

AI-Assisted Design & Optimization for Predictive Maintenance

A Case Study using Deep Learning and Search Metaheuristics for Structural Health Monitoring in Aviation

Ewald, Vincentius

DOI

[10.4233/uuid:a3070931-7512-44fa-833e-4fdc9e33da4a](https://doi.org/10.4233/uuid:a3070931-7512-44fa-833e-4fdc9e33da4a)

Publication date

2023

Document Version

Final published version

Citation (APA)

Ewald, V. (2023). *AI-Assisted Design & Optimization for Predictive Maintenance: A Case Study using Deep Learning and Search Metaheuristics for Structural Health Monitoring in Aviation*. [Dissertation (TU Delft), Delft University of Technology]. <https://doi.org/10.4233/uuid:a3070931-7512-44fa-833e-4fdc9e33da4a>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.


AI-Assisted Design & Optimization for Predictive Maintenance

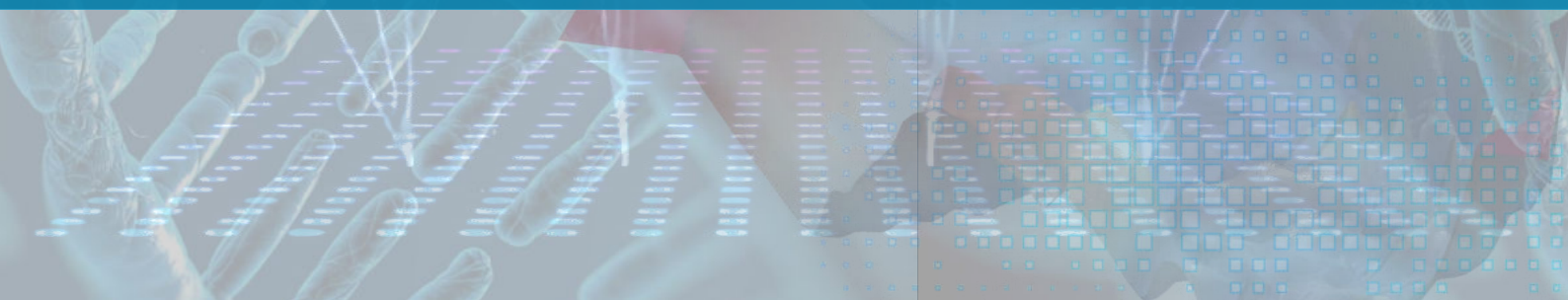
A Case Study using Deep Learning and Search
Metaheuristics for Structural Health Monitoring in Aviation

A doctoral dissertation

Vincentius Ewald



 TU Delft



AI-Assisted Design & Optimization for Predictive Maintenance

A Case Study using Deep Learning and Search Metaheuristics for
Structural Health Monitoring in Aviation

Propositions

1. Semi-supervised learning is the most feasible way to achieve success in machine learning projects (This thesis).
2. Human bias is the most crucial element to reach artificial general intelligence (This thesis).
3. Outside academia, there is no need to search an optimum by metaheuristics since greedy method is sufficient (This thesis).
4. Without AI democratization, airline and aircraft industries will stop talking about big data and Internet of Things by themselves (This thesis).
5. Stochastics is a way to express our '*Unwissenheit*'.
6. The key assumption for determining scientific truth is an agreement.
7. The best presentation slide in a scientific conference is an empty slide.
8. The boundary of science is religion and philosophy.
9. An appropriate PhD contract length to generalize a fundamental axiom e.g., from Newton 2nd law into Hamiltonian, is about 200 years.
10. The safest way to protect the Netherlands from global warming by replacing the "Deltawerken" with the Swiss Alps.

* *Unwissenheit* (DE) or *Onwetendheid* (NL) is not knowing the unknown, i.e., not to be confused with ignorance: ignorance can be understood as the act of neglecting things we know.

These propositions are regarded as opposable and defensible and have been approved as such by the promoters Prof. dr. ir. Rinze Benedictus and Dr. Roger M. Groves

AI-Assisted Design & Optimization for Predictive Maintenance

A Case Study using Deep Learning and Search Metaheuristics for SHM in Aviation

Dissertation

for the purpose of obtaining the degree of doctor

at Delft University of Technology,

by the authority of the Rector Magnificus, Prof. dr. ir. T.H.J.J. van der Hagen,

chair of the Board for Doctorates

to be defended publicly on

Friday, November 17th, 2023, at 12:30

by

Vincentius EWALD

Master of Science
Material Science and Engineering

University of Saarland
Germany

Diplôme d'ingénieur
Ecole européenne d'ingénieurs en génie des matériaux

University of Lorraine
France

born in Bandung, Indonesia

This dissertation has been approved by the promotors.

Prof. dr. R. Benedictus
Dr. R.M. Groves

Composition of the doctoral committee:

Rector Magnificus
Prof. dr. ir. R. Benedictus
Dr. R.M. Groves

Chairperson
Delft University of Technology, promotor
Delft University of Technology, promotor

Independent members:

Prof. dr. M. Vejlkovic
Dr. N. Yorke-Smith
Prof. Dr.-Ing. C. Boller
Dr. J. B. Harley
Prof. dr. ir. T. Tinga

Delft University of Technology
Delft University of Technology
University of Saarland, Germany
University of Florida, USA
University of Twente



This research has been supported by the Ministry of Economic Affairs as a part of the “Topsectoren” policy. The research team has received TKI-funding from the top-sector High Tech Systems and Materials for the Smart Sensing for Aviation Project.

Keywords: *Structural Health Monitoring, Guided Lamb Wave, Machine Learning, Deep Learning, Computational Intelligence, Metaheuristics, Optimization, Signal Processing, Sensor Network, Aircraft Inspection*

Printed by: **Ipskamp Printing**
Cover image: **Vincentius Ewald**

Copyright © 2023 by Vincentius Ewald
ISBN: 978-94-6473-293-1

An electronic version of this dissertation is available at
<http://repository.tudelft.nl/>

I have said that science is impossible without faith. No amount of purely objective and disconnected observation can show that probability is a valid notion, the laws of induction in logic cannot be established inductively. Inductive logic, the logic of Bacon, is rather something on which we can act than something which we can prove, and to act on it is a supreme assertion of faith. Science is a way of life which can only flourish when men are free to have faith.

An excerpt from *The Human Use of Human Beings*.
- Norbert Wiener

Acknowledgement

Bremen, October 14th, 2023

Since the beginning of my dissertation, there are numerous people I owe to and without them all this work would have not been possible.

My special thanks go to my supervisor Roger, Head of the Aerospace NDT Lab, who since the beginning hosted me as master student and taught me to be very critical on what I wrote. Without all his effort, the constructive comments and discussion, many of the published papers are not even worthy to be considered. Thank you, Rinze, as a chairholder of the group of structural integrity and composites (SIC) at TU Delft who hosted me as a candidate. Further thanks go to the members of Aerospace NDT Lab: Pratik, Nakash, Andrei, and Luigi for the fruitful discussion around NDT-SHM topics during the monthly colloquium.

During my research period, I was helped by Victor and Misja to do some experiments and without skilled technicians like you, some part of this thesis would be missing. Further thanks go to my previous office mates: Chirag, Camila, Nitesh, and Janos. Thanks for always having crazy conversation à la PhD: Piled Higher and Deeper. I know some of you guys are done, but some might still be writing, and we know the doctoral trajectory is roller coaster of multiple convex function, so courage to the finish line! Further, I would like to thank Huub, my fellow PhD candidate, who helped me to translate the abstract of the thesis into Dutch. Last not least, thank you Gemma for being very patient secretary who always cleaned up the mess of my administrative works.

From the University of Saarland (Germany) and the Kaunas University of Technology (Lithuania) I would like to thank my research collaborators Ramanan, Aadhik, and Christian for providing me with the simulation data and its pre-processing method. It was a very good work, and I am very proud that our collaboration yielded into a peer-reviewed publication.

I am thankful to my current colleagues at Testia: Holger, Bastien, Astrid, Ashwin, Steffen, Jonas, and my current teammates H el ene, Christian, Thomas, Samuel, and Bichu. We know all NDT and SHM is super niche world. After some years of exploring many possibilities around Europe, it seems that my current career trajectory within one of the largest aircraft OEM gave me an understanding why the current SHM approach is difficult to be implemented within this strictly controlled supply chain.

Over the years, since the beginning of my master study in Saarbr ucken and finishing the trajectory in Delft, I had friendship circle to which I am thankful for. My first thanks go to the Hodgins, I know you guys are now back in South Carolina, and I wish to visit you again one day! My bro Eric, you are always there with me in many situations, huh? Thank you as well to my fellowship in Delft: Febe, Vitali, Arjan, Mike and Nicola. Now you might be in Delft, France, Mexico, or Canada - but oh my gosh, we are very international! And Mike, I'll knock your front door the next time I'm in Vancouver, eh? Finally, my bro, fellow, ex-student, and partner in crime Xavier: After all the journey we made, I would like to express my big hug before I'm replacing you with ChatGPT v5.0.

For the fellowships in ICF Rotterdam and The Bridge Brussels: I would like to thank Fred, Henry, Bill and Gretchen. It would have been a blessing for me to know you. We meet each other at quite late stage of my dissertation, but without disengaging from my scientific day to day work, all this writing would have been meaningless to me. Thank you for all the support you provided.

Thank you to my family in Indonesia who always asked me when the dissertation is finished. Finally, a big thanks to my +1 Bea and our little one N.Y who always being with me all the time in our home. After all the years passed, this work would have been completely irrelevant without your presence.

Abstract

One of the classical solutions to maintain the aircraft structural integrity is to rely on the analysis of non-destructive testing (NDT) inspector with various inspection methods. However, it is relatively expensive in matter of time and costs to train human resources until the certification is reached. Further, in majority of the cases of aircraft scheduled and unscheduled maintenance, most of the detected damages are far below the damage tolerance limit and therefore are considered as a costly false positive because such inspections generally require additional downtime. Structural Health Monitoring (SHM) tries to reduce the wasteful resources in the maintenance, repair, and overhaul (MRO) industry by signaling such false positives during the maintenance process by becoming an integral part of the structure itself.

On the other hand, there has been an increase in using the artificial intelligence (AI) methodologies such as computational heuristics and machine learning in many areas of human civilization which includes voice and face recognition, languages translation, and automated driving. There has been a lot of interest on implementing AI to assist SHM in maintaining airworthiness while driving the cost down. Nevertheless, the maintenance of airworthiness (such as but not limited to, EASA Part 145/M and FAA CFR Part 21) is a heavily regulated area and are not easily changed.

The current state of the art was captured in the literature review. This includes recent developments of guided wave based SHM and the parameter optimization as well as recent trends and advances in artificial intelligence such as machine and deep learning. The findings from the state of the art were used as the basis to determine the research problem and to propose the solution.

The first part of the proposed solution consisted of a short review the damage growth assumption within the damage tolerance framework and the used methodology to generate and capture Lamb wave signal within Finite Element (FE) environment. This methodology is a deterministic solution that can be partially used for solving continuous optimization in deterministic sensor placement problem. It was further expanded to include a semi-stochastic approach to address non-predictable damage location that includes some metaheuristics search such as genetic algorithm and swarm intelligence. The ultimate first part of solution was a compromise between the deterministic and semi-stochastic actuator-sensor topology.

The second part of the proposed solution was the investigation on whether deep learning can be used to treat the Lamb wave signal given the configuration obtained from the first part of the proposed solution. To do so, an assumption based on converging probability measures and generalization bound in deep learning must be taken. Then, the approach is to represent the entity of the captured Lamb wave signal in time-frequency domain either as randomly sampled spectrogram or layers of joined spectrograms. After the training, the hypothesis was validated with A/B Testing.

Then, the research was expanded to understand the scalability level of deep learning for SHM for given data size, model parameters, and restriction on physical memory. In this sense, the signal representations were trained sequentially with an example of in hybrid convolutional recurrent network. The investigation was focused on stability behavior of convoluted-recurrent modelling for variable spectrogram length and the experimental validation of the model for classification of the Lamb wave spectrogram signals.

Keywords: *Structural Health Monitoring, Guided Lamb Wave, Machine Learning, Deep Learning, Computational Intelligence, Metaheuristics, Optimization, Signal Processing, Sensor Network, Aircraft Inspection*

Samenvating

Een van de klassieke oplossingen om de structurele integriteit van vliegtuigen te behouden is vertrouwen op de analyse van non-destructief onderzoek (NDO)-inspecteurs met verschillende inspectiemethoden. Dit is echter relatief duur in termen van tijd en kosten om personeel op te leiden totdat de certificering is bereikt. Bovendien liggen in de meeste gevallen van gepland en ongepland onderhoud van vliegtuigen de meeste gedetecteerde beschadigingen ver onder de schadetolerantiegrens en worden ze daarom beschouwd als een dure fout-positieve inspectie omdat dergelijke inspecties over het algemeen extra stilstandtijd vereisen. Structural Health Monitoring (SHM) probeert de verspillende hulpbronnen in de maintenance, repair, and overhaul (MRO) te verminderen door dergelijke valse positieven tijdens het onderhoudsproces te signaleren door een integraal onderdeel van de structuur zelf te worden.

Aan de andere kant is er een toename in het gebruik van kunstmatige intelligentie (AI)-methodologieën zoals computationele heuristieken en machinaal leren op veel gebieden van de menselijke beschaving, waaronder stem- en gezichtsherkenning, taalvertaling en geautomatiseerd rijden. Er is veel belangstelling voor de implementatie van AI om SHM te helpen de luchtwaardigheid te behouden en tegelijkertijd de kosten te verlagen. Niettemin is het behoud van de luchtwaardigheid (zoals, maar niet beperkt tot, EASA Part 145/M en FAA CFR Part 21) een zwaar gereguleerd gebied en kan niet gemakkelijk worden gewijzigd.

In het literatuuronderzoek is de huidige state-of-the-art vastgelegd. Dit omvat recente ontwikkelingen op het gebied van guided wave gebaseerde SHM en de parameteroptimalisatie, evenals recente trends en vooruitgang op het gebied van kunstmatige intelligentie, zoals machine- en deep learning. De bevindingen uit de stand van de techniek zijn als basis gebruikt om het onderzoeksprobleem te bepalen en de oplossing voor te stellen.

Het eerste deel van de voorgestelde oplossing bestond uit een korte beoordeling van de aanname van de schadegroei binnen het schadetolerantieraamwerk en de gebruikte methodologie voor het genereren en vastleggen van Lamb wave signalen binnen de finite elementen (FE)-omgeving. Deze methodologie is een deterministische oplossing die gedeeltelijk kan worden gebruikt voor het oplossen van continue optimalisatie van het deterministische sensorplaatsingsprobleem. Het werd verder uitgebreid met een semi-stochastische benadering om niet-voorspelbare schadelocaties aan te pakken, waaronder enkele metaheuristische onderzoeken, zoals genetische algoritmen en zwermintelligentie. Het ultieme eerste deel van de oplossing was een compromis tussen de deterministische en semi-stochastische actuator-sensor-topologie.

Het tweede deel van de voorgestelde oplossing was het onderzoek of deep learning kan worden gebruikt om het Lamb-wave signaal te verwerken, gegeven de configuratie verkregen uit het eerste deel van de voorgestelde oplossing. Om dit te doen, moet een aanname worden gedaan die is gebaseerd op convergerende waarschijnlijkheidsmetingen en generalisaties die gebonden zijn aan diepgaand leren. Vervolgens is de aanpak om de entiteit van het opgevangen Lamb-golfsignaal in het tijd-frequentiedomein weer te geven, hetzij als willekeurig bemonsterd spectrogram, hetzij als lagen van samengevoegde spectrogrammen. Na de training werd de hypothese gevalideerd met A/B-testen.

Vervolgens werd het onderzoek uitgebreid om inzicht te krijgen in het schaalbaarheidsniveau van deep learning voor SHM voor een gegeven datagrootte, modelparameters en beperkingen op fysiek geheugen. In deze zin werden de signaalrepresentaties opeenvolgend getraind met een voorbeeld van een hybrid convolutional recurrent network. Het onderzoek was gericht op het stabiliteitsgedrag van dergelijke modellering voor variabele spectrogramlengte en de experimentele validatie van het model voor classificatie van de spectrogramsignalen.

Keywords: *Structural Health Monitoring, Guided Lamb Wave, Machine Learning, Deep Learning, Computationele Intelligentie, Metaheuristiek, Optimalisatie, Signaalverwerking, Sensornetwerk, Vliegtuiginspectie*

Contents

1. Introduction	1
1.1. World of Aircraft Maintenance.....	1
1.2. The Role of Artificial Intelligence (AI) in Predictive Maintenance.....	2
1.3. Structural State Diagnostic in MRO:	3
1.3.1. Design Philosophies and Non-Destructive Testing in Aircraft Maintenance...	4
1.3.2. Role of Structural Health Monitoring in Maintenance.....	4
1.4. Thesis Organization.....	6
Literature List.....	6
2. The State of the Art	9
2.1. Recent Development on Lamb Wave SHM.....	9
2.1.1. Technological Advancement on Lamb Wave SHM.....	10
2.1.2. Advancement on Strategic Design of SHM System Parameter.....	14
2.2. Recent Trend in Machine Learning and Computational Intelligence.....	19
2.2.1. Advancement in Machine and Deep Learning.....	19
2.2.2. Computational Intelligence for Optimization.....	32
2.3. General Problem Statement & Objective.....	37
2.3.1. Recognized Problems in SHM & NDT.....	37
2.3.2. Recognized Problems in Computer Science, Machine and Deep Learning Community.....	38
2.3.3. Diagnostic Decision Logic & Maintenance Logic and Isomorphism in Finite Automata.....	39
2.3.4. Research Problems Formulation.....	43
Literature List.....	44
Appendix.....	54
3. Theoretical Background	55
3.1. Lamb Wave and Simulated Propagation.....	55
3.1.1. Acoustic Wave in Plate-Like Structure.....	55
3.1.2. Simulated Lamb-Wave Propagation in Finite Element Environment.....	59
3.1.3. Piezoelectric Actuator and Sensor.....	61
3.1.4. Lamb Wave Attenuation.....	63
3.2. Signal Representation and Data Processing.....	65
3.2.1. Signal Representation.....	66
3.2.2. Feature Extraction and Damage Index.....	70
3.3. Optimization and Search Metaheuristics.....	72
3.3.1. Fundamentals on Continuous Optimization.....	73
3.3.2. Search Metaheuristics in Discrete Optimization.....	75
3.4. Machine and Deep Learning.....	80
3.4.1. Statistical Learning Theory for Supervised Learning.....	81
3.4.2. Inductive Bias.....	85
3.4.3. Neural Network.....	87
3.4.4. Hyperparameter Tuning.....	91
3.4.5. Regularization.....	93
3.5. Features Learning and Invariant Representation.....	94
Literature List.....	97
4. Deterministic Approach Sensor Placement	103
4.1. Crack Growth in Damage Tolerance Structure.....	103
4.2. Simulation of Lamb Wave Propagation with ABAQUS FE.....	105
4.3. Image Processing.....	107
4.4. Blob Detection.....	109
4.5. Results and Discussion.....	112
4.6. Conclusion.....	115
Literatures.....	116
5. Holistic Sensor Network Topology Optimization	117
5.1. Fitness Function.....	117
5.2. Methodology.....	122

5.2.1. Global Random Search.....	123
5.2.2. Greedy Search.....	124
5.2.3. Metaheuristics Search.....	125
5.2.3.1. Genetic Algorithm (GA).....	125
5.2.3.2. Simulated Annealing.....	127
5.2.3.3. Swarm Intelligence.....	128
5.3. Preliminary Results	129
5.3.1. Comparison between Greedy Methods, Random Search, and GA.....	129
5.3.2. Comparison between Metaheuristics.....	130
5.4. Integrative Method.....	131
5.5. Experimental Validation.....	133
5.5.1. Reproducing Hotspot SHM.....	133
5.5.2. Experimental Setup.....	134
5.5.3. Impact Damage Setup.....	135
5.5.4. Damage Analysis.....	137
5.5.4.1. Hotspot Damage Detection.....	138
5.5.4.2. Impact Localization.....	138
5.6. Chapter Summary.....	144
Literatures.....	144
6. Deep Learning for Structural Health Monitoring.....	147
6.1. Research Outline Recap.....	148
6.2. Concept and Theoretical Background	149
6.2.1. Formalization of DeepSHM.....	149
6.2.2. Model Abstraction of DeepSHM Behavior.....	154
6.3. Methodology.....	155
6.3.1. Simulation Setup.....	155
6.3.2. Data Pre-Processing.....	158
6.3.3. Entity Representation.....	159
6.3.3.1. Hierarchical Representation in Multiple Sensor.....	159
6.3.3.2. Conserved Entity over Time.....	159
6.3.4. Training Setup and Parameters.....	160
6.3.4.1. Hardware.....	160
6.3.4.2. Software and Libraries.....	160
6.3.4.3. Optimization.....	161
6.3.4.4. Neural Network Architectures and Optimizers.....	161
6.4. Training Result.....	162
6.4.1. On Modelling DeepSHM as Multiple Actors with Independent Decision Making.....	163
6.4.1.1. Influence of Network Architecture and Sensor Locations.....	163
6.4.1.2. Effect of the Convolution Window Length.....	167
6.4.2. On Modelling DeepSHM as Conserved Entity over Time.....	168
6.5. Concept Validation.....	171
6.5.1. Result comparison with Random Noise Training.....	171
6.5.2. Model Testing.....	172
6.5.2.1. Hierarchical Representation in Multiple Sensors.....	172
6.5.2.2. Representation as Conserved Entity over Time.....	173
6.6. Conclusion.....	174
6.6.1. Summary.....	174
6.6.2. Conclusion and Recommendation.....	174
Source codes and online Documentation.....	176
Literatures.....	176
7. Recurrent Modelling of Time-Frequency Signal.....	179
7.1. Introduction.....	179
7.2. Related Work to Recurrent Modelling in Predictive Maintenance.....	180
7.3. Theoretical Foundation of Hybrid ConvNet-Recurrent Network.....	182
7.4. Methodology.....	183
7.4.1. Experimental Validation.....	184
7.4.2. Sensor Placement.....	185
7.4.3. Reassigned Spectrogram.....	186

7.4.4. Recurrent Training Mechanism.....	187
7.4.5. Hyperparameters Configuration.....	187
7.5. Results and Discussion.....	188
7.5.1. Simulation Dataset.....	188
7.5.2. Experimental Dataset.....	190
7.5.3. Discussion.....	192
7.6. Conclusion.....	193
Attachment.....	194
Literatures.....	206
8. Conclusion and Future Works.....	207
8.1. Summary.....	207
8.2. Conclusion.....	209
8.3. Research Outlook.....	212

List of Abbreviations

ABC	Artificial Bee Colony
AC	Advisories Circular; Alternating Current
AC(S)O	Ant Colony (System) Optimization
AE	Autoencoder; Acoustic Emission
AI	Artificial Intelligence
API	Application Programming Interface
AROR	Annualized Rate of Returns
ART	Adaptive Resonance Theory
(AL)BERT	(A Lite) Bidirectional Encoder Representations from Transformers
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BMP	Bitmap
BNC	Bayonet Neill Concelman
CBM	Condition Based Maintenance
CC	Cross Correlation
CDF	Cumulative Distribution Function
CFL	Courant-Friedrich-Lewy
CFRP	Carbon Fiber Reinforcement Plastics
CI	Computational Intelligence
CL	Classification Layer
CNC	Computerized Numerical Control
(C)(R)NN	(Convolutional) (Recurrent) Neural Network
CT	Chirplet Transformation; Computed Tomography
CV	Computer Vision
(C)WT	(Continuous) Wavelet Transform
DBN	Deep Belief Network
DDF	Digital Damage Fingerprints
DDR	Double Data Rate
DE	Differential Evolution
DEC	Deep Embedding Clustering
DFA	Deterministic Finite Automata
DI	Damage Index
DL	Deep Learning
(D)(A)NN	(Deep) (Artificial) Neural Network
DOE	Design of Experiment
DOL	Designated Operational Lifecycle
DPD	Damage Parameter Database
DWT	Discrete Wavelet Transformation
EASA	European Union Aviation Safety Agency
EDM	Electrical Discharge Machine
EEG	Electroencephalography
EFIT	Elastodynamic Finite Integration Technique
EM	Expectation Maximization
EMAT	Electromagnetic Acoustic Transducer
EMD	Empirical Mode Decomposition
EMI	Electromechanical Impedance
FAA	Federal Aviation Administration
FBG	Fiber-Bragg Grating
FDM	Finite Difference Method
FN	False Negative
FOL	First Order Logic
FP	False Positive
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GA	Genetic Algorithm
GAN	Generative Adversarial Network
GB	Gigabyte
GDOP	Geometric Dilution of Precision

GDT	Generalized Data Transformation
GPT	Generative Pre-Trained Transformer
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
HF	High-Frequency
HHC	Home Health Care
HHT	Hilbert-Huang Transform
HMM	Hidden Markov Models
HT	Hilbert Transform
(I)(F)FT	(Inverse) (Fast) Fourier Transformation
IR	Infrared
JPEG	Joint Photographic Experts Group
KL	Kullback-Leibler
LISA	Local Interaction Simulation
LSTM	Long Short-Term Memory
LW	Lamb Waves
MAD	Mean Absolute Deviation
MAPCG	Multi-Attention Parallel CNN-GRU
MAV	Micro Aerial Vehicle
MB	Megabytes
MDP	Markov Decision Process
MI	Mutual Information
ML	Machine Learning
MLP	Multi-Layer Perceptron
MPCA	Multiple Particle Collision Algorithm
MPI	Magnetic Particle Inspection
MRO	Maintenance, Repair, and Overhaul
MSE	Mean-Squared Error
MSG	Maintenance Steering Group
MTS	Minimum Time Search
NAG	Nesterov Accelerated Gradient
ND(E/I/T)	Non-Destructive Evaluation / Inspection / Testing
NFA	Non-Deterministic Finite Automaton
NFZ	Near-Field Zone
NNF	Nearest-Neighbor Fields
NP	Non-Polynomial
ODB	Output Database
OEM	Original Equipment Manufacturer
OPEX	Operating Expenditure
(PA)UT	Phased Array Ultrasonic Tomography
PAC	Probably Approximately Correct
PCA	Principal Component Analysis
PDE	Partial Differential Equations
PID	Proportional Integral Derivative
PMMA	Polymethylmethacrylate
PNG	Portable Network Graphic
POD	Probability of Detection
PSO	Particle Swarm Optimization
PVDF	Polyvinylidene Fluoride
PWAS	Piezoelectric Wafer Active Sensor
PZT	Pb[Zr _x Ti _{1-x}]O ₃ (Lead Zirconium Titanate)
QAP	Quadratic Assignment Problem
RAM	Random Access Memory
RBM	Restricted Boltzmann Machine
RGB	Red-Green-Blue
RL	Reinforcement Learning
RMSD	Root Mean Square Deviation
RUL	Remaining Useful Life

SA	Simulated Annealing
SGD	(Stochastic) Gradient Descent
(S)FE(M)	(Spectral) Finite Element (Method)
SH	Shear Horizontal
SHM	Structural Health Monitoring
SIF	Stress Intensity Factor
SIM	Sharp Interface Model
SLDV	Scanning Laser Doppler Vibrometer
SLP	Single-Layer Perceptron
SNR	Signal-to-Noise Ratio
SPWVD	(Smoothed) (Pseudo) Wigner-Ville Distribution
STDP	Spike-Timing Dependent Plasticity
STFT	Short-Time Fourier Transform
SV	Shear Vertical
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TCS	Total Cost Saving
TFD	Time-Frequency Distribution
TFR	Time-Frequency Representation
TN	True Negative
TOA	Time-Of-Arrival
TOF	Time-Of-Flight
TP	True Positive
TS	Tabu Search
TSP	Travelling Salesman Problem
UAV	Unmanned Aerial Vehicles
US	Ultrasound
VC	Vapnik-Chervonenkis
VGG	Visual Geometry Group
WDCNN	Wide First-Kernel Deep CNN
WDP	Winner Determination Problem
WFT	Warped Frequency Transform

List of Most Relevant Physical Notations in Chronological Appearance

$\frac{\partial^2 p}{\partial t^2} = \frac{\partial^2 p}{\partial x_1^2} + \frac{\partial^2 p}{\partial x_2^2}$	The second partial derivative of mechanical pressure p at time t is dependent of second derivative of p towards coordinate x_1 and x_2
$c_L = \sqrt{\frac{\lambda_{\text{Lamé}} + 2\mu}{\rho}}$	The longitudinal wave velocity c_L as a function of Lamé constant $\lambda_{\text{Lamé}}$, the material density ρ , and the shear modulus μ
$c_T = \sqrt{\frac{E}{2\rho(1+\nu)}}$	The transversal wave velocity c_T as a function of elastic modulus E , the material density ρ , and Poisson's ratio ν
$f; \mathbf{y}$	Scalar potential, Vector potential
$p^2 = \frac{\omega^2}{c_L^2} - k^2$	Non dimensional parameter p is dependent from angular frequency ω , longitudinal bulk wave velocity c_L and wavenumber k
$c_G = c_P - \lambda_{\text{wave}} \frac{\partial c_P}{\partial \lambda_{\text{wave}}}$	The dependency of wave group velocity c_G as function of wave phase velocity c_P , and wavelength λ_{wave}
$C_{ijkl}; \mathbf{e}$	General stiffness matrix of the material; strain tensor
$\mathbb{W}; \mathbf{G}$	Volumetric area; surficial integral area
$M\ddot{\mathbf{u}} + C\dot{\mathbf{u}} + K\mathbf{u} = \mathbf{F}_a$	The vector of applied loads F_a as product of the structural mass matrix M , the structural damping matrix C , and structural stiffness, where \mathbf{u} is the particle displacement, $\dot{\mathbf{u}}$ is the particle velocity and $\ddot{\mathbf{u}}$ is the particle acceleration
$\Delta t_{\text{CFL}}; \Delta t_{\text{rec}}; h; f_{\text{max}}$	Time increment Δt_{CFL} according to CFL condition; Recommended time step Δt_{rec} ; finite element width h ; maximum frequency f_{max}
$S_{ij} = S_{ijkl}^E T_{kl} + d_{kij} E_k$	Mechanical strain S_{ij} as a product of the mechanical compliance of the material s_{ijkl}^E measured at zero electric field, mechanical stress T_{kl} , and piezoelectric coupling effect d_{kij} .
$D_j = d_{jkl} T_{kl} + \epsilon_{jk}^T E_k$	Electrical displacement D_j as a function of electrical field E_k and dielectric permittivity measures ϵ_{jk}^T zero mechanical stress
$P = P_0[\alpha \cdot \exp(-\beta x)]$	Excitation signal power P as a function of geometrical attenuation factor α and material attenuation coefficient β
$E_0; A(t)$	Original excitation energy; signal amplitude at time t
$D : \{\pi, \psi, \tau, \lambda, \omega\}$	A quintuple D which consists of vector elements written as convention in actor tuple π , medium tuple ψ , transition tuple τ , phenomenon tuple λ , and environmental tuple ω
$X_\lambda(t); \tilde{X}(f)$	Signal vector X at time t influenced by parameter λ , transformed signal vector X in the frequency domain
$\Phi\left(\frac{t-b_n}{a_n}\right)$	A mother wavelet function Φ with shifting factor b and scaling factor a at time t and its discretized version for a discrete time-step n
$K_{\text{IC}}, \alpha_{\text{crit}}$	Critical stress intensity factor, critical crack length
$r_{ij}; s_{ij}; \tau; \delta; \sigma$	The Euclidian distance from the wave source at coordinate (x_i, y_i) up to an arbitrary pixel; pixel score at coordinate (x_i, y_i) ; network score; averaged Euclidian distance; standard deviation

List of Most Relevant Mathematical Notations in Chronological Appearance

$P(z)$	Superclass of probability distribution P with density z
$\forall : x, \exists : y$	For all x , there exists y
$A \supseteq \{B, C\} \neq \emptyset$	A is weak superset of B and C and is not an empty set \emptyset
$\delta : Q \times \Sigma \rightarrow F$	The state transition function δ maps state Q which contains alphabet Σ into final state F
$^{-n}; \mathbb{N}^n; \mathbb{R}^n$	Set of natural, integer, and real numbers with dimension of n
$i \in I; i \in \mathbb{N}^+$	Element i belongs to set I , element i belongs to set of positive integers
$\mathfrak{N} \cap \Omega$	Intersection elements that belong to both the sets \mathfrak{N} and Ω
$P(d_m^y f, m, A)$	Conditional probability of the ordered set d of size m of the cost values y , given function f to be optimized and algorithm A .
$\eta \leftarrow \eta_0$	Initial gradient step η_0 is assigned to variable η
$x_{i+1} = x_i - \eta_i \nabla f(x_i)$	Data point x at step $i+1$ is defined as an update of its previous value at step i , subtracted by gradient step η multiplied with partial derivative of function f
$X \subseteq 2^G$	Set X is a subset of feasible solution of ground set 2^G
$H_{\mathfrak{C}}$	Hypothesis space of concept class \mathfrak{C}
\hat{E}	Generic notation of estimate
$\sup; \inf$	Supremum; infimum
\mathfrak{R}_n	Rademacher complexity with respect to the sample n
$\min_{\theta \in [0,1]} J(\theta_{mn})$	Minimize the loss function J that has size dimension $m \times n$, where parameters θ can take any real value between 0 and 1
$J_{\text{MSE}}(\theta); J_{\text{CEE}}(\theta)$	Mean-squared based loss function $J(\theta)$; cross-entropy based loss function $J(\theta)$
$V \circ U$	Composition of function V and U
r_{y_n}	Invariant latent of label y at sample n
$\nabla^2 G(x, y); \sigma$	Laplacian (second partial derivative) of Gaussian function G at pixel x, y ; feature scaling

1. Introduction

1.1. World of Aircraft Maintenance

Besides fuel and ground services, one of the most crucial aspects in an airline operating cost is the maintenance. In 2017, it was reported that 70 billion USD was spent by airlines for maintenance, repair, and overhaul (MRO) [Michaels (2018)] and this figure was expected to grow to 115 billion USD in 2028 due to increasing number of aircraft deliveries [Ann Shay (2018), Chong (2018)]. According to [IATA (2019)], the current maintenance cost varies between 91 and 44,573 USD (with an average of 2634 USD) per flight cycle. While seemingly small, this figure translates to between 0.1 - 16 million USD (with an average of 3.6 million USD) per aircraft during its lifetime.

In general, without standards and regulations, different airlines and aircraft manufacturers would have different maintenance procedures and the situation would be a mess because every company would have their own standard. So, to harmonize these maintenance procedures, governmental bodies have been set-up to make advisory circulars (AC) to improve airworthiness. This was the reason for inception of the maintenance steering group (MSG) task that has the purpose to be aircraft maintenance logic.

The focus of MSG-2 was process orientated and it used a bottom-up approach. As mentioned earlier, humans learn from their previous experiences because they often develop a process that are far from perfect. Based on the experience and the identified weaknesses of MSG-2, this process was overhauled in MSG-3, which was first published in 1980. In contrast to the MSG-2, MSG-3 introduced a top-down approach by focusing on the consequences of failure.

An effort-aware maintenance manager would rationally not spend time and associated cost for excessive maintenance if there is no foreseeable reliability improvement or if the total effort of excessive maintenance exceeds its real value. When focusing on CBM, there is a lot of space to continuously improve and there are many approaches how to develop CBM [Tinga and Loendersloot (2014)] - this will be discussed in detail as it will be discussed in chapter 2 and 3.

Many engineering structures are designed to be operated within their design limits. In any engineering discipline be it civil, automotive, aerospace, electrical, or mechatronics, there is always a term called **lifecycle** or **lifetime**, which can be more precisely expressed as **designated operational lifecycle (DOL)**. The DOL typically signifies a quantifiable amount in term of usage cycles, within which the structure can be utilized reliably. That means, after the DOL is reached - it should be the time to write off the object or system.

On the other side however, our universe tends to behave in non-predictable way, and we can denote it as a **stochastic universe**. Pragmatically, we shall assume that uncertainties could be very likely occur at any time. A practical

application of this philosophical assumption for aircraft maintenance is that damages occur during the DOL of an aircraft and unfortunately this would likely to reduce the DOL by a certain degree, we just do not know how much because monitoring these uncertainties is very difficult.

1.2. The Role of Artificial Intelligence (AI) in Predictive Maintenance

In 2012, a new jargon was introduced by the German Ministry of Education and Research [BMBF]: **Industry 4.0** and the term **4th industrial revolution** is also used equivalently. While the 3rd industrial revolution in early 1990 was heavily focused on utilization of electronic devices to increase productivity using automation, Industry 4.0 reinforces the ability of the electronic devices to connect with each other, commonly known as Internet-of-Things (IoT). IoT has many sub-components and one of its sub-components is the volume, velocity, and variety of data involved (also called **big data**) and its corresponding advanced analytics and algorithms. In other word, the importance of using data be it either for leveraging of MRO processes by **artificial intelligence** (AI) and **machine learning** (ML), monitoring real-time performance, or accelerated information exchange within Industry 4.0 is emphasized.

Recently, people are also increasingly talking about **deep learning** (DL), a subset of ML techniques using a deep **artificial neural network** (ANN) [Frankish and Ramsey (2014), Chollet (2018)]. However, when talking about deep and machine learning, we should first understand their relationship with artificial intelligence (AI). Unfortunately, there is no universally accepted definition of these terms, however the consensus can be summarized in Fig. 1.2-1, where we can see DL is the smallest subset of ML and ML is a subset of the broader AI field. The concept of artificial intelligence arose when Alan Turing introduced the **Turing test** in 1950, which was published his paper [Turing (1950)] with an opening phrase: "*I propose to consider the question: Can machines think?*".

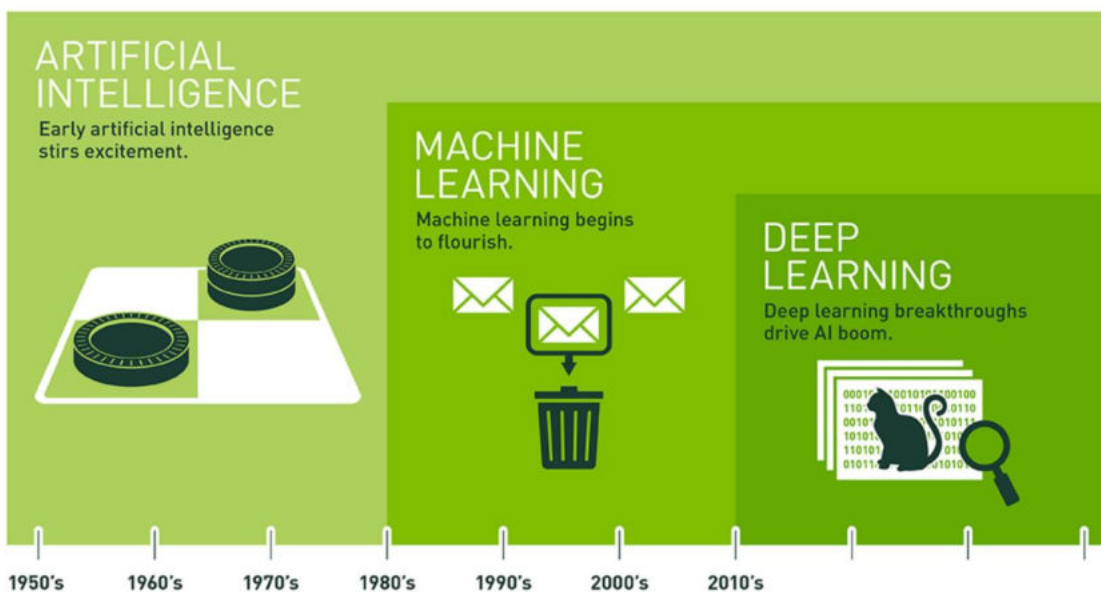


Fig. 1.2-1: Deep Learning as a subset of machine learning which is just another sub-field of artificial intelligence.

In his test, Turing proposed that a human judge makes a conversation with an artificial responder that generates human-like answers in a separate room with the condition that the conversation is limited to communication via the computer keyboard and screen. If the human cannot reliably distinguish whether the response was generated by the machine or human, the machine is said to have passed the Turing test. This test thus become one of the most important concepts in AI, although it was not free of criticism, especially when [Searle (1980)] proposed the **Chinese room** argument.

Apart from the philosophical concept, the term deep learning was first mentioned by [Dechter (1986)], albeit not within the context of multilayered neural network. Most notable work has been performed by [Hinton and Salakhutdinov (2006)], when they introduced the **restricted Boltzmann machine** (RBM), a generative stochastic ANN that learns a probability distribution from its set of inputs. However, the history of ANN can be tracked back to 1958 when Rosenblatt introduced the concept of perceptron for the first time [Rosenblatt (1958)]. The popularity of ANN grew until the late 1970's, where it was slowed down by then the slow computational ability. Eventually this popularity was overshadowed by another machine learning technique called Support Vector Machine (SVM) [Corinna and Vapnik (1995)] in the 1990's.

Like any other emerging technology, AI oversaw several cycles of hype and disappointment. Cuts in AI research funding marked the periods of disappointment, commonly referred as **AI winters** [Crevier (1993), Hendler (2008)] and it has already happened twice, during late 1970's and between the early 1990's and early 2000's. Since 2010's however, thanks to ever-increasing computational power (see Moore's Law [Moore (1965), Liddle (2006)] for a detailed explanation) including the developments of the graphical processing unit (GPU) and field programmable gate array (FPGA), the development of AI has been in steady pace.

From the low-level perspective, an example case for the technical implementation within the predictive maintenance framework is the utilization of past historical data from many sensors already installed in engineering structures to predict potential damages and if possible, their future states by using both rule-based systems including the majority of sensing signal processing techniques, and machine learning methods.

1.3. Structural State Diagnostic in MRO

The last paragraph of section 1.2 mentioned the utilization of historical sensor data to predict the potential damages contained by engineering structure. So, given a similar structure, when a similar damage occurs in the future, the resulting sensor signal would also be likely to be similar, and this would be an interesting study case for machine and deep learning as neural network is generally good at recognizing certain signal patterns. The detail explanation of this will be given later in chapter 3.

1.3.1. Design Philosophies and Non-Destructive Testing in Aircraft Maintenance

According to [Baker and Wang (2018), Bellinger and Liao (2009)], three design philosophies exist: safe-life, failsafe, and damage tolerance. Safe-life design is based on the predicted lifecycle of a component. During this period, it should be assumed that no component failure occurs. After this period, the component must be replaced, regardless of its condition. On the other side, failsafe is a design to which an error results in the least possible damage. It requires that partial failure of the structure does not cause its entire failure. Failsafe design recognizes that cracks may occur, but the structures is so arranged that the critical crack will not lead to sudden total failure before it can be repaired. An example of a failsafe design is a horizontal stabilizer [Wanhill (2003)].

Damage tolerance is defined as *“the ability of a structure to sustain anticipated loads in the presence of fatigue, corrosion or accidental damage until such damage is detected through inspections or malfunctions and is repaired”*. [Ransom et al. (2008)]. It is an extension of failsafe design, and it plays a central role in current aircraft design. Within damage tolerance, a thorough damage assessment in a possible failure scenario is necessary. Three key items in damage tolerant designs are: fatigue crack growth, crack detection with NDT, and residual strength prediction [Fatemi et al. (2001), Abdel-Latif (2009)]. While the damage tolerant design itself can be regarded as a passive protection against damage, NDT is an active intervention within the maintenance framework.

To be able to predict the residual strength before the next repair, a non-destructive evaluation of crack existence and an estimate of its growth is needed. This evaluation of the concerned component demands aircraft repair stations with open-access and closed-access man hours, which costs money and time [McFadden and Worrels (2012)]. As many aircraft parts are made to be damage tolerance, an inspection within certain interval is needed to ensure that the aircrafts are operating within its design limit [EASA AMC 20-20A].

There are a manifold of NDT technologies that have been established for decades in aircraft inspection [Fahr (2014)] such as high-frequency ultrasound (HF-US) and phased-array ultrasonic tomography (PAUT) [Rau (2006), Pohl (1998)], visual inspection combined with magnetic particle inspection (MPI) [Betz (2007)], and eddy current [Meilland (2006)] and all these techniques are used to detect flaw such as crack and delamination within aircraft parts.

1.3.2. Role of Structural Health Monitoring in Maintenance

All the above-mentioned techniques have a range of detection accuracy and their own cost in terms of man-hours and down-time [Shaloo et al. (2022)]. In aerospace domain, a qualified NDT technician is in general in very rare position and while an NDT inspection costs both man-hours and service down-time. Nevertheless, it is needed to be understood that the underlying principle here is

not to fly unsafe aircraft, but rather to have reliable decision support for MRO about whether certain parts are to be repaired or not.

In classical aircraft maintenance, this unscheduled inspection tends to be rigorous and very tedious. While this is done for understandable safety reasons, it is not very efficient either since it requires unnecessary man-hours and down-time. In line with the spirit of predictive maintenance within Industry 4.0 for aircraft MRO, a question now arises: *How can we minimize uncertainty during unscheduled maintenance with advanced novel technologies such as deep learning and big data that can be combined with increasing computational power?*

With the current approach, to detect a crack with NDT equipment, one needs access for opening and closing the relevant substructure of the aircraft and regardless of whether a crack is found or not, this will require spending man-hours. This is where a successful Structural Health Monitoring (SHM) becomes handy, where NDT is embedded the aircraft – so when certain aircraft parts to be replaced, ideally it automatically sends a warning flag to the aircraft operator to signal that that certain part must be replaced without spending man-hours for determining the uncertainty.

When looking in low-level detail regarding automated inspection by SHM, there are several potential techniques that can be used as NDT decision support while not completely replacing NDT itself. Among them are guided Lamb Waves (LW) SHM which principally uses lower-frequency acoustic waves and Fiber-Bragg Grating (FBG) [Ibrahim (2017)]. Guided LW-SHM shares the same physical phenomenon as the well-established NDT such as HF-US and PAUT, except that this technique is more suitable for larger area measurement with a capability up to several meters due to its larger wavelength in comparison to HF-US and PAUT which typically only penetrates area of several cm³ under the sensor. This is the main reason guided Lamb waves has been selected as the main physical phenomenon of this dissertation.

Like other SHM methods, the approach of using LW-SHM is not immune to criticism, particularly because there are multitudes of parameter configurations that were previously less problematic in NDT but now would affect the LW-SHM signal pattern – such as sensor positioning, frequency selection, type of sensor adhesive, etc. In chapter 5, these factors will be discussed and some methodologies will be proposed, particularly **metaheuristic optimization** [Gogna (2013), Du and Swamy (2016), Sørensen et al. (2018)] which are of interests not only to AI community but also to **operations research** [Rardin (2016)] community.

Now, while some might believe that the well-established NDT methods are a direct competitor to LW-SHM, looking back into the purpose of SHM as NDT decision support it turns out that both methods can and should complement each other rather than being competitors.

1.4. Thesis Organization

The thesis is structured as follows: The state of the art of guided LW-SHM, machine and deep learning in the last 10 years will be reported in chapter 2. Based on this state-of-the-art, the problem statement and the main research objective will be introduced. The well-established theoretical background that covers the physics and numerical simulation of guided Lamb waves, configuration factors such as sensor placement method, signal processing, inductive bias based on domain knowledge and well-known statistical learning theory is given in a separate theoretical chapter as chapter 3.

Chapter 4 will describe the preliminary results from the used methodology to numerically simulate the data generation and its experimental validation, while the preliminary result from the sensor placement methodology and its experimental validation is given in chapter 5. The results of deep learning training behavior including the discussion of different representations is given in chapter 6, while chapter 7 is a methodology extension of chapter 6 in which the representation is modelled in sequential way. Finally, the summary, conclusion, and outlook of this dissertation is given in chapter 8. For ease of reading, each chapter will have its separate literature list.

Literature list

- Abdel-Latif AM. *An Overview of the Applications of NDI/NDT in Engineering Design for Structural Integrity and Damage Tolerance in Aircraft Structures*. In *Damage and Fracture Mechanics*. Springer Science+Business Media, Berlin / Heidelberg (2009).
- Ann Shay L. *Commercial Spending Will Lead MRO Field in 2018*. Aviation Week & Space Technology (2018). Available <http://aviationweek.com/commercial-aviation/commercial-spending-will-lead-mro-field-2018> (Last online: FEB-2020).
- Baker AA, Wang J. *Adhesively Bonded Repair/Reinforcement of Metallic Airframe Components: Materials, Processes, Design and Proposed Through-Life Management*. In *Aircraft Sustainment and Repair*. Chapter 6: 191 - 252 (2018).
- Bellinger NC, Liao M. *Corrosion and Fatigue Modeling of Aircraft Structures*. In *Corrosion Control in the Aerospace Industry*. Chapter 8: 172 - 191 (2009).
- Betz CE. *Principle of Magnetic Particle Inspection*. Magnaflux (2007).
- Bundesministerium für Bildung und Forschung (BMBF). *Was ist Industrie 4.0?* Available: <https://www.plattform-i40.de/PI40/Navigation/DE/Industrie40/WasIndustrie40/was-ist-industrie-40.html> (Last online: MAR-2020).
- Chollet F. *Deep Learning with Python and Keras*. Manning Publications Inc, Shelter Island (2018).
- Chong A. *Global MRO spend to reach \$115 billion by 2028 - Wyman*. Flightglobal (2018). Available <https://www.flightglobal.com/news/articles/global-mro-spend-to-reach-115-billion-by-2028-oli-445243/> (Last online: FEB-2020).
- Corinna C, Vapnik V. *Support-Vector Networks*. J Machine Learning. Vol. 20: 273-297 (1995).
- Crevier D. *AI: The Tumultuous Search for Artificial Intelligence*. Basic Books Inc, New York (1993).
- Dechter R. *Learning While Searching in Constraint-Satisfaction-Problems*. Proc. 5th American Association of Artificial Intelligence (AAAI), Philadelphia (1986).
- Du KL, Swamy MNS. *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*. Birkhäuser, Basel (2016).
- European Union Aviation Safety Agency (EASA). *AMC 20-20A: Continuing Structural Integrity Programme*. Available: <https://www.easa.europa.eu/en/document-library/easy-access-rules/online-publications/easy-access-rules-acceptable-means?page=14> (Last online: NOV 2022).

- Fatemi A, Fuchs H, Stephens R. *Metal Fatigue in Engineering* (2nd Ed.). John Wiley & Sons, New York (2001).
- Frankish K, Ramsey WM. *The Cambridge Handbook of Artificial Intelligence*. Cambridge University Press, Cambridge (2014).
- Gogna A. *Metaheuristics: Review and Application*. J Experimental & Theoretical Artificial Intelligence. Vol. 25(4): 503-526 (2013).
- Hendler JA. *Avoiding Another AI Winter*. IEEE Intelligent Systems. Vol. 23: 2-4 (2008).
- Hinton GE, Salakhutdinov RR. *Reducing the Dimensionality of Data with Neural Networks*. J Science. Vol. 313(5786): 504-507 (2006).
- Ibrahim RA. *Handbook of Structural Life Assessment*. John Wiley & Sons Ltd., Hoboken (2017)
- International Air Transport Association (IATA). *Airline Maintenance Cost: Executive Commentary Edition 2019*. Available <https://www.iata.org/contentassets/bf8ca67c8bcd4358b3d004b0d6d0916f/mctg-fy2018-report-public.pdf> (Last online: MAR-2020).
- International Civil Aviation Organization (ICAO). *State of Global Aviation Safety 2019 Edition*. Available https://www.icao.int/safety/Documents/ICAO_SR_2019_final_web.pdf (Last online: MAR-2020).
- Liddle DE. *The Wider Impact of Moore's Law*. IEEE Solid-State Circuits Society Newsletter. Vol. 20(3): 28 (2006).
- McFadden M, Worrells DS. *Global Outsourcing of Aircraft Maintenance*. Journal of Aviation Technology and Engineering Vol. 1(2): 63-73 (2012).
- Meilland P. *Novel Multiplexed Eddy-Current Array for Surface Crack Detection on Rough Steel Surface*. Proc. 9th European Conf of Non-Destructive Testing (ECNDT), Berlin (2006).
- Michaels K. *Opinion: OEMs Focus on Mature Aircraft for Aftermarket Growth*. Aviation Week & Space Technology (2018). Available <http://aviationweek.com/commercial-aviation/opinion-oems-focus-mature-aircraft-aftermarket-growth> (Last online: FEB-2020).
- Moore GE. *Cramming More Components onto Integrated Circuits*. J Electronics. Vol. 38(8): 114-117 (1965).
- Pohl J. *Ultrasonic Inspection of Adaptive CFRP-Structures*. Proc. 7th European Conf of Non-Destructive Testing (ECNDT), Copenhagen, (1998).
- Ransom JB, Glaessgen EH, Raju IS, Knight NF, Reeder JR. *Lessons Learned from Recent Failure and Incident Investigations of Composite Structures*. Proc. 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Schaumburg (2008).
- Rardin RL. *Optimization in Operations Research* (2nd Ed.). Prentice Hall, Upper Saddle River (2016).
- Rau E, Grauvogl E, Manzke H, Cyr P. *Ultrasonic Phased Array Testing of Complex Aircraft Structures*. Proc. 9th European Conf on Non-Destructive Testing (ECNDT), Berlin (2006).
- Rosenblatt F. *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. J Psychological Review. Vol. 65(6): 386-408 (1958).
- Searle J. *Minds, Brains and Programs*. J Behavioral and Brain Sciences. Vol. 3(3): 417-457 (1980).
- Shaloo M, Schnall M, Klein T, Huber N, Reitingner B. *A Review of Non-Destructive Testing (NDT) Techniques for Defect Detection: Application to Fusion Welding and Future Wire Arc Additive Manufacturing Processes*. J Materials (Basel). Vol. 15(10): 3697 (2022).
- Sörensen K, Sevaux M, Glover F. *A History of Metaheuristics*. In *Handbook of Heuristics*. Springer, Cham (2018).
- Tinga T, Loendersloot R. *Aligning PHM, SHM, and CBM by Understanding the Physical System Failure Behaviour*. Proc. European Conf of the Prognostics and Health Management Society (PHME), Nantes (2014).
- Turing AM. *Computing Machinery and Intelligence*. J Mind. Vol. 49: 433-460 (1950).
- Wanhil RJH. *Milestone Case Histories in Aircraft Structural Integrity*. Comprehensive Structural Integrity. Vol. 1: 61-72 (2003).

2. The State of the Art

Chapter 1 introduced the background and the general scope of this dissertation, while in this chapter we will explore the more specific scope of the work. Before going too deep, recall the definition of SHM proposed by [Boller (2008)]: “*SHM is the integration of sensing and [...] actuation devices to allow the loading and damaging conditions of a structure to be recorded, analyzed, localized, and predicted in a way that NDT becomes an integral part of the structure and a material. Consequently, SHM requires [...] loads and damage monitoring [...] assessment algorithms and needs to get those merged in a holistic process such that the structural health can be accompanied during the life cycle [...]*” which can be visualized as the workflow depicted in Fig. 2-1 [Ooijselaar (2014)]:

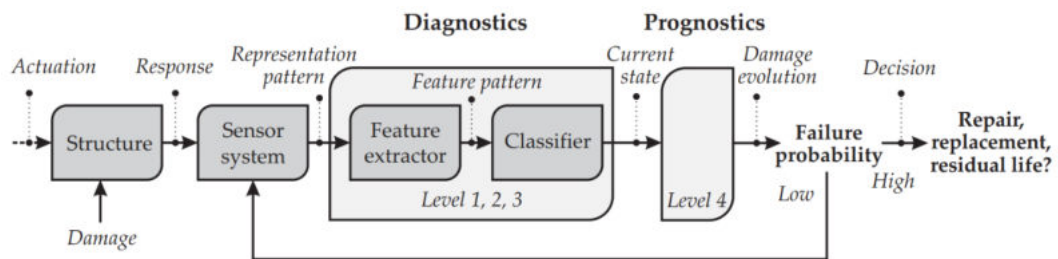


Fig. 2-1: The workflow of SHM proposed by [Ooijselaar (2014)] in which one could decide to induce a certain actuation in a structure that might contains any damage, record the structural response that is captured by sensor system, and represent the response as feature pattern.

As depicted in Fig. 1, SHM is divided into two major frameworks: diagnostic and prognostic. Diagnostic SHM concerns the feature extraction and classification that can be related to the physical current state of the structure, while prognostic SHM evaluates the failure probability of the structure given the current structural state and decides whether the concerned structure needs to be repaired, replaced, or written-off depending on its predicted residual life. Specifying the framework further, the tasks of SHM can be generally divided into 4 functional levels, where diagnostic SHM comprises the functional level 1 – 3 and prognostic SHM comprises the functional level 4 [Ooijselaar (2014)]:

SHM Level 1: the condition monitoring of structural load

SHM Level 2: the detection of damage presence

SHM Level 3: the characterization of the present damage regarding its location, type, and severity in terms of size

SHM Level 4: the prediction of failure probability and remaining useful life (RUL) of the structure given the state of SHM from levels 2 and 3.

Narrowing this down to functional level, the work will focus on SHM levels 2 and 3 only, because these tasks are the most relevant to NDT and Lamb wave.

2.1. Recent Development on Lamb Wave SHM

This section will focus on recent developments in Lamb Wave SHM, and it will be divided into three sub-sections: a review on recent developments of

experimental and simulation work on Lamb Wave SHM in section 2.1.1, and the parameter optimization of SHM systems in section 2.1.2.

2.1.1. Technological Advancement on Lamb Wave SHM

The fundamentals of guided wave propagation in solids have been described extensively in literatures, for instance in [Cesnik and Raghavan (2009), Su and Ye (2009)]. When talking about recent advances in experimental and simulation work in Lamb wave SHM, there are many research directions that have been paved for the last decade as well. For example, looking into older work, [Andrews et al. (2008)] investigated Lamb wave propagation behavior in various temperature ranging from -18°C to 107°C in a metallic plate. They concluded that, at least for metallic plates, only small changes occur in the observed waveform. A similar study was also performed by [Dodson and Inman (2013)] for a temperature ranging from 20°C to 70°C and they came to a similar conclusion delivered by [Andrews et al. (2008)]. Some of the results from [Dodson and Inman (2013)] are given in Fig. 2.1.1-1.

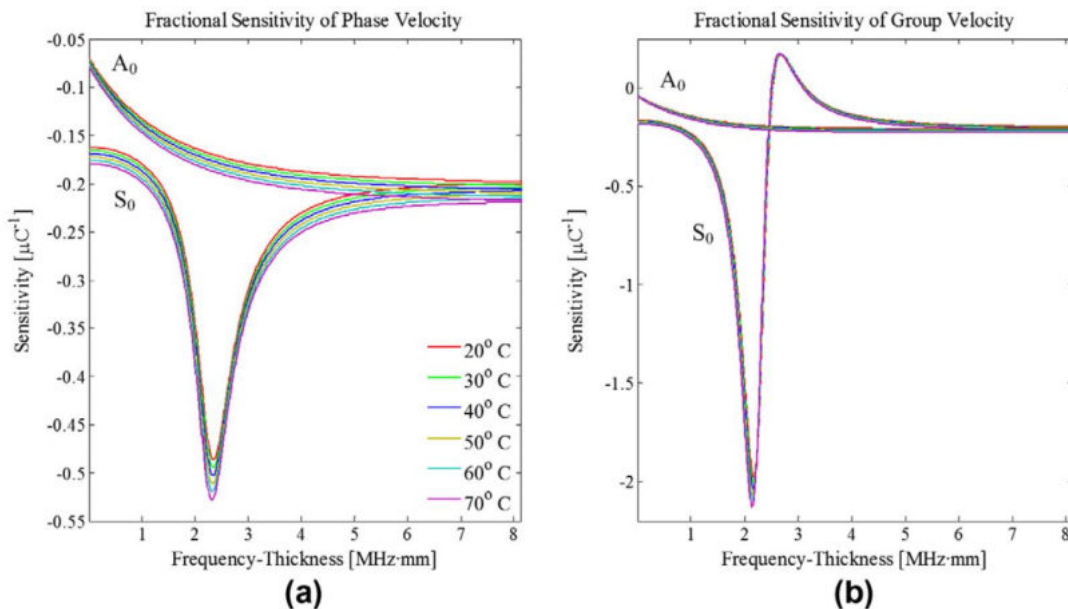


Fig. 2.1.1-1: Effect of temperature on a). phase velocity and b). group velocity as a function of frequency thickness on Al6061. [Dodson and Inman (2013)].

A more comprehensive study that also includes moisture, external vibration, and bonding condition beside the thermal effects was delivered by [Gorgin et al. (2020)]. They stated that the temperature effects can be compensated by various techniques. This is especially useful for many composite materials as composites are affected by moisture, bonding defects, and shear lag effect between the structural surface and the sensor. They stated that vibration only modifies the amplitude and can be filtered so that the time-of-flight (TOF) is not affected. This condition is obviously valid only if the operational vibration has a much lower frequency than those of ultrasonic signals. A similar study regarding the effects of structural complexities combined with environmental conditioning on anisotropic composite materials has been performed by [Schubert et al. (2014)].

The accuracy of analytical modeling based on wave function expansion and the Born approximation of Lamb wave scattering at delaminations in multilayered isotropic plates has been studied by [Ng (2015)]. The result was cross validated with an explicit numerical method and a good agreement was found. It was concluded that delamination can be modeled as a waveguide divided into two sub-waveguides. [Šofer et al. (2018)] discussed the numerical solution for Rayleigh-Lamb waves in complex wavenumbers to describe the wavemodes propagation.

[Gao et al. (2020)] studied the response features of second-harmonic generation of Lamb wave propagation to the thickness of microdamage layer in a solid plate. They verified the analytical results with a finite element simulation and a close agreement between the theoretical and numerical analysis was found. As a conclusion, they claimed that their findings provided a convenient means for accurate characterization of inhomogeneous microdamage in layered plates. [Wu et al. (2020)] investigated the surface effects of Lamb waves propagation in a nanoplate. They combined classical wave theory with the surface elasticity theory. They found that for a very thin layer, even in fundamental modes, the Lamb wave phase velocity increased if the plate thickness decreased.

It is not only the physical understanding and analytical modelling that have been an important subject. Transducer technology is also important, e.g. [Gu and Wang (2009)] investigated the feasibility of a monolithic polyvinylidene fluoride (PVDF) based transducer for generating and sensing Lamb waves, citing that PVDF is not only low-cost, but also low-power which makes it suitable for wireless-sensing and remote operation. [Othmani and Zhang (2020)] investigated the influence of initial stresses of the Lamb wave propagation in PVDF. They formulated the Legendre polynomial to calculate wave propagation characteristics and validated the analytical approach with simulation results. They concluded that for the design of a PVDF sensor, the following parameters must be considered: initial stress, laminate thickness and mass density, as they are the main factors that determine the phase and group velocities of the corresponding Lamb wave modes.

In numerical work, [Malinowski et al. (2009)] introduced the phased-array method for damage localization on a thin-aluminum plate based on the spectral element method (SEM) and reconstructed the wavefield scan. While they concluded that the damage detection and localization has been effective, the technique was purely based on the numerical method and further experimental work would be needed to validate the work. Another approach to using SEM has been proposed by [Hu and Zhou (2012)] where they modelled a transverse crack as a massless spring which was derived from a basic formulation in fracture mechanics. While they compared the results with conventional finite element analysis, further experimental work is also needed to validate their approach.

[Ambrozinski et al. (2010)] compared the numerical approach from local interaction simulation on a sharp interface model (LISA/SIM) and the

elastodynamic finite integration technique (EFIT). The difference between LISA and EFIT is that LISA is based on a finite differentiation approach, while EFIT eliminates the differential problem by using staggered grid. They concluded that both methods can be combined with experimental measurements to solve inverse material problem identification. The results of the numerical approach by employing EFIT has also been replicated by [Rappel et al. (2014)].

A different approach on numerical modelling was proposed by [Gravenkamp (2014)] in his doctoral research. While he used SEM for the latter part of problem discretization, he formulated the scaled boundary FEM (SB-FEM), which can be regarded as a semi-analytical method as only the element boundaries needed to be discretized. The work was experimentally validated in several different materials setup. The results were quite promising in the way that his formulation saves computational cost by a factor of 100 in comparison with traditional FEM.

A simpler simulation on Lamb wave propagation by using FEM on commercial software such as ABAQUS in metallic plate has been proposed [Nirbhay et al. (2017), Ding et al. (2018), Ismail et al. (2019)]. These are similar studies to what I performed before [Ewald et al. (2015)] along with several other studies that also involve composite plates [Chen et al. (2012), Wang (2014), De Luca (2016, 2018), Duan (2017)]. While these studies are not particularly novel, the method of using commercial software turns out to be reliable and replicable without involving highly complex analytical formulations that could take years to be developed. [Miguel et al. (2019)] proposed a finite element framework based on Carrera's unified formulation for simulating Lamb wave propagation in continua by using the higher-order polynomial approximation. The essence of their work was to look for a compromise between classical plate theory that is insufficiently accurate for complex geometry and common quadratic 3D finite elements that quickly takes very high computational effort for relatively large structure.

For signal processing and damage detection, there have been many works performed during the last decade. The generalities and theoretical fundamentals on Lamb wave signal processing has been thoroughly described by [Su and Ye (2009), Staszewski and Sohn (2009)]. They discussed the signal processing based on time-domain, frequency-domain, and time-frequency domain. nevertheless, Lamb wave signal processing for damage detection remains difficult to be solved by single 'catch-them-all' techniques since it involves complexity, uncertainty, and variability as stated by [Harley et al. (2017)] who proposed several methods that decompose into variability reduction, uncertainty analysis by various techniques such as sparse wavenumber synthesis, and complexity leverage by using baseline subtraction, matched field processor, and delay sum and sparse method [Nokhbatolfoghahai et al. (2019)].

Chronologically, when looking back into older works, [Soma-Sekhar et al. (2006)] used Lamb wave tomography to demonstrate the detection capability of their system on low-velocity impact damage on quasi-isotropic graphite-reinforced epoxy matrix composites. One way to characterize Lamb wave signals is to

represent signals in the time frequency domain, also known as a Time-Frequency Representation (TFR). There are several methods to do this: Short-Time Fourier Transform (STFT), Wigner-Ville distribution (WVD), Chirplet Transform (CT) and Wavelet Transform (WT) [Su and Ye (2009), Kerber et al. (2010), and Kordbacheh (2012)].

The difference between these techniques is the function that is multiplied by the time domain input signal. For instance, while in STFT the time domain input signal is multiplied by a fixed window length, in WT, the input signal is multiplied by a scalable mother wavelet. A very detailed approach on characterizing Lamb wave signals with that TFR with Matching Pursuit is described by [Karpenko (2013)] in his thesis, where the proposed TFR is designed on the basis of the reassigned spectrogram in order to improve the resolution. The work was experimentally validated on aluminum and woven composite plates.

[De Marchi et al. (2013)] proposed a novel signal processing for chirp excitation with three steps: 1). warped frequency transform (WFT) to compensate the dispersion due to propagation, 2). signal compression to remove the frequency modulation, and 3). an imaging algorithm to localize the damage. This approach is related to that which has been proposed by [Hua et al. (2015)]. The difference in the study is, that they used more pulse variations that include white noise signal in their chirped excitation.

[Ahmad (2014)] characterized Lamb wave propagation signals in thermoplastic materials using the Daubechies 4-tap (db4) based wavelet algorithm and Gabor Transform to determine the experimental group velocities and by using this technique, they were able to identify higher-order Lamb modes such as S10, A7, A9, and A10 modes. [Chen and Wang (2014)] introduced a technique for signal noise removal by calculating the fractional differential amplitude and extracted the amplitude spectrum with estimation model. They validated their study with simulation and experimental method.

From imaging techniques, [Gao et al. (2019)] introduced sparse reconstruction imaging from contactless Lamb wave excitation by a laser and reconstructed the signal using system response decomposition. While the sparse imaging result is not smooth, the achieved damage localization accuracy was below 1 cm. They concluded that the performance of this sparse imaging technique relies on the denoising parameters and the number of the excitation sources.

Lamb mode conversion is another area of research direction that has been established for a while. Typically, in this case, the work involves the characteristic of mode conversion to detect or localize the damage, as has been proposed by [Hosseini et al. (2014)]. In their work, they used the scattered coefficients from continuous wavelet transform (CWT) to separate the different Lamb modes in sandwich structures. As a conclusion, a damage localization with a maximum error of 2 mm was obtained.

In area of a developing damage index (DI), [Gao et al. (2018)] proposed an integrated impedance technique for damage classification, i.e., instead of the amplitude, they indirectly derived the DI by monitoring the impedance variation of the sensor network and then, the DI is calculated by adjusted result of the sensor self-diagnostic parameter to estimate the damage severity. This method is particularly useful to monitor 1) the structural health, and 2). the health of the sensor network itself.

We can summarize the literatures above into 4 distinct research directions: 1). physics & analytical modelling on Lamb wave based SHM, 2). signal processing for damage characterization, 3). numerical approach on Lamb wave propagation, and 4). sensor technologies for Lamb wave generation and sensing.

2.1.2. Advancement on Strategic Design of SHM System Parameter

The focus of the section 2.1.1 was on the scientific and technological novelty and advances of Lamb wave based SHM that might (or might not) come into closer realization of fully functional Lamb wave based SHM, i.e., those are still within the laboratory realm or smaller scale development level. This section will be focused more on design parameters that are relevant for larger-scale SHM deployment.

One of the very early studies on Lamb wave design parameters was the mode selection performed by [Rose et al. (1993)], where at that time, they were only talking about ultrasonic NDT a with guided wave for a composite plate, but of course without mentioning SHM since at that time, the term SHM was most likely not heard yet. The topic of wave mode tuning techniques was revisited by [Shi (2002)] in his doctoral research, where he described different methods such as single transducer tuning, phased array tuning, and synthetic phase tuning.

The reason why wave mode selection is important is due to the fact that unlike bulk waves, Lamb waves have a dispersive and multi-modal nature, i.e., not only that the propagating velocity changes as a function of frequency, but also that at least two fundamental Lamb modes are always present. In fact, signal processing of Lamb wave for damage detection is often hampered by the presence of undesired higher-order Lamb modes that must be separated first by many techniques such as those discussed in section 2.1.1. The complexity and difficulty of analyzing more than just the fundamental Lamb modes was reiterated by [Wilcox (1998)] in his dissertation.

Over two decades, the topic on wave mode selection has been revisited several times. In the slightly later article, [Wilcox et al. (2001)] pointed out that 6 crucial factors influence propagating Lamb modes: dispersion, attenuation, sensitivity, excitability, detectability, and selectivity. Also, they mentioned that a wedge transducer is generally unsuitable for a liquid-surrounded structure, and they recommended using either an electromagnetic acoustic transducer (EMAT) or a shear PZT instead. It was pointed out by [Santoni et al. (2007) and Giurgiutiu et al. (2007)] that the Lamb mode tuning can be influenced by the dimension of the

transducer, where they demonstrated this tuning with several varying transducer dimensions and calculated the effective dimension to compensate for the shear-lag effect between the sensor and surface due to bonding, and finally processed the captured signal with the time-reversal method.

A further study on calibration methods for a circular shaped PZT for Lamb wave tuning has been published by [Sohn and Lee (2010)]. They first constructed the theoretical tuning curve due to discrepancy between the bonding layer and the energy distribution: After that, they incorporated the energy distributions among the symmetric and antisymmetric modes and validated the effectiveness of the calibration method by using numerical and experimental work on an aluminum plate. An example of such a tuning curve before and after their proposed calibration is depicted in Fig. 2.1.2-1.

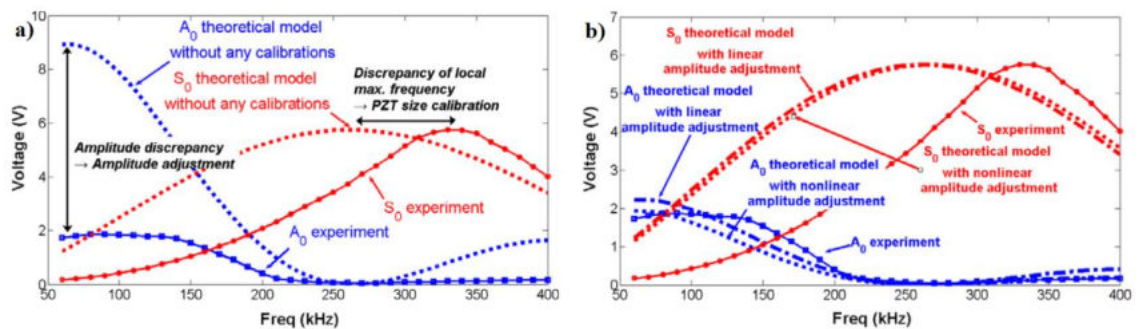


Fig. 2.1.2-1: Discrepancy between the experimental and theoretical tuning curves a). before and b). after calibration - which can be either a linear or a non-linear adjustment. [Sohn and Lee (2010)].

Other approach to generate specific Lamb modes has been proposed by [Yan and Bo (2011)]. Instead of determining the PZT dimension, they used a transducer pair on the top and bottom surface of an aluminum plate. Such an approach would allow weakening of one Lamb mode while strengthening the other, so that only a single dominating Lamb mode is propagating the plate. While they validated their numerical modelling with experimental work, such approach is not very practical for some applications in aircraft crack monitoring because it would require one of the sensors to be placed on the outside.

[Schmidt et al. (2013)] proposed a design for an interdigital transducer that consists of multiple arrays of positive and negative electrodes that are separated by wavelength distance. They first analyzed the broadband excitation frequency range between 25 - 400 kHz and validated the work on a carbon fiber reinforced composites plate accompanied with finite element modelling in the ANSYS environment. Unlike a conventional PZT which is in general very brittle, the interdigital transducers are flexible in shape and suitable to many geometries, however the disadvantage of this type of transducer is its relatively higher price and its relatively large dimension (circa 5 x 5 cm) which might not be suitable for hardly accessible areas closer to fasteners.

A related work on Lamb mode phase matching has been proposed by [Li et al. (2020)]. They first described the theoretical analysis of a second harmonic Lamb

mode in an isotropic plate and went further into the derivation of non-linear parameters for Lamb waves which were later adjusted to the phase-matching condition due to attenuation (see Fig. 2.1.2-2). Based on their experimental validation, it can be observed that higher S-modes have a tendency of amplitude perseveration, i.e., better energy efficiency.

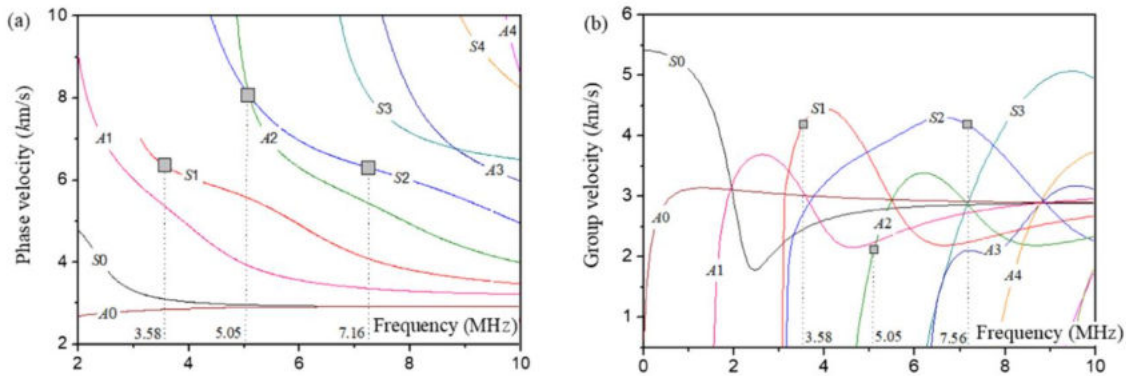


Fig. 2.1.2-2: Phase matched Lamb modes indicated by square symbols, in a). phase velocity dispersion curve and b). group velocity dispersion curve. [Li et al. (2020)]

Another parameter that can be fine-tuned for Lamb wave based SHM is the excitation waveform. There were not many works performed in this area: study on waveform design performed by [Zeng et al. (2013)] for improving resolvable resolution can be briefly mentioned. They stated that besides sensor distance, both the number of excitation cycles and the center frequency are significant factors that influence the resolvable resolution.

Another approach on frequency mixing has also been recently proposed by [Chen et al. (2020)]. In this work, they studied the response of frequency mixing from two counter-propagating Lamb waves in a two-layered plate. They found that outside the wave mixing zone, the magnitude of the combined harmonic tends to be stable and due to the relatively small size of the wave mixing zone, this technique can enhance the localization accuracy of debonding in the two-layered plate. They performed a semi-analytical model and verified this in numerical modelling via FEM, but an experimental validation still must be conducted to validate the proposed approach.

As for the design of the SHM power system, there have not been many approaches proposed: One work from [Kural et al. (2013)] describes the design and optimization of the transmission and reception circuits when using inductors. They first characterized the power consumption of the sender-receiver system in a broad excitation range between 20 and 200 kHz to determine the minimum power supply threshold, and then compensated the power throughput by using the inductor system. Conclusively, they found out that the power throughput of the system can be increased 5-fold.

Another important parameter that influences damage detectability in Lamb wave based SHM system is the PZT pattern and positioning. Depending on the detectability, performance, and cost-benefit requirement, the sensor network

arrangement can be either sparse or dense. [Croxford et al. (2007)] mentioned that for SHM, a sparse sensor network arrangement is suitable – which is very logical since the complexity and redundancy of a dense SHM sensor network would outweigh its cost-benefit performance and thus in that case an inspection by conventional NDT would suffice. Further, they mentioned that for a sparse network, the distance between the sensors is ideally far larger than the scale of the anticipated damage.

An example comparison between a sparse and a slightly dense sensor network was presented by [Ambrozinski (2012)] where they compared the damage imaging reconstruction from a sparse and a star-shaped Phased-Array sensor network. They conducted the experiment on an aluminum plate with a size of 1000 x 1000 x 2 mm and concluded that due to the complexity of aircraft structures, a baseline imaging needs to be subtracted from a reconstructed image of the damaged structure. As the study was quite in an early stage, they mentioned that further work addressing environmental condition such as temperature needed to be performed.

In a more recent article from [Kudela et al. (2018)], a novel strategy to improve imaging resolution was proposed. In this case, they used a circular sensor pattern placed in the same geometry and material as used in [Ambrozinski et al. (2012)]. In this work, two parameters were studied: the excitation waveform and the sensor density. The waveform excitation they used was also based on pulse compression as has been discussed earlier in [Lin et al. (2016)]. Their strategy for sensor positioning was to start with a dense network (assigned as a focal point), and then slightly increase the sensor distance based on signal post-compensation and the sum of the energy.

[Lee and Staszewski (2007)] used the local interaction simulation approach (LISA) to model a small number of damage scenarios and based on the result of the simulations, the locations with the highest peak to peak locations were identified as suitable locations for sensor placement. A related approach using simulation of Lamb wave propagation was proposed by [Venkat et al. (2016) and Taltavull et al. (2017)]. In this approach, the summed-up energy captured by all the individual sensors was plotted and the most optimal sensor location was determined as the one with the highest captured energy.

A similar approach was realized in an experimental setup by [Stawiarski and Muc (2019)]. However, instead of the energy, they calculated the damage index (DI) based on the correlation coefficient between the baseline signal and the signal from the defected structure. [Fendzi et al. (2014)] proposed a novel approach for sensor placement using geometric dilution of precision (GDOP), which is based on a Lamb wave ray tracing method for known damage locations. [Haynes et al. (2014)] proposed sensor placement by minimizing the Bayesian cost and thus selected the locally optimal sensor location. However, if the damage occurs outside of that area, it might fail to detect it.

[Mallardo et al. (2012)] proposed a hybrid probabilistic approach using a combination of a Genetic Algorithm (GA) and an Artificial Neural Network (ANN), where they related the fitness function to the approximate error of the ANN. This approach takes a very dense network into consideration and seems to be suitable for monitoring stringers and frames, but at the same time it can be considered an overkill and not very cost-efficient for monitoring an impact in an open area.

In more recent study, [Thiene et al. (2016)] introduced DI-free sensor placement optimization based on a fitness function that maximizes the coverage area of the sensor network. They calculated the coverage of each pixel in the geometry based on the pitch-catch technique, so that every pixel that contributes to the probability that a damage in a random location is being detected is counted. Their goal was to maximize the coverage area of the sensor network. A related approach on maximizing coverage area was recently proposed by [Soman et al. (2019), Soman and Malinowski (2019)] and [Mustapha et al. (2019)]. Some of these techniques are also mentioned in a recent review by [Ostachowitz et al. (2019)]. Some figures from these works are collated together in Fig. 2.1.2-3.

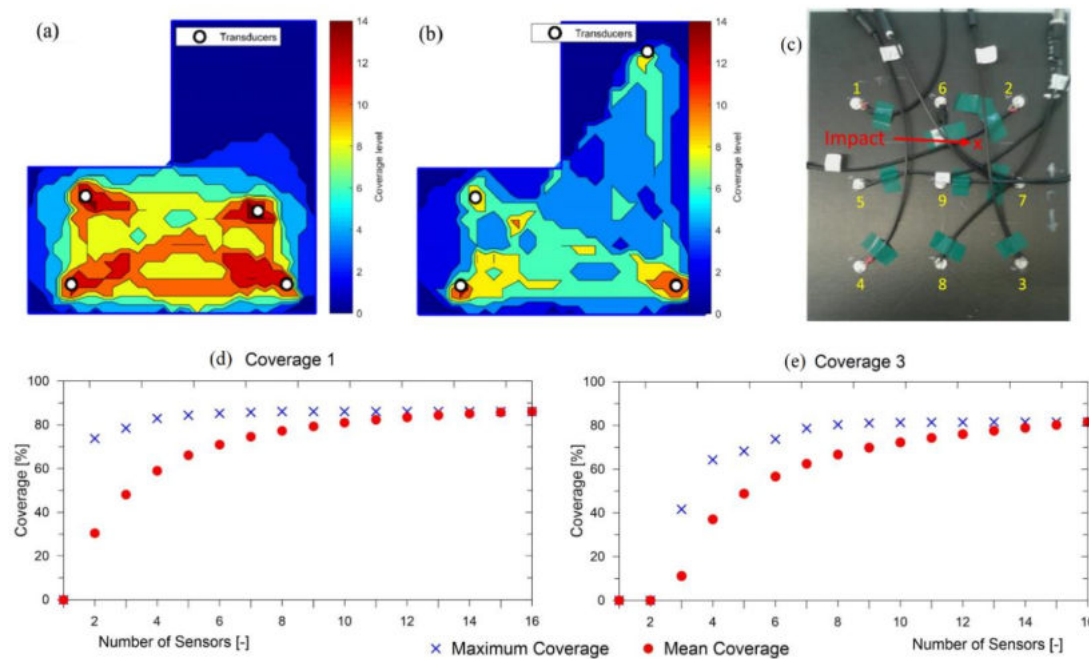


Fig. 2.1.2-3: From [Thiene et al. (2016)]: Comparison two different sensor network topology of 4 sensors, a). Denser network in lower part of plate, b). balanced network, c). experimental setup. From [Soman et al. (2019)]: Network score convergence as function of sensor density for d). highest (coverage 1), and e). lowest (coverage 3) sensor-actuator pair coverage, respectively. Detailed definition of coverage nomenclature can be read in the paper.

In a recent work from [Balamonica et al. (2020)], a study on sensor networks based on serial and parallel connections was performed. They conducted the work for a reinforced concrete beam to understand dynamic damage quantification metrics such as the moving root mean square deviation (M-RMSD), the moving cross-correlation (M-CC), and the moving mean absolute deviation (M-MAD). The beam was subjected to damage and was measured by the sensor response conductance with varying frequencies for different sensing paths and

it was concluded that the sensor node with the higher magnitude in the frequency domain is the one which is closer to the damage.

Another parameter that can be optimized to increase the efficiency and detectability of an SHM system is the geometry of the structure itself as has been proposed by [Ong and Chiu (2012), (2013)]. While the idea of optimizing the geometrical aspects for current damage tolerant design is quite wild, this approach would make sense for a new structure which is to be designed from scratch. In their work, they proposed fatigue crack detection and localization using ray tracing in a specially designed sub-structure within an aircraft wing that specifically redirects Lamb wave propagation. The study was performed in a FEM environment and experimentally validated.

As a conclusion for this sub-chapter, the design parameters of Lamb wave SHM that should be considered when designing such system has been discussed by [Carboni et al. (2015)] where they employed statistical approaches by Design of Experiment (DOE) to study the main influencing factors such as Lamb mode tuning, frequency combination, pulse-echo and pitch-catch configuration, and sensor position for damage detection in carbon fiber composites.

2.2. Recent Trends in Machine Learning and Computational Intelligence

This section contains the recent trends and advances in machine learning, evolutionary computing in heuristics optimization, and give some perspective on computational neuroscience that might be adapted to the development of autonomous systems in the sub-section 2.2.1 – 2.2.3, respectively.

2.2.1. Advancement in Machine and Deep Learning

Since the scope of machine learning is very broad and in recent years it is getting more and more chaotic than ever, before ‘getting lost’ in the jungle of machine learning terminology, we shall understand the existing paradigms of machine learning [Mohri et al. (2018), Karimpanal and Bouffanais (2018), Settles (2010), Sahoo et al. (2018)]: supervised learning, unsupervised learning, semi-supervised learning, self-supervised learning, feature learning, reinforcement learning, active learning, and online learning:

- **Supervised learning:** This is most probably the most well-known learning scenario. In supervised learning, the algorithm receives a set of labeled examples as training data and makes predictions for all unseen points. The most common applications that use machine learning are image recognition, weather prediction, and spam email detection.
- **Unsupervised learning:** In this case, the learning algorithm exclusively receives unlabeled training data and estimates whether there are relations between each point. Generally, the performance of an unsupervised learning algorithm is difficult to measure since in general no labeled example is available in that setting. An example of unsupervised

machine learning techniques is clustering for news category grouping and principal component analysis (PCA) for dimensionality reduction.

- **Semi-supervised learning:** This approach falls between unsupervised and supervised learning. The learning algorithm receives a training sample consisting of both labeled and unlabeled data and makes predictions for all unseen points. This approach is very common in settings where unlabeled data is easily accessible, but labels are expensive to obtain, which reflects many real-world situations.
- **Self-supervised learning:** Like semi-supervised learning, self-supervised learning also falls between supervised and unsupervised learning. The difference here is, self-supervised learning makes use of completely unlabeled data, but it tries to find the relationship between each data point and tries to map that relationship in a supervised way. This is commonly used in the robotics domain where a self-supervised algorithm learns the relation between multimodal data from a color and a depth camera.
- **Feature learning:** Sometimes also called representation learning, feature learning allows a system to automatically discover the representations needed for feature detection from raw data. Learning representations replace hand-engineered features and allow the algorithm to both learn the features directly from the raw data and to use them to perform a specific task. An example of feature learning is a face recognition software that detects biometric markers such as eyes and mouth location.
- **Transfer learning:** Also associated with feature learning and deep learning, transfer learning makes use of the learned feature representations. Since a deep neural network can only perform a specific task and it must be retrained if the task is changed, the learned features are sometimes transferred into a similar task in order to save training time.
- **Online learning:** Online learning is a subset of supervised learning, which involves an 'online scenario' that consists of multiple rounds of intermixed training and testing phases. However, unlike classical supervised learning, the online scenario is not done in a mini batch over an entire dataset, but rather in sequential way as soon as new data points are discovered.
- **Active learning:** Like online learning, active learning is a special case of supervised learning. The active learning algorithm interactively collects training data, typically by querying a human to request labels for new points. The goal in active learning is to achieve a performance comparable to the standard supervised learning scenario, but with fewer labeled examples.
- **Reinforcement learning:** Like active learning, a reinforcement learning algorithm actively interacts with the environment. It receives an immediate reward or penalty for each action. The objective of reinforcement learning is to maximize the reward over a course of actions and iterations with the environment. However, unlike many other machine learning scenarios, reinforcement learning does not yield a statistical model like a neural network, but rather a policy that maps the agent action to maximize its reward.

Supervised learning is most probably the most common and often talked about in machine learning and when introducing someone to machine learning, the first method that will be introduced would probably linear, polynomial, and logistic regressions as these simple techniques have existed since centuries. Indeed, almost all modern supervised techniques are at least involving regression. The most simplistic supervised algorithm would probably be a simple (also called naïve) Bayesian classifier that can also slightly be expanded into multinomial Bayes classifier. Already in 1997, [Domingos and Pazzani (1997)] discussed the optimality of a Bayes classifier, where they show it can be optimal under hinge loss. There are already enough works on Bayes classifiers and therefore will not be elaborated further here.

Another popular technique in supervised learning is called Support Vector Machine (SVM) which was introduced by [Cortes and Vapnik (1995)] as a support-vector network for pattern recognition. The original SVM was basically a binary classifier that used an optimal hyperplane which maximizes the margin in multidimensional space. In a simplified case where the data points are linearly separable, it can be simplified as a linear SVM. The work has been extended by [Lee et al. (2004)] to include multiclass classifier SVM and its statistical theory has been recently provided by [Pouliot (2018)].

The most popular and heavily researched supervised learning method is probably neural network, which was introduced for the first time by [Rosenblatt (1958)] as a perceptron and later emerged as a network and thus called multilayer perceptron (MLP). Since MLP is an imitation of a biological neural network, it is also sometimes called artificial neural network (ANN). MLP grew in popularity until the 1990's [Hopfield (1982)], where at that time the computational ability was low, and the ANN popularity was overshadowed by SVM [Lee and To (2010)].

In between, there were periods called the 'AI winters' where research in artificial intelligence suffered a lack of funding and interest, and these were notoriously between the early to late 1970's, between the late 1980's and the early 1990's, and after the dotcom bubble collapse in the early 2000's. It was not until the mid-2000's when [Hinton and Salakhutdinov (2006), Hinton et al. (2006)] introduced the deep belief network (DBN), a class of DNN, when the neural network regained popularity. Post-2012, people were increasingly talking about 'deep learning' but actually it is just a network with at least 2 hidden layers, so technically a 2-layered MLP from the 1970's would also qualify to be called deep learning.

While MLP was powerful enough to recognize the handwritten digit even in the early 1980's, it was still far away from recognizing the human face or detecting an object within an image. Not only was MLP memory consuming, but at that time there was not enough computational power to train a huge MLP. Moreover, a complex MLP was very prone to the overfitting problem. To overcome the problem, a convolutional neural network (CNN or ConvNet) called LeNet-5 was introduced by [LeCun et al. (1998)], depicted in Fig. 2.2.1-1.

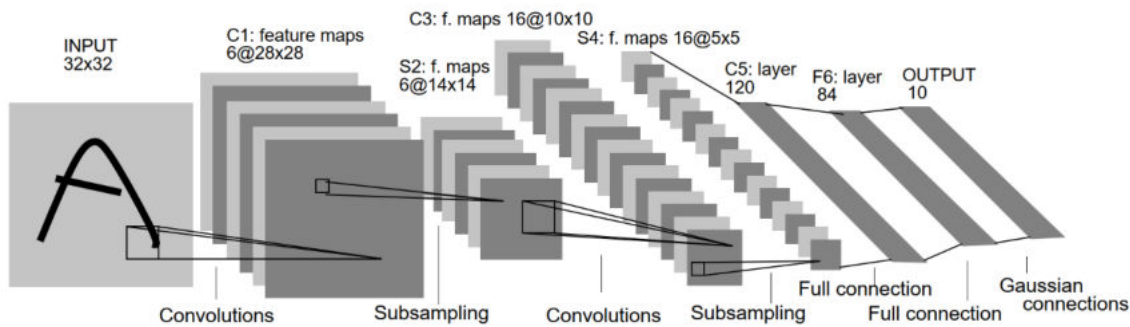


Fig. 2.2.1-1. Original LeNet-5 [LeCun et al. (1998)]

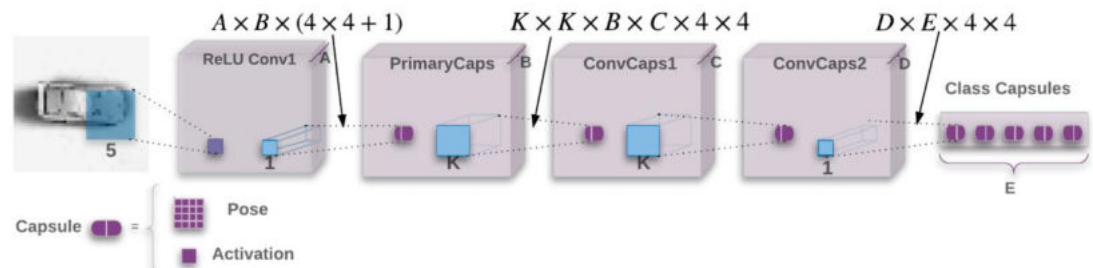
The idea behind CNN was to capture local spatial information within the input image, thus it was designed as a shared network parameter within a convolution filter that slides from the top left to the bottom right of the input image. The input representations were subsampled into several feature maps and this step was repeated in several times until a very deep abstraction level of the original input image was reached. Finally, the feature maps were attached to a smaller, but fully connected MLP at the end. LeNet-5 was published coincidentally during the peak of dotcom bubble, however as one can expect, there was a short period of AI winters afterwards. From 2010 onwards, research in CNN were flourishing. [Krizhevsky et al. (2012)] introduced AlexNet, a deeper and larger version of LeNet-5. They also trained it on GPU and thus accelerated the training process. Unsurprisingly, AlexNet was the top performer in ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2012. Since then, there exist many variants of CNN:

- 16- and 19-layers deep network from Visual Geometry Group of Oxford University, called VGG-16 and VGG-19 [Simonyan and Zisserman (2015)]
- 152-layers deep CNN with skip connections called Residual Network (ResNet) [He et al. (2016)]
- CNN a variational convolution filter size from the Google team, called GoogLeNet which also known as InceptionNet (22-layers deep) and its younger brother InceptionNet v4 and its sibling recombinant Inception-ResNet [Szegedy et al. (2015), (2017)]
- a smaller but efficient CNN for smartphone and embedded vision application called MobileNet [Howard et al. (2017)]
- similar to MobileNet: SqueezeNet [Iandola et al. (2017)] that achieved AlexNet-level of performance but only with less than 1 MB size.

One of the major breakthroughs in CNN was proposed by [Hinton et al. (2018)] in what they called as dynamically routed capsule network (CapsNet). In normal CNN, a pooling layer is typically attached after each convoluted feature map to reduce feature redundancy, but this causes information loss regarding object pose and location. So, a CNN would recognize an image that consists of eyes, mouth, nose, and ears as perfect human face even if the mouth is placed on the top of the eyes. This is known as white attack, but it is to be expected because the convolution kernel from a CNN only learns the local spatiality of an object (e.g., eye color, pupil, lens within an eye) but it does not learn the eye position relative

to the human face. CapsNet addressed this issue by learning an equivariant representation of the object, i.e., it tracks the object movements just from several samples and concatenate them in a capsule. Instead of a common non-linear activator such as sigmoid or linear rectifier, the capsule is routed to the next layer by a squash function. The architecture of CapsNet and the equivariant object representation is depicted in Fig. 2.2.1-2 a - c.

a). Architecture of CapsNet



b). 8 sample images that are fed into CapsNet



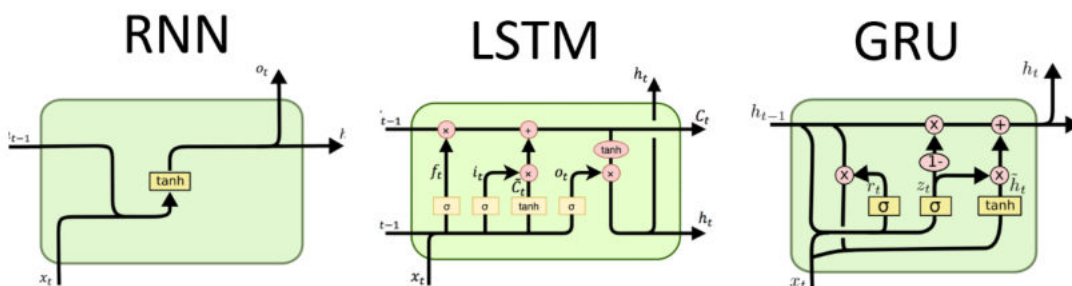
c). Artificially generated image from learned parameters



Fig. 2.2.1-2: a). Architecture of CapsNet, b). 8 sample images from 4 objects that are fed into CapsNet - each object and its equivariant representation is given in top and bottom, c). Artificially generated images from learned parameters trained with CapsNet. It can be seen quickly that CapsNet can efficiently learn the object pose. [Hinton et al. (2018)]

Another approach that commonly appears with a neural network is sequential data modelling using a recurrent network. This approach is very useful for a data block that has variable length, but also for very long data that has to be treated in sequential mode such as human speech, textbook, or a very long time-series. In a recurrent network, the information is processed in the recurrent cell. There are several variants of core recurrent cells that make up the network: the vanilla recurrent neural network (RNN), the long short-term memory (LSTM) and the gated recurrent unit (GRU) which were originally introduced by [Rumelhart et al. (1986)], [Hochreiter and Schmidhuber (1997)], and [Cho et al. (2014)], respectively. The basic recurrent cell operations and the model architecture of a generic recurrent network is depicted in Fig. 2.2.1-3.

a). Recurrent Cell Operations



b). Recurrent Network Architecture

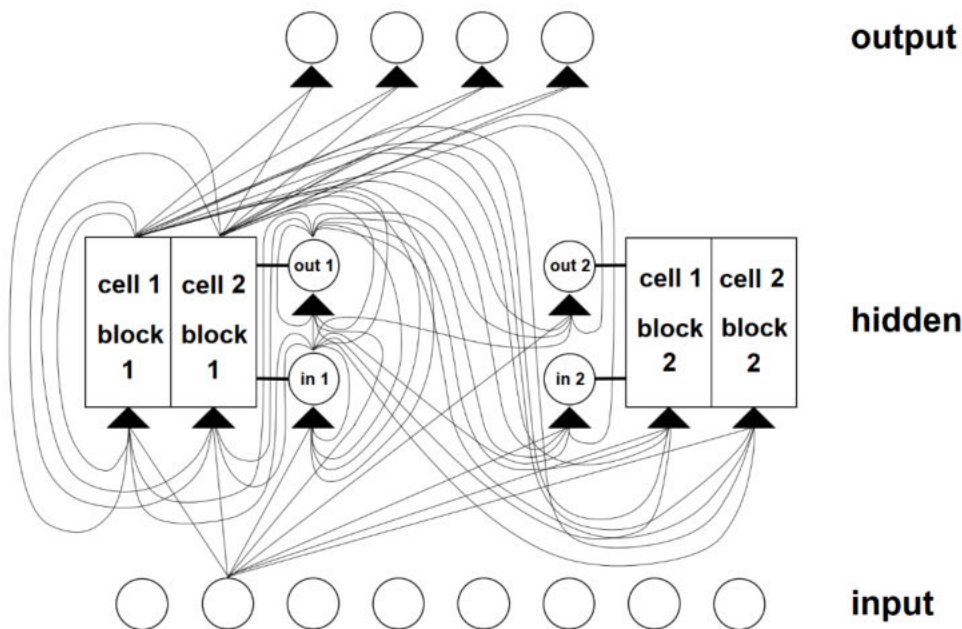


Fig. 2.2.1-3: a). Recurrent cell basic operations: Vanilla RNN, LSTM, and GRU [Rathor (2018)]. b). Recurrent network architecture [Hochreiter and Schmidhuber (2017)]. Every cell at hidden layer can be either vanilla RNN, LSTM, GRU, or combination of these.

While the theoretical detail is too complex to discuss in this literature review, it can be pointed out that the similarity of these networks is that the input into the recurrent cell not only comes from the input layer, but also from preceding output layer. This is in contrast with a feed-forward neural network (such as MLP

and CNN) which only gets its input solely from the input layer. The advantage of LSTM and GRU over vanilla RNN is that the GRU and LSTM cells have memory and thus are more suitable for applying to a long-term dependency problem such as machine translation. An example that demonstrates the capability of a 4-layers deep LSTM network to handle long term dependency has been performed by [Sutskever et al. (2014)]. They showed that there is no translation degradation for English - French translation for sentences of less than 35 words.

Another possible approach for sequential modelling on recurrent network is the combination between CNN with LSTM. There are two different approaches to this: 1). the conjunct CNN-LSTM where the MLP layer in the standard feed-forward CNN is simply replaced by LSTM cells such as demonstrated by [Xue et al. (2019), Vidal and Kristjanpoller (2020)] and 2). the ConvLSTM method where the convolutional kernel from standard CNN is made recurrent as an LSTM cell such as demonstrated by [Xu et al. (2019)]. For this reason, this can be regarded as a 2D-LSTM cell.

Given the ability possessed by CNN to extract spatial information from the input and the ability of recurrent models such as GRU and LSTM to learn temporal relationships stretching across long periods of time, it is natural to fathom a neural network that is a hybrid of both. The resulting hybrid neural network would enable the pattern learning of the spectrotemporal input in an end-to-end manner.

The first proof of this concept found its first appearance in research dating back to 2015 [Ng et al. (2015), Donahue et al. (2015)]. To best of my knowledge, the first hybrid CNN and LSTM combination was proposed by [Shi et al. (2015)]. Numerous variants of CNN-LSTM hybrid networks have since been proposed in a wide range of technical fields, such as: clinical electroencephalography (EEG) [Xu et al. (2020)], financial forecasting [Lu et al. (2020)], renewable energy production [Tovar et al. (2020)], natural language processing [Sainath et al. (2014)] and, ostensibly of course, computer vision [Ercolano and Rossi (2021)].

Theoretically, a convolutional operation of the kernel can be of arbitrary dimension, although as humans we normally perceive in either 1-, 2- or 3-dimensional space. The simplest example was by given [Xu et al. (2020)] in which they employed 1D CNN-LSTM architecture, in which they used the 1D CNN-LSTM for automatic recognition of epileptic seizures via the analysis of EEG signals. The purpose of the convolutional layer was to extract features from the preprocessed EEG input data, while the LSTM component of the architecture would then subsequently perform the extraction of temporal features. This novel application of 1D CNN-LSTM architecture for a clinical EEG task proved to have much better performance than both traditional machine learning algorithms and simpler deep neural networks and a pure CNN network.

Within the field of field of financial forecasting, [Lu et al. (2020)] used a 1D CNN-LSTM network to predict the price of a stock one day ahead. The tasks of both the

1D CNN and LSTM components are similar to those in the previous case, so unsurprisingly one may view the financial data input into the network as a suggestive signal of a company's potential profitability. As an outcome, they concluded that a 1D CNN-LSTM is capable of reliably forecasting stock prices, which was further demonstrated by comparing the results from alternative models such as multi-layer perceptron (MLP), pure CNN, pure recurrent (RNN and LSTM) and CNN-RNN.

One prominent example from the medical field using CNN-RNN recombination has been performed by [Gheisari et al. (2021)] for detecting glaucoma, a leading cause of blindness. According to their results, the combined CNN-RNN model reached an average F-score of 96.2% while the base CNN model only reached an average F-score of 79.2%. Another work from the medical field has been demonstrated by [Islam et al. (2020)] who used it to automatically diagnose X-ray imagery data of Covid-19 patients. Their experimental results show that the algorithm achieved an accuracy of 99.4% and an F1-score of 98.9%. A visual depiction of their implementation of 2D CNN LSTM is given in Fig. 2.2.1-4.

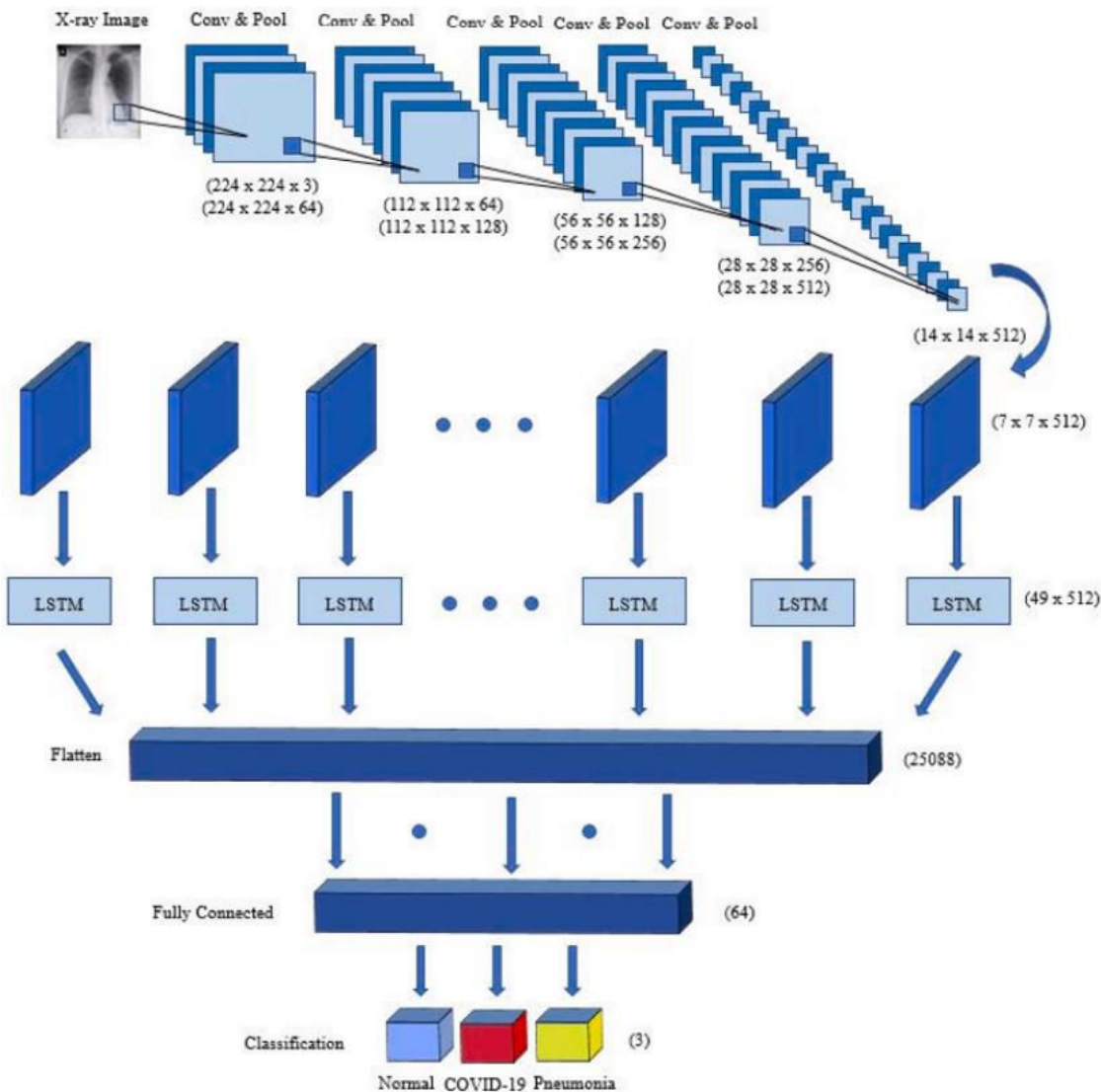


Fig. 2.2.1-4: 2D CNN-LSTM network proposed by [Islam et al. (2020)] to distinguish X-ray imaging data between a healthy (normal) lung, a Covid-19 patient, and a bacterial pneumonia.

Another application of CNN-LSTM was proposed by [Ercolano and Rossi (2021)] who developed an algorithm to recognize daily life activities in the home environment using skeleton data coming from a depth camera. In this case, they had a 3D CNN component which takes as input depth wise grid representations of skeletal data, as depicted in Fig. 2.2.1-5.

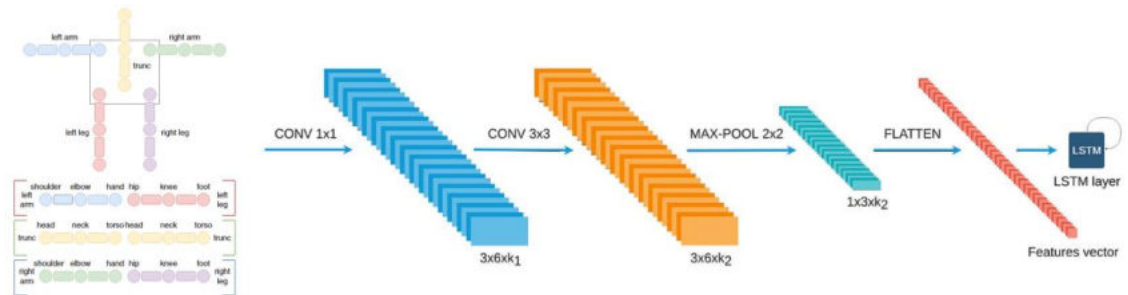


Fig. 2.2.1-5: 3D CNN-LSTM for recognizing RGB-D skeletal data [Ercolano and Rossi (2021)]

[Tovar et al. (2020)] used a similar network to predict electric energy consumption. They used multivariate datasets coming from different sensor data such as atmospheric pressure, humidity, temperature, AC voltage, radiations, etc. A visualization of the implementation of their workflow is given in Fig. 2.2.1-6.

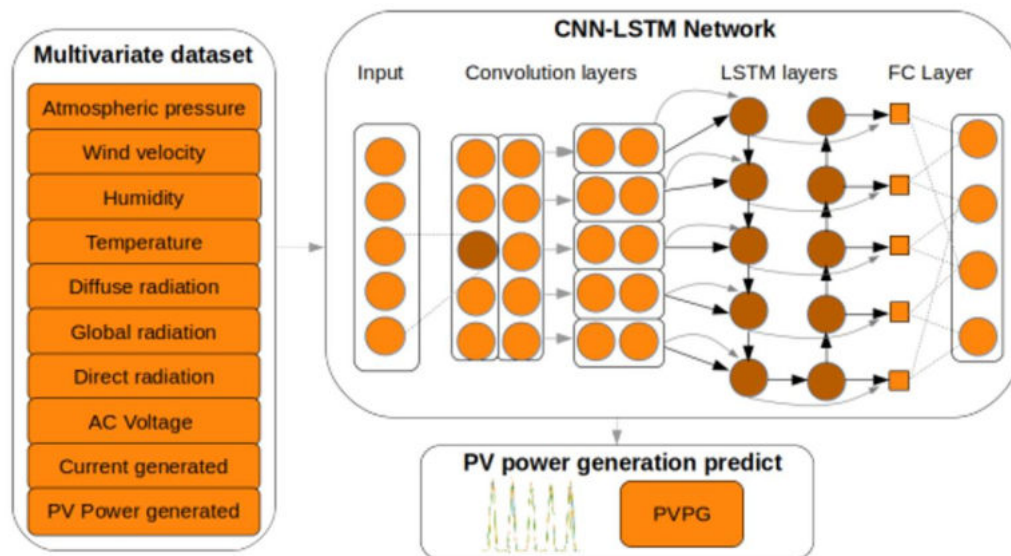


Fig. 2.2.1-6: Visual depiction of the 1D CNN-LSTM implemented work for electricity energy consumption prediction [Tovar et al. (2020)].

Within the field of SHM-NDT, there are few use cases of the CNN-LSTM architecture. One example is the work of [Khorram et al. (2019)] which focused on intelligent bearing diagnostics and prognostics and aimed at reducing unscheduled downtime, performance degradation and hazardous safety matters by detecting bearing faults using CNN-LSTM with the end-to-end approach. In their work, the data is maintained in its raw format without any pre-processing such as a Fast Fourier Transformation (FFT) or Discrete Wavelet Transformation (DWT). Notably, the result achieved by this work demonstrated a level of accuracy higher than that of any other present in literature. They collected raw accelerometer measurements as input data which are fed into the temporal sequence prediction algorithm, as depicted in Fig. 2.2.1-7.

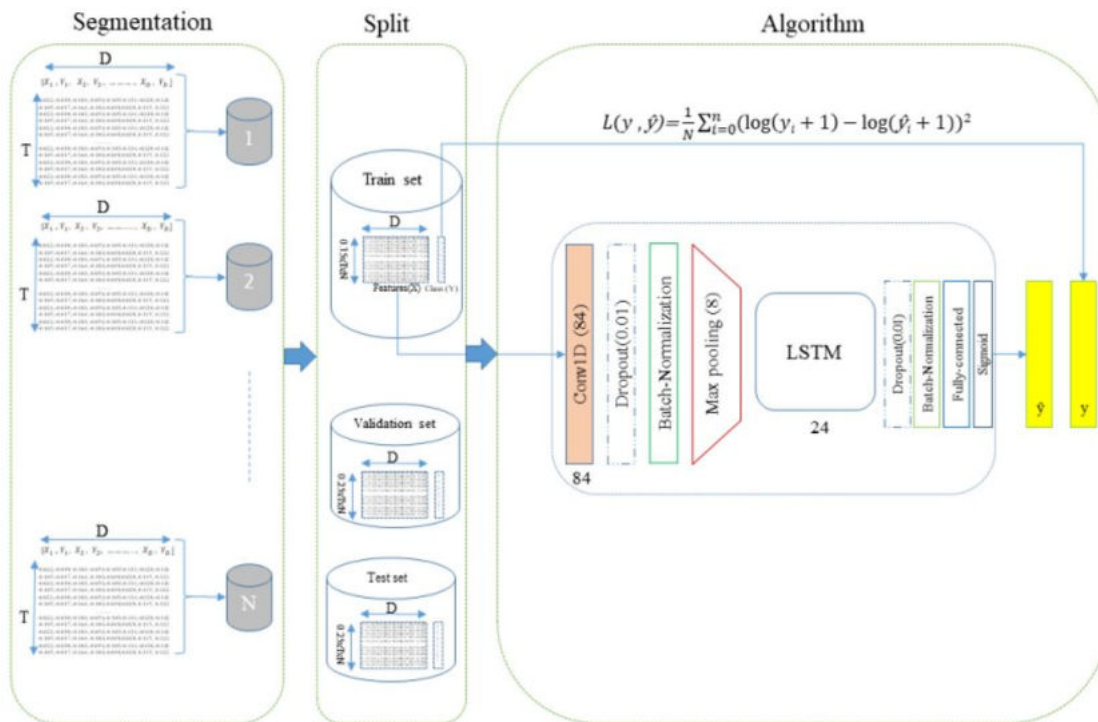


Fig. 2.2.1-7: End-to-end 1D CNN-LSTM approach for bearing fault diagnosis [Khorram et al. (2019)]. The loss function is defined as L .

In the domain of machine translation, a novel approach for neural translation has been proposed by [Bahdanau et al. (2015)]. An LSTM memory cell degrades for a very long sentence. They addressed this issue by adapting the approach of [Sutskever et al. (2014)] as variable length vectors and adaptively choose a subset of these vectors. This is called attention mechanism, and it relieves the computational load from squashing all the information from the source sentence into a single fixed-length vector. Concretely, they used a bi-directional RNN both as an encoder during sequence annotation and a decoder during context searching from the source sentence. An example result is depicted in Fig. 2.2.1-5.

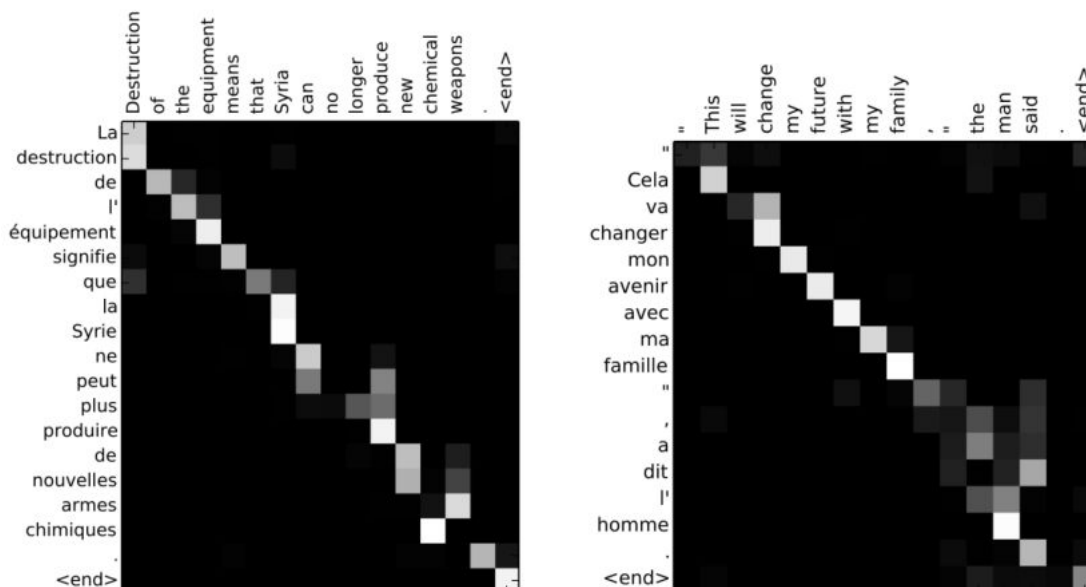


Fig. 2.2.1-8: Example translation result for English - French word matching using approach proposed by [Bahdanau et al. (2015)]

A variation for neural machine translation has been introduced by [Vaswani et al. (2017)] where they called their model Transformer network. While the work of [Sutskever et al. (2014)] relied on sequence aligned recurrence, [Vaswani et al. (2017)] exploited the self-attention mechanism, where the attention mechanism can be related to different positions within a source sentence instead of one. Their model was extended into Pre-trained Bidirectional Encoder Representations from Transformers (BERT) by [Devlin et al. (2018)] by leveraging the bidirectionality of the Transformer network which was previously only made for the left-to-right direction.

One interesting approach in supervised learning is transfer learning where a machine learning model that has been trained to solve certain task (e.g., face recognition) can be retrained and repurposed for another task (e.g., object detection). An example for this approach has been demonstrated by [Goodfellow et al. (2014)] on a generative adversarial network (GAN). The network consists of a generator and discriminator that behave in an adversarial way and can be trained in semi-supervised mode [Kingma et al. (2014)], i.e., a generalization from a small amount of labelled data for a large but unlabeled dataset.

In terms of unsupervised learning, there are works that have been performed by the computer science and machine learning community, although these works are very much overwhelmed by deep learning. One of the most popular unsupervised techniques that is very often used for data compression as well is clustering. Originally proposed by [MacQueen (1965)] by using k -means, i.e., data are partitioned into k -clusters under the assumption that each data point that is close to the nearest mean is assumed to belong to that cluster. The resulting cluster is called a Voronoi tessellation.

Beside clustering techniques, another unsupervised learning technique which is also commonly used for data compression is called principal component analysis (PCA). Originally proposed by [Pearson (1901)] in early 19th century, PCA can be done by either using singular value decomposition (SVD) or by performing eigenvalue decomposition on the covariance matrix. The method has been refined several times, e.g.: by [Ke and Kanade (2005)] who proposed alternative convex programming for robust L_1 -Norm factorization, [Johnstone and Lu (2009)] who discussed consistency and sparsity for high-dimensional PCA, and [Markopoulos et al. (2014)] who generalized multiple L_1 -maximum projection components.

While many of the works are proposing novel techniques and methodologies, [Locatello et al. (2019)] challenged some common assumptions in unsupervised learning regarding disentangled representations. Disentangled representations are narrowly defined variables of observation features which are encoded in a separate dimension, i.e., these are models that capture the very low-level features of a given observation in such a way that if one feature changes, the others remain unaffected. They first argued that *“unsupervised learning of disentangled representations is fundamentally impossible without inductive biases”*, trained

more than 12,000 models that covered many prominent methods and metrics, and concluded that there seems to be no such thing as fully disentangled methods without supervision.

Semi-supervised learning can be regarded as a generalized form of both supervised and unsupervised learning because one can either fully supervise the model during the training – or let the model find to learn the distribution without any prior label and in semi-supervised mode, both approaches are combined in a single training pipeline. One early empirical study on the semi-supervised approach involving variations of Bayes classifiers on multiple datasets and common benchmarks can be read in the work performed by [Guo et al. (2010)]. Furthermore, [Bengio et al. (2013), Reddy et al. (2018), Li and Liang (2019)] provided a literature review on recent semi-supervised techniques.

One common problem in semi-supervised approach is the duality between quasi-omnipresence of unlabeled data and sparse availability of labeled data which is tied to the expensive cost of labelling all possible datasets. [Triguero et al. (2015)] provided an extensive survey on self-labeling techniques covering the taxonomy of methodologies, multiple software packages, and various empirical studies. An example of such self-labeling approach is proposed by [Benato et al. (2018)]. In this example, they used projected feature space with a CNN-based encoder-decoder system to propagate the dataset labeling.

Briefly speaking, in deep learning, what people are interested in is in particular the feature representation of the input dataset. For this reason, the International Conference of Learning Representations (ICLR) has been invented as a research venue by the deep learning community since 2013 and consequently, many of the ICLR papers are mostly concerned with feature learning. For representation learning, the importance of mutual information (MI) maximization should be pointed out and this is reiterated by [Tschannen et al. (2020)], although they also argued that MI properties are highly dependent on used inductive bias, i.e., the choice of encoder-decoder architecture and its parametrization. [Ozair et al. (2019)] introduced the Wasserstein dependency measure instead of common KL-divergence in the mutual information. Similar to the Hellinger-Bhattacharyya distance, the Wasserstein distance measures the similarity between two probability distributions, which can be regarded as a generalization of the Euclidian and the Mahalanobis distance. They argued that the Wasserstein measure captures a more complete representations in the mutual information estimator. The state-of-the-art of MI maximization in word representation can be read in [Kong et al. (2020)].

It is also important to incorporate the inductive bias as well such that a proper architecture is selected. Inductive bias was reiterated again by [Shen et al. (2019)] on their work regarding ordered neurons in an LSTM network for different tasks such as language modeling and logical inference. [Kolesnikov et al. (2019)] argued that *“standard architecture design recipes do not necessarily translate from the fully supervised to the self-supervised setting and architecture choices which*

negligibly affect performance in the fully labeled setting, may significantly affect performance in the self-supervised setting”.

[Misra and Maaten (2019)] introduced Pretext Invariant-Representations in self-supervised mode, in such a way that the neural network learns representations from both the original image and its invariant transform and clustered the output in a memory bank about whether an image transform is similar or dissimilar. When training such a large dataset, sometimes either time or space becomes limited. [Goyal et al. (2019)] proposed to scale different dataset axis such as pre-training dataset size, model capacity, and the related problem complexity.

Not only for visual problems, but self-supervised learning can also (and probably should) be used for multi-modal problems. [Patrick et al. (2020)] introduced generalized data transformation (GDT) in the audio-visual representation learning problem to allow choice as to whether the transform is invariant or distinctive and to derive the conditions which the transformation combinations must obey. Their framework approach is depicted in Fig. 2.2.1-9.

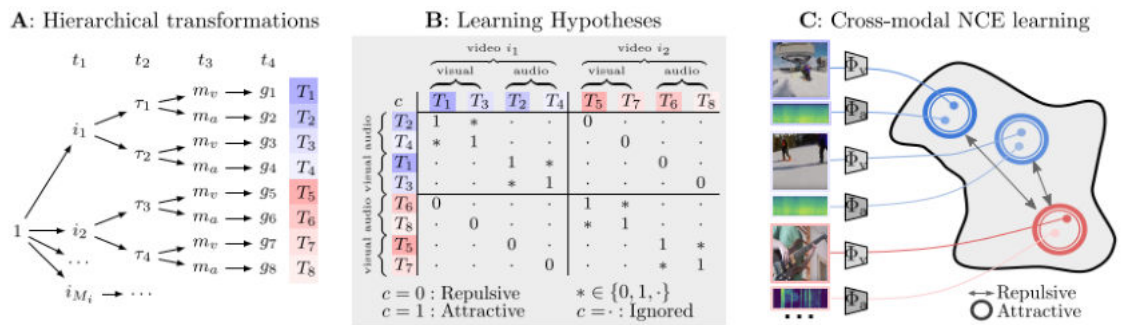


Fig. 2.2.1-9: Schematic overview of GDT. A: Hierarchical sampling of GDT for the audio-visual problem. B: The network learns the invariant and distinctive transformations. C: Result example showing which pairs are repelling and attracting. [Patrick et al. (2020)]

Finally, when talking about machine learning paradigms, one should not forget reinforcement learning (RL), where the central task is to influence agent behavior in an environment in order to maximize cumulative rewards. The formal introduction into reinforcement learning can be seen in [Busoniu et al. (2010), Mohri et al. (2018), François-Lavet et al. (2018)]. In short, in reinforcement learning an agent collects information through a sequence of actions by interacting with the environment.

One early application of reinforcement learning was proposed back in the early 90's by [Littman (1994), Tesauro (1995)] for playing games and these have been extended in recent years for many games such as chess or AlphaGo [Szita (2012), Lai (2015), Tang et al. (2017), Shao et al. (2018)] where the agent learns how to maximize the reward by winning the game. Further application of reinforcement learning can be seen in the domain of robotics and humanoids such as demonstrated by [Kormushev et al. (2013), Gu et al. (2017), Liu et al. (2018)], and for autonomous vehicle such as cars and unmanned aerial vehicles (UAV) [Ng (2003), Munoz et al. (2019), Becker-Ehmck et al. (2020), Ravi Kiran et al. (2020)].

An extensive survey on reinforcement learning on robotics can be read in [Kober and Peters (2014)].

2.2.2. Computational Intelligence for Optimization

The selection of technical and process parameter optimization for SHM such as number, location, and shape of the sensors, optimal excitation and sampling frequency, or even the geometry and constituent material of the structure, etc. can be regarded as a subset problem of a discrete mathematical optimization: it is a selection of the best element from a set of possible elements, i.e., finding a maximum (or minimum) and if possible, globally. The generalized problem in optimization can be formulated as the finding a non-convex function with non-linear programming technique for multi-objective optimization. From there, many special cases can be derived, e.g., for solving single objective linear problem with simplex algorithm.

The manifold of neural networks has been reviewed in very extensive way in the previous section - so it will not be further discussed here. Both evolutionary computing and swarm intelligence are methods that are very often used to solve optimization problems, and both can be regarded as metaheuristic methods. Generally, when solving an optimization problem, depending on the nature of the problem complexity, one can discretize these methods into one of these four approaches, including some of the common sub-techniques [Bianchi et al. (2009), Brinkhuis and Tikhomirov (2005), Cook et al. (1998)]:

- **Nonlinear programming:** function-based (such as line-search and interpolation), gradient-based (such as Trust region, Quasi-Newton, and conjugate gradient), and Hessian based methods.
- **Convex optimization:** linear programming (such as simplex and interior-point methods), quadratic programming - both are special case of non-linear programming where the functions to be solved are either linear or quadratic.
- **Combinatorial optimization:** Dynamic programming, Brute-force, Greedy method, Integer programming, State-space search, Graph algorithm
- **Metaheuristics:** Evolutionary algorithm (such as genetic algorithm), Simulated annealing, Swarm intelligence (such as ant colony, bee colony, and particle swarm optimization), Tabu search

Unfortunately, in the optimization community there are fewer resources and publications unlike the machine and deep learning domain. Nevertheless, one particular optimization technique, which is the gradient based method, serves as a fundamental backbone in many machine learning techniques especially for backpropagation in neural networks and gradient ascent for expectation maximization (EM) in other ML techniques, such as hidden Markov models (HMM) [Stamp (2017)] and mixture models [Marin et al. (2005)]. Most of modern deep learning optimization is done through gradient descent and its variant.

Some other optimization techniques beside gradient descent are the Levenberg-Marquardt and Broyden-Fletcher-Goldfarb-Shanno (BFGS) methods including its variant limited-memory BFGS (L-BFGS) [Nocedal and Wright (2006)]. Further, there is also Newton's method, but it requires a twice differentiable function since it needs to calculate the Hessian matrix \mathbf{H} , making it mostly impractical to optimize a large network.

Some well-known problems within combinatorial optimization are the vehicle routing problem (including the travelling salesman problem and the route inspection problem), the knapsack problem and the art gallery problem. One obvious method to solve a combinatorial problem is the exhaustive search (commonly known as brute force) which visits all possible options in the search space. Depending on the problem class complexity (i.e., **P** for polynomial time vs **NP** for non-polynomial time in a Turing machine), this option can be viable when the search space is extraordinarily small. In most real-world cases, brute force is computationally infeasible since the time complexity will be factorial - the time complexity is denoted as $\mathcal{O}(n!)$ - see Fig. 2.2.2-1, where n is the number of possible elements in the search space. The list of complexity of some common algorithms can be read in [BigOCheatSheet (online)].

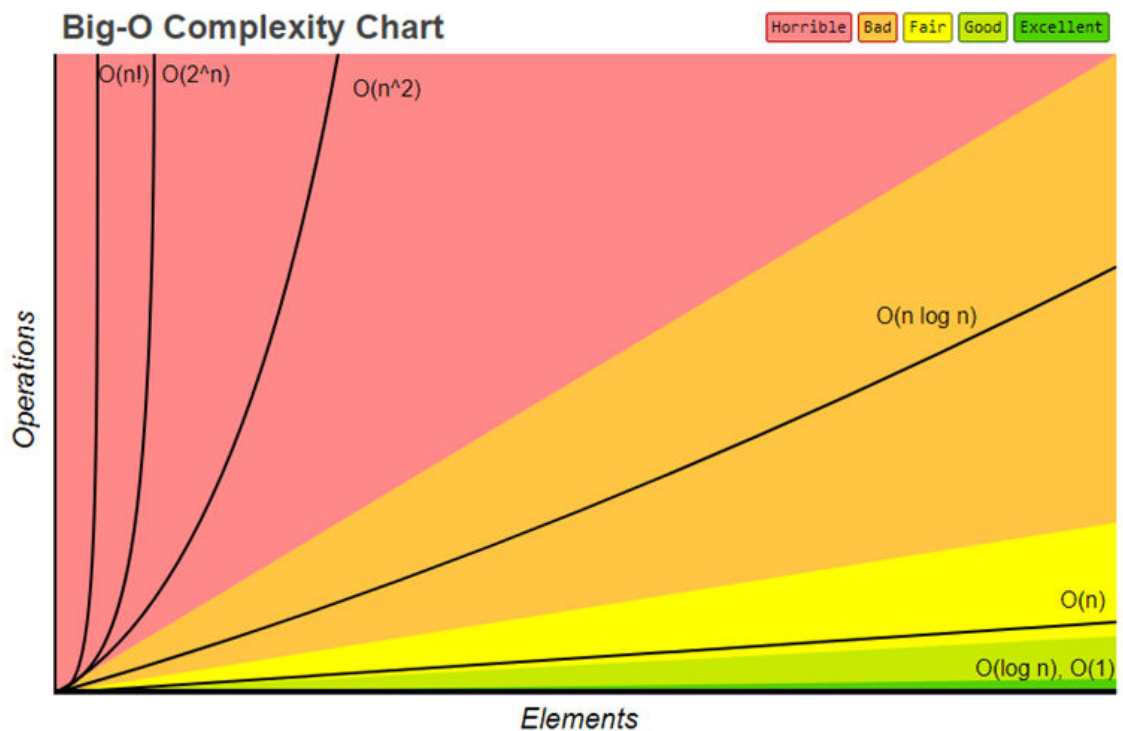


Fig. 2.2.2-1: Time complexity as a function of all possible elements in the search space. [BigOCheatSheet (online)]. Ideally, an algorithm can be considered feasible in many real-world problems if its computational complexity is less than $O(n^2)$, and possibly less than $O(n \log n)$.

Thus, we may rely on metaheuristic search algorithms such as the evolutionary algorithm to find a solution for many combinatorial problems. As summary, there are currently four mainstream approaches: 1). Evolutionary computing methods such as the genetic algorithm, neuroevolution and differential evolution, 2). Physically inspired methods such as simulated annealing and its variations, quantum annealing and stochastic tunneling, 3). Swarm intelligence-based

methods such as ant colony, bee colony, and particle swarm optimization, and 4). Socially inspired methods such as tabu search and its variants.

An application of adaptive differential evolution for dynamic optimization has been demonstrated by [Trojanowski et al. (2011)], where they modified the mutation operator to be based on an α -stable distribution. A similar approach on modified mutation has also been proposed by [Azad et al. (2011)] but they also implemented self-adaptive control parameters that influence the generated population at each run. This approach is related to what has been proposed by [Lourenço et al. (2013)]. In their work, a rule that defines how many individuals should be selected and how they should be chosen in order to adjust the selective pressure. The algorithm was tested on the knapsack problem.

[Forstenlechner et al. (2017)] proposed a similarity-based crossover technique by calculating the similarity measures such as Hamming and Levenshtein distance depending on the variable used. [Abbood and Vidal (2017)] introduced several different mutation operators in the co-evolution algorithm which they called the “Fly Algorithm”. The algorithm was tested for reconstructing a computed tomography (CT)-scan of cancer patient. A comparison of some of the adaptive selection techniques has been performed by [Jankee et al. (2015)].

On the application side, an evolutionary algorithm can be combined with a graph search algorithm such as A* for drone path planning for a surveillance mission in an urban environment as proposed by [Ghambari et al. (2019)]. The objective was to find the shortest path under the constraint that the drone must maintain a safe distance when flying between the obstacles.

A similar UAV path planning has also been proposed by [Arantes et al. (2016), Ellefsen et al. (2016)]. In similar way, [Ragusa et al. (2017)] also proposed an enhanced path planning for autonomous flight with a mini drone, the so-called micro aerial vehicle (MAV). They mentioned that common limitations of using GAs for path planning in a simulated environment are that the environment is discrete, and that the UAV motion is monotonic.

A genetic algorithm was also employed to optimize aircraft approach trajectories [Vormer et al. (2006)] and to solve automated taxi routing in an airport, such as proposed [Brownlee et al. (2018)]. They recognized that aircraft taxiing not only causes unnecessary fuel burn, but it also frustrates passengers and airport resources. For this reason, their algorithm was designed to minimize the taxiing time and they validated the approach using an example at 3 international major airport hubs.

Another interesting application of an evolutionary algorithm in an aerospace related domain beside path planning is to optimize the winglet design such as proposed by [Teixeira et al. (2016)], where several decision variables are fed into the algorithm to optimize two objectives: the ratio of drag-lift coefficients, and the wing root bending moment coefficient. This approach was similar to the later

work by [Gewehr and Sousa (2019), Zhang et al. (2020)], although the later implemented in the winglet configuration for a solar-powered aircraft.

In metaheuristics by swarm intelligence, one of the commonly used methods is the ant colony optimization algorithm (ACO) which is a probabilistic multi agent-based technique that is inspired by the behavior of real ants for solving computational problems with high complexity such as vehicle routing. [Melo et al. (2013)] divided the ant colony system (ACS) for solving the dynamic travelling salesman problem (TSP) – which is an NP-hard problem – into several cases that can have different selection strategies. An example of a real-world application of ACS optimization for logistics and transportation domain was proposed by [Luo et al. (2019)], where they aimed to minimize CO₂ emissions of transportation vehicles from home health care (HHC) companies. Not only for logistics, but ant colony optimization (ACO) has also been used for path optimization of multiple UAVs during Minimum Time Search (MTS) missions [Carabaza et al. (2017)]. While this can be used for military and defense-related topics, fortunately the example they gave was for a simulated search and rescue mission for a lost hiker.

Interestingly, ACO can also be used for optimizing recurrent neural networks such as LSTM and has been demonstrated by [El Said et al. (2018)]. In this work, they basically evolved the LSTM network by letting the artificial ants choose the best connections between the hidden layers. This technique can be viewed as a regularization for overfitting prevention that used metaheuristics search instead of a regular neuron dropout. The example results of their work for vibration prediction from a single test flight that compares optimized and non-optimized LSTM architecture is given in Fig.2.2.2-5.

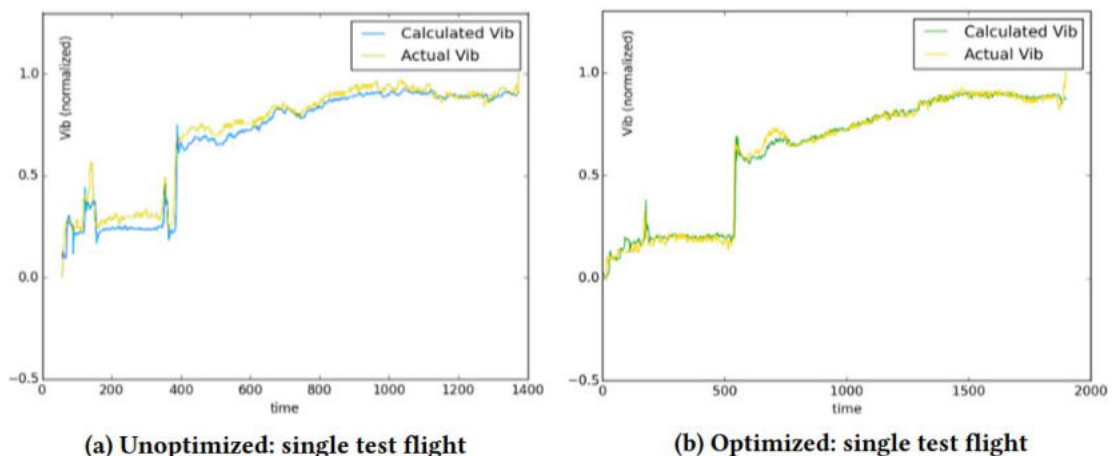


Fig. 2.2.2-1: Comparison between vibration prediction from unoptimized and ACO-optimized LSTM network. [El Said et al. (2018)].

Besides the ACO, artificial bee colony (ABC) algorithm, originally proposed by [Karaboga (2005)], is also commonly used for search metaheuristics. Rather than using a pheromone trail, in ABC the employed bee goes to a food source, evaluates the nectar amount and dances in the beehive. An onlooker bee watches the dance and goes to the food source and examines it. If the food source is abandoned, the employed bee (which then becomes a scout) will continue to search for a new

food source. The algorithm was slightly modified by [Consoli and Pavone (2013)] to solve the graph coloring problem and by [Wang et al. (2016)] for searching k-Nearest-Neighbor Fields (k-NNF).

As an alternative to the artificial ant and bee colony is the more generic particle swarm optimization (PSO), which was originally proposed by [Kennedy and Eberhart (1995), Shi and Eberhart (1998)], is also commonly used for many optimization problems. Many of the PSO variations do exist such as novelty driven PSO [Galvao et al. (2015)], where *“the particles are driven only towards instances significantly different from those found before”* to avoid local optima, i.e., they assigned the best position of particle to be the one that has the highest novelty instead of the best fitness.

PSO has also been used to optimize hyperparameter selection in a neural network, as demonstrated by [Lorenzo et al. (2017)]. This can be related to the work of [El Said et al. (2018)], although in this case they only used a standard CNN instead of LSTM for validation of their work. Not only for hyperparameter optimization in supervised learning, [Nguyen et al. (2018)] used PSO for feature selection in transfer learning with a domain adaptation approach. As for real-world PSO applications there are several interesting works that have been proposed such as: filter parameter selection for image denoising [Wang et al. (2017)] by employing a structural similarity index to calculate the intensity parameter. This approach was similar to a more recent work proposed by [Portelli and Pallez (2019)]. Recently, [Mohamed and Otero (2020)] used PSO for solving a multi-objective approach for market timing in a financial portfolio optimization such as value at risk (VaR), transactions count, and annualized rate of returns (AROR), where the solution set was presented as a Pareto set.

Another possible metaheuristic that is still used sometimes (although now outdated) is the tabu search. Originally proposed by [Glover (1986)], tabu search is a local neighborhood search, i.e., after taking a certain solution, it checks the value of its immediate neighbors. If it turns out that one of the neighbors is better than the current solution, then it will be taken as the best solution. This procedure is repeated until the termination condition is met. For this reason, tabu search is rarely employed as and apart from the contribution from [Sghir et al. (2013)] who used tabu search for the winner determination problem (WDP) and [Abdelkafi et al. (2017)] who used hybrid iterative tabu search for a quadratic assignment problem (QAP). Otherwise there seems to be not many works involving tabu search anymore.

The same is also valid for simulated annealing (SA) methods, which was originally proposed by [Pincus (1970)]. SA was inspired by the annealing process in metallurgy: after heating a metal at very high temperature, the slow cooling process ensures that the atoms have enough time to rearrange themselves according to the law of thermodynamics to form stable crystals because this results in a low-energy state - which is equivalent to the global optimum in metaheuristics. SA can be used to find the global optima of multivariable

functions. In comparison to the genetic algorithm however, SA normally performs worse as it can incorporate a candidate solution that is not improving, while genetic algorithm only accepts an evolving candidate.

There were not so many recent works involving SA apart from [Hung et al. (2008)], who demonstrated a multi-objective SA for PID controller design. [Mu et al. (2015)] proposed a memetic algorithm using combined SA and greedy optimization for community detection in networks. They argued that combining a global search with a locally concentrated SA will generate an algorithm with a better search ability. [Larsen et al. (2016), Larsen (2019)], used SA in combination with a pheromone-based perturbation strategy to identify important network substructures in the domain of biology, especially for protein structure comparison and studying human disease.

2.3. General Problem Statement & Objective

This section summarizes the general problem recognized from both the SHM & NDT community and computer science community. In the latter part of this section, the main research problem formulation will be described and then further broken down into several sub-problems.

2.3.1. Recognized Problems in SHM & NDT

For diagnostic applications, particularly in NDT, several applications of deep learning - which is largely based on CNN for crack visual detection - have been proposed. These works are generally focused on surface inspection of structures have been proposed by: [Zhang et al. (2016), Cha et. al (2017), Chaiyasarn et al. (2018), Fan et al. (2018), Panella et al. (2018), Pauly et al. (2017)] and many more. Besides for surficial crack detection at surface, there are several other works involving CNN in NDT, such as for optical phase boundary detection in shearography proposed by [Sawaf and Groves (2014)], welding detection using X-Ray images by [Hou et al. (2018)], and damaged steel and CFRP using infrared (IR) images by [Yousefi et al. (2018)].

In similar way, deep learning has also brought some wave of excitement to diagnostic SHM, although there are less works exploiting deep learning for diagnostic SHM in comparison to NDT. In recent years several works that incorporate deep learning in SHM has been proposed by of [Ebrahimkhanlou and Salomone (2019)] who used deep autoencoder (deep AE) for acoustic emission (AE) source localization, [Choy (2018), De Oliveira et al. (2018)] who used CNN for processing electromechanical impedance (EMI), and [Azimi and Pekcan (2019)], who used CNN for damage identification and localization of vibration sensor data in civil infrastructure.

However, as CNN is a discriminative model that is specifically tailored to learn how to solve certain task, once the model is trained, its parameters are fixed for solving that particular task only. Consequently, when the particular task is

slightly changed (e.g., recognizing a car in autonomous driving instead of recognizing a face in Facebook), a new deep learning model must be created. Hence, transfer learning might be a temporary solution, albeit it requires a pretrained model based on a large dataset. In SHM- NDT, such large datasets are not publicly available, and it is hardly feasible to perform transfer learning. It has been tried for recognizing crack image as has it been proposed, and in Lamb wave based SHM there was a trial done [Liu and Zhang (2019)] although there is a doubt on the efficacy of transferring image parameters for audio or acoustic wave signal processing. This is because the pre-trained model was trained specifically for recognizing images that has a physical origin of photon particles which is a fundamentally different physical phenomenon from an acoustic wave. While the network may finally learn the features from the time-frequency spectrogram, I also think that there would be no advantage of using pre-trained image recognition models in comparison to classical random start weights.

Another approach using online active learning has been proposed by [Bull et. al (2019)]. They recognized that the lack of descriptive labels made conventional supervised learning infeasible, and they proposed a novel adaptive learning process that updates the learning algorithm as soon as a new class of data is discovered by calculating the entropy. This approach bridges the gap of lack of labelled data temporarily, however it will fail at some point due to the incapability of the models to capture all possible cluster distributions since these tend to be infinite in nature.

Conclusively, while we shall appreciate the numerous works that propose deep learning approaches for SHM and NDT, a theoretical foundation that formalizes the utilization of deep learning for NDT and SHM is currently lacking. Some insight into understanding why deep learning might work for acoustic wave signal modelling for applications in structural diagnostic is needed. Further, the practicality of SHM still encounters many technical questions, and one of them is robust pattern recognition for signal classification for damage detection, which might not be robust in case of sub-optimal sensor topology, that is, when not enough wavefront is captured by one of the sensors within the sensor network. As previously described in chapter 2.1.2, currently there is only a limited amount of research in terms of sensor topology optimization. Much research has been placed on optimizing sensor positions with some good result, but to move forward it must be scalable on the SHM level.

2.3.2 Recognized Problems from Computer Science, Machine and Deep Learning Community

As for technological advancement from the AI and computer science community, we shall acknowledge that this advancement can clearly bring an improvement in the SHM and NDT community. One string that is attached to many of the works done in these communities is that many of the advanced techniques proposed are limited to publicly available benchmarks, such as:

- Image recognition and computer vision: handwriting recognition database (MNIST), MS-COCO, ImageNet, CIFAR-10 and CIFAR-100, Open Images and Street View House Numbers (SVHN).
- Text data: Amazon reviews, The Reuters Corpus, WordNet, Internet Movie Database (IMDB) Reviews, Sentiment140, and Twitter100k.
- Sound data: TIMIT, Common Voice, AudioSet, and Clotho.

They are rarely exposed to unusual, non-publicly available data such as Lamb wave signals and its representation since both SHM and NDT are quite niche areas in engineering. Thus, it is natural that they might not even be aware of what Lamb wave SHM is. For this reason, we can easily identify that it might be worthwhile to investigate further the treatability of Lamb wave signals with the advances brought by the computer science and AI community.

Before getting deeper into the investigation, one question that should be asked is what would be the starting point? Of course, it is a wish from many people not to have supervision bias, but without any incorporation of domain knowledge, meaningful research should (and probably can) not be conducted. Revisit the theorem proposed by [Locatello et al. (2019)] regarding unsupervised learning of disentangled representations:

Theorem 2.3.2-1. [Locatello et al. (2019)]

For $d > 1$, let $\mathbf{z} \sim P$ denote any distribution which admits a density $p(\mathbf{z}) = \prod_{i=1}^d p(z_i)$

Then, there exists an infinite family of bijective functions $f : \text{sup}(\mathbf{z}) \rightarrow \text{sup}(\mathbf{z})$

such that $\frac{\partial^2 f_i(u)}{\partial u_j^2} \neq 0$ almost everywhere $\forall_{i,j}$ (i.e. \mathbf{z} and $f(\mathbf{z})$ are completely entangled)

and $P(\mathbf{z} \leq u) = P(f(\mathbf{z}) \leq u) \forall u \in \text{sup}(\mathbf{z})$ (i.e. they have the same marginal distribution)

Proof: See Appendix A.

Corollary: Theorem 2.3.2-1 essentially shows that without inductive biases both on models and data sets (e.g., from prior knowledge), the task is fundamentally impossible, i.e., *biases are the part of solution when dealing a real-world problem with machine learning.*

That practically means, it is up to the SHM-NDT community to explore and exploit whether the advanced machine learning and computational intelligence techniques brought by the AI community can be employed to push the desired improvement in predictive maintenance.

2.3.3. Diagnostic Decision Logic and Isomorphism in Finite Automata

Rather than going directly with engineering details, we shall start with the formalism and mathematical logic. It consists of 6 separate different areas: **logic**, **model theory**, **computability**, **set theory**, **proof**, and **Gödel's Incompleteness theorem**. While not everything can be described in a detailed way, we can refer

to [Li (2010), Mendelson (1997), Shoenfield (2010), Boolos et al. (2002)]. The above-mentioned literatures introduce the **formal systems** and **quantifiers** and help understand **propositional calculus** and **first-order logic (FOL)**. The semantics of FOL is that interpretation is based on the context and its syntax consists of variables and quantification (e.g., ‘for all: \forall ’, ‘there exists: \exists ’). The four fundamental elements of formal systems are:

1. A finite set of symbols
2. A grammar rules
3. A set of axioms
4. A set of inferencing rules

In computational linguistics, Chomsky hierarchy represents the hierarchical class of languages that are accepted by the different abstract machines. As such, the formal languages represented in the hierarchy can be described as formal grammar. The abstract machines (often called automata) are computational models which follow a predetermined sequence of instructions. [Chomsky (1956)] classified the grammar, automaton, and language as shown in Table 2.4.1-2. Type-3 grammar is also contained by type-2 grammar, and type-2 grammar is contained by type-1 grammar, and type-1 grammar is contained by type-0 grammar. In the same way, every regular language is recursively enumerable, and every finite-state machine is a special case of Turing machine, and so on.

Grammar	Language	Automaton
Type-3 Grammar	Regular	Finite state machine
Type-2 Grammar	Context-Free	Pushdown automaton
Type-1 Grammar	Context-Sensitive	Linear-bounded automaton
Type 0-Grammar	Recursively Enumerable	Turing machine

Table 2.4.1-1: Chomsky hierarchy

Without going too deep into theoretical computer science, the formalism in aircraft maintenance logic goes this way: many parts of the Airbus A320 have been designed to be damage tolerant. Let us simplify the assumption by stating that the inspection technique has a detection limit of a crack longer than 5 cm, i.e., a crack ≥ 5 cm is said to be a damage. When a certain sub-structure is damaged, it needs to be repaired so that the aircraft is airworthy. In this situation, the binary input: {Damage, NOT-Damage} can be associated with two possible finite states: {Airworthy, Repair}. This formulation can be translated into the simplest model of automaton: the finite state machine (FSM). The formal definition of an FSM for the binary diagnostic SHM is a quintuplet $\{\Sigma, Q, q_0, \delta, F\}$, where:

Σ are input alphabet, finite \emptyset -set of symbols $\ni \{\text{Damage, NOT-Damage}\}$

Q is a finite \emptyset -set of states $\ni \{\text{Airworthy, Repair}\}$

q_0 is an initial state, $\epsilon Q \ni \{\text{Airworthy}\}$

δ is the state-transition function, $\delta: Q \times \Sigma \rightarrow F$

F is the set of final states, $\epsilon Q \ni \{\text{Airworthy}\}$

An example of FSM is deterministic finite automata (DFA), see Fig. 2.3.3-1a, while the state transition function for a given input is depicted in Fig. 2.3.3-2b. With this logic, whenever there is a damage, the next following state will be {Repair} and if it is still damaged, it will stay in {Repair}, otherwise it will return to the state {Airworthy}.

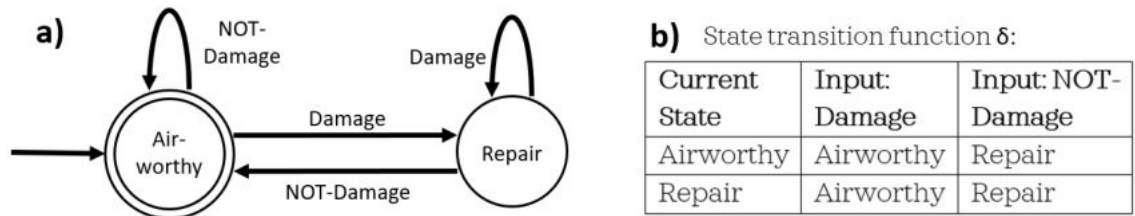


Fig. 2.3.3-1: a). Deterministic finite automaton (DFA) for diagnostic SHM binary logic, b). state transition function δ for the input alphabet {Damage, NOT-Damage}. The final state is represented as double circle, which in this case is {Airworthy}.

A finite sequence of the input alphabet is called a string and an n -set of strings is called a language $L \supseteq L_n$. For brevity, {NOT-Damage, Damage} will be abbreviated as as {0, 1}, respectively. As such, an example of strings that can be accepted by the binary diagnostics DFA are:

- The aircraft was never used so that it has no accumulated damage over time such that $L_1 = \{0, 00, 000, 0000, 00000, \dots\}$
- The aircraft was flown, and it has accumulated damage, but directly repaired afterwards such that $L_2 = \{010, 10, 0110, 01110, 10010, \dots\}$

On the contrary, an example of strings that will be rejected the DFA are:

- The aircraft was flown, and it has accumulated damage, but never directly repaired afterwards such that $L_3 = \{01, 001, 011, 0011, 000011, \dots\}$
- The aircraft has already been damaged just after it was produced such that $L_4 = \{1, 11, 111, 1111, 11111, \dots\}$
- Further, we can define a language $L_5 \subseteq L_4$ for a situation where the aircraft has been damaged after the production, repaired directly afterwards, but damaged during the service and not repaired such that $L_5 = \{1101, 1001, 1101, 100011, \dots\}$

As previously said, the above situation is simplified for a binary diagnostic only. In real-world situation, when we are not satisfied with the binary diagnostics as many would like to regress during the aircraft lifecycle, we can slightly extend the FSM into three states and a 3-input alphabet as described in Table 2.3.3-1:

Σ : {NOT-Damage, Damage-Threshold 1, Damage-Threshold 2}
Q : {Airworthy, Damage Growth, Repair}
q_0 : {Airworthy}
F : {Airworthy, Damage Growth}

Table 2.3.3-1: Three states diagnostics FSM with 3 input alphabets

For brevity, the set {NOT-Damage, Damage-Threshold 1, Damage-Threshold 2} is abbreviated as {0,1,2}. In the situation above, the state {Airworthy} is associated only for {0}, and when the input alphabet is {1}, the state would be {Damage Growth}. When the input state is {2}, the state would be {Repair}. The initial state would be {Airworthy} and the final states can be {Airworthy} or {Damage Growth}. Depending on how the airworthiness is regulated by the governmental agency, there are multiple paths possible during damage growth. The applicable logic here is to assign a multiple possibility when the state is {Damage Growth}. Such FSM is called non-deterministic finite automaton (NFA). The NFA diagram and its state transition function is given in Fig. 2.3.3-2.

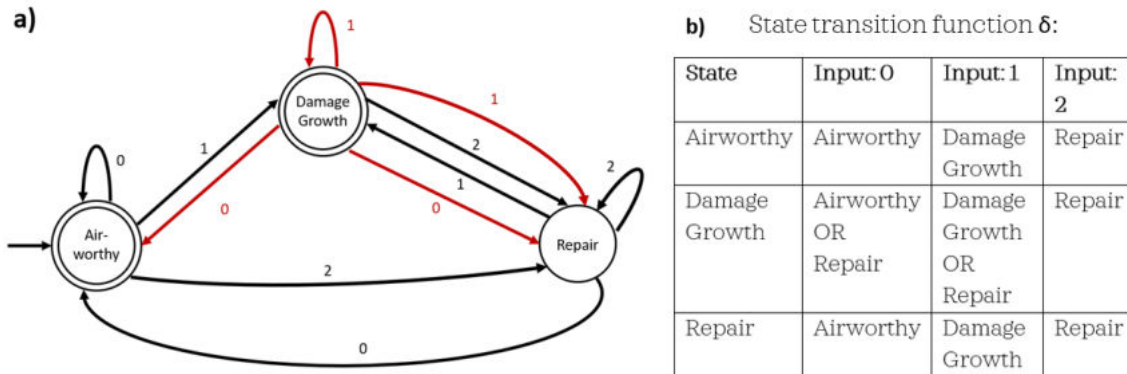


Fig. 2.3.3-2: a). Non-deterministic finite automaton (NFA) for non-binary diagnostic logic, b). state transition function δ for the input alphabet {0, 1, 2}.

There are several situations that can be associated with the non-determinism during damage growth state:

- After the state {Damage Growth} has been reached, suddenly the SHM measurement gives {0} (i.e., no damage at all) when in fact this is probably illogical since the damage state does not return to the baseline without repair, which indicates high probability of a measurement error. In this case, two possible states {Airworthy, Repair} will follow - indicated by the lower red arrows in Fig. 2.4.1-2a. An example of such a string would be either {000110} or {001102}.
- After the {Damage Growth} state has been reached, an aircraft operator can choose to let the damage grow until it reaches the second threshold where it has to be repaired or the operator chooses to repair it directly. This case is indicated by the upper red arrows in Fig. 2.4.1-2a. An example of such a string would be either {011111} or {000012}.

We can see that for the single NFA described above, there exists 4 equivalent DFA depending on which string is accepted - i.e., on the logic path the aircraft operator would like to choose. Further, the diagnostic NFA can still be expanded into 5 different states and 3 damage thresholds as depicted in Fig. 2.3.3-4 and by its formal definition in Table 2.3.3-1. In this case, it is interesting to think what state transition function δ can be applied to this logic and what kind of strings can be accepted / rejected by such automaton.

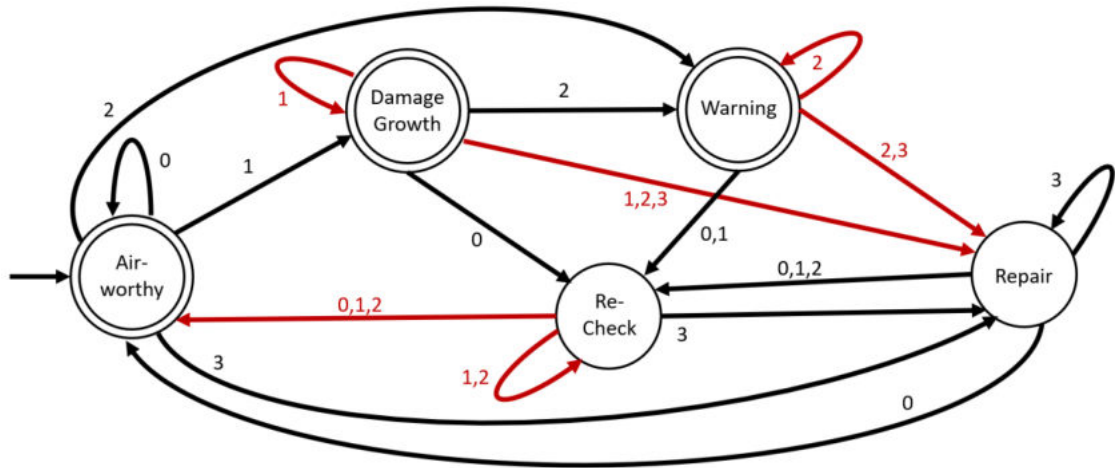


Fig. 2.3.3-4: NFA for 5-states diagnostic logic with 4 input alphabets. Red arrows indicate the multiple next possible states.

Σ : {NOT-Damage, Threshold 1, Threshold 2, Threshold 3}
Q : {Airworthy, Damage Growth, Warning, Re-Check, Repair}
q_0 : {Airworthy}
F : {Airworthy, Damage Growth, Warning}

Table 2.4.1-4: Five states diagnostics FSM with 4 input alphabets

2.3.4. Research Problems Formulation

Narrowing down this statement for NDT and SHM domain, it would mean that instead of focusing on automated decision making, AI should be used as decision support to accelerate decision making in the MRO industry. A concrete example of this can be demonstrated by automated damage detection to help to decide whether an aircraft should be repaired or not. Acknowledging the state of the art in NDT & SHM and the recent advances from the CS and ML community, the high-level question that should be asked from the NDT & SHM community is: *is it feasible to incorporate computational and artificial intelligence as a design tool for an automated diagnostic as a decision support for predictive maintenance – and if so, in what way?* Surely there are different ways to do so, and I hypothesize that it is certainly feasible although it might not always be the easy way.

To answer this question, we shall break down the problem into several manageable sub-problems, and the proposition are:

1. The design complexity and parameter optimization, particularly on sensor placement methodologies for both deterministic and semi-stochastic approaches according to what extent the structure is designed based on the premise that sensor network topology affects the damage detection capability and the overall SHM performance. In this proposal, the investigation can be rephrased as *which different sensor network topologies are needed to understand the trade-off between the strategies and if possible, to propose a compensation technique?*

2. Deep learning for SHM, i.e., an investigation as to whether deep learning can be used to treat the Lamb wave signal – and if so, whether it has certain theoretical justification. *What would be the pros and cons when using deep learning to treat Lamb wave signals and what would be the consequences for design and manufacturing of SHM system?* Further considerations on certain aspects from computational neuroscience for processing the Lamb wave signal could also be considered.
3. Eventually and worthy to be considered as a research direction as well: when combining the sub-problems to reconstruct the final solution: *Given a certain sensor topology, what would the training behavior look like from for different sensors and how do different signal representation will affect the training behavior?*

In relation to the research problem formulation described in Section 2.3.3, we shall start with a simplified solution: problem discretization in binary mode, i.e., whether a damage is being detected or not. After that, we can further refine our precision with regressive discretization, i.e., taking different possible damage states into consideration. The following chapters 3 - 7 will follow one of these case studies streams.

Literatures

Abbood ZA, Vidal FP. *Basic, Dual, Adaptive, and Directed Mutation Operators in the Fly Algorithm*. Proc. 13th Intl Conf on Artificial Evolution, Paris (2017).

Abdelkafi O, Idoumghar L, Lepagnot J. *Improved Hybrid Iterative Tabu Search for QAP Using Distance Cooperation*. Proc. 13th Intl Conf on Artificial Evolution, Paris (2017).

Ahmad R. *Wavelet Based Characterization of Acoustic Attenuation in Polymers Using Lamb Wave Modes*. Proc. 7th European Workshop on Structural Health Monitoring (EWSHM), Nantes (2014).

Ambrozinski L, Packo P, Stepinski T, Uhl T. *Ultrasonic Guided Waves Based Method for SHM Simulations and an Experimental Test*. Proc. 5th World Conference on Structural Control and Monitoring (WCSCM), Tokyo (2010).

Ambrozinski L, Stepinski T, Uhl T, Ochonski J, Klepka A. *Development of Lamb Waves Based SHM Systems*. J Key Engineering Materials. Vol. 518: 87-94 (2012).

Andrews JP, Palazotto AN, DeSimio MP, Olson SE. *Lamb Wave Propagation in Varying Isothermal Environments*. Intl J of Structural Health Monitoring. Vol. 7(3): 265-270 (2008).

Arantes MDS, Arantes JDS, Toledo CFM, Williams BC. *A Hybrid Multi-Population Genetic Algorithm for UAV Path Planning*. Proc. 2016 Genetic and Evolutionary Computation Conf (GECCO), Denver (2016).

Azad MAK, Fernanded MGP. *Modified Constrained Differential Evolution for Solving Nonlinear Global Optimization Problems*. Proc. Intl Joint Conf on Computational Intelligence, Paris (2011).

Azimi M, Pekcan G. *Structural Health Monitoring using Extremely Compressed Data through Deep Learning*. J Computer Aided Civil and Infrastructure Engineering. Vol. 12517: 1-18 (2019).

Balamonica K, Saravanan J, Priya B, Gopalakrishnan N. *Piezoelectric Sensor-Based Damage Progression in Concrete Through Serial/Parallel Multi-Sensing Technique*. Intl J Structural Health Monitoring. Vol. 19(2): 339-356.

Bahdanau D, Cho KH, Bengio Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. Proc. Intl Conf on Learning Representations (ICLR), San Diego (2015).

Becker-Ehmck P, Karl M, Peters J, van der Smagt P. *Learning to Fly via Deep Model-Based Reinforcement Learning* (2020). Available: <https://arxiv.org/abs/2003.08876> (Last online: JUL-2020)

- Benato BC, Telea AC, Falçao AX. *Semi-Supervised Learning with Interactive Label Propagation Guided by Feature Space Projections*. Proc. 31st Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), Paraná (2018).
- Bengio Y, Courville A, Vincent P. *Representation Learning: A Review and New Perspectives*. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 35(8): 1-31 (2013).
- Big O Cheat Sheet. Available: <https://www.bigocheatsheet.com/> (Last online: JUL-2020).
- Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ. *A Survey on Metaheuristics for Stochastic Combinatorial Optimization*. J Natural Computing. Vol. 8(2): 239-287 (2009).
- Boller C. *Structural Health Monitoring – An Introduction and Definitions*. In: Encyclopedia of Structural Health Monitoring, John Wiley & Sons Ltd (2009).
- Boolos G, Burgess J, Richard J. *Computability and Logic* (4th Ed.). Cambridge University Press, Cambridge / New York / Melbourne / Madrid / Cape Town (2002).
- Brinkhuis J, Tikhomirov V. *Optimization: Insights and Applications*. Princeton University Press, Princeton (2005).
- Brownlee AEI, Weiszer M, Woodward JR, Chen J. *A Rolling Window with Genetic Algorithm Approach to Sorting Aircraft for Automated Taxi Routing*. Proc. 2018 Genetic and Evolutionary Computation Conf (GECCO), Kyoto (2018).
- Bull A, Rogers TJ, Wickramarchchi C, Cross EJ, Worden K, Dervilis N. *Probabilistic Active Learning: An Online Framework for Structural Health Monitoring*. J Mechanical System and Signal Processing. Vol. 134: 106294 (2019).
- Busoniu L, Babuska R, De Schutter B, Ernst D. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press Automation and Control Engineering Series, Boca Raton / London / New York (2010).
- Carabaza SP, Ortega JB, Portas EB, Orozco JA, Cruz JM. *A Multi-UAV Minimum Time Search Planner based on ACO_R*. Proc. 2017 Genetic and Evolutionary Computation Conf (GECCO), Berlin (2017).
- Carboni M, Gianneo Andrea, Giglio M. *A Lamb Waves Based Statistical Approach to Structural Health Monitoring of Carbon Fibre Reinforced Polymer Composites*. J Ultrasonics. Vol. 60: 51-64 (2015).
- Cesnik C, Raghavan A. *Fundamentals of Guided Elastic Waves in Solids*. In: Encyclopedia of Structural Health Monitoring, John Wiley & Sons Ltd (2009).
- Cha YJ, Choi W, Büyüköztürk O. *Deep Learning Based Crack Damage Detection Using Convolutional Neural Networks*. J Computer-Aided Civil and Infrastructure Engineering, Vol. 32(5): 361-378 (2017).
- Chaiyasarn K, Sharma M, Ali L, Khan W, Poovarodom N. *Crack Detection in Historical Structures Based on Convolutional Neural Networks*. Intl J of Geomate, Vol. 15(51): 240-251 (2018).
- Chen H, Gao G, Hu N, Deng M, Xiang Y. *Modeling and Simulation of Frequency Mixing Response of Two Counter-Propagating Lamb Waves in a Two-Layered Plate*. J Ultrasonics. Vol. 104: 106109 (2020).
- Chen L, Dong Y, Meng Q, Liang W. *FEM Simulation for Lamb Wave Evaluate the Defects of Plates*. IEEE Intl Workshop on Microwave and Millimeter Wave Circuits and System Technology, Chengdu (2012).
- Chen X, Wang CL. *Noise Removing for Lamb Wave Signals by Fractional Differential*. J Vibroengineering. Vol. 16(6): 2676-2684 (2014).
- Cho KH, Bahdanau D, Bougares F, Schwenk H, Bengio Y. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. Proc. Conf on Empirical Methods in Natural Language Processing (EMNLP), Doha (2014).
- Chomsky N. *Three Models for the Description of Language*. IRE Transactions on Information Theory. Vol.2: 113-124 (1956).
- Choy AW. *Structural Health Monitoring with Deep Learning*. Proc. 2018 IAENG Intl Conf on Control and Automation, Hong Kong (2018).
- Consoli P, Pavone M. *O-BEE-COL: Optimal BEEs for COLoring Graphs*. Proc. 11th Intl Conf on Artificial Evolution, Bordeaux (2013).
- Cook WJ, Cunningham WH, Pulleyblank WR, Schrijver A. *Combinatorial Optimization*. Wiley, New York (1998).
- Cortes C, Vapnik V. *Support-Vector Networks*. J Machine Learning. Vol. 20: 273-297 (1995).
- Croxford AJ, Wilcox PD, Drinkwater BW, Konstantinidis G. *Strategies for Guided-Wave Structural Health Monitoring*. Proc. Royal Society A. Vol. 463(2087): 2961-2981 (2007).

- De Luca A, Perfetto D, DeFenza A, Petrone G, Caputo F. *Sensitivity Analysis on the Damage Detection Capability of a Lamb Waves Based SHM System for a Composite Winglet*. Procedia Structural Integrity. Vol. 12: 578-588 (2018).
- De Luca A, Sharif-Khodaei Z, Aliabadi MH, Caputo F. *Numerical Simulation of the Lamb Wave Propagation in Impacted CFRP Laminate*. Procedia Engineering. Vol. 167: 109-115 (2016).
- De Marchi L, Perelli A, Marzani A. *A Signal Processing Approach to Exploit Chirp Excitation in Lamb Wave Defect Detection and Localization Procedures*. J Mechanical System and Signal Processing. Vol. 39 (1-2): 20-31 (2013).
- De Oliveira MA, Monteiro MA, Vieira Filho J. *A New Structural Health Monitoring Strategy Based on PZT Sensors and Convolutional Neural Network*. J Sensors. Vol. 18: 1-21 (2018).
- Devlin J, Chang MW, Lee K, Toutanova K. BERT: *Pre-training of Deep Bidirectional Transformers for Language Understanding*. Proc. 2019 Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), Minneapolis (2019).
- Ding X, Zhao Y, Hu N, Liu Y, Zhang J, Deng M. *Experimental and Numerical Study of Nonlinear Lamb Waves of a Low-Frequency SO Mode in Plates with Quadratic Nonlinearity*. J MDPI Materials. Vol. 11(11): 2096.
- Dodson JC, Inman DJ. *Thermal Sensitivity of Lamb Waves for Structural Health Monitoring Applications*. J Ultrasonics. Vol. 53(3): 677-685 (2013).
- Domingos P, Pazzani M. *On the Optimality of the Simple Bayesian Classifier under Zero-One Loss*. J Machine Learning. Vol. 29: 103-130 (1997).
- Donahue J, Hendricks LA, Rohrbach M, Venugopalan S, Guadarrama S, Saenko K, Darrell T. *Long-term Recurrent Convolutional Networks for Visual Recognition and Description*. Proc. 15th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston (2015).
- Duan W, Niu X, Gan TH, Kanfoud J, Chen HP. *A Numerical Study on the Excitation of Guided Waves in Rectangular Plates Using Multiple Point Sources*. MDPI J Metals. Vol. 7: 552 (2017)
- Ebrahimkhanlou A, Dubuc B, Salomone S. *A Generalizable Deep Learning Framework for Localizing and Characterizing Acoustic Emission Sources in Riveted Metallic Panels*. J Mechanical System and Signal Processing. Vol. 130: 248-272 (2019).
- Ellefsen KO, Lepikson HA, Albiez JC. *Planning Inspection Paths through Evolutionary Multi-objective Optimization*. Proc. 2016 Genetic and Evolutionary Computation Conf (GECCO), Denver (2016).
- El Said AR, El Jamiy F, Higgins J, Wild B, Desell T. *Using Ant Colony Optimization to Optimize Long Short-Term Memory Recurrent Neural Networks*. Proc. 2018 Genetic and Evolutionary Computation Conf (GECCO), Kyoto (2018).
- Ercolano G, Rossi S. *Combining CNN and LSTM for Activity of Daily Living Recognition with a 3D Matrix Skeleton Representation*. J Intelligent Service Robotics. Vol. 14: 175-185 (2021).
- Ester M, Kriegel HP, Sander J, Xu X. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Proc. 2nd Intl Conf on Knowledge Discovery and Data Mining (KDD-96), Portland (1996).
- Ewald V, Ochôa P, Groves RM, Benedictus R. *Design of a Structural Health Monitoring System for a Damage Tolerance Fuselage Component*. Proc. 7th Intl Symposium on NDT in Aerospace, Bremen (2015).
- Fan Z, Wu Y, Lu J, Li W. *Automatic Pavement Crack Detection Based on Structured Prediction with the Convolutional Neural Network* (2018). Available online <https://arxiv.org/abs/1802.02208> (Last online: FEB-2020)
- Fendzi C, Morel J, Rébillat M, Guskov M, Mechbal N, Coffignal G. *Optimal Sensor Placement to Enhance Damage Detection in Composite Plates*. Proc. 7th European Workshop on Structural Health Monitoring (EWSHM), Nantes (2014).
- Forstenlechner S, Fagan D, Nicolau M, O'Neill M. *Semantics-Based Crossover for Program Synthesis in Genetic Programming*. Proc. 13th Intl Conf on Artificial Evolution, Paris (2017).
- François-Lavet V, Henderson P, Islam R, Bellemare MG, Pineau J. *An Introduction to Deep Reinforcement Learning*. In: Foundations and Trends in Machine Learning. Vol. 11(3-4): 1-140 (2018).
- Fuller R. *Introduction to Neuro-Fuzzy Systems*. Springer, Berlin / Heidelberg (2000).
- Galvao DF, Lehman J, Urbano P. *Novelty-Driven Particle Swarm Optimization*. Proc. 12th Intl Conf on Artificial Evolution, Lyon (2015).
- Gao D, Wu Z, Yang L, Zheng Y. *Integrated Impedance and Lamb Wave-based SHM Strategy for Long Term Cycle Loaded Composite Structure*. J Structural Health Monitoring (SHM). Vol. 17(4): 763-776 (2018).

- Gao F, Hua J, Zeng L, Lin J. *Amplitude Modified Sparse Imaging for Damage Detection in Quasi-isotropic Composite Laminates using Non-contact Laser Induced Lamb Waves*. J Ultrasonics. Vol. 93: 122-129 (2019).
- Gao G, Liu C, Hu N, Deng M, Chen H, Xiang Y. *Response of Second-Harmonic Generation of Lamb Wave Propagation to Microdamage Thickness in a Solid Plate*. J Wave Motion. Vol. 96: 102557 (2020).
- Gewehr LAC, Sousa BS. *Winglet Design Optimization Using A Multi-Objective Genetic Algorithm*. Proc. 25th ABCM Intl Congress on Mechanical Engineering, Uberlândia (2019).
- Ghambari S, Idoumghar L, Jourdan L, Lepagnot J. *Hybrid Evolutionary Algorithm for Offline UAV Path Planning*. Proc. 14th Intl Conf on Artificial Evolution, Mulhouse (2019).
- Gheisari S, Shariflou S, Phu J, Kennedy PJ, Agar A, Kalloniatis M, Golzan SM. *A Combined Convolutional and Recurrent Neural Network for Enhanced Glaucoma Detection*. Scientific Reports. Vol. 11: 1945 (2021).
- Giurgiutiu V, Yu L, Santoni GB, Xu B. *Lamb Wave Tuning for Piezoelectric Wafer Active Sensor Applications in In-Situ Structural Health Monitoring*. Proc. Annual Review of Progress in Quantitative Nondestructive Evaluation, Golden (2007).
- Glover F. *Future Paths for Integer Programming and Links to Artificial Intelligence*. J Computers and Operations Research. Vol. 13(5): 533-549 (1986).
- Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. *Generative Adversarial Nets*. Proc. 27th Intl Conf on Neural Information Processing Systems (NIPS), Montréal (2014).
- Gorgin R, Luo Y, Wu Z. *Environmental and Operational Conditions effects on Lamb Wave Based Structural Health Monitoring Systems: A Review*. J Ultrasonics. Vol. 105: 106114 (2020).
- Goyal P, Mahajan D, Gupta A, Misra I. *Scaling and Benchmarking Self-Supervised Visual Representation Learning*. Proc. IEEE Intl Conf on Computer Vision (ICCV), Seoul (2019).
- Gravenkamp H. *Numerical Methods for the Simulation of Ultrasonic Guided Waves*. Dissertation. Technical University Carolo-Wilhelmina Brunswick, Braunschweig (2014).
- Gu H, Wang ML. *A Monolithic Interdigitated PVDF Transducer for Lamb Wave Inspection*. Intl J of Structural Health Monitoring. Vol. 8(2): 137-148 (2009).
- Gu S, Holly E, Lillicrap T and Levine S. *Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-policy Updates*. Proc. 2017 IEEE Intl Conf on Robotics and Automation (ICRA), Singapore (2017).
- Guo Y, Niu X, Zhang H. *An Extensive Empirical Study on Semi-supervised Learning*. Proc. 10th IEEE Intl Conf on Data Mining (ICDM), Sydney (2010).
- Gupta D, Ramjee R, Kwatra N, Sivathanu M. *Unsupervised Clustering using Pseudo-semi-supervised Learning*. Proc. Intl Conf on Learning Representations (ICLR), Virtual Conf (2020).
- Harley JB, Liu C, Oppenheim IJ, Moura JMF. *Managing Complexity, Uncertainty, and Variability in Guided Wave Structural Health Monitoring*. SICE J of Control, Measurement, and System Integration. Vol. 10(5): 325-336 (2017).
- Haynes C. *Effective Health Monitoring Strategies for Complex Structures*. Dissertation. University of California San Diego (UCSD), San Diego (2014).
- He K, Zhang X, Ren S, Sun J. *Deep Residual Learning for Image Recognition*. Proc. IEEE Conf on Computer Vision and Pattern Recognition (CVPR), Las Vegas (2016).
- Hinton GE, Salakhutdinov RR. *Reducing the Dimensionality of Data with Neural Networks*. Science Magazine. Vol. 313: 504-507 (2006).
- Hinton GE, Osindero S, Teh YW. *A Fast-Learning Algorithm for Deep Belief Nets*. J Neural Computation. Vol. 18(7): 1527-54 (2006).
- Hinton GE, Sabour S, Frosst N. *Matrix Capsules with EM Routing*. Proc. Intl Conf on Learning Representations (ICLR), Vancouver (2018).
- Hochreiter S, Schmidhuber J. *Long Short-Term Memory*. J Neural Computation 9(8): 1735-1780 (1997).
- Hopfield JJ. *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*. Proc. Natl Academy of Science USA, Vol. 79(8): 2554-2558 (1982).
- Hosseini SMH, Duczek S, Gabbert U. *Damage Localization in Plates Using Mode Conversion Characteristics of Ultrasonic Guided Waves*. J Non-Destructive Evaluation (JNDE). Vol. 33: 152-165 (2014).
- Hou W, Wei Y, Guo J, Jin Y, Zhu C. *Automatic Detection of Welding Defects using Deep Neural Network*. J Physics. Vol. 933: 012006 (2018).

- Howard AG, Zhu ML, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications* (2017). Available: <https://arxiv.org/abs/1704.04861> (Last online: SEP-2020)
- Hu S, Zhou L. *Modeling Lamb Wave Propagation in Damaged Structures Based upon Spectral Element Method*. J Advanced Materials Research. Vol. 570: 79-86 (2012).
- Hua J, Lin J, Zeng L, Gao F. *Pulse Energy Evolution for High-resolution Lamb Wave Inspection*. J Smart Materials and Structures. Vol. 24(6): 065016 (2015).
- Hung MH, Shu LS, Ho SJ, Hwang SF, Ho SY. *A Novel Intelligent Multiobjective Simulated Annealing Algorithm for Designing Robust PID Controllers*. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans. Vol. 38(2): 319-330 (2008).
- Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K. *SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and <0.5MB Model Size* (2017). Available: <https://arxiv.org/abs/1602.07360> (Last online: SEP-2020).
- Islam MZ, Islam MM, Asraf A. *A Combined Deep CNN-LSTM Network for the Detection of Novel Coronavirus (COVID-19) using X-Ray Images*. Informatics in Medicine Unlocked. Vol. 20: 100412 (2020).
- Ismail N, Hafizi ZM, Nizwan CKE, Ali S. *Simulation of Lamb Wave Interactions with Defects in a Thin Plate*. IOP J of Physics Conf. Series. Vol. 1262: 012030 (2019).
- Jankee C, Verel S, Derbel B, Fonlupt C. *Distributed Adaptive Metaheuristic Selection: Comparisons of Selection Strategies*. Proc. 12th Intl Conf on Artificial Evolution, Lyon (2015).
- Johnstone IM, Lu AY. *On Consistency and Sparsity for Principal Components Analysis in High Dimensions*. J of the American Statistical Association. Vol. 104(486): 682-693 (2009).
- Karaboga D. *An Idea Based on Honeybee Swarm for Numerical Optimization* (2005). Available: <https://pdfs.semanticscholar.org/015d/f4d97ed1f541752842c49d12e429a785460b.pdf> (Last online: JUL-2020)
- Karimpanal TG, Bouffanais R. *Self-Organizing Maps for Storage and Transfer of Knowledge in Reinforcement Learning*. J Adaptive Behavior. Vol. 27(2): 111-126 (2018).
- Karpenko O. *Signal Analysis in Guided Wave Structural Health Monitoring*. Master Thesis. Michigan State University, Lansing (2013).
- Ke Q, Kanade T. *Robust L_1 -Norm Factorization in the Presence of Outliers and Missing Data by Alternative Convex Programming*. Proc. IEEE Conf on Computer Vision and Pattern Recognition (CVPR), San Diego (2005).
- Kennedy J, Eberhart R. *Particle Swarm Optimization*. Proc. IEEE Intl Conf on Neural Networks IV (1995).
- Kerber F, Sprenger H, Niethammer M, Luangvilai K, Jacobs LJ. *Attenuation Analysis of Lamb Waves Using the Chirplet Transform*. EURASIP J on Advances in Signal Processing. Vol. 2010: 375171 (2010).
- Khorram A, Khalooei M, Rezaghi M. *End-to-End CNN + LSTM Deep Learning Approach for Bearing Fault Diagnosis*. J Applied Intelligence. Vol. 51(1): 1-16 (2019).
- Kingma DP, Rezende DJ, Mohamed S, Welling M. *Semi-supervised Learning with Deep Generative Models*. Proc. 27th Intl Conf on Neural Information Processing Systems (NIPS), Montréal (2014).
- Kober J, Peters J. *Reinforcement Learning in Robotics: A Survey*. In: Learning Motor Skills. Springer Tracts in Advanced Robotics. Springer, Cham (2014).
- Kolesnikov A, Zhai X, Beyer L. *Revisiting Self-Supervised Visual Representation Learning*. Proc. IEEE Conf on Computer Vision and Pattern Recognition (CVPR), Long Beach (2019).
- Kong L, d'Autumne CM, Ling W, Yu L, Dai Z, Yogatama D. *A Mutual Information Maximization Perspective of Language Representations Learning*. Proc. Intl Conf on Learning Representations (ICLR), Virtual Conf (2020).
- Kordbacheh M, Yousefi-Koma A, Saleh MS, Soorgee MH. *Application of Wavelet Transform as a Signal Processing Method for Defect Detection using Lamb Waves: Experimental Verification*. Iranian J of Mechanical Engineering. Vol. 13(2): 82-95 (2012).
- Kormushev P, Calinon S, Caldwell DG. *Reinforcement Learning in Robotics: Applications and Real-World Challenges*. J MDPI Robotics, Vol. 2(3): 122-148 (2013).
- Krizhevsky A, Sutskever I, Hinton GE. *ImageNet Classification with Deep Convolutional Neural Networks*. Proc. Advances in Neural Information Processing Systems 25 (NIPS), Lake Tahoe (2012).
- Kudela P, Radzienski M, Ostachowicz W, Yang Z. *Structural Health Monitoring System Based on a Concept of Lamb Wave Focusing by the Piezoelectric Array*. J Mechanical System and Signal Processing. Vol. 108: 21-32 (2018).

- Kural A, Pullin R, Holford K, Lees J, Naylon J, Paget C, Featherston C. *Design and Characterization of an Ultrasonic Lamb-Wave Power Delivery System*. IEEE Trans on Ultrasonics, Ferroelectric and Frequency Control. Vol. 60(6): 1134-1140 (2013).
- Lai M. *Giraffe: Using Deep Reinforcement Learning to Play Chess*. Thesis. Imperial College, London (2015).
- Larsen SJ, Alkaersig FG, Ditzel HJ, Jurisica I, Alcatraz N, Baumbach J. *A Simulated Annealing Algorithm for Maximum Common Edge Subgraph Detection in Biological Networks*. Proc. 2016 Genetic and Evolutionary Computation Conf (GECCO), Denver (2016).
- Larsen SJ. *Finding Patterns in Complex Biomedical Data Using Networks and Molecular Profiling*. Dissertation. University of Southern Denmark, Odense (2019).
- LeCun Y, Bottou L, Bengio Y, Haffner P. *Gradient-Based Learning Applied to Document Recognition*. Proc. IEEE. Vol. 86(11): 2278-2324 (1998).
- Lee B, Staszewski W. *Sensor Location Studies for Damage Detection with Lamb Waves*. J Smart Material and Structures. Vol. 16: 399-408 (2007).
- Lee MC, To C. *Comparison of Support Vector Machine and Back Propagation Neural Network in Evaluating the Enterprise Financial Distress*. Intl J of Artificial Intelligence & Applications, Vol. 1(3): 31-43 (2010).
- Lee Y, Yin L, Wahba G. *Multicategory Support Vector Machines: Theory and Application to the Classification of Microarray Data and Satellite Radiance Data*. J American Statistical Association. Vol. 99: 67-81 (2004).
- Li W. *Mathematical Logic: Foundations for Information Science*. Birkhäuser, Basel / Boston / Berlin (2010).
- Li W, Chen B, Cho Y. *Non-Linear Feature of Phase Matched Lamb Waves in Solid Plate*. J Applied Acoustics. Vol. 160: 107124 (2020).
- Littman M. *Markov Games as a Framework for Multi-Agent Reinforcement Learning*. Proc. 11th Intl Conf on Machine Learning (ICML), New Brunswick (1994).
- Liu H, Zhang Y. *Deep Learning Based Crack Damage Detection Technique for Thin Plate Structures using Guided Lamb Wave Signals*. J Smart Materials and Structure. Vol. 29: 015032 (2019).
- Liu Y, Bi S, Dong M, Zhang Y, Huang J, Zhang J. *A Reinforcement Learning Method for Humanoid Robot Walking*. Proc. IEEE 8th Annual Intl Conf on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Tianjin (2018).
- Locatello F, Bauer S, Lucic M, Rätsch G, Gelly S, Schölkopf B, Bachem O. *Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations*. Proc. Intl Conf on Learning Representations (ICLR), New Orleans (2019).
- Lorenzo PR, Nalepa J, Kawulok M, Ramos LS, Pastor JR. *Particle Swarm Optimization for Hyper-Parameter Selection in Deep Neural Networks*. Proc. 2017 Genetic and Evolutionary Computation Conf (GECCO), Berlin (2017).
- Loureço N, Pereira F, Costa E. *Learning Selection Strategies for Evolutionary Algorithms*. Proc. 11th Intl Conf on Artificial Evolution, Bordeaux (2013).
- Luo H, Dridi M, Grunder O. *Ant Colony Optimization Algorithm for a Transportation Problem in Home Health Care with the Consideration of Carbon Emissions*. Proc. 14th Intl Conf on Artificial Evolution, Mulhouse (2019).
- Lu W, Li J, Li Y, Sun A, Wang J. *A CNN-LSTM-Based Model to Forecast Stock Prices*. J Complexity. Vol. 2020: 6622927 (2020).
- MacQueen J. *Some Methods for Classification and Analysis of Multivariate Observations*. Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley (1965-1966).
- Mallardo V, Aliabadi MH, Sharif Khodaei MH. *Optimal Sensor Positioning for Impact Localization in Smart Composite Panels*. J Intelligent Material Systems and Structures. Vol. 24: 559-573 (2012).
- Malinowski P, Wandowski T, Trendavilova I, Ostachowitz W. *A Phased Array-based Method for Damage Detection and Localization in Thin Plates*. Intl J of Structural Health Monitoring. Vol. 8(1): 5-15 (2009).
- Marin JM, Mengersen K, Robert CP. *Bayesian Modelling and Inference on Mixtures of Distributions*. Handbook of Statistics Vol. 25. Elsevier (2005).
- Markopoulous PP, Karystinos GN, Pados DA. *Optimal Algorithms for L_1 -Subspace Signal Processing*. IEEE Transactions on Signal Processing. Vol. 62(19): 5046-5058 (2014).
- Melo L, Pereira F, Costa E. *Effective Multi-Caste Ant Colony System for Large Dynamic Traveling Salesperson Problems*. Proc. 11th Intl Conf on Artificial Evolution, Bordeaux (2013).
- Mendelson E. *Introduction to Mathematical Logic* (4th Ed.). Chapman & Hall, London / Weinheim / New York / Tokyo / Melbourne / Madras (1997).

- Miguel AG, Pagani A, Carrera E. *Higher-order Structural Theories for Transient Analysis of Multi-mode Lamb waves with Applications to Damage Detection*. J of Sound and Vibration. Vol. 457: 139-155 (2019).
- Misra I, Maaten L. *Self-Supervised Learning of Pretext-Invariant Representations*. Proc. IEEE Conf on Computer Vision and Pattern Recognition (CVPR), Virtual Conf, (2020).
- Mohamed I, Otero FEB. *A Multiobjective Optimization Approach for Market Timing*. Proc. 2020 Genetic and Evolutionary Computation Conf (GECCO), Virtual Conf (2020).
- Mohri M, Rostamizadeh A, Talwalkar A. *Foundations of Machine Learning* (2nd Ed.). The MIT Press, Cambridge/London (2018).
- Mu CH, Xie J, Liu Y, Chen F, Liu Y, Jiao LC. *Memetic Algorithm with Simulated Annealing Strategy and Tightness Greedy Optimization for Community Detection in Networks*. J Applied Soft Computing, Vol. 34: 485-501 (2015).
- Munoz G, Barrado C, Cetin E, Salami E. *Deep Reinforcement Learning for Drone Delivery*. J MDPI Drones. Vol. 3(3): 72 (2019).
- Mustapha S, Ismail Z, Ali Fakh M, Tarhini H. *Sensor Placement Optimization on Complex and Large Metallic and Composite Structures*. Intl J Structural Health Monitoring. Vol. 19(1): 262-280 (2019).
- Ng AY. *Shaping and Policy Search in Reinforcement Learning*. Dissertation. University of California, Berkeley (2003).
- Ng CT. *On Accuracy of Analytical Modeling of Lamb Wave Scattering at Delaminations in Multilayered Isotropic Plates*. Intl J of Structural Stability and Dynamics. Vol. 15(8): 1540010 (2015).
- Nguyen BH, Xue B, Andreae P. *A Particle Swarm Optimization based Feature Selection Approach to Transfer Learning in Classification*. Proc. 2018 Genetic and Evolutionary Computation Conf (GECCO), Kyoto (2018).
- Nirbhay M, Dixit A, Misra RK. *Finite Element Modelling of Lamb Waves Propagation in 3D Plates and Brass Tubes for Damage Detection*. Russian J of Nondestructive Testing. Vol. 53(4): 308-329 (2017).
- Nocedal J, Wright SJ. *Numerical Optimization* (2nd Ed.). Springer Science+Business Media, New York (2006).
- Nokhbatolfoghahai A, Navazi HM, Groves RM. *Use of Delay and Sum for Sparse Reconstruction Improvement for Structural Health Monitoring*. J Intelligent Material Systems and Structures. Vol. 30(18-19): 2919-2931 (2019).
- Ong WH, Chiu WK. *Redirection of Lamb Waves for Structural Health Monitoring*. Hindawi J Smart Materials Research. Vol. 2012: 718686 (2012).
- Ong WH, Chiu WK. *Enhancement of Lamb Wave Based In-Situ Structural Health Monitoring through Guided Local Geometry Changes*. Intl J of Structural Health Monitoring. Vol. 12(4): 339-358 (2013).
- Ooijsaar T. *Vibration Based Structural Health Monitoring of Composite Skin-Stiffener Structures*. Dissertation. University of Twente, Enschede (2014).
- Ostachowitz W, Soman R, Malinowski P. *Optimization of Sensor Placement for Structural Health Monitoring: A Review*. Intl J of Structural Health Monitoring. Vol. 8(3): 963-988 (2019).
- Othmani C, Zhang H. *Lamb Wave Propagation in Anisotropic Multilayered Piezoelectric Laminates Made of PVDF- θ° with Initial Stresses*. J Composite Structures. Vol. 240: 112085 (2020).
- Ozair S, Lynch C, Bengio Y, van den Oord A, Levine S, Sermanet P. *Wasserstein Dependency Measure for Representation Learning*. Proc. 33rd Conf on Neural Information Processing Systems (NIPS), Vancouver (2019).
- Panella F, Boehm J, Loo Y, Kaushik A, Gonzalez D. *Deep Learning and Image Processing for Automated Crack Detection and Defect Measurement in Underground Structures*. Proc. Conf ISPRS TC II Mid-term Symposium Towards Photogrammetry 2020, Riva del Garda (2018).
- Patrick M, Asano YM, Kuznetsova P, Fong R, Henriques JF, Zweig G, Vedaldi A. *Multi-modal Self-Supervision from Generalized Data Transformations* (2020). Available: <https://arxiv.org/abs/2003.04298> (Last online: OCT-2020)
- Pauly P, Peel H, Luo S, Hogg D, Fuentes R. *Deeper Networks for Pavement Crack Detection*. Proc. Intl Symposium on Automation and Robotics in Construction, Taipei (2017).
- Pearson K. *On Lines and Planes of Closest Fit to Systems of Points in Space*. Philosophical Magazine. Vol. 2(11): 559-572 (1901).
- Pincus M. *A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems*. J of Operation Research. Vol. 18(6): 1225-1228 (1970).
- Portelli G, Pallez D. *Image Signal Processor Parameter Tuning with Surrogate-Assisted Particle Swarm Optimization*. Proc. 14th Intl Conf on Artificial Evolution, Mulhouse (2019).

- Pouliot GA. *Equivalence of Multicategory SVM and Simplex Cone SVM: Fast Computations and Statistical Theory*. Proc. 35th Intl Conf on Machine Learning (ICML), Stockholm (2018).
- Ragusa VR, Mathias HD, Kazakova VA, Wu AS. *Enhanced Genetic Path Planning for Autonomous Flight*. Proc. 2017 Genetic and Evolutionary Computation Conf (GECCO), Berlin (2017).
- Rappel H, Yousefi-Koma A, Jamali J, Bahari A. *Numerical Time-Domain Modeling of Lamb Wave Propagation Using Elastodynamic Finite Integration Technique*. J Shock and Vibration. Vol. 2014: 434187 (2014).
- Rathor S. *Simple RNN vs GRU vs LSTM: Difference Lies in More Flexible Control* (2018). <https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57> (Last online: JUL-2020)
- Ravi Kiran B, Sobh I, Talpaert V, Mannion P, Sallab AA, Yogamani S, Perez P. *Deep Reinforcement Learning for Autonomous Driving: A Survey* (2020). Available: <https://arxiv.org/abs/2002.00444> (Last online: JUL-2020)
- Reddy YCA, Viswanath P, Reddy B. *Semi-supervised Learning: A Brief Review*. Intl J of Engineering & Technology. Vol. 7(18): 81-85 (2018).
- Rose JL, Pilarski A, Ditri J. *An Approach to Guided Wave Mode Selection for Inspection of Laminated Plate*. J Reinforced Plastics and Composites. Vol. 12(5): 536-544 (1993).
- Rosenblatt F. *The Perceptron: A Probabilistic Model for Information Storage and Organization*. Brain, Cornell Aeronautical Laboratory, Psychological Review. Vol. 65(6): 386-408 (1958).
- Rumelhart DE, Hinton GE, Williams RJ. *Learning Representations by Back-Propagating Errors*. Nature. Vol. 323(9): 534-536 (1986).
- Russell S, Norvig P. *Artificial Intelligence, Global Edition (4th Ed.)*. Pearson New Jersey (2021).
- Sahoo D, Pham Q, Lu J, Hoi SCH. *Online Deep Learning: Learning Deep Neural Networks on the Fly*. Proc. 27th Intl Joint Conf on Artificial Intelligence (IJCAI), Stockholm (2018).
- Sainath TN, Vinyals O, Senior A, Sak H. *Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks*. IEEE Intl Conf on Acoustics, Speech and Signal Processing (ICASSP), Brisbane (2015).
- Santoni GB, Yu L, Xu B, Giurgiutiu V. *Lamb Wave-Mode Tuning of Piezoelectric Wafer Active Sensors for Structural Health Monitoring*. J Vibration and Acoustics. Vol. 129(6): 752-762 (2007).
- Sawaf F, Groves RM. *Phase Discontinuity Predictions Using a Machine-Learning Trained Kernel*. Applied Optics. Vol. 53(24): 5439-5447 (2014).
- Schmidt D, Sinapius M, Wierach P. *Design of Mode Selective Actuators for Lamb Wave Excitation in Composite Plates*. CEAS Aeronautical J. Vol. 4: 103-112 (2013).
- Schubert KJ, Brauner C, Herrmann AS. *Non-Damage-Related Influences on Lamb Wave-based Structural Health Monitoring of Carbon Fiber-Reinforced Plastic Structures*. Intl J of Structural Health Monitoring. Vol. 13(2): 158-176 (2014).
- Settles B. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648 (January-26-2010 Update). University of Wisconsin-Madison, Madison (2010).
- Sghir I, Hao JK, Jaafar IB, Ghédira K. *A Recombination-Based Tabu Search Algorithm for the Winner Determination Problem*. Proc. 11th Intl Conf on Artificial Evolution, Bordeaux (2013).
- Shao K, Zhao D, Li N, Zhu Y. *Learning Battles in Vizdoom via Deep Reinforcement Learning*. Proc. IEEE Conf on Computational Intelligence and Games (CIG), Maastricht (2018).
- Shen Y, Tan S, Sordoni A, Courville A. *Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks*. Proc. Intl Conf on Learning Representations, New Orleans (2019).
- Shi X, Chen Z, Wang H, Yeung DY, Wong WK, Woo WC. *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting*. Proc. 28th Intl Conf on Neural Information Processing Systems (NIPS), Montréal (2015).
- Shi Y. *Analysis of Optimum Lamb Wave Tuning*. Dissertation. Massachusetts Institute of Technology (MIT), Cambridge (2002).
- Shi Y, Eberhart RC. *A Modified Particle Swarm Optimizer*. Proc. of IEEE Intl Conf on Evolutionary Computation (1998).
- Shoenfield JR. *Mathematical Logic (2nd Ed.)*. CRC Press, Boca Raton / New York (2010).
- Simonyan K, Zisserman A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Proc. Intl Conf on Learning Representations (ICLR), San Diego (2015)
- Šofer M, Ferfecki P, Šofer P. *Numerical Solution of Rayleigh-Lamb Frequency Equation for Real, Imaginary and Complex Wavenumbers*. MATEC Web of Conferences 157: 08011 (2018).
- Sohn H, Lee SJ. *Lamb Wave Tuning Curve Calibration for Surface-bonded Piezoelectric Transducers*. J Smart Materials and Structures. Vol. 19(1): 015007 (2010).

- Soman R, Malinowski P. *A Real-Valued Genetic Algorithm for Optimization of Sensor Placement for Guided Wave-Based Structural Health Monitoring*. Hindawi J of Sensors. Vol. 2019: 9614630 (2019).
- Soman R, Kudela P, Balasubramaniam K, Singh SK, Malinowski P. *A Study of Sensor Placement Optimization Problem for Guided Wave-Based Damage Detection*. MDPI J Sensors. Vol. 19: 1856 (2019).
- Soma-Sekhar BV, Balasubramaniam K, Krishnamurthy CV. *Structural Health Monitoring of Fiber-reinforced Composite Plates for Low-velocity Impact Damage using Ultrasonic Lamb Wave Tomography*. Intl J of Structural Health Monitoring. Vol. 5(3): 243-253 (2006).
- Sutskever I, Vinyals O, Le QV. *Sequence to Sequence Learning with Neural Networks*. Proc. 27th Intl Conf on Neural Information Processing Systems (NIPS), Montréal (2014).
- Stamp M. *A Revealing Introduction to Hidden Markov Models*. In: Introduction to Machine Learning with Applications in Information Security. Chapman and Hall/CRC, New York (2017).
- Staszewski WJ, Sohn H. *Signal Processing for Structural Health Monitoring*. In: Encyclopedia of Structural Health Monitoring, John Wiley & Sons Ltd (2009).
- Stawiarski A, Muc A. *On Transducers Localization in Damage Detection by Wave Propagation Method*. MDPI J Sensors. Vol. 19: 1937 (2019).
- Su Z, Ye L. *Identification of Damage Using Lamb Waves: From Fundamentals to Applications*. Springer, Berlin / Heidelberg (2009).
- Szegedy C, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. *Going Deeper with Convolutions*. Proc. IEEE Conf on Computer Vision and Pattern Recognition (CVPR), Boston (2015).
- Szegedy C, Ioffe S, Vanhoucke V. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. Proc. 31st AAAI Conf on Artificial Intelligence, San Francisco (2017).
- Szita I. *Reinforcement Learning in Games*. In: Reinforcement Learning: Adaptation, Learning, and Optimization. Springer, Berlin / Heidelberg (2012).
- Taltavull A, Qiu L, Venkat RS, Dürager C, Boller C, Ren Y, Yuan S. *Simulation, Realization and Validation of Guided Wave SHM System Solutions for Aircraft Metallic Structural Repairs*. Proc. 11th Intl Workshop on Structural Health Monitoring (IWSHM), Stanford (2017).
- Tang Z, Shao K, Zhao D, Zhu Y. *Recent Progress of Deep Reinforcement Learning: From AlphaGo to AlphaGo Zero*. J Control Theory and Applications. Vol. 34(12): 1529-1546 (2017).
- Teixeira MAM, Goulart F, Campelo F. *Evolutionary Multiobjective Optimization of Winglets*. Proc. 2016 Genetic and Evolutionary Computation Conf (GECCO), Denver (2016).
- Tesauro G. *Temporal Difference Learning and TD-Gammon*. Communications of the ACM. Vol. 38(3): 58-68 (1995).
- Thiene M, Sharif Khodaei Z, Aliabadi MH. *Optimal Sensor Placement for Maximum Area Coverage for Damage Localization in Composite Structures*. J Smart Materials and Structures. Vol. 25: 095037 (2016).
- Tovar M, Robles M, Rashid F. *PV Power Prediction, Using CNN-LSTM Hybrid Neural Network Model. Case of Study: Temixco-Morelos, México*. J Energies. Vol. 13(24): 1-15 (2020).
- Triguero I, Garcia S, Herrera F. *Self-labeled Techniques for Semi-supervised Learning: Taxonomy, Software and Empirical Study*. J Knowledge Information System. Vol. 42: 245-284 (2015).
- Trojanowski K, Raciborski M, Kaczynski P. *Adaptive Differential Evolution with Hybrid Rules of Perturbation for Dynamic Optimization*. Proc. Intl Joint on Conf on Computational Intelligence, Paris (2011).
- Tschannen M, Djolonga J, Rubenstein PK, Gelly S, Lucic M. *On Mutual Information Maximization for Representation Learning*. Proc. Intl Conf Learning Representations, Virtual Conf (2020).
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. *Attention Is All You Need*. Proc. 30th Intl Conf on Neural Information Processing Systems (NIPS), Long Beach (2017).
- Venkat RS, Boller C, Qiu L, Ravi NB, Mahapatra DR, Chakraborty N. *Integrated Approach to Demonstrate Optimum Sensor Positions in a Guided Wave Based SHM System Using Numerical Simulation*. Proc. 8th Intl Symposium on NDT in Aerospace, Bangalore (2016).
- Vidal A, Kristjanpoller W. *Gold Volatility Prediction using a CNN-LSTM Approach*. J Expert Systems with Applications. Vol. 157: 113481 (2020).
- Vormer F, Mulder M, Mulder JA, Van Paassen MM. *Optimization of Flexible Approach Trajectories Using a Genetic Algorithm*. J Aircraft 43(4): 941-952 (2006).
- Wang T. *Finite Element Modelling and Simulation of Guided Wave Propagation in Steel Structural Members*. Master Thesis. University of Western Sydney, Sydney (2014).

- Wang C, Xue B, Shang L. *PSO-based Parameters Selection for the Bilateral Filter in Image Denoising*. Proc. 2017 Genetic and Evolutionary Computation Conf (GECCO), Berlin (2017).
- Wang Y, Qian Y, Li Y, Gong M, Banzhaf W. *Artificial Multi-Bee-Colony Algorithm for k-Nearest-Neighbor Fields Search*. Proc. 2016 Genetic and Evolutionary Computation Conf (GECCO), Denver (2016).
- Wilcox PD. *Lamb Wave Inspection of Large Structures Using Permanently Attached Transducers*. Dissertation. Imperial College, London (1998).
- Wilcox PD, Lowe MJS, Cawley P. *Mode and Transducer Selection for Long Range Lamb Wave Inspection*. J Intelligent Material Systems and Structures. Vol. 12(8): 553-565 (2001).
- Wu W, Zhang H, Jia F, Yang X, Liu H, Yuan W, Feng XQ, Gu B. *Surface Effects on Frequency Dispersion Characteristics of Lamb Waves in a Nanoplate*. J Thin Solid Films. Vol. 697: 137831 (2020).
- Xu Y, Gao L, Tian K, Zhou SG, Sun H. *Non-Local ConvLSTM for Video Compression Artifact Reduction*. IEEE Intl Conf on Computer Vision (ICCV), Seoul (2019).
- Xue N, Triguero I, Figueredo GP, Landa-Silva D. *Evolving Deep CNN-LSTMs for Inventory Time Series Prediction*. IEEE Congress on Evolutionary Computation (CEC), Wellington (2019).
- Yan ZH, Bo YJ. *Piezoelectric Transducer Parameter Selection for Exciting a Single Mode from Multiple Modes of Lamb Waves*. J Chinese Physics B. Vol. 20(9): 094301 (2011).
- Yousefi B, Kalhor D, Usamentiaga R, Lei L, Castanedo CI, Maldague X. *Application of Deep Learning in Infrared Non-Destructive Testing*. Proc. 14th Quantitative InfraRed Thermography Conf, Berlin (2018).
- Zeng L, Lin J, Lei Y, Xie H. *Waveform Design for High-Resolution Damage Detection Using Lamb Waves*. IEEE Trans on Ultrasonics, Ferroelectrics and Frequency Control. Vol. 60(5): 1025-1029 (2013).
- Zhang L, Ma D, Yang M, Wang S. *Optimization and Analysis of Winglet Configuration for Solar Aircraft*. Chinese J of Aeronautics. Vol. 1549: 1-15 (2020).
- Zhang L, Yang F, Zhang YD, Zhu YJ. *Road Crack Detection Using Deep Convolutional Neural Network*. Proc. IEEE Intl Conf on Image Processing, Phoenix (2016).

Appendix A [Locatello et al. (2019)]

Proof. To show the claim, we explicitly construct a family of functions f using a sequence of bijective functions. Let $d > 1$ be the dimensionality of the latent variable \mathbf{z} and consider the function $g : \text{supp}(\mathbf{z}) \rightarrow [0, 1]^d$ defined by

$$g_i(\mathbf{v}) = P(\mathbf{z}_i \leq v_i) \quad \forall i = 1, 2, \dots, d.$$

Since P admits a density $p(\mathbf{z}) = \prod_i p(z_i)$, the function g is bijective and, for almost every $\mathbf{v} \in \text{supp}(\mathbf{z})$, it holds that $\frac{\partial g_i(\mathbf{v})}{\partial v_i} \neq 0$ for all i and $\frac{\partial g_i(\mathbf{v})}{\partial v_j} = 0$ for all $i \neq j$. Furthermore, it is easy to see that, by construction, $g(\mathbf{z})$ is a independent d -dimensional uniform distribution. Similarly, consider the function $h : (0, 1]^d \rightarrow \mathbb{R}^d$ defined by

$$h_i(\mathbf{v}) = \psi^{-1}(v_i) \quad \forall i = 1, 2, \dots, d,$$

where $\psi(\cdot)$ denotes the cumulative density function of a standard normal distribution. Again, by definition, h is bijective with $\frac{\partial h_i(\mathbf{v})}{\partial v_i} \neq 0$ for all i and $\frac{\partial h_i(\mathbf{v})}{\partial v_j} = 0$ for all $i \neq j$. Furthermore, the random variable $h(g(\mathbf{z}))$ is a d -dimensional standard normal distribution.

Let $\mathbf{A} \in \mathbb{R}^{d \times d}$ be an arbitrary orthogonal matrix with $A_{ij} \neq 0$ for all $i = 1, 2, \dots, d$ and $j = 1, 2, \dots, d$. An infinite family of such matrices can be constructed using a Householder transformation: Choose an arbitrary $\alpha \in (0, 0.5)$ and consider the vector \mathbf{v} with $v_1 = \sqrt{\alpha}$ and $v_i = \sqrt{\frac{1-\alpha}{d-1}}$ for $i = 2, 3, \dots, d$. By construction, we have $\mathbf{v}^T \mathbf{v} = 1$ and both $v_i \neq 0$ and $v_i \neq \sqrt{\frac{1}{2}}$ for all $i = 1, 2, \dots, d$. Define the matrix $\mathbf{A} = \mathbf{I}_d - 2\mathbf{v}\mathbf{v}^T$ and note that $A_{ii} = 1 - 2v_i^2 \neq 0$ for all $1, 2, \dots, d$ as well as $A_{ij} = -v_i v_j \neq 0$ for all $i \neq j$. Furthermore, \mathbf{A} is orthogonal since

$$\mathbf{A}^T \mathbf{A} = (\mathbf{I}_d - 2\mathbf{v}\mathbf{v}^T)^T (\mathbf{I}_d - 2\mathbf{v}\mathbf{v}^T) = \mathbf{I}_d - 4\mathbf{v}\mathbf{v}^T + 4\mathbf{v}(\mathbf{v}^T \mathbf{v})\mathbf{v}^T = \mathbf{I}_d.$$

Since \mathbf{A} is orthogonal, it is invertible and thus defines a bijective linear operator. The random variable $\mathbf{A}h(g(\mathbf{z})) \in \mathbb{R}^d$ is hence an independent, multivariate standard normal distribution since the covariance matrix $\mathbf{A}^T \mathbf{A}$ is equal to \mathbf{I}_d .

Since h is bijective, it follows that $h^{-1}(\mathbf{A}h(g(\mathbf{z})))$ is an independent d -dimensional uniform distribution. Define the function $f : \text{supp}(\mathbf{z}) \rightarrow \text{supp}(\mathbf{z})$

$$f(\mathbf{u}) = g^{-1}(h^{-1}(\mathbf{A}h(g(\mathbf{u}))))$$

and note that by definition $f(\mathbf{z})$ has the same marginal distribution as \mathbf{z} under P , i.e., $P(\mathbf{z} \leq \mathbf{u}) = P(f(\mathbf{z}) \leq \mathbf{u})$ for all \mathbf{u} . Finally, for almost every $\mathbf{u} \in \text{supp}(\mathbf{z})$, it holds that

$$\frac{\partial f_i(\mathbf{u})}{\partial u_j} = \frac{A_{ij} \cdot \frac{\partial h_j(g(\mathbf{u}))}{\partial v_j} \cdot \frac{\partial g_j(\mathbf{u})}{\partial u_j}}{\frac{\partial h_i(h_i^{-1}(\mathbf{A}h(g(\mathbf{u}))))}{\partial v_i} \cdot \frac{\partial g_i(g^{-1}(h^{-1}(\mathbf{A}h(g(\mathbf{u}))))}{\partial v_i}} \neq 0,$$

as claimed. Since the choice of \mathbf{A} was arbitrary, there exists an infinite family of such functions f . \square

3. Theoretical Background

In this chapter, the theoretical background that is necessary to understand the discussion will be described. The organization of this chapter is as follows: the theoretical background of Lamb wave propagation and its numerical simulation is given in section 3.1. In section 3.2, I will describe deterministic and heuristic approaches for discrete optimization. Section 3.3 concerns the signal processing of time-series signal, and in section 3.4, I will elaborate on some theoretical perspectives from computational neuroscience. Finally, in section 3.5, the theory and some assumptions of some commonly used machine and deep learning algorithms will be described.

3.1. Lamb Wave and Simulated Propagation

This section contains four sub-sections: In sub-section 3.1.1 I will describe the physics of Lamb waves while the simulation of Lamb wave propagation with numerical modelling is given in sub-section 3.1.2, while sub-section 3.1.3 will describe the piezoelectric effect which is needed to excite the Lamb wave. Finally, in sub-section 3.1.4, a brief discussion on the attenuation of Lamb wave in different materials in order to justify the selection of certain excitation parameters is given.

3.1.1. Acoustic Wave in Plate-Like Structure

Acoustic waves in bulk mode propagates in solid media both as longitudinal and transversal waves. In 3-dimensional space, the acoustic wave propagation can be described by the general wave equation:

$$\frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} = \left(\frac{\partial^2 p}{\partial x_1^2} + \frac{\partial^2 p}{\partial x_2^2} + \frac{\partial^2 p}{\partial x_3^2} \right) \quad (3.1.1-1)$$

In Eq. 3.1.1-1, c is the speed of sound, p is mechanical pressure, t is time, and $x_1 \dots x_3$ are the spatial coordinate signifiers. A longitudinal wave oscillates in the direction of the propagation, while a transversal wave oscillates perpendicular to its direction of propagation. In isotropic media, the longitudinal bulk wave speed c_L and transversal bulk wave speed c_T are defined as [Su and Ye (2009)]:

$$c_L = \sqrt{\frac{\lambda_{\text{Lamé}} + 2\mu}{\rho}} = \sqrt{\frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}} \quad , \quad c_T = \sqrt{\frac{E}{2\rho(1+\nu)}} = \sqrt{\frac{\mu}{\rho}} \quad (3.1.1-2)$$

Further, in Eq. 3.1.1-2, E is the Young modulus of the material, ν is Poisson's ratio, $\lambda_{\text{Lamé}}$ is the Lamé constant, ρ is the material density, and μ is the shear modulus of the plate. For a damage tolerant aircraft substructure such as an aluminum fuselage, the material can be regarded as a thin isotropic and homogenous plate, i.e., when one geometrical dimension is significantly smaller than the other two dimensions as can be seen Fig. 3.1.1-1.

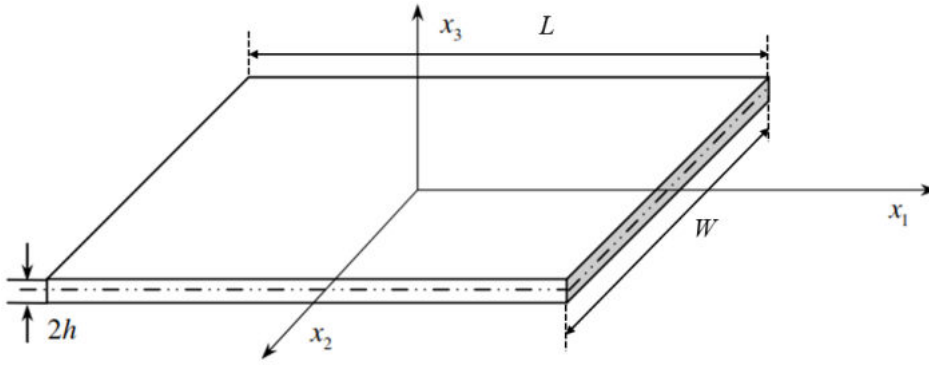


Fig. 3.1.1-1: Thin plate with a length L , width W , and thickness $2h \ll (L, W)$ in the Cartesian coordinate system [Su and Ye (2009)]. The wave propagation direction is parallel to x_1 and x_2 .

When the plate thickness $2h$ is approximately the same as the wavelength (i.e., $2h \approx \lambda_{\text{wave}}$), the general wave equation from Eq. 3.1.1-1 can be simplified as given in Eq. 3.1.1-3 [Su and Ye (2009)], where u is the particle displacement and f is the body force in the x_i direction, respectively.

$$(\lambda_{\text{Lamé}} + \mu) \cdot u_{j,ji} + \mu \cdot u_{i,ji} + \rho \cdot f_i = \rho \cdot \ddot{u}_i \quad (i, j = 1, 2, 3) \quad (3.1.1-3)$$

To solve Eq. 3.1.1-3, we need to first to decompose it into two uncoupled parts under the plane strain condition with the displacement potentials method which is based on the Helmholtz theorem [Rose (1999), Achenbach (1973)] that takes the longitudinal and transversal wave velocity c_L and c_T into account:

$$\frac{\partial^2 \Phi}{\partial x_1^2} + \frac{\partial^2 \Phi}{\partial x_3^2} = \frac{1}{c_L} \frac{\partial^2 \Phi}{\partial t^2} = \sqrt{\frac{\rho}{\lambda_{\text{Lamé}} + 2\mu}} \frac{\partial^2 \Phi}{\partial t^2} \quad (3.1.1-4)$$

$$\frac{\partial^2 \Psi}{\partial x_1^2} + \frac{\partial^2 \Psi}{\partial x_3^2} = \frac{1}{c_T} \frac{\partial^2 \Psi}{\partial t^2} = \sqrt{\frac{\rho}{2\mu}} \frac{\partial^2 \Psi}{\partial t^2} \quad (3.1.1-5)$$

Where the scalar potential Φ and vector potential Ψ are defined as:

$$\Phi = [A_1 \sin(px_3) + A_2 \cos(px_3)] \exp[i(kx_1 - \omega t)] \quad (3.1.1-6)$$

$$\Psi = [B_1 \sin(px_3) + B_2 \cos(px_3)] \exp[i(kx_1 - \omega t)] \quad (3.1.1-7)$$

A_1, A_2, B_1 and B_2 are four constants determined by the boundary conditions. The relation between the non-dimensional parameters p and q , wavenumber k , angular frequency ω , wavelength λ_{wave} and longitudinal and transversal bulk wave velocities c_L and c_T is given by:

$$p^2 = \frac{\omega^2}{c_L^2} - k^2 \quad , \quad q^2 = \frac{\omega^2}{c_T^2} - k^2 \quad , \quad k = \frac{2\pi}{\lambda_{\text{wave}}} \quad (3.1.1-8)$$

A Lamb mode is the result of the wave being constrained by two surfaces with a thickness on the same magnitude as the wavelength combined with interference phenomena that creates a standing wave pattern in the thickness direction while

propagating in horizontal direction [Giurgiutiu (2014)]. Due to this property, a Lamb wave contains both longitudinal and transverse components, i.e., the particle motions are in parallel and perpendicular directions to the propagation direction, respectively. The general description of Lamb waves in an isotropic and homogeneous plate is given by [Su and Ye (2009)]:

$$\frac{\tan(qh)}{\tan(ph)} = \frac{4k^2(pq)\mu}{(k^2\lambda_{\text{Lamé}} + p^2\lambda_{\text{Lamé}} + 2\mu p^2)(k^2 - q^2)} \quad (3.1.1-9)$$

Eq. (3.1.1-9) can be split into two equations by substituting Eq. (3.1.1-7) and (3.1.1-8) to obtain the symmetric and anti-symmetric characteristics of the Lamb wave, respectively:

$$\frac{\tan(qh)}{\tan(ph)} = \frac{-4k^2(pq)}{(k^2 - q^2)} \quad , \quad \frac{\tan(qh)}{\tan(ph)} = \frac{(k^2 - q^2)}{-4k^2(pq)} \quad (3.1.1-10)$$

The symmetric and anti-symmetric characteristics of Lamb waves refers to the particle displacement u_1 and u_3 relative to the plate mid-plane, as depicted in Fig. 3.1.1-2a-b, respectively. Since the symmetric Lamb mode (S-Mode) has a quasi-pressure displacement field, it is sometimes referred as the compressional Lamb wave mode, and it is sensitive to defects anywhere in the thickness direction [Marcos (2011)]. The anti-symmetric Lamb wave (A-Mode) has a quasi-flexural displacement field, and it is sometimes referred as a bending Lamb mode and it is more sensitive to defect in wave propagation direction such as surface cracks or delamination [Guy et al. (2003), Liu et al. (2013)].

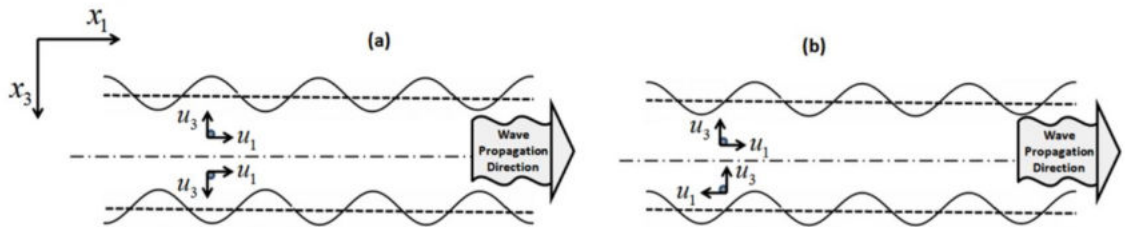


Fig. 3.1.1-2: a). Symmetric Lamb wave (S-Mode) and b). Anti-symmetric Lamb wave (A-Mode). [Pant et al. (2013)]

Lamb waves are dispersive, i.e., their velocities are dependent on the wave frequency and plate thickness. The fundamental symmetric and anti-symmetric Lamb modes (referred as the A_0 -Mode and the S_0 -Mode, respectively) are always present during the propagation, whereas at a higher frequency or for thicker plates (or the combination both), the higher order Lamb modes ($A_1, S_1, \dots, A_2, S_2, \dots$) will also occur. During propagation, these wave modes will reflect and then overlap with each other as soon as the wave front touches each free boundary.

The solution of Eq. (3.1.1-10) describes the dispersive behavior of the Lamb waves, and it can be numerically calculated by combining Eq. (3.1.1-8) and Eq. (3.1.1-10) with the material properties and plate thickness. Further, the solution can also be drawn as a dependency of wave velocity as a function of frequency-thickness

product for each Lamb mode and therefore it is referred as a dispersion curve. An example of a dispersion curve for symmetric and anti-symmetric Lamb modes with the normalized phase velocity in an aluminum plate is depicted in Fig. 3.1.1-3a-b, respectively. At higher frequency-thickness, A_0 and S_0 modes become nearly without the dispersion as can be seen from Fig. 3.1.1-3a-b. Above 2000 [Hz·m], the velocity of A_0 -Mode is quasi-constant and not very much affected by either the frequency or the plate thickness anymore. The same behavior can be observed for the S_0 -Mode above 3000 [Hz·m], see Fig. 3.1.1-3a.

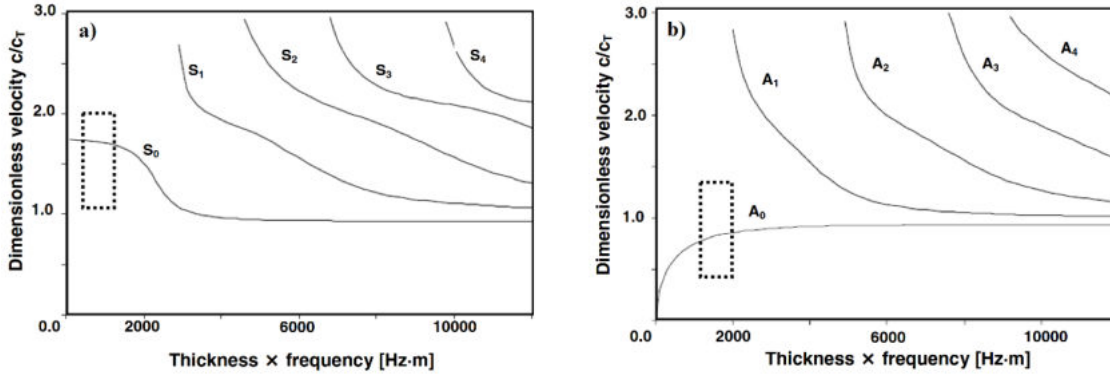


Fig. 3.1.1-3: Dispersion curve for a). S-Modes and b). A-Modes in aluminum plate [Su and Ye (2009)]. Less dispersive region is marked in dashed box.

In bulk waves, each individual wave travels with certain phase velocity and the overall envelope shape of the wave amplitudes travels with the group velocity [Su and Ye (2009)]. Analogously, each Lamb mode propagates with a different phase velocity c_P . The wave packet of Lamb modes propagates at the group velocity c_G , which is defined in Eq. (3.1.1-11).

$$c_G = c_P - \lambda_{\text{wave}} \frac{\partial c_P}{\partial \lambda_{\text{wave}}} \leftrightarrow c_G = c_P - k \frac{\partial c_P}{\partial k} \quad (3.1.1-11)$$

The dispersion curve can also be drawn in terms of group velocity or alternatively in wavenumber, as in Fig. 3.1.1-4a-b, respectively. It is important to understand the dispersion relation of the group velocity since “*it is the actual velocity captured in experiments*” [Su and Ye (2009)].

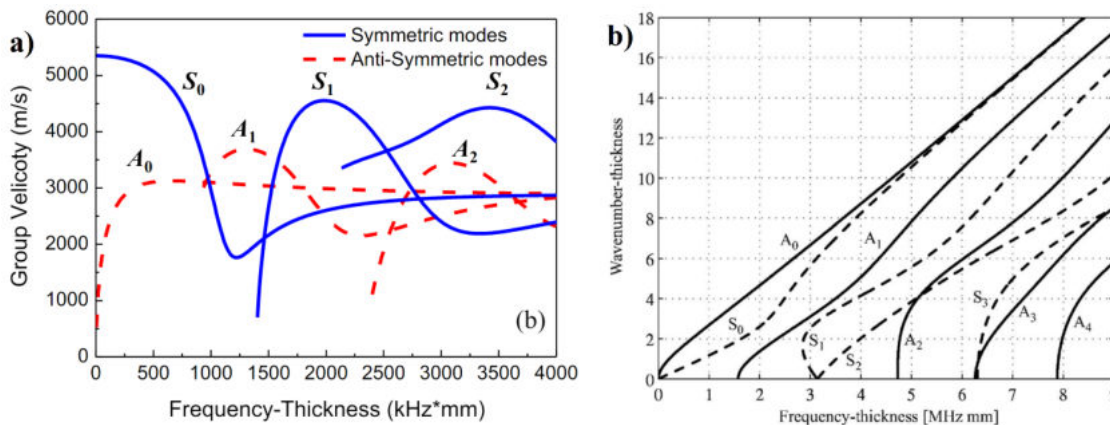


Fig. 3.1.1-4: Lamb wave dispersion curve for an aluminum plate in terms of a). group velocity [Zhao et al. (2017)] and b). wavenumber-thickness [Masserey and Fromme (2008)].

3.1.2. Simulated Lamb-Wave Propagation in Finite Element Environment

For exactly calculating the Lamb wave propagation behavior in a plate, an analytical model must be built. However, building an analytical model for many engineering structures is difficult as many engineering applications are driven by complex systems and their physics is very likely to be governed by multivariate Partial Differential Equations (PDEs). Therefore, when analytically solving multivariate PDE for real-world structures, most of the times simplifications are used resulting in either of following consequences:

1. The analytical solution quality is reduced since it becomes imprecise, or
2. The solution is guaranteed only for certain increment of the structure, thus making it not scalable from the holistic perspective

For this reason, full-analytical solutions for complex structures are many times computationally heavy and numerical solutions are often sought in those cases. According to [Giurgiutiu (2014)], for complex geometries, *“the numerical methods represent the only viable approach to understanding the multiple reflections and diffractions of the ultrasonic waves within the component”*.

There are different mathematical ways one can obtain numerical solutions of Lamb waves such as Finite element method (FEM), Finite difference method (FDM), Elastic Finite integration technique (EFIT) and spectral FEM (SFEM) [Ostachowitz et al. (2012)]. FEM is de facto the most popular numerical analysis method thanks to its flexibility of being adapted to many different engineering problems, i.e., not only for Lamb wave propagation. For this reason, most of the numerical analysis software packages that are commercially available for wide-public are FEM based.

FEM developments are based on the Hamilton's principle, which states that the motion of the system within certain interval vanishes under infinitesimal variations of the displacements and formulated as [Duczek et al. (2014), Cerniglia et al. (2010)]:

1. Lagrangian function in the volumetric integral area Ω , which describes the difference between the kinetic energy and the elastic strain energy
2. External work over volumetric area Ω and surficial integral area Γ

$$\underbrace{\left(\int_{\Omega} \left[\rho \delta u^T \cdot \ddot{u} + \delta \varepsilon^T \cdot C_{ijkl} \cdot \varepsilon \right] d\Omega \right)}_{\text{Lagrange volumetric integral}} = \underbrace{\int_{\Omega} \delta u^T \cdot F_v d\Omega + \int_{\Gamma} \delta u^T \cdot F_s d\Gamma}_{\text{External work}} \quad (3.1.2-1)$$

In Eq. (3.1.2-1), ρ is the material density, u and \ddot{u} are the particle displacement vectors in the material and their corresponding accelerations, respectively, ε is the strain tensor, and C_{ijkl} is the stiffness matrix of the material. The external forces can be classified as surface load F_s and volume load F_v . After some

calculations described in [Zienkiewicz et al. (2005)], Eq. (3.1.2-1) can be written as the well-known equation of motion:

$$M\ddot{u} + C\dot{u} + Ku = F_a \quad (3.1.2-2)$$

In Eq. (3.1.2-2), M is the structural mass matrix; C is the structural damping matrix; K is the structural stiffness matrix; F_a is the vector of applied loads; and \dot{u} is the particle velocity. For metals in the elastic deformation zone (such as Lamb waves propagation), the structural damping can be neglected, thus Eq. (3.1.2-2) can be regarded as dependent of particle displacement and acceleration only. To numerically solve the PDE in Eq. (3.1.2-2), the geometry involved is divided into mesh elements over which the equation can be approximated [Zienkiewicz et al (2005), Wriggers (2008)].

A mesh element is a spatial discrete representation of the geometry. For a three-dimensional problem, there are four element types: brick, tetrahedral, prism, and pyramid. In most commercial FE software, two computationally feasible polynomial approximation methods exist: linear ($p=1$) and quadratic ($p=2$). Both determine the number of nodes in each element type (see Fig. 3.1.2-1).

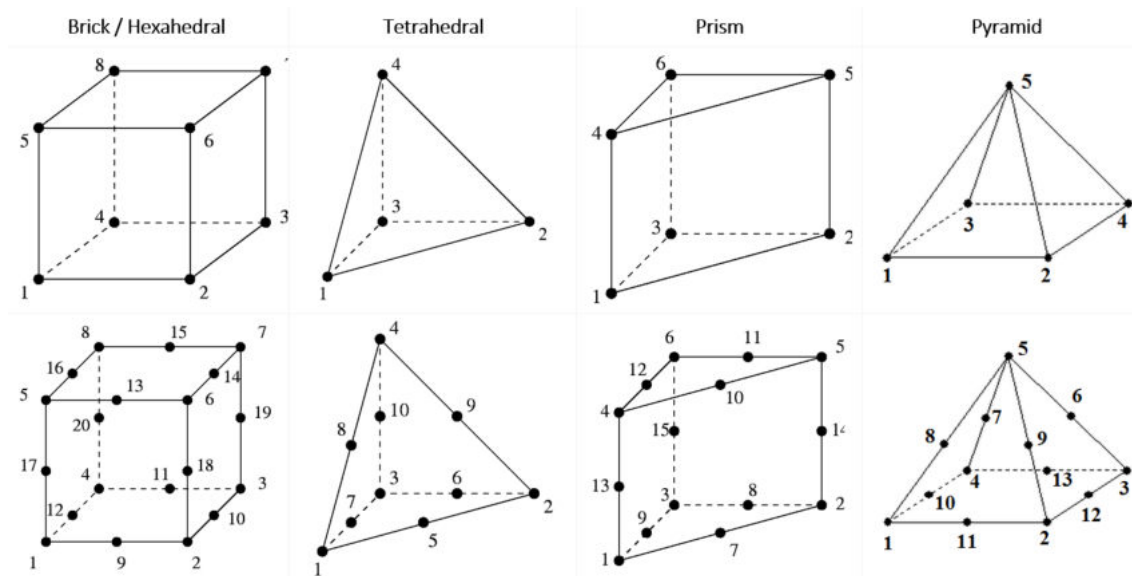


Fig. 3.1.2-1: Mesh Element Type. Top: Linear Mesh ($p = 1$), Bottom: Quadratic Mesh ($p = 2$)

To reach a better calculation result, the mesh can be refined by decreasing the distance between each node so that the size of the mesh element becomes smaller, thus increasing the required number of elements to cover the whole geometry. This is called h -type refinement, where h signifies the mesh element size. There is another refinement called p -type refinement [Gopalakrishnan et al. (2008)], which in contrast to the h -type, focuses on higher order p -polynomial approximations such as cubic ($p=3$), quartic ($p=4$), and so on. Higher order p -type refined finite element is also called spectral element (SE) or p -FE. Deep research into p -FE methods for application in Lamb wave propagation has been investigated by [Pahlavan (2012)], however this did not involve any commercial FE software, since at the moment there was no such SE-FE software available in

the market. The finite element size, h_e is derived from the smallest wavelength λ_{\min} . In the linear case, [Moser et al. (1999)] recommended 20 nodes per wavelength for good spatial resolution, which can be written as:

$$h_e = \frac{\lambda_{\min}}{20} \quad (3.1.2-3)$$

Beside the spatial discretization, the time discretization of equation Eq. (3.1.2-2) is needed as well. The minimum requirement to ensure numerical stability of time integration is sufficed by the Courant-Friedrich-Lewy (CFL) condition [Duczek et al. (2014)]:

$$\Delta t_{\text{CFL}} = \frac{h}{c_G}, \quad \Delta t_{\text{rec}} = \frac{1}{20 \cdot f_{\max}} \quad (3.1.2-4)$$

The CFL condition stipulates that the wave should not travel more than one element width h in a single time increment Δt_{CFL} . For the Newmark time integrator, the recommended time step Δt_{rec} is 20 increments per cycle of the maximum frequency f_{\max} as given in Eq. (3.1.2-4), so that solutions can be calculated in efficient manner, especially for ultrasounds with frequencies in the MHz [Cerniglia et al. (2010), Gresil et al. (2012)].

3.1.3. Piezoelectric Actuator and Sensor

There are several transducer technologies that can be used for generating Lamb waves, such as conventional and air-coupled piezoelectricity, electromagnetic, and laser transducer [Su and Ye (2009), Boller et al. (2004), Döring (2011), Güemes and Ostachowitz (2013)]. For application in ultrasonic SHM, the most common method is to use a conventional piezoelectric patch of type lead zirconate titanate or piezoelectric wafer active sensor (PWAS, Fig. 3.1.3-1 a-b) due to their relatively cheap price and their lightweight.

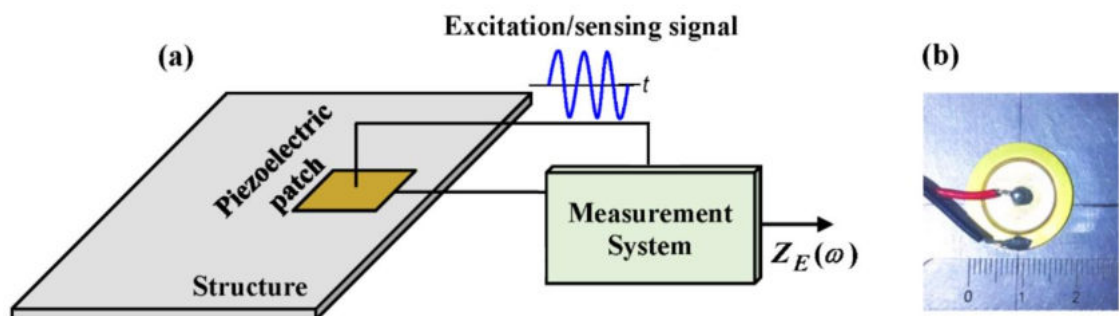


Fig. 3.1.3-1: a). piezoelectric patch mounted on a structure, b). typical thin cylindrical PZT patch [Budoya et al. (2017)]

Piezoelectricity effect was discovered by Jacques and Pierre Curie in the 1880s. “It is an anisotropic property of crystalline materials and results from non-uniform charge distributions within a crystal’s cells.” [Boller et al. (2004)]. This property can manifest itself in two ways:

1. Direct piezoelectric effect, which occurs in the materials where an electrical charge is generated due to an applied mechanical force
2. Inverse piezoelectric effect which is the occurrence of an internal strain due to an applied electrical field.

The inverse piezoelectric effect is used for generating acoustic waves, while the direct piezoelectric effect is used for detecting acoustic waves – see Fig. 3.1.3-2 a-b for illustrations.

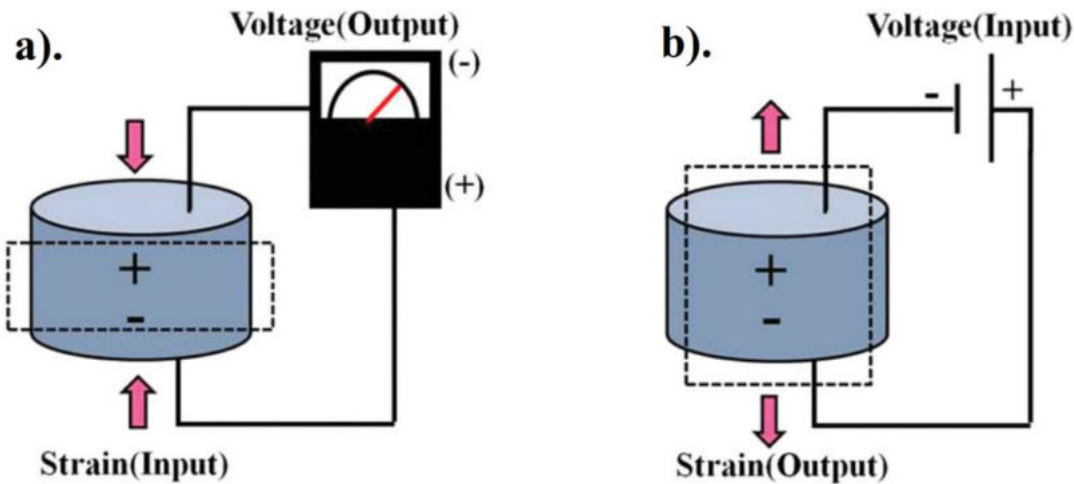


Fig. 3.1.3-2: a). direct piezoelectric effect, b). inverse piezoelectric effect [Mishra et al. (2018)]

There are several material groups which exhibit these properties. The most common are the piezoceramics with Perovskite structure (ABO_3), such as: $BaTiO_3$, $CaTiO_3$, and $SrTiO_3$. Piezoceramic sensors are particularly attractive for damage detection in a structure because they can be used both as actuator and sensor (sender and receiver) and this allows both passive and active damage detection. The most commonly used transducers come from this group: the PZT (lead zirconate titanate) or $Pb[Zr_xTi_{1-x}]O_3$, with $0 \leq x \leq 1$. The second group is the piezoelectric polymers, material being the polyvinylidene fluoride (PVDF). Also, some natural materials like quartz, topaz, dry bone, and silk exhibit piezoelectricity. Furthermore, III-V and II-VI semiconductors are also piezoelectric due to their crystal asymmetry.

The electromechanical effect in piezoelectricity is described through the tensorial piezoelectric constitutive equations [Gresil et al. (2012)]:

$$S_{ij} = S_{ijkl}^E T_{kl} + d_{kij}^E E_k \quad (3.1.3-1)$$

$$D_j = d_{jkl}^T T_{kl} + \epsilon_{jk}^T E_k \quad (3.1.3-2)$$

where, S_{ij} is the mechanical strain; T_{kl} is the mechanical stress; E_k is the electrical field; D_j is the electrical displacement; s_{ijkl}^E is the mechanical compliance of the material measured at zero electric field, ϵ_{jk}^T is the dielectric permittivity measured at zero mechanical stress, and d_{kij} represents the piezoelectric coupling effect.

3.1.4. Lamb Wave Attenuation

Like Rayleigh waves, the energy of plate waves dissipates during propagation, and this is also the case for Lamb waves and in-plane shear horizontal (SH)-waves. This phenomenon is called attenuation and it can be observed with a gradually decreasing of the signal amplitude [Su and Ye (2009)]. When Lamb waves propagate inside the absorptive material, their energy is absorbed by the material, which results from friction between material particles, thus converting the wave energy to heat during propagation [Luangvilai (2007)].

In the presence of material inhomogeneities such as stiffeners, corrosion, crack, rivet holes, etc. the attenuation is more pronounced. S-Mode Lamb waves tend to travel farther than the A-Mode due to their dominant in-plane particle displacement, whereas the energy of an A-Mode is partially leaking out along the free surfaces due to its dominant perpendicular particle displacement. The relatively high attenuation of the A-Mode becomes more pronounced when the structures are immersed in water or buried in soil [Wilcox et al. (2005)]. The time-harmonic solution to Eq. 3.1.1-3 can be written as:

$$u(x, t) = A \cdot \exp[i(\omega t - kx)] \quad (3.1.4-1)$$

Where in Eq. (3.1.3-1), A is the amplitude of the Lamb wave. The measured signal power P at point x after having an original excitation signal power P_0 in a plate that has the geometrical attenuation factor α which is proportional to $1/\sqrt{x}$ and material attenuation coefficient β is given in Eq. (3.1.4-2). Alternatively, it can also be expressed in energy E as a function of original excitation energy E_0 as given in Eq. (3.1.4-3) [Ono and Gallego (2012)].

$$P = P_0[\alpha \cdot \exp(-\beta x)] \quad , \quad \text{where} \quad \alpha = \left(\frac{1}{\sqrt{x}} \right) \quad (3.1.4-2)$$

$$E = E_0 \cdot \left(\frac{1}{x} \right) \cdot \exp(-2\beta x) \quad (3.1.4-3)$$

The amplitude of Lamb waves signals in a plate decay at a rate that is proportional to the inverse square root of the propagation distance. The ratio between amplitudes $A(d_1)$ and $A(d_2)$ of distances d_1 and d_2 at two points along the propagation path is given by Eq. (3.1.4-4). Due to the consecutive attenuation, [Su and Ye (2009), Konstantinidis et al. (2006)] suggested to compensate the energy loss due to geometrical spreading by multiplying the signal amplitude with the square root of elapsed time as given by Eq. (3.1.4-5).

$$\frac{A(d_1)}{A(d_2)} = \frac{\sqrt{d_2}}{\sqrt{d_1}} \quad (3.1.4-4)$$

$$A'(t) = A(t)\sqrt{t} \quad (3.1.4-5)$$

The material attenuation β depends on frequency and thickness, e.g. for a 1-mm thick aluminum plate, the attenuation coefficient is between 2.2 – 17 dB/m for an excitation frequency between 0.5 – 5 MHz, where for a thermoplastic material such as polymethylmethacrylate (PMMA), the attenuation is higher by a factor of 10 – 50x [Ono and Gallego (2012), Fig. 3.1.4-1a] due to its more pronounced viscoelastic property in comparison to metals such as steel [Fig. 3.1.4-1b] or aluminum which only exhibit negligible viscous properties.

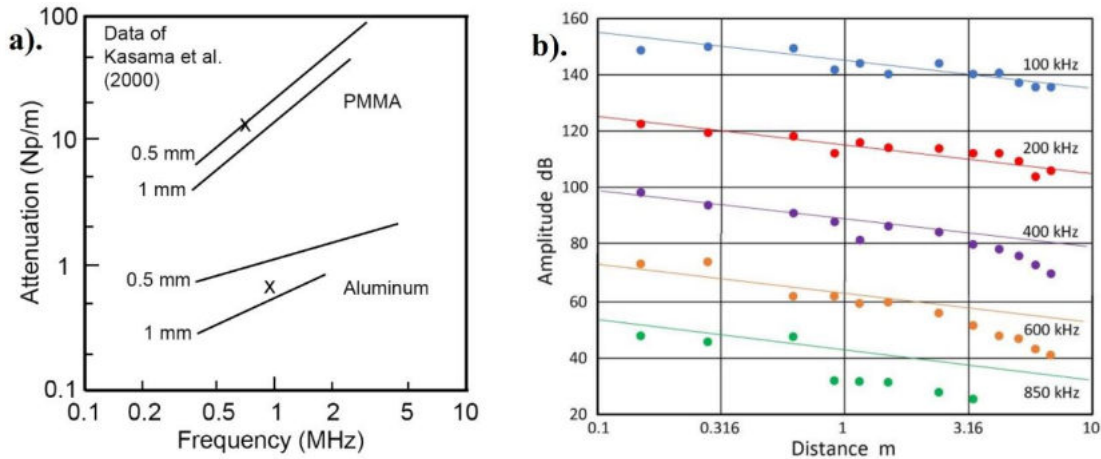


Fig. 3.1.4-1: a). Lamb wave attenuation as a function of frequency for aluminum and PMMA plate [Kasama et al. (2000), Ono and Gallego (2012)] and b). Decrease of Lamb wave amplitude of different frequencies in a steel plate as a function of propagation distance [Ono (2018)].

[Zhao et al. (2007)] gave another example of Lamb wave attenuation in an aerospace structure, where a transducer T is placed between rivet holes as depicted in Fig. 3.1.4-2a (Case 1). Given that the actuator T was excited by using a 1.8 MHz excitation frequency, Fig. 3.1.4-2b illustrates the captured S_0 -mode Lamb wave signal from a series of sensors X that are located 20 – 200 mm away from the actuator T. In this case, they calculated that the average attenuation rate was 0.044 dB/mm. In case 2, they placed a sensor series Δ across the stiffeners, and using the same frequency and S_0 -mode excitation obtained an average attenuation of 15 dB per rivet row. The distance between the rivet rows was 6.5 cm, meaning that the average attenuation was increased to 0.231 dB/mm. This calculation already included multiple scattering from the rivets.

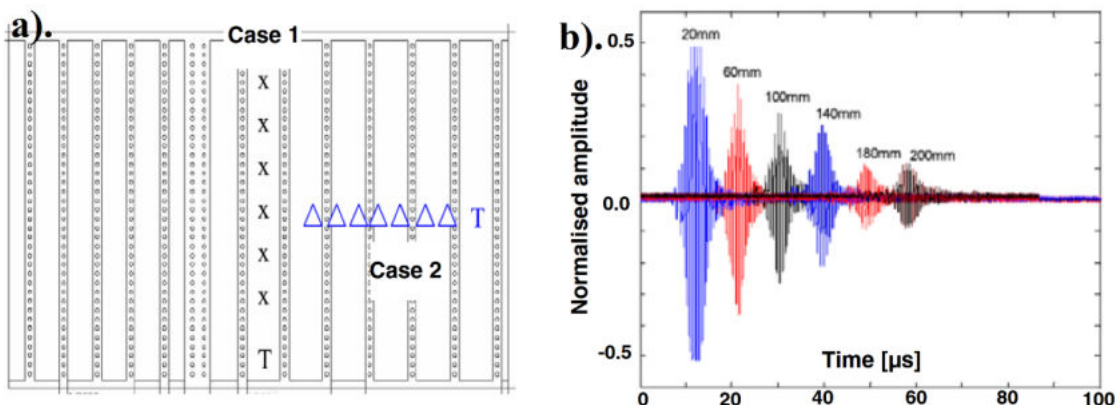


Fig. 3.1.4-2: a). Sketch of distribution of rivets and transducers in wing section ('T': actuator; 'X': sensor array in Case 1; 'Δ': sensor in Case 2); and b). integrated Lamb wave signals captured by a series of sensors in a straight line (Case 1) [Zhao et al. (2007)].

3.2. Signal Representation and Data Processing

This section describes the different ways to represent the captured Lamb wave signal and the commonly used features to calculate the damage index that can be related to predict the physical damage state. The general formulation of SHM diagnostic can be categorized into quintuple $D: \{\pi, \psi, \tau, \lambda, \omega\}$ where:

1. π is the actor domain and contains the parameters that are needed to generate and measure the physical phenomenon of interest λ ,
2. ψ is the medium domain where the phenomenon of interest λ reigns,
3. τ is the transitional domain, which separates the actor from the medium,
4. λ is the phenomenon of interest, that is the physical phenomenon, which is observed in actor domain π ,
5. ω is the environmental domain which covers π, ψ , and τ , but are separated from them

We consider here a typical Lamb wave experimental setup such as in Fig. 3.2-1 [Wang et al. (2019)], where actor π consists of the computer, amplifier, controller, and PZT patch, medium ψ consists of the aluminum plate including any inhomogeneities inside such as fasteners, rivet holes, welded regions, cracks, or corrosion spots, transitional tuple τ consists of the glue layer between the plate and the PZT, λ is the propagating Lamb wave at a certain time, and ω includes the environmental factors such as temperature, humidity, or external vibration that might affect or interfere the behavior of the Lamb wave propagation.

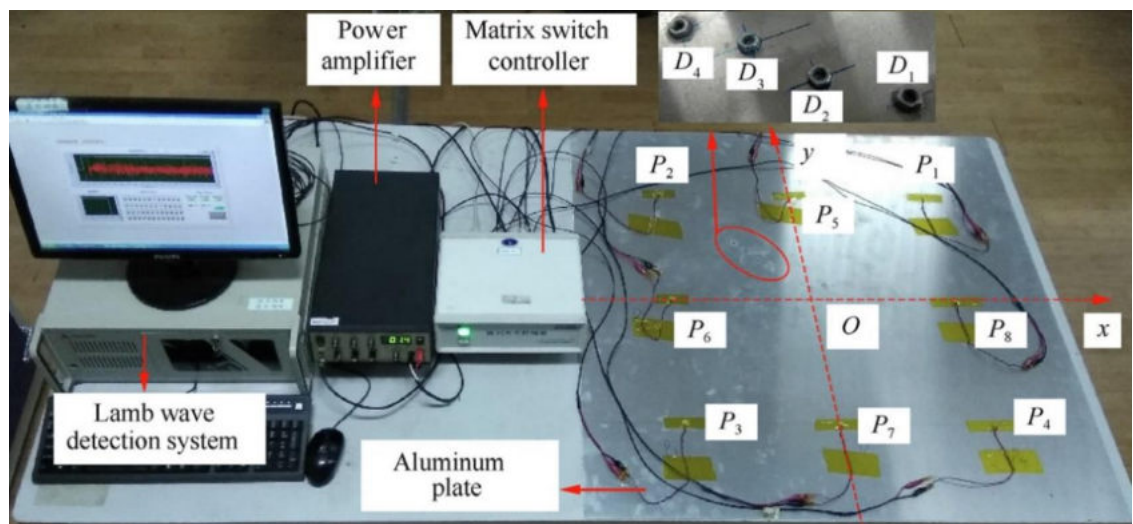


Fig. 3.2-1: Typical Example of Lamb Wave Experimental Setup [Wang et al. (2019)] where P are the sensing locations and D are the simulated defect locations.

Obviously, the Lamb wave propagation cannot be seen with naked eye, and in fact, in many real-world situations where data is only recorded offline, we can only observe the signal shown in the computer. Human perception on what occurs in the medium domain ψ during the wave propagation λ is based on signal X that is recorded by the actor tuple π . When accounting for the environmental factor ω , human perception can clearly be distorted if the signal is distorted. The most direct formulation of Lamb wave propagation is:

$$\lambda = f(\boldsymbol{\pi}, \boldsymbol{\psi}, \boldsymbol{\tau}, \boldsymbol{\omega}) \leftrightarrow \boldsymbol{\pi}, \boldsymbol{\psi}, \boldsymbol{\tau}, \boldsymbol{\omega} = f^{-1}(\lambda) \quad (3.2-1)$$

In reality, the occurrence of Lamb wave propagation λ is rather inferred through the observation of signal X at time t in the oscilloscope or computer that is contained by the tuple $\boldsymbol{\pi}$, so the formulation becomes:

$$\hat{f}(\boldsymbol{\pi}, \boldsymbol{\psi}, \boldsymbol{\tau}, \boldsymbol{\omega}) = X_{\lambda}(t) \quad (3.2-2)$$

Assuming that the behavior between actor, environmental, and transition tuple is consistent during the observation time, the relationship between inhomogeneities i in the medium $\boldsymbol{\psi}$ can be simplified as:

$$\hat{f}(\boldsymbol{\psi}_i) \simeq X_{\lambda}(t) \equiv X(t) \quad (3.2-3)$$

3.2.1. Signal Representation

An example of Lamb wave signal in the time domain $X_{\lambda}(t)$ from Eq. (3.2-3) at 100 and 200 kHz excitation frequency in an aluminum plate is depicted in Fig. 3.2.1-1a-b [Malaeb et al. (2018)], respectively.

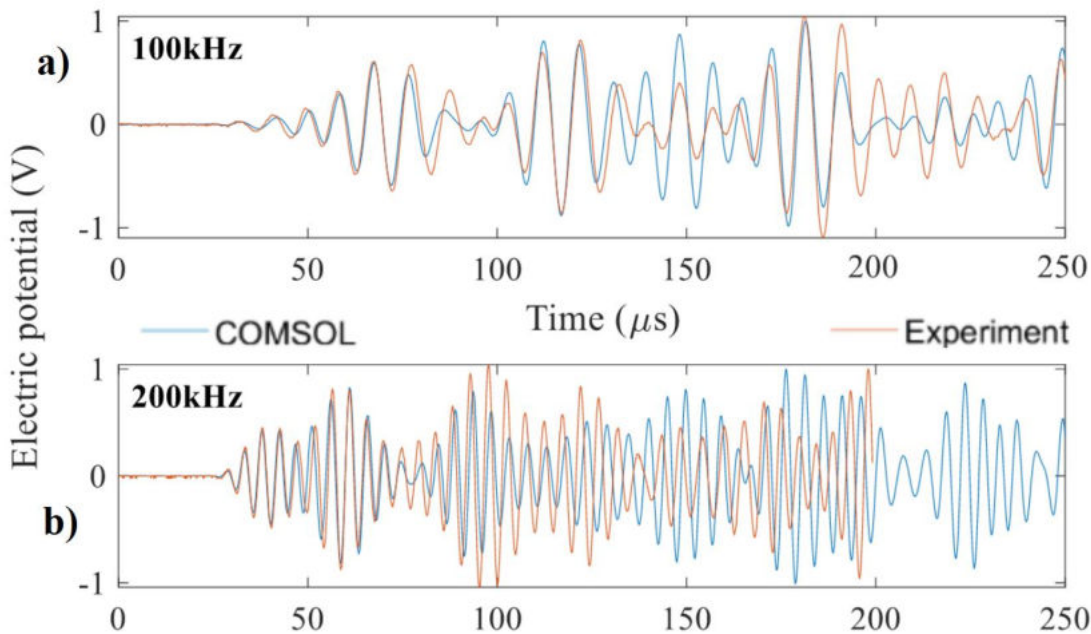


Fig. 3.2.1-1: Example of Lamb wave signal at a). 100 kHz and b). 200 kHz excitation frequency in aluminum plate [Malaeb et al. (2018)].

Such a signal representation as shown in Fig.3.2.1-1 is called a time-series as the amplitude data points are ordered in time order. A second way to represent the signal is to look at its frequency spectrum and to convert the time-series into frequency-domain spectrum. A fast Fourier transform (FFT) is commonly used to convert time-discrete signal into a frequency spectrum. Analogously, the frequency-domain spectrum can be re-converted into a time—series by using an inverse FFT (IFFT). The general equation of Fourier transformation in continuous

time t and its inverse in continuous frequency f are given in Eq. (3.2.1-1) and (3.2.1-2), respectively:

$\tilde{X}(f) = \int_{-\infty}^{\infty} X(t) \cdot \exp(-i2\pi ft) dt$	(3.2.1-1)
$X(t) = \int_{-\infty}^{\infty} \tilde{X}(f) \cdot \exp(-i2\pi ft) df$	(3.2.1-2)

Accordingly, the discretized Fourier transform and its inverse in discrete time t_n and frequency f_n can be described as:

$\tilde{X}(f) \approx \tilde{X}(f_n) = \sum_{n=0}^{N-1} X(t_n) \cdot \exp\left(\frac{-i2\pi ft}{N}\right)$	(3.2.1-3)
$X(t) \approx X(t_n) = \sum_{n=0}^{N-1} \tilde{X}(f_n) \cdot \exp\left(\frac{-i2\pi ft}{N}\right)$	(3.2.1-4)

An example of the time and frequency representation of a Lamb wave signal measured by scanning laser Doppler vibrometer (SLDV) is given in Fig. 3.2.1-2 [Tian and Yu (2014)].

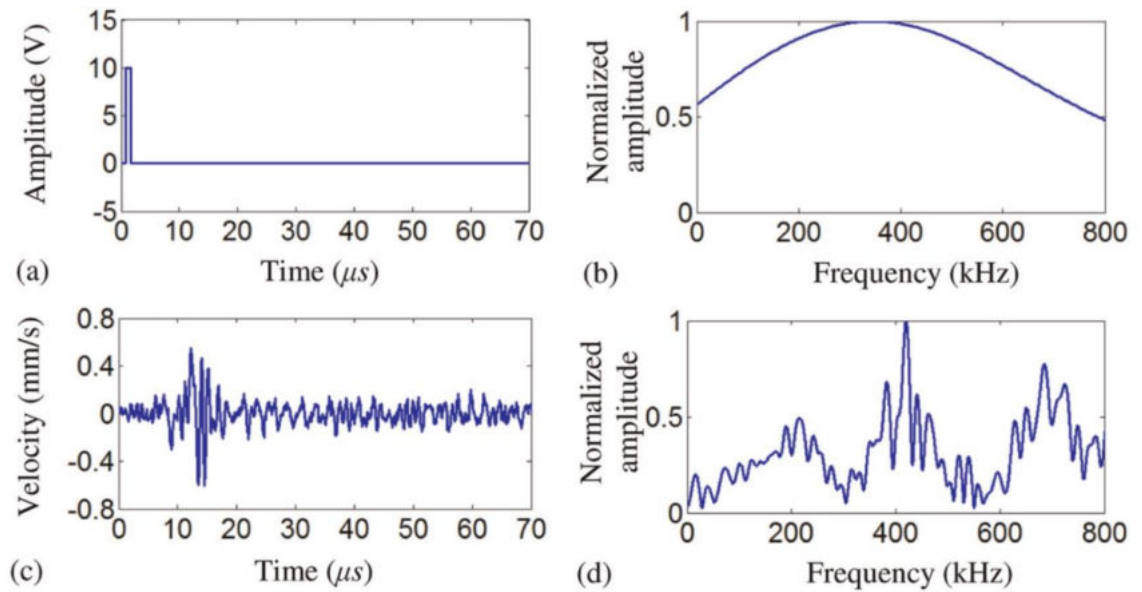


Fig. 3.2.1-2: Time-domain Lamb wave signal from a). excitation PZT and c). SLDV scanning point located on the plate. Their frequency representation is depicted in b) and d), respectively [Tian and Yu (2014)]. More detail on SLDV technique can be read in [Tian et al. (2019), Schmitt et al. (2013), Pohl and Mook (2013)].

Both the time-series and its frequency representation can be joined to create a Time-Frequency Representation (TFR), sometimes known as Time-Frequency Distribution (TFD). There are several ways to achieve a TFR [Krammer and Jones (1994), Hlawatsch (1998), Legendre et al. (2000), Debnath (2002), Shin and Song (2000), Niethammer et al. (2000), Zemmour (2006), Kehtarnavaz (2008), Li et al. (2009), Harley et al. (2015), Zhang et al. (2015), Boashash (2016), Zoubi et al. (2019)]:

1. Short-Time Fourier Transform (STFT)
2. Wavelet Transform (WT)

3. (Smoothed-Pseudo) Wigner Ville Distribution (SPWVD)
4. Hilbert-Huang Transform (HHT)

STFT is probably the simplest way to represent the TFR. While the FFT does not provide any information about the change in the frequency spectrum over time, the STFT is also suitable for signals in which the frequency changes over time. With the STFT, a window function is applied to the time series, i.e., the window is shifted at each point in time and to the frequency to be considered and thus the absolute duration and bandwidth of the window remain constant. The resolution in the time and frequency domains are therefore dependent on the window size. The continuous form of the STFT coefficient for window length w at time t and its discretized version for a discrete time-step $n \in \mathbb{Z}$ are:

$$\text{STFT}_{X(t)}(t, f) = \int_{-\infty}^{\infty} \underbrace{X(t) \cdot \exp(-i2\pi ft)}_{\text{Fourier Transform}} \cdot \underbrace{w(t - \tau)}_{\text{Window function}} dt \quad (3.2.1-5)$$

$$\text{STFT}_{X(t)}(t_n, f) = \int_{-\infty}^{\infty} \underbrace{X(t_n) \cdot \exp(-i2\pi ft)}_{\text{Fourier Transform}} \cdot \underbrace{w(t - \tau)}_{\text{Window function}} dt \quad (3.2.1-6)$$

Due to the time-frequency uncertainty, the resolution in the time domain is inversely proportional to the resolution in the frequency domain and therefore it is not possible to achieve the best possible resolution in the time domain and in the frequency domain at the same time (“No Free Lunch” principle). The STFT coefficients can be represented as spectrograms as depicted in Fig. 3.2.1-3.

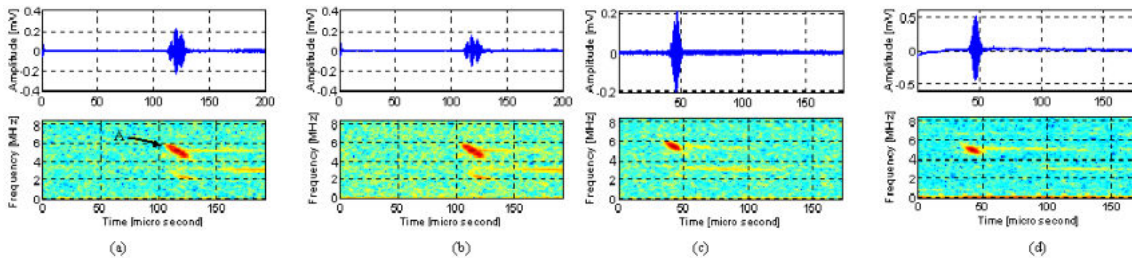


Fig. 3.2.1-3: Spectrogram of Lamb wave signals in pitch-catch configuration for (a) no defect, (b) 5%, (c) 35%, and (d) 80% through wall of electrical discharge machined (EDM) notches in aluminum plate with 1 mm thickness [Shin and Song (2000)].

To overcome the time-frequency trade-off, the wavelet transformation is used. The term wavelet describes the basic function used for the transformation. Like the STFT, a window function is applied to the time series signal. The window function, called the ‘mother wavelet’ must be selected according to the general representative shape of the time series waveform. However, instead of moving and modulating the window, the mother wavelet is shifted and scaled.

As with the STFT, the scaling also results in a frequency shift, but at the same time as the frequency increases, the duration of the window is reduced. A WT can be regarded as a flexible size TFR pixel. This results in a more detailed temporal resolution at higher frequencies and a lower time resolution at lower frequencies. The continuous form of the wavelet transforms coefficient WT for a mother

wavelet function Φ with shifting factor b and scaling factor a at time t and its discretized version for a discrete time-step $n \in \mathbb{Z}$ is:

$$\text{WT}_{X(t)}(t, f) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} X(t) \cdot \underbrace{\Phi\left(\frac{t-b}{a}\right)}_{\text{Continuous mother wavelet}} dt \quad (3.2.1-7)$$

$$\text{WT}_{X(t)}(t_n, f) = \frac{1}{\sqrt{a_n}} \sum_{n=-\infty}^{\infty} X(t_n) \cdot \underbrace{\Phi\left(\frac{t_n - b_n}{a_n}\right)}_{\text{Discretized mother wavelet}} \quad (3.2.1-8)$$

An example of a TFR of Lamb wave signal by wavelet transform is depicted in Fig. 3.2.1-4 [Pramila et al. (2007)] where 3.2.1-4a depicts the signal in the time domain while where 3.2.1-4b represents the TFR of 3.2.1.4a.

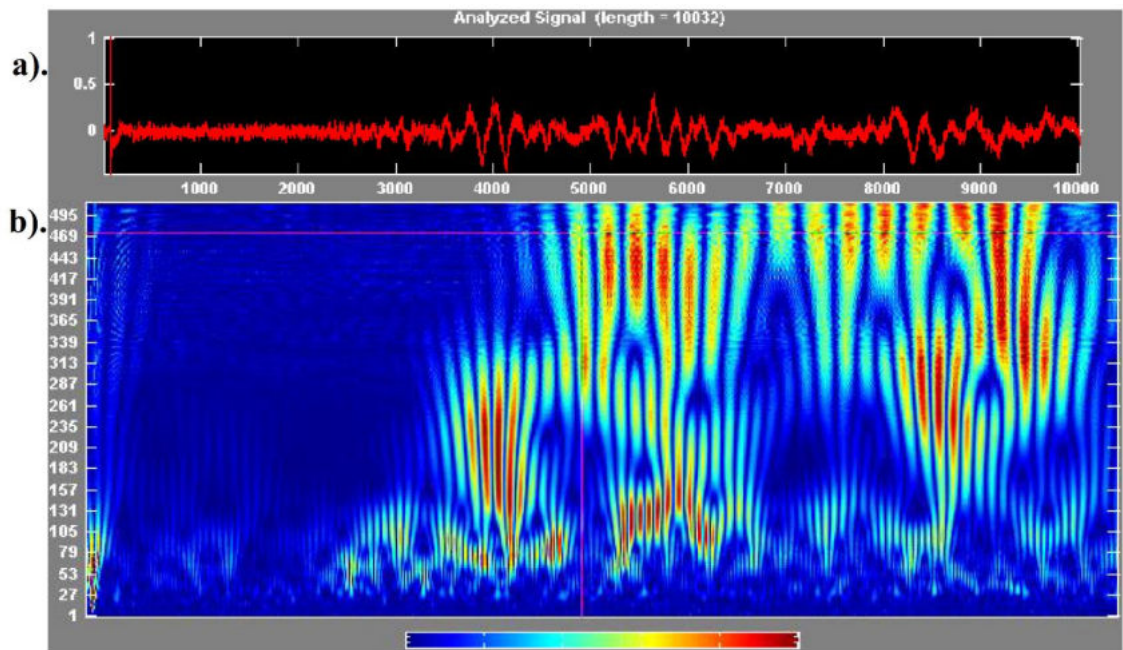


Fig. 3.2.1-4: a). Representation of Lamb wave signal in time domain and b). its representation in the time-frequency domain by using a wavelet transform [Pramila et al. (2007)].

The Wigner-Ville distribution (WVD) was introduced by Eugene Wigner in 1932 in quantum physics to introducing quantum corrections to statistical physics with the objective of replacing the wave function in the Schrödinger equation by a probability density in phase space. The shared algebraic structure between time-frequency in signal processing and position-momentum in quantum physics made it possible to adapt WVD techniques for TFR analysis.

Due to its origin in quantum mechanics, WVD can offer a very high resolution and thus detailed quasi-continuous TFR analysis. For calculating a WVD, first the non-stationary autocorrelated signal $X^c(t)$ must be defined as given in Eq. (3.2.1-7), where the bar signifies the complex conjugate. Then the Wigner distribution function for a given window length τ is given by Eq. (3.2.1-8). For a zero-mean time series (i.e., where $\mu(t) = 0$), the Wigner function can be written as Eq. (3.2.1-9).

$$X^c(t) = \left\langle \left(X(t_1) - \mu(t_1) \right) \cdot \overline{\left(X(t_2) - \mu(t_2) \right)} \right\rangle \quad (3.2.1-7)$$

$$\text{WVD}_{X(t)}(t, f) = \int_{-\infty}^{\infty} X^c\left(t + \frac{\tau}{2}, t - \frac{\tau}{2}\right) \cdot \exp(-i2\pi f\tau) d\tau \quad (3.2.1-8)$$

$$\text{WVD}_{X(t)}(t, f) = \int_{-\infty}^{\infty} X\left(t + \frac{\tau}{2}\right) \cdot \overline{X\left(t - \frac{\tau}{2}\right)} \cdot \exp(-i2\pi f\tau) d\tau \quad (3.2.1-9)$$

As per the “No Free Lunch” principle however, the WVD comes with the trade-off that it is computationally expensive. This matter becomes significant when the WVD is applied to a broadband signal and therefore makes it impractical in many signal processing applications. Several methods have been proposed to reduce this effort, and the most known one is called the smoothed-pseudo WVD (SPWVD) [Li and Liu (2008)]. An example of a SPWVD spectrogram of a Lamb wave signal is depicted in Fig. 3.2.1-5.

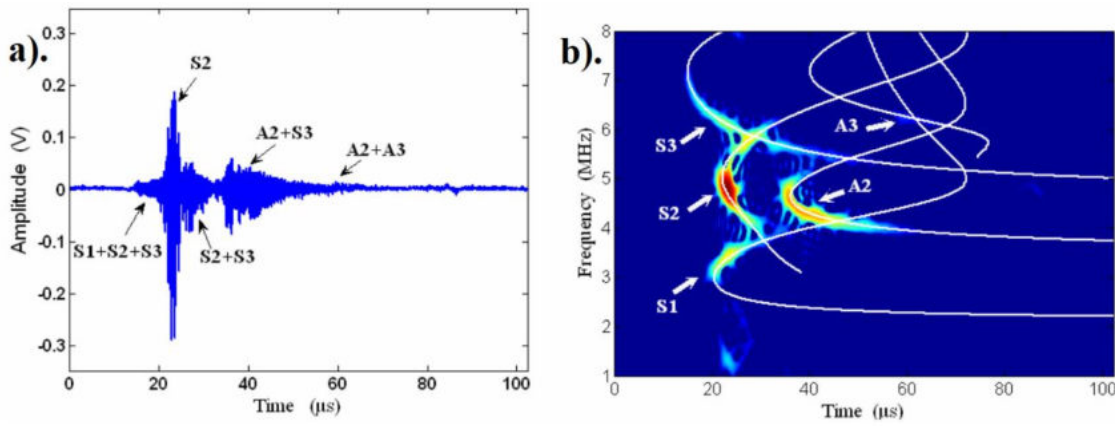


Fig. 3.2.1-5: a). Time-domain Lamb wave signal containing overlapping higher multiple modes and b). its TFR by using SPWVD [Li and Liu (2008)].

The SPWVD is given by Eq. (3.2.1-10), where q is a lowpass function for a time-shift u and PW is the pseudo-Wigner-Ville distribution which is given by Eq. (3.2.1-11), where w is the selected window function.

$$\text{SPWVD}_{X(t)}(t, f) = \int_{-\infty}^{\infty} \left[q(t - u) \cdot \text{PW}_{X(t)}(f) \right] (t) du \quad (3.2.1-10)$$

$$\text{PW}_{X(t)}(t, f) = \int_{-\infty}^{\infty} w\left(\frac{\tau}{2}\right) \cdot \overline{w\left(-\frac{\tau}{2}\right)} \cdot X\left(t + \frac{\tau}{2}\right) \cdot \overline{X\left(t - \frac{\tau}{2}\right)} \cdot \exp(-i2\pi f\tau) d\tau \quad (3.2.1-11)$$

3.2.2. Feature Extraction and Damage Index

The multitude of signal representations either in time domain, frequency domain, or time-frequency domain can be analyzed in order to extract the relevant features that might be useful to interpret the physical condition of the relevant structure. According to [Su and Ye (2009)], feature extraction is “*the process of identifying and picking up the damage-modulated properties and parameters in a signal which are called the features or characteristics of the signal*”. In this aspect,

various approaches have been developed to define and extract the features to calculate the damage index (DI) that gives an indication of the presence of damage. There are several DIs that can be linked to physical damage [Michaels and Michaels (2007), Konstantinidis et al. (2006), Betz et al. (2006), Rizzo and DiScalea (2006)], for instance the energy distribution from an extracted signal envelope with a Hilbert Transform (HT), given in Eq. (3.2.1-12), the peak-to-peak amplitude, residual mean squared deviation (RMSD) from the baseline extraction, given in Eq. (3.2.1-12) and signal variance, given in Eq. (3.2.1-13). There are two variants of energy-based DI. Both are given in Eq. (3.2.1-13). The signal correlation-based DI is given in Eq. (3.2.1-15). In these equations, $X(t)$ is the signal function in the continuous domain, x_i is the signal from a baseline structure at discrete time i , τ is the window length and μ is the signal average, respectively. The subscript tilde denotes a signal from a damaged structure.

$\text{HT}_{X(t)} = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{X(t)}{(t - \tau)} d\tau$	(3.2.1-12)
$\text{DI}_{\text{RMSD}} = \sqrt{\frac{\sum_{i=1}^N (\tilde{x}_i - x_i)^2}{\sum_{i=1}^N x_i^2}}$	(3.2.1-13)
$\text{DI}_{\text{Energy}(1)} = \left(\frac{\int_{t_0}^{t_1} \tilde{X}(t) ^2 - X(t) ^2 dt}{\int_{t_0}^{t_1} X(t) ^2 dt} \right), \quad \text{DI}_{\text{Energy}(2)} = \left(\frac{\int_{t_0}^{t_1} \tilde{X}(t) ^2 dt}{\int_{t_0}^{t_1} X(t) ^2 dt} \right)$	(3.2.1-14)
$\text{DI}_{\text{Signal_Corr}} = 1 - \frac{\sum_{i=1}^N (x_i - \mu_x)(\tilde{x}_i - \mu_{\tilde{x}})}{\sqrt{\sum_{i=1}^N (x_i - \mu_x)^2} \cdot \sqrt{\sum_{i=1}^N (\tilde{x}_i - \mu_{\tilde{x}})^2}}$	(3.2.1-15)

Another way to extract features is to build a damage fingerprinting specification from the raw signal in either time-domain, frequency-domain, or time-frequency domain. The reason to do so is because in many cases, a raw signal contains many non-characteristic points. [Su and Ye (2009)]. They mentioned that “*taking into account the most frequently occurring damage cases, a damage parameter database (DPD) hosting all DDF can be constructed for a particular type of damage in the structure under inspection, e.g., a crack or delamination*”. For a Lamb wave SHM system consisting of a certain number of actuators and sensors, a series of feature vector pairs, termed as digital damage fingerprints (DDF) can be established for all the signals captured by the sensor network, and depicted as a flowchart in Fig. 3.2.2-2 [Su and Ye (2009)].

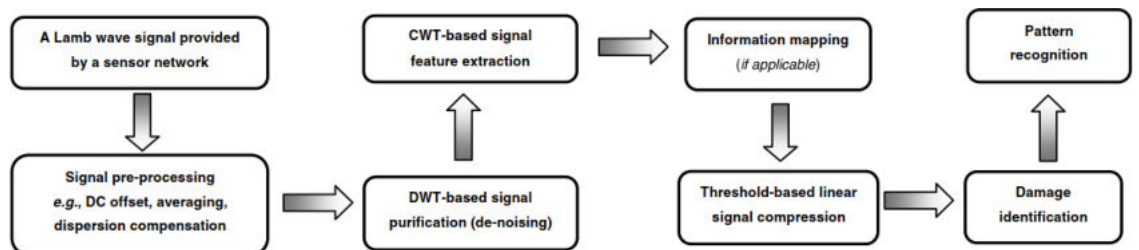


Fig. 3.2.2-2: The flowchart of DDF-based signal processing for damage identification in Lamb wave based SHM [Su and Ye (2009)].

3.3. Optimization and Search Metaheuristics

Lamb wave mode conversion and scattering occur all the time and this not only leads to the fact that the signal interpretation becomes aggravated, but also that the ToF measurement of one specific wave mode will be more difficult. To suppress wave mode conversion and subsequent mode overlapping, an actuator should be put far from any edge, holes, or structural joints. Consider a mounted sensor network on a structure, a time-harmonic interfacial shear stress [Xu and Giurgiutiu (2007)] occurs during wave propagation as depicted in Fig. 3.3-1a-b.

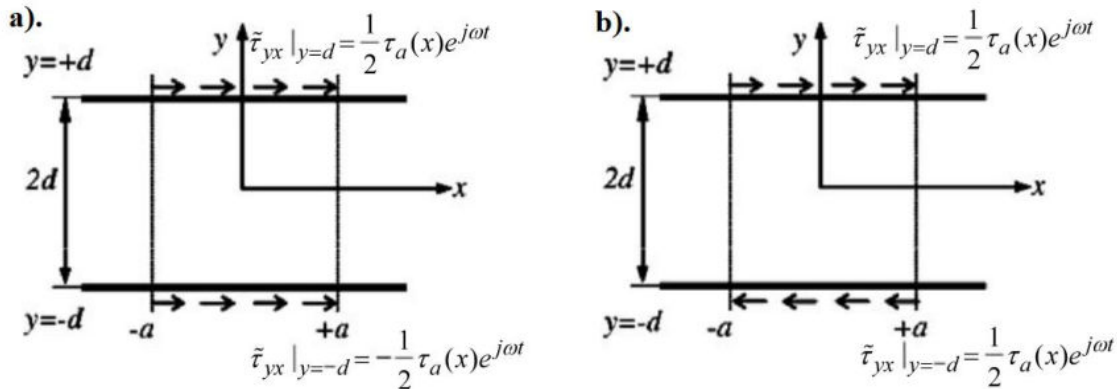


Fig. 3.3-1: Load on a plate due to the PZT actuation for a) Symmetric and b) Antisymmetric Lamb modes [Xu and Giurgiutiu (2007)]. The plate thickness is $2d$, and the PZT diameter is $2a$. In this figure, τ denotes the interfacial shear stress.

The amount of shear stress depends on the sensor diameter and the loading condition. For more sensors and the use of sensors with larger radius, the larger will be the influence of the interfacial shear stress. Thus, one should aim to use the minimum number of sensors possible. For a pulse-echo configuration, a single PZT transducer is sufficient, whereas for the pitch-catch configuration the minimum number of PZT transducers is 2. The larger structure the structure is, the more transducers would be needed to cover the entire area of inspection.

A larger number of sensors collects more data and when combined with appropriate signal processing techniques might decrease the probability of damage misdetection, but at the same time this could be uneconomical and provide redundant data. Thus, a compromise must be made between the number of PZTs to be installed and the sensor network performance.

Briefly speaking, optimization is a branch of mathematics seeking to model, analyze and solve analytically or numerically problems which consist of minimizing or maximizing a function for a certain input set [Nocedal and Wright (2006)]. The quality of the results depends on 1). the basic assumptions of the model, 2). the choice of the variables that one seeks to optimize, 3). the efficiency of the algorithm and 4). the computing power available. When speaking about mathematical optimization, there are two general streams which depend on the domain range: continuous optimization and discrete optimization, which will be described in sections 3.3.1 and 3.3.2, respectively.

3.3.1. Fundamentals on Continuous Optimization

In contrast to discrete optimization, in which the variables can take either Boolean or integer values, in continuous optimization, the variables in the model are allowed to take any value within a range of values, which are usually real numbers, although they can also be complex numbers. The formal definition of finite dimensional continuous optimization problems for searching minima can be written as:

Definition 3.3.1-1. [Stein (2016), Nocedal and Wright (2006)]

$$\min_{x \in \mathbb{R}^n, n \in \mathbb{N}} f(x) \quad \text{subject to} \quad \begin{cases} g_i(x) \leq 0 & , \quad i \in I \\ h_j(x) = 0 & , \quad j \in J \end{cases}$$

with (at least partially differentiable or convex) objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$, inequality constraint functions $g_i : \mathbb{R}^n \rightarrow \mathbb{R}^1, i \in I$ with a finite set index I , and equality constraint functions $h_j : \mathbb{R}^n \rightarrow \mathbb{R}^1, j \in J$ with a finite set index J

Define the feasible set Ω to be the set of points x that satisfy the constraints:

$$\Omega = \left\{ \begin{array}{l} g_i(x) \leq 0 \quad , \quad i \in I \\ h_j(x) = 0 \quad , \quad j \in J \end{array} \right\}$$

Then, the objective function can simply be rewritten as:

$$\min_{x \in \Omega} f(x)$$

Subsequently, to search for the function maxima, the objective function $f(x)$ can simply be multiplied by -1. The solutions of the objective function $f(x)$ are either a global or local extremum. The formal definition of global and local minima is given in Def. 3.3.1-2.

Definition 3.3.1-2: Global and Local Minima [Nocedal and Wright (2006)]

A point $x^* \in \Omega$ is a **global minimum** if $\forall_{x \in \mathbb{R}^n}$:

$$f(x^*) \leq f(x)$$

A point $x^* \in \Omega$ is a **local minimum** if $\forall_{x \in \mathcal{N}}$:

$$\exists \text{ neighborhood } \mathcal{N} \cap \Omega \text{ of } x^* : f(x^*) \leq f(x)$$

A point $x^* \in \Omega$ is a **strict local minimum** if $\forall_{x \in \mathcal{N}, x \neq x^*}$:

$$\exists \text{ neighborhood } \mathcal{N} \cap \Omega \text{ of } x^* : f(x^*) < f(x)$$

A point $x^* \in \Omega$ is a **isolated local minimum** if:

$$\exists \text{ neighborhood } \mathcal{N} \cap \Omega \text{ of } x^* : x^* \text{ is the only local minimum in } \mathcal{N} \cap \Omega$$

In the same way, the global and local maxima can be simply re-defined by multiplying the objective function $f(x)$ by -1. For convenience, the subsequent definitions will be limited to minima and any subsequent objective functions will be formulated as minimization problems. The best solution to any optimization problem is found when we reach a global minimum. But in many situations, this

is not feasible since many algorithms cannot practically visit many points. Depending on the functions, many algorithms are trapped in the manifold of local minima. The only way to be sure that the local minimum is a global minimum is to visit all possible points in the search space, making any algorithm to run this search in either take an indefinite time or space and it will be as worthless as a random generator, i.e., a blind search. Let revisit the No-Free-Lunch theorem on search and optimization proposed by [Wolpert (1997)]:

Theorem 3.3.1-1: No Free Lunch in Optimization [Wolpert and Macready (1997)]

For any pair of algorithms A_1 and A_2 :

$$\sum_f P(d_m^y | f, m, A_1) = \sum_f P(d_m^y | f, m, A_2)$$

where d_m^y denotes the ordered set of size m of the cost values $y \in Y$ associated to input values $x \in X$ and $f : X \rightarrow Y$ is the function being optimized.

$P(d_m^y | f, m, A_i)$ denotes the conditional probability of obtaining a given sequence of cost values from algorithm A_i run m times on function f .

The consequence of the No-Free-Lunch Theorem can be interpreted in following ways [Ho and Pepyne (2002), English (2000)]:

1. A general-purpose almost-universal optimizer can theoretically exist. That does not mean however, that it is necessarily practical.
2. An algorithm may outperform another on a problem when neither is specialized to the problem.
3. For almost all objective functions, specialization is essentially accidental.

Examining all possible points in the neighborhood of point x^* to make sure that x^* is local minimum seems to be impractical as the function complexity can still trap the algorithm. However, there is a more efficient way to identify local minima if f is smooth, especially when f is twice continuously differentiable, we may predict whether x^* is a local minimum only by examining the gradient $\nabla f(x^*)$ and the Hessian $\nabla^2 f(x^*)$. In essence, the study of minimization [Apostol (1974), Nocedal and Wright (2006), Bartle and Sherbert (2011)] is derived from Taylor's theorem. In any case, all algorithms seek a point where the gradient ∇f vanishes. The most direct process to find the local minima is the method of steepest gradient, sometimes also called gradient descent:

$$x_{i+1} = x_i - \eta_i \nabla f(x_i) \quad , \quad i \in \mathbb{Z}^+ \quad (3.3.1-1)$$

In Eq. (3.3.1-1), η is the gradient step. Assuming that f is continuously differentiable, x^* converges towards the local minimum. The adaptive gradient step size η at step i is defined as:

$$\eta_i = \frac{|(x_i - x_{i-1})^T [\nabla f(x_i) - \nabla f(x_{i-1})]|}{\|\nabla f(x_i) - \nabla f(x_{i-1})\|^2} \quad (3.3.1-2)$$

Gradient descent is the most used methods to find minima in many problems, including in operation research, schedule optimization, machine learning, and many more. Especially for many machine learning problems where the search space is often large, some stochasticity is often involved, in which the most common algorithm is called stochastic gradient descent (SGD) [Hoseini et al. (2019)], as given in Algorithm 3.3.1-1.

Algorithm 3.3.1-1. Stochastic Gradient Descent.

Setup initial values:

$$x \leftarrow x_0, \quad \eta \leftarrow \eta_0$$

Calculate $f(x_0)$

Until termination condition is reached:

Randomly shuffle x_i

For $i \in \mathbb{Z}^+$ until $i = M < \infty$, do:

$$x_{i+1} \triangleq x_i - \eta_i \nabla f(x_i)$$

Calculate $f(x_i)$

If $f(x_i) < f(x_{i-1})$: $f(x) \leftarrow f(x_i)$

3.3.2. Search Metaheuristics in Discrete Optimization

Unlike continuous optimization where the variables are usually real numbers, in discrete optimization the input variables are restricted to only take discrete numbers, i.e., integers. If the objective function is linear, it is possible to solve the problem with linear programming. However, some real-world problems such as sensor placement has a non-linear optimization function – albeit simplification through integer variables and boundaries. Assume a section of fuselage panel with a size of 1 m² that can be discretized into a 100 cm x 100 cm grid with a resolution of 1 cm² for each grid pixel, then we have 10.000 possible placements for each transducer involved. For N transducers and L pixels, we have C combinations of possible placements:

$$C = \frac{L!}{N!(L-N)!} \quad (3.3.1-11)$$

For 2 PZTs in 100 x 100 cm plate, there are 50 million placement combinations possible. For 3 transducers, the number of possible combinations jump to 167 billion. It would be absurdly expensive to test all these combinations to determine which sensor network topology will minimize the missed detection rate, which alternatively can be rephrased as maximization of the probability of detection (PoD). For this reason, a smarter and more efficient way to find feasible local minima is necessary – this general framework is called (meta)-heuristics. Heuristics are a technique designed for solving a problem more quickly when classic a method such as Newtonian or gradient based algorithms are too slow, or for finding an approximate solution when classic methods fail to find any exact solution, while metaheuristic can be regarded as an expansion of this approach

such that the heuristic algorithms can be controlled and fine-tuned. According to [Sörensen and Glover (2013), Sörensen et al. (2018)], “a metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms”. In no way should the definition be abused so that one might think metaheuristics can find the most optimum solution, because it never guarantees that a global minimum will be found, but rather that it can guarantee that the solution(s) found are part of the set of possible optima within the landscape of the objective function. Chapter 2 mentioned some of the common metaheuristics: genetic algorithm (GA), simulated annealing (SA), and particle swarm optimization (PSO) which includes the ant- and bee-colony optimization. Before going deeper into metaheuristics, let us recall a very standard problem-solving heuristic: the greedy approach. An example of the greedy method is given in Algorithm 3.3.2-1. An algorithm is said greedy when it satisfies the following two conditions [Charlier (1995)]:

1. The algorithm creates the solution in an incremental way.
2. At each step of the sequence, the local minimum is selected.

Algorithm 3.3.2-1. Greedy Method for Minimization [Martins and Ribeiro (2006)].

Recall the minimization problem for the objective function $f : 2^G \rightarrow \mathbb{R}$ with

$$\min f(S) \text{ subject to } S \in X$$

Where $G = \{e_1, \dots, e_n\}$ is the ground set and $X \subseteq 2^G$ is a set of feasible solutions

We are looking for an optimal solution $S^* \in X$ such that $f(S^*) \leq f(S), \forall S \in X$

The greedy method is given as follows in pseudocode:

Setup initial values:

$$S \leftarrow \emptyset$$

Evaluate the incremental cost $\forall e \in G$

Until termination condition is reached:

Select the element $s \in S$ with the smallest incremental cost

$$S \leftarrow S \cup \{s\}$$

Update the incremental cost $\forall e \in G \setminus S$

Obviously, the greedy method is not always the best method. This not to say that for some problems, the optimal solution can be found with the greedy method, e.g., for determining the minimum number of coins when giving change. Assume that 0.86 € is given as a return from the following € coin denominations: {1¢, 2¢, 5¢, 10¢, 20¢, 50¢, 1€, 2€}.

Then, the minimum number of coins can then be broken down as 1x 50¢, 1x 20¢, 1x 10¢, 1x 5¢, 1x 1¢ which sums up to 5 coins. Now assume with hypothetical € coin denominations: {2¢, 8¢, 10¢, 40¢, 60¢, 90¢, 1.50€} where 0.86€ is given as a return. Following the sorting greedy method, the given output would be: 1x 60¢, 2x 10¢, and 3x 2¢ which sums up to 6 coins. Obviously, it misses an alternative output: 2x 40¢ and 3x 2¢, which only sums up to 5 coins.

In mathematics and computer science (CS), when an exhaustive search is infeasible, metaheuristics can be introduced to partially solve the problem. As explained in chapter 2 earlier, there are two metaheuristics mainstreams [Echevarría et al. (2019)] as depicted in Fig. 3.3.2-1:

1. Non-population-based methodologies such as simulated annealing and tabu search
2. Population-based methodologies, which can be divided into evolutionary algorithms and swarm intelligence.

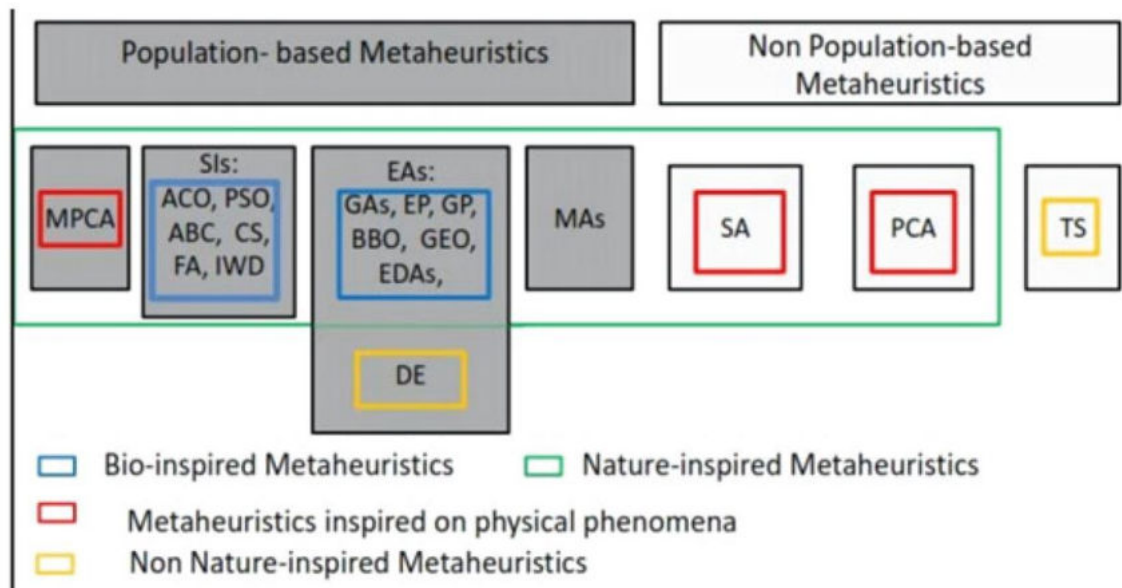


Fig. 3.3.2-1: Classification on Metaheuristics according to [Echevarría et al. (2019)]. MPCA: Multiple Particle Collision Algorithm, SIs: Swarm Intelligence, EAs: Evolutionary Algorithm, DE: Differential Evolution, MAs: Memetic Algorithm, SA: Simulated Annealing, PCA: Particle Collision Algorithm, TS: Tabu Search.

Based on the number of papers in the literature review of chapter 2, we can see that evolutionary algorithms and swarm intelligence are the most commonly used metaheuristics, while simulated annealing has a smaller significance, and the popularity of tabu search has been declining. For this reason, three major search metaheuristics: Genetic algorithm (GA), particle swarm intelligence and simulated annealing (SA), will be subsequently described.

GA is a biologically inspired algorithm from the Darwinian concept of natural evolution [Kramer (2017)]. Usually, a GA contains three main operators: mutation, crossover, and selection [Schmitt (2001)] and typically, the procedure starts with a given initial population that will be assessed against its fitness. Those individuals who have the best fitness are crossed-over to each other and/or a “genetic mutation” is applied e.g., by bit-flipping or replacement. The individuals who do not have the best fitness are not selected. This procedure is repeated several times until a specified certain termination condition is reached. A typical genetic algorithm is described in Algorithm 3.3.2-2.

Algorithm 3.3.2-2. Genetic Algorithm [Zaritsky and Sipper (2004)]Input parameter: S – set of blocks, Output: Superstring of set S

Setup initial values:

Set generation $t \leftarrow 0$ Initialize population P_t to random individuals i from S^* Do procedure **Evaluate_Fitness**(S, P_t)

Until termination condition is reached:

Do {

- Select individuals from P_t
- Procedure **Crossover**($i \in P_t, \hat{i} \in P_t$)
- Procedure **Mutate**($i \in P_t$)
- Procedure **Evaluate_Fitness**(S, \hat{P}_t)
- $P_{t+1} \leftarrow \hat{P}_t$
- $t \leftarrow t + 1$

Definition procedure **Evaluate_Fitness**(S, P_t):For each individual $i \in P$:

Do {

- Generate derived string $s(i)$
- $m \leftarrow$ All blocks from S that are not covered by $s(i)$
- $s^*(i) \leftarrow$ Concatenation of $s(i)$ and m
- $fitness(i) \leftarrow \frac{1}{\|s^*(i)\|^2}$

As defined in algorithm 2, there are 2 basic operations in a genetic algorithm: the mutation and the crossover operator. The mutation operator alters one or more values in the chromosome and its purpose is to preserve and introduce diversity, while crossover is used to combine the genetic information of two parents to generate new children. In practice, there are many other ways to conduct mutations and genetic operations. For a simplified illustration, only the most common methods are shown in Fig. 3.3.2-2.

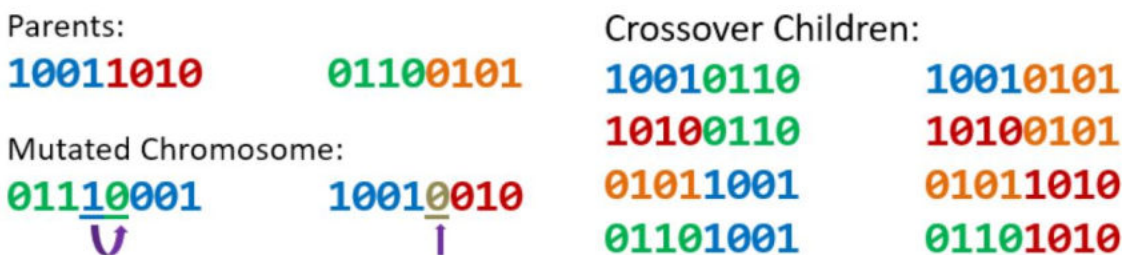


Fig. 3.3.2-2: Basic Operation of Genetic Algorithm: Mutated and Crossover from Parent Generation. [Ewald et al. (2020)]

As an example, the chromosome input value can be encoded as a binary value $\{0, 1\}$. In a single-point crossover, as shown in Fig. 3.3.2-2, the parent chromosome is divided into two sub-genomes (e.g. [1001] and [1010] from parent 1, and [0110] and [0101] from parent 2), and the genome information are permuted in order to derive the crossover children.

To create mutated children, two common methods are normally used:

1. Bit string mutation (e.g. [10011010] from parent 1 into [10010010] as the mutated children 1)
2. Bit-flip mutation (e.g. [01100101] from parent 2 into [01110001] as the mutated children 2).

The mutation operation rate is required to be larger than 0 to avoid being stuck in a local minimum, but at the same time it is kept low in such way that the algorithm does not jump too fast from one optimum to another as both of these conditions make the search unnecessary long.

The second metaheuristic method is particle swarm optimization (PSO) which is a nature-inspired algorithm that looks for a solution to an optimization problem based on biological swarm behavior, i.e., the individuals within a swarm try to stay within the group at all times to follow the group movement while maintaining a certain minimum distance from other individuals [Kennedy and Eberhart (1995), Shi and Eberhart (1998)]. Analogous to the natural phenomenon, in a PSO algorithm, a population of solution candidates is moved through the search space in order to obtain an approximately good solution to the problem. For this purpose, the position of each individual population must be recalculated in each iteration step. An example of PSO is given in Algorithm 3.3.2-3.

Algorithm 3.3.2-3. Particle Swarm Optimization [Raza and Qamar (2016)]

Input: C – set of conditional functions, D – set of decision functions

Output: X_{best} – best solution

Setup initial values $\forall i$:

$X_i \leftarrow$ random Position, $V_i \leftarrow$ random Velocity

$\text{fit} \leftarrow \text{bestFit}(X)$, $\text{global_best} \leftarrow \text{fit}$, $\text{pos_best} \leftarrow \text{bestPos}(X)$, $P_i \leftarrow X_i$

Until termination condition is met for $i \in \mathbb{N}$:

if($\text{fitness}(i) \geq \text{fit}$):

$\text{fit} \leftarrow \text{fitness}(i)$, $\text{pos_best} \leftarrow X_i$

if($\text{fitness}(i) \geq \text{global_best}$):

$\text{global_best} \leftarrow \text{fitness}(i)$

updateVelocity(), updatePosition()

The third metaheuristic method is called SA. The basic idea of SA is to simulate a cooling process that occurs during annealing in metallurgy. After a metal is heated, slow cooling ensures that the atoms have sufficient time to arrange themselves and form stable crystals to achieve a low-energy state which is close to the optimum. Adopting this natural process into a mathematical optimization, temperature T corresponds to a probability in which the intermediate optimization result may also deteriorate and the function to be minimized is the

energy E of the system. However, in contrast to a local search algorithm, SA can escape a local optimum. A simplified simulated annealing algorithm is described in Algorithm 3.3.2-4.

Algorithm 3.3.2-4. Simulated Annealing [Fraga-Gonzalez et al. (2017)]

Input: T_0 – initial temperature, J – cost function, s_0 – initial state,
 S – cooling schedule, N – neighbor state function

Setup initial values:

$$T \leftarrow T_0$$

$$s_k \leftarrow s_0$$

Until termination condition is reached:

$$\text{Do } \left\{ \begin{array}{l} s_{k+1} \leftarrow N(s_k) \\ \Delta E \leftarrow J(s_{k+1}) - J(s_k) \\ \text{if } \min \left(1, \exp \left(-\frac{\Delta E}{T} \right) \right) \geq \mathbf{random}(0,1) : \\ \quad s_k \leftarrow s_{k+1} \\ T \leftarrow S \end{array} \right.$$

3.4. Machine and Deep Learning

3.4.1. Statistical Learning Theory for Supervised Learning

In short, machine learning is a field of study of artificial intelligence which is based on mathematical and statistical approaches to give computers the capacity to learn from data without being explicitly programmed. [Mitchell (1997)] defined the algorithm of machine learning as: “A *computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E* ”. Broadly speaking, the current field of machine learning concerns with the design, analysis, optimization, development and implementation of such algorithms.

In this section, the most important concepts in statistical learning with the focus on supervised learning is highlighted. As previously known, supervised learning is a branch of machine learning in which the learning algorithm tries to find a hypothesis that makes predictions that are as accurate as possible. A hypothesis is to be understood as a mapping function that assigns a presumed output value to each input value. For supervised learning, the basic assumption is that there exists an unknown probability distribution. Now, we must consider how to associate the probability distribution with machine learning.

In general, when considering supervised learning, the following questions naturally arise: which learning problems can be solved efficiently and is it easier to solve some problems rather than others? How many N training samples do we need, and which parameters θ must be fine-tuned during the learning process? In computer science, the proper intuition would be the learnability of the function

itself, commonly known as the probably approximately correct (PAC)-learning framework. More concretely, the underlying assumption for the given assumption stated in Eq. (3.2-3) is that we know $h_\theta: f(X) \rightarrow \Psi$ belongs to the concept class \mathfrak{C} , where h_θ is a hypothesis function that belongs to the hypothesis space H . The PAC learning framework is given in Lemma 3.4.1-1.

Lemma 3.4.1-1. PAC Learning. [Valiant (1984, 2013), Moran and Yehudayoff (2015)]

For any hypothesis $h_\theta \in H_{\mathfrak{C}}$, where $H_{\mathfrak{C}}$ is the hypothesis space in the concept class \mathfrak{C} , the generalization error J is defined as $J_{h_\theta} = P(h_\theta \neq f(X))$. Such a generalization error is impossible to calculate since the hypothesis space is infinitely large. Thus, we can come up with the approximated measurable error over N samples, commonly referred as empirical error $\bar{J}_{h_\theta} = \frac{1}{N} \sum_{n=1}^N P(h_\theta \neq f(X))$.

Assume the existence of a learning algorithm \mathfrak{A} such that for any $\varepsilon > 0$ and $\delta > 0$, \forall distribution κ on input X , the following holds for N training samples:

$$\left\{ N = \frac{1}{\varepsilon} \cdot \ln \frac{|\mathfrak{C}|}{\delta} \right\} : P(J_{h_\theta} \leq \varepsilon) \geq 1 - \delta$$

Then, the concept class \mathfrak{C} is said to be PAC-learnable.

In Lemma 3.4.1-1, h_θ can be thought as a function that can inversely map X into Ψ . We denote the family of this function as generalizer H such that $h_\theta \in H$. As stated above, the assumption is that the universe tends to behave stochastically, thus H does not actually map the observation X into Ψ directly but would rather map X into the joint probability density $P(h_\theta|X)$. The formal definition of a (multidimensional) generalizer [Wolpert (1990, 1992)] is:

Definition 3.4.1-2. Multidimensional Generalizer [Wolpert (1990, 1992)]

An k -dimensional generalizer is a countable infinite set of continuous functions from a subset of $(\mathbb{R}^k \times \mathbb{R} \times \mathbb{R}^k)$ to \mathbb{R} , from a subset of $(\mathbb{R}^k \times \mathbb{R} \times \mathbb{R}^k \times \mathbb{R} \times \mathbb{R}^k)$ to \mathbb{R} , etc. Notationally, a k -dimensional generalizer is a set of continuous functions $L_{\{i\}}$ along with domains of definitions, where $i \in \mathbb{N}$ and $L_{\{i\}}$ being from $\mathbb{R}^{i(k+1)+k}$ to \mathbb{R} .

Generalization is one of the most important concepts in machine learning which can be informally defined as the model's ability to adapt properly to unseen, independently and identically distributed (i.i.d) data drawn from the distribution used to create a machine learning model. The probabilistic nature of many physical phenomenon prohibits us from computing the true underlying risk, hence what we can compute is the number of mistakes on the training data, called the empirical error (sometimes also called as the training error), which has already been explained in Lemma 3.4.1-1.

Normally, we may expect that the empirical risk is relatively small – otherwise, the learning algorithm will not be able to explain the training data. Under this assumption however, we cannot guarantee the training error will be

approximately equal to the rest of the sample set X . We shall say that a classifier generalizes well if the difference between the true error and empirical error in Lemma 3.4.1-1 is small. Note that within this context, a good generalization performance does not necessarily mean that a classifier will have a small overall empirical error, but it rather only means that the empirical error is a good estimate of the true error.

We may never be sure whether the model we created from the training sampled from the distribution is complex enough to represent the distribution or whether is too simple? In this case, we are talking about the bias-variance trade-off. Bias is the error resulting from incorrect assumptions in the learning algorithm. High distortion can cause an algorithm to fail to model the appropriate relationships between input and output, and this is called underfitting. For this reason, bias is also called approximation error. Variance is the error based on the sensitivity to minor fluctuations in the training data. A high variance causes overfitting: instead of modelling the input-output relation, the algorithm is rather memorizing the noise in the data. Mathematically, the total error can be summarized as:

$$\text{Error}(X) = \underbrace{(\underbrace{\mathbb{E}[h_\theta(X)]}_{\text{Predicted}} - \underbrace{f(X)}_{\text{True}})^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}[h_\theta(X) - \underbrace{\mathbb{E}[h_\theta(X)]}_{\text{Avg. Predicted}}]^2}_{\text{Variance}} + \underbrace{\sigma_e^2}_{\text{Irreducible Error}} \quad (3.4.1-1)$$

The total error can be decomposed into 3 different errors: 1). Bias, 2). Variance, and 3). Irreducible error. The irreducible error (sometimes called Bayesian error) is the error that cannot be removed [Kuhn and Johnson (2016)] due to statistical noise in the observations. If we choose a very large hypothesis space $H \ni h_\theta$, the bias will be small because $|H|$ will most likely contain all possible h_θ that fit the input-output relation we would like to model.

However, at the same time, since it contains all possible h_θ , the distance between the incorrect prediction and the truth will also increase, thus making the estimation error (variance) larger. The same logic applies if we choose small hypothesis space $H \ni h_\theta$. Thus, at this point, the “sweet spot” will be to find a balance between the bias and variance as depicted in Fig. 3.4.1-1a-b.

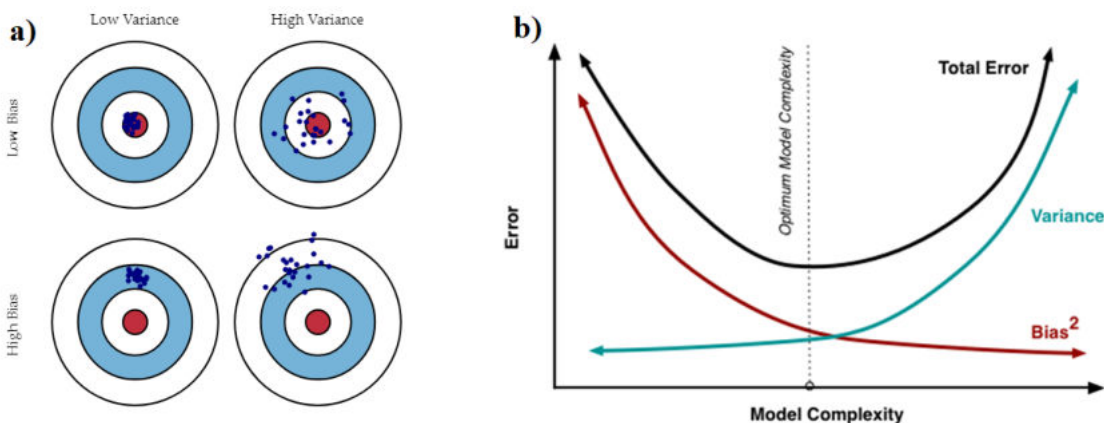


Fig. 3.4.1-1. a). Bull's eye model of four different scenarios representing combinations of both high and low bias and variance. b). Trade-off between bias² and variance.

Before going further, let us denote $|H_n|$ as the cardinality of the hypothesis space H that contains all possible h_θ with n sample points. Let the maximum number of hypothesis functions that can be distinguished $S(H, n)$ be defined as:

$$S(H, n) = \max\{|H_n| \mid X_1, \dots, X_n \in X\} \quad (3.4.1-2)$$

The quantity $S(H, n)$ is denoted as the shattering coefficient of the hypothesis space H with respect to sample size n . The shattering coefficient can be thought as “the number of ways that the function space can separate the patterns into two classes” [Luxburg and Schölkopf (2008)], i.e., the capacity measure of the hypothesis space. In other words, the larger the hypothesis space H , the larger shattering coefficient will be. From the PAC-Learning according to [Wolpert (1990, 1992)], we would like to know what is the quantifiable measure for generalization by using the shattering coefficient S ? [Vapnik and Chervonenkis (1971), Devroye et al. (1996)] have already discussed and given a proof on the convergence bound. The generalization bound that we would like to hold with a probability of at least $1 - \delta$ is defined as [Luxburg and Schölkopf (2008)]:

$$R(h_\theta) \leq R_{\text{emp}}(h_\theta) + \sqrt{\frac{4}{n} (\log(2S(H, n)) - \log(\delta))} \quad (3.4.1-3)$$

Eq. (3.4.1-3) tells us that if both the empirical risk $R_{\text{emp}}(h_\theta)$ and the square root term are small, then the total risk $R(h_\theta)$ can be guaranteed with a probability of at least $1 - \delta$ will be small. This means that even if the size hypothesis space $|H|$ is small, it is not coincidence that some essential aspects of the problem have been captured. Whether a problem is hard to learn is entirely dependent on our prior knowledge, i.e., we must come up with a suitable function class.

Another important concept in terms of generalization beside the shattering coefficient is the VC-Dimension (named after Vapnik and Chervonenkis), which is a measure of function capacity. The purpose of introducing the VC-Dimension is to “characterize the growth behavior of the shattering coefficient using a single number” [Luxburg and Schölkopf (2008)]. Simply speaking, a set of points are said to be shattered, if for each $x_n \in X$, there exists a classifier $h_\theta \in H$ such that $x_n = h_\theta \cap X$. I give a simple explanation using Fig. 3.4.1-2a-b.

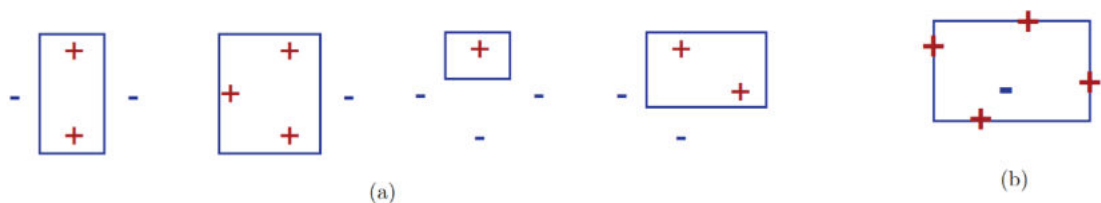


Fig. 3.4.1-2. VC-dimension of axis-aligned rectangles. (a) Examples of realizable dichotomies for four points in a diamond pattern. (b) No sample of five points can be realized if the interior point and the remaining points have opposite labels [Mohri et al. (2018)].

Formally, given a hypothesis space H , a dichotomy of a set X is defined as one of the possible ways of labeling the points of D using a hypothesis $h_\theta \in H$. A set D of n

≥ 1 points is said to be shattered by a hypothesis set H when H realizes all possible dichotomies of S , i.e., when $|H_n| = 2^n$. The VC-dimension of a hypothesis set H is defined the size of the largest set that can be shattered by H [Mohri et al. (2018)]:

$$VC_H = \max\{n \in \mathbb{N} \mid |H_n| = 2^n \text{ for some } h_\theta\} \quad (3.4.1-4)$$

The other important concept in supervised learning to measure the hypothesis capacity is the Rademacher complexity. Unlike the VC dimension, Rademacher complexity depends on the probability distribution. *The Rademacher complexity captures the richness of a family of functions by measuring the degree to which a hypothesis set can fit random noise* [Mohri et al. (2018)]. The formal definition of Rademacher complexity \mathfrak{R} is given as follows:

Definition 3.4.1-3. Rademacher Complexity [Mohri et al. (2012)].

Let H be a family of functions mapping from X to $[a, b]$ and $X = (x_1, \dots, x_n)$

a fixed n samples with elements in X . Let $\sigma_1, \sigma_2, \dots$ be independent random variables that are assigned either with $+1$ or -1 each with probability 0.5 .

Then, the empirical Rademacher complexity of H with respect to the sample n is defined as:

$$\mathfrak{R}_n \equiv \mathbb{E} \left(\sup_{h \in H} \frac{1}{n} \sum_{i=1}^n \sigma_i h(X_i) \right)$$

The interpretation of Rademacher complexity is as follows: we know that the product $\sigma_i h(X_i)$ either takes value $+1$ if $\sigma_i = h(X_i)$ and -1 if otherwise. The sum product will be therefore larger if σ_i coincides with $h(X_i)$ on many data points and this means the hypothesis function h fits to the labels σ_i and thus the empirical error R_{emp} will be minimized. If we now take the supremum into account, we look up on all possible hypothesis h within the hypothesis space H . Thus, we can expect that a model that generalizes well would have higher Rademacher complexity, i.e., Rademacher complexity measures how large the hypothesis space H is. Like the shattering coefficient, one can prove generalization bounds of the following form: with probability at least $1 - \delta$:

$$R(h_\theta) \leq R_{\text{emp}}(h_\theta) + 2 \cdot \mathfrak{R}_n + \sqrt{\frac{\ln(1/\delta)}{2n}} \quad (3.4.1-4)$$

The Rademacher complexity \mathfrak{R} , together with the VC-dimension [Goldberg and Jerrum (1995), Clayton (2014)] are a measure of richness of a class of real-valued functions - that is, the capability of generalizer L is related to how complex it is. There are many more capacity concepts, but the general form which most bounds take is composed of three different terms and have the following form:

$$R(h_\theta) \leq R_{\text{emp}}(h_\theta) + \text{capacity}(H) + \text{confidence}(\delta) \quad (3.4.1-5)$$

Finally, to wrap-up, the relation between generalization gap and capacity concept (here Rademacher \mathfrak{R} complexity is taken as an example) is formally defined in Def. 3.4.1-4.

Definition 3.4.1-4. Generalization gap [Kawaguchi et al. (2017), Mohri et al. (2012), Balcan (2011)].

Let n -samples in the sampling space of training dataset $D\{(x_1, y_1), \dots, (x_n, y_n)\}$, where $x \in X$ and $y \in Y$, and the generalizer $L(S) : X \rightarrow Y$. The expected generalization risk is denoted as $R_{L(S)}$ and the computable empirical risk is denoted as: $\tilde{R}_{L(S)} = \frac{1}{N} \sum_{i=1}^N J_\theta(L(x_i), y_i)$, where J is the error or loss function associated with L . The generalization gap G is defined as: $G \equiv R_{L(S)} - \tilde{R}_{L(S)}$

Assume $J \in [0, 1]$, we have that $\forall \delta > 0$ with probability at least $1 - \delta$:

$$\sup G \leq 2 \cdot \mathfrak{R}_N(L) + \sqrt{\frac{\ln(1/\delta)}{2N}},$$

where $\mathfrak{R}_n(L)$ is the Rademacher complexity of L .

3.4.2. Inductive Bias

Prior knowledge is incorporated in the way we design the algorithm. This is related to inductive bias. Broadly and informally, there are four categories involving the prior knowledge [Luxburg and Schölkopf (2008)]:

1. The formalization of the data space X , e.g., by its topology. Commonly, this is done by using a distance or a similarity function which tells us how “similar” different input values in X are.
2. The loss function J . This function encodes what the “goal” of learning should be, e.g., we can weight errors on individual data points more heavily than on other data points.
3. Assumptions on the underlying probability distributions, i.e., when we did not make any assumption, no matter what probability distribution generated our data, the generalization bounds would apply.
4. Any deliberate hypothesis set construction in H , i.e., we encode our assumptions on how a useful a classifier might look.

Briefly speaking, the inductive bias of a learning algorithm is the set of assumptions that the learning algorithm A uses to predict outputs for a given input that it has not yet encountered. Formally defined, the inductive bias of learning algorithm A is “*any minimal set of assertions B such that for any target concept \mathfrak{C} and corresponding training data $D_{\mathfrak{C}}$* ” [Mitchell (1997)] that the following formula holds:

$$\forall x_n \in X : (B \wedge D_{\mathfrak{C}} \wedge x_n) \vdash A(x_n, D_{\mathfrak{C}}) \quad (3.4.2-1)$$

Where \mathbf{C} the target concept, $D_{\mathbf{C}}$ set of training examples, x_i is i -th instance of input set X , and $A(x_i, D_{\mathbf{C}})$ the classification assigned to x_i by A after training on set $D_{\mathbf{C}}$ is. Consider following notations:

- y_i : i -th instance of output space Y
- P : Probability distribution on $X \times Y$
- Q : Distribution on P
- L : Loss function that maps $Y \times Y \rightarrow \mathbb{R}$
- H : Hypothesis space which is a set of function $h: X \rightarrow Y$
- \mathbb{H} : Family of hypothesis space where $H \in \mathbb{H}$
- $\Delta_Q(H)$: Loss of hypothesis space H on Q
- $\Delta_P(h)$: Loss of function h on distribution P

The loss function, also sometimes called the cost function, is the error between the actual and predicted output. For a regression problem, there are several functions that can be chosen such as mean squared error, average absolute deviation, or mean difference. For a classification problem, it is more common to choose loss functions that map an output probability between 0 and 1, such as cross-entropy error or hinge loss [Rosasco et al. (2003)]. The goal of bias learning is to find the hypothesis space $H \in \mathbb{H}$ that minimizes $\Delta_Q(H)$:

$$\begin{aligned} \Delta_Q(H) &\equiv \int_P \inf_{h \in H} \Delta_P(h) dQ(P) \\ &= \int_P \inf_{h \in H} \int_{X \times Y} L(h(x), y) dP(x, y) dQ(P) \end{aligned} \quad (3.4.2-2)$$

Typically, minimizing $\Delta_Q(H)$ cannot be done directly, therefore sampling n times from P according to Q is needed to yield: P_1, \dots, P_n . Furthermore, we sample m times from $X \times Y$ according to each P_i to yield the pairs: $\{(x_i^1, y_i^1), \dots, (x_i^m, y_i^m)\}$. In the sequel, an (n, m) -sample is denoted by z and written as a matrix in Eq. (3.4.2-3). In order to choose a hypothesis space $H \in \mathbb{H}$, one needs to minimize the empirical loss on z $\Delta_z(H)$ as in Eq. (3.4.2-4).

$$z \equiv \begin{pmatrix} (x_{11}, y_{11}) & \cdots & (x_{1m}, y_{1m}) \\ \vdots & \ddots & \vdots \\ (x_{n1}, y_{n1}) & \cdots & (x_{nm}, y_{nm}) \end{pmatrix} = \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} \quad (3.4.2-3)$$

$$\Delta_z(\mathcal{H}) \equiv \frac{1}{n} \sum_{i=1}^n \inf_{h \in \mathcal{H}} \Delta_{z_i}(h) \quad (3.4.2-4)$$

The bias induced learning algorithm \hat{A} is then defined as [Baxter (2000)]:

$$\hat{A}: \bigcup_{n > 0, m > 0} (X \times Y)^{(n, m)} \rightarrow \mathbb{H} \quad (3.4.2-5)$$

Which reads: the map that takes (n, m) -samples from the distribution $X \times Y$ as input to the hypothesis space $H \in \mathbb{H}$ as the output.

3.4.3. Neural Network

In neuroscience, a neural network is part of a biological nervous systems that forms a connection which “fires” when it gets a stimulus [Saladin (2016)], whereas in computational neuroscience, this natural phenomenon is imitated to model the biological network for which the mathematical model is often referred to artificial neural network (ANN) [Haykin (2009)]. In many models, neurons are arranged in layers one behind the other, also called single-layer perceptron (SLP). The first artificial neuron was introduced by [McCulloch and Pitts (1943)] as a logical threshold value element with several inputs and a single output. It took a Boolean variable, and thus can assume the states true and false and “fire” when the sum of the input signals exceeded a threshold value. [McCulloch and Pitts (1943)] showed that any simple propositional function (AND, OR, NOT) can be described by a combination of several such neurons as depicted in Fig. 3.4.3-1.

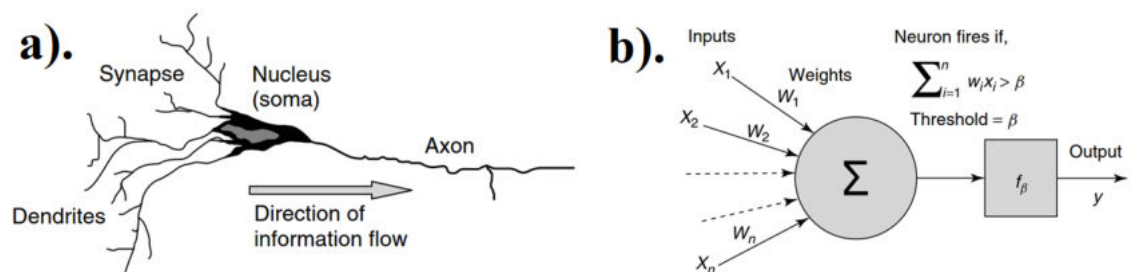


Fig. 3.4.3-1. Depiction of a) biological neuron and its mathematical modelling as b) artificial neuron according to McCulloch and Pitts [Reed (2009)].

Until today, the fundamental assumption of learning rule relies on the Hebbian postulate [Hebb (1949)] which is based on the fact that the activation and inhibition of a synapse can be calculated as the product of pre- and postsynaptic activity. There is neuroscientific evidence that long-term potentiation and spike-timing dependent plasticity (STDP) is the biological pendant to Hebb's postulate. In the SLP, there is only a single layer of artificial neurons, which also represents the output vector. Each neuron is represented by a neuron function and receives the entire input vector x_n as a parameter. The binary output value of the perceptron is learned by adjusting the weights θ_n and bias b of each neuron:

$$\text{Output} = \begin{cases} 1 & \text{if: } \sum_n \theta_n x_n + b > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.4.3-1)$$

Once the weights have been learned, a perceptron is also able to classify input vectors that differ slightly from the vector originally learned. This is precisely what makes the perceptron capable of classification, from which it owes its name. SLPs can be attached together to create a multilayer perceptron (MLP) which represents information flows from the input layer to the output layer via multiple layers. In the case where the information flows in one direction, the network is called a feedforward network as depicted Fig. 3.4.3-2a-b.

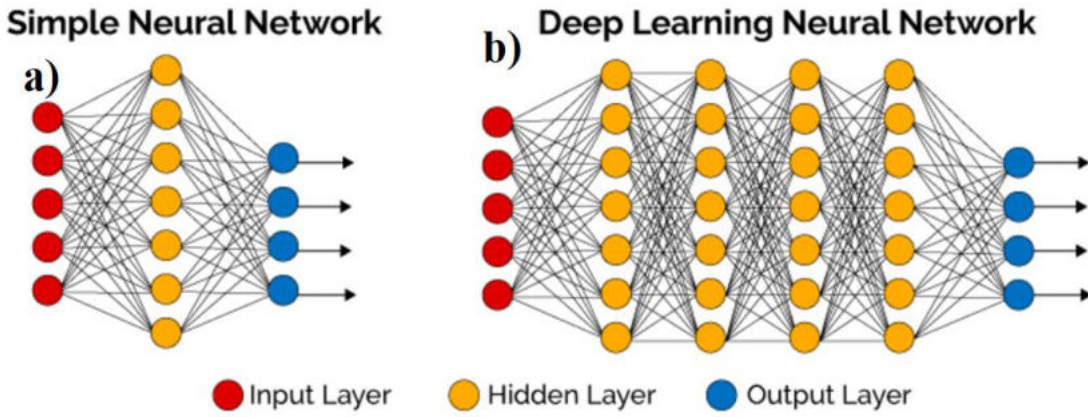


Fig. 3.4.3-2: Feedforward neural network in form of a). simple neural network or SLP and b). MLP, which is an example of a deep neural network.

Training neural network will involve loss function minimization. As mentioned in Section 3.3.1, minimization is just another optimization problem. The output z_m in layer m can be calculated from the input x_n and weight θ_n at node n in layer $m-1$, thus Eq. (3.4.3-1) can be rewritten as:

$$z_m = \varphi \left(\sum_n \theta_n x_n + b \right) \triangleq \varphi(y_n) \quad (3.4.3-2)$$

Where φ is the activation function to be selected. This is typically either as sigmoid function, a hyperbolic tangent (tanh), or a rectified linear unit (ReLU). Per Def. 3.3.1-1, the optimization problem of loss function J can be simplified as Eq. 3.4.3-3 and with the chain rule, the relationship between loss function J and neural weight θ can be rewritten as Eq. 3.4.3-4.

$$\min_{\theta \in [0,1]} J(\theta_{mn}) \quad (3.4.3-3)$$

$$\frac{\partial J}{\partial \theta_{mn}} = \frac{\partial J}{\partial z_m} \frac{\partial z_m}{\partial y_n} \frac{\partial y_n}{\partial \theta_{mn}} = \frac{\partial J}{\partial \sigma(y_n)} \frac{\partial \sigma(y_n)}{\partial y_n} \frac{\partial y_n}{\partial \theta_{mn}} \quad (3.4.3-4)$$

Since Eq. (3.4.3-4) involves the chain rule, we can imagine implementing this in multiple layers and therefore increasing the calculation steps each time we add a layer in the neural network structure. Since the loss is defined as the difference between the output and the input at the preceding layer, such an operation is therefore called backpropagation. There are many variants of how to do backpropagation, but for brevity, only the simplest form without additional algorithmic parameters is given. Concretely, the simplest loop to perform a weight update at step i with gradient descent is defined as:

$$\theta_{mn}^i = \theta_{mn}^{i-1} + \Delta \theta_{mn} = \theta_{mn}^{i-1} - \alpha \frac{\partial J}{\partial \theta_{mn}} \quad (3.4.3-5)$$

The algorithmic procedure of Eq. (3.4.3-5) is given in Algorithm 3.4.3-1.

Algorithm 3.4.3-1: Gradient Update ProcedureInput: α – learning rate

Setup initial values:

$$\theta_{mn} \leftarrow \theta_{mn}^0$$

Calculate $J(\theta_{mn})$

Until termination condition is reached:

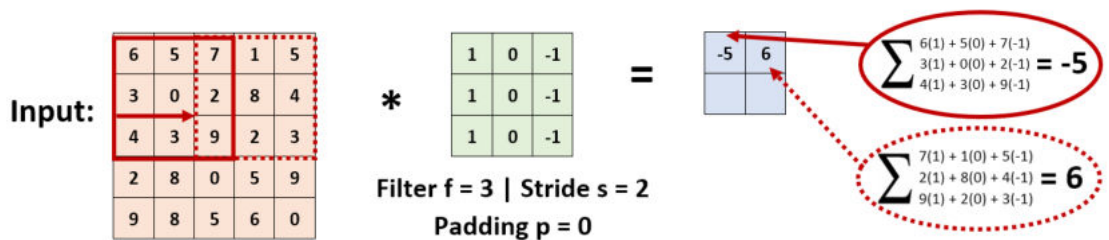
$$\theta_{mn}^{i+1} = \theta_{mn}^i - \alpha \frac{\partial J}{\partial \theta_{mn}}$$

Calculate loss $J(\theta_{mn} \leftarrow \theta_{mn}^{i+1})$ If $J(\theta_{mn}^{i+1}) < J(\theta_{mn}^i)$:

$$J(\theta_{mn}) \leftarrow J(\theta_{mn}^{i+1})$$

If the size of the input layer is small (i.e., less than 1000 nodes), the weights of the network can be easily adjusted during the training. However, as the size of the input layer grows, the number of weights becomes large, and the training becomes unbearable. For this reason, the convolutional neural network (CNN), sometimes also called ConvNet was introduced in the computer science community. It turns out that CNN has been proven to be an extremely effective tool in the field of computer vision since it has less network weights to be trained and this approach became a foundation of modern computer vision.

A CNN consists of multiple successively increasing refined data filters – the layers of a convolutional filter. Within the context of this analogy, the input data to a CNN, for instance a single black and white image can be represented as a 3D tensor of size: height \times width \times channels of pixel values ranging from 0 to 255. The transformations performed by each layer are parametrized by its neural weights and then, the loss is calculated. There are two types of layers that are normally used in CNN core architecture: the convolutional layer and the pooling layer. The core CNN is typically attached to the fully connected (FC) layer. The convolutional layer performs a convolutional operation to the input tensor that it receives from previous layer for feature extraction. The filter performs the process of feature extraction by sliding over the input data by several pixels which is also assigned and called the stride as depicted in Fig. 3.4.3-3.

Fig. 3.4.3-3: Convolution operation in CNN using stride $s = 2$ and zero-padding

CNN would theoretically accept an arbitrarily sized input, but it is normally used for data with a fixed input size. In reality there is no such thing as infinite space, i.e., we are practically limited to the current computational capacity. Thus, employing a CNN for data with variable length such as a time series (such as

weather forecast, stock market, speech signal, seismic/volcanic activity) and long-dependency input (such as long sentences, handwriting recognition, or language translation) will pose a computational problem for CNN. For this reason, a recurrent neural network (RNN) is developed. As the name says, an RNN is a neural network with recurrent connections, made up of interconnected neurons in which there is at least one recurring input cycle within the network.

There are 3 most common variants in the RNN family: the original (referred to as vanilla) RNN, long short-term memory (LSTM), and gated recurrent unit (GRU). Like a feedforward network, the units are connected by synapses and the neural output is a nonlinear combination of its inputs. The differences between these three RNNs are the basic operation within its cell. Vanilla RNN has the basic recurrent operation: at time t , the input at time t X_t and pre-current input at time $t-1$ X_{t-1} are added together to calculate the output at time t .

When training such a network, the SGD method is often used. A deep neural network implementation will involve a very long chain of training, i.e., after a long backpropagation chain, the gradient starts to vanish. The LSTM method solves this problem by using three types of gates for: an input gate, a remember-forget gate and an output gate to enable a short-term memory that lasts for a long time. The operations within LSTM cells with the input at time t X_t , bias b , hidden state h , cell state c , weights θ and activation function φ are:

$$\begin{aligned} \text{Forget gate:} \quad & f_t = \varphi(\theta_F X_t + \hat{\theta}_F h_{t-1} + b_F) \\ \text{Input gate:} \quad & i_t = \varphi(\theta_I X_t + \hat{\theta}_I h_{t-1} + b_I) \\ \text{Output gate:} \quad & o_t = \varphi(\theta_O X_t + \hat{\theta}_O h_{t-1} + b_O) \\ \text{Cell state:} \quad & c_t = F_t \circ c_{t-1} + i_t \circ \varphi(\theta_c X_t + \hat{\theta}_c h_{t-1} + b_c) \\ \text{Hidden state:} \quad & h_t = o_t \circ \varphi(c_t) \end{aligned}$$

Instead of a single neural function, the LSTM module consists of the aforementioned gates. Briefly speaking, the input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell or is forgotten and the output gate the extent to which it is in which the value in the cell is used to calculate the next module in the chain.

A Gated Recurrent Unit (GRU) network is a variant of LSTMs introduced in 2014 and it has comparable performance to that of LSTMs for the prediction of time series. The main difference is the number of parameters within the recurrent cells: the cell is only associated with a hidden state (no more cell state), while the input and forget gates of the cells are merged to become “update gate z ” and the output gate is replaced by a “reset gate r ”. The GRU cells operations are:

$$\begin{aligned} \text{Update gate:} \quad & z_t = \varphi(\theta_z X_t + \hat{\theta}_z h_{t-1} + b_z) \\ \text{Reset gate:} \quad & r_t = \varphi(\theta_r X_t + \hat{\theta}_r h_{t-1} + b_r) \\ \text{Hidden state:} \quad & h_t = (1 - z_t) \circ \varphi(\theta_h X_t + \hat{\theta}_h (r_t \circ h_{t-1}) + b_h) \end{aligned}$$

3.4.4. Hyperparameter Tuning

A hyperparameter is a parameter that is used to control the training algorithm. Model M is constructed by a learning algorithm A using the training set $D_{\text{training}}(x_i, y_i)$, where the learning algorithm is parameterized by a set of hyperparameters Φ , i.e., $M = A(D_{\text{training}}(x_i, y_i); \Phi)$. The formalized hyperparameter tuning problem is defined as [Claesen and De Moor (2015)]:

$$\Phi^* = \arg \min_{\Phi} J(D_{\text{test}}; A(D_{\text{training}}; \Phi)) \quad (3.4.4-1)$$

Where Φ^* is the optimal hyperparameter, J is the selected cost function, and D_{test} the test set. For neural network training, the most common hyperparameters are learning rate, number of training epochs, hidden layers & hidden units and types of activation function. A list of common activation functions for a Hadamard product z of transposed weights θ^T and input X are given in Table 3.4.4-1.

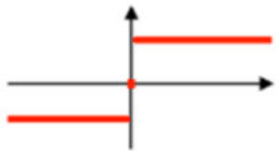

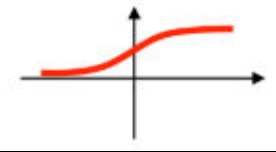
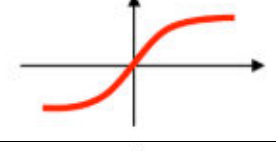
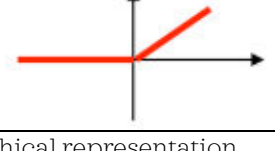
Activation Function	Equation	1D-Graph
Signum (sgn)	$\varphi(\theta^T X) = \varphi(z) = \begin{cases} -1, & z < 0 \\ 0, & z = 0 \\ 1, & z > 0 \end{cases}$	
Linear	$\varphi(\theta^T X) = \varphi(z) = z$	
Sigmoid (sigm)	$\varphi(\theta^T X) = \varphi(z) = \frac{1}{1 + \exp(-z)}$	
Hyperbolic tangent (tanh)	$\varphi(\theta^T X) = \varphi(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$	
Linear Rectifier (ReLU)	$\varphi(\theta^T X) = \varphi(z) = \max(0, z)$	

Table 3.4.4-1: Commonly used activation functions with simplified graphical representation

During backpropagation, the weights are adjusted until the error is minimized. The input data is generally divided into training sample groups called batches and reshuffled. A complete cycle of all input data after several reshuffling is referred to as an epoch. When applying the SGD during the training, it is likely that it will be slowed down in one of the local minima, especially for non-convex objective function. [Sutskever et al. (2013)] tackled this by adding momentum γ and it was further developed [Botev et al. (2017)] by adapting Nesterov accelerated gradient (NAG) to escape the local minima trap:

$$v_i = \underbrace{\gamma}_{\text{Momentum}} v_{i-1} - \underbrace{\eta \nabla J(\theta - \gamma v_{i-1})}_{\text{NAG}} \quad (3.4.4-1)$$

Update rule:

$$\theta \leftarrow \theta_{i-1} + v_i$$

Where v_i is the gradient update velocity at increment i and η is the learning rate. Another popular gradient descent is called Adam [Kingma and Ba (2015)], its variant RMSProp [Hinton et al. (2014)], Adagrad [Duchi et al. (2011)], and Adadelata [Zeiler et al. (2015)]. The difference between the gradient descent flavors is:

- Adagrad adapts the learning rate to the parameters, i.e., it has a low learning rate for connection weights that are associated with frequent features and a higher learning rate for connection weights that are associated with infrequent features.
- Adadelata is an Adagrad extension that seeks to reduce its “*monotonically decreasing learning rate. [...] Adadelata restricts the window of accumulated past gradients to a fixed size*” [Zeiler et al. (2015)]. Another alternative denomination to Adadelata is RMSprop.
- Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter. Like Adadelata, Adam also “*stores an exponentially decaying average of past squared gradients, and additionally it keeps an exponentially decaying average of past gradients similar to momentum.*” [Ruder (2016)].

The formal definition of different SGD flavors is summarized in Table 3.4.4-2. The ϵ -factor (typically small at 10^{-8}) is added to the flavors to avoid division by 0.

SGD Flavor	Formal Definition
Adagrad	$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \nabla J(\theta_{t,i})$
Adadelata (RMSProp)	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[\nabla J(\theta_t) ^2] + \epsilon}} \nabla J(\theta_t)$ $E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1 - \gamma)\Delta\theta^2_t$
Adam	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} m_t$ $m_t = \frac{\beta_1 m_{t-1} + (1 - \beta_1) \nabla J(\theta_t)}{1 - \beta_1^t}, \quad v_t = \frac{\beta_2 v_{t-1} + (1 - \beta_2) \nabla J(\theta_t) ^2}{1 - \beta_2^t}$

Table 3.4.4-2: Summary of adaptive SGD. In Adagrad, $G_{t,ii}$ is a diagonal matrix where each diagonal element i, i is the square sum of the gradients up to time step t , whereas in RMSProp and Adadelata, E signifies the running average of the gradient. In Adam, v_t is the exponentially decaying average of past squared gradients, m_t is the exponentially decaying average of past gradients, and β_1, β_2 are the bias factor of the first and second moment of estimates, respectively.

3.4.5. Regularization

To overcome the overfitting problem during training, a *regularization* technique is introduced. It is the process of regularizing the parameters that constrains, regularizes, or shrinks the coefficient estimates towards zero to prevent the algorithm from shaping an overly complex model. There are two categories of regularization: L_1 -Norm (sometimes called Lasso Regularization) and L_2 -Norm (also called Ridge Regression). In L_1 -Norm, the sum of the absolute values of each element in the weight tensor is summed. In short, in L_1 , we estimate the median of the distribution while in L_2 , we estimate the distribution mean instead. Table 3.4.5-1 summarizes the regularized cost function J expressed in mean-squared error (MSE) and cross-entropy error (CEE).

Norm	Regularized Cost Function
L_1	$J_{\text{MSE}}(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - h_{\theta}(x_i))^2 + \vartheta \sum_{j=0}^m \ \theta_j\ $ $J_{\text{CEE}}(\theta) = -\frac{1}{N} \left[\sum_{i=1}^N y_i \log(h_{\theta}(x_i)) + (1 - y_i) \log(1 - h_{\theta}(x_i)) \right] + \vartheta \sum_{j=0}^m \ \theta_j\ $
L_2	$J_{\text{MSE}}(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - h_{\theta}(x_i))^2 + \vartheta \sum_{j=0}^m \theta_j^2$ $J_{\text{CEE}}(\theta) = -\frac{1}{N} \left[\sum_{i=1}^N y_i \log(h_{\theta}(x_i)) + (1 - y_i) \log(1 - h_{\theta}(x_i)) \right] + \vartheta \sum_{j=0}^m \theta_j^2$

Table 3.4.5-1: L_1 and L_2 -Norm for both MSE and CEE cost function input x and true output y , where θ is the weight parameter, h_{θ} is the hypothesis function w.r.t θ , i.e., the predicted output, ϑ is the regularization factor, j is the weights index within sample i of a total of N -samples taken from the underlying distribution $D(x,y)$.

Another regularization methodology is called dropout which was invented by [Srivastava et al. (2014)]. When training the network, between 25% – 50% of the neurons in each layer of the network are randomly switched off to prevent the neighboring neurons from "co-approximating" each other too closely. A neural network of n units can be viewed as a collection of 2^n possible thinned networks. An example of dropped out network is depicted in Fig. 3.4.5-1.

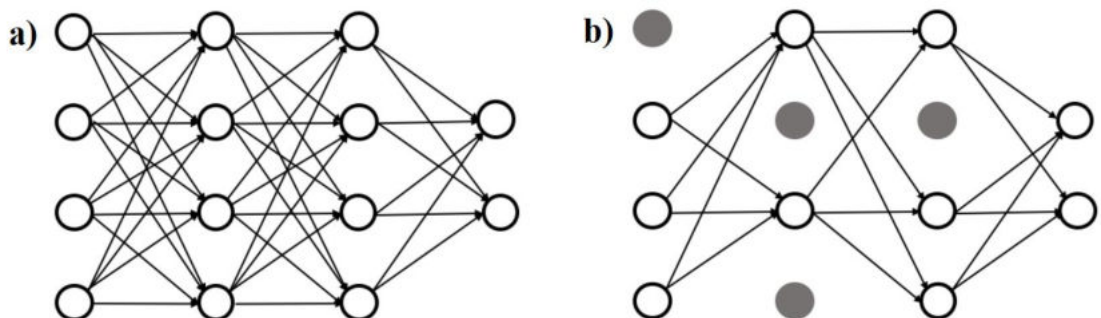


Fig. 3.4.5-1: a). A standard neural net with 2 hidden layers. b). An example of a thinned net produced by applying dropout to the network on the left [Srivastava et al. (2014), Wang et al. (2018)].

3.5. Features Learning and Invariant Representation

Consider the following definition from an example of autonomous driving perception [Pendleton et al. (2017)]: “*Environment perception is a fundamental function to enable autonomous vehicles, which provides the vehicle with crucial information on the driving environment, including the free drivable areas and surrounding obstacles’ locations, velocities, and even predictions of their future states [...]*”. We can adapt this for SHM in an analogous way: Perception in health monitoring is a fundamental function to enable the full functionality of autonomous damage evaluation, which provides the SHM system with crucial information on the changes in the sensory input such as change in amplitude, frequency, or phase-shift.

Further, consider the human hearing process [Stöver and Diensthuber (2011)] which consists of two parts: 1). Conversion of mechanical (audio) waves that travel through the concha, malleus and cochlea into electrical stimulation by stereocilia and 2). Information processing that is carried by a neurotransmitter through auditory nerves [Petralia and Wenthold (2009)] into auditory cortex located in the temporal lobe of human brain. Part 1 is analogous to the conversion of an analog Lamb wave signal into an electrical signal by a piezoelectric transducer (PZT) thanks to the piezoelectric effect. Part 2 is analogous to SHM signal processing to extract information from the electrical signal to have a meaningful interpretation.

The postulate of [Hebb (1949)] states that the connectivity between two neurons is determined by how strong they are linked to each other, i.e., the connectivity between two neurons increases if they are simultaneously activated and decreases otherwise [Wallisch et al. (2014)]. Practically, if the auditory cortex is fed with the same audio signal over time, at some point the brain would automatically recognize it. In this process, human brain is thought to create invariant representation on recognizing objects from different timeframe and locations [Quiroga et al. (2005)]. To help understand what invariant representation means, consider following lemmata:

Lemma 3.5-1. The affine transformation of multivariable function composition O (trivial proof)

Let a set F^k , where $k \in \mathbb{N}$, be defined as $F^k \triangleq \{f \mid f : \mathbb{R}^k \rightarrow \mathbb{R}^k\}$ and (X_1, \dots, X_k) affine spaces. A composition O is defined as: $O : F^k \times F^k \rightarrow F^k$. Let (X, Y, Z) be the hyperspaces of the affine spaces (X_1, \dots, X_k) , (Y_1, \dots, Y_k) , and (Z_1, \dots, Z_k) that contain point sets $(x_1^{(k)}, \dots, x_i^{(k)})$, $(y_1^{(k)}, \dots, y_i^{(k)})$, and $(z_1^{(k)}, \dots, z_i^{(k)})$ where $i \in \mathbb{N}$. Let (U, V) be a set of linear maps, and $(U : X \rightarrow Y)$, $(V : Y \rightarrow Z)$ be affine transformations. A map $U : X \rightarrow Y$ is said to be an affine map if $\exists V : Y \rightarrow Z$ such that $V(y - z) = f(y) - f(z) \forall y, z$ in (Y, Z) . Then, the composition $O : V \circ (U : X \rightarrow Y)$ is an affine transformation with tangent map $V \circ U$.

Lemma 3.5-2. The cumulative distribution function (CDF) as the estimate of invariant for affine transformation. [Liao et al. (2013)]

Assume an observation $x \in X$, where X is a set of all possible observations.

Furthermore, we consider transformation $g_t \in G$, where G is 2D affine groups that consists of all possible transformations T , such as translation, resizing, etc.

Assume a set of smallest atomic representations of observation x as τ_i where $i \in \mathbb{N}$. For the set $\{g_t \tau_i \mid t = 1, \dots, T\}$, the distribution of $\langle x, g_t \tau_i \rangle$ is invariant and unique to each observation. The empirical distribution κ of the inner products

$\langle x, g_t \tau_i \rangle$ is used as the signature: $\kappa_n^i(x) = \frac{1}{T} \sum_{t=1}^T \sigma(\langle x, g_t \tau_i \rangle + n\Delta)$, where σ is the typical sigmoid function and Δ is the resolution parameter for each observation. Since each observation x has its own characteristic empirical κ , it also shows that these signatures could be used to discriminate between them.

Lemma 3.5-3. Invariant latent. [Feige (2019)]

Let consider class label $y_{\lambda^{(n)}}$ that can be associated with the phenomenon λ in

Eq. 3.2-1. The full set of class y -labeled data $\{x_n \mid h_{\theta}(x_n) = y_{\lambda^{(n)}}\}$ will be denoted as $D_{h_{\theta}}^y$. The invariant latent $r_{y_{\lambda^{(n)}}}$ of label $y_{\lambda^{(n)}}$ is thus defined as:

$$r_{y_{\lambda^{(n)}}} = \mathbb{E}_{x \sim (D_{h_{\theta}}^y \setminus \{x_n\})} [f_{\theta}(x)] \approx \frac{1}{N} \sum_{i=1}^N f_{\theta}(x^i)$$

where N is the number of sampled observations.

The importance of these above-mentioned lemmas is following:

1. Any (non)-dispersive acoustic wave signal (including body waves such as a P- and SV-wave and/or a surface wave such as Lamb-, Rayleigh wave, and SH-wave) can be regarded as a decomposable multivariable function of multiple polynomial functions and there exists an affine function that transforms the space of signal features to affine spaces of signal features,
2. the transformation of each atomic representation of wave is invariant – these has been used e.g., in wavelet transformation, and
3. in a more high-level waveform composition such as a longer time-series, the invariant latent can be explicitly calculated to provide the information needed to learn which wave packet within the time-series from all possible damage classes have in common – e.g., the first and second wave packet for similar damage sizes are all similar, i.e., having a cosine similarity 1.

In March 2019, the innovator club at Merck proposed these research questions in the brain hack challenge that they called *The Future of AI* [Merck (2019)]. In one of the challenges, they were looking for a formalization of a cortical algorithm, a mechanism which might be able to imitate the pattern recognition process that is believed to be taking place in the grid neuron, which is in the mammal neocortex. In one of their core problems statements, two assumptions are made:

1. *“Hierarchical structure: Entities are hierarchically structured, which means that an entity E is either fundamental (i.e., it cannot be broken down any further) or it is composed of other entities E_1, E_2, \dots, E_i such that every perception p of E is associated with a set of perceptions p_1, p_2, \dots, p_i .”*
2. *“Entity conservation over time: Subsequent perceptions in time are (usually) generated by the same set of entities.”*

For the hierarchical structure, entities are subdivided into smaller features, and an analogy for this assumption would be looking at one car which can be decomposed into a sub-structure such as a wheel, door, light, windshield, etc. The same analogy for human faces can be applied: nose, eyes, ears, etc. These sub-features are in turn can be decomposed into smaller shapes such as the edges and circles. The idea is depicted in Fig. 3.5-1.

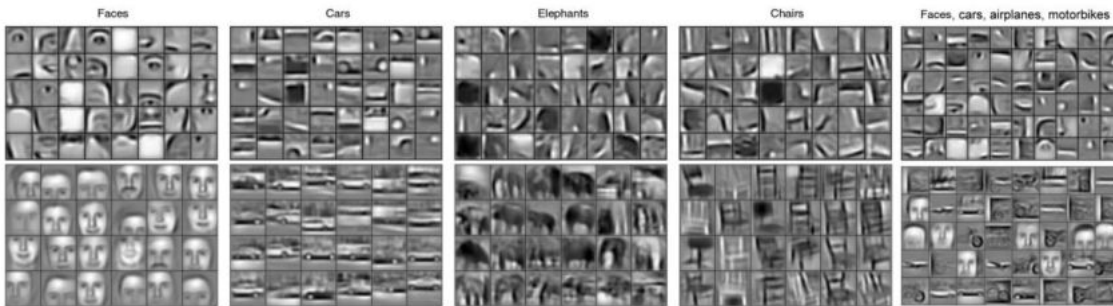


Fig. 3.5-1: Structured hierarchical representations [Lee et al. (2011)]

For the second assumption (conserved entity over time), ideally not only the full-length signal but also the different observation perspective must be considered. An analogy for this assumption would be looking at a particular car, but it can be looked from different angles, e.g., from the front, left, back, etc. – see Fig. 3.5-2 to have an easy way to imagine this. But at the end, this car represents exactly only a single entity.



Fig. 3.5-2: Representation of entity conservation over time [Ewald et al. (2022)]

The above-mentioned assumptions are taken as an inspiration to model different various sensing perceptions: a guided wave signal from different sensing locations can be represented as smaller subset of features (e.g., single wave packets), or it could also be represented as holistic datapoints encompassing the conservation over time. This will be later discussed in chapter 6.

Within this context, we must assume that each observation $x \in X$ should correspond to at least an element $y \in Y$. The relational mapping between domains X and Y can at least describe in three characteristics: surjective, injective, and bijective. The formal definition is given in Lemma 3.5.1-4.

Lemma 3.5.1-4. The bijective relation maps an input space X into output sets Y (trivial proof)

Consider $f : X \rightarrow Y$, where $X, Y \in \mathbb{R}^k$, then f is **surjective** if for

$f(x_1, \dots, x_k) = [y_1, \dots, y_k]$ there exists at least one solution $x \in X$ for each $y \in Y$,

i.e. $\forall y \in Y : \exists x \in X : [y_1, \dots, y_k] = f(x_1, \dots, x_k)$. Further, f is said to be **injective**, if

for $f(x_1, \dots, x_k) = [y_1, \dots, y_k]$ there exists at most one solution $x \in X$ for each

$y \in Y$, i.e. $\forall x^{(1)}, x^{(2)} \in X : f(x^{(1)}) = f(x^{(2)}) \Rightarrow x^{(1)} = x^{(2)}$. Finally, f is said to be

bijective if f is both **surjective** and **injective**.

Depending on the case, be it either a discriminative or a generative problem, any of these three characteristics can be modelled in machine learning. When there is a one-to-one relation between both domains, then this relation is called bijective. Assuming that the relation between each parameter combination in the formal diagnostic and its corresponding label explained means that changing any parameters in the setup would cause the phenomenon λ to change. In other words, each λ is only unique to each parameter combination. For example, the change of the material, dimension, boundary conditions, sensor geometry, etc. and/or combination of these parameters, would cause λ to behave differently, so that the observation set X would also change.

Literatures

Achenbach JD. *Wave Propagation in Elastic Solids*. North-Holland American Elsevier, New York (1973).

Apostol T. *Mathematical Analysis* (2nd Ed.). Addison-Wesley, Boston (1974).

Balcan MF. *Rademacher Complexity*. In Lecture series CS 8803: Machine Learning Theory, Carnegie Mellon University (2011)

Bartle RG, Sherbert DR. *Introduction to Real Analysis* (4th Ed.). John Wiley & Sons, Inc., Hoboken (2011).

Baxter J. *A Model of Inductive Bias Learning*. J Artificial Intelligence Research. Vol. 12: 149-198 (2000).

Betz DC, Staszewski WJ, Thursby G, Culshaw B. *Structural Damage Identification Using Multifunctional Bragg Grating Sensors: II. Damage Detection Results and Analysis*. J Smart Materials and Structures. Vol. 15: 1313-1322 (2006).

Boashash B. *Time-Frequency Signal Analysis and Processing - A Comprehensive Reference* (2nd Ed.). Elsevier, Amsterdam / Boston / Heidelberg / London / New York (2016).

Boller C, Staszewski WJ, Tomlinson GR. *Health Monitoring of Aerospace Structures: Smart Sensor Technologies and Signal Processing*. John Wiley & Sons Ltd, West Sussex (2004).

Botev A, Lever G, Barber D. *Nesterov's Accelerated Gradient and Momentum as Approximations to Regularised Update Descent*. Proc. Intl Joint Conf on Neural Networks (IJCNN), Anchorage (2017).

Budoya D, Castro B, Campeiro L, Silveira R, Freitas E, Baptista F. *Analysis of Piezoelectric Diaphragms in Impedance-Based Damage Detection in Large Structures*. Proc. MDPI. Vol.2: 131 (2018)

Cerniglia D, Pantano A, Montinaro N. *3D Simulations and Experiments of Guided Wave Propagation in Adhesively Bonded Multi-Layered Structures*. J NDT & E Intl. Vol. 43(6): 527-535 (2010).

Charlier B. *The Greedy Algorithms Class: Formalization, Synthesis and Generalization*. Lectures Notes. UCL Belgium (1995).

Claesen M, De Moor B. *Hyperparameter Search in Machine Learning*. Proc. The XI Metaheuristics International Conference, Agadir (2015).

- Clayton S. *Topic 10: Rademacher Complexity*. In Lecture series EECS 598: Lecture on Statistical Learning Theory. University of Michigan (2014).
- Debnath L. *Wavelet Transforms and Their Applications*. Birkhäuser, Boston (2002).
- Devroye L, Györfi L, Lugosi G. *A Probabilistic Theory of Pattern Recognition*. Springer, New York (1996).
- Döring D. *Luftgekoppelter Ultraschall und Geführte Wellen für die Anwendung in der Zerstörungsfreien Werkstoffprüfung*. PhD Dissertation. Universität Stuttgart (2011).
- Duchi J, Hazan E, Singer Y. *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*. J Machine Learning Research. Vol. 12: 2121-2159 (2011).
- Duczek S, Joulaian M, Düster A, Gabbert U. *Numerical Analysis of Lamb Waves Using the Finite and Spectral Cell Methods*. Intl J Numerical Methods in Engineering. Vol. 99(1): 26-53 (2014).
- Echevarría LC, Santiago OL, Velho HFC, Neto AJS. *Fault Diagnosis Inverse Problems: Solution with Metaheuristics*. Springer Intl Publishing, Cham (2019).
- English TM. *Optimization is Easy and Learning is Hard in the Typical Function*. Proc. 2000 Congress on Evolutionary Computation (CEC00), La Jolla (2000).
- Ewald V, Groves RM, Benedictus R. *Integrative Approach for Transducer Positioning Optimization for Ultrasonic Structural Health Monitoring for the Detection of Deterministic and Probabilistic Damage Location*. Intl J Structural Health Monitoring. DOI: 10.1177/1475921720933172 (2020).
- Ewald V, Venkat RS, Asokkumar A, Benedictus R, Boller C, Groves RM. *Perception Modelling by Invariant Representation of Deep Learning for Automated Structural Diagnostic in Aircraft Maintenance: A Study Case using DeepSHM*. J Mechanical System and Signal Processing. Vol 165: 108153 (2022).
- Feige I. *Invariant-Equivariant Representation Learning for Multi-Class Data*. 36th Intl Conf on Machine Learning (ICML), Long Beach, 1-5 (2019).
- Fraga-Gonzalez LF, Aguilera RQF, Gonzalez AG, Ante GS. *Adaptive Simulated Annealing for Tuning PID Controllers*. J AI Communications, Vol. 30(5): 347-362 (2017).
- Giurgiutiu V. *Structural Health Monitoring with Piezoelectric Wafer Active Sensor*. Academic Press-Elsevier, San Diego (2014).
- Gopalakrishnan S, Chakraborty A, Mahapatra DR. *Spectral Finite Element Method – Wave Propagation, Diagnostics and Control in Anisotropic and Inhomogeneous Structures*. Springer-Verlag, London (2008).
- Goldberg PW, Jerrum MR. *Bounding the Vapnik-Chervonenkis Dimension of Concept Classes Parameterized by Real Numbers*. J Machine Learning. Vol. 18(2-3): 131-148 (1995).
- Gresil M, Giurgiutiu V, Shen B, Poddar B. *Guidelines for Using the Finite Element Method for Modeling Guided Lamb Wave Propagation in SHM Processes*. Proc. 6th European Workshop on Structural Health Monitoring (EWSHM), Dresden (2012).
- Güemes JA, Ostachowicz W. *New Trends in Structural Health Monitoring*. Springer Verlag, Wien / Heidelberg / New York / Dordrecht / London (2013).
- Guy P, Jayet Y, Goujon L. *Guided Wave Interaction with Complex Delaminations: Application to Damage Detection in Composite Structures*. Proc. NDE for Health Monitoring and Diagnostics, San Diego (2003).
- Harley J, Zoubi A, Matthews VJ, Adams DO. *Lamb Waves Mode Decomposition Using the Cross-Wigner-Ville Distribution*. Intl Workshop on Structural Health Monitoring, Stanford (2015).
- Haykin S. *Neural Networks and Learning Machines* (3rd Ed.). Pearson, New York / Boston / San Francisco (2009).
- Hebb DO. *The Organization of Behavior: A Neuropsychological Theory*. John Wiley and Sons, New York (1949).
- Hlawatsch F. *Time-Frequency Analysis and Synthesis of Linear Signal Spaces*. Springer, Boston (1998).
- Hinton G, Srivastava N, Swersky S. *Lecture 6a: Overview of Mini-Batch Gradient Descent*. In Lecture: Neural Networks for Machine Learning (2016). Available: <http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf> (Last online: FEB-2021)
- Ho YC, Pepyne DL. *Simple Explanation of the No-Free-Lunch Theorem and Its Implication*. J Optimization Theory and Applications: Vol. 115: 549-570 (2002).
- Hoseini F, Shahbahrani A, Bayat P. *AdaptAhead Optimization Algorithm for Learning Deep CNN Applied to MRI Segmentation*. J Digit Imaging. Vol. 32: 105-115 (2019).
- Kasama H, Takemoto M, Ono K. *Attenuation Measurement of Laser Excited So-Lamb Wave by the Wavelet Transform and Porosity Estimation in Superplastic Al-Mg Plate*. Hihakai-Kensa Japanese J Nondestructive Testing. Vol. 49(4): 269-276 (2000).

- Kawaguchi K, Kaelbling LP, Bengio Y. *Generalization in Deep Learning*. Mathematics of Deep Learning, Cambridge University Press, to appear. Preprint available as: MIT-CSAIL-TR-2018-014
- Kehtarnavaz N. *Digital Signal Processing System Design – LabVIEW-Based Hybrid Programming* (2nd Ed.). Elsevier, Amsterdam / Boston / Heidelberg / London / New York (2008).
- Kennedy J, Eberhart R. *Particle Swarm Optimization*. Proc. Intl Conf on Neural Networks (ICNN), Perth (1995).
- Kingma DP, Ba JL. *Adam: A Method for Stochastic Optimization*. Proc. Intl Conf on Learning Representations, San Diego (2015).
- Konstantinidis G, Drinkwater BW, Wilcox PD. *The Temperature Stability of Guided Wave Structural Health Monitoring Systems*. J Smart Materials and Structures. Vol. 15: 967-976 (2006).
- Kramer O. *Genetic Algorithm Essentials*. In: Studies in Computational Intelligence. Springer International Publishing, Cham, 2017.
- Kuhn M, Johnson K. *Applied Predictive Modeling* (5th Ed.). Springer Science+Business Media, New York (2016).
- Lee H, Grosse R, Ranganath R, Ng AY. *Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Network*. Communications of the ACM. Vol. 54(10): 95-103 (2011).
- Legendre S, Massicote D, Goyette J, Bose TK. *Wavelet-transform-based Method of Analysis for Lamb Wave Ultrasonic NDE Signals*. IEEE Transactions on Instrumentation and Measurement. Vol. 49(3): 524-530 (2000).
- Li F, Meng Guang, Kageyama K, Ye L. *Optimal Mother Wavelet Selection for Lamb Wave Analyses*. J Intelligent Material Systems and Structures. Vol. 20(10): 1147-1161 (2009).
- Li J, Liu S. *Mode Identification of Lamb Waves*. Proc. 17th World Conf on Nondestructive Testing, Shanghai (2008).
- Liao Q, Leibo JZ, Poggio T. *Learning Invariant Representations and Applications to Face Verification*. 27th Conf on Neural Information Processing System (NIPS), Lake Tahoe, 1-9 (2013).
- Liu Z, Yu H, He C, Wu B. *Delamination Detection in Composite Beams using Pure Lamb Mode Generated by Air-Coupled Ultrasonic Transducer*. J Intelligent Material System and Structures. Vol. 25(5): 541-550 (2013).
- Luangvilai K. *Attenuation of Ultrasonic Lamb Waves with Applications to Material Characterization and Condition Monitoring*. PhD Dissertation. Georgia Institute of Technology (2007).
- Luxburg U, Schölkopf B. *Statistical Learning Theory: Models, Concepts, and Results*. In: Handbook of the History of Logic (Vol.: 10). Elsevier, Oxford / Amsterdam / Waltham (2011).
- Malaeb RA, Mahfoud EN, Harb MS. *Decomposition of Fundamental Lamb Wave Modes in Complex Metal Structures Using COMSOL*. Proc. COMSOL Conf, Lausanne (2018).
- Marcos EM. *Cracks Detection in Aluminium Plates by Ultrasounds using Lamb Waves*. MSc Thesis. AGH University Kraków (2011).
- Martins SL, Ribeiro CC. *Metaheuristics and Applications to Optimization Problems in Telecommunications*. In: Handbook of Optimization in Telecommunications. Springer, Boston (2006).
- Masserey B, Fromme P. *On the Reflection of Coupled Rayleigh-like Waves at Surface Defects in Plates*. J Acoustical Society of America. Vol. 123(1): 88-98 (2008).
- McCulloch W, Pitts W. *A Logical Calculus of the Ideas Immanent in Nervous Activity*. Bulletin of Mathematical Biophysics. Vol. 5: 113-133 (1943).
- Merck: *Future of AI Challenge*. Available online <https://app.ekipa.de/challenges/future-of-ai/brief> (Last online: FEB-2020).
- Michaels JE, Michaels TE. *Guided Wave Signal Processing and Image Fusion for In-Situ Damage Localization in Plates*. J Wave Motion. Vol. 44: 482-492 (2007).
- Mishra S, Unnikrishnan L, Nayak SK, Mohanty S. *Advances in Piezoelectric Polymer Composites for Energy Harvesting Applications: A Systematic Review*. Vol. 304(1): 1800463 (2018).
- Mitchell T. *Machine Learning*. McGraw-Hill, Redmond & Ithaca (1997).
- Mohri M, Rostamizadeh A, Talwaker A. *Foundations of Machine Learning* (2nd Ed.) MIT Press, Cambridge & London (2012).
- Moran S, Yehudayoff A. *Sample Compression Schemes for VC Classes*. J Communications of the Association of Computing Machinery (ACM). Vol. 63(3): 1-21 (2016).
- Moser F, Jacobs LJ, Qu J. *Modeling Elastic Wave Propagation in Waveguides with the Finite Element Method*. J NDT & E Intl. Vol. 32(4): 225-234 (1999).

- Niethammer M, Eisenhardt C, Jacob LJ. *Application of the Short Time Fourier Transform to Interpret Ultrasonic Signals*. AIP Conf Proc. Vol. 509: 703 (2000).
- Nocedal J, Wright SJ. *Numerical Optimization* (2nd Ed.). Springer Science+Business Media, New York (2006).
- Ono K, Gallego A. *Attenuation of Lamb Waves in CFRP Plates*. J Acoustic Emission. Vol. 30: 109-123 (2012).
- Ono K. *Review on Structural Health Evaluation with Acoustic Emission*. J MDPI Applied Sciences. Vol. 8: 958 (2018).
- Ostachowicz W, Kudela P, Krawczuk M, Zak A. *Guided Waves in Structures for SHM: The Time-Domain Spectral Element Method*. John Wiley & Sons Ltd., West Sussex (2012).
- Pahlavan PL. *Wave Propagation in Thin-walled Composite Structures: Application to Structural Health Monitoring*. PhD Dissertation. Delft University of Technology (2012).
- Pant S, Laliberte J, Martinez M. *Structural Health Monitoring (SHM) of Composite Aerospace Structures Using Lamb Waves*. Proc. 19th Intl Conf Composite Materials (ICCM), Montreal (2013).
- Pendleton SD, Andersen H, Du X, Shen X, Meghjani M, Eng YH, Rus D, Ang MH. *Perception, Planning, Control, and Coordination for Autonomous Vehicles*. MDPI J Machines. Vol. 5(1): 1-54 (2017).
- Petralia RS, Wenthold RJ. *Neurotransmitters in the Auditory System*. Encyclopedia of Neuroscience (2009).
- Pohl J, Mook G. *Laser-Vibrometric Analysis of Propagation and Interaction of Lamb Waves in CFRP-Plates*. CEAS Aeronautical J. Vol. 4: 77-85 (2013).
- Pramila T, Shukla R, Kishore NN, Raghuram V. *A Study of the Spectral Behavior of Laser-Generated Lamb Waves using Wavelet Transforms*. Proc. 4th Intl Conf on NDT, Chania (2007).
- Quiroga RQ, Reddy L, Kreiman G, Koch C, Fried. *Invariant Visual Representation by Single Neurons in the Human Brain*. Nature Vol. 435: 1102-1107 (2005).
- Raza MS, Qamar U. *A Hybrid Feature Selection Approach Based on Heuristic and Exhaustive Algorithms using Rough Set Theory*. ACM Conf on Internet of Thing and Cloud Computing (ICC 2016, Cambridge (2016).
- Reed S. *Chapter 32: Artificial Neural Network*. In: Encyclopedia of Structural Health Monitoring. John Wiley and Sons Ltd, West Sussex (2009).
- Rizzo P, DiScalea FL. *Feature Extraction for Defect Detection in Strands by Guided Ultrasonic Waves*. Intl J Structural Health Monitoring. Vol. 5(3): 297-308 (2006).
- Rosasco L, De Vito E, Caponnetto A, Piana M, Verri A. *Are Loss Functions All the Same?* J Neural Computation. Vol. 16: 1063-1076 (2003).
- Rose JL. *Ultrasonic Waves in Solid Media*. Cambridge University Press, New York (1999).
- Ruder S. *An Overview of Gradient Descent Optimization Algorithms* (2016). Available: <https://arxiv.org/abs/1609.04747> (Last online: FEB-2021).
- Saladin K. *Human Anatomy*. McGraw-Hill Education Ltd, New York (2016)
- Schmitt LM. *Fundamental Study: Theory of Genetic Algorithms*. J Theoretical Computer Science. Vol. 259: 1-61 (2001).
- Schmitt M, Schmidt K, Olfert S, Rautenberg J, Lindner G, Henning B, Reind LM. *Detection of Coatings within Liquid-Filled Tubes and Containers by Mode Conversion of Leaky Lamb Waves*. J Sensors and Sensor Systems. Vol. 2(1): 73-84 (2013).
- Sethuraman J. *Some Limit Theorems for Joint Distributions*. The Indian J of Statistics: Series A. Vol. 23(4): 379-386 (1961).
- Shi Y, Eberhart R. *A Modified Particle Swarm Optimizer*. Proc. IEEE Intl Conf on Evolutionary Computation Proceedings, Anchorage (1998).
- Shin HJ, Song SJ. *Observation of Lamb Wave Mode Conversion on an Aluminum Plate*. Proc. 15th World Conf for Non-Destructive Testing (WCNDT), Rome (2000).
- Sörensen K, Sevaux M, Glover F. *A History of Metaheuristics*. In: Handbook of Heuristics. Springer, Cham (2018).
- Sörensen K, Glover F. *Metaheuristics*. In: Encyclopedia of Operations Research and Management Science. Springer, Boston (2013).
- Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. J Machine Learning Research. Vol. 15: 1929-1958 (2014).
- Stein O. *What is Continuous Optimization?* Lecture Notes Karlsruhe Inst of Technology (2016). Available: http://kop.ior.kit.edu/downloads/continuous_optimization.pdf (Last online: DEC-2020).

- Stöver T, Diensthuber M. *Molecular Biology of Hearing*. J GMS Current Topics in Otorhinolaryngology - Head and Neck Surgery. Vol. 10: 1-15 (2011).
- Su Z, Ye L. *Identification of Damage Using Lamb Waves: From Fundamentals to Applications*. Springer, Berlin / Heidelberg (2009).
- Sutskever I, Martens J, Dahl G, Hinton G. *On the Importance of Initialization and Momentum in Deep Learning*. Proc 30th Intl Conf on Machine Learning (ICML), Atlanta (2013).
- Tian Z, Yu L. *Lamb Wave Frequency-Wavenumber Analysis and Decomposition*. J Intelligent Material Systems and Structures. Vol. 25(9): 1107-1123 (2014).
- Tian Z, Howden S, Ma Z, Xiao W, Yu L. *Pulsed Laser-Scanning Laser Doppler Vibrometer (PL-SLDV) Phased Arrays for Damage Detection in Aluminum Plates*. J Mechanical Systems and Signal Processing. Vol. 121: 158-170 (2019).
- Valiant LG. *A Theory of the Learnable*. J Communications of the Association of Computing Machinery (ACM). Vol. 27(11): 1134-1142 (1984).
- Valiant LG. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books Inc., New York (2013).
- Vapnik V, Chervonenkis A. *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*. Theory of Probability and its Applications. Vol. 16(2): 264-280 (1971).
- Wallisch P, Lusignan ME, Benayoun MD, Baker TI, Dickey AS, Hatsopoulos NG. *Chapter 36 - Neural Networks Part I: Unsupervised Learning*. In MATLAB for Neuroscientists: An Introduction to Scientific Computing in MATLAB (2nd Ed). Elsevier, London / Waltham / San Diego (2014).
- Wang S, Wang X, Zhao P, Wen W, Kaeli D, Chin P, Lin X. *Defensive Dropout for Hardening Deep Neural Networks under Adversarial Attacks*. Proc. Intl Conf on Computer-Aided Design (ICCAD) San Diego (2018).
- Wang X, Cai J, Zhou Z. *Lamb Wave Signal Reconstruction Method for High-resolution Damage Imaging*. Chinese J of Aeronautics. Vol. 32(5): 1087-1099 (2019).
- Wilcox P, Lowe M, Cawley P. *Omnidirectional Guided Wave Inspection of Large Metallic Plate Structures Using an EMAT Array*. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control. Vol. 52(4): 653-665 (2005).
- Wolpert DH, Macready WG. *No Free Lunch Theorems for Optimization*. IEEE Transactions on Evolutionary Computations. Vol. 1(1): 67-82 (1997).
- Wolpert DH. *A Mathematical Theory of Generalization: Part I*. J Complex Systems. Vol. 4: 151-200 (1990).
- Wolpert DH. *Stacked Generalization*. J Neural Networks. Vol. 5(2): 241-259 (1992).
- Wriggers P. *Nonlinear Finite Element Methods*. Springer-Verlag, Berlin / Heidelberg (2008).
- Xu B, Giurgiutiu V. *Single Mode Tuning Effects on Lamb Wave Time Reversal with Piezoelectric Wafer Active Sensors for Structural Health Monitoring*. J Non-Destructive Evaluation. Vol. 26: 123-134 (2007).
- Zaritsky A, Sipper M. *The Preservation of Favored Building Blocks in the Struggle for Fitness: The Puzzle Algorithm*. IEEE Transactions on Evolutionary Computation. Vol. 8(5): 443-455 (2004).
- Zemmour AI. *The Hilbert-Huang Transform for Damage Detection in Plate Structures*. Master Thesis. University of Maryland (2006).
- Zeiler A. *Adadelta: An Adaptive Learning Rate Method* (2012). Available: <https://arxiv.org/abs/1212.5701> (Last online: JAN-2021)
- Zhang Y, Wang S, Huang S, Zhao W. *Mode Recognition of Lamb Wave Detecting Signals in Metal Plate Using the Hilbert-Huang Transform Method*. J Sensor Technology. Vol. 5(1): 7-14 (2015).
- Zhao X, Gao H, Zhang G, Ayhan B, Yan F, Kwan C, Rose JL. *Active Health Monitoring of an Aircraft Wing with Embedded Piezoelectric Sensor/Actuator Network: I. Defect Detection, Localization and Growth Monitoring*. J Smart Materials and Structures. Vol. 16(4): 1208-1217 (2007).
- Zhao Y, Li F, Cao P, Liu Y, Zhang J, Fu S, Zhang J, Hu N. *Generation Mechanism of Nonlinear Ultrasonic Lamb Waves in Thin Plates with Randomly Distributed Microcracks*. J Ultrasonics. Vol. 79: 60-67 (2017).
- Zienkiewicz OC, Taylor RL, Zhu JZ. *Finite Element Method - Its Basis and Fundamentals* (6th Ed.). Elsevier Butterworth-Heinemann, Oxford / Burlington (2005).
- Zoubi AB, Kim S, Adams DO, Matthews VJ. *Lamb Wave Mode Decomposition Based on Cross-Wigner-Ville Distribution and Its Application to Anomaly Imaging for Structural Health Monitoring*. Vol. 66(5): 984-997 (2019).

4. Deterministic Approach Sensor Placement

This chapter partially contains the work that has been published as:

1. Ewald V, Groves RM, Benedictus R. *Transducer Placement Option of Lamb Wave SHM System for Hotspot Damage Monitoring*. MDPI J Aerospace. Vol. 5: 39 (2018).

Recall the formulated high-level research question in section 2.3.4: *in what way is the incorporation of computational and artificial intelligence as a design tool for automated diagnostics within predictive maintenance feasible?* One of the sub-problems to address this research question is the investigation of the design complexity, particularly on sensor placement using deterministic and semi-stochastic methods according to the purpose of its structural design. This objective can be broken down into two approaches: deterministic and semi-stochastic. Based on the literature review in Chapter 2, following two streams of research in sensor positioning have been identified:

1. Transducer placement for detecting hotspot damage from predictable locations based on fatigue analysis such as a rivet hole crack, and
2. Transducer placement for detecting stochastic damage locations that are independent of fatigue analysis, such as hail impact or tool drop.

As a full description of both approaches would be very lengthy, in this chapter, only the first approach will be described, i.e., when the consideration assumes a known damage location due to the original design of the concerned aircraft sub-structure. In particular, we must consider following aspects [Ewald (2015)]:

1. Crack growth law and critical crack size in a damage-tolerant aircraft sub-structure.
2. Numerical simulation of Lamb wave propagation in a plate-like structure to save experimental time.
3. Data processing that involves signal subtraction of damaged from undamaged baseline structures.

Chapter 4 is organized in following way: Section 4.1 briefly reviews the crack growth assumption within the damage tolerance framework, and in section 4.2, the methodology to model Lamb waves in the Finite Element (FE) environment is discussed. The simulated data processing methodology is given in sections 4.3 and 4.4, while the results and discussion from the applied methodologies are given in Section 4.5. Finally, the conclusion of this chapter is given in Section 4.6.

4.1. Lamb Wave and Crack Growth in Damage Tolerance Structure

To understand deterministic sensor positioning for a predictable damage location, firstly, it is important to understand the concept of damage tolerant design. Recall that the definition of damage tolerance is *“the ability of the structure*

to sustain design limit loads in the presence of damage caused by fatigue, corrosion and other sources until such damage is detected and repaired" [Harris et al. (2003)]. The important key elements in damage tolerant design are:

1. The assumption of initial damage existence,
2. Damage growth in the material due to structural loading, and
3. The critical damage size up to which the structure endures the loading before catastrophic failure.

These three key elements are synchronous to the regions I, II, and III in a da/dN curve [Pugno et al. (2006)], which describes crack propagation rate as a function of stress intensity factor (SIF) range during fatigue cycle (ΔK), shown in Fig. 4.1-1.

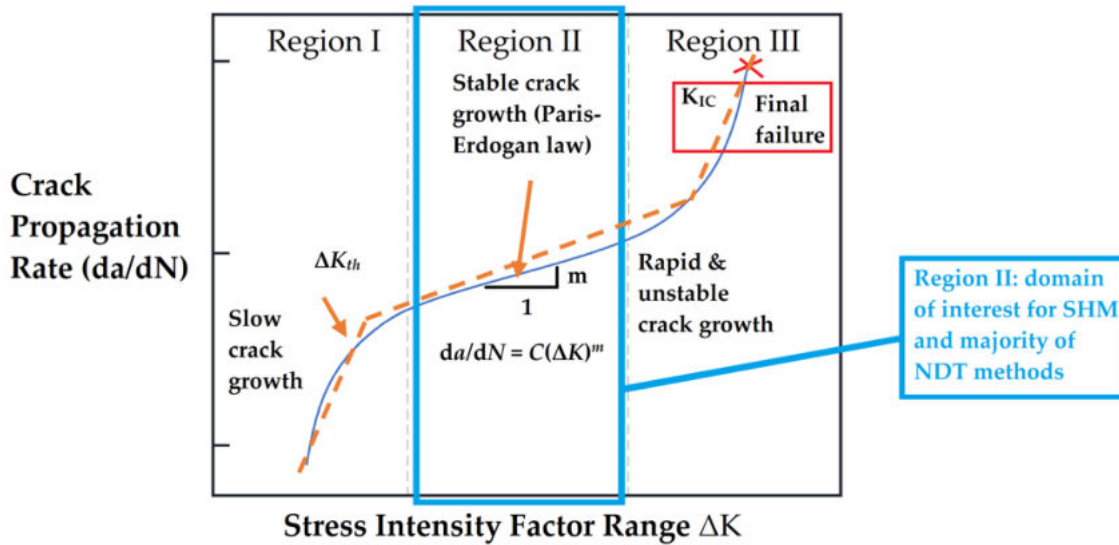


Fig. 4.1-1: Typical da/dN curve. ΔK_{th} is threshold stress intensity factor range, K_{IC} is the critical stress intensity factor in mode I crack propagation, C and m are Paris-Erdogan constants.

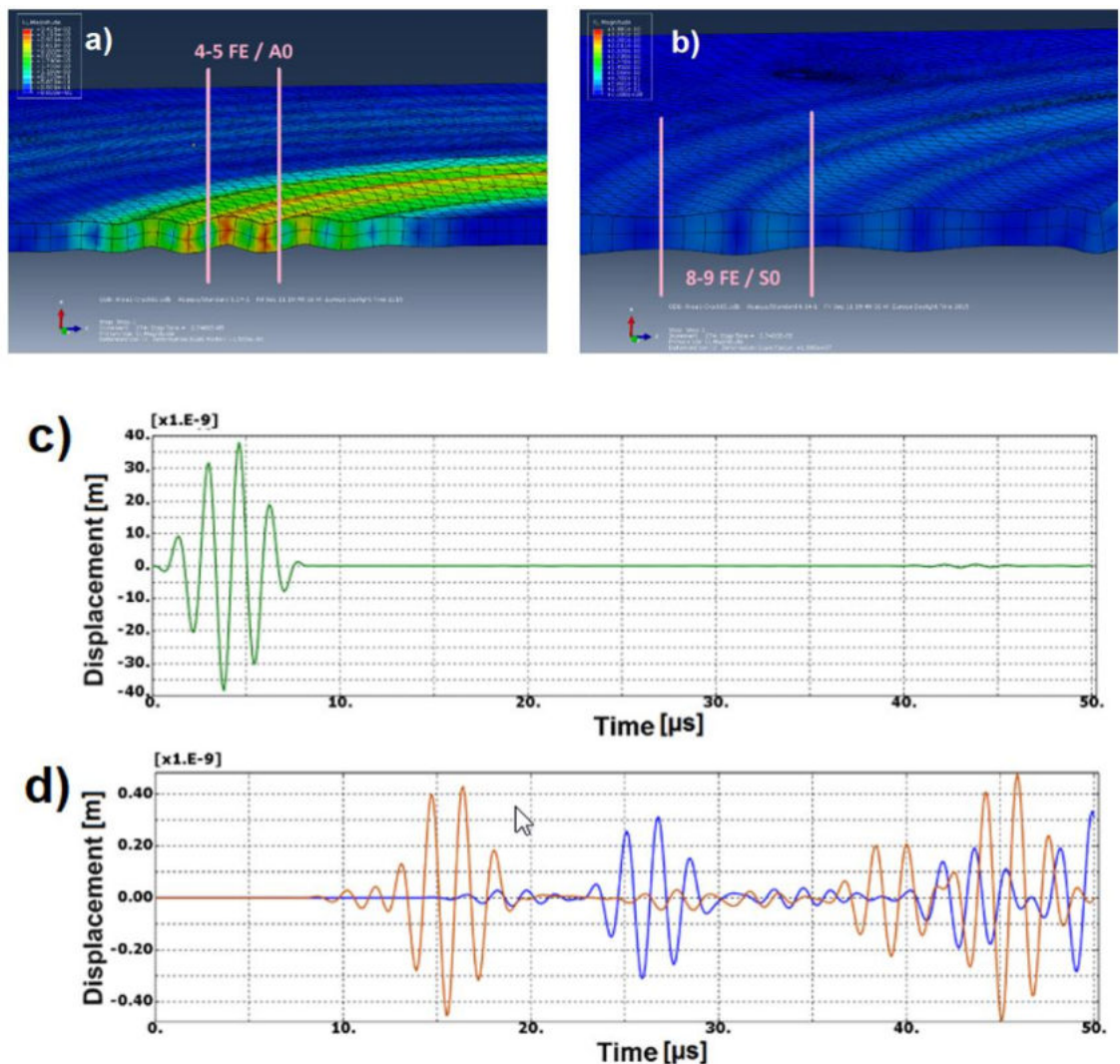
The assumption of initial damage existence falls in region I of Fig. 4.1-1, where the crack growth is typically slow. This region can be covered by some advanced NDT and material characterization techniques such as X-Ray tomography [Schors et al. (2006)] or scanning electron microscopy (SEM) [Wang et al. (2005)]. After passing the threshold SIF range ΔK_{th} , region II begins where the crack propagation rate is stable, and crack growth normally follows the Paris-Erdogan law [Pugno et al. (2003), Milella (2013)]. In this region, the crack becomes much larger and sometimes can be seen with naked eye [McMaster (2010)]. This is the domain of interest for SHM and the majority of NDT methods. The critical crack length a_c is typically connected with the critical stress intensity factor K_{IC} . After transition between region II and III, the crack propagation rate becomes rapid and unstable until final failure.

Lamb waves can interact with damage which has a size at least the half of its wavelength [Gilchrist (1999), Wang and Su (2014)], and since the critical damage tolerant size (which can be up to several hundred millimeters [Harris et al. (2003)]) is generally larger than the wavelength (which is typically less than 100 mm), it is safe to assume that Lamb waves can also interact with the critical

damage. The SIF is generally higher in the area around the notch; thus, one can expect that a fatigue crack will initiate around a notch. For example, the changes in cabin pressure in an aircraft fuselage can be approximated by the internal pressure in a thin-walled cylinder where the hoop stress is two times larger than the axial stress [Richard and Sander (2008)]. Therefore, the crack orientation is expected to be orthogonal to the direction of the hoop stress. By knowing the most probable damage orientation and location and the critical damage size for a certain geometry, two FE simulation scenarios of wave propagation can be performed: 1). Lamb wave propagation in an undamaged structure as a baseline and 2). Lamb wave propagation in a critically damaged structure.

4.2. Simulation of Lamb Wave Propagation with ABAQUS FE

The theoretical foundation of simulated Lamb wave has been given in Section 3.1.2. To reliably model an ultrasonic signal, the recommended number of mesh elements is 4 elements per A_0 - or 8 per S_0 -wavelength [Ewald (2015)]. The method for choosing the simulation parameters has been described in [Zienkiewicz et al. (2005)]. An example screen capture of Lamb wave propagation in ABAQUS simulation software is depicted in Fig. 4.2-1a-b, where a typical excitation signal is given Fig. 4.2-1c, and its response is given in Fig. 4.2-1d. Their respective frequency domain signals are given in Fig. 4.2.1e-f, respectively.



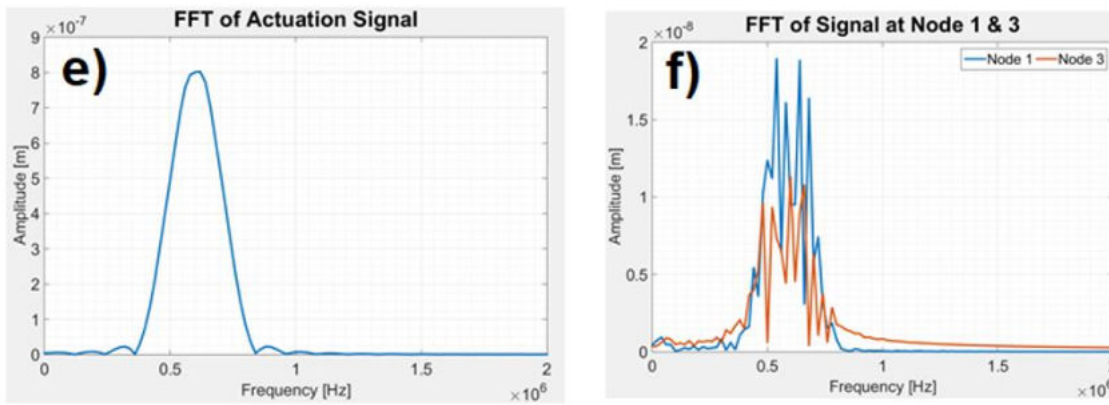


Fig. 4.2-1: Example of simulated Lamb wave propagation in ABAQUS FE for a). A_0 -mode and b). S_0 -mode. The excitation signal and its responses captured in different sensing nodes (red and blue lines) are depicted in c) and d), respectively. The frequency domain responses by using FFTs of c) and d) are given in e) and f), respectively.

For this work, the test object is an aluminum plate with a dimension of 600 mm x 400 mm x 2 mm. The following parameters were chosen based on the study conducted in the previous work [Ewald (2015)]: ABAQUS explicit, generic aluminum properties (Young's modulus of 70 GPa, Poisson ratio of 0.33, density of 2700 kg/m³), quadratic brick mesh (C3D20) with a global mesh size of 1 mm, single node out-of-plane excitation with windowed 5 sine-cycle of central frequency of 250 kHz with 1N concentrated force, dynamic implicit step, no boundary conditions imposed, time increment of 0.1 μ s (i.e., the sampling frequency is 10 MHz), total time period of 500 μ s and a single nodal output precision. The crack can simply be modelled as an elliptical material discontinuity.

The specification of the computer where the simulation was run was: Intel Xeon E5-1620 3.5 GHz (Quad-core 8-Threads), 32 GB DDR3-RAM, and NVidia NVS310M Graphic card (GPU acceleration was not activated). The wave propagation image can be later captured with automated script. The color vector of a single pixel is normally represented as an RGB array and can be used to represent different Lamb wave displacement amplitudes. The average displacement U is defined as:

$$U = \sqrt{u_x^2 + u_y^2 + u_z^2} \quad (4.2-1)$$

where u_x , u_y , and u_z are the displacements in x, y, and z-directions, respectively. Fig. 4.2-2 shows an example simulation of Lamb wave propagation 30 μ s after excitation. No displacement ($U = 0$ nm) is shown as a blue pixel, while a displacement of 2.5 nm is shown in green, and a displacement of 5 nm is shown in red. The values in-between such as 1.25 nm and 3.75 nm are shown in cyan and yellow, respectively.

This colormap 'rainbow' is the default colormap in ABAQUS. Note that this colormap is slightly different from the basic 3-bit RGB colormap depicted in Fig. 4.2-2 as it has more color transitions, i.e., there are smoother transitions between blue and cyan, cyan and green, and so on.

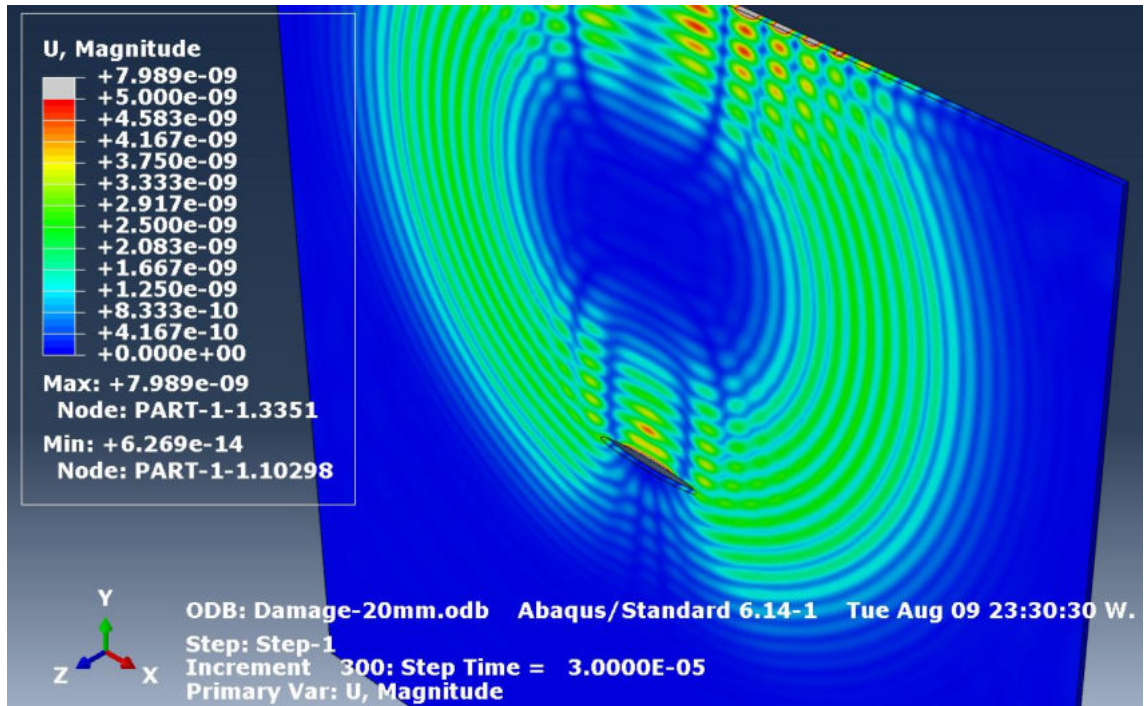


Fig. 4.2.-2: Lamb wave propagation at $30 \mu\text{s}$ after excitation in an Al-7075-T6 plate with dimensions of $200 \text{ mm} \times 200 \text{ mm} \times 1 \text{ mm}$ (shown in ABAQUS GUI Viewer)

While this image processing procedure offers less displacement information as all displacement values are translated into an RGB array, this alternative procedure much faster, and more memory efficient rather than extracting data directly from the ABAQUS ODB binary file.

4.3. Image Processing

The Lamb wave propagation at a certain moment in the ABAQUS Viewer can be captured as an image. Fig. 4.3-2a-b shows Lamb wave propagation at $t = 100 \mu\text{s}$ and $125 \mu\text{s}$ in an undamaged Al-7075 plate with 3 rivet holes, respectively. These are the baseline images. As can be seen from these figures, the wave reflection from the rivet hole is minimal, thus only showing a minimal amount of wave ripples. The scale of displacement magnitude is the same as depicted in Fig. 4.2-2.

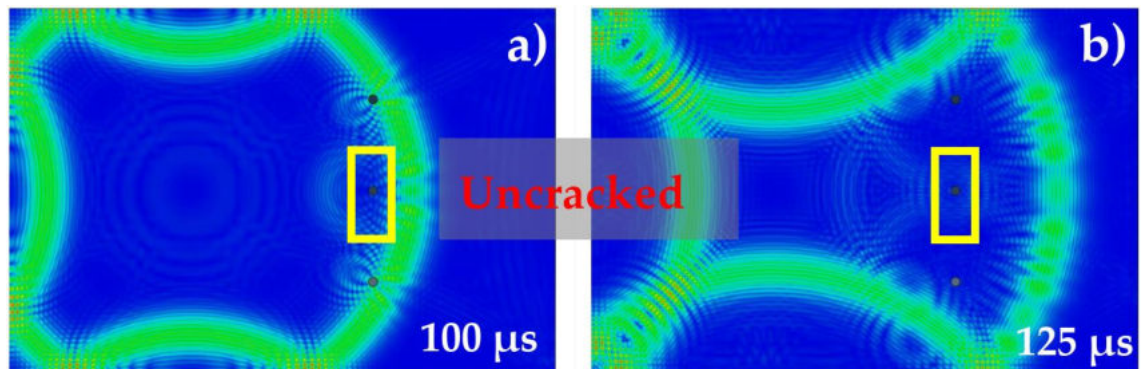


Figure 4.3-2: Lamb wave propagation at a) $t = 100$ and b) $125 \mu\text{s}$ in uncracked Al 7075 Plate.

Fig. 4.3-3a-b show the wave propagation in the same plate but with a symmetric crack (from tip-to-tip, including the hole diameter of 10 mm) of 28 mm length in the middle of the plate (marked by a yellow rectangle). The images captured have

size of a 1210 x 807 pixels, so the resolution is 2 pixel/mm. These images are stored as an array with a size of 1210 x 807 x 3, where each pixel has 3 arrays, each containing a normalized floating value between 0 and 1 for each of the RGB colors. A similar pattern of Lamb wave propagation can be observed if the crack length differs by +/- 10%, as shown in Fig. 4.3-3c-d. In this case, the crack length is 30 mm instead of 28 mm. However, if the crack is much larger, a notable change in the wave propagation pattern can be observed, as depicted in Fig. 4.3-3e-f. In this case, the crack length is 60 mm.

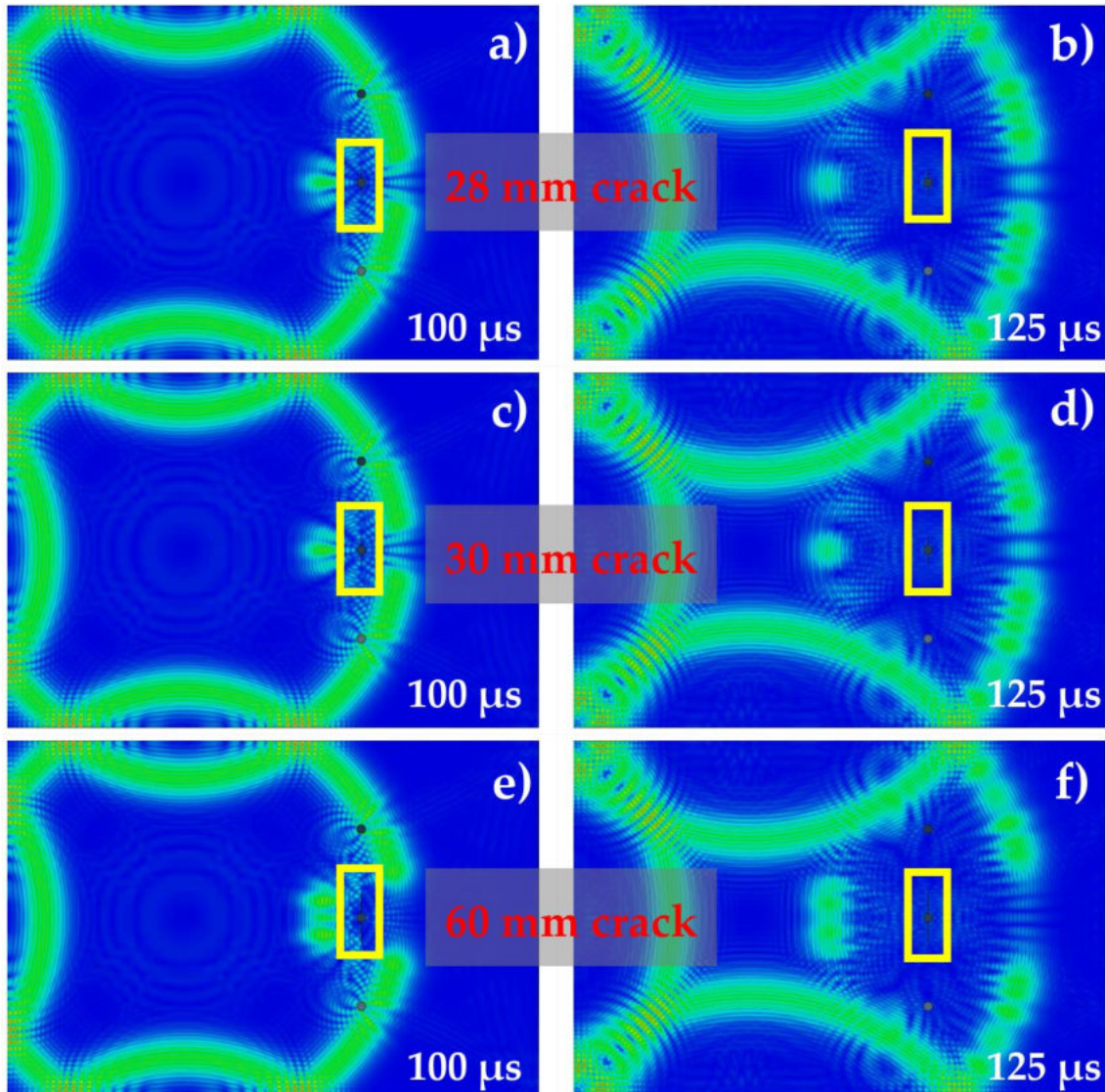


Figure 4.3-3: Lamb wave propagation at a). $t = 100$ and b). $125 \mu\text{s}$ in an Al 7075 Plate with a 28 mm crack. In a similar way, the wave propagation for an Al-7075 Plate with 30 mm and 60 mm cracks at $t = 100 \mu\text{s}$ and $t = 125 \mu\text{s}$ are depicted in c - f, respectively.

By subtracting the image of the cracked plate Fig. 4.3-3e from the baseline images in Fig. 4.3-2b, the reflected wave scatter image can be obtained, as shown in Fig. 4.3-4a. In an analogous way, we can easily subtract Fig. 4.3-2b from Fig. 4.3-2f. Note the RGB values are subtracted rather than the displacement values. Pixels, for which there is no change in wave scatter are shown as black. In Fig. 4.3-4a, the reflected wave scatter is highlighted by the yellow rectangle, while the distorted transmitted wave is obtained as well (in the red rectangle) but is not clearly visible.

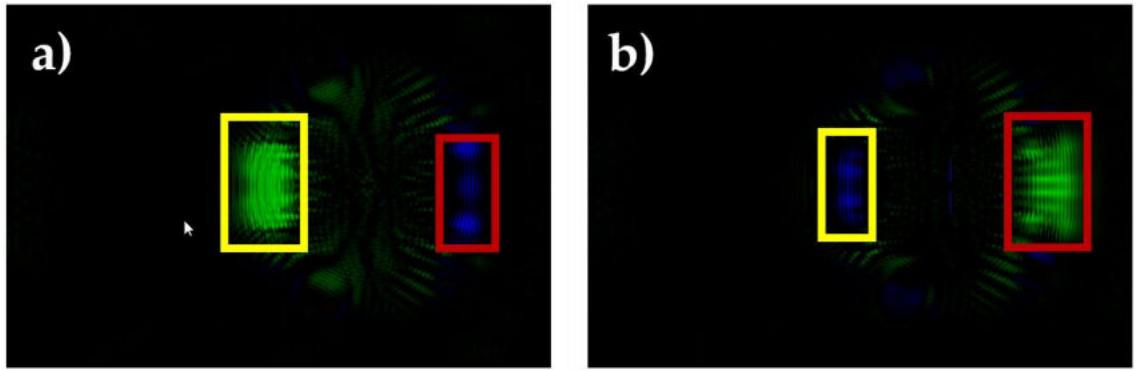


Fig. 4.3-4: Differential image of a). Fig. (4.3-3h - 4.3-2b) and b). Fig. (4.3-2b - 4.3-2h)

For the further sections, only the results from the uncracked plate and the cracked plate of 60 mm crack will be shown for conciseness. Fig. 4.3-4b can be explained in the same manner: the corruptly transmitted wave scatter image is highlighted more (red rectangle), while the reflected wave scatter can still be seen (yellow rectangle) but is less visible. To highlight both the reflected and corrupted wave scatter, the two images shown in Fig. 4.3-4a - b can be joined to form a composite image, which can be either created either via addition or image fusion. In image addition, both images are simply added to each other. The second one is called image fusion and uses the '*imfusion*' function in MATLAB, where the two images are firstly converted into greyscale mode, given a false color and then added mathematically. An example of image addition and image fusion are depicted in Fig. 4.3-5a - b, respectively. For the sensor placement procedure described in Section 4.4, the inverse fused image (Fig. 4.3-5b) is used so we can still see the representation of reflected and missing wave scatter.

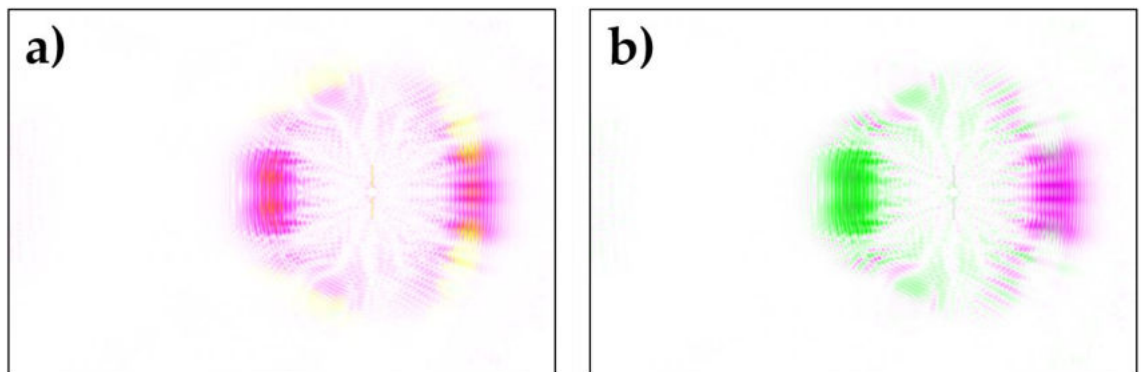


Fig. 4.3-5: a). Inverse of a). added image and b). fused images of Fig. 4.3-4.

4.4. Blob Detection

The wave scatter which was caused by both reflections and corrupted transmissions due to the crack front are represented by false color (i.e., green and magenta pixels in Fig. 4.3-5b). These pixels have a different color from the background (white). The region of interest is determined by using the MATLAB blob detection function to locate areas of adjacent green and magenta pixels. The larger the blob is, the larger the area of wave scatter. The sensor should be placed in the centroid of the largest blob, so that it will have a high probability of capturing a portion of wave scatter from cracks.

The blob detection algorithm is based on the Laplacian of Gaussian [MATLAB IPT Documentation, Kong et al. (2013), Lindeberg (1998)] with an 8-pixel connectivity kernel. The Gaussian function G of an input image $f(x,y)$ and feature scaling σ is defined as:

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (4.4-1)$$

The Laplacian operator ∇^2 is defined as:

$$\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (4.4-2)$$

By applying the Laplacian operator in Eq. (4.4-2) to the Gaussian function in Eq. (4.4-1), one obtains the Laplacian of Gaussian (LoG), as described in Eq. (4.4-3). Concretely, LoG is the edge of the blob and for this reason, many edge detection algorithm problems rely on LoG.

$$\nabla^2 G(x, y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\pi\sigma^4}\right) \cdot \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (4.4-3)$$

Recall the definition of continuous optimization described in Section 3.3.1. Since the Gaussian function is a quasi-concave, it is continuously differentiable in \mathbb{R} . Within a blob, there are several anchors point candidates. The right centroid of the blob is found when the LoG reaches the maximum. Hence, the blob centroid x_c, y_c with the scale σ_c is the simultaneous local minimum of the LoG. The objective function to search the best centroid within the blob can be thus formulated as:

$$(x_c, y_c, \sigma_c) = \arg \min \max_{(x,y,\sigma)} (\nabla^2 G(x, y)) \quad (4.4-4)$$

For the blob detection, an 8-pixel connectivity kernel (Fig. 4.4-1a) is used because it is more suitable for a larger area since the diagonal neighbor is counted as well, while a 4-pixel connectivity kernel (Fig. 4.4-1b) is typically used for line and corner detection. In Fig. 4.4-1a - b, the meaning of -1 and 0 are pixels which are counted and are not counted as a neighbor of the center pixel, respectively.

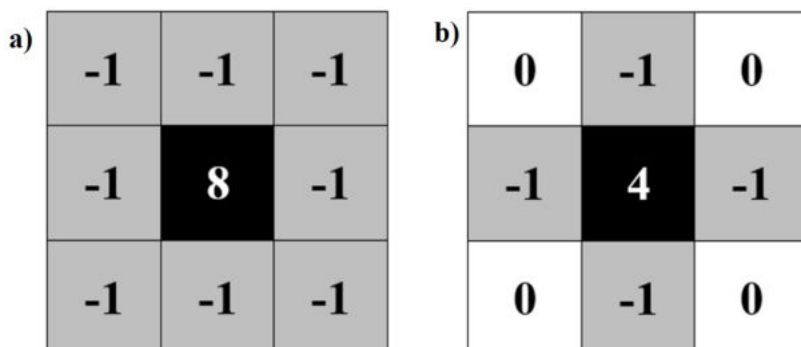


Fig. 4.4-1: Convolutional kernel of a). 8-pixel connectivity and b). 4 -pixel connectivity within 3x3 convolution.

The blob detections from various time increments are depicted in Fig. 4.4-2a-d. As mentioned before, the sensor should be placed in the location with the largest change in signal over time, i.e., the largest blob. Since it generally contains more than a single pixel, the centroid can be calculated to determine the average pixel location that would receive wave scatter. In Fig. 4.4-2a-d, the largest and the second-largest blob centroids are marked by red and green dots, respectively. Blob boundaries are marked by the yellow polylines.

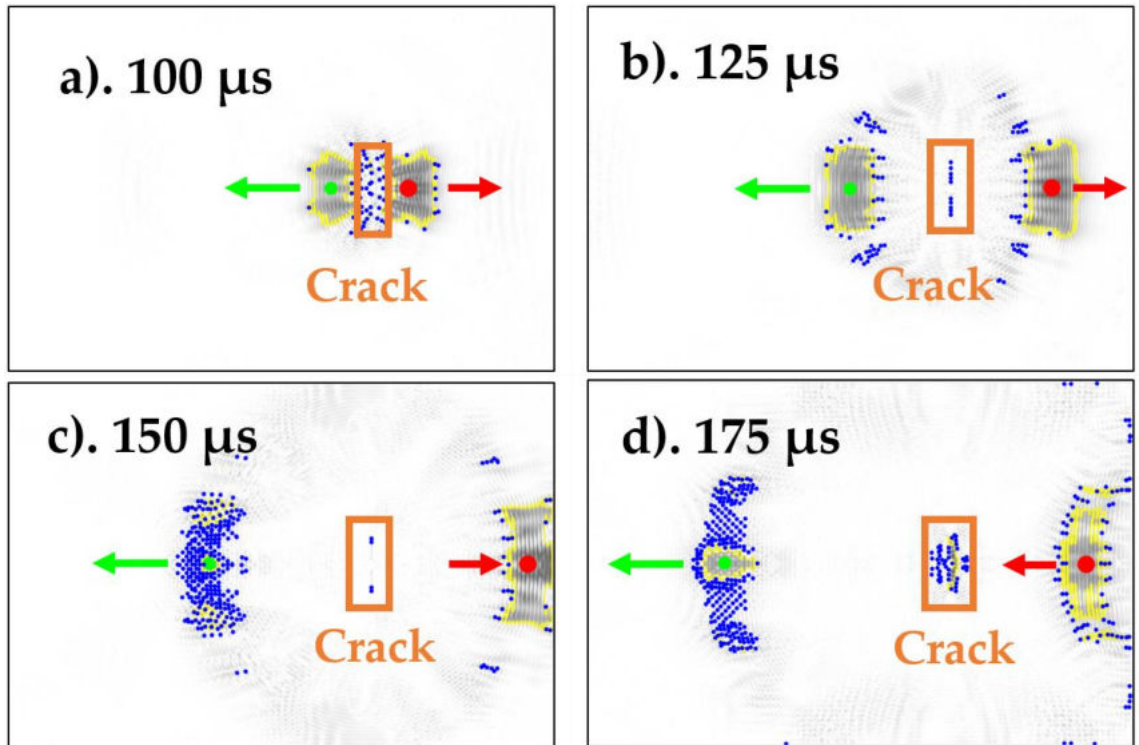


Fig. 4.4-2: Detected blobs at a). 100 μ s, b). 125 μ s, c). 150 μ s, and d). 175 μ s. The largest and second-largest blobs are marked in red and green, respectively. Arrows indicate the direction of movement of the blobs.

The rest of the centroids are marked by blue dots. In order not to lose the overview, the reader is encouraged to compare Fig. 4.4-2a with Fig. 4.3-2a and 4.3-3e, as well as Fig. 4.4-2b with Fig. 4.3-2b and 4.3-3f. The X-Y coordinates of the blob centroid and the blob size are summarized in Table 4.4-1.

Time Frame	Largest centroid			Second-largest centroid		
	Coord. [pixel]	Coord. [mm]	Area [pixel]	Coord. [pixel]	Coord. [mm]	Area [pixel]
100 μ s	888;403	440;200	15910	716;403	355;200	12917
125 μ s	1031;402	511;199	29535	582;404	289;200	18794
150 μ s	1154;402	572;199	27067	445;401	221;200	8808
175 μ s	1111;405	551;201	24214	304;402	151;199	7949

Table 4.4-1: Area and coordinates of the largest and second largest centroid. Units are in pixel and mm. Total area is in pixel. Average resolution is 2 pixel / mm.

After a certain time, the wave pattern becomes more chaotic due to multiple reflections from the crack front, rivet holes, and plate edges so that more smaller

centroids will be born that are not exactly aligned with the mid Y-axis anymore. The detailed procedure in MATLAB to find to trace the blob is described in Algorithm 4.4-1.

Algorithm 4.4-1. Blob Detection.

Input parameter: S – set of images, Output: X, Y – Blob centroid coordinates

Setup initial values:

$i \leftarrow$ Number of available images

$n_0 \leftarrow$ Number of detected blobs

Until termination condition is reached:

Do {
 Convert the RGB matrix S_i into greyscale array I_i
 Set the intensity threshold $I_i < \text{threshold}$
 Trace region with 8-pixel connectivity B_i^8
 Update $n \leftarrow n(B_i^8)$

Until termination condition is reached:

Do {
 Store area information in $A_n(B_i^8)$
 Sort blob size such that $A_n(B_i^8) > A_{n+1}(B_i^8) \forall n$
 Store blob boundaries $R_n(B_i^8)$
 Calculate blob centroids $C_n = \text{mean}(R_n(B_i^8))$
 Update centroid coordinate:
 $X \leftarrow X_{C_n}$
 $Y \leftarrow Y_{C_n}$

The smaller centroids imply that the potential energy capture by PZT is getting smaller and this will be aggravated by Lamb wave attenuation. This is the reason it is recommended to have ‘early wave scatter capture’ for hotspot SHM design. Typically, one can decide the best sensor position by considering the movement of the centroid per time increment, also known as ray tracing [Heinze et al. (2014)].

4.5. Results and Discussion

In order to get a better overview, Fig. 4.4-2a – d was fused into a single image. An example result of this operation is depicted in Fig. 4.5.1, where each pixel contains information about the normalized intensity between 0 and 1 which is then mapped into the rainbow color scale. From Fig. 4.5.1, it can be subjectively judged that the best sensor position is between $X = 44$ cm and 57 cm and the second-best sensor position is between and $X = 34$ and 38 cm, while the vertical coordinate for both positions remain at $Y = 20$ cm.

In order to demonstrate that the image processing algorithm also works for a different case of simulations, the case can slightly be modified the case for the a). the critical crack length to 30 mm, and b). the orientation of the crack by 8° . The whole procedure was repeated, and the results for the mentioned cases are

depicted in Fig. 4.5.2 and 4.5.3, respectively. From Fig. 4.5.2, it can be seen that the areas with higher pixel intensity (colored in red with value between 0.7 and 1) are smaller than those of Fig. 4.5.1. This is to be expected since the wave perturbation at the crack front due to a crack length of 30 mm is smaller than those of 60 mm.

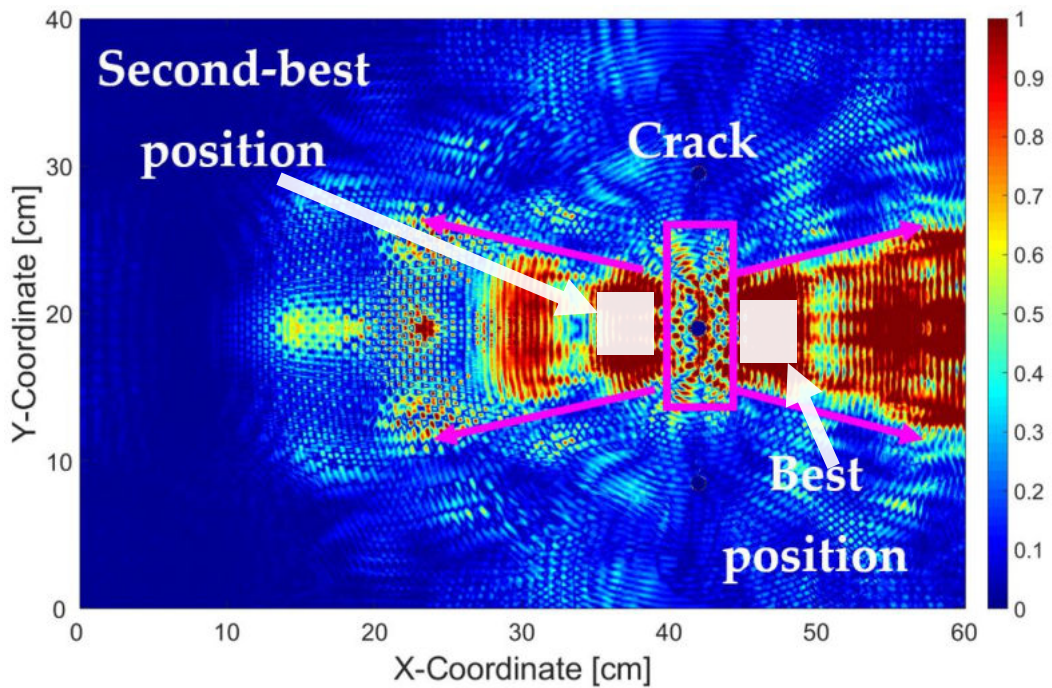


Figure 4.5.1: Fused image of Fig. 4.4.2a-d. The normalized intensity value is encoded between 0 and 1 and is therefore unitless. Intensity value of 0 indicates no residual wave scatter, while intensity value of 1 indicates the maximum residual wave scatter.

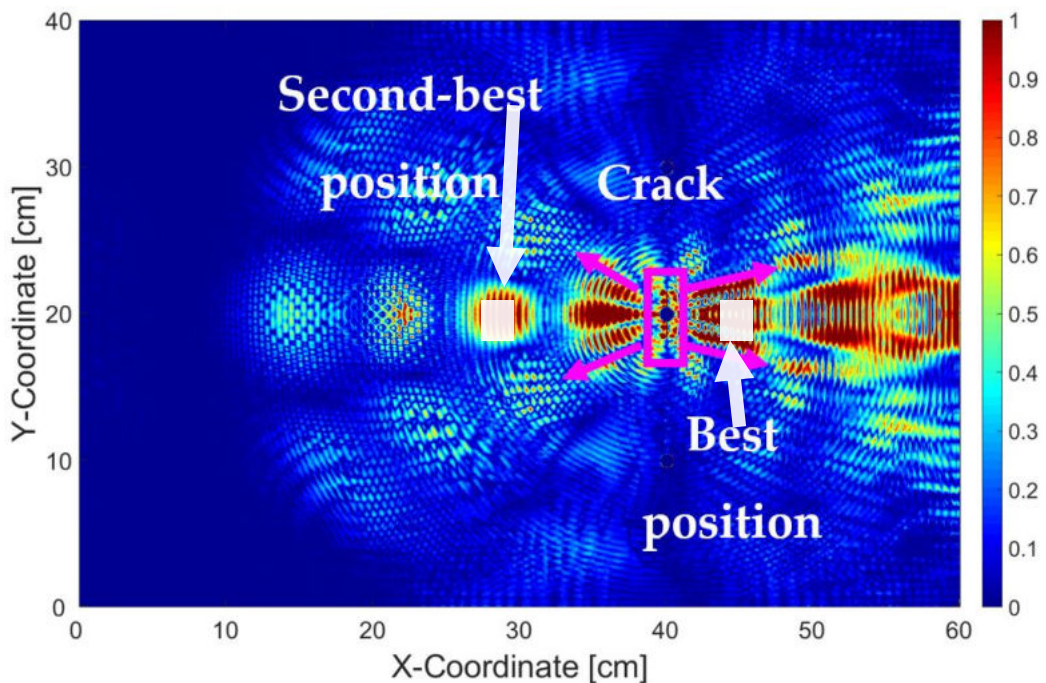


Figure 4.5.2: Fused image of differential images of a 30 mm crack. Intensity value convention from Fig. 4.5.1 applies.

Meanwhile from Fig. 4.5.3, it can be seen that the crack orientation changes the direction of the reflected wave scatter, and it is still conforming with Snell's law. However, it does not change the orientation of areas where the wave scatter is not

present. After cross validating with the original simulation data, it can be confirmed that angled crack only heavily influences the reflected wave scatter portion, but not the missing scatter portion (Fig. 4.5.4a-b, marked in yellow).

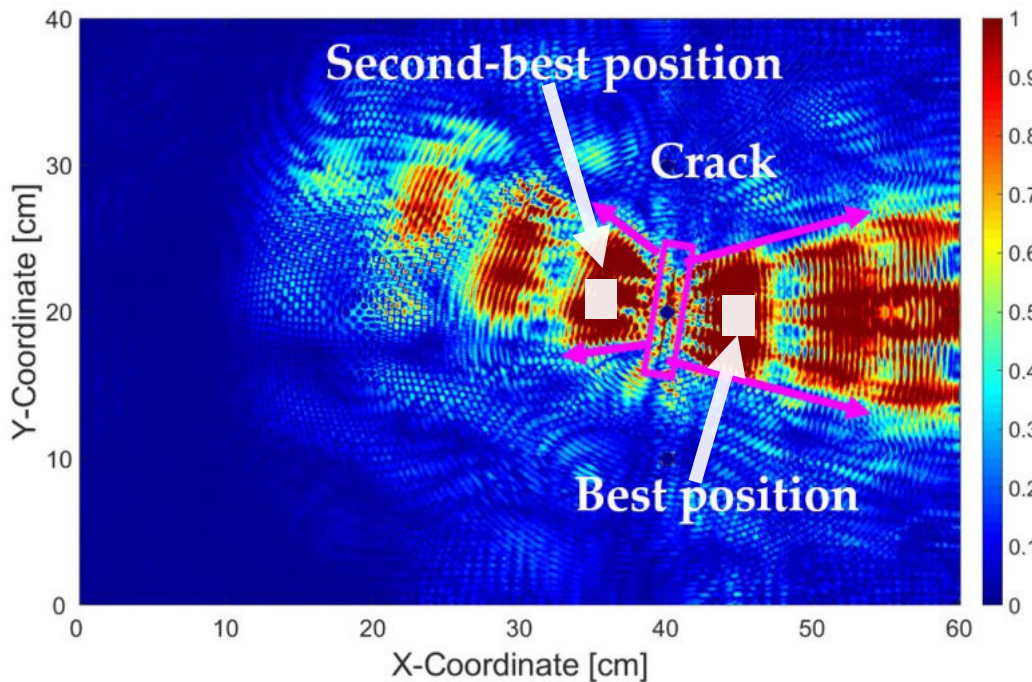


Figure 4.5.3: Fused image of differential images of 60 mm crack with 8° orientation. Intensity value convention from Fig. 4.5.1 applies.

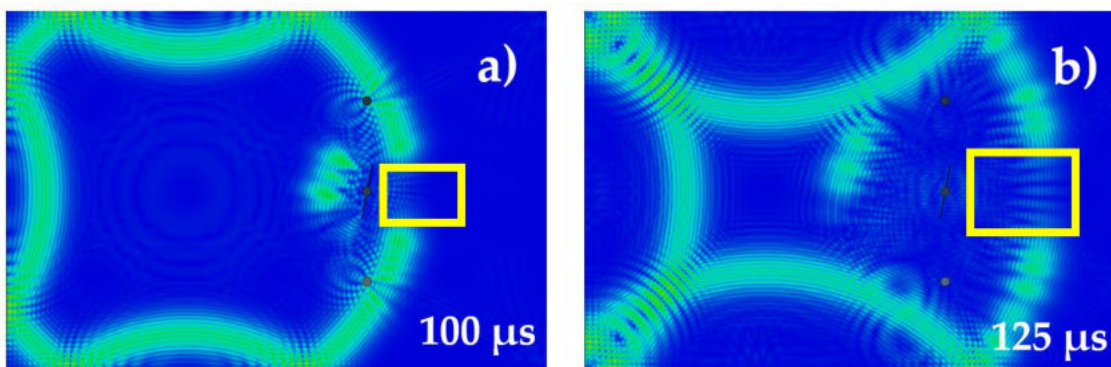


Figure 4.5.4: Lamb wave propagation at a) $t = 100$ and b) $125 \mu\text{s}$ in Al 7075 Plate

In this work, two best locations for placing the sensor are given. However, the number of sensors that are allocated for every case can be adjusted according to the manufacturer and/or aircraft operator requirement to achieve the required crack detectability. Furthermore, not only the number of sensors, but also the excitation frequency can be changed depending on the size of the critical crack that must be detected. A higher frequency means a shorter wavelength, thus enabling the wave to interact with a smaller critical crack length.

It is to be noted that such a higher frequency Lamb wave will also be more quickly attenuated than the lower frequency Lamb wave. Therefore, in order to stabilize the SHM network performance, more sensors will be required. Nevertheless, when more sensors are employed, higher procurement costs are also expected due to more weight, more data processing capability, etc. This can be regarded as

the classical trade-off between SHM investment cost and SHM network reliability and the decision should be passed back to the aircraft manufacturer or operator according to their needs.

4.6. Conclusion

This work demonstrated a novel technique to design the sensor network topology for hotspot SHM by using differential images and a blob detection algorithm. While my image processing technique does not allow a quantitative approach to observe the nodal displacement, i.e., displacement from every single FE node, I believe this technique offers a more holistic view (Fig. 4.5.1 - 4.5.3) of where to place the PZT sensors on the structure to be monitored.

Also, with this technique I believe that the sensor placement can be done more quickly without exhaustive data processing from simulation files for each surface while not sacrificing too much spatial resolution. In practice, even the extracted nodal data from the simulation must be interpolated, since in reality a PZT sensor would always occupy more than a single node (e.g., a typical PZT in our lab has a diameter of 1 cm, so would theoretically occupy about 78 FE nodes). Therefore, as concluding remark, hopefully this technique will help further research in sensor placement.

Chapter 2 stated that the main research problem formulation of this dissertation is to *investigate the feasibility of incorporating computational and artificial intelligence as a design tool for an automated diagnostic within predictive maintenance - and if so, in what way certainly?* Essentially, to connect the discussion in this chapter to the feasibility, now we need to recall the sub-research question defined in section 2.3.4:

The design complexity and parameter optimization, particularly on sensor placement methodologies for both deterministic and semi-stochastic approaches according to what extent the structure is designed based on the premise that sensor network topology affects the damage detection capability and the overall SHM performance. In this proposal, an investigation of different sensor network topologies is needed to understand the trade-off between the strategies and if possible, to propose a compensation technique.

In this chapter, the deterministic methodology for transducer placement for known damage location based on continuous optimization described in Section 3.3.1 is used. While the methodology described in this chapter can easily be repeated for many different types of geometry and material properties to partially answer the design question on deterministic sensor placement, there are several issues that still need to be addressed:

1. Experimental validation of the deterministic design of the sensor topology as in this chapter, only the simulation data from ABAQUS FE is used.

2. Stochastic strategy for non-predictable damage location, especially for quasi-instantaneous but abrupt-like events such as bird and/or lightning strike, tool drop, and hail impact.
3. Bimodal utilization of deterministic and stochastic information of the network topology, i.e., when concerning the integration of both approaches since both damage type, be it on predictable or non-predictable location are very likely to appear during aircraft operational lifetime.

As a short conclusion, the three above mentioned topics are the starting point of the next chapter.

Literatures

- Ewald V. *Post-Design Damage Tolerance Enhancement of Primary Aircraft Structures by Ultrasonic Lamb Wave Based Structural Health Monitoring (SHM) System*. MSc Thesis. Universität des Saarlandes, Saarbrücken (2015).
- Gilchrist MD. *Attenuation of Ultrasonic Rayleigh-Lamb Waves by A Symmetrical Embedded Crack in an Elastic Plate*. J Applied Mathematics and Mechanics. Vol. 79: pp 497 - 498 (1999).
- Harris CE, Starnes Jr JH, Shuart MJ. *Advanced Durability and Damage Tolerance Design and Analysis Methods for Composite Structures - Lesson Learned from NASA Technology Development Programs*. NASA Langley Research Center, Hampton (2003).
- Heinze C, Sinapius M, Wierach P. *Lamb Wave Propagation in Complex Geometries - Model Reduction with Aroximated Stiffeners*. 7th European Workshop on Structural Health Monitoring (EWSHM), Nantes (2014).
- Kong H, Akakin HC, Sarma SE. *A Generalized Laplacian of Gaussian Filter for Blob Detection and its Alications*. IEEE Transactions on Cybernetics. Vol. 43: 1719-1733 (2013).
- Lindeberg T. *Feature Detection with Automatic Scale Selection*. Technical report ISRN KTH/NA/P-96/18-SE. Kungliga Tekniska Högskolan (KTH), Stockholm (1998).
- MATLAB IPT Documentation. Available: <https://de.mathworks.com/help/images/> (Last online: 03-FEB-2018)
- McMaster RC. *Volume 9: Visual Testing (VT)*. In *Nondestructive Testing Handbook* (3rd Ed.). American Society for Nondestructive Testing (ASNT), Columbus (2010).
- Milella PP. *Fatigue and Corrosion in Metals*. Springer, Milan / Heidelberg / New York / Dordrecht / London (2013).
- Pugno N, Ciavarella M, Cornetti P, Carpinteri A. *A Generalized Paris' Law for Fatigue Crack Growth*. J Mechanics and Physics of Solids. Vol. 54: 1333-1349 (2006).
- Richard HA, Sander M. *Technische Mechanik: Festigkeitslehre* (2. Auflage). Vieweg+Teubner, Wiesbaden (2008).
- Schors J, Harbich KW, Hentschel MP, Lange A. *Non-Destructive Micro Crack Detection in Modern Materials*. Proc. 9th European Conf of Non-Destructive Testing (ECNDT), Berlin (2006).
- Wang Q, Su Z. *Crack Diagnosis and Monitoring Method Using Linear-Phased PZT Sensor Array*. Proc. 2nd Intl Conf of Structural Health Monitoring and Integrity Management (ICSHMIM), Nanjing, (2014).
- Wang XS, Wua BS, Wang QY. *Online SEM Investigation of Microcrack Characteristics of Concretes at Various Temperatures*. J Cement and Concrete Research. Vol. 35: pp 1385-1390 (2005).

5. Holistic Sensor Network Topology Optimization

This chapter partially contains the work that has been published in:

1. Ewald V, Groves RM, Benedictus R. *Integrative Approach for Transducer Positioning Optimization for Ultrasonic Structural Health Monitoring for the Detection of Deterministic and Probabilistic Damage Location*. Intl J of Structural Health Monitoring. DOI: 10.1177/1475921720933172 (2020).

In chapter 4 of this dissertation, we saw the deterministic sensor placement methodology. In the last section of chapter 4, there were still several issues that still need to be addressed:

1. Experimental validation of the deterministic design of the sensor topology as in this chapter, only the simulation data from ABAQUS FE is used.
2. Stochastic strategy for non-predictable damage location, especially for quasi-instantaneous but abrupt-like events such as bird and/or lightning strike, tool drop, and hail impact.
3. Bimodal utilization of deterministic and stochastic of the network topology, i.e., when concerning the integration of both approaches since both damage type, be it on predictable or non-predictable location are very likely to appear during aircraft operational lifetime.

This chapter will address these issues one by one. Before going deeper into the strategy, we should take the influencing physical parameters into account when determining the objective function. This will be discussed in section 5.1. The stochastic strategy for sensor placement is described in section 5.2, while the bimodal topology is described in section 5.3 and the preliminary result are given in section 5.4. Section 5.5 gives the experimental validation of sections 5.2 and 5.3 and finally, the conclusion and summary are given in section 5.6.

5.1. Fitness Function

As explained in [Ewald et al. (2020)], the deterministic approach would require too many simulations, and is computationally unfeasible, a situation that can be related to the art gallery problem [O'Rourke (1987)]. Thus, it would be useful to maximize the sensor coverage area to detect damages that occur within that coverage area. Hence, I propose a target function that describes the attenuation at a certain location in the propagation space of the Lamb wave. First, consider the measured signal power P in an infinite plate at point x where the original excitation signal power is P_0 [Ono and Gallego (2012)] with the geometrical attenuation factor α which is proportional to $1/\sqrt{r}$ [Su and Ye (2009), Mizutani et al. (2014), Schubert and Herrmann (2011), Dentith and Mudge (2014), Kerber et al. (2010)], where r is the distance from wavefront to the point x :

$$P = P_0 \cdot \alpha \cdot \exp(-\beta \cdot r) \quad \text{where} \quad \alpha \propto \frac{1}{\sqrt{r}} \quad (5.1-1)$$

The material attenuation β depends on frequency and thickness, e.g., for a 1-mm thick aluminum plate, the attenuation coefficient is between 2.2 - 17 dB/m for a frequency between 0.5 - 5 MHz [Ono and Gallego (2012), Kasama et al. (2000)]. For a given coordinate (x_i, y_j) , we can construct an effective travel distance assigned in pixel value $f(r_{ij})$ by multiplying the total attenuation $[\alpha_{ij} \cdot \exp(-\beta \cdot r_{ij})]$ by the propagating distance r_{ij} from the wave propagation source so that it is comparable to a measured Lamb wave signal amplitude attenuation profile:

$$\begin{aligned} P &= P_0 \cdot \alpha \cdot \exp(-\beta \cdot r) \quad \text{where} \quad \alpha \propto \frac{1}{\sqrt{r}} \\ f(r_{ij}) &= r_{ij} \cdot [\alpha_{ij} \cdot \exp(-\beta \cdot r_{ij})] \quad \text{where} \quad \alpha_{ij} \propto \frac{1}{\sqrt{r_{ij}}} \end{aligned} \quad (5.1-2)$$

Where the distance r_{ij} is defined as the Euclidian distance from the wave propagation source at coordinate (x_i, y_j) up to an arbitrary pixel located in coordinate (\hat{x}_i, \hat{y}_j) and α_{ij} is the dimensionless geometric spreading correction factor at the distance r_{ij} , respectively:

$$r_{ij} = \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_j - y_j)^2} \quad (5.1-3)$$

Consider a structural inhomogeneity such as rivet hole at the coordinate $(\tilde{x}_i, \tilde{y}_j)$ that acts as secondary source since a Lamb wavefront is scattered at the rivet holes, where in my assumption, the scattering occurring at the rivet is considered lossless. A scattering efficiency could be included in the calculation if a reliable value is available. The secondary source emits a lower energy as the waves have lost energy in travelling from the source PZT to the rivet hole via the indirect path \tilde{r}_{ij} defined by:

$$\tilde{r}_{ij} = r_{\text{PZT-rivet}} + r_{\text{rivet-pixel}} = \sqrt{(\tilde{x}_i - x_i)^2 + (\tilde{y}_j - y_j)^2} + \sqrt{(\tilde{x}_i - \hat{x}_i)^2 + (\tilde{y}_j - \hat{y}_j)^2} \quad (5.1-4)$$

The pixel value in Eq. (5.1-4) for the secondary source can be rewritten in Eq. (5.1-5).

$$\begin{aligned} f(\tilde{r}_{ij}) &= r_{\text{rivet-pixel}} \cdot [\tilde{\alpha}_{ij} \cdot \exp(-\beta \cdot \tilde{r}_{ij})] \\ &= r_{\text{rivet-pixel}} \cdot \left[\frac{1}{\sqrt{r_{\text{PZT-rivet}} \cdot r_{\text{rivet-pixel}}}} \cdot \exp(-\beta \cdot \tilde{r}_{ij}) \right] \end{aligned} \quad (5.1-5)$$

where $\tilde{\alpha}_{ij}$ is the recalculated geometrical spreading correction factor at the distance \tilde{r}_{ij} . As an example, consider a resolution of 1 pixel that corresponds to 1 cm in reality, the function values of Eq. (5.1-2) and (5.1-5) for different values of β with the distance $r_{\text{rivet-pixel}} = 0.25\text{m}$ are depicted in Fig. 5.1-1a - b, respectively. Remember that this pixel value is only a dimensionless construct that indicates the Lamb wave attenuation profile. The constructed pixel value is not only based on the attenuation profile (which goes toward $+\infty$ very close to the source) but also to anticipate the near-field zone (NFZ), known as the dead zone since where it is difficult to evaluate any flaws within the NFZ. For simplicity, let us only consider the NFZ to be the area which is covered directly by the PZT.

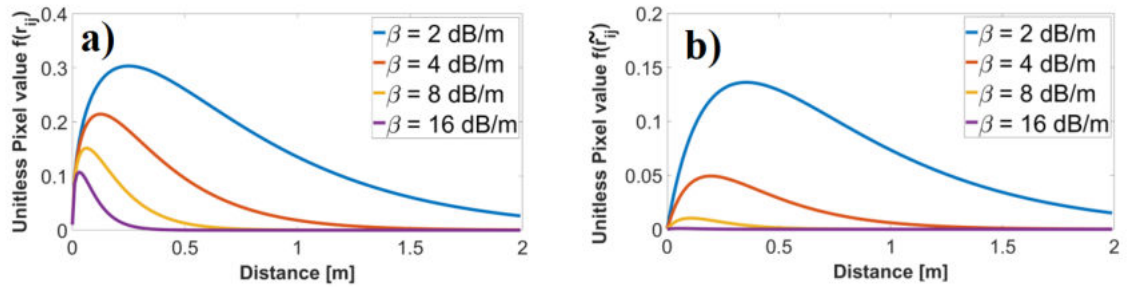


Fig. 5.1-1: Unitless pixel value of a). $f(r_{ij})$ and b). $f(\tilde{r}_{ij})$ as a function of distance which is comparable to amplitude profile. For demonstration purposes, the curve in Fig. 5.1-1b is calculated based on $\tilde{r}_{ij} = 0.25\text{m} + \text{rivet-pixel}$.

Depending on the modes, material, excitation frequency, and thickness, the attenuation β can vary between 0.001 to 0.005 dB/cm [Drinkwater et al. (2013)]. For instance, in CFRP woven (10-ply), an A_0 -mode Lamb wave excited at 285 kHz would only need to travel 85 mm until 90% decay, whereas in woven CFRP of 8-ply, an S_0 -mode Lamb wave excited at 250 kHz, it would need to travel 1700 mm until 90% decay [Su and Ye (2009)].

Generally, the S_0 -mode tends to travel further than the A_0 -mode due to the fact that the A_0 -mode is dominated by perpendicular displacement relative to the wave propagation direction, thus it is leaking more energy to the surrounding environment [Su and Ye (2009)]. This is in contrast to the S_0 -mode which is dominated by the in-plane particle displacement, so that the energy is better conserved within the plate. For the consecutive scattering, [Su and Ye (2009)] suggested to compensate the energy loss due to geometrical spreading by multiplying the measured amplitude with the square root of the time elapsed:

$$\hat{f}(t) = f(t) \cdot \sqrt{t} \quad (5.1-6)$$

Consider the example proposed by [Zhao et al. (2006)], where transducer T is placed between rivet holes as depicted in Fig. 5.1-2a (case 1). Given that actuator T was excited by using a 1.8 MHz excitation frequency, Fig. 5.1-2b illustrates the captured S_0 -mode Lamb wave signal from a series of sensors X that are located 20 – 200 mm away from the actuator T.

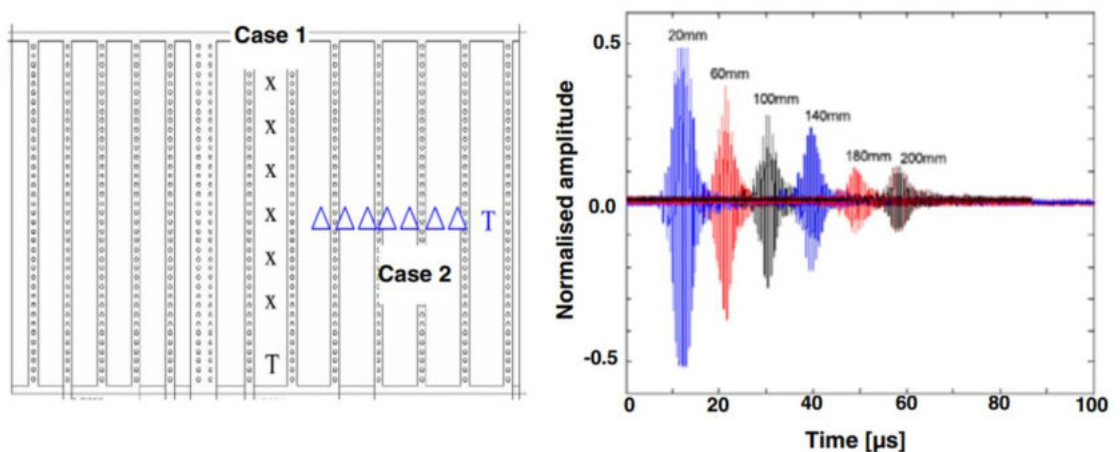


Fig. 5.1-2: Sketch of distribution of rivets and transducers in wing section ('T': actuator; 'X': sensor in Case 1; 'Δ': sensor in Case 2); and b). integrated Lamb wave signals captured by a series of sensors in a straight line (Case 1) [Zhao et al. (2007)].

In this case, they calculated that the average attenuation rate was 0.044 dB/mm. In case 2, they placed sensor series Δ across the stiffeners, and using the same frequency and S_0 -mode excitation obtained an average attenuation of 15 dB per rivet row. The distance between the rivet rows was 6.5 cm, meaning that the average attenuation was increased to 0.231 dB/mm. This calculation already included the multiple scattering across the rivets.

As simplification, only the first appearance of the wave scattering until the wave is absorbed at the boundaries of the plate is considered. The calculated pixel score s_{ij} at pixel $(\tilde{x}_i, \tilde{y}_j)$ for N transducers and B inhomogeneities can simply be defined as a summation of normalized function values $f(r_{ij})$ and $f(\tilde{r}_{ij})$:

$$s_{ij} = \sum_N \|f(r_{ij})\| + \sum_B \|f(\tilde{r}_{ij})\| \quad (5.1-7)$$

The network score τ is simply the summation of all pixel scores, excluding the pixel $P \in P_N$ or P_B , which are occupied by the transducers N and inhomogeneities B , respectively:

$$\tau = \sum_{i=1}^m \sum_{j=1}^n s_{ij}, \quad (5.1-8)$$

where $s_{ij} = 0$ if $P \in P_N(x_i, y_j) \vee P_B(x_i, y_j)$

As a limitation, only the area directly below the PZT is considered as effective NFZ. The rivet hole is idealized as a “secondary actuator”, with the simplification that the wave scatter from the rivet hole is homogeneously reflected to all directions, although in practice, it would depend on the direction of the coming wavefront. Thus, it is logical to set the pixel score to be 0 at those occupied pixels as they do not act as wave detection points. Additionally, Eq. (5.1-9) can be normalized to take any positive real number between 0 and 100:

$$\| \tau \| = \frac{100 \cdot \tau}{(m \cdot n)} = \frac{100}{(m \cdot n - (N + B))} \left(\sum_{i=1}^m \sum_{j=1}^n s_{ij} \right) \quad (5.1-9)$$

Examples of the network score mapping for transducers placed at coordinates 20|40 cm and 115|10 cm in a plate with dimension of 120x80 cm are given in Fig. 5.1-3a - b, respectively, while their alternative representations in 3D projection are depicted in Fig. 5.1-3c - d, respectively. In Fig. 5.1-3a - b, the sensor and rivet hole locations are red dots in locations indicated by white and black rectangles, respectively.

Fig. 5.1-3a shows that the whole plate is better covered if the PZT is located at 20|40 cm since the network score is 39.73, in comparison to Fig. 5.1-3b (PZT location at 115|10 cm) which has a network score of only 33.17. Our definition of coverage is any pixel location where a direct or scattered wave propagates. Thus, a network score of 39.73 can be considered as the average wave amplitude is 39.73% of the maximum. Note that until Eq. (5.1-9), we shall not consider any signal processing parameters nor algorithm yet (except the anticipation towards

the NFZ). The value of coverage level can be later adjusted once the thresholding parameter has been determined.

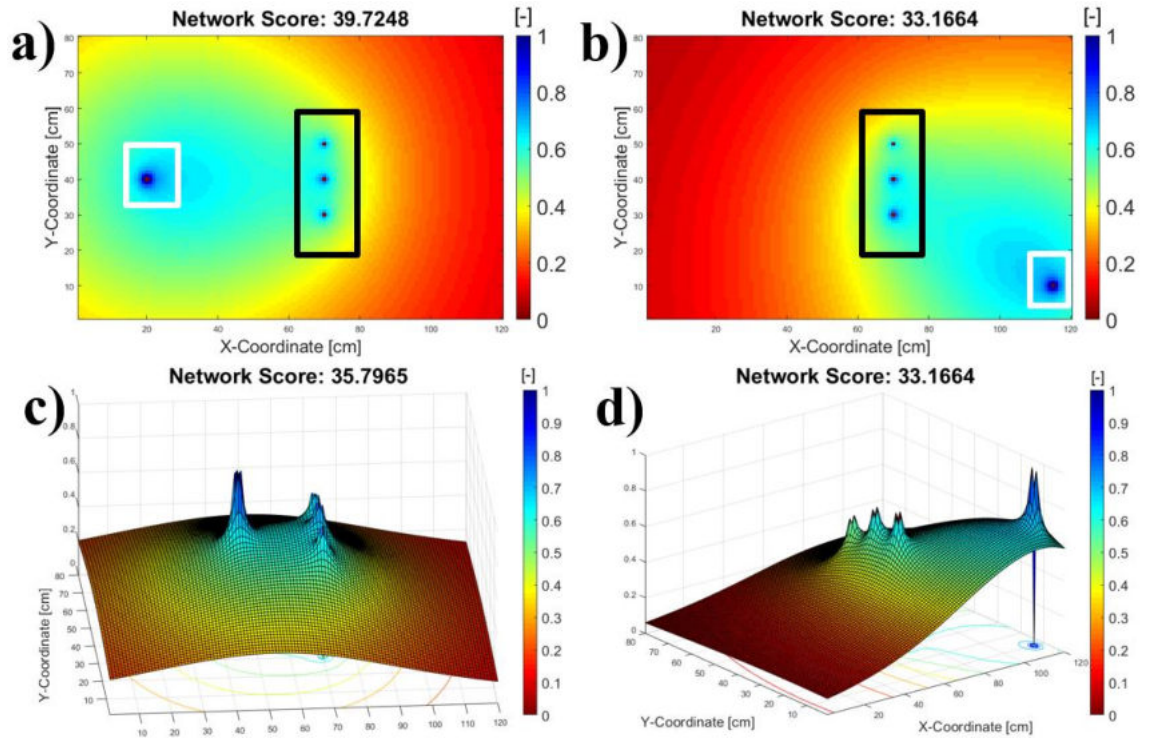


Fig. 5.1-3: Mapped unitless pixel score s_{ij} and network score τ for transducer placement at a). 20|40 cm and b). 115|10 cm. Fig. c) and d) are the alternative representations of the network score in 3-dimensional projection. The white and black rectangles signify the sensor and rivet hole locations, respectively.

Furthermore, the network score will decrease if the attenuation coefficient β is increased as depicted in Fig. 5.1-4a – b (cf. with Fig. 5.1-3a with $\beta = 0.3$). The attenuation coefficient depends on the material properties and excitation frequency [Ono and Gallego (2012)]. This implies that even if the material is the same, the network score will be lower if a higher excitation frequency is applied.

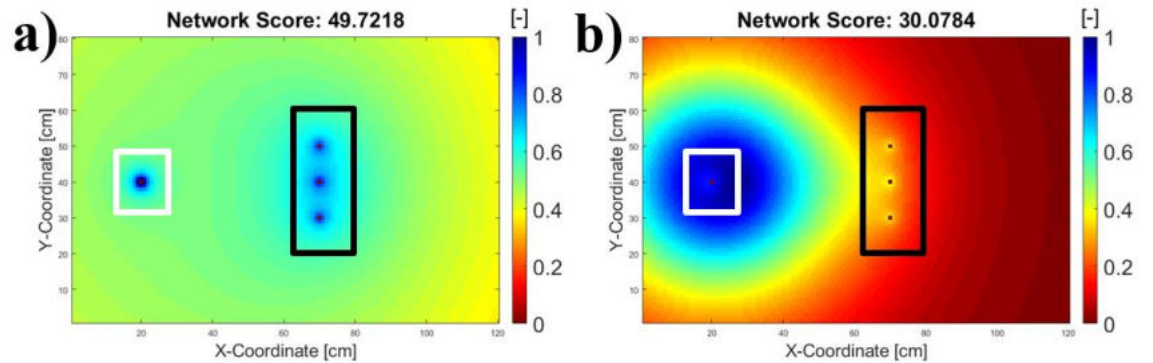


Fig. 5.1-4: Mapped pixel score s_{ij} and network score τ for transducer placement at 20|40 cm for a). $\beta = 0.1$ and b) $\beta = 0.7$. The white and black rectangles signify the sensor and rivet hole locations, respectively.

The best sensor network topology will reach a network score $\|\tau\| = 100$. However, knowing that this is quite unlikely, the objective is defined as:

$$\operatorname{argmax}_{(x_i, y_j)^{(N)}} (\|\tau\|) \quad (5.1-10)$$

Eq. (5.1-10) reads, given N number of sensors, determine the coordinate (x_i, y_j) of each actuator N that maximizes the network score τ . Theoretically, the maximum value is the total amount of pixels without $(N+B)$ as per Eq. (5.1-9). For example, a plate with a size of 120 x 80 cm and 2 mounted sensors and 3 rivet holes would have theoretical maximum score of $9600 - (2+3) = 9595$, or 99.9479 if it is normalized by using Eq. (5.1-9). The interpretation of Eq. (5.1-10) means that at the maximum network score, a minimum attenuation is reached. Assuming that the PZT sensors are able to capture any wave scatter due to the damage occurring at anywhere on the plate and coupled to adequate signal processing, the sensor network will be able to detect and predict the damage location reliably.

From Eq. (5.1-7 - 10), it is obvious that the network score is independent of the damage index DI. This at least eases the transducer placement search for the best fitness. However, to search for the best fitness, it would still take a lot of time even without determining the DI from experiment. The number of possible sensor placement combinations C of given N sensors, B inhomogeneities and L pixels is given by Eq. (5.1-11).

$$C = \frac{(L - B)!}{N!(L - B - N)!} \quad (5.1-11)$$

As an example, assume each pixel size is 1x1 cm, then for a plate of 120x80 cm for a single sensor ($N = 1$) and 3 rivet holes ($B = 3$) in which there are $C = 9597$ possible combinations, the computation time for the brute force search with my PC specification is 2.57 seconds. However, for two, three, and four sensors the calculation time would increase from 3.4 hours to 15 months and to 3000 years, respectively.

5.2. Methodology

From Eq. (5.1-11), it is easily known that it is not feasible to look for all possible transducer locations since it may take indefinitely long. When an exhaustive brute force search takes too much time, normally a heuristic search is employed to find a close-to-optimal solution within a reasonable amount of time. To find a viable solution from such a large search space, one could consider the following approaches: 1). no prior knowledge was used during the decision making; thus, the decision probability is equally distributed over the decision set, 2). prior knowledge is used in the decision making and for sorting the decision options, and 3). no prior knowledge was involved at the beginning but is gradually incorporated as the decision-making process evolves.

These approaches can be related to popular search techniques [Hopcroft et al. (2006), Rastrigin (1963), Kramer (2017), Schmitt (2001), Hung et al. (2008), Kennedy and Eberhart (1995)] such as: random search, greedy methodology some metaheuristics such as genetic algorithm (GA), swarm intelligence, and simulated annealing (SA). In this section, investigate these approaches to find a solution for Eq. (5.1-10) will be investigated.

5.2.1. Global Random Search

Global random search is the easiest method to use to solve a combinatorial problem. However, given a limited time constraint, it is also the least efficient since the optimal sensor position might not be found. The algorithm is very simple and can be demonstrated in only 4 lines of pseudocode, as shown in Algorithm 5.2.1-1. The random value in this case is a random position of x_i, y_j .

Algorithm 5.2.1-1. Random Search

Input parameter: X – random value from search space

Output: Y – output function $f(X)$, decision variable $w = \{0,1\}$

Until termination condition is reached:

Do $\left\{ \begin{array}{l} \text{Calculate } f(\text{random value}) \\ \text{Update } w \text{ if } f(\text{random value}^+) > f(\text{random value}) \end{array} \right.$

The results of the global random sensor position search are depicted in Fig. 5.2.1-1a – b. Fig. 5.2.1-1a depicts the search result for 3 sensors, while Fig. 5.2.1-1b depicts the result of a search of 6 sensors. While the result would normally change for each iteration, it is possible for the random search to converge with an increasing number of sensors (see Fig. 5.2.1-1c). Fig. 5.2.1-1c depicts the average network score after 10 searches for a 1 – 10 sensors search from 1 to 1000 iterations.

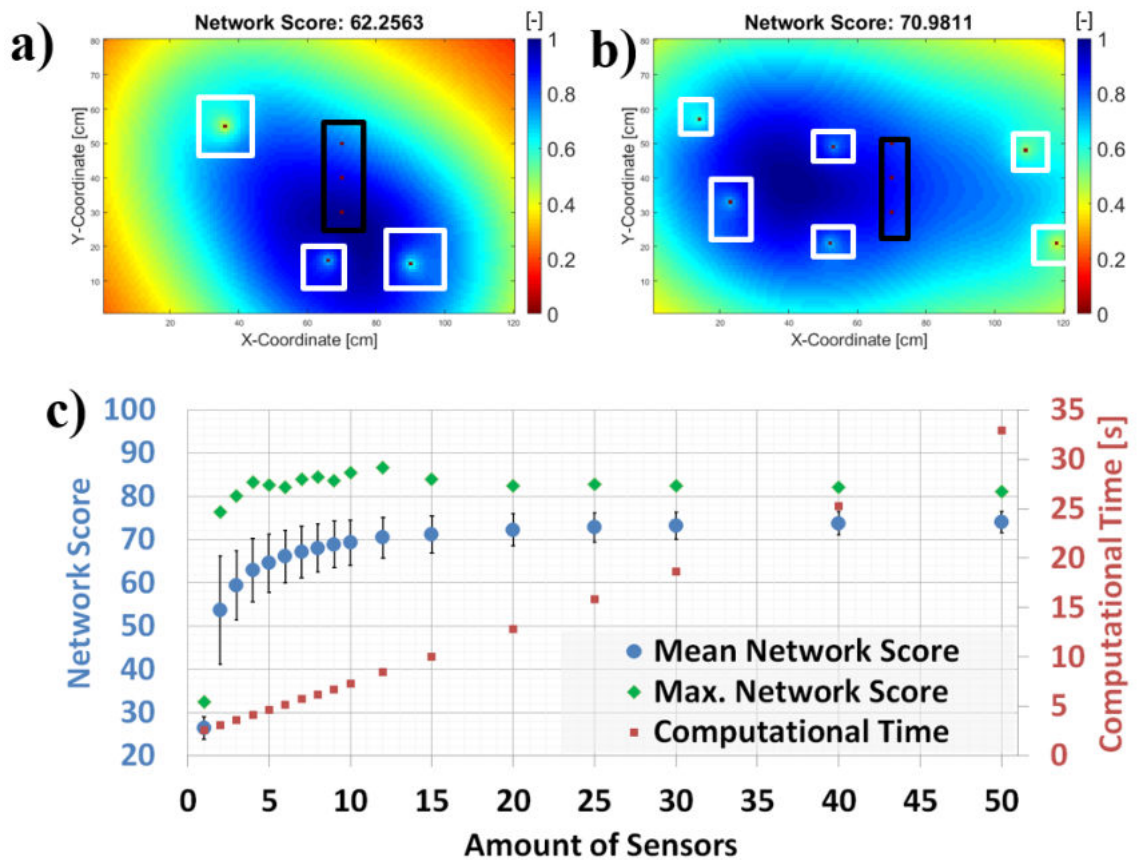


Fig. 5.2.1-1: Found sensor network pattern by random search for a) 3 sensors and b) 6 sensors with black & white rectangles convention from Fig. 5.1-4, while c) depicts the distribution of the network score for a random search of 1 – 50 sensors after 1000 iterations. Error bars indicate the standard deviation of the network score during 1000 iterations.

As one can see in Fig. 5.2.1-1c, the random search algorithm starts to converge from 5 sensors with decreasing standard deviation (indicated by the error bars) towards an average network score of around 81, which means the plate is 81% covered by the wavefront should an impact happen anywhere on the plate. The computational time, as expected, is linear since the calculation effort is the same for every iteration.

5.2.2. Greedy Search

According to [Cormen et al. (2009)], “a greedy algorithm always makes the choice that looks best at the moment”, i.e., it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution”. For some problems, the greedy algorithm can provide an optimal solution, while in other cases it does not, because sometimes the selected solutions reach a local optimum. An example pseudocode of a greedy algorithm was shown in Algorithm 3.3.2-1 (Chapter 3).

In the greedy algorithm, the function value will be sorted from the minimum to maximum, and the argument maximum is chosen as the optimal solution. For multiple sensors, the greedy algorithm will search the next optimal sensor position step by step. That is, the next sensor position is determined by the previous sensor position by considering the last previous sensor position. In practice, this will lead to locally optimal solutions that might still be globally optimal solution within a reasonable amount of time.

The mode of operation of the greedy algorithm is depicted in Fig. 5.2.2-1a – d. The algorithm finds the best position of the first sensor (Fig. 5.2.2.1-a), then calculates the next best sensor position based on the position of the previous sensor.

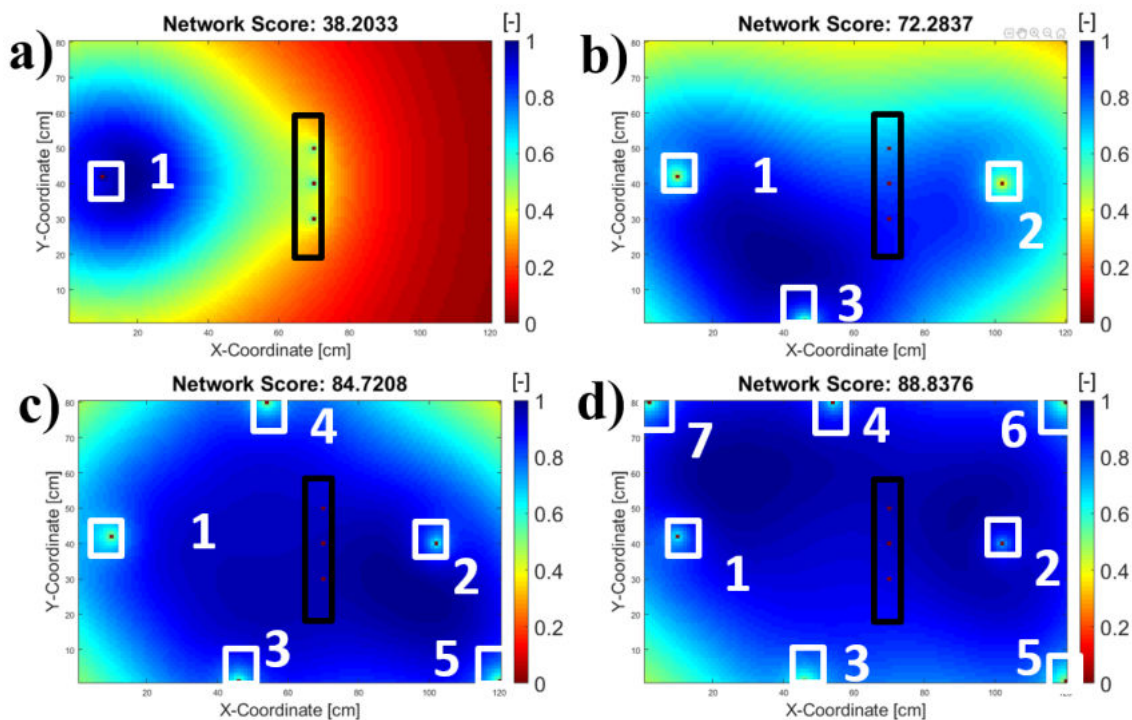


Fig. 5.2.2-1a - d: Found sensor network pattern by greedy methods for 1, 3, 5, and 7 sensors, respectively. The black and white rectangles convention applies.

As can be seen from Fig. 5.2.2-1a vs Fig. 5.2.2-1b, the greedy algorithm for 3 sensors search (network score = 72.28) performs better than a global random search (average network score = 62.26). This is expected because the random search does not have clear strategy to find the maximum except by saving the best possible solution for each iteration, while the strategy of finding the maximum in the greedy algorithm is by dividing the problem into smaller sub-problems. The greedy algorithm was able to determine the maximum theoretical network value thanks to the sorting function which in this case is 38.2033 as can be seen in Fig. 5.2.2-1a. The problem with this approach is when more than 1 sensor search is applied, the required calculation time is comparable to $9600N$ for a plate size of 120 x 80 cm, where N is the amount of the sensors. Sorting is therefore not always feasible for a multivariable search.

5.2.3. Metaheuristics Search

5.2.3.1. Genetic Algorithm (GA)

Usually, a GA contains three main operators: mutation, crossover, and selection [Schmitt (2001)] and typically, the procedure starts with a given initial population that will be assessed against its fitness. Those individuals who have the best fitness are crossed-over to each other and/or a “genetic mutation” is applied e.g., by bit-flipping or replacement. The individuals who do not have the best fitness are not selected. This procedure is repeated several times until a specified certain termination condition is reached. A pseudocode example of GA has been described in Algorithm 3.3.2-2 in the chapter 3. For the sensor placement problem, the sensor coordinates x_i, y_j are first encoded as a chromosome (Fig. 5.2.3.1-1) that will be assessed against the fitness function. The genome length is $2N$, where N is the amount of the sensors to be installed.

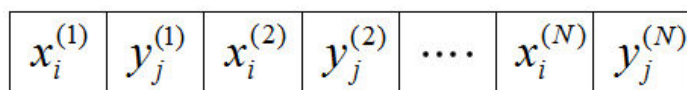


Fig. 5.2.3.1-1: Sensor position as chromosome in genetic algorithm for N sensors

Typically, the aircraft manufacturer or operator will determine how many sensors are to be installed based on the balance between cost, additional weight, POD / sensor network performance and safety. Generally, the more sensors that are installed means a higher Lamb wave coverage, but this also means higher costs and energy consumption, and more weight since every sensor is attached to a cable. Also, after a certain number of sensors, the coverage will only slowly increase up to the upper limit of the sensor network performance. The results from a genetic algorithm for 1 – 8 sensor searches are depicted in Fig. 5.2.3.1-1a – f, respectively. From these figures, one can clearly see that the genetic algorithm tends to outperform the global random search and greedy algorithms. For instance, for 3 sensor searches, the genetic algorithm reaches a network score of 84.12 (Fig. 5.2.3.1-1c) after 21 seconds, while the greedy algorithm only reaches a network score of 72.28 (Fig. 5.2.2-1b).

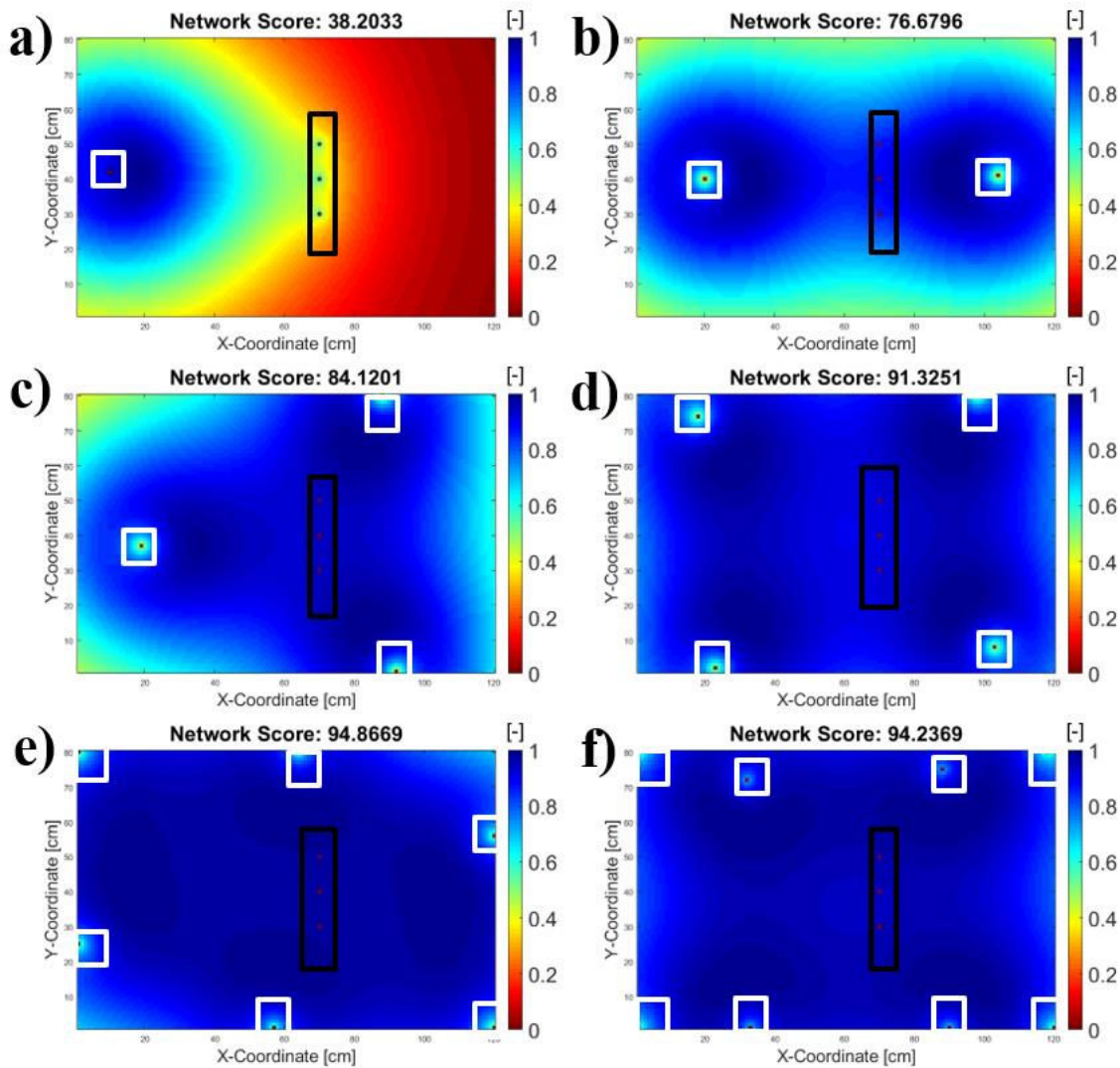


Fig. 5.2.3.1-1a - f: Found sensor network pattern by genetic algorithm search result for 1, 2, 3, 4, 6, 8 sensors. The black and white rectangles convention applies.

The random search performed even worse as it reaches a network score of 62.26 (Fig. 5.2.1-1a). Note that for the special case of a 1 sensor search, the genetic algorithm was successful in finding the maximum theoretical value of the sorting greedy algorithm, which is 38.2033 (cf. Fig. 5.2.1-1a to Fig. 5.2.3.1-1a) – thus there is no difference in this case between the greedy and the genetic algorithm, which proves that the construction of the genetic algorithm worked in a consistent way.

However, it should be noted that the genetic algorithm performs slowly, especially when the number of sensors increases. While the greedy algorithm and random search give a result almost immediately, a search with a genetic algorithm for 6 sensors took almost 4 minutes. Not only that, but the more sensors are also employed, the more computer memory is needed, sometimes forcing earlier termination of the algorithm and resulting a lower score, such as in Fig. 5.2.3.1-1f.

In a larger plate where more sensors are to be installed, a genetic algorithm would deliver a high network score, however this will be neutralized by its slower performance. This is actually in line with the **No Free Lunch Theorem** proposed by [Wolpert and Macready (1997)]: “Given a finite set V and a finite set S of real

numbers, assume that $f: V \rightarrow S$ is chosen at random according to uniform distribution on the set S^V of all possible functions from V to S . For the problem of optimizing f over the set V , then no algorithm performs better than a blind search.”

5.2.3.2. Simulated Annealing

As explained in chapter 3, simulated annealing (SA) is another heuristic method like GA. It was inspired by a process used in metallurgy, where steel blocks are slowly cooled down and reheated in alternating cycles in order to minimize the energy of the material. This method is thus used as an inspiration in the optimization problem to find the extrema of a function. For this sensor optimization method, the algorithmic approach is given in Algorithm 5.2.3.2-1. Note that in Algorithm 5.2.3.2-1, **random(0,1)** is a function that randomly takes any real number between 0 and 1. The search result obtained by using simulated annealing is depicted in Fig. 5.2.3.2-1a - f.

Algorithm 5.2.3.2-1. Simulated Annealing

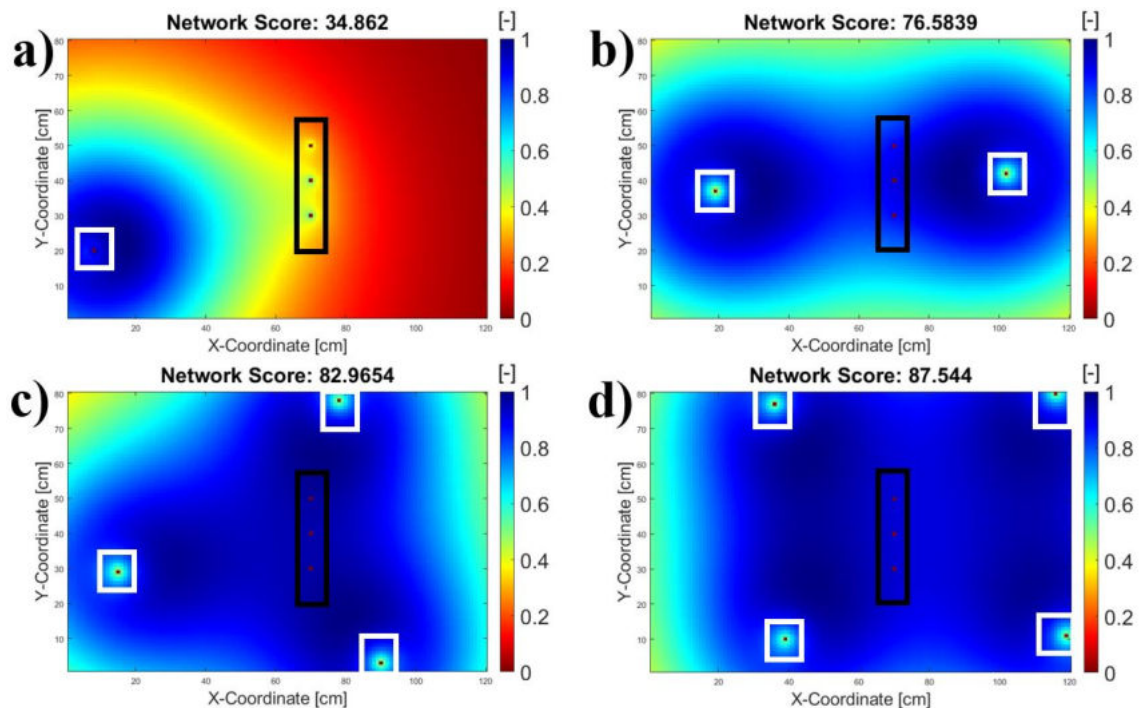
Input parameter: X – sensor position as x and y coordinate,

T_0 – initial temperature, $T \leftarrow T_0, X \leftarrow X_0$

Output: Y – output function $f(X)$

Until termination condition is reached:

Do $\left\{ \begin{array}{l} \text{Pick a random neighboring sensor position } \tilde{X} \neq X_0 \\ \text{Calculate differential energy } \Delta E(X_0) \leftarrow J(\tilde{X}) - J(X_0) \\ \text{If } \min \left[1, \exp \left(\frac{\Delta E}{T} \right) \right] \geq \mathbf{random}(0,1) : \\ \quad X \leftarrow \tilde{X} \end{array} \right.$



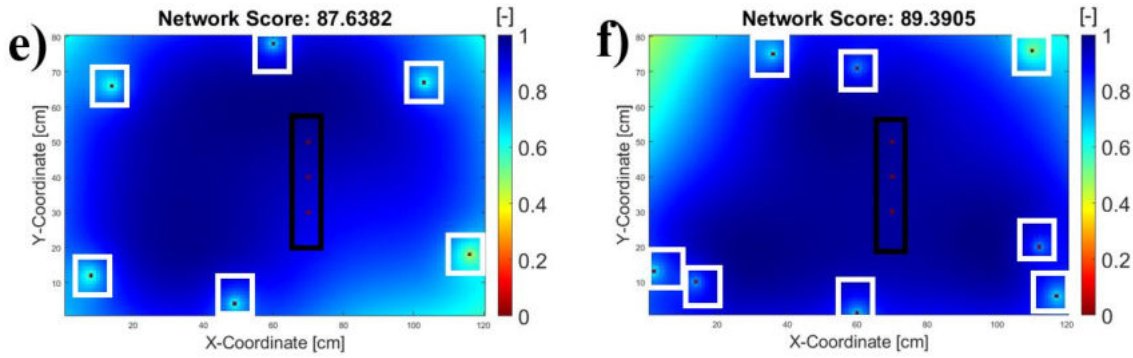


Fig. 5.2.3.2-1a - f: Found sensor network pattern by simulated annealing search result for 1, 2, 3, 4, 6, 8 sensors. The black and white rectangles convention applies.

Comparing these figures with GA (Fig. 5.2.3.1-1a - f), we can see that simulated annealing (SA) performs slightly worse than GA, but it is better than random search.

5.2.3.3. Swarm Intelligence

The last metaheuristics method being used is the swarm intelligence, which was inspired by biological swarm behavior. In this case, every individual within the swarm will act in accordance with certain simple rules, but the interaction or synergy between simple individual actions can result in various ways of collectively complex behaviors. The algorithmic approach for solving the sensor placement problem is given in Algorithm 5.2.3.3-1 and the search results for particle swarm optimization (PSO) are depicted in Fig. 5.2.3.3-1a - f.

Algorithm 5.2.3.3-1. Particle Swarm Optimization

Initial parameters: sensor coordinate \equiv particle position X_i , particle velocity v_i

Assign inertia factor w , and cognitive learning rate c_1 and c_2

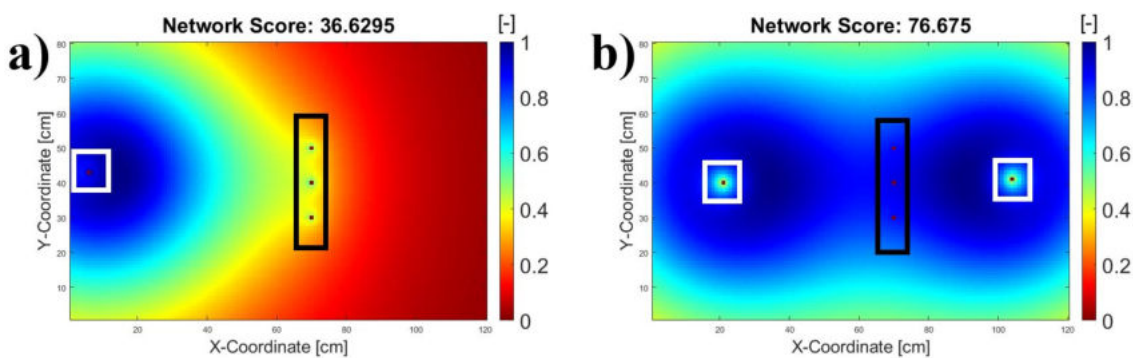
Assign sensor best position \equiv particle best position $p_i \leftarrow X_i$

Until termination condition is reached:

Do {

- Calculate fitness value $f(X_i)$
- Update p_i if $f(X_i^+) > f(X_i = p_i)$
- Choose X_i with best fitness value and assign as g_i
- Calculate new velocity $v_i = (w \cdot v_i + c_1 \cdot \mathbf{random}(p_i - X_i) + c_2 \cdot \mathbf{random}(g_i - X_i))$
- Update particle position $X_i \leftarrow X_i + v_i$

}



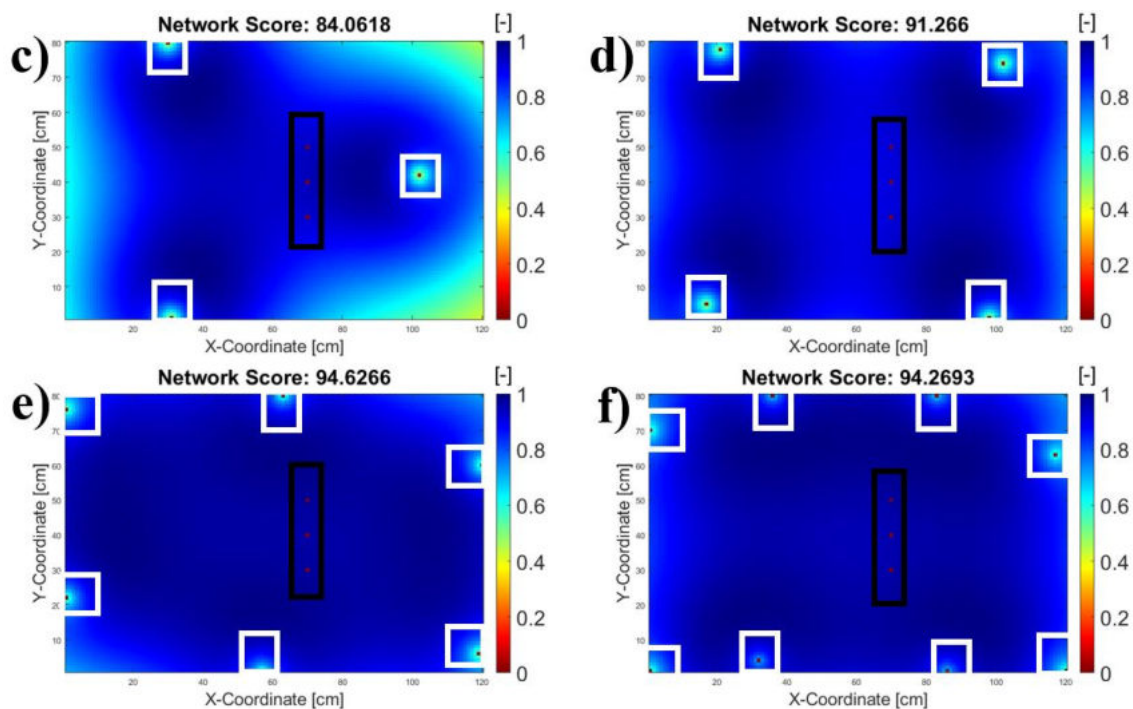


Fig. 5.2.3.2-1a - f: Found sensor network pattern by swarm intelligence for 1, 2, 3, 4, 6, 8 sensors. The black and white rectangles convention applies.

5.3. Preliminary Results

5.3.1. Comparison between Greedy Methods, Random Search, and GA.

As a summary, a performance comparison of global random search (after 1000 runs), greedy, and genetic algorithms (after 10 runs) is presented in Fig. 5.3.1-1, where only the standard deviation (stdev) σ from the random search is shown. Note that the stdev σ of the genetic algorithm (GA) is too small to be visualized in the graph.

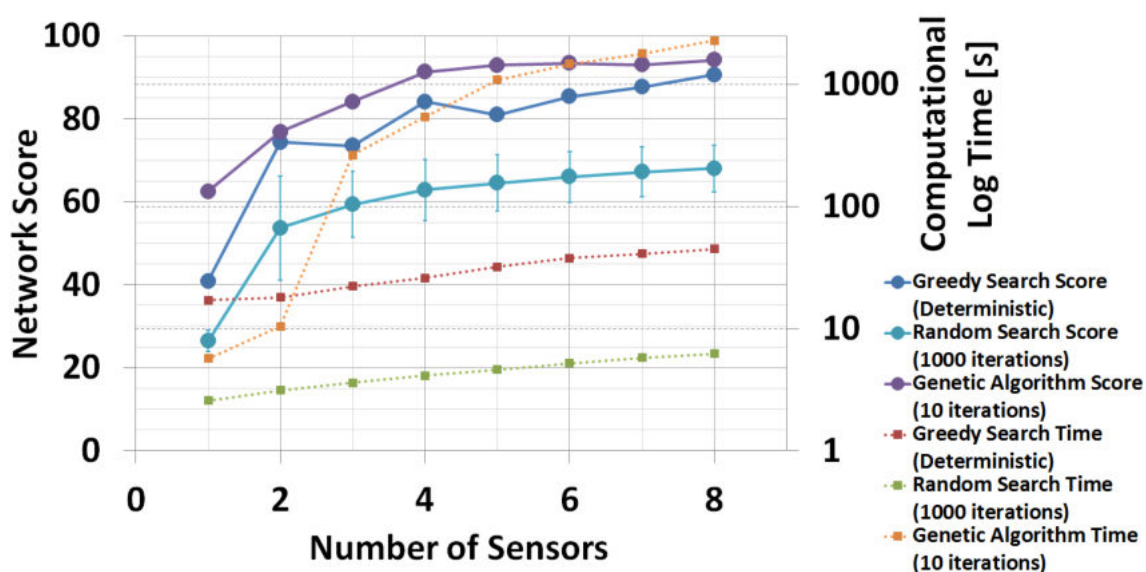


Fig. 5.3.1-1: Network score for random search algorithm of 1 - 8 sensors

The stdev of the greedy algorithm is 0 because each step of the algorithm is a deterministic method in which no stochastic factor is involved. The x -axis

represents the number of sensors while the left γ -axis represents the network score reached by each algorithm. The right γ -axis represents the computational time needed for each algorithm. Neglecting the computation time, obviously the GA has the best performance from the three algorithms. However, taking the computation time into account, the greedy algorithm is competitive with the GA. Note that the right γ -axis has a logarithmic scale. Conversely, the random search took the lowest computational time while it has the lowest network performance.

It can be seen from Fig. 5.3.1-1, that GA starts to outperform the greedy algorithm from 3 sensors onwards, however the 10-run genetic algorithm took about 12 times longer (about 264 seconds) than the greedy algorithm. Also, note that in Fig. 5.3.1-1, the standard deviation for the GA cannot be shown as these are too small to be visualized (2.04 or less)

One could argue about the best terminating conditions of these algorithms. For this reason, 4 different thresholds of network score are set: 80, 85, 90, and 95 which can be understood as coverage levels between 80% – 95% of the surface – which when coupled with adequate signal processing can yield a global probability of damage detection (POD) of the network between 80% – 95%. The trade-off between the network score, number of sensors N , and the required computational time is presented in Table 5.3-1. For brevity, let us not consider the hardware weights and the potential data redundancy as well as the system energy and/or power required as the number of sensors increases.

Algorithm	Network Score: 80		Network Score: 85		Network Score: 90		Network Score: 95	
	N	Time [s]	N	Time [s]	N	Time[s]	N	Time[s]
Random	3	3.62	10	7.32	n/r	n/r	n/r	n/r
Greedy	4	26.03	6	37.95	8	44.96	n/r	n/r
GA*	3	132.09	4	268.64	4	268.64	10	1442.25

Table 5.3.1-1: Number of sensors (N) and time needed to reach network score from 80 to 95. The result for random search is based on 1000 iterations, whereas for GA is based on 10 iterations. *n/r: not reached. (*): GA: Genetic Algorithm

As can be seen in Table 5.3.1-1, neglecting the required computational time, it is obvious that the genetic algorithm has the best performance from the three algorithms, however, it also requires the most computational time. Conversely, the random search has the lowest computational time and the lowest network performance.

5.3.2. Comparison between Metaheuristics

Analogous to section 5.3.1, I will compare the results between the metaheuristics search result: genetic algorithm (GA), particle swarm optimization (PSO), and simulated annealing (SA). In order to save time, only 5 iterations instead of 10 are run. The results are given in Table 5.3.2-1.

N	PSO			GA			SA		
	Mean	Stdev	Time	Mean	Stdev	Time	Mean	Stdev	Time
1	36.9031	0.0068	53.4	38.2034	0.0000	26.2	34.7713	2.1256	14.5
2	76.7108	0.0028	68.4	76.6800	0.0000	38.5	76.6192	0.0752	75.0
3	84.1839	0.0036	149.8	84.1225	0.0012	81.1	83.1327	0.5466	79.2
4	91.3353	0.0145	271.5	91.3135	0.0142	214.4	89.3785	0.9662	77.4
5	92.1262	2.2475	570.7	92.5737	0.5127	468.9	87.0934	1.8111	192.4
6	95.0607	0.1544	422.1	94.4081	0.8687	341.1	87.4289	0.9803	288.1
7	93.9077	1.0875	845.8	93.1091	0.8386	461.7	88.9369	0.5013	338.4
8	94.5158	0.8002	662.1	94.4317	0.0630	1102.8	87.9961	0.6909	523.8

Table 5.3.2-1: Comparison of search results after 5 iterations by using 3 metaheuristics: Particle swarm optimization (PSO), genetic algorithm (GA), and simulated annealing (SA). N is the number of sensors. The time is expressed in [s].

As we can see from Table 5.3.2-1, both GA and PSO seem to outperform SA, especially for a search of more than 3 sensors. This is to be expected since below 3 sensors, the search space is still relatively small, and SA only tracks a single solution at time. Accordingly, the computation time needed for SA is relatively lower than GA and PSO (Recall: *No Free Lunch*). Starting from 4 sensors however, SA seems to converge and is not able to reach a network score of 90. Conclusively, after 5 iterations only GA and PSO are able to reach a network score of 90 with a minimum of 4 sensors in a plate 120 cm x 80 cm, as given in Table 5.3.2-2.

Algorithm	Network Score: 80		Network Score: 85		Network Score: 90		Network Score: 95	
	N	Time [s]	N	Time [s]	N	Time[s]	N	Time[s]
PSO	3	149.8	4	271.5	4	271.5	6	422.1
GA	3	81.1	4	214.4	4	214.4	n/r	n/r
SA	3	79.2	4	77.4	n/r	n/r	n/r	n/r

Table 5.3.2-2: Number of sensors (N) and time needed to reach network score from 80 to 95. Search results are based on 5 iterations. *n/r: not reached.

5.4. Integrative Method

The hotspot SHM design described in chapter 4 has the job of monitoring hotspot. On the contrary, the probabilistic approach for SHM described until section 5.3 in this chapter is useful for detecting random damage locations. The proposed SHM system design is to integrate both approaches in one. This is because when aircrafts are in service, they are prone to both types of damages whose occurrence are likely to be independent of each other.

The procedure described in chapter 4 can be reproduced for an aluminum plate with dimensions of 120 x 80 cm and the two best hotspot sensor locations were determined to be 45|40 cm and 80|40 cm. As a reference, the network score for the hotspot SHM configuration with the sensors located at 45|40 cm and 80|40 cm is 52.22, as depicted in Fig. 5.4-1a. This is clearly inferior to a 2-sensor network generated by the genetic algorithm (cf. Fig. 5.2.3.1-1b). That means, the proposed

hotspot SHM network would have a relatively poor coverage for the detection of damage at a random location.

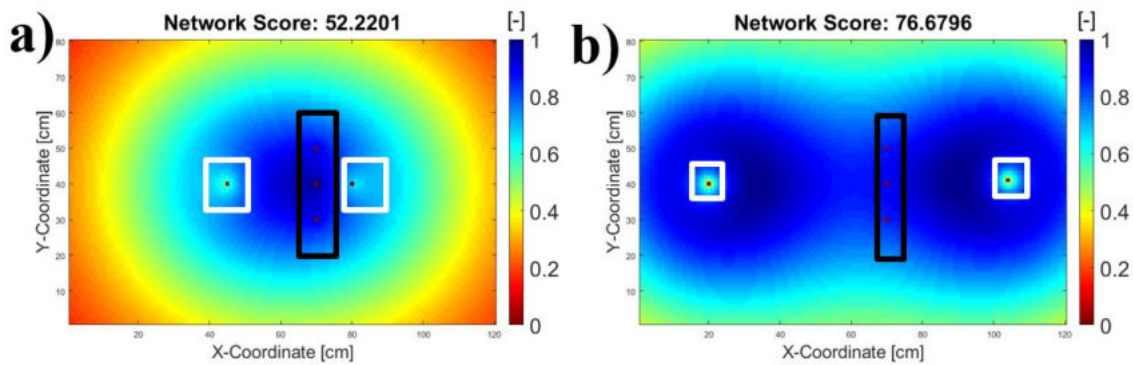


Fig. 5.4-1: Sensor network score and topology for a) hotspot SHM network for detection of hotspot crack and b) stochastic SHM network for random impact detection.

In an analogous way, the network pattern for detecting random damage locations depicted in Fig. 5.4-1b might have a relatively lower detectability for the detection of cracks from the rivet holes. As a side note, Fig. 5.4-1b is an exception: this sensor configuration might still have a good detectability of hotspot crack detection considering that the wave scatter from the rivet hole is coming in a perpendicular direction to the sensor. After first putting the two hotspot SHM sensors at 45|40 cm and 80|40 cm, the network score can be increased by adding several other sensors using the metaheuristics such as GA and PSO described in section 5.1 – 5.3. As an example, a demonstration by adding 1, 2, 3, and 4 additional sensors using GA is depicted in Fig. 5.4-2a – d. Note that in these figures, the first and second-best hotspot SHM sensor coordinates are denoted by numbers 1 and 2, respectively and the locations do not change in every iteration.

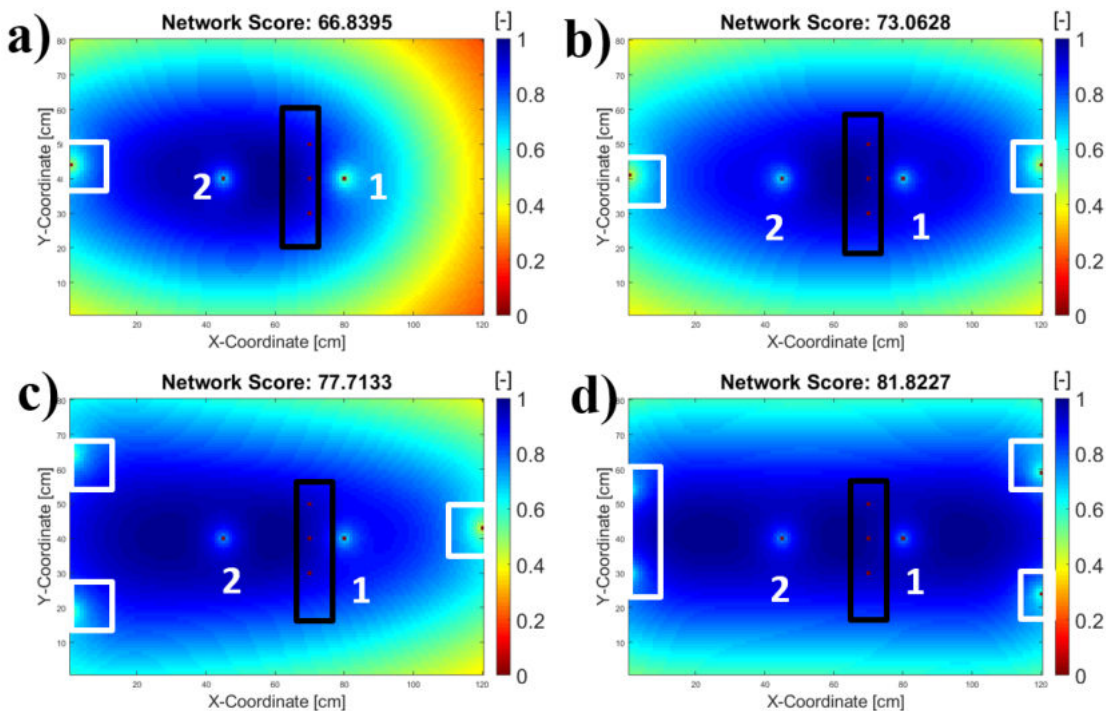


Fig. 5.4-2: Network score and topology for integrated SHM with 2 sensors for hotspot and a) 1, b) 2, c) 3, and d) 4 sensors for random damage detection, respectively. The first and second-best hotspot SHM sensors are denoted by number 1 and 2.

The black and white rectangles signify the rivet hole and stochastic SHM sensor network topology, respectively. It can be concluded from Fig. 5.4-2a - d, that the network scores are lower than those from the solution generated by the pure genetic algorithm, however I think that this hybrid approach is the most plausible way to compensate the conflicting objectives between hotspot and global SHM sensor placement.

5.5. Experimental Validation

5.5.1. Reproducing Hotspot SHM

As described in section 5.4, a hybrid approach that combines hotspot and global SHM sensor network design was made for an aluminum plate of size 100 x 50 cm with 8 rivet holes using the simulation parameters given in [Ewald (2015)] to demonstrate that the algorithm works for different geometrical size. The hotspot location was assumed to be at 75|20 cm with a maximum damage tolerance size of a 16 cm fatigue crack (from tip to tip). The rest of the process is merely a reproduction of chapter 4 for different geometry.

Fig. 5.5.1-1a - b depicts the wave propagation at $t = 100 \mu\text{s}$ in the baseline and artificially cracked plate, respectively, where Fig. 5.5.1-1c is the subtracted result of Fig. 5.5.1-1a and 5.5.1-1b. There are several blob centroids of interest as depicted in Fig. 5.5.1-1c, marked by the blue dots, where the largest and second-largest blobs are marked by green and red dots, respectively. The fused images from wave propagation between 25 and 250 μs is depicted in Fig. 5.5.1-1d, and after averaging the blob centroids from all time frames, the best hotspot sensor coordinates found after using blob detection algorithm were 65|21 cm and 84|20 cm.

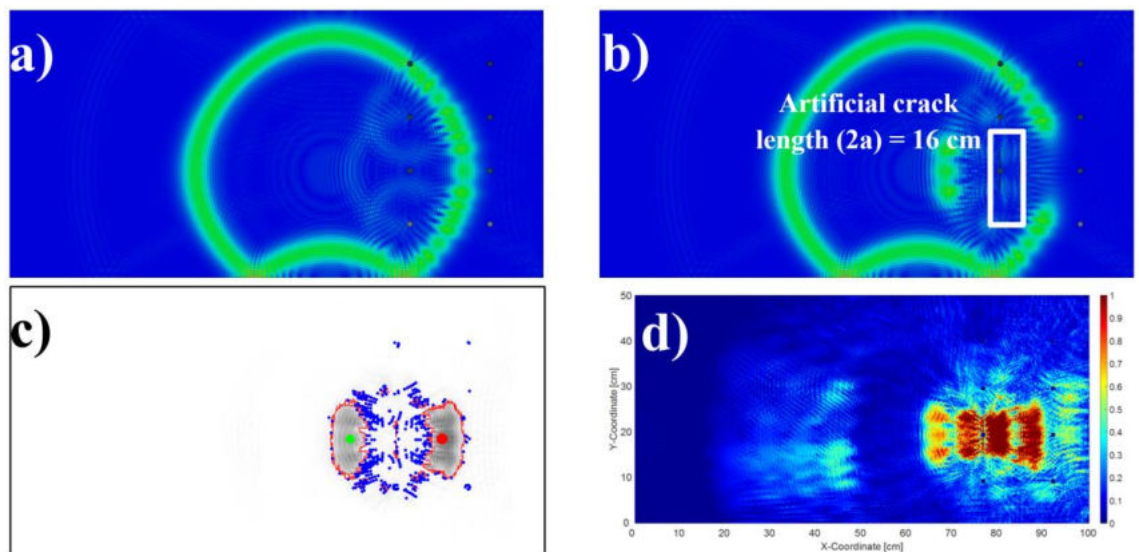


Fig. 5.5.1-1: Simulated wave propagation in a). baseline / pristine and b). artificially cracked plate. c). The subtracted result of image a) and b), where the centroids of largest and 2nd-largest blob are marked by green and red dots, respectively, d). Fused differential image from all time frames between 25 μs and 250 μs .

5.5.2. Experimental Setup

After finding the hotspot sensors coordinates at 65|21 cm and 84|20 cm, the rest of the locations were determined by the GA since it has lower stdev than PSO as can be seen in Table 5.3.2-1. To minimize the number of PZTs used in the experiments, I tested two sample topologies: 1). 3 global + 2 hotspot sensors, and 2). 5 global + 2 hotspot sensors. For conciseness, the first network will be denoted as the “3+2” network, while the latter will be denoted as the “5+2” network. The sensor coordinates determined by the genetic algorithm are depicted in Fig. 5.5.2-1a - b.

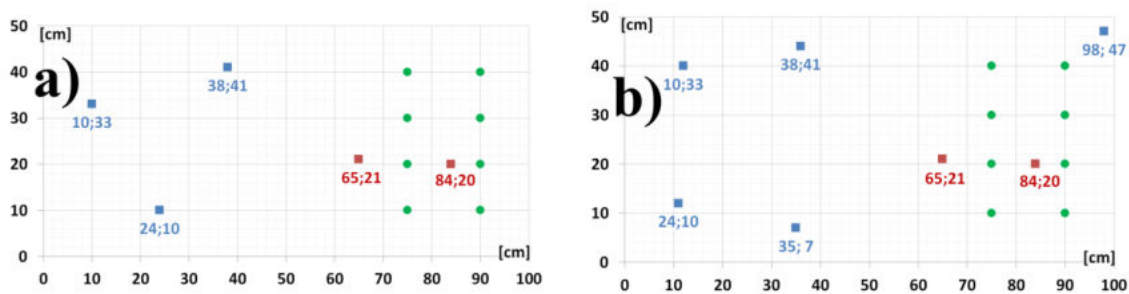


Fig. 5.5.2-1: Sensor network pattern for a). 3+2 network (Scenario 1A) and b). 5+2 network (Scenario 1B). The red dots signify the hotspot sensors, and the blue dots signify the rest of the global sensors that are determined by the genetic algorithm (GA) since it has slightly lower stdev than the PSO.

Multiple PZTs could additionally act as actuators to send excitation signals, and this would generate a larger dataset. Ideally, to reach the energy level that corresponds to the maximum network score, all actuators should be excited at the same time. For the 3+2 network, this would require 5 waveform generators and this amount of necessary hardware was not available, and thus for the sake of demonstration, only the hotspot sensor located at 65|21 cm is excited. This only corresponds a fraction of the previously described signal power was used. The available hardware during the test were a Picoscope 6402A oscilloscope, an Agilent Waveform generator 33500B, standard BNC cables, a radial PZTs American Piezo APC-850 (\varnothing 9.52 mm, thickness = 1 mm, resonance frequency $f_r = 207$ kHz), a desktop PC installed with Waveform Builder Pro and software from Picoscope. The experimental setup is depicted in Fig. 5.5.2-2.

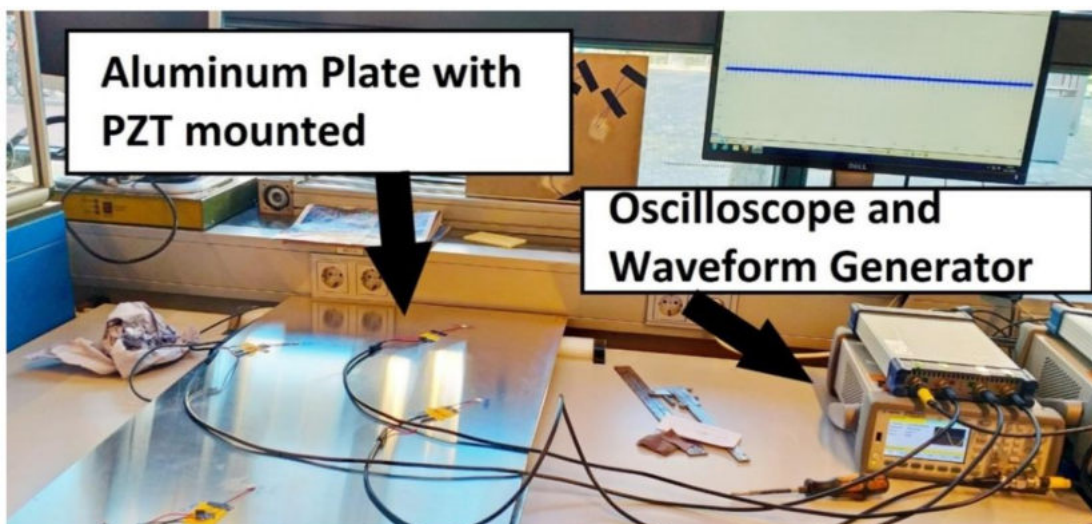


Fig. 5.5.2-2: Experimental Setup

As we have radial mode PZTs with a thickness smaller than the diameter, then normally it is the S_0 -mode which will be the predominant waveform that will be actuated and sensed by the PZT [Pohl et al. (2012)]. From 8 specimens, 2 plates must be assigned as baseline, otherwise the residual time-trace cannot be calculated. The baseline from the 3+2 and 5+2 network will be designated as Scenarios 1A and 1B, respectively. As scenarios 2 and 3 have very large and visible damage (almost 30 cm), we can expect that the 3+2 network will be more than sufficient to identify this damage, and accordingly since scenarios 6 and 7 have a hardly noticeable dent, the denser 5+2 network is assigned to them. Finally, to compare the damage localization performance from both networks, the 3+2 and 5+2 network were assigned to scenarios 4 and 5, respectively.

5.5.3. Impact Damage Setup

To experimentally validate the sensor network configuration to detect both random and hotspot damage occurrences, several damage scenarios are tested as given in Table 5.5.3-1. An artificial fatigue crack was created by milling a slot adjacent to the rivet hole. The length of the artificial crack in product applications will be determined according to the damage tolerance criteria. For this study, it is assumed to be 8 cm from tip to tip, i.e., including the rivet hole with a diameter of 1 cm. Due to the limitations of the dimensions of the fixation table of the impact tower (see Fig. 5.5.3-1a), the dimension of the plate is reduced to 100 cm x 50 cm (mentioned earlier in section 5.5.1) as depicted in Fig. 5.5.3-1b.

Scenario	Height h [m]	Energy E_{Impact} [J]	Velocity v [m/s]	Description of Impact	Impact Epicenter [cm]	Hotspot Crack Coordinates [cm]	Sensor Network
1A	None						3+2
1B	These are baselines						5+2
2	2.0	80.4	6.3	ca. ½ plate breaks	27;24	None	3+2
3	2.0	80.4	6.3	ca. ½ plate breaks	27;24	75;20	3+2
4	1.7	68.4	5.8	ca. ¼ plate breaks	20;35	None	3+2
5	1.7	68.4	5.8	ca. ¼ plate breaks	20;35	None	5+2
6	1.6	64.3	5.6	only dent	27;24	75;20	5+2
7	1.6	64.3	5.6	only dent	27;24	None	5+2

Table 5.5.3-1: Damage scenarios tested by using aluminum 7075-T6 of size 100 x 50 x 0.2 cm.

Depending on the impact type (hail impact, tool drop, ground collision), the impact energy could vary. For instance, a tool drop has a typical energy lower than 28 Joule [Li et al. (2016)], while hail impact energy during taxiing can reach up to 162 Joule [Li et al. (2016), USDOT-FAA (2017)] and this can reach 3900 Joule

[Li et al. (2016)] during cruise. The TU Delft impact tower was operated at up to its height limit $h = 2.0$ m, which corresponds to an impact energy of 80.4 J, as given in Table 5.5.3-1. This impact energy was sufficient to cause a large visible damage on the test coupon (see Fig. 5.5.3-2a).

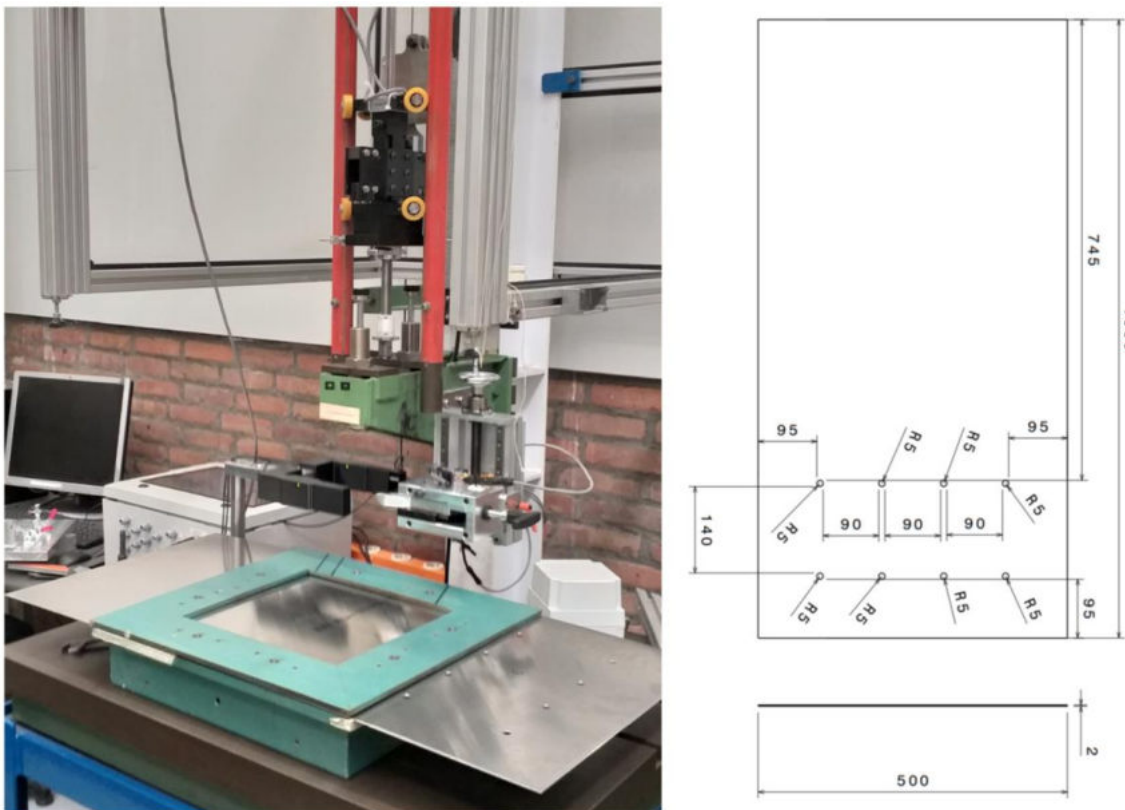


Fig. 5.5.3-1: a). Impact test setup and b). Dimension of the specimen in mm (including the rivet holes)

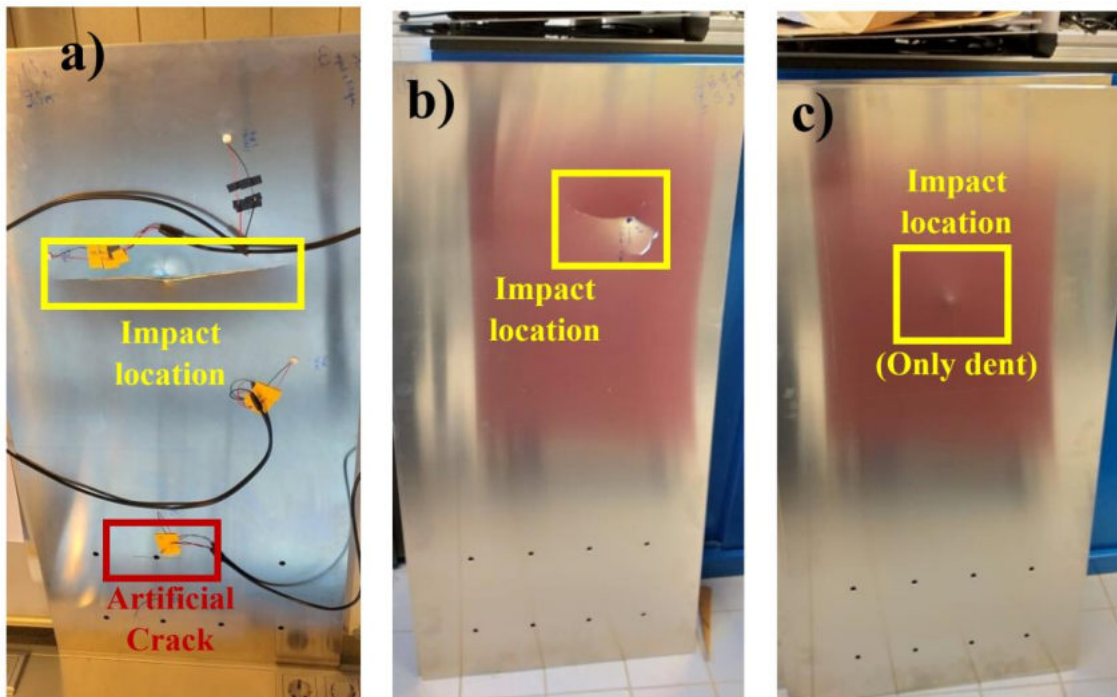


Fig. 5.5.3-2: Example of damaged aluminum plates from scenario a). 3, b). 5, and c). 7 of Table 5.5.3-1, respectively.

The maximum mass m that can be attached to the impactor is 2.4 kg (made of Tungsten), making the total impact mass 4.1 kg (including the mass fixation). During the testing period, a spherical impactor shape and fixator holder with sharp corner were available and therefore these were chosen. This mass is slightly heavier than typical hail, but assuming that the potential energy is fully converted to kinetic energy, this corresponds to impact velocities between 5.60 and 6.26 m/s – which is typically slower than hail impact even during taxiing. As an illustration, the damaged aluminum plates from scenarios 3, 5, and 7 are depicted in Fig. 5.5.3-2a – c.

5.5.4. Damage Analysis

To validate the consistency of each specimen setup, the cumulative correlation coefficient (CC) between the baseline and each damage scenario was calculated. The PZT pulsing actuation used was the trivial 5-cycles Hann sinusoid (at $f = 200$ kHz) and the signal amplitude was recorded by the oscilloscope. As the hotspot sensor at 65|21 cm is used as an actuator, the only sensor which is available at the same location on every specimen is the one located at 84|20 cm. The normalized baseline signals, their envelope, and the corresponding cumulative CC are depicted in Fig. 5.5.4-1.

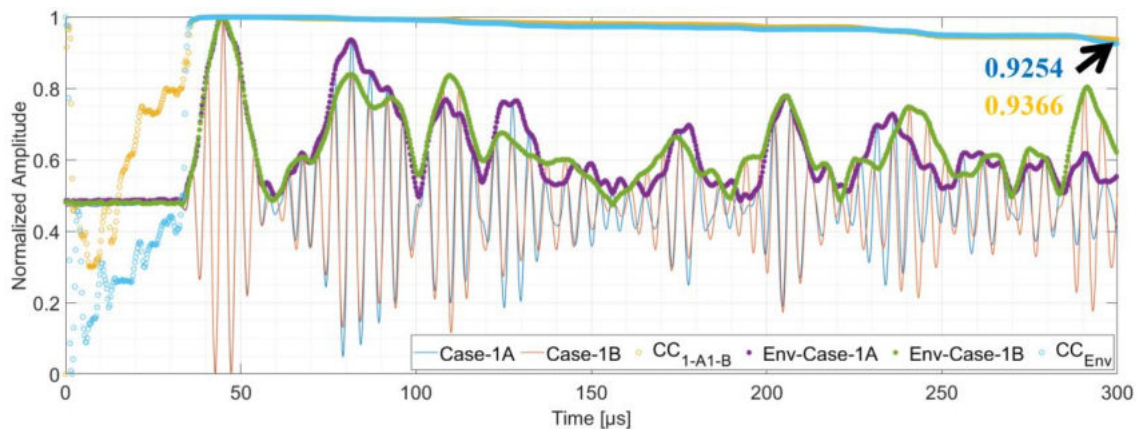


Fig. 5.5.4-1: Baseline for scenario 1A and 1B at sensor located at 84|20 cm.

The CC can be calculated for any time length, for instance the CC of the baseline signals and between the baseline envelopes until 300 μs are 0.9254 and 0.9366, respectively. This is to be expected, since even in an ideal experimental case, two similar and pristine plates can still have a difference in measurement due to their material properties or the presence of minor defects inside the materials.

Moreover, background noise and vibration from nearby equipment in the laboratory can cause a low-frequency signal oscillation. Intuitively, we need to set a band pass the signal between half and double of the resonance PZT frequency (100 – 400 kHz) to isolate both the low and high-frequency noise. Theoretically, an optimization on the band pass is needed, however this was not included as it was not the main purpose of this study. Since the plate dimensions are 100 x 50 cm and assuming that the S_0 -mode is travelling at 5300 m/s, at 300 μs the wavefront would have covered a distance of 2.12m.

5.5.4.1. Hotspot Damage Detection

There is neither a need for damage localization nor damage classification for hotspot SHM placement as both the location and critical damage size will have been predicted according to damage tolerance design. Logically, only the SHM detection function applies here. As can be seen from Fig. 5.5.4.1-1a, there is an 80% decrease in amplitude of both signals and envelope between 36 and 53 μs , which corresponds to a travelling distance of 19 cm for the S_0 -mode which is exactly the distance between the actuator and the sensor.

Accordingly, the CC of the signals and the envelope drops below 0.3 at 36 μs . Assuming that the measurement instrument is working properly (e.g., no defective cables or equipment), such a huge decrease in amplitude (80%) clearly signifies the lack of wave scatter in the propagation path between sensor and actuator. As such, it can be assumed that the crack emerging from the rivet hole has already blocked a significant portion of the wave propagation path. Fig. 5.5.4.1-1b can be explained in the same way.

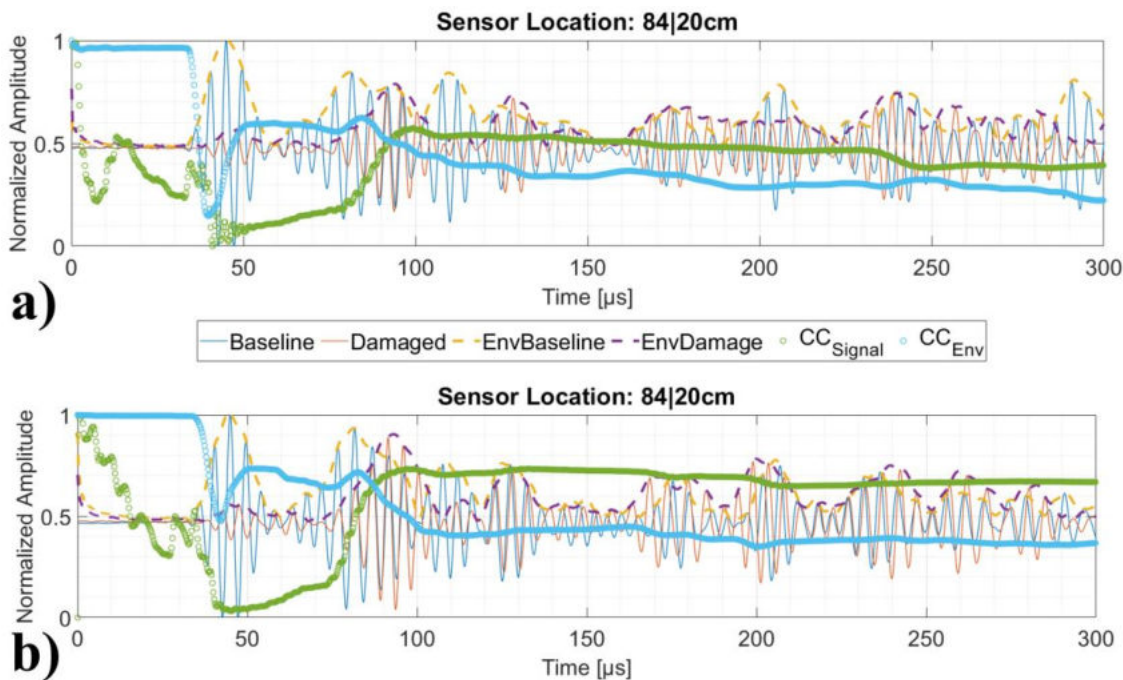


Fig. 5.5.4.1-1a). Amplitude time-series and the CC from Scenario 1A vs 3 and b). Scenario 1B vs 6

5.5.4.2. Impact Localization

While the detection of hotspot damage typically only requires an observation of amplitude change, impact localization also requires signal observation regarding phase shift to extract information to calculate the travel distance of a particular Lamb wave mode.

As an example, a comparison of the signal waveforms, envelopes, and their corresponding CC between the baseline (Scenario 1A) and scenario 2 from network 3+2 at two different sensing locations are given in Fig. 5.5.4.2-1a - b. Since the $CC_{\text{Envelopes}}$ only consider half of the signals and does not consider the

incremental variation of the amplitude within the envelope, it is less sensitive towards the time shifts in the original signal waveforms, as can be seen in Fig. 5.5.4.2-1a between 60 – 90 μs , marked in red dotted rectangle.

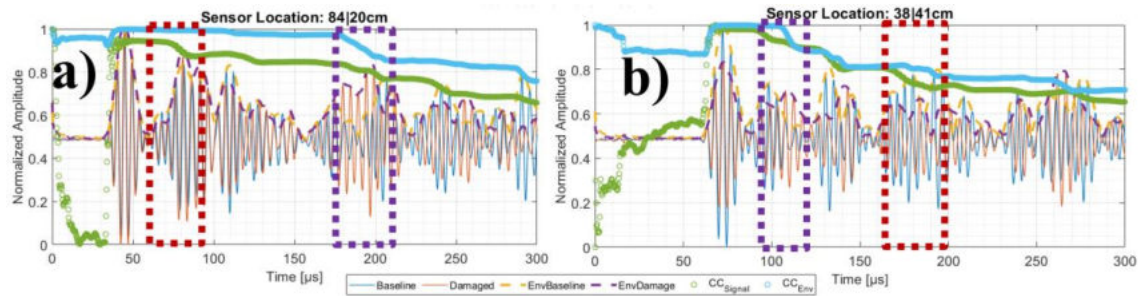


Fig. 5.5.4.2-1: Comparison between signals waveforms, envelopes, and CC of the scenario 1A (Baseline) and 2 (Damaged) at the sensor located at a) 84|20 cm and b) 38|41 cm.

On the other hand, $CC_{\text{Envelopes}}$ is quite sensitive towards amplitude change, especially when the amplitude suddenly drops, such as between 180 μs – 200 μs , marked in Fig. 5.5.4.2-1a in the purple dotted rectangle. During this period, CC_{Signal} also drops, although it is occurring in less dramatic way, i.e., 0.9678 to 0.8713 for $CC_{\text{Envelopes}}$ in comparison with 0.8337 to 0.7942 for CC_{Signal} . Fig. 5.5.4.2-1b can be explained in the same manner.

As stated in section 5.3, the CC of the baseline signals and between the baseline envelopes until 300 μs are 0.9254 and 0.9366, respectively. For this reason, we shall only consider that a damage would occur if the CC dropped below these numbers.

However, the CC would not only drop just because of the damage, we shall also consider all error propagation factors, such as an inhomogeneous amount of applied superglue between the PZT and plate surface, geometrical tolerances such as length, width, thickness of the plate and the rivet holes, the exact coordinate of the sensor placement, potential micro-defects within the plate, etc. For this reason, it would be wise to consider a threshold CC that is slightly below these numbers, but still above 0.5 ($CC = 0.5$ means 50% correlation), otherwise all information that is contained below the threshold will be suppressed, too.

Along with the objective stated in chapter 2 and [Thiene et al. (2016)], the purpose of this chapter is definitely not to propose a novel signal processing method or new feature to calculate DI, but rather to propose a sensor network placement method that is DI-free and can be coupled with any signal processing.

As an example, let us take the threshold of $CC_{\text{Envelope}} = 0.9$ (which is < 0.9254) and a $CC_{\text{Signal}} = 0.8$ (which is < 0.9366) as an example. Thus, if both CC values drop below these thresholds, it is considered as significant. The original waveform and the signal envelope from the baseline (1B) and the damaged plate (scenario 5) and their CC captured by PZTs located at 12|40 cm and 11|12 cm is depicted in Fig. 5.5.4.2-2a – b. After determining the threshold, the first thing to consider is the Time-of-Arrival (TOA).

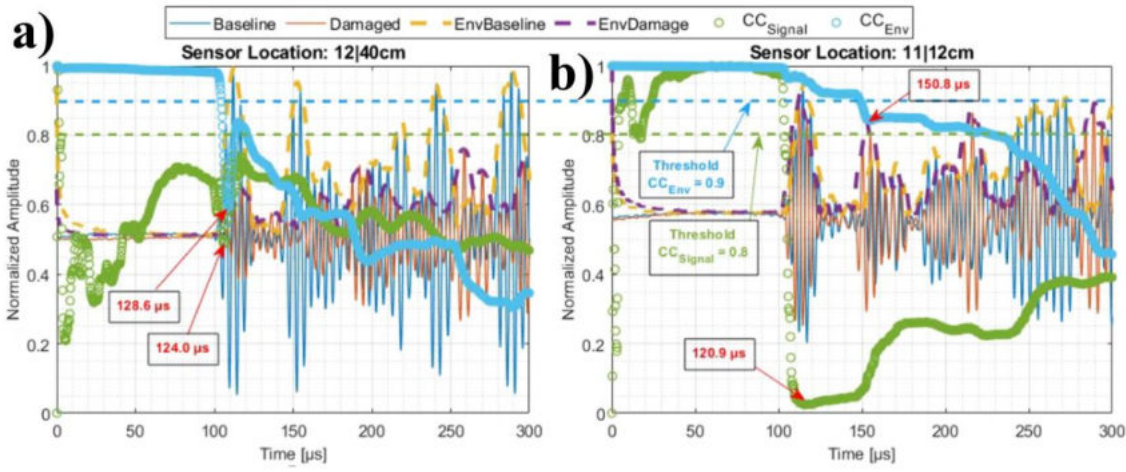


Fig. 5.5.4.2-2: Comparison between signals waveforms, envelopes, and CC of the scenario 1B (Baseline) and 5 (Damaged) at the sensor located at a). 12|40 cm and b). 11|12 cm.

Since both CC values are always changing at every time increment, it is wise to take the TOA where the CC either: 1). reaches its local optima or, 2). Stabilizes as a local plateau. For brevity, in both these cases CC is denoted as CC^* . An example for case 1) is given in Fig. 5.5.4.2-2a, where the TOA of the CC^*_{Signal} and CC^*_{Envelope} are at $124.0 \mu\text{s}$ and $128.6 \mu\text{s}$, respectively. An example for case 2) is given in Fig. 5.5.4.2-2b, where the TOA of the CC^*_{Signal} and CC^*_{Envelope} is at $120.9 \mu\text{s}$ and $150.8 \mu\text{s}$, respectively. The TOA of course might change if the threshold is lowered or raised.

For simplification, let us consider only the TOA of the first local optima and the first local plateau. In future work, the desired technique can be combined with more advanced signal processing such as sparse reconstruction by sum-and-delay technique [Nokhbatolfoghahai et al. (2019)], however in line with the objective: I would like to know how well the hybrid sensor placement method works if it is coupled with conventional signal processing. The localization of the impact damage can be triangulated by calculating the elliptical distance between the actuator and 2 different sensor positions, according to Eq. (5.5.4.2-1a - b).

$$d_{S_0(\text{Path}_1)} = \left\| P_{\text{Actuator}} - P_{\text{Damage}} \right\| + \left\| P_{\text{Damage}} - P_{\text{Sensor}_1} \right\| = v_{S_0} \cdot \text{TOA}_{\text{Sensor}_1} \quad (5.5.4.2-1a)$$

$$d_{S_0(\text{Path}_2)} = \left\| P_{\text{Actuator}} - P_{\text{Damage}} \right\| + \left\| P_{\text{Damage}} - P_{\text{Sensor}_2} \right\| = v_{S_0} \cdot \text{TOA}_{\text{Sensor}_2} \quad (5.5.4.2-1b)$$

In Eq. (5.5.4.2-1a - b), $d_{S_0(\text{Path}_1)}$ and $d_{S_0(\text{Path}_2)}$ are the sums of the Euclidian distance from the actuator to the damage and the damage to the sensor indexed with location 1 and 2 measured from each position P of either the actuator or damage, respectively. P_{Actuator} , P_{Sensor} , and P_{Damage} are the x - and y - coordinates of the actuator, corresponding sensor, and damage, respectively. v_{S_0} , and $\text{TOA}_{\text{Sensor}}$ are the velocity of the S_0 -wavemode and the time of arrival at the corresponding sensor location, respectively. P_{Damage} can be obtained by solving Eq. 5.5.4.2-1a - b simultaneously. By repeating this step for all actuator - sensor pairs, a distribution of predicted P_{Damage} can be obtained. For every solved quadratic equation, there are a maximum of 2 solutions. An example of these calculations

by using TOA from $CC^*_{Envelope}$ for a single actuator – sensor pair is given in Table 5.5.4.2-1. Note that the full table is too long to be presented here.

Scenario	x - y -Coordinate (in cm) of:			TOA of $CC^*_{Envelope}$ (in μ s) from		Predicted Damage x - y - Coordinates (in
	Actuator	Sensor 1	Sensor 2	Sensor 1	Sensor 2	
2	65 21	84 20	10 33	197.9	113.7	22 26 OR 21 39
3	65 21	84 20	24 10	40.5	131.4	79 78 OR 16 25
4	65 21	24 10	10 33	200.5	110.7	<i>Complex Root</i>
5	65 21	84 20	12 40	202.3	128.6	21 31 OR 18 52
6	65 21	35 7	36 41	208.9	150.8	22 56 OR 59 67
7	65 21	84 20	12 40	204.3	131.4	21 31 OR 17 53

Table 5.5.4.2-1: Predicted Damage Location based on TOA-Triangulation from Various Damage Scenario

Not every single solution is useful, for instance in scenario 1A-3, the first predicted coordinate at 79|78 cm lies outside of the plate, thus the other solution which is at 16|25 cm is taken as the accepted predicted location. In scenario 1A-4, both solutions are complex roots so they cannot be considered anymore. In this case, only the roots that fulfill the constraint (i.e., positive real numbers within the dimensions of the plate) are taken as an accepted solution. The reason that sometimes there are no accepted solutions is because in Eq. (5.5.4.2-1a – b), all TOA(CC^*) are multiplied by the S_0 -wavemode velocity, assuming that the dominant Lamb mode for excited radial PZT is the S_0 -mode.

This is an oversimplification because generally, both fundamental Lamb modes are always present, i.e., after the wavefront comes in contact with any inhomogeneities such as rivet holes and plate boundaries, wave mode conversion occurs. To avoid this, a Lamb wave mode separation technique such as [Xu and Ta (2012)] can be used to sort in which group the TOA belongs to. For now, it is enough to consider all accepted predicted damage locations and to calculate the distribution of these predictions.

The distribution of the predicted damage locations is summarized in Fig. 5.5.4.2-3a – d, which shows predicted damage locations from scenarios 2, 4, 5 and 7, respectively. As can be seen from Fig. 5.5.4.2-3a and 4a, even a simple algorithm can easily localize a large damage size (about $\frac{1}{2}$ of plate impacted, i.e., 30 cm), although there is an area that was not covered by the distribution as the damage itself is quite large. Furthermore, for the localization of smaller impact damage (about $\frac{1}{4}$ plate, i.e. impacted in scenario 4 and 5) which are depicted in Fig. 5.5.4.2-3b – c and Fig. 5.5.4.2-4b – c, respectively, the determination of damage location based on TOA calculation outputs provides a relatively reliable localization.

This is in contrast with small impact, which barely causes a smaller dent on the surface where in this case, even the denser network is not able to predict the damage location in a sufficient manner as depicted in Fig. 5.5.4.2-3d and 5.5.4.2-4d.

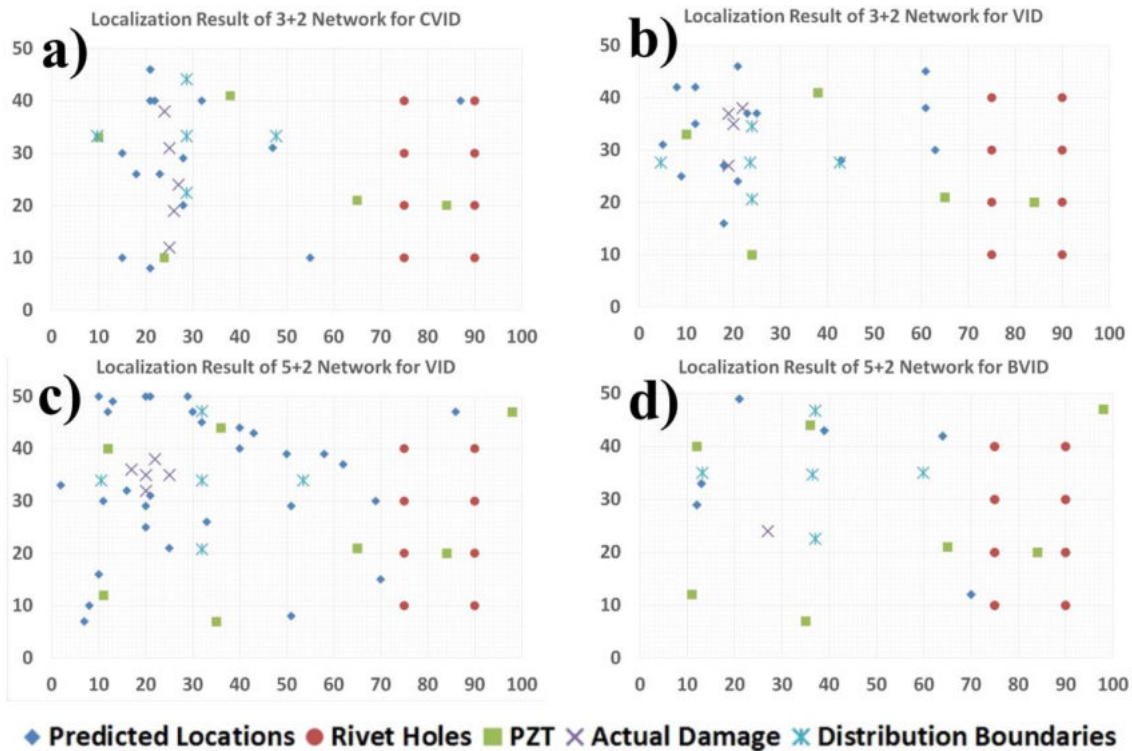


Fig. 5.5.4.2-3 (clockwise from top left): Damage localization result from scenario a). 2, b). 4, c). 5 and d). 7.

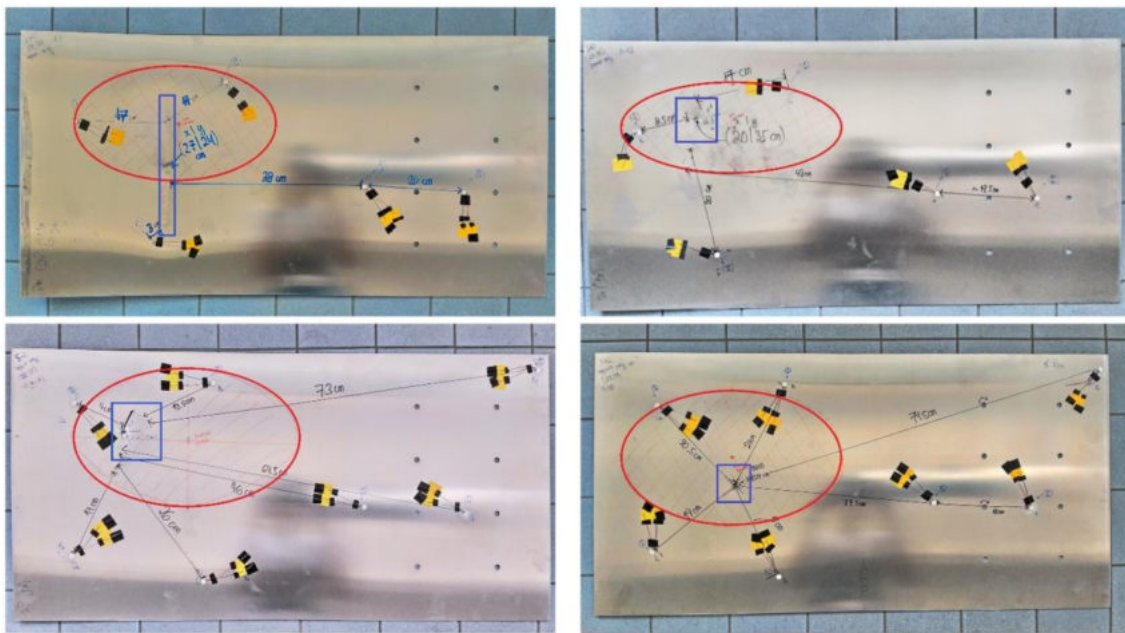


Fig. 5.5.4.2-4 (clockwise from top left): Real plate from scenario a). 2, b). 4, c). 5 and d). 7. The blue rectangles indicates the impact damages where the red ellipse are the distributions of predicted damage locations. Note: the aluminum surface works like a mirror, giving unwanted reflection in the photo

Taking a closer look into the quantification of the localization performance, I calculated several criteria as given in Table 5.5.4.2-2:

1. Averaged Euclidian distance δ : the distance between each predicted damage location $P_{x,y}$ (which is just the distribution mean) and the epicenter of the actual damage epicenter $A_{x,y}$ without considering the multi-site damage

2. Mahalanobis distance M : the distance between actual damage epicenter without considering the multi-site damage and the probability distribution of all predicted damage locations.
3. Standard deviation σ_x and σ_y for both x - and y - axis
4. Percentage R which is ratio between elliptical area covered by vertex σ_x and σ_y , divided by plate size, which is $100 \times 50 \text{ cm} = 5000 \text{ cm}^2$, for scenario 2, 4, 5, and 7: these elliptical areas can be seen in Fig. 5.5.4.2-4 already.

Scenario	$A_{x,y}$ [cm cm]	$P_{x,y}$ [cm cm]	δ [cm]	M [unitless]	$\sigma_x \sigma_y$ [cm cm]	R
2	27 24	29 33	9.5	4.438×10^{-3}	19 11	13.0%
3	27 24	24 28	5.0	4.665×10^{-3}	19 7	8.3%
4	20 35	27 34	6.8	2.990×10^{-3}	20 8	10.2%
5	20 35	33 33	12.8	3.824×10^{-3}	21 12	16.2%
6	27 24	26 30	5.8	2.033×10^{-3}	21 13	17.2%
7	27 24	37 35	14.3	1.531×10^{-3}	23 12	17.8%

Table 5.5.4.2-2: Euclidian and Mahalanobis distance between the predicted and actual impact epicenter

As it can be seen from Table 5.5.4.2-2, the Euclidian distances between the predicted and actual epicenter varies between 5.0 cm and 14.3 cm. Note that in scenarios 2 – 5, the damage is quite large, i.e., not only dents and therefore it occupies multiple locations and the actual damage locations are covered by R .

In scenarios 6 and 7, R is large, but the damage size is small (dents $\varnothing = 1 \text{ cm}$), which poses a limitation of the damage detection algorithm, because the Lamb wave mode at this particular PZT frequency (200 kHz, wavelength $\lambda = 2.65 \text{ cm}$) is not a good match with the damage size. The difference between scenarios 6 and 7 is only the artificial fatigue crack located at 75|20 cm for scenario 6 has been described in section 5.3.1. For scenario 7, the average error on Euclidian distance is 14.3 cm and the ratio R is 17.8%.

Practically, when scaling this hybrid approach for a larger-sized structure (e.g., 5 x 5 m), then at least 4 m² (about 1/6 or 16.7% of the surface) must still be scanned manually. As stated previously, the sensor network pattern is designed to work independently of any signal processing, thus, to increase accuracy in the future, the signals could be first separated by using method described in [Xu and Ta (2012)] and then processed further by using the delay and sum method for sparse reconstruction described in [Nokhbatolfighahai (2019)].

It is often forgotten that the purpose of SHM is not to replace NDT completely, but to determine whether a further NDT inspection of a certain aircraft part is needed during unscheduled maintenance or not. Therefore, I believe that by reducing the inspection man-hours by at least 83.3%, we may think that the hybrid sensor placement method with a minimum number of sensors for hotspot and global damage detection contributes to design strategies for Lamb wave SHM.

5.6. Chapter Summary

This chapter demonstrated that a sensor network topology for hotspot SHM for detection of predictable crack location can be merged with the probabilistic approach without sacrificing too much of the global sensitivity. To do so, first, the hotspot sensor locations are determined according to the largest centroid based on blob detection algorithm. Then, to determine the sensor positions for detecting random damage, 5 search algorithms were compared: global random search, greedy methodology, and 3 common metaheuristics: genetic algorithm (GA), particle swarm optimization (PSO), and simulated annealing (SA).

Global random search has the lowest performance, and GA and PSO are on par, and they have the best performance, while the greedy methodology and SA have a search performance which lies in between GA/PSO and global random search. Accordingly, as per the *No Free Lunch* theorem, both GA and PSO took the most computational resources - this can be either in time or space while the random search took the least computational resources. Unsurprisingly, the required computational resources of the greedy method and SA also lie in between global random search and GA and PSO.

Since the specimen size used in this work was not too large and the computational time for every iteration search was below 1 hour, the genetic algorithm is used to determine the integrated sensor network topology. This hybrid approach demonstrated that sensor networks can detect fatigue cracks and locate randomly occurring damage, if these do not occur at the same time. I believe this likelihood is small, but nevertheless, it might be interesting in future study to understand the probability of fatigue crack and impact occurring at the same time.

Given the results in section 5.3 to 5.5, it is safe to think that the hybrid approach based on blob detection algorithm and search metaheuristics can partially address the sensor positioning problem in active ultrasonic SHM in scalable manner - especially when the detection requirement is not too high. However, for placing a much larger numbers of sensors in a larger and complex structure, the suggestion would be using the greedy algorithm instead of the genetic algorithm to compensate for the network performance and the computational effort required.

Literature

Cormen TH, Leiserson CE, Rivest RL, Stein C. *Introduction to Algorithms* (3rd Ed.). Ch. 16: Greedy Algorithms. MIT Press, Cambridge / London (2009).

Dentith M, Mudge ST. *Geophysics for the Mineral Exploration Geoscientist* (1st Ed.). Cambridge University Press, Cambridge (2011).

Drinkwater BW, Castaings, Bernard Hosten. *The Measurement of and Lamb Wave Attenuation to Determine the Normal and Shear Stiffnesses of a Compressively Loaded Interface*. J Acoustical Society of America. Vol. 113: 3161 (2003).

- Ewald V. *Post-Design Damage Tolerance Enhancement of Primary Aircraft Structures by Ultrasonic Lamb Wave Based Structural Health Monitoring (SHM) System*. MSc Thesis. Universität des Saarlandes, Saarbrücken (2015).
- Ewald V, Groves RM, Benedictus R. *Integrative Approach for Transducer Positioning Optimization for Ultrasonic Structural Health Monitoring for the Detection of Deterministic and Probabilistic Damage Location*. Intl J of Structural Health Monitoring. DOI: 10.1177/1475921720933172 (2020).
- Hopcroft JE, Motwani R, Ullman JD. *Introduction to Automata Theory, Languages, and Computation* (3rd Ed.). Addison-Wesley Longman Publishing Co., Inc., Boston (2006).
- Hung MH, Shu LS, Ho SJ, Hwang SF, Ho SY. *A Novel Intelligent Multiobjective Simulated Annealing Algorithm for Designing Robust PID Controllers*. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans. Vol. 38(2): 319-330 (2008).
- Kasama H, Takemoto M, Ono K. *Attenuation Measurement of Laser Excited S_0 -Lamb Wave by the Wavelet Transform and Porosity Estimation in Superplastic Al-Mg Plate*. Hihakai-Kensa (Japanese J Nondestructive Testing). Vol. 49: pp 269-276 (2000).
- Kennedy J, Eberhart R. *Particle Swarm Optimization*. Proc. IEEE Intl Conf on Neural Networks IV (1995).
- Kerber F, Sprenger H, Niethammer M, Luangvilai K, Jacobs LJ. *Attenuation Analysis of Lamb Waves Using the Chirplet Transform*. J on Advances in Signal Processing: 375171 (2010).
- Kramer O. *Genetic Algorithm Essentials*. In: Studies in Computational Intelligence. Springer Intl Publishing, Cham (2017).
- Li S, Jin F, Zhang W, Meng Z. *Research of Hail Impact on Aircraft Wheel Door with Lattice Hybrid Structure*. J Physics Conf Series. Vol. 744: 012102 (2016).
- Mizutani Y, Suenaga K, Todoroki A, Suzuki Y. *Estimation of Viscoelastic Properties by Lamb Wave Analysis*. Proc 31st Conf of the European Working Group on Acoustic Emission (EWGAE), Dresden (2014).
- Nokhbatolfoghahai A, Navazi HM, Groves RM. *Use of Delay and Sum for Sparse Reconstruction Improvement for Structural Health Monitoring*. J Intelligent Material Systems and Structures. pp 1 – 13 (2019).
- O'Rourke J. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York (1987).
- Ono K, Gallego A. *Attenuation of Lamb Waves in CFRP Plates*. J Acoustic Emission. Vol. 30: 109-123 (2012).
- Pohl J, Willberg C, Gabber U, Mook G. *Experimental and Theoretical Analysis of Lamb Wave Generation by Piezoceramic Actuators for Structural Health Monitoring*. J Experimental Mechanics. Vol. 52: 429-438 (2012).
- Rastrigin, LA. *The Convergence of the Random Search Method in the Extremal Control of Many Parameter System*. J Automation and Remote Control. Vol. 24: 1337-1342 (1963).
- Schubert KJ, Hermann AS. *On Attenuation and Measurement of Lamb Waves in Viscoelastic Composites*. J Composite Structures. Vol. 94: 177-185 (2011).
- Schmitt LM. *Fundamental Study: Theory of Genetic Algorithms*. J Theoretical Computer Science. Vol. 259: 1-61 (2001).
- Su Z, Ye L. *Identification of Damage Using Lamb Waves: From Fundamental to Applications*. Springer-Verlag, Berlin / Heidelberg (2009).
- Thiene M, Sharif Khodaei Z, Aliabadi MH. *Optimal Sensor Placement for Maximum Area Coverage for Damage Localization in Composite Structures*. Smart Materials and Structures. Vol. 25: 095037 (2016).
- USDOT Federal Aviation Administration (FAA). *Detection and Characterization of Hail Impact Damage in Carbon Fiber Aircraft Structures*. Final Report DOT/FAA/TC-16/8 (2017).
- Wolpert DH, Macready WG. *No Free Lunch Theorems for Optimization*. IEEE Transactions on Evolutionary Computation. Vol. 1: 67-82 (1997).
- Xu K, Ta D. *Mode Separation of Lamb Waves Based on Dispersion Compensation Method*. J Acoustical Society of America. Vol. 131: 2714-2722 (2012).
- Zhao X, Gao H, Zhang G, Ayhan B, Yan F, Kwan C, Rose JL. *Active Health Monitoring of an Aircraft Wing with Embedded Piezoelectric Sensor/Actuator Network: I. Defect Detection, Localization and Growth Monitoring*. J Smart Materials and Structures. Vol. 16: 1208- 1217 (2006).

6. Deep Learning for Structural Health Monitoring

This chapter partially contains the work that has been published in:

1. Ewald V, Groves RM, Benedictus R. *DeepSHM: A Deep Learning Approach for Structural Health Monitoring Based on Guided Lamb Wave Technique*. Proc. SPIE Smart Structures + NDE: 10970, Denver (2019).
2. Ewald V, Venkat RS, Asokkumar A, Benedictus R, Boller C, Groves RM. *Perception Modelling by Invariant Representation of Deep Learning for Automated Structural Diagnostic in Aircraft Maintenance: A Study Case using DeepSHM*. J Mechanical System and Signal Processing. Vol 165: 108153 (2022).

As stated previously in chapter 2, the high-level problem formulation that should be asked from the NDT & SHM community is: *what is the feasibility of incorporating artificial intelligence (AI) as a design tool for an automated diagnostic within predictive maintenance – and if so, in what way?* For this chapter, let us focus on the following lower-level research problem formulations mentioned in section 2.3 of this dissertation:

1. *“Investigation whether deep learning can be used to treat the Lamb wave signal – and if so, does it have certain theoretical justification? What would be the pros and cons when of deep learning to treat Lamb wave signals and what would be the consequence for the design and manufacturing of the SHM system? Further considerations on certain aspects from computational neuroscience for processing the Lamb wave signal should also probably taken into account.”*
2. *“When combining the sub-problems to reconstruct the final solution: given certain sensor topology, what would be the training behavior for different sensors and whether some perspective from computational neuroscientific could also be considered as well?”*

Section 2.4 introduced some formalism such as automata, logic, and their consequences for diagnostics. These formal concepts are important to understand so that we, the empirical science community which consists of physicists, chemists, and biologists and various types of engineers, do not merely accuse the formal science community (e.g., mathematicians, computer scientists, and software engineers) with prejudices such as *“AI is a total Blackbox”*.

Understanding the concepts of rules, logics, and languages as well as high-level abstraction of (human) perception and common sense are sometimes unfortunately taken for granted by the empirical science community. Only when the formal concepts have been understood, one can successfully proceed to incorporate domain specific knowledge with a domain agnostic approach.

6.1. Research Outline Recap

The theoretical background of machine and deep learning as well as features and invariant representations has been described in sections 3.4 and 3.5 of chapter 3. To begin this chapter, assume the following premises to determine the AI architecture strategy:

1. More complexity in geometry and material properties would require enhanced signal processing to capture signal features,
2. A pure physical model is normally more powerful, but typically requires a lot of effort and sometimes it also idealizes some assumptions that might not always correspond to the real-world situation, and
3. A pure statistical model can only find correlation, but not causation (also known as the ‘black-box property’), thus conclusions are difficult to understand. Therefore, a compromise between a physical and a statistical model must be made in order to further progress the advancement of automated damage detection, be it in SHM or NDT.

The previous work [Ewald et al. (2018a)] demonstrated how to bias CNN with appropriate aerospace domain knowledge for both NDT and SHM applications. This was also in line with the approach proposed by [Gardner et al. (2020)]. For SHM applications using active Lamb waves, I previously proposed a hybrid model called DeepSHM framework [Ewald et al. (2019)]. Specifically, it is a statistical signal modelling based on deep learning biased by a physical nature and easily worked for a complex signal classification.

Like any other deep learning algorithm, the advantage of DeepSHM is its *agnosticism*: it treats any input, and it gives any output given the input, so no matter how complex the signal is, the classification accuracy is tendentially very high. The biggest disadvantage of DeepSHM is also its *agnosticism*: for any given bad input, the outcome would be a poor output, which is known in computer science as a *Garbage in – Garbage out* [Kim et al. (2016)] process.

While deep learning would work given any input sequence, to align this research with the previous work and for this reason, the use of DeepSHM is limited solely for active Lamb wave based SHM. One specific problem that was encountered in [Ewald et al. (2019)] was that some of the algorithms could not make a distinction between signals that come from a slightly geometrically similar defect, i.e., statistically speaking, they come from the similar distribution. The reason for this was the physical limitation that one particular ultrasonic wavelength is in general only suitable for detecting damage in a certain size range. The proposed hypothesis to overcome this problem are:

1. *Applying broadband frequency excitation since this will involve a broader wavelength distribution.*
2. *Varying the sensing locations to potentially obtain more information.*

To do so, a typical deep learning workflow will be performed: the classification performance metrics in the confusion matrix will be compared with given

captured signals from different sensing locations. One associated problem with signal processing with multiple sensing locations is that it could result in different sensor responses which might give conflicting predictions (e.g., no damage based on the response of one sensor and damage based on the response of the other sensor). This leads to the following research questions:

1. *How much do the varying sensing location and the different sensing representations of the time-frequency Lamb wave signal influence the deep learning training behavior?*
2. *Given the ‘a posteriori knowledge’ from question (1), what consequences can be drawn for engineering applications in SHM and why should this approach work?*

This chapter is organized as follows: my thoughts about modelling SHM perception from a neuroscientific perspective is given in section 6.2. The necessary methodologies (such as generating data with simulation, signal pre-processing, hyperparameter configuration, etc.) is given section 6.3. The results and discussion, as well as the concept validation are given in section 6.4 and 6.5, respectively. Finally, the conclusion is given in chapter 6.6.

6.2. Theoretical Background

The general formulation of diagnostic has been described in section 3.2 and the relationship between inhomogeneities i in the medium ψ can as a function of time dependent observable $X_\lambda(t)$ containing phenomenon λ at time t is:

$$\hat{f}(\psi_i) \approx X_\lambda(t) \equiv X(t) \quad (6.2-1)$$

The observable $X_\lambda(t)$ is the measured signal and can be mathematically expressed in either vector form or more generically a finite-dimensional observation tensor. Every observation X perpetually changes for any given domain parameters. Thus, it would be naturally logical to describe the behavior of X as probabilistic variables rather than as deterministic ones. For brevity, let us assume the null hypothesis h_θ where the existence of λ is caused only by changing parameters in ψ . Due to the first assumption of the stochastic nature of observation X , the relation can be formulated via Bayes conditional probability P :

$$P(h_\theta(\psi_i) | X_\lambda(t)) = \frac{P(X_\lambda(t) | h_\theta(\psi_i)) \cdot P(h_\theta(\psi_i))}{P(X_\lambda(t))} \quad (6.2-2)$$

Where in Eq. 6.2.1-3, $P(h_\theta(\psi_i) | X_\lambda(t))$ is the posterior probability of the existence of i -th inhomogeneities ψ given observations $X_\lambda(t)$, $P(X_\lambda(t) | h_\theta(\psi_i))$ is the prior probability where $X_\lambda(t)$ occurs given the hypothesis h_θ . $P(X_\lambda(t))$ and $P(h_\theta(\psi_i))$ are the marginal probabilities of observing $X_\lambda(t)$ and $h_\theta(\psi_i)$ independently, respectively. Furthermore, in Eq. 6.2.1-3, θ is the fitting parameters (which in deep learning are normally called neural network weights) that are to be optimized during the learning process. Given the fact that in most of the case, the observable

X is multi-dimensional, it is logical to formulate the problem as a multivariate distribution rather than a uni- or bivariate distribution.

Further, let us consider the most suitable expression of the joint probability distribution. Naturally, there is an infinite number of possible combinations within ψ (e.g., for the length of a crack, the size of delamination, or corrosion depth) and that a very small variation within ψ normally only causes small variation. So, instead of a discrete probability mass distribution, the probability density function is the more suitable formulation for a joint distribution in diagnostics since it expresses the density of a continuous random variable.

Recall section 3.4.1 from chapter 3: when considering any machine learning algorithms, the following questions naturally arise: which learning problems can be solved efficiently and which are easier to solve than others? How many N training samples do we need, and which parameters θ must be tuned during the learning process? In computer science, the proper intuition would be the learnability of the function itself, as explained from Lemma 3.4.1-1 [Valiant (1984, 2013)]: The probably approximately correct (PAC)-learning framework [Valiant (1984, 2013), Gormley (2016), Moran and Yehudayoff (2016)].

[Mitchell (1997)] defined the algorithm of the machine learning process as: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E ”. Specifically, this means that a trained algorithm, which is just the generalizer L in disguise from Def. 3.4.1-2 [Wolpert (1990, 1992)], that has learned during machine learning process E and is said to be able to generalize on the class of tasks T from a certain probability distribution,

In the diagnostic realm, task T is the diagnostic itself, that is retrieving the information from the observable variables X_ψ regarding the damage state, experience E is the iterative process for enhancing the algorithm to increase the accuracy performance of the trained algorithm that best generalizes the distribution over X_ψ . The probability for the i -th class of information given the X_ψ in k -dimensional Hilbert space is typically written as a logit or sigmoid function [Mitchell (1997), MIL-HDBK-1823 (2009)] and can be generalized in a SoftMax function:

$$P(h_{\theta}(\psi_i) = i | X_{\lambda}(t)) = \frac{\exp([X_{\lambda}(t)]^T \cdot \theta_i)}{\sum_{k=1}^K \exp([X_{\lambda}(t)]^T \cdot \theta_k)} \quad (6.2-3)$$

The posterior of $P(h_{\theta}(\psi_i)|X_{\lambda}(t))$ is taking real value within $[0, 1]$, but can also sometimes be expressed as percentage value. The posterior is maximized by minimizing the information difference, which can be referred to either as loss, cost or error between the predicted value $h_{\theta}(\psi_i)$ and the true information contained X . The central task of machine learning is to minimize the loss function by iteratively adjusting θ . Depending on the problem formulation, different loss functions must be defined. For instance, in a regression problem, the mean squared error is typically chosen, while when looking further into a classification

problem, a cross-entropy loss is chosen since it maps a logistic output between 0 and 1 and thus is an appropriate measure to calculate a similarity between two distributions. The cross-entropy $H(p,q)$ between the true distribution $p(X_\psi)$ and the estimated distribution $q(X_\psi)$ is defined as [Rubinstein and Kroese (2004)]:

$$H(p,q) = - \sum_{X_\lambda(t)} p(X_\lambda(t) = \psi_i) \cdot \log[q(X_\lambda(t) = h_\theta(\psi_i))] \quad (6.2-4)$$

Within the context of classification problem, the minimization of the cross-entropy loss J_θ for N training samples can be rewritten from Eq. 6.2-4 as:

$$\arg \min J_\theta = \min \left[- \frac{1}{N} \left(\sum_{i=1}^N p(X_{\psi_i} = \lambda) \cdot \log[q(X_{\psi_i} = h_\theta(\lambda))] \right) \right] \quad (6.2-5)$$

Depending on the algorithm complexity, θ can be infinitely dimensional, meaning that from Eq. 6.2-5, it can be easily assumed that learning in machine learning is a NP-hard as it contains a highly dimensional combinatorial problem that is $\sim O(\theta!)$. While it is impractical to reach $J_\theta = 0$, learning means we are striving for $J_\theta \rightarrow 0$, so there must be an upper limit where the probability measures converge [Sethuraman (1961)] given by Lemma 6.2.1-1, while the consequence can be summarized in Corollary 6.2.1-1.

Lemma 6.2.1-1: Converging probability measure [Sethuraman (1961)]

Let $C \in U$ and P_1, P_2, \dots , be the sequence of probability measures defined on a measurable space (M, V) , then P_i converges strongly to $P(P_i \rightarrow P$ in symbols) if $P_i(C) \rightarrow P(C) \forall C \in V$.

Corollary 6.2.1-1: A proper probabilistic generalizer (upwardly compatible HERBIE) is said to have an upper converging generalization bound if the probability measures over the defined measure spaces strongly converge according to Lemma 6.2-1.

One statistical metric that is useful to indicate the upper convergence limit of the generalization bound is the statistical classification accuracy A , defined as:

$$A = \frac{\sum TP + TN}{\sum (TP + TN + FP + FN)} \quad (6.2-6)$$

Where in Eq. 6.2-6, TP, TN, FP, FN is the true positive, true negative, false positive, and false negative rate, respectively. The definition of TP, TN, FP , and FN can be found in [Ting (2017)]. As mentioned before, it should be noted that often it is useful to influence the algorithm by domain knowledge, as demonstrated in previous work [Ewald et al. (2018a)].

Biasing the algorithm also includes determining the data distribution which is fed into the learning algorithm to have interpretable outcome and to avoid a **Garbage in – Garbage out** process. Therefore, it is very important to determine which task T the algorithm should perform at the beginning. For this reason, we can expand

the definition of an upwardly compatible generalizer in Def. 3.4.1-2 on deep learning by considering Def. 3.4.1-3 regarding the generalization gap and the Rademacher complexity \mathfrak{R} . As explained in section 3.4.1, the Rademacher complexity and VC-dimension [Goldberg and Jerrum (1995), Clayton (2014)] are a measure of the richness of a class of real-valued functions. To guarantee the upper bound of the generalizer, [Kawaguchi et al. (2017)] proposed a theorem via the validation error as given in Lemma 6.2.1-2.

Lemma 6.2.1-2. Generalization bounds of deep learning via validation [Kawaguchi et al. (2017)]

Let the sampling space S_N be split according to the true distribution $P_{(x,Y)}$ in S_N^{train} and S_N^{val} that denote the training and validation datasets, respectively.

Let $\mathfrak{N}_{L,i} = R_L - J_{\#}(L(x_i), y_i)$ for each pair $(x_i, y_i) \in S_N^{val}$. For any $\delta > 0$, with probability at least $1 - \delta$, then the following holds $\forall L \in L^{val}$:

$$R_L \leq R_{L(S_N^{val})} + \frac{2 \cdot \sup |\mathfrak{N}_{L,i}| \cdot \ln \left(\frac{|L^{val}|}{\delta} \right)}{3N_{val}} + \sqrt{\frac{2 \cdot \sup \mathbb{E}(\mathfrak{N}_{L,i}^2) \cdot \ln \left(\frac{|L^{val}|}{\delta} \right)}{N_{val}}}$$

where L^{val} is defined as a set of models L that is independent of a held-out validation dataset S_N^{val} , but can depend on the training dataset S_N .

Lemma 6.2.1-2 practically explains why deep learning could generalize well if the generalization bound is reached, which is guaranteed by the converging validation error despite possible sharp local minima or non-robustness. Thanks to corollary 6.2.1-1, we can now summarize the conclusion in corollary 6.2.1-2.

Corollary 6.2.1-2: *An upwardly compatible deep learning model reaches its upper generalization bound when the validation loss converges.*

[Kawaguchi et al. (2017)] mentioned following worst cases:

$\sup \mathfrak{N}_{L,i} = 1$	(6.2-7a)
$\sup \mathbb{E}(\mathfrak{N}_{L,i}^2) = 1$	(6.2-7b)

Given a large hyperparameter cardinality (e.g., 10^6) and 1000 training epochs in a larger dataset $N_{val} = 10000$, the second and third terms of the equations sum up only to 6.94%:

$\frac{2 \cdot \sup \mathfrak{N}_{L,i} \cdot \ln \left(\frac{ L^{val} }{\delta} \right)}{3N_{val}} + \sqrt{\frac{2 \cdot \sup \mathbb{E}(\mathfrak{N}_{L,i}^2) \cdot \ln \left(\frac{ L^{val} }{\delta} \right)}{N_{val}}} \leq 6.94\%$	(6.2-8)
---	---------

In the non-worst case, this figure decreases to 0.49%. Extrapolating this for the case where the dataset amount is way smaller (e.g., $N_{val} = 100$) with a less complex

deep neural network architecture with a smaller cardinality (e.g., 10^2) and 100 training epochs, we have a probability of 0.9 that:

$$R_L \leq R_{L(S_N^{val})} + 55.66\% \tag{6.2-9}$$

This means that there is 55.66% chance that the model does not generalize. The implication of an upwardly generalizable of a diagnostic algorithm, is to increase the true positive (TP) and decrease the false negative (FN , equivalent to statistical error of type II) detection rate, thus maximizing the probability of detection (POD), sometimes called the sensitivity or recall rate, defined as:

$$POD = \frac{\sum TP}{\sum (TP + FN)} \tag{6.2-10}$$

The current standard practice for diagnostic NDT according to [MIL-HDBK1823 (2009)] is a $POD = 0.9$ (or 90%) within 95% statistical confidence σ , although this might not be suitable for diagnostic SHM [Hayo et al. (2011)]. To be more generic with the formulation, we can redraw the proposition of [Ooijevaar (2014)] for an active Lamb wave SHM system, shown in Fig. 6.2-1. The processing framework of the observable X_ψ containing damage information captured by the actor π using a trained deep learning to predict the hypothesis $h_\theta(\lambda)$ can be seen in Fig. 6.2-2.

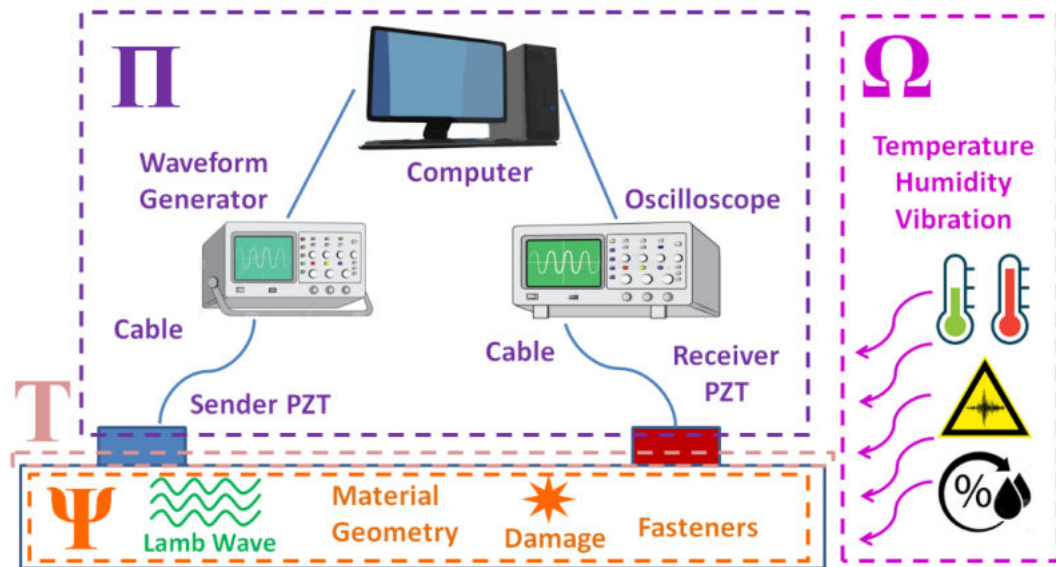


Fig. 6.2.1-1: Diagnostic SHM by using an active guided Lamb wave as physical phenomenon.

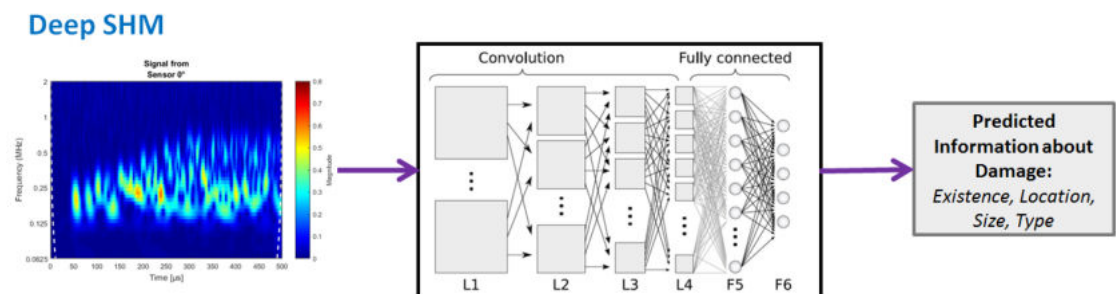


Fig. 6.2.1-2: DeepSHM Framework

Like any machine learning process, the central task of the DeepSHM framework is to find generalizable parameters θ to fit the correlation between $X_\lambda(t)$ and $h_\theta(\psi_i)$, where in guided Lamb wave SHM, $h_\theta(\psi_i)$ is defined as the hypothesis of the damage information contained in medium domain ψ that is influenced by interaction between phenomenon Lamb wave (λ) and the damage itself.

6.2.2. Model Abstraction of DeepSHM Behavior

As SHM itself is inspired by biology, maybe DeepSHM should not just be capable of actuating and sensing, but it should have its own perception. Section 3.5 mentioned the definition of *machine perception* by [Pendleton et al. (2017)]. The definition of machine perception can then be simply adapted for the SHM domain: *Perception in diagnostic SHM is a fundamental function to enable the full functionality of an autonomous damage detection system, which provides the SHM system with crucial information on changes in the sensory input (such as changes in amplitude, frequency, or phase-shift).*

Interestingly for any ultrasound-based technology, such perception is similar to the hearing process in our brain. We might be staying in a room when someone is calling our name, but no matter what frequency or volume our name was called, we broadly understand it is our name. So, there must be an assumption of the existence of shared invariant features between each name calling. Same with Lamb waves, the recorded signals would be likely to have entirely different amplitudes, phase shift, and frequency from each other if it is recorded at different sensing location.

The explanation of invariant representation and its importance to Lamb wave signal processing can be found in section 3.5 and Lemmata 3.5.1-1 – 3.5.1-4. Recall two assumptions made by *The Future of AI* challenges [Merck (2019)], in which they were looking for a formalization of the cortical algorithm, a mechanism which might be able to imitate the pattern recognition process that is believed to be taking place in the grid neuron, which is located in the mammalian neocortex:

1. *“Hierarchical structure: Entities are hierarchically structured, which means that an entity E is either fundamental (i.e., it cannot be broken down any further) or it is composed of other entities E_1, E_2, \dots, E_i such that every perception p of E is associated with a set of perceptions p_1, p_2, \dots, p_i .”*
2. *“Entity conservation over time: Subsequent perceptions in time are (usually) generated by the same set of entities.”*

The reason to consider this fringe idea from neuroscience is because it gives us a hint about how we should treat the SHM signals. Concretely, given any arbitrary structure with k -sensors, let us adapt these two assumptions as follows:

1. ***Hierarchical Representation of signals.*** The signals can be captured anywhere in the structure, but they are represented separately. As such, treating such images would require multiple learners in parallel as no sensor data fusion is required because each node acts as an independent actor. Thus, multiple outputs come from k -sensory inputs forming k -

possible perceptions. In this case, the invariant representation is the atomic decomposition of each observation.

2. *Conserved Entity over Time*. In this case, the observable $X(t)$ is represented as a single entity. Assuming a bijective projection as given by Lemma 3.5.1-4, any arbitrary signal can only be associated with that structure, no matter where the sensors are placed. As such, k -sensory inputs can be represented in a stack of a k -dimensional array. Consequently, treating such images would only require a single model and thus DeepSHM would only act as a single actor. In this case, each layer of the k -dimensional image becomes the invariant representant of the whole observations.

The experimental design in the next sections will follow either one of the above-mentioned assumptions.

6.3. Methodology

This section will describe the methodology used to obtain the result. The simulation parameters to obtain a Lamb wave signal are described in section 6.3.1. Section 6.3.2 explains the data processing method while in section 6.3.3, the input based on the assumptions of entity representation mentioned in section 6.2.2 will be explained. The training setup and parameters such as hardware and libraries selection, loss function and hyperparameter optimization are described in section 6.3.4.

6.3.1. Simulation Setup

I would like to give the credit for the simulation part to my research colleagues from Saarland University, Germany: Aadhik Asokkumar, Dr. Ramanan Venkat and Prof. Dr.-Ing. Christian Boller since many works on simulation have been performed at the Chair of Non-Destructive Testing and Quality Assurance. As a study case for the preliminary concept DeepSHM, the case study was based on the previous works [Ewald et al. (2015, 2018a)] using aluminum 2024 (Al-2024) plate, a commonly used material in aerospace fuselage structures.

Since the energy of the Lamb waves are mostly confined within the plate, they can be efficiently used to monitor a relatively large area and depending on the wavelength, it can be between less than a meter up to several meters [Wilcox (1998)]. Lamb waves are dispersive in nature and analytically the dispersion phenomenon can be expressed in terms of a wavenumber vs frequency relation [Rose (2014)], from which relationships such as phase velocity vs frequency and group velocity vs frequency, as shown in Fig. 6.3.1-1 (left and right, respectively), can be derived. To save computational resources, the size of the aluminum plate is limited to 600 x 400 x 2 mm. Additionally, a crack with a full-length of $2a$ is assumed to grow from the rivet hole which is located at 400|200 mm, as depicted in Fig. 6.3.1-2.

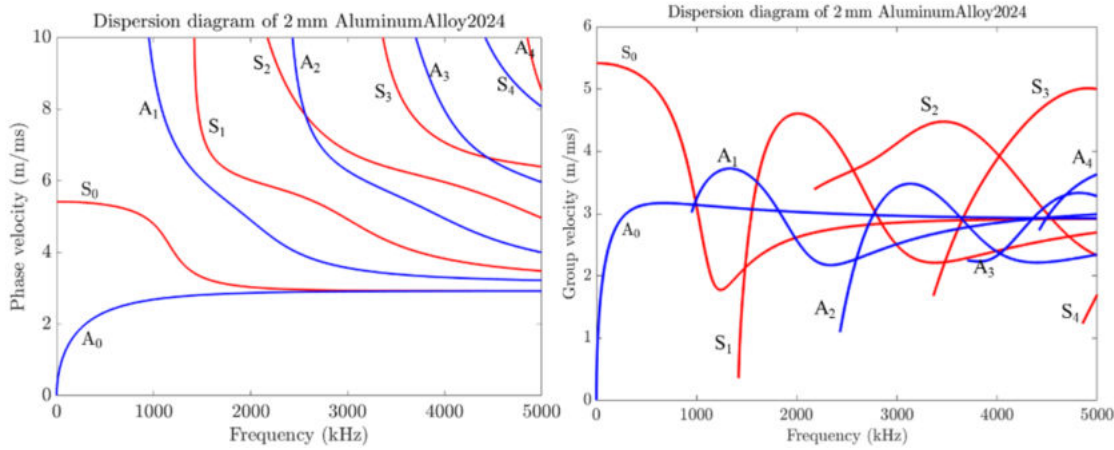


Fig. 6.3.1-1: Dispersion diagram for Aluminum 2024 with a thickness of 2 mm. Red: Symmetric modes Blue: Anti-symmetric modes. The dispersion curves are generated using the German Aerospace Center (DLR) dispersion calculator [Huber (Online)]

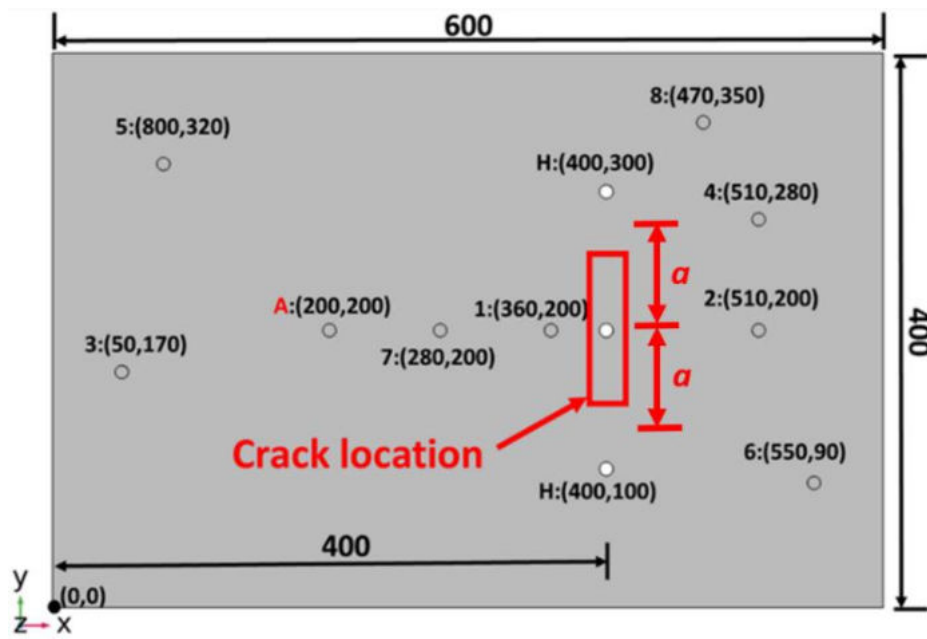


Fig. 6.3.1-2: Sensor positions are numbered from 1 - 8. The coordinates of each sensor position are written in brackets. A is the actuator and H are the rivet holes. All dimensions are expressed in mm. The sensor locations are numbered from 1 - 8.

The half-crack length including the notch is denoted as a , which is measured from the center of the rivet hole. Due to space constraints, only three crack growth scenarios are demonstrated. They vary from 0 mm (pristine plate) up to 200 mm length which is assumed to be the critical crack length a_{crit} . The total crack length $2a$ in percentage as a proportion of 200 mm is presented for multiple classification scenarios in Table 6.3.1-1.

Scenario 1	6 damage classes: Baseline - 20% - 40% - 60% - 80% - 100% of a_{crit}
Scenario 2	9 damage classes: 10% - 20% - 30% of a_{crit} with varying angle (0° , 15° , and 45°)
Scenario 3	15 damage classes: Combination of scenarios 1 and 2

Table 6.3.1-1: Classification scenarios with different percentage of crack length with reference to $a_{crit} = 200$ mm

To determine the hotspot sensing location, the previously developed blob detection algorithm [Ewald et al. (2018b)] was used. The algorithm to obtain the fused image of the wave propagation as depicted can be downloaded from the Github repository. The two sensing locations of the centroids with maximal blob area were used: at 1). 360|200 mm and 2). 510|200 mm. To compare the training performance, we can assign additional random sensing locations (numbered as location 3 – 8), see Fig. 6.3.1-2.

In each simulation, there are two sub-simulations running: 1). The Lamb wave propagation in the mechanical regime, and 2). The piezoelectrical conversion from the mechanical regime into the electrical regime. Sub-simulation 1 runs only once because there was only one plate, however, sub-simulation 2 will take a longer time as the number of sensor position increases. Depending on the number of assigned PZT locations, the total simulation time varies between 6 – 12 hours. Since there are 6 crack length classes (including the baseline) in scenario 1, 6 simulations must be performed, which will take at least 36 hours to complete. Accordingly, the complete scenario took approximately 54 hours to complete on an Intel Core i7 processor and 32 GB RAM.

To reduce the simulation time and to ensure that there was enough space in our hard drive, in the simulation of scenario 1, only the sensors located at positions 1 – 4 are simulated, as the classification scenarios for the crack lengths are more distant from each other (i.e., the simulation request in sensor location 5 – 8 was disabled). In scenario 2, however, classifications become more difficult because sensor signals from a plate with 10% a_{crit} are similar to the one of 20% a_{crit} , so I simulated the response from all 8 sensor positions, making each simulation twice longer due to the FE calculation of additional electromechanical conversion in the position 5 – 8. The combined sensor data from scenarios 1 and 2 are made into scenario 3, although due to the lack of the data of the other 4 sensors (i.e., sensor locations no. 5 – 8), only the 4 sensors data with the same location from scenarios 1 and 2 (i.e., sensor locations no. 1 – 4) could be used.

To generate the Lamb waves using the piezoelectric effect in the numerical model, COMSOL Multiphysics was used. The PZT has a diameter of 9.52 mm and a thickness of 1 mm, respectively. The material properties are Young modulus $E = 73.1$ GPa, Poisson's ratio $\mu = 0.33$ and density $\rho = 2780$ kg/m³ [Bauccio (1993)]. In COMSOL, the predefined PZT properties (orthotropic, Type 5A) are defined as follows (values in GPa): $C_{11} = C_{22} = 120.35$; $C_{33} = 110.88$; $C_{44} = C_{55} = 21.05$; $C_{66} = 22.58$; $C_{12} = C_{13} = C_{23} = C_{21} = C_{31} = C_{32} = 75.1$; the coupling matrix values are as follows (values in C.m⁻²): $e_{15} = e_{24} = 12.3$; $e_{31} = e_{32} = -5.35$; $e_{33} = 15.78$; and the relative permittivity matrix elements are as follows: $\epsilon_{11} = \epsilon_{22} = 919.1$ and $\epsilon_{33} = 826.6$ with density $\rho = 7750$ kg/m³. The excitation pulse is a chirped Gaussian pulse with a central frequency of 310 kHz and bandwidth of 100 to 500 kHz.

To fulfil the Courant-Friedrichs-Lewy condition [Duczek et al. (2014)], a sampling frequency of 10 MS/s, which is 20 times above the Nyquist frequency to ensure

there are enough sampling points. This approach has been already validated by the time of arrival method and a previous experiment [Taltavull (2017)].

The simulation was modeled without considering external environmental influences, and human factors, which results in an identical output signal every time the simulation is repeated. As a data augmentation technique, a random white Gaussian noise was added to the sensor signals to make them closer to the signals obtained from an experiment. This method is demonstrated with a sensor signal obtained from the simulation in Fig. 6.3.1-3, from which we can consider that an SNR of 15 is quite noisy and an SNR of 25 to resemble a typical non-averaged signal measured from an experiment.

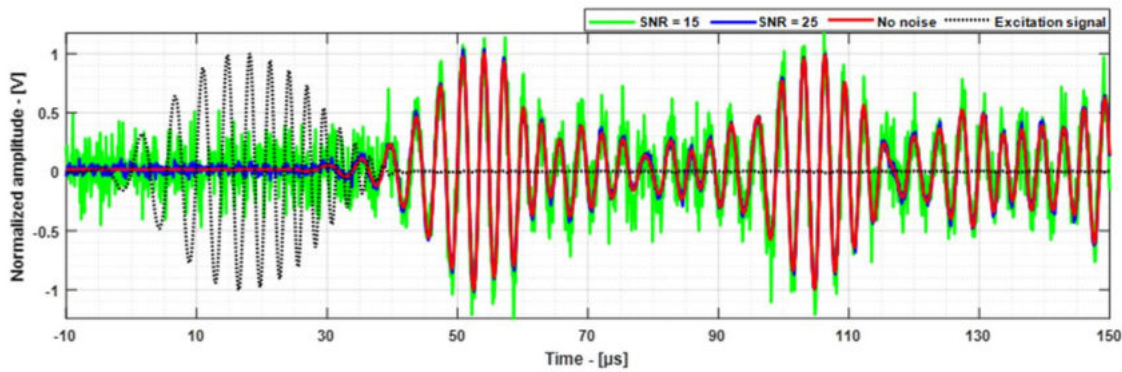


Fig. 6.3.1-3: Example of simulated signal without noise, with SNR = 15 (green) and SNR = 25 (blue)

6.3.2. Data Pre-Processing

This section covers the pre-processing method for the Lamb wave signal generated through the simulation described in section 6.3.1. As mentioned, zero-mean noise was used to simulate external dynamic disturbances with varying SNR between 5 and 25. Note that zero-mean noise is used for a simplification and that the purpose of this work is not focused on the data augmentation technique, i.e., once a better noise representation is available, it can be used in a future study to generate a more realistic simulated signal.

The signals from the sensors can be transformed from a 1D representation (time domain signal) into a 2D Time-Frequency representation (TFR) and signals from each crack condition will result in distinct images that can be fed into a CNN. There are quite a few methods that can be used to perform TFR for Lamb waves. The reassignment method is a technique used in TFRs to sharpen and to localize the frequencies nearer to their true regions along time of the signal [Niethammer et al. (2001)]. Therefore, we used the reassignment method implemented on Short Time Fourier Transform (STFT) in the Github repository of [Fedotenkova (2016)]. An example of such a transformation is shown in Fig. 6.3.2-1.

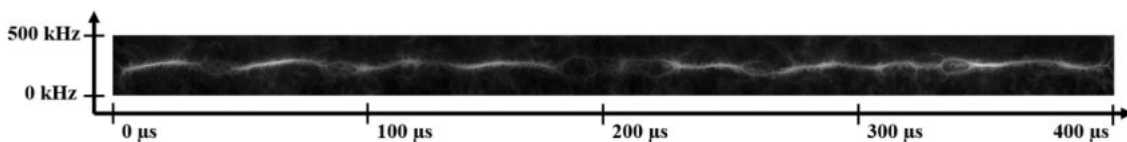


Fig. 6.3.2-1: Example of normalized reassigned STFT. Black pixel signifies an STFT coefficient around 0 and it contains mostly noise and no meaningful information, where the white pixel represents an STFT coefficient close to 1, which contains the waveform information.

6.3.3. Entity Representation

The entity representation is described in section 6.2.2 and there are two assumptions that can be made: 1). *The entity is hierarchically represented over smaller sub-entities*, and 2). *The entity is conserved over time*. This sub-section will describe how to represent the entity in each way.

6.3.3.1. Hierarchical Representation in Multiple Sensor

The analogy of a hierarchical representation of human face recognition is that a face consists of eyes, mouth, noses, ears, chin, etc. The eye consists of an eyebrow, pupil, iris, sclera, etc. In-line with this analogy when a larger entity can be broken down into smaller sub-entities, we can randomly sample the signal over a certain time window W and then pre-process the randomly sampled sequence by the method described in section 6.3.2.

Examples of randomly sampled TFRs from a sensor that is located at 51|20 cm in an undamaged aluminum plate are depicted in Fig. 6.3.3.1-1a - d, where in these figures, W is varied between 20 and 320 μs and thus corresponds to a varying image size between 200 and 3200 pixels in length and between 7 and 253 pixels in width. The width represents the STFT coefficient of the frequency component. Applying a smaller window length would imply the probability that certain features that occur in other classes increase, leading to a higher invariant estimate. For data augmentation, the invariant transformation would be shifting the features to the left or right to represent small sensor displacements.

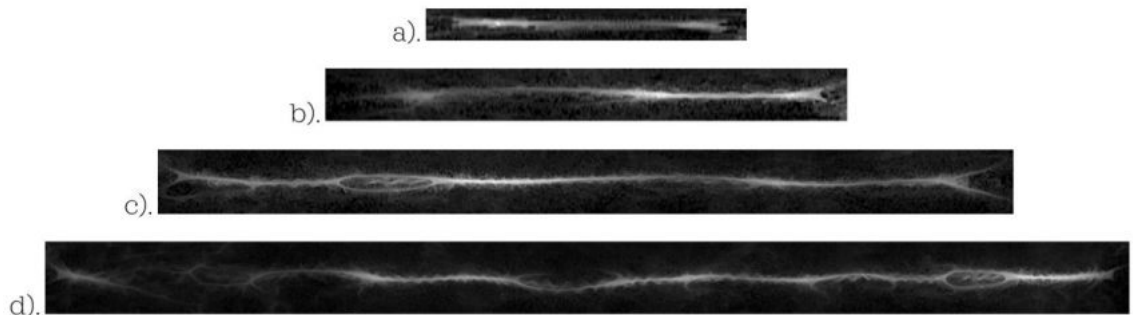


Fig. 6.3.3.1-1: Spectrogram of randomly sampled signals from the sensor that is located at 51|28 cm over a convolution window length W of a). 20 μs , b). 40 μs , c). 80 μs , and d). 160 μs . 1 μs input length corresponds to 10 input samples. Sampling frequency = 10 MHz. The meaning of greyscale color scaling is analogous to Fig. 6.3.2-1.

6.3.3.2. Conserved Entity over Time

An ideal representation of the conserved time-entity signals would be a stack(s) of pre-processed data from all possible observation points, that is all signals are ideally captured by a grid neuron. However, this is very difficult to realize in SHM, particularly because in SHM one would typically only have small number of sampling observations (e.g., less than 10 sensors for a monitoring area of 1 m^2).

For ultrasonic based SHM, it might be possible in the future to combine an actuator PZT and a moving phased array (PA)-probe as a sensor that acts as a grid neuron, where the smallest discretization unit is the size of PZT actuator.

Nevertheless, we can only partially reconstruct the responses by simplifying the conserved representation, e.g., for 3 sensors, the pre-processed data from section 6.3.2 can be visualized by stacking them together in an RGB array, depicted in Fig. 6.3.3.2-1. For 4 sensors or more, the data cannot be visualized without reducing the information content – but it can be stacked in a k -dimensional array.

Applying a larger window length means that the probability that certain features occur in other classes decreases (i.e., equivalent to a lower invariant estimate). For data augmentation, the invariant transformation would be left or right shifting the features to represent a small sensor displacement but also swapping the channels (e.g., the red, green, and blue channels first represent sensors 1,2,3 respectively and then are swapped to represent sensors 2,3,1 respectively). The reason for this channel equivalence is because the feature representations of each channel will be summed together before being passed through to the next layer. To simplify: what would be the perception difference in hearing music when the left and right speaker of a stereo headset are turned around?

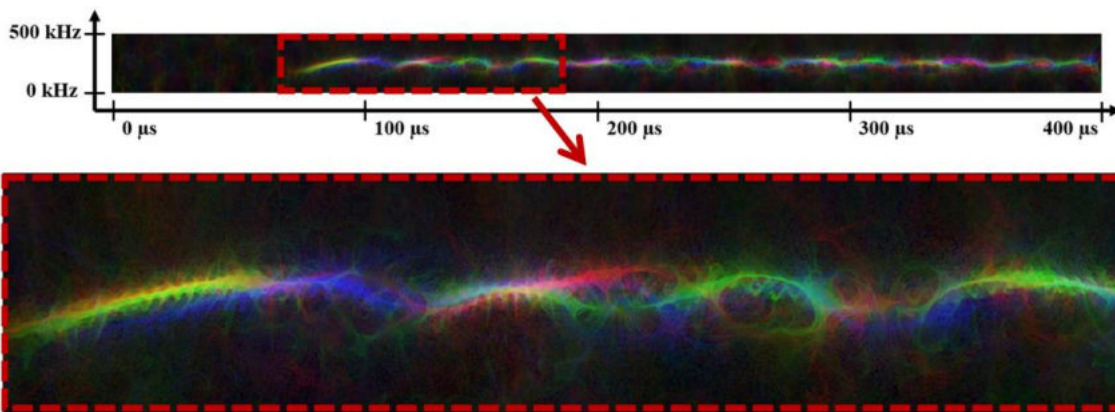


Fig. 6.3.3.2-1: Feature representation of merged input signal from 3 sensors channeled in RGB array.

6.3.4. Training Setup and Parameters

6.3.4.1. Hardware

The analysis was performed on a standard TU Delft PC, which is a Dell Precision T5810 running with Intel Xeon(R) E5-1620 3.5 GHz and 32 GB DDR-RAM running on Windows 10 which is equipped with an additional NVidia GPU GeForce GTX1080Ti to increase the computational performance. At the time of purchase (June 2018), the graphic card had the highest performance on the end-user market. At that time, this was sufficient large-scale GPU such as the NVidia DGX-2 or the cloud services such as Azure or Amazon AWS. Another alternative for training a deep neural network would be a Field Programmable Gate Array (FPGA) as it promises even faster processing, but it would require hardware customization which is out of scope of this project.

6.3.4.2. Software and Libraries

At the time the experiment made (mid-2018), the software selection was decided for the de-facto market leader TensorFlow (developed by Google) and the Keras Wrapper API because they were the richest libraries available on Python market. An alternative library such as PyTorch is also available, albeit it was less rich than

TensorFlow, but recently gaining its popularity among the deep learning academics due to its simpler syntax. Some “academic friendly” scripting languages such as MATLAB and Julia have also launched their own GUI-based deep learning toolbox, making it is even more practical to use.

6.3.4.3. Optimization

Training a neural network is NP-hard and a hyperparameter grid search would take non-polynomial effort, so we shall prioritize converging iterative methods over search metaheuristics given the high dimensionality of parameters θ . When talking about converging iterative methods, the natural choice for a high-dimensionality of θ would be the 1st order methods, also known as the gradient-based method which is a single-dimensional Jacobian. Involving any second-order methods (i.e., involving Hessian matrix) would likely be able to solve the problem faster than gradient-methods, however as there is no such thing as a free lunch, the Hessian method would also require much larger amount of computational space (i.e., RAM/memory).

Among the 1st order methods, some popular techniques are (L)-BFGS [Morales (2002)], Gauß-Newton [Shalev-Shwartz and Ben-David (2014)], the (steepest) gradient descent [Ruder (2016)], and Levenberg-Marquardt [Zayani et al. (2008)]. Like other libraries, currently only the gradient descent methods are implemented in Keras. The native optimizer is called stochastic gradient descent (SGD) with variable batch training size which supports momentum and Nesterov acceleration [Botev et al. (2017)] as given in Eq. (6.3.4.3-1) to escape local minima.

$$\begin{aligned}\delta_{t+1} &= \gamma \cdot \delta_t + \eta \cdot \left[\nabla_{\theta} J(\theta - \gamma \cdot \delta_t) \right] \\ \theta_{t+1} &= \theta_t - \delta_{t+1}\end{aligned}\tag{6.3.4.3-1}$$

In Eq. (6.3.4.3-1), δ_{t+1} is the update vector for parameter θ at iteration $t+1$, η is the learning rate, J is the assigned cost function and γ is the momentum which is typically set to 0.9 [Srivastava et al. (2014)]. There are more optimizers available in Keras such as RMSProp, adaptive moment estimate (Adam) and its variants (Adagrad, Adadelta, Adamax, Nesterov-Adam). More detailed explanations of these techniques can be found in Keras documentation as well as [Géron (2017)].

6.3.4.4. Neural Network Architectures and Optimizers

The convolutional filter of CNN is designed to capture spectrotemporal features of the TFR and after that, the learned filters are fed into a fully connected layer (MLP). Without capturing the spectrotemporal features, the information coherence within a certain time-frequency window will be lost. To demonstrate that spectrotemporal features are learned during the training, we must compare the training results from CNN+MLP and pure MLP architectures without convolutional kernel for each of the 3 different classification scenarios.

Analogous to neural network parameters optimization, finding a suitable network architecture has an NP-hard property since the architecture

combinations are infinitely extendable and unfortunately, there is no strict rule to design the number of layers. Nevertheless, given the theoretical detail regarding function capacity in chapter 3, the network should be designed as compact as possible. Thus, it would be logical to start the choice of architecture with the less complex series. Therefore, the previous architectures from previous work [Ewald (2019)] are re-used, as described in Table 6.3.4.5-1.

MLP	D(128)-DO(0.5)-D(16)-CL
CNN	C(8)-MP-DO(0.5)-C(16)-MP-DO(0.5)-C(32)-MP-DO(0.5)-D(16)-DO(0.5)-CL

Table 6.3.4.5-1. Neural network architectures used. $C(i)$: i -filter convolutional kernel; MP: MaxPooling layer; $DO(j)$: dropout regularization with rate of j ; $D(k)$: dense (fully connected) layer with k -neurons, CL: Classification layer

The utilization of more sophisticated CNN such as VGG-16 [Simonyan and Zisserman (2014)], inception layers [Szegedy et al. (2015)], or ResNet [He et al. (2016)] are out of scope of this work.

The sample code can be cloned from the repository [Github(online)]. For training purposes, the data should be normalized between 0 and 1 to avoid an exploding gradient. For a deep neural network, it is recommended to always activate the dropout regularization and according to [Srivastava (2014)], 0.5 is the best rate found. The default parameter for the optimizers is presented in Table 6.3.4.5-2.

SGD	$\eta = 0.01, \gamma = 0.0, \eta_{\text{decay}} = 0.0, \text{NESTEROV} = \text{FALSE}$
Adam	$\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \eta_{\text{decay}} = 0.0, \text{AMSGrad} = \text{FALSE}$

Table 6.3.4.5-2. Default optimizers parameters in Tensorflow library. η : Learning rate, γ : neural momentum, η_{decay} : learning rate decay, NESTEROV: Nesterov momentum parameter, β_1, β_2 : exponential decay rate for 1st and 2nd moment estimates.

6.4. Training Result

The training results for the given parameters in section 6.3.3 are presented in this section and are organized as follows:

- In section 6.4.1, the results and discussion for modelling DeepSHM as a hierarchical entity, as discussed in section 6.3.3.1, are presented, where the discussion includes the training results trained under both the SGD and Adam optimizers for different sensing locations in each given damage case scenario for variable window length.
- In section 6.4.2, the results for modelling DeepSHM as a conserved entity, as discussed in section 6.3.3.2, that gets the input from fused sensor data, are presented. In this case, the training results for different sensing locations will be given, however as a consequence of the assumption of the conserved entity over time, only the full window length can be used.

For sake of brevity, the training samples are limited to only 100 samples per class to see the network behavior. During the trial-and-error phase, I actually tried with 1000 samples first, but it took too long for preprocessing (about 2 days per

dataset). In our case, the input size varied between 251×7 pixels up to 4101×247 pixels, depending on the dataset generated.

6.4.1. Results of Hierarchical Representation

6.4.1.1. Influence of Network Architecture and Sensor Locations

To demonstrate the superiority of CNN in comparison to multilayer perceptron (MLP) to handle spectrotemporal data, first the training result of MLP handling the training set is presented. The MLP is trained for all the damage scenarios given in Table 6.3.1-1 with various reserved memory from 800 to 3200 input samples. For brevity, only the training results for the training and validation datasets from the presumably best sensor location are shown, which is the sensor at location 2 (cf. Fig. 6.3.1-2). For simplification, the training process will be limited up to 50 epochs only. The training results are depicted in Fig. 6.4.1.1-a - f.

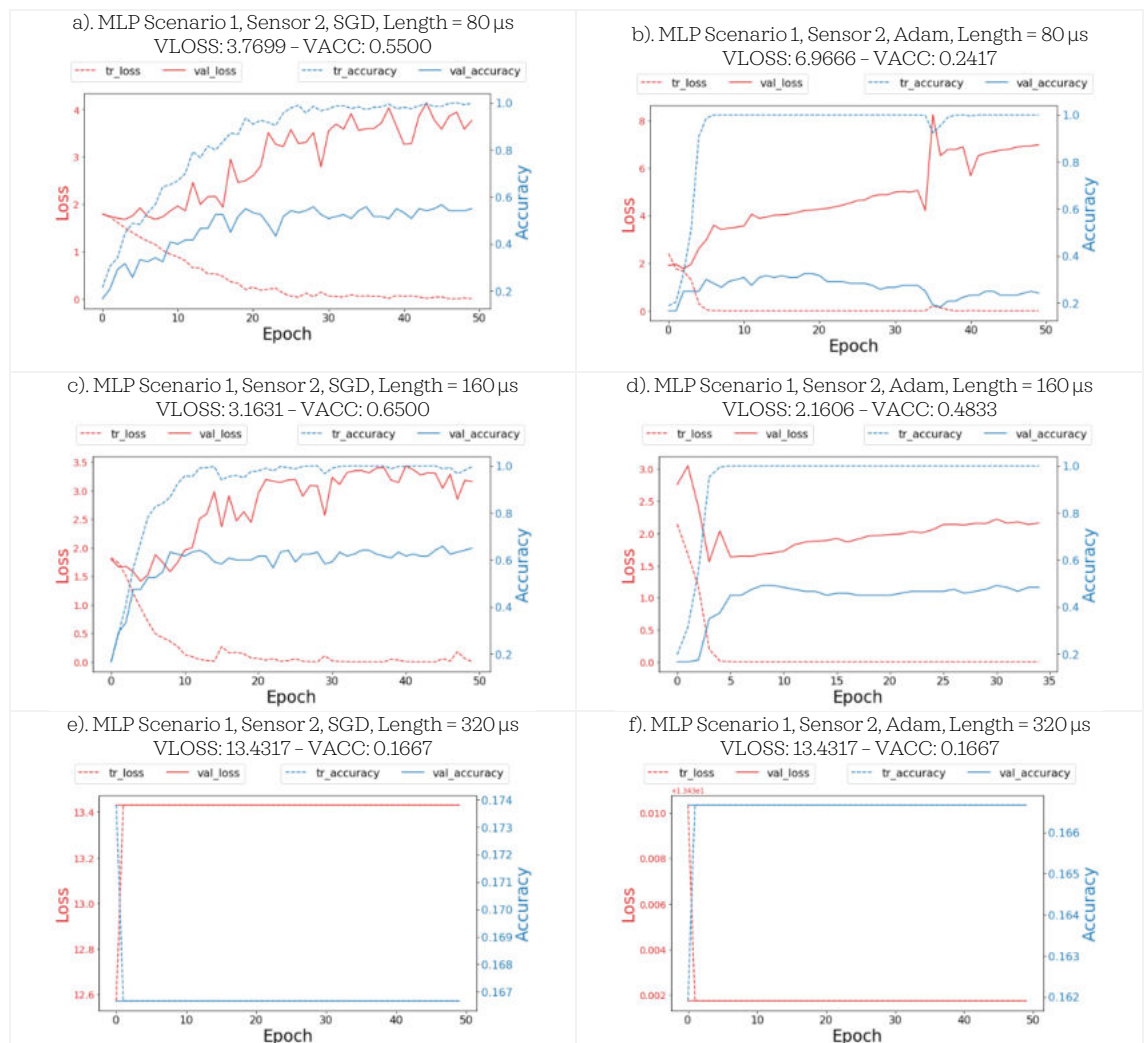


Fig. 6.4.1.1-a - f: Training results of MLP for signal from sensor 2 with various window length trained under Adam and SGD optimizer up to 50 epochs. $1 \mu\text{s}$ input length corresponds to 10 input samples. VACC = Final validation accuracy, VLOSS = Final Validation Loss. Note that the scaling is not uniform due to the default library settings.

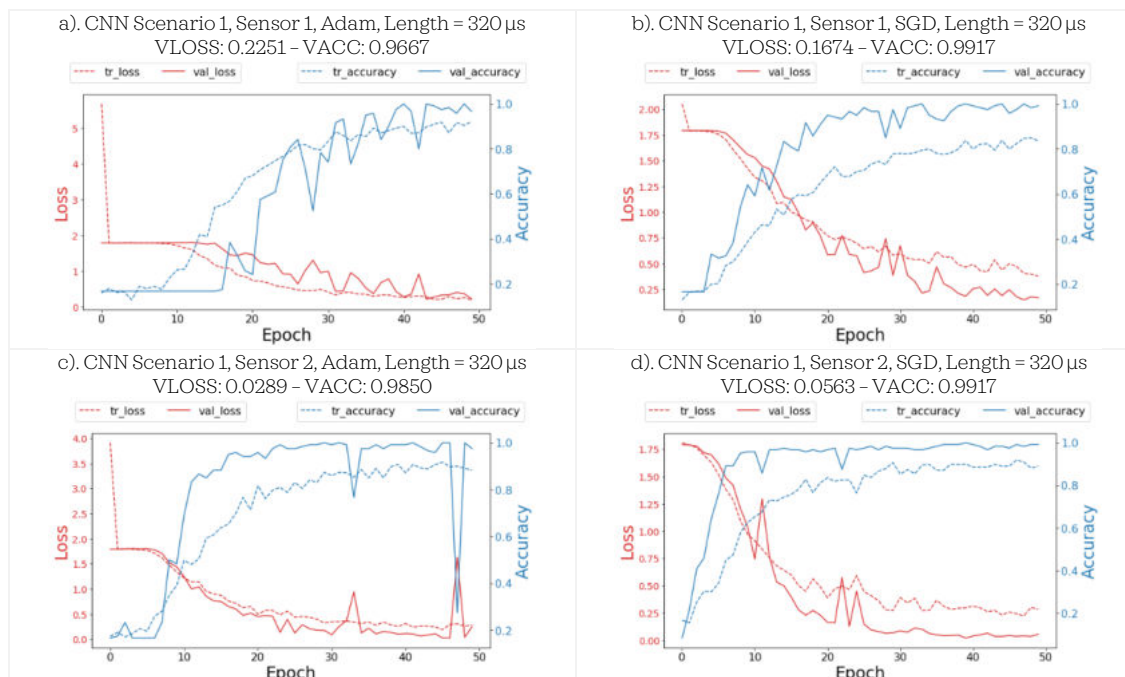
SGD and Adam optimizer shall be compared in this case. Adam uses the adaptive learning rate from the moments of the gradients estimates thus it is generally faster in finding the gradient path in the beginning epochs and keeps an exponentially decay of prior gradients average. On the other side, SGD is slightly

slower in finding an optimal gradient in the beginning epochs as it only computes the gradient without the estimate approximation.

For a general training purpose, Adam is therefore recommended as it can find the gradient path faster. From Fig. 6.4.1.1-1a – b, for a window length of $80 \mu\text{s}$, it can be seen that the MLP model quickly overfits regardless of the optimizer used. While the training accuracy reaches almost 1.0 (or 100%) in Fig. 6.4.1.1-1a – b, it can be seen that the validation accuracy stays below 0.6 (or 60%). This behavior is confirmed again in Fig. 6.4.1.1-1c – d, where the window length was extended to $160 \mu\text{s}$ and in both cases, the difference between training and validation accuracy is more than 35%.

When the window length was extended once more to $320 \mu\text{s}$ (see Fig. 6.4.1.1-1e – f), only a constant flat line over the whole training epoch was reached, meaning that the GPU was overloaded. This problem can be solved by adding more physical memory by using a better GPU until it meets a larger input dimension where we start another loop of using a more powerful GPU. With such an unacceptable training behavior, it is decided not to further proceed with exploiting the MLP since a simple MLP would highly be likely to fail to capture a spectrotemporal feature due to its lower function capacity.

For the CNN architecture, the CNN was kept with 3 hidden convolutional layers attached to the same MLP. Then, each convolutional layer was followed by pooling and dropout layers as described in Table 6.3.4.5-1. In general, the network may learn the pattern representation after long training, however in reality, we always have a time-cost limit, and a training budget threshold must be determined depending on application and budget resources. To demonstrate a simplified training budget, assume limit the training epochs to 50. The training results from CNN trained both under Adam and SGD optimizer for damage scenario 1 (Table 6.3.1-1) are depicted in Fig. 6.4.1.1-2a – h, where in this case the CNN is trained to distinguish 6 different damage classes including one from the baseline.



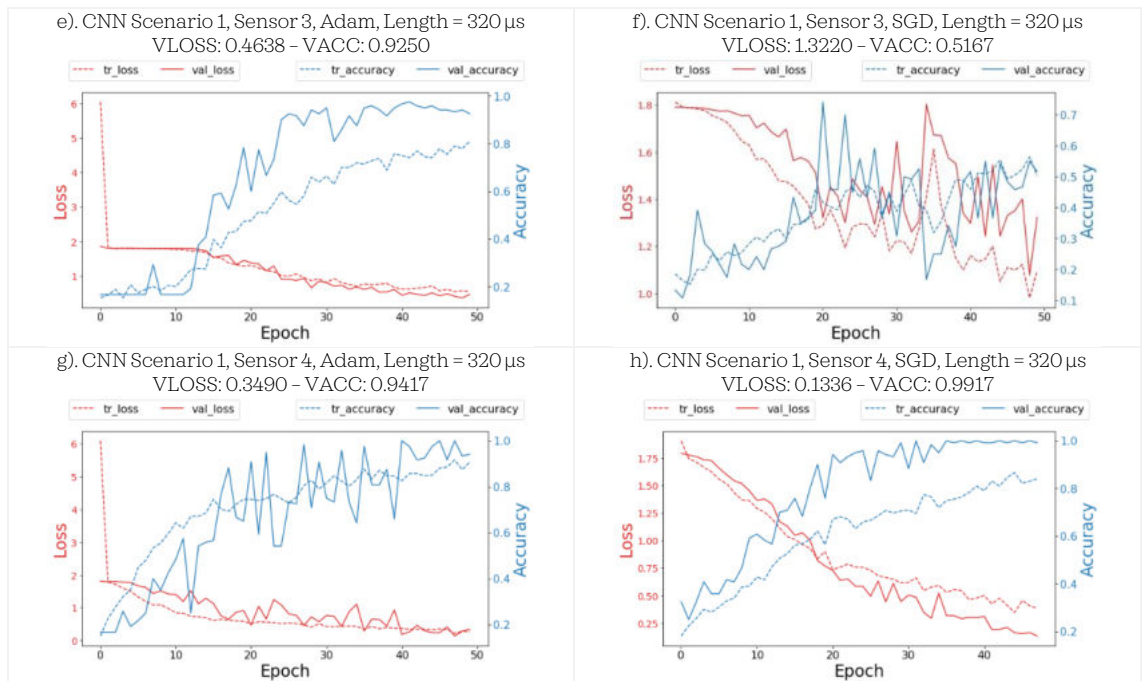


Fig. 6.4.1.1-2a - h: Training results of CNN for signal from sensor 1 - 4 in scenario 1 with window length of $320 \mu\text{s}$ trained with Adam and SGD up to 50 epochs. Sensor locations are given in Fig. 12. $1 \mu\text{s}$ input length corresponds to 10 input samples. VACC = Final validation accuracy, VLOSS = Final Validation Loss. Note that the scaling is not uniform due to the default library settings.

The highest validation accuracy reached was 0.9917 (or 99.17%) from data captured by sensors 1, 2, and 4 when optimized by SGD, while the lowest validation accuracy reached was 51.67% from data captured by sensor 3. However, when using Adam, the highest validation accuracy reached was 98.50% from data captured by sensor 2 (which is presumably the best sensor position) followed by 96.67%, 94.17%, and 92.50% for sensors 1, 4, and 3, respectively. Because the Adam optimizer yields a better result than the SGD, from this point on, only the result of the trained network under Adam optimizer will be shown.

For the classification of damage scenario 2, the PZT response was also simulated from sensor locations 5 - 8, because in this scenario, the CNN is trained to classify less distinguishable signals from each class as per Table 6.3.1-1. Normally, more sensor responses would give an advantage because there is more data available. Nevertheless, for brevity only the training results using Adam optimizer from sensor location 1 - 4 in Fig. 6.4.1.1-3a - d are shown, which can be directly compared to Fig. 6.4.1.1-2a, c, e, and g, respectively.

The results depicted in Fig. 6.4.1.1-3a - d met the expectation because validation accuracy is on average lower than in scenario 1, ranging from 32.22% in sensor 3 (where it also overfits - see Fig. 6.4.1.1-3c and sensor 3 is one of the worst PZT sensing locations) to 93.33% in sensor 1, depicted in Fig. 6.4.1.1-3a which is one of the best PZT sensing locations. This is to be expected, because if we look closely into scenario 2, it can be easily assumed that the TFR signal from the baseline plate and the damaged plate with a smaller crack length (e.g., only 10% a_{crit}) and smaller deviated angle (i.e., 15°) are likely to be similar to each other, especially because the recording length was short ($320 \mu\text{s}$). Thus, the neural network would not be able to distinguish these TFR.

Nevertheless, unlike classical signal processing, the TFRs from better sensing locations such as sensor positions 1, 2, and 4 were able to be trained to reach a better validation accuracy (Fig. 6.4.1.1-3a, b, and d). As a side note, by doubling or even quadrupling the window length from $320\ \mu\text{s}$ to $640\ \mu\text{s}$ or $1280\ \mu\text{s}$, I believe the neural network will be very likely to be able to learn the distinguishing pattern between the TFR and thus further increase the validation accuracy. This is because in the later time-series response, the wave scatter from the crack would eventually travel through sensor location 3, too.

While this might be an advantage of deep learning over classical signal processing, recall that doubling or quadrupling the window length would require double or quadruple the amount of data storage. If one TFR image takes up 2 MB, doubling this would mean at least 4 MB. For small-scale research, this is surely not a problem but scaling this up on industrial level would mean double or quadruple the required investment for data storage. So, to be fair we shall not forget that in classical signal processing, multiplying the data storage might not be necessary.

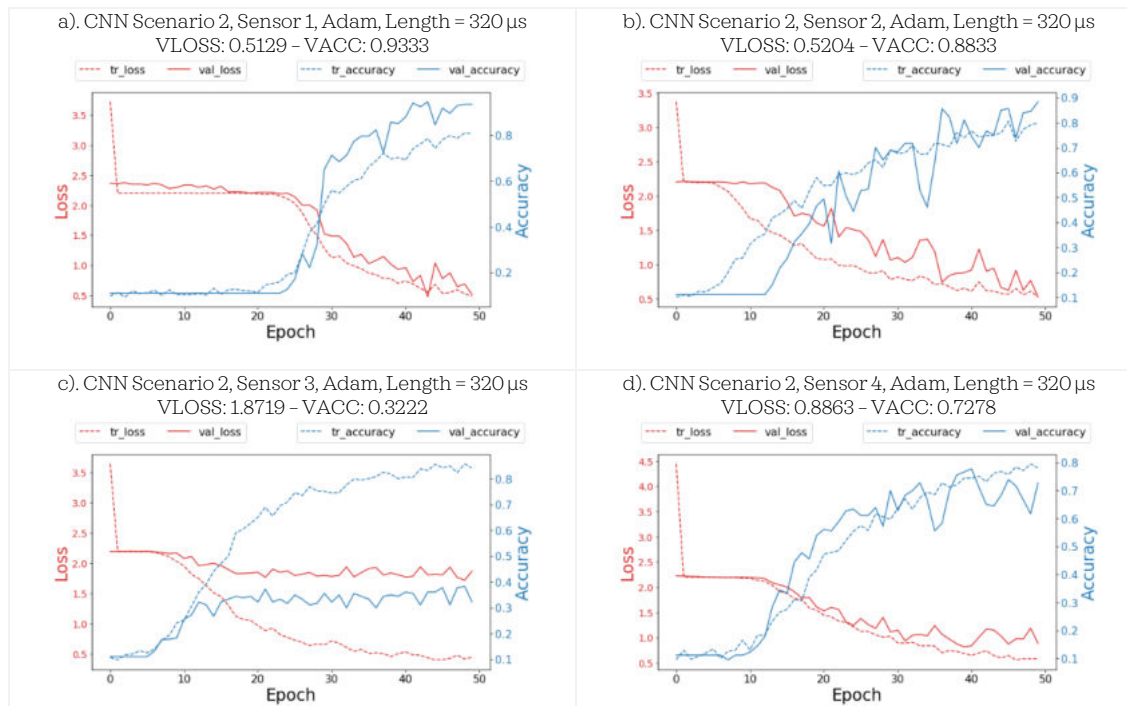


Fig. 6.4.1.1-3a - d: Training results of CNN for signal from sensor 1 - 4 in scenario 2 with window length of $320\ \mu\text{s}$ trained with Adam optimizer up to 50 epochs. Sensor locations are given in Fig. 12. $1\ \mu\text{s}$ input length corresponds to 10 input samples. VACC = Final validation accuracy, VLOSS = Final Validation Loss. Note that the scaling is not uniform due to the default library settings.

As previously mentioned in Table 6.3.1-1, damage scenario 3 is just the combination of scenario 1 (a very distinctive classification ranging from 0% - 100% length of critical crack in each 20% step) and scenario 2 (a less distinguishable signal between each class with varying angled crack), the training results are slightly better than scenario 2, but worse than scenario 1. Note that in this case, only the data from the 4 sensors in scenario 2 which are located in the same location as in scenario 1 can be added to the training set. Also, the importance of sensor positioning is also now clearly highlighted. The training results are depicted in Fig. 6.4.1.1-4a - d. As mentioned before, sensors 1 and 2 are at one of the best sensing locations and thus, the validation training accuracy

reached was more than 0.9 or 90% (Fig. 6.4.1.1-4a – b), while sensor 3, which is located very far away from the crack location, only reached a validation accuracy of 43.67% (Fig. 6.4.1.1-4c). Again, this is far from surprising, because we can expect that at the sensing location occupied by the sensor 3, the accumulated energy from the scattered wave from the crack to be far less than that captured by sensors 1 and 2. The data from sensor 4, which is located close to the crack location but not directly placed along the wave scatter propagation path, reaches a final validation accuracy of 69.67%, as depicted in Fig. 6.4.1.1-4d.

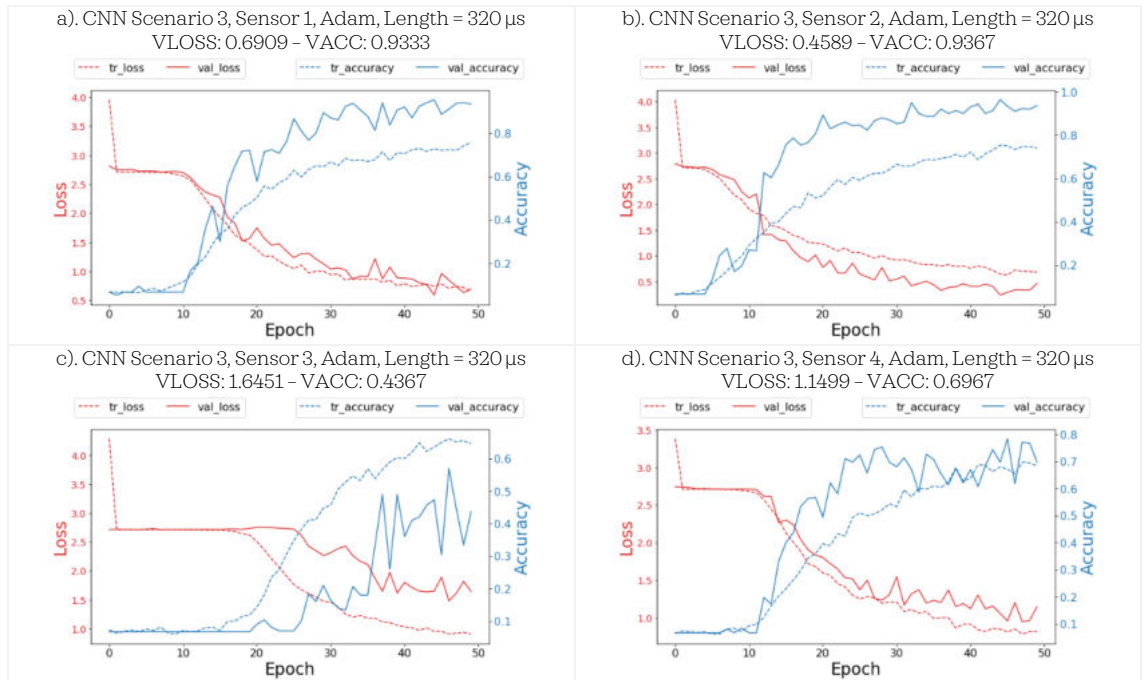


Fig. 6.4.1.1-4a – d: Training results of CNN for signal from sensor 1 – 4 in scenario 3 with window length of 320 μs trained under Adam optimizer up to 50 epochs. 1 μs input length corresponds to 10 input samples. VACC = Final validation accuracy, VLOSS = Final Validation Loss. Note that the scaling is not uniform due to the default library settings.

From all these experiments, we can safely conclude that one cannot rely solely on a deep or any other advanced machine learning algorithm to tackle a physical limitation and thus there must be a physical intervention (such as adding more PZT sensors until detectability convergence increases) to improve the result. The good news is, however, that the previous work regarding the sensor placement strategy for hotspot sensor placement using blob detection [Ewald (2018b)] can be used because a sensor placement strategy is still needed even when applying a sophisticated machine learning techniques such as deep learning. The bad news on the other side is that a hotspot sensor placement strategy is still needed – thus a design effort for an SHM sensor placement strategy is needed and there would be a cost for this both in time and money.

6.4.1.2. Effect of the Convolution Window Length

This scenario describes the influence of the reserved convolutional window length of the signal on the required time per training step and on the final validation accuracy (VACC). For brevity, only the data from the best sensing location is used, which is sensor 2. The convolutional window length varies between 10 μs to 320 μs sampled from the full signal as previously explained in section 6.3.3. All the scenarios are trained with Adam optimizer and to save time,

the training is limited to only 50 epochs. The results for all scenarios are summarized in Fig. 6.4.1.2-1a and Table 6.4.1.2-1, where it can be seen that at least a window length of 200 μs is necessary to surpass a 90% training accuracy threshold, while for the more difficult scenarios 2 and 3, a windows length of 320 μs is needed. The computation time per time step varies between around 16 ms for TFR with up to 60 μs window length and increases quadratically as the window length is increased as depicted in Fig. 6.4.1.2-1b.

Window Length [μs]	Scenario 1: 6 classes		Scenario 2: 9 classes		Scenario 3: 15 classes	
	Time/Step [ms]	Final VACC	Time/Step [ms]	Final VACC	Time/Step [ms]	Final VACC
10	16	0.1667	14	0.1111	14	0.0633
15	16	0.1833	14	0.1111	14	0.0667
20	16	0.1500	14	0.1278	15	0.0800
25	16	0.1750	14	0.1167	15	0.0833
30	16	0.1917	15	0.1444	15	0.0733
40	16	0.2167	15	0.1556	17	0.0933
50	16	0.3000	16	0.1389	18	0.0733
60	25	0.3917	17	0.1611	20	0.0867
80	28	0.4917	18	0.2111	22	0.2300
100	24	0.5250	24	0.1811	28	0.2267
120	29	0.6000	22	0.1611	33	0.3500
160	46	0.7593	47	0.4889	54	0.3833
200	67	0.9167	68	0.5086	70	0.5600
320	185	0.9833	212	0.9044	190	0.9467

Table 6.4.1.2-1: Effect of Window Length on Training Time per Epoch and Final VACC* at 50 Epochs. *VACC = Validation Accuracy

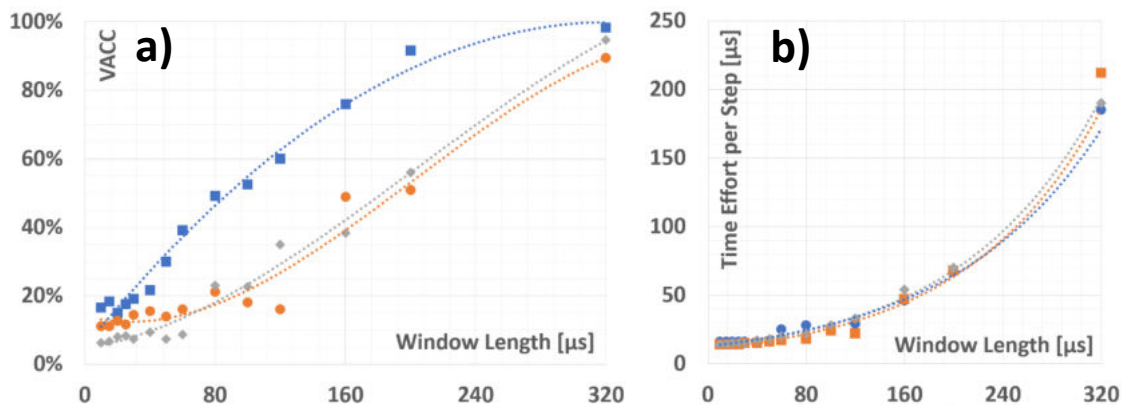


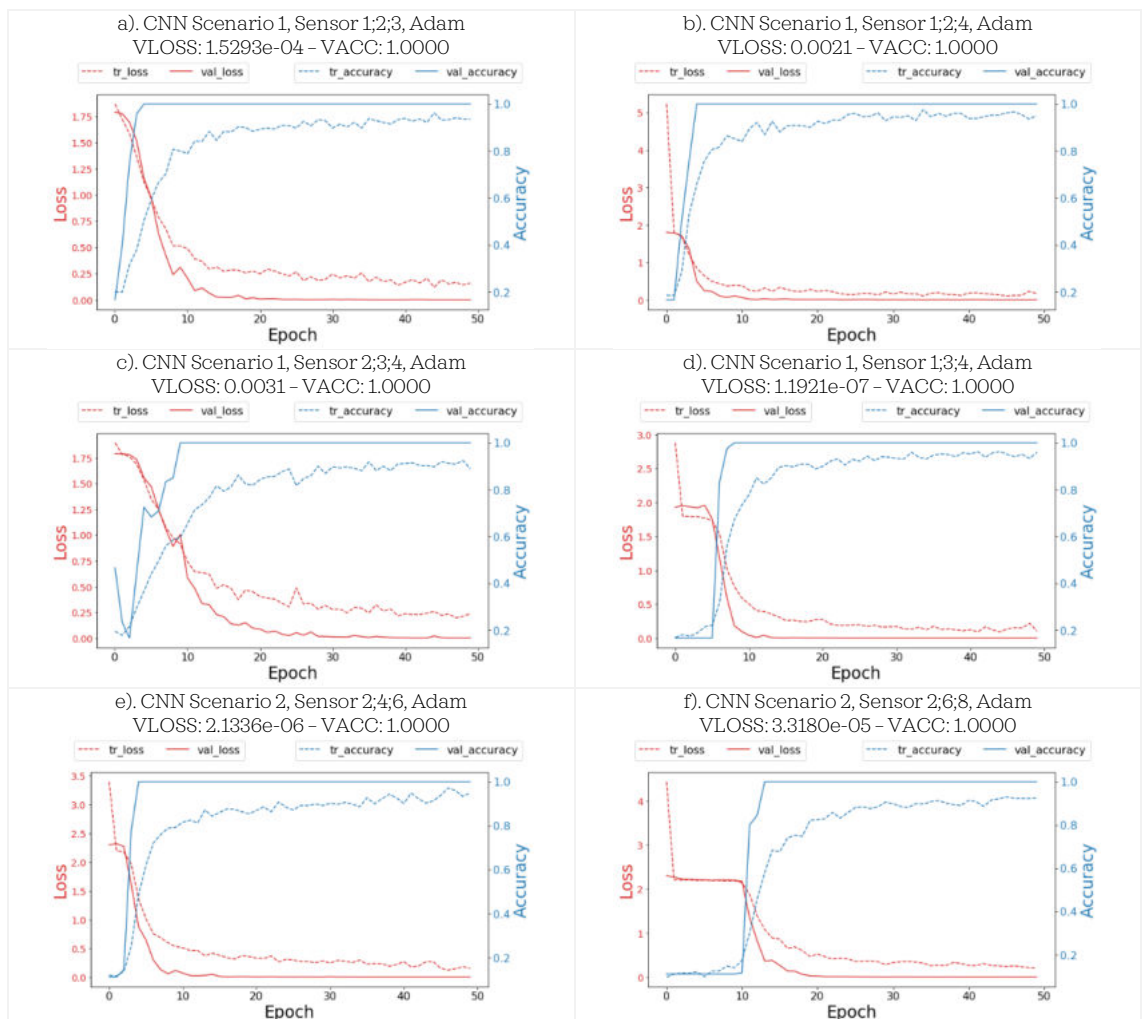
Fig. 6.4.1.2-1: a). The relation between final validation accuracy (VACC) and reserved window length in [μs]; b). The relation between time effort needed per training step and reserved window length in [μs].

6.4.2. Results of Conserved Entity over Time

Now it is interesting to see the training behavior when the perception is modelled as a single entity conserved over time. The consequence for this assumption is that the training set consists of a k -dimensional data cube composed of multi-layer full-length TFRs, e.g., see Fig. 6.3.3.2-1. Theoretically, the data cube should comprise all possible layers which represents all possible responses from each sensor. However, for brevity, the data cube is represented in 3-layers (meaning that only 3 sensor responses are used) so that it can easily be converted into an

image and trained with the libraries (Keras and Tensorflow). While Tensorflow can read bitmap (BMP) files, it is not recommended to convert STFT coefficient matrix into BMP since it took too much space. Thus, a portable network graphic (PNG) format, which is a lossless compression format, was chosen to balance between the information richness contained in BMP and the compression sparsity of JPEG format. Most of the ready-to-use deep learning libraries only support a 3-channel color image because many works in which the deep learning community are focused on using RGB images. By customizing the library, it is possible in the future to train an arbitrarily sized dataframe in Python, such as demonstrated by [Paoletti et al. (2018)] for classification of hyperspectral images by using CNN. Unfortunately, those works are in general very niche and often not made publicly accessible.

For this section, several sensor responses are combined to create the representation. Since there are only 4 sensors in scenario 1, all possible combinations of the training behavior of the sensor response (1;2;3, 1;3;4, 1;2;4, and 2;3;4) can be shown, as depicted in Fig. 6.4.2-1a - d. However, as there are too many possible combinations in scenario 2, only the result of several combinations are shown in Fig. 6.4.2-1e - h. The rest of the results are available in the dataset, or alternatively the readers can also download the code from the repository and try to run it. Finally, since scenario 3 is simply a combination of scenario 1 and 2, only 4 sensor responses can be shown. The corresponding training results from scenario 3 are given in Fig. 6.4.2-1i - l.



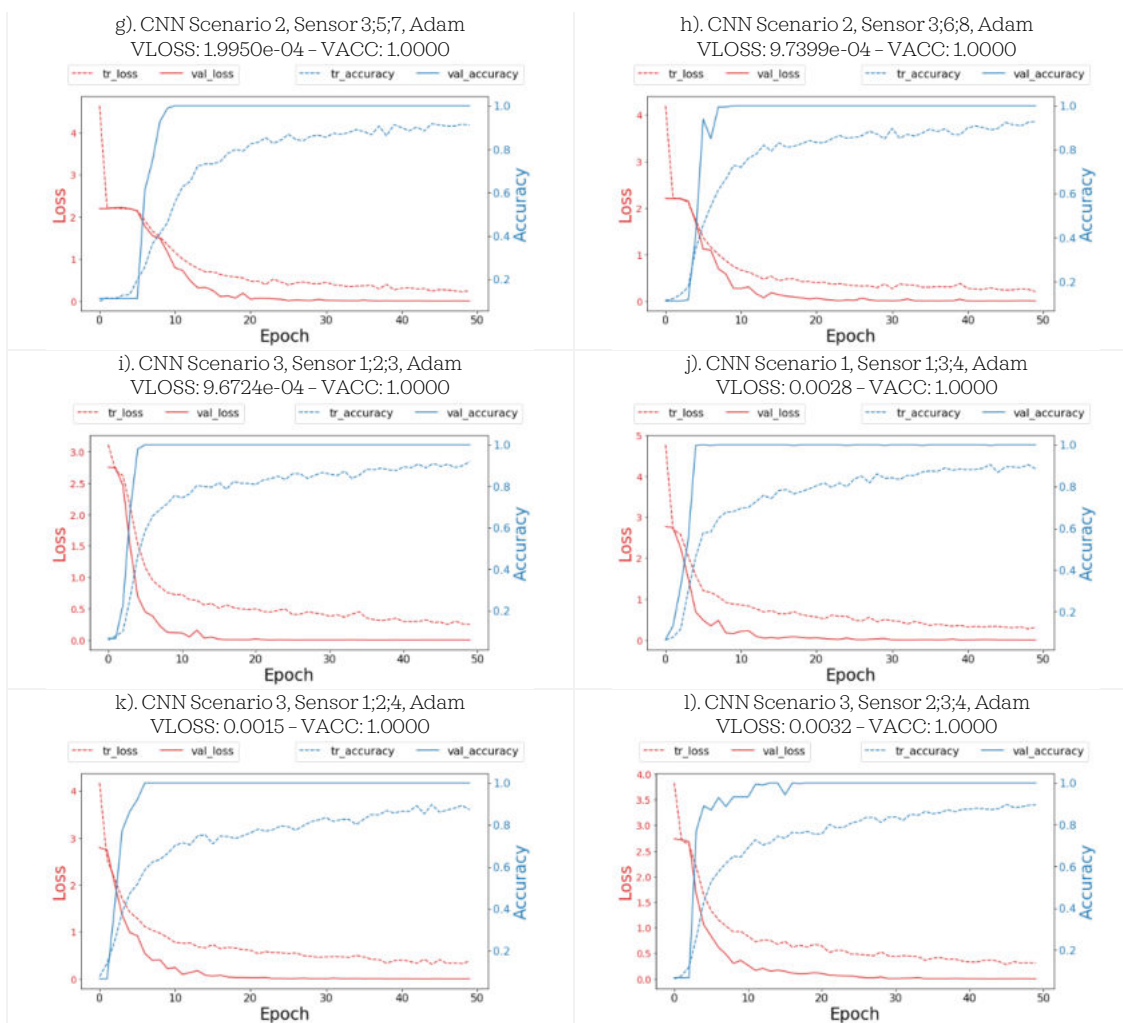


Fig. 6.4.2-1a - 1: Training results of CNN for combined signal from various sensor combination in all damage scenarios trained under Adam optimizer up to 50 epochs. VACC = Final validation accuracy, VLOSS = Final Validation Loss. Note that the scaling is not uniform due to the default library settings.

From all sub-figures in Fig. 6.4.2-1, it can be clearly concluded that modelling the SHM perception as a single conserved entity has a quickly converging training rather than modelling the perception as hierarchically ordered but separate entities. For the first case, it is obvious that the sensing locations now becomes the less important issue. There are clearly several different behaviors before the training accuracy reaches its 1.0 plateau as depicted in Fig. 6.4.2-1a - 1.

My assumption regarding on why CNN was able to capture the correlated 3D-spectrotemporal features within certain regions in spectrograms is the existence of the correlated 3D-spectrotemporal features which turns out to be the evidence of invariant latent [Feige (2019)] in which the estimate is proposed in Lemma 3.5-3. It is difficult to deductively prove the applicability of this lemma because neural network parameters can only propose correlations without explanatory power, but for the time being this is *de facto* accepted in the computer science community. A plausible explanation to my hypothesis has already been stated previously in section 6.2.2 regarding music recognition in the human brain.

While a Lamb wave signal is not a music or a song, it is not merely random noise either - there is an interconnection between each particle wave movement during guided wave propagation because an acoustic wave in a continuum can

be regarded as harmonic motion and thus do not fall into a sudden singularity, so I assume the existence of locally connected spectrotemporal features between each layer in Fig. 6.3.3.2-1. We shall be aware that this assumption is of course not valid for randomly occurring singularities within a continuum – but normally we never expect sudden singularities within acoustic wave signals either (except when the sensor is broken, but that is a separate topic).

It is difficult to understand what is happening inside the deep neural network, but I postulate that these local spectrotemporal features, while being very deeply embedded in the CNN model, are very distinct to each other for every different damage class. This was due to previous assumption that every different damage class would produce a varying, non-homogenous TFR at each different sensing location and therefore allow the neural network to easily capture these (cf. Rademacher complexity) and learn from them after several training epochs. Each TFR is then a special distinctive signature of each probability classes as supported in Lemma 3.5-2.

In general, it is common to use a single-frequency centered Hann window as an excitation wave because it would be easier to analyze the sensor response based on single-frequency signal. As this has always been most of the case, the training results gives evidence that it might be an advantage to try out the less exploited and more “adventurous” chirplet-like excitation signal for Lamb wave SHM. The training results suggest that these distinctive spectrotemporal features within the representation that are more heavily augmented due to the quasi-chirplet excitation in comparison to when it is excited by simple Hann window signal, which in general produces more complex material response and thus more complex information embedded in the signal representations that helps the CNN to learn from this distinctive spectrotemporal information. Currently this is my speculation, so I think that there would be a future opportunity to quantitatively investigate this matter in more rigorous research.

6.5. Concept Validation

6.5.1. Result comparison with Random Noise Training

To deliver further evidence regarding the invariant latent, a mock-up test is set up to compare the training behavior between 1). the simulated signal with added white Gaussian noise with varying SNR between 5 – 15 and 2). pure Gaussian white noise. The reason to do random noise training comes from statistical A/B testing. Without a trial like this, we will never know whether the network actually learns the pattern, or it merely remembers the noise behind the pattern. An example of TFR of Gaussian white noise is depicted in Fig. 6.5.1-1. The noise was trained under exactly the same training parameters as before and split into 6 different folders like in scenario 1.

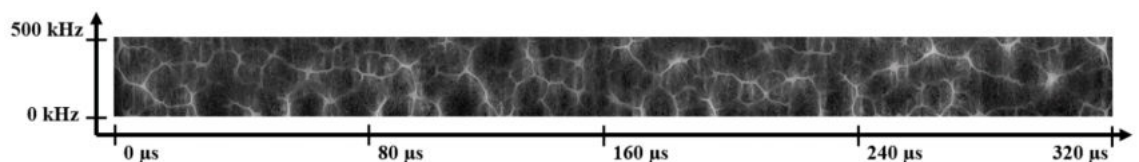


Fig. 6.5.1-1: TFR of a white Gaussian noise with a length of 320 μ s

The white noise training were repeated several times and give two sample results in Fig. 6.5.1-2a - b which depict the training behavior for a white noise signal with a window length of $320 \mu\text{s}$. It can be seen from Fig. 6.5.1-2a - b that the networks failed to reach a validation accuracy of more than 60% even after 50 training epochs, while the training accuracy reached a value much higher than the validation accuracy.

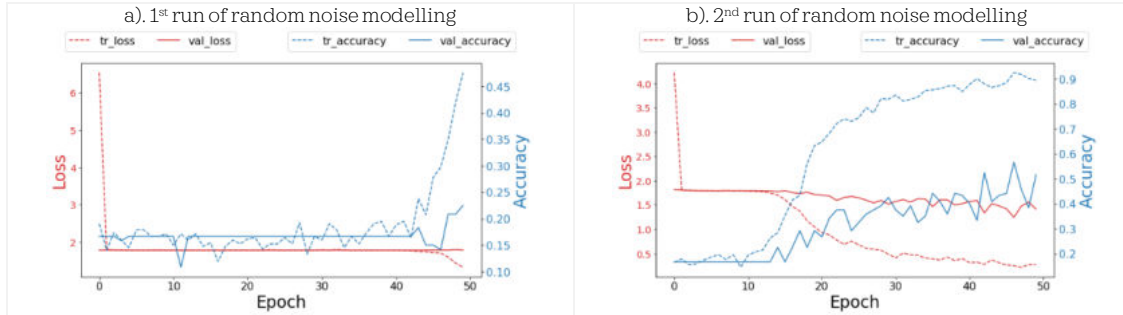


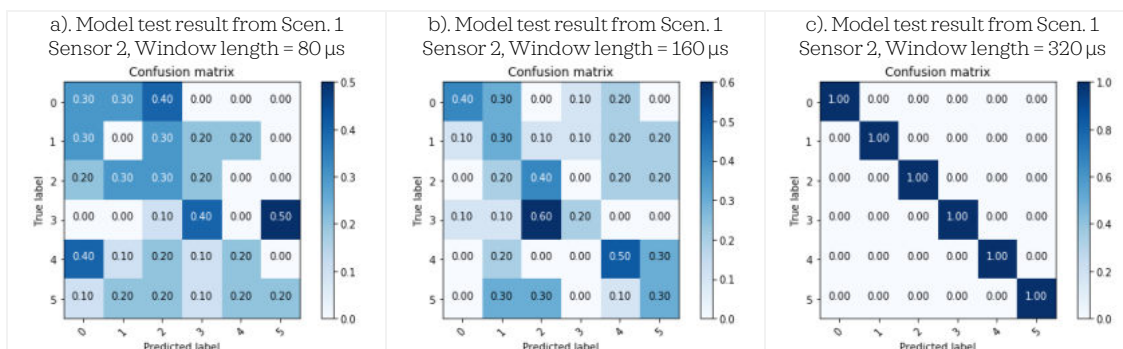
Fig. 6.5.1-2a - b: Sample training results of random noise modelling. Note that the scaling is not uniform due to the default library settings.

The distance between the training and validation accuracy is relatively far (between 20% and 40%), which is a clear sign of heavy network overfitting, yet this is not the most important fact. What is more important to know is that it failed to deliver a consistent result, i.e., the network did not learn anything from the TFR pattern because there is not locally connected spectrotemporal features in the TFR of a noise, indicating that the invariant latent assumption within the noise group does not hold. In fact, the network simply memorized the noise. More figures are available upon demand, or the readers are welcome to clone the repository [Ewald et al. (2019)] to self-experiment with it.

6.5.2. Model Testing

6.5.2.1. Hierarchical Representation in Multiple Sensors

To test the validity of the models as to whether they can classify the signal or not, a confusion matrix must be calculated. As a sample result, the test results were shown from scenarios 1 and 2 for sensor 2 for convolutional window lengths of $80 \mu\text{s}$, $160 \mu\text{s}$, and $320 \mu\text{s}$ which are depicted in Fig. 6.5.2.1-1a - f, respectively. Further, the labels “0” to “5” in Fig. 6.5.2.1-1a - c correspond to the damage condition of scenario 1 described in Table 6.3.1-1 with “0” as the baseline and “5” as the critical crack length a_{crit} . The same logic applies for Fig. 6.5.2.1-1d - f from scenario 2 described in Table 6.3.1-1, where in this case, “0” is the 0° -oriented crack with a length of $10\% a_{\text{crit}}$ and “8” is the 45° -oriented crack with length of $30\% a_{\text{crit}}$.



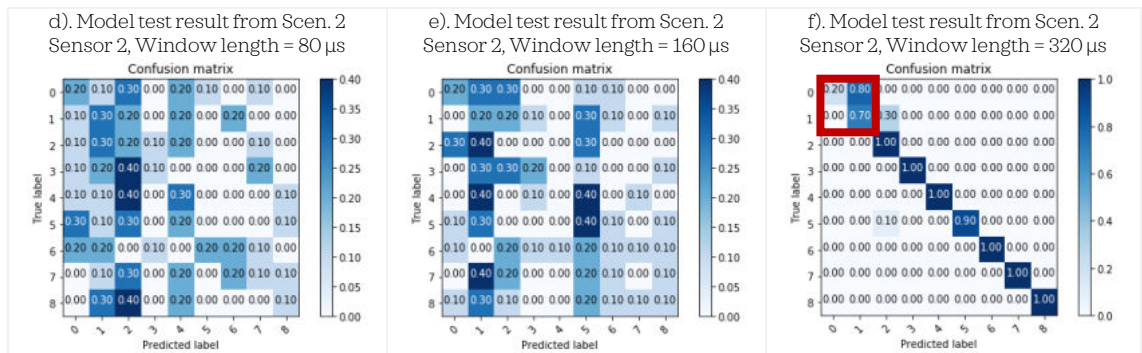


Fig. 6.5.2.1-1a - f: Model test result from scenario 1 and 2 for sensor 2 with varying window length between 80 and 320 μs .

The labelling itself is currently not important as this can be changed in the code. A diagonal with an average of 1.00 such as Fig. 6.5.2.1-1c corresponds to 100% POD. In Fig. 6.5.2.1-1f (marked in red rectangle), it is obvious that the CNN cannot correctly classify the smaller cracks, which in this case “0” is an 0° -oriented crack of 10% a_{crit} , whereas “1” is a 15° -oriented crack of 10% a_{crit} . This is to be expected since the captured signals from a setup where there are minimal differences in crack length and orientation will be likely similar to each other. For a detailed proof, the relative entropy and the cosine similarities between two time-series can be calculated, however this is not the focus of this work. As the result is very coherent to validation accuracy depicted in Fig. 6.4.1.2-1a, we can conclude that the generalization guarantee via Lemma 6.2.1-2 [Kawaguchi et al. (2017)] holds.

6.5.2.2. Representation as Conserved Entity over Time

When modelling the feature representation in a single conserved entity, the network does not seem to have a problem at all classifying the test data in an accurate way with 100% POD as depicted in Fig. 6.5.2.2-1a - c. The meaning of the labels “0” to “8” is analogous to our explanation in section 6.5.2.1.

While more test results are available, for brevity only three example results are shown in Fig. 6.5.2.2-1a - c that depict result from scenario 2 for conserved entity representation that fused responses from sensor 1;2;3, 1;5;7, and 2;4;6, respectively. All other tests yield a 100% POD. Also, here I hypothesize that it learned the locally spectrotemporal features within the entity in a more subtle and concise way as there is more information embedded in this ‘data cube’.

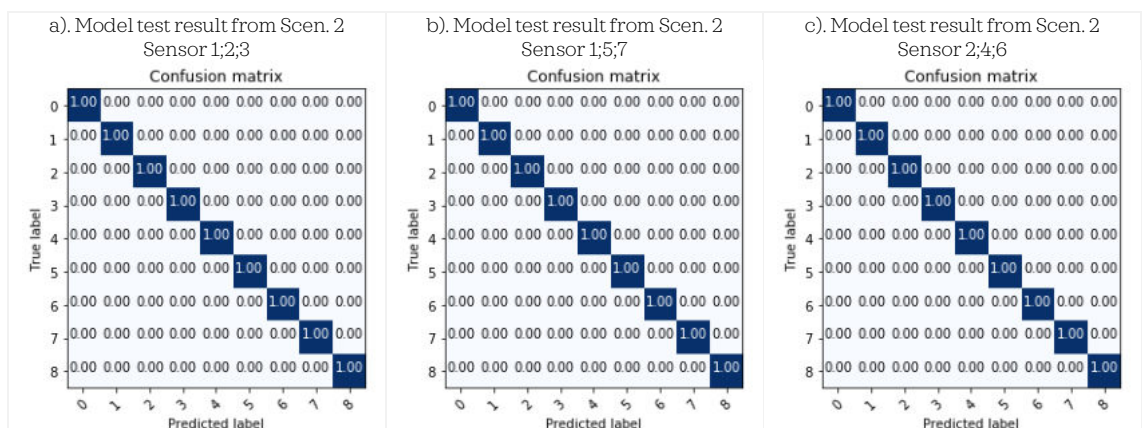


Fig. 6.5.2.2-1a - c: Model test results from scenario 2 for conserved entity representation that fused different sensor responses.

6.6. Conclusion

6.6.1. Summary

This chapter is an extension to the previous work [Ewald (2019)], in which the generalized idea about diagnostics was conceptualized and then formalized as the DeepSHM framework. In section 6.2, I elaborated my thoughts about perception from a neuroscientific perspective incorporated into the SHM and NDT domains, while the methodology used such as the numerical model, data pre-processing, and neural network training are described in section 6.3. The results and discussion are given by section 6.4, and the validation of the concept is given by section 6.5.

6.6.2. Conclusion and Recommendation

Before making any recommendation, recall the hypothesis stated in section 6.1:

... that some of the algorithms could not make a distinction between the signals that come from slightly similar a distribution. I hypothesized that this problem might be overcome by:

1. *Applying broadband frequency excitation since broadband excitation frequency will normally invoke more variable wavenumber, and broader wavelengths.*
2. *Varying sensing location to obtain more potential information.*

Based on the results, we can state that the CNN captures more information due to the varying sensing location and varying wavelength thanks to the chirplet-like excitation. However, this approach encounters its limit as well, as can be seen in Fig. 6.5.2.1-1f. It is possible to enlarge the excitation band frequency to include more information, however this would have two drawbacks: 1). A more expensive wideband PZT must be used, and 2). Assigning a human data analyst to analyze such a complex signal with traditional signal processing would require more time and it would be a disadvantage if we would like to understand and relate meaningful signal features to the physical domain of the structure.

My recommendation is therefore to use the quasi chirp-like excitation only when it is coupled with advanced machine learning, which in any case is already a black-box. Employing a machine learning algorithm such as CNN requires a quality control of both input and output. Further, the following questions were stated the beginning:

1. *How much do the varying sensing locations and the different sensing representations of the time-frequency Lamb wave signal influence the deep learning training behavior?*
2. *Given “a posteriori knowledge” from (1), what consequence can be drawn for the engineering application in SHM and why should this approach work?*

Objectively I am firm that there is not a single ultimate answer, but that we can see the highly optimally located set of sensors give the most desired training behavior due to better response capture. On the other side, when the information is fused first from different sensing locations, that is in the case of conserved entity representation, it can clearly be seen that the sensing position does not matter anymore as has already been explained in section 6.4.2.

Revisiting the term “no free lunch”, it is fair here to state that the training cost in terms of time also increases when the network must learn more parameters. At this moment, I believe decreasing the neural network size is the most feasible way to make deep learning for structural diagnostics is scalable for industrial production, with the caveat that limiting the amount of model parameters would risk that the training would be longer. In the worst case where the size of data is limited, this would lead into a less generalizable model. In future work, it would be of interest to investigate the scalability level of deep learning for SHM for a given data size, model parameters, and restrictions on physical memory.

Also, another consideration that should be discussed is the actuator position. If the actuator position is changed, the signal representation also changes. While it is possible to determine the damage even if the actuator position has changed, the model needs to be changed as well. This is a limitation of deep learning where it learns the statistical distribution from given samples. This principle applies to all type of supervised networks such as MLP and its derivatives, CNN and its derivatives (LeNet, VGG, ResNet, etc.), recurrent and its derivatives (LSTM, GRU, RNN), etc. To ensure that the damage types from different actuator locations can be detected, there are two possibilities:

1. A new model based on that actuator-sensor pair must be created, or
2. The training data from all possible actuator-sensor pair must be included during the training.

It is possible to train data coming with 2 sensors only and we can expect that the performance lies between a single sensor response and a 3-sensor response (cf. Section 6.5.2.1 and section 6.5.2.2). As mentioned, the entity is represented via two different perceptions:

1. The behavior of Lamb wave propagation is sampled for a single sensor and different window lengths are applied to generate multiple perceptions, resulting in different chopped time-series which were converted via reassigned STFT to generate greyscale array (Fig. 6.3.3.1-1)
2. The behavior of Lamb wave propagation is sampled in n -sensor locations (e.g., $n = 3$ in our case) with a full window length (full length is adjustable, but we have 400 μ s). These time-series were again converted into reassigned spectrogram and joined together to form image in Fig. 6.3.3.2-1. It is possible to join signals from more sensors ($n > 3$), but of course these cannot be depicted as RGB images anymore.

Conclusively, the second perception modelling can be regarded as the extension of the first modelling and as academia, we might be tricked into thinking “higher performance = better” so we will be tempted to use more and more sensors data ($n > 20$) to improve performance, until we meet the big O in the time-space complexity.

A further limitation in this current study is that only damage classification techniques being employed. For damage localization, the label in the datasets can be expanded with the location of the corresponding damages such as proposed by [Hu et al. (2020), Zhang et al. (2021)] and retraining the models. In this case, the model will be assigned to classification and regression tasks at the same time. Computationally, this is not a problem per se. However, the availability of the data in such case is very scarce so that the proposed DeepSHM may work well only in the medium to long-term run when enough data is available.

For the short-term, some augmentation techniques and generative modelling should probably be incorporated to bypass the data scarcity. For a conservative industry like aerospace, this is a perfect pace because the industry and its people and process are moving slowly – this will provide enough time to the major OEMs to place a strategy on their investment and to make a right prioritization.

Source codes and online Documentation

Fedotenkova M. Available online <https://github.com/mfedoten/reaspectro> (Last online: FEB-2020).

Github repository. Available online <https://github.com/vewald/DeepSHM> (Last online: SEP-2020).

Keras Optimizers Documentation. Available online <https://www.keras.io/optimizer> (Last online: JUL-2020).

PyTorch Documentation. Available online www.pytorch.org (Last online: AUG-2020).

Tensorflow Documentation. Available online <https://www.tensorflow.org> (Last online: AUG-2020).

Literatures

Bauccio ML. *ASM Metals Reference Book* (3rd Ed). ASM International, Materials Park (1993).

Botev A, Lever G, Barber D. *Nesterov's Accelerated Gradient and Momentum as Approximations to Regularised Update Descent*. International Joint Conference on Neural Networks (IJCNN), 1-5 (2017).

Clayton S. *Topic 10: Rademacher Complexity*. In Lecture series EECS 598: Lecture on Statistical Learning Theory. University of Michigan (2014).

Duczek S, Joulaian M, Düster A, Gabbert U. *Numerical Analysis of Lamb Waves Using the Finite and Spectral Cell Methods*. Intl J for Numerical Methods in Engineering. Vol. 99: 26-53 (2014).

Ewald V, Goby X, Jansen H, Groves RM, Benedictus R. *Incorporating Inductive Bias into Deep Learning: A Perspective from Automated Visual Inspection in Aircraft Maintenance*. Proc. 10th Intl Symposium on NDT in Aerospace, Dresden, 1-9 (2018a).

Ewald V, Groves RM, Benedictus R. *Transducer Placement Option of Lamb Wave SHM System for Hotspot Damage Monitoring*. MDPI J Aerospace. Vol. 5(2): 39 (2018b).

Ewald V, Groves RM, Benedictus R. *DeepSHM: A Deep Learning Approach for Structural Health Monitoring Based on Guided Lamb Wave Techniques*. Proc. SPIE Smart Structures and NDE, Denver, 1-16 (2019).

- Ewald V, Ochoa P, Groves RM, Benedictus RM. *Design of a Structural Health Monitoring System for a Damage Tolerance Fuselage Component*. Proc. 7th Intl Symposium on NDT in Aerospace, Bremen, 1-9 (2015).
- Ewald V, Venkat RS, Asokkumar A, Benedictus R, Boller C, Groves RM. *Perception Modelling by Invariant Representation of Deep Learning for Automated Structural Diagnostic in Aircraft Maintenance: A Study Case using DeepSHM*. J Mechanical System and Signal Processing. Vol 165: 108153 (2022).
- Feige I. *Invariant-Equivariant Representation Learning for Multi-Class Data*. 36th Conference on International Conference on Machine Learning (ICML), Long Beach, 1-5 (2019).
- Gardner P, Worden K, Liu X. *On the Application of Domain Adaptation in Structural Health Monitoring*. J Mechanical Systems and Signal Processing. Vol. 138: 106550 (2020).
- Géron A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Sebastopol CA (2017).
- Goldberg PW, Jerrum MR. *Bounding the Vapnik-Chervonenkis Dimension of Concept Classes Parameterized by Real Numbers*. J Machine Learning, Vol. 18(2-3): 131-148 (1995).
- Gormley M. *Lecture 28: PAC Learning*. In Lecture series 10-601: Introduction to Machine Learning. Carnegie Mellon University (2016).
- Hayo T, Frankenstein B, Boller C, Bockenheimer C. *Approach to the Technical Qualification of a SHM System in Terms of Damage Detection in Aerospace Industry*. Proc. Intl Workshop Smart Materials, Structures & NDT in Aerospace, Montreal, 1-9 (2011).
- He K, Zhang X, Ren S, Sun J. *Deep Residual Learning for Image Recognition*. Proc. Conf on Computer Vision and Pattern Recognition (CVPR), Las Vegas (2016)
- Hu C, Yang B, Yan J, Xiang Y, Zhou S, Xuan FZ. *Damage Localization in Pressure Vessel by Guided Waves Based on Convolution Neural Network Approach*. J. Pressure Vessel Technol. Vol. 142(6): 061601 (2020).
- Huber A. *Dispersion Calculator Software*. Available online https://www.dlr.de/zlp/en/desktopdefault.aspx/tabid-14332/24874_read-61142 Last online: JUN-2020).
- Kawaguchi K, Kaelbling LP, Bengio Y. *Generalization in Deep Learning*. Mathematics of Deep Learning, Cambridge University Press, to appear. Preprint available as: MIT-CSAIL-TR-2018-014
- Kim Y, Huang J, Emery S. *Garbage In, Garbage Out: Data Collection, Quality Assessment and Reporting Standards for Social Media Data Use in Health Research, Infodemiology and Digital Disease Detection*. J Med Internet Res., Vol. 18: e41 (2016).
- Merck: *Future of AI Challenge*. Available online <https://app.ekipa.de/challenges/future-of-ai/brief> (Last online: FEB-2020).
- MIL-HDBK-1823A. *Non-Destructive Evaluation System Reliability Assessment*. US Department of Defense, Wright-Patterson (2009).
- Mitchell T. *Machine Learning*. McGraw-Hill, Redmond & Ithaca (1997).
- Moran S, Yehudayoff A. *Sample Compression Schemes for VC Classes*. J Communications of the Association of Computing Machinery (ACM). Vol. 63(3): 1-21 (2016).
- Morales JL. *A Numerical Study of Limited Memory BFGS Methods*. J Applied Mathematics Letter. Vol. 15(4): 481-487 (2002).
- Niethammer M, Jacobs LJ, Qu J, Jarzynski J. *Time-Frequency Representations of Lamb Waves*. J Acoustical Society of America. Vol.109(5): 1841-7 (2001).
- Ooijselaar T. *Vibration-based Structural Health Monitoring of Composite Skin-stiffener Structures*. PhD Diss, University of Twente (2014).
- Paoletti ME, Haut JM, Plaza J, Plaza A. *A New Deep Convolutional Neural Network for Fast Hyperspectral Image Classification*. J of Photogrammetry and Remote Sensing (ISPRS), Vol. 145(A): 120-147 (2018).
- Pendleton SD, Andersen H, Du X, Shen X, Meghjani M, Eng YH, Rus D, Ang MH. *Perception, Planning, Control, and Coordination for Autonomous Vehicles*. MDPI J Machines, Vol. 5(1): 1-54 (2017).
- Rose JL. *Ultrasonic Guided Waves in Solid Media*. Cambridge University Press (2014).
- Rubinstein RY, Kroese D. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*. Springer-Verlag, New York (2004).
- Ruder S. *An Overview of Gradient Descent Optimization Algorithms* (2016). Available online <https://arxiv.org/abs/1609.04747> (Last online: FEB-2020)
- Shalev-Shwartz S, Ben-David S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, Cambridge (2014).

- Simonyan K., Zisserman A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Proc. Intl Conf on Learning Representations (ICLR), San Diego (2014)
- Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. J Machine Learning Research, Vol. 15: 1929-1958 (2014).
- Szegedy G, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. *Going Deeper with Convolutions*. Proc. IEEE Conf on Computer Vision and Pattern Recognition, Boston (2015).
- Taltavull A. *Structural Health Monitoring Solutions for Patched Metallic Aircraft Repairs Based on Guided Ultrasonic Waves*. MSc Thesis, University of Saarland (2017).
- Ting KM. *Confusion Matrix*. In Encyclopedia of Machine Learning and Data Mining. Springer, Boston (2017).
- Valiant LG. *A Theory of the Learnable*. J Communications of the Association of Computing Machinery (ACM). Vol. 27(11): 1134-1142 (1984).
- Valiant LG. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books Inc., New York (2013).
- Wilcox PD. *Lamb Wave Inspection of Large Structures Using Permanently Attached Transducers*. PhD Dissertation, Imperial College University of London (1998).
- Wolpert DH. *A Mathematical Theory of Generalization: Part I*. J Complex Systems. Vol. 4: 151-200 (1990).
- Wolpert DH. *Stacked Generalization*. J Neural Networks. Vol. 5(2): 241-259 (1992).
- Zayani R, Bouallegue R, Roviras D. *Levenberg-Marquardt Learning Neural Network for Adaptive Predistortion for Time-Varying HPA with Memory in OFDM Systems*. 16th European Signal Processing Conf (EUSIPCO), Lausanne (2008).
- Zhang S, Li CM, Ye W. *Damage Localization in Plate-like Structures using Time-Varying Feature and One-Dimensional Convolutional Neural Network*. J Mechanical Systems and Signal Processing. Vol. 147: 107107 (2021).

7. Recurrent Modelling of Time-Frequency Signal

This chapter partially contains the work that has been submitted as:

1. Ewald V, Goby X, Venkat RS, Benedictus R, Groves RM. *On Stability of Sequential Modelling in Data Driven Predictive Maintenance: A Study Case Using CNN-LSTM for DeepSHM*. Intl J of Structural Health Monitoring (In Review)

Chapter 7 will be divided as follows: The introductory part will be given in section 7.1, while the state-of-the art from which the research problems are derived will be described in section 7.2 and a short theoretical foundation on hybrid convolutional recurrent network will be given in section 7.3. Section 7.4 covers methodology to generate the data which includes the simulation parameters, experimental setup, and data pre-processing methodology. The results will be discussed in section 7.5, and finally, the conclusions of this chapter will be given in section 7.6.

7.1. Introduction

In chapter 6, we have already seen a lengthy discussion about convolutional neural networks (CNN) and their training & test methodology. The data treated came from simulation and there was no experimental validation yet. The issue of scalability has also not been addressed yet, especially for a continuous stream of data. The objective of this chapter is to introduce recurrent modelling of time-frequency signals, i.e., where the latent representation of the CNN is treated as continuous input.

The previous CNN model introduced in chapter 6 poses a static behavior, i.e., it treats the input signals of certain given lengths, despite being hardly trainable on/for inputs of variable lengths. Practically, we will always be limited by the current available hardware. All deep learning libraries require a memory placeholder, that means the acceptable input size cannot be larger than the available physical memory. To overcome this, the computer science community has pioneered the recurrent models. As mentioned early in section 2.2.1 regarding the advances made in machine & deep learning, the two algorithms which are currently most established (or maybe to say instead highly regarded) & commonly employed in sequence-to-sequence modelling [Hochreiter and Schmidhuber (1997), Cho et al. (2014), Vaswani et al. (2017), Devlin et al. (2018)] are:

1. **Recurrent neural network** (RNN), with its vanilla variant, long short-term memory (LSTM), and gated recurrent unit (GRU)
2. **Transformer network** various derivatives of it, such as Bidirectional Encoder Representations from Transformers (BERT), its lighter variant LiteBERT, and Generative Pre-trained Transformer (GPT)

Unlike textbooks, which can be effortlessly scanned & provided as an input to BERT, there is no such luxury within SHM & NDT, and even if the data happens to be available, the lack of annotations thereof remains, simply because the world of SHM-NDT is a relatively niche area in comparison to computer science. In this chapter, I would like to specifically address the usage of recurrent modelling for SHM-NDT applications.

7.2. Related work to Recurrent Modelling in Predictive Maintenance

Since most of the literatures have been covered in chapter 2, in this section only a select works related to health monitoring is taken into consideration.

The earliest work I found for recurrent modelling for machine monitoring is from [Zhao et al. (2016)]. They compared various machine learning models from multi-layer perceptron (MLP), support vector machine (SVM), and shallow and deep LSTM for treating time-series data from Computerized Numerical Control (CNC) milling machine to predict the tool wear. As a result, they concluded that 3-layer LSTM outperforms single layer LSTM which in turns outperforms MLP and SVM. They further extended the work by including convoluted bi-directional LSTM to treat the signal from the milling machine [Zhao et al. (2017)].

It is also interesting that they remarked that while a solely physics-based model such as the Paris or Forman crack growth model have been proven to be successful in industry, their performance is also highly tied to the *“quality and accuracy of domain knowledge about the practical mechanical systems”*. Further, they mentioned that *“in real life, due to complexity and noisy working conditions, such high-quality domain knowledge is often unavailable, which hinders the robustness of these physics-based models”* [Zhao et al. (2016, 2017)]. I also agreed with their argument against solely-physics-based models regarding their inability to be updated with on-line measurements, which limits the flexibility of the applications.

Further, physics-based modelling can only consider the current known domain knowledge, which means it needs either an update or a completely different approach when new knowledge is discovered. While there is nothing wrong with that, but it just highlights the evolution of scientific process. For this reason, hybrid data-driven physics-based models, which are based on historical measured data as decision making support tool from the online data collected from sensors, are necessary to facilitate the Internet of Things (IoT).

[He et al. (2019)] used a 2D CNN component to detect well rod pumping system faults and gradually changing faults, a special type of fault which only becomes apparent once irreversible damage has occurred. The framework consisted mainly of classifying the gradually changing state of the faults present in the pumping system based on visual information with various viewing angles. Their framework is depicted in Fig. 7.2-5.

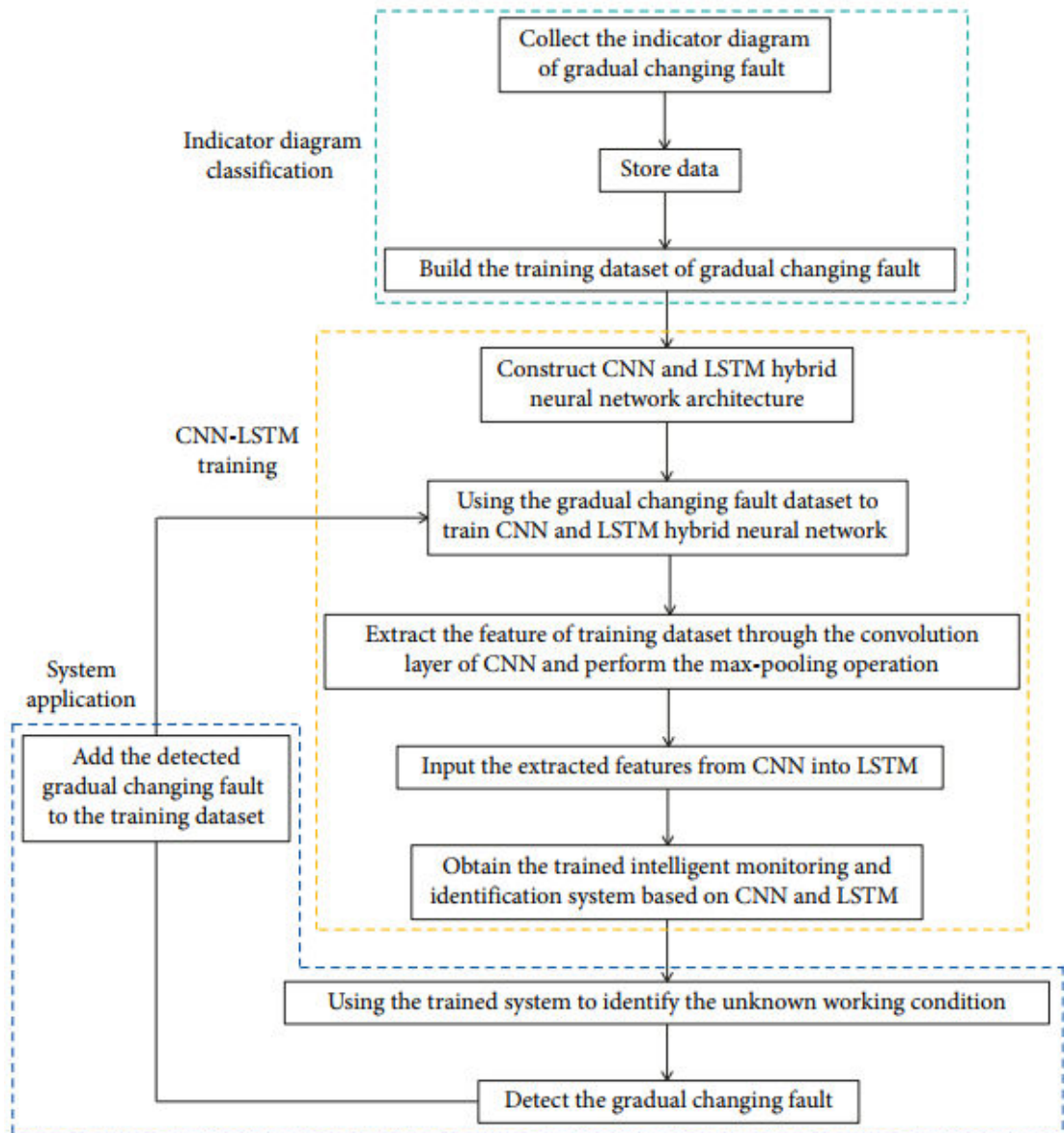


Fig. 7.2-5: End-to-end 2D CNN-LSTM approach for detecting gradual changing fault in rod pump system proposed by [He et al. (2019)].

As seen in Fig. 7.2-5, the 2D CNN extracts features from the images. The extracted features are then amalgamated by LSTM component which enabled so-called “indicator diagrams” to be constructed. These indicator diagrams, along with a SoftMax classifier are used to establish a correspondence of the input data with the type of fault that might be present. Conclusively, their result delivered evidence of the effectiveness of the CNN-LSTM architecture in being able to learn characteristic spatiotemporal features in order to make accurate system fault predictions. A 1D CNN-LSTM hybrid model was also implemented by [Dang et al. (2021)] for the development of a data-driven SHM method incorporating various feature fusion such as the autoregressive model (AR), discrete wavelet transforms (DWT), and empirical mode decomposition (EMD).

The work of [Shenfield and Howarth (2020)] for diagnostics and prognostics focused on a hybrid model consisting of an RNN path and a wide first-kernel Deep CNN (WDCNN). They claimed that this architecture is robust against variable environmental noise and has a low cost of time for both inference &

training. Training & testing was carried out for 4 different RNN type variants of the RNN-WDCNN architecture and of two simpler standard ML models. The end results obtained conclusively demonstrated that the LSTM-WDCNN variant performed the best across nearly the entire board of evaluations.

[Chen et al. (2020)] proposed multi-scale CNN and LSTM model (MCNN-LSTM) to predict bearing faults given a raw vibration signal as an input. They applied 6 convolutional layers followed by 2-LSTM layers and as a result, the F1-score varied between 95.10% to 99.45%. Further, they also tested the proposed model with various signals with SNR ranging from -2 to 10 dB and conclude that the average classification accuracy varies from 81.41 dB (SNR = 0) to 95.25 (SNR = 10). A very similar idea was proposed by [Zhang et al. (2021)], in which some finetuning and hyperparameters differ.

[Huo et al. (2021)] proposed multi-attention parallel CNN-GRU (MAPCG) fault diagnosis by using variational modal decomposition of a rolling bearing as an input to the network. The classification accuracy of their proposed model is 98.5% which is on average higher than other models such as pure CNN, pure GRU, and autoencoders. Not only for acoustic emission signals, a hybrid CNN also-LSTM architecture has been used in NDT-SHM domain for online defect recognition in CO₂-welded joint, as proposed by [Liu et al. (2018)]. The algorithm converges at 300 epochs and the accuracy of defects detection in CO₂ welding molten pool is 94%.

Unlike pure time-domain recurrent modeling approaches such as AE signal classification [Haile et al. (2018)], bridge vibration health monitoring [Xiao et al. (2021)], or system temperature prediction [Mousavi and Gandomi (2021)], there have not been many works in the SHM-NDT domain that involve hybrid CNN-recurrent models. Thus, we can safely conclude that there is currently a lack of usage of convoluted sequential modelling for treatment of Lamb wave signals. Moreover, there is currently no explanation on how these sequential modelling approaches will behave. For this reason, the purpose of this work is to:

1. Demonstrate the capability of hybrid convoluted sequential modelling for treating the spectrogram of Lamb wave signal.
2. Investigate of the stability behavior of convoluted-recurrent modelling for variable spectrogram length
3. Perform the experimental validation of the model for the classification of Lamb wave spectrogram signals.

7.3. Theoretical Foundation of Hybrid CNN-Recurrent Network

As we know from chapter 3, CNN is a special class of feed-forward neural network, the hallmark of which being in the processing of data that is either visually or topologically represented. The two main layers which perform the set of key operations within a CNN are a convolutional layer and a pooling layer. The convolutional layer convolves multiple filters with raw input data and generates

features. The 2D feature map output F resulting from the convolutional operation is defined at pixel (i, j) as:

$$F(i, j) = (K \cdot X)(i, j) = \sum_m \sum_n X(i + m, j + n) (K(m, n)) \quad (7.3-1)$$

In Eq. 7.3-1, K is the 2D convolutional kernel of size $m \times n$, and X is the 2D input signal. The activation function ϕ within the convolutional layers can be chosen from Table 3.4.4-1, although the current most popular activation function is the ReLU function since it offers a good balance between an accurate approximation and the computational cost of traditional non-linear activation such as hyperbolic tangent or sigmoid. The ReLU activation function is defined as:

$$\phi_{\text{ReLU}}(\theta^T X) = \phi(z) = \max(0, z) \quad (7.3-2)$$

In Eq. 7.3-2, z is the product of the transposed neural weight matrix θ and original input signal X . Next, the activated output is passed through a pooling layer which performs a down-sampling of the activated feature map. This yields both a slight improvement in computational performance and a slight boost to invariance to the transformed feature map representation [Lee et al. (2016)]. A commonly used pooling function is the max-pooling function, which selects only the maximum value within the feature matrix region obtained by the convolution filters. The resulting pooling value G is given by:

$$G_i = \max_i(0, F_i) = \max_i \left[0, (K \cdot X)(i, j) \right] \quad (7.3-3)$$

There are two ways to treat the down-sampled feature representation before it is passed through to the FC-layer at the end of the CNN component of the network: either via a flattening function can be employed in order to transform the input into a 1D flattened vector, or with global pooling (GP) layer in order to preserve the spatiality before the classification layer. In either case, the resulting classed representation is provided as input to the fully connected layer which adjusts feature map such that it can be input with a time-series representation to the recurrent segment of the hybrid network. As known from chapter 3, there are currently 3 types of recurrent network: the vanilla RNN, LSTM, and GRU. LSTM and GRU have the advantage over RNN that both have distinctive use of memory blocks and cell states, which are designed to deal with temporal relationships present in data. Since LSTM and GRU overperformed the vanilla RNN, the use of the vanilla variant will not be further discussed, and the work will focus on the LSTM and GRU variants in this chapter. The detailed mathematical operations of LSTM and GRU are given in Section 3.4.3.

7.4. Methodology

This section will describe the methodology used to generate the data, convert it into time-frequency domain signal that will be used as an input, and the

hyperparameter configuration to train the network. There are two ways to generate the data: via Finite Element (FE) simulation and lab experiment. Both methods have its own advantages and disadvantages, which are summarized in Table 7.4.1.

Simulated Data	Experimental Data
<p>Advantages:</p> <ol style="list-style-type: none"> 1. Easily reproducible 2. Simulation parameters can be fine-tuned. 3. Relatively “cheap” in terms of physical resources, i.e., only PC is needed as hardware. <p>Disadvantages:</p> <ol style="list-style-type: none"> 1. Does not represent real-world data, thus needs to be the simulated signal noise. 2. Relatively slow to calculate the response, i.e., highly depends on PC specification. 3. Signal quality highly depends on selected methodology (e.g., FDM vs FEM, etc.) 	<p>Advantages:</p> <ol style="list-style-type: none"> 1. Close to real-world operation data. 2. Huge amounts of data can be easily and quickly captured (within several minutes) 3. Only a standard PC is needed. <p>Disadvantages:</p> <ol style="list-style-type: none"> 1. Relatively “expensive” in terms of hardware setup (waveform generator, PZT sensors, cables, oscilloscope) and materials 2. Environmental conditions are not always easily reproducible. 3. Lab & factory testing conditions do not always represent real-world operational condition

Table 7.4-1: Advantages and disadvantages of using simulation vs experimental data.

Since the simulation methodology has been thoroughly described in Section 6.2.1, it will not describe in this section any further. Instead, only the methodology to generate the data with lab equipment will be described here.

7.4.1. Experimental Validation

The geometrical and physical parameters of the aluminum plate have been described in Section 6.2.1. For the experimental setup, the following hardware is available at the TU Delft Aerospace NDT Laboratory: PZT sensors from American Piezo (Material: APC-850, $\varnothing = 9.52$ mm, 1 mm thickness), standard BNC cables, waveform generator Agilent 33500B, amplifier Agilent 33502A, Picoscope Oscilloscope 7402, cutting machine, aluminum plate with 3 rivet holes.

The crack is assumed to start from the middle rivet hole, and it is simulated by cutting from the middle rivet hole as only the geometrical change is of concern here. Note that this approach is valid only for active scheduled SHM where data is captured on-demand, i.e., thus in reality it is not suitable for online passive monitoring. The data was captured at baseline level, and increasingly in an incremental way at 20%, 40%, 60%, 80%, and 100% critical crack length a_{crit} to simulate 6 distinctive damage classes. Several photos of the experiments are given in Fig. 7.4.1-1a - d.

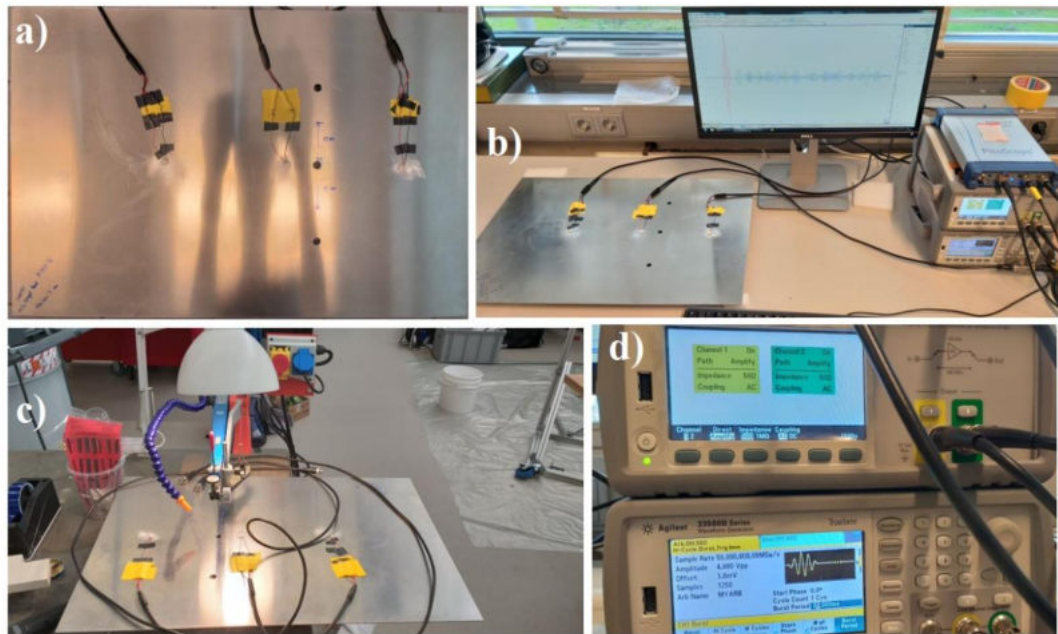


Fig. 7.4.1-1: a). Position of the sensors relative to the 3 rivet holes, b). Experimental setup which consists of an aluminum plate, PC, waveform generator, oscilloscope, and amplifier, c). Simulating the crack propagation process with a cutting machine, and d). Setup of amplifier (top) and waveform generator (bottom).

7.4.2. Sensor Placement

The sensor placement methodology has been described in chapter 6. For brevity of this section, only the results for the two best sensing locations are taken: position 1, located at (360|200) mm and 2, located at (510|200) mm. The actuator is located at (200|200) mm, as can be seen in Fig. 7.4.2-1.

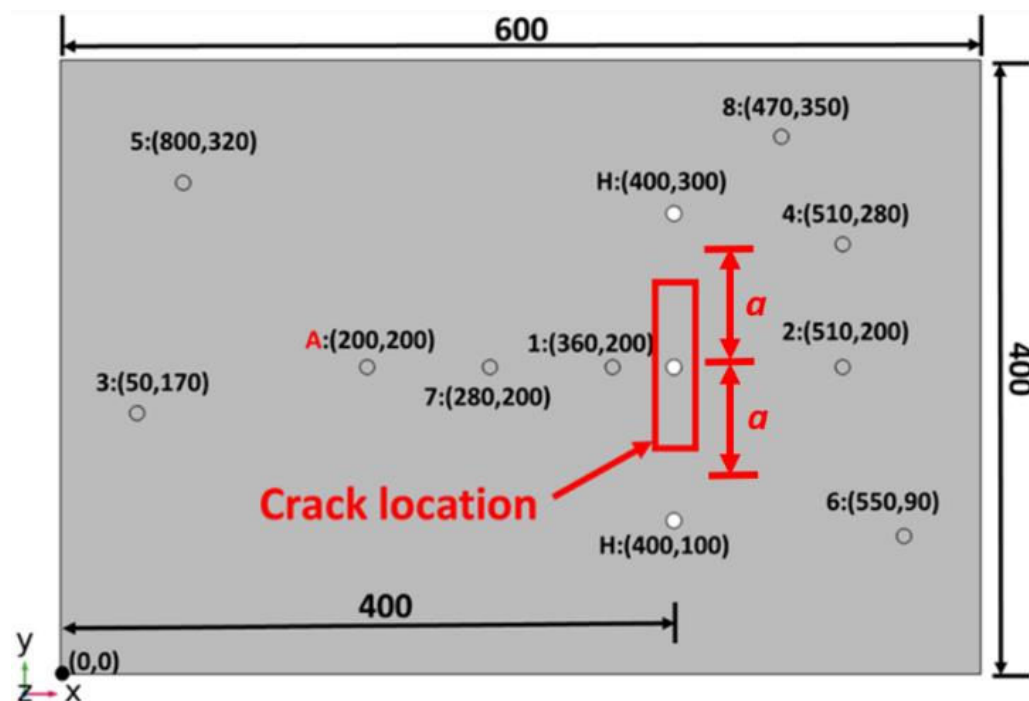
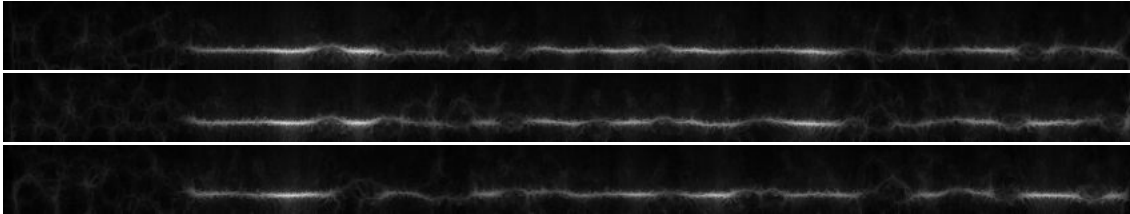


Fig. 7.4.2-1: Sensor positions are numbered from 1 - 8. The coordinates of each sensor position are written in brackets. A is the actuator; and H are the rivet holes. All dimensions are expressed in mm. The sensor locations are numbered from 1 - 8 and a is the half crack length.

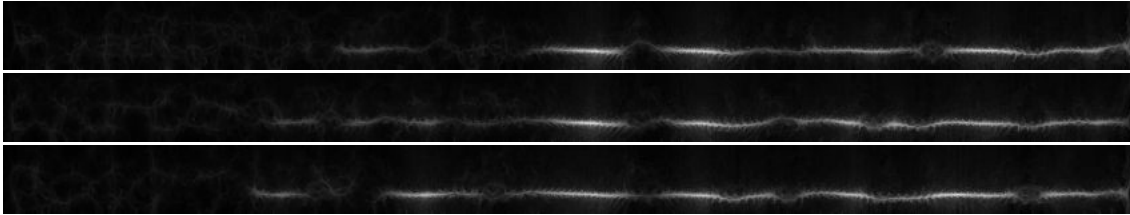
7.4.3. Reassigned Spectrogram

To make a compromise between the time-resolution and frequency-resolution while keeping both with a relatively high resolution, a reassigned spectrogram methodology [Flandrin et al. (2002)] is used, which is based on multitaper technique to estimate the power spectrum of a stationary signal [Thomson (1982), Percival et al. (1993)]. As described in section 6.3.2, the reassignment method is a technique used in TFRs to sharpen and to localize the frequencies nearer to their true regions along time of the signal [Niethammer et al. (2001)]. Examples of the reassigned spectrograms are given in Fig. 7.4.3-1a - d. In each figure, the first, second, and third rows indicate the baseline signal, the signal from the damaged plate with $2a = 40\%$ critical crack length a_{crit} , and the signal from the damaged plate with $2a = 100\%$ critical crack length a_{crit} , respectively. The network topology and the crack location can be seen from Fig. 7.4.2-1.

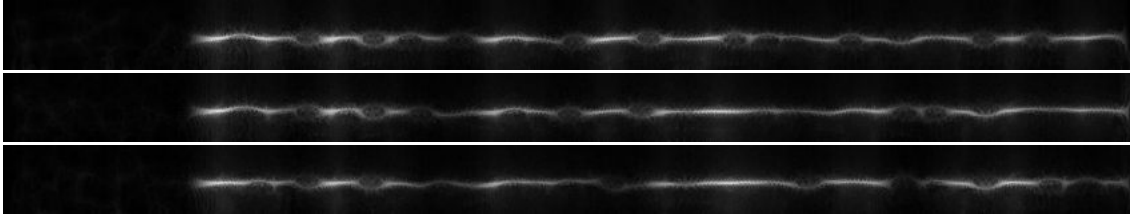
a). Sensor 1, $f = 150$ kHz. 1st row: $\underline{X}(2a = 0)$, 2nd row: $\underline{X}(2a = 40\% a_{crit})$, 3rd row: $\underline{X}(2a = a_{crit})$



b). Sensor 2, $f = 150$ kHz. 1st row: $\underline{X}(2a = 0)$, 2nd row: $\underline{X}(2a = 40\% a_{crit})$, 3rd row: $\underline{X}(2a = a_{crit})$



c). Sensor 1, $f = 250$ kHz. 1st row: $\underline{X}(2a = 0)$, 2nd row: $\underline{X}(2a = 40\% a_{crit})$, 3rd row: $\underline{X}(2a = a_{crit})$



d). Sensor 2, $f = 250$ kHz. 1st row: $\underline{X}(2a = 0)$, 2nd row: $\underline{X}(2a = 40\% a_{crit})$, 3rd row: $\underline{X}(2a = a_{crit})$

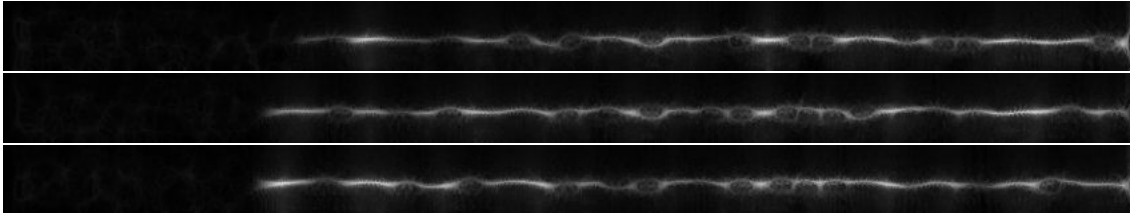
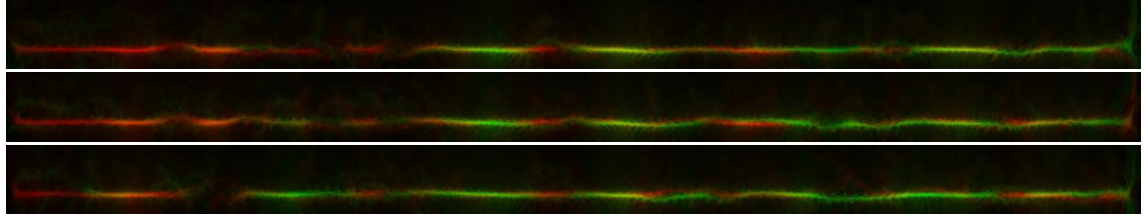


Fig. 7.4.3-1: Reassigned spectrogram for sensing location 1 and 2 with the actuation frequency f of either 150 kHz or 250 kHz. $\underline{X}(2a)$ indicates the spectrogram of the captured signal X for the plate with crack length of $2a$. The geometrical configuration can be depicted in Fig. 7.4.2-1.

When considering the entity representation described in section 6.3.3, the signal depicted in Fig. 7.4.3-1a can be joined together with Fig. 7.4.3-1b to create a joint representation, known as a conserved entity over time. The result is depicted in

Fig. 7.4.3-2a. This can be analogously applied for activation frequency $f = 250$ kHz, as depicted in Fig. 7.4.3-2b.

a). Sensor 1 and 2, $f = 150$ kHz. 1st row: $\underline{X}(2\alpha = 0)$, 2nd row: $\underline{X}(2\alpha = 40\% \alpha_{\text{crit}})$, 3rd row: $\underline{X}(2\alpha = \alpha_{\text{crit}})$



b). Sensor 1 and 2, $f = 250$ kHz. 1st row: $\underline{X}(2\alpha = 0)$, 2nd row: $\underline{X}(2\alpha = 40\% \alpha_{\text{crit}})$, 3rd row: $\underline{X}(2\alpha = \alpha_{\text{crit}})$

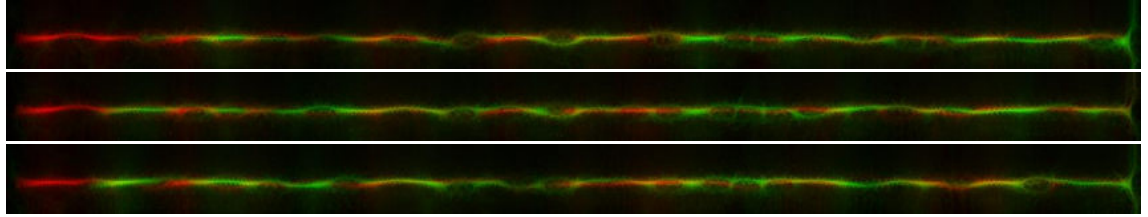


Fig. 7.4.3-2: Reassigned spectrogram for joined captured signal at sensing locations 1 and 2 with the actuation frequency f of a). 150 kHz and b). 250 kHz. The nomenclature is analogous to Fig. 7.4.3-1. The image is saved as portable network graphic (PNG) to conserve the quality of information.

In both examples, the blue channel is filled with the O-matrix, the red channel represents data captured by sensor 1 and the green channel represents data from sensor 2. The convolutional operation is not affected by the color channel assignment, however, for the consistency only the red and green channel in the following section is used.

7.4.4. Recurrent Training Mechanism

At time t_n , the reassigned spectrogram is divided into several sliding windows of different length $s = 128$ pixels, 256 pixels, 512 pixels, and 1024 pixels. The size of the input length into the network is $s \times$ spectrogram bandwidth (239 frequency components) \times number of channels (either 1 or 3). The sliced spectrogram is put as an input into the convolutional layer of the neural network and the abstract representation from the convolutional layer serves as an input for the recurrent layer, which can be either LSTM or GRU. At time t_{n+1} , the sliding windows are moved by several strides (e.g.: 10 x -pixels, i.e.: time-component) and the procedure mentioned before is repeated until the whole spectrogram is covered.

7.4.5. Hyperparameters Configuration

To understand the training stability of the network, we shall compare several hyperparameter configurations:

1. Number of recurrent layers: 2 vs 4
2. Type of recurrent layers: LSTM vs GRU
3. Type of transition layers: Dense (Flatten) vs global pooling (GP)
4. Dropout rate: 0.10 vs 0.25 vs 0.50

The logic to use 2 vs 4 recurrent LSTM and/or GRU layers is due to the vanishing gradient problem. In fact, LSTM and GRU are designed to tackle the problem RNN has by slowing down the vanishing gradient. However, if more LSTM / GRU recurrent layers are embedded, then the vanishing gradient is accelerated again and hence becomes counterproductive against the purpose for it is intended to.

Without any dropout (i.e., 0.0) the network might be prone to overfitting, however when the dropout is = 1, then all neuronal parameters will be reset to some random vectors, rendering the training impossible. The goal is to find some sweet spots between these values without exhaustive search. Note that these combinational lists are not exhaustive, and practically it is almost impossible to cover all possible parameters due to combinatorial variations.

7.5. Results and Discussion

This section will cover the training results for the CNN-recurrent models to understand their training stability. As previously explained, there are two datasets: one from simulation and the other from the lab experiments. These training results will be discussed in sections 7.5.1 and 7.5.2, respectively. The discussion on what essential information as well as knowledge that can be extracted is given in section 7.5.3. Like any machine learning method, training a deep neural network involves stochasticity and sometimes it involves some anecdotal artifacts that might be difficult to verify *a-priori*. To make a conclusion, it is therefore wise to make a slightly larger-scale training to detect any consistent behavior rather than to focus on a certain random artifact. Some important notation in this section with regards to the data is: single-channel spectrogram (e.g., Fig. 7.4.3-1) denotes the converted time-series data from sensor 1 and 3-channel spectrogram (e.g., Fig. 7.4.3-2) denotes the converted time-series data from sensor 1 and 2. The 3rd (blue) channel as mentioned in section 7.4.3 is a matrix filled with zero values.

7.5.1. Simulation Dataset

We start to discuss the training stability based on the selected hyperparameters given in section 7.4. Only a single hyperparameter selection is observed at one time while keeping the other parameters constant to isolate the observation, e.g.: when comparing two different networks with a different number of layers, the dropout rate and the type of layers should be kept the same. The result of training curves can be seen from attachment A.

- a. Comparison between **2-layer GRU vs 4-layer GRU**, both networks have a dropout rate of 0.10 with flattening for transition (see figures in attachment A1 and A2):
 - It can be observed that the training curve of 2-layer GRU behaves more stably for a both single-channel (i.e.: Lamb wave response from single sensor location) and a 3-channel spectrogram (i.e.: Lamb wave

- response from 3 different sensing locations), except for a sliding window length $s = 512$ pixels (Fig. A1-3).
- For a shorter sliding window, the training of 2 channels data converges more quickly than that of single channel only (cf. Fig. A1-1 vs A1-5 and Fig. A1-2 vs Fig. A1-6). This can be confirmed by looking at Fig. A1-4 vs Fig. A1-8, although we can see that in Fig. A1-4, the convergence of the training curve is faster than that of Fig. A1-1 and A1-2.
 - From Fig. A2-1, we can see that training of 4-layer GRU failed for a single channel spectrogram with a sliding window length of $s = 128$, while $s = 512$ and 1024 pixels fails after some epochs (Fig. A2-3 and A2-4, respectively). The 3-channel spectrogram with $s = 256$ pixels also fails after some epochs (Fig. A2-5).
- b. Comparison between **2-layer GRU vs 2-layer LSTM**. Both networks have a dropout rate of 0.10 with flattening as transition layers (see figures in attachment A1 and A3):
- It can be observed that the training of the 2-layer GRU is more stable than the LSTM. From attachment A, we can see that only training for single channel data with $s = 512$ pixels does not converge (Fig. A1-3), while from attachment B, we can see either some very rough training curves or that the training accuracy suddenly falls (Fig. A3-1, Fig. A3-2, Fig. A3-5 and Fig. A3-6).
 - For a sliding window length $s = 512$ and 1024 pixels for both single-channel and 2-channel data, the difference of the training behavior between LSTM and GRU is not significant (cf. Fig. A1-4 vs A3-4, Fig. A1-7 vs A3-7, and Fig. A1-8 vs A3-8).
 - Training with LSTM results in smoother training curve (cf. Fig. A1-3 vs Fig. A3-3).
- c. Comparison between 2-layer LSTM with **flattening and GP as transition layers** (see figures in attachment A3 and A4):
- No significant difference can be observed that except that for the GP layer, the training is slightly noisier, although it successfully converges for single-channel spectrogram with $s = 256$ and 512 pixels.
 - The training failed for a single-channel spectrogram with $s = 1024$ pixels (Fig. A4-4), which might be due to a memory issue.
 - The training for a 3-channel spectrogram succeeds for both GP and flattening transition layers for $s = 256, 512,$ and 1024 pixels (Fig. A3-6 to A3-8 and Fig. A4-6 to A4-8, respectively). However, it can be seen that $s = 1024$ pixels results in the smoothest training curve of all.
- d. Comparison between 2-layer LSTM with flattening layers as transition layer with **varying dropout rate: 0.10 vs 0.25 vs 0.50** (see figures in attachment A3, A5, and A6):

- In general, it can be observed that training with a dropout rate of 0.25 is noisier than 0.10 and this is to be expected. At the same time, training with a lower dropout rate would mean that there is a fair likelihood that a dropout rate of 0.10 can also cause parameter overfitting.
- Training with a dropout rate of 0.25 fails for a single-channel spectrogram with $s = 128$ as the window might be too short (Fig. A5-1 and A5-5), and it behaves inconsistently for $s = 256, 512,$ and 1024 pixels (Fig. A5-2 to Fig. A5-4), as well as for a 3-channel spectrogram with $s = 128$ pixels (Fig. A5-5). The training convergence for 3-channel spectrogram starts at $s = 256$ pixels (Fig. A5-6 to A5-8).
- Training with a dropout rate of 0.50 completely fails (see Fig. A6-1 to A6-8), likely due to the fact too many neurons are dropped in a such way that the network could not learn anymore during the training.

7.5.2. Experimental Dataset

Analogous to section 7.5.1, the description of the result is organized in the same manner. The difference here between Section 7.5.2 is that the dataset used is from the oscilloscope, as described in Section 7.4.1. The result of training curves can be seen from attachment B.

- a. Comparison between 2-layer LSTM with flattening layers as transition layers with **dropout rate of 0.10 vs 0.25** (see attachment B1 and B3):
 - It can be observed from Fig. B3-1 and B3-5 that both training curves of the single-channel and 3-channel spectrogram with $s = 128$ pixels and dropout rate of 0.25 converge, while when using the dropout rate of 0.10, the training failed as can be seen in Fig. B1-1 and B1-5.
 - From Fig. B3-5 to B3-8, the training for all 3-channel spectrograms with dropout rate of 0.25 successfully converge to >98% training accuracy, while for training with dropout rate of 0.10, only the training for the 3-channel spectrogram with $s = 512$ and 1024 pixels converges.
 - The training for a single-channel spectrogram for $s = 256$ pixels and 512 pixels failed (Fig. B3-2 and B3-3, respectively), while for $s = 1024$ pixels the curve converges to >98% training accuracy although the training itself is very noisy (Fig. B3-4).
- b. Comparison between 2-layer GRU with GP layer as transition layer and **dropout rate of 0.10 vs 0.25** (see attachment B2 and B4):
 - From Fig. B2-1 and B2-2, it can be observed that the training for a single-channel spectrogram with $s = 128$ and 256 pixels with dropout rate of 0.10 converges, while when it is increased to 0.25, the training curve became very noisy.
 - From Fig. B2-5 to B2-7, it can be observed that the significant difference in training the 3-channel spectrogram for $s = 128, 256,$ and 512 pixels,

- respectively, lies in the noisiness of the training. However, when training the 3-channel spectrogram with $s = 1024$, it failed.
- From Fig. B2-3, B2-4, B4-3, and B4-4, it can be seen that training the single-channel spectrogram with $s = 512$ pixels and 1024 pixels with dropout rate of 0.10 and 0.25 failed.
- c. Comparison between **4-layer GRU vs LSTM**, both networks had a dropout rate of 0.25 and a GP layer as a transition layer (see attachment B5 and B6):
- The training for both 4-layer GRU and LSTM fails in for all single-channel spectrogram of all window lengths ($s = 128, 256, 512$, and 1024 pixels) as depicted in Fig. B5-1 to B5-4 and B6-1 to B6-4, respectively.
 - On the contrary, when training a 3-channel spectrogram with a 4-layer GRU, the accuracy goes well above $> 98\%$ (Fig. B6-5 to B6-8) for all window lengths ($s = 128, 256, 512$, and 1024 pixels) albeit some noisiness in the training but this is to be expected since the 4-layer recurrent network is more difficult to train.
 - In the meanwhile, only the training with 4-layer LSTM for the 3-channel spectrogram with $s = 512$ and 1024 pixels converged (Fig. B5-7 and B5-8, respectively), while it failed to converge when $s = 128$ and 256 pixels (Fig. B5-5 and B5-6, respectively).
- d. Comparison between **2- vs 4-layer LSTM** with dropout rate of 0.25 and GP layer as transition layer (see attachment B3 and B5):
- The training of the single-channel spectrogram with a 4-layer LSTM failed for all window lengths ($s = 128, 256, 512$, and 1024 pixels) as depicted in Fig. B5-1 to B5-4, while only the training with a 2-layer LSTM for a single-channel spectrogram with sliding window length of 128 pixels was successful (Fig. B3-1).
 - It can be observed that training the 3-channel spectrogram with a 4-layer LSTM only starts to converge when the spectrogram has the window length $s = 512$ pixels (Fig. B5-7 and B5-8).
 - Finally, the training of 3-channel spectrogram was successful for 2-layer LSTM in for all window lengths ($s = 128, 256, 512$, and 1024 pixels) as depicted in Fig. B3-5 to B3-8, respectively.
- e. Comparison between **2- vs 4-layer GRU** with a dropout rate of 0.25 and GP layer as transition layer (see attachment B4 and B6):
- It can be observed that training a single-channel spectrogram with a 4-layer GRU did not converge for all window lengths ($s = 128, 256, 512$, and 1024 pixels), as depicted in Fig. B6-1 to B6-4, while training 2-layer GRU failed with window lengths $s = 512$ and 1024 pixels (Fig. B4-3 and B4-4) – most likely due to a memory issue.
 - It is worth mentioning that the spectrogram training with a 2-layer GRU results in very noisy training behavior (Fig. B4-1 and B4-2).

- No significant difference for training 3-channel spectrogram with 2-layer and 4-layer GRU for all window lengths ($s = 128, 256, 512, \text{ and } 1024$ pixels) can be seen in Fig. B4-4 to B4-8 and B6-4 to B6-8, respectively.

7.5.3. Discussion

This section covers some essential observations found from section 7.5.1 and 7.5.2:

1. Comparison between the **flattening function vs global pooling (GP) layer as transition layer** between the convolutional feature and recurrent part: Using the flattening function typically results in more stable and smoother training curve, but this is to be enjoyed with caution as it might overfit more quickly than a GP layer. The GP layer averaged the final spatial features and by doing so, the convoluted features become more invariant, preventing the overfitting in the fully connected layers. At the same time, it may also introduce more noisy training behavior.
2. The effect of **varying dropout rates**: a dropout rate of 0.10 generally causes the training curve to be smoother than a dropout out rate of 0.25, and a dropout rate of 0.25 is generally making the training curve smoother than that of 0.50. This is general knowledge, but not in coherence with Hinton's group initial proposal [Srivastava et al. (2014)] that the dropout rate of 0.50 is the best. Concludingly, the guideline stating that dropout should be 0.50 cannot not confirmed as all trainings failed (attachment A6).
3. Comparison between **recurrent network mechanisms**: The LSTM type is sometimes smoother than GRU when the dropout rate is set correctly, but from what we can observe from the result, the stability difference is minimal. The difference is only the computational memory used. Since only a very small amount of data is trained, no noticeable difference is observed. However, there is need for scalability study in the future.
4. The **effect of the number of layers**: along with general deep learning knowledge, adding more layers allows a higher network complexity that increases the estimation capacity (cf. Rademacher complexity and VC-Dimension from section 3.4.1). Allowing a higher complexity will allow a more complex pattern to be analyzed, however it also increases the likelihood of overfitting. Therefore, steering the network robustness by adding some regularization such as dropout is necessary to prevent overfitting. From the results, we can see that a 2-recurrent layer network has a better training stability than a 4-recurrent layer network. This is most likely due to the fact that the space-time coherence is becoming "blurrier" as the number of recurrent layers is increased. The pattern complexity that is available in the spectrogram, be it in the single or 3-channel, is likely to be far lower than estimation capacity of the 4-recurrent layer.

As a general remark, for this type of data, especially for a Lamb wave spectrogram, the suggestion is to use 2 recurrent layers with a dropout rate between 0.10 and 0.25. Both the flattening function and GP can be used interchangeably, but the preference should be given for the GP as it may result into more robust network albeit giving slightly noisier training. For the recurrent network type, we can safely conclude that both GRU and LSTM can also be used interchangeably except when it involves large-scale training where the GRU might be more computationally efficient than the LSTM.

7.6. Conclusion

This chapter aims to investigate the training stability of the hybrid convoluted sequential modelling by influencing the hyperparameters involved. The introductory part and the state-of-the art from which the research problems are derived is given in section 7.1 and 7.2. With briefly described theory on the hybrid network in section 7.3 (and partially some others which were mentioned in chapter 3), I hope to have given enough overview on the theoretical foundation, while the research methodology was given in section 7.4. All in all, this leads to the result and discussion given in section 7.5.

As a reiteration from the beginning, hybrid data-driven physics models based on historical data to support decision making from the online data collected from sensors are necessary to facilitate the Internet of Things (IoT) and we know that due to the current limitation of non-recurrent CNN, a continuous Lamb wave signal from an aircraft cannot easily be handled as a one-shot problem, but we must rather align to the problem nature of such signal, i.e., some sequential modelling should be involved. Revisiting the purpose of this work given at the end of section 7.2, my conclusion is following:

1. The hybrid convoluted sequential modelling has clearly a capability to treat the spectrogram of Lamb wave signal opening the potential not only for application in active SHM only, but it could be used for a solution that requires passive monitoring such as acoustic emission.
2. Based on the investigation on stability of training behavior of convoluted-recurrent modelling for a variable spectrogram length, we can safely confirm that training such a hybrid convoluted recurrent network is more difficult as none of the unstable training behavior (such as in attachment A6) occurs with non-recurrent CNN (cf. the results with chapter 6).
3. To increase the training curve stability, there is the possibility to involve more channels in the spectrogram, although it would mean if it has more than 4-channels, a customization for the current deep learning software such as Tensorflow or PyTorch is needed as the image processing library can only take PNG at most (i.e., 4-channels including the alpha layer).

Attachment A

Fig. A1: Training Curves of Simulated Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 2
Type of recurrent layers: GRU	Type of transition layers: Dense (Flatten)
Dropout rate: 0.10	Input: either 1 channel or 3 channels

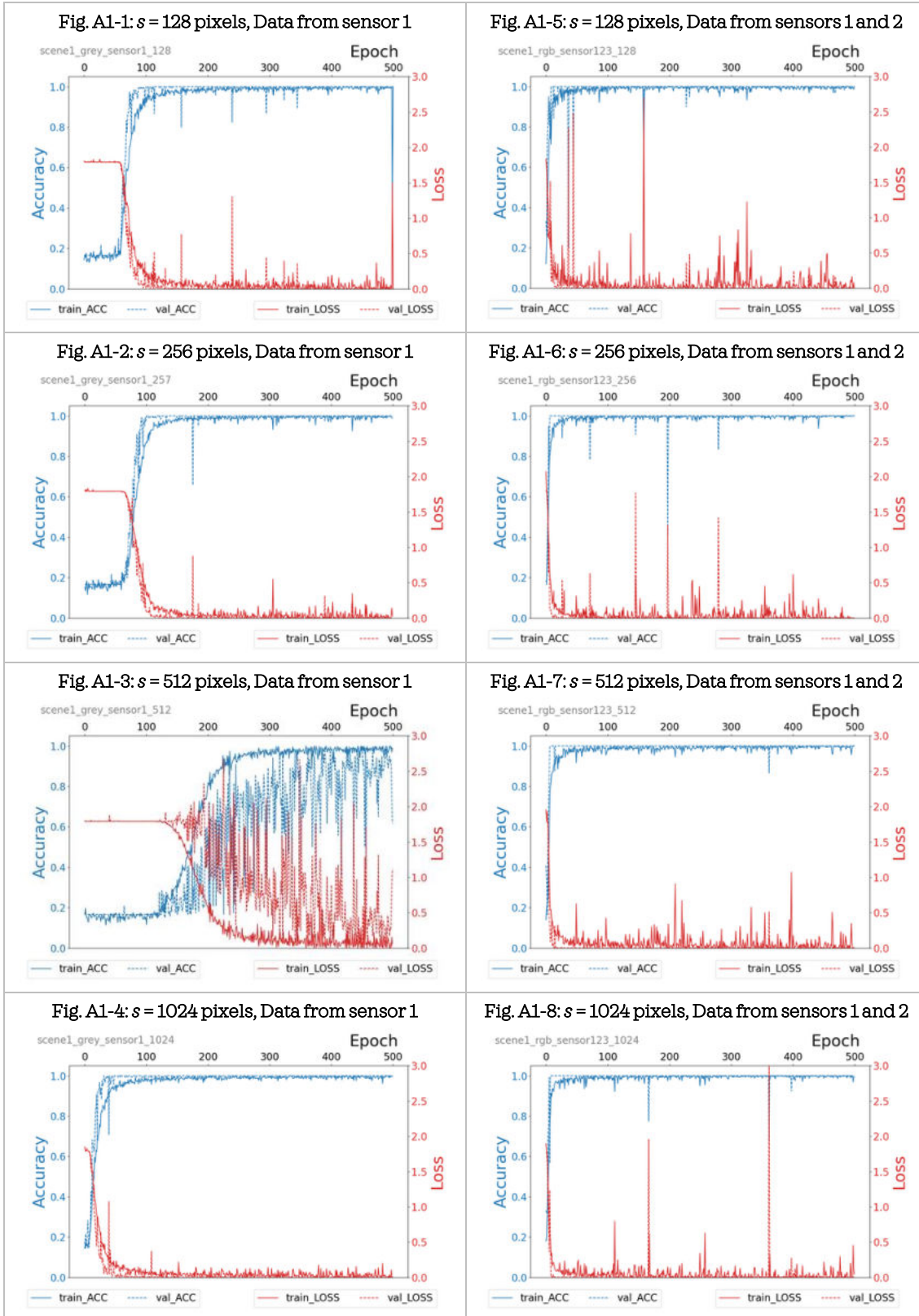


Fig. A2: Training Curves of Simulated Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 4
Type of recurrent layers: GRU	Type of transition layers: Dense (Flatten)
Dropout rate: 0.10	Input: either 1 channel or 3 channels

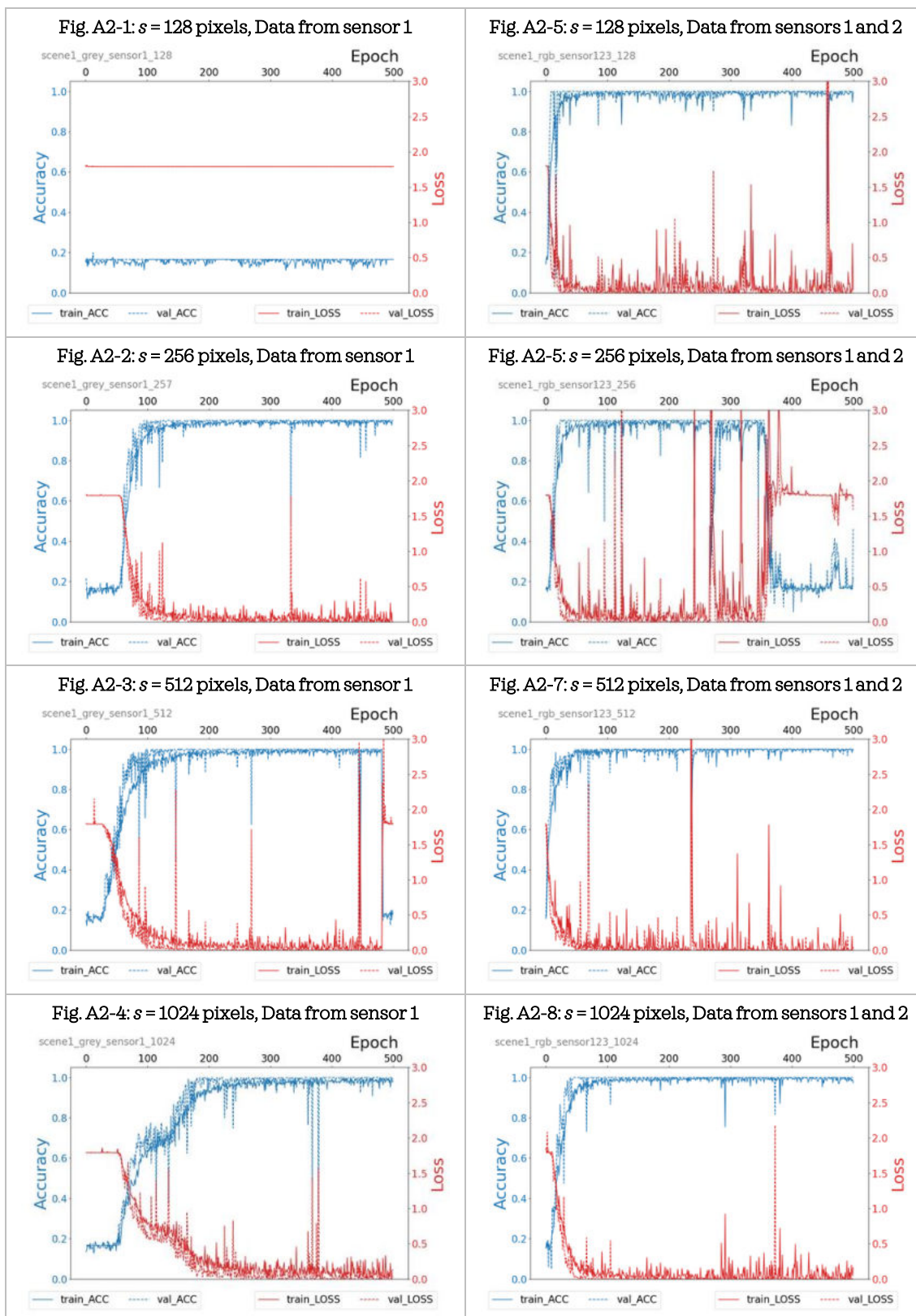


Fig. A3: Training Curves of Simulated Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 2
Type of recurrent layers: LSTM	Type of transition layers: Dense (Flatten)
Dropout rate: 0.10	Input: either 1 channel or 3 channels

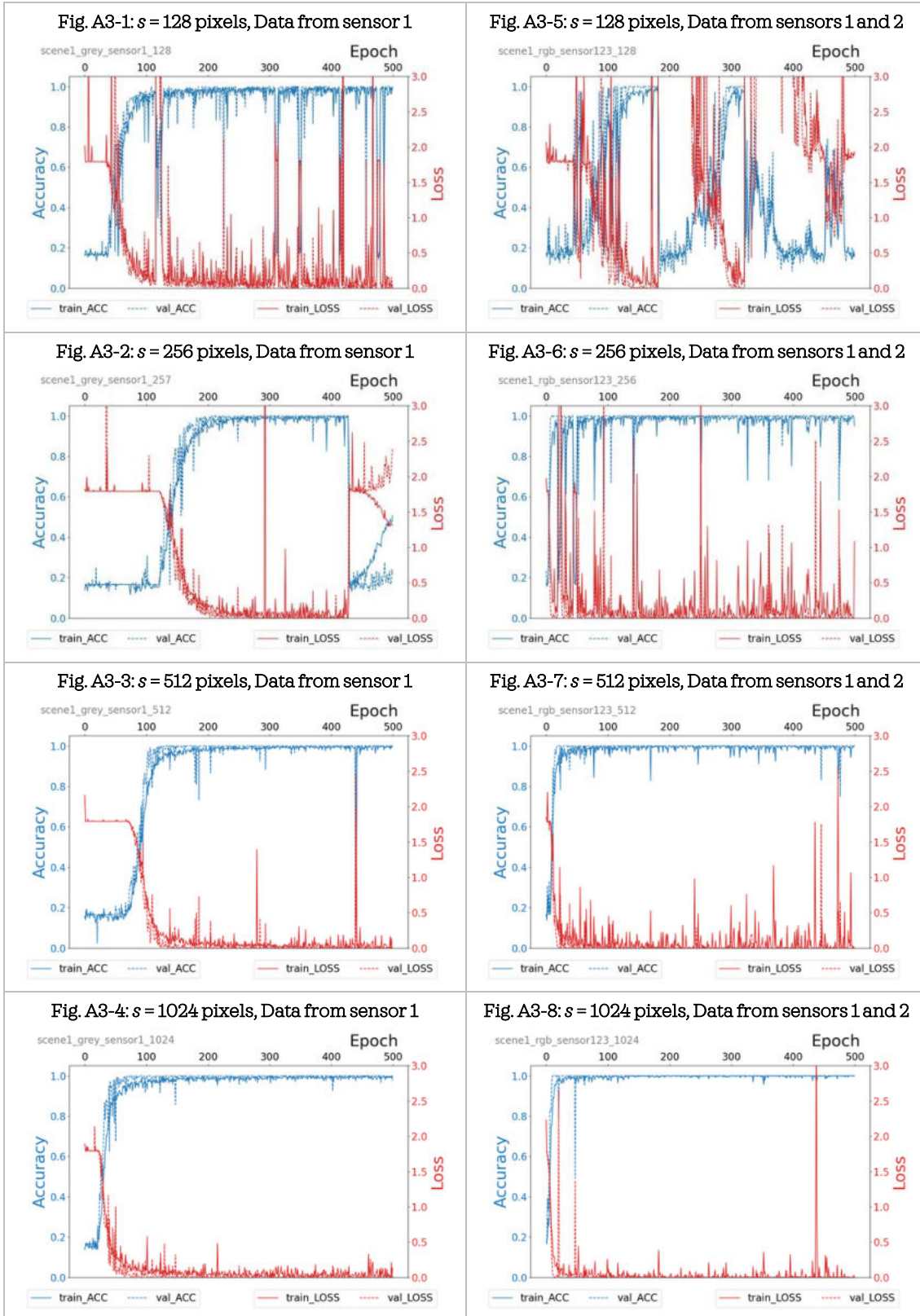


Fig. A4: Training Curves of Simulated Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 2
Type of recurrent layers: LSTM	Type of transition layers: GP
Dropout rate: 0.10	Input: either 1 channel or 3 channels

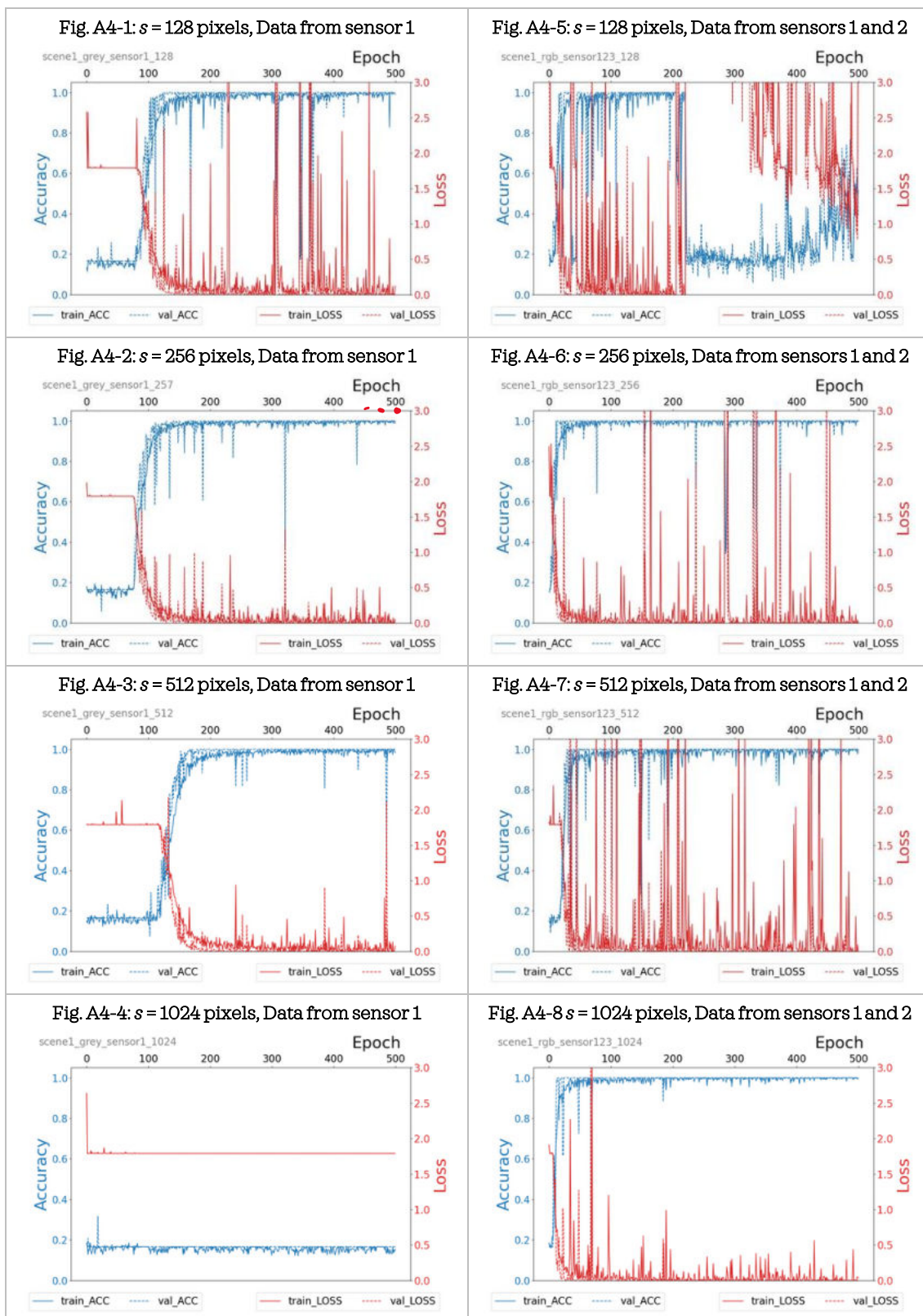


Fig. A5: Training Curves of Simulated Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 2
Type of recurrent layers: LSTM	Type of transition layers: Dense (Flatten)
Dropout rate: 0.25	Input: either 1 channel or 3 channels

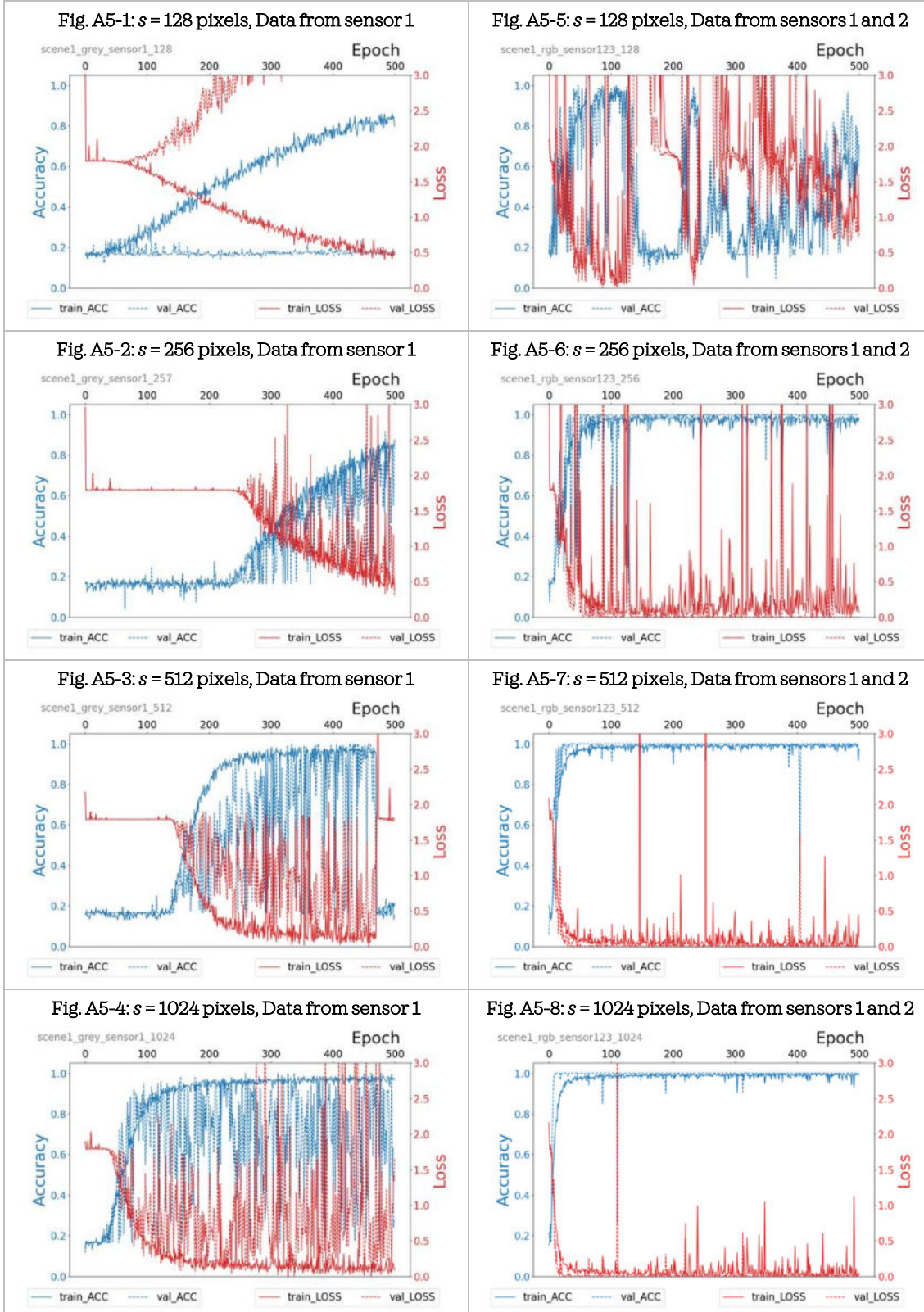
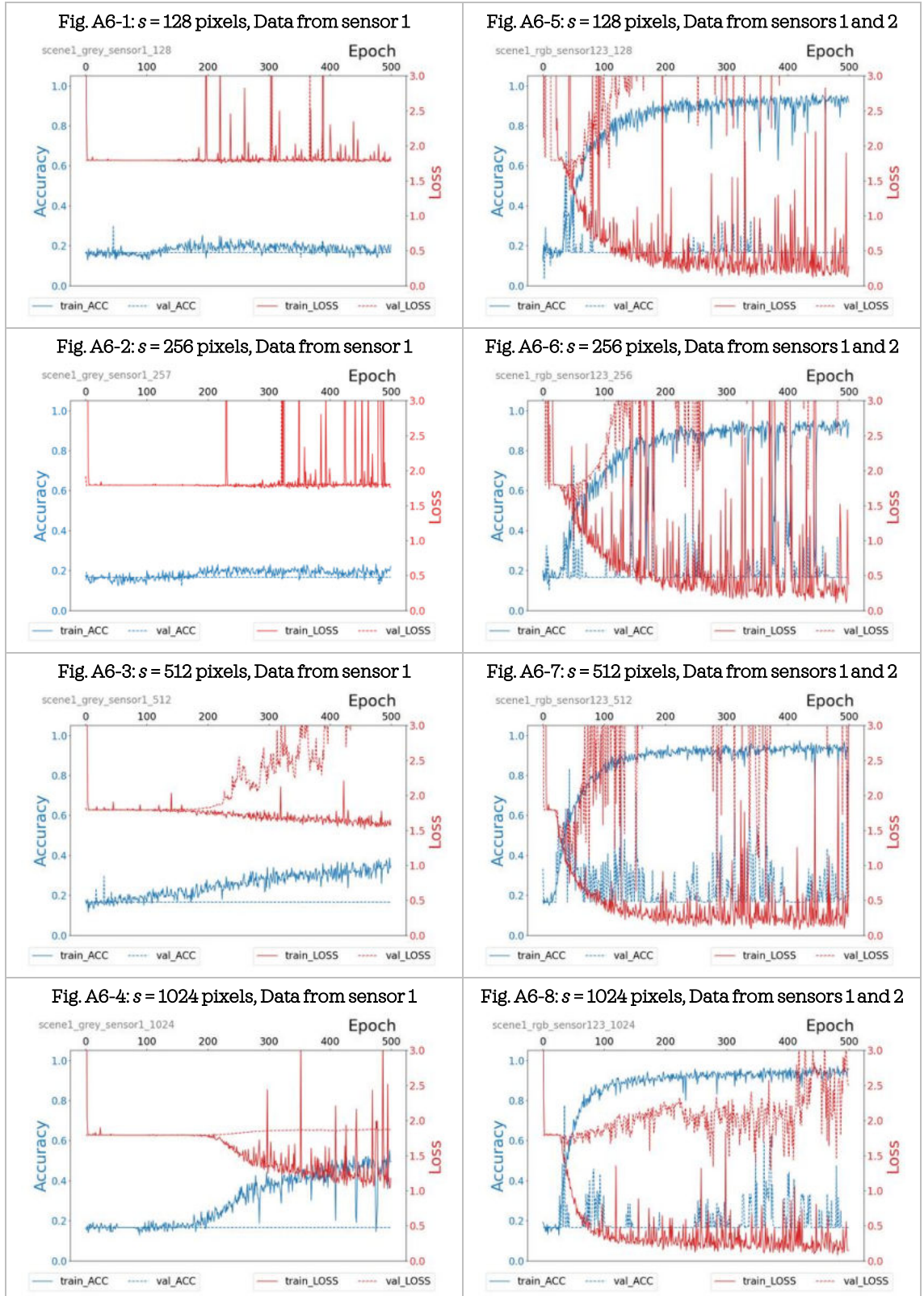


Fig. A6: Training Curves of Simulated Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 2
Type of recurrent layers: LSTM	Type of transition layers: Dense (Flatten)
Dropout rate: 0.50	Input: either 1 channel or 3 channels



Attachment B

Fig. B1: Training Curves of Experimental Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 2
Type of recurrent layers: LSTM	Type of transition layers: Dense (Flatten)
Dropout rate: 0.10	Input: either 1 channel or 3 channels

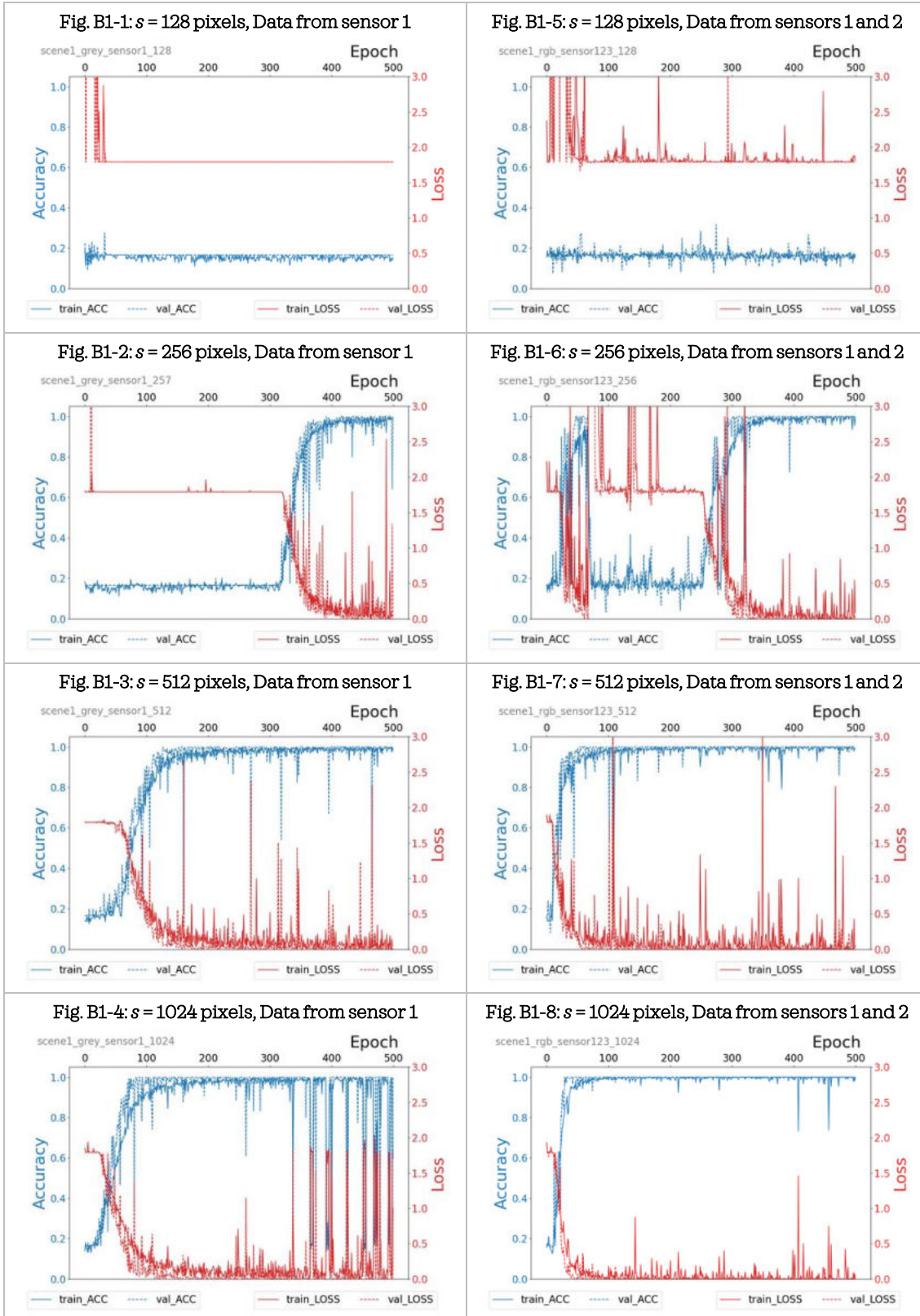


Fig. B2: Training Curves of Experimental Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 2
Type of recurrent layers: GRU	Type of transition layers: GP
Dropout rate: 0.10	Input: either 1 channel or 3 channels

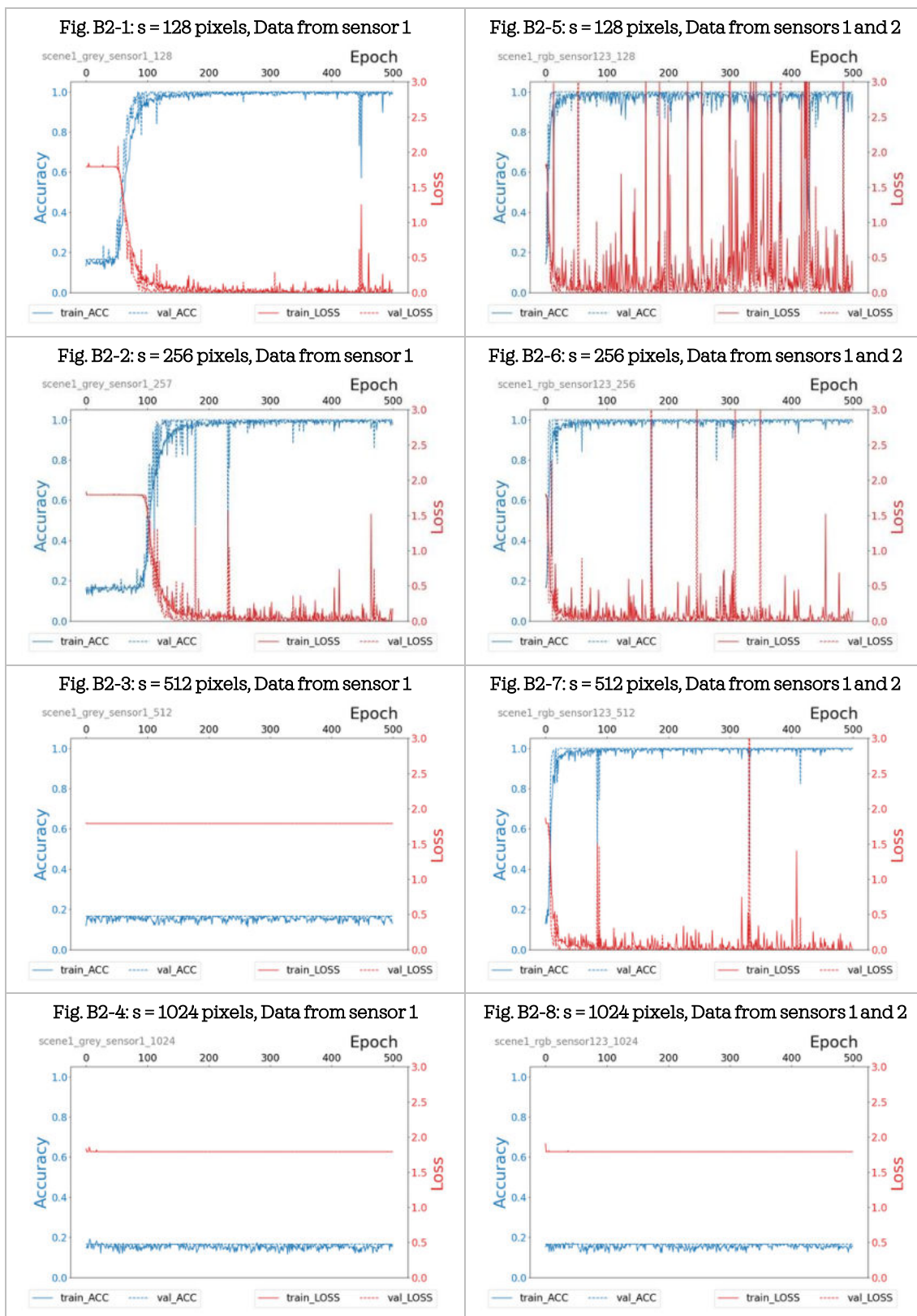


Fig. B3: Training Curves of Experimental Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 2
Type of recurrent layers: LSTM	Type of transition layers: Dense (Flatten)
Dropout rate: 0.25	Input: either 1 channel or 3 channels

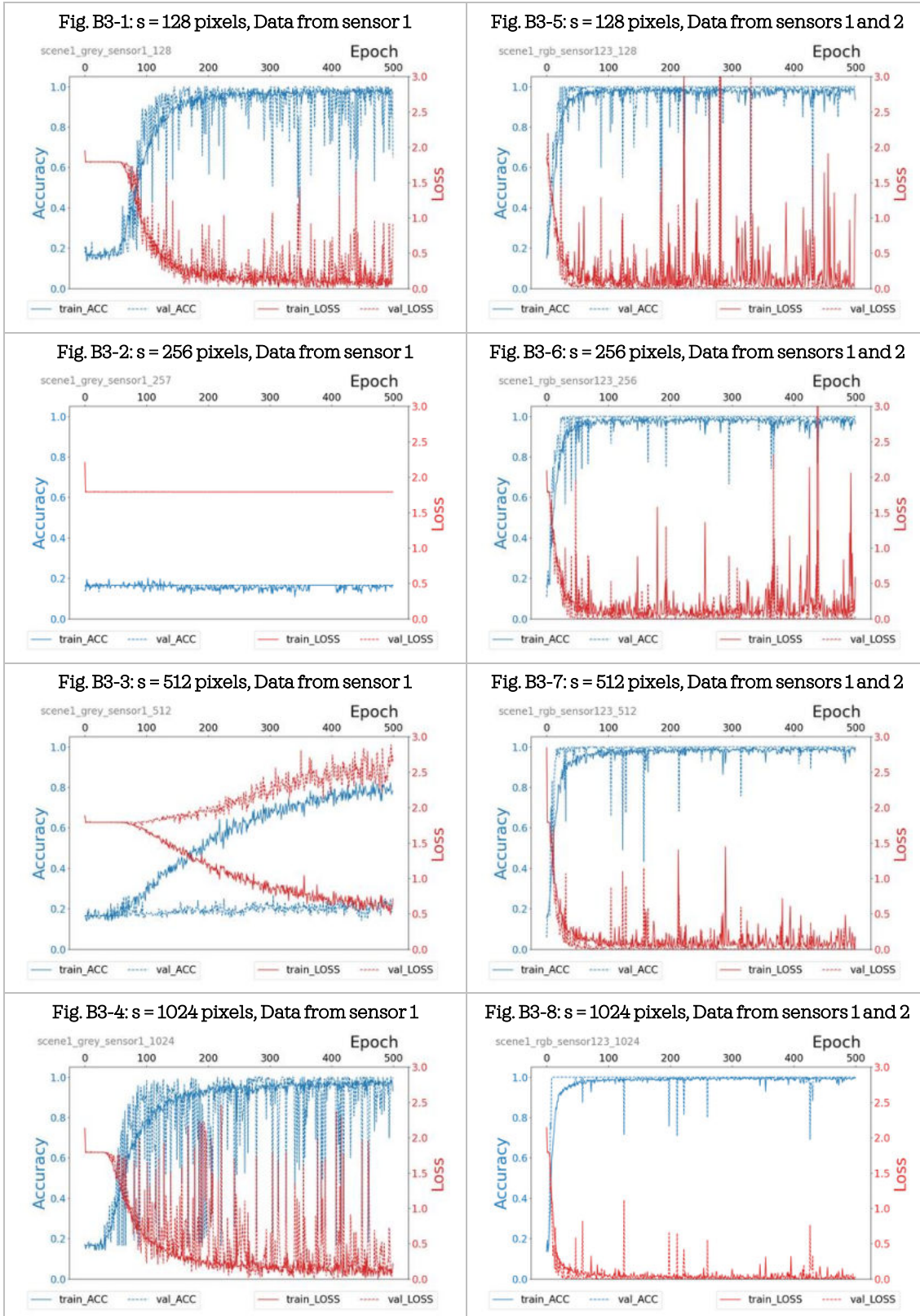


Fig. B4: Training Curves of Experimental Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 2
Type of recurrent layers: GRU	Type of transition layers: GP
Dropout rate: 0.25	Input: either 1 channel or 3 channels

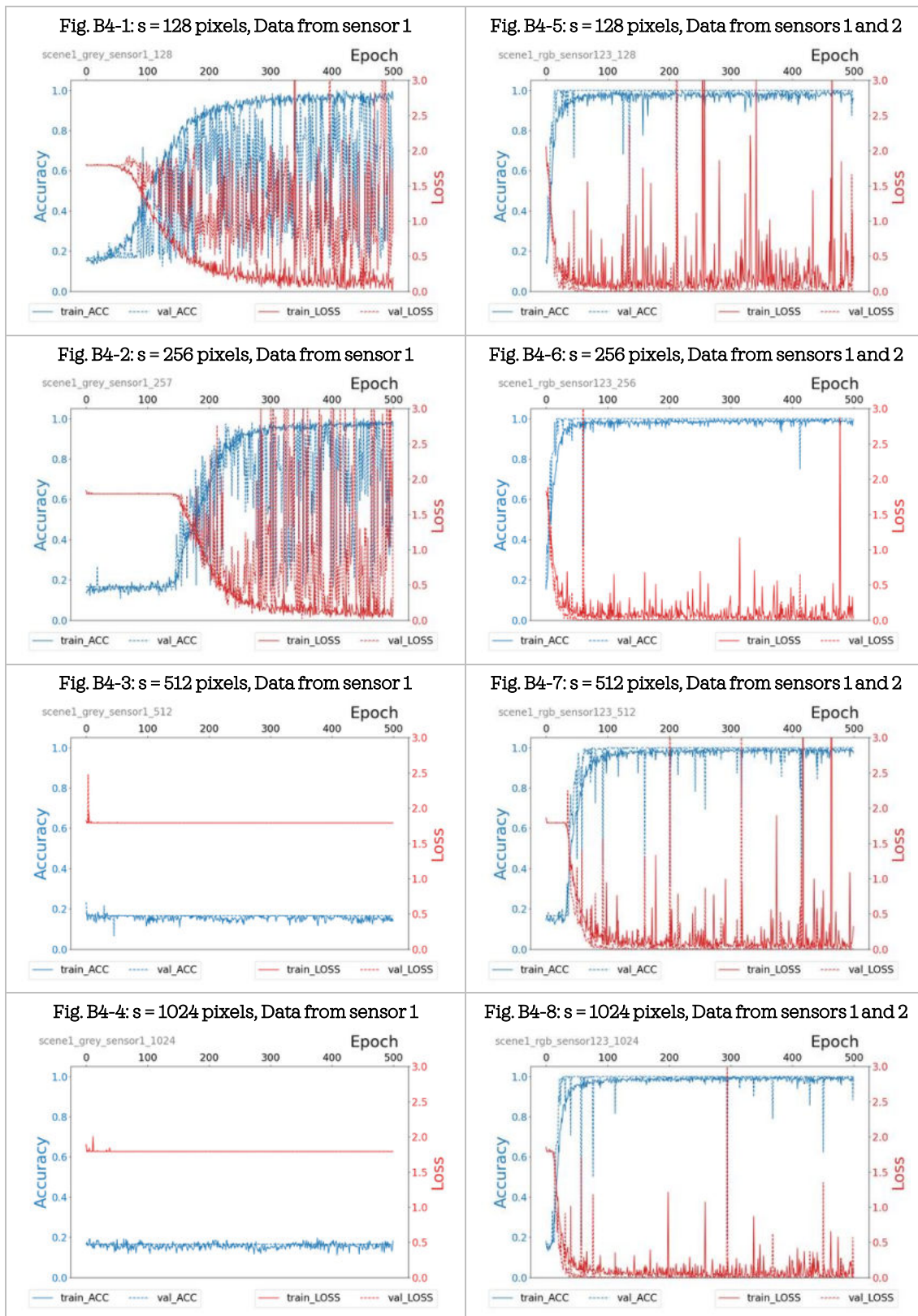


Fig. B5: Training Curves of Experimental Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 4
Type of recurrent layers: LSTM	Type of transition layers: GP
Dropout rate: 0.25	Input: either 1 channel or 3 channels

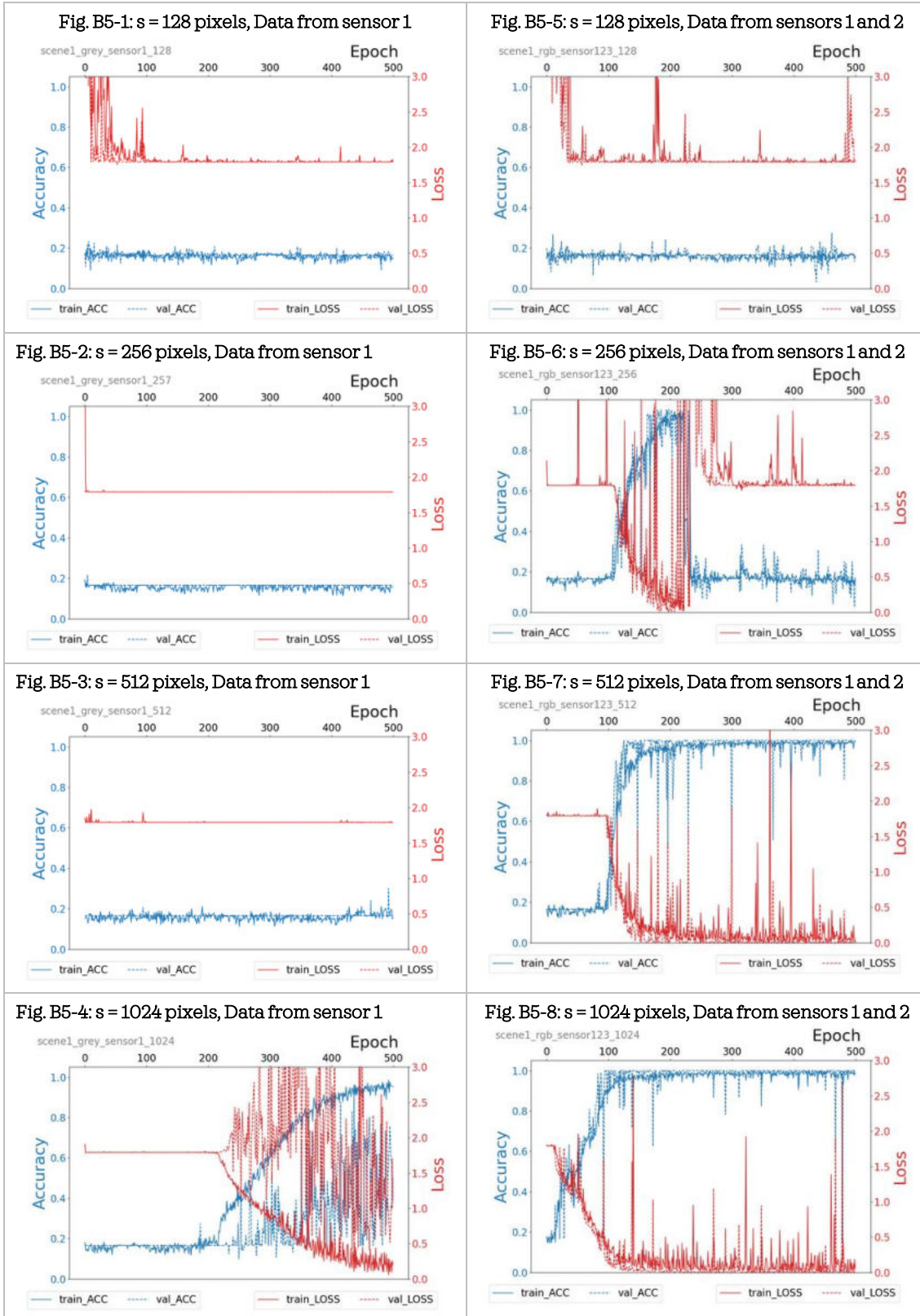
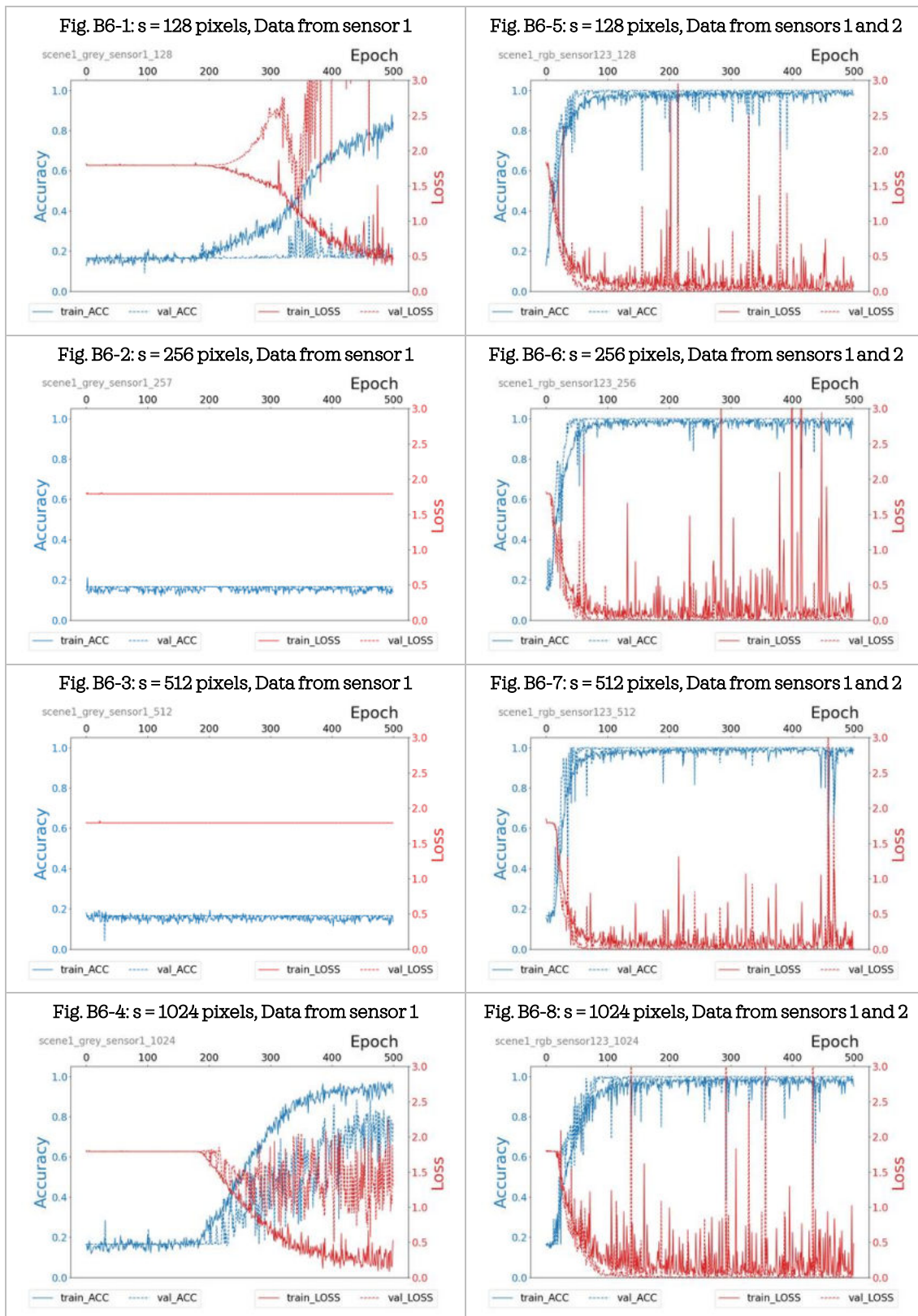


Fig. B6: Training Curves of Experimental Data with various sliding window lengths s from sensor locations 1 and 2 with following the hyperparameters:

Number of convolutional layers: 7	Number of recurrent layers: 4
Type of recurrent layers: GRU	Type of transition layers: GP
Dropout rate: 0.25	Input: either 1 channel or 3 channels



Literatures

- Chen X, Zhang B, Gao D. *Bearing Fault Diagnosis Base on Multi-scale CNN and LSTM Model*. J Intelligent Manufacturing. Vol. 32: 971-987 (2021).
- Cho KH, Bahdanau D, Bougares F, Schwenk H, Bengio Y. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. Proc. Conf on Empirical Methods in Natural Language Processing (EMNLP), Doha (2014).
- Dang HV, Tran-Ngoc H, Nguyen TV, Bui-Tien T, De Roeck G, Nguyen HX. *Data-Driven Structural Health Monitoring Using Feature Fusion and Hybrid Deep Learning*. IEEE Transactions on Automation Science and Engineering. Vol. 18(4): 2087-2103 (2021).
- Devlin J, Chang MW, Lee K, Toutanova K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Proc. 2019 Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), Minneapolis (2019).
- Flandrin P, Auger F, Chassande-Mottin E. *Time-Frequency Reassignment: From Principles to Algorithms*. Applications in Time-Frequency Signal Processing. Vol. 10: 179-204 (2002).
- Haile M, Hsu Cu, Bradley N, Chen J. *Recurrent Neural Networks for Identification of Acoustic Wave Reflections*. Proc. 9th European Workshop on Structural Health Monitoring (EWSHM), Manchester (2018).
- He Y, Liu Y, Shao S, Zhao X, Liu G, Liu L. *Application of CNN-LSTM in Gradual Changing Fault Diagnosis of Rod Pumping System*. Mathematical Problems in Engineering. Vol. 2019: 4203821 (2019).
- Hochreiter S, Schmidhuber J. *Long Short-Term Memory*. J Neural Computation 9(8): 1735-1780 (1997).
- Huo Z, Yang X, Zhang T, Wang Y, Zheng Y. *Multi-attention Parallel CNN-GRU Fault Diagnosis Method for Rolling Bearing*. Proc. 12th CAA Symposium on Fault Detection, Supervision, and Safety for Technical Processes (SAFEPROCESS), Chengdu (2021).
- Lee CY, Gallagher PW, Tu Z. *Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree*. Proc. 19th Intl Conf on Artificial Intelligence and Statistics (AISTATS), Cadiz (2016).
- Liu T, Bao J, Wang J, Zhang Y. *A Hybrid CNN-LSTM Algorithm for Online Defect Recognition of CO₂ Welding*. MDPI J Sensors. Vol. 18(12): 4369 (2018).
- Mousavi M, Gandomi A. *Deep Learning for Structural Health Monitoring under Environmental and Operational Variations*. SPIE Proc. Nondestructive Characterization and Monitoring of Advanced Materials, Aerospace, Civil Infrastructure, and Transportation XV: 115920H (2021).
- Niethammer M, Jacobs LJ, Qu J, Jarzynski J. *Time-Frequency Representation of Lamb Waves*. J Acoustical Society of America. Vol. 109(5): 1841-1847 (2001).
- Percival DB, Walden AT. *Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques*. Cambridge University Press, Cambridge (1993).
- Shenfield A, Howarth M. *A Novel Deep Learning Model for the Detection and Identification of Rolling Element-Bearing Faults*. MDPI J Sensors. Vol. 20(18): 5112 (2020).
- Shi X, Chen Z, Wang H, Yeung DY, Wong W, Woo W. *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting*. Proc. 28th Intl Conf on Neural Information Processing Systems (NIPS), Montreal (2015).
- Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. J Machine Learning Research, Vol. 15: 1929-1958 (2014).
- Thomson DJ. *Spectrum Estimation and Harmonic Analysis*. Proc. IEEE, 70(9): 1055-1096 (1982).
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. *Attention Is All You Need*. Proc. 30th Intl Conf on Neural Information Processing Systems (NIPS), Long Beach (2017).
- Xiao S, Qiao S, Chen H, Liu X, Li Y, Tang M, Hu H, Li G. *Data Prediction Model Based on LSTM Neural Network in Bridge Health Monitoring System*. IOP Conf Series Earth and Environmental Science 668(1):012068 (2021).
- Zhang X, Cong Y, Yuan Z, Zhang T, Bai X. *Early Fault Detection Method of Rolling Bearing Based on MCNN and GRU Network with an Attention Mechanism*. J Shock and Vibration. Vol. 2021: 6660243 (2021).
- Zhao R, Wang J, Yan R, Mao K. *Machine Health Monitoring with LSTM Networks*. Proc. 10th Intl Conf on Sensing Technology (ICST), Nanjing (2016).
- Zhao R, Yan R, Wang J, Mao K. *Learning to Monitor Machine Health with Convolutional Bi-Directional LSTM Networks*. MDPI J Sensors Vol. 17: 273 (2017)

8. Conclusion and Future Works

This chapter will briefly summarize the content of the PhD work in section 8.1, while section 8.2 will give the conclusions of the scientific work regarding the utilization of computational and artificial intelligence for applications in SHM domain. As a final word, the outlook and challenges of future AI-driven SHM system will be given in section 8.3.

8.1. Result Summary

To conclude the dissertation, I will provide a flashback into the results obtained during my research.

Chapter 4 of covers the assumption of the damage growth within the damage tolerance, and the used methodology to generate and capture Lamb wave signal within a Finite Element (FE) environment. The data processing includes color theory, image processing, and blob detection was also given in this chapter. While the image processing algorithm proposed is not a quantitative approach, this technique can be regarded as a first step for studying sensor placement in a more holistic way. The methodology proposed depicts a deterministic solution that can be partially used for solving continuous optimization in sensor placement problem. This solution, however, does not end there. As the conclusion of chapter 4, the following issues were still to be addressed:

1. *“Stochastic strategy for non-predictable damage location, especially for quasi-instantaneous but abrupt-like events such as bird strike, ...”*
2. *“Bimodal utilization of deterministic and stochastic of the network topology, with regard to the integration of both approaches ...”*

Therefore, a further approach was developed in chapter 5, in which I mentioned the important influencing physical parameters to determine the objective function that is to be minimized. Chapter 5 covers the stochastic strategy for sensor placement by using random search, greedy methods, genetic algorithm, simulated annealing, and swarm intelligence, and then elaborated the results from chapter 4 to create the bimodal topology for both predictable and non-predictable damage locations. The experimental validation was also given in this chapter which results in following conclusive statements:

- Global random search has logically the lowest search performance, while GA and PSO are on par, and they have the best performance. The greedy methodology and SA have a search performance which lies in between GA/PSO and global random search.
- The hybrid approach that combines blob detection algorithm and search metaheuristics is a good fit to address sensor positioning problem in active ultrasonic SHM in a limited albeit scalable manner, especially when the detection requirement is not too high.

- Unsurprisingly, the hybrid approach described in chapter 5 fulfills its duty as long as the purpose of (active)-SHM is to complement NDT and not functioning as its replacement.

In chapter 6, we saw the investigation on whether deep learning can be used to treat the Lamb wave signal. The chapter started with the main questions: “*How does the varying sensing location and the different sensing representations of a Lamb wave signal influence the deep learning training behavior?*” and “*What consequence can be drawn for the engineering application in SHM and why should this approach work?*”. To preliminarily answer these questions, the proposition of modelling SHM perception from a neuroscientific perspective has been made. Also included in chapter 6 is the concept and theoretical background such as converging the probability measure and generalization bound in deep learning.

The approach to represent the entity of the captured Lamb wave signal in the time-frequency domain are either hierarchical (which only consists of a randomly sampled spectrogram) or a conserved entity over time (which consists multiple layers of spectrograms joined in an image). These entities are trained using the parameters listed in section 6.3.4, while the training results from both entity representations are given in section 6.4. To validate the hypothesis, a simple A/B Testing is conducted in section 6.5. The conclusions were:

- An optimal sensor network topology will give the most desired training behavior due to a better response capture. However, as far as we can see from the results, when training the representation of the signals from multiple sensing locations in one single entity seemingly outweighs the previous assumption. Given the approximating capacity of a deep neural network as described before, we shall actually be not surprised.
- The training cost in terms of time would increase when the network must learn more parameters. Currently, limiting the neural network size is very likely to be the only feasible way to make deep learning for structural diagnostics scalable for industrial application.

Finally, chapter 6 also stated: “*In the future work, it would be of interest to investigate the scalability level of deep learning for SHM for a given data size, model parameters, and restriction on physical memory*”. Thus, this backlog was partially tackled in chapter 7 especially by taking sequential modelling into account. The methodology contains experimental setup, several simulation parameters, and data pre-processing which is basically the same as in chapter 6.

Based on the conducted literature review, we can safely conclude that there have been not many works in SHM-NDT domain that involve the hybrid CNN-recurrent models. As of 2021, the usage of convoluted sequential modelling for treatment of Lamb wave signal is practically limited and more importantly, there is currently no explanation on how these sequential modelling will behave. Section 7.5 divided the training results coming from 2 datasets: from experiment and from simulation. In each sub-section, we saw the comparison of the training

stability by modifying several hyperparameters. Based on the results, the conclusions of chapter 7 can be summarized as follows:

- Hybrid convoluted sequential modelling can cope with the spectrogram of a Lamb wave signal as input, and as such it opens up the potential not only for application in active SHM, but also for continuous monitoring such as acoustic emission (AE). In fact, it will be even more useful to embed the spectrotemporal capture via a convolutional kernel in the model rather than relying only on a pure time-domain recurrent model.
- The caveat is however: from the training results, it seems that training such a hybrid CNN-recurrent network is more difficult due to the unstable training behavior. Further, parallelization via GPU does not help because unlike convolutional operations that can be performed simultaneously, recurrent network operations are designed to follow each after.

An approach to involve more channels in the spectrogram to increase the training stability is also proposed. This would mean that if the spectrogram has more than 4- channels, some software engineering works such as deep learning library customization is needed.

8.2. Conclusion

To conclude this dissertation in general, let us revisit the research purpose stated in chapter 2 which covers the state of the art in NDT & SHM and the recent advances from the CS and ML community. This resulted into a high-level question that should be asked from the NDT & SHM community is:

What is the feasibility of incorporating computational and artificial intelligence as a design tool for an automated diagnostic within predictive maintenance - and if so, in what way?

As already pointed out, to (partially) solve certain domains with the help of AI, we shall break down the main problem into several manageable sub-problems, commonly known as the **divide and conquer** strategy that has been regularly used in many aspects of human civilization. The smaller sub-questions are:

1. *“The design complexity and parameter optimization, particularly on sensor placement methodologies for both deterministic and semi-stochastic approach according to what extent the structure is designed based on the premise that sensor network topology affects the damage detection capability and the overall SHM performance, i.e., which different sensor network topologies are needed to understand the trade-off between the strategies and if possible, to propose a compensation technique?”*

Two assumptions were made to answer question 1 and they naturally came from the structural design: A hazard in the structure can occur both in its weakest point due to design constraints or in another location that might be less predictable due

to the stochastic nature of the environment. Consequently, the sensor network topology problem is broken down into three possible approaches:

- For the localized problem such as hotspot damage detection, we should maximize the sensor response based on the isolated location.
 - For a statistically less predictable damage location such as hail impact, the logical choice would be to maximize the detection area in such way that any sudden impact will be detected on the first instance.
 - In realistic world, both kinds of hazards would likely to occur. I therefore proposed to compensate both by using the combination of topology that have hotspot & global damage detection capability.
2. *“Utilization of deep learning for SHM, i.e., an investigation as to whether deep learning can be used to treat the Lamb wave signal – and if so, whether it has a certain theoretical justification. What would be the pros and cons when using deep learning to treat Lamb wave signals and what would be the consequences for design and manufacturing of SHM system?”*

To answer question 2 an assumption that a set of a labeled data is available must be made. Within the supervised learning approach, the relevant foundation needed to understand the concept of learning is given so that we shall know that the high accuracy or classification performance of the deep neural network is not merely because of network overfitting, since there is a baseline for the theoretical guarantee as demonstrated in chapter 6. Consequently, by now we should be aware of what deep learning can do (and what it is not). An A/B Testing is needed to ensure that the network does not simply overfit the noise. To summarize:

- We can clearly see that the potential advantage of deep learning lies in its relatively high performance and currently as of 2022, it is still the best in-class for pattern recognition within supervised learning. Given the generalization guarantee via testing and validation, we know that at least the network is not simply memorizing the spectrogram.
 - On the negative side: Speaking from my personal experience working in industry and given the standard practice where many labelled data are hidden (i.e., not accessible in a public repository), we shall raise the doubt about the scalability of using the deep learning approach – not only for Lamb wave SHM, but also for other SHM-NDT related topics.
3. *“Eventually and worthy to be considered as a research direction as well: when combining the sub-problems to reconstruct the final solution: Given a certain sensor topology, what would the training behavior look like for different sensors and could a philosophical aspect from neuroscientific perspective be considered as well?”*

The answer to question 3 has been largely answered in chapters 6 and 7. Chapter discussed the training behavior from a single-layer and a 3-layered spectrogram for a given sensor topology. Also as mentioned earlier, this chapter highlighted the theoretical foundation which focused on the learning capacity and the

generalization gap via validation. Statistical A/B Testing was involved as well to ensure that the network does not overfit the input.

Chapter 7 focused on the training stability of sequential modelling with varying hyperparameters. The hyperparameter selection can be regarded either as a guideline or an anchor for further a hyperparameter search study. The results from chapter 6 and 7 can be summarized in the following two statements:

- The training behavior does not only depend on the network topology, but also the approach taken as to whether the network is fed with the whole signal representation such as ConvNet or whether it is modelled sequentially. The latter one has generally less stable training behavior and it takes more time to reach performance convergence, but it also depicts the only possible modelling approach for continuous input.
- Taking the two entity representations inspired by the neuroscientific perspective into account (i.e., hierarchical representations of entity and conserved entity over time), the organization of the spectrogram clearly influences the training behavior in such a way that a more complex input pattern yields in a faster training convergence by creating a very distinctive pattern in each probability class.

Finally, it is now the time to deliver a general conclusion for this PhD thesis. I stated in the beginning that there is nothing as a free lunch and as a staunch believer that there is really no such thing as a free lunch, not only in mathematics or computer science, but literally *everything*, I would like to point out both the strengths and weaknesses of employing AI and computational intelligence for designing SHM in predictive maintenance. The strength of using AI (especially deep learning and metaheuristics) as a design tool for SHM is:

1. From the results, it is obvious that deep learning and metaheuristics empowered the search of design parameters, thus it is safe to say that both techniques are suitable methods for designing sub-SHM system. Guided by domain knowledge, it will help many aircraft designers to accelerate their work. Note that we shall acknowledge that the domain knowledge here is not an option, but a requirement.
2. Currently it is known that aircraft operations are not all about maintenance since aircraft maintenance only accounts for up to 20% of the operating expenditure (OPEX). Thus, even with a high-efficiency functioning system (e.g., 80% man-hours reduction), the total cost saving (TCS) from OPEX will be only account to 16%. On the bright side, this gives an opening for the aircraft designer to adapt and/or (re)-design the future of maintenance framework to include AI & computational intelligence elements.

As always, with good news, there must be some bad news since otherwise our life might become too easy:

1. On the practicality aspect, some assumptions such as that data labels must be available is quite difficult to be fulfilled, if not absurd. While in-flight sensor data is typically available, they are mostly hidden and restricted to the OEM and getting an access to it another step of bureaucracy. As there is no to little public data available, the further development on using AI for SHM in aircraft maintenance will probably be hindered since it lacks (open) community support.
2. Generally, as for many other AI techniques in other domains, the design of SHM by involving deep learning and/or metaheuristics is not immune from the computational capacity problem, thus the question of scalability is not yet answered since it was not the part of my PhD research.

8.3. Research Outlook

Looking forward, an approach using AI might not be yet ripe enough at a scalable level for industrial demand, however the good news is that there are still many challenges that lie ahead, and this shall not prevent further research and development. In fact, such challenges open a new research direction and will further encourage more science and engineering knowledge exploration. Within the short- to medium-term from an academic perspective (i.e., within 5 – 10 years although this is typically regarded as medium- to long-term from industrial perspective), the outlook I can describe given the general conclusion in section 8.2 and considering the strength and weaknesses of employing ML/CI in SHM system design indicates 2 possible future work directions:

1. On the scientific side, I suggest that future research should include a semi-supervised approaches to generate more data labels from the known probability distribution. However, as it combines both supervised and unsupervised approach, another theoretical guarantee beyond the PAC framework and the generalization guarantees via validation that encompasses both learning methods, must be taken into consideration as well.
2. On the application & engineering side: we shall put a tremendous research effort on scalability that encompasses the interface between big data management, software engineering, and SHM system as Internet-of-Things (IoT) solution in order to prevent that the previous efforts stay only within labs and/or academia without proper industrial exploitation.



AI-Assisted Design & Optimization for Predictive Maintenance

Vincentius Ewald

