Synthetic Data for Smarter RUL Prediction

Deep Generative Models in Turbofan Analysis

D.C. Saadeldin

Technische Universiteit Delft

TU Delft

Synthetic Data for Smarter RUL Prediction

Deep Generative Models in Turbofan Analysis

by

D.C. Saadeldin

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday June 6th, 2025 at 9:00 AM.

Student number:	4163672	
Project duration:	January 1, 2023 – June 1, 2025	
Thesis committee:	Dr. ir. I. I. De Pater,	TU Delft, supervisor
	Dr. ir. J. Ellerbroek,	TU Delft, chair
	Dr. ir. O. A. Sharpans'kykh,	TU Delft, examiner

Preface

This thesis marks the final step of my Master's degree, which has been a challenging but rewarding experience. My initial exploration of generative models in predictive maintenance evolved into a comprehensive research project, testing my resolve, intellectual curiosity, and dedication. Beyond the specific technical knowledge gained in machine learning, this process underscored the importance of perseverance and independent problem-solving, from initial data immersion with the CMAPSS dataset to debugging complex GAN architectures.
This work critically compares CTGAN and TVAE models for synthetic data generation within the domain of Remaining Useful Life (RUL) prediction. The core objective was a dual evaluation using both statistical and predictive methodologies. This resulted in deep dives into technical issues requiring the development of skills and insights extending beyond classroom academic curricula.
I extend my profound gratitude to my supervisor, Ingeborg de Pater, whose expert guidance, patience, and incisive critical feedback were vital in shaping the clarity and quality of this research. Her input consistently directed the project toward completion.
My sincere thanks also go to my friends and family for their unwavering support and understanding, and for their extraordinary patience in listening to detailed explanations of Wasserstein Distance and Residual Box Plots.
Finally, I am indebted to the Delft University of Technology for providing essential resources and support. I also acknowledge the critical role of open-source communities and their contributors, whose tools like SDV, scikit-learn, CMAPSS, and PyTorch were fundamental to the successful execution of this project.
I hope that this thesis contributes meaningfully to the advancement of prognostic health management systems and encourages further investigation into the transformative potential of synthetic data in real-world engineering applications.

D.C. Saadeldin
Delft, May 2025

Contents

List of acronyms

G

GAN generative adversarial network. ix, xi, 35, 43, 49–51, 54, 55, 62, 63

GMM Gaussian mixture model. 46, 53–55

GPT generative pre-trained transformer. 35, 48, 49, 54

GRAN generative recurrent adversarial network. 51

H

HMM hidden Markov model. 46, 53, 55

I

img2img image to image generation. 54

InfoGAN information maximising generative adversarial network. 51

K

KNN k-nearest neighbour predictive model. 62

L

LAPGAN Laplacian pyramid of adversarial network. 50

LDA latent Dirichlet allocation. 47, 53, 55

LDM latent diffusion model. 54

LSTM long short-term memory. 54, 55

M

MCMC Markov chain Monte Carlo. 47

N

NICE non-linear independent components estimation. 53

P

PCA principal components analysis. 53

PHM prognostics and health management. 35, 37, 38, 43, 45, 53–55, 61, 62

R

RTF run-to-failure. 37, 39

RUL remaining useful life. 38, 39, 54

S

ST-GAN spatial transformer GAN. 54

T

T5 text-to-text transfer transformer. 54

TGAN temporal GAN. 55, 62

txt2img image to image generation. 54

V

VAE variational autoencoder. 35, 49, 54, 55, 62

VT vision transformer. 54

List of Figures

List of Tables

Introduction

This document serves as the full technical report for this research project, covering its methodology, results, and conclusions in detail. For this, a scientific article has been developed. Additionally, the antecedent literature study on generative models in prognostics is appended, providing theoretical background and research questions used in this thesis. Therefore, while the literature study establishes the conceptual basis and the article conveys key findings, this report offers the complete context, implementation intricacies, and comprehensive analysis. **Please note that the literature study is already graded and just serves as a stepping stone to the scientific article.**

I

Scientific Paper

Highlights

Synthetic Data for Smarter RUL Prediction: Deep Generative Models in Turbofan Analysis

D.C. Saadeldin

- Enhanced RUL Predictions: Evaluates the impact of deep generative models (DGMs) on Remaining Useful Life (RUL) prediction for turbofan engines.
- Synthetic Data Generation: Compares Conditional Tabular GANs (CTGAN) and Tabular Variational Autoencoders (TVAE) for generating realistic sensor data.
- Data Quality Assessment: Assesses the validity of synthetic datasets using statistical tests (Wasserstein distance, KS test), and visual distribution (t-SNE, KDE).
- Integration with Regression Models: Investigates the effect of synthetic data on RUL regression using Random Forest Regressors (RFR) and Convolutional Neural Networks (CNN).
- Scalability for Data-Limited Scenarios: Demonstrates the effectiveness of synthetic data augmentation when only limited training data is available.

Synthetic Data for Smarter RUL Prediction: Deep Generative Models in Turbofan Analysis

D.C. Saadeldin[a]

[a]*Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, Delft, 2629 HS, The Netherlands*

ARTICLE INFO

Keywords:
CTGAN
TVAE
RUL predictions
CMAPSS
Data augmentation
PHM

ABSTRACT

Scarce failure data often causes unreliable results when making predictions concerning Remaining Useful Life (RUL). This study explores the use of deep generative models (DGMs) for augmenting turbofan engine datasets by CMAPSS to improve these RUL predictions. By implementing Conditional Tabular GANs (CTGAN) and Tabular Variational Autoencoders (TVAE), synthetic data is generated and validated using statistical metrics such as Wasserstein distance and Kolmogorov-Smirnov tests. Then, these new datasets are used in several compositions of both real and synthetic data to train regressors and subsequently let them make RUL predictions. The regressors, such as Random Forest Regressors (RFR) and Convolutional Neural Networks (CNN), evaluate performance improvements through RMSE and MAE metrics. Results indicate that adding synthetic data improves prediction robustness, particularly when data is limited. This highlights the potential of DGMs for Prognostics and Health Management (PHM) applications.

1. Introduction

Maintenance is a crucial element in Prognostics Health Management [12]. As in other industries, specific parts are rigorously tested and checked for safety and endurance in aerospace engineering. This is mainly executed by the use of sensors, brake testing and other traditional maintenance techniques. However, data might not be available for exceptional situations due to the lack of real-time and real-size testing possibilities, which is often the case when testing aeroplanes towards the end of their service. This might be mitigated by extensive simulations and other novel techniques to establish the right time for repair and replacement, improving overall maintenance performance.

A major component of an aeroplane is its engines. They are measured by sensors that indicate pressure, temperature and other relevant variables to monitor. The data acquired can then be used for simulating the End of Life (EoL), Remaining Useful Life (RUL) and others. Here, the EoL states the moment an engine is no longer useful, and the RUL defines the time or cycles estimated until that particular EoL. However, specific data on engine properties during specific situations are sparse and often classified and unavailable to the public. Any insufficient data is a significant limitation in Prognostics and Health Management (PHM) for aviation [22]. This is especially applicable to the prediction of RUL for turbofan engines. It is costly and time-consuming to harvest extensive datasets from turbofan engines that require either high-computational simulation or elaborate testing of operational cycles until failure.

In real-world scenarios, most engines are repaired or retired well before complete failure. As a result, the degradation process of these engines in their last phase is often not documented comprehensively. Therefore, RUL prediction models can only rely on sparse failure data or on synthetic datasets, such as the CMAPSS dataset generated by simulations [7]. While these simulations can mimic degradation under specific conditions, they cannot capture the full complexity and variability of real-world scenarios. The lack of sufficient failure data impacts model generalizability and accuracy.

Instead of focusing on the quality of the simulation software in case of small available datasets, generative modelling can be a solution to enlarge insufficient or incomplete datasets. Especially deep generative modelling has evolved since the 2010s to create credible synthetic data [6] [13] [5]. There are multiple promising types of data augmentation [27] available, and the combination of unsupervised learning with expanding aeroplane engine datasets has undiscovered opportunities. Deep generative models (DGMs) could help enhance the sparse data into a dataset that will give easier prognoses for maintenance.

Therefore, the core aim of this study is to evaluate the utility of synthetic datasets generated by DGMs for turbofan engine degradation monitoring. Specifically, the study hypothesises that data generated by DGMs, such as generative adversarial networks (GANs) [6] [8] and variational autoencoders (VAEs) [2], can serve as credible substitutes or complements to original datasets, maintaining similar statistical and dynamic properties.

To address this, the research explores two primary questions: (1) Can synthetic datasets improve the robustness and accuracy of RUL predictions when integrated into traditional modelling pipelines? (2) How closely do the synthetic datasets mimic real-world degradation data in terms of variability and structure? These questions guide the overarching hypothesis that leveraging DGMs can enhance prediction performance while mitigating the challenges of limited datasets in PHM applications.

Many generative models can be studied, and today, a couple of deep generative models, as well. Two DGMs are considered specifically: the conditional tabular generative adversarial network (CTGAN) [30] & the tabular variational autoencoder (TVAE) [29]. The former uses a generator and discriminator to create synthetic data and is promising when trained on large datasets with complex patterns, such as rare categories or unstable data infractions. However, training a GAN is a tedious process and requires both a large dataset as well as properly set parameters.

The latter uses encoding and decoding for synthetic generation and performs quite reasonably with smaller datasets, and thus does not need as much training data as a GAN. With increased complexity and enlargement of the input data, a VAE becomes less flexible, though.

This study aims to enhance Remaining Useful Life (RUL) predictions by addressing insufficient data challenges through synthetic data generation. The approach begins with the generation of synthetic data by deep generative models (DGMs), using commonly used datasets for turbofan engine diagnostics. Subsequently, the generated data is validated through statistical and performance metrics, ensuring its similarity to the original datasets.

This synthetic data is then incorporated into regression models to improve RUL forecasting accuracy [3], using a Random Forest Regressor (RFR) [11] and a convolutional neural network (CNN) [20]. The performance of the models is then evaluated using standard performance metrics such as RMSE and MAE to quantify improvements introduced by synthetic data augmentation [25].

The case study treats different setups of DGMs and regressors fed several dataset compositions. The CTGAN-RFR combination showed the most promising results in the scenario of only 10% of the training data available, making the predictions more robust and improving on overall performance in terms of RMSE by 8%.

In Section 2, the methodology consists of an overview of the full process followed by an extensive description of set-up, choices/assumptions and calculations. Then, in Section 3, the consequent results are published and critiqued on after which conclusions are drawn in Section 4.

2. Methodology

During the research, several considerations were taken into account, from the choice of datasets to the comparison of RUL predictions. In the order of the process, the test setups and associated methodology will be elaborated on, together with all choices we made and their explanations.

2.1. Overview

The proposed approach will use the NASA Commercial Modular Aero-Propulsion System Simulation (CMAPSS) [7], a commonly used degradation dataset source. For RUL prediction of turbofan engines, the CMAPSS FD001 dataset [24] consists of training data with extensive sensor readings, test data and corresponding RUL values. Therefore, this dataset can lay a foundation for evaluating the impact of synthetic data on RUL prediction in comparison to the original RUL.

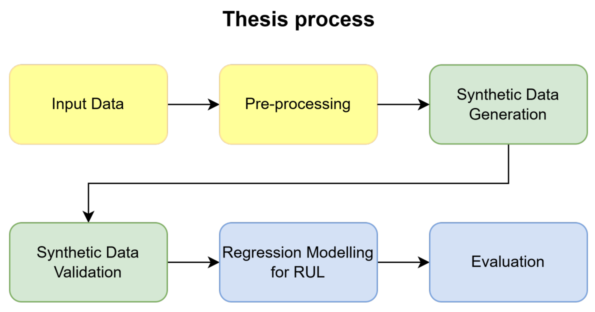

Thesis process
Input Data
Pre-processing
Synthetic Data Generation
Synthetic Data Validation
Regression Modelling for RUL
Evaluation

Figure 1: Steps total process

In Subsection 2.3, two deep generative models will be applied to generate synthetic data, a Conditional Tabular GAN (CTGAN) [30] and a Tabular Variational Autoencoder (TVAE) [29]. Their synthetic datasets will be validated using the statistical measures as the KS-test [14] and Wasserstein distance [28], as well as visual inspection by t-SNE distribution [19] and KDE plots [23].

Then, in Subsection 2.4, the Random Forest Regressor (RFR) [11] and Convolutional Neural Network (CNN) [20] will be trained on different subset combinations of synthetic and training data to predict RUL values. Their output will then be evaluated by comparing the predictions with the original RUL and describing its performance in terms of Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), showing whether synthetic data generated by deep generative models contributes to prediction accuracy, in Subsection 3.4.

2.2. Data pre-processing

In CMAPSS [7], four Fault Datasets concerning turbofan engines are explicitly available. The first one, FD001, has a single operating condition and one fault mode [7]. Therefore, this relatively simple dataset is usable as a baseline for testing generational modelling and subsequently predicting RUL values. FD002 and FD004 both have multiple operating conditions embedded [7] and are thus equipped to investigate more intricate fault progressions. The extra complexity can affect the similarity of synthetic data generation and obscure the impact of accurate RUL prediction. For clarity, Figure 2 shows a standard turbofan overview and its standard sensors that are available in CMAPSS, elaborated on in Table 1.

FD003 is set in a single operating condition and has the same dimensions and sensor readings as FD001 [7]. It can therefore be substituted conveniently to validate the model. However, FD003 has two fault modes, whereas FD001 only has one [7]. Although it can serve as a logical extension, it introduces more complex failure scenarios. We chose to ensure an analysis focused solely on the impact of synthetic data and model performance, and hence the FD001 dataset is best suited [7].

The FD001 data consists of 100 engines, with each engine described as multiple cycles until End-of-Life (EoL).

Table 1
Operational settings and sensor measurements in the CMAPSS FD001 dataset [7]

Type	Abbreviation	Description
Op1	Altitude	Operational Setting 1
Op2	Mach number	Operational Setting 2
Op3	Throttle resolver angle (TRA)	Operational Setting 3
Sensor 1	T2	Total temperature at fan inlet
Sensor 2	T24	Total temperature at LPC outlet
Sensor 3	T30	Total temperature at HPC outlet
Sensor 4	T50	Total temperature at LPT outlet
Sensor 5	P2	Pressure at fan inlet
Sensor 6	P15	Total pressure in bypass-duct
Sensor 7	P30	Total pressure at HPC outlet
Sensor 8	Nf	Physical fan speed
Sensor 9	Nc	Physical core speed
Sensor 10	epr	Engine pressure ratio (P50/P2)
Sensor 11	Ps30	Static pressure at HPC outlet
Sensor 12	phi	Ratio of fuel flow to Ps30
Sensor 13	NRf	Corrected fan speed
Sensor 14	NRc	Corrected core speed
Sensor 15	BPR	Bypass ratio
Sensor 16	farB	Burner fuel-air ratio
Sensor 17	htBleed	Bleed enthalpy
Sensor 18	Nf_dmd	Demanded fan speed
Sensor 19	PCNfR_dmd	Demanded corrected fan speed
Sensor 20	W31	HPT coolant bleed
Sensor 21	W32	LPT coolant bleed

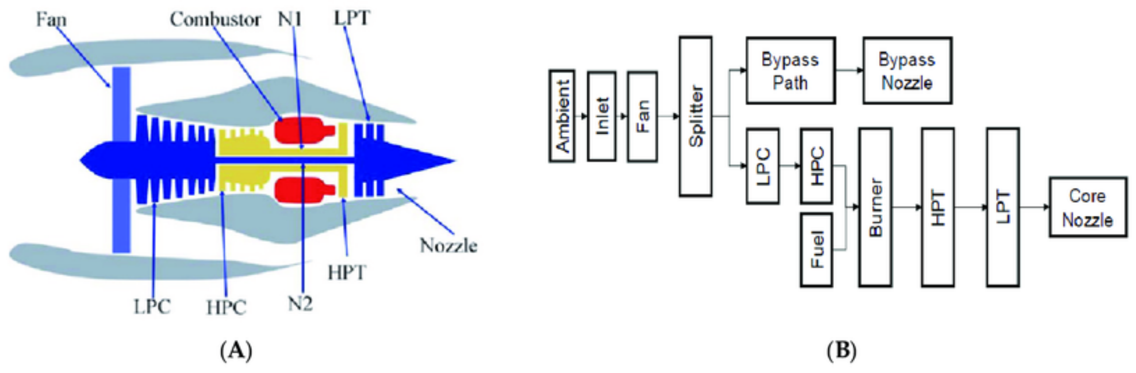


Figure 2: Overview turbofan engine [7] [16]

The number of cycles per engine differs, as would be expected in real life. A small representation of the training dataset is shown in Table 3.

Apart from the engine IDs and their respective cycles (a cycle per row), three operational settings (altitude, Mach number, throttle resolver angle) are available. They represent the operating conditions of the engine and are continuous variables.

Next, 21 columns represent sensors in and on the engine. They measure, for example, internal temperature, vibrations, flow rates, torque, etc. The dimensions of the FD001 dataset are listed in Table 2.

Normalisation of datasets before using them as training for DGMs can be advantageous, but the TVAE and CTGAN do not require this [29, 30]. Also, the elimination of constant columns is not needed for the selected DGMs, and therefore both types of pre-processing are omitted [29, 30].

The simulated dataset provided by NASA consists of smooth sensor data and does not have any missing values. Noisy data is also not assumed, which makes the handling of the data more convenient. Therefore, the raw data is assumed to be clean.

Table 2
Dataset dimensions

CMAPSS	Datasets	rows	columns
FD001	Training	20631	26
	Test	13096	26
	RUL test	100	1

Table 3
Layout of FD001: Cycle data $x_i^p = [a_i^p, b_i^p, \dots, z_i^p]$ for engine p.

Engine (p)	**Op1 (a_i^p)**	..	**Sensor2 (e_i^p)**	..	**RUL (z_i^p)**
1	-0.0007	..	641.82	..	191
1	0.0019	..	642.15	..	190
..	..	..	..	..	..
1	0.0000	..	643.34	..	1
1	0.0009	..	643.54	..	0
2	-0.0018	..	641.89	..	301
..	..	..	..	..	..
100	-0.0032	..	643.85	..	0

Where:

- x_i^p: Data for a sample from engine p.
- $a_i^p, b_i^p, \dots, z_i^p$: Features representing operational settings and sensor measurements.
- p: Engine index ($p \in \{1, 2, \dots, 100\}$).

2.2.1. *Set-up RUL as target variable*

While the test dataset has a given RUL per engine, the training dataset does not [7]. However, it is assumed that after the last cycle of each engine, that particular engine reaches EoL [7]. Therefore, one cycle before the last cycle, the RUL of this engine is one. Thus, at the first cycle, the RUL is as high as the maximum (and last) cycle of that specific engine [7]. This RUL per cycle row is as such:

$$\texttt{RUL}_i = \max(\texttt{Cycle}) - \texttt{Cycle}_i, \quad (1)$$

Where:

- $\texttt{RUL}_i$ denotes the remaining useful life for the engine at time step i,
- $\texttt{Cycle}_i$ is the current cycle number at time step i,
- $\max(\texttt{Cycle})$ represents the maximum cycle number for the specific engine instance, corresponding to its end of life (EoL).

The calculated RULs are then appended to the training dataset as the 27th column and will serve as target variables in a later stage, as seen in Table 3.

2.3. Type of DGMs

The two chosen deep generative models have their specific characteristics in how they work, handle datasets and put out synthetic data. Both their differences as well as their similarities will be discussed. Also, distinct features that affect the process are treated.

2.3.1. *CTGAN*

The first deep generative model that will be used to generate synthetic data in this research is a generative adversarial network. More specifically, a GAN that is specialised in tabular data such as the CMAPSS datasets, a so-called Conditional Tabular Adversarial Network (CTGAN).

A GAN [8] is built of two neural networks that compete as adversaries to one another in a zero-sum game. The first neural network is a generator, whereas the second neural network functions as a discriminator. The generator creates synthetic data to be as indistinguishable from real data as possible. The discriminator will try to differentiate real and generated samples. Through training, the generator improves its abilities to imitate the real data distribution. Meanwhile, the discriminator becomes more proficient in spotting fake from real. This game continues throughout training and afterwards enables a GAN to generate a realistic dataset that is purely synthetic. A schematic overview of the CTGAN infrastructure is shown in Figure 3.

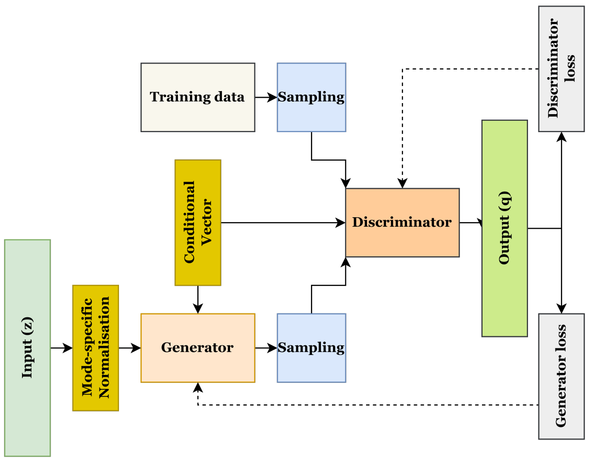


Figure 3: CTGAN layout

The generator takes a noise vector z from a latent space and puts it into the data space. This noise vector is normally randomly sampled from a simple distribution such as a Gaussian distribution ($\texttt{N(0,1)}$). This will be the input for the generator. The architecture of the generator starts with fully connected layers. Here, the input z is multiplied by learned weights and biases. These layers expand its dimensionality and enable it to capture intricate patterns in the real data distribution. After each layer non-linear ReLU activations ($\texttt{ReLU}(x) = \max(0, x)$) are implemented. These activations introduce non-linearity, an elementary part of learning from complex data. ReLU allows the model to learn hierarchical features and prevents gradient vanishing, effective characteristics for a GAN generator. Then, normalisation is applied per batch to stabilise the training between the fully connected layers. Next, to portray an output in the data space, the so-called output layer uses linear activation or Tanh for continuous data such as dataset FD001, and a softmax layer for other discrete data. To learn, the generator utilises the feedback from the discriminator to update its weighting to improve its output samples.

The discriminator acts as a binary classifier. As input, it is fed real samples from training data and synthetic samples by the generator during the training process and has to decide whether the sample is labelled as real or fake. The output is thus a single binary value of 0 or 1, where 0 means "fake" and 1 means "real". Normally, a `Sigmoid` activation function represents the produced binary probabilities in the output layer, and is defined as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2}$$

Where:

- $\sigma(x)$ is the output of the sigmoid function, representing a probability value between 0 and 1.
- x is the input to the sigmoid function, often the discriminator's raw output (logit).
- The exponential term e^{-x} ensures the output smoothly maps any real-valued input to the range $(0, 1)$.

The sigmoid function's S-shape flattens the discriminator output into a valid probability score. This allows the discriminator to learn from the feedback of having classified a sample correctly, and it updates its weighting to improve the classification accuracy accordingly.

A generator and discriminator neural network work with loss functions to describe the adversarial training process and update iteratively. The generator tries to maximise the probability of the discriminator labelling synthetic samples as real. A high $D(G(\mathbf{z}))$ in a negative logarithm function gives the ultimate goal of minimising the generator loss. The generator loss function is given by Equation 3:

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z}\sim p_z}\left[\log\left(D(G(\mathbf{z}))\right)\right] \tag{3}$$

Where:

- $\mathcal{L}_G$ is the generator loss.
- $\mathbf{z}$ refers to a noise vector sampled from the latent space distribution p_z (e.g., Gaussian or uniform).
- $G(\mathbf{z})$ indicates the synthetic data generated by the generator from the noise vector $\mathbf{z}$.
- $D(G(\mathbf{z}))$ is the discriminator's prediction for the synthetic data, representing the probability that it classifies the synthetic data as real.

The discriminator is trained to correctly classify real data as real and synthetic data as fake. Its loss function, therefore, consists of two terms. The first term represents the discriminator's goal of maximising the probability of real samples as real. The second term consists of the probability of fake samples labelled as real, where the goal is to minimise, being subtracted from 1. The discriminator loss function is given by Equation 4:

$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log\left(D(\mathbf{x})\right)\right] - \mathbb{E}_{\mathbf{z}\sim p_z}\left[\log\left(1 - D(G(\mathbf{z}))\right)\right] \tag{4}$$

Where:

- $\mathcal{L}_D$ refers to the discriminator loss.
- $\mathbf{x}$ represents real data samples drawn from the true data distribution p_{data}.
- $D(\mathbf{x})$ indicates the discriminator's prediction for the real data, representing the probability that it classifies the real data as real.
- $1 - D(G(\mathbf{z}))$ is the discriminator's prediction for the synthetic data, representing the probability that it classifies the synthetic data as fake.

A CTGAN excels over a standard GAN when tabular data such as CMAPSS is involved. It distinguishes itself through several key techniques [30]. For one, it samples conditionally to guarantee boundaries of the distributions of categorical features. It uses conditional vectors and constraints to impose this and prevents mode collapse by targeting specific categories during generation, shown in Figure 3 by the highlighted box Conditional Vector. It also applies these vectors to the discriminator to impose the same conditions on both real and generated samples.

$$G(z, c) \rightarrow \tilde{x} \tag{5}$$

$$D(x, c) \rightarrow p_{\text{real}} \tag{6}$$

Where:

- z is a latent noise vector sampled from $\mathcal{N}(0, I)$.
- c is a one-hot encoded vector representing a randomly chosen categorical feature.
- $G(z, c)$ is the generator function that outputs a synthetic sample $\tilde{x}$.
- $D(x, c)$ is the discriminator function that predicts whether x is real, conditioned on c.
- $p_{\texttt{real}}$ is the probability assigned by the discriminator that x is real.

However, in the case of continuous-only data such as FD001, the conditional mechanism becomes redundant. CTGAN then defaults to mode-specific normalisation, still shown in Figure 3, where input is normalised per GMM mode to align with underlying data distributions. A mode in this context refers to a peak or dense region in the feature distribution, captured by a Gaussian component in the mixture. Each input is standardised relative to the parameters of its most likely mode, which helps enhance feature representation even without discrete variables.

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \sigma_k^2) \tag{7}$$

$$\gamma_k(x) = \frac{\pi_k \mathcal{N}(x \mid \mu_k, \sigma_k^2)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x \mid \mu_j, \sigma_j^2)} \tag{8}$$

$$\tilde{x} = \frac{x - \mu_{k^*}}{\sigma_{k^*}}, \quad \text{where } k^* = \arg\max_k \gamma_k(x) \tag{9}$$

Where:

- $p(x)$ represents the probability density function of the feature x under a Gaussian Mixture Model (GMM) with K components.
- π_k denotes the mixture weight for Gaussian k.
- $\mathcal{N}(x \mid \mu_k, \sigma_k^2)$ represents a Gaussian distribution with mean μ_k and variance σ_k^2.
- $\gamma_k(x)$ is the posterior probability that x belongs to mode k.
- $\tilde{x}$ is the normalised feature value, standardised based on its most probable mode k^*.

While these enhancements remain useful, the absence of discrete columns limits the CTGAN to its continuous processing capabilities as a tabular DGM. In such cases, a TVAE might offer more stable results, as it is natively optimised for continuous data.

2.3.2. TVAE

The second deep generative model that is used for data generation is the variational autoencoder. It also incorporates a latent space to learn and uses probabilistic mapping to generate synthetic samples. The tabular variational autoencoder distinguishes itself for being optimised for tabular datasets that may consist of both continuous and categorical data, and its layout is displayed in Figure 4.

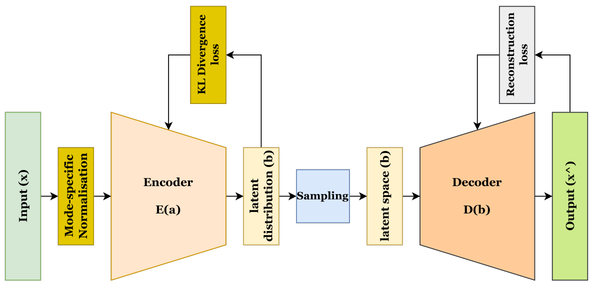


Figure 4: TVAE Layout

A VAE [13] has an encoder and decoder to perform this. The encoder maps high-dimensional input data through fully connected layers and makes use of ReLU activations, just as the GAN architecture. The main difference here is the type of input. Where a GAN starts with a random noise vector z, the VAE takes a real data sample x. The output into the latent space differs, as well. Now two outputs are created, the mean $\mu(\mathbf{x})$ and the log-variance. The log-variance is shown in Equation 10, of which the standard deviation $\sigma(\mathbf{x})$ can be derived for later use.

$$\sigma^2(x) = \exp(\text{log\_var}(x)) \tag{10}$$

Where:

- $\sigma^2(x)$ is the variance of the latent distribution output by the encoder.
- $\text{log\_var}(x)$ indicates the log-variance output by the encoder.
- The exponential function, $\exp(\cdot)$, is used to convert the log-variance to the variance.

In the latent space, sampling is introduced where a reparametrisation trick is used to enable backpropagation. A latent variable $\mathbf{z}$ is expressed in Equation 11 by stochastic nodes using the output of the encoder.

$$\mathbf{z} = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \odot \epsilon \tag{11}$$

Where:

- $\mathbf{z}$ is a latent variable sample for the decoder input.
- $\mu(\mathbf{x})$ represents the mean vector output by the encoder for input x.
- $\sigma(\mathbf{x})$ is the standard deviation vector output by the encoder.
- $\odot$ denotes the element-wise (Hadamard) product, where corresponding elements in the vectors are multiplied.
- ϵ refers to random noise sampled from a standard Gaussian distribution.

The decoder then reconstructs the input data by transforming the latent variables back. The latent variable $\mathbf{z}$ thus serves as input, is mapped by applying fully connected layers while using the aforementioned often used `Tanh` or linear activation for continuous features and the `softmax` function for categorical outputs. Then, the reconstructed data is applied to find the probability distribution of the data. Mathematically, this is represented by Equation 12 with $\hat{\mathbf{x}}$ as reconstructed output.

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}\left(\hat{\mathbf{x}}; \boldsymbol{\mu}_{\text{decoder}}(\mathbf{z}), \boldsymbol{\sigma}^2_{\text{decoder}}(\mathbf{z})\mathbf{I}\right) \tag{12}$$

Where:

- $p(\mathbf{x}|\mathbf{z})$ is the probability distribution of the data point x given the latent variable z.
- $\hat{\mathbf{x}}$ refers to a reconstructed data point sampled from the predicted Gaussian distribution.
- $\boldsymbol{\mu}_{\text{decoder}}(\mathbf{z})$ represents the mean vector of the Gaussian distribution, produced by the decoder, given the latent variable z.
- $\boldsymbol{\sigma}^2_{\text{decoder}}(\mathbf{z})$ is the variance vector output by the decoder network, controlling the spread of the distribution for each dimension of x.

- **I** indicates the identity matrix, which represents independent dimensions of x.

Next to the VAE's architecture, the learning process is addressed. Similar to GANs, a loss function guides learning by iteratively minimising a combination of reconstruction loss and a regularisation term, as shown in Equation 13:

$$\mathcal{L}_{VAE} = \mathcal{L}_{\text{reconstruction}} + \mathcal{L}_{\text{KL}} \tag{13}$$

Where:

- $\mathcal{L}_{VAE}$ is the total loss function for the VAE.
- $\mathcal{L}_{\text{reconstruction}}$ measures the negative log-likelihood, evaluating how well the decoder reconstructs input data from latent variable $\mathbf{z}$.
- $\mathcal{L}_{\text{KL}}$ refers to the Kullback-Leibler divergence regulariser, aligning the learned posterior distribution with a standard normal prior $\mathcal{N}(0, I)$.

For continuous data, the reconstruction loss typically takes the form of the negative log-likelihood of the data conditioned on the latent variable $\mathbf{z}$, as shown in Equation 14:

$$\mathcal{L}_{\text{reconstruction}} = -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] \tag{14}$$

The KL divergence component for a Gaussian posterior $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu, \sigma^2)$ and a standard normal prior $p(\mathbf{z}) = \mathcal{N}(0, I)$ is given by Equation 15:

$$\mathcal{L}_{\text{KL}} = \frac{1}{2}\sum_j \left(\mu_j^2 + \sigma_j^2 - \log(\sigma_j^2) - 1\right) \tag{15}$$

Where:

- $-\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}$ represents the expectation taken over the learned posterior distribution $q(\mathbf{z}|\mathbf{x})$.
- $p(\mathbf{x}|\mathbf{z})$ is the likelihood of reconstructing the input data $\mathbf{x}$ given the latent variable $\mathbf{z}$.
- $q(\mathbf{z}|\mathbf{x})$ denotes the learned posterior distribution over latent variable $\mathbf{z}$, conditioned on input $\mathbf{x}$.
- $\mathbf{z}$ refers to a latent variable sampled from the learned posterior distribution.
- μ_j, σ_j^2 represent the mean and variance for dimension j of the latent variable distribution.
- $\log(\sigma_j^2)$ indicates the log-variance of the latent variable distribution.

The reconstruction loss solely measures the performance of the encoder-decoder combination, comparing the $\hat{\mathbf{x}}$ to its original $\mathbf{x}$ while the KL divergence measures the latent space to smooth the process. The total loss is then implemented by the aforementioned optimisers to update the weighted parameters of the VAE, equal to a GAN, to improve the model each iteration. For a VAE, most common issues coincide with those of a GAN, such as mode collapse and overfitting.

The specific properties that set a TVAE apart from a traditional VAE are similar to those of a CTGAN versus a GAN. It involves the ability to handle mixed data types, the highlighted KL divergence loss, and the same mode-specific normalisation technique as previously described. This optimises the VAE for tabular datasets.

Another difference in TVAE versus VAE is the way it models categorical data. Instead of directly using a Gaussian output for categorical features, TVAE models them using a log-softmax transformation, which represents categorical variables through a probability distribution over possible categories. However, since during this research only continuous data is handled, this TVAE feature will not be utilised.

For continuous data, TVAE uses a Gaussian likelihood assumption similar to standard VAEs, and incorporates the same mode-specific normalisation to improve feature representation:

$$\tilde{x} = \frac{x - \mu_{k^*}}{\sigma_{k^*}}, \quad \text{with } k^* = \arg\max_k \gamma_k(x) \tag{16}$$

where $\gamma_k(x)$ represents the posterior probability of x belonging to mode k, defined as:

$$\gamma_k(x) = \frac{\pi_k \mathcal{N}(x \mid \mu_k, \sigma_k^2)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x \mid \mu_j, \sigma_j^2)} \tag{17}$$

The updated continuous loss for the TVAE is now:

$$\mathcal{L}\text{continuous} = -\mathbb{E}q(\mathbf{z}|\mathbf{x})\left[\log p_{\text{cont}}(\mathbf{x}|\mathbf{z})\right] \tag{18}$$

Where:

- the equation is essentially the same as the general VAE Equation 14 apart from the specification $\log p_{\text{cont}}(\mathbf{x}|\mathbf{z})$

The reconstruction loss would now theoretically consist of both the continuous and categorical components, but the categorical part is in this case zero:

$$\mathcal{L}_{\text{reconstruction}} = \mathcal{L}_{\text{continuous}} \tag{19}$$

And the total TVAE loss thus extends the VAE loss by incorporating these different reconstruction objectives:

$$\mathcal{L}_{TVAE} = \mathcal{L}_{\text{reconstruction}} + \mathcal{L}_{\text{KL}} \tag{20}$$

With these adaptations, TVAE ensures that any categorical variables can be properly modelled, continuous variables respect their underlying distributions, and the latent space remains regularised, making it more suitable for tabular data than a standard VAE.

Even though CMAPSS is fully continuous, TVAE can still outperform a standard VAE because of its mode-specific normalisation, better feature handling, and improved training dynamics. Advantages may lie in capturing different operational conditions of the engines more effectively in the latent space.

2.4. Type of Regressors

The regression predicts a target output (RUL) based on feature input, using test data to generate RUL predictions. Comparing these predictions to the actual RULs of the test data helps evaluate the efficiency of the regressors. By comparing different combinations of DGM types, subset compositions, and regressors on specific datasets, the models' efficiency in capturing patterns and accurately predicting RUL can be assessed.

After the subsets are generated and analysed, regressors will be implemented for prediction modelling. The subsets are divided into feature and target data (all columns apart from RUL, and the RUL column, respectively) to be used as training datasets for the regressors to learn their correlations and subsequently set their internal parameters. Then, after training, a regressor will generate a target RUL prediction when it has been given a feature input of test data. Its predictions are meant to be as minimally different to the target data before the regressor is trained sufficiently.

Two types of regressors are chosen for this research. Firstly, the Random Forest Regressor (RFR) [11] [4], which is an ensemble learning method. It uses multiple decision trees for predictions, which helps robustness and decreases the risk of overfitting. The RFR is a common and versatile regressor used throughout maintenance projects and can handle non-linear data such as turbofan engine datasets.

Secondly, a convolutional neural network (CNN) [20] is selected. A CNN performs well with time series and can process large datasets. It uses convolutional layers to detect trends in the feature input and find patterns.

Together, they complement one another. The RFR is reliable for tabular non-temporal data analysis, while a CNN is better equipped to detect temporal and spatial patterns. They are both adaptable to different data distributions, such as the subsets partly generated by the chosen DGMs. Also, both regression methods are widely used throughout PHM prediction and CMAPSS research in particular, making cross-referencing easier.

2.4.1. RFR

The Random Forest Regressor utilises multiple decision trees to make predictions. It builds this collection of decision trees during the training phase. Every decision tree is built using a random sample of the training data (bootstrap data) and, after selecting a random set of feature data at each branch, determines the best split. The randomiser guarantees uncorrelated multiple trees and mitigates overfitting. A simple diagram of a decision tree including splits is shown in Figure 5.

In the RFR, each individual decision tree makes its prediction and subsequently all predicted target values are averaged in Equation 21.

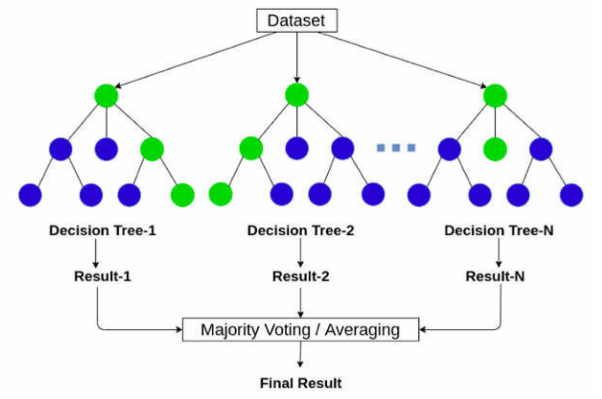


Figure 5: Decision tree diagram [9]

$$\hat{y}_i = \frac{1}{M}\sum_{m=1}^{M}\hat{y}^{(m)}(x_i) \tag{21}$$

Where:

- M: The number of trees in the Random Forest.
- $\hat{y}^{(m)}(x_i)$: The prediction of the m-th tree for the input vector x_i.
- x_i: The feature vector for the i-th engine-cycle pair, i.e., $x_i = [a_i, b_i, \dots, z_i]$, where:
  - i: The index of the sample (a specific engine and cycle).
  - $a_i, b_i, \dots, z_i$: Features representing operational settings and sensor measurements for that sample.
- $\hat{y}_i$: The final predicted output, i.e., the estimated Remaining Useful Life (RUL) for the i-th sample.

The Mean Squared Error (MSE) is then commonly used as a split criterion. It inserts the earlier calculated means, and its goal is to minimise the variance of the target variable after splitting:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 \tag{22}$$

Where:

- N is the number of samples in the node.
- y_i represents the true target values.
- $\hat{y}$ indicates the mean prediction for the node.

For a node t being split into left t_L and right t_R child nodes, the loss is calculated as follows using the MSE:

$$\text{Loss} = \frac{|t_L|}{|t|}\cdot \text{MSE}(t_L) + \frac{|t_R|}{|t|}\cdot \text{MSE}(t_R) \tag{23}$$

Where:

- N is the number of samples in the node.
- y_i represents the true target values.
- $\hat{y}$ indicates the mean prediction for the node.

Each decision tree is set to minimise this loss by updating its parameters, improving its performance and therefore the overall performance of the RFR.

2.4.2. CNN

Convolutional Neural Networks (CNNs) [17] are deep learning models designed to capture spatial patterns in data. They are particularly well-suited for feature extraction in temporal sequences, such as time-series data. CNNs use convolutional layers to apply filters (kernels) that slide over the input data, identifying local patterns and creating feature maps. This enables CNNs to automatically learn hierarchical features, from low-level patterns to high-level abstractions, without requiring manual feature engineering. The input to output layer architecture is visualised in Figure 6.

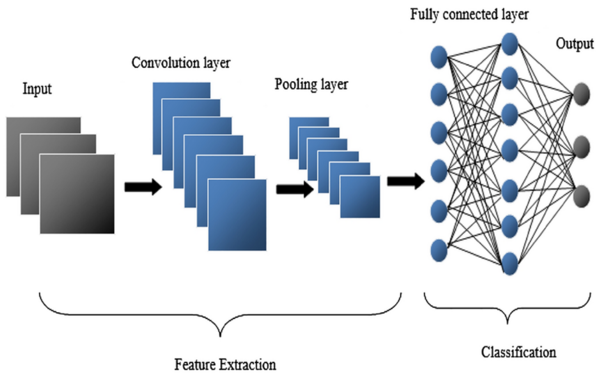


Figure 6: CNN layer architecture [15]

CNNs are well-suited for RUL prediction because they can effectively analyse sensor data over several cycles to extract complex patterns. Instead of requiring hand-designed features, the CNN model learns to recognise the most relevant trends and relationships between operational conditions and sensor measurements for RUL prediction.

Before feeding data into the CNN, the input is preprocessed. Each engine's time-series data is normalised to ensure uniform scaling across features, and sequences of cycles are reshaped into fixed-length input windows. For instance, data from the last n cycles of an engine can be used to predict the RUL at the current cycle, capturing temporal dependencies.

The prediction of the CNN is expressed as follows in Equation 24:

$$\hat{y} = f(x; \theta) \tag{24}$$

Where:

- $\hat{y}$: The predicted Remaining Useful Life (RUL).
- $f(x;\theta)$: The CNN function mapping input x to output $\hat{y}$, parameterised by θ.
- x: The input data representing sensor measurements and operational settings for a given engine and cycle, $x_{ip} = [a_{ip}, b_{ip}, \dots, z_{ip}]$, with:
  - p: The engine index.
  - i: The cycle index.
  - $a_{ip}, b_{ip}, \dots, z_{ip}$: Features representing operational settings and sensor measurements for engine p at cycle i.
- θ: The learnable parameters of the CNN, including filter weights and biases.

The CNN was trained using the Mean Squared Error (MSE) loss function, just as the RFR. The training process again involves minimising this loss by updating the parameters θ using backpropagation. This allows the CNN to iteratively improve its ability to predict RUL based on the patterns learned from the training data.

The architecture of the CNN implemented in this thesis consists of two convolutional layers, each followed by `ReLU` activation and max-pooling. The kernel size was set to 3, with a stride of 1, ensuring fine-grained pattern detection. The first layer uses 32 filters, and the second layer increases to 64 filters, allowing for the extraction of increasingly complex features. Dropout is applied after the convolutional layers to prevent overfitting, and batch normalisation ensures stable training. The final feature maps are flattened and passed through fully connected layers to predict the RUL.

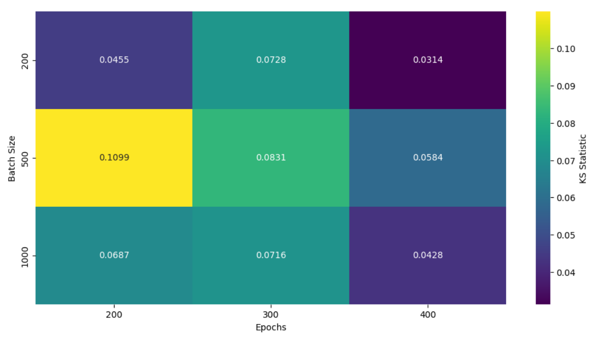
3. Results

The generational data created by the DGMs from the Synthetic Data Vault (SDV) [26] will be analysed, including the setup used from the Methodology Section 2. Then, the next step of data regression is addressed, and ultimately, the final results will be disclosed and discussed.

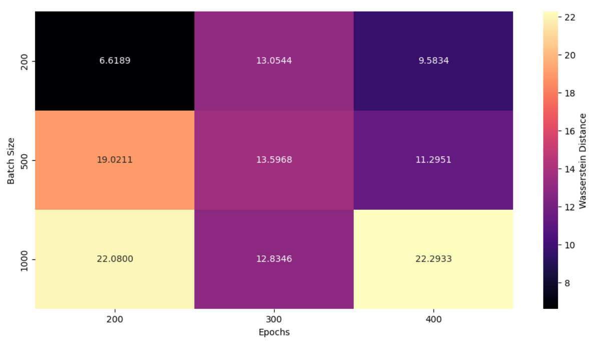
3.1. Hyperparameter optimisation DGMs

Hyperparameters help counteract generating issues by providing fitting properties for a GAN to learn. Often used hyperparameters are batch sizes, number of epochs, learning rates, choice of optimiser, among others. The batch size is an adjustable hyperparameter and represents the number of samples, real or fake, to be fed to the model before an iterative feedback update is performed. A small batch size counters overfitting but can make the model unstable, whereas a larger size causes the opposite. Another hyperparameter is the set number of epochs a GAN will have. It signifies how many times the total training dataset is used as input for training. If the number of epochs is low, the model might not have had enough time to learn the data correlations enough to generate realistic data. However, if the number of epochs is very high, overfitting is an often-seen issue.

To establish the best batch size and epoch combination, a generic dataset of 80% training data with 100% synthetic data is chosen to be evaluated in different epoch-to-batch size combinations. The training data is fed to the TVAE



(a) CTGAN Grid search (KS test)



(b) TVAE Grid search (Wasserstein)

Figure 7: Grid search results for CTGAN by KS-test and TVAE by Wasserstein hyperparameter tuning.

and CTGAN with these hyperparameter settings, after which they will produce a dataset with a 100% volume of the original training dataset. Since the ultimate target variable is the RUL, that column will be optimised using the Wasserstein and KS metric.

The Wasserstein distance calculates the cost required to transform one distribution into another. This provides insight into the global differences, such as mean and variance shifts, and largely ignores small fluctuations in density.

The KS test measures the maximum difference between the cumulative distribution functions (CDFs) of training and synthetic data, making it sensitive to early and local differences. Combined with other metrics, it provides a more complete view of how well the synthetic data replicates the training data's properties, capturing both local and global deviations.

The Wasserstein distance $W(p,q)$ and KS-test $D_{n,m}$ are calculated using the following equations to determine their values between a given train dataset compared to the synthetic dataset created by either the CTGAN or the TVAE, with the Wasserstein distance denoted in Equation 25 and the KS test in Equation 26.

$$W(p,q) = \inf_{\gamma \in \Pi(p,q)} \int |x-y| \, \mathrm{d}\gamma(x,y) \tag{25}$$

Where:

- $W(p,q)$: The Wasserstein distance between the two probability distributions $p(x)$ and $q(y)$.
- inf: The infimum (greatest lower bound) over all valid transport plans.
- γ: A joint distribution (also called a transport plan) over (x,y) that defines how much "mass" is transported from x (in p) to y (in q).
- $\Pi(p,q)$: The set of all joint distributions with marginals $p(x)$ and $q(y)$.
- $|x-y|$: The ground cost, i.e., the "effort" or "distance" to move a unit of mass from x to y.
- $\int |x-y| \, \mathrm{d}\gamma(x,y)$: The total transport cost under plan γ.

$$D_{n,m} = \sup_x \left| F_n(x) - G_m(x) \right| \tag{26}$$

Where:

- $D_{n,m}$: The Kolmogorov–Smirnov (KS) statistic, which measures the maximum absolute difference between the two empirical cumulative distribution functions (CDFs).
- $F_n(x)$: The empirical CDF of the first sample, based on n observations (e.g., training data).
- $G_m(x)$: The empirical CDF of the second sample, based on m observations (e.g., synthetic or test data).
- $\sup_x$: The supremum (least upper bound), which in this context is equivalent to the maximum value of the absolute difference over all values of x.

Both metrics will be utilised again for confidence intervals in Subsubsection 3.2.3.

In search of the lowest possible output, a grid search of batch size [250, 500, 1000, 2000] with a range of [10, 100, 200, 300, 400, 500] epochs is conducted. The CTGAN outcome by KS test is shown in7a. For the TVAE, the same grid search is executed, and the Wasserstein test results are shown in Figure 7b.

For both the TVAE and the CTGAN, the increase of batch size improved the score in the grid search marginally. A lower number of epochs shows a small improvement in score, but not enough to allow for a much higher computational time. Therefore, the combination of 300 epochs with a batch size of 500 was set to be the default for both the CTGAN as well as the TVAE. During the remainder of the research, these two key hyperparameters were kept at this value. Comparing the two DGMs to one another, their KS-test grid search produces comparable scores. When applying Wasserstein, the CTGAN can score lower. This is important to keep in mind during the data validation and subsequently the regression.

Another important hyperparameter feature is the choice of optimiser. An optimiser controls learning rates, gradient

Table 4
Hyperparameter settings for CTGAN and TVAE models.

Hyperparameter	CTGAN	TVAE
`Epochs`	300	300
`Batch size`	500	500
`Embedding dim`	128	128
`Generator dim`	(256, 256)	(256, 256)
`Discriminator dim`	(256, 256)	–
`Discriminator step`	1	–
`Pac`	10	–
`L2 scale`	–	1e-5
`Learning rate`	2e-4	2e-4
`Loss function`	`cross entropy`	`MSE`
`Optimizer`	Adam	Adam
`log frequency`	True	–

clipping, weight decay, and momentum from previous gradients. Two common optimisers are the Stochastic Gradient Descent (SGD) and the Adam (Adaptive Moment Estimation). The Adam optimiser is the most widely used one in GANs for both the generator as well as the discriminator and will be utilised in both the TVAE and CTGAN during the research. Other optional hyperparameters, such as activation functions and dropout rates, are deduced by the used DGMs from the SDV [26] when it is calculating its metadata or kept as default, as the most important key hyperparameters did not significantly change the scores in Figure 7. They can be found in Table 4.

Both the TVAE and the CTGAN can be subjected to imposing constraints on the output. The target feature RUL cannot be negative, and thus a constraint of a minimal output of 0 seems logical, as might be a maximal output constraint when inspecting the maximum RULs in the training dataset. However, the constraints interfere with the fully connected layers of both DGMs and increase the computational time drastically. Also, a well-configured DGM should generate credible outputs in the first place. Therefore, a Min-Max RUL constraint was abandoned.

3.2. Analysis of generated synthetic data

When a new set of synthetic data is generated, it will be compared to the original available data that was used by the DGMs. The composition of the subset (part training data + part synthetic data) is noted, together with its main characteristics such as the mean, standard deviation, min, max, and quartiles per column.

Also, the Kolmogorov-Smirnov test (KS-test) and the Wasserstein distance are performed to provide statistical comparisons between the training and synthetic datasets' distributions.

The newly generated synthetic data are analysed and compared to one another using several metrics. The mean, standard deviation, minima, maxima and quartiles of the target RUL column per subset will visually give more insight into whether its properties are realistic.

3.2.1. Descriptive validation

The newly generated data can now be compared on multiple bases. Where a visual and statistical validation will be discussed in Subsubsection 3.2.2 and 3.2.3 respectively, first, a table with the most important measures per variable is created. This includes a direct comparison between the mean and standard deviation from the real training dataset, the synthetic dataset generated by the CTGAN and finally the synthetic data from the TVAE, displayed in Table 5. The full table, including the variance, can be found in Appendix A.

Three variables are singled out and printed in bold. First Operational setting 1, the best performing variable in terms of standard deviation. Secondly, Sensor 14, a variable that seemingly changes in value randomly or at least, without a discernible pattern by the DGMs. Therefore, Sensor 14 performs the least when comparing the standard deviation among all variables. Lastly, RUL is chosen to take a closer look at because this variable will become key during the regression and the following results.

It appears the DGMs can quite clearly mimic the real data, with not only the mean close to the original but also a comparable standard deviation. However, the following subsections give more insight into whether the underlying data is distributed as precisely as the data the DGMs were trained on. RUL is more complicated to imitate because the sequential decrease of RUL in the real data is not apparent in the synthetic data. This means that a RUL value is connected to the specific values on that same row from other values and is not vertically dependent. Therefore, when one RUL column is directly compared against the original, the standard deviation might be off.

3.2.2. Visual validation

The synthetic data can be compared visually by plotting the generative data versus the original training data. However, because the datasets consist of many variables with separate columns, a distinction is created between first comparing one variable at a time per KDE plot and subsequently combining the separate variables into a multivariate distribution for broader analysis.

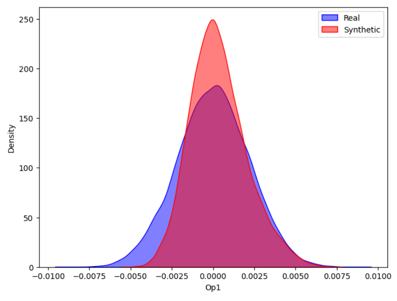
KDE plots

For specific visual validation, a couple of variables are selected to compare the original training data to the synthetic data. Both the worst as well as the best-performing variables in terms of standard deviation were chosen. Apart from the constant columns, the best is Operational setting 1, and the least performing is found to be Sensor 14. These variables represent the altitude and NR_c, respectively (see Table 1). Together with the target variable RUL, the synthetic data of the CTGAN is shown with the training data in Figure 8a, Figure 8b and Figure 8c. These plots show the similarity of the data in density to their value.

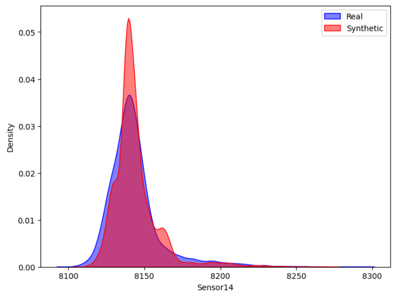
The second series of plots follows the same chosen variables but now for the TVAE in Figure 8d, Figure 8e and Figure 8f. Although the synthetic data is covering a similar range of values, it does not replicate the full distribution of

Table 5
Descriptive Validation of TVAE and CTGAN

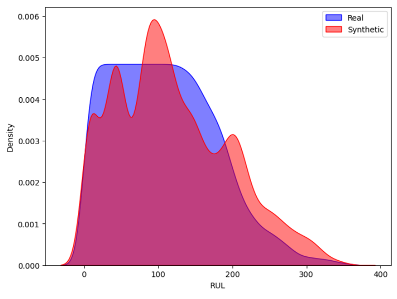
Variable	Real Data		TVAE (Synthetic Data)		CTGAN (Synthetic Data)	
	Mean	Std	Mean	Std	Mean	Std
Cycle	108.81	68.88	55.72	43.44	100.30	62.52
Op1	**9.0e-6**	**3.37e-3**	**-3.0e-6**	**2.07e-3**	**-4.8e-5**	**1.48e-3**
Op2	2.0e-6	2.59e-4	-2.2e-5	2.87e-4	2.3e-4	3.50e-4
Op3	100.00	0.00	100.00	0.00	100.00	0.00
…	…	…	…	…	…	…
Sensor14	**8143.75**	**19.08**	**8137.18**	**7.95**	**8145.38**	**16.95**
…	…	…	…	…	…	…
RUL	**107.81**	**68.88**	**84.73**	**44.45**	**102.25**	**65.78**



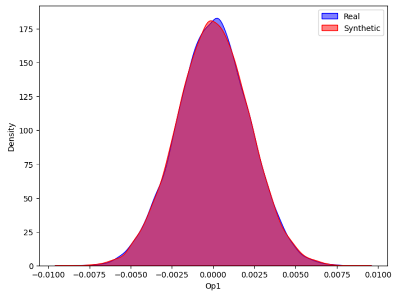
(a) Op1 (CTGAN)



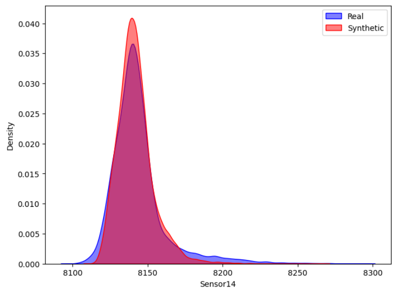
(b) Sensor 14 (CTGAN)



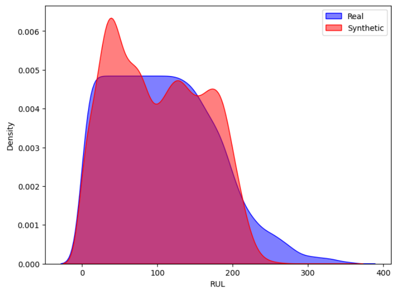
(c) RUL (CTGAN)



(d) Op1 (TVAE)



(e) Sensor 14 (TVAE)



(f) RUL (TVAE)

Figure 8: KDE distributions for CTGAN (top) and TVAE (bottom) across different features.

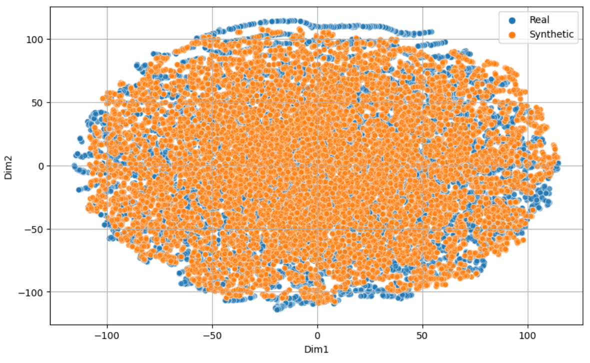
the original training data at all times. Especially Figure 8a, 8b, and both KDE plots for RUL show a coarse distribution of synthetic data versus their real data counterparts. In the regression phase, this discrepancy will be addressed on how it affects the ability to predict the final target variable RUL, as these KDE plots only show one column's variable at once, whereas during regression whole of the data serves as input simultaneously. Nonetheless, the KDE plots give an intuitive means of preliminary comparison to estimate whether a synthetic dataset has similar characteristics.

t-SNE distribution

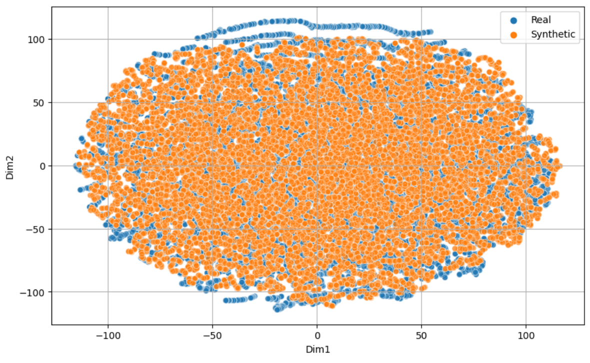
To visualise multivariate data, t-Distributed Stochastic Neighbour Embedding is chosen. t-SNE preserves similarities between data points and is used as a two-dimensional representation of the high-dimensional feature space of both the real as well as synthetic datasets. The axes (Dim1 and Dim2) have no physical meaning, but the points placed close together mean that they are likely similar in the feature space. Therefore, a t-SNE plot illustrates how well the approximation of the synthetic data distribution compares to the training data. Ideally, all points would overlap, which indicates alignment and successful capture of dataset patterns. Perplexity of the t-SNE distribution adjusts the method to focus on local or global tendencies.

A low perplexity of 5 was chosen to better highlight the local structure of the data, revealing close alignment between the real and synthetic distributions, as shown in Figure 9a and 9b. These figures show emphasis on the generative model's ability to replicate small-scale patterns observed in the original data.

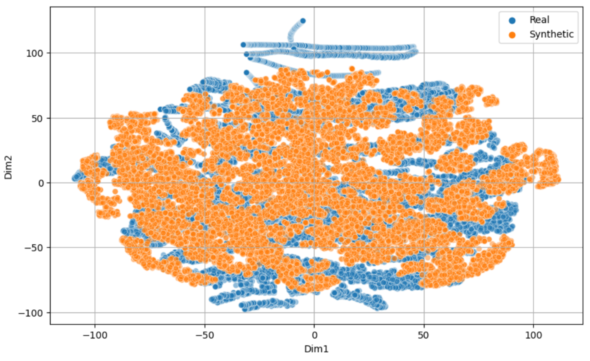
Also, a higher perplexity value of 30 is shown in Figure 9c and 9d to provide a broader overview. These figures show a less distinct overlap, indicating that focusing on



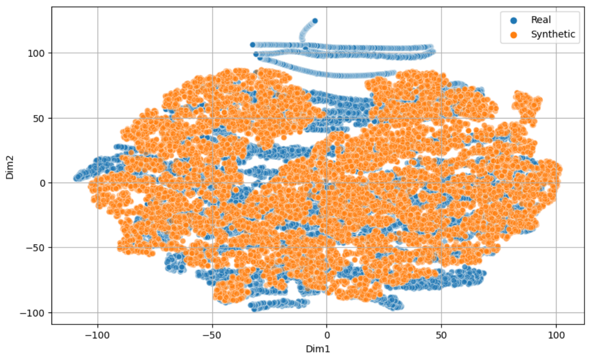
(a) t-SNE distribution CTGAN with perplexity 5



(b) t-SNE distribution TVAE with perplexity 5

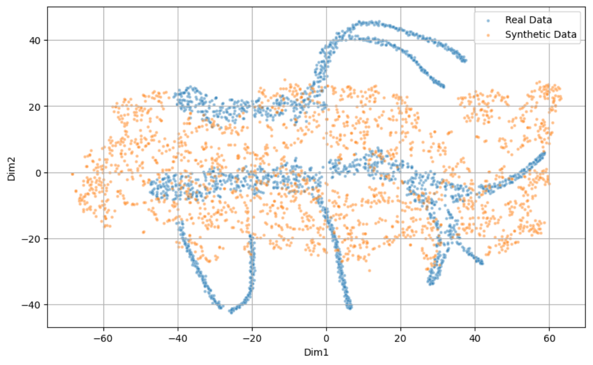


(c) t-SNE distribution CTGAN with perplexity 30

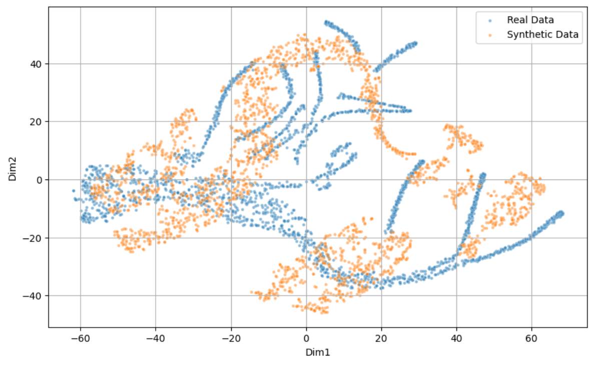


(d) t-SNE distribution TVAE with perplexity 30

Figure 9: Overview of T-SNE distributions



(a) t-SNE distribution 10% CTGAN with perplexity 5



(b) t-SNE distribution 10% TVAE with perplexity 5

Figure 10: t-SNE visualizations comparing CTGAN and TVAE distributions with 10% data and perplexity 5.

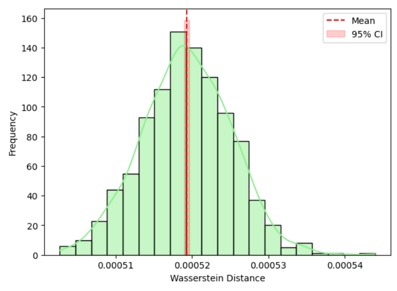
local structures offers a clearer assessment of the model's performance compared to a global approach.

To illustrate the deterioration of the dataset's properties, a t-SNE distribution is also conducted on a 10 % fraction of the training data and its subsequent synthetic data to simulate a scenario with scarce training data available. In Figure 10a and Figure 10b, the t-SNE distribution plots of CTGAN and TVAE generation are shown, respectively. The synthetic data points do not overlap the real data points and do not show an expected circle through Dim1 over Dim2. This indicates that with such a low input of real data to train a CTGAN or TVAE, the models are not able to grasp the correlations of said training data to implement during their data generation.

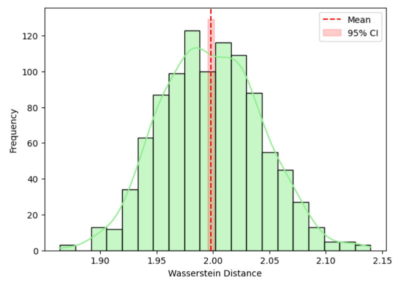
3.2.3. Statistical validation

When statistically validating synthetic data against the training data, the Kolmogorov-Smirnov (KS) test and the Wasserstein distance serve as complementary metrics to find similarities in distribution per variable chosen in Subsubsection 3.2.2.

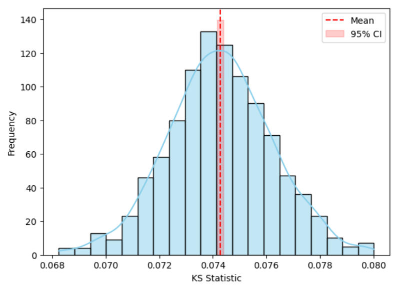
Apart from the grid searches, both metrics are useful for investigating confidence intervals. The metrics are specified



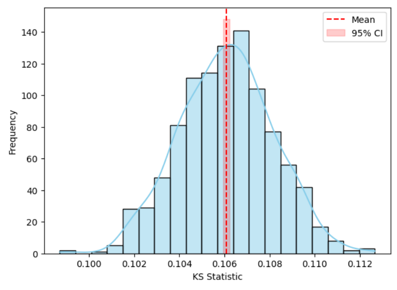
(a) Op1 (altitude) Wasserstein CTGAN



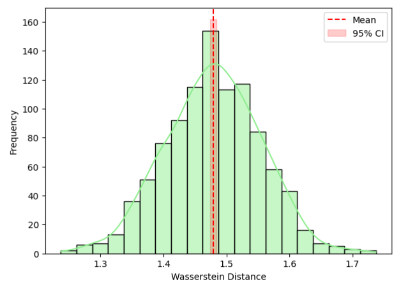
(b) Sensor 14 (NR$_c$) Wasserstein CTGAN



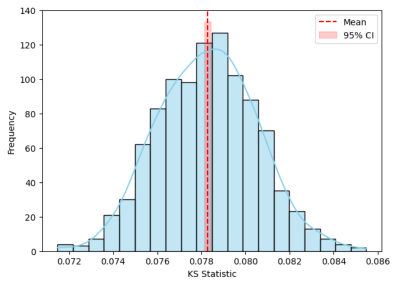
(c) RUL KS CTGAN



(d) Op1 (altitude) KS CTGAN



(e) Sensor 14 (NR$_c$) Wasserstein TVAE



(f) RUL KS TVAE

Figure 11: Confidence interval plots for CTGAN and TVAE using Wasserstein and KS metrics for Op1 (altitude), Sensor 14 (NR$_c$), and RUL

on validating one target variable at a time, and therefore, just as with the KDE plots, the Wasserstein distance and KS test show one specific variable per plot. For consistency, again, the best and worst performing variable according to its standard deviation, together with the ultimate target variable to depict, respectively, Operational Setting 1, Sensor 14 (NR_c) and RUL. To visualise the results of the Wasserstein distance and KS test calculation, confidence intervals are implemented. Both metrics are run a 1000 times and their values are grouped and shown in a histogram of 30 bins, shown in Figure 11a up until Figure 11f.

All plots show a desirable pyramid of bins where most of the 1000 values are situated in the middle bins and only a small portion on the outer edges. The values of KS are low across all shown variables, expressing a well-performing local distribution. For example, the bins in Figure 11a follow the same compartmentalisation as the bins in Figure 11d, derived from the same synthetic CTGAN dataset. Comparing the Sensor 14 results in Figure 11b and Figure 11e, the TVAE seems to have created data more resembling the train data, compliant with the results from the KDE plots in Subsubsection 3.2.2. However, when performing the KS test for RUL in Figure 11c and Figure 11f, the values are similarly low.

In Table 6 and Table 7, a selection of the most important and plotted CI values can be found in values of Wasserstein and KS, respectively. The values of the Wasserstein distance for RUL differ compared to the other variables. This is to be expected, as RUL ranges between 0 to 400+, and is shuffled during the generational process. This gives a much greater offset than a sensor with only a small range between minimum and maximum. Full tables for all variables after 1000 runs are included in the appendix A, one table per metric as well (Table 13 & Table 12).

With the generated synthetic data evaluated, we move on to utilise it to predict and analyse RUL. First, the regressors will have to be optimised, trained, and tested in Subsection 3.3. The input datasets will also be adjusted by establishing subsets and trimming.

Table 6
Wasserstein Distance for CTGAN and TVAE

Variable	CTGAN			TVAE		
	Mean	CI Lower	CI Upper	Mean	CI Lower	CI Upper
Cycle	6.99	6.97	7.00	6.93	6.90	6.96
Op1	**7.8e-4**	**7.8e-4**	**7.8e-4**	**1.2e-4**	**1.2e-4**	**1.2e-4**
Op2	3.7e-5	3.7e-5	3.7e-5	4.2e-5	4.2e-5	4.2e-5
...	...	...	...	...	...	...
Sensor14	**2.07**	**2.06**	**2.07**	**2.42**	**2.42**	**2.43**
...	...	...	...	...	...	...
RUL	**10.61**	**10.59**	**10.62**	**17.00**	**16.97**	**17.03**

Table 7
KS Test for CTGAN and TVAE

Variable	CTGAN			TVAE		
	Mean	CI Lower	CI Upper	Mean	CI Lower	CI Upper
Cycle	0.055	0.055	0.055	0.039	0.039	0.039
Op1	**0.190**	**0.190**	**0.190**	**0.025**	**0.025**	**0.025**
Op2	0.049	0.049	0.049	0.075	0.075	0.075
...	...	...	...	...	...	...
Sensor14	**0.044**	**0.044**	**0.044**	**0.068**	**0.067**	**0.068**
...	...	...	...	...	...	...
RUL	**0.066**	**0.066**	**0.066**	**0.089**	**0.088**	**0.089**

3.3. Hyperparameter optimisation Regressors

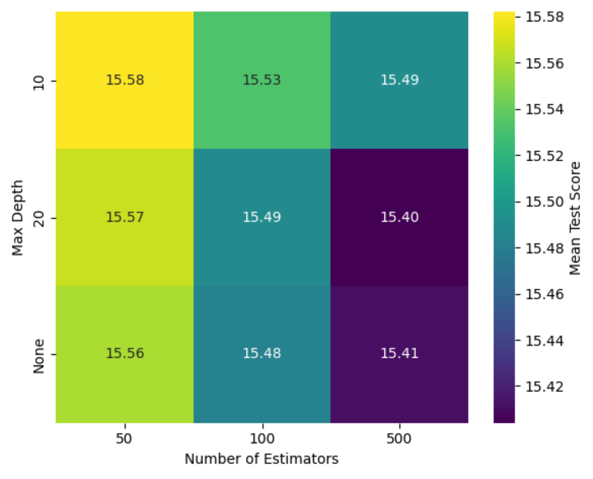
Most hyperparameters are set to default, as this research is focused on the analysis of RUL results after incorporating synthetic data. While adjusting the regressor to optimal settings, consistency in the results is most important. The random state of the used RFR is explicitly set to 42 (an arbitrary number, commonly used), and only the following hyperparameters were selected for a grid search due to their significant impact on model behaviour.

The first key hyperparameter is `min samples split`, which sets the minimum number of samples required to split an internal node. Lower values allow for deeper trees with smaller nodes, potentially capturing complex patterns but increasing the risk of overfitting. Higher values result in larger node sizes, improving generalisation but preventing overly complex splits. The grid search explores a range of values to balance model complexity and performance. The `Number of Estimators` hyperparameter determines the number of decision trees in the forest. Generally, increasing the number of trees improves model performance, but also increases computational cost. A range of values is tested to find an optimal trade-off between accuracy and efficiency. Finally, `Max Depth` controls the maximum depth of individual trees. Deeper trees can model more intricate patterns but are more prone to overfitting. Shallower trees may fail to capture essential relationships in the data. The grid search evaluates different depths to identify the ideal model capacity by fitting the regressor on the same training data. The regressor's predicted RUL output is then compared to the actual RUL data using RMSE. The results of this grid search are visualised in Figure 12a and Figure 12b, corresponding to `min samples split` values of 2 and 5, respectively.

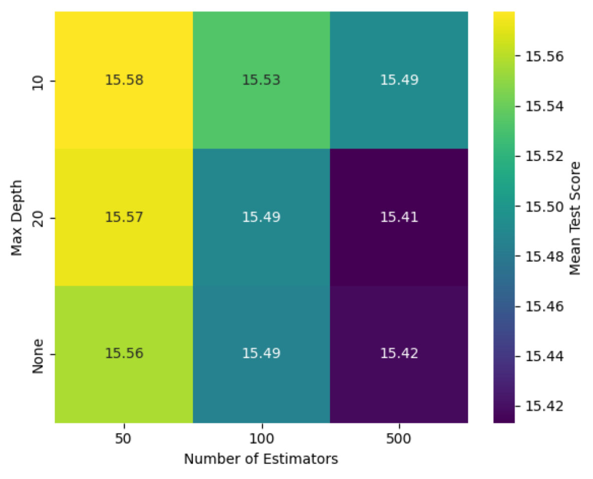
The CNN's architecture also requires careful tuning of hyperparameters that control its ability to extract meaningful features from data. For this grid search, the following hyperparameters were chosen:

The `Kernel Size` determines the size of the convolutional filter. Smaller kernels (e.g., 3) capture finer details, while larger kernels (e.g., 5) provide a broader view, which is useful for detecting larger patterns. Both options were explored to assess their impact on feature extraction. Also, the `Batch Size` is investigated. It controls the number of training samples processed before updating the model's weights. Smaller batch sizes provide more frequent updates, potentially improving convergence but increasing variance. Larger batch sizes offer more stable updates but may require longer training times. Lastly, the `Number of Filters` defines the number of filters in the convolutional layers. More filters increase the model's capacity to extract complex features but add to computational cost. The grid search again explores a range of values to find the optimal balance. Just as with the RFR grid search, the RMSE metric is chosen as the output value. The resulting heatmaps, shown in Figure 12c and Figure 12d, illustrate the model performance for `Kernel Size` values of 3 and 5, respectively.

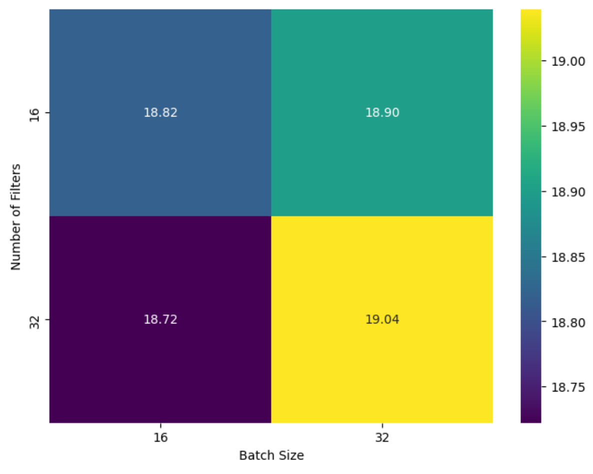
After evaluating the values in the RFR grid search, it can be concluded that adjusting these particular key hyperparameters did not change the RMSE output significantly. When utilising the Random Forest Regressor, all values are around 15.50. For the Convolutional Neural Network, the largest offset between values was less than 0.55, also not



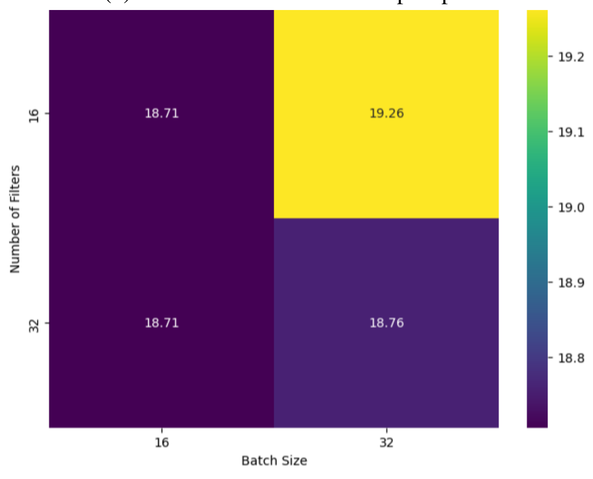
(a) RFR Grid Search with sample split 2



(b) RFR Grid Search with sample split 5



(c) CNN Grid Search with kernel size 3



(d) CNN Grid Search with kernel size 5

Figure 12: Grid Search RMSE results for RFR and CNN hyperparameters.

a significant indicator to choose a specific regressor setup. Therefore, whilst trading off computational time, as well, we chose the hyperparameter values split=2, estimators=100 and depth=20 for RFR. For the CNN, hyperparameter values were set at kernel=3, batch=32 and filter=32 for CNN. A more substantial overview of the key hyperparameters for both the RFR as well as the CNN can be found in Table 8, where the most important default hyperparameters are included, too. While not extensively tuned, these defaults have demonstrated good performance across various RUL and time series prediction studies (e.g. [31] [1]).

3.3.1. K-fold validation

To ensure the robustness and reliability of the regression models, a 5-fold cross-validation procedure was applied. This method divides the training data into five equally sized folds, which are each used once as a validation set, while the remaining four folds are used for training. Therefore, this process repeats five times. The primary objective of this validation was to assess the regressors' performance on their training data before evaluating them on synthetic data. By using cross-validation, the model's stability across different data splits can be confirmed.

The 5-fold cross-validation process was performed for both the Random Forest Regressor (RFR) and the Convolutional Neural Network (CNN). For each model, the average RMSE and MAE scores were recorded across all five folds to provide a comprehensive performance assessment.

The results of the RFR model evaluated on synthetic data generated by CTGAN are shown in Figure 13, while the corresponding 5-fold validation for TVAE-generated data presented comparable results. These results illustrate that the RFR model achieved consistent performance across all folds, indicating stable and reliable behaviour. A similar trend was observed for the CNN model, further supporting the effectiveness of the trained regressors.

Table 8
Hyperparameter settings for RFR and CNN models.

Hyperparameter	RFR	CNN
`Min samples split`	2	–
`No of estimators`	100	–
`Max depth`	20	–
`Criterion`	`squared error`	–
`Min samples leaf`	1	–
`Max features`	1.0 (all features)	–
`Bootstrap`	True	–
`Random state`	42	–
`Kernel size`	–	(3, 3)
`Batch size`	–	32
`No of filters`	–	32
`Activation`	–	ReLU
`Pooling`	–	MaxPooling2D (2, 2)
`Dropout`	–	0.5
`Dense layer units`	–	64
`Optimizer`	–	Adam
`Learning rate`	–	0.001
`Epochs`	–	300

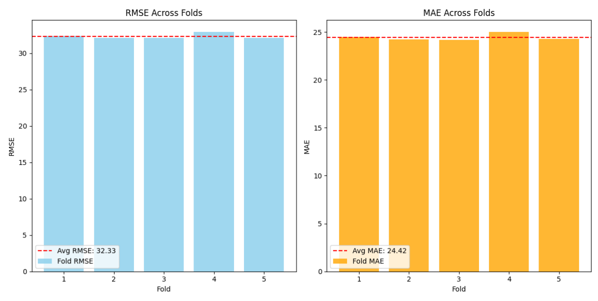


Figure 13: RFR 5-fold validation CTGAN

3.3.2. *Subsets*

After the synthetic data generation, different compositions of training data and synthetic data are fed to the regressors. To measure improvements on results, the following compositions are taken into account, shown in Table 9. The first and smallest subset depicts a scenario where training data is scarce, consisting of only 10% of the original training dataset. Other subsets contain 25%, 50%, 75%, and ultimately 100% of only training data to establish the baseline of the DGMs and investigate the effect of scarce data. When sorting the subsets, the percentages concern the number of engines. So 75% means 75 out of the 100 available engines in the CMAPSS training dataset. The engines are chosen randomly to counter model dependency on properties of a specific engine.

Next, synthetic data is added to the training data. Apart from just training data with 0%, the sizes of 50%, 100%, and 200% synthetic data addition were chosen to inspect the consequences, up until a dominant display of synthetic data in the subset. The added synthetic data is generated by DGMs trained with the same percentage of training data as it is combined with. For instance, for all 10% training subset combinations, all added synthetic data is generated by DGMs fitted on just 10% of the training data. All in all, 20 different subsets are set up for regression to have their results compared.

Table 9
Various compositions of subsets

		Training data percentage				
		10	25	50	75	100
Synthetic data percentage	0	×	×	×	×	×
	50	×	×	×	×	×
	100	×	×	×	×	×
	200	×	×	×	×	×

3.3.3. *Other*

After the synthetic data is generated, it is common practice to cap RUL outliers. Firstly, the final cycles of an engine are more informative than the initial ones in terms of degradation behaviour. Moreover, extremely high RUL values can mislead regressors by encouraging them to treat these anomalies as meaningful training signals. Therefore, a standardised cap is applied, setting all RUL values above 125 to a maximum of 125. This approach was first applied by Babu et al. (2016) [1] and later adopted in the 2017 IEEE Conference on Prognostics and Health Management [31], which has since become common practice in the literature. This transformation is mathematically expressed in Equation 27:

$$\text{cap}(\mathbf{RUL}) = \begin{cases} \mathbf{RUL}, & \text{if } \mathbf{RUL} \leq 125 \\ 125, & \text{if } \mathbf{RUL} > 125 \end{cases} \tag{27}$$

This way, all RUL values greater than 125 are capped at 125 and will not exceed this threshold.

3.4. Analysis of RUL predictions

Now, the RUL predictions by these regressors are compared, predicted RUL values (`ypred`) against actual RUL (`ytest`), and the performances of the DGM-regressor combinations are analysed.

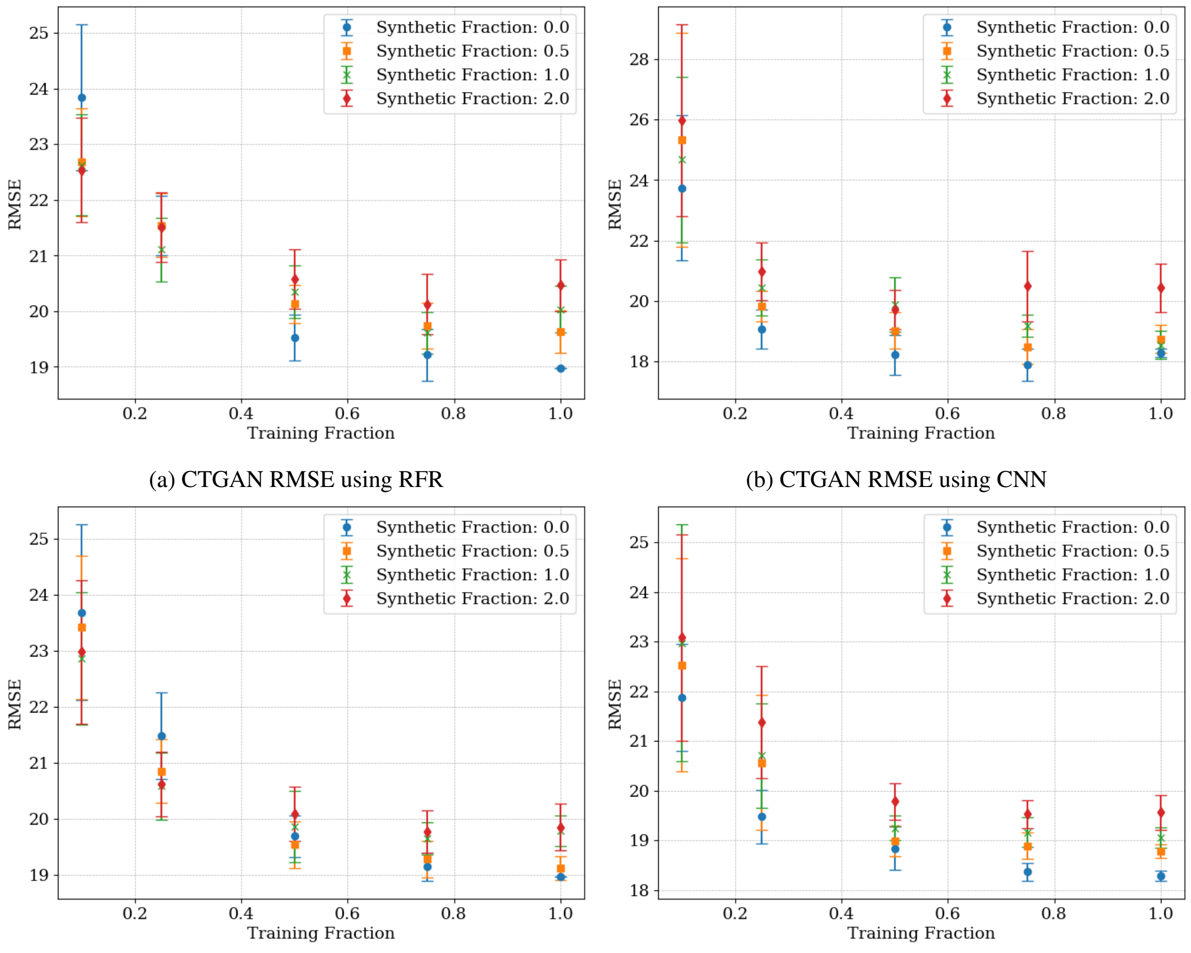
3.4.1. *RMSE*

The Root Mean Squared Error (RMSE) is a widely used metric for evaluating the performance of regression models, particularly in predicting the Remaining Useful Life (RUL).

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2} \tag{28}$$

Where:

- N is the number of instances in the test set.
- y_i represents the actual RUL values.
- $\hat{y}$ is the predicted RUL value.



(a) CTGAN RMSE using RFR

(b) CTGAN RMSE using CNN

(c) TVAE RMSE using RFR

(d) TVAE RMSE using CNN

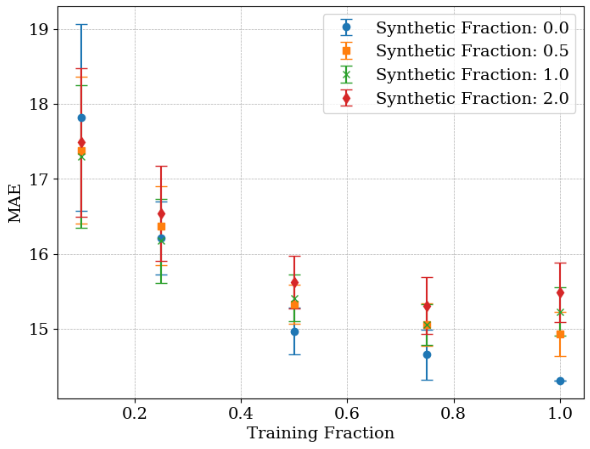
Figure 14: RMSE results for CTGAN and TVAE using RFR and CNN

The RMSE is particularly sensitive to large errors due to its squaring and therefore gives an accurate display of a surplus of outliers in the prediction versus the original data.

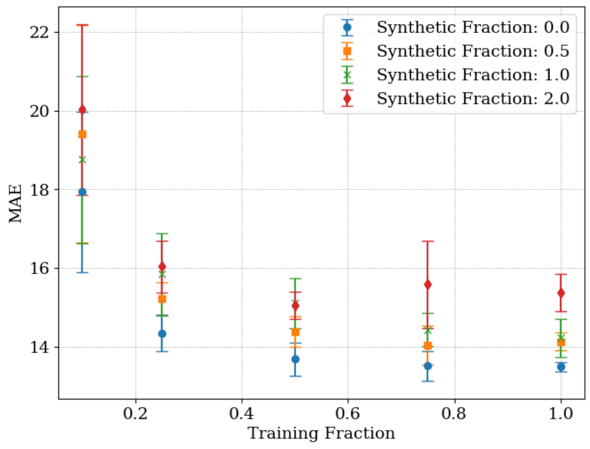
The RMSE results from all available combinations of DGM and regressor are plotted in Figure 14. All subsets are represented per plot, where a different symbol represents the synthetic fraction, the x-axis shows the percentage of training fraction, and the y-axis shows the ultimate values in terms of RMSE. All calculations have been performed 10 times to find an overall mean and insight into any anomalies. This is displayed by the corresponding bars, which indicate the 95% confidence interval of the 10 trials per subset combination.

The RMSE values after CNN regression are lower than those by RFR. During the grid search and K-fold validation, the preliminary RMSE scores were higher when utilising CNN, but in combination with the subset design, the CTGAN-CNN and TVAE-CNN performances change this around. Overall, all plots show similar trends throughout. The model performs better with increasing training data, which follows intuitive thinking. The appending of synthetic data does change all RMSE outputs, but seems to do so favourably when data is scarce and negatively if the training data is sufficiently available. The TVAE is less prone to the addition of synthetic data than the CTGAN: the means of all subsets with the same training fraction are much closer to one another in Figure 14c and 14d than Figure 14a and 14b, as well as the spread of the 10 trials resulting in lower standard deviations. As different properties of a CTGAN are not fully used, explained in Subsubsection 2.3.1, the TVAE performance was expected to be better.

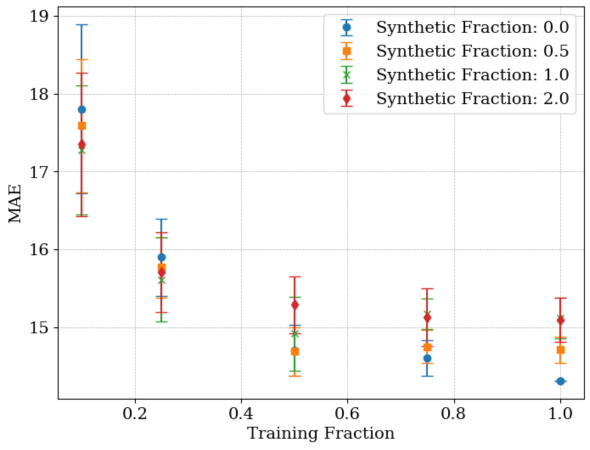
The CNN regressor has the overall best RMSE performance, but the addition of synthetic data seems to worsen its performance significantly for any training fraction. However, the use of RFR has more volatile results and larger confidence intervals: with the CTGAN-RFR combination specifically, a decrease of RMSE can be seen when the synthetic fraction is increased at the small training fractions (0.1 or 0.25). Thus, synthetic data seems to help improve RUL prediction performance, but the results have to be analysed thoroughly. More experiments are to be conducted to deem these results significant, though, as well as more specific trials and subset combinations.



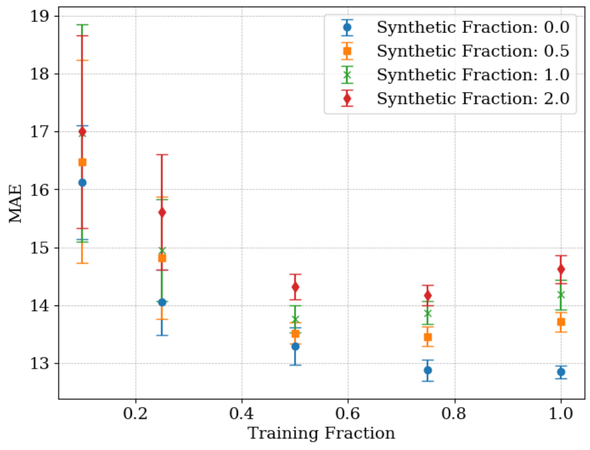
(a) CTGAN MAE using RFR



(b) CTGAN MAE using CNN



(c) TVAE MAE using RFR



(d) TVAE MAE using CNN

Figure 15: MAE results for CTGAN and TVAE using RFR and CNN

3.4.2. *MAE*

The Mean Absolute Error (MAE) is another popular metric for evaluating the accuracy of RUL predictions. Unlike RMSE, MAE calculates the average of the absolute differences between predicted and actual RUL values, providing a more straightforward measure of prediction accuracy without heavily penalising larger errors.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}| \tag{29}$$

Where:

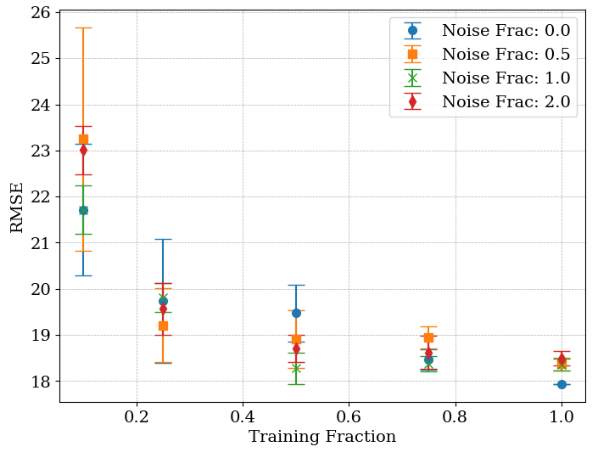
- N is the number of instances in the test set.
- y_i represents the actual RUL values.
- $\hat{y}$ is the predicted RUL value.

Therefore, MAE is more robust to outliers compared to RMSE and gives a more balanced view of the model's performance.

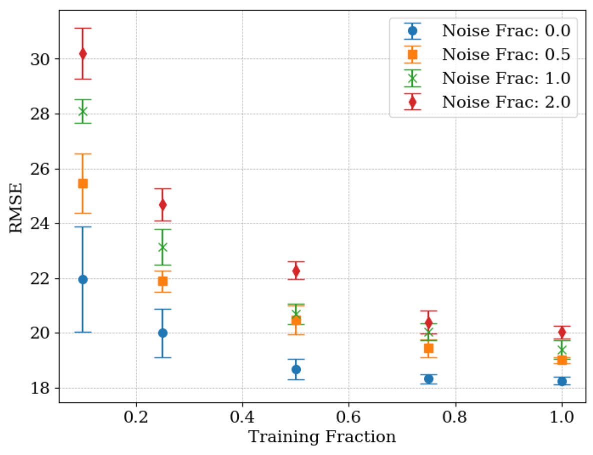
The plots in Figure 15 are generated by the same outputs as in Figure 14, and therefore can be compared to one another. Firstly, the MAE results are consistently lower than the RMSE results. This indicates that outliers in either the real RUL data or the predicted RUL are present. Nevertheless, the trend in decreasing MAE as the training fraction increases is again apparent. Also, the influence of synthetic data addition is similar to the RMSE results, a positive effect when training data is scarce and a negative impact when training data is sufficiently available. Again, the TVAE and CTGAN both perform roughly the same, but the CNN has better results than the RFR as a regressor overall. To give the plots more meaning, both performance metrics will be compared to a standard data augmentation technique in Subsubsection 3.4.3. Also, to take a deeper look into the RMSE and MAE results, the underlying will be discussed more in Subsubsection 3.4.4.

3.4.3. *Noise*

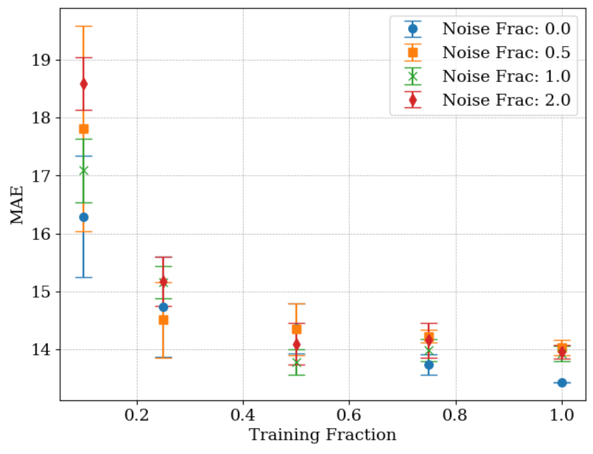
A widely used method of data augmentation, the addition of noise, was implemented to give a more sophisticated understanding of the behavioural changes of altered datasets. A noise created by a Gaussian distribution as large as its standard deviation was added to the training dataset in the



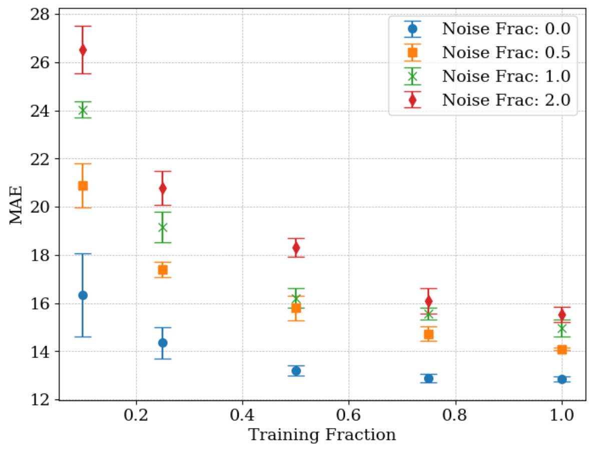
(a) Noise RMSE using RFR



(b) Noise RMSE using CNN



(c) Noise MAE using RFR



(d) Noise MAE using CNN

Figure 16: RMSE and MAE results for Noise Regression using RFR and CNN

same combination setup as the earlier used subsets. Again, 10 trials were carried out to find a stable mean and standard deviation, expressed similarly to the RMSE plots in Figure 14.

The model works as validation for the subset regression, too. The blue round subsets with no added noise should be similar to or equal to their blue round subsets without any added synthetic data. The RMSE values for the CTGAN-RFR and CTGAN-CNN are showcased in Figure 16, where the matching trend of decreasing RMSE when increasing the amount of training data is repeated. Also, the range of RMSE for both the RFR and CNN outputs is the same as for the DGMs. However, the addition of noise has not helped lower RMSE for any subset combination. Therefore, in terms of the RMSE metric, data augmentation by either a CTGAN or TVAE gives better results than noise addition. For TVAE, the same trend is visible: starting RMSE values on par with the plots in Figure 14c and 14d, with increasing noise fractions causing higher outcomes.

The noise regression was not only specified for RMSE, but also gives insights into MAE as well. In Figure 16c and 16d, the noise regression with metric MAE shows the same general trend of decreasing values when increasing the training fraction, and the increase of noise fraction causes higher MAE values. Thus, the same conclusion can be drawn for MAE as it is for RMSE: the use of DGMs can be helpful in RUL prediction in comparison to simply data augmentation methods such as noise addition. Also, the CNN shows even more distinct results than the RFR, where the addition of noise causes significantly worse results.

3.4.4. Residuals / Deeper insight in RMSE & MAE

To gain a deeper understanding of the regression models' performance and the distribution of errors, the residuals are analysed. The bar plot in Figure 18 shows the distribution of the average residuals per training fraction. It reveals that models trained with a low training fraction tend to systematically underestimate RUL, leading to more negative residuals and greater spread. As the training fraction increases, the residuals stabilise and move closer to zero, indicating improved prediction quality.

The scatter plot confirms this pattern by displaying residuals against predicted RUL, seen in Figure 17. It shows that at higher RUL values, the spread increases, particularly for lower training fractions, suggesting heteroscedasticity. The addition of synthetic data introduces extra variability but does not change this fundamental pattern. The red dashed line at zero serves as a reference, highlighting potential systematic biases in the predictions.

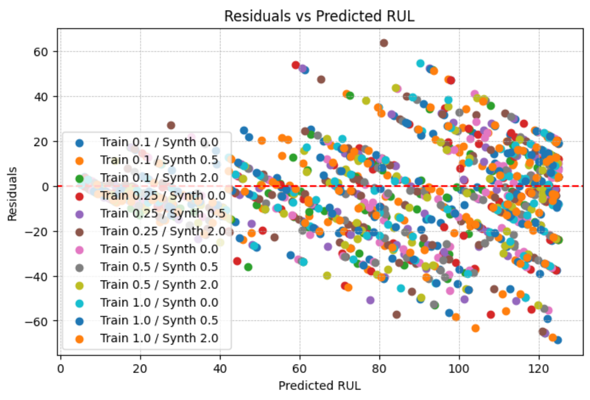


Figure 17: Residual Scatter plot

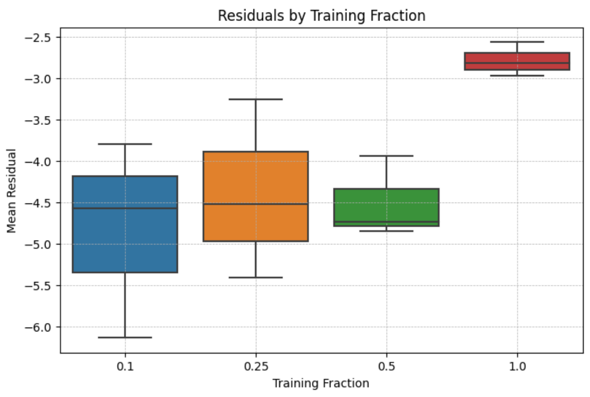


Figure 18: Residual bar plot

4. Conclusion

The conducted research aimed to evaluate the effectiveness of synthetic data generation using CTGAN and TVAE in combination with two distinct regression models, RFR and CNN, for predicting Remaining Useful Life (RUL) in the CMAPSS FD001 dataset. By examining synthetic data quality, assessing regression performance, and introducing noise as a comparison method, several key findings emerged.

The generated synthetic data was evaluated using descriptive statistics, visual validation, and statistical metrics. Across all these methods, the synthetic data generally demonstrated characteristics closely resembling the original training data. While both CTGAN and TVAE produced data that aligned well with the real dataset's mean and standard deviation, the TVAE tended to provide a more stable distribution when combining synthetic and real data. The CTGAN, however, performed better in terms of Wasserstein distance, suggesting superior handling of global distribution properties. Despite their comparable KS-test results, the CTGAN's advantage in Wasserstein distance may provide more robust data for regression tasks, especially when smaller subsets of training data are used.

Noise-based data augmentation served as a baseline comparison to synthetic data generation. Results indicated that noise addition generally worsened RMSE and MAE scores across all subsets. This outcome emphasises that data generated via CTGAN or TVAE provides a more effective augmentation strategy than noise alone. Both the noise regression plots and the performance of DGMs in data-scarce scenarios further show the value of synthetic data for improving predictive accuracy when limited training data is available.

After a robust regression across multiple subsets while maintaining reasonable computational costs, the analysis of RMSE and MAE revealed consistent trends across both metrics:

- Increasing the training fraction consistently improved model performance (lower RMSE and MAE).
- Synthetic data marginally enhanced performance when real data was limited (e.g., 10% or 25% training data).
- When combined with larger training fractions (e.g., 75% or 100%), synthetic data hindered performance.
- TVAE's performance was less sensitive to the addition of synthetic data compared to CTGAN.
- In all cases, CNN outperformed RFR, achieving lower RMSE and MAE scores.
- Overall, synthetic data is most beneficial when data scarcity is a concern. When sufficient real data is available, the addition of synthetic data offers diminishing or even adverse effects.

The research demonstrates that combining synthetic data generation with regression models can offer meaningful improvements when training data is limited. The CTGAN's superior Wasserstein distance results suggest stronger global alignment with the training data, making it preferable for scenarios where broader distribution accuracy is essential. Meanwhile, the TVAE's stability across data compositions makes it a robust option when a balanced approach is required.

Future work could be to further delve into better accommodating and training GANs and VAEs for a specific dataset, such as FD001. Also, more exploration of other generative models (e.g., Diffusion Models [10], specialised time-series models [18], Normalising Flows [21]) and assessment of their potential value for prognostics tasks could prove interesting. Also, better integration of domain knowledge during the pre-processing of the data to guide the generation process may enhance realistic properties and utility purposes of the synthetic data further.

References

[1] Babu, G.S., Zhao, P., Li, X., 2016. Deep convolutional neural network based regression approach for estimation of remaining useful life, in: Annual Conference of the Prognostics and Health Management Society 2016.

[2] Bank, D., Koenigstein, N., Giryes, R., 2020. Autoencoders. URL: https://arxiv.org/abs/2003.05991, doi:10.48550/ARXIV.2003.05991.

[3] Baptista, M., Goebel, K., Henriques, E., 2022. Relation between prognostics predictor evaluation metrics and local interpretability shap values. Artificial Intelligence 306, 103667. doi:10.1016/j.artint.2022.103667.

[4] Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

[5] Cao, H., Tan, C., Gao, Z., Chen, G., Heng, P.A., Li, S.Z., 2022. A survey on generative diffusion model. arXiv e-prints URL: https://arxiv.org/abs/2209.02646, doi:10.48550/ARXIV.2209.02646.

[6] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., Bharath, A.A., 2018. Generative adversarial networks: An overview. IEEE Signal Processing Magazine 35, 53–65. doi:10.1109/MSP.2017.2765202.

[7] Frederick, D., de Castro, J., Litt, J., 2007. User's Guide for the Commercial Modular Aero-Propulsion System Simulation (CMAPSS). Technical Report. NASA. Available online: https://ntrs.nasa.gov/citations/20070034949 (accessed on 30 September 2021).

[8] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. stat 1050, 10.

[9] Ha, V., 2023. Experimental study on remaining useful life prediction of lithium-ion batteries based on three regressions models for electric vehicle applications. Preprints doi:10.20944/preprints202306.0999.v1.

[10] Ho, J., Jain, A., Abbeel, P., 2020. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems 33, 6840–6851. URL: https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d96effd7fc1a54b38-Paper.pdf.

[11] Ho, T.K., 1995. Random decision forests, in: Proceedings of the 3rd International Conference on Document Analysis and Recognition, IEEE. pp. 278–282.

[12] Kim, N.H., An, D., Choi, J.H., 2017. Prognostics and health management of engineering systems. Springer.

[13] Kingma, D.P., Welling, M., 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 .

[14] Kolmogorov, A.N., 1933. Sulla determinazione empirica di una legge di distribuzione. Giornale dell'Istituto Italiano degli Attuari 4, 83–91.

[15] Kumar, P., Pooja, Chauhan, N., Chaurasia, N., 2023. A vision-based pothole detection using cnn model. SN Computer Science 4. doi:10.1007/s42979-023-02153-w.

[16] Lazarova-Molnar, S., Niloofar, P., Barta, G., 2020. Data-driven fault tree modeling for reliability assessment of cyber-physical systems, in: Proceedings of the 2020 Winter Simulation Conference (WSC), pp. 2719–2730. doi:10.1109/WSC48552.2020.9383882.

[17] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

[18] Lim, B., Zohren, S., 2021. Deep learning for time series forecasting: A survey. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 379, 20200207. doi:10.1098/rsta.2020.0207.

[19] Van der Maaten, L.J.P., Hinton, G.E., 2008. Visualizing high-dimensional data using t-sne. Journal of machine learning research 9, 2579–2605.

[20] O'shea, K., Nash, R., 2015. An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458 .

[21] Rezende, D.J., Mohamed, S., 2015. Variational inference with normalizing flows. International Conference on Machine Learning 37, 1530–1538. URL: http://proceedings.mlr.press/v37/rezende15.pdf.

[22] Rodrigues, L.R., Yoneyama, T., Nascimento, C.L., 2012. How aircraft operators can benefit from phm techniques, in: 2012 IEEE Aerospace Conference, pp. 1–8. doi:10.1109/AERO.2012.6187376.

[23] Rosenblatt, M., 1956. Remarks on some nonparametric estimates of a density function. Annals of Mathematical Statistics 27, 832–837.

[24] Saxena, A., Goebel, K., 2008. Turbofan engine degradation simulation data set. NASA Ames Prognostics Data Repository URL: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/.

[25] Shorten, C., Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. Journal of big data 6, 1–48.

[26] Synthetic Data Vault (SDV) Project, 2024. Sdv: Synthetic data generation for machine learning. URL: https://sdv.dev/. accessed: 2025-02-17.

[27] Van Dyk, D.A., Meng, X.L., 2001. The art of data augmentation. Journal of Computational and Graphical Statistics 10, 1–50.

[28] Villani, C., 2008. Optimal transport: old and new. Springer Science & Business Media.

[29] Xu, L., Lei, S., Vzaimosvyaz, K., 2019a. Modeling tabular data using conditional variational autoencoder. arXiv preprint arXiv:1907.00503 .

[30] Xu, L., Skoularidou, M., Cuesta-Infante, A., Veeramachaneni, K., 2019b. Modeling tabular data using conditional gan, in: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, pp. 7339–7349.

[31] Zheng, S., Ristovski, K., Farahat, A., Gupta, C., 2017. Long short-term memory network for remaining useful life estimation, in: 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), IEEE. pp. 1–7.

A. Appendix

A.1. Descriptive validation tables

Table 10
Full descriptive validation CTGAN

Variable	mean Real	std Real	var Real	mean Synthetic	std Synthetic	var Synthetic
Cycle	108.807862	6.888099e+01	4.744591e+03	100.299792	6.251703e+01	3.908379e+03
Op1	-0.000009	2.187313e-03	4.784340e-06	-0.000048	1.475783e-03	2.177936e-06
Op2	0.000002	2.930621e-04	8.588541e-08	0.000231	3.496139e-04	1.222299e-07
Op3	100.000000	0.000000e+00	0.000000e+00	100.000000	0.000000e+00	0.000000e+00
Sensor1	518.670000	6.537152e-11	4.273435e-21	518.670000	6.537152e-11	4.273435e-21
Sensor2	642.680934	5.000533e-01	2.500533e-01	642.722626	4.265636e-01	1.819565e-01
Sensor3	1590.523119	6.131150e+00	3.759099e+01	1590.927466	6.810537e+00	4.638342e+01
Sensor4	1408.933782	9.000605e+00	8.101089e+01	1408.054137	8.647853e+00	7.478536e+01
Sensor5	14.620000	3.394700e-12	1.152399e-23	14.620000	3.394700e-12	1.152399e-23
Sensor6	21.609803	1.388985e-03	1.929279e-06	21.609928	8.439523e-04	7.122554e-07
Sensor7	553.367711	8.850923e-01	7.833883e-01	553.515977	8.073305e-01	6.517825e-01
Sensor8	2388.096652	7.098548e-02	5.038938e-03	2388.087815	6.673351e-02	4.453361e-03
Sensor9	9065.242941	2.208288e+01	4.876536e+02	9064.615385	1.984636e+01	3.938778e+02
Sensor10	1.300000	4.660829e-13	2.172333e-25	1.300000	4.660829e-13	2.172333e-25
Sensor11	47.541168	2.670874e-01	7.133568e-02	47.521325	2.553851e-01	6.522153e-02
Sensor12	521.413470	7.375534e-01	5.439850e-01	521.506963	8.370096e-01	7.005850e-01
Sensor13	2388.096152	7.191892e-02	5.172330e-03	2388.075943	7.364034e-02	5.422899e-03
Sensor14	8143.752722	1.907618e+01	3.639005e+02	8145.383295	1.694860e+01	2.872550e+02
Sensor15	8.442146	3.750504e-02	1.406628e-03	8.445521	4.939359e-02	2.439727e-03
Sensor16	0.030000	1.556432e-14	2.422479e-28	0.030000	1.556432e-14	2.422479e-28
Sensor17	393.210654	1.548763e+00	2.398667e+00	393.007222	1.534131e+00	2.353559e+00
Sensor18	2388.000000	0.000000e+00	0.000000e+00	2388.000000	0.000000e+00	0.000000e+00
Sensor19	100.000000	0.000000e+00	0.000000e+00	100.000000	0.000000e+00	0.000000e+00
Sensor20	38.816271	1.807464e-01	3.266927e-02	38.834747	1.677327e-01	2.813425e-02
Sensor21	23.289705	1.082509e-01	1.171825e-02	23.291030	1.080587e-01	1.167668e-02
RUL	107.807862	6.888099e+01	4.744591e+03	102.253550	6.578449e+01	4.327599e+03

Table 11
Full descriptive validation TVAE

Variable	mean Real	std Real	var Real	mean Synthetic	std Synthetic	var Synthetic
Equipment	51.506568	1.922703e+01	8.542545e+02	78.201541	1.962570e+01	3.851676e+02
Cycle	108.807962	6.888099e+01	4.744591e+03	55.716440	4.343902e+01	2.414661e+03
Op1	0.000009	3.373713e-03	4.748340e-06	-0.000003	2.068468e-03	4.278550e-06
Op2	0.000002	2.593062e-04	8.588541e-08	-0.000022	2.872634e-04	8.252028e-08
Op3	100.000000	0.000000e+00	0.000000e+00	100.000000	0.000000e+00	0.000000e+00
Sensor1	518.670000	6.537152e+11	4.273435e-21	518.670000	6.537152e-11	4.273435e-21
Sensor2	642.680934	5.000533e+01	2.500533e+01	642.621655	3.049414e-01	9.298924e-02
Sensor3	1590.523119	6.131150e+00	3.759099e+01	1590.144743	3.544630e+00	1.256404e+01
Sensor4	1408.933782	9.000605e+00	8.101068e+01	1408.334849	5.429744e+00	2.948212e+01
Sensor5	14.620000	3.394700e-12	1.152398e-23	14.620000	3.394700e-12	1.152398e-23
Sensor6	21.609803	1.388985e-03	1.929279e-06	21.610000	9.702696e-12	9.414231e-23
Sensor7	553.367711	8.850923e-01	7.833883e-01	553.415078	5.799003e-01	3.362844e-01
Sensor8	2388.096652	7.098548e-02	5.038938e-03	2388.096823	4.960659e-02	2.460814e-03
Sensor9	9065.242941	2.208288e+01	4.876536e+02	9056.169373	7.730767e+00	5.976477e+01
Sensor10	1.300000	4.660829e-13	2.172333e-25	1.300000	4.660829e-13	2.172333e-25
Sensor11	47.541168	2.670874e-01	7.133568e-02	47.525806	1.657651e-01	2.747808e-02
Sensor12	521.471470	7.375534e-01	5.439850e-01	521.470767	4.473883e-01	2.001563e-01
Sensor13	2388.096652	7.098548e-02	5.038938e-03	2388.096823	4.960659e-02	2.460814e-03
Sensor14	8143.752722	1.907618e+01	3.639005e+02	8137.178497	7.947285e+00	6.315933e+01
Sensor15	8.442146	3.750504e-02	1.406628e-03	8.436551	2.293593e-02	5.260567e-04
Sensor16	0.030000	1.556432e-14	2.422479e-28	0.030000	1.556432e-14	2.422479e-28
Sensor17	393.210654	1.548763e+00	2.398667e+00	393.091464	6.626379e-01	4.390890e-01
Sensor18	2388.000000	0.000000e+00	0.000000e+00	2388.000000	0.000000e+00	0.000000e+00
Sensor19	100.000000	0.000000e+00	0.000000e+00	100.000000	0.000000e+00	0.000000e+00
Sensor20	38.816271	1.807464e-01	3.266927e-02	38.814986	1.321085e-01	1.745266e-02
Sensor21	23.289765	1.082059e-01	1.171825e-02	23.293768	6.633278e-01	4.400038e-03
RUL	107.807862	6.888099e+01	4.744591e+03	84.729049	4.445125e+01	1.975913e+03

A.2. Statistical validation tables

Table 12
Full Wasserstein Distance Results for CTGAN and TVAE

Variable	CTGAN			TVAE		
	Mean	CI Lower	CI Upper	Mean	CI Lower	CI Upper
Cycle	6.986	6.966	7.005	6.928	6.895	6.961
Op1	0.001	0.001	0.001	0.000	0.000	0.000
Op2	0.000	0.000	0.000	0.000	0.000	0.000
Op3	0.000	0.000	0.000	0.000	0.000	0.000
Sensor1	0.000	0.000	0.000	0.000	0.000	0.000
Sensor2	0.043	0.043	0.043	0.023	0.023	0.023
Sensor3	0.961	0.959	0.963	0.276	0.274	0.277
Sensor4	1.219	1.217	1.220	0.379	0.378	0.381
Sensor5	0.000	0.000	0.000	0.000	0.000	0.000
Sensor6	0.002	0.002	0.002	0.002	0.002	0.002
Sensor7	0.321	0.321	0.322	0.286	0.286	0.287
Sensor8	0.014	0.014	0.014	0.012	0.012	0.012
Sensor9	4.538	4.533	4.543	2.264	2.257	2.270
Sensor10	0.000	0.000	0.000	0.000	0.000	0.000
Sensor11	0.044	0.044	0.044	0.010	0.010	0.010
Sensor12	0.258	0.257	0.258	0.132	0.131	0.132
Sensor13	0.014	0.014	0.014	0.008	0.008	0.008
Sensor14	2.065	2.061	2.069	2.424	2.418	2.431
Sensor15	0.009	0.009	0.009	0.003	0.003	0.003
Sensor16	0.000	0.000	0.000	0.000	0.000	0.000
Sensor17	0.199	0.199	0.199	0.167	0.167	0.167
Sensor18	0.000	0.000	0.000	0.000	0.000	0.000
Sensor19	0.000	0.000	0.000	0.000	0.000	0.000
Sensor20	0.034	0.034	0.034	0.009	0.009	0.009
Sensor21	0.008	0.008	0.008	0.007	0.007	0.007
RUL	10.606	10.590	10.623	16.999	16.965	17.033

Table 13
Full KS Test Results for CTGAN and TVAE

Variable	CTGAN			TVAE		
	Mean	CI Lower	CI Upper	Mean	CI Lower	CI Upper
Cycle	0.055	0.055	0.055	0.039	0.039	0.039
Op1	0.190	0.190	0.190	0.025	0.025	0.025
Op2	0.049	0.049	0.049	0.075	0.075	0.075
Op3	0.000	0.000	0.000	0.000	0.000	0.000
Sensor1	0.000	0.000	0.000	0.000	0.000	0.000
Sensor2	0.048	0.048	0.048	0.023	0.023	0.023
Sensor3	0.085	0.085	0.085	0.022	0.022	0.022
Sensor4	0.065	0.065	0.065	0.019	0.019	0.019
Sensor5	0.000	0.000	0.000	0.000	0.000	0.000
Sensor6	0.065	0.065	0.065	0.060	0.060	0.061
Sensor7	0.062	0.062	0.062	0.059	0.059	0.059
Sensor8	0.044	0.044	0.044	0.040	0.040	0.040
Sensor9	0.152	0.152	0.152	0.045	0.044	0.045
Sensor10	0.007	0.007	0.007	0.004	0.004	0.004
Sensor11	0.095	0.095	0.095	0.017	0.017	0.017
Sensor12	0.052	0.052	0.052	0.018	0.018	0.018
Sensor13	0.052	0.052	0.052	0.033	0.033	0.033
Sensor14	0.044	0.044	0.044	0.068	0.067	0.068
Sensor15	0.050	0.050	0.050	0.023	0.023	0.023
Sensor16	0.000	0.000	0.000	0.000	0.000	0.000
Sensor17	0.064	0.064	0.064	0.036	0.036	0.036
Sensor18	0.000	0.000	0.000	0.000	0.000	0.000
Sensor19	0.000	0.000	0.000	0.000	0.000	0.000
Sensor20	0.086	0.086	0.086	0.023	0.023	0.023
Sensor21	0.037	0.037	0.038	0.023	0.023	0.023
RUL	0.066	0.066	0.066	0.089	0.088	0.089

This Literature Study has been completed and graded as part of the Master’s Program requirements.

II

Literature Study

1 Introduction

The opportunities for data augmentation in the field of **prognostics and health management (PHM)** are studied in this thesis Kim et al., 2017, especially the use of deep learning methods Deng, Yu, et al., 2014 for data generation. In this project, generative data techniques are investigated on how they augment data sets for predictive models and on how the different generative techniques compare to each other. In preparation for this thesis, a preliminary literary review was conducted in the field of PHM and generative modelling. The literature study ensures that the work will be novel and not performed before.

PHM in aviation is covered in chapter 2. The document describes different sorts of applicable maintenance and how they relate and differ from one another. The isolation of the end-of-life (EoL) points and the health of systems are discussed in chapter 2 as well. In section 2.3 **health monitoring trajectories** and their characteristics (monotonicity vs non-monotonicity, trendability, prognosability) are addressed Baraldi et al., 2018. Furthermore, the major causes of the lack of **data quality** are also addressed in chapter 2, followed by an explanation of the importance of data quality in PHM. Consequently, the importance of quality in data generation is addressed as well Jain et al., 2020.

Next, **data augmentation** will be explored in chapter 3 Van Dyk and Meng, 2001 Shorten and Khoshgoftaar, 2019. Several techniques are considered, each with its specificities, disadvantages, and advantages. Furthermore, chapter 3 will focus on the field of **data generation** and the existing generative modelling techniques. Different generative models are mentioned, including Hidden Markov models, Bayesian networks, and Latent Dirichlet Allocation. Also, four deep generative techniques are discussed in depth in section 3.2: namely, the **generative pre-trained transformer (GPT)**, **autoencoder (AE)**, **variational autoencoder (VAE)**, and **generative adversarial network (GAN)** Q. Zhu and Luo, 2022 Bank et al., 2020 Creswell et al., 2018.

In chapter 6, the **problem statement** and research for this thesis are presented. First, the research gap in this field of research is discussed as well as the objective of this thesis. The **research question** and its **sub-questions** can also be found in chapter 6 as well as the underlined objective.

Next, the **approach** is described in chapter 5. Here the methodology on how to achieve the goals set in the problem statement can be found. The planning on when these goals should be achieved is presented in section 5.2. The goals include the mid-term meeting, the ”Research Methodology” course deadline, and the Green Light. This is visualised in a **calendar** and a **Gantt chart**.

Finally, in chapter 7 there is the conclusion on the literature study. This includes future recommendations and a final reflection.

2

Prognostics & health management

In the field of machinery, maintenance is essential to provide safe conditions while disrupting productivity as little as possible Saltoglu et al., 2016. In aviation, maintenance can be implemented using the technology of PHM, a discipline that strives to schedule repairs or preventive measures by the means of prediction.

2.1. Maintenance in aviation

Most components in aviation are subject to maintenance, whether it is an aircraft valve, a fuselage part, or cylinders in an engine. In Figure 2.1, a common differentiation of maintenance is shown in a diagram ToolSense, 2023 Rodrigues et al., 2012.

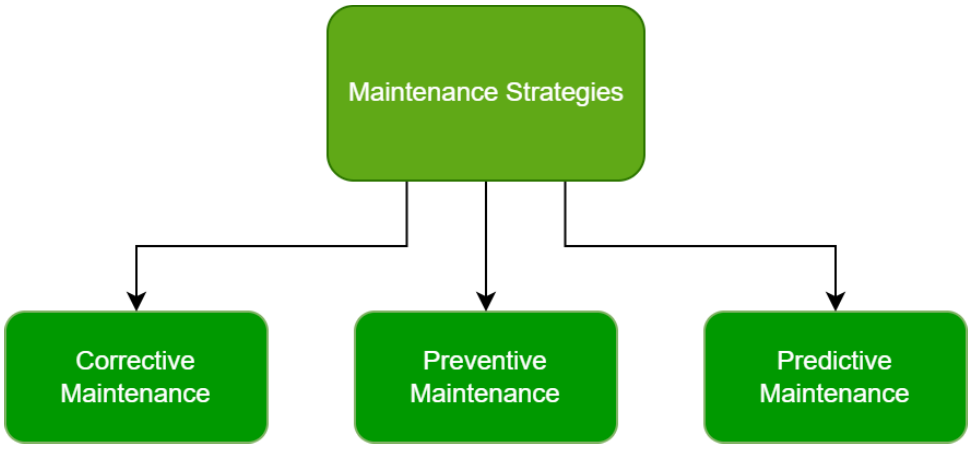


Figure 2.1: Maintenance strategies in aviation

The majority of maintenance activity can be described by the following strategies:

- Corrective maintenance
- Preventive maintenance
- Predictive maintenance

Corrective maintenance entails the repair of damaged systems. It only happens after a problem has been found. Even though inspections are scheduled and the run-to-failure (RTF) can be anticipated, the repair itself is not. Therefore, corrective maintenance has a reactive approach. The corrective repair can be from just restoring the system to operation to disassembling and rebuilding the system in total.

A benefit of reactive maintenance is the reduction of planning due to its unscheduled nature, and the simplification of the process by only intervening after the failure. A disadvantage is the difficulty to predict and prepare for failures as well as the lack of safety when a component or system fails during operation.

Another alternative strategy to corrective maintenance is preventive measures to keep systems in service. While reactive maintenance lets the failures occur first preventive maintenance action is taken to prevent failures. Preventive maintenance can be achieved by time-based or usage-based routine, by scheduled examination of a specific system's condition IBM, 2023.

Repair based on time, usage, or another fixed quantity of time can be seen as predetermined maintenance. Instead of focusing on the performance of a system, scheduled repairs are programmed. These schedules are mostly based on thresholds based on statistical knowledge, assuming that the system will perform satisfactorily until the predetermined repair event. This type of maintenance reduces planning since there are predetermined repair dates and minimises time and resources during the predictive maintenance stage.

Another type is predictive maintenance. This focuses more on finding the moment of failure of a system and how to solve the extension of its life cycle. In this field of maintenance, new methods are developed to better anticipate the behaviour of the maintained system with increasing precision. The following section 2.2 is dedicated to this type of maintenance. The noted types of maintenance and their (dis)advantages are listed in Table 2.1.

Table 2.1: Advantages and disadvantages of maintenance

	Example	Advantages	Disadvantages
Reactive	Replacement of broken parts	Reduced planning Simple process Helpful for non-critical parts	Difficult to predict Safety issues
Predetermined	Repair scheduled on time or usage	Reduced Planning Easy to implement on larger scale	Not tailored Rigid
Predictive	Smoke detection	Decrease of downtime Tailored Accurate prognostics	Real-time monitoring Uncharted technology

2.2. Predictive maintenance

Another type of maintenance utilises the state of a system as an indication for repair. When the condition of an observed system is taken into account coupled with predictive algorithms, it is called predictive maintenance. Through observation, data analysis, and other types of inspection an assessment is made. Either the system must be repaired or scheduled for a new moment of inspection. Sensors help find system conditions that are out of the ordinary and require maintenance. A small example is smoke detectors: the detection of smoke is a change of conditions causing an alarm and subsequent maintenance procedures such as turning on sprinklers.

This type of maintenance is beneficial to systems that can be observed full-time. With integrated sensors that check the status of such a system at all times, the overall downtime is generally lower. This consequently results in higher productivity. However, predictive maintenance involves more cost than the aforementioned types. The system has to be observed all the time instead of at predetermined moments or just when it is not working anymore and the fix must be available as well to enjoy the noted advantages of this type.

Predictive maintenance can also be more data-driven. By analysing data from the observed system as well as model-based data from comparable situations a prediction on when to intervene can be produced.

Predicting the moment of failure is the maintenance discipline that contains the discipline of PHM. The time predicted until failure is described as remaining useful life (RUL). To predict the RUL and thus the time frame for intervening, PHM uses modelling to represent and find useful information in the available data. Prognostics can be categorised as physics-based prognostics, data-based prognostics, or a hybrid combination of the two. Here physics-based prognostics use data from physical hand-built models and data-based prognostics stem directly from the data Schwabacher, 2005.

2.3. Health monitoring trajectories

When performing predictive maintenance, monitoring the health of the operated systems is important. To predict when failure will occur and consequently plan maintenance before the occurrence, health monitoring trajectories are obtained. These trajectories help visualise the degradation of systems and help establish the RTF and RUL. Several characteristics of the trajectory can describe how the respective system behaves. Features such as monotonicity, trendability, and prognosability are defined to give more insight into what their properties tell about the investigated system Baptista et al., 2022.

2.3.1. Monotonicity

Monotonicity is a property or a function that describes how strictly a system behaves as its input values increase or decrease. A trajectory is considered monotonic when it is always increasing, or always decreasing whilst variable input is changed (in one direction). So if a strictly monotonic function is visualised by a graph, its output will always move only up or only down as its input values change to the right or just to the left.

A function is said to be monotonically increasing if, for all values of the input variable, as the input increases, the output either stays the same or increases. More formally, a function $f(x)$ is monotonically increasing if for all $x1$ and $x2$ such that $x2 > x1$, we have $f(x2) \geq f(x1)$. Similarly, a function is said to be monotonically decreasing if, for all values of the input variable, as the input increases, the output either stays the same or decreases. More formally, a function $f(x)$ is monotonically decreasing if for all $x1$ and $x2$ such that $x1 < x2$, we have $f(x1) \geq f(x2)$.

$$monotonicity = \frac{1}{\mathbf{M}} \sum_{j=1}^{M} \sum_{k=1}^{N_j-1} \left| \frac{sng(x_j(k+1) - x_j(k))}{N_j - 1} \right| \tag{2.1}$$

The monotonicity equation is displayed in Equation 2.1. The mean of M trajectories is multiplied by the absolute differences between these trajectories divided by the number of observations Coble and Hines, 2009. Examples to illustrate monotonicity are: the function $f(x) = x^2$ is a monotonic function because it consistently increases as its input values (x) increase. The function $g(x) = 1/x$ is also monotonic, but it consistently decreases as its input values increase. However, when taking positive values in the range $[0 : 1]$, this is reversed.

On the other hand, non-monotonicity refers to a function that does not have a consistent pattern of increase or decrease while changing its variable in one direction. For example, oscillations such as sinuses are non-monotonic functions as they have turning points where the function changes direction as are $f(x) = x^3$ and $f(x) = x^5$, etc.

A value of 1 for monotonicity is called strictly monotonic whereas a value of 0 is defined as non-monotonic. This is an important property in mathematics and science. This also holds for optimisation and statistics, as it helps us understand the behaviour of functions to better predict their values. In Figure 2.2 the aforementioned equations are shown to give a visualisation.

$$monotonicity = \frac{1}{M} \sum_{j=1}^{M} \sum_{k=1}^{N_j-1} \frac{sng(x_j(k+1) - x_j(k))}{N_j - 1}$$

Figure 2.2: Four basic (polynomial) functions

2.3.2. Trendability

This feature is used to measure how much a predictor of a system's degradation shows the same trend throughout multiple trajectories for that system. This way the similarity between all the health monitoring trajectories is quantified. Therefore, trendability is very useful for explaining this kind of trajectory data Rigamonti et al., 2016.

$$trendability = \min_{j,k} \left|corr(x_j, x_k)\right| j, k \,\epsilon\{1, ..., M\} \tag{2.2}$$

The trendability equation also ranges between 0 and 1, with 1 setting the predictor as absolutely right and 0 making the predictor unreliable for discovering resemblances. The absolute value of correlation ($corr(x_j, x_k)$) between trajectories is computed between them all and ultimately the minimum value of all those is chosen, as shown in Equation 2.2. Naturally, trendability shows how similar the featured trajectories are compared to each other and then the least comparable values are chosen. For example, if in Figure 2.2 all functions would be seen as trajectories the largest difference between output is at $x = -2$ between x^2 and x^3. This would then be the least similar data point between trajectories.

2.3.3. Prognosability

Lastly, prognosability is useful for measuring the anticipated moment of failure. It utilises the standard deviation ($std, dev(x_{j(N_j)})$) and subsequently the variation of a health monitoring trajectory to find a value to quantify the degree of prognosability Rigamonti et al., 2016. This is described in Equation 2.3.

$$prognosability = \exp\left(-\frac{std, dev(x_{j(N_j)})}{mean\left|x_j(1) - x_j(N_j)\right|}\right), j \,\epsilon\{0, 1, ..., M\} \tag{2.3}$$

Again, the equation of prognosability ranges between 0 and 1, with 1 setting the prognoses of failure for all tested systems at the same time, namely, their EoL is similar. The value of 0, however, means that the moments of failure are very different for all tested and compared systems thus complicating prognostics.

2.4. Data quality

Model-based prognostics, data-based prognostics, and hybrid forms are all dependent on collected input data. To ensure results that are verified and validated, this input data must be of satisfactory quality. Also, there must be enough data to train a model to avoid problems such as overfitting. Several problems encountered in guaranteeing the quality of data can be insufficient sensor data, missing data, economic restrictions, lack of quality, and others Duffuaa and Ben□Daya, 1995 Lesage and Dehombreux, 2012.

2.4.1. Insufficient sensor data

As the main source of data, sensors are important for data quality. However, when using sensors different challenges can arise. Instalment of the sensors is performed in logical places to optimise the amount of data input, its produced data is still a simplified metric of reality. A common source of insufficient sensor data is just the lack of sensors installed or operating at the same time. This naturally causes a lack of sensor data, or at least insufficiently.

Where sensors are operated to monitor and maintain a process or system, they are also subjected to wear and tear, deficiencies, and other causes that require maintenance. During the downtime of sensors during maintenance no reliable data can be expected. Perhaps the acquired data before discovering malfunctioning sensors must also be discarded. Another cause of insufficient data by sensors can be the difficulties encountered when collecting, merging, and preparing the data from different sensors and their different outputs. This would not be likely in custom systems or products by a sole manufacturer. However, end products that combine different types of sensors could produce outputs that are not directly compatible or easy to interpret.

2.4.2. Missing data

Closely related to insufficient data by sensors is missing data. While technically lack of data by sensors can be seen as missing as well, this may also occur in other situations. Requested data might not be available as input for systems that are not used professionally yet. Also testing data for extremely rare situations might be difficult to come by.

Another reason for missing data could be cost-driven. Testing thoroughly is expensive both in time as well as in financial measures. This applies throughout all industries but aviation might stand out even more. First, the consequences of failure are likely to be very harmful. So, to ensure safety as much as possible elaborate testing is compulsory. Secondly, creating a test space that replicates the real circumstances can also be a tedious and costly process. Both make testing in the field of aeronautics much more expensive, resulting in only testing what is relevant for the modelling at hand. Further investigations or follow-up tests are therefore not handed abundant test data to choose from.

2.4.3. Obsolete data

The next cause for data not being reliable is the sheer lack of quality not by circumstances as described in the aforementioned paragraphs but by becoming obsolete to new methods and techniques used in maintenance. Data that was harvested and used for models a decade ago might not be compatible or useful to newly developed models nowadays, e.g. through the other use of variables or differently set parameters.

In aviation, many aspects are monitored and maintained by one of the types of maintenance mentioned just as in other industries. But aviation has several difficulties that others might not encounter. E.g. testing the end product in a true-to-life environment is difficult and very costly. Furthermore, safety is critical whereas a failure of a system in an aircraft might prove to be directly catastrophic. The latter difficulty makes scrutinous testing obligatory while the former suggests that generating an abundance of real test data is next to impossible.

To be able to test as much and as well as possible other ways of generating reliable data are researched. One of the methods to procure more useful data is an analysis technique called data augmentation (DA) which will be further elaborated on in chapter 3. During this thesis, that area of data prognostics will be researched more in-depth.

In Figure 2.3 a simple compass is shown. On the x-axis, data quality is described from range $[-1 : 1]$, -1 being very poor and 1 sublime. On the y-axis, a sufficient amount of data is shown, again from range $[-1 : 1]$, -1 being non-sufficient and 1 perfect. If your data set would be located in the lower left plane of the diagram, this means that the data set scores low in both sufficiency as well as quality. Upper left and lower right planes only are satisfactory in one of the characteristics. So, ultimately a proper data set to train a model with should be ranked in the upper right plane, preferably as close to the corner as possible. In this case, the blue dot represents such an outcome.

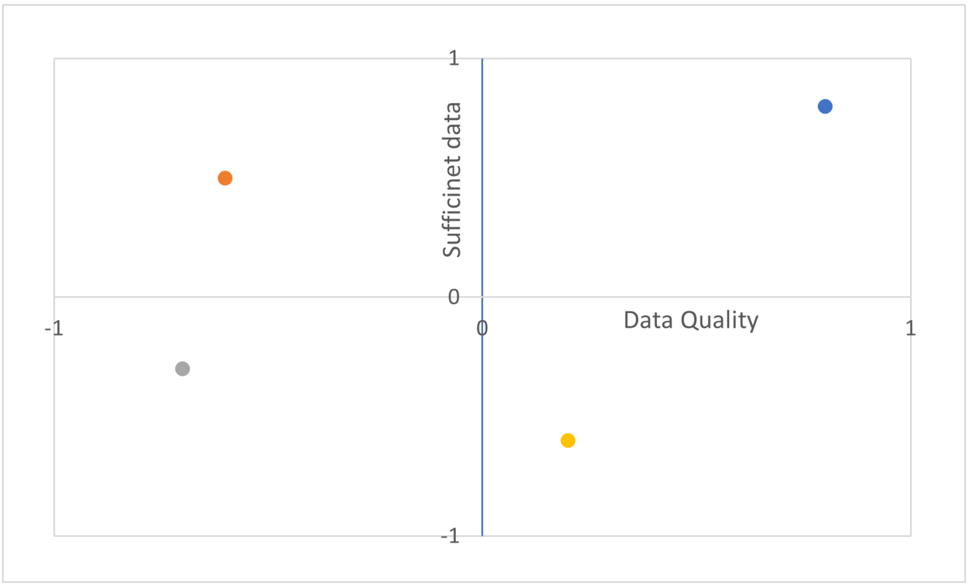


Figure 2.3: Data quality compass

3 Data augmentation

For the modelling in PHM, valid data are essential. As mentioned in chapter 2, some end products in aviation cannot always be tested until failure due to cost and safety conditions while actual failures during operation are disastrous. Therefore other approaches should be investigated. In data analysis, different techniques under the name of data augmentation can help provide data that is not easily retrievable by physical testing. The task of data augmentation is to take translational invariances in data sets and overcome issues such that the resulting models will have reliable outputs despite any challenges Van Dyk and Meng, 2001 Shorten and Khoshgoftaar, 2019 Xie et al., 2020.

Apart from providing larger datasets, data augmentation is also helpful in the prevention of overfitting, improving the accuracy of results, and operating a raw dataset.

Data augmentation normally uses existing data to augment at such a level that it can be fed into a prognostic model and produce effective results. It is used in several processes today already. For instance in signal processing and imaging but also in speech recognition.

3.1. Data augmentation techniques

To visualise the different data augmentation techniques and their subsequent methods to produce effective prognostic models, the diagram in Figure 3.1 is created.

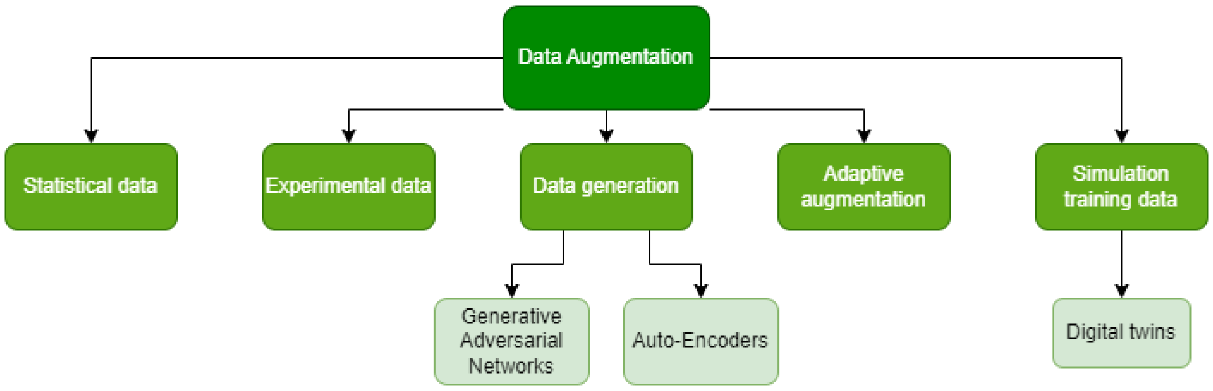


Figure 3.1: Data Augmentation related to GAN and AE

The field of data augmentation can roughly be divided into the following categories:

- Statistical augmentation of training data
- Training data with experimental data augmentation
- Generation of new training data from generative models
- Adaptive augmentation of training data
- High-fidelity simulation training data

The content of each of these categories and their differences are explained below.

3.1.1. Statistical augmentation

An often-used type of data augmentation is statistically augmenting training data. For example, extra data is added to the initial dataset by injection of Gaussian noise or a novel approach called double-noise injection C. Zhang and Baan, 2021. Another statistical technique to use is the Anscombe transform (AT) to transform noise to relevant data Eckert et al., 2020. Also randomly rotating images to enlarge a dataset of those images falls under this type of data augmentation.

This method is also applied in datasets that describe 3D graphics, but instead of input data in pixels now voxel data is augmented by a chosen statistical transformation. Voxel stands for volume-element (3D) as opposed to pixel, a picture element (2D). An example of 3D graphic augmentation can be found in Figure 3.2.

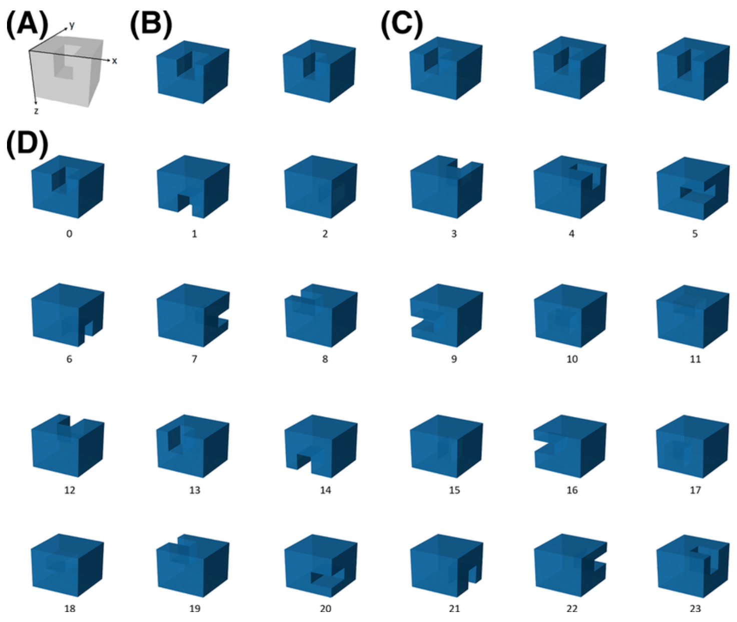


Figure 3.2: Data Augmentation for 3D structures Shi et al., 2020

3.1.2. Experimental data

Another type is the use of experimental methods to accumulate more data for the training dataset. Take image data augmentation as an example: whereas the aforementioned type uses statistical methods to manipulate, odd methods such as randomly zooming in or changing the lighting or dihedral angle of an image are also researched as possibilities for data augmentation. Also, data warping is investigated, for instance Wong et al., 2016.

The setup to develop a virtual counterpart to the real situation can be investigated. The scarce data harvested in reality combined with the data produced with the aforementioned techniques will test the virtual counterpart. Its purpose is to digitally create validated and verified results from prognostic models concerning safety indications, planned checkups, and EoL predictions. If the data, the modelling, and the subsequent interpretation are correct, the results gathered from the so-called digital twin should also apply to the real twin.

3.1.3. Adaptive augmentation

A relatively new type is adaptive data augmentation. While running the existing training data, the data augmentation's parameters are updated with information on the model's results. Therefore, the parameters adapting during the process aim to minimise training data loss and maximise the accuracy of the model Xu et al., 2022.

In visual tasks with images as data set adaptive augmentation can be very useful, particularly when these images differ extensively. This method is easily automated and can handle large data sets but needs human analysis to set relevant parameters during manual selection.

3.1.4. Simulation training data

Simulations with high fidelity can produce training data that is imbalanced: data describing failure is clouded by the abundance of regular test data. To counter this data augmentation can be utilised to deter the imbalance and improve the accuracy and robustness of the simulated predictions Lim et al., 2022.

Simulation training data augmentation is particularly applicable in specific scenarios that cannot be analysed easily during a simulation. For example, if a computer visualisation of an object is created out of several images, other camera angles that have not been shot can be generated by this type of data augmentation.

3.1.5. Data generation

The last technique described in Figure 3.1 is data generation produced through generative models. Where the abovementioned types of data augmentation also generate new data, this method specifically uses self-learning models. In Table 3.1 the aforementioned data augmentation techniques are listed including some of their respective benefits and disadvantages.

Table 3.1: Data augmentation methods

	Advantages	Disadvantages
Statistical augmentation	Fast iterations Developed DA method	Rigid
Experimental augmentation	Creative	Trivial
Generative augmentation	Cost-effective State-of-the-art techniques Promising results Rapid academic developments	Underdeveloped DA method
Adaptive augmentation	Instantaneous updating	Underdeveloped DA method Lack of academic knowledge
Simulation augmentation	Fully digital Emphasising on robustness	High cost

During this thesis, this type of artificial intelligence will be explored further, especially in investigating generative models. In the following section 3.2 several sorts of generative models, their peculiarities, and their abilities to generate useful training data will be elaborated on.

3.2. Generative models

Generating new data is important for PHM processes where real (test) data is scarce. Sampling will help self-learning models see patterns and make more qualitative predictions. However, since the new data are not real, validation and verification of the newly generated data are even more essential. A predictive model will not be useful if its input data is unreliable. Therefore any generative used in data augmentation must be reliable and ideally be able to generate new data unrecognisable from the real data in terms of authenticity. Several methods used in data generation will be discussed in chapter 5 concerning their pros and cons and how to implement them apart from each other as well as hybrid combinations.

To generate new data that is reliable and close to authentic data, different methods are available. Several of those techniques are fairly novel in the engineering world or were used for different purposes in the past. The following list names a couple:

- Gaussian mixture model
- Hidden Markov model
- Bayesian network
- Latent Dirichlet allocation
- Boltzmann machine
- Diffusion model
- Flow-based generative model
- **Deep generative models:**
  - *Generative pre-trained transformer*
  - *Variational autoencoder*
  - *Autoencoder*
  - *Generative adversarial network*

Below all listed types of generative models are briefly discussed, apart from the four deep generative models. They will be elaborated on in section 3.3.

3.2.1. Gaussian mixture model

The Gaussian mixture model (GMM) is used as a probability density function, a mixture of different Gaussian normal distributions and their characteristics. The Gaussian mixture model is the weighted combination to include these distributions into one result. For instance, a data set contains different distributions that are to be combined such as the weight of people categorised in different age groups. A GMM will use the mean, median, standard deviation, (co)variance, and other properties of each distribution to model an overall fit by iteration Reynolds et al., 2009.

Normally the estimation of a GMM is performed by an expectation-maximisation (EM) that uses an expectation step and a maximisation step to iterate. During the expectation step, the probability for each data point belonging to any of the treated Gaussian distributions is estimated, so-called responsibilities. The maximisation step will update the estimations by maximising the likelihood of the data concerning the model parameters Han et al., 2021. This two-step algorithm will iterate between one another until the result has converged into a stable solution.

The GMM is used in many fields of engineering as well as in finance or anywhere where statistics including Gaussian distributions are relevant.

3.2.2. Hidden Markov model

The hidden Markov model (HMM) represents a Markov process in which the states are not detailed beforehand. A data set or a sequence of events in time can be described without the information of how it was created, thus the generation process is not observable. Instead, it is assumed that a hidden process is responsible for the data. This hidden process can be provided by using the technique of an HMM.

There are two elements that act together as HMM. First is the hidden state process, a Markov process in which the state of $t+1$ is only dependent on the state of t. Secondly, there is an observable output process that visualises the hidden states from the hidden state process. An HMM can be used to generate new data by transforming input data from a training data set. After the HMM is trained and the right parameters are set for both the hidden state process as well as the observable output process, new data can be simulated. This new data is then combined with the original data set, hence performing data augmentation Eddy, 2004.

3.2.3. Bayesian network

A Bayesian network (BN) is a probabilistic graphical model that represents probabilities between variables graphically. A BN consists of a set of nodes, each representing one of the considered variables. Also, edges are in place that represent the probabilistic dependency between these variables.

In this model, a set of variables have a joint probability distribution and also conditional probabilities can be displayed, giving an insightful visualisation of dependencies and reason in complex systems with various variables. Also, new scenarios or updated probabilities are easily incorporated into an existing Bayesian network. They are therefore especially useful when dealing with uncertainty issues by representing possibilities of events in clear visuals.

In data augmentation, BNs are also applied to generate new training data when sampling the input data and desired output data. A BN provides the conditional dependencies between the input and output and ensures that newly generated data fit these properties when fed to a distribution model. The parameters of the Bayesian network are thus important to indicate the extent of dissimilarity for the new data Stephenson, 2000.

3.2.4. Latent Dirichlet allocation

The latent Dirichlet allocation (LDA) acts as a generative statistical model using topics and a Dirichlet distribution. It can be used to discover hidden patterns in large data collections, particularly text. An LDA assumes a model that is represented as documents generated from a set of topics, with each topic a vocabulary distribution of a word set.

The LDA method can be applied in multiple tasks such as detecting keywords, arranging documentation as well as text generation. In data augmentation, existing training data is again used to increase the data set in size. An LDA is first trained by treating real data, gathering information on the set of topics, and how these topics are distributed over the considered documents. Subsequently, this is sampled to generate new data that is similar to the original data set. For example, the way how consumers write a product review on a specific product is to be investigated. If there is an abundance of product reviews to be found and the investigation demands a larger data collection, LDA can generate that. First, the existing reviews are fed to the LDA, forming a set of topics and a distribution of these topics in each considered review. Then there will be sampled to generate new reviews that are essentially expressing the same conclusions, treating the same topics but in different text D. Blei et al., 2001.

3.2.5. Boltzmann machine

A Boltzmann machine (BM) is a neural network using stochastics as an approach using a random process to decide how to adjust its parameters. A BM consists of a set of connected nodes that can be turned on or off. These so-called neurons are arranged per layer and subsequently, each layer is connected to the following layer by weighted links. The weighting ensures the specific strength of a neuron connection and can be modified for optimal performance.

A Boltzmann machine is an energy-based model. Each possibility of connections throughout the model is given an energy value based on the weights between those connections and the states of the connected neurons. When the energy value is lower, the likelihood of occurrence is higher and vice versa. This resembles physical systems where lower energy indicates stability.

For the learning process of a BM, a Markov chain Monte Carlo (MCMC) is used. This MCMC flips the states of the neurons and calculates the resulting energy changes. Whenever a new configuration has a lower energy than previously, the new state is chosen. Higher energy configuration might also be still applicable but with a lower probability that is lowered as the energy differences become larger. This way the BM learns to adjust the weights of the connections to optimise the best low-energy configuration possible in the considered network. BMs are useful throughout data augmentation and can be applied to complex and high-dimensional data. However, this comes at the expense of cost and time and the model can be, unlike for example the Bayesian network, difficult to understand without proper insight J. Zhang et al., 2019.

3.2.6. Diffusion model

The diffusion model (DM) describes the spread of a distribution through a sampled population. Transformations sequences applied to an initial noise distribution are added to ultimately generate new data. The transformations gradually increasing the noise input to the data is called the diffusion process.

First, the diffusion model is trained using an original data set. The maximum likelihood estimation is applied to find the optimal parameters of the diffusion process. After the DM is accustomed to the training data set, random noise is slowly added. This results in newly generated data that is almost similar but slightly different from the original data points. By repeating this process the data set is ultimately enlarged.

To determine the proportion of added noise in a DM a diffusion coefficient is introduced. This key parameter scales the uncertainty in each step. The higher the coefficient, the higher variability and diverseness and vice versa. The diffusion coefficient can be controlled manually but generally, self-learning is preferred in generative modelling Cao et al., 2022.

3.2.7. Flow-based generative model

flow-based generative model (FM) generates new data using invertible transformation sequences or flows to characterise fed training data. These transformations take relatively simple probability distributions and alter them into complex ones that capture the characteristics of the simple probability distribution's data set.

After the FM is trained in using the applicable invertible transformation sequences, it is then able to generate new data similar to the original data set. This process is repeated until enough new data is generated. Because of the invertibility, an FM is able to deliver exact samples rather than approximations generated by other generative models. Flow-based generative models can produce high-quality samples and can cover the total variation in an original data set into newly generated data points B. Zhang et al., 2022.

3.3. Deep generative models

Generative models that have many complexities, hidden layers, and/or high dimensional inputs for instance, can be referred to as deep generative model (DGM)s. Autoencoders, GANs but also generative pre-trained transformer (GPT)s are examples of this.

3.3.1. Generative pre-trained transformers

One of the more popularised deep generative models at this moment is OpenAI chat interfaces. They are categorised as GPTs and the most commonly used one is called ChatGPT OpenAI, n.d. It runs from a freely accessible website where you can start chats with the chat bot that will react to you using self-generated answers. It learns from the information on the internet combined with experiences on the platform. Below is a transcript about what ChatGPT does, generated by ChatGPT itself:

As an AI language model, ChatGPT works by processing and analyzing large amounts of text data to learn patterns and relationships between words and phrases. The model is based on a deep learning architecture known as a transformer, which is trained using a process called unsupervised learning.

During the training process, ChatGPT is fed with massive amounts of text data and learns to predict the probability distribution of the next word in a sentence given the preceding words. This process allows the model to develop a deep understanding of the structure of natural language and the relationships between words and phrases.

When you interact with ChatGPT, it uses its knowledge and understanding of language to generate a response based on the input you provide. It does this by using algorithms to analyze the input text, identifying the key concepts, and generating a response that is most likely to be relevant and coherent based on its training data.

ChatGPT is continually learning and improving, as it is trained on ever-growing amounts of text data from the internet and other sources. This enables it to generate increasingly accurate and sophisticated responses over time.

ChatGPT or any other kind of generative pre-trained transformer will not be included in the ultimate assessment between generative models for this thesis, as GPTs specialise in text generation whereas a new high-dimensional data generation is the aim.

3.3.2. Autoencoders

The autoencoder (AE)) is a prime example of a DGM. The model is fed in data to compress and encode and subsequently decode it as output. The goal is to have the autoencoder learn from the results, trying to make the difference between output and input as small as possible. Since the input data is without any other information, the AE figures out any patterns or anomalies by itself. Because of this characteristic, autoencoders are often used to ignore white noise and find interesting data in a vast data set. The other way around, autoencoders are also useful for finding disrupting anomalies in a traditional data set.

In Figure 3.3 an autoencoder is visually displayed. An input (a) into the encoder results in a latent vector (b). It is then fed to the decoder, generating a reconstructed input $(\hat{a})$.

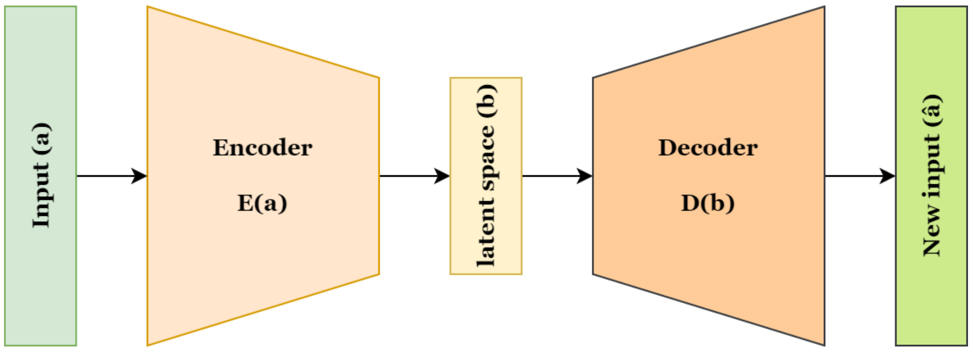


Figure 3.3: Visualisation of an autoencoder

The autoencoder can be used in combination with GANs by producing the input. A GAN could however also be implemented into an encoder by being the decoder part.

3.3.3. Variational autoencoders

A specific type of AE is the variational autoencoder (VAE). Where the AE has latent vectors between the encoder and decoder, the VAE has outputs in parameters that are predefined for every input. Constraining the output in latent space results in a normal distribution and also ensures regulation.

In Figure 3.4 the method of a VAE is visualised, again with an input, encoder, decoder, and reconstructed input as in a normal AE. However, the latent space is now different. Instead, there is now a latent distribution and sampling to construct a latent vector (b), which is then fed to the decoder.

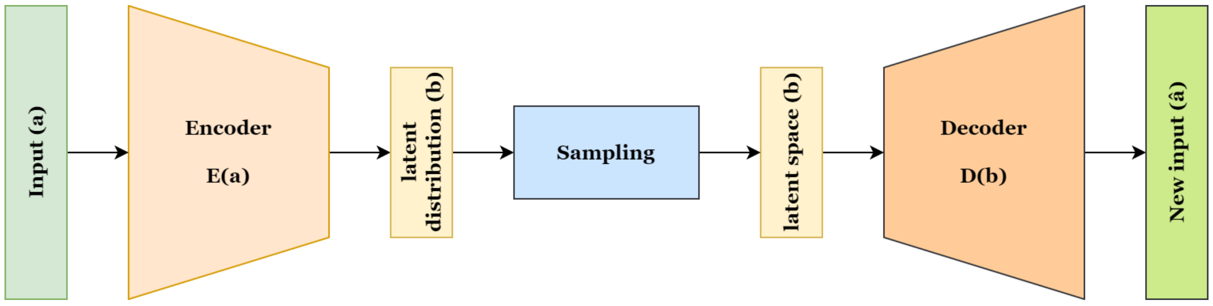


Figure 3.4: Visualisation of a variational autoencoder

3.3.4. Generative adversarial networks

In machine learning, a generative adversarial network (GAN) is used to improve robustness in prognostic modelling. It essentially learns itself to generate new data by the use of (initially) two neural networks. The first is a generative network that produces whilst the second discriminative network judges its authenticity. The generator normally is fed a random input (of Gaussian noise) and will generate a sample. This is added to the training data set sampling and put into the discriminator which then has to value the samples. Its output will then be fed back to both the discriminator as well as the generator to learn. This feedback is respectively the discriminator loss and generator loss. This process is visualised in Figure 3.5.

By iteration, the generative network will produce better data to fool the discriminative network. Reversely, the discriminative network will tailor itself to not be fooled easily by the gained experience. This process should result in new data sets that can be utilised in prognostic models given the right parameters and conditions in which the GAN is executed. The ultimate goal of training the GAN is to make the generator produce samples that are indistinguishable from the real data. Once the model is trained, the generator can be used to generate new samples that can be added to enlarge the data collection.

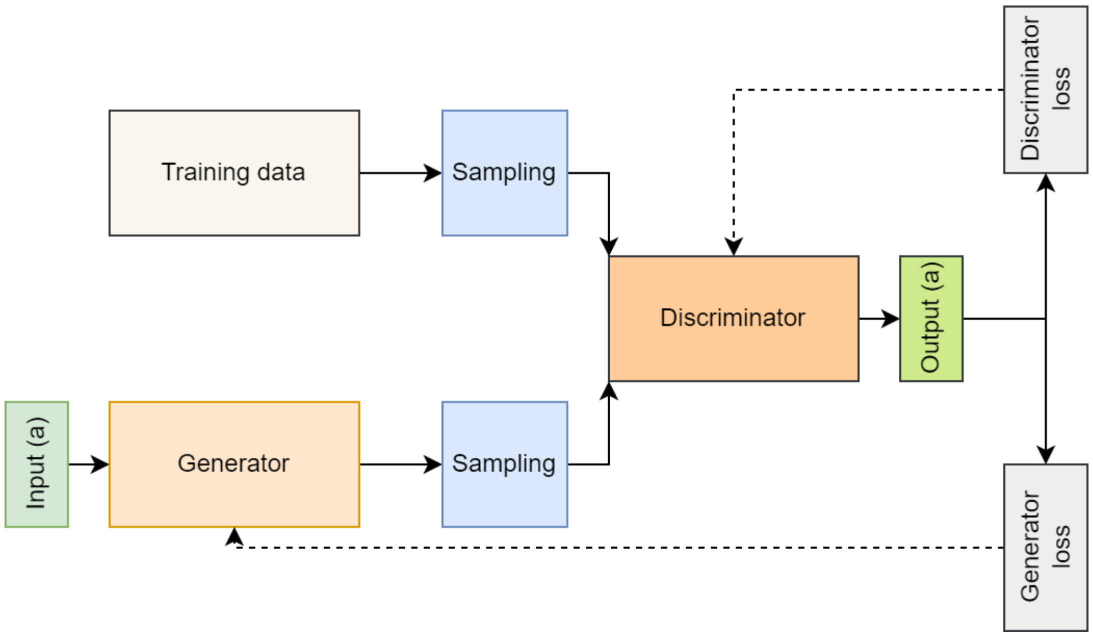


Figure 3.5: Visualisation of a GAN

GANs are used in several applications, not only in engineering but also in finance, the medical world de Farias et al., 2021, and others. They have promising results in generating new images, of non-existing persons for instance, and can be applied in other industries such as video game development.

3.4. Overview of different generative adversarial networks

There are different GAN models with different architectures that serve distinctive purposes. Firstly, there are **fully connected GANs**. The discriminator as well as the generator utilises neural networks that are not constrained in connections. This is one of the least complex architectures and is applied to plain data sets in particular. The Vanilla GAN is a well-known fully connected GAN Creswell et al., 2018.

There are also **conditional GAN (CGAN)s**. Extra information, descriptions, and labels are fed to a GAN which makes it conditioned. These conditions can be forced upon the generator as an input together with the noise input, upon the discriminator as extra help for identifying and classifying samples, or upon both. A CGAN works relatively well with image-to-image translation, text-to-image generation, and video synthesis. A well-known example of a CGAN specialising in image-to-image translation is the CycleGAN Creswell et al., 2018. Generation of images can also be performed by **Laplacian pyramid of adver-**

sarial network (LAPGAN). It uses a Laplacian pyramid framework and builds a series of generative models using upsampling. This is also a convolutional model Alqahtani et al., 2021.

Furthermore, there is a type of **convolution GANs**. This works the same as fully connected GANs but now by using convolutional neural networks in both the generator and discriminator to generate and rate samples. Convolutional GANs are specialised in image data sets, training themselves by producing a layer above another. A special type is the **deep convolutional GAN (DCGAN)**, which uses down-sampling and up-sampling spatially and uses deep learning. The DCGAN characteristics in Table 3.2 will represent the standard convolution GAN too Creswell et al., 2018.

Another type is the **GAN with interference model**. These are GANs with a built-in mechanism for interference to improve the data quality of the output. The interference lets the generator focus on specific input data elements. A well-known interference GAN is the attention GAN (AttnGAN). The interference comes in the form of an attention mechanism to generate images from descriptions in writing. The attention mechanism is multi-level and focuses on different parts of the input text during the process of image generation. The generator produces more realistic images that are consistent with the input description when being helped with focusing on relevant parts of the data. This GAN will not be compared in Table 3.2 as the interference model is out of scope.

Next, there is the method of an **adversarial autoencoder (AAE)** that utilises GAN for variational interference by finding similarities between hidden code vectors Alqahtani et al., 2021. The AAE is normalised and both the adversarial network as well as the autoencoder are trained together in two phases. First, a reconstruction phase is followed by a regularisation phase. Another GAN is the **generative recurrent adversarial network (GRAN)**. Now the encoder is a convolutional network that extracts images of the current computation, whereas the decoder will decide how to update it. At every step of time, a sample from the prior distribution state is saved in a hidden state. With all samples, the final sample can be composed of the hidden state that acts as a decoder Alqahtani et al., 2021.

Also, the **information maximising generative adversarial network (InfoGAN)** can be used to learn features unsupervised. The objective is to learn representations by maximising the information of a noise subset and observations. A domain is semantically decomposed while considering the features of the data. InfoGANs decompose the input by the same approach, resulting in a latent vector and a noise vector. Then by regularisation, the information found is maximised. Lastly, the **bidirectional generative adversarial network (BiGAN)** is a method that uses feature representation of projected data in latent space by inverse mapping. Here, the objective is learning to invert the generator.

Table 3.2: Different GANs

GANs	Learning	Network	Gradient	Objective	Performance
Fully connected	Supervised	Multilayer	Step	Minimise	Log-likelihood
Conditional	Supervised	Multilayer	Step	Minimise	Log-likelihood
Laplacian	Unsupervised	Laplacian pyramid	None	Image generation	Log-likelihood
Deep convolutional	Unsupervised	Convolutional networks	Step	Hierarchy learning	Accuracy
Adversarial AE	Mixed	Autoencoders	Step	Matching hidden code	Log-likelihood
Recurrent	Supervised	Recurrent convolutional networks	Step	Image generation	Adversarial metric
Info	Unsupervised	Multilayer	Step	Maximising	Information metric
Bidirectional	Mixed	Deep multilayer neural networks	None	Feature learning	Accuracy

Apart from the aforementioned types there are many more GANs widely available and developed constantly during the last decade. However, for now, the most common and least complicated GAN will be used in further investigating as it is appropriate for the research objective in chapter 5. This would be the fully connected GAN and specifically the Vanilla GAN.

4 Usage of generative models

In this section, the general history of the generative models treated in section 3.2 and section 3.3 is described. Furthermore, the usage of DGMs in PHM is described. Finally, a comparison between the different methods is included in deciding which ones are most suitable to support the research objective.

4.1. Brief history of generative models

Though many of the considered generative models bear the names of mathematicians and statisticians from the 1700s and 1800s, it is generally recognised that the first work of artificial intelligence was accomplished in 1943 Russell, 2010. The article *A logical calculus of the ideas immanent in nervous activity* by Warren S. McCulloch and Walter Pitts proposed a model of artificial neurons McCulloch and Pitts, 1943. Then the term ”machine learning” was first used in 1959 by Arthur Samuel in *The IBM journal of research and development* Samuel, 1959. Since then new methods using theorems, laws, and rules of Bayes, Dirichlet, Markov, and many others have emerged to help generate reliable augmented data.

In 1966 Baum and Petrie published *Statistical inference for probabilistic functions of finite state Markov chains*, in which the hidden Markov model and associated algorithms such as the forward-backward algorithm were developed Baum and Petrie, 1966. The next generative model discussed in section 3.2 is the Gaussian mixture model. That method was introduced by Duda and Hart in 1973, exploring the possibilities of combining Gaussian distributions Duda, Hart, et al., 1973. Furthermore, diffusion models were developed in the 1970s, as well. The article *A theory of memory retrieval* by Ratcliff discusses the use of a diffusion process to accomplish the retrieval of memories Ratcliff, 1978.

In the 1980s Bayesian networks were explored, first named by Judea Pearl in his article *Bayesian networks: A model of self-activated memory for evidential reasoning* from 1985 Pearl, 1985. Also, the Boltzmann machine as a generative model is developed in 1985. An article called *A Learning Algorithm for Boltzmann Machines* researches a general parallel search method to apply a learning algorithm for BMs Ackley et al., 1985. In 1991 a paper by Kramer was published in *AIChE journal* that proposed the autoencoder as a nonlinear generalisation of principal components analysis (PCA) for the first time, that would expand to be a deep learning generative model during the next decades Kramer, 1991.

The next considered generative model to be developed is latent Dirichlet allocation. In 2003, LDA was proposed by Blei in the *Journal of Machine Learning Research* D. M. Blei et al., 2003. It describes LDA as a three-level hierarchical Bayesian model and presents early results. Then, the flow-based generative model was developed en published by Dinh et al. during the proposition of non-linear independent components estimation (NICE) in 2014, a deep learning framework Dinh et al., 2014. The important feature here is that individual invertible transformations were stacked to compose a flow model Ho et al., 2019.

At this point, the VAE was just invented at the end of 2013 by Kingma and Welling Kingma and Welling, 2013. Together with the development and description of the first use of GAN six months later by Goodfellow, the field of generative modelling became revolutionised Goodfellow et al., 2014. In the years thereafter GAN models were introduced rapidly such as the deep convolutional GAN and the conditional GAN for image-to-image translation CycleGAN, in 2015 and 2017 respectively Radford et al., 2015 J.-Y. Zhu et al., 2017. In the following period, transformers became prominent enabling the building of larger models. Notably, language models such as the generative pre-trained transformer, text-to-text transfer transformer (T5), and the use of the spatial transformer GAN (ST-GAN) are developed in 2018 Radford et al., 2018 Xue et al., 2020 Lin et al., 2018. In Figure 4.1 a timeline of proposed GAN methods is shown.

In recent years the field of deep generative modelling saw new approaches proposed, including merging of different DGMs. For instance, the vision transformer (VT) utilises mostly language-used transformers for image generation, a theory that was first published in 2020 Dosovitskiy et al., 2020. Also, the DM method was incorporated by two large models for large image generation during the same year, namely the denoising diffusion probabilistic model (DDPM) and the denoising diffusion implicit model (DDIM) Ho et al., 2020 Song et al., 2020.

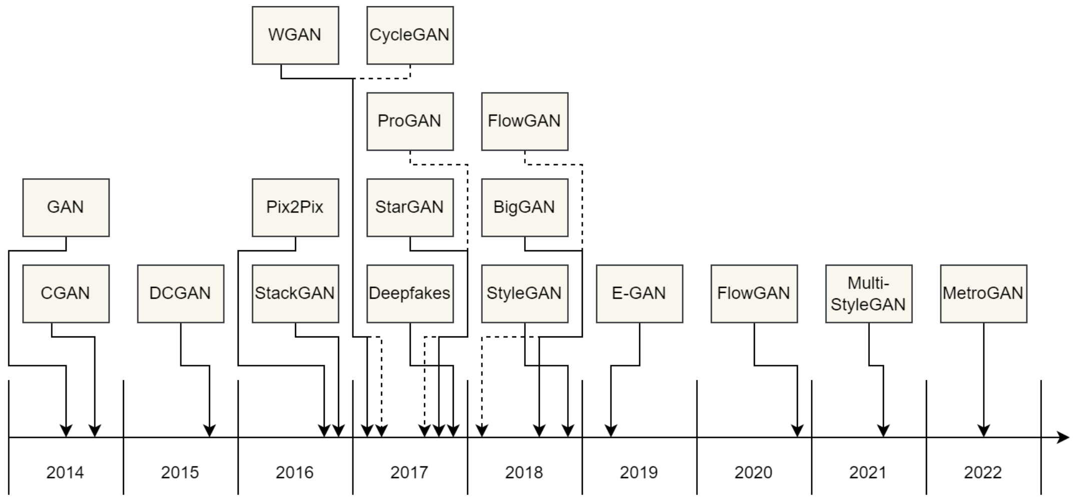


Figure 4.1: Timeline of last decade concerning GANs

Today, improvements in large language models are released. In particular, GPT-4 by OpenAI, PaLM by Google, and OPT and LLaMA by Meta are becoming popular among consumers at the beginning of 2023 OpenAI, 2023 Chowdhery et al., 2022 S. Zhang et al., 2022 Touvron et al., 2023. Meanwhile, latent diffusion was explored by combining the diffusion model method with the autoencoder's latent space. This resulted in a latent diffusion model (LDM), the Stable Diffusion by Stability AI Rombach et al., 2022a. This model is deployable for both image to image generation (txt2img) as well as image to image generation (img2img) Rombach et al., 2022b.

4.2. Generative models in PHM

Today, there are already several examples of generative modelling used in prognostics and health management (PHM). For instance, the deep convolutional GAN is applied in time-series regeneration to estimate remaining useful life better X. Zhang et al., 2020. The degradation and thus RUL of batteries can also be managed by using deep learning generative modelling such as GANs, suggested in the article *Prognostics and health management of Lithium-ion battery using deep learning methods: A review* by Zhang and Li Y. Zhang and Li, 2022. Also, a variational autoencoder could help find the RUL and EoL, using a long short-term memory (LSTM) network and GMM to provide probabilistic predictions Huang et al., 2021.

Another main purpose in PHM for which a deep generative model (DGM) can be implemented is the generation of missing data. This is accurately described in the paper *Reliable machine prognostic health management in the presence of missing data*, exploring the use of GANs in combination with VAEs to model the irregularity of an incomplete time-series Huang et al., 2020. Furthermore, PHM also can concern anomaly detection. This can be performed through several techniques that include generative methods. For example, AEs with LSTM have been researched as well as VAEs and the use of GANs Basora et al., 2019.

Air traffic trajectory prediction models are also developed by generative modelling. The wind and weather predictions can be provided by approaches such as the CGAN, as suggested in Pang and Liu, 2020. In this paper, multiple set-ups are proposed that produce better results than conventionally obtained. Another example is the usage of DGMs in rolling element bearing PHM. In the article *A comprehensive review of artificial intelligence-based approaches for rolling element bearing PHM: shallow and deep learning* multiple approaches are reviewed, including the benefits of using deep learning methods under which a DCGAN, an AE, and other GANs Hamadache et al., 2019.

Lastly, the problem of detecting coating breakdown and corrosion (CBC) for aircraft is taken as an example that can be treated by a deep generative model. Fink et al. describe several directions for deep learning in an article from 2020 Fink et al., 2020, in which GMMs, GANs, VAEs, the use of LSTM and much more are discussed to find an aircraft's coating breakdown and corrosion.

4.3. Comparison of treated generative methods

All considered generative models are now compared to each other to inquire about which models are best suited to help answer the questions stated in chapter 6. In Table 4.1 several common characteristics are listed in the first column with all treated generative models next to one another in the first row. They are named by their abbreviations mentioned in section 3.2 and section 3.3 that can also be found in the List of acronyms. When an x is placed in the table, that means that that specific generative model handles that characteristic relatively better than average.

First, the preferred type of data input is covered. Next, the easiness to train and quickly start using the generative model is compared to extensive training but with more diverse results. Furthermore, specific characteristics that can belong to one or almost all generative models are described such as being able to handle high-dimensional data or computational cost.

Table 4.1: Comparison of generative models

Characteristics:	GMM	HMM	BN	LDA	BM	DM	FM	VAE	AE	GAN
Discrete data		□		□						
Continuous data	□					□	□			□
Mixed data			□		□			□	□	
Easy to train	□	□	□	□						
Extensive training					□	□	□	□	□	□
Complex variable relationships			□				□	□		
Causality		□	□							
High-dimensional data				□	□		□	□	□	□
Clustering	□	□		□				□		
Large data sets								□	□	□
Visualisation		□	□				□	□	□	□
Computational cost					□	□	□	□		□
Handling of missing data			□	□				□	□	□
Handling of noise			□	□	□		□	□	□	□

During the analysis of the considered generative models, it became apparent that especially DGMs are promising in data generation for complex conditioned systems and/or data sets. While the other generative models have their benefits and might be preferred in a specific field of expertise, the VAE, AE, and GAN can be especially helpful during this thesis.

The two most versatile DGMs will be compared to each other in the same test setup. Between the (variational) autoencoder and generative adversarial network, the AE seems the least widely applicable in advance. Therefore it was concluded that the VAE and GAN will be used for further research. As described in subsection 3.3.4, there are many types of GANs to choose from. At first, the most standard and commonly used GAN will be utilised, the Vanilla GAN. Also, the TGAN will initially be investigated concerning time-series data sets. One of the aforementioned standard generative models is chosen to be compared to the DGMs. Due to its ability to have both discrete and continuous data as input, to handle missing data and noise, and its versatility overall, the Bayesian network (BN) is chosen at this stage.

5 Approach

With the problem statement defined and the literature review as preparation, this section gives an outline of how to answer the raised research questions and in which time frame, respectively called the Methodology and Planning.

5.1. Methodology

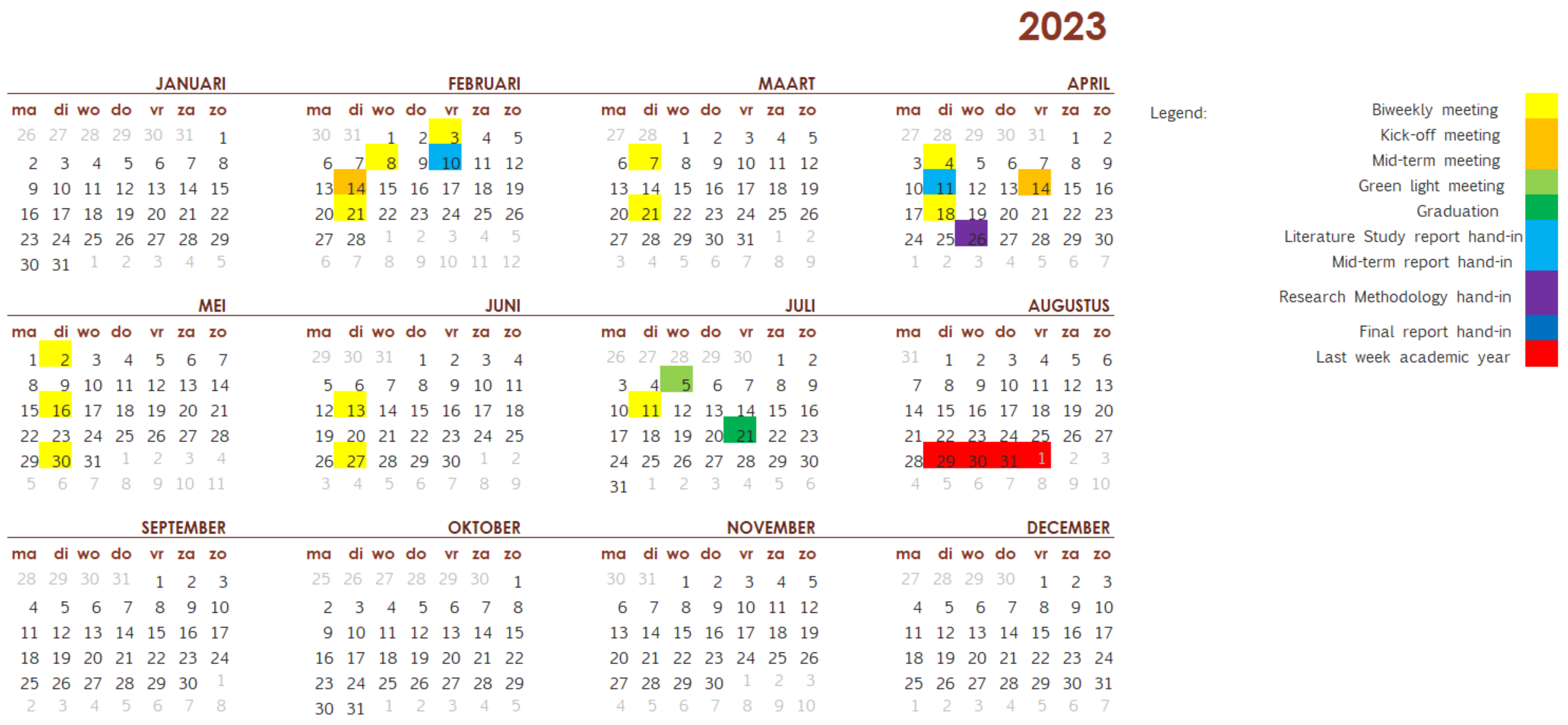
In conjunction with Research Methodologies

5.2. Planning

The calendar shows the important dates in the next 6 months to keep on track. Below the most crucial ones are listed. Also, a Gantt Chart is provided to show the schedule of the thesis. It consists of multiple tasks that have to be finished, both dependent on or apart from each other.

- **January 2023** Start Thesis
- **March 2023** Kick-off
- **April 2023** Research Methodology
- **May 2023** Mid-term
- **July 2023** Green Light

5.2.1. Calendar



2023

JANUARI

ma	di	wo	do	vr	za	zo
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

FEBRUARI

ma	di	wo	do	vr	za	zo
30	31	1	2	🟨 3	4	5
6	7	🟨 8	9	🟦 10	11	12
13	🟧 14	15	16	17	18	19
20	🟨 21	22	23	24	25	26
27	28	1	2	3	4	5
6	7	8	9	10	11	12

MAART

ma	di	wo	do	vr	za	zo
27	28	1	2	3	4	5
6	🟨 7	8	9	10	11	12
13	14	15	16	17	18	19
20	🟨 21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

APRIL

ma	di	wo	do	vr	za	zo
27	28	29	30	31	1	2
3	🟨 4	5	6	7	8	9
10	🟦 11	12	13	🟧 14	15	16
17	🟨 18	19	20	21	22	23
24	25	🟪 26	27	28	29	30
1	2	3	4	5	6	7

MEI

ma	di	wo	do	vr	za	zo
1	🟨 2	3	4	5	6	7
8	9	10	11	12	13	14
15	🟨 16	17	18	19	20	21
22	23	24	25	26	27	28
29	🟨 30	31	1	2	3	4
5	6	7	8	9	10	11

JUNI

ma	di	wo	do	vr	za	zo
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	🟨 13	14	15	16	17	18
19	20	21	22	23	24	25
26	🟨 27	28	29	30	1	2
3	4	5	6	7	8	9

JULI

ma	di	wo	do	vr	za	zo
26	27	28	29	30	1	2
3	4	🟩 5	6	7	8	9
10	🟨 11	12	13	14	15	16
17	18	19	20	🟩 21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

AUGUSTUS

ma	di	wo	do	vr	za	zo
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	🟥 29	🟥 30	🟥 31	🟥 1	2	3
4	5	6	7	8	9	10

SEPTEMBER

ma	di	wo	do	vr	za	zo
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

OKTOBER

ma	di	wo	do	vr	za	zo
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

NOVEMBER

ma	di	wo	do	vr	za	zo
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

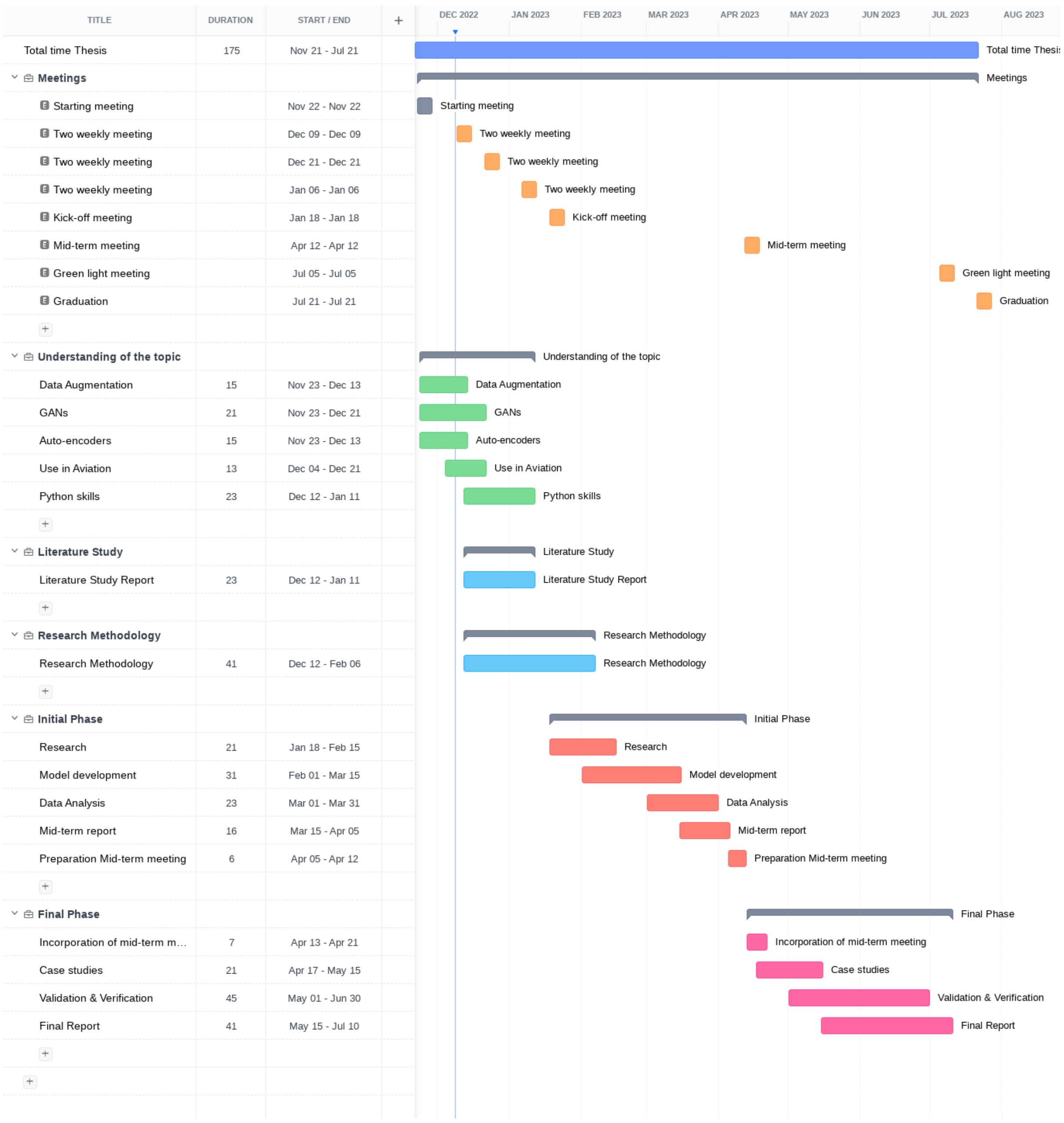
DECEMBER

ma	di	wo	do	vr	za	zo
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Legend:

Biweekly meeting	🟨
Kick-off meeting	🟧
Mid-term meeting	🟧
Green light meeting	🟩
Graduation	🟩
Literature Study report hand-in	🟦
Mid-term report hand-in	🟦
Research Methodology hand-in	🟪
Final report hand-in	🟦
Last week academic year	🟥

Figure 5.1: Calendar for the upcoming 6-7 months



TITLE	DURATION	START / END
Total time Thesis	175	Nov 21 - Jul 21
Meetings		
Starting meeting		Nov 22 - Nov 22
Two weekly meeting		Dec 09 - Dec 09
Two weekly meeting		Dec 21 - Dec 21
Two weekly meeting		Jan 06 - Jan 06
Kick-off meeting		Jan 18 - Jan 18
Mid-term meeting		Apr 12 - Apr 12
Green light meeting		Jul 05 - Jul 05
Graduation		Jul 21 - Jul 21
Understanding of the topic		
Data Augmentation	15	Nov 23 - Dec 13
GANs	21	Nov 23 - Dec 21
Auto-encoders	15	Nov 23 - Dec 13
Use in Aviation	13	Dec 04 - Dec 21
Python skills	23	Dec 12 - Jan 11
Literature Study		
Literature Study Report	23	Dec 12 - Jan 11
Research Methodology		
Research Methodology	41	Dec 12 - Feb 06
Initial Phase		
Research	21	Jan 18 - Feb 15
Model development	31	Feb 01 - Mar 15
Data Analysis	23	Mar 01 - Mar 31
Mid-term report	16	Mar 15 - Apr 05
Preparation Mid-term meeting	6	Apr 05 - Apr 12
Final Phase		
Incorporation of mid-term m…	7	Apr 13 - Apr 21
Case studies	21	Apr 17 - May 15
Validation & Verification	45	May 01 - Jun 30
Final Report	41	May 15 - Jul 10

Figure 5.2: Gantt Chart: all tasks unfolded

6 Problem statement

In this section, the problem is once more stated, including the motivation to research this specifically in section 6.1. Furthermore, the objective of this research will be explained along with the respective research questions to be answered during this thesis.

6.1. Research gap

In PHM, data augmentation can be crucial to elevate one of the most promising types of PHM. However, data augmentation by unsupervised learning is sometimes not researched in-depth, also because of new technologies and various novel methods emerging rapidly today.

There is not always consensus on what kind of data augmentation is most suitable for specific scenarios in the aviation industry. In maintenance as a whole, there might be some insight on when to use what. However, for specific situations, different techniques should be compared. Especially the use of deep generative models is new territory but has promising preliminary results.

The lack of information on how to utilise DGMs and the comparison of their respective results in aviation maintenance is a gap that needs more research to establish what type of model can accurately generate augmented data for specific scenarios.

6.2. Research objective

One of the objectives of this thesis is to provide more insight into how to use data augmentation in such a way as to better fulfil the prediction of failure in aviation.

To research this, different deep generative models must be explored on how to use them for generating new degradation data. Two of these DGMs are considered: the generative adversarial network & the variational autoencoder. Publicly available training data combined with newly generated input by the said DGMs are compared to further investigate the benefits of maintaining PHM.

Publicly available training data input that can be used by these models. The following datasets can be found free of charge:

- turbofan engine degradation Orzech, 2022
- bearings, roll or mill Lu et al., 2021
- aviation maintenance data sets Yang and Desell, 2022

A road map must be established labelling the chosen different (deep) generative models (GAN, AE, BN) on how to use them for specific prognostics problems, including pros and cons measured in the modelling prediction results. This will include a thorough analysis and discussion. In order to compare and rate the aforementioned generative models, their newly generated data will be fed to a standard predictor (i.e. a KNN Imandoust, Bolandraftar, et al., 2013 or a BPN Erb, 1993).

Different types of GANs must also be reviewed and weighed. Also, their competitiveness with other DGMs will have to be proved. In the wake of answering this question, the thesis paper will give more insight into the use of DGMs in the field of aeronautics, hopefully narrowing the existing research gap in this relatively new chapter of science.

Lastly, considerations for future work will be discussed as well as some recommendations for potential following projects.

6.3. Research question

To be able to achieve the objectives stated in the aforementioned section, the following research question is formulated:

- **When using the data set in a turbofan case study, how could the data be improved using generative techniques?**

6.3.1. Sub-questions

A couple of sub-questions are formulated to support the research question, including auxiliary questions. First altering the used models is addressed:

- What is the best method for data generation for turbofan prognostics (VAE, GAN, temporal GAN (TGAN), BN) using the same test setup?
  - To define best, do health monitoring trajectories need different weighting?
  - When comparing different scenarios for the same training data set of a turbofan case study, does the outcome of the best method change?
- How does the accuracy of the prognostics change with the hyperparameter changes of the generative network model?
  - What hyperparameters are most impactful?
  - Is the accuracy change different per investigated generative method when the hyperparameters are changed identically?

Secondly, questions concerning the data input are as follows:

- How does input noise play a role in obtaining better quality data with generative models?
  - Is the impact of Gaussian noise different per generative method?
  - If noise would not be normally distributed, does that change the scoring of the investigated methods?
- What is the minimum volume of input data needed to generate quality output data with generative models?
  - Is there a correlation or even causation between the volume of input data and the accuracy of the prognostics?
  - Is there a difference in the minimum volume of input data per treated generative method, and if so, how does this difference occur?

Lastly, a question regarding the future of DGMs in PHM is considered:

- Are the chosen GAN and VAE applicable and reliable enough to be a promising approach for PHM in a turbofan case study?
  - Does this specific turbofan case study stand for general turbofan case studies?
  - Is it observable at this moment already that testing with data from DGMs will outpace traditional testing by quantifiable measures?

7 Conclusion

Maintenance is a vital part of almost every product's life. Products in the aviation industry are prone to be maintained throughout their useful life to last them longer, be safer and cause no more downtime than necessary. Maintenance can roughly be categorised into three types, being corrective, preventive, and predictive maintenance. It is predictive maintenance that might be best suited to fulfil the goals set in the aviation industry. A well-known method of predictive maintenance is prognostics and health management. It monitors the health and tries to predict failure to plan maintenance beforehand. This is managed by health monitoring trajectories. These trajectories are based on several features including monotonicity, trendability, and prognosability.

However, prognostics and health management cannot deliver adequate results if the data it bases its estimations on is biased or missing. Therefore data quality is crucial. Several obstacles might cause quality loss, such as insufficient sensor data, missing data, economic restrictions, and lack of desired quality. A method that can help fill in the gap for insufficient data in prognostics and health management is called data augmentation. This method uses existing data and augments it at such a level that it can be used as data for a prognostic model while producing effective results.

Data augmentation can be categorised into the following types. Statistical data augmentation, experimental data augmentation, data generation, adaptive augmentation, and high-fidelity simulation data training. All types have augmented data as output. However, the most forward type of producing actual new realistic data that can be added to the already existing data collection is data generation.

Data generation is commonly performed by all kinds of generative models. Several of the most well-known generative models were covered and accompanied by deep generative models. After consideration and comparison, it was concluded that the deep generative models would be more suitable for the research goal of this thesis. The variational autoencoder and generative adversarial network were chosen to act as subjects for realistically generating data.

The objective of the thesis is to provide more insight into how data augmentation can be used better to optimise the prediction of failure in prognostics and health management. The chosen deep generative models will be installed in a test setup and during each test, a different variable will be changed. That can be the noise input for the generators, the existing training data set, the change of GAN type, or the change of predictor.

Also, as a goal for the coming thesis, obstacles of today for the use of this specific type of data augmentation in prognostics and health management are reviewed to assess whether this new approach is promising for aviation products in the nearby future. The thesis including the necessary conditions is set to be finished during the summer of 2023.

Bibliography

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive science*, *9*(1), 147–169.

Alqahtani, H., Kavakli-Thorne, M., & Kumar, G. (2021). Applications of generative adversarial networks (gans): An updated review. *Archives of Computational Methods in Engineering*, *28*, 525–552.

Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. https://doi.org/10.48550/ARXIV.2003.05991

Baptista, M., Goebel, K., & Henriques, E. (2022). Relation between prognostics predictor evaluation metrics and local interpretability shap values. *Artificial Intelligence*, *306*, 103667. https://doi.org/10.1016/j.artint.2022.103667

Baraldi, P., Bonfanti, G., & Zio, E. (2018). Differential evolution-based multi-objective optimization for the definition of a health indicator for fault diagnostics and prognostics. *Mechanical Systems and Signal Processing*, *102*, 382–400. https://doi.org/https://doi.org/10.1016/j.ymssp.2017.09.013

Basora, L., Olive, X., & Dubot, T. (2019). Recent advances in anomaly detection methods applied to aviation. *Aerospace*, *6*(11), 117.

Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, *37*(6), 1554–1563.

Blei, D., Ng, A., & Jordan, M. (2001). Latent dirichlet allocation. In T. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems* (Vol. 14). MIT Press. https://proceedings.neurips.cc/paper/2001/file/296472c9542ad4d4788d543508116cbc-Paper.pdf

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Cao, H., Tan, C., Gao, Z., Chen, G., Heng, P.-A., & Li, S. Z. (2022). A survey on generative diffusion model. *arXiv e-prints*. https://doi.org/10.48550/ARXIV.2209.02646

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2022). Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Coble, J., & Hines, J. W. (2009). Identifying optimal prognostic parameters from data: A genetic algorithms approach. *Annual Conference of the PHM Society*, *1*.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, *35*(1), 53–65. https://doi.org/10.1109/MSP.2017.2765202

de Farias, E. C., di Noia, C., Han, C., Sala, E., Castelli, M., & Rundo, L. (2021). Impact of gan-based lesion-focused medical image super-resolution on the robustness of radiomic features. *Scientific Reports*, *11*(1).

Deng, L., Yu, D., et al. (2014). Deep learning: Methods and applications. *Foundations and trends® in signal processing*, *7*(3–4), 197–387.

Dinh, L., Krueger, D., & Bengio, Y. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Duda, R. O., Hart, P. E., et al. (1973). *Pattern classification and scene analysis* (Vol. 3). Wiley New York.

Duffuaa, S. O., & Ben□Daya, M. (1995). Improving maintenance quality using spc tools. *Journal of Quality in Maintenance Engineering*, *1*(2), 25–33. https://doi.org/10.1108/13552519510089565

Eckert, D., Vesal, S., Ritschl, L., Kappler, S., & Maier, A. (2020). Deep learning-based denoising of mammographic images using physics-driven data augmentation. *Bildverarbeitung für die Medizin 2020: Algorithmen–Systeme–Anwendungen. Proceedings des Workshops vom 15. bis 17. März 2020 in Berlin*, 94–100.

Eddy, S. R. (2004). What is a hidden markov model? *Nature Biotechnology*, *22*(10), 1315–1316. https://doi.org/10.1038/nbt1004-1315

Erb, R. J. (1993). Introduction to backpropagation neural network computation. *Pharmaceutical research*, *10*, 165–170.

Fink, O., Wang, Q., Svensen, M., Dersin, P., Lee, W.-J., & Ducoffe, M. (2020). Potential, challenges and future directions for deep learning in prognostics and health management applications. *Engineering Applications of Artificial Intelligence*, *92*, 103678.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *stat*, *1050*, 10.

Hamadache, M., Jung, J. H., Park, J., & Youn, B. D. (2019). A comprehensive review of artificial intelligence-based approaches for rolling element bearing phm: Shallow and deep learning. *JMST Advances*, *1*, 125–151.

Han, M., Wang, Z., & Zhang, X. (2021). An approach to data acquisition for urban building energy modeling using a gaussian mixture model and expectation-maximization algorithm. *Buildings*, *11*(1). https://doi.org/10.3390/buildings11010030

Ho, J., Chen, X., Srinivas, A., Duan, Y., & Abbeel, P. (2019). Flow++: Improving flow-based generative models with variational dequantization and architecture design. *International Conference on Machine Learning*, 2722–2730.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 6840–6851, Vol. 33). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf

Huang, Y., Tang, Y., & VanZwieten, J. (2021). Prognostics with variational autoencoder by generative adversarial learning. *IEEE Transactions on Industrial Electronics*, *69*(1), 856–867.

Huang, Y., Tang, Y., VanZwieten, J., & Liu, J. (2020). Reliable machine prognostic health management in the presence of missing data. *Concurrency and Computation: Practice and Experience*, *34*(12), e5762.

IBM. (2023). What is preventive maintenance. https://www.ibm.com/topics/what-is-preventive-maintenance

Imandoust, S. B., Bolandraftar, M., et al. (2013). Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. *International journal of engineering research and applications*, *3*(5), 605–610.

Jain, A., Patel, H., Nagalapatti, L., Gupta, N., Mehta, S., Guttula, S., Mujumdar, S., Afzal, S., Sharma Mittal, R., & Munigala, V. (2020). Overview and importance of data quality for machine learning tasks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 3561–3562.

Kim, N.-H., An, D., & Choi, J.-H. (2017). *Prognostics and health management of engineering systems*. Springer.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, *37*(2), 233–243.

Lesage, A., & Dehombreux, P. (2012). Maintenance & quality control: A first methodological approach for maintenance policy optimization. *IFAC Proceedings Volumes*, *45*(6), 1041–1046.

Lim, D., Jung, W., Bae, J., & Park, Y. (2022). Utilization of high-fidelity simulation data for data augmentation of artificial neural net-based rotor faults diagnosis. In J.-H. Han, S. Shahab, & J.

Yang (Eds.), *Active and passive smart structures and integrated systems xvi* (120431A). SPIE. https://doi.org/10.1117/12.2612314

Lin, C.-H., Yumer, E., Wang, O., Shechtman, E., & Lucey, S. (2018). St-gan: Spatial transformer generative adversarial networks for image compositing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9455–9464.

Lu, H., Barzegar, V., Nemani, V. P., Hu, C., Laflamme, S., & Zimmerman, A. T. (2021). Gan-lstm predictor for failure prognostics of rolling element bearings. *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 1–8. https://doi.org/10.1109/ICPHM51084.2021.9486650

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*, 115–133.

OpenAI. (n.d.). Introducing chatgpt. https://openai.com/blog/chatgpt

OpenAI. (2023). Gpt-4 technical report.

Orzech, G. (2022, July). Prognostics center of excellence data set repository. https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository

Pang, Y., & Liu, Y. (2020). Conditional generative adversarial networks (cgan) for aircraft trajectory prediction considering weather effects. *AIAA Scitech 2020 Forum*, 1853.

Pearl, J. (1985). Bayesian netwcrks: A model of self-activated memory for evidential reasoning. *Proceedings of the 7th conference of the Cognitive Science Society, University of California, Irvine, CA, USA*, 15–17.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological review*, *85*(2), 59.

Reynolds, D. A., et al. (2009). Gaussian mixture models. *Encyclopedia of biometrics*, *741*(659-663).

Rigamonti, M., Baraldi, P., Zio, E., Roychoudhury, I., Goebel, K., & Poll, S. (2016). Echo state network for the remaining useful life prediction of a turbofan engine. *PHM Society European Conference*. http://www.papers.phmsociety.org/index.php/phme/article/view/1623

Rodrigues, L. R., Yoneyama, T., & Nascimento, C. L. (2012). How aircraft operators can benefit from phm techniques. *2012 IEEE Aerospace Conference*, 1–8. https://doi.org/10.1109/AERO.2012.6187376

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022a). High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10684–10695.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022b, November). Compvis/stable-diffusion: A latent text-to-image diffusion model. https://github.com/CompVis/stable-diffusion

Russell, S. J. (2010). *Artificial intelligence a modern approach*. Pearson Education, Inc.

Saltoglu, R., Humaira, N., & Inalhan, G. (2016). Scheduled maintenance and downtime cost in aircraft maintenance management. *International Journal of Aerospace and Mechanical Engineering*, *10*(3), 602–607. https://publications.waset.org/vol/111

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210–229.

Schwabacher, M. (2005). A survey of data-driven prognostics. *Infotech Aerospace*, 7002. https://doi.org/10.2514/6.2005-7002

Shi, P., Qi, Q., Qin, Y., Scott, P., & Jiang, X. (2020). A novel learning-based feature recognition method using multiple sectional view representation. *Journal of Intelligent Manufacturing*, *31*, 1291–1309. https://doi.org/10.1007/s10845-020-01533-w

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, *6*(1), 1–48.

Song, J., Meng, C., & Ermon, S. (2020). Denoising diffusion implicit models. *International Conference on Learning Representations*.

Stephenson, T. A. (2000). *An introduction to bayesian network theory and usage* (tech. rep.). Idiap.

ToolSense. (2023). The 6 types of maintenance. https://toolsense.io/maintenance/the-6-types-of-maintenance-definitions-benefits-examples/

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Van Dyk, D. A., & Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, *10*(1), 1–50.

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016). Understanding data augmentation for classification: When to warp? *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 1–6. https://doi.org/10.1109/DICTA.2016.7797091

Xie, Q., Dai, Z., Hovy, E., Luong, T., & Le, Q. (2020). Unsupervised data augmentation for consistency training. *Advances in neural information processing systems*, *33*, 6256–6268.

Xu, X., Zhao, H., & Torr, P. (2022). Universal adaptive data augmentation. *arXiv preprint arXiv:2207.06658*. https://doi.org/10.48550/ARXIV.2207.06658

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2020). Mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Yang, H., & Desell, T. (2022). A large-scale annotated multivariate time series aviation maintenance dataset from the ngafid. *arXiv e-prints*. https://doi.org/10.48550/ARXIV.2210.07317

Zhang, B., Wang, T., Zhou, C., Conci, N., & Liu, H. (2022). Human trajectory forecasting using a flow-based generative model. *Engineering Applications of Artificial Intelligence*, *115*, 105236. https://doi.org/https://doi.org/10.1016/j.engappai.2022.105236

Zhang, C., & Baan, M. (2021). Complete and representative training of neural networks: A generalization study using double noise injection and natural images. *GEOPHYSICS*, *86*, 1–43. https://doi.org/10.1190/geo2020-0193.1

Zhang, J., Wang, H., Chu, J., Huang, S., Li, T., & Zhao, Q. (2019). Improved gaussian–bernoulli restricted boltzmann machine for learning discriminative representations. *Knowledge-Based Systems*, *185*, 104911. https://doi.org/https://doi.org/10.1016/j.knosys.2019.104911

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022). Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhang, X., Qin, Y., Yuen, C., Jayasinghe, L., & Liu, X. (2020). Time-series regeneration with convolutional recurrent generative adversarial network for remaining useful life estimation. *IEEE Transactions on Industrial Informatics*, *17*(10), 6820–6831.

Zhang, Y., & Li, Y.-F. (2022). Prognostics and health management of lithium-ion battery using deep learning methods: A review. *Renewable and Sustainable Energy Reviews*, *161*.

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proceedings of the IEEE international conference on computer vision*, 2223–2232.

Zhu, Q., & Luo, J. (2022). Generative pre-trained transformer for design concept generation: An exploration. *Proceedings of the Design Society*, *2*, 1825–1834. https://doi.org/10.1017/pds.2022.185