Delft University of Technology

Thesis Project
SET3901

Multi-Modal End-to-End Learning for Real-Time Monitoring of Sustainable Energy Systems

Master Thesis

Author: Rushil Vohra
Student number: 4534506
Supervisor: Dr. Jochen Cremer

December 1, 2022

Thesis Committee
Dr. Jose Rueda Torres
Dr. Jochen Cremer
Dr. Wim Bierbooms
Ali Rajaei

Table of Contents

Acknowledgements

I would firstly like to thank Dr. Jochen Cremer for supervising the project every step of the way, and always providing open and honest advice. I would also like to thank Ali Rajaei and Haiwei Xie for their guidance and help on parts of the thesis. Without my parents and sister I would never have reached this stage of university (or life), and for them I am forever grateful. Finally, spending time outside university with my friends in Delft has been almost as fun as training neural networks, so I am glad to have them around too.

Abstract

The growth of renewable energy technologies is leading to energy systems that are more reliant than ever on renewables such as Wind and Photovoltaic (PV) power. Despite their obvious benefits in terms of sustainability, their ubiquity poses challenges in the form of maintaining grid stability given their inherent intermittency, making the prediction of power fluctuations more important. Physical models and statistical approaches, especially for nowcasting (forecasting for 0-6 hours in the future), may have been superseded by Machine Learning (ML) methods in terms of forecast accuracy (below 3% Root Mean Squared Error (RMSE)) [RSS15, ASMA20, RNE16]. Within ML, Artificial Neural Network (ANN) methods have been shown to perform particularly well for nowcasting. This project focuses not only on predicting solar and wind meteorology with that level of accuracy, but also on how to best use the prediction to minimize the cost of maintaining a balanced energy system, i.e. one where electricity consumption matches production at any moment. Producing accurate power predictions based on Multi-Modal (MM) data and the extent to which prediction accuracy reduces system cost are challenges to be addressed in this thesis. MM and End-to-End (E2E) training (with the system cost as the task of an ANN based algorithm) are investigated to this end. MM learning involves handling information from multiple types of input (audio and visual, for example) for performing a ML task such as regression or classification [CPZ19]. It is of interest for this project because it has been shown to outperform other Neural Network (NN) approaches in predicting sudden changes in solar irradiance [LWL+19]. E2E learning entails an algorithm design which predicts the end goal of a ML process directly from the raw inputs (directly predicting generation cost rather than a generation forecast in a power system from meteorological data, for example) [DAK17]. This form of training is pursued because it addresses the true task (cost minimization) of power plant operators as the focus of the ML algorithm.

The proposed method consists of a NN architecture that, through training, learns to fuse features from MM data (sky imagery and meteorological sensor data) at intermediate layers of the network in order to predict PV or Wind generation. This prediction is then used as an input to an Optimal Power Flow (OPF) problem (which seeks to minimize generation costs in a power system, considering power balance and transmission network constraints to ensure the twin goals of economic and secure system operation) [eco17]. The proposed model is trained E2E, therefore it is informed by the minimized cost solved by the optimization, rather than the intermediate power prediction (as conventional approaches would involve). In an IEEE 6-bus power system with PV generation, it is found that a sequential training approach (which trains a NN to predict PV generation and uses the prediction to solve the OPF) results in costs 10% higher than a perfect forecast, while our proposed MM4-E2E approach achieves costs only 7% higher, a significant improvement. This is explained by the added information about the system cost function received by the NN during E2E training. The intermediate prediction of PV power by MM4-E2E is also improved, with 18% lower RMSE by the proposed model compared to a Uni-Modal (UM) baseline, primarily explained by the enhancement of features from one modality by those of the other through MM learning. In a power system with two renewable sources (Wind and PV), costs are again reduced through the proposed model compared to a conventional approach (4% excess cost compared to 7%, measured against a perfect forecast), but power prediction accuracy is worse, demonstrating that, depending on the cost function, power prediction accuracy and cost minimization are not necessarily aligned. Again, the improvement in cost optimization is attributed to the information of the cost function provided to the NN through E2E learning. Power prediction accuracy may have been lost, in this case, because of the NN not converging to a unique cost minimum during E2E training.

Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

ARIMA Autoregressive Integrated Moving Average

ARMA Autoregressive Moving Average

CNN Convolutional Neural Network

DCOPF DC Optimal Power Flow

DL Deep Learning

E2E End-to-End

ED Economic Dispatch

FC Fully-Connected

FNN Feedforward Neural Network

GHI Global Horizontal Irradiance

HPO Hyperparameter Optimization

LSTM Long-Short Term Memory

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

ML Machine Learning

MLP Multilayer Perceptron

MM Multi-Modal

MSE Mean Squared Error

MT Multi-Task

MVR Multi-Variable Regression

NN Neural Network

NWP Numerical Weather Prediction

OPF Optimal Power Flow

PV Photovoltaic

ReLU Rectified Linear Activation

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

STC Standard Test Conditions

SVM Support Vector Machine

UM Uni-Modal

List of Figures

List of Tables

1 Introduction

Sustainable energy systems of the future will be, in many regions, heavily reliant on Photovoltaic (PV) power, Wind power, or both. These forms of energy generation are inherently uncontrollable due to their dependence on the local weather. Therefore, to ensure a balanced electrical system, predicting the supply is essential. Conventional Numerical Weather Prediction (NWP) models and statistical methods such as Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA) do not always provide the most accurate forecasts, especially in the short term, which has led to increased application of Artificial Intelligence (AI) and Machine Learning (ML) to these predictions in recent years [RNE16]. The ability of Artificial Neural Network (ANN) approaches, in particular, to capture the sharp changes in outputs compared to inputs with the help of intelligent training yields improved performance. Adaptive and robust NN training methods improve the capability of the network to learn complex relationships between input and output variables.

This thesis focuses not only on power prediction, but also the task of energy system optimization. The true task for power plant and transmission network operators is not necessarily the prediction of PV/Wind power, but the economic dispatch that minimizes generation costs given the predicted power. This opens the viability of End-to-End (E2E) learning to minimize costs. The E2E approach combines weather prediction scenarios and an Optimal Power Flow (OPF) calculation in a non-sequential way, such that the optimization informs the prediction [DAK17]. In addition, the use of Multi-Modal (MM) data sources, namely imagery and sensor data, to improve E2E forecasts is investigated in this study. The application of MM deep learning, which entails the combined use of data of different types (e.g. imagery and audio) to make predictions, to solar or wind forecasting is not yet ubiquitous. Its use has been argued as a method to improve the prediction of severe short-term fluctuations that statistical and other ANN approaches might miss [LWL+19]. The aim is to understand how to best combine MM data and E2E learning in order to accelerate the energy transition by reducing the cost of operating renewable energy systems.

The report is structured as follows. Chapter 2 defines the central research focus, the research questions and their respective sub-questions. This is followed by Chapter 3 which is a literature review focused on weather prediction, machine learning, and energy system optimization methods relevant for the thesis. Chapter 4 describes the procedures used, and the case studies designed to answer the research questions. Chapter 5 contains further details about the settings of each case study, and their results. Those results are discussed in Chapter 6, along with an outlook on further work; and the thesis is concluded with a summary Chapter 7.

2 Research Objective

This thesis is focused on improving the current E2E approaches to intra-hour power balancing by incorporating MM data sources for the intermediate power prediction, building on previous work done by the likes of Dariush Wahdany [Wah21], who focused on E2E learning for a power system with Wind generation. This work is extended by, firstly, including MM data (imagery and meteorological sensor data, as opposed to only the latter), and secondly, testing on systems with PV generation and systems with both PV and Wind generation. The main research question can thus be formulated as follows:

How can multi-modal data be used to improve end-to-end learning for forecasting renewable generation and minimizing costs in a power system with renewables such as PV or Wind?

The above cannot be answered directly, and is instead dissected into a list of sub-questions that are addressed over the course of the thesis. The conclusions from each of the following are used to ultimately answer the main research question.

1. How can Uni-Modal (UM) data, i.e. data of a single type such as imagery, be effectively applied to predicting PV power?
   (a) Which meteorological data are appropriate for prediction power injection?
   (b) How many previous time instances should be included to predict the future?
   (c) What are the requirements on using imagery, regarding computational load?
   (d) How can a neural network be designed to predict Global Horizontal Irradiance (GHI) from the individual modalities?
   (e) How do the prediction performance compare for the different modalities, to each other?
2. How can the multiple modalities be effectively combined to predict wind or PV power, and which data would enable this combination?
   (a) What are the methods for combining these modalities?
   (b) How can a neural network be designed for MM learning to predict PV power?
3. How does multi-modal learning outperform trivial uni-modal learning baselines?
   (a) Does MM learning produce more accurate power predictions than combining multiple uni-modal predictions in an ensemble method?
   (b) How much more or less accurate are predictions using MM learning compared to the ensemble method?
   (c) How do these compare to a persistence prediction of GHI?
4. How does pairing multi-modal learning and E2E learning advance optimized cost prediction?
   (a) How does a MM E2E trained model compare with a uni-modal sequential (conventionally trained) model in terms of system cost minimized via an Optimal Power Flow (OPF) problem?

2.1 Overview of Methodology

The research is conducted in a stepwise manner to address the research questions above. The steps are outlined in Table 1 and further described thereafter.

Table 1: Overview of the methodology, including the steps conducted to answer the respective research questions and the expected outcomes at each stage.

Stage	Research Questions Addressed	Methods	Outcomes
1	RQ1	- Data acqusition and pre-processing - Literature review - Training and testing different models	- UM power forecasting baselines - Assessment of predictions from imagery and sensor data
2	RQ2 and RQ3	- Literature review - Training and testing different models	- Proposed MM model for PV/Wind power forecasting - Assessment against UM baselines
3	RQ4	- Data acquisition and pre-processing - Implementation of OPF - Training and testing different models	- Proposed MM-E2E model for cost minimization - Assessment against sequentially trained models

The stages listed above are further described as follows.

1. Stage 1: the aim of this stage is to design NN-based models which make predictions of PV or Wind power from UM data (imagery or metorological sensor data). To this end, firstly, data of the two modalities are acquired and pre-processed (so that each dataset has the same temporal resolution, for instance). Using literature review, comparable existing models, and experiments, a variety of model configurations (for either modality) is compared. The best performing models in terms of prediction error are considered as UM baselines for the subsequent steps of the methodology.

2. Stage 2: here, MM learning for predicting PV or Wind power is investigated. The data used is the same as in stage 1. MM methods are again studied through literature and experiments to yield the proposed MM model for predicting power. This model is assessed against the UM baselines in terms of forecast error to answer RQ3, quantifying the benefit (or lacktherof) of MM learning for PV/Wind power prediction.

3. Stage 3: at this stage, E2E learning is to be incorporated with MM learning. E2E learning here involves solving an OPF based on predicted power generation. To this end, energy load data is acquired and pre-processed. Then, the OPF formulation is implemented as differentiable layers in the NN. The proposed MM-E2E model is trained (to directly minimize system cost from imagery and sensor data) and tested, and the results are compared to sequentially trained models (which are trained to predict PV/Wind power, and the predictions are used outside the NN to solve the OPF).

Much of the literature used to study the questions at each stage of the methodology is summarized in Chapter 3. These stages are described in further detail in Chapter 4. The case studies used to investigate different models, and their results, are given in Chapter 5.

3 Background

This chapter describes basic theory behind, and the current state of the relevant aspects of weather and energy system prediction for this thesis. Section 3.1 briefly summarizes several methods for weather prediction, followed by Sections 3.2 to 3.4 which focus on deep machine learning methods for making predictions. Aspects related to energy system optimization are described in Sections 3.5 to 3.8.

3.1 Weather Prediction

Several methods are used for forecasting renewable energy generation and the meteorology relevant to renewables. These range from conventional applications of NWP models, to ML techniques and ANNs. Although this thesis project will focus on the latter, it is worthwhile describing the methods generally used for solar and wind predictions. The applicability of these methods depends on factors such as the time for which the forecast is required and the data available. The main types can be summarized as follows [ASMA20, RNE16]:

- Persistence forecast: ideal for very short- and short-term forecasts, this method essentially assumes that recent very data will be repeated in the near future. It is mostly used as a tool to benchmark other methods against.
- Physical models: these NWP models combine a large amount of meteorological data and atmospheric modeling equations to retrieve solar and wind predictions. They are more suited to long-term, large-scale predictions, and often miss the very short-term fluctuations in energy.
- Statistical techniques: these make extensive use of time series data from the past and real-time generated data. The appeal of these methods is that they are far less computationally intensive than Deep Learning (DL) approaches while also performing better than NWP models in short-term forecasts. Within this field there exist methods with widely varying complexity, from basic time series methods like ARIMA to conventional ML techniques such as support vector machine Support Vector Machine (SVM) and Multi-Variable Regression (MVR).
- Deep learning: DL is an unsupervised form of ML which finds patterns in its inputs to make predictions. Amongst others, the Convolutional Neural Network (CNN) is a commonly used architecture for forecasting which is particularly relevant [Edu20]. It usually consists of a convolutional layer (responsible for feature extraction), a pooling or dimension reduction layer (which aggregates features), and a Fully-Connected (FC) layer (used to transfer the learned distributed feature representations and perform classification).

ML methods (including DL) are commonly applied to nowcasting (forecasting for up to a few hours ahead), short-term forecasting (for up to a few days ahead), and aiding NWPs with certain tasks [ASMA20]. They do not require any physical understanding of meteorological dynamics, and instead rely on resolving patterns from large datasets for which the prediction target is known [Due19]. The data used are commonly meteorological data in the form of time series, measuring variables such as air temperature, cloud opacity, humidity, surface pressure, and so on. In addition, particularly for solar power, ground-based sky imagery is useful in nowcasting. These are images taken of the sky from the ground. The ML algorithm perceives patterns from training sets of meteorological and/or visual data, which it uses to make predictions on new samples for which the solar power is not known. In general, the independence from rules and the understanding of governing equations allows ML approaches to have several advantages over conventional modeling methods [Ger19]:

1. When applied to problems with for which model solutions require fine-tuning or long lists of rules, an ML algorithm can simplify code and perform better than the model approach.
2. In problems with fluctuating environments or situations, ML solutions will adapt to new data
3. ML solutions can yield insights from large amounts of data that may not have been noticed through conventional models.

A study by Sabzehgar et al. has compared the performance of a number of techniques for making solar forecasts [SAR20]. Namely, a NN-based model is compared with SVM and MVR approaches. Using each of the three statistical methods, the authors aim to forecast solar irradiance based on weather parameters. This forecast is then used to predict the amount of generated power in a smart residential microgrid. The same data is used for the three approaches: meteorological data for prediction from the National Renewable Energy Laboratory, and electrical power data (for error estimation) from their local (San Diego) energy provider. The accuracy of the forecast models is evaluated and compared using Mean Absolute Percentage Error (MAPE) and Mean Squared Error (MSE). The study

found the NN-based method to be significantly more accurate than the other two methods (MVR and SVM). It performed better in the prediction of solar irradiance and of power generated. Sobri et al. also conducted a review of the state of solar forecasting techniques [SKKR18]. They reach a similar conclusion about the improvements of NN models as compared to traditional methods. These findings are primarily explained by the ability of NNs to model complex and non-linear functions, as its vast number of neurons can capture patterns relatively better than MVR or SVM approaches.

Section 3.2 focuses on deep learning theory which is most relevant for this thesis.

3.2 Deep Learning

Deep Learning refers to the field of ML using of ANNs, or artificial neuron-based architectures that mimic the networks found in brains. These tend to be much more effective with increasingly large datasets than other ML methods. Each artificial neuron has a set of inputs and a single output. The output is calculated by applying a function, known as an activation function, to the weighted sum of its inputs (plus a bias term). The weights are variable, which allows the network to improve during training by updating the weights and biases. Equation 1 summarizes the function of a neuron.

$$output = f(\sum_{i=1}^{n_{inputs}} w_i \cdot x_i + b) \tag{1}$$

Here f is the activation function, n_{inputs} is the number of inputs to the neuron, x_i is each input value, w_i is the weight attributed to each input, and b is the bias term. Figure 1 visualizes the function of a type of artificial neuron called a threshold logic unit, where the activation function is a step function.

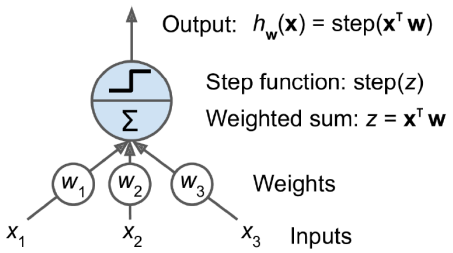

Figure 1: A threshold logic unit, or an artificial neuron which calculates its output by applying a step function to the weighted sum of its inputs [Ger19].

The step function, along with other commonly used activation functions are illustrated in Figure 2.

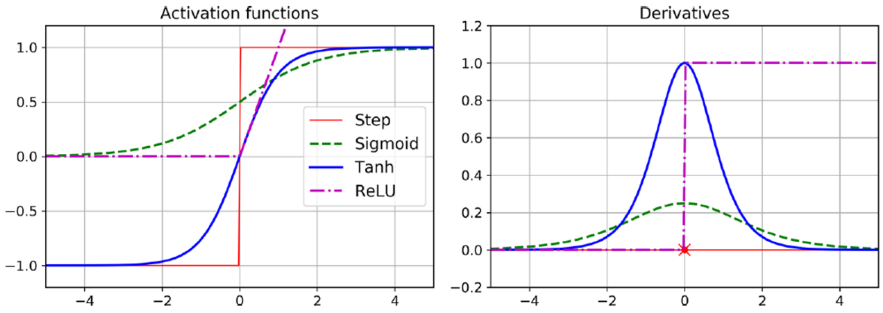

Figure 2: Plots of common activation functions and their derivatives [Ger19].

The properties of the activation functions are central to producing an effective ANN. Training the ANN requires 'backpropagating' gradients calculated on the outputs of the neurons (further described in Section 3.3), which are directly dependent on the activation function. A few pertinent properties of the shown functions are mentioned below.

- Step: the simplest one mentioned, but the lack of derivative means that this function is not effective when working with backpropagation (further described in Section 3.3) and gradient descent.

- Sigmoid: well-defined and differentiable everywhere, with non-zero derivatives; this makes it easy to calculate gradients. However, the diminishing derivatives further from the origin make this function susceptible to the vanishing gradients problem (further described in Section 3.3).
- Tanh: continuous and, like Sigmoid, differentiable everywhere, but also may suffer from vanishing gradients.
- Rectified Linear Activation (ReLU): continuous but not differentiable at the origin (which can hinder gradient descent). It is very fast to compute and does not have a maximum output (which aids gradient descent), and it does not have saturating gradients like Sigmoid or Tanh.

Neurons are commonly compiled in parallel in layers, which are then connected sequentially to produce a neural network architecture. When the neurons of a layer are connected to each of the neurons in the layers above and below it, the layer is called fully connected or dense. A Multilayer Perceptron (MLP) (also known as Feedforward Neural Network (FNN)) is composed of exclusively FC layers. A basic MLP, with 2 inputs and 3 outputs, is shown below.

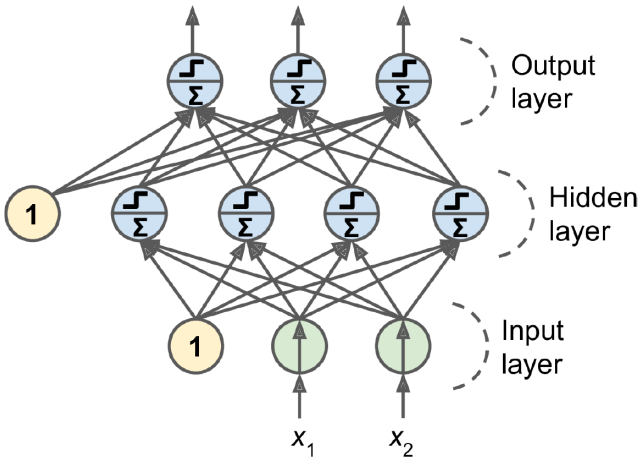

Figure 3: A multi-layer perceptron network with 2 inputs, 1 hidden layer, and 3 outputs.

While MLPs only use FC layers, there are other types of NN such as the CNN, which use convolutional layers in addition to FC layers of neurons.

Convolutional Neural Networks

CNNs, unlike FNNs, use convolutional and pooling layers in addition to FC layers, making them more suited to certain applications, such as learning from visual data [NRB20]. A generic architecture is visualized in Figure 4; the convolutional and pooling layers are useful at constructing higher dimension feature maps from the inputs, which are used by the FC layers to predict a label. The feature maps, through training, extract information from the imagery that is most relevant to the prediction.

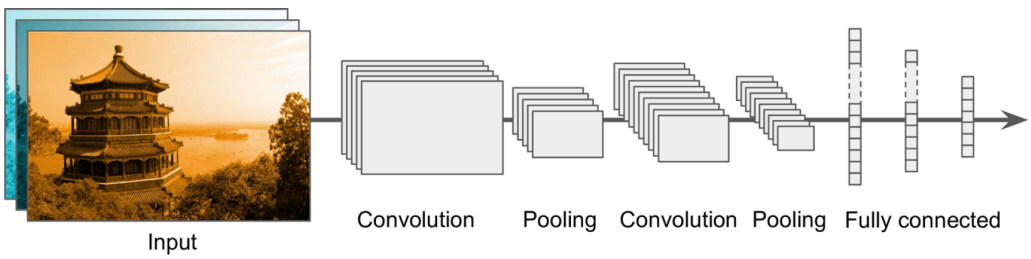

Figure 4: A typical CNN architecture [Ger19].

When dealing with a 2-dimensional image, for instance, a convolutional layer within the CNN will contain a set of kernels (2-dimensional maps with sizes smaller than the image) which pass over the width and height of the image. As a kernel passes over a dimension, the dot product of the kernel and the image pixels is computed. The passing of each kernel over the image, therefore, generates an *activation map* or *feature map* [KLB+18]. The feature maps for

all the kernels are stacked to produce the output of the convolutional layer. The weights of each of the kernel matrices are learnable parameters for the network during training. The kernel size, stride (steps taken in each direction by the kernel), and padding (pixel distance from the edge of the image that is not covered by the kernel) are hyperparameters.

The pooling layers have the effect of 'shrinking' the image by aggregating input pixels, most often by replacing a set of pixels by their maximum or mean value. This practice reduces the number of parameters of the network, therefore reducing the computational load of training it and the risk of overfitting (overfitting is described in Section 3.3). Again, the kernel size and stride are hyperparameters.

Using an architecture such as the one shown in Figure 4 will make an image smaller and smaller (because of the pooling layers) as it progresses through the forward pass, while also making it 'deeper' (due to each convolutional kernel adding feature maps). After the last pooling layer, the feature stack is flattened to one dimensional, and then passed through a FNN (the FC layers shown in the figure), the last layer of which makes a prediction. A particular advantage of CNNs is that once it has learned to recognize a pattern in one location, it can recognize it in other locations as well. This is not immediately possible with an FNN [Ger19]. In the context of forecasting PV power generation from imagery, as an example, the CNN's pattern recognition may enable it to associate certain visual patterns with objects like clouds and others with the sun. In principle, given enough data with high enough resolution, the NN can also differentiate between types of clouds. The recognition of such objects is correlated with moments of relatively high or low solar irradiance through training, and therefore the NN 'learns' to predict solar power from an unseen image by recognizing the level of solar visibility or cloud coverage.

3.3 Training and Testing

Supervised learning for an ML algorithm follows a common methodology regardless of the chosen algorithm. To fit an algorithm for the task at hand, given a set of data, commonly the data is split into a *training set* and a *test set*. The former contains samples and their respective labels that the NN can learn from. The test set is used to check the performance of the trained NN by comparing its predicted labels to the 'true' labels.

During training, usually a labeled sample from the training data is provided as input to the algorithm, and the prediction it makes is compared with the label. The error between the two is calculated (using a loss function suited to the problem), and the parameters of the model are updated in order to reduce this error. The weight updates can be calculated using gradient descent (or one of its variants), an optimization technique which seeks to find the values of the weights that minimize the cost. In an ANN, where there are several layers of neurons and perhaps millions of parameters to update, an algorithm called backpropagation enables a computationally efficient weight update by avoiding repeated gradient calculations [RHW86]. It is summarized in Figure 5 and described thereafter.

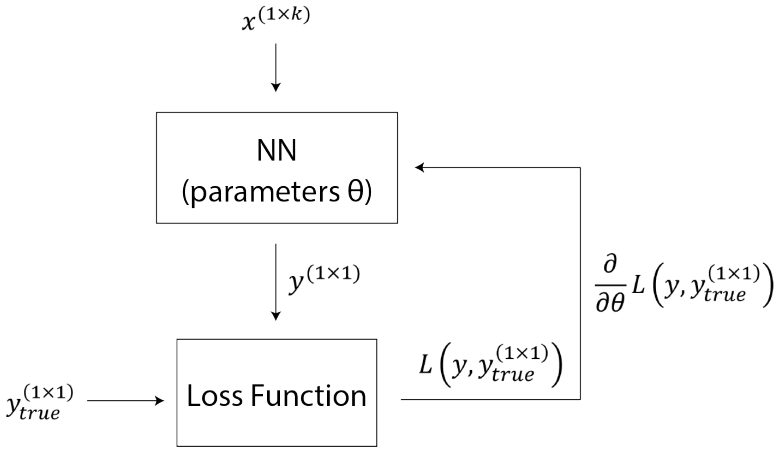

Figure 5: Summary of backpropagation routine.

Briefly, it works as follows:

1. During the forward pass, each the inputs sample ($x^{(1\times k)}$) is passed through the layers of the ANN with parameters θ. Here, k is the number of features in each input sample. The outputs and weights at each layer are preserved.

2. The error between the network's output ($y^{(1\times1)}$) and the true label ($y_{true}^{(1\times1)}$) is computed using a selected loss function. This is $L(y, y_{true})$.

3. The error contributions of each parameter in the output layer, as well as the hidden layers below are calculated using partial derivatives ($\frac{\partial}{\partial\theta}L(y, y_{true})$). These gradients are calculated in reverse starting from the output layer and applying the chain rule going backwards. This is the process to which the algorithm's name is attributed.

4. The error gradients are used by an optimization algorithm to update the weights such that the loss is minimized. Notable methods are gradient descent, Stochastic Gradient Descent (SGD), Adam, and AdamW; these are described in the following section.

5. The steps above are conducted for all training samples, which constitutes one training *epoch*. It is common practice to also make predictions on a set of samples reserved as a 'validation' set, from which the NN weights are not updated. Monitoring the accuracy of predictions on the validation data allows the optimal model configuration to be found over dozens of epochs without *underfitting* or *overfitting* the training set (these are explained ahead).

If a ML model performs exhibits a low prediction error with the training data, but then predicts with higher error on unseen samples (i.e. does not generalize well), it is considered to be overfitted to the training set. This can be a consequence of using a model that is too complex (too many trainable paramaters) for the size of the available training set [Ger19]. On the other hand, an underfitted model is too simple (not enough trainable parameters) to adequately recognize patterns in the data and therefore does not perform well enough on training or test data. This can be understood visually from Figure 6 which shows an example of underfitted, well-fitted, and overfitted models for approximating a given function.

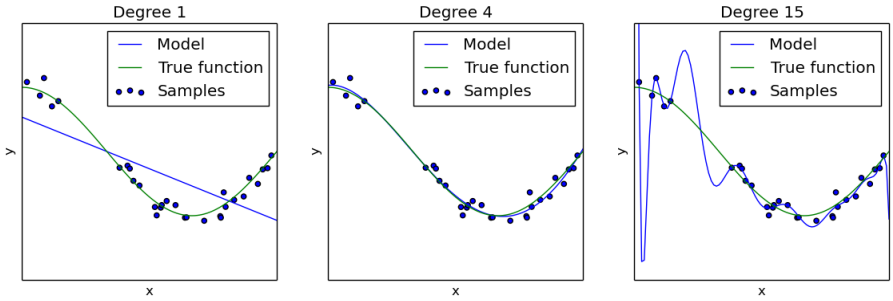

Figure 6: An underfitted, a well-fitted, and an overfitted model for the approximation of a function [sl14].

Over the course of training a NN, as the model parameters are updated to better predict the training data, the model is initially underfitted. If it is trained for too many epochs on the same data, it will eventually be overfitted. An overfitted model will have a significantly higher error when predicting unseen samples than it does on the seen (training) samples. Continuously testing it (at every epoch) on a smaller set of data that it has not learned from allows the onset of this phenomenon to be noticed, as visualized in Figure 7.

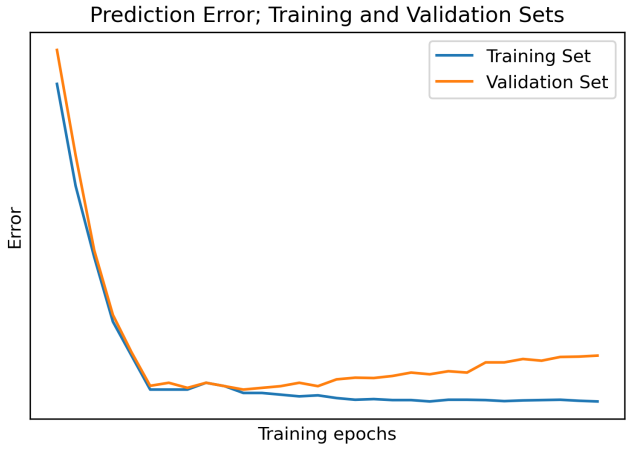

Figure 7: Learning curve depicting a NN being overfitted to the training set.

This type of plot, known as a *learning curve*, shows the performance of a NN in terms of its prediction error. The error on the training set continually decreases as the model parameters learn from it, however, it soon gets worse at predicting the validation samples. This demonstrates overfitting; it can be avoided through hyperparameter choices and the thoughtful design of the architecture, as well as by halting the training before the divergence of training and validation error (known as *early stopping*). Early stopping can be implemented, for instance, by tracking whether the validation error has decreased over the last 5 epochs, and if not, then training is stopped.

Optimizers

Minimizing the loss during training can theoretically be done directly by solving a matrix equation. When applying a MSE loss function to fit a linear regression, for instance, the error of predictions on a training set X with m samples is measured as follows:

$$MSE(X) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 \tag{2}$$

Here, θ represents the trainable parameters of the linear regression, therefore $\theta^T x^{(i)}$ is the prediction at instance i, and $y^{(i)}$ is the true value at instance i. Minimizing this function yields the following closed-form solution for θ.

$$\hat{\theta} = (X^T X)^{-1} X^T y \tag{3}$$

Equation 3 provides a solution that minimizes the loss for a training set, but in practice, this can be very computationally expensive to compute when dealing with many features. Matrix inversion and *singular value decomposition* (an alternative method to solve the above equation using pseudoinverses instead) can be between $O(n^2)$ and $O(n^3)$ to compute, where n is the number of features [Ger19]. In order to find minima and avoid the computationally expensive solutions, the *gradient descent* algorithm can be employed. Considering the same example of an MSE loss for training a linear regression, gradient descent would work as follows:

1. $MSE(X, \theta)$ is computed for a training set X and parameter vector θ.
2. The variation of the loss due to a change in each of the parameters, measured by the partial derivatives $\frac{\partial}{\partial \theta_j} MSE(X, \theta)$ is computed. These gradients are collected in the vector $\nabla_\theta MSE(X, \theta)$.
3. In order to decrease the loss, the parameters should logically be shifted in the direction opposing their partial derivatives, therefore they are updated by subtracting the gradient multiplied by a step ν (also called the *learning rate*):
$$\theta^{update} = \theta - \eta \cdot \nabla_\theta MSE(X, \theta) \tag{4}$$

The learning rate is an important hyperparameter when training, as it can cause the algorithm to converge very slowly (η too low) or repeatedly overshoot the minimum (η too high). These phenomena are illustrated in Figure 8.

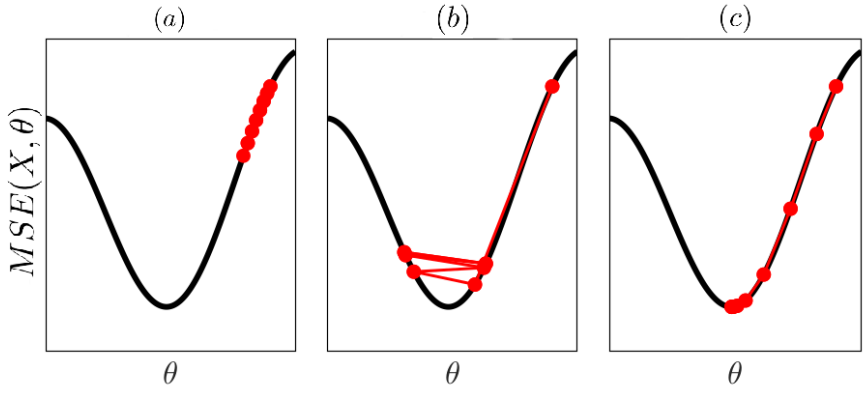

Figure 8: The effects of a learning rate set too low (a), too high (b), or just about right (c).

Given a convex loss function and sufficiently low learning rate, gradient descent is guaranteed to find the optimal parameter vector. However, this process may be slow because the entire training set X is used to update the parameters. SGD improves on this by using only a single random sample of training data on each iteration of the parameter update. This makes the 'path' to the minimum more chaotic, but as long as the chosen samples

are independent and identically distributed, the algorithm will reach the vicinity of the minimum. How close it reaches depends on the number of allowed iterations. Another similar approach, *mini-batch gradient descent*, uses a set of random samples rather than just a single one. This makes its path to the minimum less erratic than that of SGD.

More recent developments include the *Adam* optimizer, which uses varied learning rates for different parameters [KB14]. *AdamW* improves on it further by including weight decay to the parameter update, which seeks to avoid overfitting by penalizing high parameter weights and has been shown to speed up training [GH18, LH17].

Normalization

An issue that commonly occurs while training NNs, and was eluded to previously, is that of vanishing gradients. This occurs when the error gradients computed for all or many neuron parameters are 0 or close enough to 0 that the weight update does not yield improvements in the network's predictive performance. The choice of activation functions is central to avoiding this phenomenon; the ReLU and Leaky ReLU functions have been shown to be effective in this sense thanks to their nonzero derivatives throughout the positive domain (positive and negative domain for Leaky ReLU) [XWCL15]. Here, batch normalization and inputs normalization are explained as two techniques to mitigate vanishing gradients.

Batch normalization addresses the high outputs of hidden layers which may contribute to vanishing gradients. When dealing with layers with sigmoid or tanh activation functions, high values within the matrix of $|x| = |(Wu + b)|$ (where x is the output of the layer, u is its input, W is the weight vector, and b is the bias) would cause the gradients to approach 0. This leads to the vanishing gradients problem and slow learning. With batch normalization, a layer is included between the hidden layers of a network that normalizes the output from the layer below it. This is advantageous because it ensures the input u to a hidden layer above a batch normalization layer will only contain values within a controlled range, thereby avoiding the vanishing gradients [IS15]. In addition to alleviating the risk of vanishing gradients, batch normalization has also been shown to reduce a NN's sensitivity to weight initialization (described further in the following section).

Another pertinent mechanism to avoid vanishing gradients is inputs normalization, or the scaling of data before it enters the network. *Min-max normalization* and *standardization* are two common procedures for scaling inputs to ML algorithms. The former ensures that the entire data set is bounded by 0 and 1, by applying the following formula to each point x:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5}$$

Standardization, on the other hand, ensures a unit variance but not a bounded data set by applying the following (where μ is the mean of the data set and σ is its standard deviation):

$$x_{std} = \frac{x - \mu}{\sigma}$$

When working with NNs, since they expect input values between 0 and 1, min-max normalization is generally preferred [Ger19].

Initialization

Another aspect to consider while training NNs is the weight initialization of the neurons. It has been shown that to avoid vanishing or exploding gradients, the variance of the outputs of each layer should be equal to the variance of its inputs (during the forward pass), and the variance of the gradients should be equal before and after flowing through a layer (in the backwards pass) [GB10]. This can be explained by considering the sample during the forward pass as a signal flowing towards the output; to make a sensible prediction, the signal should not die out, nor should it be uncontrollably amplified. Glorot et al., along with other authors, found that (depending on the chosen activation function) the weights of a layer should be initialized by randomly sampling a uniform or normal distribution whose parameters are dependent on the number of inputs (fan_{in}) and neurons (fan_{out}), or the average of these two (fan_{avg}) [HZRS15b]. For instance, for ReLU activation, the suggested initialization (dubbed *He Initialization* after Kaiming He) is a random sampling from a uniform distribution with lower limit $-r$ and upper limit $+r$, where:

$$r = \sqrt{\frac{3}{fan_{avg}}}$$

While this initialization strategy is developed for the ReLU activation function, similar ones have been devised for other functions such as sigmoid or tanh [LBBH98].

3.4 Multi-Modal Learning

MM learning involves combining different data sources (or modalities) to predict a single concept, e.g. combining meteorological data and sky imagery to forecast GHI [LWL+19]. Meanwhile, Spiess et al. have demonstrated the applicability of the same modalities (sensor data and sky imagery) to predict an optimal control policy for battery operation [SBPK18]. That research investigated the use of DL and MM data to minimize the stress on a battery that is used to stabilize PV power fluctuations. The core appeal of MM, in general, is the enabling of learning from heterogeneous data streams in a manner that the data from either modality enhance each other.

An overview of the primary steps in MM learning is provided by Baltrusaitis et al. [BAM19]. Particularly relevant for this project is the approach to representing/fusing modalities (representation and fusion are sometimes conducted as separate stages). This refers to the distillation of each modality to features which can then be compared with one another. Coordinated representation is a method suited for a maximum of two modalities, whereby the modalities would be separately projected to an abstract space within which the similarity of the modalities would be maximized. The similarity can be quantified by different metrics, e.g. Euclidean distance in the abstract space. Joint representation, on the other hand, relies on applying a common function to all the modalities together to produce a single representation vector. The 'function' in this context is often an ANN. Within joint representation, a few different fusion methods can be distinguished [LAJ15, Mor20].

- Early fusion: here, the raw or pre-processed data from different modalities are fused (most simply by concatenation) before being used as inputs to an ML model. A challenge of this method is the loss of data from one or more of the modalities that may be incurred in order to make common ground before fusion. Therefore it is not well suited to using imagery and sensor data as modalities, since the latter cannot be directly concatenated with the pixel values of an image.
- Late fusion: instead of combining the data before they are input to a model, when utilizing late fusion, a prediction is made separately from modality, and these predictions are fused. There are several choices for how to fuse the predictions; a basic approach such as a weighted ensemble could be used, or a more complex ANN one such as a CNN or FNN.
- Intermediate fusion: late fusion introduced the idea of employing a multimodal layer to make predictions. This MM layer could also instead be implemented lower in the NN, as a hidden layer. The position within the NN of this layer (the layer index) therefore must be selected, keeping in mind that a MM layer can be implemented multiple times, and at different indices for the different modalities.

Fusing different modalities through joint representation, in practice, requires combining different feature tensors for each instance. This can be done simply by concatenating the features. Another suggested approach (for two modalities) is bilinear pooling, where the outer product of the two feature tensors produces the MM feature tensor [ZCP+17]:

$$h_{mm} = \begin{bmatrix} h_1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} h_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_1 \otimes h_2 \\ 1 & h_2 \end{bmatrix} \tag{6}$$

Here h_{mm} is the MM feature tensor, h_1 is the feature tensor for the first modality, and h_2 is the feature tensor for the second. A potential disadvantage of this method is that it can result in a very large and imbalanced feature tensor (if h_1 is n dimensional and h_2 is m dimensional, then h_{mm} is $(n+1)\cdot(m+1)$ dimensional). However, it has also been argued that imbalances in the tensor allow the network to identify important features more quickly.

The variety of methods described above opens a range of questions regarding design choices since there is no universally preferred approach. The choice is dependent on factors such as the problem context (what the model is tasked with predicting), types of modality (the available data), and computational requirements. It should be noted that, particularly during training, the approaches to normalization become doubly important to consider when dealing with MM learning. Fusing modalities, potentially at different stages of the NN, must not cause the unintentional loss or amplification of features due to mismatched feature scaling.

3.5 Energy Systems

Energy systems generally consist of loads (which demand energy), generators (which supply energy), transmission lines (either AC or DC) and distribution networks [eco17]. At times storage units are also included, which can act

as loads or generators depending on whether there is a surplus or shortage of generation. The frequency of the grid by which loads and generators are connected must maintain stability at all times, meaning that consumption and generation need to be balanced in real-time. The increasing inclusion of renewable generation in European grids is accompanied by increasing stochasticity, especially on the very short term (minutes to hours) due to the inherent intermittency of renewable energy. Energy mismatches in real-time can be alleviated by exchanges in the intraday spot market, where market participants can trade energy at up to 15 minute frequency [EPE22]. This form of trading motivates the research into forecasting techniques for energy generators to run optimal schedules - for example, in terms of minimal overall generation cost, specifically minimal cost of renewable generator operation, etc.

Economic Dispatch (ED) is the problem of determining the generation levels of the committed generators in a power system to satisfy the load demand at the lowest possible cost, while considering the limits of the generating units. Due to the nature of the cost functions of conventional thermal generation plants, the problem is canonically nonlinear. Optimal Power Flow (OPF) formulations extend ED problems by including power balance and transmission network constraints. These are therefore nonlinear problems which determine the control variables and state variables that minimize generation costs and ensure secure system operation [eco17]. The DC approximation of this problem, the DC Optimal Power Flow (DCOPF), given in Equations 7a through 7d, simplifies the formulation by ignoring reactive power balance and line power losses. The variables are described in Table 2.

$$\text{minimize} \quad C(p^g) = \sum_{i \epsilon G} c_i(p_i^g) \tag{7a}$$

$$\text{subject to} \quad -p^g + p^d + g(\theta) = 0, \tag{7b}$$

$$h(\theta) \leq s, \tag{7c}$$

$$p_i^{g-} \leq p_i^g \leq p_i^{g+}, \quad \forall i \epsilon G. \tag{7d}$$

Table 2: Variables involved in DCOPF

Variable	Description
θ	Vector of voltage angles of all nodes
$g(\theta)$	Real power flow functions at all nodes
p^g	Vector of power injections at each node
c_i^g	Cost of power generation at node i
p^d	Vector of power withdrawal at each node
p_i^{g-}	Lower limit of generation at node i
p_i^{g+}	Upper limit of generation at node i
$h(\theta)$	Real power branch flow functions
s	Vector of limits for real power branch flows
G	Number of generator nodes

Equation 7a represents the total generation cost from all generation units, which is the objective to be minimized. The power balance constraint (Equation 7b) ensures that the power injected and withdrawn at each node is balanced. Constraint 7c limits the power flow in each branch of the network due to the thermal limits of the lines. Finally, Equation 7d constrains the level of power generation at each generator node to its respective lower and upper limits.

3.6 E2E learning

End-to-end learning is a form of ML model training where the model is trained (or 'learned') based on the ultimate task of the user, as opposed to an intermediate output such as a loss function [DAK17]. In the context of this research, this implies learning based on the ultimate value of an optimization operation (such as a monetary cost) following a model prediction, rather than learning based only on the prediction. Applying E2E learning to energy system dispatching involves the evaluation of a stochastic programming problem following the prediction of time series variables such as wind power or energy demand. Let the variables and task be labeled as follows:

- Inputs to the predictive model are called X. These can include energy load in the past, meteorological data, or other variables useful for predicting energy load.
- Output of the prediction is y. This can be energy load for the current problem description.

- Optimized actions are Z. These are the results of the stochastic programming problem, and describe the energy scheduling to solve the economic dispatch and unit commitment problem for the power plants.
- Task loss is a measure of the performance of the learned system, in this case, it can be considered the system cost of the optimized actions Z.

The planned actions Z are the result of optimizing the problem described by Equation 8 (called the generator problem by Donti et al.):

$$\begin{aligned} \min_{z \in R^{24}} \quad & \sum_{i=1}^{24} E_{y \sim p(y|x;\theta)}[\gamma_s[y_i - z_i]_+ + \gamma_e[z_i - y_i] + \frac{1}{2}(z_i - y_i)^2] \\ \text{s.t.} \quad & |z_i - z_{i-1}| \leq c_r, \forall i \end{aligned} \tag{8}$$

This stochastic formulation is minimizing task loss (due to under- (γ_s) and over- (γ_e) deployment penalties) while quadratically regularizing Z as close to y (energy demand) as possible. The planned actions Z* are calculated using three ways and the task loss of each of the three results is compared:

1. Use an ANN to first forecast y for upcoming day based on the previous day's load, temperature, and other meteorological variables. Then solve the generator problem for Z* using y such that task loss is minimized. This is the implementation of E2E learning.
2. Use an ANN to first forecast y for upcoming day based on the previous day's load, temperature, other meteorological variables. Then solve for Z* just based on Root Mean Squared Error (RMSE) (i.e. last term of the generator problem) of the difference between Z and y.
3. Use an ANN to first forecast y for the upcoming day based on the previous day's load, temperature, other meteorological variables. Then solve for Z* just based on the cost-weighted RMSE of the difference between Z and y. Here, the training samples are periodically reweighted given their task loss.

The results of method 1 produce the lowest mean and variance of task loss, indicating the value of E2E learning. The same paper describes a similar implementation to battery arbitrage, whereby the user (a battery operator) must decide on how much to charge and discharge for each hour of a day based on energy price predictions. Compared to conventional RMSE loss learning, task-based learning produced somewhat lower losses and more reliability (less variance).

3.7 Forecasting Data

Imagery Dataset

All sky images captured at TU Delft were used for making predictions of the GHI at the site [Del22]. The dataset consists of images captured at a minute-frequency between 01 May 2021 and 24 February 2022. The images are captured between 04:00 and 23:00 for every day of the dataset. Each image is in full color, with a resolution of 1536x1536. An example of an image, taken on 01 May 2021 at 11:00 is provided in Figure 9.

Figure 9: All sky image captured by the camera at 11:00 on 01 May 2021 [Del22]

Meteorological Dataset

A variety of meteorological variables can be used to predict GHI using ANNs. Demirtas et al., for instance, use air temperature, humidity, and barometric pressure to forecast solar radiation 10 minutes ahead of time [DYSC12]. For this project, meteorological data were acquired from Solcast, and all the available variables were included as features used for forecasting, which are listed below, along with their respective units [Sol21]:

- Time [seconds]: the moment of the day when the instance was measured.
- Air Temperature [°C]: measured at 2 meters above surface level.
- Cloud Opacity [%]: the attenuation of incoming sunlight due to cloud. Varies from 0% (no clouds) to 100% (full attenuation of incoming sunlight).
- Relative Humidity [%]: measured at 2 meters above ground level. Relative humidity is the amount of water vapor as a percentage of the amount needed for saturation at the same temperature. A value of 50% means the air is 50% saturated.
- Wind Direction [°]: measured at 10m altitude.
- Wind Speed [m/s]: measured at 10m altitude.
- Precipitable water [kg/m^2]: a measure of the precipitable water of the entire atmospheric column.
- Surface Pressure [hPa]: air pressure measured at sea level.

Further details about each parameter and its measurement and processing can be found in the documentation of the European Centre for Medium-Range Weather Forecasts database [ECM22].

Load Demand

For modeling load demand, the dynamic profile of a normal household in Germany is used [BDE17], which is available at 15 minute temporal resolution. This is assumed to be a similar profile to that of a Dutch household. The profiles for 01 January and 01 June are shown in Figure 10.

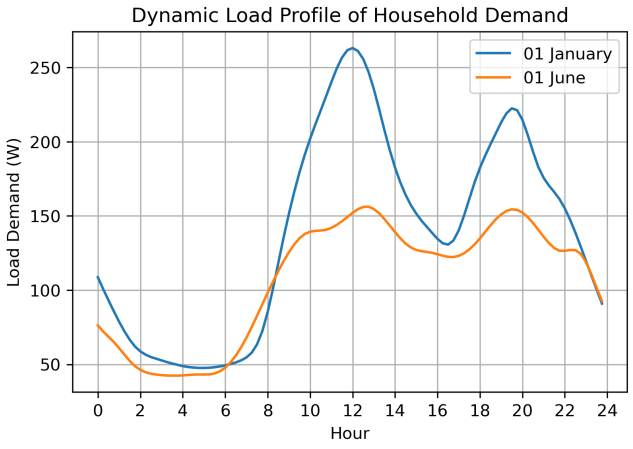

Figure 10: Dynamic load profile of household demand [BDE17]

As can be expected for Germany, the energy consumption of a household is higher in the winter than in the summer. It is a challenge in renewables-based energy systems to match the peaks in demand to intermittent supply.

3.8 Renewable Power Generation

Thus far the computational background for forecasting weather patterns has been described. Ultimately using these forecasts in energy systems requires informing power calculations from the forecast conditions. This is done for both PV and wind power generation at different stages of the thesis; the basic theory for such power conversions is described below.

PV Power

The power output of a solar panel based on GHI can be simply estimated using the power rating of the panel. Equation 9 provides the calculation, where P_{PV} is the output of a panel in W and P_{Rated} is its power rating (in W) at Standard Test Conditions (STC) [Isa21].

$$P_{PV} = \frac{GHI}{1000W/m^2} \times P_{Rated} \tag{9}$$

This is a simplified formula for calculating the output and does not consider a variety of factors, for instance, the effect of panel temperature on its output. Since this thesis project is focused on developing a methodology for cheaper operation of renewable energy systems, this simplified approach to power conversion is acceptable and does not have a significant effect on the investigation.

The panel used for the project is one that is commonly installed in Northern Europe, the LG NeON BiFacial 440. It has a rated power of $440W$ and an area of $2.2m^2$ [LG 21].

Wind Power

The power generation from wind turbines has a somewhat less straightforward relationship with wind speed. Turbines have a 'cut-in' speed, the lowest speed at which they generate electricity, a 'rated speed', which is the lowest wind speed at which the turbine achieves its rated power, and finally a 'cut-out' speed, beyond which the turbine is disconnected and produces no power in order to protect its components. Between the 'cut-in' speed and rated speed, the power generated follows a theoretically cubic relationship with wind speed. In practice, the curve is not exactly cubic, to ensure smooth operation of the turbine as it transitions between non-operation, its non-rated wind speed region, and rated wind speed region. For the Vestas V112-3.45 turbine, which has been used for farms in the Netherlands and the rest of Northern Europe, the power curve is provided in Figure 11 [owe22].

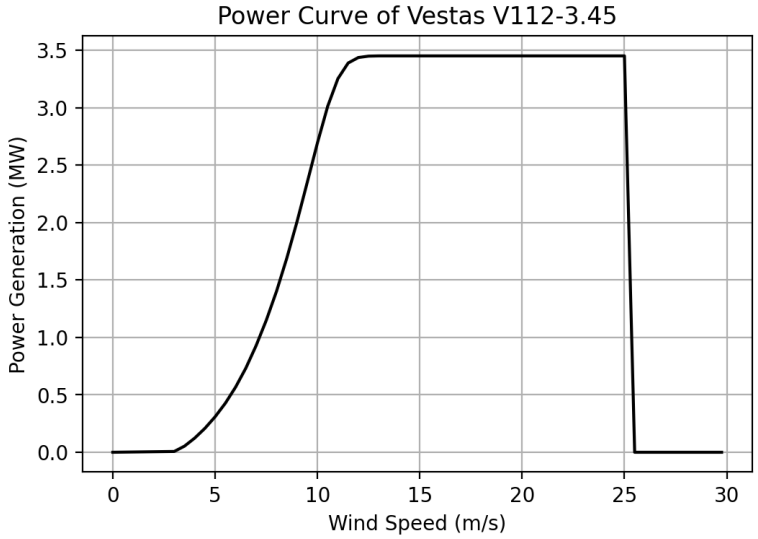

Figure 11: Power curve of Vestas V112-3.45 turbine [win21]

The wind speed in this context refers to the wind speed at the hub height of the turbine. Measurements, including the ones used for this thesis, are normally available for a height of 10m. The wind speed profile in a statically neutral atmospheric boundary layer (as is assumed for this setting) is described by Equation 10, where M_1 is the speed at height z_1, M_2 is the speed at height z_2, and z_0 is the surface roughness. The latter depends on the type of surface, for instance, open sea or built environment, and is approximated by the Davenport-Wieringa roughness-length classification [Stu95]. A higher value for z_0 implies a rougher surface.

$$M_2 = M_1 \cdot \frac{\ln(z_2/z_0)}{\ln(z_1/z_0)} \tag{10}$$

The Vestas turbine, for example, has a hub height of 84m; applying Equation 10 implies that it faces higher wind speeds than those measured at 10m.

4 Methodology

To reiterate, this project aims to investigate the suitability of multi-modal learning for end-to-end learning for power system cost minimization. This is carried out in a stepwise fashion (as outlined in Table 1) - first, the use of MM learning compared to UM learning for 'sequential' prediction (training to predict solar power output measured by GHI) is analyzed (to answer research questions 1, 2, and 3). Then, research question 4 is addressed by using the best-performing UM or MM algorithm for E2E learning. The effectiveness of E2E training for minimizing system costs is compared to that of sequential training.

4.1 Uni-Modal Baseline Models

To ultimately develop a MM Learning model for forecasting power output and grid scheduling, it is worthwhile first describing how a UM model is designed and implemented. This section describes the use of ANNs to predict GHI from sky imagery and meteorological data individually.

4.1.1 Forecasting from Imagery

Imagery data described in Section 3.7 is used to forecast GHI by employing an ANN. The workflow for forecasting from imagery is summarized schematically in Figure 12.

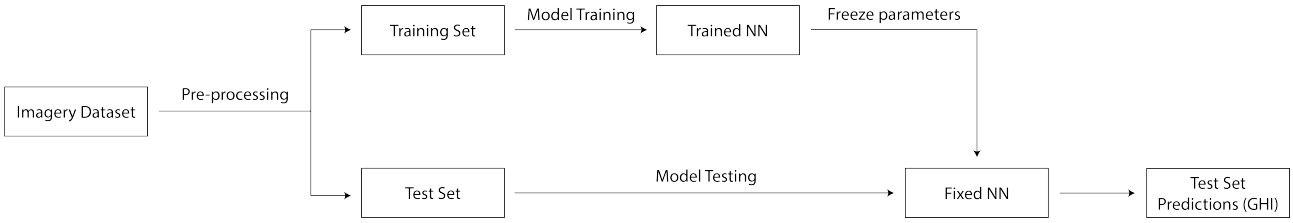

Figure 12: Procedure followed for predicting GHI from imagery using a NN.

The model training procedure was described in Chapter 3 using Figure 5. The design of the NN is conducted through literature review and experiments, this is detailed in later sections. Before training a model, the imagery is pre-processed primarily to reduce computational load; this process is detailed below.

Imagery Pre-Processing

As can be seen in Figure 9, the sun and clouds are easily visible, which would (in concept) provide features for a ML method to use to predict irradiation or solar power. Additionally, there are buildings and trees visible in the images - since these are constantly in the same positions throughout the period of interest and thus throughout the dataset, their presence should not interfere with the prediction.

To address research question 1, specifically 1(c), the following requirements are placed on the imagery dataset due to the computational resources available:

1. The size of the original dataset must be limited to 100 GB due to the disk space available.
2. The size of the pre-processed dataset must be limited to 7 GB due to the limited memory that can be consumed when subsequently processing the dataset in Python using NumPy.

The first requirement above is easily satisfied since the dataset is 30 GB in its entirety. The second is satisfied by converting the images to grayscale from color. This conversion may also improve predictions as grayscale imagery has been shown to be more effective for object recognition using CNNs than colored imagery [BLC+16]. In addition, the resolution of the imagery is reduced from 1536×1536, also in order to satisfy the second requirement. A case study was designed to compare the predictive performance of the model when using 64×64 resolution imagery, as compared to 128×128. The results are presented in Case Study 1 in Chapter 5. The pre-processed image of 64×64 resolution captured on 01 May 2021 at 11:00 is provided in Figure 13.

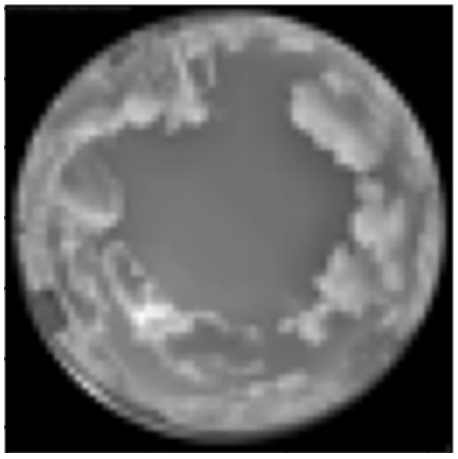

Figure 13: Pre-processed all sky image captured by the camera at 11:00 on 01 May 2021 [Del22]

The pixel values in images conventionally range from 0 (darkest) to 255 (brightest). As mentioned in Chapter 3, ANNs perform better when dealing with feature values between 0 and 1 [Ger19]. Therefore, the images are normalized using min-max normalization before being used as inputs to the ANN.

Time Horizon of Predictions

Since this project is focused on nowcasting, and given the highest frequency of data available is once every 10 minutes (as mentioned in Chapter 3), this is also the frequency chosen to make predictions. However, another related question is: how many data points in the past should be used to predict for the future? This is addressed through Case Study 2 in Chapter 5, where the performance of using three previous images (measured over 30 minutes) vs. only a single image to predict GHI 10 minutes in the future is compared. Both Case Study 1 and 2 are conducted using a CNN architecture for making the GHI prediction. The design of this network is discussed below.

Neural Network Architecture

As mentioned in Chapter 3, CNNs are adept at finding patterns and making predictions from images. Therefore it is chosen to develop a CNN to forecast GHI from the visual dataset. At this stage, there are a variety of design choices to be made; a CNN architecture can be feasibly any NN containing convolutional layers. A typical architecture will contain stacks of convolutional layers with an activation function, followed by a pooling layer, and optionally a batch normalization layer [Ger19]. A FNN with multiple FC layers (each with a specified activation function) is stacked on top to make predictions. If a single value is to be predicted, as is the case in this study, the final layer will have only 1 output. A generic CNN is illustrated in Figure 4. For this project, certain aspects of the architecture were fixed based on literature and best practices. These are detailed below:

- **Choice of activation function**: the ReLU function is chosen for the convolutional layers and the FC (except for the final layer) due to its low computational requirements, faster training time, non-saturating nature which prevents vanishing gradients, and ubiquity in image interpretation applications [Beh03, KSH12]. In the final layer, a Sigmoid activation is used to ensure the output is within $[0, 1]$, since the network is trained to make min-max normalized predictions.
- **Convolutional layer hyperparameters**: there is a number of hyperparameters that can be set differently for each convolutional layer in the network. Their choices are primarily justified by an underlying principle of CNNs, that the image should shrink but get 'deeper' as it moves forward through the network [Ger19]. The depth (or number of *channels*) initially for a colored image is 3 (RGB) and for a grayscale image is 1; increasing the channels enables the network to notice more abstract features. Simultaneously, the image size should decrease in order to reduce the number of overall features; therefore the most important ones are selected to be used by the FNN at the top of the network. Thus it is advised that the size of the image generally decreases as it progresses through the forward pass. The following hyperparameters were set based on this philosophy, and taking inspiration from established architectures such as *LeNet* and *AlexNet* [LBBH98, KSH12].
  - **Kernel size**: by convention, the kernel size is set to odd numbers. It is suggested to avoid using very large kernels (especially when applied to relatively low resolutions such as 64×64) as this risks losing pertinent features. Additionally, using multiple layers with smaller kernels is less computationally expensive than

using a single layer with a large kernel [Ger19]. Therefore 3×3 kernels are used for the convolutional layer; this also matches the choice made in AlexNet for layers dealing with comparable image sizes.

- **Number of output channels**: The number of channels (c) of a grayscale input image is 1, as mentioned above, so a 64×64 resolution image is represented as a tensor with dimensions 64×64×1 ($d_{in} \times d_{in} \times c$). A convolution layer makes the image deeper by adding channels, and this number of output channels (c') can be selected. As a result, a kernel of dimension $k \times k \times c \times c'$ is applied to the input image, yielding an output feature map of dimension $d_{out} \times d_{out} \times c$ (the calculation of d_{out} is described below).
- **Stride and padding**: setting these in a coordinated way makes it easy to control the size of the image (not its depth) that is output from a layer relative to its input. Equation 11 defines this relationship.

$$d_{out} = \left[\frac{d_{in} + 2p - k}{s}\right] + 1 \tag{11}$$

Here d_{out} is the output size (for a square image, this is the length of the side of the image), d_{in} is the input size, p is the padding, k is the kernel size, and s is the stride. To preserve the edges of the image, a low padding should be used; therefore $p = 1$ is chosen. In addition, $d_{out} = d_{in}$ can then be achieved with $s = 1$. This is the configuration used for all the convolutional layers except the first one, where $s = 2$ in order to significantly reduce the image dimension as it enters the network.

- **Pooling layers**: the two common methods for pooling are *average pooling* and *max pooling*. The former entails pooling pixels together by replacing them with one pixel of their average brightness. The latter does the same, but instead using the maximum brightness amongst the chosen pixels. Max pooling has been shown to be generally more effective for image classification [Cho17]. This can be understood through Figure 14. The contrast between light and dark pixels, particularly around edges, is greater when using max pooling. By replacing several pixels with their mean value, average pooling will 'smoothen' the edges and textures present in an image, losing potentially crucial features. Max pooling preserves such details and can therefore more effectively extract extreme features. The pooling layers also include the hyperparameters of kernel size and stride. To reduce the image size as it progresses through the network, a kernel size of 2×2 and a stride of 2 were chosen. This configuration (which is also employed in Figure 14) decreases the length of the side of the image by half when a pooling layer is applied.

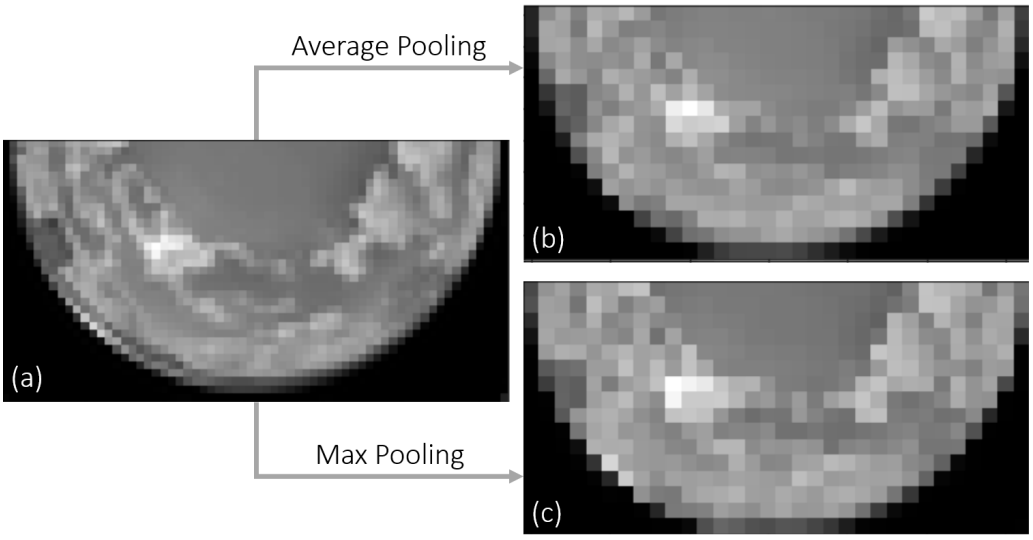

Figure 14: The bottom half of the sky image from 11:00 on 01 May 2021 in 3 forms; (a) the pre-processed image as shown in Figure 13, (b) the result of applying average pooling to this image (c) the result of applying max pooling to the image.

Evidently, many factors can be fine-tuned to configure the network architecture. A number of them, such as the hyperparameters of the convolutional and pooling layers listed above, can be optimized through Hyperparameter Optimization (HPO). This normally involves conducting either a grid search or a random search algorithm to find the hyperparameter configuration that leads to the best performance of the network. In order to avoid settling into local optima during HPO, the ANN should be initialized according to the practices described above; then the network is already configured in a theoretically appropriate way, although it has not been optimized for the specific problem.

Due to time constraints, a systematic HPO was deemed out of the scope of this thesis, however, a case study was conducted to justify the size of the CNN that is ultimately used.

Case Study 3 (with the results presented in Chapter 5) compares the performance of a relatively simple CNN with 3 convolutional layers (hereafter referred to as *CNN-base*) to a more complex architecture (*CNN-Alex*) with 5 convolutional layers and 3 FC layers that mimics AlexNet [KSH12]. The architecture with better performance in this comparison is selected to continue with for the remainder of the investigation. CNN-base is depicted in Figure 15 and Table 3.

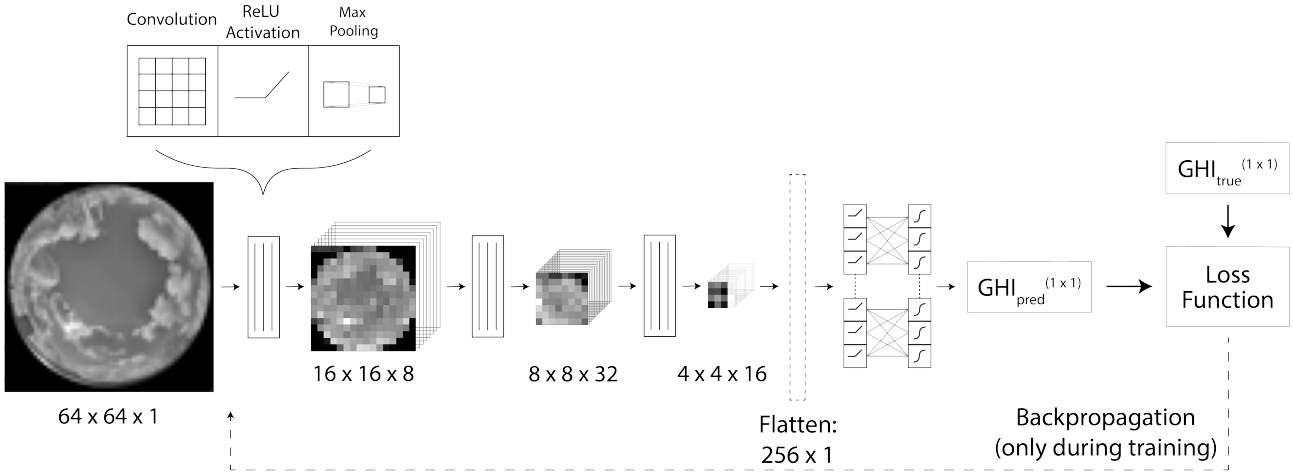

Figure 15: The CNN-base architecture investigated in Case Study 3.

Table 3: CNN-base architecture.

Layer	Type	Input Size	Kernel Size	Stride	Padding	Activation	Output Size
C1	Convolutional	64×64×1	3×3	2×2	1×1	ReLU	32×32×8
P2	Max. Pooling	32×32×8	2×2	2×2	0×0	-	16×16×8
C3	Convolutional	16×16×8	3×3	1×1	1×1	ReLU	16×16×32
P4	Max. Pooling	16×16×32	2×2	2×2	0×0	-	8×8×32
C5	Convolutional	8×8×32	3×3	1×1	1×1	ReLU	8×8×16
P6	Max. Pooling	8×8×16	2×2	2×2	0×0	-	4×4×16
Flatten							
Layer	Type	Input Size	Num. of Neurons	-	-	Activation	Output Size
F7	Fully Connected	256×1	64	-	-	ReLU	64×1
F8	Fully Connected	64×1	1	-	-	Sigmoid	1×1

Training is conducted using batches, where, as described in Chapter 3, the outputs (and prediction losses) of a batch of samples is calculated before updating the model parameters. The training process of this network from a batch GHI^B of images and paired GHI labels is described as follows.

1. GHI^B has a batch size S_B, therefore the batched input has dimension $S_B \times 64 \times 64 \times 1$. Each image in this batch (of dimension $64 \times 64 \times 1$) is passed through the convolutional layer, followed by ReLU activation, and finally a max. pooling layer.

   (a) Each neuron in the convolutional layer applies the function 12 to the input image.

$$z_{i,j,k} = b_k + \sum_{u=0}^{f-1} \sum_{v=0}^{f-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \quad \text{with} \begin{cases} i' = i \times s + u \\ j' = j \times s + v \end{cases} \tag{12}$$

   $z_{i,j,k}$ is the output of the neuron at row i, column j in feature map k of the convolutional layer l. s is the stride, considered identical for the horizontal and vertical directions for all the models in this thesis, and f is the height and width of the receptive (again considered identical as 64 due to the square images). $f_{n'}$ is the number of channels in the image ($f_{n'} = 1$ in this case). $x_{i',j',k'}$ is the output of the neuron at

location row i', column j' in feature map k' in layer l-1. b_k is the bias applied to feature map k, in layer l. Finally, $w_{u,v,k',k}$ is the connection weight between a neuron in feature map k of layer l and its input located at u, v in feature map k'.
With the hyperparameters given in Table 3, this results in a feature mapping of dimension $32 \times 32 \times 8$ from the first convolutional layer C1.

(b) The ReLU activation function is visualized in Figure 2 and expressed in Equation 13.

$$ReLU(z) = \max(0, z) \tag{13}$$

This is applied to every output $z_{i,j,k}$ of a layer l as it enables fast computations and avoids vanishing gradients. Therefore the dimension of the feature mapping is unchanged.

(c) The max pooling layer, as depicted in Figure 14 is subsequently applied and results in a feature mapping of dimension $16 \times 16 \times 8$.

The above three steps (convolution-activation-pooling) are applied two more times, resulting in a $4 \times 4 \times 16$ mapping.

2. The feature space is then flattened from $4 \times 4 \times 16$ to 256×1 and used as input to a fully-connected network. Layer F7 applies the ReLU activation to the weighted sum of the neuron inputs, as expressed in Equation 14.

$$z_i = ReLU(\sum_k w_{k,i} \cdot z_k + b_i) \tag{14}$$

Here z_i represents the output of neuron i in layer l, z_k is the value of input feature with index k, and $w_{k,i}$ is the connection weight between feature k and neuron i. In layer F7, for example, $k\epsilon[1, 256]$ and $i\epsilon[1, 64]$.

3. Layer F8 applies a similar procedure to the output features of layer F7, however applying the Sigmoid activation instead, given by Equation 15.

$$z = \frac{1}{1 + e^{(\sum_k w_k \cdot z_k + b)}} \tag{15}$$

Here, since only 1 neuron is employed, there is a 1×1 output which represents the GHI predicted by the NN, GHI_{pred}.

4. GHI_{pred} is compared to the true GHI at that moment, GHI_{true}, using a loss function such as the MSE (given in Equation 2). For a training batch GHI^B, the MSE can be rewritten as follows:

$$MSE(GHI^B) = \frac{1}{S_B} \sum_{i=1}^{S_B} (GHI^{B,i}_{pred} - GHI^{B,i}_{true})^2 \tag{16}$$

Using Mean Absolute Error (MAE) as a loss function, on the other hand, would mean computing Equation 17 for the batch GHI^B.

$$MAE(GHI^B) = \frac{1}{S_B} \sum_{i=1}^{S_B} |GHI^{B,i}_{pred} - GHI^{B,i}_{true}| \tag{17}$$

5. With one of the above loss functions computed for a training batch, the backpropagation algorithm described in Chapter 3 is applied for updating the model parameters (weights and biases in each layer). The AdamW optimizer alluded to in Chapter 3 is employed to find the parameters that minimize the loss given by either Equation 16 or 17. The NN then applies steps 1 to 5 for the next training batch, until the entire training set has been used for updating the model parameters. This signifies the end of one training epoch. This iterative process is continued for several epochs, shuffling the batches every time until an optimized model is found. At the end of every epoch, the loss on a validation set is also calculated, to track the performance of the NN. To avoid overfitting, early stopping is applied here, whereby training is halted if the validation loss does not decrease for 5 consecutive epochs.

6. During testing, the NN that has been learned on the entire training set is fixed (model parameters are no longer updated using backpropagation), and the predictive performance is quantified using a loss function such as MSE or MAE on the test set. Therefore Equations 16 or 17 are applied to the test set to compute $MSE(GHI^{test})$ or $MAE(GHI^{test})$.

4.1.2 Forecasting from Meteorological Data

The other modality that is employed in this thesis is numerical data measured by meteorological instruments. The procedure for predicting GHI is summarized in Figure 16.

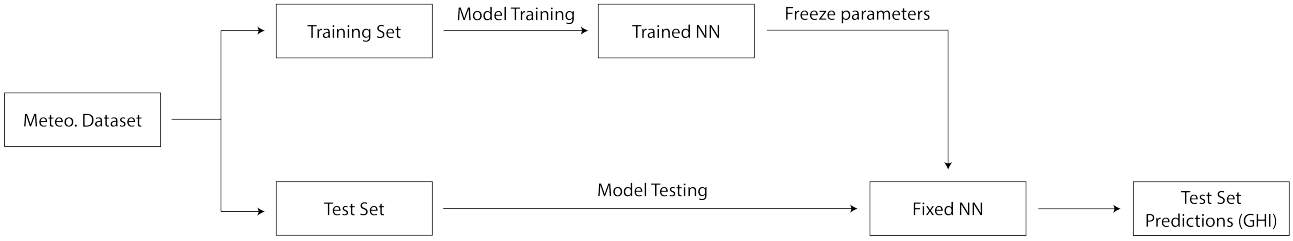

Figure 16: Procedure followed for predicting GHI from meteorological sensor data using a NN.

Eight different variables (listed in Section 3.7) are used. Computationally, since each of these is just a single float per instance, there is no strain on the requirements alluded to in Section 4.1.1. The samples are normalized as before, using min-max normalization.

Neural Network Architecture

Forecasting from meteorological data is conducted using a FNN (hereafter referred to as *FNN-base*). Its architecture is depicted schematically in Figure 17 and detailed in Table 4.

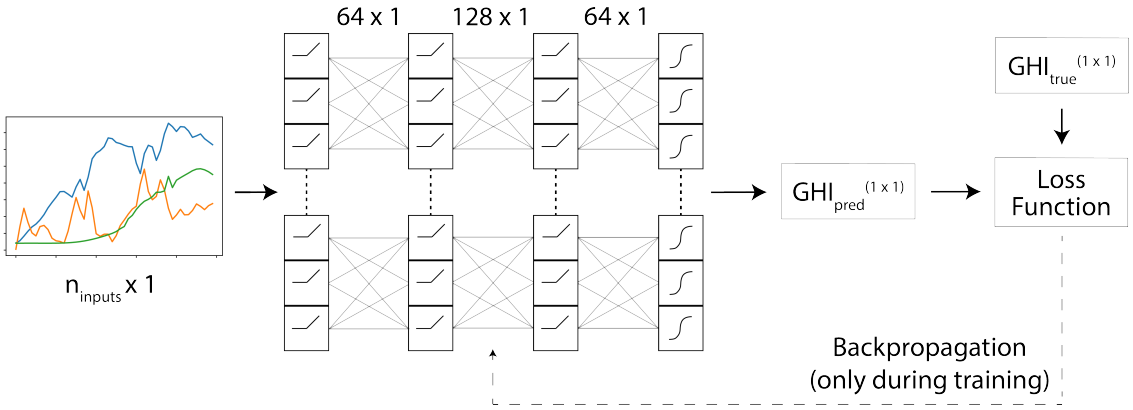

Figure 17: The FNN-base architecture used for predicting GHI from meteorological sensor data.

Table 4: FNN-base architecture.

Layer	Type	Input Size	Num. of Neurons	Activation	Output Size
F1	Fully Connected	$n_{inputs}\times1$	64	ReLU	64×1
F2	Fully Connected	64×1	128	ReLU	128×1
F3	Fully Connected	128×1	64	ReLU	64×1
F4	Fully Connected	64×1	1	Sigmoid	1×1

The functioning of this network is very similar to the process described for CNN-base. It is briefly explained as follows.

1. During the forward pass, the input features for a single sample are provided as a flat tensor of dimension $n_{inputs} \times 1$. Here n_{inputs} is the number of features; if 8 meteorological variables measured at 3 time instances are used, for example, then $n_{inputs} = 24$. Again, mini-batches are employed during training, so a batch of size S^B would have dimension $S^B \times n_{inputs} \times 1$.

2. The first layer contains 64 neurons with a weight for each input and a single bias term, therefore a $(64 \times n_{inputs})$ matrix transformation is applied to each sample. The following layer contains 128 neurons, then a layer of 64 neurons, and finally the output layer of 1 neuron. This results in a single number as the predicted GHI

perceived from the meteorological features. The neurons in each of the layers, except the output layer, uses ReLU activation, applying Equation 14. The neurons in the output layer employ Sigmoid activation, given in Equation 15, to ensure the prediction is within $[0, 1]$.

3. The error of the prediction is computed using one of the loss functions expressed in Equations 16 and 17.

4. While training, backpropagation with AdamW is applied to update the weights in each neuron to minimize the loss of a training batch. Once the algorithm goes through the entire training set, validation loss is calculated on a reserved set of samples, and the next epoch is begun.

5. The above steps are repeated until the model validation loss no longer decreases, as determined by early stopping.

This procedure yields the NN optimized to predict GHI. After training, the model is applied to the samples in the test set. In the testing phase, no backpropagation is involved; the model is fixed. The predicted GHI can then be used to calculate PV power output via Equation 9 and system costs by solving the DCOPF thereafter. Case Study 4 was conducted to analyze the effect of using 3 vs. 1 previous instances as meteorological features.

4.1.3 Uni-Modal Baselines

The described neural network architectures outlay a method for predicting GHI from two different modalities (imagery and sensor data). These predictions then can be compared to gain insight into which one performs better and thereby answer Research sub-question 1(e); this is further elucidated in Case Study 5. To compare UM algorithms to MM ones in other studies, two baseline models are developed from the UM methodology.

1. The naive ensemble baseline (*UM-base1*): here, the predictions from imagery and meteorological data are combined as an average, exemplifying the simplest late fusion method to combine the two modalities. This is summarized in Equation 18.

$$GHI_{UM} = \frac{1}{2} \cdot (GHI_{UM1} + GHI_{UM2}) \tag{18}$$

   GHI_{UM1} and GHI_{UM2} are the predictions from the individual modalities, and GHI_{UM} is their weighted combination.

2. The weighted ensemble baseline (*UM-base2*): in this setting the predictions from the two modalities are used to fit a linear regressor as described in Equation 19.

$$GHI_{UM} = c_1 \cdot GHI_{UM1} + c_2 \cdot GHI_{UM2} \tag{19}$$

   This can therefore be considered a slightly more 'intelligent' ensemble as compared to the naive method, due to the variability of the weights c_1 and c_2 based on the predictive performance of either modality.

4.2 Multi-Modal Learning for Power Forecasting

Employing multi-modal learning in the context of power forecasting involves changing the neural network architecture such that features from both modalities are used simultaneously to make, and learn from predictions. To effectively answer Research Question 2, the features selected from each modality and the method used to combine these features must be investigated. For the former topic, some possible options include:

1. Combine the images and meteorological data directly, without any feature extraction.

2. Use a CNN to extract features from the imagery and a FNN to extract features from the meteorological data and combine these features.

3. Use a NN to extract features from the meteorological data and combine these with the image directly.

4. Use a NN to extract features from the imagery and combine these with the meteorological data directly.

Option 1 and 3 do not make sense in practice because of the nature of the images - they have a high dimensionality and a often lot of irrelevant features. The value of CNNs is in extracting important aspects of an image such as patterns and edges, and disregarding the rest. Therefore, to avoid the curse of dimensionality (whereby an exponentially larger dataset would be required to learn from a large, sparse feature space), only options 2 and 4 are pursued. Case Study 6 investigates the options in terms of their effectiveness at predicting GHI. Figure 18 depicts

the proposed architecture used when features are only extracted from the images, and these are combined with the meteorological data (hereafter referred to as the *MM4* architecture).

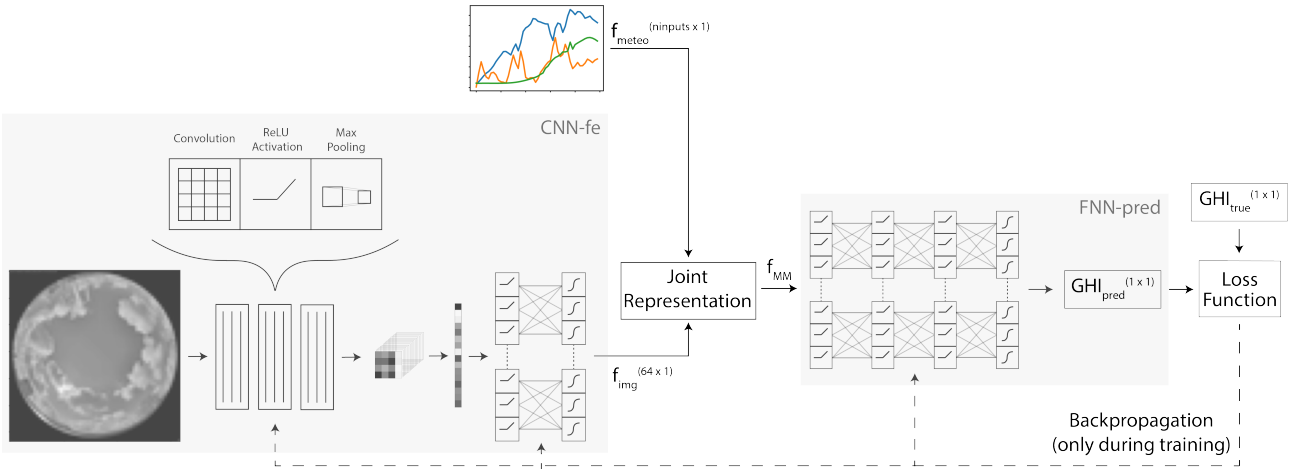

Figure 18: Proposed multi-modal learning architecture MM4 to predict GHI from imagery and meteorological data, using a CNN for feature extraction from the imagery (and no NN for extracting features from the meteorological data).

The proposed MM4 architecture is described briefly below.

1. A feature extraction network (denoted in Figure 18 as *CNN-fe*) is applied to an image input during the forward pass. It consists of 3 stacks of convolutional, ReLU activation, and max pooling layers, followed by a FC layer with ReLU activation and another with Sigmoid activation. This architecture is shown in detail in Table 5.

   Table 5: CNN-fe architecture used for feature extraction from imagery within the MM4 neural network.

Layer	Type	Input Size	Kernel Size	Stride	Padding	Activation	Output Size
C1	Convolutional	64×64×1	3×3	2×2	1×1	ReLU	32×32×8
P2	Max. Pooling	32×32×8	2×2	2×2	0×0	-	16×16×8
C3	Convolutional	16×16×8	3×3	1×1	1×1	ReLU	16×16×32
P4	Max. Pooling	16×16×32	2×2	2×2	0×0	-	8×8×32
C5	Convolutional	8×8×32	3×3	1×1	1×1	ReLU	8×8×16
P6	Max. Pooling	8×8×16	2×2	2×2	0×0	-	4×4×16
Flatten							
Layer	Type	Input Size	Num. of Neurons	-	-	Activation	Output Size
F7	Fully Connected	256×1	64	-	-	ReLU	64×1
F8	Fully Connected	64×1	64	-	-	Sigmoid	64×1

   The above structure mimics CNN-base, with the exception of the last layer, which has 64 neurons instead of just 1. This results in an output feature vector ($\mathbf{f_{img}}$) with dimension 64×1 for a single sample.

2. The meteorological features ($\mathbf{f_{meteo}}$) are fused with $\mathbf{f_{img}}$ in joint representation either using concatenation or bilinear pooling, depending on the case study. Concatenation is expressed in Equation 20 while bilinear pooling is given by Equation 21.

$$\mathbf{f_{MM}}^{((64+n_{inputs})\times 1)} = \mathbf{f_{img}}^{(64\times 1)} \frown \mathbf{f_{meteo}}^{(n_{inputs}\times 1)} \tag{20}$$

$$\mathbf{f_{MM}}^{((64+1)\times(n_{inputs}+1))} = \begin{bmatrix} \mathbf{f_{img}}^{(64\times 1)} \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{f_{meteo}}^{(n_{inputs}\times 1)} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{f_{img}} & \mathbf{f_{img}} \otimes \mathbf{f_{meteo}} \\ 1 & \mathbf{f_{meteo}} \end{bmatrix} \tag{21}$$

   If the latter (bilinear pooling) is used, the resultant feature vector is flattened before continuing, since the FC network that follows requires a one-dimensional vector. $\mathbf{f_{MM}}$ in this context, therefore, ultimately has a size of $(65 \cdot (n_{inputs} + 1) \times 1)$.

3. GHI is predicted from the fused feature vector, $\mathbf{f_{MM}}$, using a FNN similar to FNN-base, referred to as *FNN-pred* in Figure 18. It is detailed in Table 6.

Table 6: FNN-pred network used for predicting GHI from multi-modal feature vector $\mathbf{f_{img}}$ within MM4 NN.

Layer	Type	Input Size	Num. of Neurons	Activation	Output Size
F1	Fully Connected	Dependent on fusion method	64	ReLU	64×1
F2	Fully Connected	64×1	128	ReLU	128×1
F3	Fully Connected	128×1	64	ReLU	64×1
F4	Fully Connected	64×1	1	Sigmoid	1×1

The first layer notably receives inputs with size dependent on the fusion method (concatenation or bilinear pooling) applied. As discussed above, these will result in different sizes for f_{MM}, which is the input to FNN-pred. The proceeding layers F2-F4 function as previously described concerning FNN-base.

4. During training, the loss (computed by either MSE or MAE, specified in the case studies) compares GHI_{pred} to GHI_{true}. This is then differentiated according to the backpropagation algorithm and the gradients are passed backwards to update the weights of all parameters within CNN-fe and FNN-pred.

The *MM2* architecture, which also uses a NN for feature extraction from the meteorological inputs, employs a FNN between that modality and the joint representation stage. That is the only difference between the two architectures. MM2 is illustrated in Figure 19.

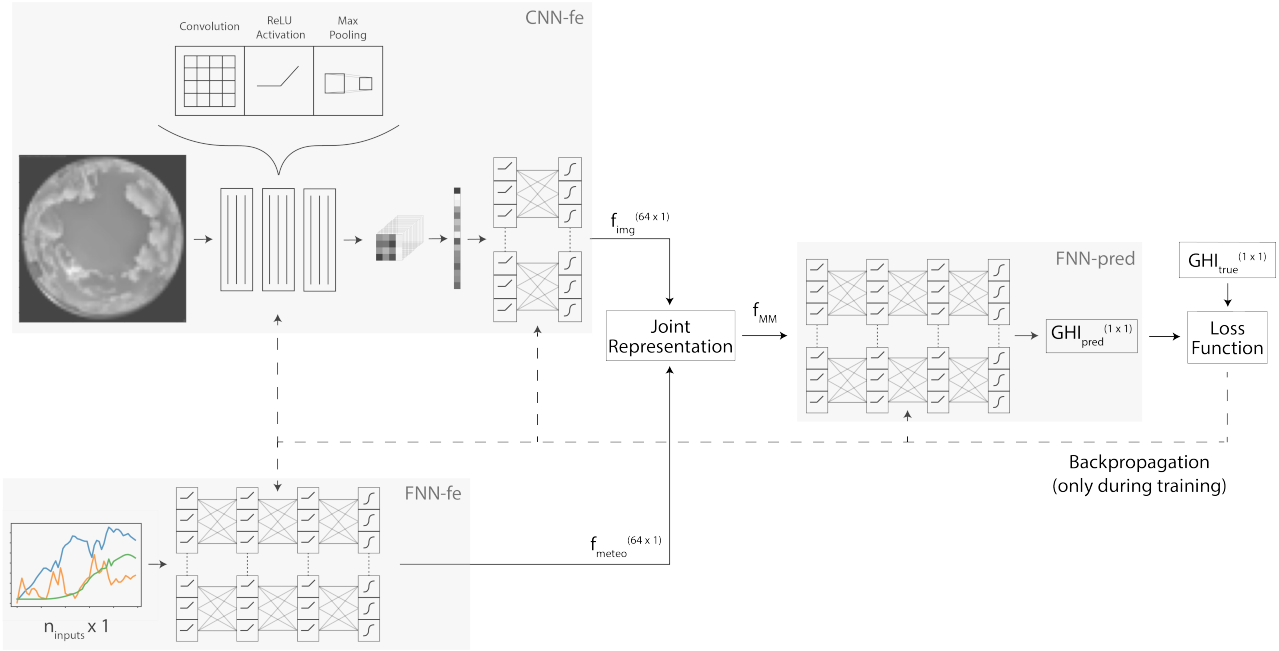

Figure 19: MM2 Multi-modal learning architecture to predict GHI from imagery and meteorological data, using a CNN for feature extraction from the imagery and a FNN for feature extraction from the meteorological data.

The FNN used for feature extraction from the meteorological data, FNN-fe, is identical in architecture to the first three layers of FNN-base. This is described explicitly in Table 7.

Table 7: FNN-fe architecture.

Layer	Type	Input Size	Num. of Neurons	Activation	Output Size
F1	Fully Connected	n_{inputs}×1	64	ReLU	64×1
F2	Fully Connected	64×1	128	ReLU	128×1
F3	Fully Connected	128×1	64	ReLU	64×1

Therefore, MM2 involves an equal number of features extracted from both the modalities ($\dim(\mathbf{f_{img}}) = \dim(\mathbf{f_{meteo}})$ = (64 × 1)), however, this is not necessarily the optimal configuration. In addition, as described in Chapter 3, fea-

tures could be extracted from different stages of the network and fused. There are, in this sense, unlimited design tweaks to the MM learning architecture that can be investigated but are out of the scope of this thesis project.

Regarding the method used to combine the features from either modality, the two prevalent techniques for joint representation identified in Chapter 3 (concatenation and bilinear pooling) are compared in Case Study 7, within the MM4 architecture.

Finally, the optimal MM model from these two studies (either MM2 or MM4) is tested in comparison to UM learning. Case Study 8 includes the predictive performance of the two baseline ensemble UM methods (UM-base1 and UM-base2), a persistence forecast, as well as that of one of the MM methods. In order to comprehensively answer Research Question 3 (specifically addressing 3c), Case Study 9 compares the persistence forecast against the performance of one UM baseline (UM-base2) trained with past GHI data as a feature. Case Study 10 considers the prediction of wind power in a similar manner to Case Study 8 - comparing the performance of UM and MM algorithms.

4.3 Economically Optimized Predictions with End-to-End Learning

The methodologies presented thus far have investigated the effectiveness of different neural network architectures at predicting GHI. In order to use these predictions in an energy system, the GHI must be used to calculate PV power output (in Watts) and an optimal power flow problem must be solved using this value. This allows the focus of the investigation to be shifted from power forecasting to the 'task': the cost of running an energy system. The method followed to predict the task from MM data is shown schematically in Figure 20.

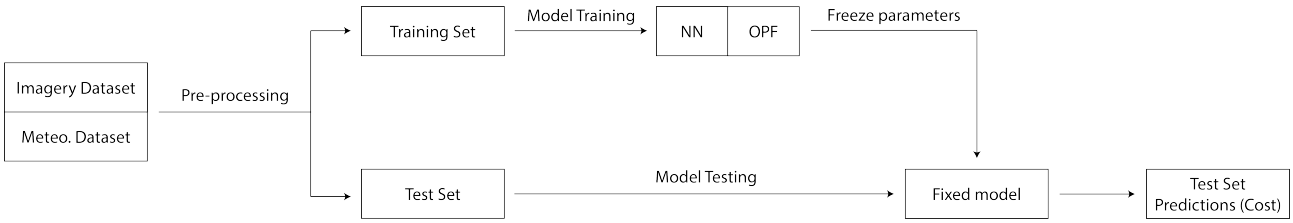

Figure 20: Procedure followed for minimizing system cost from imagery and meteorological sensor data using a NN. As before, GHI (and/or Wind power) is predicted, however, it is now only an intermediate prediction. The model is trained on minimizing cost.

For the implementation of this phase of the project in Python, particular credit is given to Ali Rajaei, a researcher at the IEPG group, TU Delft. The energy system costs can be solved via a pair of DCOPF formulations, given values for load demand, PV, and wind power, as was briefly introduced in Chapter 3. The first optimization, referred to as *DCOPF-Schedule*, solves for a generation schedule (and its respective costs) using the forecasted renewable power sources. This optimization is defined as follows.

$$\text{minimize} \quad C_{sch} = C_{gen} + C_{inf} = \sum_{b \epsilon B} \left(c_b^g(P_b^g) + \gamma_1 \lambda_{1_b} + \gamma_2 \lambda_{2_b} + \gamma_3 \lambda_{3_b} + \gamma_4 \lambda_{4_b} \right) \tag{22a}$$

$$\text{subject to} \quad -P_b^d + P_b^g + (P_b^{PV} - \lambda_{3_b}) + (P_b^W - \lambda_{4_b}) = \lambda_{1_b} - \lambda_{2_b} + P_b^f \qquad \forall b \epsilon B, \tag{22b}$$

$$\lambda_{3_b} \leq P_b^{PV}, \tag{22c}$$

$$\lambda_{4_b} \leq P_b^W, \tag{22d}$$

$$|P_r^f| \leq P_r^{f,lim}, \tag{22e}$$

$$P_r^f = \frac{\delta_{r,i} - \delta_{r,j}}{x_r} \qquad \forall r \epsilon R, \tag{22f}$$

$$P_b^{g-} \leq P_b^g \leq P_b^{g+} \tag{22g}$$

Table 8: Variables involved in DCOPF-Schedule

Variable	Description
C_{sch}	Scheduling cost
C_{gen}	Generation cost
C_{inf}	Infeasibility cost
R	Number of lines
B	Number of nodes
P_b^g	Thermal power injection at node b
c_b^g	Cost of power generation at node b
λ_1 & λ_2	Auxiliary variables for imbalance
γ_1 & γ_2	Prices on thermal generation imbalance
λ_3 & λ_4	PV and Wind power curtailment
γ_3 & γ_4	Prices on PV and Wind curtailment
P_b^d	Power withdrawal at node b
P_b^{PV}	Predicted PV power generation at node b
P_b^W	Predicted Wind power generation at node b
P_b^f &	Net power flow from all lines into node b
P_r^f & $P_r^{f,lim}$	Power flow and power flow limit in line r
$\delta_{r,i}$ & $\delta_{r,j}$	Phase angles at line r
P_b^{g-} & P_b^{g+}	Lower and upper limits of generation at node b

Equation 22b expresses the node balance at each of the buses in the power system - ensuring that power injection and withdrawal (P^g and P^d) are balanced by the power flows through lines connected to the respective bus. The constraints given by Equations 22c and 22d express that renewables curtailment cannot be greater than the predicted generation. Equation 22e represents the maximum thermal limit for the lines, and 22f relates power flow in the branches to the respective phase angles. Finally, equation 22g gives the generator limits.

When this DCOPF formulation is integrated as a layer in a NN, P_b^{PV} and P_b^W are included as learnable parameters. There are therefore varied by the NN during backpropagation according to the loss ultimately calculated (which also depends on a subsequent optimization layer). Load demand (P_b^d) is always assumed to be deterministic in this project. After solving DCOPF-Schedule for the various costs and the generation schedule (P_d^g), the true values for renewable generation are compared to the forecast in order to quantify the cost of inaccurate predictions. This leads to a redispatch of power sources to account for the mismatch between the predicted and actual renewable generation. This redispatch is formulated in *DCOPF-Redispatch* given below.

$$\text{minimize} \quad C_{RD} = C_{RD,gen} + C_{RD,inf} = \sum_{b \epsilon B} \left(c_{up}(dP_b^{g+}) + c_{down}(dP_b^{g-}) + \gamma_5 \lambda_{5_b} + \gamma_6 \lambda_{6_b} + \gamma_7 \lambda_{7_b} + \gamma_8 \lambda_{8_b} \right) \tag{23a}$$

$$\text{subject to} \quad -P_b^d + (P_b^g + \Delta P_b^g) + (P_b^{PV,act} - \lambda_{7_b}) + (P_b^{W,act} - \lambda_{8_b}) = \lambda_{5_b} - \lambda_{6_b} + P_b^{f,RD} \qquad \forall b \epsilon B, \tag{23b}$$

$$\Delta P_b^g = dP_b^{g+} - dP_b^{g-}, \tag{23c}$$

$$\lambda_{7_b} \leq P_b^{PV,act}, \tag{23d}$$

$$\lambda_{8_b} \leq P_b^{W,act}, \tag{23e}$$

$$|P_r^f| \leq P_r^{f,lim}, \tag{23f}$$

$$P_r^f = \frac{\delta_{r,i} - \delta_{r,j}}{x_r} \qquad \forall r \epsilon R, \tag{23g}$$

$$P_b^{g-} \leq P_b^g + \Delta P_b^g \leq P_b^{g+}, \tag{23h}$$

$$dP_b^{g+} \leq dP_b^{g+,max}, \tag{23i}$$

$$dP_b^{g-} \leq dP_b^{g-,max} \tag{23j}$$

Table 9: Additional variables involved in DCOPF-Redispatch

Variable	Description
C_{RD}	Redispatch cost
$C_{RD,gen}$	Generation cost in redispatch
$C_{RD,inf}$	Infeasibility cost in redispatch
dP_b^{g+}	Thermal power increase at node b
dP_b^{g-}	Thermal power reduction at node b
c_{up}	Cost of increasing thermal power generation at node b
c_{down}	Cost of reducing thermal power generation at node b
λ_5 & λ_6	Auxiliary variables for imbalance
γ_5 & γ_6	Prices on thermal generation imbalance
λ_7 & λ_8	PV and Wind power curtailment
γ_7 & γ_8	Prices on PV and Wind curtailment
$P_b^{PV,act}$	Actual PV power generation at node b
$P_b^{W,act}$	Actual Wind power generation at node b
$dP_b^{g-,max}$ & $dP_b^{g+,max}$	Lower and upper rate limits at node b

This optimization is analogous to the one formulated in Equations 22a to 22g. However, the inclusion of ΔP_b^g, c_{down} and c_{up} facilitate corrections to the shortages or surpluses in power generation attributed to inaccurate prediction. After solving DCOPF-Redispatch for minimized costs, the *system cost* C_{sys} can be calculated via Equation 24.

$$C_{sys} = C_{sch} + C_{RD} = (C_{gen} + C_{inf}) + (C_{RD,gen} + C_{RD,inf}) \tag{24}$$

C_{sys} is the cost that the neural network is now trained on using backpropagation. It is costly to both increase or decrease generation in real-time to maintain grid stability, therefore C_{RD} can be generally reduced by obtaining accurate predictions of the renewable generation. Depending on the exact formulation of the redispatch cost relative to scheduling cost (the values of $\gamma_1 ... \gamma_8$), however, minimizing prediction error may not universally minimize monetary cost. Due to the linear cost functions used here (represented by γ), it can be expected that the relationship between cost and prediction accuracy is linear. This curve is plotted in Figure 21.

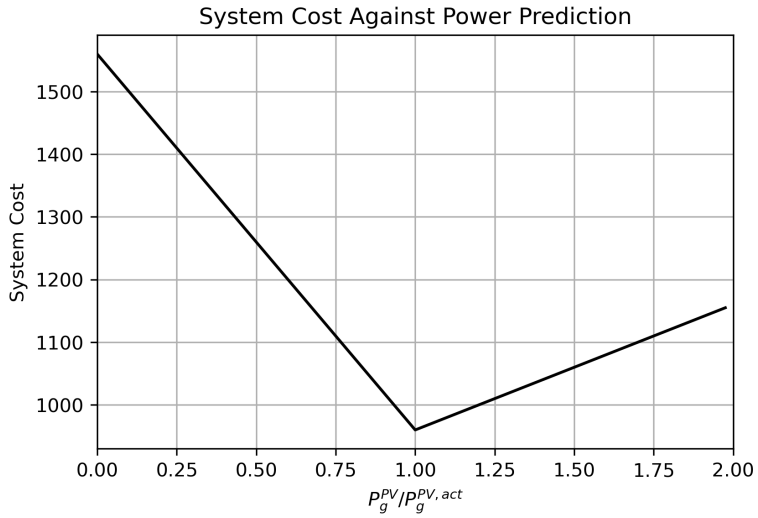

Figure 21: The expected variation of system cost based on renewable power forecast accuracy. This plot is for a system with 145 MW of load demand and 25 MW of PV generation.

The magnitude of the cost on the y-axis depends on the load demand and the amount of renewable generation available in the system. The plot in Figure 21 is for a moment with 145 MW of load and 25 MW of PV generation.

4.3.1 Proposed MM E2E Architecture

A NN which predicts GHI using the MM4 architecture and subsequently solves the pair of DCOPF problems is illustrated in Figure 22. This proposed architecture, named MM4-E2E, includes a single renewable generation source

(PV).

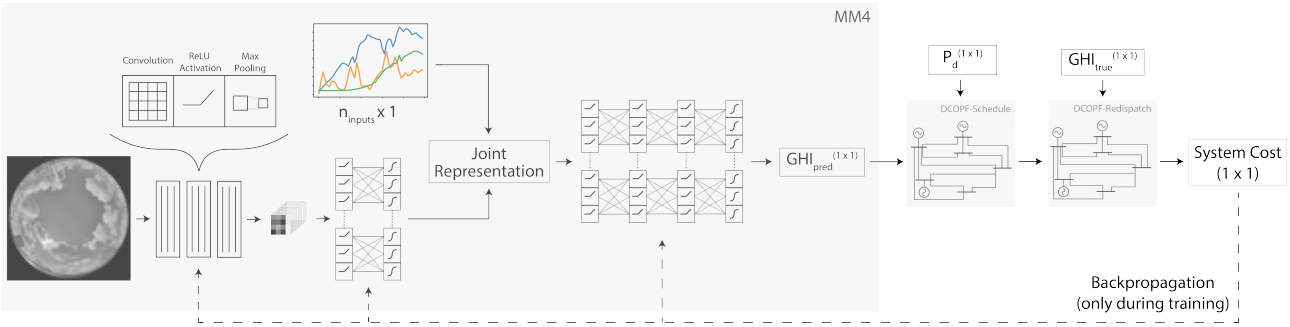

Figure 22: Proposed MM4-E2E architecture to minimize system cost, using the MM4 architecture for PV forecasting.

On the other hand, Figure 23 depicts a system with PV and wind generation. PV power is forecast as in the MM4-E2E architecture, using MM4, but now a wind forecast is also included using FNN-base.

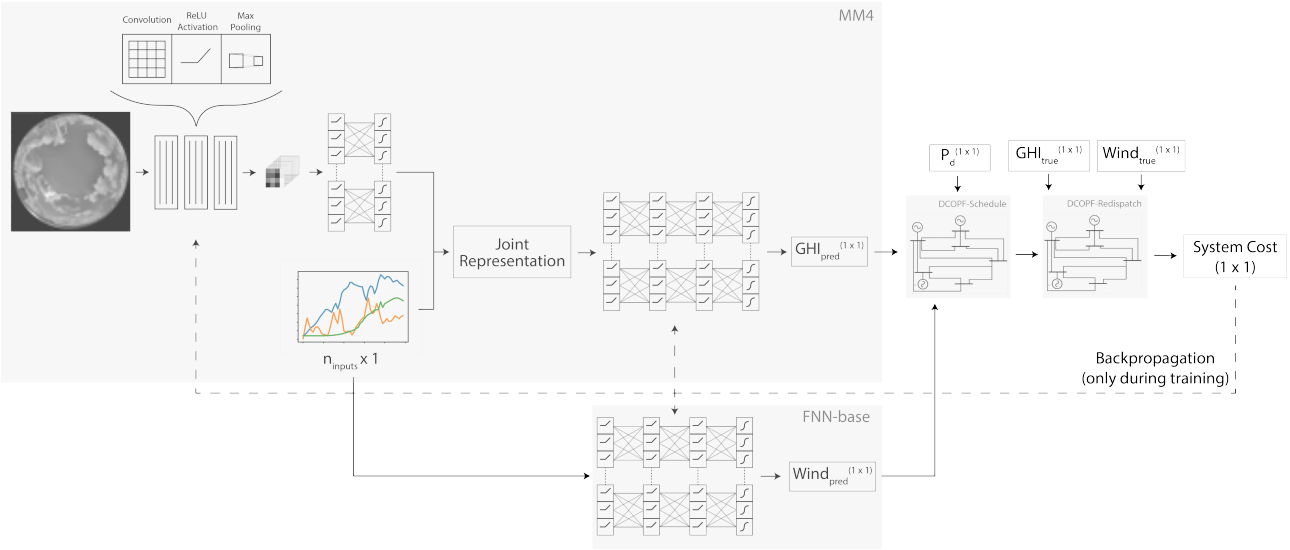

Figure 23: Proposed MM4-PVWind-E2E architecture to minimize system cost, using the MM4 architecture for PV forecasting and the FNN-base architecture for wind forecasting.

To assess this model against a sequential trained one, MM4-PVWind is designed. As above, it predicts PV from the MM4 architecture, and Wind from FNN-base.

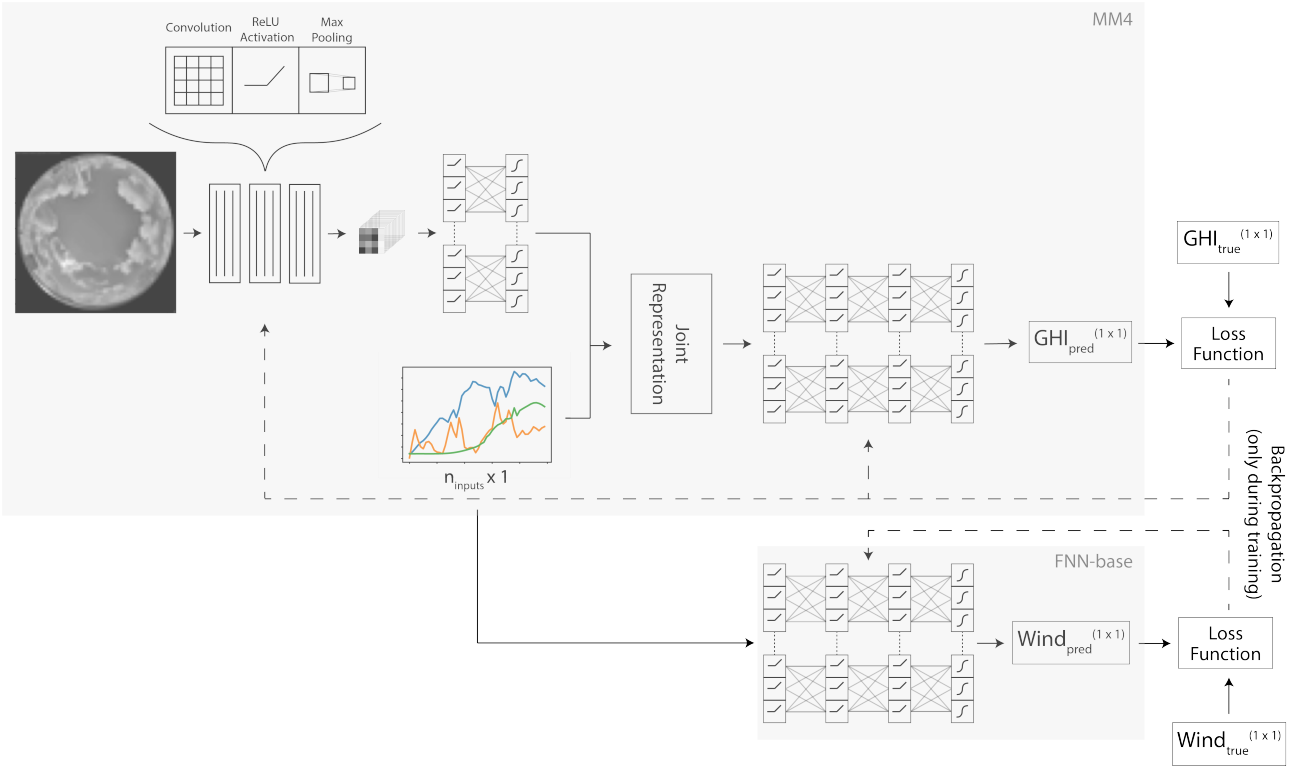

Figure 24: MM4-PVWind architecture to predict renewables generation, using the MM4 architecture for PV forecasting and the FNN-base architecture for wind forecasting.

Case Study 11 compares E2E and sequential learning in terms of C_{sys}. This study only considers PV generation, so the MM4-E2E architecture is compared with *MM4-Seq*, where the MM4 network is trained as before, on making predictions of GHI, and subsequently, the DCOPF is solved from the GHI forecast and load data. Also in this study, the UM baseline algorithms UM-base1 and UM-base2 are compared to MM4-Seq and MM4-E2E. Case Study 12 investigates MM4-PVWind-Seq and MM4-PVWind-E2E, which use two renewable sources (PV and Wind) rather than one. We expect that in both Case Studies 11 and 12, E2E learning benefits cost minimization, since it provides the NN with information about the cost function. Case Study 13 investigates whether this expectation is validated by the results of studies 11 and 12, and helps explain the benefits and pitfalls caused by E2E learning.

5 Case Studies

This chapter presents the several case studies used to investigate different aspects of the methodology and therefore answer the research questions. Designing a neural network, the data preparation, and the training workflow requires making certain design choices. The studies are designed along the most important of these design choices. The chapter begins with case studies that focus on the UM baseline models, then on the multi-modal ones, and finally on the end-to-end learning frameworks. A summary of the case studies and their respective aims is given in Table 10.

Table 10: A summary of the case studies conducted, the research questions they address, and their respective aims.

Case Study	Research Questions Addressed	Purpose
1	RQ1	Compare the error on predicting GHI from lower and higher resolution images.
2	RQ1	Compare the error on predicting GHI from 3 or 1 recent image.
3	RQ1	Investigate the optimal depth of CNN for predicting GHI from imagery.
4	RQ1	Compare the error on predicting GHI from 3 or 1 recent instance of sensor data.
5	RQ1	Compare the prediction of GHI and wind power from imagery via a CNN vs. from sensor data via a FNN.
6	RQ2	Investigate the effectiveness of extracting features from sensor data for making predictions with MM learning.
7	RQ2	Compare two methods for combining features from either modality during MM learning.
8	RQ3	Compare proposed MM learning model for predicting GHI against UM baselines.
9	RQ3	Investigate GHI forecasting using recent GHI sensor data as a feature.
10	RQ3	Compare proposed MM learning model for predicting wind power against UM baselines.
11	RQ4	Assess the proposed MM-E2E model (MM4-E2E) in terms of system cost in a system with PV generation.
12	RQ4	Assess the proposed MM4-E2E model in a system with PV and Wind generation.
13	RQ4	Compare the minimized costs from MM4-E2E to the expected (true) cost function.

5.1 Settings and Test Network

5.1.1 Power System

A simple power system - the IEEE 6-bus system with 3 generator and 3 load buses - is employed to model the functioning of a power system with renewable generation. The line diagram of this system is illustrated in Figure 25.

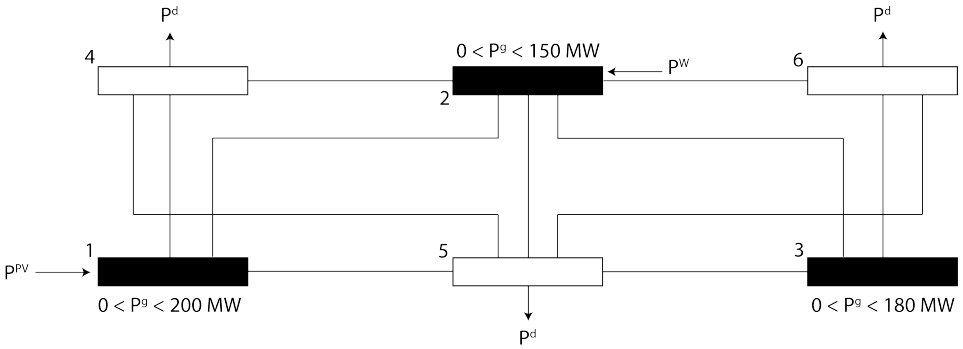

Figure 25: Line diagram of IEEE 6-bus system with PV (P^{PV}) and Wind (P^{W}) generation. The limits on thermal generation are labelled at their resective nodes. Each of the lines has a limit of 100 MW.

Thermal generators inject at buses 1,2,3, and load is drawn at buses 4,5,6. PV power is injected at bus 1, or Wind is injected at bus 2, or both, depending on the case study. Details of the parameters used in the DCOPF-Schedule and DCOPF-Redispatch optimizations are given in Table 11.

Table 11: Values of essential parameters (first described in Tables 8 and 9) used in the optimization problems.

Parameter	Value	Unit
P_1^{g-}	0	MW
P_1^{g+}	200	MW
P_2^{g-}	0	MW
P_2^{g+}	150	MW
P_3^{g-}	0	MW
P_3^{g+}	180	MW
c_1^g	12	Euro/MW
c_2^g	10	Euro/MW
c_3^g	8	Euro/MW
$P_r^{f,lim}$	100	MW
R	11	
B	6	
$\gamma_{1,2,5,6}$	10	Euro/MW
$\gamma_{3,4,7,8}$	0.1	Euro/MW

The details of the scale of load demand, PV generation, and wind generation are given below.

- Load is modeled from the dynamic load profiles of 1.5 million identical households (as described in Chapter 3), distributed unevenly over 3 buses (48% at bus 4, 28% at bus 5, and 24% at bus 6). During the time period of the available data (2021 and 2022) the maximum power load demand was 293MW.
- PV generation is based on the LG panel cited in Chapter 3, with a rated power of 440W and panel area of 2.2 m^2. Equation 25 is used to approximate the PV power output of a solar farm with N_{panels} panels of area A_{panel}, given their rate power P_{rated} and the GHI.

$$P_{PV} = GHI \times N_{panels} \times A_{panel} \times \frac{P_{rated}}{1000W/m^2} \tag{25}$$

  We assumed that 250,000 panels are used, resulting in a solar farm rating of 110MWp.

- Wind generation is based on the Vestas turbine also mentioned in Chapter 3, and on the existing OWEZ offshore farm [owe22]. The curve is approximated by a hyperbolic tangent function in order to calculate wind power from wind speed. The power curve and its approximation are depicted in Figure 26.

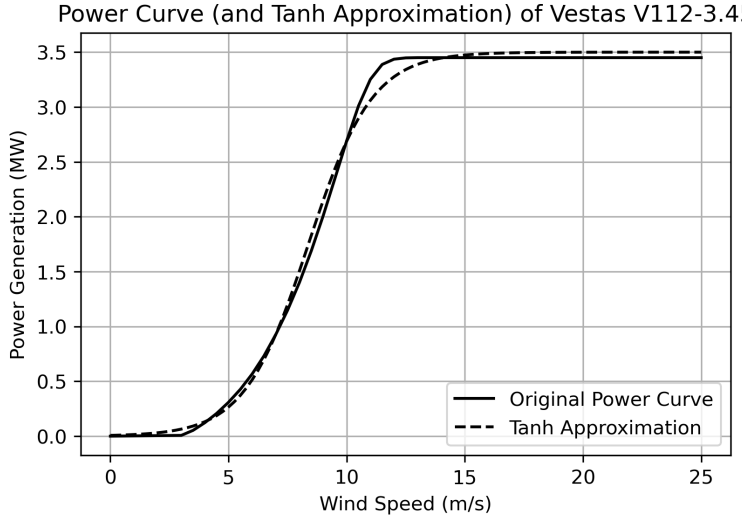

Figure 26: Power curve of Vestas V112-3.45 turbine [win21] and a tanh approximation of it.

Comparing the curves in Figure 26 and Figure 11, it is notable that the cut-out wind speed is not present in the tanh approximation. Although this is not physically accurate, it does not affect the studies, since the maximum wind speed occurring in the datasets is 23 m/s, well below the cut-out speed of 25 m/s. The model simulates 36 turbines of rating 3.45MWp, thereby implying a wind farm rated at 124MWp.

For prediction from the modality of meteorological data, the variables given in Table 12 are always used. Each one is measured at a temporal resolution of 10 minutes.

Table 12: Meteorological features used for predicting GHI

Meteorological Variable	Description	Unit
Time	The moment of the day when the instance was measured	seconds
Air temperature	Measured at 2 meters above surface level	°C
Cloud opacity	The attenuation of incoming sunlight due to clouds. Varies from 0% (no cloud) to 100% (full attenuation of incoming sunlight)	%
Relative humidity	Measured at 2 meters above ground level. Relative humidity is the amount of water vapour as a percentage of the amount needed for saturation at the same temperature. A value of 50% means the air is 50% saturated	%
Wind direction	Measured at 10m altitude	°
Wind speed	Measured at 10m altitude	m/s
Precipitable water	A measure of the precipitable water of the entire atmospheric column	kg/m^2
Surface pressure	Air pressure measured at sea level	hPa

Depending upon the case study, a single or multiple previous instances of these features are included. Additionally, Case Study 9 also includes previous GHI measurements as features to compare the predictive performance of a NN to a persistence forecast. Each of the above variables is normalized using min-max normalization (Equation 5) before being used by a NN such that they are within [0,1].

5.1.2 Training and Testing

The hyperparameters settings that are common to all case studies are listed in Table 13.

Table 13: Training and testing settings for Case Study 1.

Hyperparameter	Value
Training set size	6750
Validation set size	750
Test set size	2500
Learning rate	0.0025
Mini-batch size	64
Maximum training epochs (if not early stopped)	100
Number of train and test trials	10

During training, early stopping is implemented so that if the validation error does not decrease for 5 consecutive epochs, training is halted. The test set used is the same for all trials. Between trials, the separation of the validation and training split is varied, as is the mini-batch shuffling. The plots and tables in each case study that summarize the results show the distribution of test set predictions over the 10 separate trainings. The boxplots show a line at the median of the results, the box extends from the lower to the upper quartile of the data, and the whiskers extend to indicate the range of the results.

5.1.3 Computational Setup

The various NN models described in Chapter 4 are implemented in PyTorch version 1.10.2. For the formulation of the DCOPF with differentiable parameters, CVXPY 1.1.18 and cvxpylayers 0.1.4 were used. Other important packages for data processing are Pandas 1.3.5 and NumPy 1.21.5. The computer used for training and testing the case studies is equipped with an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz with 8 GB of RAM.

5.2 Uni-Modal Baseline Models

These case studies (1-5) are designed to find effective NN model settings to predict PV power, represented by GHI. As such, the loss function that the models are trained on (which is the same for all Case Studies 1-5, except for 5b) is $MSE(GHI_{pred}, GHI_{true})$ (Equation 16). Study 5b investigated wind power prediction, therefore it uses $MSE(P_g^W, P_g^{W,act})$ as a loss function. Here, P_g^W is the predicted wind power, and $P_g^{W,act}$ is the true value, calculated from wind speed using Figure 26. The target variables GHI_{true} and $P_g^{W,act}$ are min-max normalized (Equation 5) such that they are within [0,1].

5.2.1 Case Study 1: Image Resolution

This case investigates the effect of using images of different resolution for predicting GHI using a convolutional neural network with the CNN-base architecture described in Figure 15 and Table 3. The MSE measured on the test set on each trial is presented in Figure 27, and the summary of these trials is given in Table 14.

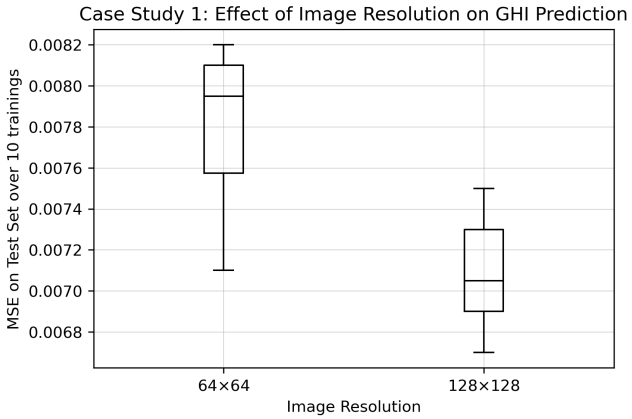

Figure 27: MSE on test set for 10 different trainings of CNN-base to predict GHI from images of either 64×64 or 128×128 resolution.

	64×64	128×128
Average MSE	0.0078	0.0071
STD MSE	0.00040	0.00026

Table 14: Average and Standard Deviation of MSE on predicted GHI from the 10 trials presented in Figure 27.

Using the higher resolution images shows a marginal improvement in predictive performance (9% lower MSE), however, this improvement also comes at the cost of increased computational demands. The memory and time required during training were both about 4× higher when using images of 128×128 resolution as compared to 64×64. Despite the improvement in prediction, to facilitate the rest of the investigation, the 64×64 resolution imagery is used for the remainder of the case studies. Nevertheless, this comparison gives an idea of the sensitivity of this model to image quality, should it be employed in a real setting.

5.2.2 Case Study 2: Time Horizon of Imagery

This study investigates the timing of taking images before predicting PV power. Here, the performance of using three previous images (measured over 30 minutes) and using a single image to predict GHI 10 minutes in the future is compared. As in Case Study 1, the CNN-base architecture is used. The combination of three images was done by calculating the average pixel value at each location of the image. Figure 28 demonstrates this image combination for three images captured consecutively on 01 May 2021. The results of the study are summarized in Figure 29 and Table 15.

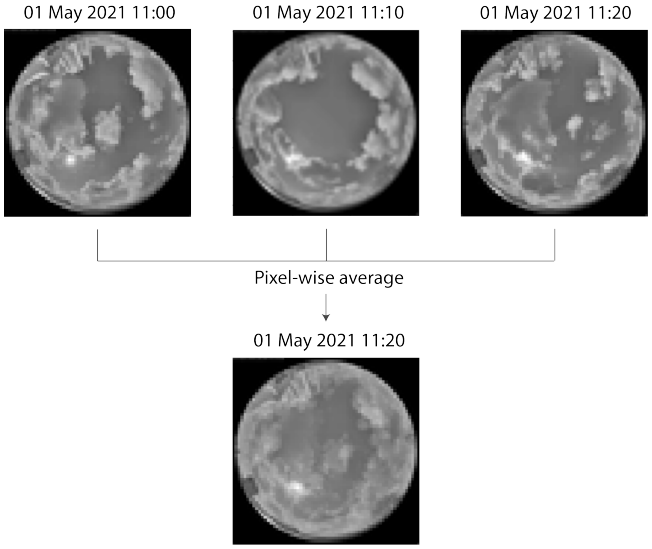

Figure 28: The combination of three images captured on 01 May 2021 using pixel-wise averaging.

Figure 29: MSE on test set for 10 different trainings of CNN-base to predict GHI from either three or one previous images.

	1 Image	3 Images
Average MSE	0.0107	0.00836
STD MSE	0.000675	0.000682

Table 15: Average and Standard Deviation of MSE on predicted GHI from the 10 trials presented in Figure 29.

This method of combining the three images (with pixel-wise averaging) does not change the data volume when comparing the use of three or one previous images (since the input at one instance is still of dimension 64×64×1), however, it does arguably sacrifice information in each of the three images by only considering the mean value of each pixel. Still, the predictions show consistent improvement when using multiple previous images (about 17% lower MSE over the 10 tests), at no additional computational cost since the image dimensionality is identical. The improvement could indicate that losing some feature information in the pixels, in favor of a 'smoothing' effect, is advantageous for making predictions using CNN-base. However, this cannot be conclusively stated without further comparison with other methods of combining the images. For example, an alternative approach could be to stack three images for each instance, such that the input is of dimension 64×64×3; thereby not compromising any pixel information from the separate images (but this would also involve slightly altering the CNN architecture, specifically the input layer hyperparameters). Additionally, investigating the effect of using more than three images could yield an optimal number of images; this would be a form of hyperparameter tuning.

Aspects such as the method used to combine the images, and the choice of the number of images employed are also closely related to the problem setting; in a location with different meteorology or different temporal frequency of the imagery dataset, the ideal hyperparameters could be different from the optimal set for this project.

In summary, this case study has demonstrated that averaging sky images from the recent past (30 minutes) enables the CNN-base architecture to make more accurate predictions than if only the most recent image were used.

In light of this result, for the remainder of the case studies, three superimposed images are used, as opposed to a single image.

5.2.3 Case Study 3: CNN Depth

Case study 3 focuses on investigating the depth of convolutional layers for which we compare a deep neural network (CNN-Alex) with the CNN-base. Since the CNN-Alex architecture has more convolutional and pooling layers, along with more channels within the convolutional layers as compared to CNN-base, there is a large discrepancy in the total number of trainable parameters in either one - CNN-base has about 20,000 while CNN-Alex has over 1,000,000. The architectures are summarized in Tables 3 and 16.

Table 16: CNN-Alex architecture.

Layer	Type	Input Size	Kernel Size	Stride	Padding	Activation	Output Size
C1	Convolutional	64×64×1	7×7	4×4	1×1	ReLU	15×15×96
P2	Max. Pooling	15×15×96	3×3	2×2	1×1	-	8×8×96
C3	Convolutional	8×8×96	5×5	1×1	0×0	ReLU	8×8×128
P4	Max. Pooling	8×8×128	3×3	2×2	1×1	-	4×4×128
C5	Convolutional	4×4×128	3×3	1×1	1×1	ReLU	4×4×256
C6	Convolutional	4×4×256	3×3	1×1	1×1	ReLU	4×4×256
C7	Convolutional	4×4×128	3×3	1×1	1×1	ReLU	4×4×128
P8	Max. Pooling	4×4×128	3×3	2×2	1×1	-	2×2×128
Flatten							
Layer	Type	Input Size	Num. of Neurons	-	-	Activation	Output Size
F9	Fully Connected	512×1	512	-	-	ReLU	512×1
F10	Fully Connected	512×1	64	-	-	ReLU	64×1
F11	Fully Connected	64×1	1	-	-	Sigmoid	1×1

For either architecture, the top layer represents the input, and the bottom one represents the output. Layers C1-P6 are the feature extractive layers of CNN-base and F7-F8 perform regression. Regarding CNN-Alex, C1-P8 perform feature extraction. Here the network constructs much deeper feature maps than CNN-base (the maximum depth is 256, compared to 32 in CNN-base). CNN-Alex therefore also contains far more trainable parameters, as previously noted. Fully connected layers F9-F11 conduct regression in CNN-Alex.

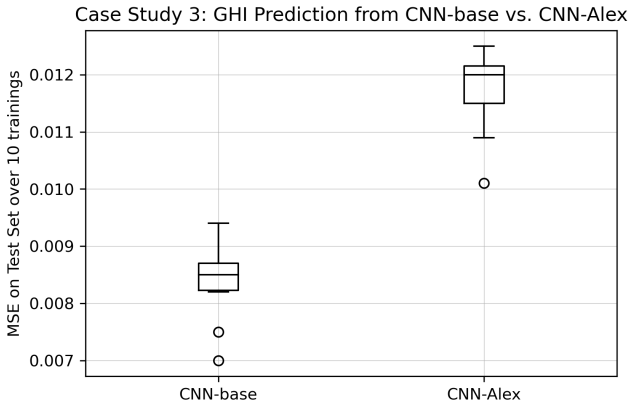

Figure 30: MSE on test set for 10 different trainings of CNN-base and CNN-Alex.

	CNN-base	CNN-Alex
Average MSE	0.00836	0.01172
STD MSE	0.000682	0.000732

Table 17: Average and Standard Deviation of MSE on predicted GHI from the 10 trials presented in Figure 30.

The deeper NN, CNN-Alex, yields consistently worse predictions (about 22% higher MSE as compared to CNN-base). This may be explained by the CNN-Alex having surpassed a critical depth, beyond which performance declines (for the given input data). This phenomenon is investigated by Nichani et al., who find that CNNs that are *too deep* may not generalize well [NRU21]. Provided with larger datasets and higher resolution imagery (which contain more features), the CNN-Alex may be able to outperform CNN-base, but that investigation is out of the scope of this thesis. In response to the results of this case study, the subsequent case studies use the CNN-base architecture (if imagery is involved).

5.2.4 Case Study 4: Time Horizon of Meteorological Data

Case Study 4, much like Case Study 2, investigates whether including recent meteorological data measurements over multiple time steps benefits prediction, as compared to including only the most recent set of measurements. The FNN-base architecture illustrated in Figure 17 and Table 4 is trained to predict GHI, with the hyperparameters as shown in Table 13. When using a single previous instance, eight meteorological features are used (as given in Table 12), therefore, $n_{inputs} = 8$. For the case where three previous instances are used as features, $n_{inputs} = 24$. This changes the input layer of FNN-base; no other alterations to the network are required. The results for making predictions on the test set after 10 different trainings are presented in Figure 31 and Table 18.

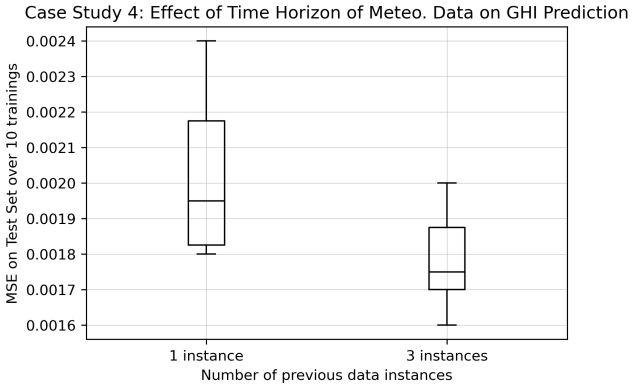

Figure 31: MSE on test set for 10 different trainings of FNN-base to predict GHI from either three or one previous instances.

	1 Instance	3 Instances
Average MSE	0.00201	0.00177
STD MSE	0.000208	0.000134

Table 18: Average and Standard Deviation of MSE on predicted GHI from the 10 trials presented in Figure 31.

Using three previous instances consistently enables the network to make more accurate (by 11% lower MSE) solar forecasts. This is not a surprising result, since the provision of more instances directly means more features (and therefore more relevant information) for the network to perform regression with. It would be an interesting study to also continually include more instances until there is no (or very marginal) corresponding improvement in predictive performance. This time horizon may also differ for the individual meteorological variables (for example the NN may benefit from knowing a longer history of wind speed rather than cloud opacity). Such a study could provide insight into how long certain aspects of the weather can be predicted with a network as simple as FNN-base.

In response to the results of Case Study 4, three previous instances were used for all case studies that follow.

5.2.5 Case Study 5: Forecasting from Imagery vs. Meteorological Data

The prediction of GHI (and subsequently wind power) from two distinct modalities - imagery and meteorological sensor data - is compared in this study. The former modality is used in the CNN-base architecture, while the latter is employed by the FNN-base architecture. The results are shown in Figure 32 and Table 19.

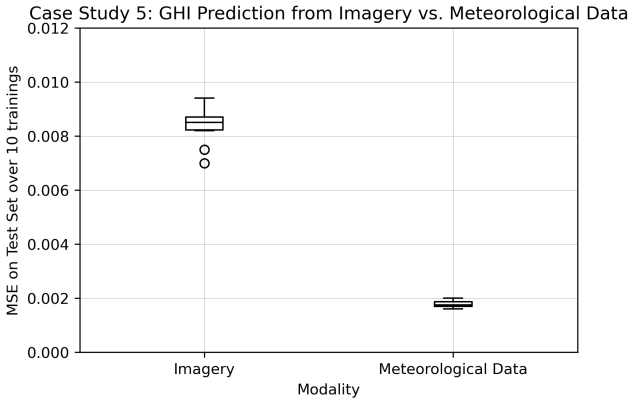

Figure 32: MSE on test set for 10 different trainings of CNN-base (which predicts GHI from images) and FNN-base (which predicts GHI from meteorological data).

	Imagery	Meteorology
Average MSE	0.00836	0.00177
STD MSE	0.000682	0.000134

Table 19: Average and standard deviation of MSE on predicted GHI from the 10 trials presented in Figure 32.

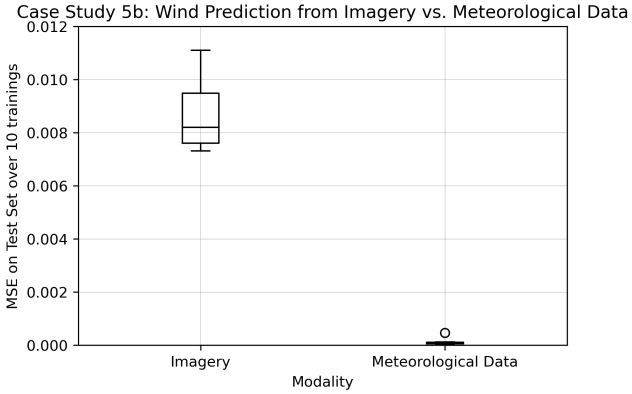

Figure 33: MSE on test set for 10 different trainings of CNN-base (which predicts wind power from images) and FNN-base (which predicts wind power from meteorological data).

	Imagery	Meteorology
Average MSE	0.00867	0.00011
STD MSE	0.00131	0.00013

Table 20: Average and standard deviation of MSE on predicted wind power from the 10 trials presented in Figure 33.

Forecasting from meteorological data clearly outperforms forecasting from images. This can be explained by the fact that images do not contain as many obvious and distinct features, such as wind speed and humidity, as are directly present in the meteorological data. Some of those features (such as wind speed) can possibly be inferred by the NN as it retrieves patterns, but that process cannot be as effective as instrument measurements. This is exacerbated by the fact that the images used in this study are of 64×64 resolution; further limiting the ability of the NN to recognize deeply embedded atmospheric features.

When forecasting wind power, meteorological data is still more effective as a modality than imagery. This result is also influenced by the fact that previous wind speeds (which directly related to wind power via the plot in Figure 26) are included as meteorological features, whereas previous GHI data was not included for predicting GHI.

5.3 Multi-Modal Learning for Power Forecasting

Case studies (6-10) investigate the use of MM learning for power prediction, both for PV and Wind. Studies 6 through 9 use the same loss function as Case Studies 1 through 5, i.e. $MSE(GHI_{pred}, GHI_{true})$ (Equation 16). Case Study 10 focuses on Wind prediction, therefore the loss function that is used to train the model is $MSE(P_g^W, P_g^{W,act})$. Here, P_g^W is the predicted wind power, and $P_g^{W,act}$ is the true value, calculated from wind speed using Figure 26. $P_g^{W,act}$ is normalized using min-max normalization (Equation 5) such that it is within [0,1].

5.3.1 Case Study 6: Meteorological Feature Extraction

The MM2 and MM4 architectures defined in Chapter 4 provide two alternatives for the feature extraction process of a MM learning NN. The former includes a network for extracting features from both modalities separately (MM2), before combining them, whereas the latter only uses a network to extract features from imagery (MM4), and then fuses them with the meteorological data. The method used for fusion in both architectures for this study is concatenation, given by Equation 20. Figure 34 and Table 21 present the results of this study.

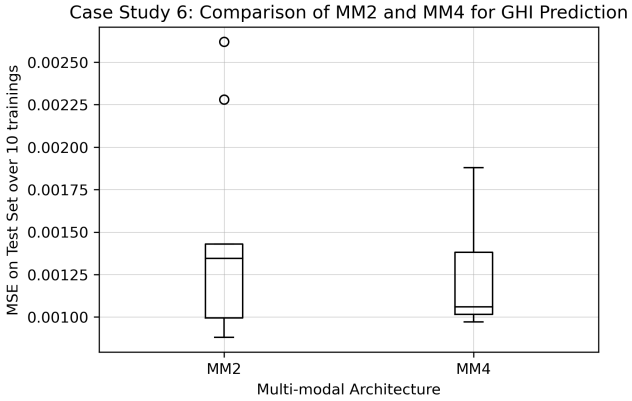

Figure 34: MSE on test set for predicting GHI over 10 different trainings of MM2 and MM4.

	MM2	MM4
Average MSE	0.00143	0.00129
STD MSE	0.00058	0.00031

Table 21: Average and standard deviation of MSE on predicted GHI from the 10 trials presented in Figure 35.

Despite the added complexity of the MM2 architecture (more layers and therefore trainable parameters), there is no improvement in predictive performance. MM4 produces forecasts with 10% lower MSE and a lower variance. Both of these results may be attributed to the fact that there are few meteorological features to begin with (n_{inputs} = 24), thus the added FNN-fe network that is included in MM2 only overfits the training data, without revealing new information from the features. This effect may be alleviated with much larger sets of training data, or the use of more previous instances of meteorological features (also mentioned as an extension to Case Study 4); both would require extensive investigation that was not conducted during this thesis [Ipp19]. For the subsequent studies, in light of these results, the MM4 architecture has been selected.

5.3.2 Case Study 7: Joint Representation Method

This study investigates bilinear pooling and concatenation for feature fusion within the MM4 architecture. This fusion occurs, as shown in Figure 18, after feature extraction from the imagery modality. The results are shown in Figure 35 and summarized in Table 22.

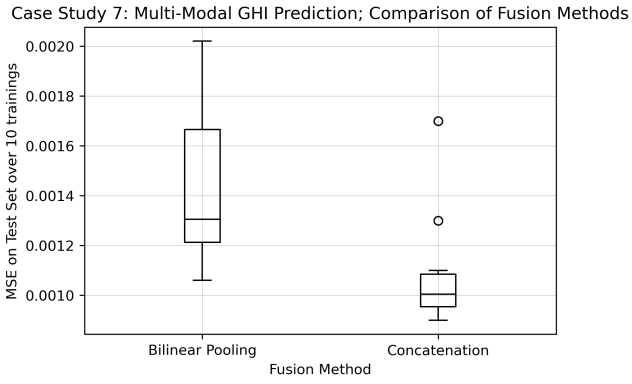

Figure 35: MSE on test set for 10 different trainings of MM4, using either bilinear pooling or concatenation for joint representation of the features.

	Bil. Pooling	Concatenation
Average MSE	0.00144	0.00109
STD MSE	0.00033	0.00024

Table 22: Average and standard deviation of MSE on predicted GHI from the 10 trials presented in Figure 35.

The NN performs considerably more accurately (about 24% lower MSE) and more consistently (lower variance of error) over these 10 trials using concatenation. Bilinear pooling includes, by definition, all the features represented

in the fused feature space of concatenation. However, it results in a quadratically larger feature space with greater sparsity and more irrelevant features than implied by concatenation. This may partially explain the poorer performance of using bilinear pooling - the NN could be missing important information amongst the vast feature space. Selecting the right features for a regression problem has been shown to not only improve computational times and efficiency but also predictive performance [Ole22]. Feature selection improves performance as it enables the NN to avoid making spurious correlations between irrelevant features and the target (a form of overfitting) during training.

5.3.3 Case Study 8: Comparison of Methods for PV Forecasting

This case study is central to quantifying the utility of MM learning for power forecasting. A naive UM baseline (UM-base1, which predicts GHI as an average of the prediction from the imagery and sensor data modality), a more intelligent UM baseline (UM-base2, which uses a linear regression-based weighted average of the two predictions), the proposed MM learning model MM4, and a persistence forecast are compared in terms of prediction error. Regarding the MM4 architecture, the features from imagery are fused with the meteorological ones via concatenation. The results are presented below.

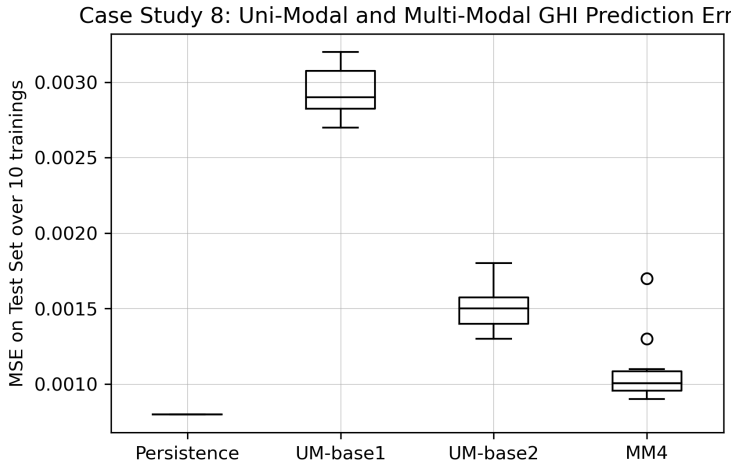

Figure 36: MSE on test set for 10 different trainings of a persistence model, UM-base1, UM-base2, and MM4.

Table 23: Average and standard deviation of MSE and %RMSE on predicted GHI over 10 trials presented in Figure 36.

	Persistence	UM-base1	UM-base2	MM4
Average MSE	0.0008	0.00294	0.00151	0.00109
STD (MSE)	0	0.000157	0.000179	0.000242
Average %RMSE	2.8%	5.4%	3.9%	3.2%
STD (%RMSE)	0	0.14%	0.23%	0.34%

The 10 different trained models clearly show that MM learning improves predictions. Given the same input data, the NN can combine information from either modality in a way that neither a naive ensemble nor a regression can achieve. Possible explanations for why this behavior occurs, considering that the same data are made available to all three NNs, are that MM4 has far more trainable parameters for combining the modalities, and that the intermediate stage feature combination enables the imagery to enhance the sensor data. These theories are discussed in greater depth in Chapter 6. The prediction performance of UM-base1, UM-base2 and MM4, in terms of %RMSE, is in line with state-of-the-art NN-based methods for predicting solar power [RNE16]. The persistence forecast outperforms the other 3 methods consistently, however, it must be noted that this is because it is predicting GHI using the GHI from 10 minutes prior, assuming it is known. With the NN methods, the GHI from 10 minutes before is not used as a feature, to test their applicability in environments where the GHI data is not continuously available.

5.3.4 Case Study 9: Forecasting using GHI Features

Case Study 8 has compared the performance of neural networks predicting GHI from UM and MM data, where GHI is not included as a predictive variable, against a persistence forecast of GHI. It is worth investigating how the NN-based

algorithms perform if GHI data is also made available to them. Case Study 9 investigates this, looking at the same persistence forecast as before and UM-base2 with a previous GHI as a feature (in addition to those in Table 12). Three previous instances are used again for the NN, therefore $n_{inputs} = 27$. Figure 37 and Table 24 summarize the results.

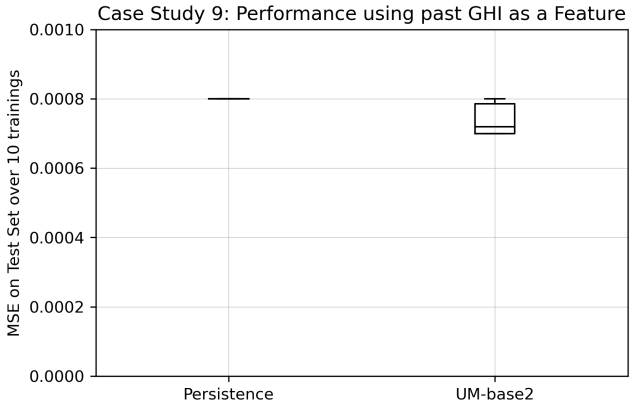

Figure 37: MSE on test set for 10 different trainings of a persistence model and UM-base2 (with GHI as a feature).

	Persistence	UM-base2
Average	0.0008	0.00073
STD	0	0.000046

Table 24: Average and standard deviation of MSE on predicted GHI from the 10 trials presented in Figure 37.

Using the NN-based UM algorithm UM-base2 improves upon the persistence forecast by a 9% reduction in MSE, if GHI is included as a feature. This result demonstrates that whether GHI sensor data is available or not, a NN can predict better than the simplest persistence approach. Although the multi-modal models were not included in this case study, MM4 would also presumably improve upon persistence considering the relative performance of UM-base2 and MM4 in Case Study 8. Thus, in response to Research Question 3c, it can be concluded that UM and MM methods can achieve better forecasts than a persistence model.

5.3.5 Case Study 10: Comparison of Methods for Wind Forecasting

This case study is conducted in a similar manner to Case Study 8, as several forecast models are compared in terms of their error in predicting wind power. Due to the relatively weak performance of UM-base1 in Case Study 8, it is disregarded in this comparison. Instead, an alternative MM learning neural network model is included, MM-Shallow, which contains only one neuron for forecasting.

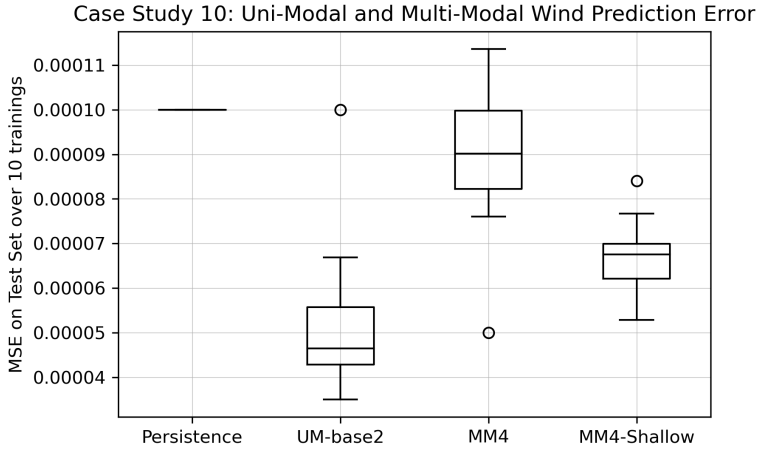

Figure 38: MSE on test set for 10 different trainings of a persistence model, UM-base2, MM4, and MM4-Shallow.

Table 25: Average and standard deviation of MSE on predicted Wind Power over 10 trials presented in Figure 36.

	Persistence	UM-base2	MM4	MM4-Shallow
Average	0.0001	0.000053	0.000090	0.000060
STD	0	0.000019	0.000018	0.000009
Average %RMSE	1%	0.7%	0.9%	0.8%
STD (%RMSE)	0	0.11%	0.10%	0.05%

It is visible that MM learning does not benefit wind power forecasting. This behavior may be explained by the fact that, during training, the features are amalgamated through layers and, due to vanishing gradients, only the last couple of layers are effectively updated. As a result, the less useful imagery features cannot be discarded as easily as when using UM-base2. This hypothesis is also supported by results from the single neuron MM model (MM4-Shallow), which performs as well or better than MM4. Therefore, perhaps there is a threshold beyond which, if one modality is more useful than the other, MM learning will be detrimental to overall performance.

It is also possible that no strong conclusion can be made from this study since the level of prediction error is very low compared to other studies. Thus, the differences between the various models may not be significant enough to be explained by methodological aspects. Again, each of the NN-based models (UM-base2, MM4, and MM4-Shallow) perform similar to recent studies applying NNs for short-term wind forecasting in terms of RMSE [SZMM10]. Due to differences in input data, geographical location, model complexity, and time horizons, it is difficult to make exact benchmarks.

5.4 Economically Optimized Predictions with End-to-End Learning

The following case studies focus on system cost minimization. Enabling cheap and stable distribution is central to the design of any energy system forecasting methodology. Implementing E2E learning with the system cost as the task objective can be expected to reduce costs compared to methods which are trained to predict power and subsequently solve the cost minimization. Case Studies 11 through 13 investigate four distinct algorithms for predicting system cost C_{sys} from MM data.

1. **UM-base1-Seq**: this model consists of the UM UM-base1 model to predict power, and the predicted value is used to solve for system cost C_{sys} in a 6-bus system.
2. **UM-base2-Seq**: this model consists of the UM UM-base2 model to predict power, and the predicted value is used to solve for C_{sys} in a 6-bus system.
3. **MM4-Seq**: here the multi-modal MM4 model is trained to predict power, and subsequently this prediction is used to solve for C_{sys} in a 6-bus system. When predicting for a system with PV and Wind, as in Case Study 12, the MM4-PVWind-Seq is used.
4. **MM4-E2E**: the model is now trained directly on C_{sys}, as shown in Figures 22 and 23. When predicting for a system with PV and Wind, as in Case Study 12, the MM4-PVWind-E2E is used.

Case Study 11 only considers PV generation in the power system. As such, UM-base1-Seq, UM-base1-Seq and MM4-Seq are trained on $MAE(P_g^{PV}, P_g^{PV,act})$. MAE is used as opposed to MSE because of the linear nature of the system cost function (explained by the constant values for $c^g_{1,2,3}$ and $\gamma_{1,\dots,8}$ in Table 11 and the plot in Figure 21). A similar procedure is followed for Case Study 12, where Wind generation is also included. Here a pair of UM-base1 models (trained on $MAE(P_g^{PV}, P_g^{PV,act})$ and $MAE(P_g^{W}, P_g^{W,act})$) are used by UM-base1-Seq to predict PV and Wind, a pair of UM-base2 models is used by UM-base2-Seq, and the MM4-PVWind model (Figure 24) is used by MM4-PVWind-Seq. MM4-PVWind-E2E, as before, is trained directly on C_{sys}.

The results from the algorithms listed above are compared with a hypothetical **Perfect Forecast** scenario, where the system cost is calculated using the known PV and Wind power as the prediction (therefore $MAE(P_g^{PV}, P_g^{PV,act}) = MAE(P_g^{W}, P_g^{W,act}) = 0$). This simulation, therefore, calculates the minimum system cost against which the various algorithms may be assessed. We expect that E2E learning, with the added information about the system cost function, would enable the prediction of lower costs by the NN. From Figure 21, it is clear that a perfect forecast would enable the lowest cost. Amongst the predictive models, MM4-E2E (and MM4-PVWind-E2E) may be expected to sacrifice some power prediction accuracy to achieve more consistently reduced costs.

5.4.1 Case Study 11: Minimized System Cost; only PV Generation

This study compares the system costs minimization by E2E learning against the baselines for predicting PV power which are established already. The results for cost minimization are given in Figure 39 and Table 26.

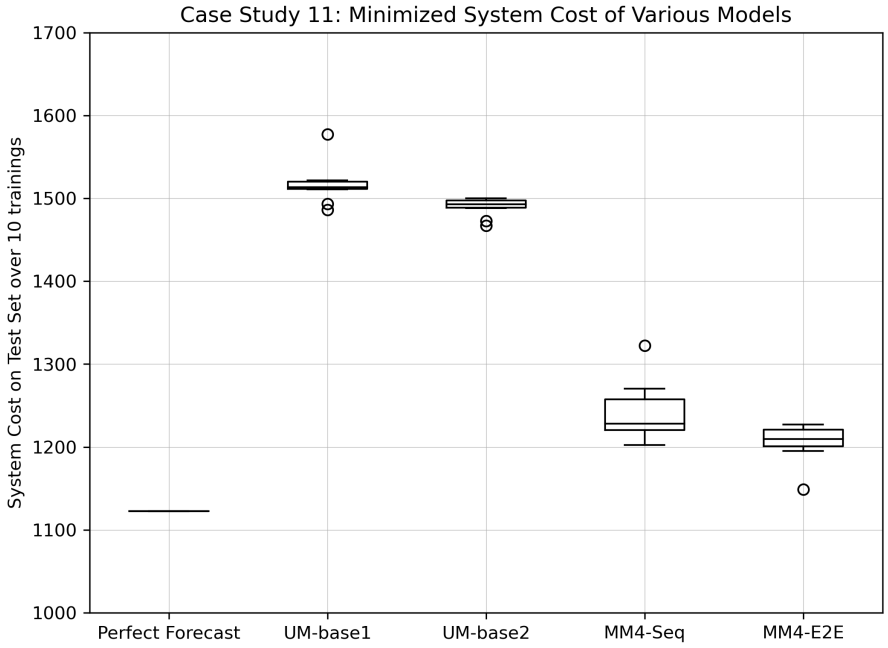

Figure 39: System cost on test set for 10 different trainings of four different algorithms.

Table 26: Average and standard deviation of system cost C_{sys} from the 10 trials presented in Figure 39.

	Perfect Forecast	UM-base1-Seq	UM-base2-Seq	MM4-Seq	MM4-E2E
Average C_{sys} [Euro]	1122.4	1516.7	1489.9	1240.3	1205.6
STD of C_{sys} [Euro]	0	24.3	11.4	36.3	22.6

Here multi-modal learning is shown to not only improve power output prediction but also benefit power system planning in terms of system cost as both the MM models outperform the UM ones. E2E training (MM4-E2E) seems to yield marginally improved results in terms of system cost (about 1.5% lower) when compared to sequential training (MM4-Seq). However, comparing the results of MM-Seq and MM-E2E with the Perfect Forecast indicates that E2E training makes a significant improvement. C_{sys} is increased by 10.5% when training sequentially, and increased by only 7.4% when training E2E. This 'excess cost' is plotted against power prediction accuracy (measured by $MAE(GHI_{pred}, GHI_{true})$) in Figure 40 for the various models in Table 26.

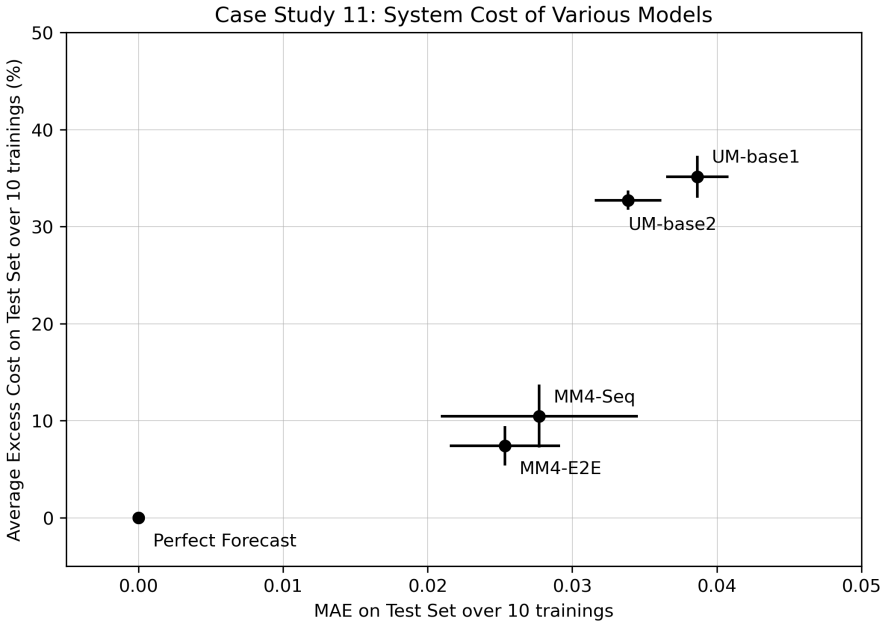

Figure 40: Excess system cost compared to power forecast error.

A conventional approach, represented by UMbase2-Seq, where PV power is predicted from two modalities separately and subsequently combined with a linear regression to solve for system costs results in a 32% cost increase compared to a perfect knowledge scenario. The proposed model, MM4-E2E, which uses MM and E2E learning only increases the cost by 7%, which is a significant performance improvement. Much of this cost reduction, it can be argued from the difference between UMbase2-Seq and MM4-Seq in Figure 40, is due to the intelligent combination of MM features using a NN. E2E learning also contributes a clear reduction in cost (comparing the 10% excess cost from MM4-Seq to 7% from MM4-E2E). In addition, E2E training enables a more consistently reduced cost, shown by the lower variance in cost. This improved performance of MM4-E2E can be explained by the fact that E2E training provides the MM4 NN with information about the generation cost functions (c_b^g) of the various thermal generators, as well as the infeasibility costs ($\gamma_1 ... \gamma_8$). The test in Case Study 11 can also be conducted in a power system with multiple renewable generation sources that need prediction.

5.4.2 Case Study 12: Minimized System Cost; PV and Wind Generation

Case Study 12 investigates the cost minimization in a system with PV and Wind. The results of training and testing the models previously described 10 separate times are depicted below.

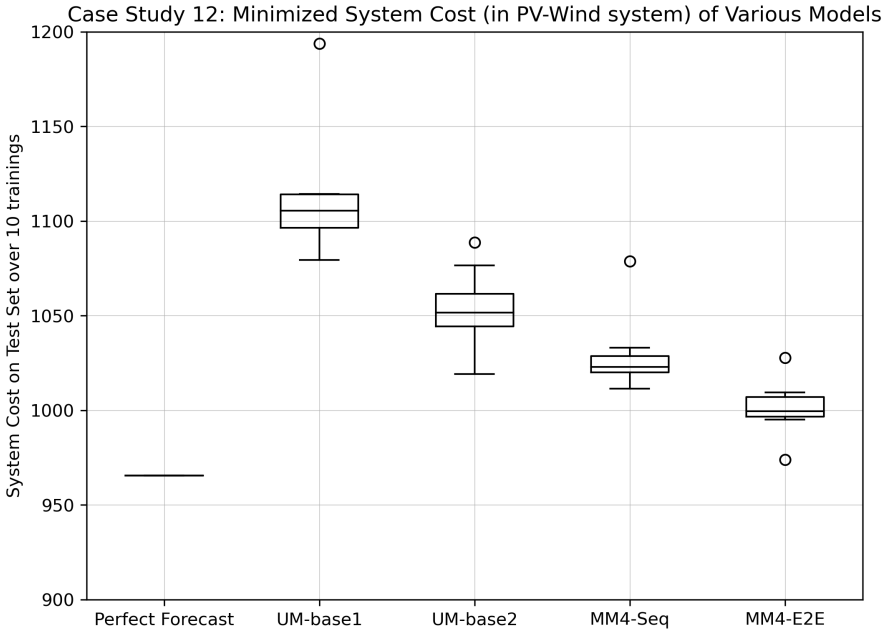

Figure 41: System cost on test set for 10 different trainings of four different algorithms.

Table 27: Average and standard deviation of system cost C_{sys} from the 10 trials presented in Figure 41.

	Perfect Forecast	UM-base1-Seq	UM-base2-Seq	MM4-PVWind-Seq	MM4-PVWind-E2E
Average C_{sys} [Euro]	965.5	1112.9	1053.5	1029.3	1001.5
STD of C_{sys} [Euro]	0	34.7	21.7	20.8	15.1

The minimized system cost is plotted against prediction error in Figure 42. In this plot, the error is measured as the average of $MAE(P_g^{PV}, P_g^{PV,act})$ and $MAE(P_g^{W}, P_g^{W,act})$.

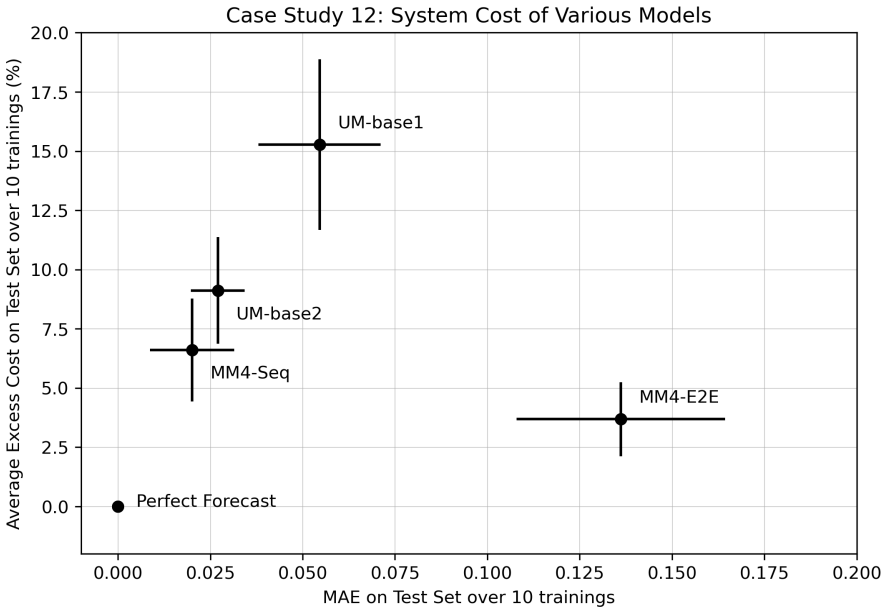

Figure 42: Excess system cost compared to power forecast error.

While E2E (represented by MM4-E2E) training results in lower system costs than sequential learning (represented by MM4-Seq), it is notable that the power prediction error is significantly greater in the former. This result is in contrast to the results of Case Study 11, where MM4-E2E also reduces prediction error in comparison to MM4-Seq.

The magnitude of prediction error (between 10 and 15% MAE on PV and Wind power prediction) is also starkly higher than the errors seen in any model from other case studies, which is a surprising result.

Another interesting feature from the results is that the standard deviation on system cost is lowest from the MM4-E2E predictor, however, the standard deviation on the power prediction is the largest for MM4-E2E. The first observation is logical because E2E learning trains the NN to predict minimized system cost, therefore the model should consistently find costs. The second observation indicates that perhaps the NN often finds the minimized cost by overestimating PV and underestimating Wind, or vice versa, ultimately 'balancing' the error on both such that the power grid is stable without redispatch. This hypothesis could therefore explain the relatively high power prediction error, and it is further investigated in Case Study 13.

5.4.3 Case Study 13: System Cost Functions

Rather than comparing the performance of different model configurations, as the majority of the above case studies have done, Case Study 13 seeks to explain why E2E learning outperforms conventional learning in terms of minimizing system cost. In particular, the results found in Figures 40 and 42 are worth investigating as they both show MM4-E2E achieving the lowest costs but at varying levels of power prediction accuracy. MM4-E2E achieves low prediction error compared to other models in a system with one renewable generation source (Figure 40), but much higher error when there are two renewable generators (Figure 42). To understand why this may be the case, the system cost functions under either power system setting (single and dual generators) are plotted along with the predicted minimized cost, against prediction accuracy. The following levels of load and generation in System 1 (single generator) and System 2 (dual generators) are applied to produce the theoretical system cost graphs.

System 1 (single PV generator):

- Load (P_d) = 145 MW
- True PV generation ($P_g^{PV,act}$) = 25 MW.
- Test samples with P_d in the range [100, 190] MW are plotted to have sufficient instances.

System 2 (PV and Wind generators):

- Load (P_d) = 145 MW
- True PV generation ($P_g^{PV,act}$) = 25 MW.
- True Wind generation ($P_g^{W,act}$) = 25 MW.
- Test samples with P_d in the range [100, 190] MW are plotted to have sufficient instances.

Figure 43 illustrates the theoretical and predicted costs for System 1.

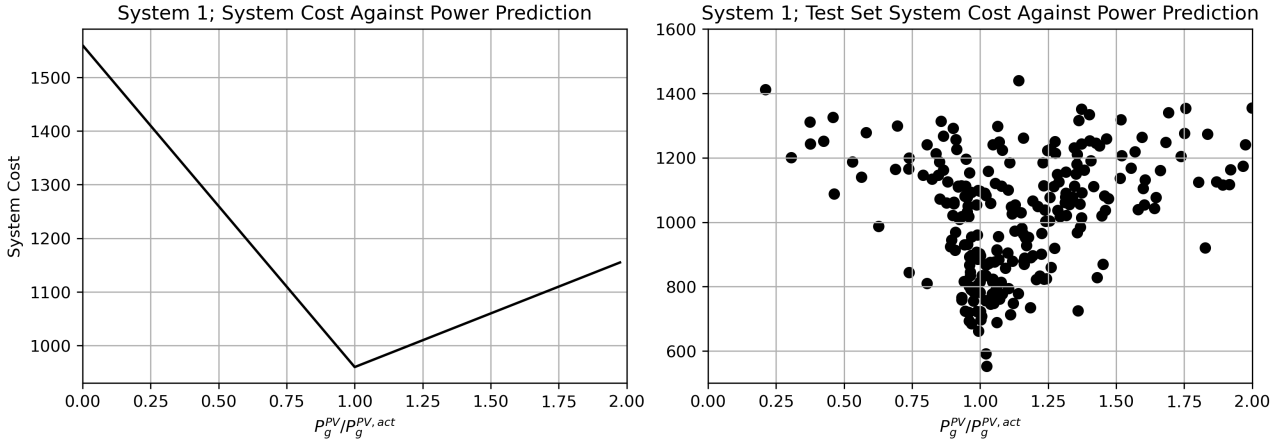

Figure 43: (Left) Theoretical variation of system cost against PV power prediction for a system with $P_d = 145$MW and $P_g^{PV,act} = 25$MW (repeated from Figure 21) and (Right) Minimized system cost by the MM4-E2E network.

The distribution of minimized costs from the test set follows a somewhat similar pattern to the theoretical system cost, showing that the NN mimics the expected cost function after being trained. The magnitude of costs on the two plots is different because the results on the right are for instances with P_d in [100, 190] MW, and any level of PV generation, however, the shape is still well replicated. The minimal solution is clearly achieved by perfectly forecasting power, but when this is not possible, it is valuable to know the cost of over- or underestimating. Herein lies the utility of E2E learning - considering the cost shown in Figure 43, E2E learning enables the information about the relatively expensive underestimation to be used to train the NN. Sequential learning lacks this facility as common loss functions such as MAE and MSE are symmetric.

The same comparison, between theory and practice, can be made for System 2. Figure 44 illustrates the theoretical and predicted costs for the system with PV and Wind generation.

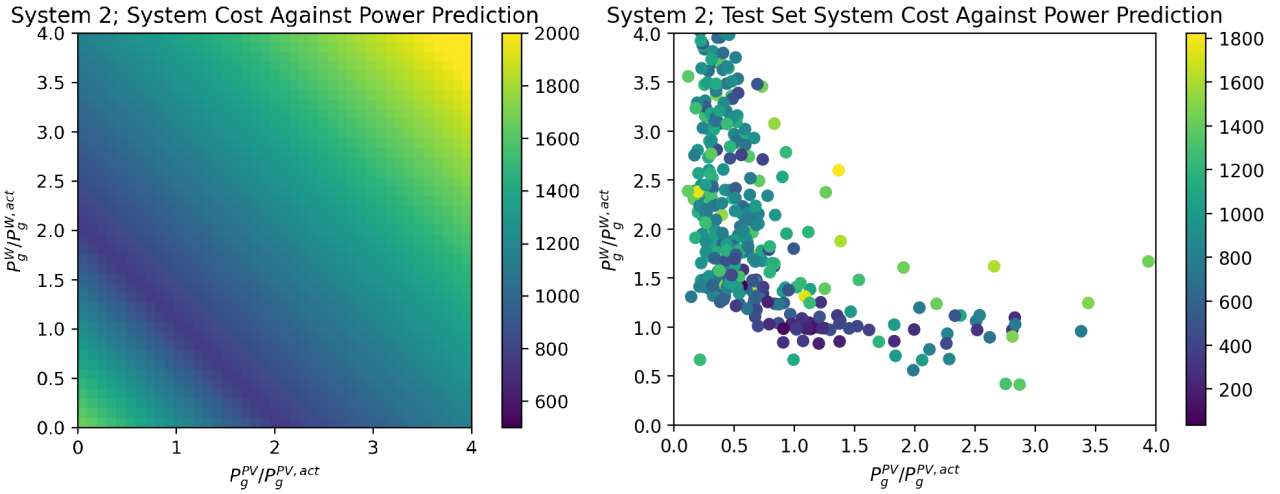

Figure 44: (Left) The expected variation of system cost based on PV and Wind power forecast accuracy. This plot is for a system with $P_d = 145$MW and $P_g^{PV,act} = P_g^{W,act} = 25$MW. (Right) Minimized system costs predicted by MM4-E2E on test set against PV and Wind power forecast accuracy.

Focusing first on the left plot, this three-dimensional system cost does not have a clearly defined minimum as there is in Figure 43. Instead, it seems possible to achieve similar low costs (along the blue valley) by overestimating PV and underestimating Wind, or vice-versa. In such a scenario (where one source is overestimated and the other is underestimated), it is possible to incur errors such that the power system is balanced and therefore no (or very low) redispatch costs are incurred by the forecast errors. This behavior explains the results in Figure 42, where relatively large prediction errors are not seriously detrimental to system cost minimization.

The linear valley of optimal solutions is somewhat visible on the right graph of Figure 44. There is a discrepancy in the magnitude of costs (lower system costs are present in the latter plot) and in the location of minimized solutions along the axes. This difference can be explained by the fact that the plot on the left of Figure 44 is for a specific test setting ($P_d = 145$MW and $P_g^{PV,act} = P_g^{W,act} = 25$MW), whereas the one on the right includes instances with any level of $P_g^{PV,act}$ or $P_g^{W,act}$, and P_d in [100, 190] MW. Instances from this range of power combinations are included because there are not enough test set instances with values close to $P_d = 145$MW.

6 Discussion

In this chapter, the results from the various case studies are analyzed and commented upon in the context of the research questions written in Chapter 2. The analysis of the results and answers to our research questions is followed by a set of recommendations for further work.

6.1 Answering the Research Questions

How can uni-modal data be effectively applied to predicting PV power?

Establishing effective models to predict power from UM data was an important step in the project to, firstly, understand the suitable NN methods for the given modalities, and secondly, to have appropriate baselines against which the proposed models can be assessed. Two modalities of data, sky imagery and meteorological sensor data, were used separately to predict power, and several insights were drawn from the case studies that investigated their use in NNs.

Sky Imagery: The sky imagery modality was used by a CNN to predict power. It was found, through Case Study 1, that higher resolution imagery, as can be expected, enabled more accurate forecasts. This is explained first by the higher number of features available to the algorithm, and second by the greater potential for the NN to retrieve higher-level features (such as objects, patterns) within the image. A deep CNN is adept at detecting higher-level features and is therefore used for making predictions from imagery. Although images with greater resolution enable better forecasting, their use comes at the cost of computational time and requirements. Depending on the practical setting of using such algorithms, these restrictions vary, therefore the resolution used can be fitted to the practical context. It has also been found that when using a CNN, combining multiple recent images (captured over 30 minutes) into a single one via pixel-wise averaging improves forecasts. This implies that, although some level of detail is lost in the combined image from each of the individual ones, a form of temporal smoothing is beneficial, which can indicate that very short term (10 minutes) fluctuations in the condition of the sky are not as valuable as slightly longer term (30 minutes) trends. Between 10 minute intervals, the position of the sun is virtually unchanged (and its position would not have a great effect on the function of a CNN, since it learns to detect objects regardless of where they are in the image), but cloud conditions can vary greatly. The temporal variation of cloud cover is visible in Figure 28, where 3 sky images captured at a 10 minute interval are shown. The solar position in the same images is virtually stationary. The results of Case Study 2 show that providing the CNN with the mean sky conditions of the recent past mitigates the inaccurate predictions that may be caused by very short term aberrations in cloud cover.

Meteorological Sensor Data: For this modality, a FNN is used as opposed to a CNN. The FNN is a simpler NN architecture which is commonly used for regression problems based on numerical data [WLB+14]. Similar to the conclusion about the effectiveness of using multiple images for forecasting, it was found in Case Study 4 that the use of multiple recent meteorological data (from the previous 30 minutes) produced more accurate forecasts than using only the most recent (from 10 minutes prior). However, it is unclear how much of the improvement can be attributed to the NN achieving a better understanding of the trends in meteorology, and how much of it is due to simply having more features to forecast from. Regarding both Case Studies 2 and 4, the optimal number of recent samples has not been found (nor investigated, due to time constraints). Three previous ones were used in both (in comparison to a single one), but more than three may yield still better results.

The results of Case Study 5 demonstrate that more accurate predictions of PV power can be made from the modality of meteorological data than from sky imagery. As mentioned, this may be because the latter modality does not have important features as clearly available to the NN, while the former consists of explicit measurements of those features. In summary, it has been found that a FNN is suitable for predicting PV power from numerical sensor data. Using a CNN is more suitable for the sky imagery modality, and averaging multiple recent images produces better forecasts than only using the most recent one.

How can the multiple modalities be effectively combined to predict wind or PV power, and which data would enable this combination?

The proposed MM learning model utilized intermediate fusion of features from either modality before making a power prediction. Feature extraction from the imagery is necessary, as mentioned in Chapter 4 because pixel values cannot be directly combined with sensor data [LLXN18]. Case Study 6 investigated the effectiveness of a NN for feature extraction from the sensor data. The outcomes of this case study assert that using a FNN (in this case FNN-base) for feature extraction from the meteorological data was not beneficial to forecasting performance. This may be due,

as mentioned, to the overfitting of a training set with few (24) features [Ipp19]. Another explanation could be related to the vanishing gradients problem, which disproportionately affects early layers of an NN [BJZP20]. Examining the mean (over the parameters) of the backpropagated error gradients of each layer within the MM2 network during training shows that this is a relevant phenomenon. Figure 45 captures these means for the layers of FNN-fe and FNN-pred of the MM2 architecture.

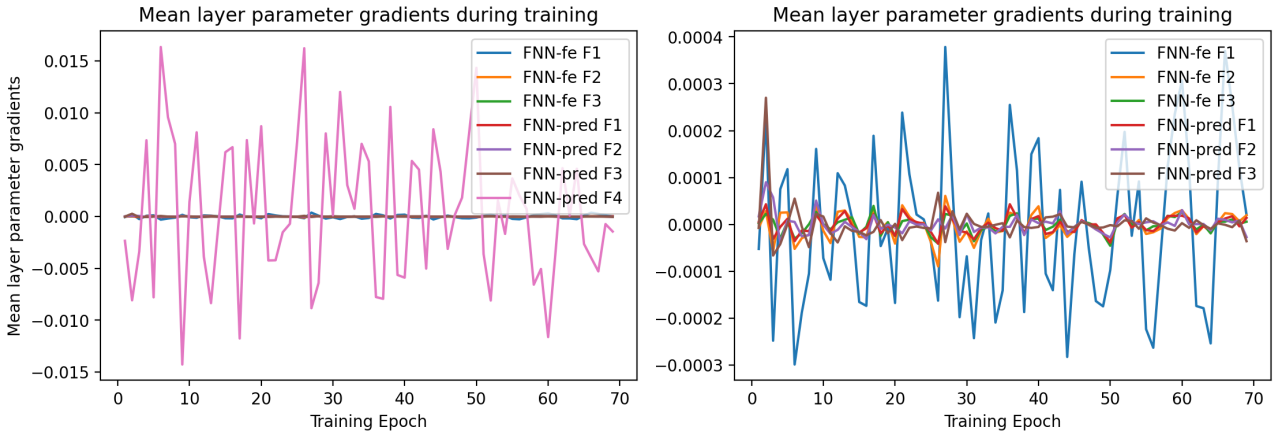

Figure 45: Mean error gradients over all parameters within each FC layer of MM2 during training (left), and the same plot with all except the output layer FNN-pred F4 (right).

The left plot of Figure 45 shows the vast difference in gradients over the layers. The output layer FNN-pred F4 has, by far, the largest magnitude error gradients during training. The diminished magnitudes for the layers before the output demonstrate the vanishing gradients problem. The plot on the right shows that, although they are relatively small in magnitude, the gradients of the other layers do vary during training. This behavior shows that the design of the NN architecture may need to be varied to mitigate the vanishing gradients (perhaps using batch normalization, other activation functions, or fewer hidden layers) and fully utilize the sensor data modality. In addition, the architecture could be altered by using features from different stages (higher and lower) of the network in the MM joint representation. This could enable the backpropagation algorithm to more effectively update parameter weights, as error gradients for the various layers could be expected to have larger magnitudes. From Case Study 6 it has been found that feature extraction is not useful for the given set of meteorological data, because there are relatively few features to begin with, and due to the vanishing gradients problem. Therefore, combining the data directly with the imagery features is found to be a more appropriate form of MM fusion.

After having shown which features are most useful to fuse, another open question is how they should be fused. The results of Case Study 7 found that concatenating the features produced better predictions than using bilinear pooling. This is explained by the large and mostly impertinent set of features produced by bilinear pooling (f_{MM}^{BP}). While f_{MM}^{BP} contains all the features that the concatenated feature space (f_{MM}^{CC}) does, it also contains several others which are less relevant (for instance the cloud cover % multiplied with a high-level imagery feature). This irrelevance causes the model to learn spurious correlations during training and therefore ultimately has a detrimental effect on predictive performance [Ole22]. The results of Case Study 7 established concatenation as the preferred method to combine features from separate modalities.

To summarize, the results of Case Studies 6 and 7 addressed research question 2 by investigating different feature extraction and feature fusion techniques. We found that feature extraction from the imagery modality exclusively, along with concatenation for fusing visual and sensor features, produced the most effective MM architecture (MM4) for forecasting PV power.

How does multi-modal learning outperform trivial uni-modal learning baselines?

As shown in the results of Case Study 8, there is potential to enhance prediction accuracy by intelligently combining the modalities. The most naive way, using a simple average of the prediction from meteorology and imagery (UM-base1) severely lacks in performance compared to using a linear regression of the two predictors (UM-base2, which yields a 50% reduction in MSE). UM-base1 also performs worse than using UM learning with the meteorological data, proven by comparing the errors in Figure 31 and 36. This is due to the significantly worse performing imagery modality (also shown in Figure 31) being equally weighed with the sensor data in UM-base1.

Employing a NN to combine the features from either modality *before* making the PV power forecast, i.e. using multi-modal learning via the MM4 architecture, improved on the performance of UM-base1 by 40% and upon UM-base2 by 18% in terms of RMSE. The linear regression algorithm that UM-base2 is reliant upon to make combined predictions has only two trainable parameters (as given in Equation 19), while the NN-based method MM4 has thousands. This added complexity, given enough training data and features, partially explains why MM learning produces better performance. A second factor that explains the performance difference is the stage at which the modalities are being fused in the two algorithms: UM-base2 employs the latest possible fusion, only combining the predicted GHI from either modality, while MM4 employs intermediate fusion, concatenating features from meteorological data and imagery and subsequently using this joint representation as input to FC layers. Importantly, the MM4 architecture includes a CNN (CNN-fe) for feature extraction from the imagery, the outputs of which are fused with the sensor data modality. During training, this architecture enables CNN-fe to learn not only which visual features are most important for accurately predicting GHI (therefore attributing the respective parameters with higher weights) but also which ones are most beneficial to be considered alongside the sensor data for forecasting. While CNN-base is informed on how to best forecast GHI, CNN-fe within the MM4 architecture learns how to best enhance the meteorological data in order to forecast GHI.
It is difficult to ascertain the extent to which each of these two factors (increased complexity and intermediate feature fusion) benefits the performance. For instance, a study could be designed where the predicted outputs from each modality in UM-base2 (GHI_{UM1} and GHI_{UM2}) are fused and used as inputs to a NN; comparing the results to those of MM4 would isolate the effect of intermediate fusion. However, such a study would likely yield trivial results because of the minimal features in the joint representation (if using concatenation for feature fusion, $[GHI_{UM1} \frown GHI_{UM2}]$ has dimension (2×1)) made available to the NN.

When forecasting wind power instead of GHI, MM learning does not make a significant improvement. This may be because the less useful features from the imagery are amalgamated with the more useful sensor data through the layers and, due to vanishing gradients, effectively are detrimental to performance. In this context, the implementation of feature selection would be a possible improvement. This could be done, for instance, by calculating the correlation between each feature and the prediction target, and only including features which achieve a certain threshold of correlation.

In summary, MM learning improved upon UM baselines by providing a larger number of trainable parameters, and enabling one modality to enhance the other in the context of predicting PV power. For prediction wind power, MM learning is not significantly beneficial. This is likely due to the greater relative underperformance of one modality (imagery) compared to the other (meteorological sensor data).

How does pairing multi-modal learning and E2E learning advance optimized cost prediction?

Aside from the prediction of renewable generation, the use of MM data for reducing energy system costs is of interest in this thesis. Case Study 11 finds that MM learning can drastically reduce C_{sys}, as MM4-Seq lowers cost by 16% compared to UM-base2-Seq. Incorporating E2E learning with MM learning (through the MM4-E2E architecture) modestly reduced this by a further 1.5%. However, as mentioned, the bulk of C_{sys} is C_{sch}, which could not be reduced even with perfect renewables forecasting. The MM4-E2E architecture yields system costs 7% higher than perfect forecasting, whereas MM4-Seq is about 10% higher, demonstrating the utility of E2E learning in reducing redispatch cost. Simultaneously, in Case Study 11, we find that E2E learning makes more accurate power predictions. This paired behavior is logical given the unique minimum present in the system cost function illustrated in Figure 21.

On the other hand, Case Study 12 finds that when minimizing costs in a system with PV and Wind generation (both unknowns to be predicted by the NN), E2E reduces cost while making less accurate power predictions. The performance improvement achieved by E2E learning, for both types of systems (single renewable generator or two renewable generators), can be explained by the information about the cost function that is made available to the NN, which would not be true for sequential learning. The counterintuitive outcome of Case Study 12, whereby costs are reduced despite power prediction errors increasing, exemplifies the contrasting nature of E2E and sequential learning. The singular aim of E2E learning to optimize a task has been shown to improve performance, but if the intermediate power prediction is also of importance in the given problem setting, this form of training may not be ideal. For example, in a scenario where System 2 (with two renewable sources) from Case Study 13 has power limits in branches ($P_r^{f,lim}$) reduced to 50%, the dependence of system cost on prediction performance is drastically altered.

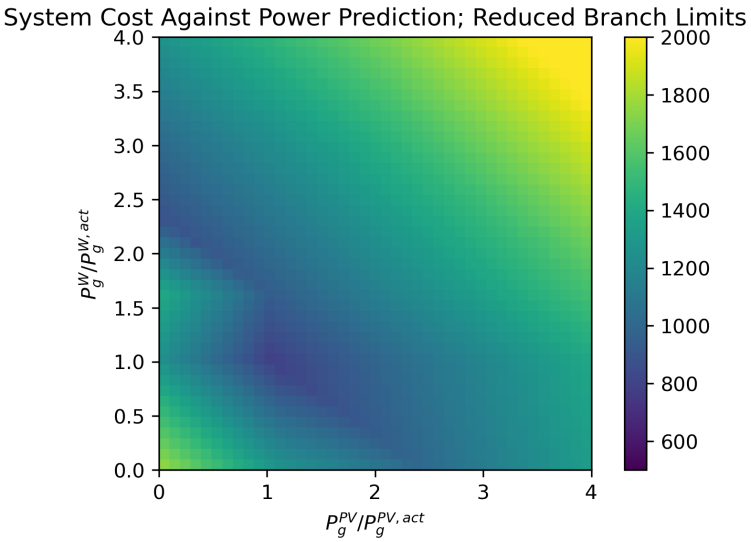

Figure 46: The expected variation of system cost based on PV and Wind power forecast accuracy. This plot is for a system with $P_d = 145$MW and $P_g^{PV,act} = P_g^{W,act} = 25$MW, with $P_r^{f,lim}$ reduced to 50%.

Figure 46 shows a clearly different plot to Figure 44 as the set of minimal solutions is no longer linear. There is a minimum at perfect power forecasting ($P_g^{PV}/P_g^{PV,act} = P_g^{W}/P_g^{W,act} = 1$), and apparently a local minimum when PV is underestimated ($P_g^{PV}/P_g^{PV,act} < 1$) and Wind is overestimated ($P_g^{W}/P_g^{W,act} > 1$). E2E training in this context would enable the learning of a more complex cost function, but would also risk often finding solutions in the local minimum. In this sense, it is a double-edged sword. E2E offers the possibility of learning such a function which would not be possible by just training on forecast accuracy. This variety of cost functions also demonstrates that when using an E2E trained algorithm in practice, the objective of the minimization should be carefully decided depending on the problem context. For example, if forecast accuracy needs to be prioritized, a multi-objective cost function could be considered (combining system cost and power prediction). Additionally, in an energy system where the levels of more renewable generators need to be independently forecasted, e.g. a system with PV, Wind, and concentrated solar power, the plot in Figure 44 would be extended to four dimensions. Similarly, a system with 10 generators at different locations would incur local cost minima in an 11 dimensional space.

Considering the viability of MM and E2E learning together, we can conclude that in the context of an optimization with a clearly defined unique optimum (as in Case Study 11), MM learning enhances the intermediate prediction, and therefore enables improved cost minimization results. However, if the cost function lacks a unique optimum (as in Case Study 12), then the E2E trained model is no longer necessarily aligned with prediction accuracy. MM learning is useful for enhancing prediction accuracy, and would therefore be less consequential in an E2E problem with non-unique optima.

6.2 Limitations

Through the analysis of the results from the case studies, several limitations of the methodology can be identified. Some of these are aspects that could be improved upon without drastically extending the scope of the project.

- **Method of combining multiple images**: the current methodology utilizes pixel-wise averaging to combine multiple images, thereby applying a form of temporal smoothing. This has not been compared with other methods, such as making a larger image out of multiple smaller ones, i.e. three 64×64 could become one 64×192 image. Alternatively, a deeper image could be constructed (with each one as a channel, therefore with an ultimate dimension of 64×64×3). Studying these alternatives would provide insight into the value (or lack thereof) of temporal smoothing.
- **Shallower networks**: the insights provided in Figure 45 show that, at least for some of the model architectures, the neuron weights of early layers are not effectively updated during training. This observation, along with the result of Case Study 10, where linear regression and MM4-Shallow outperformed MM4, indicates that a deeper network is not necessarily always better. It has been generally established that deeper networks have improved representational ability as they can detect more abstract features [HZRS15a]. But, in a setting where only low-resolution imagery and a fairly small set of sensor data is available, it is possible that abstract features are

lacking and therefore a less network with fewer parameters may train better [GBDK21]. This hypothesis could be studied by testing NNs with only 1 or 2 layers for feature extraction and prediction. On the other hand, experiments could be done with higher resolution imagery (if greater computational facilities are available) and in that scenario, we could expect deeper networks to generally outperform shallower ones.

- **Feature selection**: as mentioned previously, intelligent feature selection would be a possible improvement [Ole22]. Calculating the correlation between each feature in Table 12 and each other, as well as with the prediction target, would yield keen insights into the relevance and redundancy of each one. For instance, only including features which achieve a certain threshold of correlation with the target could be included, and features with too high correlation with each other cannot both be included. This is a valuable part of the data exploration phase which could later enable more efficient training of the NNs, as spurious correlations are avoided.
- **E2E cost function definition**: MM4-E2E is currently trained directly on minimizing the system cost, but this may not be the ideal training objective. An analogous metric, designed through normalizing costs, for instance, could yield better outcomes. It has been shown that the absolute difference between the perfect forecast cost and the E2E training cost is relatively small (965.5 vs. 1001.5, or about 7%). The perfect forecast cannot be improved upon, so perhaps training the network on the 'excess cost' would yield larger error gradients and more effective parameter update.
- **Computational time**: although E2E learning has been shown to produce better outcomes in terms of cost minimization, training the MM4-E2E NN takes about 80$\times$ longer than training MM4-Seq. This discrepancy should be considered in the context of the problem setting; if the model needs to be frequently re-trained, sequential learning may be more suitable.

The subsequent section describes broader improvements or extensions to the methodology.

6.3 Further Work

There are several extensions that can be made from this work concerning the broad scope of E2E learning from MM data for energy system optimization. Case Studies 2 and 4 demonstrated the added benefit of using several recent subsequent measurements as inputs to a NN for forecasting renewable generation, regardless of the modality of data. This observation could be developed by employing a Recurrent Neural Network (RNN) architecture, for instance, which takes advantage of the sequential nature of its inputs for forecasting the future. The Long-Short Term Memory (LSTM) RNN architecture may be particularly effective as it enhances the ability of the NN to capture long-term trends by mitigating the vanishing gradient problem [GSC99]. With or without a RNN, a deeper investigation into the optimal number of previous instances of data could be considered (only the uses of three and one previous instances were studied thus far). This would be a form of HPO which should be conducted in general for a variety of hyperparameters such as learning rate, batch size, number of visual modality features extracted by CNN-fe, etc.

Regarding the fusion of MM features, the outcome of Case Study 6 asserted that a FNN for feature extraction from meteorological sensor data was not beneficial for GHI forecasting. Figure 45 exhibits the possible prevalence of the vanishing gradients problem during the training of the MM2 architecture, which may also occur while training the MM4 network. Mitigating this issue could involve changing the design of the NNs, for instance by including batch normalization, and should be further investigated. In addition, pre-training the feature extraction networks CNN-fe and FNN-fe to predict GHI may also improve the ultimate performance of MM2 and MM4. MM2 may also perform better with a larger training set and longer time horizon (thereby more features), since the NN parameters are likely overfitting the current training set. The added complexity of the network, compared to MM4, probably requires more data to show its value. Another improvement to both MM2 and MM4 could be the inclusion of features from different stages of the network architectures in the MM joint representation [Mor20]. For example, involving early features from the images in the MM4 architecture would effectively bring them closer to the prediction and loss function calculation. This change could increase the magnitudes of error gradients of the early layers during backpropagation, thereby mitigating the vanishing gradients problem and enabling more effective parameter update.

Looking ahead from Case Study 11 and 12, it has been demonstrated that E2E learning reduces costs compared to sequential learning. The effect of E2E training when applying a variety of cost functions within DCOPF-Schedule and DCOPF-Redispatch is also an interesting avenue of research. In practical energy systems such cost functions may be quadratic as opposed to linear, and (specifically for infeasibility costs) may not be symmetric for positive and negative redispatch. Given this knowledge of more complex cost functions, it is foreseeable that E2E learning would again outperform sequential learning, but this cannot yet be ascertained. In addition, it is worth reminding

that proportion of C_{sch}/C_{sys} (= 1 for the Perfect Forecast) and the reduction of redispatch cost is dependent on the setting of thermal generation cost parameters (c_b^g, c_{up}, c_{down}) and the infeasibility costs ($\gamma_1 ... \gamma_8$). In a context where c_b^g are increased relative to other costs, the ratio C_{sch}/C_{sys} can be expected to increase, and as a result, E2E learning may have an even smaller effect on reducing C_{sys}. However, in an energy system where renewable generators represent a growing portion of total power generation (as is the case in many Northern European countries), scheduled thermal generation is less significant. This can be expected to increase the scope for E2E learning to reduce costs.

Finally, the scope of the research may be expanded by including Multi-Task (MT) learning in addition to MM and E2E learning. In this domain, multiple tasks are predicted simultaneously, and the similarities between the different prediction problems are leveraged to improve each of them. For this reason, the methodology is also referred to sometimes as 'hints' [Car97, Rud17]. MT learning could, in the context of this thesis, incorporate the tasks of minimizing system cost and maximizing renewable power prediction accuracy.

7 Conclusion

This thesis has investigated the use of Multi-Modal (MM) data alongside End-to-End (E2E) learning in the context of energy system optimization. The modalities studied are sky imagery and meteorological sensor data. The optimization seeks to minimize the total monetary cost of meeting power load while maintaining grid stability in a system with thermal generators and at least one renewable generator (either Photovoltaic (PV) or Wind). A conventional Machine Learning (ML) based approach would involve training an algorithm to forecast renewable generation, and use the prediction as one of the inputs to the optimization, thus solving for the minimized cost. The training of a Neural Network (NN) for such a task entails quantifying the error of the predicted power against the true power using a loss function, and differentiating this error with respect to the model's parameters. These parameters are then updated over several iterations (based on the error gradient) to minimize the error. Employing E2E learning involves training instead on the 'true' task, i.e. minimizing the system cost, rather than the intermediate task of predicting power. We expected that using MM data enhances power predictions and therefore also benefits E2E learning. Our proposed methodology involves using a Convolutional Neural Network (CNN) to extract relevant features from the sky imagery, followed by a MM learning approach to fuse those with the sensor data. The fused features are used by a Feedforward Neural Network (FNN) to predict power, and this prediction is used as an input to the E2E-based power system optimization.

To test the methodology, several case studies were designed. These case studies investigate the optimal design of the NNs, as well as the suitability of different models for E2E economical optimization. It is found that, in a system with a single renewable source (PV), using the proposed MM4 architecture for forecasting power reduces RMSE by 18% compared to a linear regression approach for combining the MM data. We explain this improved performance by the ability of MM learning to enhance one modality with the other; since the features are combined at an intermediate stage before predicting, the NN learns to optimally weigh the diverse features to predict power. A linear regression approach only learns to make optimal predictions from each modality separately, and subsequently weighs the predictions. In addition, the proposed MM4-E2E approach outperforms a conventional training approach, as it yields system costs 7% higher than a perfect forecast, while the conventional approach yields costs 10% higher (for the system with PV generation). When testing in a system with two renewable sources (PV and Wind), the proposed model (MM4-E2E) again results in the lowest costs (4% higher than perfect forecasting, compared to 7% from a sequential training approach). However, in this setting, power predictions by MM4-E2E were less accurate, indicating that, depending on the power system and its cost functions, minimal costs can be achieved without necessarily predicting power more accurately. This result demonstrated the keen dependence of E2E learning performance on the nature of the system cost function.

References

[ASMA20] R. Ahmed, V. Sreeram, Y. Mishra, and M. D. Arif. A review and evaluation of the state-of-the-art in PV solar power forecasting: Techniques and optimization, 2020.

[BAM19] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis Philippe Morency. Multimodal Machine Learning: A Survey and Taxonomy, 2019.

[BDE17] BDEW. Standardlastprofile Strom, 2017. data retrieved from World Development Indicators, `https://www.bdew.de/energie/standardlastprofile-strom/`.

[Beh03] Sven Behnke. *Hierarchical Neural Networks for Image Interpretation*, volume 2766. 01 2003.

[BJZP20] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan. Gradient amplification: An efficient way to train deep neural networks. volume 3, pages 196–207, 2020.

[BLC+16] Hieu Minh Bui, Margaret Lech, Eva Cheng, Katrina Neville, and Ian S. Burnett. Using grayscale images for object recognition with convolutional-recursive neural network. *2016 IEEE 6th International Conference on Communications and Electronics, IEEE ICCE 2016*, pages 321–325, 9 2016.

[Car97] Rich Caruana. Multitask Learning. *Machine Learning*, 28, 1997.

[Cho17] Francois Chollet. *Deep Learning with Python*. Manning, 2017.

[CPZ19] Sauhaarda Chowdhuri, Tushar Pankaj, and Karl Zipser. Multinet: Multi-modal multi-task learning for autonomous driving. *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, pages 1496–1504, 3 2019.

[DAK17] Priya L. Donti, Brandon Amos, and J. Zico Kolter. Task-based end-to-end model learning in stochastic optimization. volume 2017-December, 2017.

[Del22] TU Delft. Photovoltaic Materials and Devices, Monitoring Station Dataset, 2022.

[Due19] Peter Dueben. AI and Machine learning for weather predictions, 2019.

[DYSC12] Mehmet Demirtas, Mehmet Yesilbudak, Seref Sagiroglu, and Ilhami Colak. Prediction of solar radiation using meteorological data. *2012 International Conference on Renewable Energy Research and Applications, ICRERA 2012*, 2012.

[ECM22] ECMWF. European Centre for Medium-Range Weather Forecasts, 2022.

[eco17] *Electricity Markets: Theories and Applications [Power System Economic Dispatch]*, chapter 5, pages 147–171. John Wiley & Sons, Ltd, 2017.

[Edu20] IBM Cloud Education. Convolutional Neural Networks. `https://www.ibm.com/cloud/learn/convolutional-neural-networks`, October 2020.

[EPE22] EPEXSpot. Basics of the Power Market. `https://www.epexspot.com/en/basicspowermarket#day-ahead-and-intraday-the-backbone-of-the-european-spot-market`, 2022.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[GBDK21] Ankit Goyal, Alexey Bochkovskiy, Jia Deng, and Vladlen Koltun. Non-deep networks, 2021.

[Ger19] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly, 2019.

[GH18] Sylvain Gugger and Jeremy Howard. AdamW and Super-convergence is now the fastest way to train neural nets, 2018.

[GSC99] F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999.

[HZRS15a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[HZRS15b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[Ipp19] Pier Ippolito. Feature Extraction Techniques. https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be, October 2019.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[Isa21] Isabella, Olindo and Ziar, Hesan. Lectures in Photovoltaic Systems, February 2021.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[KLB+18] Qiuhong Ke, Jun Liu, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. Chapter 5 - computer vision for human–machine interaction. In Marco Leo and Giovanni Maria Farinella, editors, *Computer Vision for Assistive Healthcare*, Computer Vision and Pattern Recognition, pages 127–145. Academic Press, 2018.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[LAJ15] Dana Lahat, T. Adalı, and Christian Jutten. Multimodal data fusion: An overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103:1449–1477, 2015.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LG 21] LG Electronics. LG NeON Bifacial, 2021.

[LH17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017.

[LLXN18] Kuan Liu, Yanen Li, Ning Xu, and Prem Natarajan. Learn to combine modalities in multimodal deep learning, 2018.

[LWL+19] Zhuo Li, Kejie Wang, Chenchen Li, Miao Zhao, and Jiannong Cao. Multimodal Deep Learning for Solar Irradiance Prediction. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 784–792, 2019.

[Mor20] Louis-Philippe Morency. Multimodal Machine Learning, September 2020.

[NRB20] Ricardo Nanculef, Petia Radeva, and Simone Balocco. *Chapter 9 - Training Convolutional Nets to Detect Calcified Plaque in IVUS Sequences*. Elsevier, 2020.

[NRU21] Eshaan Nichani, Adityanarayanan Radhakrishnan, and Caroline Uhler. Do deeper convolutional networks perform better?, 2021.

[Ole22] Michal Oleszak. Feature Selection Methods and How to Choose Them. https://neptune.ai/blog/feature-selection-methods, September 2022.

[owe22] OWEZ. https://en.wikipedia.org/wiki/OWEZ, 2022.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature 1986 323:6088*, 323:533–536, 1986.

[RNE16] Muhammad Qamar Raza, Mithulananthan Nadarajah, and Chandima Ekanayake. On recent advances in pv output power forecast. *Solar Energy*, 136:125–144, 10 2016.

[RSS15] Ye Ren, P.N. Suganthan, and N. Srikanth. Ensemble methods for wind and solar power forecasting—a state-of-the-art review. *Renewable and Sustainable Energy Reviews*, 50:82–91, 2015.

[Rud17] Sebastian Ruder. An Overview of Multi-Task Learning for Deep Learning, 2017.

[SAR20] Reza Sabzehgar, Diba Zia Amirhosseini, and Mohammad Rasouli. Solar power forecast for a residential smart microgrid based on numerical weather predictions using artificial intelligence methods. *Journal of Building Engineering*, 32, 2020.

[SBPK18] Robin Spiess, Felix Berkenkamp, Jan Poland, and Andreas Krause. Learning to compensate photovoltaic power fluctuations from images of the sky by imitating an optimal policy, 2018.

[SKKR18] Sobrina Sobri, Sam Koohi-Kamali, and Nasrudin Abd Rahim. Solar photovoltaic generation forecasting methods: A review, 2018.

[sl14] scikit learn. Underfitting vs. Overfitting. `https://scikit-learn.org/0.15/auto_examples/plot_underfitting_overfitting.html`, 2014.

[Sol21] Solcast. Global solar irradiance data and PV system power output data, 2021.

[Stu95] Stull, Roland. *Meteorology for Scientists and Engineers*. The University of British Columbia, 1995.

[SZMM10] Saurabh S. Soman, Hamidreza Zareipour, Om Malik, and Paras Mandal. A review of wind power and wind speed forecasting methods with different time horizons. *North American Power Symposium 2010, NAPS 2010*, 2010.

[Wah21] Dariush Wahdany. End-to-End Learning for Sustainable Energy Scheduling. 2021.

[win21] wind-turbine-models.com. Vestas V112-3.45. `https://en.wind-turbine-models.com/turbines/1247-vestas-v112-3.45#powercurve`, 2021.

[WLB+14] Pascal Wallisch, Michael E. Lusignan, Marc D. Benayoun, Tanya I. Baker, Adam S. Dickey, and Nicholas G. Hatsopoulos. Chapter 36 - neural networks part i: Unsupervised learning. In Pascal Wallisch, Michael E. Lusignan, Marc D. Benayoun, Tanya I. Baker, Adam S. Dickey, and Nicholas G. Hatsopoulos, editors, *MATLAB for Neuroscientists (Second Edition)*, pages 489–500. Academic Press, San Diego, second edition edition, 2014.

[XWCL15] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.

[ZCP+17] Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Tensor fusion network for multimodal sentiment analysis, 2017.