POLITECNICO DI TORINO

Master degree in Energy and Nuclear Engineering

Master Degree Thesis

Data driven models for renewable energy and load forecasting

Supervisors
Prof. Maurizio Repetto
Prof. Francesco Grimaccia

Co-supervisor
Ivan Mariuzzo

Candidate
Andrea Margiotta

Academic Year 2020/2021

Summary

The progressively advanced diffusion of energy from renewable sources leads to increasingly articulated networks and their more complex management. Among the main issues to be addressed there are the aleatory of renewable energy sources (RES), how to predict the power they feed into the grid, and how to match all the different types of current energy production. Also, both electrical and thermal load curves have been shaped to new lifestyles and consequently bring with them greater difficulties in being able to forecast these loads. Thus from these themes, this thesis is born to be able to use more innovative methods in making predictions. These methods adopted are based on machine learning and artificial neural networks applying data driven models.

Therefore researches were done to find forecasting methods using artificial neural networks. Next, two renewable energy sources and a load curve were selected in order to fine-tune these forecasting techniques. The RES selected are a photovoltaic plant with data collected online, and a solar thermal collector system with data provided by `SOLID Solar Energy Systems GmbH, Graz, Austria`. Eventually, the load curve forecast concerns a district heating system located in *Alba, Italy* with data provided by `EGEA SPA`. For each part, results were presented regarding the accuracy of the models built.

Contents

List of Figures

Introduction

Climate change is a constant reality nowadays, ever stronger storms, droughts advancing everywhere on the planet, rising sea levels, social gaps widening between populations, are just some problems we will have to face in the future. Human activities have already caused approximately $1°C$ increase in global temperature, with the risk of reaching $1.5°C$ between 2030 and 2050 [1]. This rise, as is well known, is caused by the increase in greenhouse gases. First and foremost, carbon dioxide, which has never reached its current level[10] in the last 650000 years[11].
According to the *IPCC*, to mitigate the impacts of climate change as far as possible, CO_2 emissions will have to reach the net zero global emissions around 2050 [1].
In addition to the reasons mentioned above, an energy transition is also needed to fight air pollution (which causes 7 million deaths worldwide[12] every year) and the depletion of fossil fuels.
This transition can be achieved if action is taken on all sectors that cause these emissions, such as energy, transport, agriculture, forest & land use, industry and waste.

The energy sector is the one that most produces and releases greenhouse gases and pollutants into the environment. Within this, there is a large portion for electricity and another one for thermal energy production. Taking into account the fact that even today almost 1 billion people do not have access to electricity, electrification, the continuing increase in population and the need for a level playing field, it is clear that the energy sector will always continue to grow up [2].
Therefore it is very important to actively increase different actions as shift all electricity production to renewables, change current building heating systems, electrifying transport, increase carbon capture & sequestration, encourage the use of green hydrogen and work for the possibility in the future of using nuclear energy from fusion, which is still a technology under study.

[10] December 2020 value: 414.49 ppm, Global Monitoring Laboratory. `https://www.esrl.noaa.gov/gmd/ccgg/trends/global.html`

[11] Global climate change `https://climate.nasa.gov/`

[12] `https://www.who.int/health-topics/air-pollution#tab=tab_1`

Among the renewables (excluding hydro), photovoltaic and wind power are the most widespread and the most promising, also thanks to the continuous drop in production costs.
For building energy sector, district heating (DH)[13] is one of the most important approaches, also because thanks to it, it is possible to integrate well different renewable thermal energy systems such as solar thermal collectors, geothermal energy and heat pumps. It will also be possible to integrate a district cooling network that in the future (around 2070) will exceed the current worldwide heat demand, will therefore become an increasingly important part of the energy demand in buildings [3].
However, cooling can also be produced from renewable sources or through what is called trigeneration. Using electrical energy via electrical chillers, or using heat via absorption chillers. This latter method is suitable for warm climates where the thermal demand is very low compared to the cooling one, with the only limitation being the type of system to dissipate excess heat [5].
Several communities have relied on district heating based on renewables. For instance in Iceland, almost 90% of citizens are served by a geothermal network[14]; in the Austrian city of Graz, solar thermal energy covers 20% of the district heating network's needs [7]; or in a small community of 52 homes called Drake Landing Solar Community[15], which through a solar collector system and seasonal borehole thermal energy storage, manages to have a solar fraction for district heating of over 90% with peaks of almost 100% [68].

However a massive and intensive use of renewables creates some issues, including low predictability of energy production, fast variability in time and weather sensitive [8]. According to the *World Energy Council*, this energy transition involves complying with three paradigms simultaneously, called *Energy Trilemma*, that are [4]:

- **Energy Security**: ability to meet current and future energy demand, as well as to withstand and react to system shocks minimizing disruption to supplies.
- **Energy Equity**: capacity to provide universal access to reliable, affordable and abundant energy all over the world.
- **Environmental Sustainability**: relevance to mitigate and avoid environmental degradation and impacts of climate change.

[13] According to [6] reference, DH it is placed at number 27 among the top 100 solutions against global warming.

[14] `https://www.euroheat.org/knowledge-hub/district-energy-iceland/`

[15] `https://www.dlsc.ca/`

Precisely because of low predictability of renewables (due to the difficulty in having an accurate forecast of generated power) this thesis was born, going to analyse innovative systems to make predictions. The different forecasting methods were then analysed, going into specifics of machine learning with artificial neural networks. These methods have been found to achieve excellent results and accuracy in many different applications.
After learning the methodologies and carrying out various tests in many areas, three different energy problems were specifically sought to be predicted. Starting with simple models and then moving on to more complex ones.

The first part of the thesis therefore focuses on describing the structure and functioning of data driven models, mainly based on neural networks and tools used during this work.
In the second part, the three selected cases are the following, composed of photovoltaic, solar thermal energy and district heating. In all three cases the methodology followed is the same with research and obtaining the data to be used; analysis and cleaning of these data; creation of a network and critical analysis of the results obtained.

Part I

Theoretical Background

Chapter 1

Machine learning

Machine learning is a branch of Artificial intelligence. It takes its cue from the learning mechanism of the living being, where the experiences made lead to knowledge. In normal programming the inputs are algorithms or rules, and data given to the program using the algorithm itself providing as results the solutions. Instead in machine learning, data becomes the inputs and as results the program gives the algorithms in order to obtain the specific solutions from that data. Moreover the machine does not have to be programmed but rather it has to be trained. The training can be defined as the heart of machine learning. It is partly based on statistical methods, but the main difference is that machine learning deals with a huge amount and heterogeneous data, so one of the important issues is Data Analysis. This process is composed of three important phases: the collection, the organization and the cleaning of the data. Only after that, machine learning could see these data and actually use them [10].

1.1 Applications

The main uses of machine learning are Classification, Clustering and Regression [10].

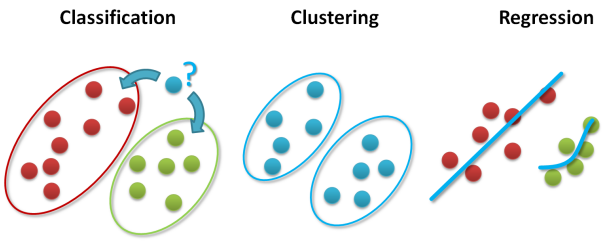


Figure 1.1: Applications of Machine Learning[1]

[1] http://www.big-data.tips/machine-learning-methods

Classification It's about the categorization of some data into specific classes. As input has continuous number and as output discrete variables. The most used algorithms in this application are Decision Tree, Artificial Neural Networks and K-nearest neighbor.

Clustering When data are part of the same groups and therefore can be divided and grouped according to their attributes. Unlike Classification the classes are not provided a-priori but the program has to be able to identify if the data are similar to each other and eventually grouped all together. The most adopted algorithms in this application are K-means Clustering, Mean-Shift Clustering and Agglomerative Hierarchical Clustering.

Regression Unlike Classification and Clustering, Regression works only in continuous variables both as input as output. It can works with simple or multiple models, having more than one feature, and with linear or non linear models. For this application some of the most important algorithms are Decision Tree Regression, Random Forest Regression and Artificial Neural Networks.

1.2 Types of learning

Supervised Learning This is mostly used for Classification and Regression problems. The learning consists of two phases, *Training* and *Prediction*. The first represents the most important part of the learning because is the phase in which the model is really created and trained with the dataset. The inputs for the computer in Training phase are constituted by the so-called *Training Set*, composed of input and output data. After the model is well trained Prediction phase could be applied, where a new set of input data is sent to the model providing the desired output. For example, this type of learning is extremely used for image recognition: a large amount of picture is sent to the program with the explanation of what every picture means, and after the model is trained, it is able, by viewing a new picture correlated to the ones seen in training, to classify it according to previously defined qualities. Other applications are to predict the price of a certain stock in the market or to make meteorological predictions.

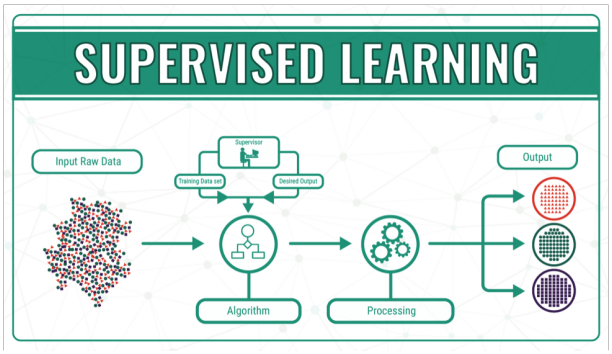


Figure 1.2: Scheme of Supervised Leaning [16]

Unsupervised Learning This is normally used for Clustering applications where there are several sets of data not labelled, and the purpose is to group these data. It is up to the program to understand the pattern behind the data, so human intervention, in this case, is limited. Examples are credit card fraud detection or email spamming. Another possible great application is for disease recognition, like a tumor, where without labelling each figure that is sent to the model, this learns by itself if, for example, a person may or may not has lung cancer. This type of learning also falls under Semi-supervised learning, a mix of Unsupervised and Supervised learning, that requires the labelling of a portion of the data and helps the program to find the pattern of the unlabeled data.

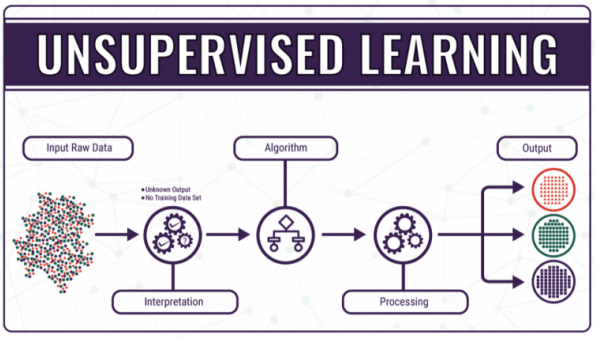


Figure 1.3: Scheme of Unsupervised Leaning [16]

Reinforcement Learning It is similar to Unsupervised learning because the raw data are unlabelled, but the process is different. Through a loop scheme, an agent is guided thanks to feedback in the environment where it is located. These feedback are positive or negative according to an evaluation system. This type of learning is used mainly for Regression, especially for robotics or automatic car without a driver.

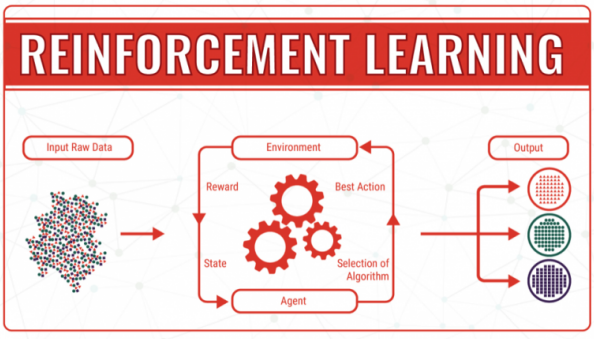


Figure 1.4: Scheme of Reinforcement Leaning [16]

Chapter 2

Artificial Neural Network

Artificial Neural Network (ANN) is a tool used in Machine Learning, in specific in Supervised Learning. The main applications are Regression and Classification. The concept of ANN starts right from the architecture and functioning of the human brain neural network. The human brain is extremely complicated and can be approximated to a vast neural network. Almost 100 billions of neurons form this network [12]. Each neuron is linked with thousands of other neurons, with connections (called synapses), and this leads to millions of billions of links.

2.1 Structure

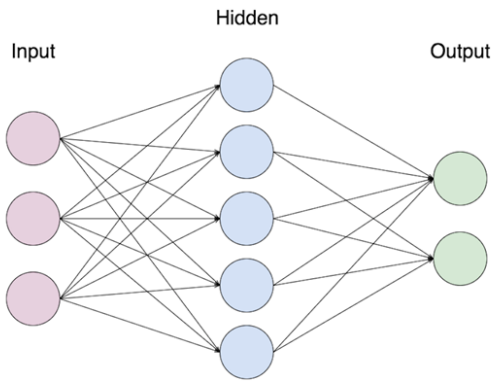


Figure 2.1: Scheme of a simple Artificial Neural Network[1]

The main components of an ANN are layers, neurons and connections. Layers are the fundamental structure of the network, and they are composed of a certain number of neurons. Different types of layers exist for different applications. Each neuron is generally connected with all the neurons in adjacent layers.

[1] https://laptrinhx.com/titanic-prediction-with-artificial-neural-network-in-r-3087367370/

Data are supplied to the input layer, composed only with samples (that are the numbers of elements available) and features (that are the variables available) of the dataset[2]. Then one, or more, layers (called *Hidden layers*) are composed of a certain number of neurons that are connected to the input layer. Finally, there is the output layer, which can be composed of one single neuron or more than one [19].

Each neuron in a layer interacts with the other through a specific unit, called Linear Threshold Unit, composed of Transfer and Activation function.

Transfer and Activation function

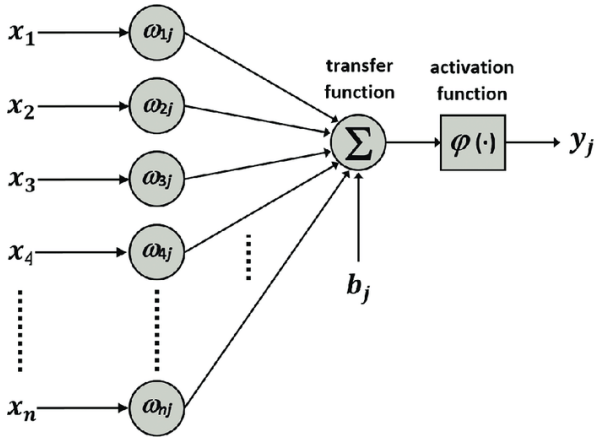


Figure 2.2: Mathematical scheme of a layer[3]

The Transfer function performed the sum of each neuron multiply by its weight $[W]$ and its bias $[b]$, as explained in equation 2.1. The weight indicates the importance of a specific connection and consequently the importance of the feature involved. So a low value, or zero value, of weight means that the specific feature is less important.
The bias is a coefficient useful to shift the activation function. The generic output from a layer will be:

$$z_i = \sum_{i=1}^{N} W_{ij} x_i + b_i \tag{2.1}$$

[2]Indeed the input data in the input layer could be a vector or a matrix, having as rows the samples of the dataset and as columns the features.

[3]`https://www.intechopen.com/books/sleep-apnea-recent-updates/usefulness-of-artificial-neural-networks-in-the-diagnosis-and-treatment-of-sleep-apnea-hypopnea-synd`

Where:

- **N**: is the total number of neurons in a layer.
- **W**: is the weight associated to that specific neuron.
- **x**: is the input of the layer.
- **b**: is the bias.

The activation function [ϕ] instead plays a key role in the ANN, introducing a non-linearity into the network [23].

$$y_i = \phi(z_i) \tag{2.2}$$

Several kinds of functions can be arranged and they determine if a neuron should be activated (and so its value stored) or not, according to its weight and consequently its importance for the model.
The most simple function is the linear activation where the input and output are equal. This requires the lowest possible computational cost compared to non-linear functions but would limit the ability of the model to generalise as much as possible.
The choice of the best activation function for a hidden layer depends on the type of network used, while the choice for the output layer depends on the type of application [17].
It is therefore understood that the same activation function is not used for all layers of an ANN.

The subdivision of the activation function for hidden layer is highlighted in table 2.1.

Type	**Activation**
Multilayer Perceptron	ReLU
Convolutional ANN	ReLU
Recurrent ANN	Sigmoid
	Tanh

Table 2.1: Activation function for hidden layer

While the subdivision for the output layer is highlighted in table 2.2.

Type		**Activation**
Regression		Linear
Classification	Binary	Sigmoid
	Multiclass	Softmax
	Multilabel	Sigmoid

Table 2.2: Activation function for output layer

The most commonly activation functions are described.

Sigmoid function Called also the logistic function it's described with the equation:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

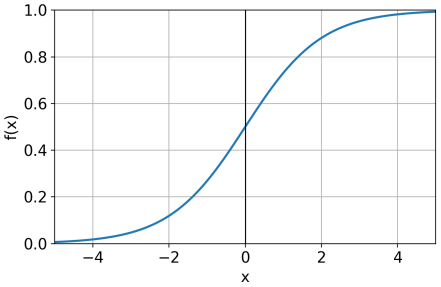


Figure 2.3: Sigmoid function

It is the most common activation function even if it is no longer used as much as in the past. The output range of the function is [0,1], therefore for small values sigmoid returns values close to 0, and for large values, the output is close to 1 [15]. However, for high or low values of input it causing a problem called *Vanishing gradient*[4].

Tanh function It is the hyperbolic tangent activation function, with the equation:

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.4}$$

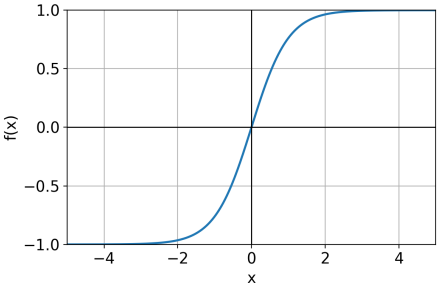


Figure 2.4: Tanh function

The output range is [-1,1], and also tanh has a s-shaped curve. It has the same disadvantages for Sigmoid function but the advantages of making it easier to model inputs with great negative or positive values.

[4]i.e. the model can no longer learn or becomes very slow with high computational costs.

Softmax It is a function that converts the numeric array into probabilities, where these are proportional to the numeric input value. This probability is useful to understand if input is in a specific class or not. The equation that described the function is:

$$softmax(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} for i = 1, ..., K \tag{2.5}$$

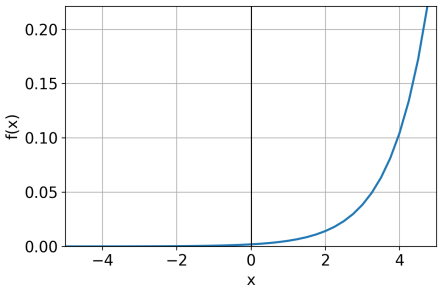


Figure 2.5: Softmax function

The main utilization is for Classification problems and as described in table 2.2 it is used for the output layer mainly.

ReLU It stands for Rectified Linear Units and despite its name, it's not linear but it works similar to Sigmoid function but with better performances. The equation is quite simple:

$$relu(x) = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases} \tag{2.6}$$

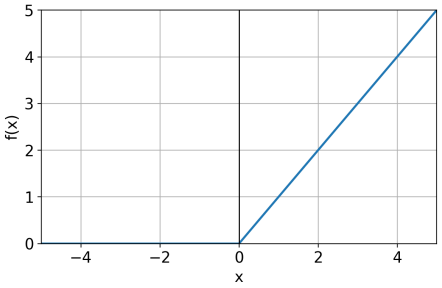


Figure 2.6: ReLU function

It's the most widely used activation function nowadays. As output has range in [0,∞], and respect to the others two functions described above have the most efficient computational cost due to its fast convergence. On the other hand, though it has a problem called *Dying ReLU* where for inputs negative or near zero, the model cannot learn.

Leaky ReLU It is a variant of ReLU in which instead of having 0 for $x < 0$ this function gave a small constant line with slope equal to α[5]. The equation is:

$$leakyrelu(x) = \begin{cases} x & \text{for } x > 0 \\ \alpha x & \text{for } x \leq 0 \end{cases} \tag{2.7}$$

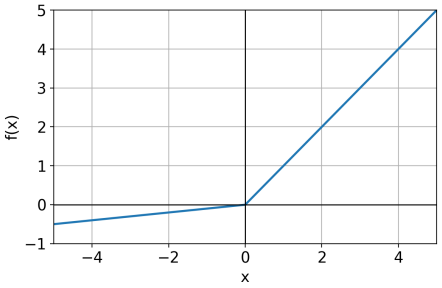


Figure 2.7: LeakyReLU function

This modification respect to ReLU function involves the disappearance of the *Dying ReLU* problem. Negative input values are allowed to pass trough the activation function with a small gradient, therefore its range output is in [-∞,+∞].

[5]In Keras default to 0.3 [15].

2.2 Training process

The main concept of an artificial neural network is the process of learning which leads to finding the best weights and biases for the specific problem. This process is called training, and it is the phase in which losses are calculated and through optimisation methods are then minimised [20].

2.2.1 Loss function

The purpose of this function is to calculate the losses $[C(W)]$ that a model should seek to minimize during training process [15]. The most important are [18]:

- **Probabilistic losses**
  - *Binary Crossentropy*: it's used for binary classification, where there are only two possible classes output.
  - *Categorical Crossentropy*: it adopted for multi-class classification. The neurons in output layer must be equal to the number of classes.
- **Regression losses**
  - *Mean Square Error [MSE]*: it's the sum of square distances between the target and predicted values. It's more stable to find solution compared to others but it's more sensitive to outliers.
  - *Mean Absolute Error [MAE]*: it's the sum of absolute differences between the target and predicted values. Respect to *MSE* is more robust to outliers but less efficient to find the solution.
  - *Mean Absolute Percentage Error [MAPE]*: it's the percentage ratio between *MAE* and the target value. It gives a very intuitive result being a percentage error thus giving the ability to compare different models for different applications. On the other hand, in case of target values close or equal to zero, it cannot be used leading to infinite values.

2.2.2 Optimizer

After calculating the errors via the loss function, these must be minimized (as mentioned above) via optimization methods. This is done by the so-called optimizer, which updates the coefficients (weights and biases) of the ANN. The model initially attributes a random value to its coefficients[6], then the task of the optimizer is to find in which direction moving to reach the minimum of the losses [25]. The main optimization algorithms are:

Gradient descent

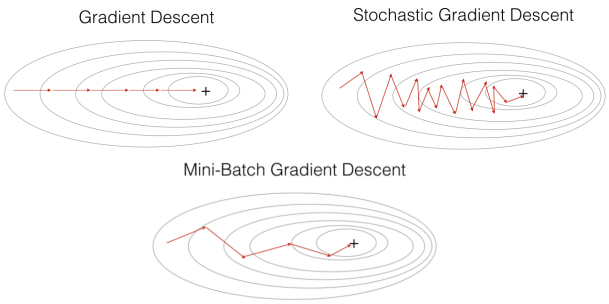


Figure 2.8: Optimizer based on gradient descent algorithms[7]

- *Gradient Descent (GD)*: it's the most simple to implement and the eldest invented. Through the first-order derivative of a function, it is possible to find the minimum or maximum values simply to find where this derivative is equal to zero. The same principle can be adapted to a function of multiple variables where the derivatives are called gradient. This gradient is then calculated for the whole dataset, updating every single coefficient of the ANN. It is very slow to achieve convergence and requires a high computational cost [25].

$$W = W - \alpha \cdot \nabla C(W; x, y) \tag{2.8}$$

- *Stochastic Gradient Descent (SGD)*: instead of updating the network parameters one by one, it updates them all at once at each epoch[8]. This consequently leads to continual changes in the coefficients and also to large fluctuations in finding the minimum as can be seen in figure 2.8. It reaches convergence in much less time than GD but given the large fluctuations it is very random to get better results [25].

$$W = W - \alpha \cdot \nabla C(W; x_i, y_i) \tag{2.9}$$

[6]Set by default according to the type of layer used [15].

[7]`https://edgeaiguru.com/Batch-vs-Mini-batch-vs-Stochastic-Gradient-Descent`

[8]Which represents the number of times that the algorithm works over all the training data [19].

- *Mini-Batch Gradient Descent (SGD)*: it is the middle way between the two previous algorithms by updating the parameters of every group of samples called batch (B). This leads to obtaining the advantages of both algorithms without their respective disadvantages. It still has a random component that leads to oscillations that are anyway smaller than SGD and furthermore the number of batches can be adapted to the specific problem [25].

$$W = W - \alpha \cdot \nabla C(W; B_i) \tag{2.10}$$

Momentum based

It consists of an addition to the gradient descend algorithm, with the aim of speeding up the training. Especially in the favourable direction while reducing oscillations in the other one. The equation describing this concept is:

$$V(t) = \gamma V(t-1) + \alpha \cdot \nabla C(W) \tag{2.11}$$

With the weights updating according to $W = W - V(t)$.
As can be seen from the equation the updating of weights take into account the current and the previous step.

Adaptive Moment Estimation [ADAM] [27]

It uses both first ($\hat{m}_t$) and second ($\hat{v}_t$) moments of gradients in order to obtain different propagation speeds. In fact it uses adaptive learning rates, that are speeded up in region away from the minimum and are slowed down closer to the minimum.

$$\begin{cases} \hat{m}_t = \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t = \frac{v_t}{1-\beta_2^t} \end{cases} \tag{2.12}$$

β_1 and β_2 are hyper-parameters that control the exponential decay rates. Finally the parameters could be update.

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.13}$$

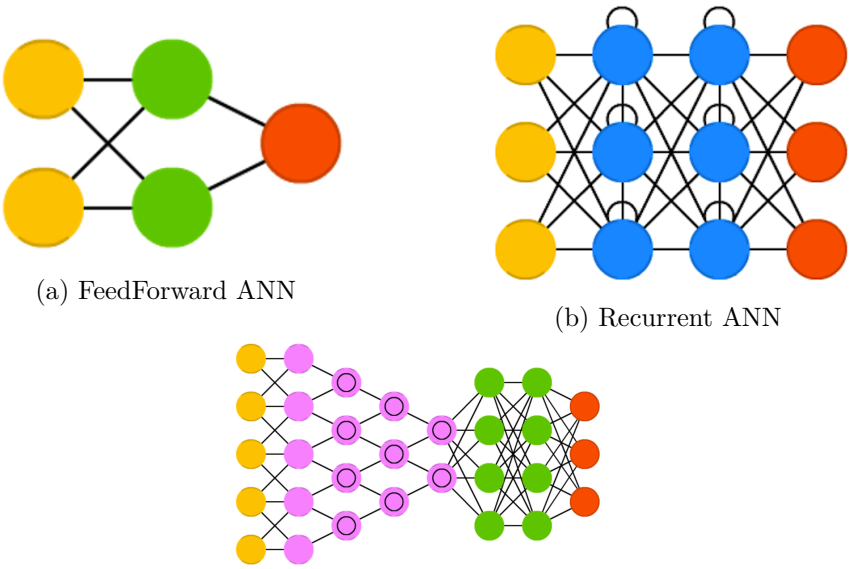
These algorithms explained above required that the objective function is differentiable, convex and continuous in all the solution domain. As these properties are not often known a-priori, other heuristic optimization algorithms are used, as for instance Genetic Algorithms [70].

2.2.3 Backpropagation

The procedure in which an artificial neural network is able to provide outputs from inputs, by means of weights & biases, is called forward propagation because the information travels from the input to the output layers of the model. This process eventually generates a scalar cost $C(W)$. Instead, the backpropagation algorithm allows the return of this scalar cost information obtained, calculating the derivative of the loss function with respect to the parameters of the ANN [20].
The gradient shows how much a parameter must change (in a positive or negative direction) to minimise the function of interest. However calculate the gradient numerically may require excessive computational costs, and the objective function (i.e. the cost function obtained from the loss function) depends on several variables. To overcome this problem, the chain rule is applied to calculate this derivative of the composite cost function. Therefore the algorithm would have to resolve already known derivatives that would then go on to create this chain from the output back to the input, thus giving the optimiser the ability to understand how to update the parameters to get closer to the predicted output [26].

2.3 Types of network

Nowadays there are so many types of ANN that can be called them the *Neural network zoo* [21]. The main types of ANN are now described in detail and visualized in figure 2.9.



(a) FeedForward ANN

(b) Recurrent ANN

(c) Convolutional ANN

Figure 2.9: Main artificial neural networks schemes [20]

2.3.1 FeedForward

They represent the simplest neural networks and the first to be used. Inputs propagate through the network in a unidirectional way (forward) entering the input layers and leaving in the output ones. In figure 2.9a, the input layers are indicated by the orange circles, the output by the red one and the green ones represent the hidden cells. The layers in this type of network are composed of these three cells and each neuron is fully connected to all the other neurons in the adjacent layers. A feedforward network with more than one hidden layer is called Multi-layer Perceptron (MLP).

2.3.2 Recurrent

In contrast with feedforward, in this type of network the neurons are connected recursively, creating cycles. This means that the network becomes much more complex and with a considerable increase in connections, and that the information for a neuron is provided by both the previous layer and the neuron itself. Therefore the inputs are both the weights and biases of the previous timestep[9] and the previous layer. One type of recurrent neural network is the Long Short Term Memory (LSTM) that can catch long and short time dependencies, and this property is shown to be crucial for timeseries forecasting. In the recurrent network, the repeating module is quite simple and formed by a single layer (for instance a tanh layer) instead for the LSTM this module is more complex and formed by 4 layers (as described in figure 2.10).

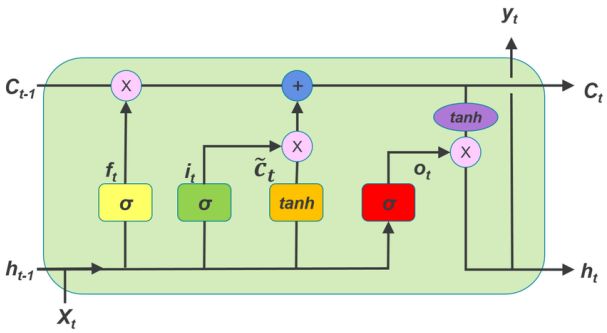


Figure 2.10: Repeating module in a LSTM and its main components[10]

The operation of LSTM is [28]:

1. Decide if an information is stored or not and this is done by the *forget gate layer* $[f_t]$. Both h_{t-1} and x_t pass trough this gate output in range [0,1].

2. Decide what new information will be stored. This operation is divided into two phases, the first one is a sigmoid layer called *input gate layer* $[i_t]$ where its output is the decision on the choice to update the value; the second one is a tanh layer $[\tilde{C}_t]$ that creates new available values that will be added to the state.

3. The C_{t-1} is now updated to the new C_t with the information of the two previous steps.

4. Finally decide what information will be sent as output through a sigmoid layer $[o_t]$ then to a tanh layer.

[9] Represents the amount of lag in the recurrent network and thus the number of previous observations to predict the next one.

[10] `https://www.researchgate.net/publication/324600237_Improving_Long-Horizon_Forecasts_with_Expectation-Biased_LSTM_Networks`

2.3.3 Convolutional

Compared to the other two networks, they are very different in many ways. The main use of this type of network is for image or audio classification. The cells drawn in violet in figure 2.9c are the ones that make it characteristic. First, these layers are not fully connected to each other as in the previous networks but are only connected to neighbouring neurons. The operation is based on the mathematical operator of convolution[11], which in practice behaves like a scanner that reduces the number of elements in the image component matrix. This reduction is performed to make image processing easier and as the information passes through the layers, it is reduced more and more and then provides classification information via a classical feedforward network.

[11] i.e. the operation between two functions that consists of integrating the product of the first and second convolution by a given value.

2.4 Data pre-processing and ANN working

The flow chart in figure 2.12 summarises a complete operation for creating a model data driven. The first block called **data handling** consists of a complete analysis of the data, a study to understand the usefulness of certain variables and the handling of missing or *Not A Number (NaN)* data.

Next, knowing how the optimisation algorithm works, it is important to carry out a **feature scaling** by modifying and adapting the range of each feature. In fact, with heterogeneous data, the optimiser would make large oscillations at each step with different distributions. With feature scaling, however, these fluctuations are reduced and the search for the minimum cost function is improved. The most common are [29]:

- *Normalization*: computed with Min-Max scaler, it maps the data in range [0,1].

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.14}$$

- *Standardization*: it creates a data distribution with zero mean and unit standard deviation.

$$x' = \frac{x - x_{mean}}{x_{std}} \tag{2.15}$$

It is then important to perform the **dataset split** into Training, Validation and Test subsets in order to be able to use the training process.

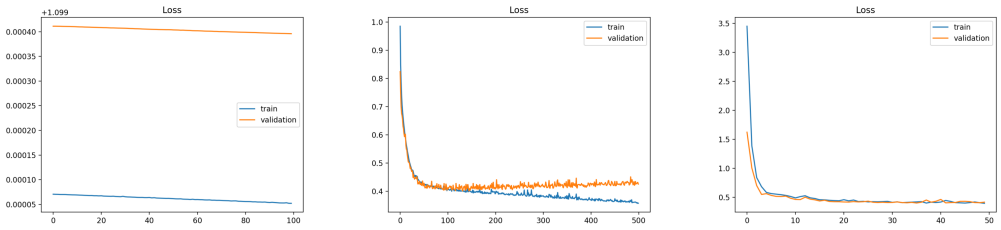
- Data used to build up the ANN model:
  - *Training*: this is the most important part of the set that is really used for training of the ANN. The program sees these data and tries to fit the model at each step.
  - *Validation*: a dataset used during training phase, for evaluating the ANN model at every iteration and necessary for fine-tuning of the architecture of the model.
- *Test*: it is the part helpful for the evaluation of the model accuracy since the program never sees these data during training and the scores are calculated after the training process is ended and ANN coefficients set.

Each subset is then divided into inputs and outputs[12], where the first are the variables used by the model to predict the second ones.

Training subset inputs enter the ANN structure and thanks to the backpropagation the information about the loss score is propagated to the layers, updating the values of weights and biases (according to the selected optimizer).

[12]Which will be called x and y respectively.

Once a convergence is reached or a set number of epochs have been completed, **learning curves** are created through which it is studied whether the network parameters have been selected well and were able to generalise the problem studied [30].
The most important cases are [31]:



(a) Underfit (b) Overfit (c) Good fit

Figure 2.11: Different types of learning curves [31]

1. *Underfit*: it happens when the model is stopped too early or when it is too simple for the complexity of the problem.
2. *Overfit*: it occurs when the model is too trained and the validation loss starts to increase instead of decreasing.
3. *Good fit*: it's the goal of the ANN model, in which for all the training process the training and validation curves are stacked means that the model is able to perfectly generalised the problem.

It may also be the case that the datasets used are not representative of the problem being addressed. Unrepresentative training dataset leads to a validation curve that is always above the training one while unrepresentative validation dataset leads to the opposite case. Methods for solving overfit problem are:

- *Dropout*: it consists of randomly discarding a part of the parameters obtained at the output of a layer and this results in a slowdown in finding convergence but also a remedy to overfitting. The more the omitted part increases, the more chaotic and random the learning process becomes.
- *Regularization*: it applies a penalty that is added up to the loss function. This penalty could be calculated as *L1* (calculated as the sum of the absolute of the weights) or as *L2* (calculated as the sum of the square of the weights). It could be applied in kernel[13], bias or output of the layer.

[13]The name the program gives to the weight.

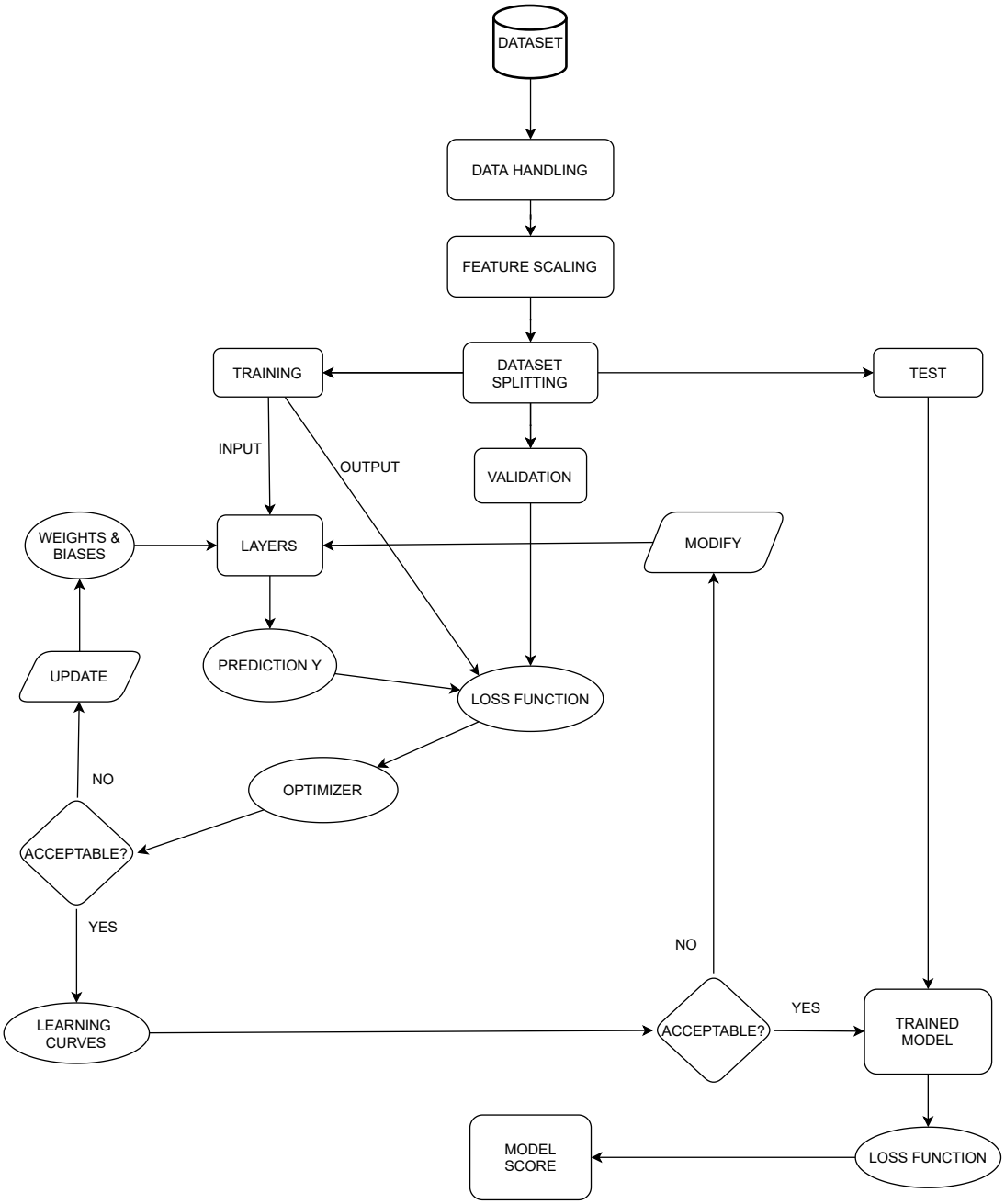


Figure 2.12: Flow chart of a complete operation for creating a model data driven

Chapter 3

Python tools

The only environment used is *Python* thanks to *Spyder* program, with different libraries that performed different task.

3.1 Data analysis

In this section are presented the tools for data processing, the first step while using an ANN since it works with the data provided as input and it is essential that these are provided in an appropriate manner.

3.1.1 Pandas

Through this library, it is possible to perform all possible actions to obtain a dataset suitable for the network [11]. The first step is to import this library:

```
import pandas as pd
```

then read the excel or CSV dataframe[1] with the proper command:

```
df = pd.read_excel
df = pd.read_csv
```

depending on the type of file used, one command will be used rather than the other. It's important to use only the sheet containing the data useful for the model. Also, the rows and columns are selected, skipping the rest.
Working with data measured by instruments that collect measurements at every available interval, the datetime columns must be treated properly:

```
df['datetime'] = pd.to_datetime(df['datetime'])
df.set_index(df['datetime'], inplace=True)
```

With this command, the program understands that the data in that specific column is a datetime type and sets that column as the index of the dataframe.

[1] Indicated from now on as df.

Afterward all the missing data, if they are present, must be found and substituted with a real number. The possibility of missing data came from errors during the measurement, or outages and they are read by the program as *Not A Number (NaN)* mainly.

```
df.fillna(method='backfill', inplace=True)
df.interpolate(method='linear', inplace=True)
```

There are two common methods to fill the missing values, the first is called *fillna* and the second works trough specific interpolation. The most common and widely used methods are *backfill*, *fowardfill*, *linear*, *quadratic* and so on.

Finally the dataset must be cleaned from the variables, or features, that are not relevant in the specific applications. A first tool to understand if a variable is importance or not, is the Pearson correlation coefficient:

$$\rho_{x,y} = \frac{cov(x,y)}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x_i})(y_i - \bar{y_i})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x_i})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y_i})^2}} \tag{3.1}$$

Thanks to this coefficient a value that goes from +1 (positive linear regression) to -1 (negative linear regression) is obtained and more a variable is near to these two values more it is correlated with the variable of interest and therefore it has meaning for the model. In python thanks to pandas the command line is `coeff = df.corr()`. In this way, a matrix with a unit diagonal will be created.

3.2 Build the ANN

Artificial neural networks can be constructed using different libraries; in this thesis, *keras* was mainly used. Before building the actual ANN, feature scaling and data splitting must be realized, as shown well in the diagram in figure 2.12.
Feature scaling:

- *Normalization*

```
min_df, max_df = df.min(), df.max()
scaled = (df - min_df) / (max_df - min_df)
```

- *Standardization*

```
mean_df, std_df = df.mean(), df.std()
scaled = (df - mean_df) / (std_df)
```

For the data splitting there is not a unique code but depends highly on how perform the division and also where the output variable is placed in the dataframe.

In the code used in this thesis, the desired variable is always put at the last columns of *df*, and the division is made as follows:

```
df_train, df_test = scaled[train], scaled[test]
input_train, output_train = df_train[:,:-1], df_train[:,-1]
input_test, output_test = df_test[:,:-1], df_test[:,-1]
x_train, y_train = input_train, output_train
x_test, y_test = input_test, output_test
```

train and *test* are 1D-array containing instructions for proper separation.
This procedure is only valid if the network is feedforward, if it is recurrent the last two lines of the code must be changed:

```
x_train, y_train = supervised(input_train, output_train, time_steps)
x_test, y_test = supervised(input_test, output_test, time_steps)
```

Where the *time_steps* is a value chosen according to the application and *supervised* is a function that correctly creates the three-dimensional matrix for the correct operation of the LSTM network.

```
import numpy as np
#
def supervised(x, y, time_steps):
    x, y = [], []
    for i in range(len(x) - time_steps):
        v = x[i:(i + time_steps)]
        x.append(v)
        y.append(y[i + time_steps])
    return np.array(x), np.array(y)
```

The libraries **numpy** is always called up at the beginning of each script because it allows various mathematical operations to be carried out and to work with arrays and matrices.

3.2.1 Keras

It's a *TensorFlow* library, designed specifically for deep neural networks. It contains a wide range of functions and the possibility of implementing recurrent and convolutional networks as well.
First of all, the library must be loaded into the program.

```
from tensorflow import keras
```

Then via the code `model = keras.Sequential()` which allows the network to be constructed by stacked one layer after another [15].
The first is the input one:

```
# for feedforward
model.add(keras.Input(shape=x_train.shape[1]))
# for recurrent
model.add(keras.Input(shape=x_train.shape[1],x_train.shape[2]))
```

After that all the hidden layers are defined. Normally if a recurrent network is used, only the first hidden layer is defined differently.

```
# for feedforward and LSTM second hidden layer forward
model.add(keras.layers.Dense(units, activation))
# for LSTM
model.add(keras.layers.LSTM(units, activation))
```

For each layer the number of neurons is defined as well as for the activation function. Eventually there is the output layer that in this thesis is set always as 1 neuron with linear activation. Thanks to the line, `model.summary()` is easy to display the constructed model in a full way.
After that the loss function and the optimizer must be chosen, for instance:

```
model.compile(loss='mse', optimizer='Adam')
```

The optimizer could be adapted by changing the value of learning rate, or the exponential decay for first or second moment.
Finally, the model can be trained.

```
model.fit(x_train, y_train, batch_size=batchsize, epochs=epoch, verbose=2,
                                        callbacks=my_callbacks,
                                    validation_data=(x_val,y_val))
```

Where the number of batchsize and epochs are set by the user, verbose print the training process and callbacks allows different actions to be used at different training stages.
Using a batchsize equal to 1 the gradient descend is performed, otherwise equal to the number of samples present is the stochastic gradient descend or with a number in the range between the two is the mini-batch GD. Two callbacks have always been used, which are:

- *EarlyStopping*: stop training when a specific metric stops to improve after a number of epochs. It also allows the best network parameters to be restored to avoid normal fluctuations.
- *TerminateOnNaN*: simply terminates training when a *NaN* loss is found.

To make prediction with the model trained:

```
y_pred = model.predict(x_test)
```

Once the model is well trained it can be saved in a specific file and load in a second moment, in other file without re-training the model.

```
model.save("model.h5")
# in another program
model = keras.models.load_model('model.h5')
```

3.3 Scores

All the different parameters used for scoring are now described [22].

Hourly error

$$e_h = P_{m,h} - P_{p,h} \tag{3.2}$$

The difference between the measured and the predicted power at the hth hour.

Absolute hourly error

$$e_{h,abs} = |e_h| \tag{3.3}$$

The absolute of the hourly error.

Percentage hourly error

$$e_{\%,m} = 100 \cdot \frac{|e_h|}{P_{m,h}} \tag{3.4}$$

The absolute hourly error divided by the hourly measured power.

Mean square error

$$MSE = \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n} \tag{3.5}$$

The mean squared distance between the observed (y_i) and the estimated ($\hat{y}_i$) values.

Root mean squared error

$$RMSE = \sqrt{MSE} \tag{3.6}$$

The root of MSE in order to have same dimension of power measured.

Normalized root mean square error

$$nRMSE_{\%} = 100 \cdot \frac{RMSE}{max(P_{m,h})} \tag{3.7}$$

The normalization of RMSE respect to the maximum value of the hourly measured power in the time interval, expressed in percentage.

Coefficient of determination

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y_i})^2} \tag{3.8}$$

A coefficient useful to estimate the quality of the curve obtained compared to the measured one. It's dimensionless and goes from 0 (wrong model) to 1 (perfect model).

Coefficient of determination adjusted

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{3.9}$$

The adjusted value of Coefficient of determination, taking into account the number of variables (p) and samples (n). The R^2_{adj} could be negative but always smaller than R^2.

3.3.1 Scikit-Learn

It's a machine learning library designed for classification, regression and clustering problems (Random Forest, Gradient Boosting, Decision Tree and many others). It also makes it possible to evaluate the performance of models used and in this thesis it will be used mainly for this feature [13].
Indeed it's easy to calculated some of the scores described above.

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

3.4 Plot

Each figure created was carried out with *Matplotlib* library [14].

3.4.1 Matplotlib

The command lines required to call up the library.

```
import matplotlib.pyplot as plt
# to manage the axis as datetime
import matplotlib.dates as mdates
```

Subsequently, the library offers many possibilities for obtaining different graphics and customising them.

Part II

Case Studies

Chapter 4

Photovoltaic

Photovoltaic technology (PV) is the direct conversion of photon's energy deriving from sunlight into electric power, and nowadays it's one of the most exploited renewable energy sources. The main disadvantages of this technology are the uncertainty and discontinuity of production, as well as for most of the RES (Renewable Energy Sources). The most disruptive element is cloudiness, which leads to a drop in power production, partly due to a drop in solar radiation and partly due to the operation of the solar panels [37]. Given such problems, a PV plant may be not able to produce electricity continuously over the entire daytime, and this limits the capacity factor (the energy produced in a year divided by the theoretical energy at full rated power continuously over time). Moreover, the production curve follows the solar radiation curve closely.

Figure 4.1: Example of a Photovoltaic Plant located in Boulder City, Nevada[1]

[1] https://www.imeche.org/news/news-article/top-10-solar-photovoltaic-plants-in-the-world

4.1 Solar radiation

The Sun gets its energy from a nuclear fusion process, where hydrogen is converted into helium, thanks to a reaction called *proton-proton chain*. This reaction releases an incredible amount of energy[2] that makes life on this planet possible by providing energy to every living being. The energy produced leaves the Sun as a radiative component, while a fraction is in matter form (solar wind). This electromagnetic radiation is released as if the Sun is considered as a black body at a temperature of $5777K$. The most important part of the solar spectrum is visible light. Once the radiation reaches the Earth's surface several processes take place reducing its actual intensity and also changing the spectrum that reaches the ground [46].

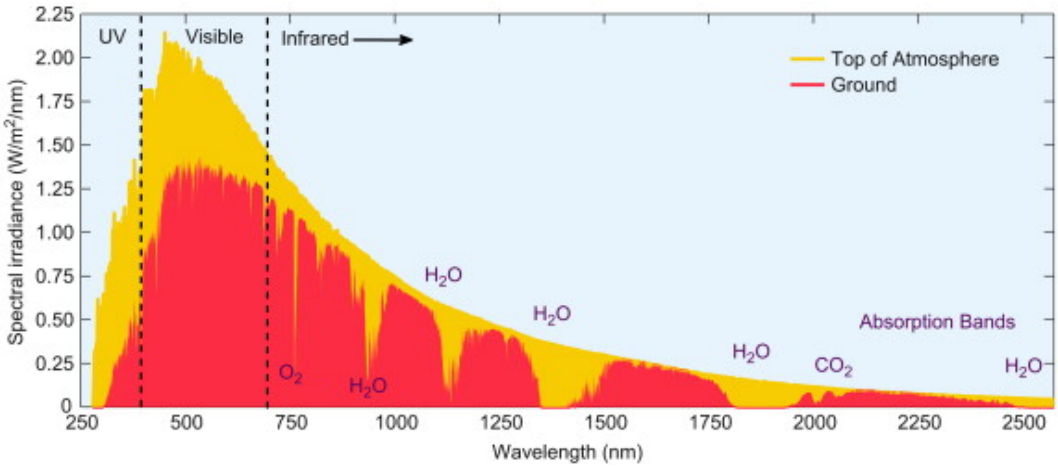


Figure 4.2: Solar spectrum [43]

Figure 4.2 shows in yellow the solar spectrum before reaching the atmosphere and in red the spectrum measured on the ground. The main mechanisms that modify this curve are due to:

- **Absorption**: when a photon hits a particle, the collision results in an energy transfer from the photon and a change in particle internal energy occurs. This process also occurs directly in the sun by forming the so-called Fraunhofer lines in the Sun's spectrum (i.e. black lines in specific wavelengths) due to the Sun's chemical composition. The molecules that absorb radiation in the Earth's atmosphere are shown in figure in purple; they are mainly *Ozone* (0_3), *Water vapour* and *Carbon dioxide*. The portion of incoming energy absorbed is about 23% [44].

[2] A radiation emission equal to $3.85 \cdot 10^{26} W$.

- **Scattering**: consists of a process where radiation is absorbed and then emitted in another direction by specific molecules [45]. A portion of the light scattered is diffused in the atmosphere, however a portion is reflected out the atmosphere and consequently lost in space. About 23% of the light coming from Sun is reflected [44]. There are two types of scattering:
  - *Rayleight scattering*: consists of the scattering of radiation by hitting particles that are smaller than its wavelength. It is the process responsible for the blue colour of the sky, since the blue light is the one with the shortest wavelength. And also because of the reddish colour of the Sun on the horizon, as the light has to pass through a larger portion of the atmosphere being more scattered, leaving red as the main colour of the Sun.
  - *Mie scattering*: occurs instead when the particles are bigger than the wavelength of the radiation. Unlike Rayleigh, which scatters light in all directions, Mie component has a predominant one. It is responsible for the white colour of clouds, as they are composed of water droplets that cause this process to occur.

Thus, the energy available at the surface depends on the composition of the atmosphere, but also on the amount of atmosphere it has to pass through (called Air Mass). In addition, there is a dependence on the geographical location, the astronomical season, the time of day, and the clear sky index. The latter is influenced by cloudiness in the first place; clouds having the most important effect in modifying the amount of light available at the ground level, due to all the processes above mentioned. Other aspects that contribute to radiative energy loss is the presence of suspended particles (like aerosol) and humidity.

Irradiance is defined as the amount of Sun radiative energy flux incident on a surface per unit area of surface, thus measured as $^{W}/_{m^2}$. This parameter is formed by three components that are beam (direct) radiation (G_b), diffuse radiation (G_d) and reflected radiation (G_r). The total radiation reached the surface is the sum of these three terms:

$$G = G_b + G_d + G_r \tag{4.1}$$

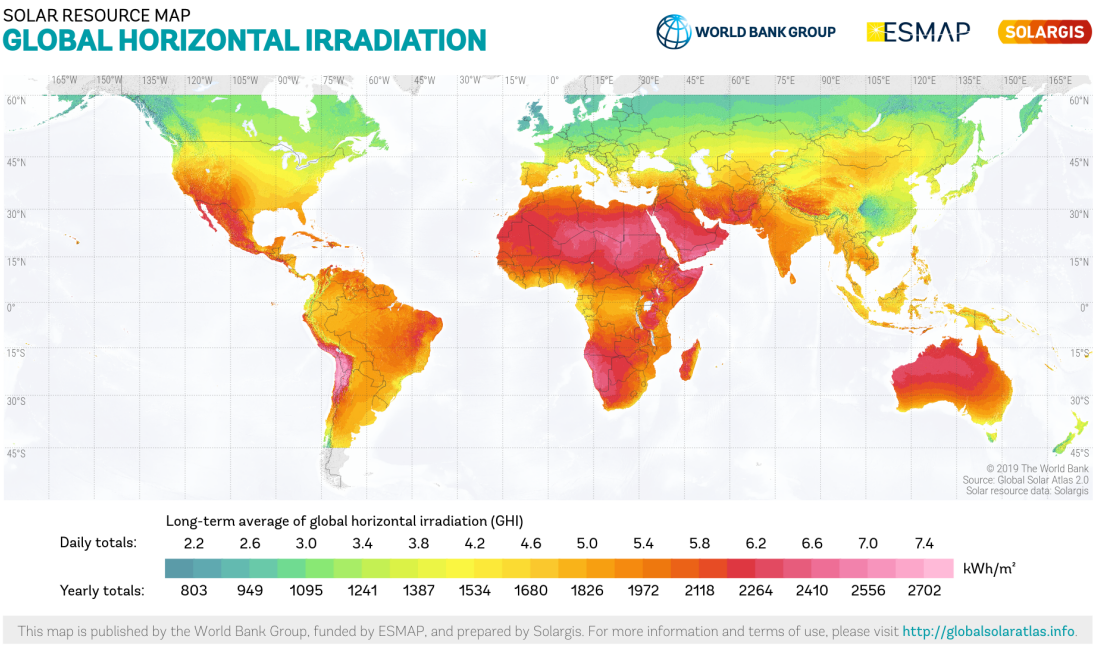


Figure 4.3: Global solar irradiation

Figure 4.3 shows the global irradiation[3] available on the earth's surface.
Another aspect to consider is the difference between the angle of the incident direct radiation and the normal of the interesting surface because it involves losses proportional to the cosine of the angle hence limiting the amount of energy reaching the surface. To overcome this issue, the surface should be kept the more perpendicular as possible with respect to sunbeams so that this angle can be pushed towards near zero values.

[3]It is the incident energy per unit area of surface.

4.2 Operation

The Photovoltaic effect is the physical phenomenon regulating how a solar cell works. When a photon, with a proper amount of quantum energy (expressed in elettronvolts, eV), hits the semi-conducting material through which the cell is structured, electrons are separated from their atoms moving from the valence band to the conduction band. This creates an electric potential due to shifting of charges, and collecting the electrons and producing an electric current.
Due to the discrete phenomena of Photovoltaic effect, a solar cell is not able to use the whole solar spectrum, as the infrared light does not have enough energy to overcome semi-conductor energy band. Some improvements consist of creating stratified cells with the addition of layers that is called Multi-junction Solar Cell. Through these two additional layers the cell is able to recover the infrared energy otherwise lost and increase the fraction of the exploited ultraviolet light for the current production [42].
All three types of solar radiation described above create the Photovoltaic effect. Most of it is produced by direct radiation, but also both diffuse and reflected radiations can also bring an important contribution. The former is prevalent in cloudy conditions, while the latter contribution is called Albedo, which depends on the reflective characteristics of the environment surrounding the solar cell. On the other hand, not all of the photons that reach the solar cell are absorbed; some are reflected by all of its layers.
The value of the current produced depends directly on the value of solar radiation that reaches the solar cell, in fact it can be seen as a current generator directly proportional to the irradiance.

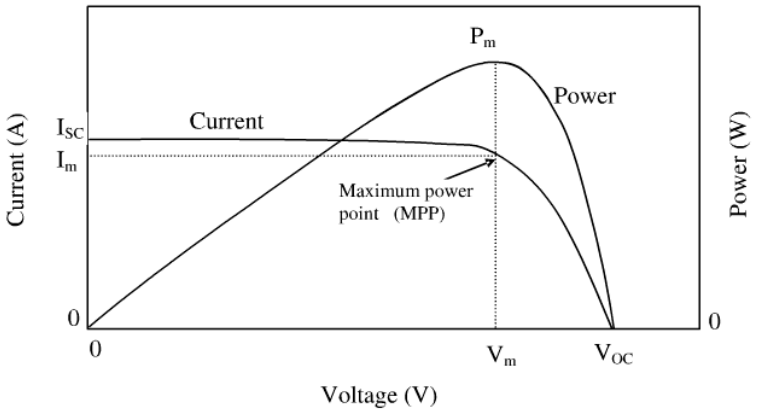
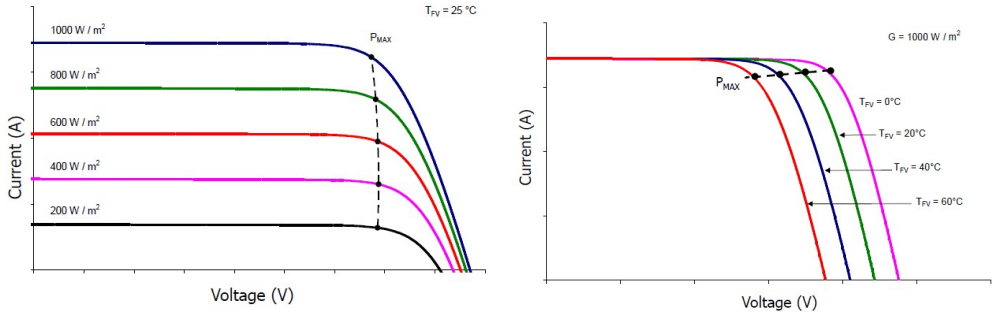


Figure 4.4: P-V characteristics of a solar cell [39]

A PV system consists of many cells connected forming a module, this is set in series with other module in order to increase the voltage creating a string. Finally the strings are placed in parallel to increase the current.

4.3 PV panel behaviour

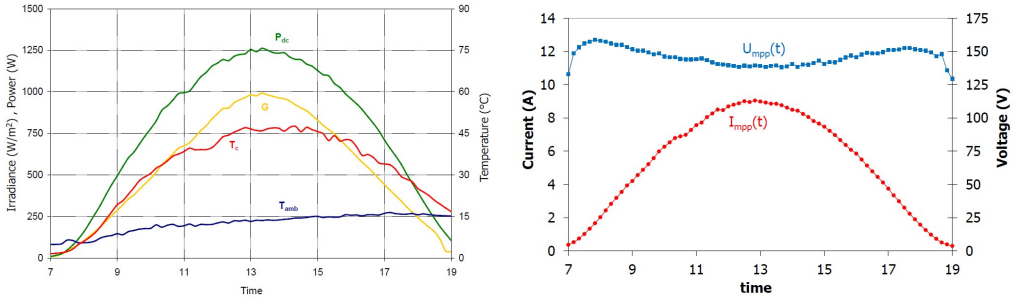
The electrical power produced by a PV is associated to the irradiance and to the air temperature (that leads to modify the module temperature), but also to the electrical load connected to the plant. In a PV system a MPTT (Maximum Power Point Tracker) is installed in order to find the point in the P-V characteristic where current and voltage allow to produce the maximum possible power at the given environmental conditions.



(a) Dependence on irradiance (b) Dependence on module temperature

Figure 4.5: PV power varying irradiance and module temperature [41]

Figure 4.5a shows that as the irradiance decreases, the value of the current produced decreases proportionally while the voltage remains almost constant. On the other hand, as the temperature of the module increases, the voltage decreases significantly and consequently also the power produced decreases.



(a) Trend of main parameters during the day (b) Trend of voltage and current during the day

Figure 4.6: PV daily trend [41]

Figure 4.6b summarises the trend of the main features of a photovoltaic. The dependence of the power produced on irradiance is clearly visible, as is the module temperature with irradiance and air temperature. As the power is evaluated given the instantaneous voltage and current values, it can be noticed how current is mostly affected by the irradiance and the voltage by the module temperature. During the day, as explained above, the voltage and current produced change as shown in figure 4.6a.

4.4 Forecasting

Physical models based on relatively simple mathematical equations can be used for photovoltaic parameters estimation, thanks to [41] one equation available to calculated the output power of a PV panel is:

$$P_m(G, T_c) = P_M(STC) \cdot G(1 + \lambda_{pm} \cdot \Delta T_c) \tag{4.2}$$

Where:

- ΔT_c is the difference between the module temperature and $25°C$.
- P_M is the nominal power of the panel in standard test condition (i.e.: with cell temperature of $25°C$, an irradiance of 1000 W/m^2 with an air mass 1.5).
- G is the vale of irradiance (that in this formula need to be in kW).
- λ_{pm} is the thermal coefficient of maximum power.

In order to have as variables only meteorological ones the temperature of collector could be estimated as follow:

$$T_c = T_a + \frac{G}{U_0} \tag{4.3}$$

Where U_0 is a parameter estimated.

Consequently, using physical models is not so simple because module temperature, shadowing and electrical parameters have to be taken into account in addition to irradiance. That's why most of them strictly depends on the operative instantaneous conditions of the photovoltaic and both climatic and environmental characteristics of the site into which it is installed (i.e. shadowing by external objects), which cannot be predicted a-priori but require further customized evaluations.
However PV can also be predicted by statistical or machine learning methods. There are various statistical methods such as linear regressions, support vector machines, auto-regressive moving averages, regression trees and others that will not be dealt with in this thesis. In any case, even for these methods it is essential to have a proper quantity of historical data such as solar radiation measurements regarding the three components previously described. For example, direct and reflected irradiance measurements are essential to capture cloudiness variation over a certain period. Other parameters which could likely be monitored are operative conditions (current and voltage) and external temperature, so that machine learning-based methods can be trained. In fact, with weather forecast it will be possible to accurately predict the power output of a panel for both short and long periods of time [22].

4.5 Case study

An application case of photovoltaic electrical power forecasting using data-driven models with neural networks is now described. The data was obtained through `https://www.kaggle.com/anikannal/solar-power-generation-data`. Two solar plants are available and only the first has been selected to develop following analysis.

4.5.1 Data processing

From the website two excel files are available, one for the generation data and for the weather sensor data. Both sets were recorded with a 15 minutes resolutions with relative instrumentation.
In the first there are 68779 rows while in the second only 3183. The huge rows number in the first dataset is due to the fact that in the plant there are 22 inverters and each of them is store in the file. The features of the first dataset are:

```
DATE_TIME, PLANT_ID, SOURCE_KEY, DC_POWER, AC_POWER, DAILY_YIELD,
                                        TOTAL_YIELD
```

while in the second one:

```
DATE_TIME, PLANT_ID, SOURCE_KEY, AMBIENT_TEMPERATURE, MODULE_TEMPERATURE,
                                        IRRADIATION
```

Therefore a procedure has been developed with the aim to group together these inverters, summing them up and then combining this data with that of the other file downloaded from the website. Then variables to be saved next are `DATE_TIME, AC_POWER` from the first dataset, while for the second one all except `PLANT_ID, SOURCE_KEY`.

The dataset created is composed by 3158 rows and represents a period from 15-05-2020 to 17-06-2020. The variables available are:

- *AMBIENT_TEMPERATURE* $[°C]$: the air temperature measured nearby the plant.
- *MODULE_TEMPERATURE* $[°C]$: the temperature of the module of PV.
- *IRRADIATION* $[W/m^2]$: the value of Irradiance measured nearby the plant.
- *AC_POWER* $[MW]$: the output power of the plant after the electronic devices that convert the DC power in AC power.

Next step consists in fill the empty or *NaN (Not A Number)* values. Using the command `df.isna().sum()` is possible to count how *NaN* values are present and in which columns. Initially the sum is zero, but with the command `df = df.asfreq('15min')` 106 new *NaN* elements are discovered, meaning that some rows were missing. Thus a interpolation is needed to fill this not a number, more precisely backward fill interpolation was used, since the amount of consecutive missing data was considerable. The variable *MODULE_TEMPERATURE* is the only discarded because cannot be obtained through weather forecasts.

In order to obtain a link to the time of the measurements, one variable [*Hour*] is created and added to the dataset representing the hour of the day, useful for the daily pattern of the curve. After the preliminary editing of the dataset, it is useful to visualize if there is linear correlation between the variables thanks to the Pearson correlation coefficient. Figure 4.7 shows that Irradiation (*Irr*) is extremely correlated to the output power, as expected. Also the air temperature (*T_air*) is quite correlated. Instead is very low the correlation for the variable added, but it has proven to be useful for the model to have better predictions.

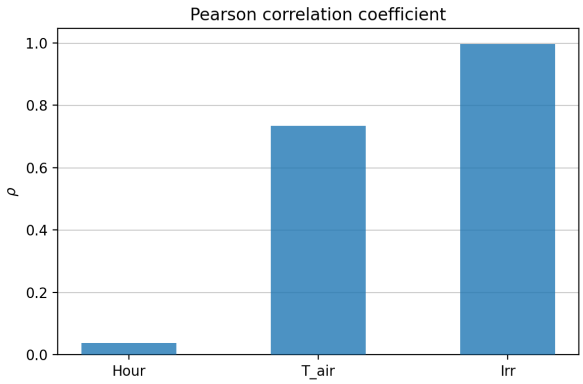


Figure 4.7: Pearson correlation coefficient respect PV Power

After that the composition of the dataset has been selected, it is useful to re-scale all the values according to a specific feature scaling. In this case the *Normalization Scaler* is used, so each value range goes from 0 to 1 and this is advantageous for the neural network, especially since all variables thus have the same range. It's important then after the training process, to compute the inverse scale transformation to return the real value of the features.
Next, for the processing of the data, it's necessary to split into training, validation and test datasets. The numbers of days available are 34, therefore not having many days the algorithm used to test the model is the K-Fold Cross Classification, a scheme is present in figure 4.8.

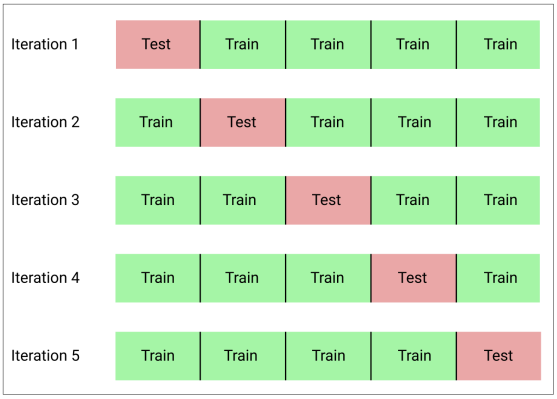


Figure 4.8: K-Fold Cross Validation scheme [40]

In practice the entire dataset has been divided into 5 blocks, where each block is almost one week long. After that, through a for cycle, the model on the first block with the training executed on the others three comes tested and so on until arriving to the last iteration. The validation part is selected to have the same length of the test part and at the end of the training dataset.
All possible scores were then calculated by averaging the results obtained on the 5 different iterations performed.

The variables `x_train`, `y_train`, `x_test` and `y_test` are created in the for loop and will be later used in the artificial neural network. The `x` stands for the input features while the `y` for the desired output variable (that is the power of the plant produced).

The total number of samples for each feature is then 3264, while for the test/validation part there are 653 samples and for the training part 1959.

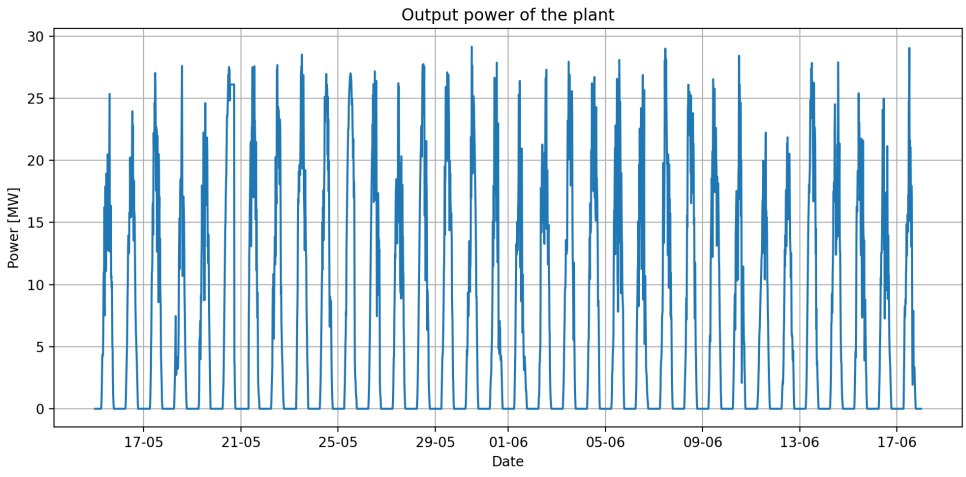


Figure 4.9: Total output power of the plant

Figure 4.9 shows the output variable to be predicted over the entire available period.

4.5.2 Layout description

An analysis is now performed to find the best architecture for ANN. Given the randomness of the results, 10 run were performed with the same hyperparameters and the mean and the variance of the results are collected. The different steps followed during this phase and the order in which they were carried out are:

1. Number of hidden layers.
2. Number of neurons in each layers.
3. Value of learning rate.
4. Batchsize.
5. Activation function.

A rigorous test would be using a grid method by making changes to all the selected parameters but that would mean hours and hours of loop cycles on the computer, hence increasing computational requirements of the procedure. Therefore, it was decided to find the best parameter for each step, leaving the others unchanged except for those that had already been improved previously. With this procedure, computation times are drastically reduced but any correlations between all hyperparameters are neglected.

Having 34 days available, a splitting strategy regarding training, validation and test set has been chosen. In particular, the split is 70% for training, 20% for validation and the last part for test. The adopted subdivision can be seen in Figure 4.10. The number of epochs was not determined a-priori, but through `EarlyStopping` method as soon as after 50 epochs the validation loss does not improve the training stops and the best weights obtained during training are restored.
The loss function chosen is `MSE`, while the optimizer selected is `Adam` and for the activation part each layer have `ReLU` function.

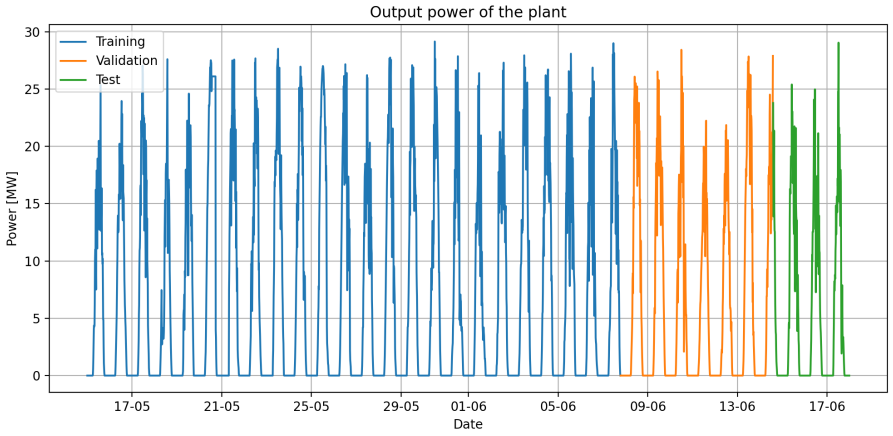
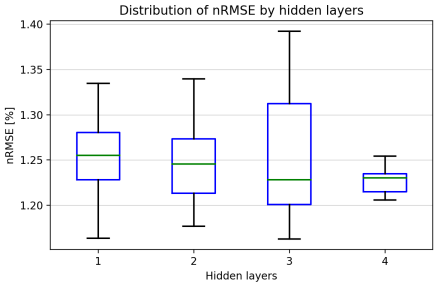


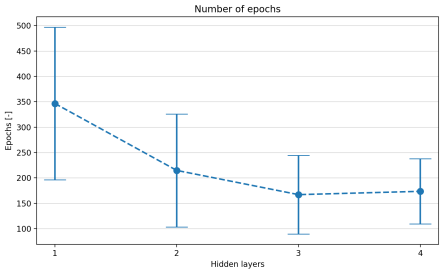
Figure 4.10: Split used for the layout built for PV

Hidden layers

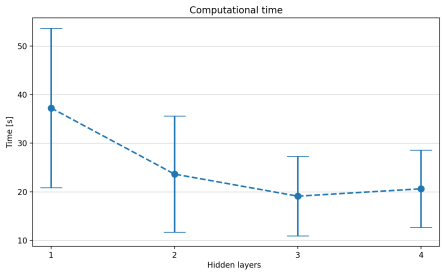
The first point concerns the number of layers in the ANN. The number of layers range from 1 to 4. The value of neurons was kept constant in each run, as well as the value of learning rate and batchsize (left at default values, respectively 10^{-3} and 32). For the first test the number of neuron is set to (8), for the second to (8, 16), for the third to (8, 16, 8) and for the last to (8, 16, 16, 8).



(a) Distribution of results



(b) Number of epochs



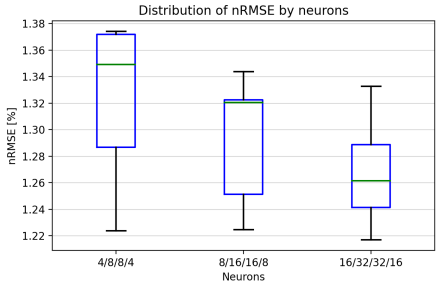
(c) Computational time

Figure 4.11: Results for the number of hidden layers

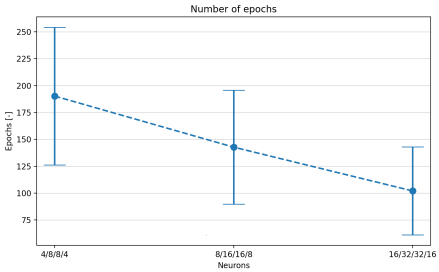
Figures 4.11 show that the best solution concerning the number of hidden layer is 4. The number of epochs gradually decreasing as the model converges more quickly, and consequently also the computational time thus making 4 hidden layers the best solution for the investigated problem.

Neurons

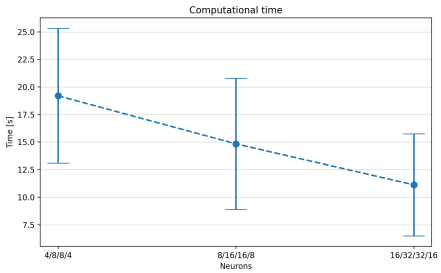
Here the optimal number of neurons for the model is found and then selected. The number of hidden layers is 4, as found in the previous analysis, while all the other parameters remain unchanged. The interval into which the neurons will be varied ranges from 4 to 16 in the first layer, doubling each time, obtaining three possible different configurations of the network. The other layers maintaining the proportion used before.



(a) Distribution of results



(b) Number of epochs



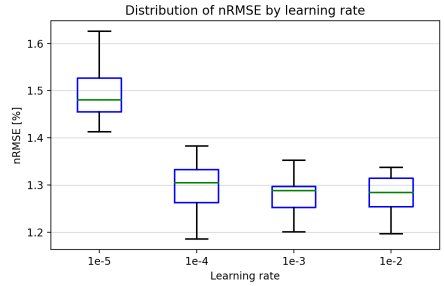
(c) Computational time

Figure 4.12: Results for the number of neurons

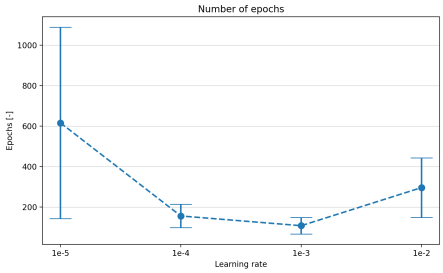
From figures 4.12 the best number of neurons is (16, 32, 32, 16), for both the best score and the lowest computational time because a lower number of epochs through through the training set are needed in order to achieve the best model performance before stopping, given the EarlyStopping algorithm.

Learning rate

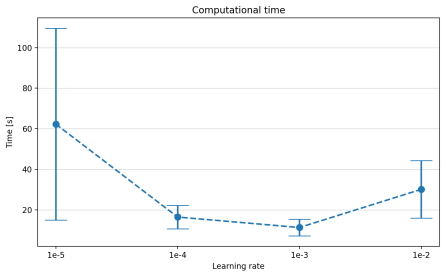
In this section, the learning rate is varied from 10^{-2} to 10^{-5} by finding the best among those selected. As before all other parameters are left unchanged and the number of neurons comes from the previous paragraph.



(a) Distribution of results



(b) Number of epochs



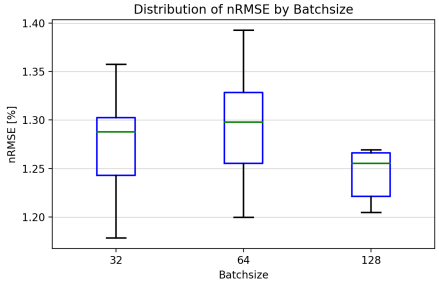
(c) Computational time

Figure 4.13: Results varying the learning rates

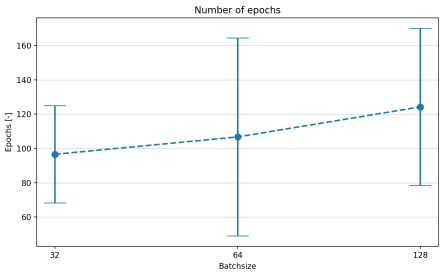
From figures 4.13 the learning rate chosen is $1 \cdot 10^{-3}$, which seems to lead to a better result while also decreasing both epochs and computational times. A lower learning rate brings to longer computational times because the optimization algorithms updates too slowly the model parameters (weights and biases) requiring more epochs for the global minimum search of the loss function. On the other hand, values that are too high also lead to a computational increase because the learning parameters are updated with too emphasis, meaning that the optimal value is likely to be skipped due to larger values of gradients update.

Batchsize

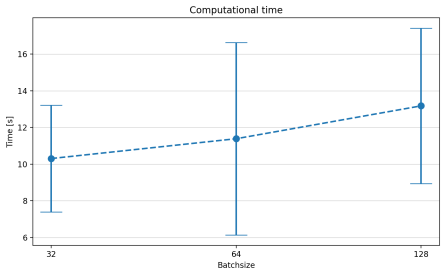
The batchsize is chosen to be varying from 32 to 128 doubling each time.



(a) Distribution of results



(b) Number of epochs



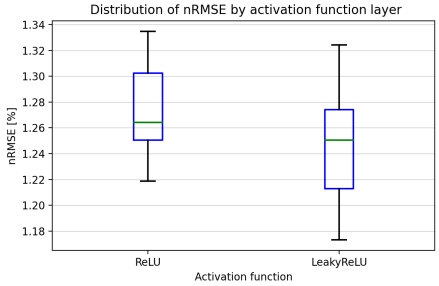
(c) Computational time

Figure 4.14: Results varying the batchsize

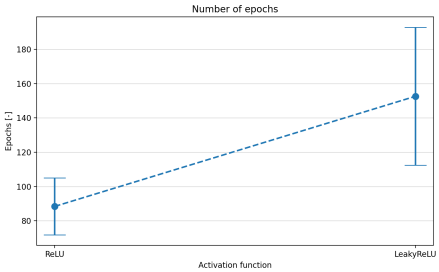
The results seem not to lead big changes in the score, and increasing the batchsize increases the computational time. In any case from figures 4.14 the batchsize chosen is 32. Increasing the batchsize value adds a stochastic component to the training process but with very large datasets the default value is not normally the best choice.

Activation function

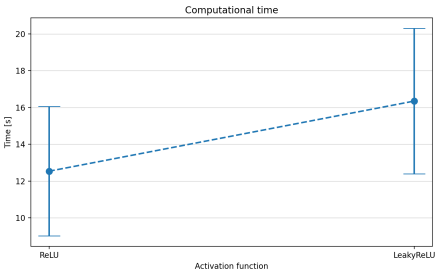
Two test are run, one with `ReLU` and one with `LeakyReLU` activation function.



(a) Distribution of results



(b) Number of epochs



(c) Computational time

Figure 4.15: Results varying the activation function

From figure 4.15 the best activation function seems to be `LeakyReLU`, although it takes a little longer, also because this activation function allows more weights to pass from one layer to another, thus making the process slower.

4.5.3 Final layout

Following results obtained by previous analysis, the best architecture found for this application is thus composed by 4 hidden layers with (16, 32, 32, 16) neurons. Eventually, there is an output layer composed on one single neuron. The total number of parameters that composed the model is 2209 and all the layers are Dense, meaning that neurons in a layer has connections (weight) with each neuron of the next layer. The activation function for hidden layer is `LeakyReLU`. The summary of the network built in python is:

```
model = keras.Sequential()
model.add(keras.Input(shape=x_train.shape[1]))
model.add(keras.layers.Dense(16))
model.add(keras.layers.LeakyReLU())
model.add(keras.layers.Dense(32))
model.add(keras.layers.LeakyReLU())
model.add(keras.layers.Dense(32))
model.add(keras.layers.LeakyReLU())
model.add(keras.layers.Dense(16))
model.add(keras.layers.LeakyReLU())
model.add(keras.layers.Dense(1))
```

After building the model, it has to be compiled choosing the right loss function and optimizer. In this problem, Mean Square Error is chosen as loss function, while Adam is selected to be the training algorithm optimizer. The learning rate of Adam is changed to $1 \cdot 10^{-3}$, while the batch size selected is set to 32. The epochs are 2000 but with a function `EarlyStopping`, as soon as after 100 epochs the validation loss does not improve the training stops and the best weights obtained during training are restored.

```
opt = keras.optimizers.Adam(learning_rate=1e-3)
model.compile(loss='mse', optimizer=opt)
model.summary()
#
my_callbacks = [keras.callbacks.TerminateOnNaN(),
        keras.callbacks.EarlyStopping(monitor='val_loss', patience=100,
        restore_best_weights=True, verbose=1)]
#
batchsize = 32
history = model.fit(x_train, y_train, epochs=2000, batch_size=batchsize,
                    validation_data=(x_val,y_val),
                    callbacks=my_callbacks, verbose=2)
```

4.5.4 Results

After the training process the learning curves of the model, both for training and validation dataset, is created. The mean computational time for training is about 31 seconds for each of the five iteration.

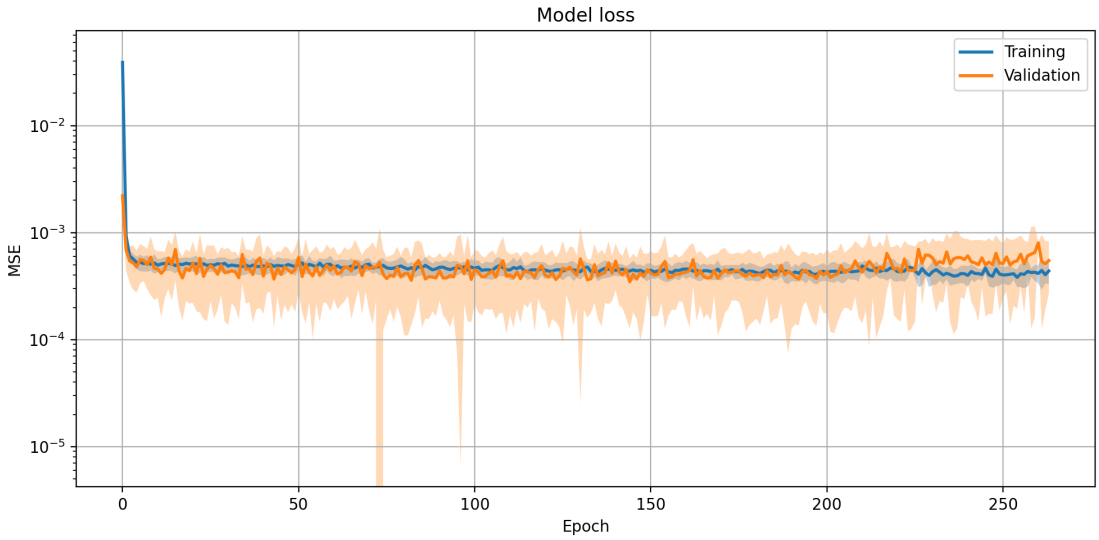
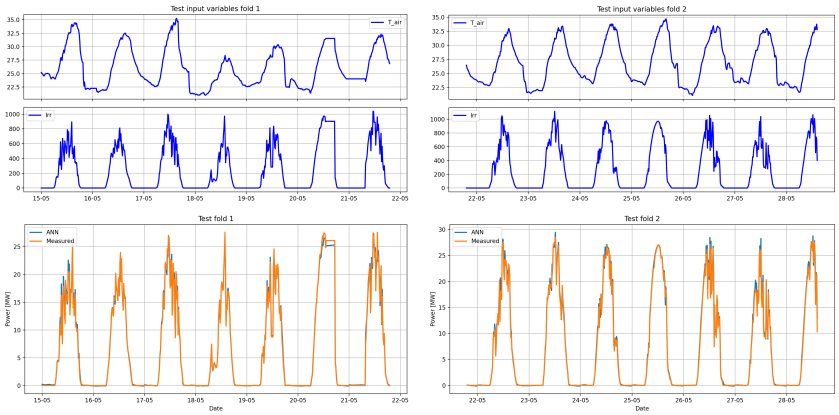


Figure 4.16: Learning curves for PV

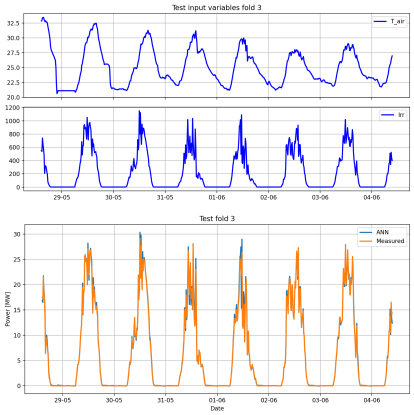
Figure 4.16 shows the mean and the interval of confidence for all the iterations done. In all of the tested data the training and validation curves are almost stacked and that means that the model is properly fitted.

Figure 4.17 show all the part tested, with the inputs used by the model (expect for the *Hour* features that is always equal for all the test) and the power curve of the plant with the measured and predicted (ANN) curves. It can be seen the two curves are much indistinguishable to indicate the fact that the model is accurate. The orange curve is for the power measured while the blue curve is the predicted one by the model (ANN). The similarity between PV electrical power and irradiance curves can also be noticed, remarking the strong correlation between the photovoltaic produced power and the irradiance over its surface.

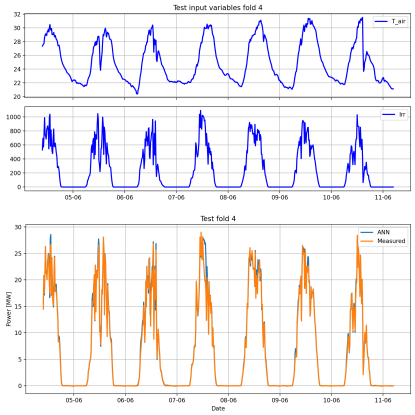


(a) Inputs and test for PV fold number 1

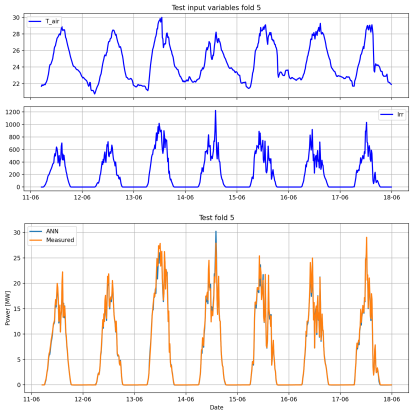
(b) Inputs and test for PV fold number 2



(c) Inputs and test for PV fold number 3



(d) Inputs and test for PV fold number 4



(e) Inputs and test for PV fold number 5

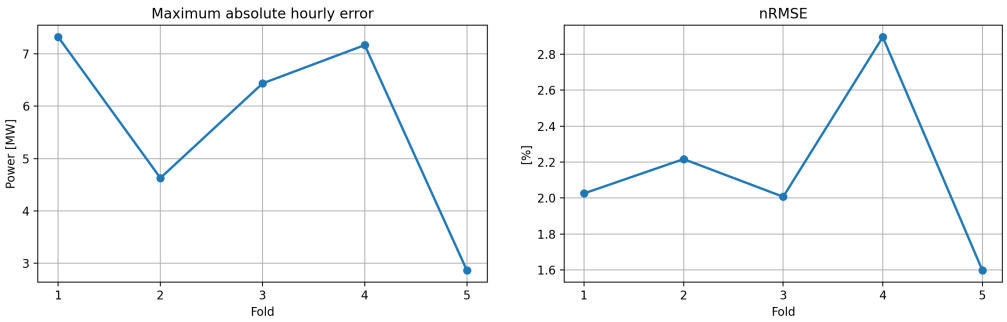
Figure 4.17: Summarize of total tested folds

Later scores values for each fold of the test data are calculated in order to better understand the prediction results. These parameters are summarized in Table 4.1, instead their calculations are explained in section 3.3.

Score	Mean	Deviation
$\max(\lvert e_h \rvert)$ $[MW]$	5.68	1.06
RMSE $[MW]$	0.62	0.12
nRMSE [%]	2.15	0.42
R^2_{adj} [-]	0.995	0.002

Table 4.1: Summarize of PV final scores

Since the problem of predicting photovoltaic can be explained with good accuracy given wheather conditions, as explained before, the score values confirm the good match between predicted and measured PV power. The R^2_{adj} is almost equal 1, while all the other scores are very low and with small dispersion. The values came out very high for the method used of training and also due to the fact that the data were measured and recorded every 15 minutes making it more accurate the power curve.

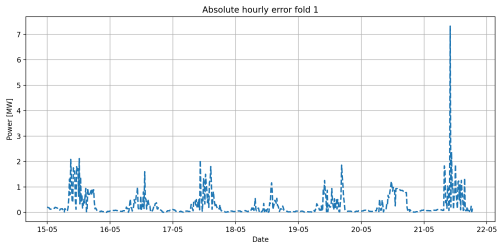


(a) $\max(\lvert e_h \rvert)$ (b) nRMSE

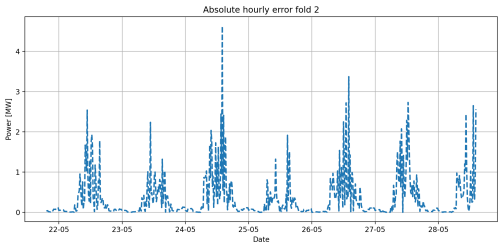
Figure 4.18: Trend of the scores for PV

Instead, figure 4.18 shows the trend during the 5 datasets tested for the maximum absolute hourly error and the normalized root mean squared error. The last dataset results to be the best of all, while the forth resulted to be the worst.

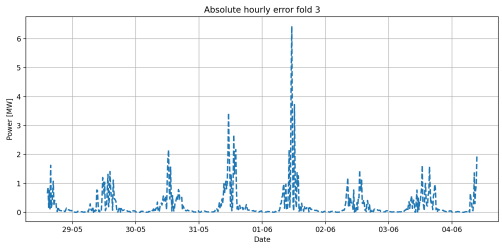
Figure 4.19 shows all the absolute hourly error. The max value of $|e_h|$ is about 7 MW and it occurs during the peak, meaning that in that precise moment the ANN is wrong in its prediction with a maximum error of 30% (being the production of about 25 MW).



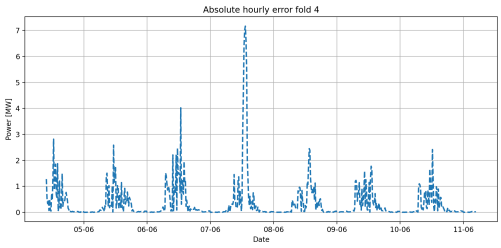
(a) Inputs and test for PV fold number 1



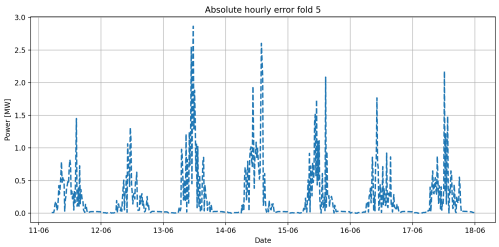
(b) Inputs and test for PV fold number 2



(c) Inputs and test for PV fold number 3



(d) Inputs and test for PV fold number 4



(e) Inputs and test for PV fold number 5

Figure 4.19: Summarize of total folds tested

In conclusion, the built model estimates in a precise way the production of electric energy from a photovoltaic, having as only meteorological inputs the air temperature and the irradiance. The most difficult moments of the day to predict, by comparing the various curves obtained and analyzing the various scores, are during changes in irradiance probably due to clouds passing over the plant.

Chapter 5

Solar thermal energy

A solar thermal collector consists in the direct transformation of radiative energy into thermal energy by absorbing sunlight. From a thermodynamic point of view, therefore, a solar thermal panel behaves like a black body absorbing photon energy. Thermal energy is transferred to a fluid[1] circulating through the panel and it is then used for heating purposes by means of heat exchangers. Using concentric solar collectors, solar radiation capture can be significantly increased in order to produce high enthalpy thermal energy and thus generate electricity through thermodynamic cycles. The problem with concentric technology is that only direct irradiance can be used because only direct rays can be directed and concentrated. On the other hand, common solar thermal collectors are also able to use the other two forms of radiant energy [48]. The applications depends on the temperature of the working fluid, starting from low value (less than $80°C$) for domestic hot water and space heating, mid value (from $80°C$ to $150°C$) for solar cooling and high value (higher than $150°C$) for solar thermal power system [47].

Figure 5.1: Solar thermal plant located in Okotoks, Alberta, Canada[2]

[1]That can be air, water or oil.

5.1 Working principle

The solar collector behaves as a heat exchanger, where the main difference lies in the fact that the energy source is a radiative energy flux. In addition, unlike a solar cell, it is able to absorb the entire solar radiation spectrum, from the ultraviolet to infrared light. The type of collectors is mainly divided into the use of concentric or non-concentric; in the form of flat plate or vacuum tube collectors and eventually in fixed collectors or with tracking systems. All concentration systems are based on solar tracking being their operation based on direct radiation as mentioned above [50].
Another important aspect concerns, as for photovoltaic, the geographical location of the system and the inclination of the collectors determining the incidence of direct light hitting the panels. However, for solar collectors, the air temperature plays a very important role with regard to losses, as conduction and convection losses are involved as a term proportional to the temperature difference between circulating fluid and external one. For this reason, the insulation of the pipes through which the working fluid flows plays a very important role [49].
A system based on solar thermal energy has several other components in addition to the solar collector. These could be a tank to store the thermal energy for later reuse, piping systems and pumps to move the fluid, and auxiliary systems to maintain the desired temperature in the storage tank. In addition, systems can be subdivided according to their operation [49]:

- *Natural circulation*: fluid circulation is obtained by its different density at different temperatures (exploiting the thermosiphon effect), so no electrical consumption by the use of a pumping system is needed. Typically the storage tank is integrated together with the collector for small sizes, but it needs protection against freezing as it is exposed to the open air. Consequently, depending on the geographical location, it may be open circuit (where there is no risk of freezing) or closed circuit (using an anti-freeze solution).

- *Forced circulation*: in this case, however, a pump is used to make the heat transfer fluid circulate and keep the system at a proper pressure. The overall system becomes more complex, but the storage tank can be placed indoors with less heat loss. In addition, the collectors can be positioned in parallel and/or in series, which increases the size considerably.

[2] https://www.okotokstoday.ca/local-news/drake-landing-a-model-for-whitehorse-study-1526567

5.1.1 Flat plate

Solar flat plate collectors are one of the most solar thermal technologies nowadays investigated.

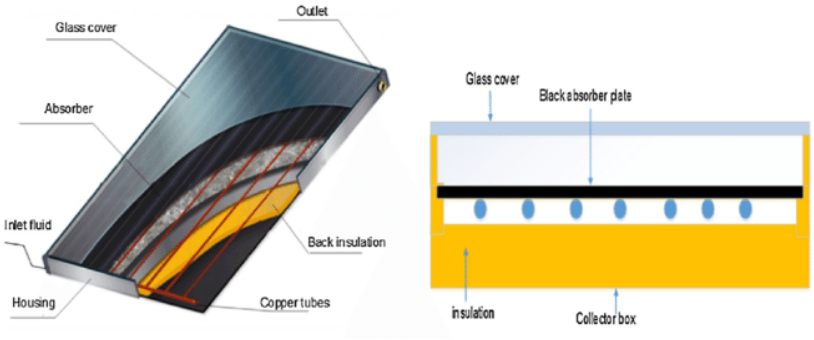


Figure 5.2: Cross section of a Flat Plate collector[3]

From scheme of figure 5.2 the main components of solar collectors are [50]:

- *Absorber plate*: it is a thin layer placed over the pipes and attached to them. Its task is to absorb as much energy as possible and transfer it to the fluid. Consequently, it must have the highest possible absorptivity (in the UV, light and low IR part) and the lowest possible emissivity (in the IR part). It is made of a typically metallic material with high thermal conductivity.
- *Glass cover*: it is a very important element because it must be transparent to light of a small wavelength but opaque to infrared light. This is to keep the radiation emitted (in the form of infrared radiation) by the tubes and the absorber plate inside the collector (creating a greenhouse effect). It also acts as an insulator from the outside environment.
- *Back and side insulation*: it is important to reduce the energy losses from sides and back of collector, so it must have a high thermal resistance.
- *Tubes*: where the working fluid can circulate and therefore must be mechanically connected to the absorber plate and dissipate as little heat as possible.

[3] https://www.researchgate.net/publication/321013607_Numerical_Investigation_and_Experimental_Verification_of_Performance_Enhancement_of_Flat_Plate_Solar_Collector_Using_Nanofluids

The operating temperature of flat plates is usually placed up to $100°C$. They are not mounted on solar tracking systems and the optimal tilt and direction is chosen according to the specific application and geographical location. Typically, a good solution would be to tilt the collector at an angle equal to the geographical latitude. However this depends on several factors and there are specialized software to carry out various simulations taking into account even the surrounding environment and possible shadows [47].

5.1.2 Evacuated tube

Unlike a flat plate collector, the tubes are placed in a vacuum environment. This is done to decrease convection losses that would otherwise occur inside the collector. Having a cylindrical shape (to better handle the stresses given by external pressure) they are more sensible to optical losses. To overcome these problems, tubes with a parabolic reflecting mirror placed underneath are used in order to recover as much energy as possible and a U-tube system (i.e. one tube placed above the other where one is the inlet and the other the outlet) is adopted [47].

5.2 Forecasting

As explained before, the power produced by a solar thermal collector strongly depends on the Sun irradiance, as well as to the sky condition and air temperature. This leads to a seasonal trend of the power produced, where for instance in summer the irradiance is high and therefore also the output thermal power. So a complete model of the physics of the problem can be quite difficult to realise, as there are also thermal transients.
A model based on mathematical equations that therefore relies on physical models uses the heat balance under steady-state conditions for a solar collector, and this is:

$$\dot{Q}_s = \dot{Q_{opt}} + \dot{Q}_l + \dot{Q}_u \tag{5.1}$$

Where $\dot{Q}_s$, is the incident solar power, $\dot{Q_{opt}}$ are the optical losses, $\dot{Q}_l$ are the heat losses and $\dot{Q}_u$ is the useful power [47].
An empirical equation which can be used to predict the thermal output of a solar collector system can be written as:

$$P_{sh} = S \cdot [\eta_0 G - a_1(T_f - T_a) - a_2(T_f - T_a)^2] \tag{5.2}$$

Where:

- η_0 is a coefficient for the yield estimation.
- a_1 and a_2 are coefficients to take into account thermal losses.
- T_f is the mean temperature of the working fluid.
- T_a is the air temperature.
- G is the solar irradiance.

As with PV, statistical and machine learning methods are used to make forecasts for solar collectors. In addition, there is the thermal transient problem of having to work with thermal energy and not electricity. However, as already mentioned, solar thermal panels are also able to use other forms of radiation and are less affected by cloudiness.

5.3 Case study

The data used were provided by `SOLID Solar Energy Systems GmbH, Graz, Austria`. The values are presented in normalized scale being owned by a third party and having signed an *NDA*, we have pledged not to disclose them.

5.3.1 Data processing

Similarly to PV analysis approach, it is first useful to identify which features are included in the working dataset. It is composed by following variables:

```
timestamp, Irradiation, Ambient_Temperature,
heatmeter_value_solar_primary_circuit,
volume_meter_solar_primary_circuit, power_solar_primary_circuit,
volume_flow_solar_primary_circuit,
return_temperature_solar_primary_circuit,
flow_temperature_solar_primary_circuit,
temperature_difference_solar_primary_circuit
```

Therefore the dataset made by 10 columns per 8761 rows. Since the problem is to predict the power generated by a solar collector system, all variables except time, irradiance, ambient temperature and heat output are eliminated. After being renamed then the dataset becomes:

```
datetime, Irr, T_air, Th_pow
```

In order to obtain a link to the time of the measurements, two variables are created and added to the dataset: these are the hour of the day, useful for the daily pattern of the curve, and the number of week of the year, effective to catch seasonal variations of the production.
Wanting to handle weeks as periods, the dataset is shortened to exactly 52 weeks of 168 hours. Then the last 25 hours are eliminated, making the period from 01-01-2018 00:00 until 30-12-2018 23:00. After the preliminary editing of the dataset, it is useful to visualize if there is linear correlation between the variables thanks to the Pearson correlation coefficient.

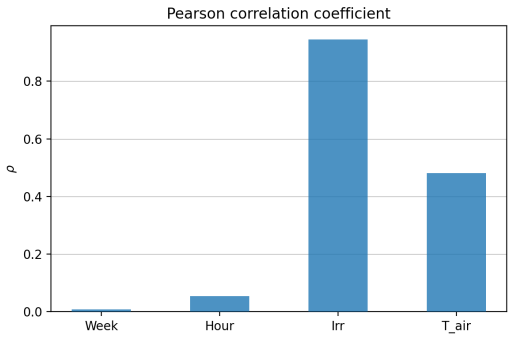


Figure 5.3: Pearson correlation coefficient respect to the thermal power

Figure 5.3 shows that Irradiation (*Irr*) variable is extremely correlated to the Thermal power, as expected, as well as air temperature (*T_air*) is quite correlated and justify previous choice to keep only these features from original dataset. Instead a low the correlation for the variables added can be observed, but it has proven to be useful for the model to have better predictions.

Chosen the composition of the dataset, it is useful to rescale all the values according to a specific feature scaling process. In this case the *Normalization Scaler* is used, so each value range goes from 0 to 1 and this is advantageous for the neural network, especially since all variables thus have the same range and learning algorithm could perform faster. In this specific case, the data will not be rescaled to maintain the agreement signed with the supplier company.

Next, for the processing of the data, it's necessary to split into training, validation and test datasets, as already suggest for PV forecasting model. Starting from the first day available (i.e.: 1 January 2018) the 52 weeks of the year are subdivided using as training 36 weeks, as validation 10 weeks and as test 6 weeks.

So the training dataset has 5712 elements for each feature, the validation 840 and the test 168 for each of the 6 weeks.

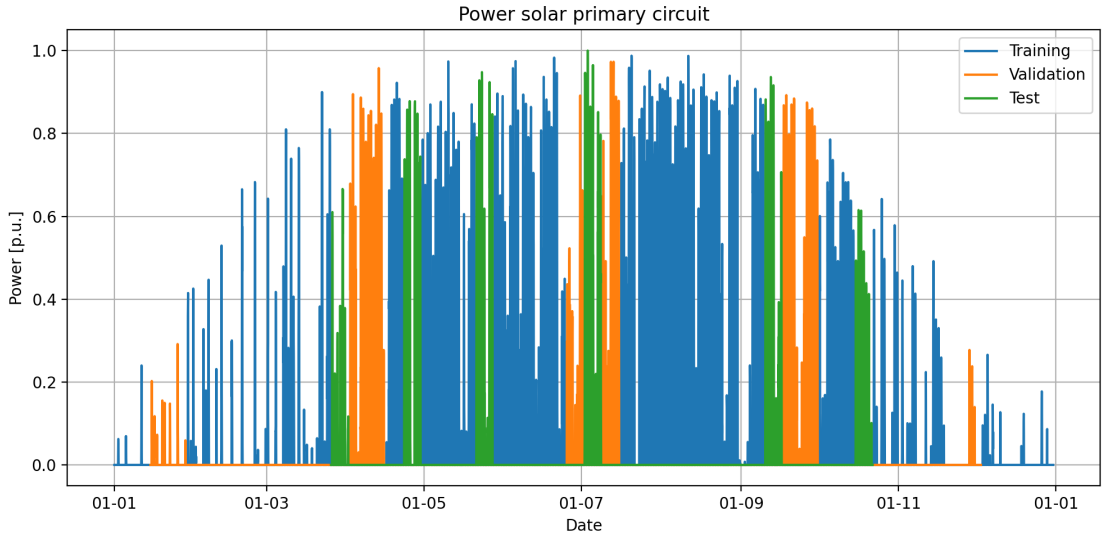


Figure 5.4: Total measured power of the year

Figure 5.4 shows the thermal production of the plant during the year. Furthermore it is also possible to view the division explained above, where the blue curve is for the training, the orange for the validation and the green for the test dataset.

5.3.2 Layout description

As for photovoltaic, an analysis of the best architecture is performed. The different steps followed during this phase and the order in which they were carried out are:

1. Number of hidden layers.
2. Types of network.
3. Number of neurons in each layers.
4. Activation function.

The procedure followed is the same as that explained in the photovoltaic section 4.5.2, but other hyperparameters have been modified in this case. Batchsize and learning rate were eliminated from the trials given that in PV they resulted comparable in terms of nRMSE. Since a significant dependence on time factors is present, the use of a recurrent network can be justified and lead to better results. So a comparison between using the first hidden layer as a feedforward or recurrent one is analyzed.

Having 52 weeks, it has been chosen that the split is 37 weeks for training, 10 weeks for validation and 5 weeks for test. Figure 5.5 shows the training, the validation and the test sets. The number of epochs was not determined a-priori, through the use of `EarlyStopping`, as soon as after 50 epochs the validation loss does not improve significantly the training stops and the best weights obtained during training are restored.
The loss function chosen is `MSE`, while the optimizer selected is `Adam` and for the activation part each layer have `ReLU` function. In the output layer linear activation function has been used. Batchsize is 128 because in this case the number of samples is more consistent with respect to photovoltaic case study, and the number of batchsize can be increased so as not to compromise computational times too much.

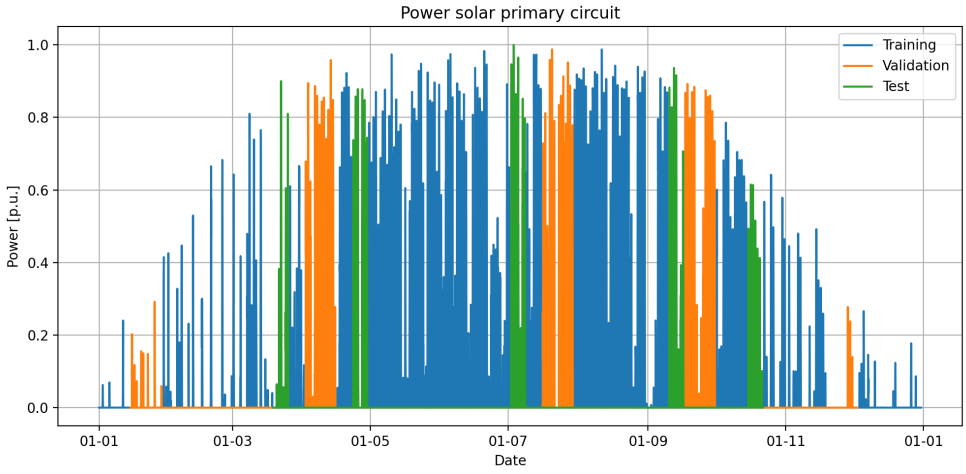
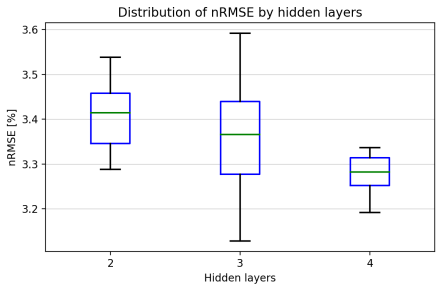


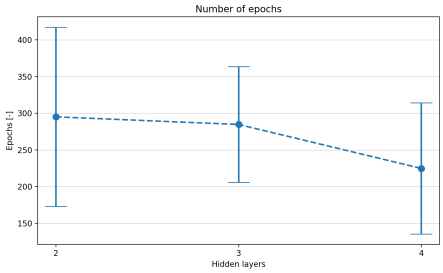
Figure 5.5: Split used for layout architecture

Hidden layers

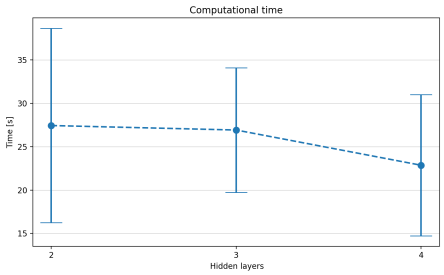
The first point to be addressed is the optimal number of layers in the ANN. The number of layers ranges from 2 to 4. The value of neurons was kept constant in each run, as well as the value of learning rate (left at default values to 10^{-3}). The number of batchsize is increased to 128 from the default value of 32 according to the number of samples in dataset. For the the first test the number of neuron is set to (8, 16), for the second to (8, 16, 8) and for the last to (8, 16, 16, 8).



(a) Distribution of results



(b) Number of epochs



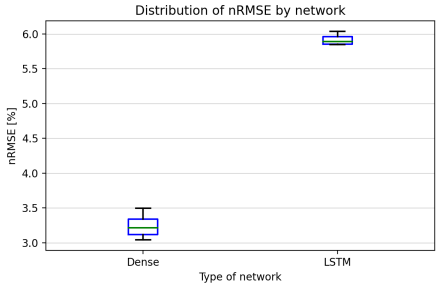
(c) Computational time

Figure 5.6: Results for the number of hidden layers

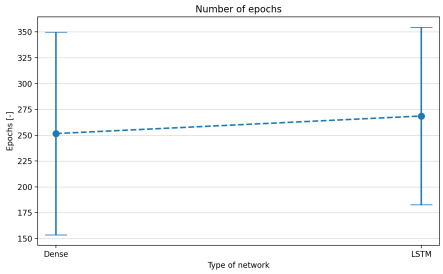
Figures 5.6 show that the best solution concerning the number of hidden layer is 4. The number of epochs gradually decreasing as the model converges more quickly, and taking slightly lower time to reach the convergence.

Type of network

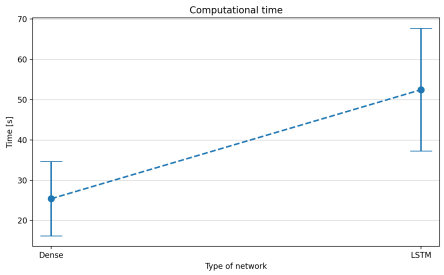
The number of hidden layer is set to 4, as a result of previous analysis, and the other hyperparameters are kept constant. For the recurrent network a LSTM layer is used, with time step set equal to 3. With the notions regarding the operation of the LSTM network unlike the Dense layers the activation function for LSTM is chosen equal to tanh.



(a) Distribution of results



(b) Number of epochs



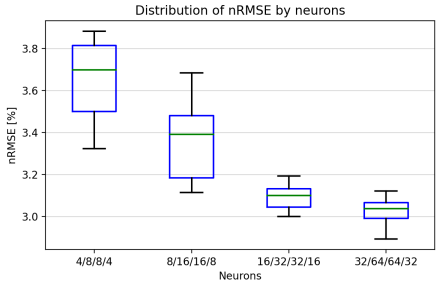
(c) Computational time

Figure 5.7: Results changing type of network

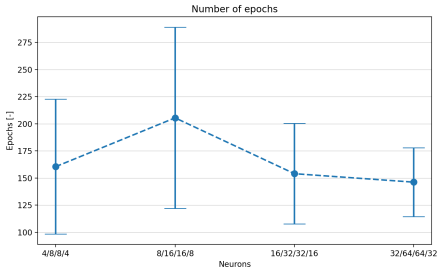
Despite the adoption of LSTM layer, the normal dense layer works better so it remains unchanged. Furthermore, since the LSTM layer involves more parameters and different internal steps, although the number of epochs is almost the same, in the second case the convergence times are almost doubled.

Neurons

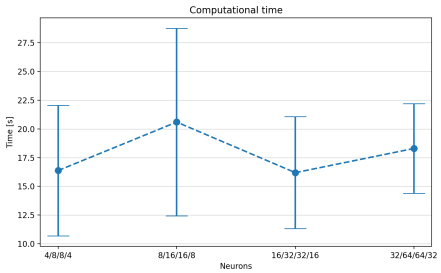
The range for the neurons is set to be varying from 4 to 32 in the first layer, doubling each time in order to use same approach as PV case study, wile the other layers keep the proportion used before. As usual, all the other parameters are unchanged.



(a) Distribution of results



(b) Number of epochs



(c) Computational time

Figure 5.8: Results for the number of neurons

From figures 5.8 the number of neurons that it is decided to select is the (32, 64, 64, 32), which as it can see leads to a better result without compromising computational time.

5.3.3 Final layout

The best architecture found then is composed by 4 hidden layers with (32, 64, 64, 32) neurons. Eventually, there is an output layer composed on one single neuron. The total number of parameters that composed the model is 8545. All the layers are Dense. The activation function for the hidden layer is `ReLU`. The summary of the network built in python is

```
model = keras.Sequential()
model.add(keras.Input(shape=x_train.shape[1]))
model.add(keras.layers.Dense(32, activation='relu'))
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dense(32, activation='relu'))
model.add(keras.layers.Dense(1))
```

After building the model, it has to be compiled choosing the right loss function and optimizer. In this problem Mean Squared Error still is the chosen loss function, while also Adam remain the selected optimizer. The learning rate of Adam is changed to $1 \cdot 10^{-3}$, while the batch size selected is set to 128. The epochs are 2000 but with a function `EarlyStopping`, as soon as after 100 epochs the validation loss does not improve the training stops and the best weights obtained during training are restored.

```
opt = keras.optimizers.Adam(learning_rate=1e-3)
model.compile(loss='mse', optimizer=opt)
model.summary()
#
my_callbacks = [keras.callbacks.TerminateOnNaN(),
        keras.callbacks.EarlyStopping(monitor='val_loss', patience=100,
        restore_best_weights=True, verbose=1)]
#
batchsize = 128
history = model.fit(x_train, y_train, epochs=2000, batch_size=batchsize,
                    validation_data=(x_val,y_val),
                    callbacks=my_callbacks, verbose=2)
```

5.3.4 Results

After the training process the learning curves of the model, both for training and validation dataset, are created to visualize how the loss function evolves during the epochs. The average computational time for training is about 20 seconds.

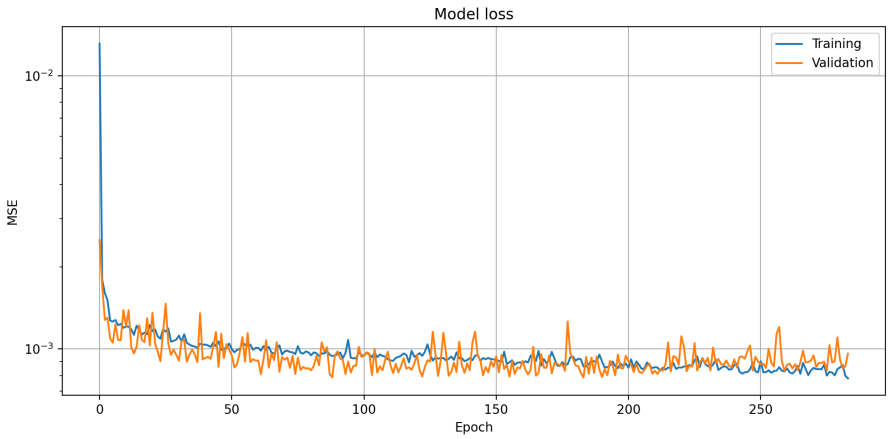


Figure 5.9: Learning curves

Figure 5.9 shows that the validation loss curve follows quite well the training loss one and this means that the model gives a good fitting and a good agreement between model's prediction and measured values.
Later some score values for each week of the test data are calculated in order to better understand the prediction results.

Table 5.1 summarizes the mean and the standard deviation for the scores calculated.

Score	Mean	Deviation
e_p [%]	4.83	3.38
nRMSE [%]	3.62	0.82
R^2_{adj} [-]	0.98	0.01

Table 5.1: Summarize of final scores

The hourly percentage error is only calculated when the measured power exceeds a certain value, otherwise the calculation of this error would have infinite values.
The results show that the neural network is quite able to predict with sufficiently accuracy the output power of the plant, and that the predicted curve is following quite closely the measured one.
Instead, figure 5.10 shows the trend during the 6 weeks tested for the normalized root mean squared error.

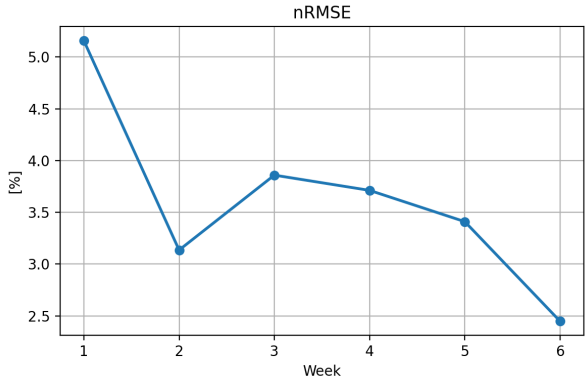
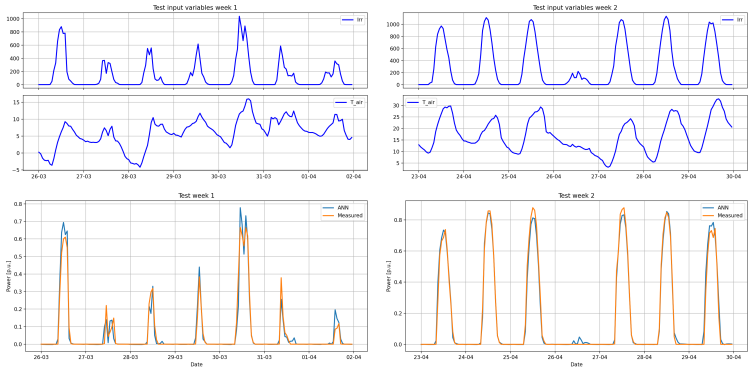


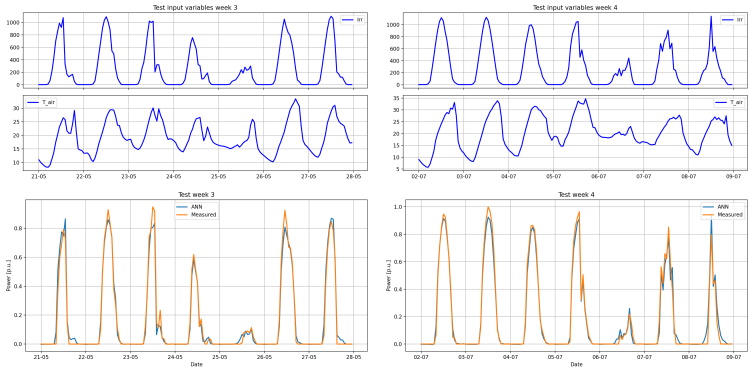
Figure 5.10: nRMSE

The first week is the least accurate, and the model gradually becomes more precise with its predictions.
Graphs with meteorological inputs and measured and predicted power curves for all 6 weeks tested are now presented.



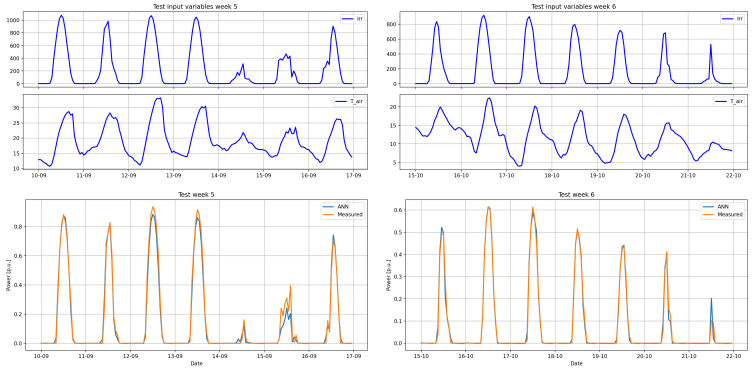
(a) Inputs and test for STE week 1

(b) Inputs and test for STE week 2



(c) Inputs and test for STE week 3

(d) Inputs and test for STE week 4

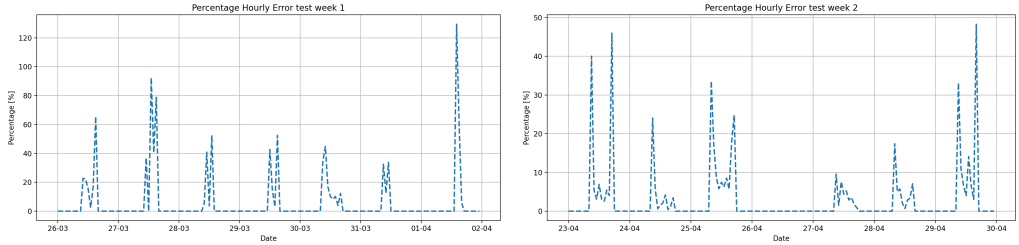


(e) Inputs and test for STE week 5

(f) Inputs and test for STE week 6

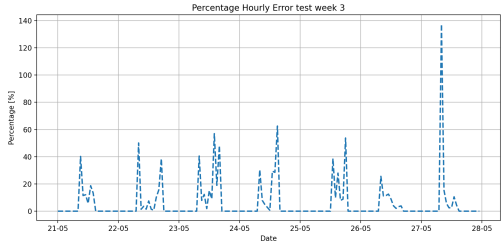
Figure 5.11: Summarize of total tested weeks

Figures 5.11 show the strong dependence of the power produced on the irradiance value and also how well the model estimates this power. To see in detail when the model is less accurate, the graphs of all 6 weeks are now presented for the hourly percentage error.

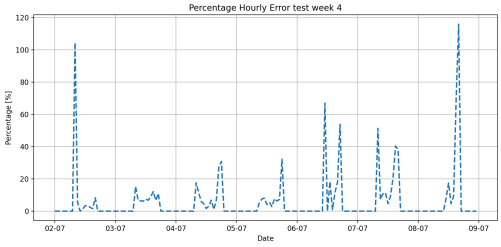


(a) e_p for STE week 1

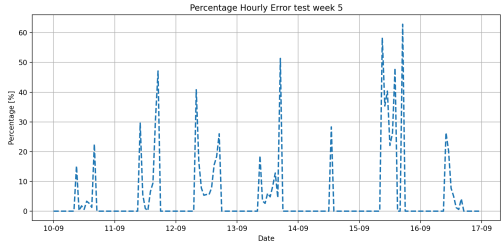
(b) e_p for STE week 2



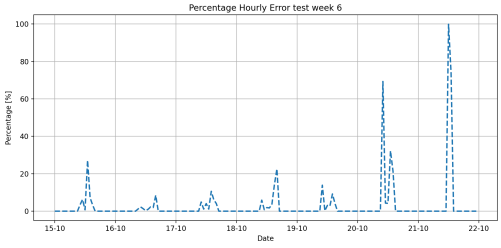
(c) e_p for STE week 3



(d) e_p for STE week 4



(e) e_p for STE week 5



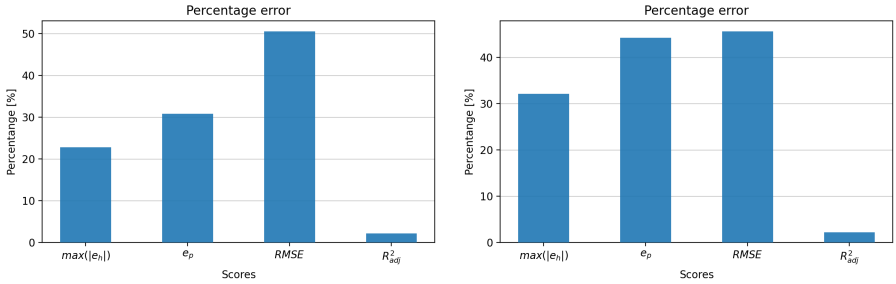
(f) e_p for STE week 6

Figure 5.12: Trend of percentage hourly error

From these graphs one can see a particular shape in which the moments with the greatest error are at dawn and dusk. Apparently, predicting output thermal power at low irradiance values or with rapid irradiance drops (mainly due to clouds) seems to represent a more challenging task for the implemented neural network architecture.

5.3.5 Offset

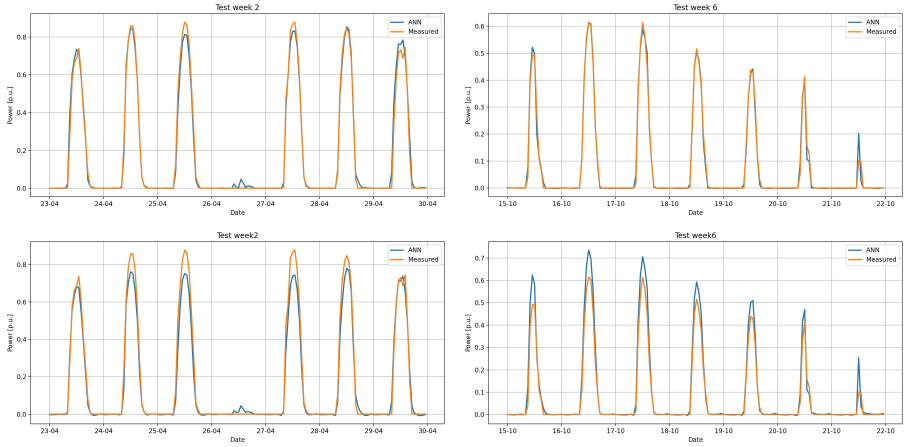
In this section, a study is made to see how the model uses irradiance values for power calculation. The trained model is then saved to a *.h5* file, after which it is called up in another program in which the original dataset has been modified. In the first case this the irradiance is reduced by 10%, in the second is increased by 10%.



(a) Decrease irradiance errors (b) Increase irradiance errors

Figure 5.13: Percentage error for both cases

In figures 5.13 it can be seen how all the parameters calculated as a score have deteriorated in absolute percentage terms. It can be seen that even if the worsening is positive or negative, the scores change differently.



(a) Test week 2 reduction irradiance (b) Test week 6 increasing irradiance

Figure 5.14: Comparison selected weeks both cases

Finally in figures 5.14 two weeks were depicted for the two cases. In figure 5.14a it can be seen the difference between the predicted normal power value and the predicted power with decreased irradiance. It is noticeable the underestimation of power and how the blue curve is always below the orange curve. The opposite case is shown in figure 5.14b where the blue curve overestimates the power produced.

Chapter 6

District heating

District heating consists on the supplying of thermal energy in the form of hot or superheated water or steam to a distributed network of users, typically in urban centres. A key element is the use of cogeneration power plants, through which electricity and heat productions can be combined so that it is possible to satisfy both thermal and electrical loads. In facts, without a district heating network, each individual user would require both electricity and heat from two different sources, as the use of thermal boilers for heating purposes and access to electric national grid for electric. Thus, this latter configuration involves two different processes efficiencies (thermal boiler and electrical national grid generation) and a decentralized production regarding thermal users. The use of this district heating instead involves the centralised production of thermal energy to cover a wide range of distributed users amongst a piping system. The main components are therefore the various sources of production, the distribution network, thermal storage systems, heat exchangers for the users and pumping systems for the heat transfer fluid. There are historically 4 generations of DH [51]:

- *First generation* was based mainly on steam, therefore it had a high flow temperature with large pipes cross sections and very low efficiencies. The main source of heat came from coal. Old New York City still uses steam as a heat transfer fluid.
- *Second generation* was based on pressurised water with a temperature of around 100 degrees Celsius. Pipes diameters are smaller than in the previous case, and there is an increase in efficiency due to fewer losses in the network. In this case, systems based on cogeneration or combined heat and power are used as a thermal energy source.
- *Third generation* still use pressurised water as the heat transfer fluid but with insulated metal pipes which allow lower temperatures to be used, lower thermal losses and resulting in an increased global efficiency.

- *Fourth generation* represents the future of the district heating network and is still mostly under development. It is based on large-scale renewable systems, smart grids, the integration of cooling networks[1] and the need to use lower and lower temperatures to increase efficiency and thus decrease losses. They also allow heat produced by the prosumer[2], making the DH network capable of handling thermal energy in both directions.

A system based on the latter technology consists mainly of production sites, a distribution network and end-users. This leads to major advantages including lower economic cost, better control of pollutants and improved system management. The reliability of the network is very high with many different heat sources and interconnections branching off. There is no risk to people's health as there are no chimneys for each individual house with risk of leakage [59].

6.1 Energy efficiency indicator

As mentioned above Combined Heat and Power (CHP), commonly named cogeneration, is a key element of a district heating network. In order to evaluate the performance of a CHP plant, an index called *Index of Primary Energy Saving* (PES), was introduced by the EU Commission with Directive 2004/08/EC [64] which if greater than 0 leads to the convenience of using cogeneration. This Regulation gives a *High Efficiency Cogeneration* status (HEC) to those plants that meet the following condition [63]:

- $PES > 10\%$ when $P_{el} \geq 1MW_{el}$
- $PES > 0$ when $P_{el} < 1\ MW_{el}$

In fact, if a plant meets these requirements it is entitled to receive economic incentives [61]. The Index is defined as follows:

$$PES = \frac{F^{SP} - F}{F^{SP}} = 1 - \frac{F^{SP}}{\frac{W}{\eta_e^{SP}} + \frac{Q}{\eta_t^{SP}}} \tag{6.1}$$

Through this index it is possible to calculate the energy saving for a cogenerator producing the same quantities of useful energy (electricity W and heat Q) by using the fuel F, with respect to the separate production (SP) requiring F^{SP} of fuel. Where η_e^{SP} is the electrical efficiency of the reference electricity production system and η_t^{SP} is the thermal efficiency of the reference boiler. A similar formulation can also be adopted for trigeneration systems or even polygeneration systems [57].

[1] Called District Heating And Cooling (DHC). This system is based on trigeneration, where in addition to the production of electricity and heating, cooling is also produced. Therefore this configuration leads to high overall efficiency. Cooling can be provided by absorption chillers, that convert the part of the excess heat that anyway would have been wasted into useful energy in the form of chilled energy [60].

[2] i.e. a user that both uses and supplies energy [57].

6.2 Components

In this section, the main components of a district heating network above mentioned are explained.

6.2.1 Production

Thermal energy can be produced by means of different production systems today, as illustrated in figure 6.1.

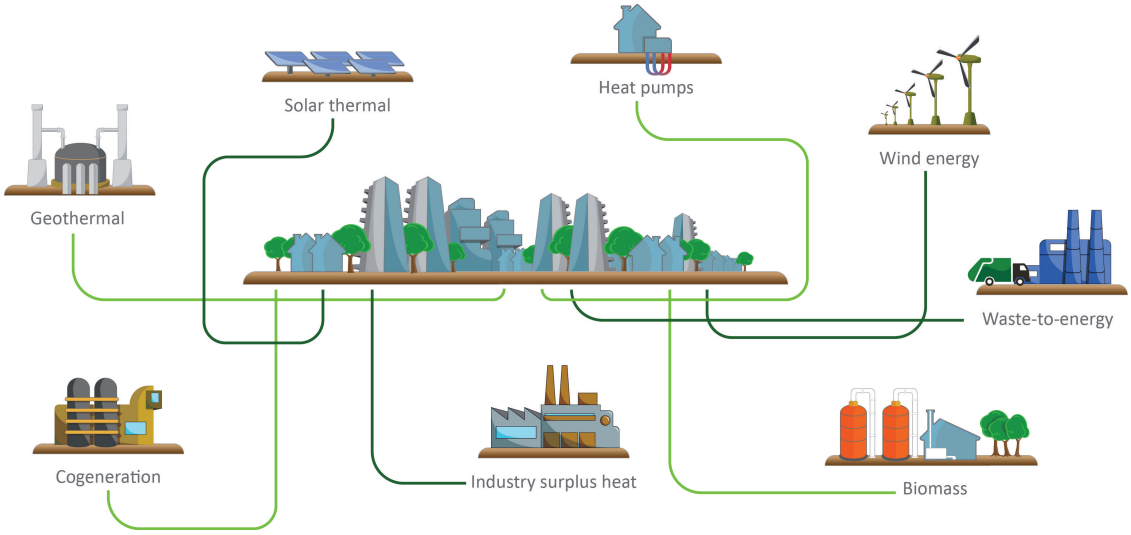


Figure 6.1: Heat sources for a district heating network[3]

- *Combined Heat and Power*: as explained above, it is a plant where simultaneous production of electrical and thermal energy occurs. They can be Combined cycle power plant, Internal combustion engines (ICE), steam power plants, nuclear power plants and so on.

- *Boiler plants*: they are thermal power plants based on fossil fuels such as coal, oil or methane (the most prevalent nowadays). In the past, they were the main source of thermal energy, while today they are mainly used to cover peak loads [62].

- *Waste-to-energy*: they are incineration plants that provide electricity and thermal energy like a CHP-based system, reducing at the same time the waste volume and mass. The thermal energy produced is then fed into the district heating network [54].

[3] `https://www.reuseheat.eu/district-heating-cooling/`

[3] The most common is Combine cycle gas turbines (CCGT).

- *Waste heat from industry*: many industrial activities produce waste heat that needs to be disposed of, which can be exploited in a district heating network. Typical industrial activities that produce this waste heat are process industries, iron and steel making, food industries, data center, etc. [53].

- *Biomass*: they are thermal power plants that instead of fossil fuels use biomass, mainly firewood and wood residues from agricultural or industrial processing. There is an environmental advantages as the biomass is renewable, releasing the same CO_2 into the environment as it has stored over a lifetime [52].

- *Heat pumps*: they are devices that allow transport of heat from a lower temperature level to a higher one, employing mechanical work. Heat can be withdrawn from the ground, from rivers or the sea, from the atmosphere and be provided to a DH network. The electricity needed to run the heat pumps can be supplied from RES, making this systems totally renewables [55].

- *Geothermal sources*: they exploit the energy stored, in form of heat, below the earth's surface. Depending on the enthalpy value, this heat can be used directly via high-temperature steam to support a district heating system. Or in addition, geothermal heat pumps can be used to recover thermal energy from the subsoil [55].

- *Solar thermal collectors*: with a large-scale system based on solar thermal collectors, a large amount of thermal energy can be produced. As the peak of thermal energy produced will be in summer, while the peak demand of a DH network is in winter, seasonal storage systems are used. Therefore these storage systems can release the energy stored in summer during the heating season [7].

- *Fuel cells*: in a fuel cell chemical oxide-reduction reactions take place, producing electrical energy. These reactions are exothermic and therefore the excess heat has to be removed. Therefore they can be seen as a CHP-based system producing both electricity and heat but with significantly higher efficiencies. This heat can be recovered via heat exchangers and fed into a local district heating network. However the size available on the market of fuel cells is not yet able to support a large DH network, but can only be exploited as micro-CHP for residential application [56].

6.2.2 Distribution

The distribution network is represented by a piping infrastructure typically buried in the soil, starting from the energy production plant and reaching each thermal user heat exchanger. Distribution network has to account for both supply and return piping system, where the former is at a higher temperature than the latter due to the supplied energy at the thermal user. Three types of configurations for the shape of the distribution exist, and they are [65]:

- *Mesh network*: composed of interconnected closed circuits. The best for heat regulation and distribution but also the most expensive.
- *Branched network*: as a tree structure it has a main line from which many secondary lines branch off to feed the consumers.
- *Ring network*: the system is a closed system and the fluid flows directly from the power station to the user and back again. It allows supplies in both directions.

As explained above, the heat transfer fluid is hot or superheated water in most of the cases. The system can be directly connected to the heat consumers without the need for heat exchangers, or it can be indirectly connected. In the latter case, a heat exchanger is placed in a heat substations to transfer heat from the DH network fluid (primary circuit) to the fluid inside the users circuit (secondary circuit). The indirect case allows the use of hot water, which lowers costs compared to superheated water, mainly because of the higher pressures that the pipes would have to withstand. In addition, the higher the temperature of the fluid, the higher dispersions through thermal losses into the environment will be in the distribution system.

6.2.3 Storage

Thanks to thermal storage systems, it is possible to decouple the instantaneous production and consumption profile of thermal energy, with the aim to reduce operational stresses due to the need to follow continuously the corresponding. In fact, this leads to a lower number of hours that the plant would run at partial load conditions and resulting in lower thermal efficiencies. The use of storage systems allows to reduce oversizing problem of power plants and allows introduction of further generation systems. In addition, cogeneration plants operate at all hours while the demand for thermal energy at night is extremely low. By means of thermal storage, thermal energy from cogeneration is stored at night and released during the day to cover peaks. These systems also benefit production from renewables [62].

6.2.4 Substations

There are pumping substations and user substations. The first ensure that there is no pressure drop in the heat transfer fluid along the entire network. The latter are the heat exchangers that separate the two circuits.

6.3 Forecasting

District heating, in addition to space heating, supply thermal energy for domestic hot water. The former depends mainly on external weather conditions while the latter depends more on sociological reasons and is therefore difficult to predict if a proper internal temperature management system is not equipped. In contrast to the two cases dealt with so far, there is no equation capable of providing through parameters the trend in thermal energy demand. This is because the system is much more complex and depends on many more variables.
The shape of a typical demand for DH depends strongly on the time of day, having a peak in the morning, which is usually the highest daily value. External weather conditions determine how marked this peak should be, but as there are also a huge number of users there are parameters that are impossible to determine and measure.
For this reason, methods to predict district heating demand are based on static methods or machine learning solutions. The important thing for these methods is therefore to have a large amount of data and as many linked variables as possible [69].

6.4 Case study

The application is made through the district heating network of the city Alba, in Italy, thanks to data provided by `EGEA SPA` and meteorological data from `Arpa Piemonte`. The comparison of the two dataset is done because the measurement tools have different precision, in fact the values of the two temperature curves are slightly different. The temperature of the first case is almost always higher than the second case, especially in summer where the temperature reached higher values. Eventually a comparison of the two datasets is done in order to fully understand how the input data could be advantageous or not for a model of artificial neural network.

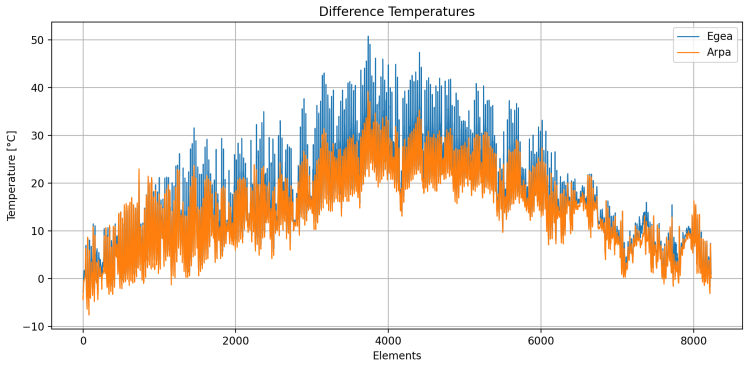


Figure 6.2: Temperatures for the two datasets

The blue curve, representing data from `EGEA SPA`, is almost always higher than the orange curve (`Arpa Piemonte`). The trend, however, is mostly the same, and for the functioning of a data driven model the trend can cover a significant relevance such as measured absolute values.

6.4.1 EGEA SPA case study

In this part the analysis of the data given by EGEA SPA is done.

Data processing

There are two files containing the dataset so this two are to be joined. In the first file the variables present are:

```
Date_Hour, External_Temperature, External_Dew_Point, External_Enthalpy,
External_Absolute_Humidity, External_Relative_Humidity
```

Using the command `dataframe.info()`, it can be seen that in the dataset there are present 8241 non-null elements for each of the features available. After renaming the variables in short form, and eliminating those considered not very influential for the problem, the final dataset is:

```
datetime, T_air, Hum
```

The selected humidity variable is the relative one and not the absolute one, while the *External Enthalpy* and *External Dew Point* variables are eliminated. The datetime range is from 22-01-2019 12:00 to 31-12-2019 22:00.
The other file containing the value of solar irradiance. There are two different solarimeter located in the city, and the one located in *Via Tanaro, Alba* is chosen. There are 8711 elements of irradiance and 8760 elements for datetime. Therefore the function `dataframe.fillna(method='bfill')` is used in order to fill the missing rows. The datetime range is from 01-01-2019 00:00 to 01-01-2020 00:00.
Next, to merge the two files, the start and end dates as well as the sampling frequency of each dataset must match. Using the `dataframe.asfreq('H')` method in both datasets ensures that there are no missing hours and using the `fillna` method these missing data are interpolated. Then after removing the first 22 days in the irradiance dataset the two datasets can be merged.
Finally, the excel file containing the district heating demand data also needs to be analysed. There are 8760 non-null elements, and the data range is from 01-01-2019 01:00 to 01-01-2020 00:00. Here again the first 22 days have to be eliminated and finally the whole dataset can be used and the model can be built.
The EGEA SPA dataset is thus composed by 8230 rows that represents almost a year, the 2019, with 3 input feature and one output variable.

Since the demand during some weeks of spring and autumn and all weeks of summer is almost constant and low, the dataset is cleaned by the data from 30-04 to 9-10 , thus remaining with 4368 rows. In order to obtain a link to the time of the measurements, two variable are created and added to the dataset: these are the hour of the day, useful for the daily pattern of the curve and a value equal to 1 for weekdays and 2 for weekend, effective for the lower value of demand during weekend.
The dependency between variables is evaluated with the linear correlation thanks to the Pearson correlation coefficient.

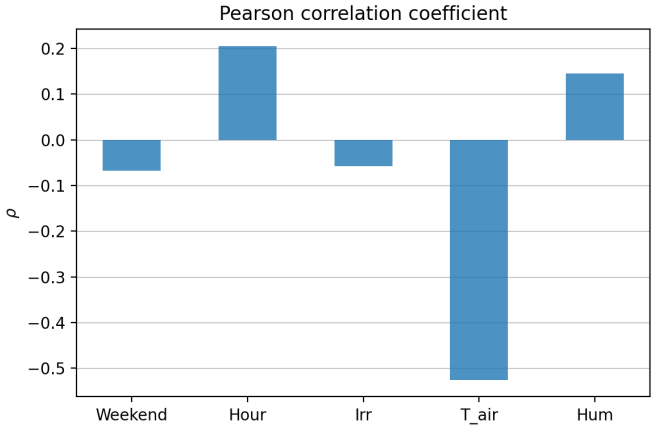


Figure 6.3: Pearson correlation coefficient respect to DH load, `EGEA SPA` dataset

From Figure 6.3 the variables most important are *T_air* [Air Temperature] and *Hour*.
Chosen the composition of the dataset, it is useful to rescale all the values according to a specific feature scaling. In this case the *Normalization Scaler* is used, so each feature will have range from 0 to 1. It's important then after the training process to compute the inverse scale transformation to return the real value of the features.

Next, for the processing of the data, it's necessary to split into training, validation and test datasets.
Using a week as forecast period, the dataset is adjusted so that it consists of an integer number of weeks. Thus the new number of elements that make up the dataset is still 4368. Consequently, as there are 168 hours in a week, there are 26 weeks in total. Therefore 14 weeks are used for training, 8 for validation and 4 for testing. The test weeks will then be evaluated individually in order to check the accuracy of the model.
So the training dataset has 2352 elements for each feature, the validation 1344 and the test 168 for each of the four week.

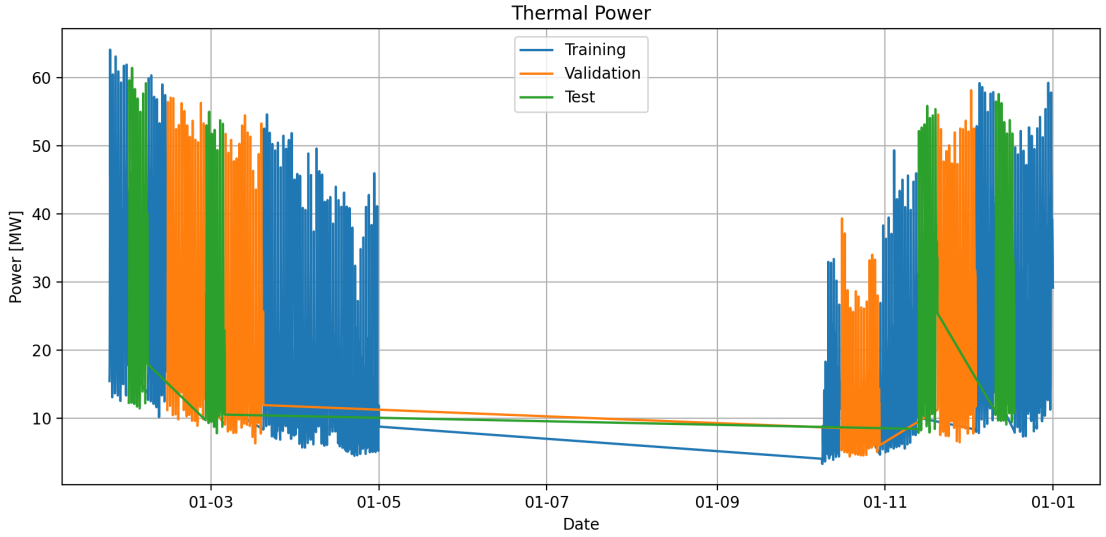


Figure 6.4: Demand curve power

Figure 6.4 shows the demand of the DH and it is also possible to view the division explained above, where the blue curve is for the training, the orange for the validation and the green for test dataset.
The period selected for validation tries to be as complete as possible for validating the model in both high-load and low-load weeks.

6.4.2 `Arpa Piemonte` case study

In this part the analysis of the data given by `Arpa Piemonte` is done.

Data processing

From the website `http://www.arpa.piemonte.it/rischinaturali/accesso-ai-dati/Richieste-dati-formato-standard/richiesta-dati/Richiesta-automatica/Dati-giornalieri-richiesta-automatica.html` it is possible to download hourly weather data for several cities of Piedmont (Piemonte) Italian region. The parameters that can be downloaded are precipitation, air temperature, wind speed, wind direction and gust. As the problem is related to district heating only the air temperature variable has been downloaded.
So here, the dataframes with the DH load curve and the meteorological variable were merged. In addition, the same two variables *Hour* and *Weekend* have been added and, in order to have the same number of samples, the length of dataset is limited making it equal to the `EGEA SPA` one.
The correlation coefficient is also calculated for this case.

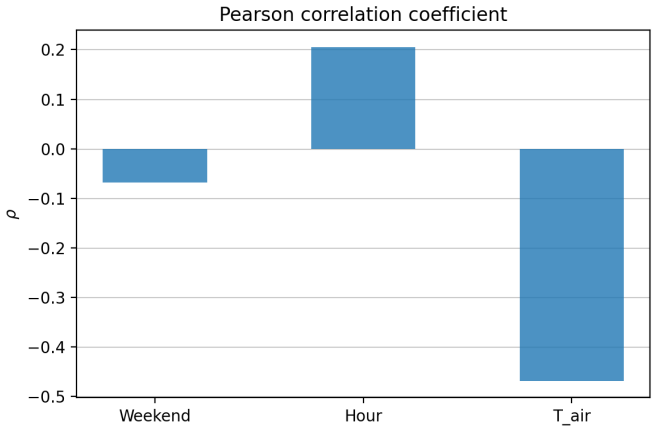


Figure 6.5: Pearson correlation coefficient respect to DH load, `Arpa Piemonte` dataset

The correlation for the added variables is the same as for `EGEA SPA` case, while *Air temperature* has a slightly lower correlation. This is because, as explained above, the two cases have slightly different temperature values. This could lead to different results even though the shape of the demand curve is not only strictly dependent on temperature but mainly on hour of the day, keeping a more or less constant shape.

6.4.3 Layout description

As for photovoltaic and solar thermal energy case studies, an analysis is performed to discover the best architecture for the given district heating forecasting problem. The used dataset is the one from `EGEA SPA`. The different steps followed during this phase and the order in which they were carried out are:

1. Number of hidden layers.
2. Type of network.
3. Timesteps.
4. Number of neurons in each layers.

The procedure followed is the same as that explained in the photovoltaic section 4.5.2, but other hyperparameters have been modified in this case. Since a significant dependence on time factors is present, the use of a recurrent network can be justified and could be leading to better results. So a configuration comparison between using a feedfoward or a recurrent first hidden layer is hence performed. After that it is important to calculated the better timesteps required for this application and finally the number of neurons.

For the model purposes, 26 weeks has been chosen and the split is 17 weeks for training, 6 weeks for validation and 3 weeks for test. Figure 6.6 shows the training, the validation and the test sets. The number of epochs was not predetermined a priori because of `EarlyStopping` will stop the procedure as soon as after 50 epochs the validation loss does not improve significantly the training and its loss function, and the best weights obtained are restored.
The loss function chosen is `MSE`, while the optimizer selected is `Adam` and for the activation part each layer have `ReLU` function. An exception is made for the output layer, where for similar regression problems the linear activation can provide accurate predictions.
The number of batchsize is increased to 128 from the default value of 32 due to the number of data and feature in the dataset, as was also the case with solar thermal energy. As well as cases where the learning rate and batchsize vary have been omitted.

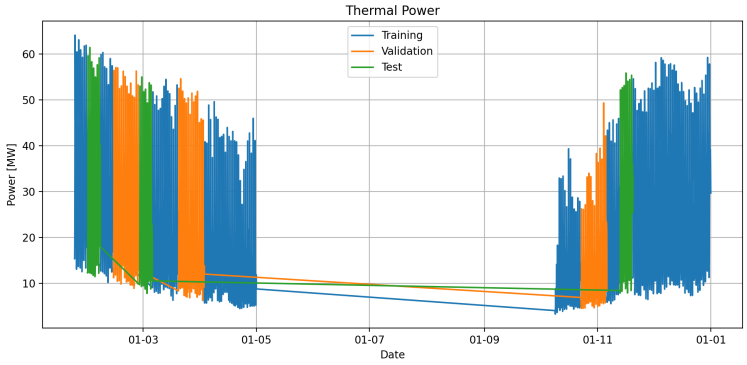
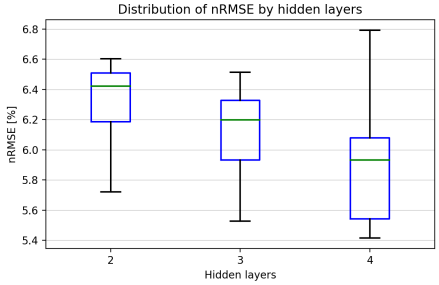


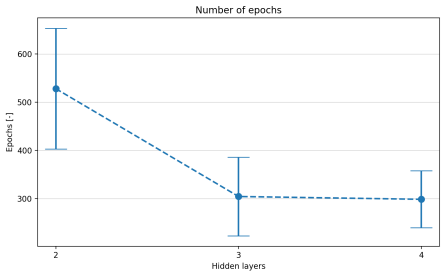
Figure 6.6: Split used for the layout built for DH

Hidden layers

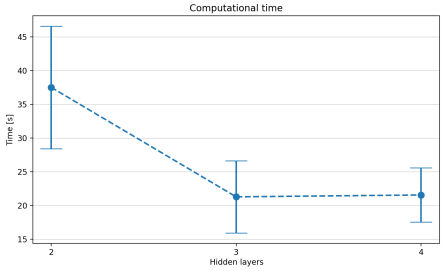
The first point concerns the number of layers in the ANN. The number of layers range from 2 to 4. The value of neurons was kept constant in each run, as well as the value of learning rate (left at default value of 10^{-3}). For the the first test the number of neuron is set to (8, 16), for the second to (8, 16, 8) and for the last to (8, 16, 16, 8).



(a) Distribution of results



(b) Number of epochs



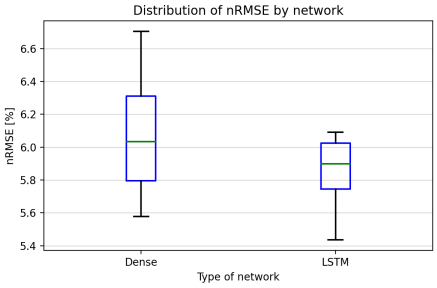
(c) Computational time

Figure 6.7: Results for the number of hidden layers

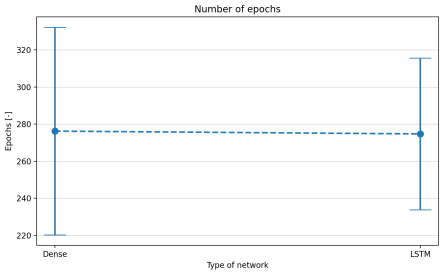
Figures 6.7 show that the best solution concerning the number of hidden layer is 4. Epochs gradually decreasing as the model converges more quickly, as well as a lower required computational time has been observed.

Type of network

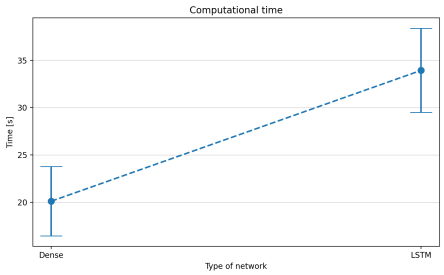
The number of hidden layer is set to 4, as result of previous analysis, and the other hyperparameters are maintained constant. For the recurrent network a LSTM layer is used, with time step initially set equal to 3. The latter means that the given prediction at a generic time instant will be function of the previous 3 monitored values. The activation function for LSTM is chosen equal to tanh.



(a) Distribution of results



(b) Number of epochs



(c) Computational time

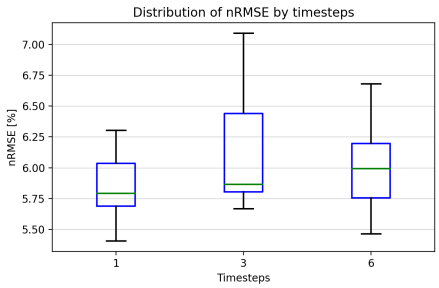
Figure 6.8: Results changing type of network

From results it can be seen that LSTM layer leads to better results, in contrast to the case of solar thermal energy. This is due to the shape of the DH demand that depends mainly on the time of day and it is more suitable for the purpose of a recurrent network (timeseries forecasting)[4]. In fact the shape of the district heating demand has a lower dependence from temperature than the irradiance has for solar thermal energy, thus maintaining the same curve shape repetitively. Consequently, LSTM was chosen as the first hidden layer, even though it involves higher computational costs.

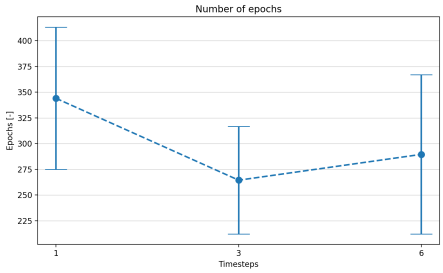
[4]As explained in section 2.3.2

Timesteps

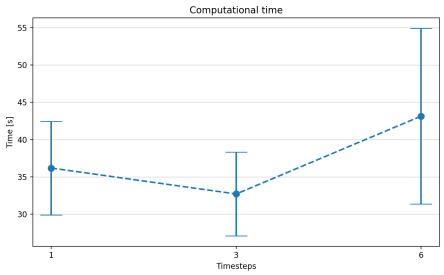
Having therefore chosen the first hidden layer as a LSTM, it is important to establish which is the best timestep to use, therefore is made to vary with three different values: 1, 3 and 6.



(a) Distribution of results



(b) Number of epochs



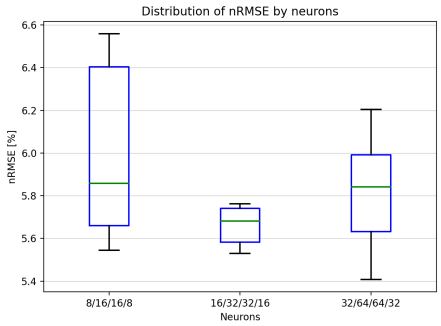
(c) Computational time

Figure 6.9: Results for different timesteps

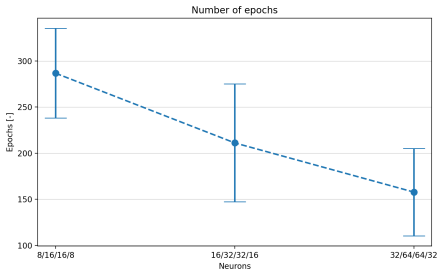
A value equal to 3 leads to convergence earlier but a slightly worse result, therefore the timesteps selected is then 1.

Neurons

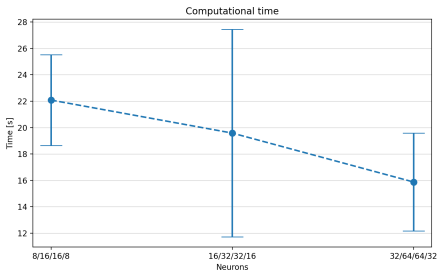
The range for the neurons is set to be varying from 8 to 32 in the first layer, doubling each time. The other layers maintaining the proportion used before, in line with the study performed with photovoltaic and solar thermal. As usual, all the other parameters remain unchanged.



(a) Distribution of results



(b) Number of epochs



(c) Computational time

Figure 6.10: Results for the number of neurons

From figures 6.10 the number of neurons that it is decided to select is the (16, 32, 32, 16), which as it can be seen leads to a better result without compromising computational time.

6.4.4 Final layout

The final layout is composed by 4 hidden layers composed by (16, 32, 32, 16) neurons. Eventually, there is a output layer composed on one single neuron. The total number of parameters that composed the model is 3553.
All the layers, except the first hidden layer, are *Dense*, instead the first one is chosen as *LSTM* to capture time dependency of the district heating profile. In this application the time step, also known as *lag*, selected is equal 1.
The activation function for the last three hidden layer is *ReLu*, while for the first hidden layer is *tanh*.

The summary of the network built in python is

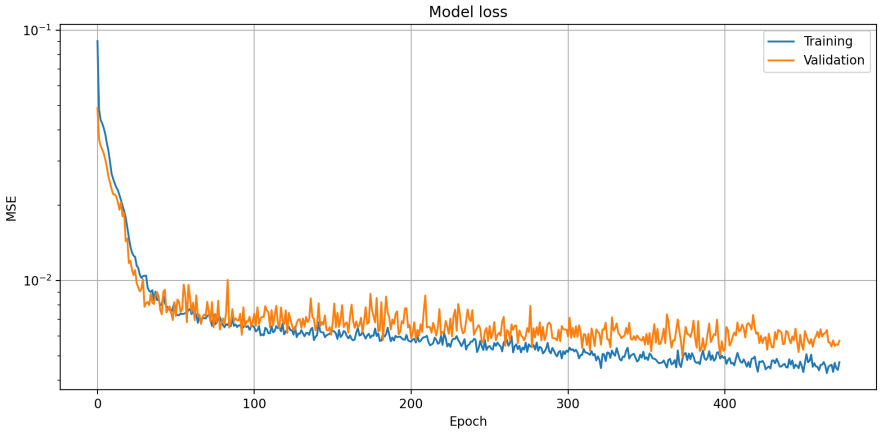
```
model = keras.Sequential()
model.add(keras.Input(shape=(x_train.shape[1],x_train.shape[2])))
model.add(keras.layers.LSTM(16, activation='tanh'))
model.add(keras.layers.Dense(32, activation='relu'))
model.add(keras.layers.Dense(32, activation='relu'))
model.add(keras.layers.Dense(16, activation='relu'))
model.add(keras.layers.Dense(1))
```

After building the model, it has to be compiled choosing the right loss function and optimizer. In this problem the loss function chosen is the Mean Square Error, while the optimizer chosen is Adam. The learning rate of Adam is changed to $1 \cdot 10^{-3}$, while the batch size selected is set to 128. The epochs are 2000 but with function `EarlyStopping`, as soon as after 100 epochs the validation loss does not improve the training stops and the best weights obtained during training are restored.

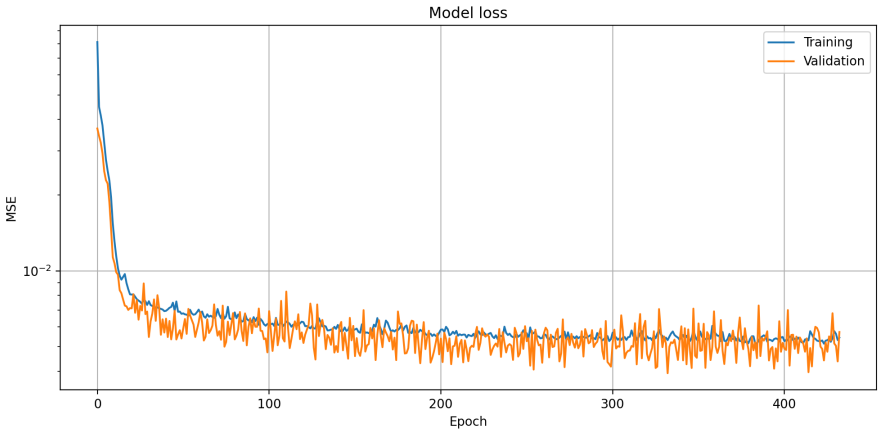
```
opt = keras.optimizers.Adam(learning_rate=1e-3)
model.compile(loss='mse', optimizer=opt)
model.summary()
#
my_callbacks = [keras.callbacks.TerminateOnNaN(),
        keras.callbacks.EarlyStopping(monitor='val_loss', patience=100,
        restore_best_weights=True, verbose=1)]
#
batchsize = 128
history = model.fit(x_train, y_train, epochs=2000, batch_size=batchsize,
                    validation_data=(x_val,y_val),
                    callbacks=my_callbacks, verbose=2)
```

6.4.5 Results

Results for both cases are now shown. After the training process the learning curves of the model, both for training and validation dataset, is plotted to visualize how the loss function evolves during the epochs. For `EGEA SPA` case the first curves drawn showed overfitting problems (problems that could not be found during the architecture phase as the learning curves were not printed). Therefore a proper systems against overfitting is used, and in this case after second hidden layer a dropout one is used[5]. While for `Arpa Piemonte` problems of overfitting were not found[6].
The average computational time for training is about 45 seconds for both cases.



(a) `EGEA SPA`



(b) `Arpa Piemonte`

Figure 6.11: Learning curves

[5]Equal to 0.25, thus discarding 25% of the weights at the output of that layer.

[6]This is consistent since in the first case more features are present and it becomes easier for the network to overfit.

Figure 6.11a shows the learning curves for the first `EGEA SPA` and for the second dataset `Arpa Piemonte`. In both figures the validation loss curve follows quite well the training loss one and this means that the model gives a good fitting.

Later some score values for each week of the test data are calculated in order to better understand the prediction results.

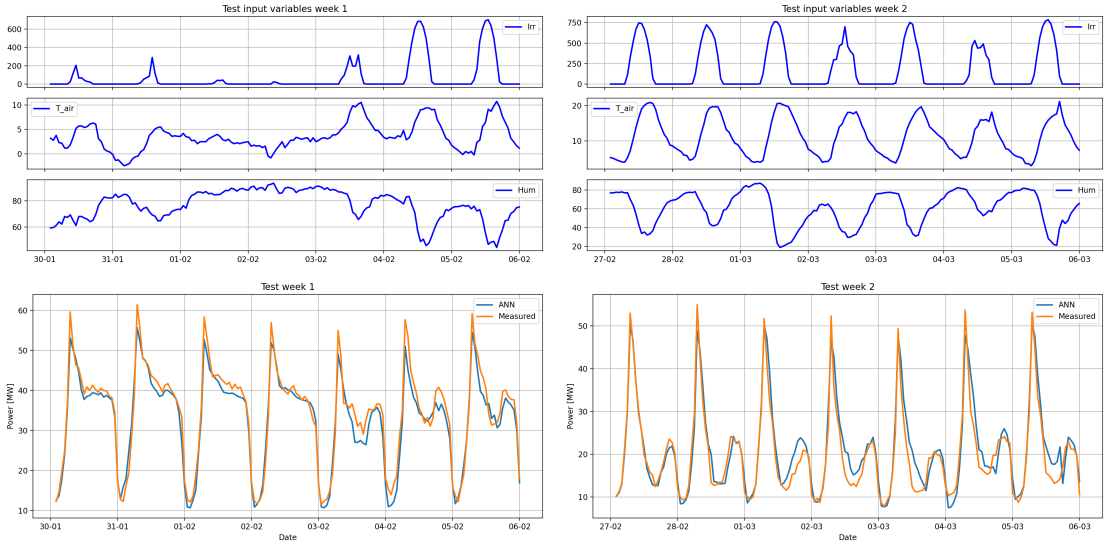
Table 6.1 summarizes the mean and the standard deviation for the scores calculated for both cases studied. The absolute hourly error is a vector with the same length of the predicted power and so for simplicity only the maximum value of this error for each week is used. Instead for the percentage hourly error the values used are the mean of the 4 weeks based on the mean of all hour in each week.

Score	EGEA SPA		Arpa Piemonte	
	Mean	Deviation	Mean	Deviation
$\max(\lvert e_h \rvert)$ $[MW]$	9.43	2.01	11.54	2.28
e_p [%]	10.48	3.00	10.60	2.17
RMSE $[MW]$	3.14	0.36	3.36	0.43
nRMSE [%]	5.49	0.81	5.89	0.97
R^2_{adj} [-]	0.92	0.02	0.91	0.03

Table 6.1: Summarize of final scores

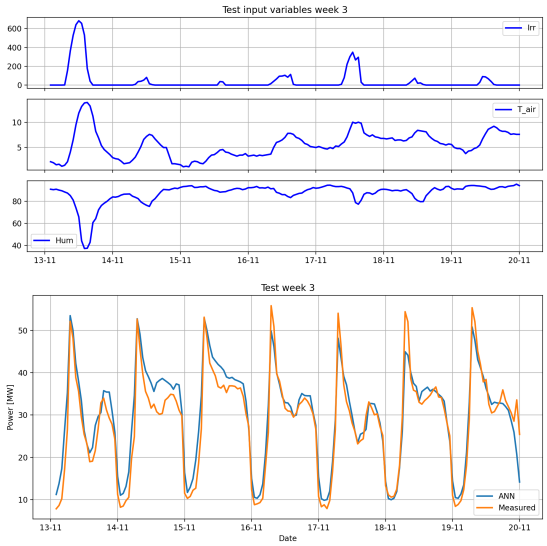
The results show that the neural network is able to predict with sufficient accuracy the demand power curve, and that the predicted curve is almost following the measured one. But also that the results are poorer in comparison with previously tested models as is known from the difficulty of predicting this load curve.
Furthermore the first dataset have better scores, maybe due to the larger number of features that help the model to predict the curve.

The results obtained for the two cases are now presented, showing first the case with the data provided by `EGEA SPA` and then those from `Arpa Piemonte`.

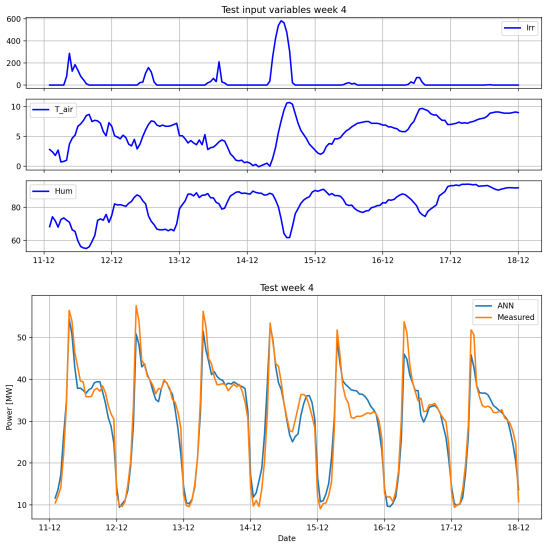


(a) Inputs and test for DH week 1

(b) Inputs and test for DH week 2



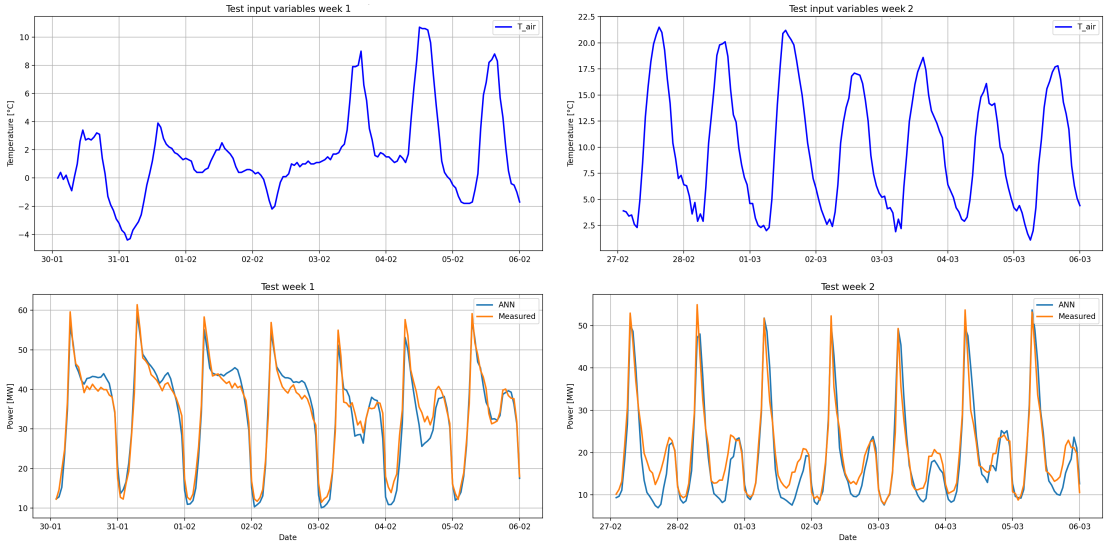
(c) Inputs and test for DH week 3



(d) Inputs and test for DH week 4

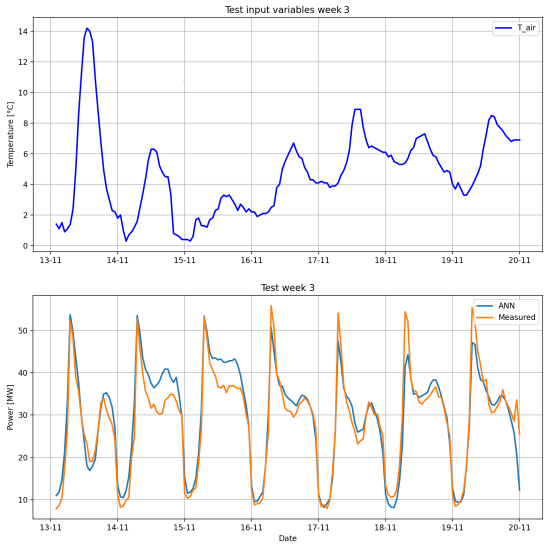
Figure 6.12: Summarize of total tested weeks with `EGEA SPA` dataset

Figure 6.12 then show all the week tested for the first case, with the meteorological inputs used by the model and the power curve of the plant with the measured and predicted (ANN) curves. In three cases the load curve has the same trend while in march the demand has an overall different shape. The model manages to predict well except for some peaks where it underestimates the power, and in some afternoon moments where the demand remains more or less constant.

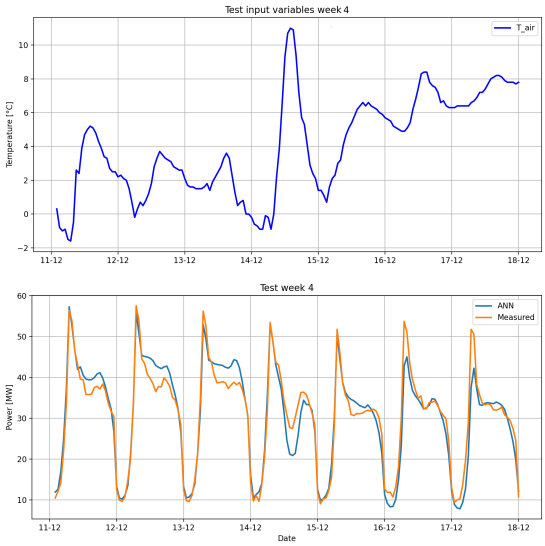


(a) Inputs and test for DH week 1

(b) Inputs and test for DH week 2



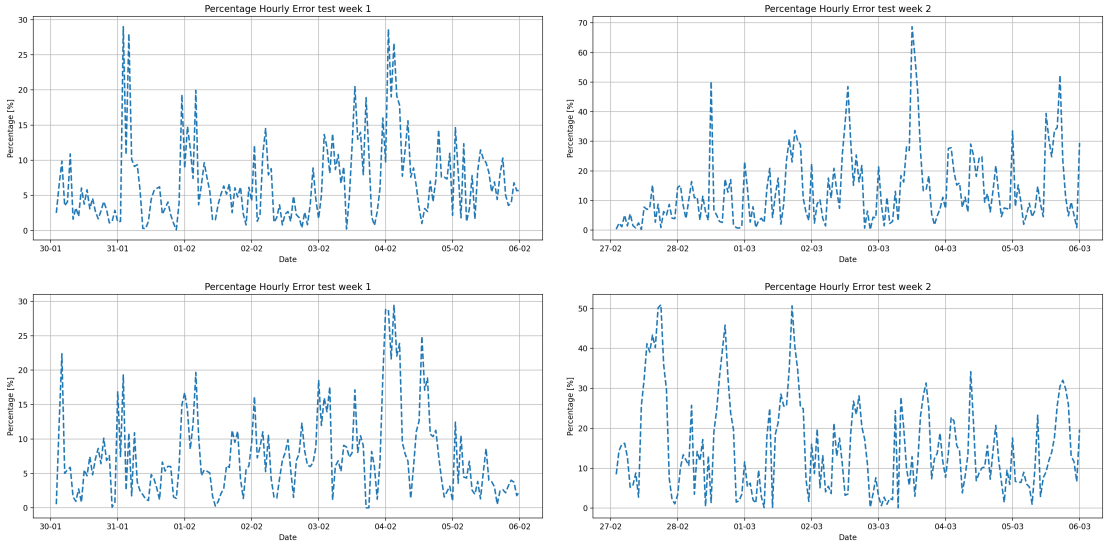
(c) Inputs and test for DH week 3



(d) Inputs and test for DH week 4

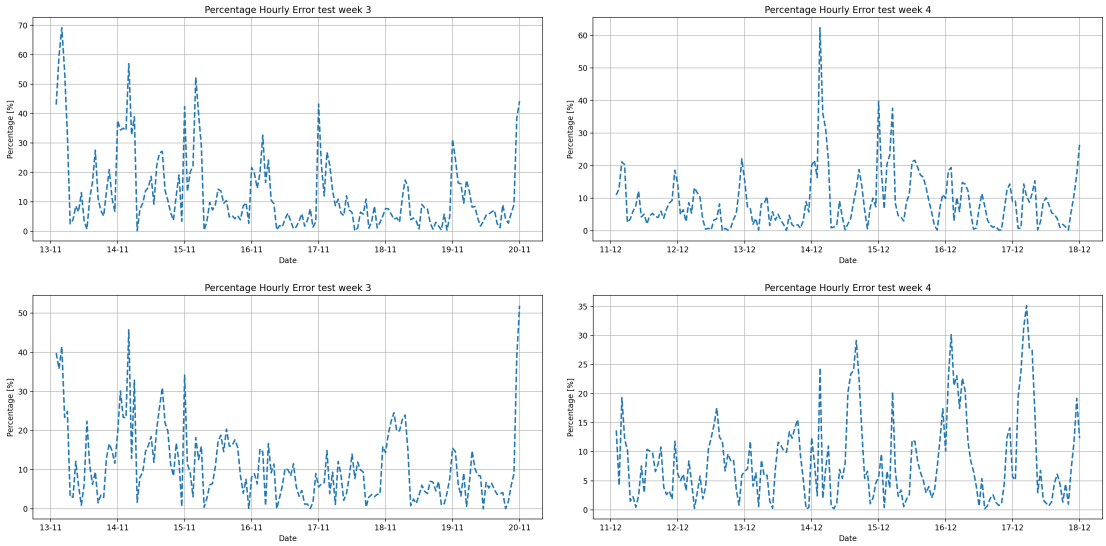
Figure 6.13: Summarize of total tested weeks with `Arpa Piemonte` dataset

Figure 6.13 instead shows the same graph explained above for the second case. Overall, the model struggles more in this case to predict the demand curve in an accurate manner. As explained, the only meteorological input is the air temperature. From these graphs one can see very well the dependence of the thermal power demand on the outside temperature.



(a) `EGEA SPA` and `Arpa Piemonte` e_p week 1

(b) `EGEA SPA` and `Arpa Piemonte` e_p week 2



(c) `EGEA SPA` and `Arpa Piemonte` e_p week 3

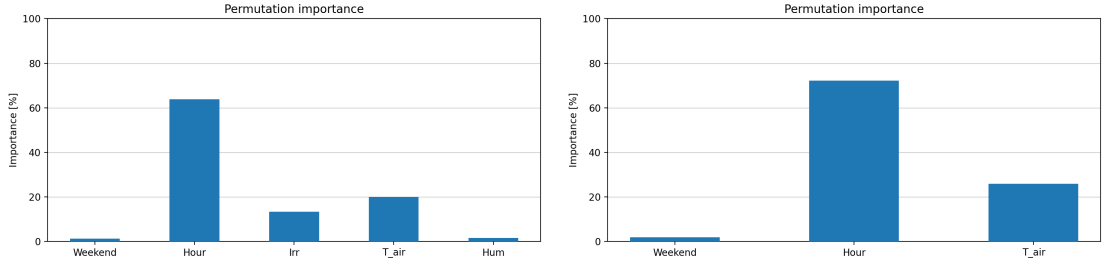
(d) `EGEA SPA` and `Arpa Piemonte` e_p week 4

Figure 6.14: Percentage error of all tested weeks

Figures 6.14 show how much and when the models are wrongs in predicting the demand curve. Finally it can see that the model based on `Arpa Piemonte` data is worse, especially during the lowest demand in march.

For both models the most difficult moments of the day are during rapidly changes of the curve, such mornings and evenings where daily load curves slopes suffers from rapid variation with quick increase and decrease steps. But also when there is an increase of the air temperature and so to a reduction of the thermal demand of the DH.

A procedure called permutation importance makes it possible to understand how the model uses the input variables and with what entity. Then one by one the features are randomly mixed and the model is re-evaluated. This procedure is carried out for all variables present and the result compared with the nRMSE of the unchanged dataset. From there a percentage of importance for the model is obtained. The procedure is carried out in both cases.

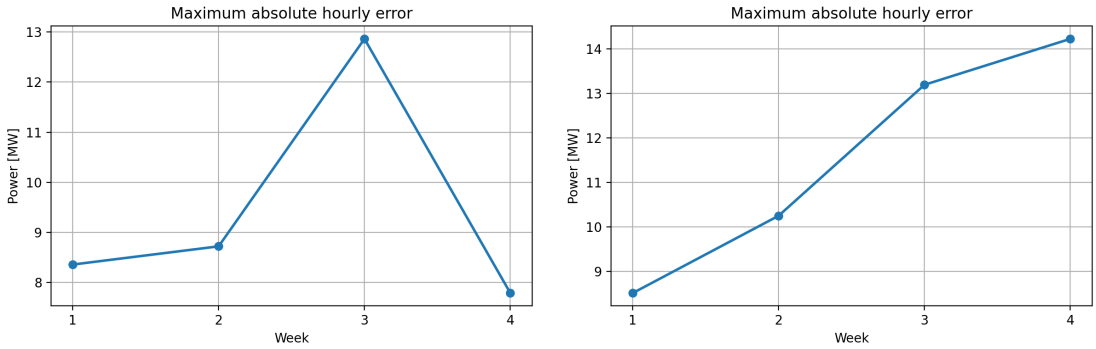


(a) `EGEA SPA` (b) `Arpa Piemonte`

Figure 6.15: Permutation importance

The variable *Hour* in both cases is quite important, followed by *T_air* [Air temperature]. In the first case, *Irr* [Irradiance] is also important, while *Hum* [Humidity] is very little. In both cases, however, *Weekend* is very unimportant, yet without that variable on weekends there would always be an overestimation of the thermal energy demand.

Eventually the trend of maximum absolute hourly error for the 4 weeks is shown.



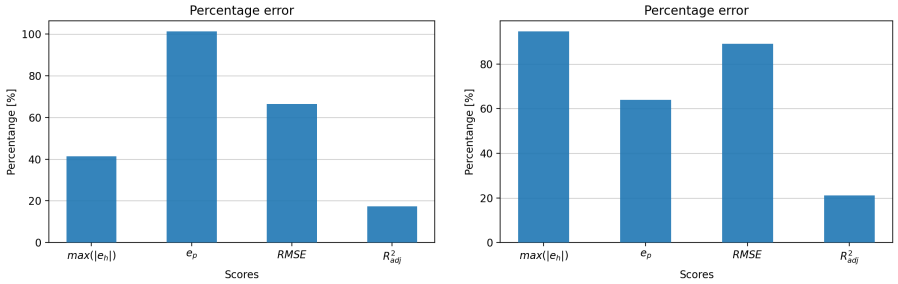
(a) `EGEA SPA` (b) `Arpa Piemonte`

Figure 6.16: Max absolute hourly error for both datasets

Under this parameter the last week tested reaches a much higher value in the second case than in the first.

6.4.6 Offset

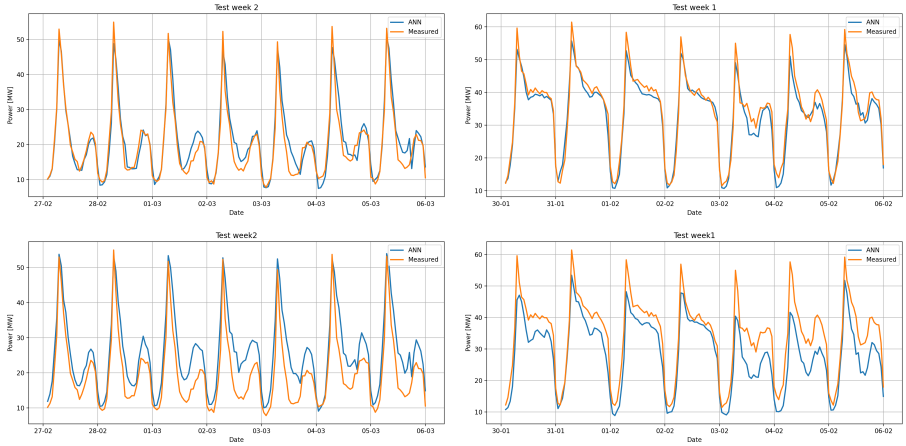
The procedure followed is the same as that explained in sub section 5.3.5. In this case, however the feature modifies is the air temperature, where in the first case it is decreasing of -3 and in the second case increasing of +3 degree Celsius.
The analysis is computed with `EGEA SPA` dataset.



(a) Decrease temperature errors (b) Increase temperature errors

Figure 6.17: Percentage error for both cases

In figures 6.17 it can be seen how all the parameters calculated as a score have deteriorated in absolute percentage terms. It can be seen that even if the offset is positive or negative, the scores change differently.



(a) Test week 2 reduction temperature (b) Test week 1 increase temperature

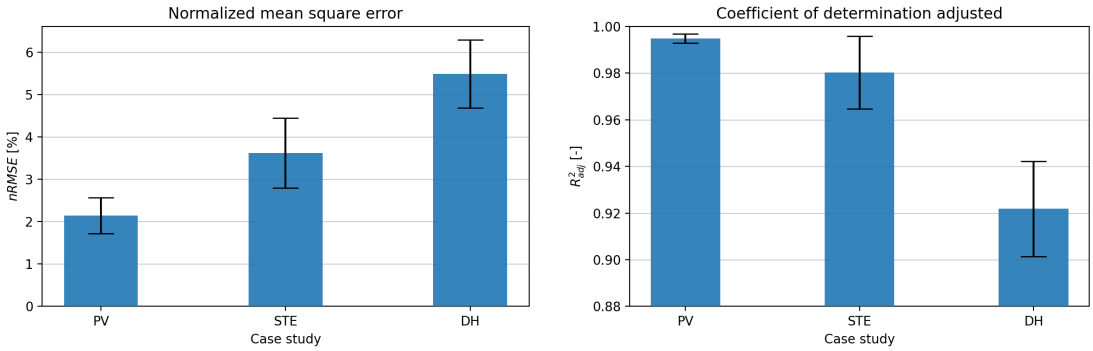
Figure 6.18: Comparison selected weeks both cases

Finally in figures 6.18 two weeks were depicted for the two cases. When the air temperature decreases, the model estimates a higher power demand; otherwise when the temperature increases, the model underestimates the demand. These results show how the network actually uses the air temperature feature, which modifies the DH demand by adapting it to the new inputs.

Conclusion

This thesis has shown that machine learning-based methods are able to make accurate predictions on both load curves and renewables energy. Furthermore, the importance of the quality of the data used and the appropriate use of the right variables for the specific case has been demonstrated. It has also been found that for not complex systems, such as photovoltaics, the results are outstanding, but are just as achievable with less complex and statically based methods.
The improvements that can be made in these areas will facilitate the continued growth of renewables in different sectors and mitigate all the possible drawbacks. These are due to the difficulty in integrating with the network because of their randomness and consequently more difficult control than traditional methods.

The graph in figure 6.19 summarises two main score, dimensionless, which make it possible to compare the different applications tested. The results were obtained by averaging all the test periods used in the individual cases, and also checking for dispersion using the standard deviation.



(a) $nRMSE$ (b) R^2_{adj}

Figure 6.19: Scores comparison for the three cases

All three calculated scores show that from photovoltaic to district heating they get worse. This is because the complexity of the problem and the models built struggle more to achieve comparable results.

Anyway, it must be remarked how the error values are low, showing that results obtained are very promising and with the addition of more historical data and systems that automatically re-train the network as soon as more data is added, increasingly high accuracy values would be achieved.
Once the models have been created and trained, they can be saved in specific files and called up whenever necessary. It is only important to keep the same number of features used and, if they are used with feature scaling, that they are of the same type, otherwise they would be scaled differently.

This thesis has therefore shown the capabilities that these models have in this area. But these same tools can be used in many other areas (still in the energy sector). For example, wind power, which relies heavily on the presence and direction of wind (which is much less predictable than solar radiation). Forecasting the amount of pollutants produced by a power plant by knowing the load and weather conditions.

A future evolution of the treated models would use algorithms no longer based on error back propagation but on metaheuristic methods. A major shortcoming of error back propagation is the need to determine the initial values (which are made randomly by the program itself) and which then determine the accuracy of the model. This is why more than one complete run has always been performed to obtain comparable results.
Thus, models based on Genetic Algorithm or Particle Swarm Optimisation have been implemented in some studies leading to very good results using also hybrid methods combining these two algorithms [70].
The use of these innovative algorithms allows good fusion with artificial neural networks, such as the possibility of using them to find the best architecture, and therefore the best hyperparameters, without having to search for the best solution using time-consuming methods [71].

Acknowledgements

This space is dedicated to those who, with dedication and patience, have contributed to the realization of this work.

A special thanks goes to my supervisors, *Prof. Maurizio Repetto*, who followed me, with his infinite availability, in every step of the realization of the thesis, from the choice of the topic, and *Prof. Francesco Grimaccia* for the references and articles provided for this thesis.
Thanks also to my co-supervisor, *Ivan Mariuzzo*, for his grateful advice and for having suggested the right changes to make to my thesis.

I would also like to thank `SOLID Solar Energy Systems GmbH, Graz, Austria` for providing the necessary data to estimate the solar thermal panels power and `EGEA SPA` for data concerning the meteorological measurements and district heating load of *Alba, Italy.*

Last but not least, I also want to thank my all Family, without their help and support I would not have made it this far.

Bibliography

[1] V. Masson-Delmotte, P. Zhai, H. O. Pörtner, D. Roberts, J. Skea, P.R. Shukla, A. Pirani, W. Moufouma-Okia, C. Péan, R. Pidcock, S. Connors, J. B. R. Matthews, Y. Chen, X. Zhou, M. I. Gomis, E. Lonnoy, T. Maycock, M. Tignor, T. Waterfield, (2018). *Global warming of 1.5°C. An IPCC Special Report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty.* IPCC.

[2] Hannah Ritchie and Max Roser, (2019). *Access to Energy.* Our World in Data. `https://ourworldindata.org/energy-access`

[3] Morna Isaac, Detlef P.van Vuuren, (2009). *Modeling global residential sector energy demand for heating and air conditioning in the context of climate change.* Energy Policy, Volume 37, Issue 2, Pages 507-521.

[4] World Energy Council, (2020). *World Energy Trilemma Index (2020).* In Partnership with Oliver Wyman. Used by permission of the World Energy Council.

[5] Nicolás Pérez-Mora, Paolo Lazzeroni, Victor Martínez-Moll, Maurizio Repetto, (2017). *Optimal management of a complex DHC plant.* Energy Conversion and Management, Volume 145, Pages 386-397.

[6] Hawken, P., (2017). *Drawdown: the most comprehensive plan ever proposed to reverse global warming.* New York, Penguin Books.

[7] Patrick Reiter , Hannes Poier, Christian Holter, (2016). *BIG Solar Graz: Solar District Heating in Graz – 500,000 m2 for 20% Solar Fraction.* Energy Procedia, Volume 91, Pages 578-584.

[8] Ettore Francesco Bompard, (2020). *Slides of course Smart electricity systems.* Polytechnic University of Turin.

[9] IEA, (2012). *Technology Roadmap - Solar Heating and Cooling.* IEA, Paris.

[10] Nishant Shukla, with Kenneth Fricklas, (2018). *Machine Learning with Tensorflow.* Manning publication.

[11] Wes McKinney, (2010). *Data Structures for Statistical Computing in Python.* Proceedings of the 9th Python in Science Conference, Pages 56-61.

[12] Herculano-Houzel, Suzana, (2009). *The Human Brain in Numbers: A Linearly Scaled-up Primate Brain.* Frontiers in human neuroscience, Volume 3, Pages 31.

[13] Pedregosa, F., Varoquaux, G., Gramfort, A. and Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., (2011). *Scikit-learn: Machine*

Learning in Python. Journal of Machine Learning Research, Volume 12, Pages 2825-2830.

[14] Hunter, J. D., (2007). *Matplotlib: A 2D graphics environment.* Computing in Science & Engineering, Volume 9, Pages 90-95.

[15] François Chollet and others, (2015). *Keras.* `https://keras.io`

[16] Ronald van Loon, (2018). *Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning.* Big Data Made Simple. `https://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/`

[17] Jason Brownlee, (2021). *How to Choose an Activation Function for Deep Learning.* Machine Learning Mastery. `https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/`

[18] Jason Brownlee, (2019). *How to Choose Loss Functions When Training Deep Learning Neural Networks.* Machine Learning Mastery. `https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/`

[19] François Chollet, (2017). *Deep Learning with Python.* Manning publication.

[20] Ian Goodfellow, Yoshua Bengio, Aaron Courville, (2016). *Deep Learning*, MIT Press.

[21] Fjodor Van Veen, (2016). *The neural network zoo.* The Asimov Institute. `https://www.asimovinstitute.org/neural-network-zoo/`

[22] S. Leva, A. Dolara, F. Grimaccia, M. Mussetta, E. Ogliari, (2017). *Analysis and validation of 24 hours ahead neural network forecasting of photovoltaic output power.* Mathematics and Computers in Simulation, Volume 131, Pages 88-100.

[23] Avinash Sharma V., (2017). *Understanding Activation Functions in Neural Networks.* The Theory Of Everything. `https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0`

[24] E. Isaksson, M. Karpe Conde, (2018). *International Journal of Machine Learning and Computing.* KTH Royal Institute of Technology.

[25] Sanket Doshi, (2019). *Various Optimization Algorithms For Training Neural Network.* Towards Data Science. `https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6`

[26] Simeon Kostadinov, (2019). *Understanding Backpropagation Algorithm.* Towards Data Science. `https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd`

[27] Diederik P. Kingma and Jimmy Ba, (2017). *Adam: A Method for Stochastic Optimization.* Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego.

[28] Christopher Olah, (2015). *Understanding LSTM Networks.*COlah's blog.

[29] Baijayanta Roy, (2020). *All about Feature Scaling, Scale data for better performance of Machine Learning Model.* Towards Data Science. `https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35`

[30] Michel Jose Anzanello, Flavio Sanson Fogliatto, (2011). *Learning curve models and applications: Literature review and research directions.* International Journal of Industrial Ergonomics, Volume 41, Issue 5, Pages 573-583.

[31] Jason Brownlee, (2019). *How to use Learning Curves to Diagnose Machine Learning Model Performance.* Machine Learning Mastery. `https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/`

[32] François Chollet, (2018). *Introduction to Keras.* Tensorflow for Deep Learning Research, Stanford University.

[33] Ahmad Alzahrani, Pourya Shamsi, Cihan Dagli, and Mehdi Ferdowsi, (2017). *Solar Irradiance Forecasting Using Deep Neural Networks.* Procedia Computer Science, Volume 114, Pages 304-313.

[34] Mahdi Sharifzadeh, Alexandra Sikinioti-Lock, Nilay Shah, (2019). *Machine-learning methods for integrated renewable power generation: A comparative study of artificial neural networks, support vector regression, and Gaussian Process Regression.* Renewable and Sustainable Energy Reviews, Volume 108, July 2019, Pages 513-538.

[35] Huaizhi Wang, Zhenxing Lei, Xian Zhang, Bin Zhou, Jianchun Peng, (2019). *A review of deep learning for renewable energy forecasting.* Energy Conversion and Management, Volume 198.

[36] Fei Wang, Zhiming Xuan, Zhao Zhen, Kangping Li, Tieqiang Wang, Min Shi, (2020). *A day-ahead PV power forecasting method based on LSTM-RNN model and time correlation modification under partial daily pattern prediction framework.* Energy Conversion and Management, Volume 212.

[37] Lo Brano, V., Ciulla, G., Di Falco, M, (2014). *Artificial Neural Network to Predict the Power Output of a PV Panel.* International Journal of Photoenergy.

[38] Raza, Muhammad Qamar & Nadarajah, Mithulananthan & Ekanayake, C., (2016). *On recent advances in PV output power forecast.* Solar Energy, Volume 136, Pages 125-144.

[39] Mohammed S, Sheik & Devaraj, D., (2014). *Simulation and Analysis of Stand-alone Photovoltaic System with Boost Converter using MATLAB/Simulink.* International Conference on Circuits, Power and Computing Technologies, Nagercoil, India, 2014, Pages 814-821.

[40] Raheel Shaikh, (2018). *Cross Validation Explained: Evaluating estimator performance.* Towards Data Science. `https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85`

[41] Spertino Filippo, (2019). *Slides of course Power generation from renewable sources.* Polytechnic University of Turin.

[42] Vasilis Fthenakis, (2012). *Third Generation Photovoltaics.* Renewable Energy, IntechOpen.

[43] Cutler J. Cleveland, Christopher Morris, (2013). *Section 10 - Solar.* Handbook of Energy, Pages 405-450.

[44] Rebecca Lindsey, (2009). *Climate and Earth's Energy Budget.* Earth Observatory. `https://earthobservatory.nasa.gov/features/EnergyBalance`

[45] Detlev Heinemann, (2002). *Energy Meteorology.* Lecture notes Renewable Energy, Carl Von Ossietzky Universitat, Oldenburg.

[46] Günther, M., Janotte, N., Mezrhab, A., Hennecke, K., Schillings, C., Wilbert, S., &

Wolferstätter, F., (2011). *Advanced CSP Teaching Materials.* Chapter 2 Solar Radiation.

[47] Simonetti Marco, (2018). *Slides of course Technology for renewable energy sources.* Polytechnic University of Turin.

[48] D.R. Myers, (2012). *Solar Radiation Resource Assessment for Renewable Energy Conversion.* Comprehensive Renewable Energy, Volume 1, Pages 213-237.

[49] J. Paul Guyer, (2018). *Introduction to Solar Collectors for Heating & Cooling Buildings and Domestic Hot Water.* CEDengineering.

[50] Soteris A. Kalogirou, (2004). *Solar thermal collectors and applications.* Progress in Energy and Combustion Science, Volume 30, Issue 3, Pages 231-295.

[51] Henrik Lund, Sven Werner, Robin Wiltshire, Svend Svendsen, Jan Eric Thorsen, Frede Hvelplund, Brian Vad Mathiesen, (2014). *4th Generation District Heating (4GDH) Integrating smart thermal grids into future sustainable energy systems.* Energy, Volume 68, Pages 1-11.

[52] Ioannis Vallios, Theocharis Tsoutsos, George Papadakis, (2009). *Design of biomass district heating systems.* Biomass and Bioenergy, Volume 33, Issue 4, Pages 659-678.

[53] Jelena Ziemele, Roberts Kalnins, Girts Vigants, Edgars Vigants, Ivars Veidenbergs, (2018). *Evaluation of the industrial waste heat potential for its recovery and integration into a fourth generation district heating system.* Energy Procedia, Volume 147, Pages 315-321.

[54] Tobiasen Lasse, Kamuk Bettina, Meyers, Robert A., (2012). *Waste-to-Energywaste-to-energy (WTE)for District Heatingwaste-to-energy (WTE)for district heating and power generation.* Encyclopedia of Sustainability Science and Technology, Springer New York.

[55] Lo Russo Stefano, (2019). *Slides of course Geothermal Energy.* Polytechnic of Turin.

[56] Santarelli Massimo, (2020). *Slides of course Polygeneration and advanced energy systems.* Polytechnic of Turin.

[57] Chicco Gianfranco, (2020). *Slides of course Smart electricity systems.* Polytechnic of Turin.

[58] Li, H., Svendsen, S., Gudmundsson, O., Kuosa, M., Rämä, M., Sipilä, K., ... Bevilacqua, C. (2017). *Future low temperature district heating design guidebook: Final Report of IEA DHC Annex TS1. Low Temperature District Heating for Future Energy Systems.* International Energy Agency.

[59] Euroheat, (2016). *Energy Distribution: District Heating and Cooling - DHC.*

[60] Joel Hernández-Santoyo, Augusto Sánchez-Cifuentes, (2003). *Trigeneration: an alternative for energy savings* Applied Energy, Volume 76, Issues 1–3, Pages 219-227.

[61] Gambini M, Vellini M, Stilo T, Manno M, Bellocchi S., (2019). *High-Efficiency Cogeneration Systems: The Case of the Paper Industry in Italy.* Energies, Volume 12, Number 3, Article number 335.

[62] Elisa Guelpa, Vittorio Verda, (2019). *Thermal energy storage in district heating and cooling systems: A review.* Applied Energy, Volume 252.

[63] M. Badami, F. Camillieri, A. Portoraro, E. Vigliani, (2014). *Energetic and economic assessment of cogeneration plants: A comparative design and experimental condition study.* Energy, Volume 71, Pages 255-262.

[64] Council of European Parliament, (2014). *Directive 2004/8/EC of the European Parliament and of the Council of the 11 February 2004 on the promotion of cogeneration based on the useful heat demand in the internal energy market and amending.* Directive 92/42/EEC, Official Journal of the European Union.

[65] ExpoClima, (2018). *Caratteristiche delle reti di teleriscaldamento.* `https://www.expoclima.net/special/146/il_teleriscaldamento_e_teleraffrescamento/caratteristiche_delle_reti_di_teleriscaldamento.htm`

[66] Alberto Poggio Tecnoapi, Giuseppe Serratì, Luciano Filippi, Cinzia Maga, Livia Manzone, Paola Benedetti, (2006). *Studio sul TELERISCALDAMENTO IN PROVINCIA DI TORINO. Stato di fatto e potenzialità di sviluppo.* Provincia di Torino.

[67] Dominik Rutz, Carlo Winterscheid, Thomas Pauschinger, Sebastian Grimm, Tobias Roth, Borna Doračić, Gillian Dyer, Thomas A. Østergaard, Reto Hummelshøj, (2019). *Upgrading the performance of district heating networks. Technical and non-technical approaches. A Handbook.* WIP Renewable Energies, Munich, Germany.

[68] Etienne Saloux, José A. Candanedo, (2018). *Forecasting District Heating Demand using Machine Learning Alrgorithms.* Energy Procedia, Volume 149, Pages 59-68.

[69] Christian Johansson, Markus Bergkvist, Davy Geysen, Oscar De Somer, Niklas Lavesson, Dirk Vanhoudt, (2017). *Operational Demand Forecasting In District Heating Systems Using Ensembles Of Online Machine Learning Algorithms.* Energy Procedia, Volume 116, Pages 208-216.

[70] Davide Caputo, Francesco Grimaccia, Member, IEEE, Marco Mussetta, Riccardo E. Zich, (2010). *Photovoltaic Plants Predictive Model by means of ANN trained by a Hybrid Evolutionary Algorithm.* The 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, pp. 1-6.

[71] Suryansh S., (2018). *Genetic Algorithms + Neural Networks = Best of Both Worlds.* Towards Data Science. `https://towardsdatascience.com/gas-and-nns-6a41f1e8146d`