CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

Post-Processing Air Temperature Weather Forecast using Artificial Neural Networks with measurements from Meteorological Stations

Gustavo Antonio Luna de Araujo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientador:
Fabio Augusto de Alcantara Andrade, PhD

Rio de Janeiro,
Agosto, 2022

Post-Processing Air Temperature Weather Forecast using Artificial Neural Networks with measurements from Meteorological Stations

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Gustavo Antonio Luna de Araujo

Aprovada por:

Presidente, Fabio Augusto de Alcantara Andrade, Ph.D. (orientador)

Eduardo Bezerra da Silva, Ph.D.

Diego Barreto Haddad, Ph.D.

Thomas Spengler, Ph.D.

Rio de Janeiro,

Agosto, 2022

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

A663 Araujo, Gustavo Antonio Luna de

Post-processing air temperature weather forecast using artificial neural networks with measurements from meteorological stations / Gustavo Antonio Luna de Araujo – 2022.

xi, 42f. il. : il. (algumas color.), tabs. ; enc.

Dissertação (Mestrado), Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2022.

Bibliografia: f, 40–42

Orientador : Fábio Augusto de Alcântara Andrade

1. Inteligência artificial. 2. Redes neurais (Computação). 3. Aprendizado do computador. I. Andrade, Fabio Augusto de Alcantara (Orient.). II. Título.

CDD 006.3

Elaborada pelo bibliotecário Leandro Mota de Menezes CRB-7/5281

ACKNOWLEDGMENTS

God, thank You for showing that the impossible is just a point of view.

Paula, my wife, thank you for dreaming my dreams with me, no matter how crazy they are.

Maria, my daughter, thank you for filling me with joy with your smiles.

Gustavo Antonio Luna de Araujo

"All we have to do is decide what to do with the time that is given to us."

Gandalf

RESUMO

Post-Processing Air Temperature Weather Forecast using Artificial Neural Networks with measurements from Meteorological Stations

Gustavo Antonio Luna de Araujo

Orientadores:

Prof. Fábio Augusto de Alcântara Andrade, PhD.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Em um tempo de eventos climáticos extremos mais frequentes, uma previsão do tempo acurada pode trazer benefícios que vão além do planejamento da melhor posição de equipamentos que usam as forças da natureza para produzir energia elétrica, mas também contribui para segurança da vida humana. Dos vários institutos que se dedicam ao estudo e previsão do tempo ao redor do globo, o Instituto de Meteorologia da Noruega é o órgão responsável pelos assuntos relacionados ao clima naquele país, e entre as suas contribuições está a previsão numérica do tempo apresentada no formato de uma grade de 2,5 km. De modo a realizar um pós processamento que melhore a resolução dessa previsão e diminua o erro da mesma, o Instituto desenvolveu uma ferramenta chamada GRIDPP que realiza a redução da escala para 1 km. O presente trabalho objetivou melhorar a abordagem de pós-processamento atual da temperatura do ar usando Redes Neurais valendo-se de medições de estações meteorológicas. Duas arquiteturas de redes neurais foram testadas: uma Multilayer Perceptron e uma Convolutional Neural Network. Ambas as arquiteturas conseguiram alcançar uma taxa de erro menor do que as abordagens originais. Dentre as amostras de teste onde o erro do GRIDPP foi maior que 3°C, sendo 3°C um limite crítico definido pelo Instituto, a solução proposta conseguiu um erro menor em 74.8% dos casos.

Redes Neurais, Aprendizado Profundo, Meteorologia, Pós-Processamento.

ABSTRACT

Post-Processing Air Temperature Weather Forecast using Artificial Neural Networks with measurements from Meteorological Stations

Gustavo Antonio Luna de Araujo

Advisors:

Prof. Fábio Augusto de Alcântara Andrade, PhD.

Abstract of Master Dissertation submitted to Electrical Engineering Graduate Program, Federal Center of Tecnologic Education Celso Suckow da Fonseca, CEFET/RJ, as partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

In a time of more frequent extreme weather events, an accurate weather forecast can bring benefits that go beyond planning the best position for equipment that uses the forces of nature to produce electricity, but also contribute to the safety of human life. Among the various institutes dedicated to the study and forecasting of the weather around the globe, the Institute of Meteorology of Norway is the one responsible for matters related to the climate in that country, and among its contributions is the numerical weather forecast presented in the format of a 2.5 km grid. Intending to carry out a post-processing that improves the resolution of this forecast and reduces its error, the Institute developed a tool called GRIDPP that downscale the grid to 1 km. The present work aimed to improve the current post-processing approach to air temperature using Neural Networks with measurements from meteorological stations. Two neural network architectures were tested: a Multilayer Perceptron and a Convolutional Neural Network. Both architectures managed to achieve a lower error rate than the original approaches. Among the test samples where the GRIDPP error was greater than 3°C, with 3°C being a critical limit defined by the Institute, the proposed solution achieved a lower error in 74.8% of the cases.

Neural Networks, Deep Learning, Meteorology, Post-Processing.

Contents

I Introduction **1**

I.1 Initial Considerations and Motivation 1

I.2 Objective and Contributions 3

I.3 Work Organization 4

II Background and Related Works **5**

II.1 Numerical Weather Prediction 5

II.2 Neural Networks 7

II.2.1 Perceptron and Multilayer Perceptron 7

II.2.2 Convolutional Neural Networks 9

II.2.3 Training Process 12

II.2.4 Optimizers 13

II.2.5 Metrics 15

II.2.6 Gradient Issues 16

II.3 Related Works 16

III Proposed Methodology **18**

III.1 Data Acquisition 18

III.1.1 Get List of Forecasts 19

III.1.2 Get Stations 19

III.1.3 Get All Forecasts 20

III.1.4 Get All Observations 22

III.1.5 ETL 22

III.2 Preprocessing 23

III.2.1 Exploratory Data Analysis 23

III.2.2 Data Transforming 25

III.3 Neural Network 26

III.3.1 Multi Layer Perceptron 27

III.3.2 Convolutional Neural Network 28

IV Results and Discussion **29**

IV.1 Experiment Environment 29

IV.2 Benchmarks 29

IV.3 Training Results 29

IV.3.1 Multilayer Perceptron 31

IV.3.2 Convolutional Neural Network 32

IV.4 Discussion 34

V Conclusions and Future Work **37**

V.1 Conclusions 37

V.2 Future Propositions 39

Bibliographic References 40

List of Tables

IV.1 Benchmarks 29
IV.2 Test set RMSE on MLP models. 31
IV.3 Test set RMSE on CNN models 33
IV.4 Performances on Test set 35

List of Figures

I.1 Modern Renewable Energy Production. 2

II.1 Perceptron. 8
II.2 MLP. 8
II.3 Convolutional Layer Representation. 9
II.4 Gradient Descent pitfalls. 13

III.1 Data Acquisition Pipeline. 19
III.2 Norway Station's 2D Histogram. 24
III.3 Norway Station's. 24
III.4 Temperature distribution. 24
III.5 Larger GRIDPP Errors. 25
III.6 Modelling Time with Sine Function. 26
III.7 Multilayer Perceptron. 27
III.8 Convolutional Neural Network. 28

IV.1 MEPS - RMSE Distribution. 30
IV.2 GRIDPP - RMSE Distribution. 30
IV.3 MLP - Learning Curve. 31
IV.4 Comparing Errors of MLP with the Benchmark. 32
IV.5 GRIDPP error higher than 3°C. 32
IV.6 MLP error in cases when GRIDPP error is higher than 3°C. 33
IV.7 CNN - Learning Curve. 33
IV.8 Comparing Errors of CNN with the Benchmark. 34
IV.9 CNN error in cases when GRIDPP error is higher than 3°C. 34
IV.10 Comparing RMSE by hours ahead. 35

Abbreviation

CNN	Convolutional Neural Network 9, 11, 16, 26, 28, 32, 33, 35, 39
COSMO	Consortium For Small-scale Modeling 6
CSV	Comma Separate Value 20, 22, 37
DOI	Digital Object Identifier 38
ETL	Extract-Tranform-Load 18, 19
FC	Full Connected 7, 11, 27, 28
GNN	Graph Neural Network 17
GRIDPP	Gridded Post-Processor 3, 6, 7, 18, 19, 20, 22, 23, 29, 31, 34, 35, 37, 38, 39
INMET	Instituto Nacional De Meteorologia 2
INPE	Instituto Nacional De Pesquisas Espaciais 2
IOT	Internet Of Things 39
MAE	Mean Absolute Error 15
MEPS	MetCoOp Ensemble Prediction System 3, 6, 18, 19, 22, 29, 31, 32, 37, 38
MET	Norwegian Meteorological Institute 2, 3, 4, 6, 16, 18, 37
METCOOP	Meterological Cooperation On Operational Numeric Weather Prediction 3, 6
MLP	Multilayer Perceptron 7, 8, 26, 28, 31, 32, 33, 35
MSE	Mean Square Error 15, 26, 29
NETCDF	Network Common Data Form 20, 21, 22
NN	Neural Network 3, 4, 7, 11, 12, 14, 16, 18, 23, 25, 29, 35, 37, 38, 39
NOAA	National Oceanic And Atmospheric Administration 2
NWP	Numeric Weather Prediction 2, 3, 5, 6
RMSE	Root Mean Square Error 15, 29, 31, 35
RNN	Recurrent Neural Network 39
UWC-EAST	United Weather Centers - East 3, 6
WMO	World Meteorological Organization 2

Chapter I Introduction

I.1 Initial Considerations and Motivation

> The world has changed.
> I see it in the water.
> I feel it in the Earth.
> I smell it in the air.
> Much that once was is lost,
> For none now live who remember it.
>
> The Lord of the Rings
> J. R. R. Tolkien

Back in 2014 (1), a heavy blizzard paralyses Washington. A particularly harsh winter has hit this region of the country, and according to the American press, 800,000 houses and businesses would be without electricity, and a few dozen dead are estimated. In 2021 (2), Iraqi authorities announced a holiday, including the capital Baghdad, because of the extreme heat. At the same time, in British Columbia, Canadian authorities estimated nearly 800 deaths between June 25 and July 1, 500 more than normal for that period and likely heat-related.

More recently, in the mountainous region of the state of Rio de Janeiro in Brazil, in the historic city of Petrópolis, two massive rains hit the city and left hundreds of people dead and nearly one thousand homeless in less than 1 month. Despite the rain warning, the amount of rain had not been accurately predicted (3).

"The world has changed" (4). In fact, the world has changed and the climatic relationships that the world presents us have also changed. At a time when weather events have been more intense and unexpected, the study of climate has become more important than ever. In addition to studying how the climate is changing and its different behaviours, an accurate weather forecast has become gold.

For a long time, humans have been trying to make accurate forecasts about the weather. From weather stations to satellites, today we have fine systems around the globe that can make accurate forecasts and modern sensors with reliable measures of the weather. And now, weather prediction can be used for navigation safety by the sea and preparation for extreme weather events like hurricanes and blizzards.

But the human relationship with the climate has not just turned on the bad side. With climate change, human beings are also learning how to use nature itself as a source of clean and renewable energy. As showed in the Figure I.1, in the last ten years, the production of electrical energy using the wind and solar powers significantly increased.

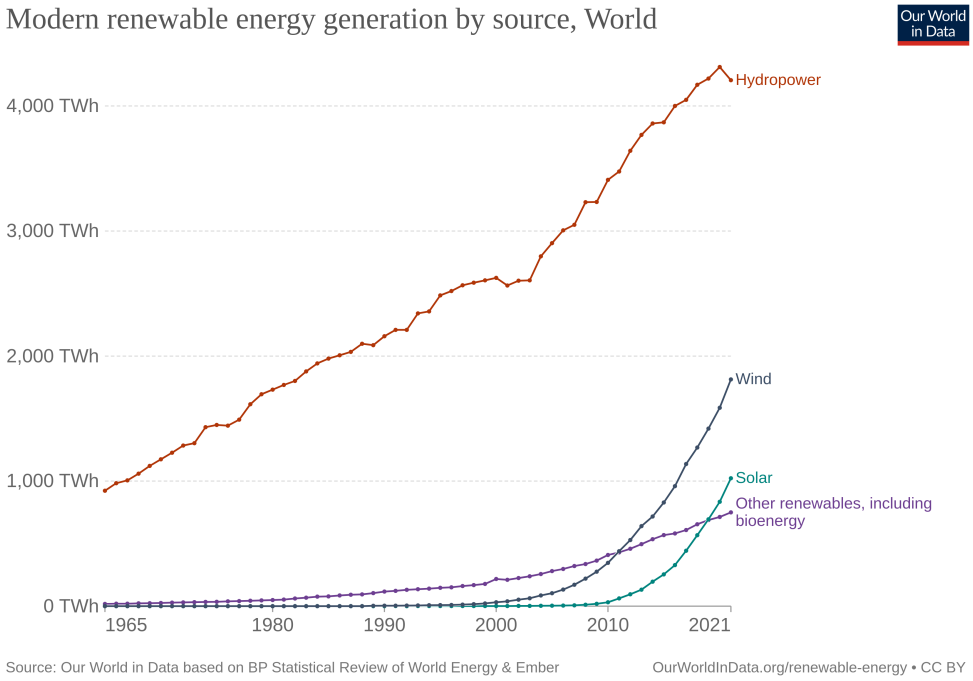


Figure I.1: Modern Renewable Energy Production.
Published in https://ourworldindata.org/renewable-energy *(accessed: 28.05.2022)*

Following this path, having a good idea of the climate and weather of a specific region has become sorely necessary for planning when using electricity generated through natural means such as the wind, the sun or the tides. So, having a good idea of weather changes has become of great importance.

Due to the importance of studying the climate and providing weather forecast services, countries usually have official institutes dedicated to this subject. In Brazil, it is possible to cite the Instituto Nacional de Meteorologia (INMET) and the Instituto Nacional de Pesquisas Espaciais (INPE). France has the Météo-France. In the United States, they have the National Oceanic and Atmospheric Administration (NOAA). In Germany, Deutscher Wetterdienst. Even the United Nations has a meteorological agency called World Meteorological Organization (WMO).

Norway specific case, for having a more extreme climate due to the high latitudes, having an accurate weather forecast is very important, because it is not unusual having ice on the roads or strong winds, and these elements can threaten security.

The Norwegian Meteorological Institute (MET), located in Oslo, is the institute in Norway in charge of weather predictions, monitoring the climate and conducting research since 1866 (5). Because of this, among others responsibilities, the MET is in charge to produce the Numeric Weather Prediction (NWP)

in Norway.

And for this purpose MET has joined the Meterological Cooperation on Operational Numeric Weather Prediction (MetCoOp) that is a cooperation between the Finnish Meteorological Institute, MET Norway and Swedish Meteorological and Hydrological Institute. The collaboration between MET and the Swedish Meteorological and Hydrological Institute, MetCoOp, started in 2010. Since 2014, the departments have benefited from working together in the weather forecast. In 2017, the Finnish Meteorological Institute also joined MetCoOp. From 2022, the Baltic States Estonia, Latvia and Lithuania will also join, and it might then be called United Weather Centers - East (UWC-East).

MET runs two different weather forecasting systems: The MetCoOp Ensemble Prediction System (MEPS) and the AROME-Artic. MEPS covers Scandinavia and Finland and provides 30 ensemble members every six hours with lead times out to 61 hours and a spatial grid spacing of 2.5 km, and the AROME-Artic covers the Svalbard area and provides 1 single deterministic run for 66 h lead time and 2.5 km grid spacing (6).

Despite all the efforts made by the MET, the meteorological models calculated by the MET come up against problems common to all models: forecast resolution and prediction errors.

Seeking to solve these initial problems, the MET developed a post-process system called Gridded Post-Processor (GRIDPP) that can downscale the MEPS grid of 2.5 km to a grid of 1 km using interpolation and another methods that will be presented later in this work.

Despite the work already developed by MET for post-processing the meteorological data and its excellent results, it is of interest to be able to improve the current post-processing and minimizing the error between the NWP and the observed weather. Rather than replacing the current approach, this work intends to join efforts and propose a mixed approach using the current methods added to Neural Network (NN).

MET cooperates not only with the data made available on its platforms and API, but also with feedbacks and advices as the study was developed.

I.2 Objective and Contributions

The general objective of this work is to understand if it is possible to use a NN approach to improve the meteorological data post-processing, reducing the error between the weather prediction and the measured temperature and use non-standard sources of weather observation to improve the post-processing performance. The contributions of this work are the following:

- **Development of a data pipeline that makes possible to train and evaluate Machine and Deep Learning models for this kind of problem**
- **Development and Test of different of NN architectures capable of refining the weather**

forecast

I.3 Work Organization

Chapter II details the theoretical background necessary for the understanding of this work. This chapter will describe the actual approach of MET

Chapter III presents the development of a data pipeline and its details, and the proposed NN.

Chapter IV presents the results of the research.

Chapter V presents the conclusions and final considerations.

Chapter II Background and Related Works

II.1 Numerical Weather Prediction

The history of NWP began in 1904, when Vilhem Bjerknes said the atmosphere's future state could be predicted by integrating the differential equations that governed its behaviour. Bjerknes proposed use as initial conditions of the differential equations the data that describe an observed state of the atmosphere, more precisely, observations from meteorological stations, also called by the acronym SYNOP[1]. The scientist Lewis Fry Richardson was the first who actually solved that equation using meteorological data from the central region of Germany. The result was not perfect, the predicted pressure variation was much greater than that observed, but this effort was useful to identify several issues that would be faced on the road of weather forecast (7, 8).

Richardson's experiment showed that the data used was not enough to represent the actual atmosphere's state. It was possible to perceive that the process required a large amount of calculation and that calculation should be fast, and minor errors at the beginning of the process could be turned into huge errors in the final forecasts.

But only in 1950 in the United States of America, scientists made the first weather forecast for one day ahead using a computer, more precisely the famous Electronic Numerical Integrator and Computer, better known as ENIAC. Five years later, the United States started to calculate the weather forecast continuously using computers.

Today, it is possible to divide the weather prediction in three main steps: Analysis, Forecast and Post-processing.

- The Analysis step is necessary because meteorological observations are not regular, and mathematical models require the data provided to be uniform, that is, with regular horizontal spacing over the globe, so mathematical and computational procedures are used so that the data have the necessary format.

- In the Forecast stage, the goal is to use the numerical models to solve the equations that describe the behaviour of the atmosphere. This step is often solved using a grid point form. To minimize the effects of atmospheric chaos and considering that atmospheric measurements are not fully

[1]The acronym METAR represent the airport weather stations, the acronym SHIP refers to weather stations embeds on ships, and SYNOP represents the other conventional sources

accurate, several simulations are performed for the same periods with small changes in the initial conditions of the atmosphere, finally the average of the results is considered.

- The post-processing, which is the focus of this work, consists in using the output of the numerical models either for study or for assembling meteorological bulletins that will be used by the final user. Post-processing is a common step that is taken in several fields, and meteorology is not an exception. Vannnistem (9) says that statistical post-processing techniques are nowadays key components of the forecasting suites in many National Meteorological Services, with, for most of them, the objective of correcting the impact of different types of errors on the forecasts.

In Norway, MET is the public institute responsible for weather prediction, monitoring the climate and conducting research since 1866 (5). Because of this, MET is in charge to provide the NWP on that country. So, in 2010 MET started to cooperate with others meteorological institutes in a fellowship called MetCoOp and since 2014 these departments have benefited from working together in the weather forecast. Initially, the partnership was between MET and Swedish Meteorological and Hydrological Institute only, but the Finnish Meteorological Institute also joined the group in 2017. From 2022, the Baltic States Estonia, Latvia and Lithuania will also join, it might then be called UWC-East.

Therefore, MET produces two different weather forecast models:

1. The **MetCoOp Ensemble Prediction System**, also known as MEPS, that covers Scandinavia and Finland and provides 30 ensemble members every six hours with lead times out to 61 hours and a spatial grid spacing of 2.5 km.

2. The **AROME-Artic** that covers the Svalbard area and provides 1 single deterministic run for 66 h lead time and 2.5 km grid spacing (6). The AROME-Artic will not be used in this dissertation.

With the aim to improve the forecast result, MET started to develop the Gridded Post-Processor, known as GRIDPP, that is a tool for post-processing gridded weather forecasts (10). This tool can downscale the MEPS grid of 2.5 km to a grid of 1 km, and also implements different improvements on many parameters, such as altitude corrections on the air temperature parameter, and bias correction using sensor measurements. It should be noted that the primary resolution of MEPS would already be considered considerably good compared to other regional models such as Consortium for Small-scale Modeling (COSMO) which has a 7 km or a 2.8 km grid and BRAMS with a 5 km grid, both used officially in Brazil (11, 12, 13).

By the repository definition (10), GRIDPP is a tool for performing statistical post-processing of gridded weather forecasts written in C++ but offers Python bindings to the functions in the library. As main features, we have the following:

- Methods for downscaling a forecast from a coarse grid to a fine grid.

- Methods for calibrating a downscaled grid.
- Neighbourhood methods to compute neighbourhood min, mean, max and any quantile.
- Functions for diagnosing variables
- Data assimilation methods such as optimal interpolation to merge observations and gridded forecasts.
- Efficient data structures for nearest location lookup in a vector or grid of locations.

On the current version of GRIDPP, we find three methods for downscaling a grid. First, that is the Nearest Neighbour Method that uses the closest neighbour for each point in the fine grid. Second, that is the Bilinear Method that uses the four nearest neighbours and interpolates bilinearly between those. Lastly, that is the Gradient-based Method that is based on the knowledge that some meteorological variables such as temperature change predictably with elevation, so when a high resolution topography is available, coarse resolution models can be downscaled by fitting the variable to the new topography.

II.2 Neural Networks

II.2.1 Perceptron and Multilayer Perceptron

Originally proposed by Warren McCulloch and Walter Pitts (14, 15), the idea of an Artificial Neuron, and after the idea of a NN, were inspired by biology in order to become a mathematical model based on the functioning of the human brain. Since the Frank Rosenblatt Perceptron represented on Equation II.1 and on Figure II.1 where x represents the neuron's input and W their weights with the step function defined on Equation II.2, this study field had some ups and downs. In 1969 (16), it was proved the limitations of using 1 layer perceptron's against the XOR problem. But some important propositions such as the using of several layers of fully connected neurons giving birth to the Multilayer Perceptron (MLP), represented in Equation II.3, took the NN back to the game. In this case, a MLP, based on Figure II.2 would consist of several layers of Perceptrons, where one layer will be fully connected to the next layer of Perceptrons. Because of this particular design, the MLP layers became known as Full Connected (FC) layers, or Dense layers.

$$h_w(x) = step(x^T W) \tag{II.1}$$

$$y = \begin{cases} 0, \text{ if } x < 0 \\ 1, \text{ if } x \geq 0 \end{cases} \tag{II.2}$$

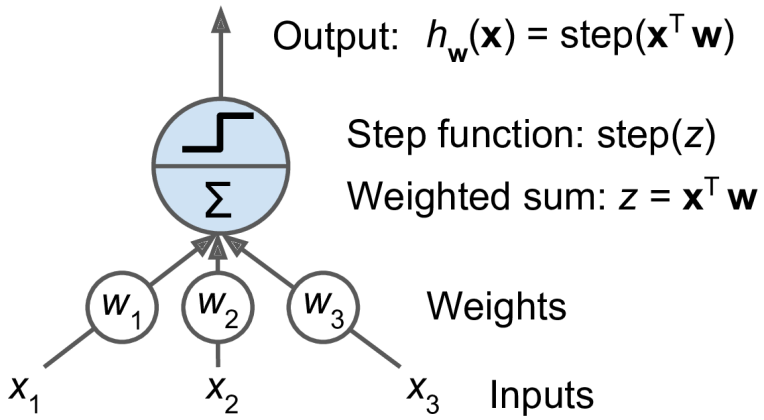


Figure II.1: Perceptron.
Taken from (15)

$$h_{w,b}(x) = \phi(x^T W + b) \tag{II.3}$$

In the MLP Equation II.3, x continues to represent a matrix of inputs and W continues to represent the neuron's weights, but some important changes happen like the bias creation represented by the letter b on the equation, and changing the step function originally used by Rosenblatt for the activation function concept represented here by the Greek symbol ϕ. This concept of MLP was very interesting because how the output of a neuron's layers becomes the input of a new neuron's layers, this make possible solve the classical XOR problem.

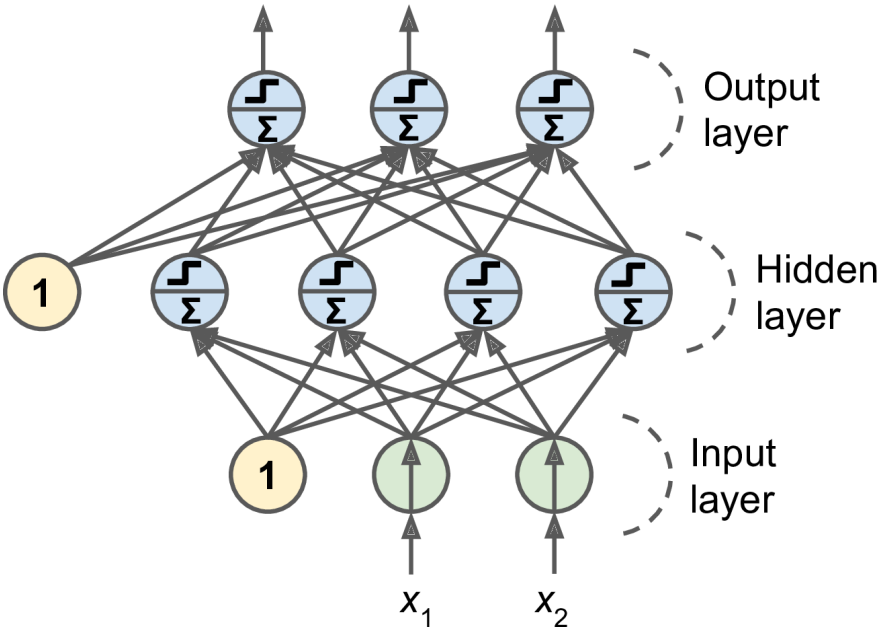


Figure II.2: MLP.
Taken from (15)

II.2.2 Convolutional Neural Networks

Hubel e Wiesel (17, 18, 19, 15) performed some experiments that show some information about the visual cortex's structure. More precisely, the neurons of the visual cortex only react when stimulated by some limited region of the visual field. It was also noticed that different neurons had different receptive fields, so that some neurons could respond to larger patterns and others only to smaller details. The combination of these characteristics constitutes an architecture capable of recognizing different patterns of different images of different sizes by our brain.

These discoveries inspired what we know as Convolutional Neural Network (CNN), using Convolutional Layers that could extract features of some specific part of the data, that is widely used in the image recognition field. The main block in the CNN architecture is the convolutional layer, here the convolutional neurons of the first layer are not connected to every pixel for the input data, but only to a specific region as represented on Figure II.3.

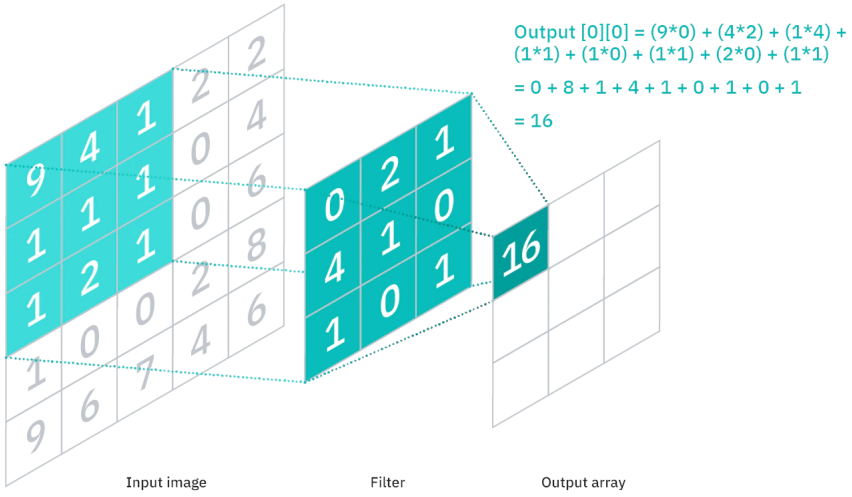


Figure II.3: Convolutional Layer Representation.
Published in *https://www.ibm.com/cloud/learn/convolutional-neural-networks (accessed: 25.05.2022)*

The convolutional neural weights are known as filter or kernel, and it is the matrix in the middle of Figure II.3. The convolutional kernel will slide over the input in a pace called stride, multiplying their content by the input data to generate the output array, as represented in Equation II.4.

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f'_n-1} x_{i',j',k'} \cdot w_{u,v,k',k} \text{ with } \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases} \tag{II.4}$$

It is possible to apply Equation II.4 in example II.5 using a stride value equals to 1.

$$\begin{bmatrix} 3 & 1 & 5 & 4 \\ 11 & 3 & 5 & 11 \\ 23 & 6 & 2 & 2 \\ 5 & 3 & 55 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 6 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 3 & 1 & 5 \\ 11 & 3 & 5 \\ 23 & 6 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 6 & 0 \end{bmatrix} = \begin{cases} 3 \times 1 + 1 \times 0 + \\ 5 \times 1 + 11 \times 0 + \\ 3 \times 2 + 5 \times 1 + \\ 32 \times 0 + 6 \times 6 + \\ 2 \times 0 \end{cases} = 55$$

$$\begin{bmatrix} 1 & 5 & 4 \\ 3 & 5 & 11 \\ 6 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 6 & 0 \end{bmatrix} = \begin{cases} 1 \times 1 + 5 \times 0 + \\ 4 \times 1 + 3 \times 0 + \\ 5 \times 2 + 11 \times 1 + \\ 6 \times 0 + 2 \times 6 + \\ 2 \times 0 \end{cases} = 41$$

$$\begin{bmatrix} 11 & 3 & 5 \\ 23 & 6 & 2 \\ 5 & 3 & 55 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 6 & 0 \end{bmatrix} = \begin{cases} 11 \times 1 + 3 \times 0 + \\ 5 \times 1 + 23 \times 0 + \\ 6 \times 2 + 2 \times 1 + \\ 5 \times 0 + 3 \times 6 + \\ 55 \times 0 \end{cases} = 48$$

$$\begin{bmatrix} 3 & 5 & 11 \\ 6 & 2 & 2 \\ 3 & 55 & 5 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 6 & 0 \end{bmatrix} = \begin{cases} 3 \times 1 + 5 \times 0 + \\ 11 \times 1 + 6 \times 0 + \\ 2 \times 2 + 2 \times 1 + \\ 3 \times 0 + 55 \times 6 + \\ 5 \times 0 \end{cases} = 350$$

$$\begin{bmatrix} 3 & 1 & 5 & 4 \\ 11 & 3 & 5 & 11 \\ 23 & 6 & 2 & 2 \\ 5 & 3 & 55 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 6 & 0 \end{bmatrix} = \begin{bmatrix} 55 & 41 \\ 48 & 350 \end{bmatrix} \tag{II.5}$$

The kernel represented on the figure above is a 2 dimensional kernel because it is represented by a 2D matrix, but there also exist 1 dimension kernels and 3 dimensions kernels design for inputs that have 1 dimension and 3 dimensions, respectively. With the goal to use all available data from the input, whether the model input or from the previous layer, an optional parameter called padding can be used.

This tool consists of adding a margin around the input matrix, generally with values equals to zero, for the combination of kernel size and stride to not leave any element out, as represented in the Equation II.6 using a padding size equal to 1.

$$\begin{bmatrix} 4 & 2 & 5 & 4 & 4 \\ 11 & 3 & 5 & 7 & 11 \\ 23 & 6 & 5 & 2 & 2 \\ 2 & 2 & 8 & 3 & 6 \\ 5 & 3 & 55 & 43 & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 2 & 5 & 4 & 4 & 0 \\ 0 & 11 & 3 & 5 & 7 & 11 & 0 \\ 0 & 23 & 6 & 5 & 2 & 2 & 0 \\ 0 & 2 & 2 & 8 & 3 & 6 & 0 \\ 0 & 5 & 3 & 55 & 43 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{II.6}$$

Intending to reduce the computational cost of the CNN, there is another layer that is quite common in the matter of the convolutional architecture: the Pooling layer. The Pooling layer is a quite simple concept, and it consists in subsample a previous layer to reduce redundant information. As the Convolutional layers, it has to define a size, a stride and padding, but only a simple operation is performed on the defined window, and that operation defines the name of the Pooling layer. The most common Pooling layers are the Max Pooling layer, which gets the maximum value, and the Average Pooling layer, which gets the mean value from the input matrix. In the Equation II.7, it is possible to see an example of a Max Pooling Layer with kernel size equals to 2 and stride equals to 1.

$$\begin{bmatrix} 4 & 2 & 5 & 4 & 4 \\ 11 & 3 & 5 & 7 & 11 \\ 23 & 6 & 5 & 2 & 2 \\ 2 & 2 & 8 & 3 & 6 \\ 5 & 3 & 55 & 43 & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} 11 & 5 & 7 & 11 \\ 23 & 6 & 7 & 11 \\ 23 & 8 & 8 & 6 \\ 3 & 55 & 55 & 43 \end{bmatrix} \tag{II.7}$$

Often, after some convolutional layers interspersed with pooling layers, FC layers are added in the output model to perform tasks like classification or regression.

LeCun (20) proposed in 1998 the famous LeNet-5 architecture that uses the CNN concept, and was popularly used to recognized handwritten characters. After LeNet-5, several new architectures were proposed like the famous ResNet (21) and VGG-16 (22) in 2015 and Xception (23) in 2017, that achieved great results in image recognition but also present a very complex convolutional architecture combining several layers. All this complexity raised the training effort to train these NN, as a result, a practice known as transfer learning (24) became very popular. This method consists in using pretrained layers to build a Deep Learning model, for example, it is possible to use the trained weights of a well-trained VGG-16 to build a new model, either to perform the original VGG-16 classifications, or to fine tune

replacing the dense layers to recognize new objects. By doing this, training new NN for a specific task became more feasible, because the computational power and time required to train a complex network largely decreases.

II.2.3 Training Process

The new proposal of a deep network with hidden layers, the layers between the input and the output, had an issue, how will be possible to train the hidden layers weights? Almost twenty years after the discredit of the Perceptron, a new method called backpropagation (25) came to make it possible to train NN with multiple layers. The method consists in two steps:

1. **Feed Forward:** Once given the input data, all matrix multiplications are applied to achieve an output, saving all the intermediate results because they are needed for the backward pass. Then, the error is calculated with the loss function chosen.

2. **Back Forward:** Using the chain rule in the loss function, the algorithm computes how much each connection in the layer below contribute to this error. After this, realize a Gradient Descent step to update the NN weights.

Considering the need to derive the functions that compose the NN, the classical step function originally used by Rosenblatt was replaced by the sigmoid function, represented on Equation II.8, and that implementation became very popular, but were not the only available. Functions as the $tanh$ (Equation II.9) and the $ReLU$ (Equation II.10) work well with the backpropagation method.

$$Sigmoid(x) = \frac{1}{1 + e^{-1}} \tag{II.8}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{II.9}$$

$$ReLU(x) = max(0, x) \tag{II.10}$$

In the training process, a specific point requires attention, the overfitting. Complex models such as NN can detect subtle patterns in the data, but if the training set is not big enough, or noisy, the model probably will start to detect the noise itself (15), in this case the model performs great on the training data, but it does not generalize well. In general, overfitting happens when it is chosen a model that is too complex relative to the amount of data.

The existence of overfitting can be noticed through the difference between performances between the training and test sets. A common way of perceiving overfitting during the training phase is to use a validation set that is presented to the model during this phase with the purpose of measuring performance

on non-training data. If we look at the learning curves during this process, it can be noticed that while the training set error continues to decrease, the validation set error starts to increase in case of overfitting after a certain point. Among the possible solutions to avoid overfitting are simplifying the model used, increasing the training data and removing outliers and other errors from the training data.

II.2.4 Optimizers

Gradient Descent

The Gradient Descent is a generic algorithm designed for optimization, capable of finding local optimal solutions (15). The idea consists in adjusting parameters iteratively to minimize a cost function, and for this the algorithm measures the gradient of the error function when using the vector parameter θ and takes a step into the descending gradient direction, as seen in Equation II.11. To define how big will be the step that will be taken, it is possible to define η as the learning rate that will be multiplied by the gradient times cost function, represented by $J(\theta)$.

$$\theta \leftarrow \theta - \eta \nabla_\theta J(\theta) \tag{II.11}$$

Some notes are important to take, if the learning rate η were too small, the algorithm may take too long to converge, but if the η were too large, it may never converge to an optimal solution.

Unfortunately, not all cost functions are convex like a bowl which would allow the gradient to go directly to the global minimum. FigureII.4 shows if the random initialization starts the algorithm on the left, then it will converge to a local minimum, that is not as good as the global minimum.

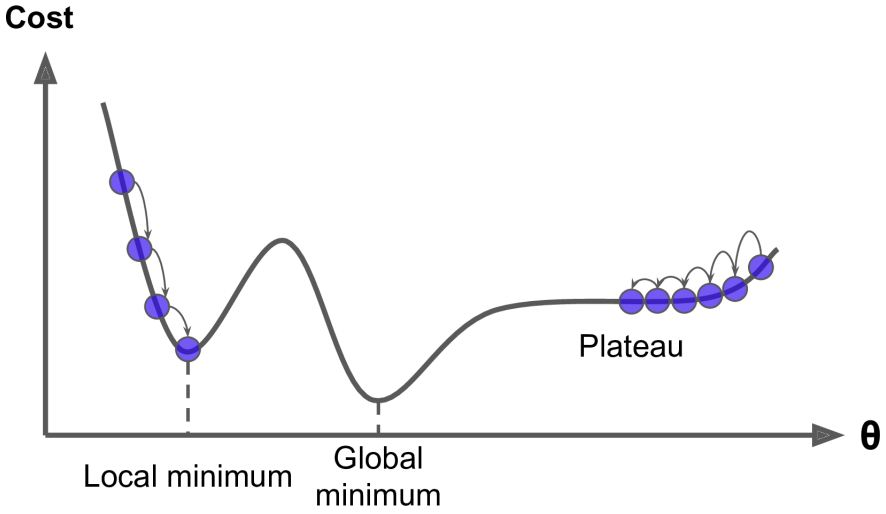


Figure II.4: Gradient Descent pitfalls.
Taken from (15)

Momentum Optimization

"Foot orienteering is an endurance sport which involves a huge mental element. There is no marked route – the orienteer must navigate with map and compass while running" (26). In an orienteering race, the athlete do not run with the same intensity during the entire race. First, it is necessary to look at the map and understand where the person is, this step is usually made at low speed. After, when the athlete defines the route he is going to take, he increases his speed. Finally, when the athlete is sure about the path and sees clearly his objective, he increases his speed even more, but when he is just 2 meters from his control point, he slows down to stop at the control point and register his path, and after start all over again. The Momentum Optimization proposed by Boris Polyak (27) works a little like that, while the Gradient Descent takes regular steps during the optimization process, the Momentum Optimization uses the gradient to set the increase of the speed, the acceleration, represented in the Equation II.12 by the letter m, based on the previous gradients. To prevent the momentum from growing too big, the parameter β, called *momentum*, is used to work like a friction force.

$$\begin{cases} m \leftarrow \beta m - \eta \nabla_\theta J(\theta) \\ \theta \leftarrow \theta + m \end{cases} \tag{II.12}$$

RMSprop

The RMSProp Algorithm (28), represented in the Equation II.13, is a modification of the Adagrad optimizer (29). The Adagrad optimizer proposes a solution when the Gradient Descent starts to converge quickly because of a high slope, but in the bottom of the valley of the cost function plot they slow down and takes too long to converge. To solve the proposed problem, the Adagrad optimizer accumulates the square of the gradients to calculate the step that will be taken. This approach works well in simple quadratic problems, but it stops too early in NN training. Remember that the gradient descent is a generic optimization algorithm and is not specific to neural networks. So, intending to perform better in NN, the RMSProp accumulates only the gradients from the most recent iterations. Here, β represents the decay rate.

$$\begin{cases} g_t \leftarrow \nabla_\theta J(\theta) \\ s \leftarrow \beta s + (1-\beta) g_t^2 \\ \theta \leftarrow \theta - \eta \frac{g_t}{\sqrt{s+\epsilon}} \end{cases} \tag{II.13}$$

Adam

Adaptive moment estimation, or Adam (30) represented in Equation II.14, combines the concepts of the Momentum Optimization and the RMSProp. Like the Momentum approach, it keeps track of

an exponentially decaying average of past gradients, and inspired by RMSProp, it keeps track of an exponentially decaying average of past squared gradients with the different decay parameters β_1 and β_2.

$$\begin{cases} g_t \leftarrow \nabla_\theta J(\theta) \\ m \leftarrow \beta_1 m - (1-\beta_1) g_t \\ s \leftarrow \beta_2 s + (1-\beta_2) g_t^2 \\ \hat{m} \leftarrow \frac{m}{1-\beta_1^T} \\ \hat{s} \leftarrow \frac{s}{1-\beta_2^T} \\ \theta \leftarrow \theta - \eta \frac{\hat{m}}{\sqrt{\hat{s}+\epsilon}} \end{cases} \tag{II.14}$$

II.2.5 Metrics

Metrics are an important part of several fields of study. With them, we can measure how well a certain action or method is doing, and in the Machine and Deep Learning area this is no different, so below are some common metrics used for regression problems.

Mean Squared Error

In the Equation II.15 represented below, m represents the amount of values that will be compared with their respective targets, $h(x^{(i)})$ represents the output of the function being worked on, where h is the function, and finally $y^{(i)}$ represents the target number.

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2 \tag{II.15}$$

However, it is common to use a variation of Mean Square Error (MSE) called Root Mean Square Error (RMSE) that is showed below on Equation II.16.

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2} \tag{II.16}$$

As we can see, the difference remains in the square root of the result.

Mean Absolute Error

The Equation II.17 shows the definition of Mean Absolute Error (MAE), where, as the MSE function, m represents the amount of samples that will be compared with their respective targets, $h(x^{(i)})$ represents the output of the function being worked on, where h is the function, and finally $y^{(i)}$ represents the target

number.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|h(x^{(i)}) - y^{(i)}| \tag{II.17}$$

II.2.6 Gradient Issues

Glorot (31) proposed a solution for the two main issues in training NN. The vanishing gradients consist of decreasing gradients as the backpropagation algorithm advances to lower layers of the network, so the weights of the lower layers remain barely unmodified, which makes the NN never converges to a good solution. On the other hand, sometimes the gradients can grow largely, so each layer receive major updates and the NN diverges.

Glorot and Bengio found that the combination of logistic sigmoid function and the initialization weights technique, a normal distribution with $\mu = 0$ and $\sigma = 1$, may be the cause for those problems. They demonstrated that with that combination, the output variance was way larger than the input variance, doing that over the layers the activation function saturates at the top layers, and this situation gets worse because the sigmoid function has a mean of 0.5 and not 0. Despite it is not possible to ensure that this problem will not happen, intending to remedy this Glorot and Bengio proposed a new weights initialization process described in Equation II.18 using a Uniform Distribution and the Equation II.19 using the Normal Distribution. In both cases, the value fan_{avg} is defined in Equation II.20, where fan_{in} represents the number of input and fan_{out} represents the number of neurons.

$$X \sim U(-r, +r), \text{ where } r = \sqrt{\frac{3}{fan_{avg}}} \tag{II.18}$$

$$X \sim N(\mu, \sigma^2), \text{ where } \sigma^2 = \frac{1}{fan_{avg}} \tag{II.19}$$

$$fan_{avg} = (fan_{in} + fan_{out})/2 \tag{II.20}$$

II.3 Related Works

Vannnistem et al. (9) speak of several approaches for post-processing meteorological data, including Machine Learning, citing Forest algorithms, CNN and more. Khodayar et al. (32) proposed a Spatio-temporal Graph Deep Neural Network for Short-term Wind Speed Forecasting. Abrahamsen (33) used meteorological data from MET to train NN in his master thesis, using Auto Regressive NN models to predict the next values of weather time series. He also made available a Python code with documentation. Grönquist (34) has shown the use of Deep Learning on several parts of the meteorological process, mainly on post-processing, and described how this could largely decrease the computational cost of the process.

Kudo (35) used an encoder-decoder based convolutional model to perform a post-processing in a gridded forecast. Ruiter (36) also uses a 2D convolutional model to post-processing weather data in his master thesis. Han (37) introduces a model based on U-net for correction named as CU-net to accomplish the model forecast correction task using weather variables from the European Centre of Medium-Range Weather Forecast Integrated Forecasting System global. Rasp (38) used data from weather stations in Germany to help the post-processing. The neural network approach significantly outperformed the proposed benchmark, being computationally more accessible.

In an effort to predict frost events, Lira et al. (39) has used a Graph Neural Network (GNN) with spatio-temporal attention architecture for frost forecast. The main objective were predict frost for an experimental site, for that were used air temperature and humidity from the site with data from 10 weather stations. They mapped weather stations' locations to nodes on a graph and defined the edges bases on geographical proximity. However, the adjacency matrix was optimized during the training phase.

Wilson et al. (40) in the paper A Low Rank Weighted Graph Convolutional Approach to Weather Prediction also propose an approach to weather forecast with graphs where the adjacency matrix is optimized during the training phase, using the weighted matrix as a model parameter that can be learned from the training data.

Chapter III Proposed Methodology

This chapter will detail the entire data acquisition process, the Extract-Tranform-Load (ETL) process, construction of the NN and train process. All these steps are made using the programming language Python. The following steps were carried out aiming only the meteorological data of Norway, excluding Norwegian islands such Svalbard.

III.1 Data Acquisition

First, It was defined that the features used as input of the NN will be:

- **Location:** Latitude and Longitude
- **Elevation:** Geographical altitude
- **Date and Time info:** Year, Month, Day and Hour
- **MEPS output:** Temperature forecast
- **GRIDPP output:** Temperature forecast

And as desirable output will be use:

- **Meteorological Station data:** observed temperature

Despite MET making all the meteorological data publicly available, first it is needed to gather all the data and transform into a shape that makes sense for training a NN. The measured temperatures were obtained using the Frost API[1] that is a public API provided by MET, and the remain data was available on the thredds website[2], also owned by MET. The exception remains on Elevation Data, and for this data will be used the Google Elevation API[3] which provides Elevation data once informed latitude and longitude.

The process will happen as described in the diagram below on Fig. III.1

[1] https://frost.met.no/api.html/
[2] https://thredds.met.no/thredds/catalog.html
[3] https://developers.google.com/maps/documentation/elevation/overview

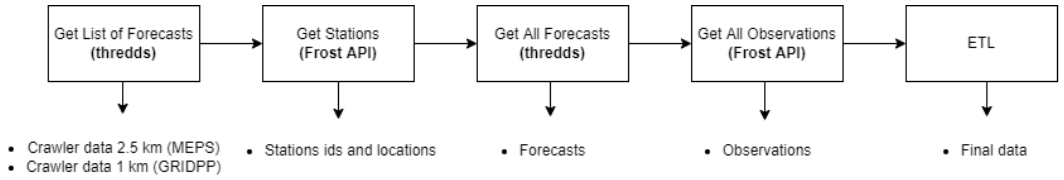


Figure III.1: Data Acquisition Pipeline.

III.1.1 Get List of Forecasts

Trying to make the process more efficient and to consume less computational processing, the first step was to build a Crawler that was capable of entering the thredds website and scrape all the URL files for MEPS and GRIDPP. There were three main reasons for this decision:

1. First, there might be missing files on thredds, so it was preferable to know all forecast files that were available before wasting time scraping the data or querying the Frost API for data that was unavailable to compare.

2. Secondly, the forecast files were extremely large, one single file could be larger than 80 GB. Considering there are 4 forecast files a day, 365 days per year and considering a range of 3 years that will be used in this work, it is about 4380 files that must be used just for MEPS, and it was not feasible to download all the files. The number of files cannot be accurately quantified, as there is a possibility that there are missing files in the met database.

3. That leads us to the third reason, it is possible to open and use the files with the URL. By doing that, it is possible to save much time and there is no need for this locally occupied space.

In this step, the years of interest are set. Afterwards, the program searches inside every month page for every set year what are the days available on that month, this results with the URL of every day page saved as a list. So, the program iterates over the list, accessing each URL and searching for the files links. As there are many files in thredds, the results were filtered using Regular Expressions to focus only on the main files that are broadcast every six hours.

Finally, the found results are stored in different JSON files, one for MEPS and another for GRIDPP. Beyond the file URL, some metadata such as date and time information was also saved to make the ETL process more feasible. This workflow can be visualized on Algorithm 1.

III.1.2 Get Stations

This step consists in get the stations that have air temperature data for declared years and their latitudes and longitudes.

As the endpoint that return the air temperature do not inform the latitude and longitude, before getting any further it was needed to make requests to the endpoint "/availableTimeSeries". Here it was

Algorithm 1 Get List of Forecasts

```
Require: years
  months ← [1...12]
  for year in years do
      for month in months do
          for existing days on page do
              list ← Save year, month, day, url as an item
          end for
      end for
  end for

  for item in list do
      Access url saved on item
      for file in url do
          if filename matchs RegularExpression then
              output ← year, month, day, hour, url, filename and fileurl
          end if
      end for
  end for
```

possible to query for stations that have air temperature data and set the available dates as a parameter, and by doing that we have a list of the available stations and their Id's. Stations Id's are used to query the latitude and longitude of the endpoint "/sources" and due to the large number of stations it was necessary to split the endpoint request into a few different requests to not exceed the URI length limit.

In order to get the elevation data, which was not available on Frost API, it was used the Google Elevation API. Their use is quite simple, once you had an API Key, it is only necessary to query using latitude and longitude for a point. The number of requests needed for this work were in the free API usage quota.

In the end, Station Id, Latitude, Longitude and Elevation were saved as a Comma Separate Value (CSV) file. An example of what has been described can be seen in Algorithm 2

III.1.3 Get All Forecasts

Here, the air temperature forecasts are obtained from the files scraped on the first step by every station found on the previous step.

First, the JSON file that was the first step output was loaded and the CSV output file for the last step was loaded too. So, file by file described on JSON, the code uses the netCDF4 library to open each URL file to get the temperature forecast for each station from the nearest point on the grid, and also the latitude and longitude of that grid point. Because of the different format of Network Common Data Form (netCDF) File, before getting the temperature, it was needed to extract all latitude and longitude from the file and build a grid object with the help of the GRIDPP Library. This grid object was used to translate the stations locations to file indexes, and it is possible to only extract the data from the file

using those indexes.

Some key points to consider:

- Getting the forecast of the nearest point of the station in the grid is possible by using a method of gridpp object, and this method makes the procedure easier than deploying a function of its own.
- The netCDF files have the results of all 10 ensemble member on the temperature forecast, to summarize the results, the mean forecast value was used.
- Despite a single file having the forecast for several hours, thinking about the amount of final data, it was only used the forecast hours 0, 1, 2, 6, 12, 24 from each file. That is, in the file of 1 July 2019 at 12Z, it was used the forecast for the hour 12:00, 13:00, 14:00, 18:00, 24:00, 12:00 (from 2 July). These values were arbitrarily defined with the intention of understanding how the model would work with nowcast data, that is, data for a very close event horizon and with farther prediction data.
- As we will have several forecast values for the same date and time from different files, in order to be able to take advantage of all the results and model the uncertainty of the forecast for hours ahead, the date and time of each forecast and the uncertainty of this forecast, represented by the variable hours ahead, were stored. That is, we may have 2 entries for January 1, 2022 at 12:00, one from the forecast that came out at that exact time and another that was forecast 6 hours earlier. This value that indicates the advance of the forecast in question is saved in a variable called "hour ahead".

Algorithm 2 Get Stations

Require: $initial_year$
$available_stations \leftarrow request('/availableTimeSeries', referencetime = initial_year)$

for row in $available_stations$ **do**
  if $validTo$ not in row **then** ▷ *Only stations that are no longer available have the field* $validTo$
    $active_stations \leftarrow row$
  end if
end for

for $station_id$ in $active_stations$ **do**
  $stations \leftarrow request('/sources', ids = station_id)$ ▷ *Get Latitude and Longitude values*
end for

for $station$ in $stations$ **do** ▷ *Google Elevation API*
  $elevation \leftarrow request('/elevation', locations = station['latitude']\&station['longitude'])$
end for

$output \leftarrow merge(station, elevation)$

- This whole procedure works for MEPS and GRIDPP data.
- It was noticed that the execution time varied greatly depending on the location of the user executing the code, probably because of the distance from the file servers. For the purpose of replicating results, it was noted that the execution time of requests in Google Colaboratory was considered acceptable.
- netCDF format is a popular file extension to storage meteorological data (41, 42, 43, 44).

At the end, the result was saved as several CSV files.

This procedure can be visualized on Algorithm 3.

Algorithm 3 Get All Forecasts

Require: $list_of_forecasts$
  $hours_to_get \leftarrow [0, 1, 2, 6, 12, 24]$
  for $file$ in $list_of_forecasts$ **do**
    if First loop **then**
      Mount the Grid using the GRIDPP library
      Find the indexes on the Grid for each Station Latitude and Longitude
    end if
    for $hour_to_look$ in $hours_to_get$ **do**
      $temperature \leftarrow mean(temperatures)$ ▷ *Each file have 10 different temperatures resulting of 10 different simulations*
      $output \leftarrow temperaute + year + month + day + hour + hour_to_look(hour_ahead)$
    end for
  end for

III.1.4 Get All Observations

This step consists in getting the air temperature observed for each station using the FROST API.

Initially, the crawler output from the first step and the metadata from the stations from the second step are needed as input. The crawler's output is used to know which date and time to query in FROST API, and the stations' metadata serves to know which station to query in FROST API.

The output of this step is a JSON file. An example may be found on Algorithm 4

III.1.5 ETL

The final step consists in gathering the results of the previous steps in one single CSV file with the features and the label as columns. This transformation was made by joining the station id, year, month, day and hour columns through the files.

Intending to bring the temperature data to the same type of measure, a final transformation is performed by bringing the temperature of the forecasts, which are in Kelvin to degrees Celsius, thus becoming compatible with the information extracted from the Frost API.

Algorithm 4 Get All Observations

Require: $stations$
Require: $forecast_files$
  $years \leftarrow forecast_files['years']$
  for $year$ in $years$ **do**
    for $station$ in $stations$ **do**
      $output \leftarrow$ request$('/observations',$
        $sources = stations,$
        $referencetime = years,$
        $elements =' air_temperature',$
        $levels = 2,$
        $qualities = 0)$
    end for
  end for

Therefore, as the final format of the data, we will have a table with 9 columns that will represent the characteristics of measurements and forecasts for a specific weather station at a given time. Such data in this work will not be modelled as a time series, since it is not possible to guarantee a regular spacing of the data in time.

III.2 Preprocessing

Despite all data acquisition process, after gathering all data in a single file, it is needed to make some preprocessing in order to be used on a NN.

III.2.1 Exploratory Data Analysis

Before start transforming the data itself, it is necessary to understand what is the data and his behaviour.

Counting the number of stations collected, it can be seen that there are 704 stations with more than 11 millions records across the Norway received in the Frost API, distributed according to the two-dimensional histogram in Figure III.2. Gathering this histogram with a plot of the stations, represented by red crosses, in a map found on Figure III.3, it is possible to see that big part of the stations are found on the Norway south region near a big city as Oslo and Bergen.

In Figure III.4 it is possible to see the distribution of the observed temperatures. Despite some extreme temperatures, most of them happen between -16°C and 27°C with the highest temperatures during the summer, and the lowest during the winter with median values near to zero.

It is curious to see that the highest differences between the observed temperatures and the GRIDPP forecast happen when the sensors of the weather stations found extreme temperatures. In Figure III.5 which is a scatter plot with the higher errors of GRIDPP and the observed temperature, it is possible to see that the largest error happen on extreme temperatures in most cases. That could indicate that the

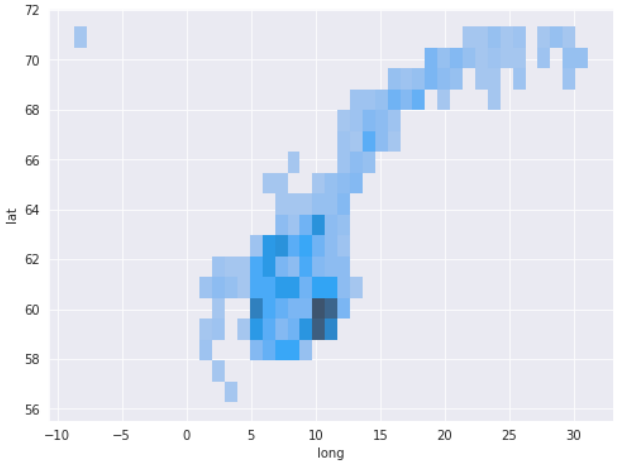


Figure III.2: Norway Station's 2D Histogram.

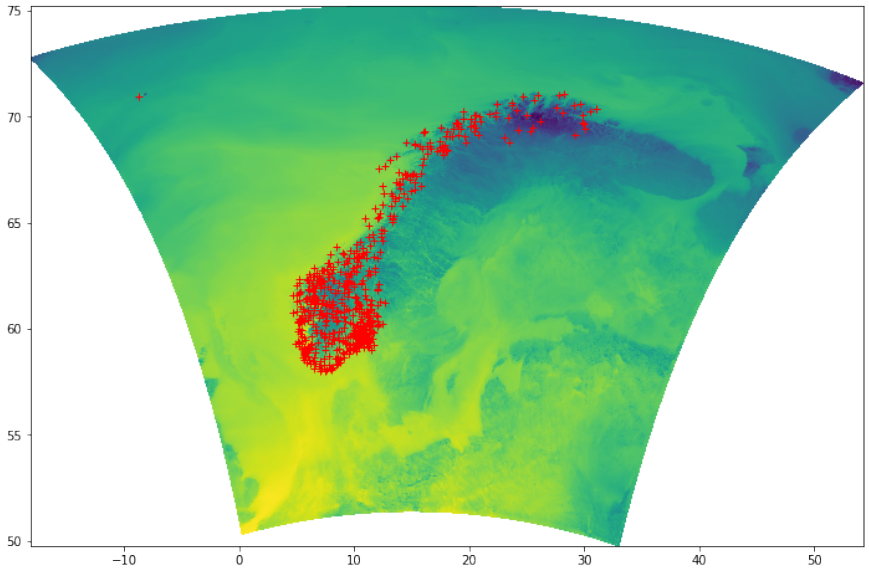


Figure III.3: Norway Station's.

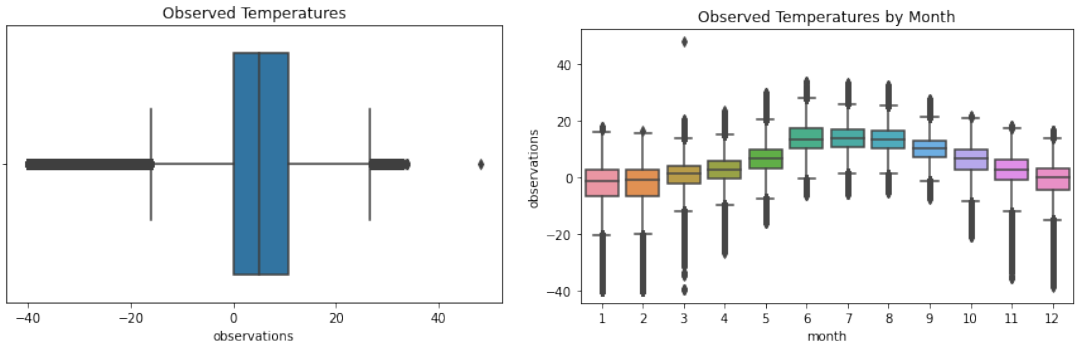


Figure III.4: Temperature distribution.

highly extreme events are really difficult to predict.

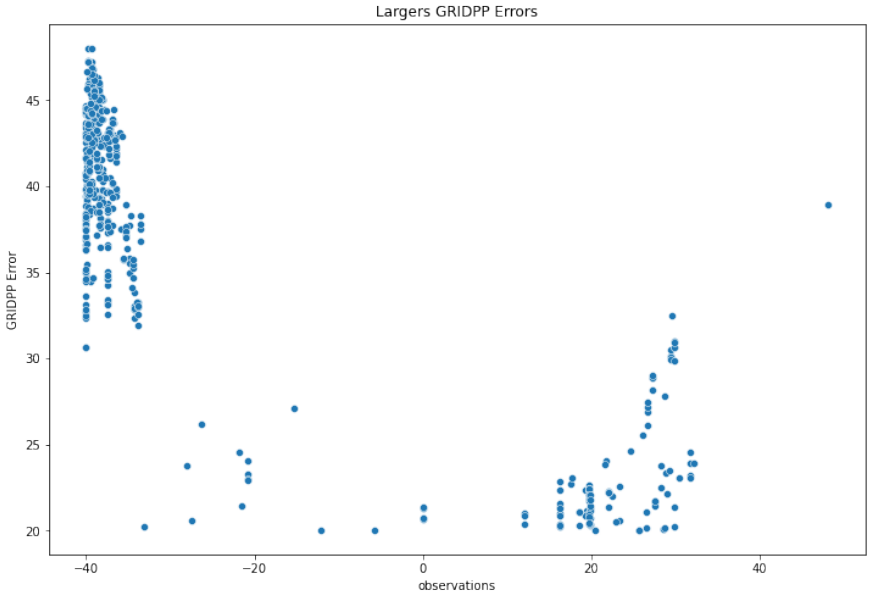


Figure III.5: Larger GRIDPP Errors.

III.2.2 Data Transforming

Here, intending to prepare the data to be consumed by the NN, the library Scikit-Learn (45) was used to perform the follow transformations.

Split Train, Validation and Test Set

As it is known, to properly evaluate any supervised Machine Learning model, it is needed to split the data in a train set and in a test set at least, and in this case was chosen to split a validation set for looking for over fitting over the epochs. The 2020 data was set aside for training, The 2019 data for validation and the 2021 data for test.

The 2019 data was selected as the validation set because, on thredds website, the year of 2019 had fewer data available.

Considering that during GRIDPP post-processing data from weather stations is already used, if the training, validation and test sets were separated by station and not by year, there could be a possibility that information from one station used in the GRIDPP fitting that was in the training set could influence the gridpp result of another station in the validation or test set. Because of this, the separation was by years.

Numerical data transformation

First, the dataset was divided into date and not date data. About the date, this information was modelled as a sin function, means that the first day of a year was interpolated to 0 and the last day to 2π, and the value that will be used is the result of the sin function of the interpolated result. By doing

this, we have a cyclic numerical representation for the date in such a way that the value assigned on December 31st will be close to the value assigned on January 1st, as represented in Figure III.6.

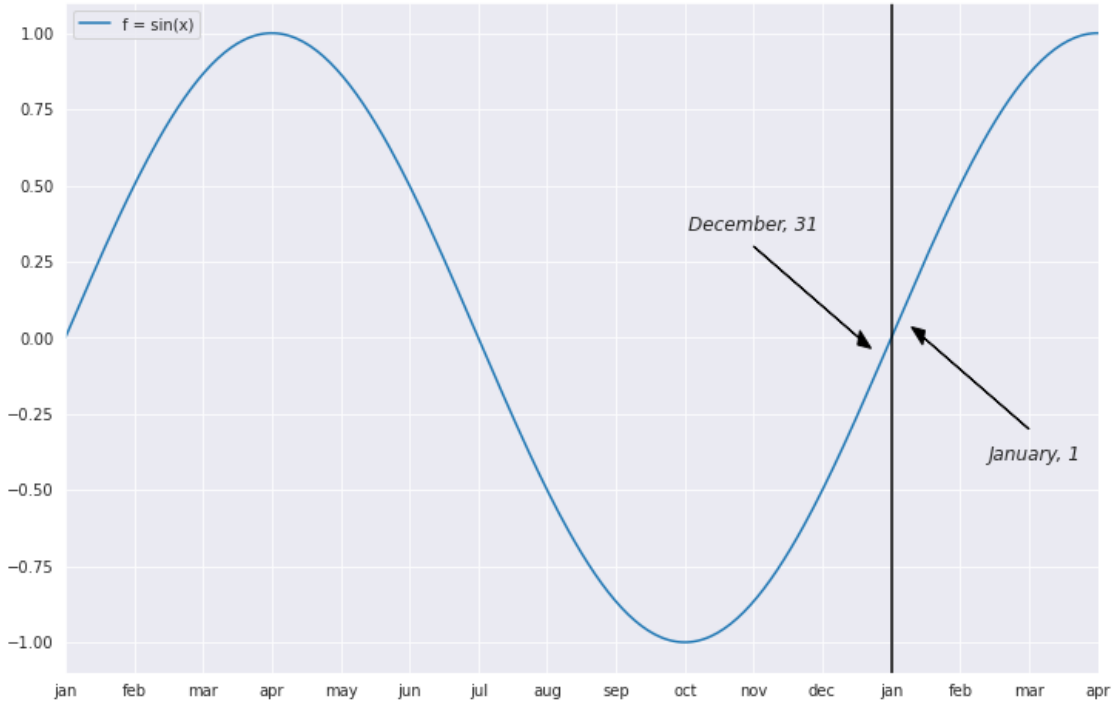


Figure III.6: Modelling Time with Sine Function.

For the rest of the columns, these numerical values were normalized using equation III.1, where x represents the original value, μ represents the mean of the numbers, σ represents the standard deviation and z represents the new value for that variable.

$$z = \frac{x - \mu}{\sigma} \tag{III.1}$$

III.3 Neural Network

For building the Neural Network in this section, the Tensorflow (46) and Keras (47) libraries were used. This library is largely used in the Deep Learning field. Two different architectures were tried: a MLP; and a CNN as described below. Both networks were trained for a limit of 20000 epochs using the MSE as loss function, the algorithm Adam as optimizer with a learning rate of 0.001, and it was used an early stopping method with 2000 epochs of patience in order to avoid overfitting.

The patience of the early stopping method was chosen through experiments, where the models were first trained for 20000 epochs without the early stopping and the convergence of the validation curves evaluated. It was noticed that the curve of the error of the validation set starts to increase systematically at around 4.5k epochs for the CNN model and 2.5k epochs for the MLP model, but decreases again, stagnating only after 12000 epochs in MLP and 16000 in CNN. Even though the curves stagnated very

late, we should not expect the training to go up to so many epochs because of overfitting issues. However, choosing a very small patience would not allow the training to converge to its best, as it was noticed in the experiments that the error of the validation set decreases significantly until a couple of thousand epochs. Therefore, based on this analysis, we chose a patience of 2000 epochs.

The choice of these architectures, especially the number of neurons and kernel size, was performed considering that the first layers should be bigger than the range of the number of inputs (features) for generalization purposes, but not too big to not potentially increase the computational cost unnecessarily. The focus of this work is to present the first post-processing solution of this type and to explore different combinations of inputs as presented in the following section. Due to the amount of data, training time and computational cost, no experiments were conducted to choose the best architecture, although there are already consolidated methods of neural network hyperparameter optimization, many already implemented in the Keras library, such as Random Search, Bayesian Optimization and Hyperband. However, the application of these methods can be considered as future work.

III.3.1 Multi Layer Perceptron

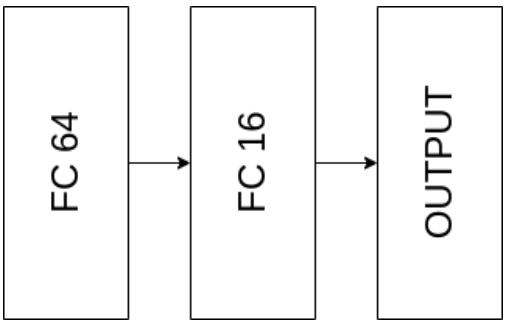


Figure III.7: Multilayer Perceptron.

As represented by Fig. III.7, The proposed MLP has only two FC hidden layers with 64 and 16 neurons, respectively. These two hidden layers use the $\tanh$ function as activation function, Glorot initialization with a uniform distribution for the parameters, and the output layer does not use any activation function. The choice for the $tanh$ activation function instead of the usual $sigmoid$ function was primarily due to what has already been explained in the weights' initialization method proposed by Glorot, and this decision was reinforced by neglecting the also usual $ReLU$ function due to the negative values of temperatures that are important and cannot be ignored. This same situation does not occur in image processing, where the $ReLU$ function is widely used, since the images are composed of pixels that, for the most part, are represented by matrices ranging from 0 to 255.

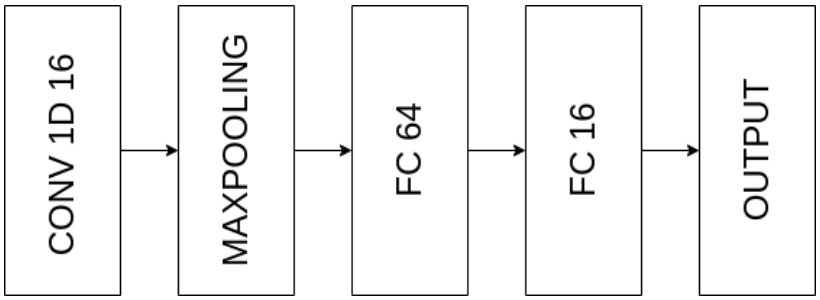


Figure III.8: Convolutional Neural Network.

III.3.2 Convolutional Neural Network

As represented by Fig. III.8, the proposed CNN has one 1D convolutional layer with 16 kernels of size 3, followed by a Max Pooling Layer with 2 of pool size and stride, and two FC layers with 64 and 16 neurons respectively and Glorot initialization with a uniform distribution for the parameters. Like the MLP construction, all layers, except the last, have $\tanh$ functions as activation functions. The reason for choosing the $tanh$ activation function in this case is the same as described in section III.3.1.

The use of 1-dimensional and not 2-dimensional convolutions was due to the data format, since the data used are from meteorological stations and these are not evenly distributed, geographically speaking, as shown in Figure III.3 and III.2. Thus, it would not be possible to directly represent these data in a 2-dimensional matrix, as expected by a CNN with 2-dimensional convolutions.

Chapter IV Results and Discussion

IV.1 Experiment Environment

All training and test steps were performed in Pop!_OS 22.04 LTS 64-bits, with an Intel Core i7-7700HQ CPU @ 2.80GHz x 8, 16 GiB of memory, an NVIDIA GeForce GTX 1070 with 8 GiB of VRAM running CUDA 11.6.

IV.2 Benchmarks

Before start any training, it was necessary to define a benchmark. Looking for today solutions, that are MEPS and GRIDPP, makes sense to use the error of these solutions as benchmark, because if the general error of the NN was higher than the error of anyone of those solutions, there will be no reason to use a NN approach, being better stay with the current post-processing approach. As showed by the table IV.1, despite the error function chosen for training to be the MSE, in order to make the error more readable and on the same scale of the data, the future comparisons will be made using the RMSE. The percentage of errors greater than 3°C will also be considered, since errors greater than this limit are more worrisome.

Table IV.1: Benchmarks

Model	RMSE	Percentage of errors greater than 3
MEPS	5.258	54.05%
GRIDPP	1.886	9.38%

Analysing the distance between the predicted by MEPS and GRIDPP and the one measured by the meteorological stations, we have the graphs IV.1 and IV.2.

In the next sections, the learning curves of all training procedures are presented. The monitoring of the graphs were made using the tool called Tensorboard (46) and their plots are made using the Python libraries Matplotlib (48) and Seaborn (49).

IV.3 Training Results

Intending to understand if the MEPS and GRIDPP values actually provides an upgrade for the training process, for each proposed architecture 3 models were trained:

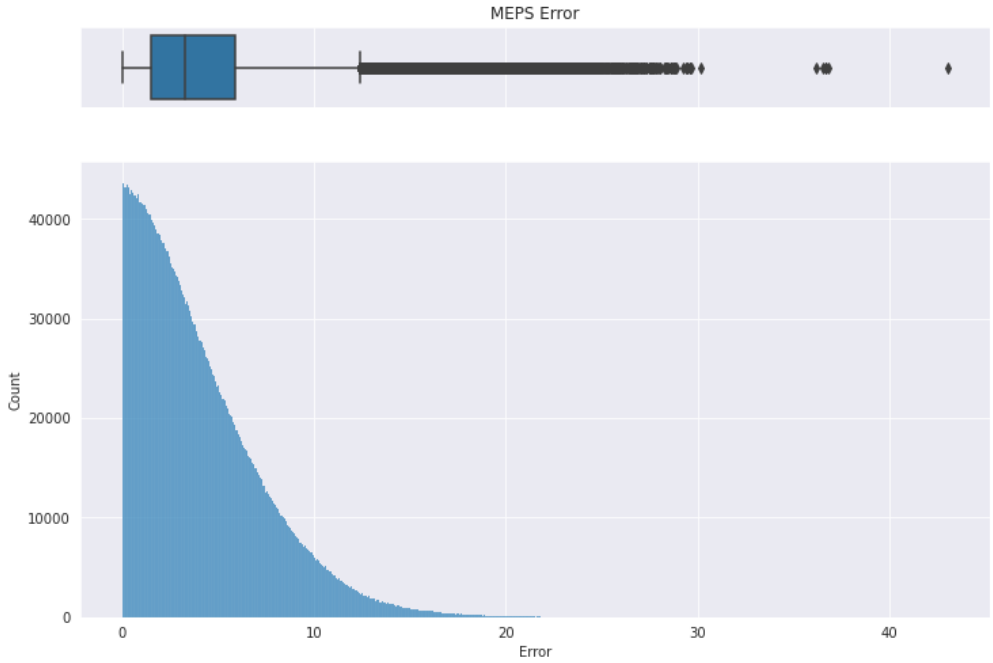


Figure IV.1: MEPS - RMSE Distribution.

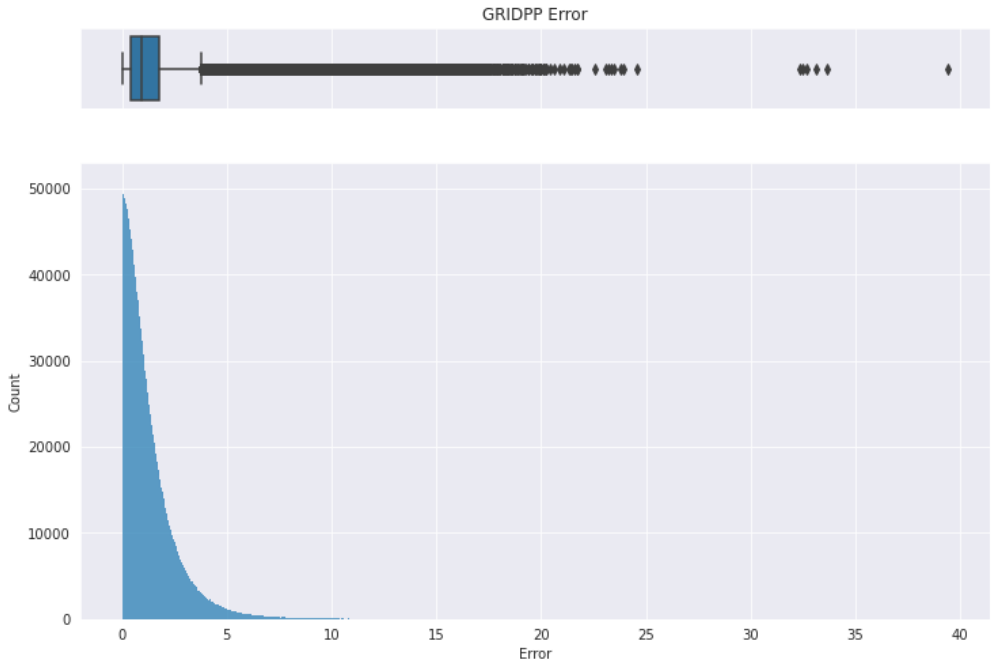


Figure IV.2: GRIDPP - RMSE Distribution.

1. A model only with MEPS forecast, excluding the post-processed value from GRIDPP
2. A model only with GRIDPP post-processed value, excluding the original forecast from MEPS
3. A model with both values, MEPS and GRIDPP

IV.3.1 Multilayer Perceptron

Comparing the MLP models, we have the following results in table IV.2 and the learning curves in Figure IV.3.

Table IV.2: Test set RMSE on MLP models.

Model	RMSE	Stopped Epoch
Only with MEPS	3.78	2005
Only with GRIDPP	1.797	5504
With MEPS and GRIDPP	1.786	9929

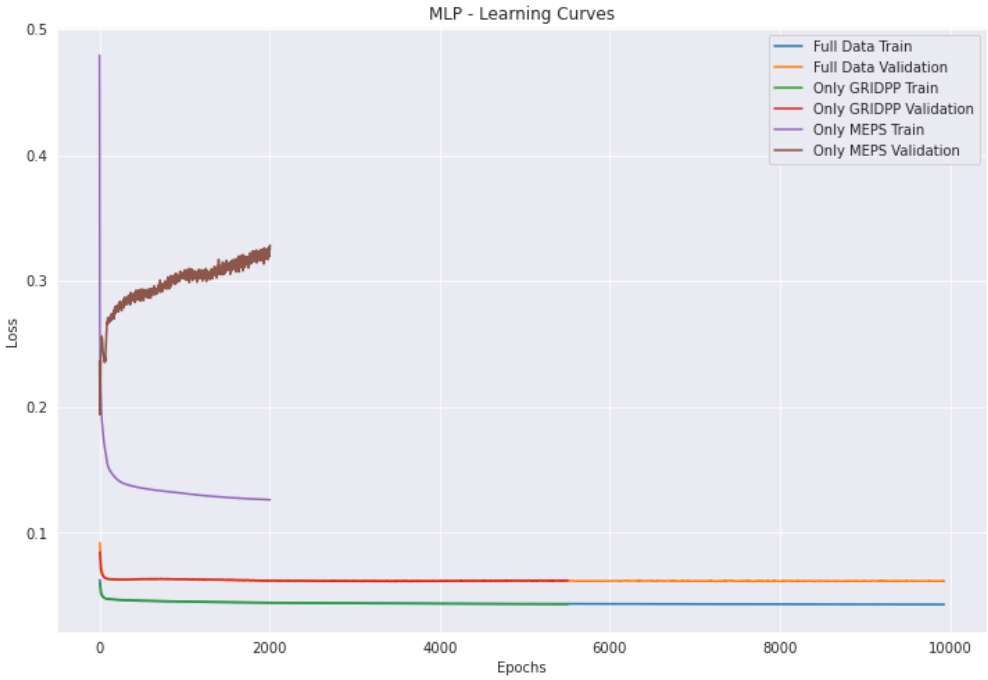


Figure IV.3: MLP - Learning Curve.

Looking at table IV.2, it was possible to notice that a MLP cannot replace the GRIDPP as the only post-processing step, since the RMSE of the pure GRIDPP was half compared to the MLP using only MEPS data and the Network was not able to converge much further than the early stopping patience, as seen on Figure IV.3. On the other hand, using MEPS and GRIDPP together allowed a longer period of convergence and a smaller error than GRIDPP alone.

Since the combination of MEPS and GRIDPP performed better, deeper analysis comparing the Network with the Benchmarks was made, counting the number of errors greater than 3°C in each case divided by the variable "Hour Ahead" to understand how the error increases as the hours pass and uncertainty increases. That analysis was represented by Figure IV.4.

Knowing the critical error threshold of 3°C, the behaviour of GRIDPP and neural networks was plotted in situations where the GRIDPP error was higher than 3°C for the test sample. Figure IV.5 represents the error higher than 3°C of GRIDPP and Figure IV.6 represents the error o MLP when GRIDPP error is

higher than 3°C.

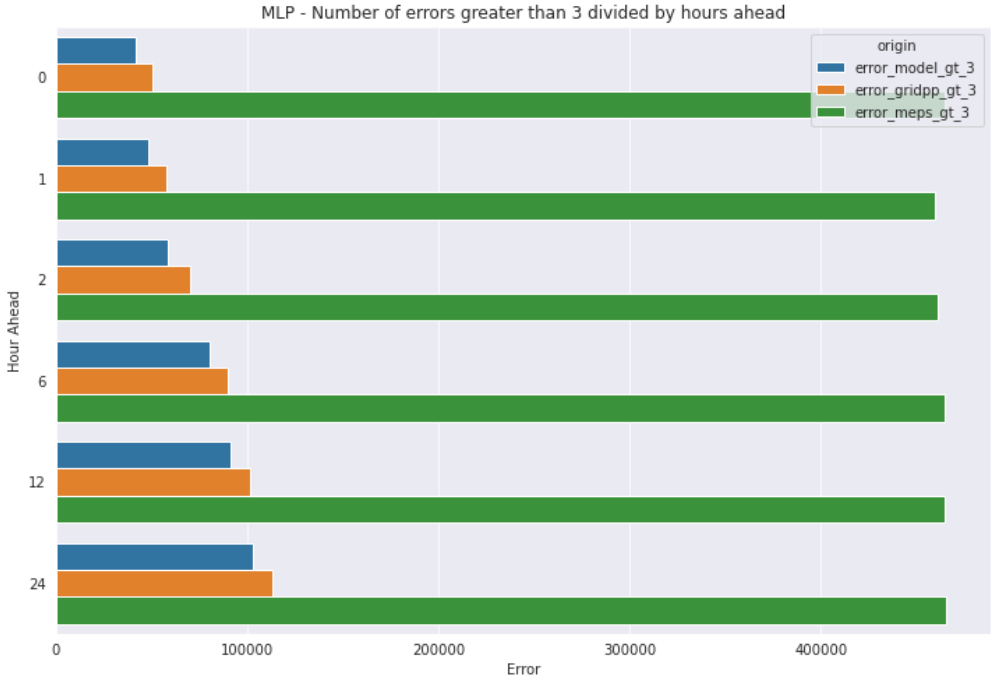


Figure IV.4: Comparing Errors of MLP with the Benchmark.

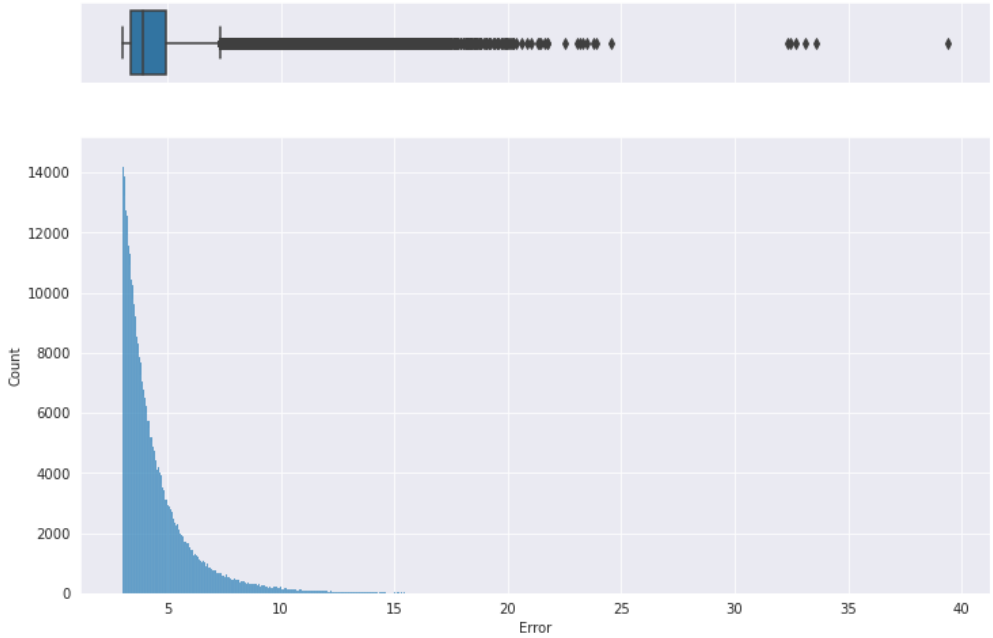


Figure IV.5: GRIDPP error higher than 3°C.

IV.3.2 Convolutional Neural Network

Comparing the CNN models, we have the following results in table IV.3 and the learning curves in Figure IV.7.

The results above were quite similar with the MLP results, once more the combination of MEPS

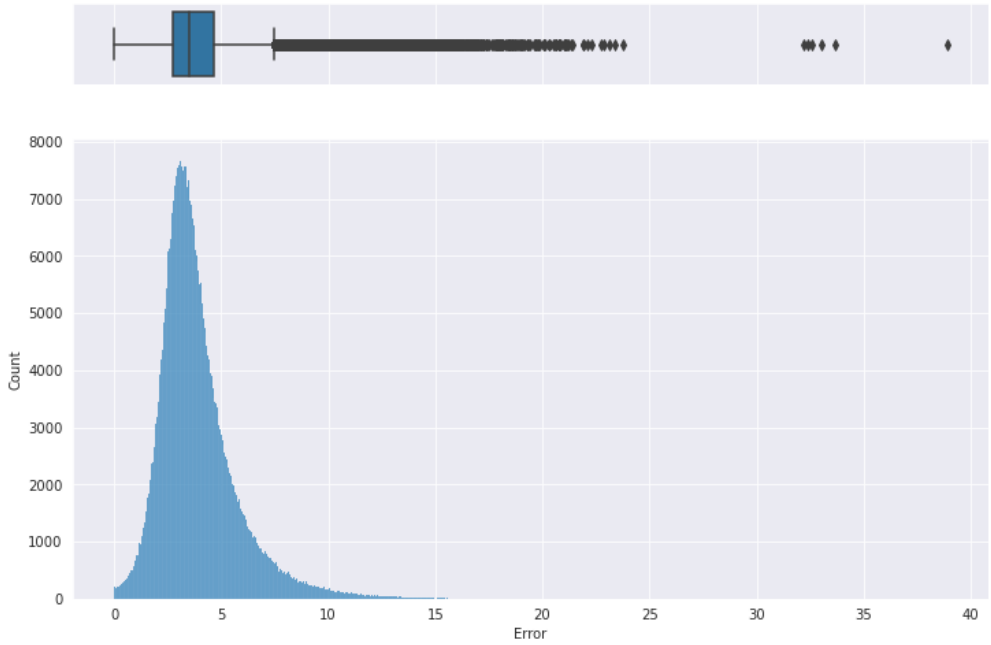


Figure IV.6: MLP error in cases when GRIDPP error is higher than 3°C.

Table IV.3: Test set RMSE on CNN models

Model	RMSE	Stopped Epoch
Only with MEPS	3.984	2100
Only with GRIDPP	1.808	3249
With MEPS and GRIDPP	1.790	12063

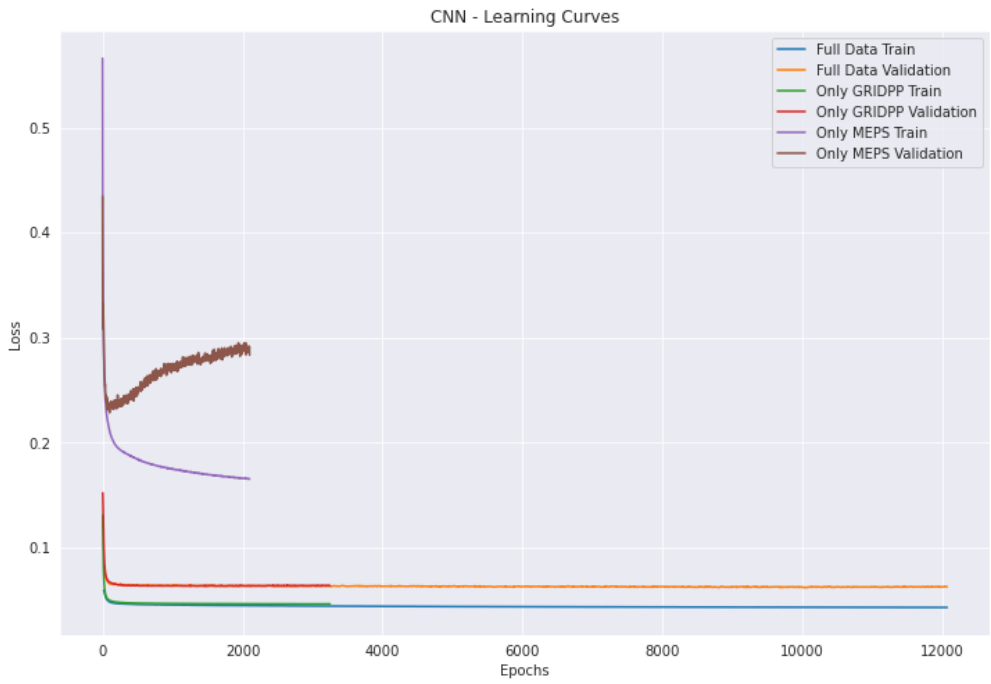


Figure IV.7: CNN - Learning Curve.

by Figure IV.8.

In the same way as it was plotted for the MLP results, the Figure IV.9 represents the CNN error

distribution when the GRIDPP has errors greater than 3°C.

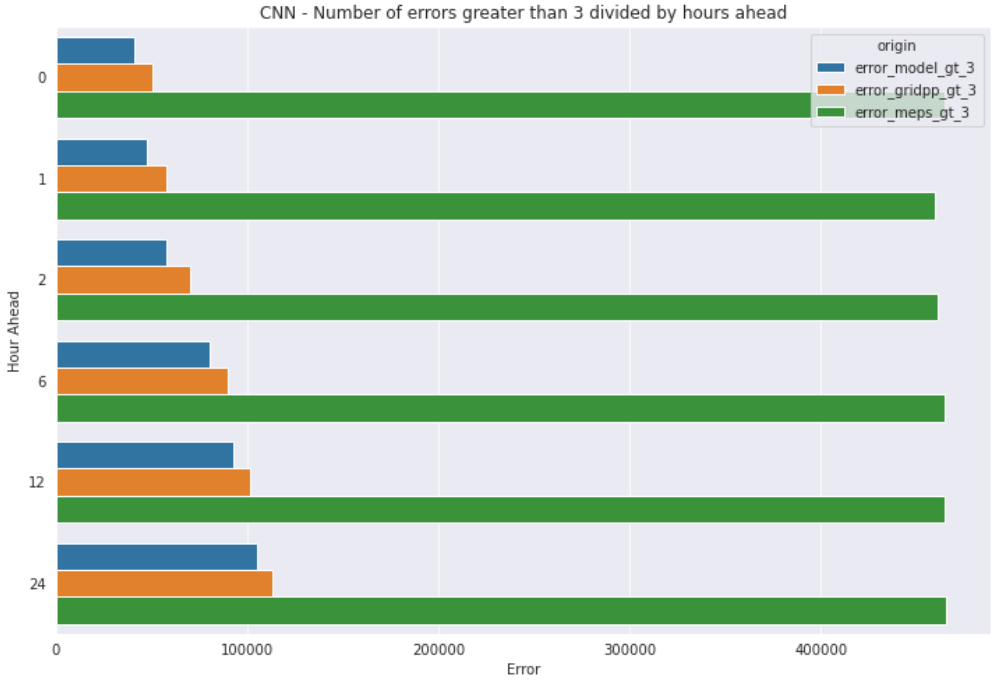


Figure IV.8: Comparing Errors of CNN with the Benchmark.

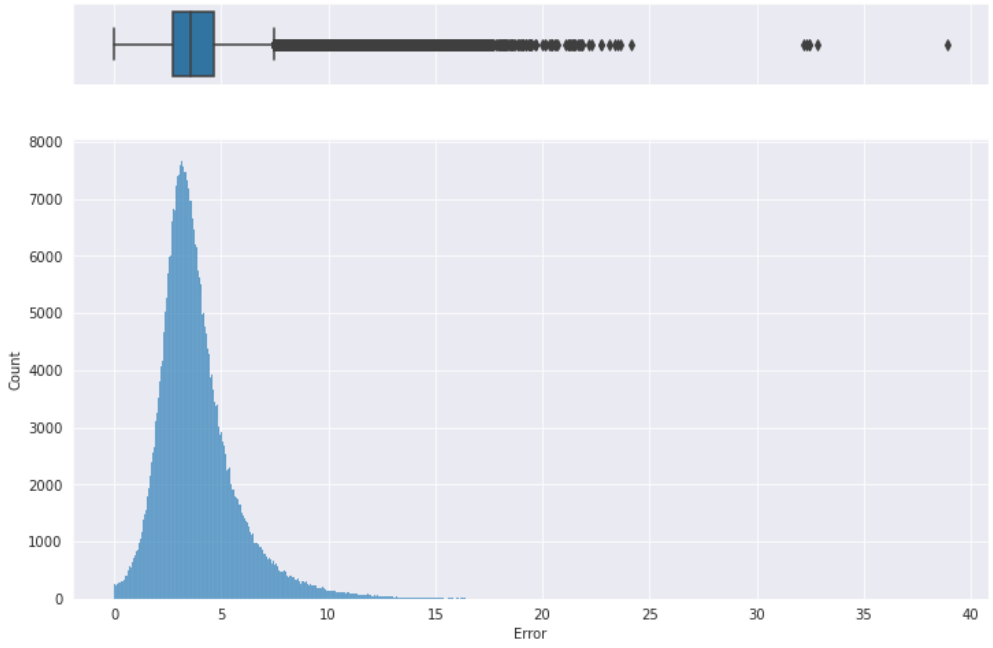


Figure IV.9: CNN error in cases when GRIDPP error is higher than 3°C.

IV.4 Discussion

To guide the discussion, the Table IV.4 organize the results of the experiments.

Despite the current forecast method are excellent comparing to other regional forecast methods, yet a post-processing method are mandatory. For purposes of post-processing, the GRIDPP has showed

Table IV.4: Performances on Test set

Model	RMSE	Percentage of errors greater than 3	Percentage of smaller errors than GRIDPP where GRIDPP error is greater than 3	RMSE where GRIDPP error is greater than 3
MEPS	5.258	54.05%	-	5.995
GRIDPP	1.886	9.38%	-	4.484
MLP	1.786	8.20%	74.8%	3.912
CNN	1.790	8.23%	73.8%	3.902

as a wonderful tool to improve and downscaling the forecast result, greatly reducing the forecast error. Thanks to the GRIDPP result, one can train neural networks in a better way.

First, looking at the Table IV.4, it is possible to see that both proposed architectures performed well on the test set, achieving a smaller RMSE and fewer error greater than 3°C. It is also interest to notice on Figures IV.4 and IV.8 how the quantity of errors greater than 3°C increases how the forecast range grows, as expected on real data, and we can see in each range value both NN achieved a better result.

Some similar results can be seen in the Figure IV.10, that shows the RMSE raising with the passing hours as expected. It is possible to see how the RMSE grows over the hours due to the increase in uncertainty. Here, the MLP and CNN curves are represented by the same line (the blue one) because both results were really close, so it would not be possible to distinguish the 2 lines. On the X axis, only the "hour ahead" values predicted by the model were represented.

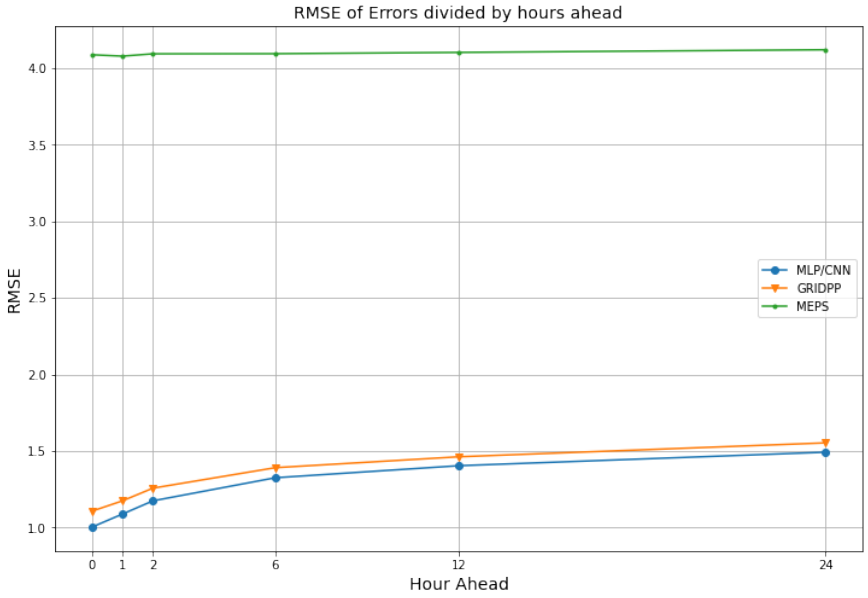


Figure IV.10: Comparing RMSE by hours ahead.

Although the MLP and CNN has some errors greater than 3°C, both Networks obtained very expressive results, reaching a smaller error in 74.8% and 73.8%, respectively, of the samples between the test

samples where the GRIDPP error was greater than 3°C. The RMSE in these two cases also decreased by approximately 0.5°C as shown in Table IV.4.

Chapter V Conclusions and Future Work

V.1 Conclusions

As pointed out earlier, an accurate weather forecast is a matter of importance for anyone, because it has several real applications in the economics field, energy field, disaster prevention etc.

The present work has two objectives: development of a data pipeline that makes possible to train and evaluate Machine and Deep Learning models for this kind of problem; and Development and Test of different NN architectures capable of refining the weather forecast value.

First, the programming language Python was used to build a workflow that access three different services: the thredds websites from MET, the FROST API and the Elevation API from Google. It was detailed where the required data are, how it is possible to consult it, how to download it and finally how to ensemble the raw data in a shape that makes sense and in useful way for deep analysis. All the process was designed to be the more computational feasible as possible.

The data pipeline built made it possible not only to build models that help post-process the data, but also to analyse the behaviour and evolution of the climate in Norway. It also made it possible to build models with other purposes, for example, it is possible to isolate only the observations taken by the weather stations and build a time series forecast model to predict the next observations from the stations. This last application used as an example could even open the path for the post-processing calculation of the coming hours of the weather forecast, since the measurements of the stations are used to correct the MEPS outputs in the post-processing of the GRIDPP, or even the weather forecast for a known point with continuous observations.

Despite the chosen output format to be a CSV file, this format is not mandatory. The parquet format has proved extremely popular when we talk about Big Data, so with more collected data (more scraped years or new scraped columns) the parquet format may be an interesting option to store the data. The relational databases as PostgreSQL, MySQL etc are also a possible, and more robust and resilient, way to store this data, but some additional efforts may be necessary choosing this path such as installing and configuring a database service as well as code changes required for interoperability between the code and the database service.

About refining the weather forecast values with NN, two different architectures were proposed, trained and tested. The results showed that only the proposed architectures of NN could not achieve a better

performance than GRIDPP with the MEPS results alone, actually both networks performed poorly without the GRIDPP result and overfitting faster than other trails.

On the other hand, combining the GRIDPP technique with the NN not only reduce the error, but decrease the number of errors greater than 3°C which is very good.

One of the major challenges to build the previous models and explore the data was the large amount of data. More than 11 millions rows in a single dataset with 9 columns just to work with temperature data. This quantity of data makes it more difficult not only to train Machine and Deep Learning models, but the plot task too. Generate visualizations with this mass of data became more difficult for two main reasons:

1. With a lot of data, the visualizations easily turns a little messy and difficult to see.

2. It demands a lot of computational power to plot this quantity of data.

For both cases, some strategies were needed to generate the plots. Some plots changed, and others turned into tables.

These results showed that it is possible to reduce the temperature prediction error with NN, opening the way for more accurate meteorological products, both in the result calculated for a certain area, and for the size of the area of interest.

All data that was used, was storage publicly on Kaggle Website[1] for public access with a Digital Object Identifier (DOI) register to be used in other works. And the preliminary results of this work were published in the journal MDPI[2] in a paper with the same name[3].

[1] https://doi.org/10.34740/kaggle/dsv/3635866
[2] https://www.mdpi.com/
[3] https://doi.org/10.3390/app12147131

V.2 Future Propositions

Both objectives achieved on this work open possibilities for new studies. About the data pipeline topic, talking about the process to extract and manipulate the information, the implementation of the framework Apache Spark[4] may allow the manipulation of a larger amount of data in a shorter time, which could allow the use of other meteorological features as wind direction and precipitation amount.

In this work, only observed data from weather stations was considered, but there are other data sources that might be included that are not weather stations, like homemade weather stations as sold by Netatmo[5] which is a resource already used by GRIDPP, sensors embed on cars and several Internet of Things (IoT) devices that could read meteorological data as temperature, to improve the power of generalization of the NN. And beyond the observed temperature used in this work, it could be possible to use other meteorological features as wind speed, wind direction, pressure and others.

One of the main inputs were the forecast weather, represented by the mean of all available ensemble members. But it is possible to change the data acquisition process and instead of just getting the average of the ensemble members, get the results of all ensemble members and use the different results to try to update the results using not only the mean but other statistics, such as the standard deviation of the results.

As mentioned in the conclusion section, it is possible to use the collected data to predict a time series of the observed temperature using some kind of NN that could realize this work, like CNN or Recurrent Neural Network (RNN). A possibility is to use, but not exclusively, the time series forecast to predict possible anomalies on the time series that could indicate either an error in the sensors of the stations, which would require some special attention of a repair squad, or some unknown meteorological event that could be studied.

As mentioned during the work, the use of hyperparameter optimization algorithms is also encouraged for future work once the computational power needed for this search is available during the experiments.

As well, some new architectures of NN that could use the neighbourhood information like the Graph Convolutional Network could show some interest performance, but by considering the surroundings of each data and how each input relates to the others, once the 2 dimensional CNN could not achieve that because the input are not regular as grid of pixels in picture.

[4] https://spark.apache.org/
[5] https://www.netatmo.com/en-us

Referências Bibliográficas

1 EXAME. *Nevascas paralisam Washington e deixam 10 mortos nos EUA*. [S.l.]. Avaiable at https://exame.com/mundo/nevascas-paralisam-washington-e-deixam-10-mortos-nos-eua/ (accessed: 28.05.2022). Citado na página 1.

2 CNN. *Os desastres naturais que impactaram o mundo recentemente*. [S.l.]. Avaiable at https://www.cnnbrasil.com.br/tecnologia/os-desastres-naturais-que-impactaram-o-mundo-recentemente/ (accessed: 28.05.2022). Citado na página 1.

3 CNN. *Chuvas em Petrópolis deixam pelo menos seis mortos, informa Defesa Civil*. [S.l.]. Avaiable at https://www.cnnbrasil.com.br/nacional/chuvas-em-petropolis-deixam-mortos-informa-defesa-civil/ (accessed: 28.05.2022). Citado na página 1.

4 TOLKIEN, J. R. R. *The lord of the rings: Boxed set*. London, England: HarperCollins, 1991. ISBN 9780261102385. Citado na página 1.

5 INSTITUTE, N. M. [S.l.]. Avaiable at https://www.met.no/en (accessed: 1.12.2021). Citado 2 vezes nas páginas 2 e 6.

6 NIPEN, T. *NWP Docs*. [S.l.]. Avaiable at https://github.com/metno/NWPdocs/wiki (accessed: 1.12.2021). Citado 2 vezes nas páginas 3 e 6.

7 YNOUE, R. Y. et al. *Meteorologia: noções básicas*. [S.l.]: Oficina de Textos, 2017. Citado na página 5.

8 OFFICE, M. *History of numerical weather prediction*. [S.l.]. Avaiable at https://www.metoffice.gov.uk/weather/learn-about/how-forecasts-are-made/computer-models/history-of-numerical-weather-prediction (accessed: 1.12.2021). Citado na página 5.

9 VANNITSEM, S. et al. *Statistical Postprocessing for Weather Forecasts – Review, Challenges and Avenues in a Big Data World*. 2020. Citado 2 vezes nas páginas 6 e 16.

10 NIPEN, T. *Gridded post-processor*. [S.l.]. Avaiable at https://github.com/metno/gridpp (accessed: 1.12.2021). Citado na página 6.

11 METEOROLOGIA, I. N. de. *PREVISÃO NUMÉRICA - MODELO*. [S.l.]. Avaiable at https://portal.inmet.gov.br/servicos/previs%C3%A3o-num%C3%A9rica-modelo (accessed: 1.12.2021). Citado na página 6.

12 NAVY, B. *MODELO COSMO*. [S.l.]. Avaiable at https://www.marinha.mil.br/chm/dados-do-smm-paginas-modelagem-numerica/modelo-cosmo (accessed: 1.12.2021). Citado na página 6.

13 CPTEC/INPE. *About BRAMS*. [S.l.]. Avaiable at http://brams.cptec.inpe.br/about/ (accessed: 1.12.2021). Citado na página 6.

14 MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 7.

15 GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. [S.l.]: " O'Reilly Media, Inc.", 2019. Citado 5 vezes nas páginas 7, 8, 9, 12 e 13.

16 MARVIN, M.; SEYMOUR, A. P. *Perceptrons*. [S.l.]: MIT Press Cambridge, MA, 1969. Citado na página 7.

17 HUBEL, D. H. Single unit activity in striate cortex of unrestrained cats. *The Journal of physiology*, Wiley-Blackwell, v. 147, n. 2, p. 226, 1959. Citado na página 9.

18 HUBEL, D. H.; WIESEL, T. N. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, Wiley-Blackwell, v. 148, n. 3, p. 574, 1959. Citado na página 9.

19 HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, Wiley Online Library, v. 195, n. 1, p. 215–243, 1968. Citado na página 9.

20 LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 1998. Citado na página 11.

21 HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado na página 11.

22 SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na página 11.

23 CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1251–1258. Citado na página 11.

24 CHOLLET, F. *Deep learning with Python*. [S.l.]: Simon and Schuster, 2021. Citado na página 11.

25 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. [S.l.], 1985. Citado na página 12.

26 IOF. *Orienteering*. [S.l.]. Avaiable at https://orienteering.sport/orienteering/ (accessed: 1.06.2022). Citado na página 14.

27 POLYAK, B. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, v. 4, p. 1–17, 12 1964. Citado na página 14.

28 TIELEMAN, T.; HINTON, G. Neural networks for machine learning. *Coursera (slide 29 in lecture 6 -RMSprop)*, 2012. Citado na página 14.

29 DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, v. 12, n. 7, 2011. Citado na página 14.

30 KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 14.

31 GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. [S.l.], 2010. p. 249–256. Citado na página 16.

32 KHODAYAR, M.; WANG, J. Spatio-temporal graph deep neural network for short-term wind speed forecasting. *IEEE Transactions on Sustainable Energy*, IEEE, v. 10, n. 2, p. 670–681, 2018. Citado na página 16.

33 ABRAHAMSEN, E. B. *Predicting weather using ANN with free open weather data in python Erik Boye*. Dissertação (Mestrado) — Universitetet i Sørøst-Norge, 2018. Citado na página 16.

34 GRÖNQUIST, P. et al. Deep learning for post-processing ensemble weather forecasts. *Philosophical Transactions of the Royal Society A*, The Royal Society Publishing, v. 379, n. 2194, p. 20200092, 2021. Citado na página 16.

35 KUDO, A. Statistical post-processing for gridded temperature prediction using encoder–decoder-based deep convolutional neural networks. *Journal of the Meteorological Society of Japan. Ser. II*, Meteorological Society of Japan, 2022. Citado na página 17.

36 RUITER, B. de. Post-processing multi-model medium-term precipitation forecasts using convolutional neural networks. *arXiv preprint arXiv:2105.07043*, 2021. Citado na página 17.

37 HAN, L. et al. A deep learning method for bias correction of ecmwf 24–240 h forecasts. *Advances in Atmospheric Sciences*, Springer, v. 38, n. 9, p. 1444–1459, 2021. Citado na página 17.

38 RASP, S.; LERCH, S. Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review*, American Meteorological Society, v. 146, n. 11, p. 3885–3900, 2018. Citado na página 17.

39 LIRA, H.; MARTÍ, L.; SANCHEZ-PI, N. Frost forecasting model using graph neural networks with spatio-temporal attention. In: *AI: Modeling Oceans and Climate Change Workshop at ICLR 2021*. [S.l.: s.n.], 2021. Citado na página 17.

40 WILSON, T.; TAN, P.-N.; LUO, L. A low rank weighted graph convolutional approach to weather prediction. In: IEEE. *2018 IEEE International Conference on Data Mining (ICDM)*. [S.l.], 2018. p. 627–636. Citado na página 17.

41 UCAR. *THREDDS Data Server (TDS)*. [S.l.]. Avaiable at https://www.unidata.ucar.edu/software/tds/ (accessed: 1.06.2022). Citado na página 22.

42 DAAC, O. *THREDDS Data Server (TDS) Catalog Service*. [S.l.]. Avaiable at https://daac.ornl.gov/cgi-bin/service_dataset_lister.pl?svc_id=4 (accessed: 1.06.2022). Citado na página 22.

43 ESRI. *What is netCDF data?* [S.l.]. Avaiable at https://pro.arcgis.com/en/pro-app/2.8/help/data/multidimensional/what-is-netcdf-data.htm (accessed: 1.06.2022). Citado na página 22.

44 NOAA. *THREDDS Enhanced Access*. [S.l.]. Avaiable at https://www.ncei.noaa.gov/access/thredds-user-guide (accessed: 1.06.2022). Citado na página 22.

45 PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 25.

46 ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Citado 2 vezes nas páginas 26 e 29.

47 CHOLLET, F. et al. *Keras*. 2015. https://keras.io. Citado na página 26.

48 CASWELL, T. A. et al. *matplotlib/matplotlib: REL: v3.5.2*. Zenodo, 2022. Disponível em: <https://doi.org/10.5281/zenodo.6513224>. Citado na página 29.

49 WASKOM, M. L. seaborn: statistical data visualization. *Journal of Open Source Software*, The Open Journal, v. 6, n. 60, p. 3021, 2021. Disponível em: <https://doi.org/10.21105/joss.03021>. Citado na página 29.