applied sciences

MDPI

*Article*

# Post-Processing Air Temperature Weather Forecast Using Artificial Neural Networks with Measurements from Meteorological Stations

**Gustavo Araujo [1,†] and Fabio A. A. Andrade [2,3,*,†]**

[1] Federal Center of Technological Education of Rio de Janeiro (CEFET/RJ), Rio de Janeiro 20271-110, Brazil; gala.gustavoantonio@gmail.com
[2] Department of Microsystems, Faculty of Technology, Natural Sciences and Maritime Sciences, University of South-Eastern Norway (USN), 3184 Borre, Norway
[3] NORCE Norwegian Research Centre, 5838 Bergen, Norway
* Correspondence: fabio.a.andrade@usn.no
† These authors contributed equally to this work.

**Abstract:** Human beings attempt to accurately predict the weather based on their knowledge of climate. The Norwegian Meteorological Institute is responsible for climate-related matters in Norway, and among its contributions is the numerical weather forecast, which is presented in a 2.5 km grid. To conduct a post-processing process that improves the resolution of the forecast and reduces its error, the Institute has developed the GRIDPP tool, which reduces the resolution to 1 km and introduces a correction based on altitude and meteorological station measurements. The present work aims to improve the current post-processing approach of the air temperature parameter by employing neural networks, using meteorological station measurements. Two neural network architectures are developed and tested: a multilayer perceptron and a convolutional neural network. Both architectures are able to achieve a smaller error than the original product. These results open doors for the Institute to plan for the practical implementation of this solution on its product for specific scenarios where the traditional numerical methods historically produce large errors. Among the test samples where the GRIDPP error is higher than 3 K, the proposed solution achieves a smaller error in 74.8% of these samples.

**Keywords:** neural network; post-processing; weather; meteorological data; air temperature



**Citation:** Araujo, G.; Andrade, F.A.A. Post-Processing Air Temperature Weather Forecast Using Artificial Neural Networks with Measurements from Meteorological Stations. *Appl. Sci.* **2022**, *12*, 7131. https://doi.org/10.3390/app12147131

Academic Editor: Hariton-Nicolae Costin

Received: 2 June 2022
Accepted: 14 July 2022
Published: 15 July 2022



## 1. Introduction

For a long time, humans have been attempting to make accurate forecasts about the weather. From weather stations to satellites, today, there are sophisticated systems around the globe that can make accurate forecasts, and modern sensors with reliable measures of the weather.

Weather prediction can be used for safe navigation on the sea, preparation for extreme weather events such as hurricanes and blizzards or even for planning when to use electricity generation through natural means such as wind and tides. Therefore, having a precise estimation of weather changes has become highly necessary, especially when dealing with major weather events.

The history of Numerical Weather Prediction (NWP) [1,2] began in 1904, when Vilhem Bjerknes stated that the atmosphere's future state could be predicted by integrating the differential equations that governed its behavior. Bjerknes proposed to use, as initial conditions of the differential equations, data that describe an observed state of the atmosphere—more precisely, observations from meteorological stations. The scientist Lewis Fry Richardson was the first who actually solved this 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 path toward accurate weather forecasting.

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

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.

Today, it is possible to divide weather prediction into three main steps: analysis, forecast and post-processing.

- The analysis step is necessary because meteorological observations are not regular, and mathematical models require that the data provided are 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 behavior 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 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 [3] suggests that statistical post-processing techniques are nowadays key components of the forecasting suites in many National Meteorological Services, for most of them with the objective of correcting the impact of different types of errors on the forecasts.

The Norwegian Meteorological Institute (MET) is the public institution in Norway responsible for weather prediction, monitoring the climate and conducting meteorological research since 1866 [4]. Because of this, the MET is in charge of providing the NWP in Norway. Therefore, the MET produces the MetCoOp Ensemble Prediction System (MEPS, which covers Scandinavia and Finland, and the AROME-Artic, which covers the Svalbard area [5]). With the aim of improving the forecast result, the MET started to develop the Gridded Post-Processor (GRIDPP), which is a tool for post-processing gridded weather forecasts [6]. This tool can downscale the MEPS grid of 2.5 km to a 1 km grid, and also implements different improvements on many parameters, such as altitude corrections on the air temperature parameter, and bias correction using several measurements from sensors spread over Norway. It should be noted that the primary resolution of MEPS would already be considered great compared to other regional models, such as COSMO, which has a 7 km grid, and BRAMS with a 5 km grid, both used officially in Brazil [7–9].

Furthermore, the power of generalization of artificial neural networks is widely known. Grönquist [10] has shown the use of deep learning in several parts of the meteorological process, mainly in post-processing, and described how this could largely decrease the computational cost of the process. Kudo [11] used an encoder–decoder-based convolutional model to perform post-processing in a gridded forecast. Ruiter [12] also used a 2D convolutional model to post-process weather data in his Master's thesis. Han [13] introduced a model based on U-net for correction, named 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 [14] used data from weather stations in Germany to assist the post-processing. The neural network approach significantly outperformed the proposed benchmark, being computationally more accessible.

The GRIDPP is the most used weather product in Norway, and the aim of this work is to use artificial neural networks to further improve its accuracy, especially for challenging

scenarios—for example, where it may not achieve precise forecasts with under 3 K of error. Therefore, the main contributions of this work are highlighted as follows:

1. To develop and test two architectures of neural networks for estimating the air temperature based on publicly available data;
2. To present a solution that performs better in 74.8% of the samples where the current post-processing solution achieves large errors;
3. To develop an air temperature historical database for training machine learning models.

Only real data are used in this work. Therefore, the air temperatures measured by weather stations were chosen as the target of the model. With the aim of trying to predict the measured temperature, the data selected as features included latitude, longitude, date, time, elevation, the output from the MEPS and the output from the GRIDPP.

Next, a section will be devoted to describing how the data were obtained and transformed, as well as the description of the neural network architectures used. After this, a section will discuss the benchmarks used and their performance during the training process. Finally, a discussion of the achieved results and how to access the data used in this work is presented in the following section.

## 2. Materials and Methods

### *2.1. Data Acquisition*

Although the MET makes all the data publicly available, first, it was necessary to gather all the data and transform them into a shape that was appropriate for training a neural network. The measured temperatures were obtained using the Frost API (https://frost.met.no/api.html accessed on 14 July 2022), which is a public API provided by MET; and the remaining data were available on the thredds website (https://thredds.met.no/thredds/catalog.html accessed on 14 July 2022), also owned by the MET.

#### 2.1.1. Obtain List of Forecasts

In order to make the process more efficient and require less computational processing, the first step was to build a Crawler that was capable of accessing the thredds website and scraping all the URL files for MEPS and GRIDPP. There were three main reasons for this decision. 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 were unavailable. Secondly, the forecast files were extremely large: one single file could be larger than 80 GB. Considering that over 1000 files must be used, it was not feasible to download all the files. This leads to the third reason: it is possible to open and use the files with the URL. By doing this, it is possible to save much time and there is no need to use local large storage space.

In this step, the years of interest were set. Afterwards, the program searched every month's page for every set year, which included the days available for each month. The result was a list with the URL of every day's page. Thus, the program iterated 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 results were stored in different JSON files, one for MEPS and another for GRIDPP. Beyond the file URL, some metadata, such as date and time information, were also saved to make the ETL process more feasible.

#### 2.1.2. Obtain Stations

This step consisted in obtaining the stations that were publicly available and had air temperature data for the declared years and their latitudes, longitudes and elevations.

As the endpoint that returns the air temperature does not provide the latitude and longitude, before progressing any further, it was necessary to make requests to the endpoint "/availableTimeSeries". Here, it was possible to query for stations that had air temperature data and set the available dates as a parameter, and then acquire the stations' IDs. Stations'

IDs were used for querying the latitude and longitude on endpoint "/sources" and, due to the large number of stations, it was necessary to split the endpoint request into a few different requests so as to not exceed the URI length limit.

To obtain the elevation data, which were not available in the Frost API, the Google Elevation API (https://developers.google.com/maps/documentation/elevation/overview accessed on 14 July 2022) was used. Their use is simple: once one has an API Key, it is only necessary to query using latitude and longitude for a point.

In the end, station ID, latitude, longitude and elevation were saved as a Comma Separated Value (CSV) file.

#### 2.1.3. Obtain All Forecasts

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

First, the JSON file that was the first step's output and the CSV output file of the last step were loaded. Thus, for the files described on JSON, the code used the netCDF4 library to open each URL file to obtain the temperature forecast for each station from the nearest point in the grid, and also the latitude and longitude of each grid point. Because of the special format of the netCDF4 file, before obtaining the temperature, it was necessary to extract all latitude and longitude coordinates from the file and build a grid object with the help of a function in the GRIDPP library. This grid object was used to translate the stations' locations into file indexes, and then it was possible to extract the data from the file using only these indexes.

Some key points to consider are listed below:

- Obtaining the forecast of the nearest point of the station in the grid is possible more easily using the method of GRIDPP object.
- The netCDF4 files have the results of all 10 ensemble members on the temperature forecast. To summarize the results, the mean forecast value was used.
- Despite a single file having the forecast for several hours, considering the amount of final data, we 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, we used the forecast for the hours 12:00, 13:00, 14:00, 18:00, 24:00 and 12:00 for the 2nd of July.
- As the neural network models will estimate the forecast of air temperature for several future hours, there are several forecast values for the same date and time from different files. That is, there may be multiple entries for 1 January 2022 at 12:00: one from the forecast that was made at that exact time, and another that was forecast 6 h earlier. The value that indicates the advance of the forecast in question is saved as a variable called "hour ahead".
- 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. For the purpose of replicating results, it is noted that the execution time of requests in Google Colaboratory was acceptable.

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

#### 2.1.4. Obtain All Observations

This step consisted in obtaining the air temperature observed by each station using the Frost API.

Initially, the Crawler output of the first step and the metadata from the stations of the second step were needed as input. The Crawler's output was used to determine which date and time to query in the Frost API, and the stations' metadata indicated which station to query.

The output of this step was a JSON file.

#### 2.1.5. ETL

The final step consisted in gathering the results of the previous steps in one single CSV file with the features and the label as columns. This transformation was performed by combining the station ID, year, month, day and hour columns through the files. At the end of the process, the final dataset had 11,543,361 samples.

### *2.2. Data Transforming*

Intending to prepare the data to be consumed by the neural network, the library Scikit-Learn [15] was used to perform the following transformations.

#### 2.2.1. Split Train, Validation and Test Set

As is known, in order to properly evaluate any supervised machine learning model, it is necessary to split the data into a train set and a test set at least, and in this work, it was chosen to use also a validation set to identify overfitting over the epochs. The 2020 data were set aside for training, the 2019 data for validation and the 2021 data for testing.

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

#### 2.2.2. Numerical Data Transformation

The features were transformed following two different procedures: one for the date-type features and another for the non-date features. For the date-type features, this information was modeled as a sine function, which means that the first day of a year was interpolated to 0 and the last day to $2\pi$, and the value that was used was the result of the sine function of the interpolated result. By doing this, we obtained a cyclic numerical representation for the date in such a way that the value assigned on the 31st of December would be close to the value assigned on the 1st of January, as represented in Figure 1.

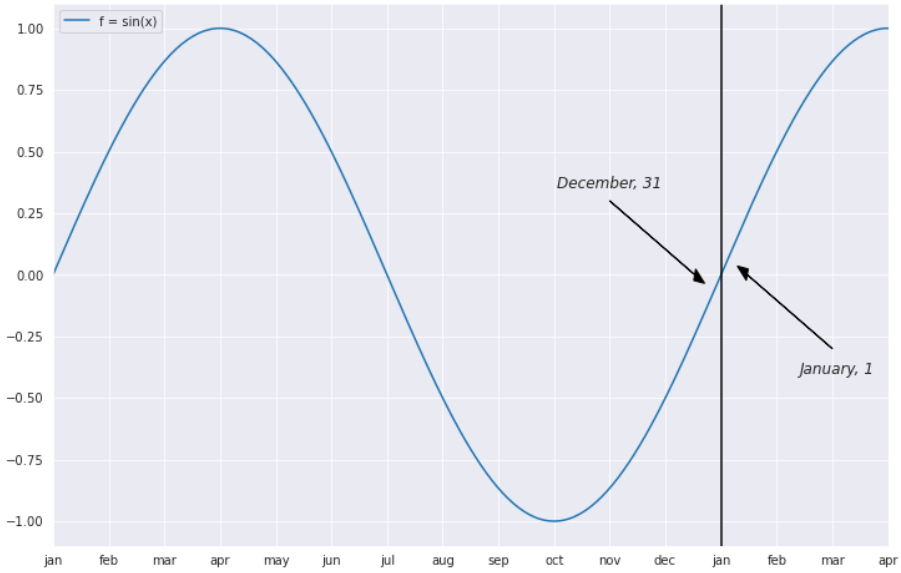


**Figure 1.** Sine function.

For the rest of the features, the numerical values were normalized using Equation (1), where $x$ represents the original value, $\mu$ represents the mean of the numbers, $\sigma$ represents the standard deviation and $z$ represents the new value for this variable.

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

### 2.3. Neural Network Architectures

For building the neural network in this section, the Tensorflow [16] and Keras [17] libraries were used. These libraries are largely used in the deep learning field. Two different architectures were evaluated: a multilayer perceptron (MLP) and a convolutional neural network (CNN), as described below. The maximum number of epochs was set as 20,000 epochs, the Mean Square Error (MSE) was used as the loss function, and the Adam algorithm was used as the optimizer, with a learning rate of 0.001. In addition, an early stopping method with 2000 epochs of patience was used.

The patience of the early stopping method was chosen through experiments, where the models were first trained for 20,000 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 started to increase systematically at around 4.5 k epochs for the CNN model and 2.5 k epochs for the MLP model, but decreased again, stagnating only after 12,000 epochs in the MLP and 16,000 in the CNN. Although the curves stagnated very late, we did not expect the training to rise 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 decreased significantly until a couple of thousand epochs. Therefore, based on this analysis, we chose a patience of 2000 epochs.

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, this work did not use these methods because of the large amount of input data, which would require a large amount of work and computational power. However, the application of these methods can be considered as future work.

1. Multilayer Perceptron: As represented by Figure 2, the proposed MLP has only two fully connected (FC) hidden layers with 64 and 16 neurons, respectively. These two hidden layers use the hyperbolic tangent (tanh) function as the activation function, Glorot initialization with a uniform distribution [18] for the parameters, and the output layer does not use any activation function.

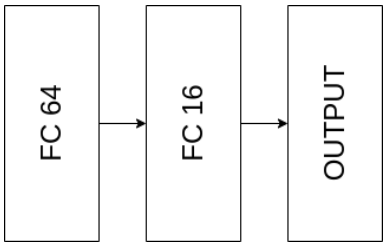


**Figure 2.** MLP representation.

2. Convolutional Neural Network: As represented by Figure 3, 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, with Glorot initialization with a uniform distribution [18] for the parameters. As with the MLP construction, all layers, except the last, have hyperbolic tangent (tanh) functions as activation functions. The use of one-dimensional and not two-dimensional convolutions was due to the data format, since the data used were from meteorological stations and they were not evenly distributed, geographically speaking. Thus, it would not be possible to directly represent these data in a two-dimensional matrix, as expected by a CNN with two-dimensional convolutions.

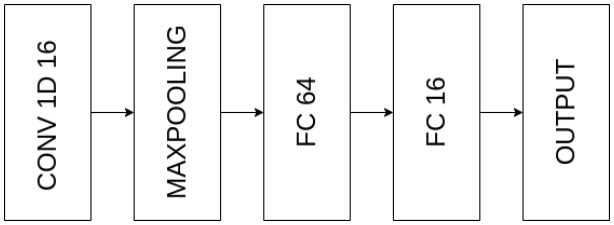


**Figure 3.** CNN representation.

The choice of these architectures, especially the number of neurons and kernel size, was made considering that the first layers should be larger than the range of the number of inputs (features) for generalization purposes, but not too large as to potentially increase the computational cost unnecessarily. The focus of this work was 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. This is encouraged for future work and practical implementation by the Norwegian Meteorological Institute.

## 3. Results

### *3.1. Experimental 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.

### *3.2. Benchmarks*

Before starting any neural network development, it is necessary to define a benchmark. Considering that the current solutions are the MEPS and the GRIDPP products, it is a good idea to use the error of these solutions as a benchmark, because, if the general error of the NN model is higher than the error of any of these solutions, there will be no reason to use an NN approach, as it would be more practical to stay with the current post-processing approach alone. As shown by Table 1, despite the error function chosen for training being the MSE, in order to make the error more readable and on the same scale of the data, future comparisons will be made using the Root Mean Square Error (RMSE). The percentage of errors greater than 3 degrees will also be considered, since errors greater than this limit are more worrisome.

**Table 1.** Benchmarks.

| Model | RMSE | Percentage of Errors Greater than 3 |
|---|---|---|
| MEPS | 5.258 | 54.05% |
| GRIDPP | 1.886 | 9.38% |

Analyzing the distance between the values predicted by MEPS and GRIDPP and the ones measured by the meteorological stations, we have Figures 4 and 5.

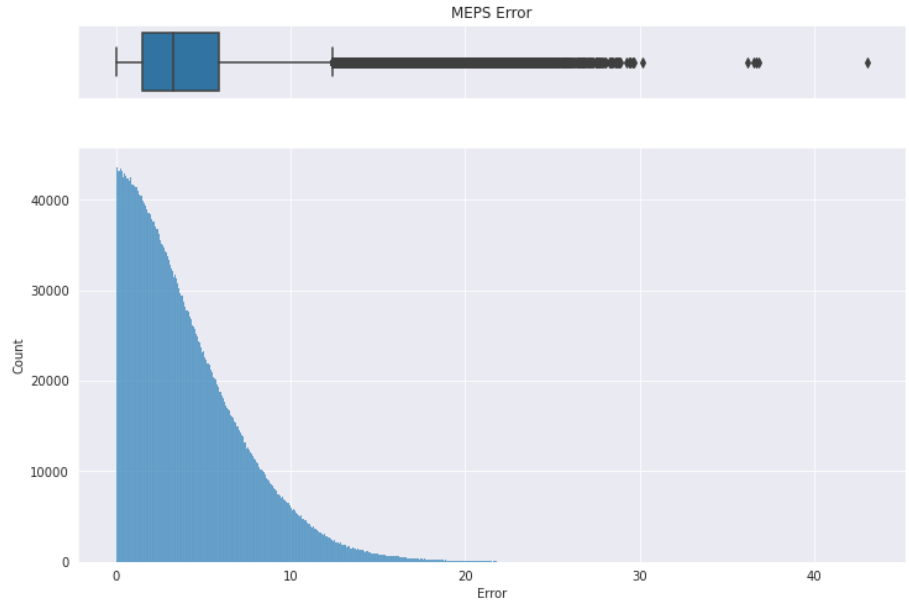


**Figure 4.** MEPS RMSE distribution.

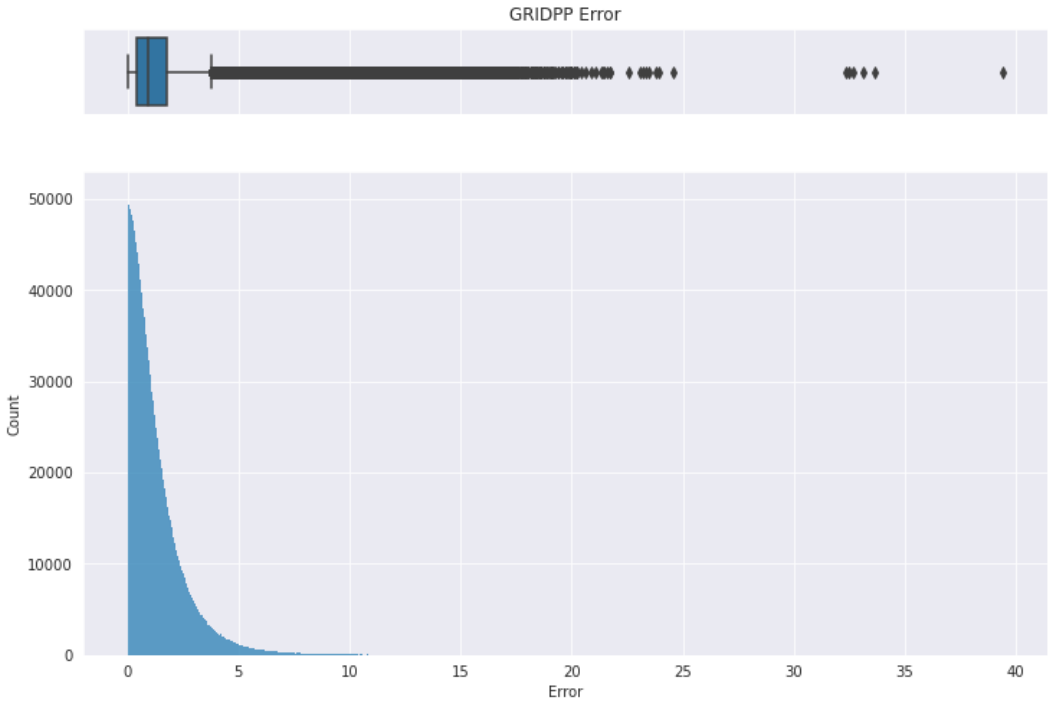


**Figure 5.** GRIDPP RMSE distribution.

### *3.3. Training Results*

Intending to understand whether the MEPS and GRIDPP values actually provide an upgrade on the training process, for each proposed architecture, three different models were evaluated:

1. A model using only the MEPS forecast, excluding the post-processed value from GRIDPP, as a feature;
2. A model using only the GRIDPP post-processed value, excluding the original forecast from MEPS; and
3. A model using both values, MEPS and GRIDPP, as features.

#### 3.3.1. Multilayer Perceptron

Comparing the MLP models, we have the following results in Table 2 and the learning curves in Figure 6.

**Table 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 |

Looking at Table 2, it is possible to notice that an MLP cannot replace the GRIDPP as the only post-processing step, since the RMSE of the pure GRIDPP was halved compared to the MLP using only MEPS data, and the NN was not able to converge much further than the early stopping patience, as seen in Figure 6. 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 analyses comparing this architecture with the benchmarks were conducted, counting the number of errors greater than 3 K in each case, split by the variable "hour ahead" to understand how the error increases as the hours pass and uncertainty increases. This analysis is represented in Figure 7.

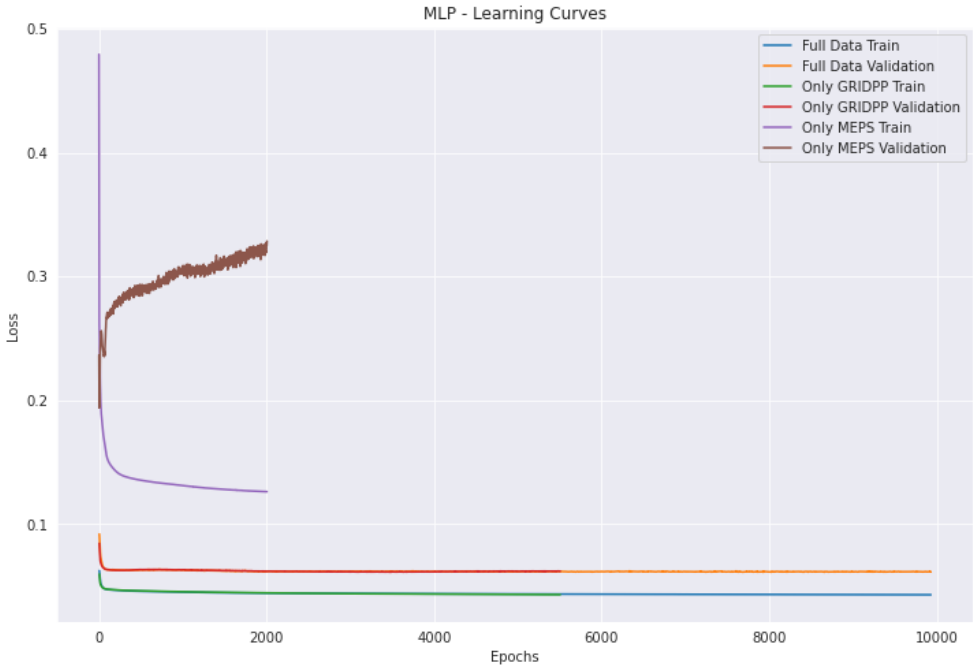


**Figure 6.** MLP learning curve.

Given the critical error threshold of 3 K, the behavior of GRIDPP and the neural network models was plotted in situations where the GRIDPP error was higher than 3 K for the test sample. Figure 8 represents the error higher than 3 K for GRIDPP and Figure 9 represents the error of the MLP model when the GRIDPP error was higher than 3 K. The MLP model achieved a smaller error in 74.8% of the samples where the GRIDPP error was greater than 3 K.

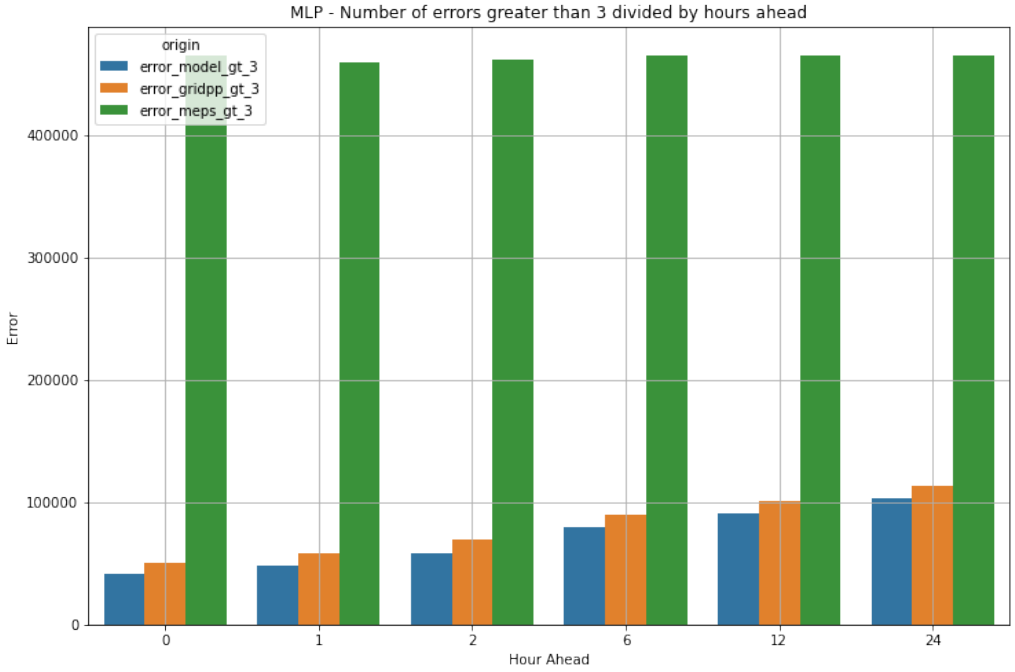


**Figure 7.** Comparing errors of MLP to the benchmark.

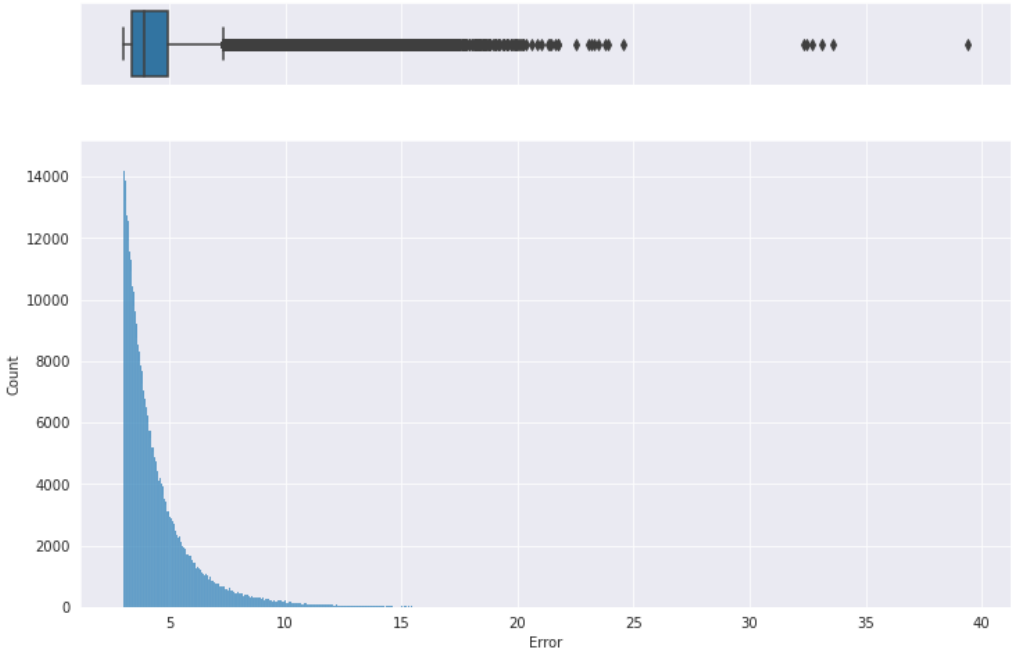


**Figure 8.** GRIDPP error higher than 3 K.

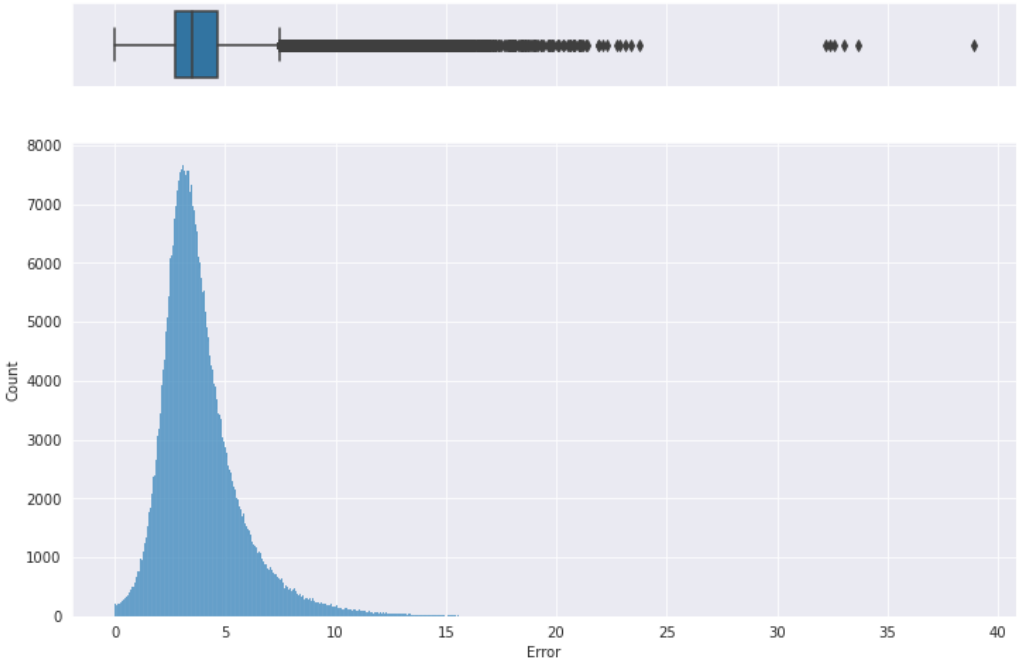


**Figure 9.** MLP error in cases when GRIDPP error is higher than 3 K.

### 3.3.2. Convolutional Neural Network

Comparing the CNN models, we have the following results in Table 3 and the learning curves in Figure 10.

**Table 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 | 12,063 |

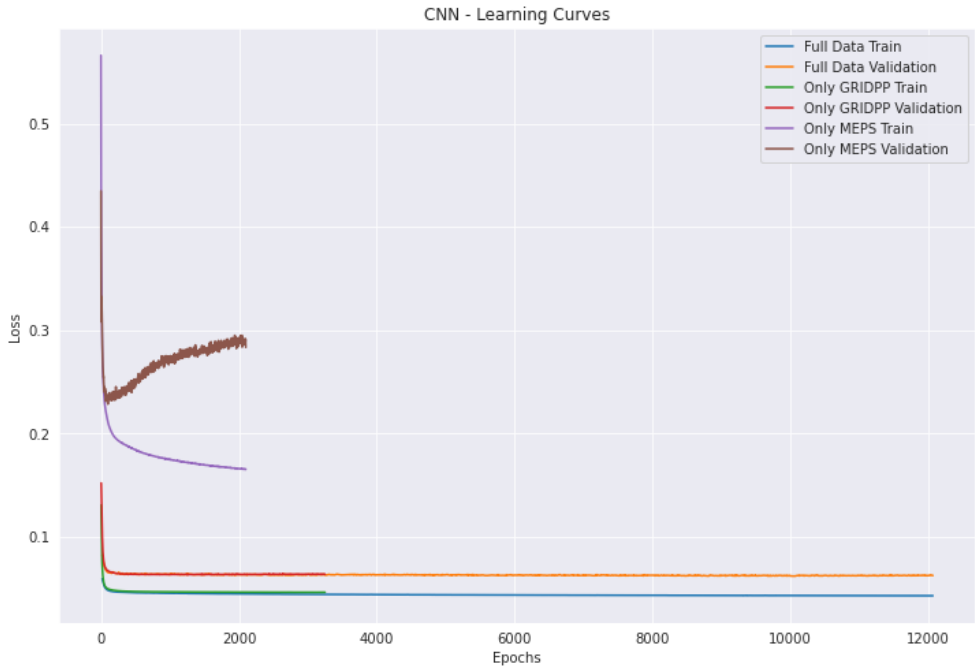


**Figure 10.** CNN learning curve.

The results above were actually similar to the MLP results. Once more, the combination of MEPS and GRIDPP as inputs of the neural network showed better performance than either of them alone. Therefore, the same combination was chosen to be compared to the benchmark, as shown in Figure 11.

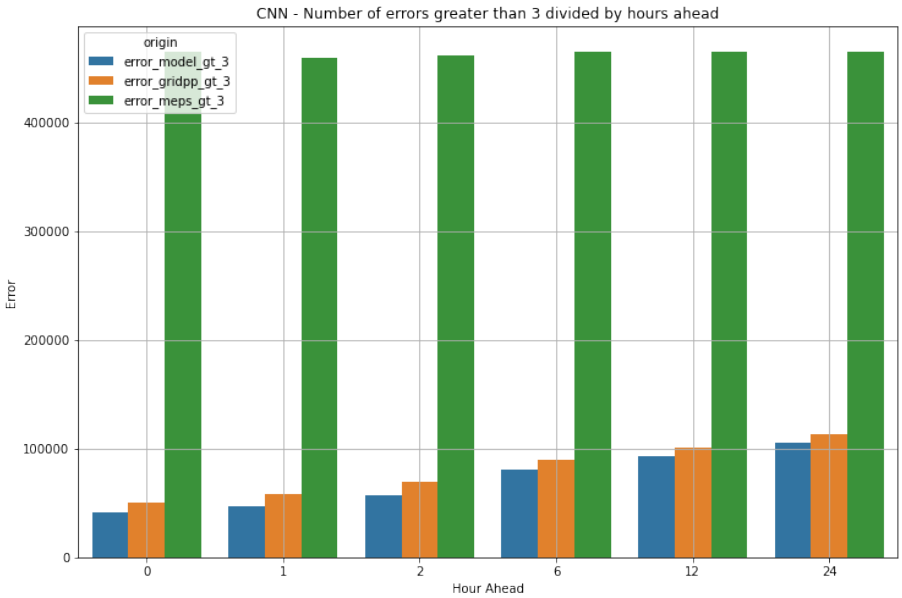


**Figure 11.** Comparing errors of CNN to the benchmark.

In the same way as was plotted for the MLP results, Figure 12 represents the CNN error distribution when the GRIDPP has errors greater than 3 K. The CNN achieved a smaller error in 73.8% of the samples where the GRIDPP error was greater than 3 K.

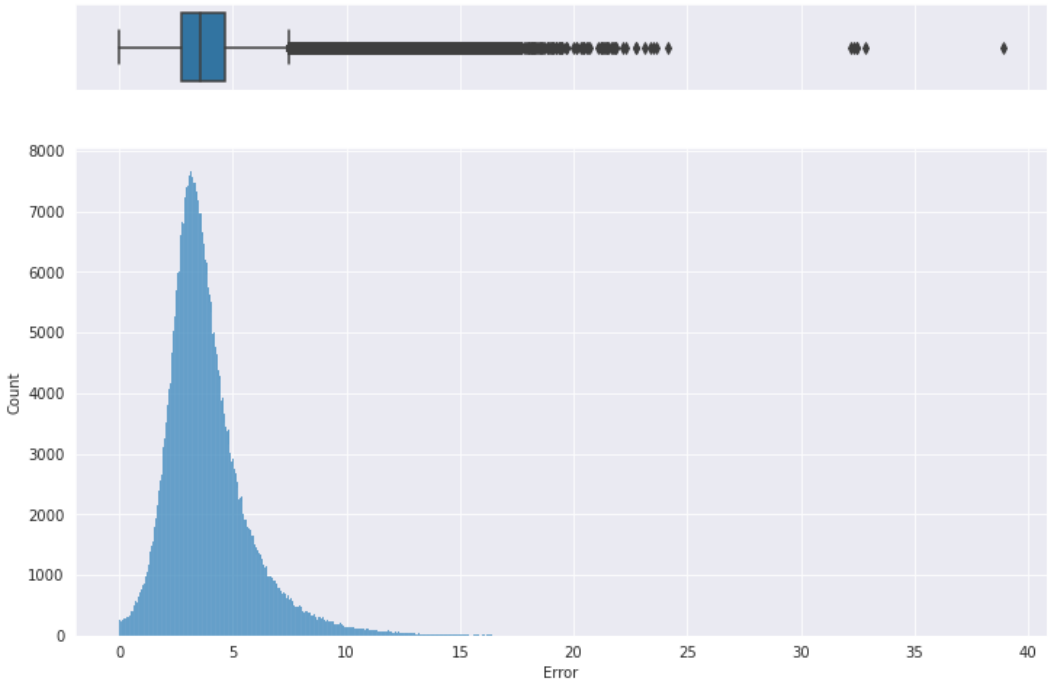


**Figure 12.** CNN error in cases when GRIDPP error is higher than 3 K.

In Figure 13, 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 very close, so it was not possible to distinguish the two lines.

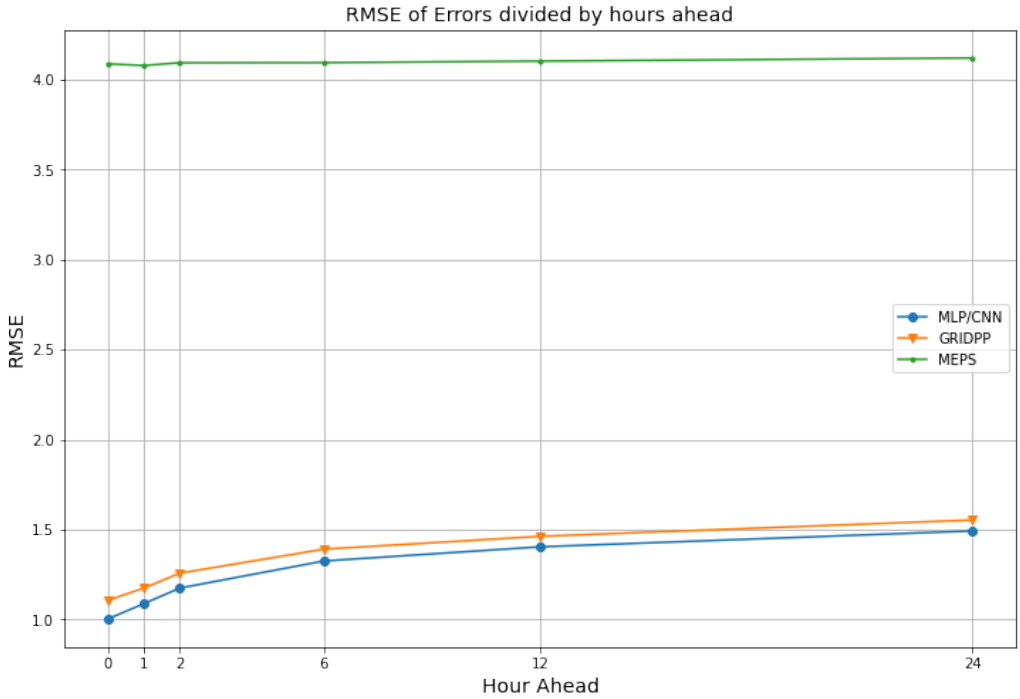


**Figure 13.** Comparing the RMSE by hours ahead.

## 4. Discussion

To guide the discussion, Table 4 provides the results of the experiments.

**Table 4.** Test set results.

| 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 |

Despite the current numerical weather forecast method being excellent compared to other regional forecast methods, a post-processing method could increase the accuracy significantly. For the purposes of post-processing, the GRIDPP has been shown to be a wonderful tool to improve and downscale 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 Table 4, it is possible to see that both proposed neural network architectures performed well on the test set, achieving a smaller RMSE than the benchmark and significantly fewer errors greater than 3 K. It is also interesting to notice in Figures 7 and 11 how the quantity of errors greater than 3 K increases with the growth of the forecast range. This is expected on real data, and we can see in each range value that both NN models achieved a better result.

Although the MLP and CNN models have some errors greater than 3 K, both models obtained excellent results, reaching a smaller error in, respectively, 74.8% and 73.8% of the samples between the test samples where the GRIDPP error was greater than 3 K. The RMSE in these two cases also decreased by approximately 0.5 K, as shown in Table 4.

## 5. Conclusions

This work proposed a neural network solution for improving the post-processing of air temperature forecasts. Results showed that it is possible to reduce the forecast error, paving the way for more accurate meteorological products. In addition, it allows the forecast estimation for any area of interest, since the proposed model uses latitude and longitude coordinates as inputs. This has the potential of reducing the computational effort for calculating the forecast for a point outside the original grid, since traditional numerical models use static grids. This should be investigated in future work.

The MLP model achieved an RMSE of 1.786 K, against 1.886 K for the benchmark, which is a tool developed by the Norwegian Meteorological Institute that uses many more weather stations than the proposed solution in this work. In addition, it was able to achieve a smaller error in 74.8% of the samples where the error of the GRIDPP product was greater than 3 K. In these cases, the reduction in the RMSE was of approximately 0.5 K. In terms of air temperature forecast, 0.5 K is a significant error reduction. This opens the way for the practical implementation of the proposed solution in combination with the current GRIDPP product of the Norwegian Meteorological Institute.

As future work, it is desirable to use more air temperature observations, such as public and private sensor data providers, homemade weather stations, sensors embedded in cars and IoT devices that could make the measurements available in the cloud. In addition, other architectures of CNN and MLP models should be evaluated, as well as new architectures, such as graph convolutional neural networks. These do not rely on the input being a regular matrix and could take advantage of neighborhood information, as the air temperature likely exhibits spatial behavior.

**Author Contributions:** Conceptualization, F.A.A.A. and G.A.; methodology, F.A.A.A. and G.A.; software, G.A.; validation, F.A.A.A. and G.A.; formal analysis, F.A.A.A. and G.A.; investigation, F.A.A.A. and G.A.; resources, F.A.A.A. and G.A.; data curation, F.A.A.A. and G.A.; writing—original draft preparation, G.A.; writing—review and editing, F.A.A.A.; visualization, G.A.; supervision, F.A.A.A.; project administration, F.A.A.A.; funding acquisition, F.A.A.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** Funding for this work was provided by the Research Council of Norway (AutonoWeather project, grant number 301575).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The final dataset resulting from the steps described here in Section 2.1 is available at https://doi.org/10.34740/kaggle/dsv/3635866 (accessed on 14 July 2022).

**Acknowledgments:** We would like to acknowledge the support from the Norwegian Meteorological Institute.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| NN | Neural Network |
| MLP | Multilayer Perceptron |
| CNN | Convolutional Neural Network |
| MSE | Mean Square Error |
| RMSE | Root Mean Square Error |
| CSV | Comma Separated Value |
| ETL | Extract-Transform-Load |
| MET | Norwegian Meteorological Institute |

| | |
|---|---|
| NWP | Numerical Weather Prediction |
| MEPS | MetCoOp Ensemble Prediction System |
| GRIDPP | Gridded Post-Processor |

## References


1. Ynoue, R.Y.; Reboita, M.S.; Ambrizzi, T.; da Silva, G.A. *Meteorologia: Noções Básicas*; Oficina de Textos: São Paulo, Brazil, 2017.
2. Met Office. History of Numerical Weather Prediction. Available online: https://www.metoffice.gov.uk/weather/learn-about/how-forecasts-are-made/computer-models/history-of-numerical-weather-prediction (accessed on 1 December 2021).
3. Vannitsem, S.; Bremnes, J.B.; Demaeyer, J.; Evans, G.R.; Flowerdew, J.; Hemri, S.; Lerch, S.; Roberts, N.; Theis, S.; Atencia, A.; et al. Statistical Postprocessing for Weather Forecasts: Review, Challenges, and Avenues in a Big Data World. *Bull. Am. Meteorol. Soc.* **2021**, *102*, E681–E699. [CrossRef]
4. The Norwegian Meteorological Institute. Available online: https://www.met.no/en (accessed on 1 December 2021).
5. Nipen, T. NWP Docs. Available online: https://github.com/metno/NWPdocs/wiki (accessed on 1 December 2021).
6. Nipen, T. Gridded Post-Processor. Available online: https://github.com/metno/gridpp (accessed on 1 December 2021).
7. Instituto Nacional de Meteorologia. Previsão Numérica-Modelo. Available online: https://portal.inmet.gov.br/servicos/previs%C3%A3o-num%C3%A9rica-modelo (accessed on 1 December 2021).
8. Navy, B. MODELO COSMO. Available online: https://www.marinha.mil.br/chm/dados-do-smm-paginas-modelagem-numerica/modelo-cosmo (accessed on 1 December 2021).
9. CPTEC/INPE. About BRAMS. Available online: http://brams.cptec.inpe.br/about/ (accessed on 1 December 2021).
10. Grönquist, P.; Yao, C.; Ben-Nun, T.; Dryden, N.; Dueben, P.; Li, S.; Hoefler, T. Deep learning for post-processing ensemble weather forecasts. *Philos. Trans. R. Soc. A* **2021**, *379*, 20200092. [CrossRef] [PubMed]
11. Kudo, A. Statistical Post-Processing for Gridded Temperature Prediction Using Encoder–Decoder-Based Deep Convolutional Neural Networks. *arXiv* **2022** , arXiv:2103.01479 .
12. de Ruiter, B. Post-processing Multi-Model Medium-Term Precipitation Forecasts Using Convolutional Neural Networks. *arXiv* **2021**, arXiv:2105.07043.
13. Han, L.; Chen, M.; Chen, K.; Chen, H.; Zhang, Y.; Lu, B.; Song, L.; Qin, R. A deep learning method for bias correction of ECMWF 24–240 h forecasts. *Adv. Atmos. Sci.* **2021**, *38*, 1444–1459. [CrossRef]
14. Rasp, S.; Lerch, S. Neural networks for postprocessing ensemble weather forecasts. *Mon. Weather Rev.* **2018**, *146*, 3885–3900. [CrossRef]
15. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
16. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Available online: tensorflow.org (accessed on 14 July 2022).
17. Keras. 2015. Available online: https://keras.io (accessed on 14 July 2022).
18. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, Sardinia, Italy, 13–15 May 2010; pp. 249–256.