STATISTICAL POSTPROCESSING OF LOCAL NUMERICAL WEATHER PREDICTION MODEL FORECASTS USING DEEP LEARNING

by

Yaşar Harun Kıvrıl

B.S., Industrial Engineering, Boğaziçi University, 2020

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor Mustafa Gökçe Baydoğan for his invaluable guidance, support, understanding, and patience. Without his continuous effort, it would be impossible to complete such a study.

My gratitude extends to the Algopoly team for their support. Especially I owe special thanks to my colleagues Burak Tabak, and Uğur Parkın for their friendship, motivation, and generous help.

I would like to thank my partner Ceren Cinek for always being there. Her endless support was precious in this course and more.

I also would like to thank my close friends Mustafa, Salih, and Haki for their motivation and help. Their support helped me a lot to go through stressful days.

Last but not least, I would like to thank my family for their indispensable love, support, and understanding. I appreciate their encouragement and their confidence in me. My sister, Betül deserves special thanks for cheering me up with Azu photos every day.

ABSTRACT

STATISTICAL POSTPROCESSING OF LOCAL NUMERICAL WEATHER PREDICTION MODEL FORECASTS USING DEEP LEARNING

Accurate weather forecasts play a crucial role in many decision-making processes. Currently, the main supply of weather forecasts is numerical weather prediction (NWP) which solves physical equations to predict future states of the atmosphere. However, the NWP models are prone to rapidly growing errors from the initial states, boundary conditions, and model structures. In order to fix these systematic errors in the forecasts, statistical postprocessing methods are used. In this study, three alternative deep learning architectures are proposed to statistically postprocess Global Ensemble Forecasting System (GEFS) forecasts of the Aegean Region of Turkey. The postprocessing is done to multiple weather variables at multiple pressure levels. The input and output structure of the models also introduced an extrapolation capability. The models are trained with sixteen years of data, and the hyperparameters are tuned with one-year validation data and tested over the last three years. The results are investigated from the variable, pressure level, and location aspects. Fully convolutional and its U-Shaped extension present promising results in every aspect. The U-Shaped architecture is chosen over the others considering its lower mean, and lower variance in error distributions. Also, the error distribution of extrapolated values validates the extrapolation capability of the model. Finally, a case study on wind power forecasting of 19 power plants shows that the method obtains better forecasts in a real-world application.

ÖZET

BÖLGESEL SAYISAL HAVA DURUMU TAHMİN MODELLERİNİN TAHMİNLERİNİN DERİN ÖĞRENME KULLANARAK İSTATİSTİKSEL ARDİŞLEMESİ

Doğru hava tahminleri, birçok karar verme sürecinde çok önemli bir rol oynamaktadır. Şu anda hava tahminlerinin ana kaynağı, atmosferin gelecekteki durumlarını tahmin etmek için fiziksel denklemleri çözen sayısal hava tahmin (NWP) modelleridir. NWP modelleri, başlangıç durumlarından, sınır koşullarından ve model yapılarından kaynaklı hızla büyüyen hatalara sebebiyet verebilmektedir. İstatistiksel ardişleme yöntemleri tahminlerdeki bu sistematik hataları düzeltmek için kullanılan yöntemlerden biridir. Bu çalışmada, Türkiye'nin Ege Bölgesi'nin Global Ensemble Forecasting System (GEFS) tahminlerini istatistiksel olarak ardişlemek için üç alternatif derin öğrenme mimarisi önerilmiştir. Ardişleme, çoklu basınç seviyelerinde çoklu hava değişkenlerine yapılmıştır. Ayrıca, modellerde kullanılan veri yapısı modellere bir ekstrapolasyon yeteneği vermiştir. Modeller on altı yıllık verilerle eğitilmiş, hiperparametreler bir yıllık doğrulama verileriyle ayarlanmış ve son üç yılda test edilmiştir. Sonuçlar hava durumu değişkeni, basınç seviyesi ve konum yönlerinden incelenmiştir. Tamamen evrişimli model ve onun U-Şekilli uzantısı, analizlerde umut verici sonuçlar sunmuştur. U-Şekilli mimari, hata dağılımlarında daha düşük ortalama ve daha düşük varyans içermesi sebebiyle diğer modellere tercih edilmiştir. Ayrıca, tahmin edilen değerlerin hata dağılımı, modelin ekstrapolasyon kapasitesinin etkisini göstermiştir. Son olarak, 19 enerji santralinin rüzgar enerjisi tahmini üzerine bir vaka çalışması yapılmış ve yöntemin gerçek hayattaki bir uygulamada daha iyi sonuçlar ortaya çıkardığı görülmüştür.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$L(.)$	Loss function
M	Number of predictors
N	Number of instances
q	Specific humidity
r	Relative humidity
tmp	Temperature
u	U component of wind
V_t	Momentum values at iteration t
v	V component of wind
W_t	Weight matrix at iteration t
w	Vertical component of wind
X	Input matrix
x	Input values
Y	Output matrix
y	Output values
$\hat{y}$	Model output values
α	Numerical weather prediction model
δ	Momentum parameter
λ_1	Penalization parameter for the absolute sum of weights
λ_2	Penalization parameter for the square sum of weights
ρ	Learning rate
∇	Gradient operator

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
ADAM	Adaptive Moment Optimization
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
ECMWF	European Centre for Medium-Range Weather Forecasts
EMOS	Ensemble Model Output Statistics
ERA5	ECMWF Reanalysis 5th Generation
GAMLSS	Generalized Additive Models for Location Scale and Shape
GEFS	Global Ensemble Forecasting System
GWh	Gigawatt Hour
MB	Milibar
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
MWh	Megawatt Hour
NOAA	National Oceanic and Atmospheric Administration
NWP	Numerical Weather Prediction
RMSProp	Root Mean Square Propagation
WMAPE	Weighted Mean Absolute Percentage Error

1. INTRODUCTION

Weather forecasts are significant for many applications as a part of their decision-making processes. While they are everyday commodities to plan activities for the public, they also have specific applications in many sectors. In aviation and marine, they are used for scheduling flights/trips and ensuring the safety of the passengers or crews. In agriculture, they allow to plan irrigation as well as planting and harvest times. Especially in the energy sector, the forecasts have a crucial role. The demand for electricity is highly affected by the weather conditions. On the other side, the renewable sources mainly generate electricity using weather events. Having an idea about the demand and supply beforehand allows future projections of the energy market. These projections are used for planing the energy trades and many other energy related events. Additionally, the forecasts save lives with extreme weather predictions that warn people about disastrous events. It is estimated that the weather forecasting services in the US create a value of $31.5 billion per year [1]. Since the weather forecasting methods are getting more and more accurate with time, the decisions supported by them get better and they create more value for the people.

The main sources of accurate forecasts are the numerical weather prediction (NWP) models. NWP models start with an initial condition and solve the atmospheric motion equations through time to obtain the future states of the atmosphere. While solving the equations, the models create a grid of the area of interest and provide forecasts for different weather variables at different vertical values on the grid (i.e value of relative humidity variable at longitude 25.75, latitude 32.5, and 800 mb pressure level). NWP models are published by various meteorological institutes operationally and are widely used throughout the world. However, they are prone to errors from different sources. Firstly, they are sensitive to the initial conditions [2]. The errors made in the initial conditions grow rapidly and the errors make the forecasts useless after a level. Also, boundary condition errors and model structural errors diminish the model accuracy [3, 4]. All of these systematic and random errors grow with a snowball effect

due to the chaotic nature of the atmospheric dynamics. The institutes developed the ensemble NWP systems where the forecasts consist of different initial conditions and different model structures. The ensembles achieved to have more robust forecasts by quantifying the uncertainties. However, even with the recent developments, the NWP forecasts reveal systematic biases and dispersion of ensembles that need postprocessing to enhance the forecast accuracy [5].

Statistical postprocessing methods are widely used for correcting these problems of the NWPs. The statistical processing methods can be categorized as methods with distribution assumption and without distribution assumption or in other terms parametric and non-parametric methods. Approaches that make distribution assumptions about the forecasts, try to estimate the parameters of the distribution. Bayesian model averaging [6], ensemble model output statistics (EMOS) [7], boosting approach [8], general additive models for location shape and scale [9], random forests, [10] and artificial neural networks [11] are all employed to fit parameters of a predetermined forecast distribution. On the other side, there are methods where no distribution assumption is made. The studies for these postprocessing methods mainly focus on quantile regression and its variants [12–16]. Also, there exist analog models [17–20] and artificial neural network models [21] that estimate directly the forecasts instead of their distribution which are shown to be promising. The methods for statistical postprocessing need to deal with some challenges. The weather events are the combination of spatial, temporal, and multivariate relations and the output of the postprocessing model is expected to be physically viable. Also, the NWP models are continuously developed and their new versions are released. Since these version changes affect the systematic bias, the postprocessing method should be designed to adapt to these changes or the NWP sources need to release the historical reforecasts of the latest version. The reforecasts create proper training data by providing the forecasts with the same bias structure for decades. Global Ensemble Forecasting System (GEFS) is one of the NWP models that provide publicly available reforecasts that goes back to 2000. Lastly, the postprocessing method is required to be timely. Getting the NWP outputs already takes reasonable time and after a long postprocess, the forecasts can become useless. Since

most machine learning methods can make predictions in seconds, they are considered proper candidates for timeliness.

In this thesis, three alternative non-parametric artificial neural network structures are implemented to postprocess multiple weather variables in a local area in the Aegean Region of Turkey consisting of 17×19 grid points. The multivariate structure is used to allow the networks to model the relations between variables. As input, GEFS control reforecasts from 2000 to 2019 are used to have large training data without version changes. Additionally, GEFS forecasts are operational which makes this postprocessing study operationally available. Just like [21–24], ERA5 Reanalysis in $0.25° \times 0.25°$ resolution is selected as the target. Reanalysis datasets are the corrections of weather forecasts after obtaining the actual observations. The resolution of the ERA5 is the same as GEFS which makes it a suitable target choice. Then, a multi-layer perceptron structure, a fully convolutional structure, and U-Shaped skip connections added as an extension to the fully convolutional model are tuned to minimize the mean square error. After that, a detailed analysis of the outputs for each structure is made for unseen data consisting of three years. Finally, a simple case study to predict wind power generation in 19 plants in the region is made to show the effectiveness of the postprocessing method.

This thesis is organized as follows: Chapter 2 consists of a detailed literature review of statistical postprocessing methods. Background information about artificial neural networks, penalized regression approaches and performance metrics are explained in Chapter 3. Chapter 4 briefly describes the methodology under the data sources, data preprocessing, and proposed architecture headings. The experiment settings for modeling pipeline, hyperparameter tuning, and wind power forecasting are included in Chapter 5. In Chapter 6, the results of each architecture and wind power forecasting task are discussed in depth. Lastly, the conclusion of the thesis together with potential future works are presented in the final chapter.

2. LITERATURE REVIEW

The methods for statistical postprocessing can be categorized as methods with distribution assumption and without distribution assumption. These categories are also called as parametric and non-parametric methods. The approaches with distributional assumptions select a proper distribution family based on the weather variable and the parameters of the distribution are related to NWP outputs using regression coefficients. After that, the coefficients are estimated by minimizing a loss function such as continuous ranked probability score [25]. Bayesian model averaging is used to calibrate the forecast ensembles [6]. Ensemble model output statistics (EMOS) as covariates to postprocess the variables [7]. A study proposes a boosting approach for algorithmic selection of the most useful features to estimate the regression coefficients without overfitting [8]. Generalized additive models for location scale and shape (GAMLSS) method uses additive functions to create an estimation of the distribution parameters [9]. Another study blends decision trees and random forests to this method to recursively partition the space and fit different forecast distributions to each partition [10]. Artificial neural networks (ANN) are used for forecast distribution parameter estimation [10]. The flexible structure of ANNs and their ability to estimate almost any function using a sequence of nonlinearities in activation functions make them a perfect candidate for postprocessing tasks. This method is able to model arbitrary relations between the predictors and the distribution parameters. Also, the kernel dressing approach [26] and fitting kernel estimations of ensemble members [27] can be counted as a parametric approach.

On the other hand, the methods without distributional assumptions work directly on values and try to come up with a predictive distribution to asses forecast uncertainty. A pioneering study suggests the use of quantile regression to approximate the predictive distribution [27]. This method allows for the estimation of specified quantile values. Later, the quantile regression approach is further extended. The Bayesian approach is used to regularize the model [13]. The constrained spline quantile regression method

is employed to extract information from all ensemble members [14]. An extreme level-based local quantile regression method is developed to fill the insufficiency of quantile regression in extreme values [15]. The quantile regression forest method is also used in statistical postprocessing [15]. In this method, the quantile estimates come from random forests. Another decision tree-based method called ecPoint is developed to postprocess precipitation forecast for the whole world and it is currently operational [28]. Apart from prespecified quantiles, some studies try to estimate a quantile function of predictions. Neural networks and Bernstein polynomials are combined together to fit a quantile function [29]. Another method tries to approximate the histogram of the distribution using convolutional neural networks after discretizing the target [30]. Analog methods are also used for postprocessing [17–20]. These methods try to find a similar historical case for the input. However, the computational cost of finding an analog grows as the training sets get larger and larger. Lastly, deep learning is used in one study to postprocesses the whole world for bias correction and uncertainty quantification [21]. The study shows that artificial neural networks give promising results in postprocessing tasks.

There are a few challenges in statistical postprocessing. Weather events are a combination of spatial, temporal, and multivariate information and postprocessing methods are expected to preserve these relations. Despite the analog models are automatically solving this problem, the other methods need adjustments to have this relation. As a method with distributional, a special multivariate distribution is proposed to preserve the physical relationships in low dimensions [31]. On the non-distributional side, ensemble copula coupling [32] and the Schaake shuffling [33, 34] methods are proposed to avoid physically unrealistic outputs. Another issue is the version changes of the NWP models. The NWP models are improving every day and new versions of the models are released based on these improvements. However, the bias structure may be subject to change with the new models. This makes the outputs of the previous versions less useful for postprocessing and limits the training data. In order to have an ideal training set, some NWPs release the reforecasts of the new versions. These reforecasts are the historical forecasts of the new version with the previous initial con-

ditions. The reforecasts create proper training data for statistical postprocessing that extends for many decades. Large-scale data also raises problems about data volumes and timeliness. Dealing with decades of multivariate data for a large grid on different atmospheric levels is computationally expensive. Furthermore, the postprocessing of this data is useful if it is released on time. Therefore, the postprocessing times should be as small as possible to allow operational usage. Machine learning methods are good candidates for having timely postprocessed forecasts. Even though the training time can be long, most of the algorithms make the prediction in a few seconds.

In this thesis, three different alternative artificial neural network structures are implemented to postprocess multiple weather variables in a local area consisting of 17×19 grid points in the Aegean Region of Turkey. The multivariate structure is used to allow the networks to model intervariable relations. Unlike many other studies that focus on one or two variables at one or a few pressure levels, this thesis conducts postprocessing in multiple variables at multiple pressure levels. As input, GEFS control reforecasts from 2000 to 2019 are used to have large training data without version changes. Additionally, GEFS forecasts are operational which makes this postprocessing study operationally available. Just like [21–24], ERA5 Reanalysis in $0.25° \times 0.25°$ resolution is selected as the target. Reanalysis datasets are the corrections of weather forecasts after obtaining the actual observations. The resolution of the ERA5 is the same as GEFS which makes it a suitable target choice. Additionally, ERA5 has more time resolution and more pressure levels. In order to preserve information on the physical relations in ERA5, the extra values are kept during the training. This allows the proposed models to have an extrapolation capability. The extrapolation capability of the model also separates this thesis from the previous studies. After determining the input and the target, a multi-layer perceptron structure, a fully convolutional structure, and U-Shaped skip connections added as an extension to the fully convolutional model are tuned to minimize the mean square error. Then, a detailed analysis of the outputs for each structure is made for unseen data consisting of three years. Finally, a case study to predict one year of wind power generation in 19 plants in the region is made to show the effectiveness of the postprocessing method.

3. BACKGROUND

3.1. Tensors

In mathematics, a single value with no index is a scalar. An array of values with a single index i is called a vector and usually denoted as $\mathbf{a}_i$. If values in the array is indexed with two indices i j, it is called as a matrix and denoted as $\mathbf{A}$ with elements $\mathbf{a}_{ij}$. For the arrays with more than two indices, a more general term tensor is used. In general, a tensor is an array with a variable number of indices. Tensors are denoted as $\mathbf{A}$ with its elements with variable number of indices $\mathbf{a}_{ijk\ldots}$.

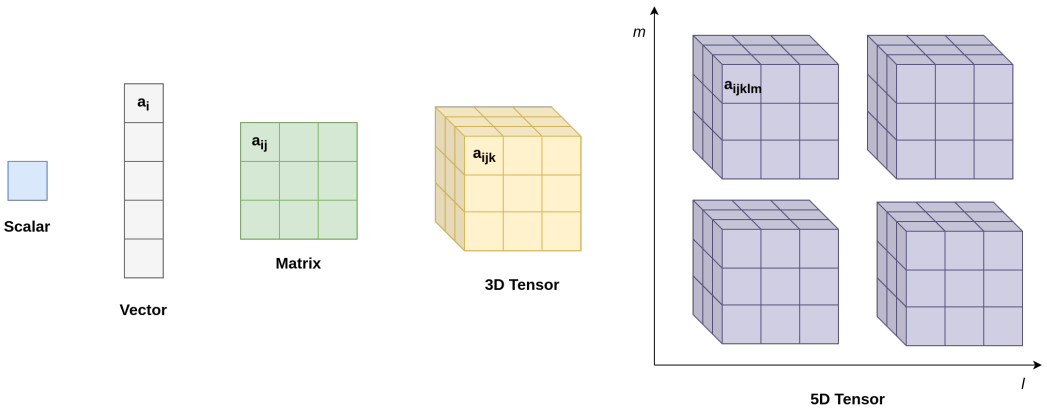

Figure 3.1. Illustration of scalar, vector, matrix, and tensor.

Tensors are practical data representations. Allowing multiple indices, the data can be stored in tensors with its relations (i.e spatial relations, temporal relations). Some of the learning algorithms employ these relations to enhance their performance.

3.2. Artificial Neural Networks (ANN)

Artificial neural networks are computational models that are inspired by the working principle of the nerve cell in the brain. Even though they are not proper

models for nerve cells, ANNs have interconnected artificial neurons with activation functions similar to the nerve system. Combining these artificial neurons in various ways returned successful results in many pattern recognition tasks. This success led to the ANN's popularity and its application in diverse areas. From image processing to natural language processing, time series forecasting to audio processing, variations of ANN are used to create state-of-the-art works.

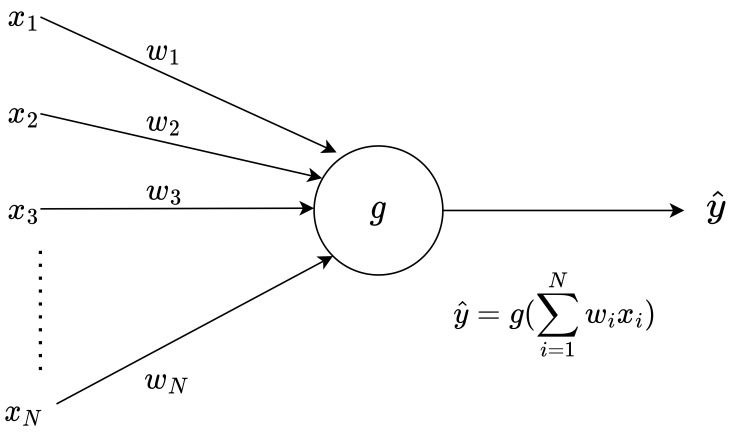

Figure 3.2. Demonstration of a single artificial neuron.

Essentially, ANNs try to estimate a function f between its input X and its output y using the weights W in neuron connections. The activation function g in the neurons introduces non-linearity and allows the estimation of the complex structures. During the estimation, the performance of the network is measured by a quantity called loss value l. The loss value is calculated using a loss function L (i.e. mean squared error). The ultimate goal of an ANN is to minimize the expected loss value of the estimation by minimizing the loss function. The loss function can be denoted as

$$\min_{W} \quad L(f(X)|y), \tag{3.1}$$

where W is the weights of the network, X is the input data and y is the target values.

The optimization of the loss function is generally made by gradient-based algorithms. The directed graph structure of the ANNs allows using a method called

back-propagation. Back-propagation algorithm allows for calculating the gradients of the weights with respect to the loss function using the chain rule. Firstly, the weights are initialized randomly. After obtaining the gradients the connection weights of the network are updated using a gradient descent algorithm. Gradient descent algorithms adjust the weights in the opposite direction of the gradient to minimize the loss. However, using these algorithms introduce a hyperparameter called learning rate (ρ) to determine the amount of the update. Having too small learning rate can cause long training times and larger learning rates create convergence problems. Therefore, this parameter needs to be tuned properly. The weight update using gradient descent algorithm is expressed as

$$W_{t+1} = W_t + \rho \nabla_W L(f(X)|y), \tag{3.2}$$

where W_t is the weights at current iteration and W_{t+1} is the weights of next iteration.

Although the learning rate is tuned, the minimization of the loss function can be enhanced and accelerated. Using momentum (δ) is one way to do that. Instead of going to the inverse gradient direction at each iteration separately, this approach calculates another direction V_t from all previous gradients by weighting them in an exponentially decaying manner. This new direction accelerates the training by providing a general direction for minimization. Additionally, it helps to avoid local minimum and converges to better loss values. However, this method introduces a momentum parameter δ between 0 and 1 that needs to be tuned. Using the momentum the weight update procedure can be modified as

$$\begin{aligned} V_{t+1} &= \delta \nabla_W L(f(X)|y) + (1-\delta) V_t \\ W_{t+1} &= W_t + \rho V_{t+1}, \end{aligned} \tag{3.3}$$

where V_t is the direction in the current iteration and V_{t+1} is the new direction.

Another way to accelerate the training is using a stochastic version of the gradient descent algorithm (SGD). Since the data in machine learning tasks consist of similar instances, instead of calculating the gradient from all of the instances, the gradient can be estimated from a subset of them. But, gradient estimation from the subset needs

to be unbiased. This property can be obtained by selecting subsets randomly. This random selection part makes a gradient decent variation stochastic. In the stochastic gradient descent algorithm, all the instances are shuffled at the beginning of each epoch. Then each iteration is made by selecting a subset of instances called mini-batch and calculating the gradient estimation from that batch. When the gradients of all instances are calculated during the optimization, $\frac{N}{Batch\ Size}$ weight updates are done instead of just one. This saves a large amount of computational cost during the training. However, using SGD introduces batch size as a hyperparameter which requires tuning. There are different variations of SGD to enhance the optimization process. Root Means Square Propagation (RMSProp) is one of them [35]. It adjusts the learning rate for each parameter separately. Another one is Adaptive Moment Optimization (ADAM) which combines the momentum idea with learning rate adjustments in RMSProp [36].

The training of the neural networks gets more complex as the network gets deeper. Since, the updates on the weights are done simultaneously, the effect of the update is dependent on the changes in the other layers. Especially in deep architectures, this dependence creates unexpected results. In order to address this issue, batch normalization is proposed as an adaptive reparametrization of the network during training [37]. By reparametrization, it coordinates the values across many layers. Basically, batch normalization standardizes the input using the standard deviation and mean of the mini-batch to have a stabilization effect [38].

Lastly, the dropout method is also used to increase the model performance. It is used as a regularization method to control the model complexity and to deal with overfitting. The idea of the dropout is randomly removing nodes from the network during the training. This allows the construction of different thinned architectures and sampling from them. Therefore, it behaves like an ensemble of multiple thinned models to regularize the model and improve its performance [39].

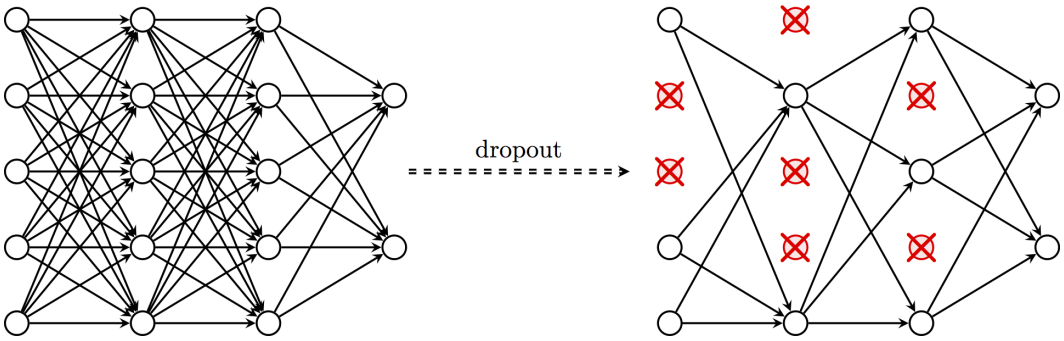

Figure 3.3. Applying dropout to a standard neural network [40].

3.2.1. Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron is a neural network layout where every neuron is connected to all neurons in the succeeding layer. This layout is also called the standard neural network architecture. The number of layers and the hidden size of the layers are can be arbitrary integers. Figure 3.4 demonstrates an example of MLP structure.

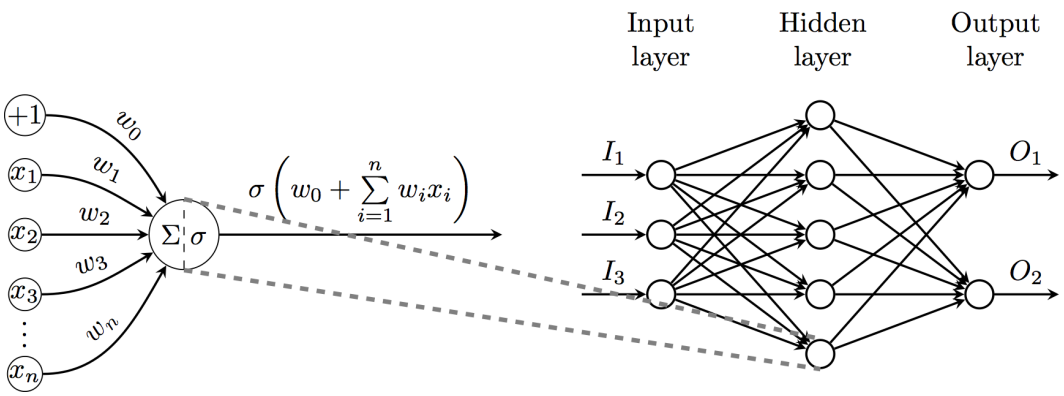

Figure 3.4. Demonstration of an example MLP [40].

3.2.2. Convolutional Neural Network (CNN)

Convolutional Neural Networks are a special kind of neural network to deal with the data that has a grid-like topology (i.e. time series data, image data). It employs an

operation called convolution (even though many libraries implement a similar function called cross-correlation function) to introduce parameter sharing and sparse interactions. During the convolution operation, the input is multiplied with a weight matrix called kernel to calculate the output as shown in Figure 3.5. Since the kernel is applied to all inputs by sliding it, the parameters in the kernel interact with multiple inputs to construct the output. In other words, unlike MLP, this operation shares the parameters along many input-output relations. Also, this implies sparse interactions since the contribution of an input to a single output is only made if the kernel size is large enough. The kernel size, stride amount of the kernel, and padding of the input before the operation are hyperparameters of the convolution operation.

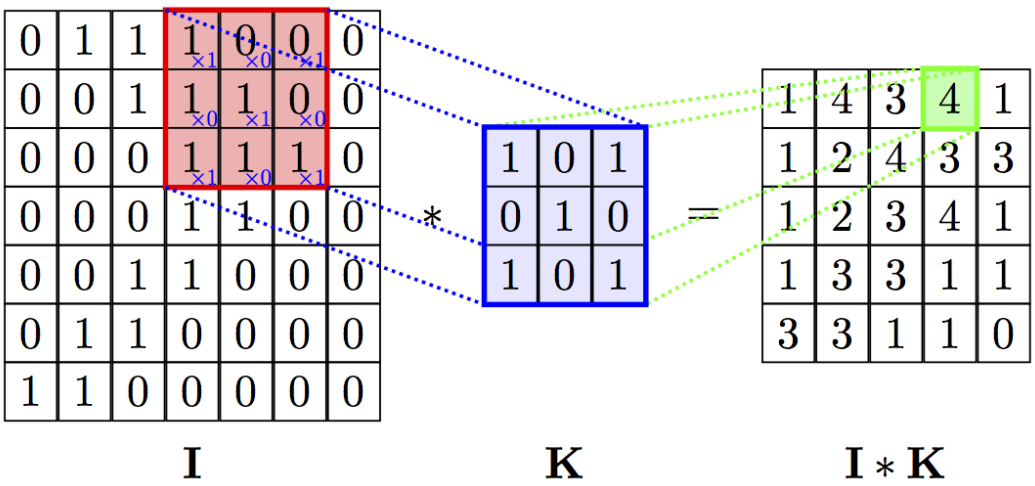

Figure 3.5. Demonstration of a convolution operation in 2D [40].

Another concept introduced by CNNs is pooling. Pooling is made by replacing the output with the summary statistics of itself and its neighborhood. This operation makes the transformation invariant to small changes in the neighborhood. In pooling, the size of the neighborhood is treated as a hyperparameter.

3.2.3. Skip Connections

As the name suggests, skip connections are the connection from the previous layers to deeper layers. These connections are used to flow the information to deeper parts of the network to obtain performance improvements. In ResNet [41], the skip connections are called residual connections and they are used in addition form. These connections in ResNet architecture allow the model to deal with the degradation problem by smoothing out the loss surface. In DenseNet architecture [42], the skip connections are named dense connections and they are used for feature reusability. The aim of these dense connections is to make use of the previously learned features in deeper layers. Skip connections are also used in biomedical image segmentation. The Popular U-Net [43] model employs the skip connections between its encoder and decoder to deal with medical images. In this model, the connections are arranged in a U-Shaped manner.

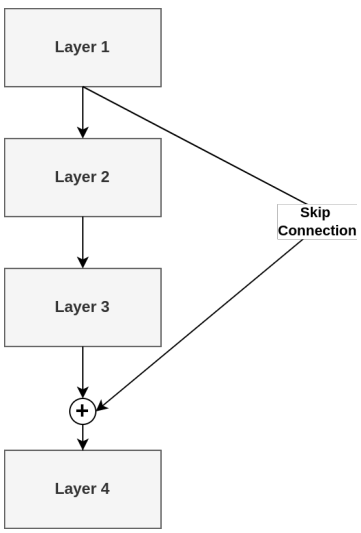

Figure 3.6. Skip connection example.

3.3. Penalized Regression Approaches

The classic regression models suffer from having large multivariate predictors when the number of samples is not sufficient enough. In order to solve this problem, penalized regression approaches are developed. In penalized regression models, a penalization term behaves as regularization and forces the model to emphasize the important predictors. Even though the aim is the same, the effect of the penalization differs depending on the penalization method. There are three different popular penalization methods. First one, Lasso regression uses the $L1$ norm of the coefficients as the penalization term. This term forces the weights of the unnecessary predictors towards zero. In other terms, it makes a subsample selection to predictors. Another one is Ridge regression. This method employs the $L2$ norm to regularize the regression coefficients. Using the square of the coefficients penalizes the large coefficients more and ensures a coefficient gets large values if it is really necessary. Lastly, the ElasticNet [44] uses the linear combination of $L1$ and $L2$ penalty terms to obtain the benefits of Lasso and Ridge regressions. All of these methods introduce one or two hyperparameters to determine the strength of the penalty. The penalty hyperparameters are usually tuned using cross-validation methods. The loss function of ElasticNet is expressed as

$$L = \frac{1}{2}\sum_{i=1}^{N}(y-\hat{y})^2 + \lambda_1 \sum_{j=1}^{M}|w_j| + \lambda_2 \sum_{j=1}^{M} w_j^2, \tag{3.4}$$

where for $\lambda_1 = 0$ the function becomes the Ridge objective and for $\lambda_2 = 0$ it expresses the Lasso objective.

There are many implementations of penalized regression approaches. GLMNet [45] package of R is prominent among them. The package is designed for fitting generalized linear models and it also fits penalized regression models using penalized maximum likelihood. It works with many alternative distribution families. Its parallelized and fast optimization algorithms exploit sparsity to further increase the training speed. Additionally, the package comes with cross-validation and plotting utilities.

3.4. Performance Metrics

There are two different performance metrics used in this thesis. The first one, mean squared error (MSE) is used for training and evaluating the postprocessing model. The other one, weighted mean absolute percentage error (WMAPE) is used for evaluating the wind power forecasts.

3.4.1. Mean Squared Error (MSE)

Mean squared error is a metric to measure the mean deviation from the real values. Its square term allows punishing the large deviations more. Since in weather forecasting case avoiding large deviations are critical, MSE is preferred for both training and performance evaluation tasks. The MSE metric is expressed as

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

3.4.2. Weighted Mean Absolute Percentage Error (WMAPE)

In wind forecasting task, rather than the performance of a single observation the total performance within the forecast period is important due to the unstable nature of the wind power generation. Therefore, instead of using mean absolute percentage error, its weighted form is preferred. In WMAPE, the weights are set as $\frac{|y_i|}{\sum_{i=1}^{N}|y_i|}$ which implies the proportion of generation over a total generation in the period. The WMAPE metric for a period is expressed as

$$WMAPE = \frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{\sum_{i=1}^{N}|y_i|}. \tag{3.6}$$

4. METHODOLOGY

Numerical Weather Prediction (NWP) postprocessing is a method to refine the weather forecast resulting from physical models. The statistical postprocessing models try to correct the errors caused by initial conditions and physical simplifications. In order to perform this task, an NWP model together with its historical forecasts, and the ground truth of these forecasts are required. Then a statistical model is constructed to map the forecasts to the ground truth. In this process, the mapping is optimized using a loss function. However, different weather variables have different scales. For example temperature in Kelvin can be between 270 and 315 for a location and relative humidity is between 0 and 1. Therefore, in order to have an equal contribution to the loss function, the weather data is needed to be scaled. Also, the weather data is needed to be shaped properly to benefit from its multidimensional structure before optimizing the model.

This chapter summarizes the methodology followed in this thesis to perform statistical NWP postprocessing without any distributional assumption. Firstly, the data sources are introduced in detail. This step involves the selection of weather variables and levels together with preprocessing such as scaling and tensor shaping. Secondly, it introduces proposed architectures and their parametrization.

4.1. Data

The data sources in this thesis can be divided into three categories. Two of them are considered as the main ones since they construct the input and output of the postprocessing task. These two sets are global weather sets and they are in the form of multidimensional tensors where the dimensions are time, weather variable (ie. temperature, humidity, etc.), atmospheric level (ie. 975 mb, etc.), latitude and longitude. A value in a global set is represented by α_{tijkl} where α is the NWP source, t is the time index, i is the variable index, j is the pressure level index, k is the latitude

index and l is the longitude index. Another data set is the wind power generation data used in the validation part of the main task. This set includes the hourly generation data in MWh for different wind farms. The data section presents the information about these data resources and how they are preprocessed to proper form.

4.1.1. Data Sources

4.1.1.1. Global Ensemble Forecasting System (GEFS). Global Ensemble Forecasting System forecasts are published by National Oceanic and Atmospheric Administration (NOAA) and are publicly available [46]. The three hourly forecasts are global and the resolution of the forecasts is $0.25° \times 0.25°$ in latitudes and longitudes. It consists of the forecasts of 6 different weather variables at 25 different pressure levels between 1 mb to 1000 mb. The name GEFS includes the word "ensemble" because NOAA releases a control model together with 30 perturbations of it to obtain more robust forecasts. The latest version of the GEFS is v12 and it has been released with daily reforecasts from 2000 to 2020 with 5 or 11 perturbations. Having the reforecast allows having the same simplifications and biases throughout the whole period. This property of GEFS makes it a perfect candidate as an input to a postprocessing task. Also, NOAA continues to publish the forecasts from GEFS which creates an opportunity to use the proposed postprocessing method in real-time. GEFS data is the input to the postprocessing models and it is represented by variable X with values x_{ijkl}.

4.1.1.2. ECMWF Reanalysis 5th Generation. Unlike GEFS, ECMWF Reanalysis 5th Generation (ERA5) is not a forecast model but a reanalysis of the forecasts after collecting actual measurements from alternative sources all around the globe [47]. Therefore, it creates a consistent global weather data set of the past and many studies treat it as ground truth weather [21–24]. The reanalysis starts in 1958 and the archive is updated regularly to include recent dates. ERA5 is also $0.25° \times 0.25°$ resolution and it includes hourly pressure level values. In the hourly pressure levels set there are 16 different weather variables and 39 different pressure levels between 1 mb and 1000 mb. ERA5 data is the output of postprocessing models and it is represented by Y with values y_{ijkl}.

4.1.1.3. Wind Power Generation. Wind Power Generation data is public and obtained from the transparency platform of Energy Exchange Istanbul [48]. About 355 wind farms with approximately 11 GWh installed capacity are included in the data [49]. For every farm, the set has the hourly generation data in MWh since their operation start date. The generation data is used as a target during the predictive performance measures of the postprocessed weather data. It is denoted as P with values p_{tf} where t is the time index and f is the farm index.

4.1.2. Data Preprocessing

Data preprocessing is applied to weather data sets in four steps. Firstly, a subset of weather variables and atmospheric pressure levels is extracted from the data. Later, a local region is selected, then the values are scaled between 0 and 1 using min-max scaling. Finally, ERA5 tensors are shaped into three hourly groups to match with GEFS inputs in the time dimension.

4.1.2.1. Variable-Level Selection. The selection of variables and levels is made because of two reasons. Firstly, ERA5 has 16 different variables at 39 different pressure levels, in a total of 624 values at each location at each hour. Similarly, GEFS has 6 different variables at 25 alternative pressure levels, in a total of 150 values per location at every third hour. Thus, the size of the data introduces extra computational complexity without selection. Secondly, it is aimed to focus on more common variable level combinations that have larger operational use cases in wind power forecasting tasks. Therefore as variables; temperature (tmp), u component of wind (u), v component of wind (v), vertical component of wind (w) and humidity (q) or relative humidity (r) are selected as the variables. For the pressure levels, the values between 1000 mb and 800 mb are selected since they are closer to the surface where more operational needs occur. Using these selections, GEFS is reduced to five variables and seven pressure levels, and ERA5 is reduced to the same variables but nine pressure levels since it has more pressure levels in the selected range.

4.1.2.2. Region Selection. Region selection is a necessary step to make the postprocessing task local. Wind farm intensity of the regions is considered as selection criteria to validate postprocessing performance in multiple wind power forecasting tasks. As a result, the Aegean Region of Turkey which has about 37% of the wind generation capacity of the country [50] is selected as the local region. The region is between 36.5 and 40.5 in latitudes and 25 and 29.5 in longitudes and it creates a 17×19 grid in 0.25° resolution. Figure 4.1 shows the boundary box of the selected area in western Turkey.

Table 4.1. Selected variables, levels and region.

Dimension	**ERA5**	**GEFS**
Variable	tmp, u, v, w, r	tmp, u, v, w, q
Level	1000, 975, 950, 925, 900, 875, 850, 825, 800 mb	1000, 975, 950, 925, 900, 850, 800 mb
Latitude	[36.5, 40.5]	[36.5, 40.5]
Longitude	[25, 29.5]	[25, 29.5]

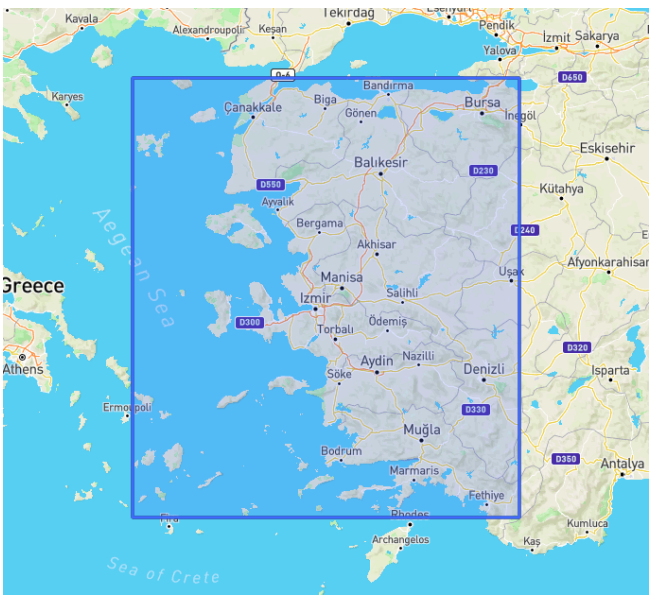

Figure 4.1. Selected region.

4.1.2.3. Scaling. The weather data sets are scaled for two reasons. Firstly, the scale of the variables at alternative pressure levels and location combinations is different. This difference creates an unfair contribution to the loss function and the values in larger domains mainly decrease or increase the total loss during the optimization. Scaling makes the contribution to loss function equal for every point in the data. Secondly, the model comparison is distorted with the evaluation metrics that do not scale invariant. In that case, a model that processes variables with a larger range slightly better would be preferred to the model that outperforms the first model in variables with a smaller range.

As the scaling method min-max scaling is selected. Min-max scaling transforms the values in every feature between 0 and 1 by subtracting the minimum observation value and dividing the result by the observed range. The equation of the scaling operation can be expressed as

$$\hat{\alpha}_{tijkl} = \frac{\alpha_{tijkl} - min(\alpha_{ijkl})}{max(\alpha_{ijkl}) - min(\alpha_{ijkl})}. \tag{4.1}$$

Additionally, in order to preserve the comparability of GEFS and ERA5 values the common $ijkl$ combinations in both of the data sets are scaled together. This allowed comparing the improvement of the postprocessed values with respect to GEFS in the scaled domain.

4.1.2.4. Tensor Shaping. GEFS is published three hourly however ERA5 is released hourly as discussed in Section 4.1.1. In order to train a model, these two sets need to be matched in the time dimension. Subsampling the hours that are not present in GEFS from ERA5 is a straightforward solution. Another way is to group ERA5 into three hourly values and try to predict three hours at once from a single GEFS hour. Even though subsampling is the easiest way, it causes the loss of one-third of ERA5 data. Considering the data requirements of the deep learning architectures, subsampling is not preferred. Also, predicting three hours of ERA5 adds an extrapolation capability to the models. Considering these benefits, the three hourly prediction scenario is favored. Thus, the ERA5 tensors are shaped into three hourly groups. Figure 4.2 shows how three hourly GEFS instances are matched with hourly ERA5 instances.

At the end, GEFS tensors are fed into the model in (*3 hourly time* × *variable* × *pressure level* × *latitude* × *longitude*) dimensions with sizes (*batch size* × 5 × 7 × 17 × 19) and ERA5 tensors are fed as target in(*3 hourly time* × *[t, t+1, t+2]* × *variable* × *pressure level*×*latitude*×*longitude*) dimensions with sizes (*batch size*×3×5×9×17×19).

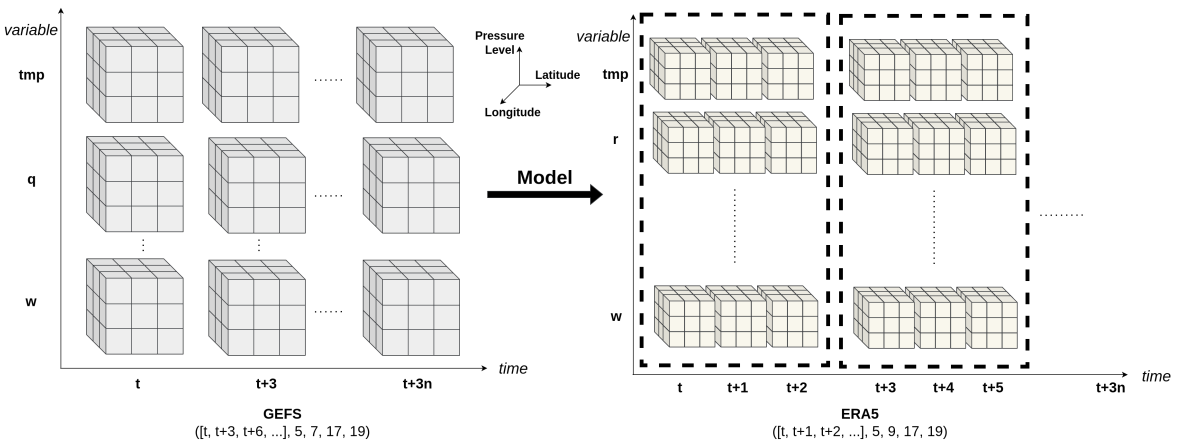

Figure 4.2. Input and output schema.

4.2. Proposed Architectures

This section briefly summarizes three alternative architecture proposals for the postprocessing model. The first one, Multi-Layer Perceptron (MLP) is chosen as a base model. The second architecture, Fully Convolutional ANN introduces convolution operation to share parameters along the spatial dimension. The last one, U-Shaped ANN adds residual connections to Fully Convolutional ANN in a U-shaped manner inspired by U-Net [43].

4.2.1. Multi-Layer Perceptron (MLP)

MLP architecture consists of multiple fully connected blocks that share a constant hidden size. The blocks consist of a fully connected layer, batch normalization, ReLU activation function, and a drop out layer. The drop out ratio and batch normalization usage are set as hyperparameters. Figure 4.3 shows the proposed MLP architecture.

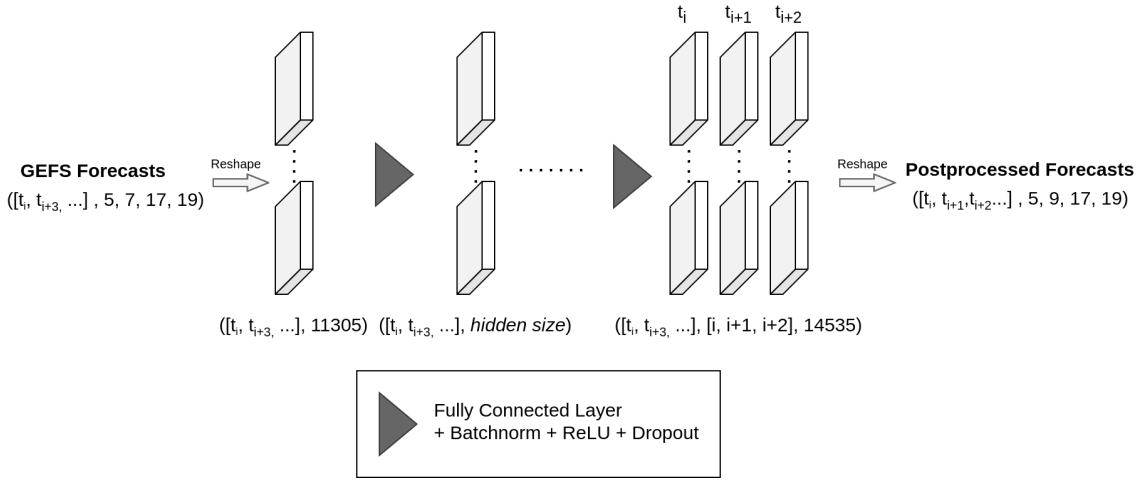

Figure 4.3. MLP architecture.

In order to pass the GEFS input to this architecture, the data is flattened along all dimensions and passed to the network. Similarly, the output of the network is in a single dimension. Therefore, it is needed to reshape the output of this model to ERA5 output size. Since, the size of GEFS tensor is about 11.3K ($5\times7\times17\times19$) the first layer needs to have $11.3K \times$ *hidden size* parameters. On the output side ERA5 tensor size is about 43.6K ($3 \times 5 \times 9 \times 17 \times 19$) and the last layer needs to have $43.6K \times$ *hidden size* parameters. Having these many parameters restricts the hidden size and the depth options with limited resources. In this architecture, the hidden size value is considered as a hyperparameter to control the model complexity. Additionally, the number of layers is not predetermined but tuned for the same reason.

4.2.2. Fully Convolutional Artificial Neural Network

In order to address the problems that come with the high number of parameters in MLP, the Fully Convolutional ANN architecture in Figure 4.4 is proposed. Using convolutional blocks instead of fully connected ones allows sharing of parameters along the spatial dimension. Besides reducing the number of parameters this architecture utilizes spatial relations by having kernels that extract features from nearby locations. Like fully connected blocks, convolutional blocks consist of convolution layer, batch

normalization, ReLU activation, pooling and drop out where drop out value, batch normalization usage, and whether to use average or max pooling are hyperparameters to tune.

The convolution operation is made in 2D and along latitude and longitude dimensions. The other dimensions are flattened into channels. Similar to hidden size in MLP, the number of output channels in all blocks is held constant. Additionally, the number of output channels and number of blocks are considered as hyperparameters to control the model complexity.

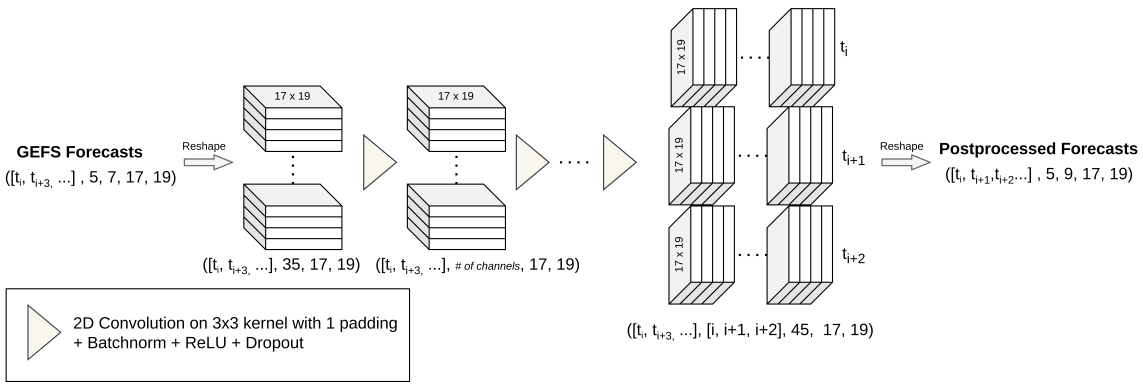

Figure 4.4. Fully convolutional ANN architecture.

4.2.3. U-Shaped Artificial Neural Network

The input of a postprocessing task is expected to be similar to the output values however there is a place for improvement. Basically, the postprocessing model needs to adjust all the values in the input by correcting the systematic errors and biases. Thus the input values create a good starting point for the output values. However, carrying all the information to the deeper layers is a challenge. When the architectures get deeper the degradation problem starts to occur and the information from previous layers cannot be passed to the deeper layers properly [41]. In order to tackle this problem adding skip connections to the fully convolutional ANN is considered [41] [42]. By connecting previous outputs to deeper layers, skip connections helps to preserve the information throughout the whole network. One of the previous studies that utilize

this idea is U-Net [43]. U-Net is an extension of fully convolutional networks and it is used for medical image segmentation. It employs a U-shaped network that introduces skip connections between the sides of the U. Considering the success of these skip connections in the U-net, the skip connections in this architecture are constructed in a U-shaped structure as shown in Figure 4.5. Unlike previous architectures, the number of blocks is not used to control the model complexity and is held constant at eight. For the number of channels, again a single value is used and its value is tuned.

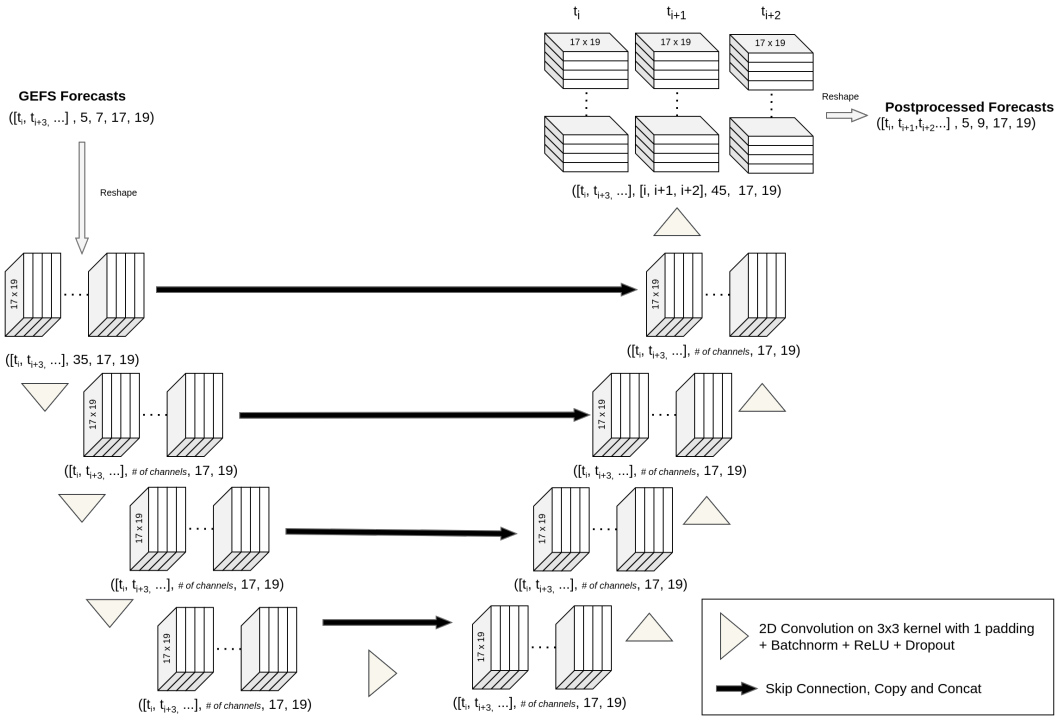

Figure 4.5. U-Shaped ANN architecture.

Table 4.2. Summary of proposed architectures.

	Block Type	**Parameters**
MLP	Fully Connected	Dropout Ratio Batch Normalization Usage # of Blocks Hidden Size
Fully Conv. ANN	Convolution	Dropout Ratio Batch Normalization Usage Pooling Type # of Blocks # of Output Channels
U-Shaped ANN	Convolution	Dropout Ratio Batch Normalization Usage Pooling Type # of Output Channels

5. EXPERIMENTS

5.1. Modeling Pipeline

This section explains the postprocessing modeling pipeline for a single experiment in detail. A single experiment is considered as the combination of training, validation, and testing of a postprocessing model in a specified parameter setting. The codes for the experiments can be found at the github repository [51].

Since the obtaining and preprocessing data is independent of the model parameters, it is done for a single time and the prepossessed versions are stored. GEFS data is downloaded in grib2 [52] format. Then, its levels and variables are filtered as mentioned in Subsection 4.1.2.1. Also, the region is selected according to Subsection 4.1.2.2. Similarly, ERA5 is downloaded in grib2 format, its variables and levels are filtered and the same region is extracted. The resulting data from both data sets are saved in a daily manner to allow parallel data loading during the training. After that, the data is split into three periods before training. Between 2000-2015 are used for training, the year 2016 is picked for validation, and 2017-2019 are reserved for testing. Lastly, the min-max value for each location, variable, and level combination is calculated and saved to be used in scaling.

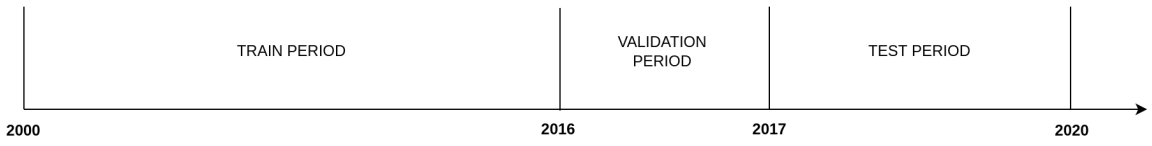

Figure 5.1. Train validation and test periods.

During the experiments, Python 3.9.1 is used as the programming language. The training is made by using the Pytorch Lightning [53] framework which is an extension of Pytorch [54]. For each experiment, the models are initialized with random weights and

moved to the GPU for faster training. Then, a data loader scaled and fed the data to models in batches. The batch size is considered as a hyperparameter for training. For each batch, the gradients are calculated with respect to the loss function via Pytorch's autograd functionality. As the loss function, mean square error is used because of its ability to penalize larger deviations more. After that, the weights in the model are updated using the Adam [36] optimizer. The amount of updates is determined by a hyperparameter called the learning rate. Also, in order to regularize the models, the weight decay parameter is employed to penalize the weights according to the $L2$ norm.

The weight updates are made in batches and using all batches for the update is called an epoch. At the end of each epoch, the performance of the validation period is calculated and reported. As the training stopping condition, a maximum of 30 epochs threshold is set. Also, an early stopping condition is introduced to terminate unpromising trials and to avoid overfitting. The early stopping is performed if the improvement in validation loss is less than 0.0001 at the last five epochs. Finally, when the training ends, the model weights are saved. Both scaled and unscaled performance on the test period is calculated and reported. Lastly, the model outputs of the test period are dumped for further analysis.

Table 5.1. Training parameters.

Training Parameters	**Value**
Batch Size	Hyperparameter
Learning Rate	Hyperparameter
Weight Decay	Hyperparameter
Max Epoch	30
Early Stop Patience	5 epochs
Early Stop Delta	0.0001

5.2. Hyperparameter Tuning

The performance of the deep learning models is highly sensitive to the hyperparameters used in the model development. Therefore, tuning these parameters is essential for obtaining high-performing models. The hyperparameters in proposed architectures can be divided into two categories. First, the model hyperparameters are mentioned in Section 4.2. They are used for controlling the model complexity and each proposed architecture has its own model hyperparameters. The other category is the training hyperparameters mentioned in Section 5.1. These parameters are used for the optimization of the weights. The training hyperparameters are not architecture-specific and tuned in every architecture. In order to tune the hyperparameters, a parameter space is required. Considering the training time of the proposed architectures, a preliminary study is made to shrink the search space. The final search space is presented in Table 5.2.

For each architecture, the hyperparameters are tuned for 48 hours using a parameter tuning framework called Optuna [55]. Even though Optuna has a Bayesian optimization schema, it is not preferred due to concerns about Bayesian hyperparameter optimization of deep learning models [38]. Instead, random search is employed during the optimization. While evaluating the trials, the test data set is not used and the optimization is made based on validation loss value. After tuning the hyperparameters, the architectures with the best validation score are chosen and postprocessed values for the test period are extracted.

5.3. Wind Power Forecasting

Wind power forecasting is an important problem for energy traders and power plant operators. While forecasting wind power, having well-forecasted wind variables is essential for high-accuracy models. This makes wind forecasting a useful task to show the effect of the proposed postprocessing model.

Table 5.2. Hyperparameter search space.

Parameter Source	Hyperparameter	Candidate Values
Training	Batch Size	8, 16, 32, 64, 128
	Learning Rate	0.001, 0.01, 0.1
	Weight Decay	0.0001, 0.001, 0.01
MLP	Dropout Ratio	0, 0.1, 0.2
	Batch Normalization Usage	True, False
	# of Blocks	2, 3, 4, 5, 6, 7, 8
	Hidden Size	32, 64, 128, 256, 512, 1024
Fully Conv. ANN	Dropout Ratio	0, 0.1, 0.2
	Batch Normalization Usage	True, False
	Pooling Type	Average, Max
	# of Blocks	2, 3, 4, 5, 6, 7, 8
	# of Output Channels	32, 64, 128, 256, 512, 1024
U-Shaped ANN	Dropout Ration	0, 0.1, 0.2
	Batch Normalization Usage	True, False
	Pooling Type	Average, Max
	# of Output Channels	32, 64, 128, 256, 512, 1024

The power forecasting is made for 19 farms located in the selected region. Figure 5.2 demonstrates the location of these farms. For every farm, $0.25° \times 0.25°$ boundary box is extracted from GEFS, model outputs, and ERA5 in the test period. As variables, u and v components of wind are selected. For pressure levels, 1000 mb, 975 mb, and 950 mb are included. From the u and v components of the wind, wind speed and wind direction are calculated. Then, all of these selected values are shaped into a tabular format where time is the row index and other dimensions are represented in columns. The square and cube of the wind speeds are extracted as features of the tabular data.

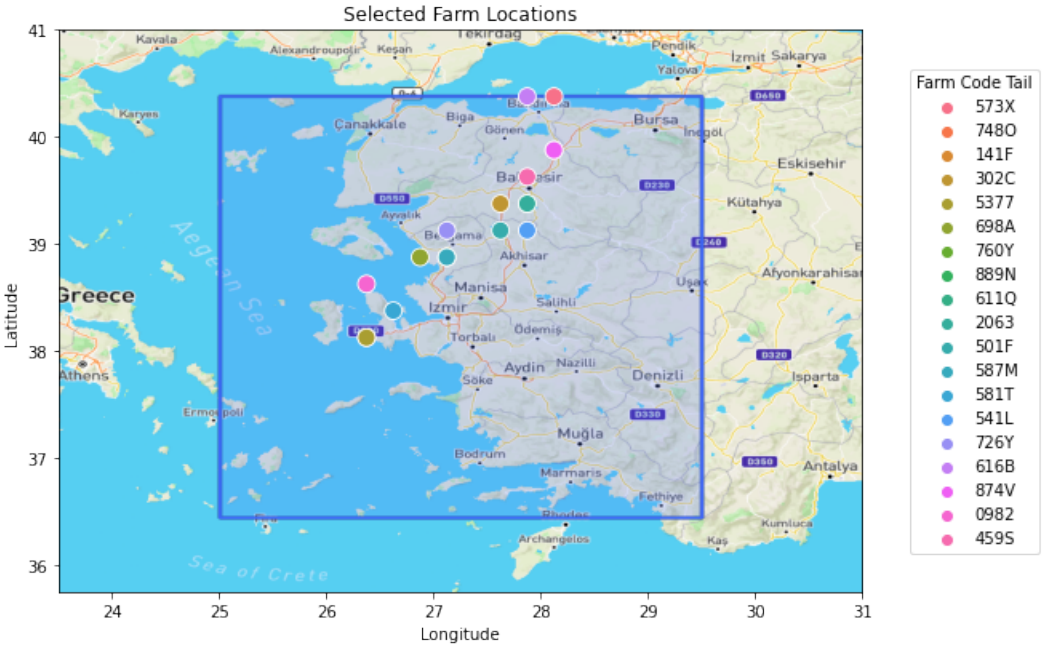

Figure 5.2. Location of the selected farms in the selected region.

The production values mentioned in the Section 4.1.1.3 are used as the target in this task. A stable production period is determined for every farm. If there is no stable period, the farm is eliminated from the tests since it would require additional preprocessing steps. In total, 19 wind power plants are selected for comparison.

For the regression task, a generalized linear model with lasso regularization is used from R's glmnet [45] package. The regularization parameter is determined by using ten-fold cross-validation in the training period. As the training period, 2017 and 2018 are used and 2019 is left out for the test. Since the weather has yearly seasonality, having one year of the test period is preferred to eliminate seasonal effects.

For all three weather sources and all wind farms, total and monthly WMAPE is calculated for 2019 values. Finally, all WMAPE values are reported for weather source comparison.

6. RESULTS

This chapter summarizes the results obtained from the experiments. Firstly hyperparameter tuning results are presented for each architecture. Secondly, the overlapping postprocessed values are compared with GEFS and the effect of postprocessing is measured. This comparison is made for each variable separately. Also, pressure level-based, location-based, and monthly comparisons are presented. After that, the performance of alternative architectures is compared and the best architecture is selected for further analysis. Then the outputs of the best postprocessing model is compared for non-overlapping hours to show the performance of the extrapolation capability of the model. Finally, the postprocessed wind variables from the best model are used to predict wind power generation in multiple wind farms and the predictive performance is reported.

6.1. Multi-Layer Perceptron

This section presents the results from the MLP architecture mentioned in the Subsection 4.2.1. Firstly, hyperparameter tuning is made according to the experiment setting mentioned in Section 5.2. The minimum MSE loss for the validation period is observed as 0.00265 in the scaled domain. The best parameters in terms of validation loss are presented in Table 6.1. After, the model trained with the best hyperparameters is evaluated in the test period. During the evaluation, the values are converted back to the real scale. Therefore, the evaluation is made for each variable separately.

Table 6.1. MLP best hyperparameters.

Hyperparameter	**Value**
Batch Size	32
Learning Rate	0.001
Weight Decay	0
# of Blocks	4
Hidden Size	1024
Dropout Ratio	0.1
Batchnorm Usage	True

6.1.1. Variable-Based Performance

MLP architecture obtained better wind speed variables than the initial values as shown in Figure 6.1. However, *tmp* estimates are worse. This situation can be explained by the model's hidden size. The MLP architecture has an information flow bottleneck in the hidden layers. When the hidden size is not large enough all necessary information cannot flow through the output. As mentioned in Section 4.2, increasing the size of the hidden dimension by one introduces 43.6K additional parameter. These extra parameters make training longer and harder. They also require additional computational resources to deal with the model size.

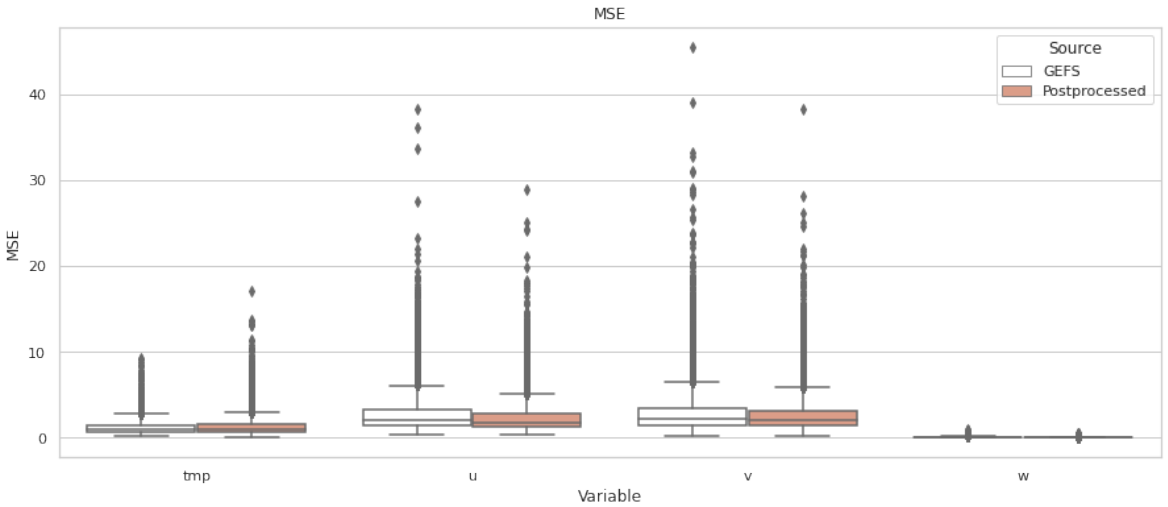

Figure 6.1. MSE performance of MLP model for each variable.

6.1.2. Variable-Pressure Level-Based Performance

The pressure level-based performances of variables *tmp*, *u*, *v*, and *w* are presented in Figures 6.2, 6.3, 6.4 and 6.5 respectively. The MSE values of the MLP Model tend to increase as the pressure level drops. This causes worse estimations of variables *tmp*, *u*, and *v* at lower pressure levels. However, the model achieves better performance for every variable at higher pressure levels. For *w*, this model achieves significant improvements in MSE values.

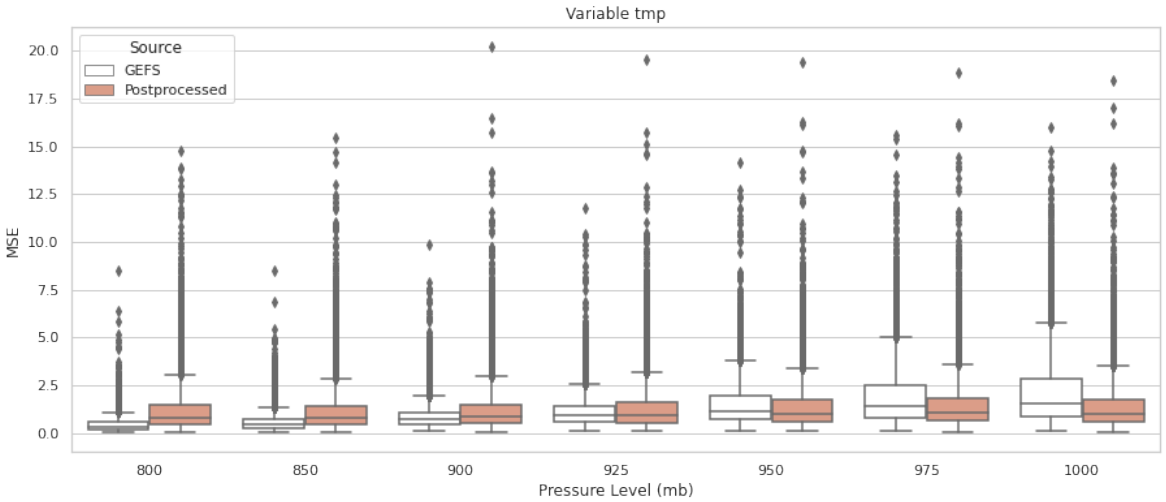

Figure 6.2. MSE performance of MLP model for variable *tmp* at each pressure level.

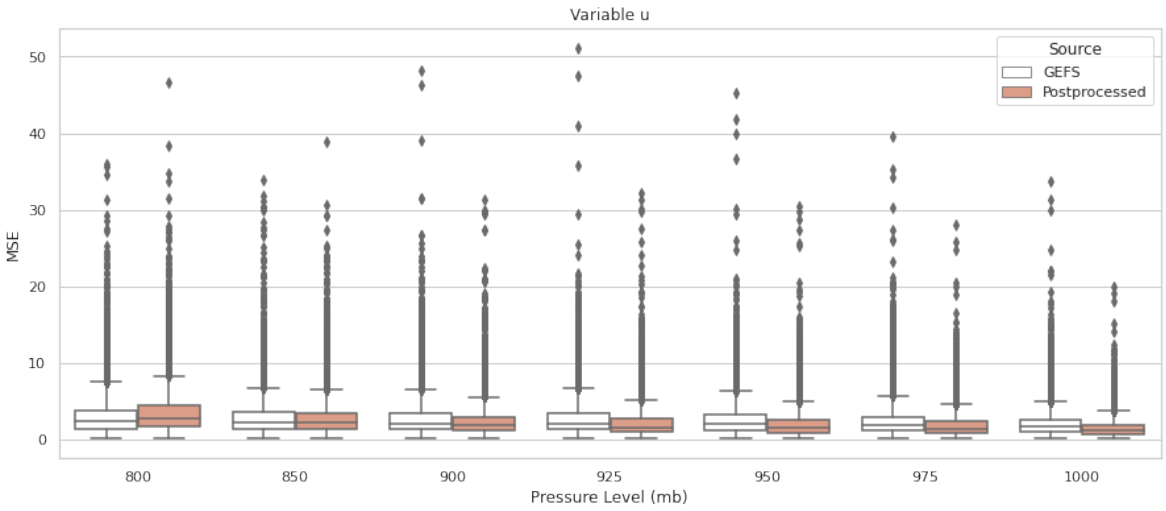

Figure 6.3. MSE performance of MLP model for variable u at each pressure level.

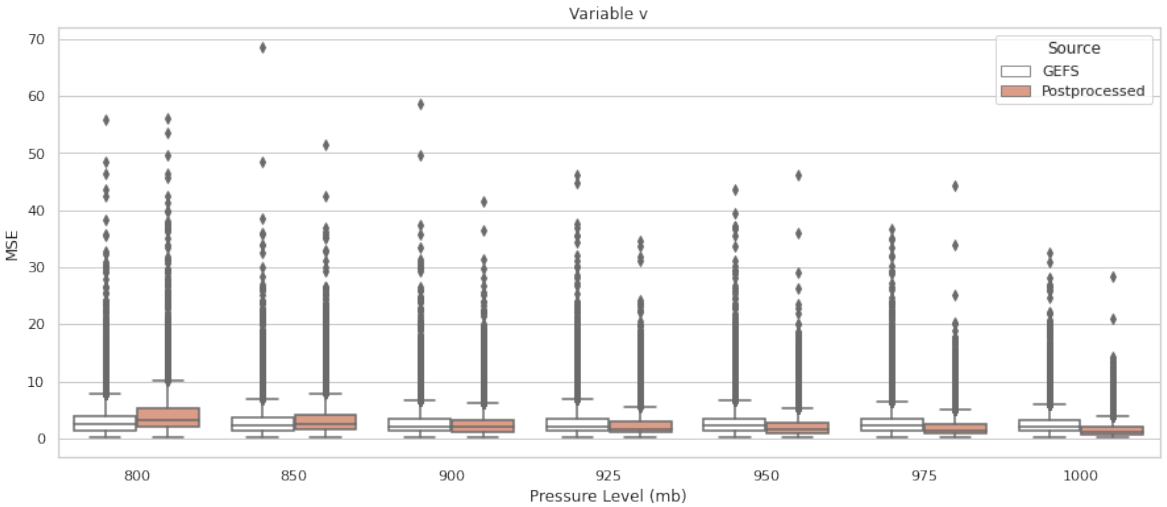

Figure 6.4. MSE performance of MLP model for variable v at each pressure level.

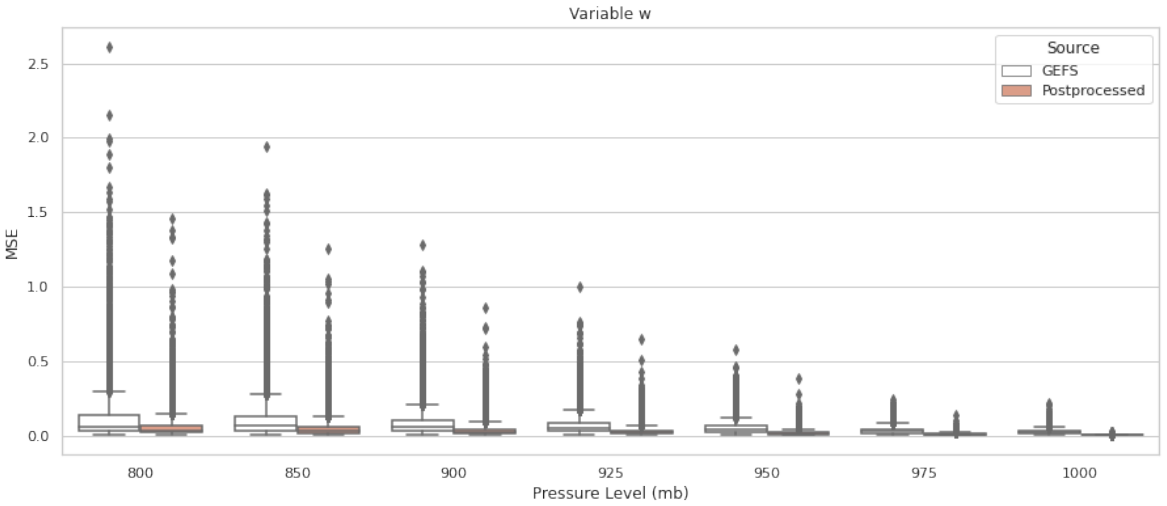

Figure 6.5. MSE performance of MLP model for variable w at each pressure level.

6.1.3. Variable-Location Based Performance

The location-based performances of variables *tmp*, u, v and w are presented in Figures 6.6, 6.7, 6.8 and 6.9 respectively. The layout of the locations in the figure has geographic correspondence with the region in Figure 4.1. Therefore, the left (west) sides of the heatmaps are equivalent to the coastal region, and the right (east) sides represent the inland region. The performance of GEFS in *tmp* degrades from coastal to inland regions. The MLP model smooths the error of this variable through the region. As a result, the errors in the coastal region degrade and the ones in the inland improve. The model also behaves similarly for variable v. For u the north and south borders are the locations with worse MSE values but the remaining area improves with the model. Lastly, the model reduces the errors of variable w almost in all locations.

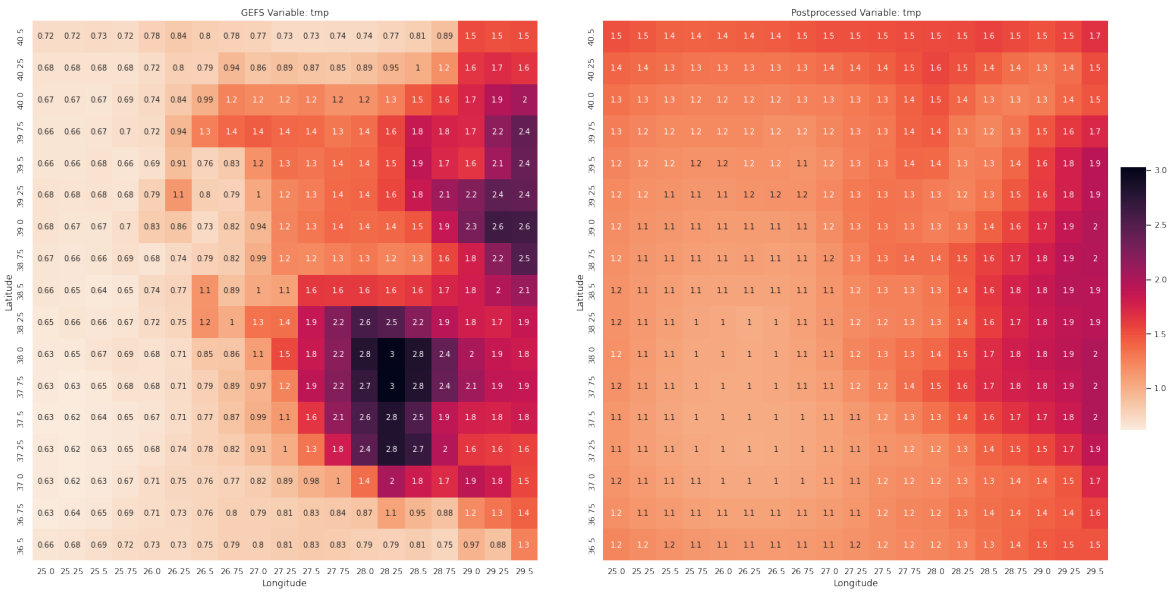

Figure 6.6. MSE performance of MLP model for variable *tmp* at each location.

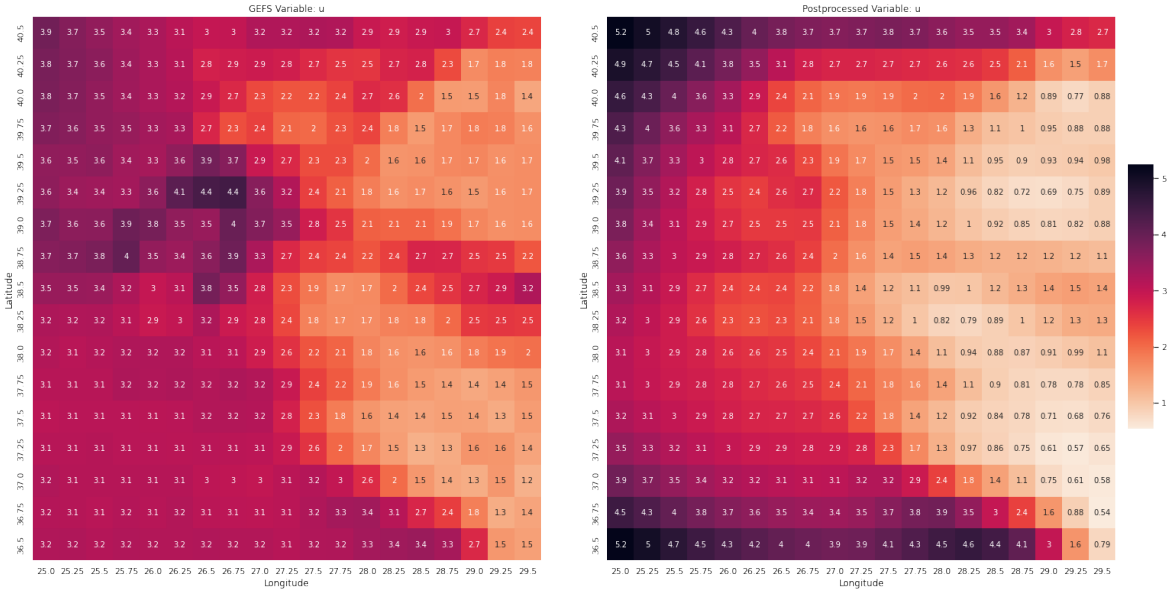

Figure 6.7. MSE performance of MLP model for variable u at each location.

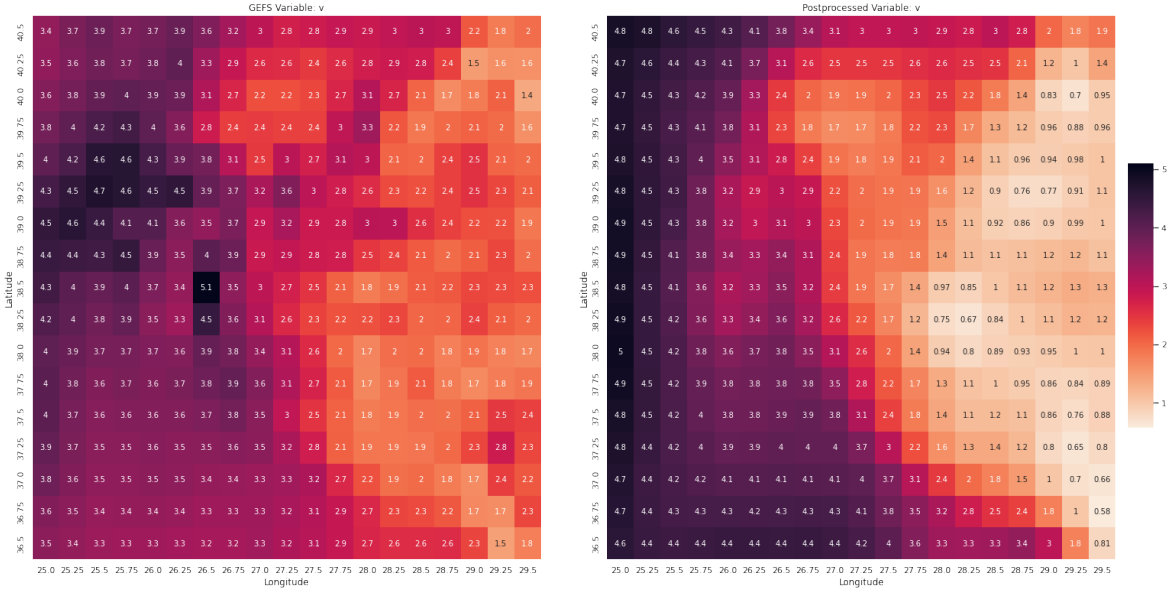

Figure 6.8. MSE performance of MLP model for variable v at each location.

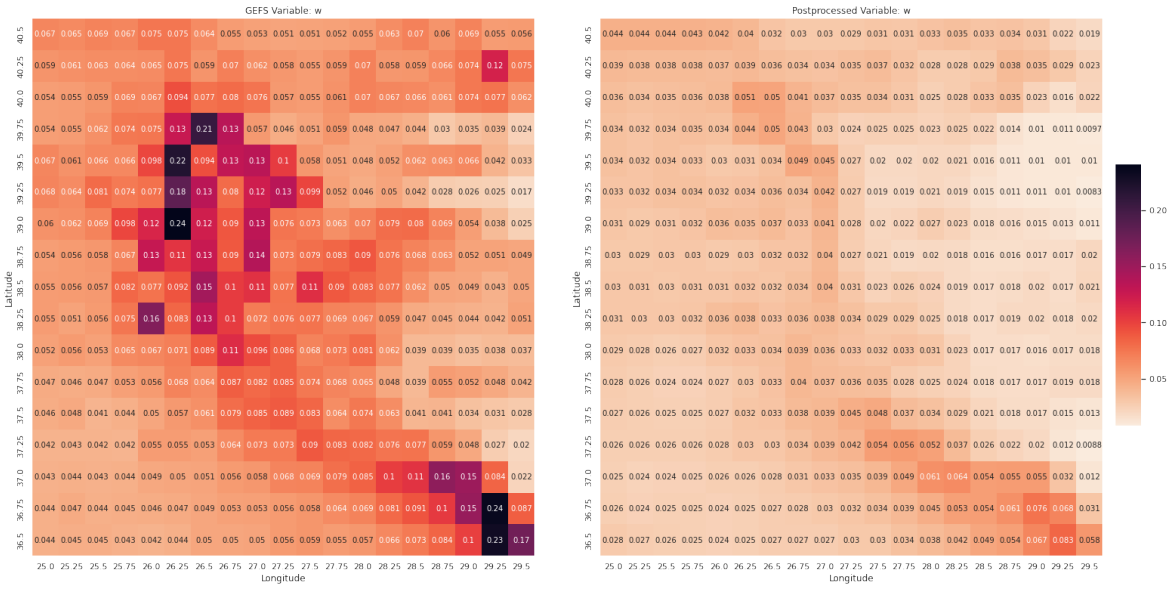

Figure 6.9. MSE performance of MLP model for variable w at each location.

6.1.4. Variable-Month Based Performance

The month-based performances of variables tmp, u, v, and w are presented in Figures 6.10, 6.11, 6.12 and 6.13 respectively. This seasonality analysis shows that the errors of GEFS are not uniform throughout the year. For each variable, the median error and the variance of it change depending on the month of the year. Figure 6.10 demonstrates that postprocessing tmp with the MLP model increases both variance and the median of the errors in almost all months. However, for u and v, the situation is the opposite. Even though the improvement is small, the MLP model achieves to deal with outlier models better. For variable w, apart from carrying the median to lower values, it successfully reduces the variance of the errors in most of the months.

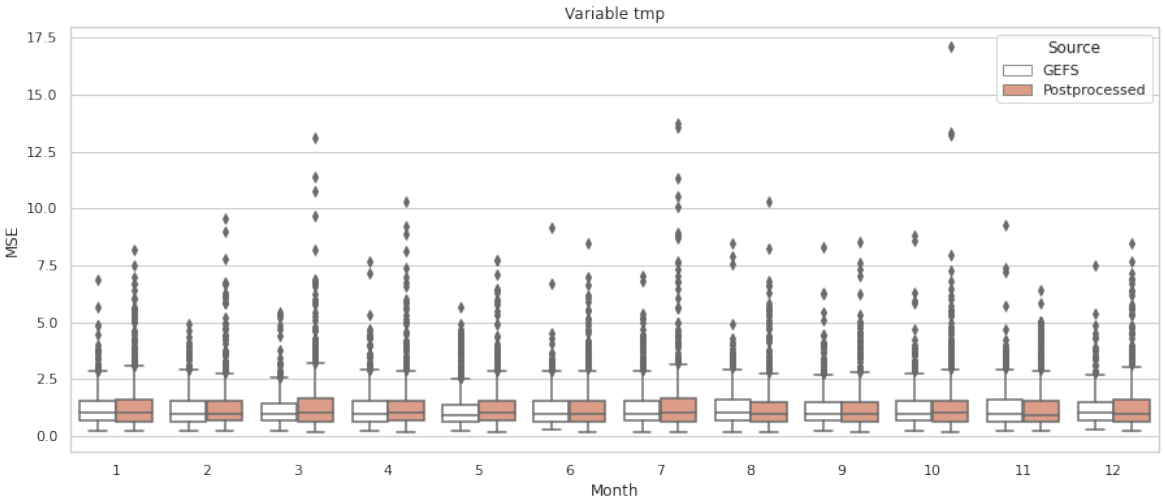

Figure 6.10. Month-based performance of variable *tmp* in MLP architecture.

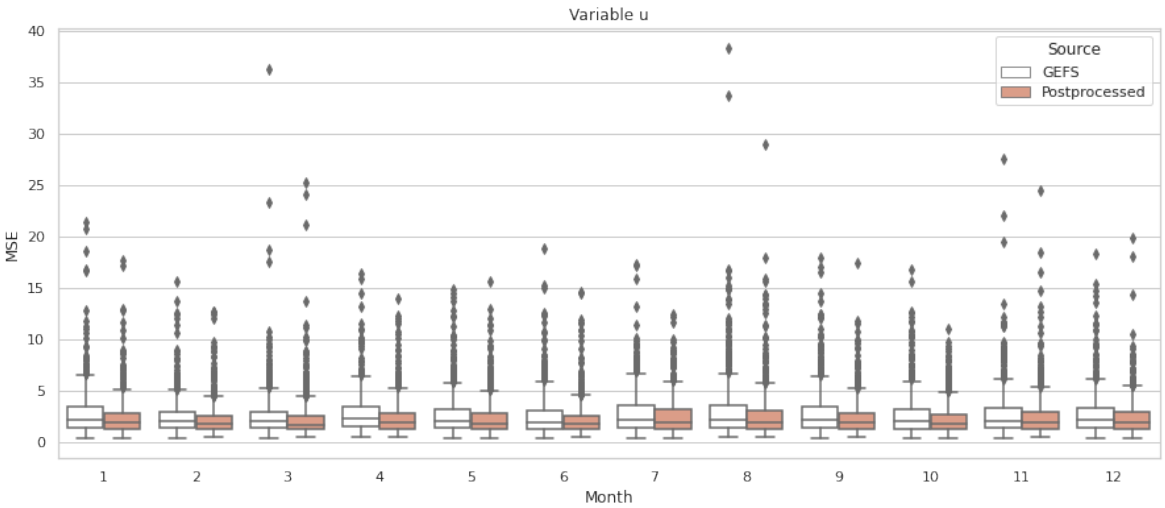

Figure 6.11. Month-based performance of variable u in MLP architecture.

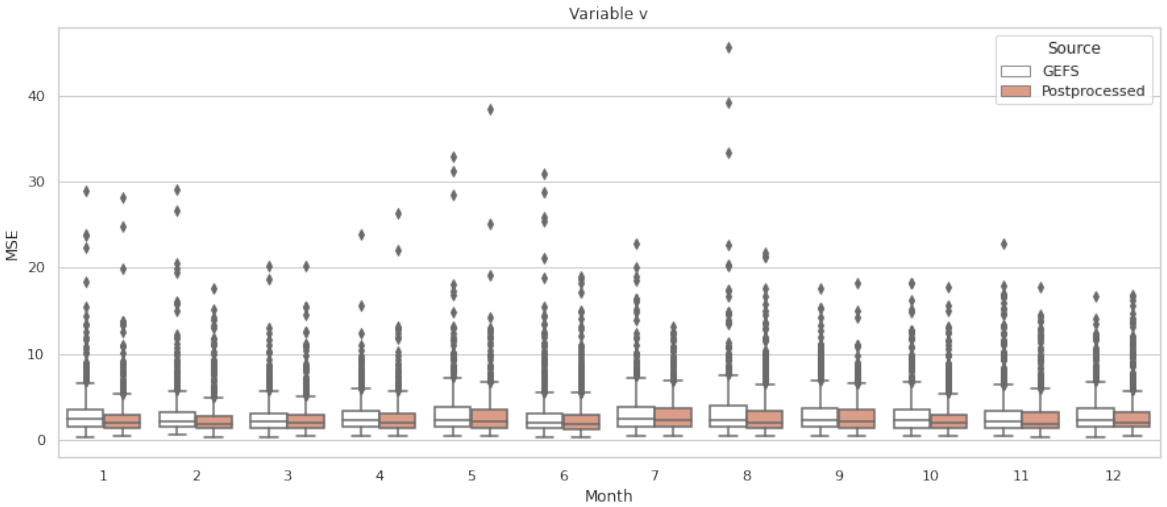

Figure 6.12. Month-based performance of variable v in MLP architecture.

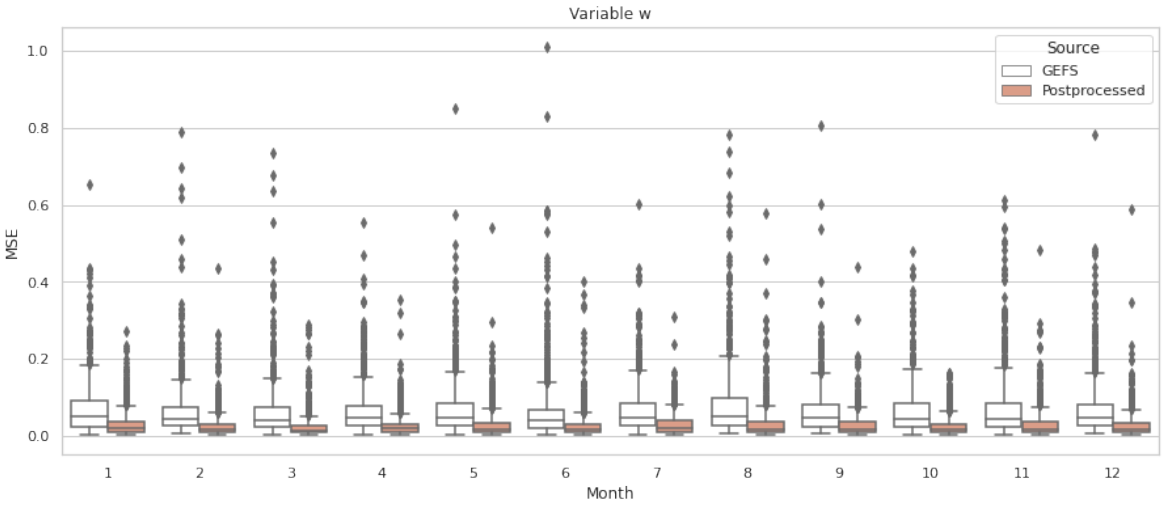

Figure 6.13. Month-based performance of variable w in MLP architecture.

6.2. Fully Convolutional Artificial Neural Network

This section presents the results from the Fully Convolutional architecture mentioned in the Section 4.2.2. The hyperparameter tuning of the model is made according to the experiment setting mentioned in Section 5.2. The minimum MSE loss for the validation period is observed as 0.00227 in the scaled domain. The best parameters in terms of validation loss are presented in Table 6.2. After the model trained with the best hyperparameters is evaluated in the test period. During the evaluation, the values are converted back to the real scale. Therefore, the evaluation is made for each variable separately.

Table 6.2. Fully convolutional architecture best hyperparameters.

Hyperparameter	**Value**
Batch Size	8
Learning Rate	0.001
Weight Decay	0
# Number of Blocks	4
# of Channels	512
Dropout Ratio	0
Batchnorm Usage	True
Pooling Function	Max Pooling

6.2.1. Variable-Based Performance

Figure 6.14 shows that Fully Convolutional architecture obtained better MSE values in all variables when compared with GEFS.

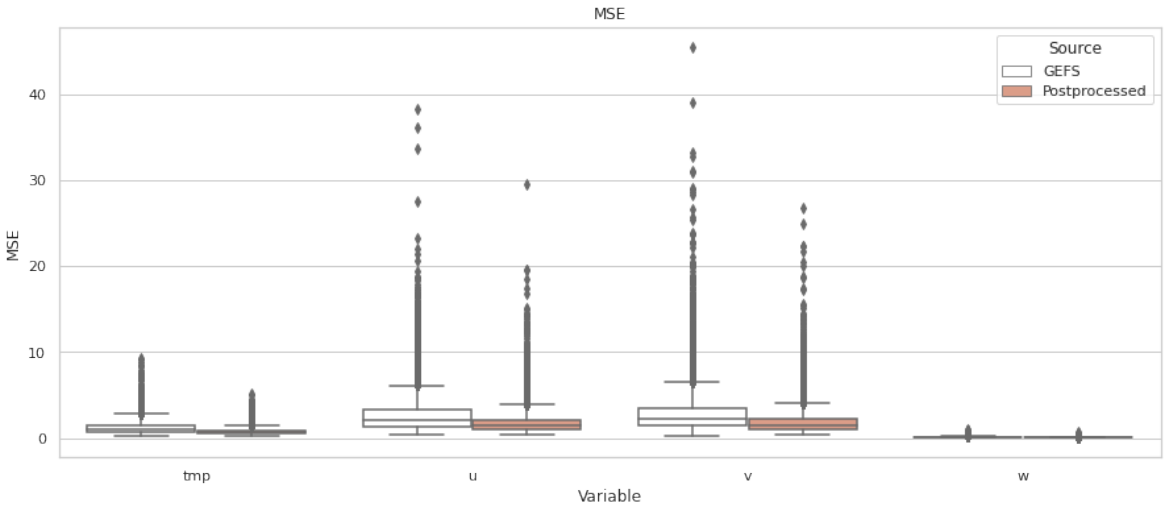

Figure 6.14. MSE performance of fully convolutional model for each variable.

6.2.2. Variable-Pressure Level-Based Performance

The pressure level-based performances of variables *tmp*, u, v, and w are presented in Figures 6.15, 6.16, 6.17 and 6.18 respectively. The model achieves better errors for all wind variables at each pressure level. For *tmp* the MSE values are better than GEFS for high pressures however, there is a small degradation in 800 and 850 mb levels.

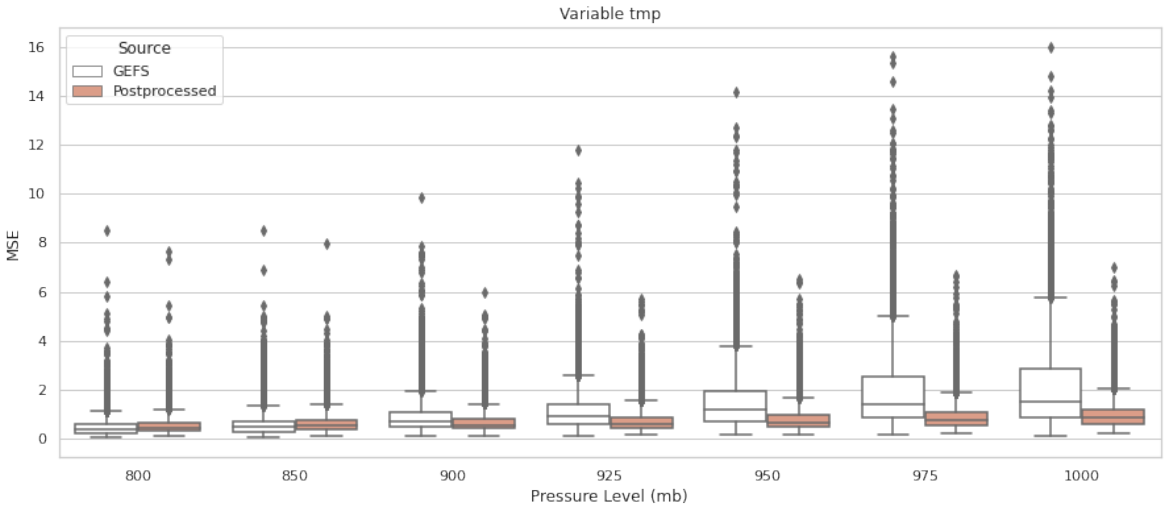

Figure 6.15. MSE performance of fully convolutional model for variable *tmp* at each pressure level.

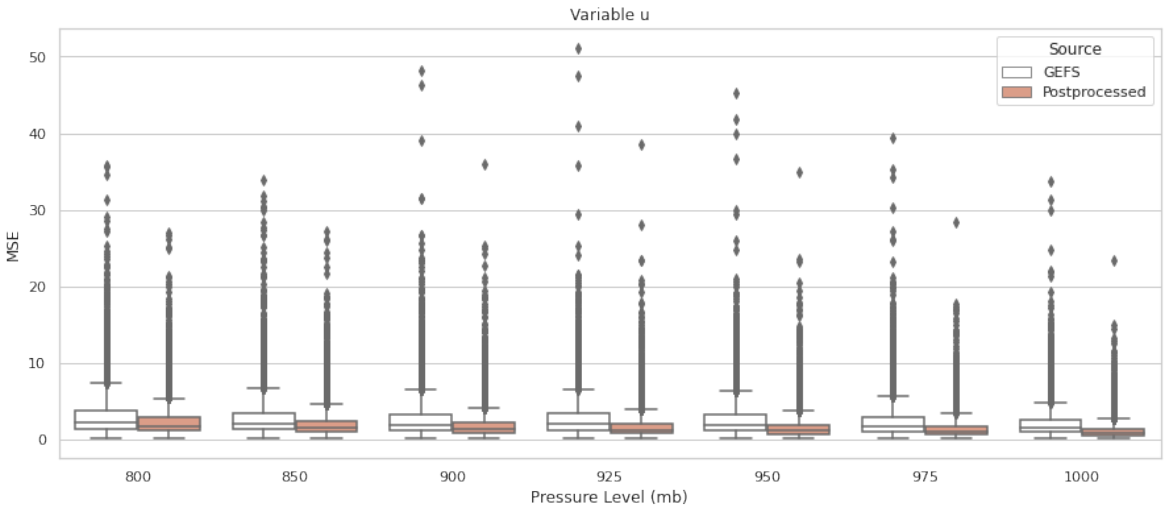

Figure 6.16. MSE performance of fully convolutional model for variable u at each pressure level.

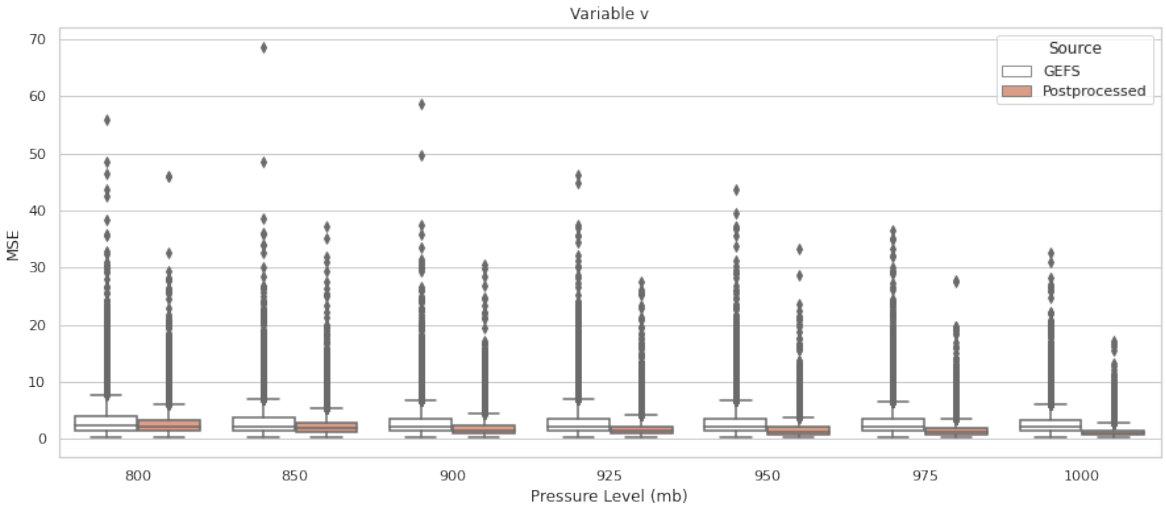

Figure 6.17. MSE performance of fully convolutional model for variable v at each pressure level.

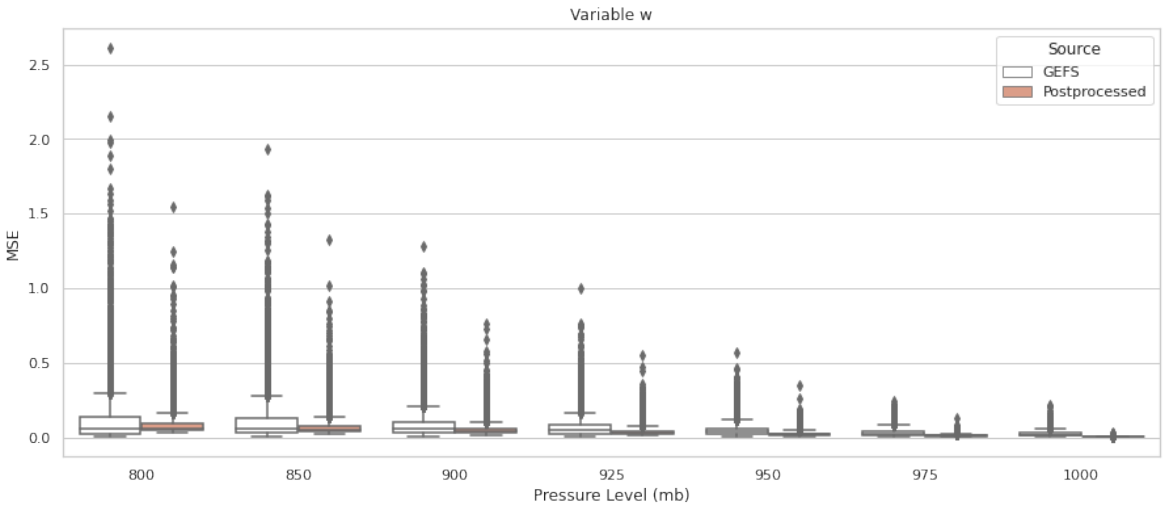

Figure 6.18. MSE performance of fully convolutional model for variable w at each pressure level.

6.2.3. Variable-Location Based Performance

The location-based performances of variables *tmp*, u, v and w are presented in Figures 6.19, 6.20, 6.21 and 6.22 respectively. The layout of the locations in the figure has geographic correspondence with the region in Figure 4.1. Therefore, the left (west) sides of the heat maps are equivalent to the coastal region, and the right (east) sides represent the inland region. For all variables, the MSE values of almost every location are reduced after postprocessing with the Fully Convolutional model. It fixes the inland region of variable *tmp* while improving the others. For u and v, all locations gain lighter colors, and for w high error region in the middle is almost fixed.

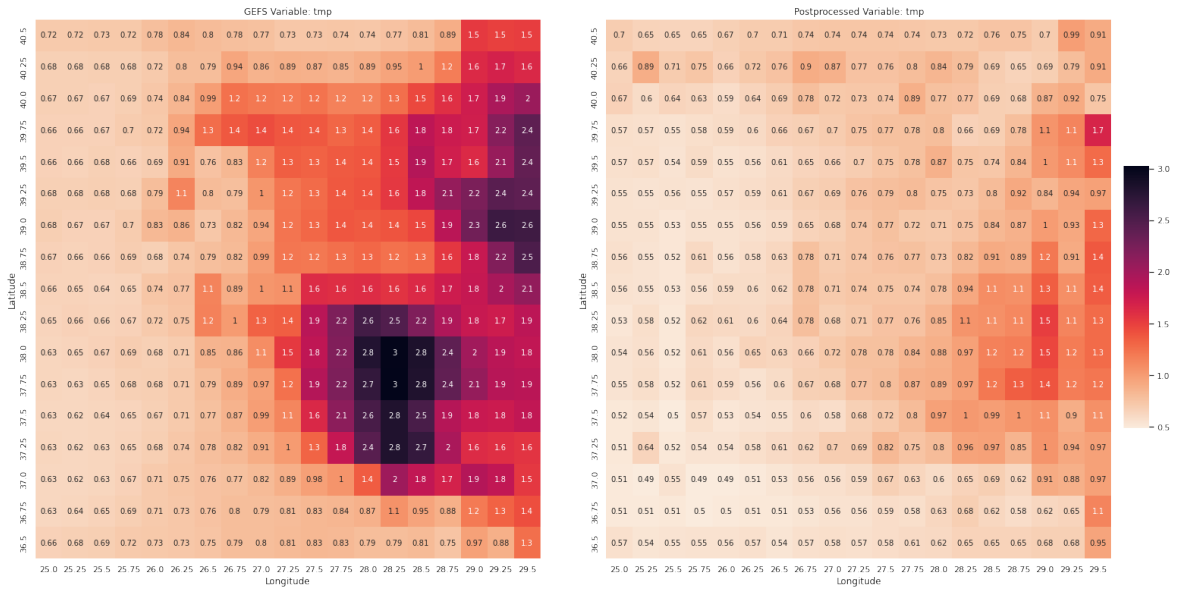

Figure 6.19. MSE performance of fully convolutional model for variable *tmp* at each location.

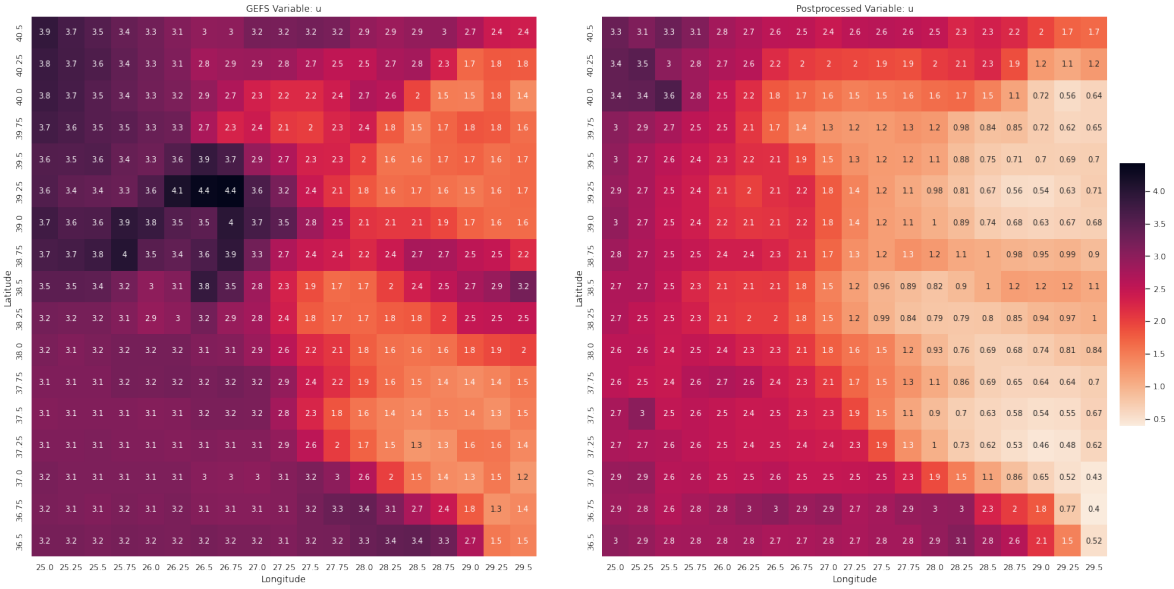

Figure 6.20. MSE performance of fully convolutional model for variable u at each location.

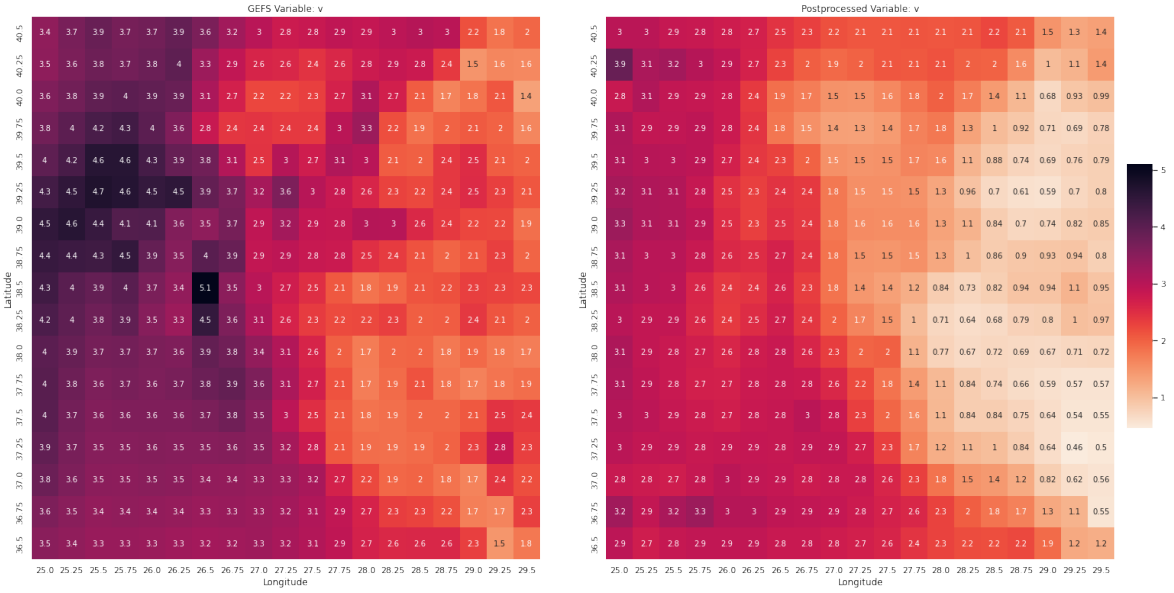

Figure 6.21. MSE performance of fully convolutional model for variable v at each location.

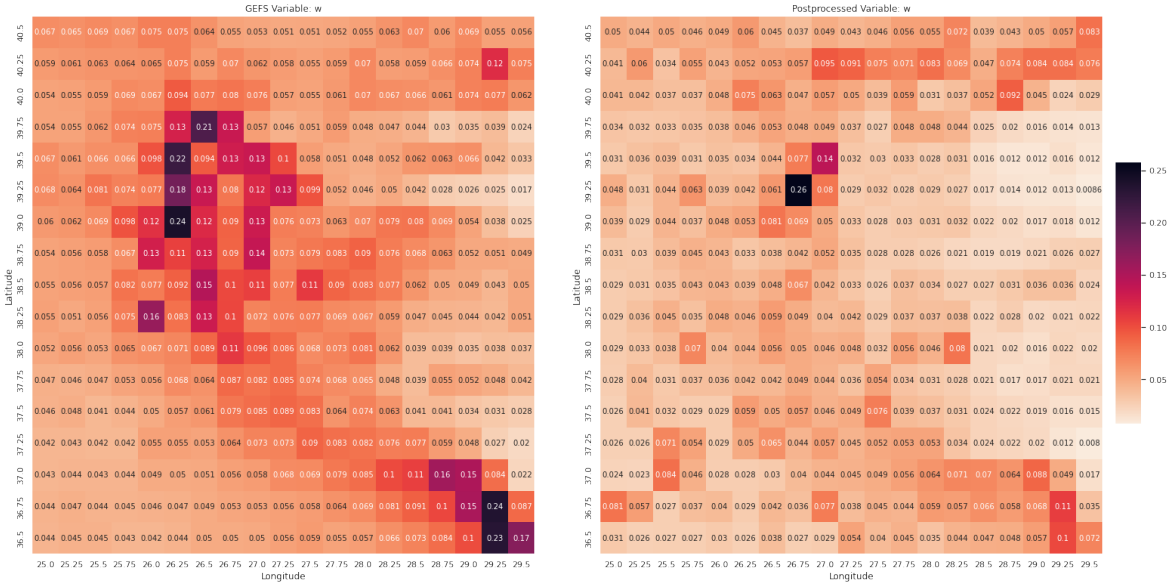

Figure 6.22. MSE performance of fully convolutional model for variable w at each location.

6.2.4. Variable-Month Based Performance

The month-based performances of variables *tmp*, u, v, and w are presented in Figures 6.23, 6.24, 6.25 and 6.26 respectively. The figures show that the model achieves to decrease in the median value and the variance of errors throughout the year for all variables.

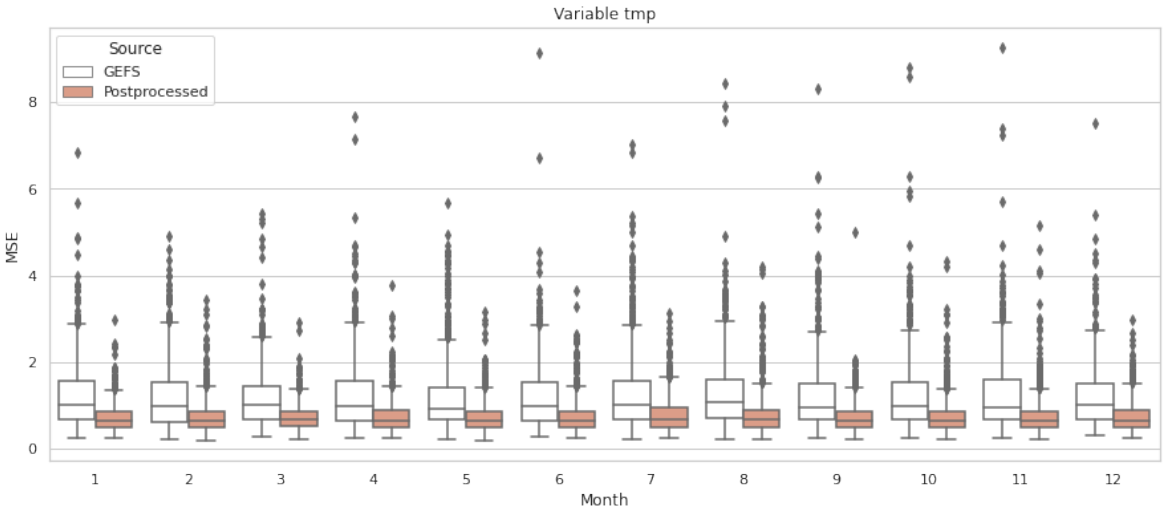

Figure 6.23. Month-based performance of variable *tmp* in fully convolutional architecture.

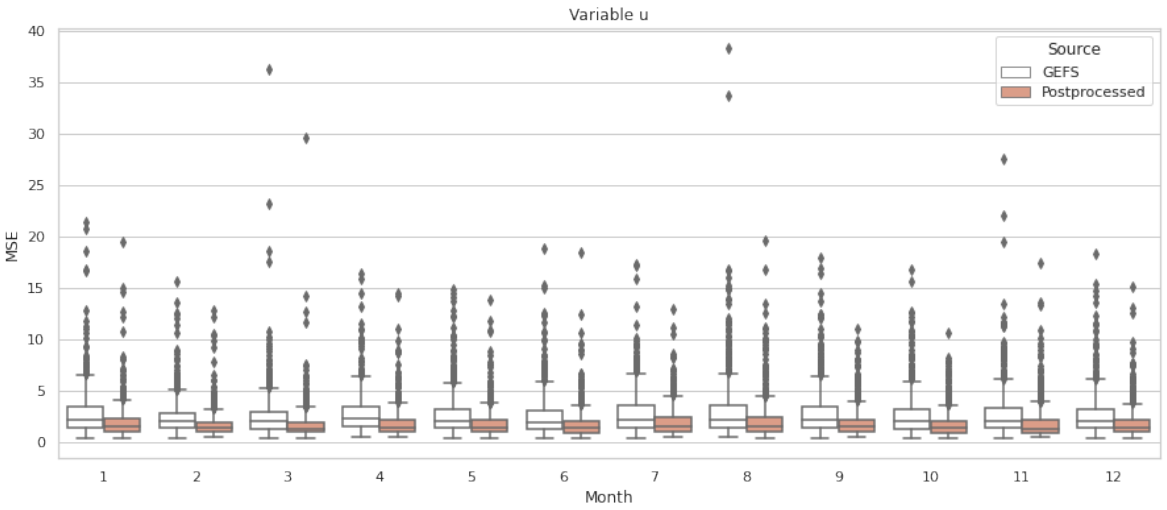

Figure 6.24. Month-based performance of variable u in fully convolutional architecture.

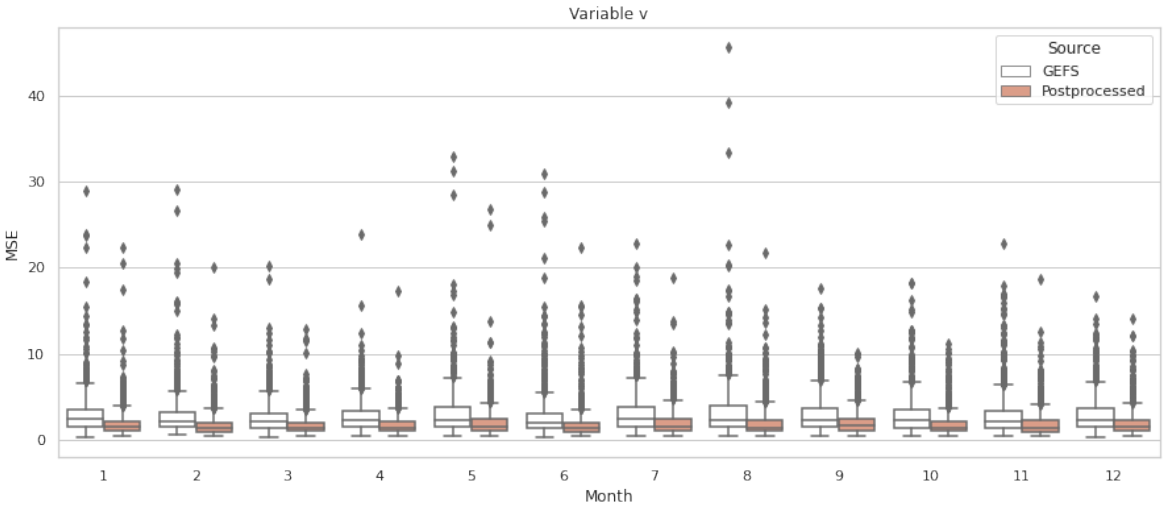

Figure 6.25. Month-based performance of variable v in fully convolutional architecture.

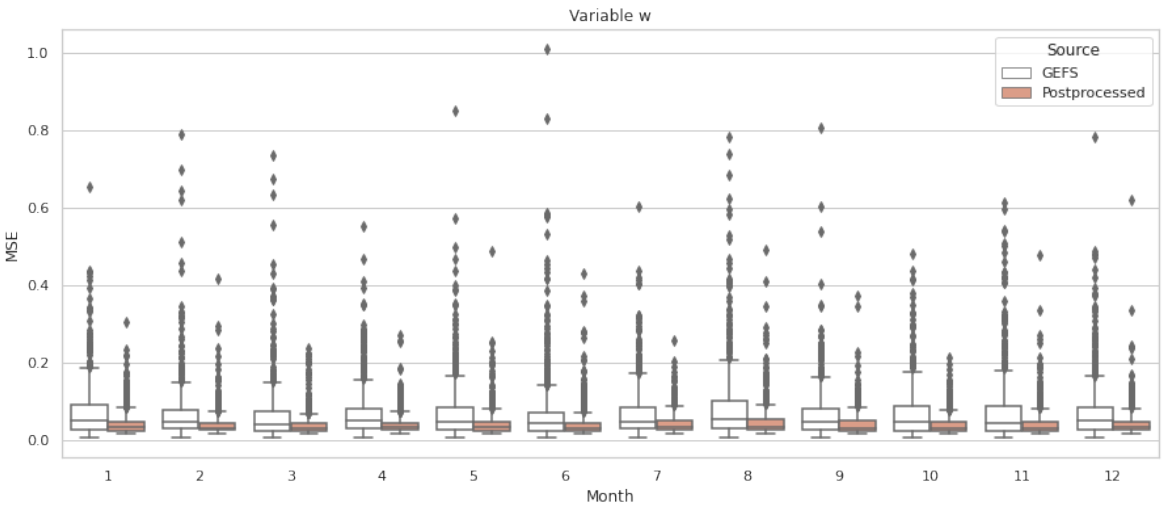

Figure 6.26. Month-based performance of variable w in fully convolutional architecture.

6.3. U-Shaped Artificial Neural Network

This section presents the results from the U-Shaped architecture mentioned in Section 4.2.3. The hyperparameter tuning of the model is made according to the experiment setting mentioned in Section 5.2. The minimum MSE loss for the validation period is observed as 0.00213 in the scaled domain. The best parameters in terms of validation loss are presented in Table 6.3. After the model trained with the best hyperparameters is evaluated in the test period. During the evaluation, the values are converted back to the real scale. Therefore, the evaluation is made for each variable separately.

Table 6.3. U-Shaped architecture best hyperparameters.

Hyperparameter	**Value**
Batch Size	16
Learning Rate	0.001
Weight Decay	0
# of Channels	512
Dropout Ratio	0
Batchnorm Usage	True
Pooling Function	Average Pooling

6.3.1. Variable-Based Performance

Figure 6.27 shows that U-Shaped architecture obtained better MSE values in all variables when compared with GEFS.

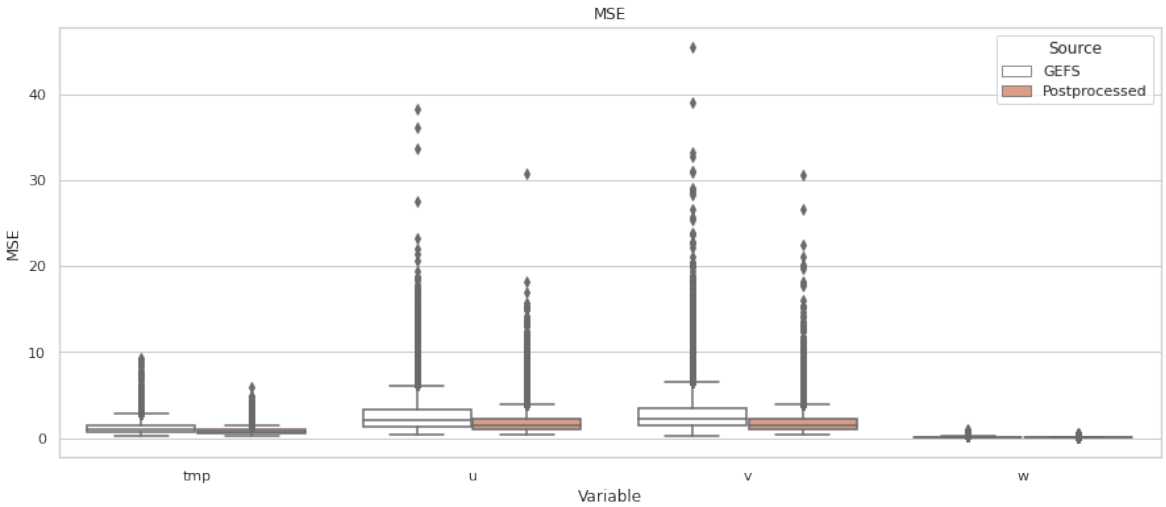

Figure 6.27. MSE performance of U-Shaped model for each variable.

6.3.2. Variable-Pressure Level-Based Performance

The pressure level-based performances of variables *tmp*, *u*, *v*, and *w* are presented in Figures 6.28, 6.29, 6.30 and 6.31 respectively. The model achieves better errors for all wind variables at each pressure level. For *tmp* the MSE values are better than GEFS for high pressures however, it falls behind in 800 and 850 *mb* levels.

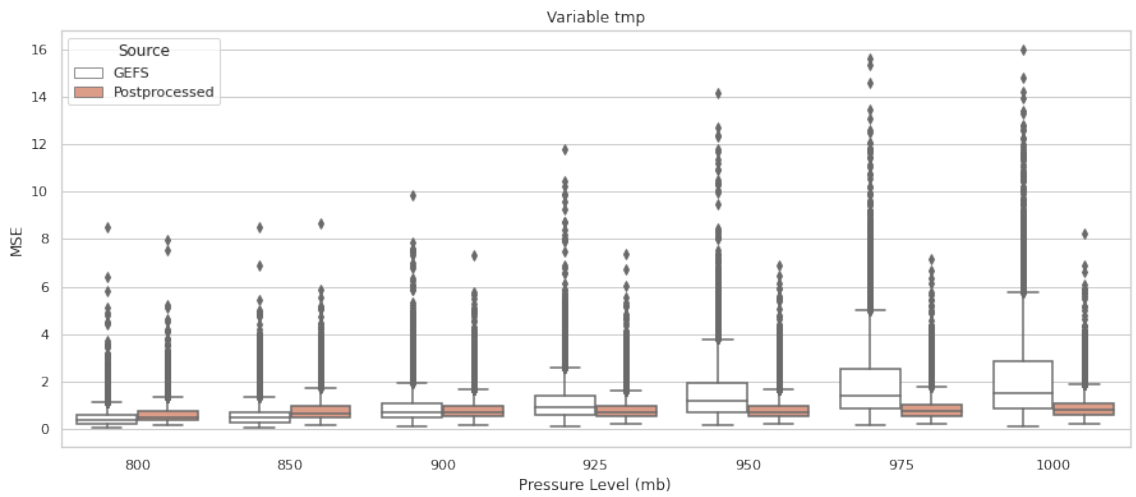

Figure 6.28. MSE performance of U-Shaped model for variable *tmp* at each pressure level.

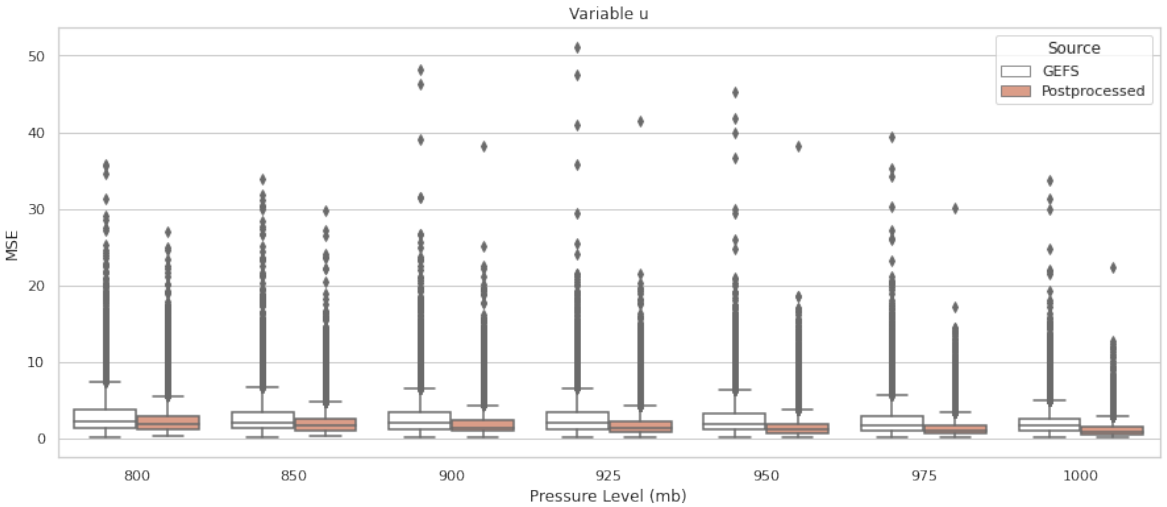

Figure 6.29. MSE performance of U-Shaped model for variable u at each pressure level.

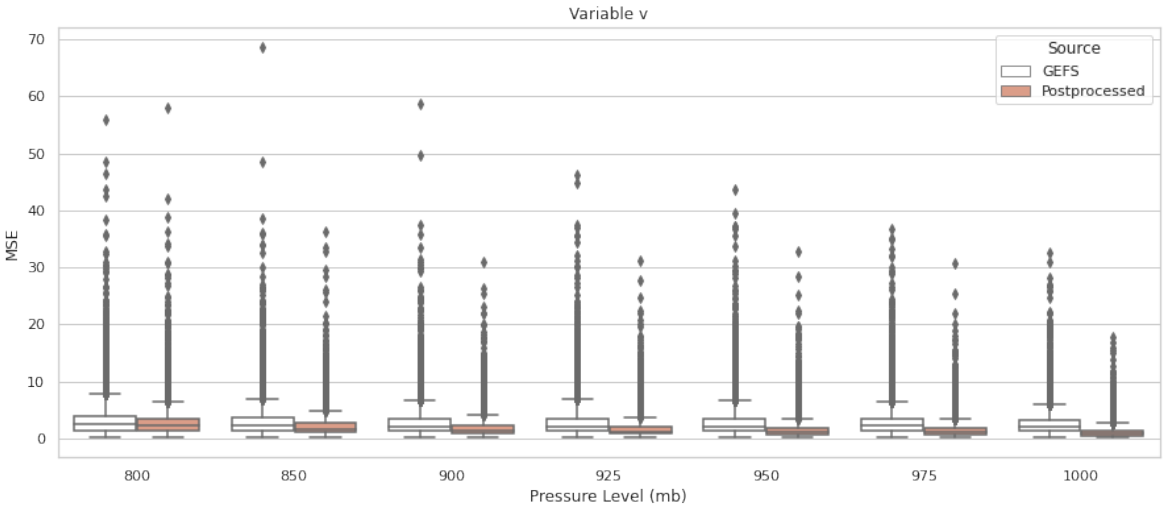

Figure 6.30. MSE performance of U-Shaped model for variable v at each pressure level.

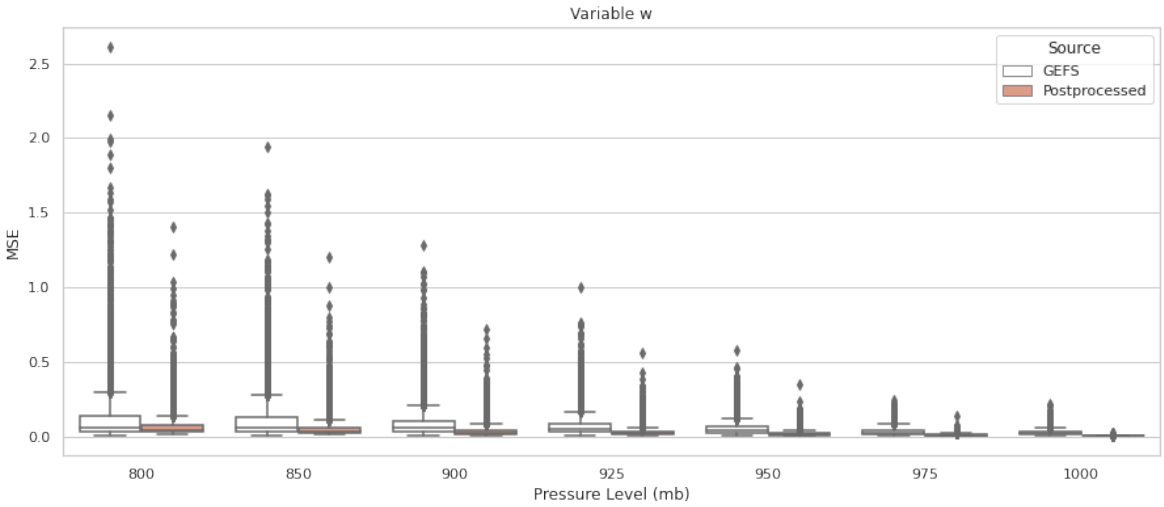

Figure 6.31. MSE performance of U-Shaped model for variable w at each pressure level.

6.3.3. Variable-Location Based Performance

The location-based performances of variables *tmp*, u, v and w are presented in Figures 6.32, 6.33, 6.34 and 6.35 respectively. The layout of the locations in the figure has geographic correspondence with the region in Figure 4.1. Therefore, the left (west) sides of the heat maps are equivalent to the coastal region, and the right (east) sides represent the inland region. For all variables, the MSE values of almost every location are reduced with the model. It fixes the inland region of variable *tmp* while improving the others. For u and v, all locations gain lighter colors, and for w high error region in the middle is almost fixed.

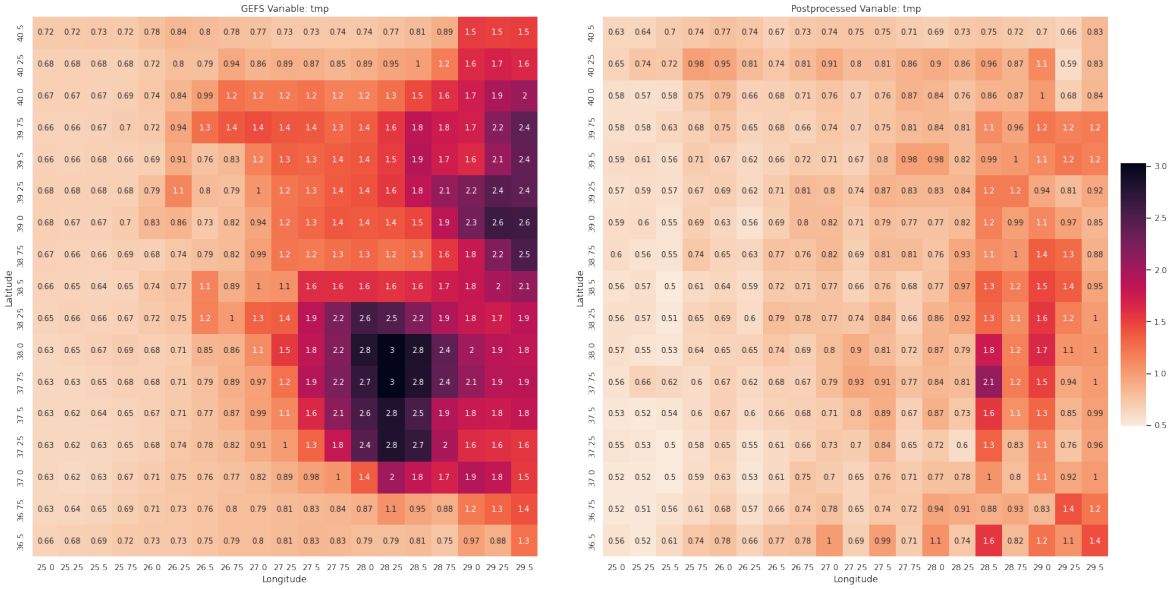

Figure 6.32. MSE performance of U-Shaped model for variable *tmp* at each location.

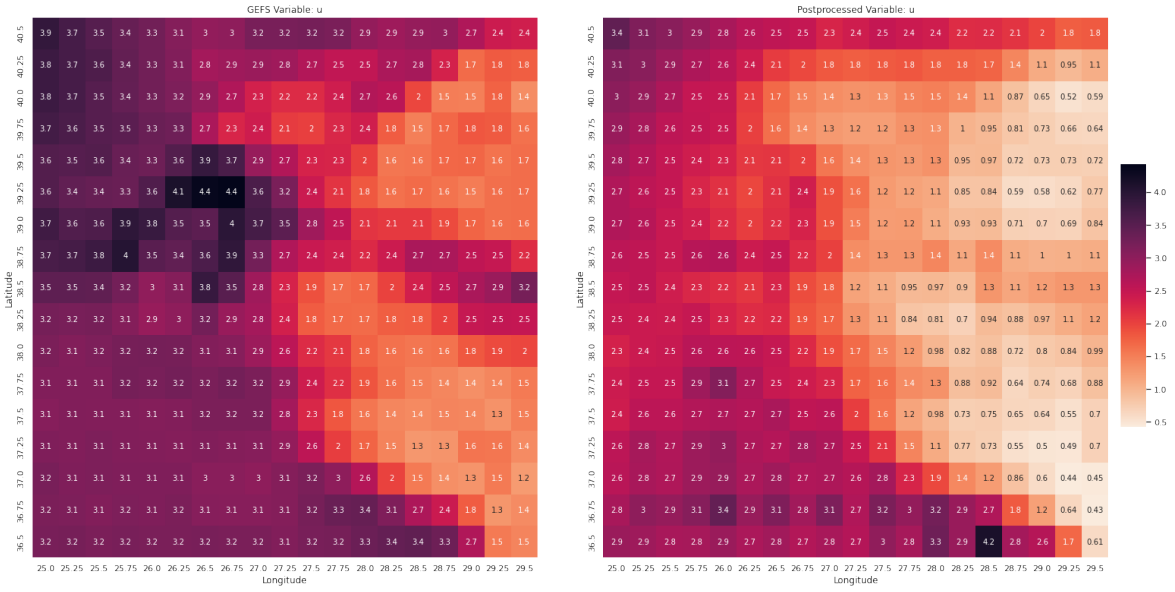

Figure 6.33. MSE performance of U-Shaped model for variable *u* at each location.

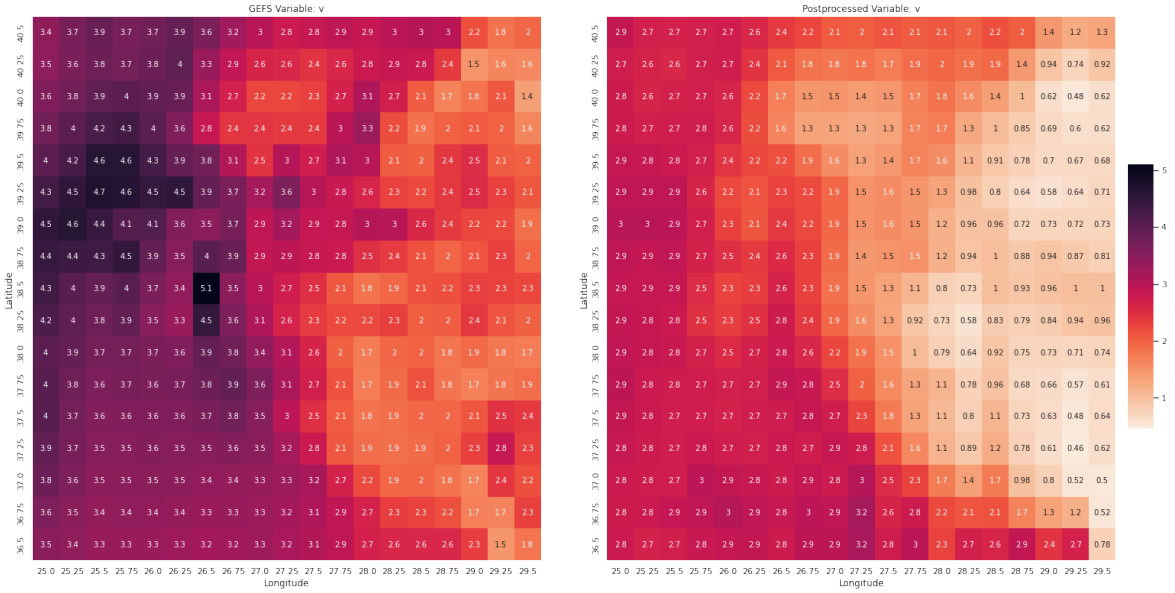

Figure 6.34. MSE performance of U-Shaped model for variable v at each location.

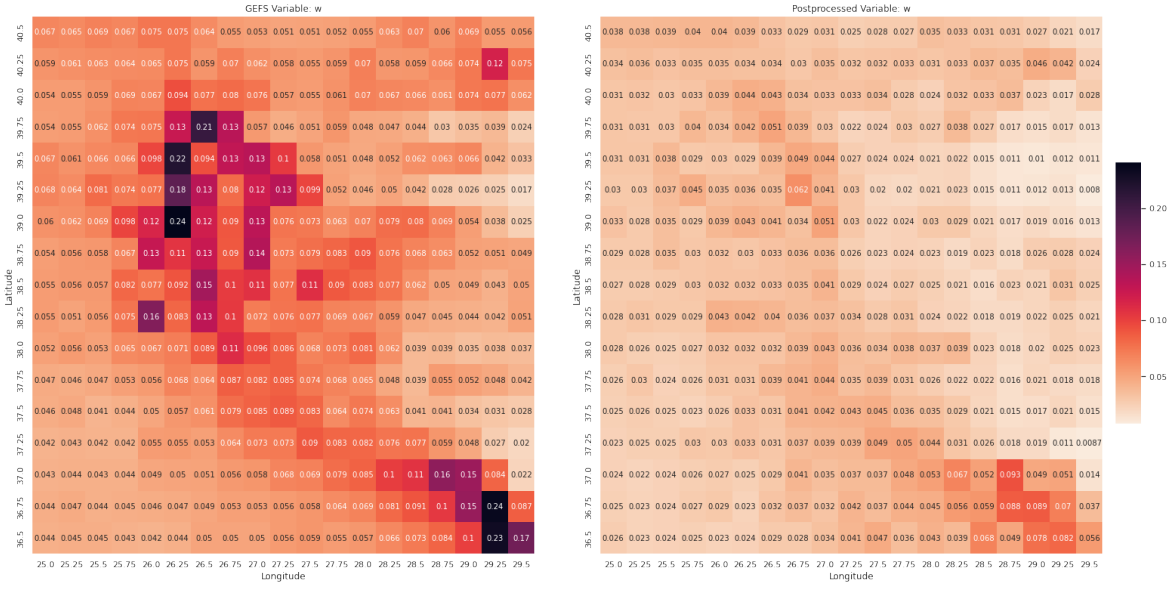

Figure 6.35. MSE performance of U-Shaped model for variable w at each location.

6.3.4. Variable-Month Based Performance

The month-based performances of variables *tmp*, *u*, *v*, and *w* are presented in Figures 6.23, 6.24, 6.25 and 6.26 respectively. The figures show that the model achieves to decrease in the median value and the variance of errors throughout the year for all variables.

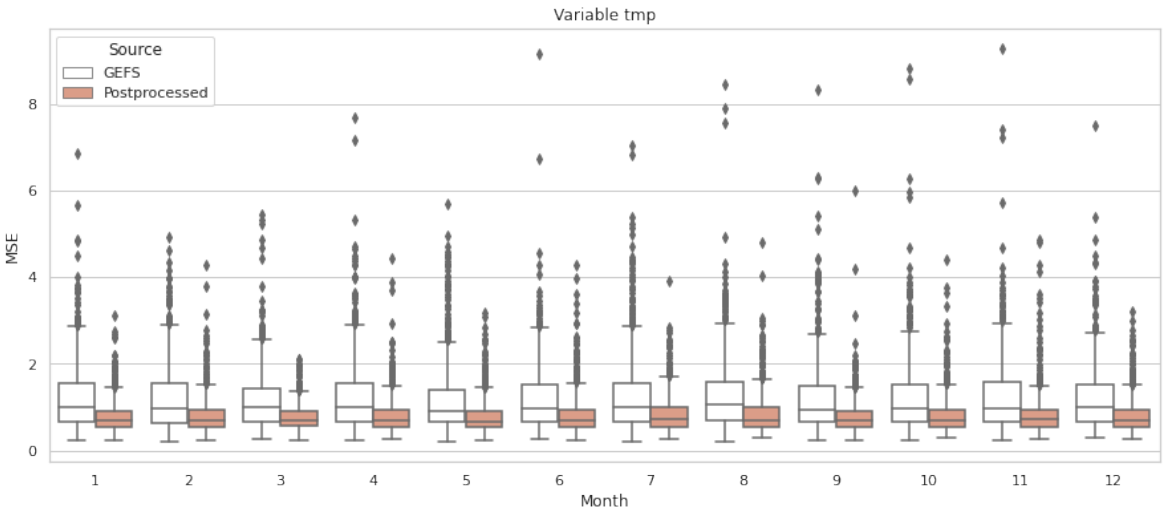

Figure 6.36. Month-based performance of variable *tmp* in U-Shaped architecture.

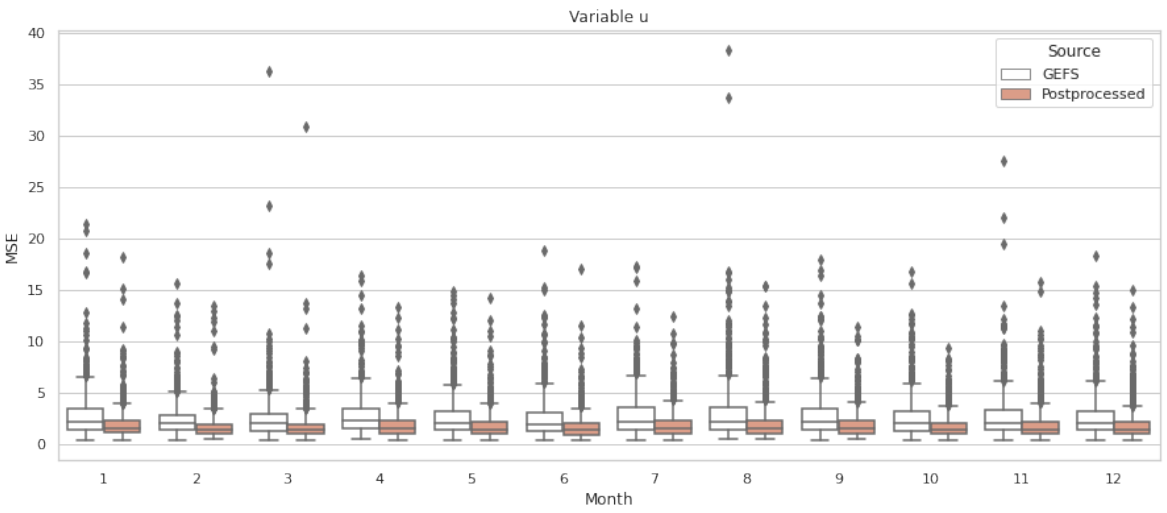

Figure 6.37. Month-based performance of variable *u* in U-Shaped architecture.

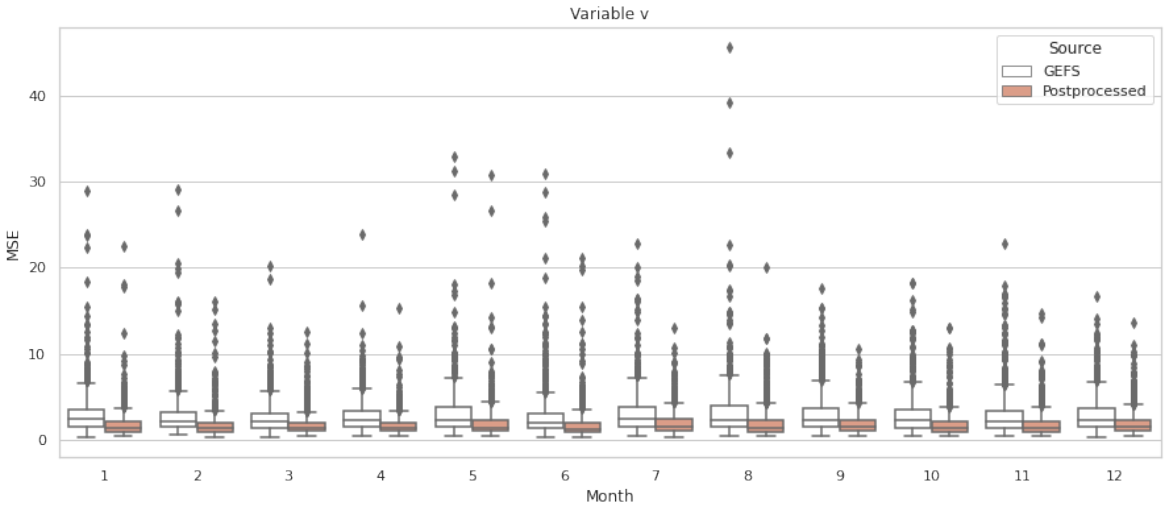

Figure 6.38. Month-based performance of variable v in U-Shaped architecture.

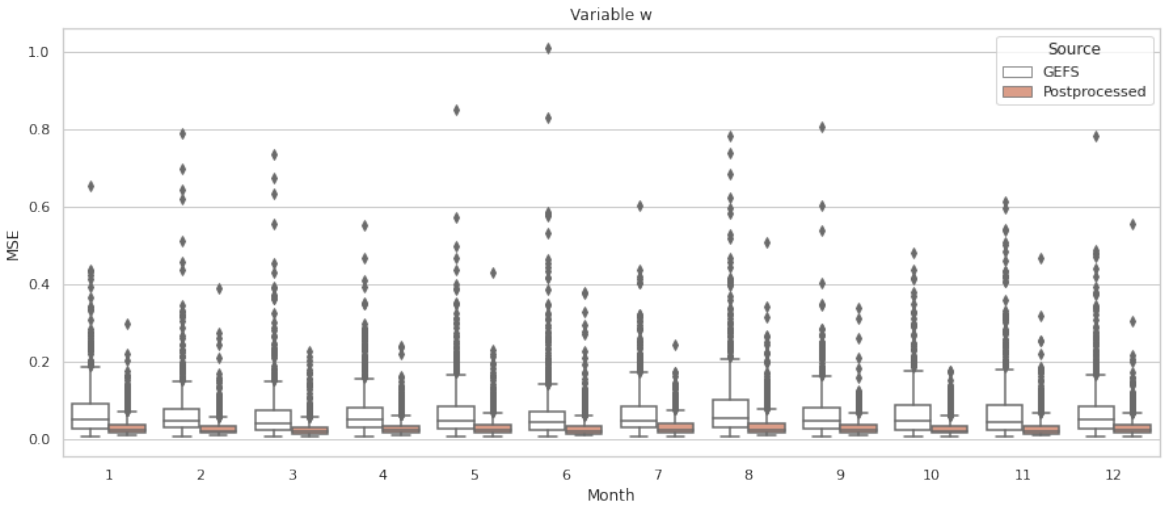

Figure 6.39. Month-based performance of variable w in U-Shaped architecture.

6.4. Comparing Architectures

After analyzing the details of each proposed architecture in detail, the final comparison of the models is made based on the MSE values of all variables. Table 6.4 presents these values with the errors before any postprocessing (GEFS). The results indicate that except for variable *tmp* in MLP, all of the methods decrease the MSE

values of every variable. MLP gets the best error for variable w, however, it performs poorly on the other ones. U-Shaped and Fully Convolutional models achieve similar MSE values. They both achieve great improvement when they are compared with GEFS. Especially at w, the U-Shaped model almost performs as well as MLP which mainly focuses on postprocess variable w. Additionally, the U-shaped performs best at variable v. The performance of the error of extrapolated values is showed in Table 6.5. For extrapolated values, U-Shaped model achieves better results than other architectures in all of the wind variables. As a result, the U-Shaped model is selected as the best model by considering its performance in both postprocessed and extrapolated values.

Table 6.4. MSE values of variables for different weather sources.

Model Name	**tmp**	**u**	**v**	**w**
GEFS	1.2175	2.6998	2.9829	0.0698
MLP	1.3164	2.3715	2.7288	0.0302
Fully Convolutional	0.7415	1.8493	1.9789	0.0421
U-Shaped	0.8070	1.8789	1.9112	0.0315

Table 6.5. MSE values of extrapolated variables for different weather sources.

Model Name	**tmp**	**u**	**v**	**w**
MLP	1.3240	2.5305	2.9515	0.0317
Fully Convolutional	0.8104	2.0983	2.2935	0.0434
U-Shaped	0.8687	2.0383	2.1870	0.0354

6.5. Extrapolation Capability of the Best Model

Previous sections have presented the results based on overlapping times, variables, and levels of GEFS and ERA5. But, the model also includes times, variables, and levels that don't present in GEFS to enrich the information during training. The

outputs of these dimensions are considered extrapolated. Unlike the overlapping values, there is no prior benchmark for them. However, comparing the error distribution of these values with the error distribution of the overlapping hours gives an idea about the extrapolation capability of the model. Figure 6.40 shows the box plot errors of each variable for GEFS, postprocessed values, and extrapolated values. GEFS and postprocessed values only include the beginning hour of the 3 hourly periods. The extrapolated values consist of the remaining two hours. The results demonstrate that the extrapolation is not as good as the postprocessing but the median error values are close. Additionally, when it is compared to GEFS, the error distribution of extrapolated values has a better median value and less variance.

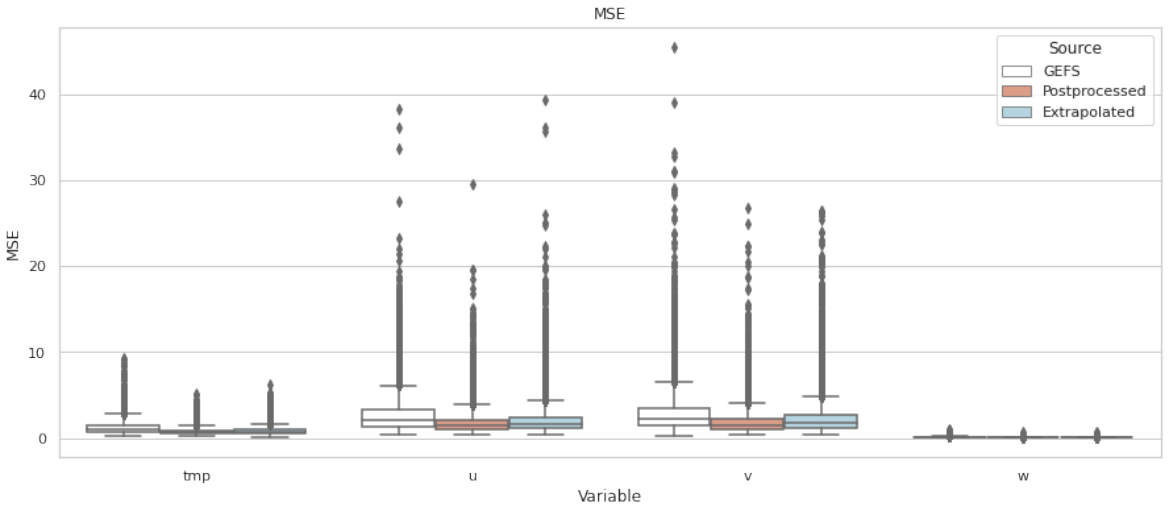

Figure 6.40. Distribution comparison of GEFS, postprocessed values, and extrapolated values of U-Shaped model.

6.6. Wind Power Forecasting

After selecting the U-Shaped model as the best model, the wind variables of all sources from the test period are used for wind power forecasting. As the training period, 2017 and 2018 are selected and the test performances are reported for the year 2019. The forecasting experiments are done as described in Section 5.3 for five different

sources. The resulting WMAPE values of each source for 19 different power plants are reported in Table 6.6.

The power generation of a wind farm is also affected by operational problems however, having better forecasts in many alternative power plants makes these effects ignorable while evaluating the quality of the weather source. The results support that ERA5 is a superior weather source. The postprocessed versions enhances the GEFS values and generates better forecasts for almost every farm. Especially, U-Shaped model achieves the best results for 13 out of 19 farms which supports the selection of U-Shaped model over the others. Additionally, the U-Shaped model usually obtains WMAPE values close to ERA5. In summary, the results demonstrate that postprocessing has an effect on a real-world problem and not only on MSE values.

Table 6.6. WMAPE values for different weather sources.

Farm Code	GEFS	ERA5	U-Shaped	Fully Conv.	MLP
40W000000000573X	0.3425	0.3080	**0.3307**	0.3496	0.3551
40W000000000587M	0.3466	0.2992	**0.2862**	0.3014	0.2891
40W000000000726Y	0.3911	0.3713	0.3882	**0.3747**	0.3945
40W000000000748O	0.3942	0.3684	0.3852	**0.3795**	0.3799
40W000000000760Y	0.2997	0.2569	**0.2658**	0.2705	0.2804
40W000000001581T	0.3561	0.3316	**0.3458**	0.3485	0.3460
40W000000002141F	0.3685	0.3123	**0.2982**	0.3159	0.3073
40W000000003302C	0.5182	0.4950	**0.5021**	0.5047	0.5061
40W0000000042063	0.3064	0.2779	0.2899	**0.2880**	0.3048
40W000000004889N	0.4336	0.3687	**0.3525**	0.3690	0.3721
40W000000005541L	0.2976	0.2796	0.2849	**0.2755**	0.2851
40W000000005611Q	0.3524	0.2972	0.3145	**0.3135**	0.3299
40W000000005874V	0.3779	0.3446	**0.3597**	0.3611	0.3765
40W0000000065377	0.3718	0.3403	0.3611	0.3544	**0.3536**
40W000000006616B	0.3468	0.3289	**0.3723**	0.3837	0.3922
40W0000000070982	0.3816	0.3637	0.3864	0.3930	**0.3795**
40W000000008459S	0.3213	0.2939	**0.3091**	0.3116	0.3247
40W000000008698A	0.4021	0.3640	0.3659	**0.3608**	0.3707
40W000000010501F	0.3283	0.2941	**0.2919**	0.2991	0.2992

7. CONCLUSION

Weather forecasts play a crucial role in many decision-making processes. The accuracy of the forecasts has a direct effect on the quality of the decisions. Currently, the main supply of weather forecasts is the numerical weather prediction (NWP) models. These models start from an initial state and solve the equations of atmospheric motion to come up with the future states of the atmosphere. However, these models are prone to errors from the initial states, boundary conditions, and model structures. These errors in the models grow rapidly due to the chaotic nature of the atmosphere dynamics. Therefore, the output of the NWP models needs postprocessing to fix these systematic errors and have more accurate forecasts.

In this thesis, alternative deep learning architectures are evaluated to statistically postprocess the forecast of an NWP model called Global Ensemble Forecasting System (GEFS). The Aegean Region of Turkey is selected as a local area to postprocess multiple weather variables at multiple pressure levels. The first architecture, the multilayer perceptron (MLP) failed to obtain better forecasts than GEFS for some of the variables at some pressure levels. On the other side, the fully convolutional architecture and its extension with skip connections achieved error distributions with significantly less mean and variance than the initial values for each level and pressure. Location-based analyses showed that the improvement applies to whole points in the region. The extension, the U-Shaped model is chosen over the fully convolutional architecture due to its more preferable error distributions.

Additionally, the models have extrapolation capability since they are trained to a response that has more resolution in terms of time and pressure levels. These extra values are kept to allow the model to learn from this information and they introduced an extrapolation capability to the postprocessing models. The extrapolation capability of the chosen model is validated by comparing the error distributions of the extrapolated values with the error distribution of the GEFS forecasts and postprocessed values.

The results showed that extrapolated values are significantly better than GEFS and slightly worse than postprocessed ones. Lastly, a case study on 19 wind power plants demonstrated that the forecasts made by the U-Shaped model outputs are not far from the ones made by ERA5 and are significantly better than using GEFS in almost all of the plants.

There is a broad space for future works in statistical postprocessing using deep learning. First of all, the effect of the variable-level selection can be investigated further. Secondly, the proposed models are not capable of modeling temporal relations. Experimenting on how to model temporal relations and measuring the effect of modeling them could lead to better postprocessing methods. The convolutions in this study are made in 2D along latitude and longitude values, however 3D convolutions including pressure levels may lead to better spatial modeling. Since all values are the results of physical processes, the method can be combined with physics-informed learning to come up with physically more satisfying results. Also, there can be improvements in model inputs. Apart from the control GEFS forecasts, the ensemble members of GEFS can be fed to the models. Using these ensembles is expected to behave like adversarial training and make the models more robust to input changes. Another study can be made by using different sources of NWP models to blend the sources during the postprocessing. By using multiple NWPs, models can make use of more information to come up with superior postprocessed values. Lastly, the input values can be perturbed and fed to the model to create postprocessing-based ensembles that define the forecast uncertainty.

REFERENCES

1. Lazo, J. K., R. E. Morss and J. L. Demuth, "300 Billion Served: Sources, Perceptions, Uses, and Values of Weather Forecasts", *Bulletin of the American Meteorological Society*, Vol. 90, No. 6, pp. 785 – 798, 2009.

2. Vannitsem, S., "Predictability of Large-Scale Atmospheric Motions: Lyapunov Exponents and Error Dynamics", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, Vol. 27, No. 3, p. 032101, 2017.

3. Nicolis, C., R. A. Perdigao and S. Vannitsem, "Dynamics of Prediction Errors Under the Combined Effect of Initial Condition and Model Errors", *Journal of the Atmospheric Sciences*, Vol. 66, No. 3, pp. 766–778, 2009.

4. Nicolis, C., "Dynamics of Model Error: The Role of the Boundary Conditions", *Journal of the Atmospheric Sciences*, Vol. 64, No. 1, pp. 204–215, 2007.

5. Vannitsem, S., J. B. Bremnes, J. Demaeyer, G. R. Evans, J. Flowerdew, S. Hemri, S. Lerch, N. Roberts, S. Theis, A. Atencia, Z. B. Bouallègue, J. Bhend, M. Dabernig, L. D. Cruz, L. Hieta, O. Mestre, L. Moret, I. O. Plenković, M. Schmeits, M. Taillardat, J. V. den Bergh, B. V. Schaeybroeck, K. Whan and J. Ylhaisi, "Statistical Postprocessing for Weather Forecasts: Review, Challenges, and Avenues in a Big Data World", *Bulletin of the American Meteorological Society*, Vol. 102, No. 3, pp. E681–E699, 2021.

6. Raftery, A. E., T. Gneiting, F. Balabdaoui and M. Polakowski, "Using Bayesian Model Averaging to Calibrate Forecast Ensembles", *Monthly Weather Review*, Vol. 133, No. 5, pp. 1155–1174, 2005.

7. Gneiting, T., A. E. Raftery, A. H. Westveld III and T. Goldman, "Calibrated Probabilistic Forecasting using Ensemble Model Output Statistics and Minimum

CRPS Estimation", *Monthly Weather Review*, Vol. 133, No. 5, pp. 1098–1118, 2005.

8. Messner, J. W., G. J. Mayr and A. Zeileis, "Non-homogeneous Boosting for Predictor Selection in Ensemble Post-Processing", *Monthly Weather Review*, Vol. 145, pp. 137–147, 2017.

9. Lang, M. N., G. J. Mayr, R. Stauffer and A. Zeileis, "Bivariate Gaussian Models for Wind Vectors in a Distributional Regression Framework", *Advances in Statistical Climatology, Meteorology and Oceanography*, Vol. 5, No. 2, pp. 115–132, 2019.

10. Schlosser, L., T. Hothorn, R. Stauffer and A. Zeileis, "Distributional Regression Forests for Probabilistic Precipitation Forecasting in Complex Terrain", *The Annals of Applied Statistics*, Vol. 13, No. 3, pp. 1564–1589, 2019.

11. Rasp, S. and S. Lerch, "Neural Networks for Postprocessing Ensemble Weather Forecasts", *Monthly Weather Review*, Vol. 146, No. 11, pp. 3885–3900, 2018.

12. Bremnes, J. B., "Probabilistic Forecasts of Precipitation in Terms of Quantiles using NWP Model Output", *Monthly Weather Review*, Vol. 132, No. 1, pp. 338–347, 2004.

13. Wahl, S., *Uncertainty in Mesoscale Numerical Weather Prediction: Probabilistic Forecasting of Precipitation*, Ph.D. Thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2015.

14. Bremnes, J. B., "Constrained Quantile Regression Splines for Ensemble Postprocessing", *Monthly Weather Review*, Vol. 147, No. 5, pp. 1769–1780, 2019.

15. Velthoen, J., J.-J. Cai, G. Jongbloed and M. Schmeits, "Improving Precipitation Forecasts using Extreme Quantile Regression", *Extremes*, Vol. 22, No. 4, pp. 599–622, 2019.

16. Taillardat, M., O. Mestre, M. Zamo and P. Naveau, "Calibrated Ensemble Forecasts using Quantile Regression Forests and Ensemble Model Output Statistics", *Monthly Weather Review*, Vol. 144, No. 6, pp. 2375–2393, 2016.

17. Hamill, T. M. and J. S. Whitaker, "Probabilistic Quantitative Precipitation Forecasts Based on Reforecast Analogs: Theory and Application", *Monthly Weather Review*, Vol. 134, No. 11, pp. 3209–3229, 2006.

18. Delle Monache, L., F. A. Eckel, D. L. Rife, B. Nagarajan and K. Searight, "Probabilistic Weather Prediction with an Analog Ensemble", *Monthly Weather Review*, Vol. 141, No. 10, pp. 3498–3516, 2013.

19. Alessandrini, S., L. Delle Monache, C. M. Rozoff and W. E. Lewis, "Probabilistic Prediction of Tropical Cyclone Intensity with an Analog Ensemble", *Monthly Weather Review*, Vol. 146, No. 6, pp. 1723–1744, 2018.

20. Odak Plenković, I., I. Schicker, M. Dabernig, K. Horvath and E. Keresturi, "Analog-Based Post-Processing of the ALADIN-LAEF Ensemble Predictions in Complex Terrain", *Quarterly Journal of the Royal Meteorological Society*, Vol. 146, No. 729, pp. 1842–1860, 2020.

21. Grönquist, P., C. Yao, T. Ben-Nun, N. Dryden, P. Dueben, S. Li and T. Hoefler, "Deep Learning for Post-Processing Ensemble Weather Forecasts", *Philosophical Transactions of the Royal Society A*, Vol. 379, No. 2194, p. 20200092, 2021.

22. Pathak, J., S. Subramanian, P. Z. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z.-Y. Li, K. Azizzadenesheli, P. Hassanzadeh, K. Kashinath and A. Anandkumar, "FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators", ArXiv:2202.11214 [physics.ao-ph], 2022.

23. Arcomano, T., I. Szunyogh, J. Pathak, A. Wikner, B. R. Hunt and E. Ott, "A Ma-

chine Learning-based Global Atmospheric Forecast Model", *Geophysical Research Letters*, Vol. 47, No. 9, p. e2020GL087776, 2020.

24. Keisler, R., "Forecasting Global Weather with Graph Neural Networks", ArXiv:2202.07575 [physics.ao-ph], 2022.

25. Gneiting, T. and A. E. Raftery, "Strictly Proper Scoring Rules, Prediction, and Estimation", *Journal of the American Statistical Association*, Vol. 102, No. 477, pp. 359–378, 2007.

26. Roulston, M. S. and L. A. Smith, "Combining Dynamical and Statistical Ensembles", *Tellus A: Dynamic Meteorology and Oceanography*, Vol. 55, No. 1, pp. 16–30, 2003.

27. Bröcker, J. and L. A. Smith, "From Ensemble Forecasts to Predictive Distribution Functions", *Tellus A: Dynamic Meteorology and Oceanography*, Vol. 60, No. 4, pp. 663–678, 2008.

28. Hewson, T. D. and F. M. Pillosu, "A New Low-Cost Technique Improves Weather Forecasts Across the World", ArXiv:2003.14397 [physics.ao-ph], 2020.

29. Bremnes, J. B., "Ensemble Postprocessing using Quantile Function Regression Based on Neural Networks and Bernstein Polynomials", *Monthly Weather Review*, Vol. 148, No. 1, pp. 403–414, 2020.

30. Veldkamp, S., K. Whan, S. Dirksen and M. Schmeits, "Statistical Postprocessing of Wind Speed Forecasts using Convolutional Neural Networks", *Monthly Weather Review*, Vol. 149, No. 4, pp. 1141–1152, 2021.

31. Pinson, P. and R. Girard, "Evaluating the Quality of Scenarios of Short-Term Wind Power Generation", *Applied Energy*, Vol. 96, pp. 12–20, 2012.

32. Schefzik, R., T. L. Thorarinsdottir and T. Gneiting, "Uncertainty Quantification in

Complex Simulation Models using Ensemble Copula Coupling", *Statistical Science*, Vol. 28, No. 4, pp. 616–640, 2013.

33. Clark, M., S. Gangopadhyay, L. Hay, B. Rajagopalan and R. Wilby, "The Schaake Shuffle: A Method for Reconstructing Space–Time Variability in Forecasted Precipitation and Temperature Fields", *Journal of Hydrometeorology*, Vol. 5, No. 1, pp. 243–262, 2004.

34. Sperati, S., S. Alessandrini and L. Delle Monache, "Gridded Probabilistic Weather Forecasts with an Analog Ensemble", *Quarterly Journal of the Royal Meteorological Society*, Vol. 143, No. 708, pp. 2874–2885, 2017.

35. Tieleman, T. and G. Hinton, "Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude", *COURSERA: Neural Networks for Machine Learning*, Vol. 4, No. 2, pp. 26–31, 2012.

36. Kingma, D. and J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference on Learning Representations*, 2014.

37. Ioffe, S. and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *International Conference on Machine Learning*, pp. 448–456, PMLR, 2015.

38. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT press, 2016.

39. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a Simple Way to Prevent Neural Networks from Overfitting", *The Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929–1958, 2014.

40. Veličković, P., "TikZ", 2018, `https://github.com/PetarV-/TikZ`, accessed on July 27, 2022.

41. He, K., X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recog-

nition", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

42. Huang, G., Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

43. Ronneberger, O., P. Fischer and T. Brox, "U-net: Convolutional Networks for Biomedical Image Segmentation", *International Conference on Medical Image Computing and Computer-assisted Intervention*, pp. 234–241, Springer, 2015.

44. Zou, H. and T. Hastie, "Regularization and Variable Selection via the Elastic Net", *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 67, No. 2, pp. 301–320, 2005.

45. Friedman, J., T. Hastie and R. Tibshirani, "Regularization Paths for Generalized Linear Models via Coordinate Descent", *Journal of Statistical Software*, Vol. 33, No. 1, p. 1, 2010.

46. Hamill, T. M., G. T. Bates, J. S. Whitaker, D. R. Murray, M. Fiorino, T. J. Galarneau, Y. Zhu and W. Lapenta, "NOAA's Second-Generation Global Medium-Range Ensemble Reforecast Dataset", *Bulletin of the American Meteorological Society*, Vol. 94, No. 10, pp. 1553 – 1565, 2013.

47. Hersbach, H., W. Bell, P. Berrisford, A. Horányi, M.-S. J., J. Nicolas, R. Radu, D. Schepers, A. Simmons, C. Soci and D. Dee, "Global Reanalysis: Goodbye ERA-Interim, Hello ERA5", *ECMWF Newsletter*, pp. 17–24, 2019.

48. "Gerçek Zamanlı Üretim - Energy Exchange Istanbul", 2017, `https://seffaflik.epias.com.tr/transparency/uretim/gerceklesen-uretim/gercek-zamanli-uretim.xhtml`, accessed on June 29, 2022.

49. "Yük Tevzi Dairesi Başkanlığı - Aralık Kurulu Güç Raporu", 2021,

https://webapi.teias.gov.tr/file/9e6326b1-5273-45c9-8f90-6d511cac882a?download, accessed on June 29, 2022.

50. "8th Turkish Wind Energy Congress", 2022, https://tureb.com.tr//lib/uploads/4e77501b714739a9.pdf, accessed on August 12, 2022.

51. Kıvrıl, Y. H., "Statistical Postprocessing of Local Numerical Weather Prediction Model Forecasts using Deep Learning", 2022, https://github.com/harunkivril/nnPostProcess.

52. "NCEP WMO GRIB2 Documentation", https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc/, accessed on August 12, 2022.

53. Falcon, W., "Pytorch Lightning", 2022, https://github.com/PyTorchLightning/pytorch-lightning.

54. Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Vancouver, BC, Canada, 2019.

55. Akiba, T., S. Sano, T. Yanase, T. Ohta and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework", *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

APPENDIX A: WIND FARMS METADATA

Table A.1. Farm codes, names, and bounding boxes of the power plants.

Farm Code	Farm Name	Left	Right	Bottom	Above
40W000000000573X	BARES	28.00	28.25	40.25	40.50
40W000000000748O	MAZI RES	26.25	26.50	38.00	38.25
40W000000002141F	BERGAMA RES	27.00	27.25	38.75	39.00
40W000000003302C	SOMA1-2 RES	27.50	27.75	39.25	39.50
40W0000000065377	ZEYTİNELİ RES	26.25	26.50	38.00	38.25
40W000000008698A	PİTANE RES	26.75	27.00	38.75	39.00
40W000000000760Y	SOMA RES	27.75	28.00	39.25	39.50
40W000000004889N	SEYİTALİ RES	27.00	27.25	38.75	39.00
40W000000005611Q	POYRAZ RES	28.00	28.25	39.75	40.00
40W0000000042063	KUYUCAK RES	27.75	28.00	39.25	39.50
40W000000010501F	KIRKAGAC RES	27.50	27.75	39.00	39.25
40W000000000587M	YUNTDAG RES	27.00	27.25	38.75	39.00
40W000000001581T	KOCADAG RES	26.50	26.75	38.25	38.50
40W000000005541L	GERES	27.75	28.00	39.00	39.25
40W000000000726Y	DUZOVA RES	27.00	27.25	39.00	39.25
40W000000006616B	EDINCIK RES	27.75	28.00	40.25	40.50
40W000000005874V	GUNAYDIN RES	28.00	28.25	39.75	40.00
40W0000000070982	SALMAN RES	26.25	26.50	38.50	38.75
40W000000008459S	ORTAMANDIRA RES	27.75	28.00	39.50	39.75