Academic Year 2020 – 2021
Research Unit: LEGI
Team: MEIGE
Supervisor: Prof. Chantal Staquet
Co-supervisors: Dr. Mikhail Krinitskiy, Dr. Sara Bacer

ENVIRONMENTAL FLUID MECHANICS

Prediction of Wintertime Atmospheric Stability in the Grenoble Valley around 2030 via Machine Learning

M2 Internship Report

Milton Salvador Gómez Deldagillo

Abstract

Decreasing air quality is a major cause of concern due to its links to negative health effects. There have been studies linking a decrease in air quality to persistent cold air pools (CAPs), also referred to as persistent temperature inversions, periods of air stagnation during the wintertime characterized by calm weather conditions and the presence of highly stable layers of air. Grenoble, an urbanized valley in the Alps, is affected by these pollution events and studies on the region have estimated the temperature inversions via the bulk temperature gradient across the valley. The objective of this internship was to predict the wintertime temperature gradient across the valley for the next decade using deep learning methods. The wintertime temperature gradient serves as an indicator of the impact of climate change on atmospheric stability, and therefore an indicator of air quality. The application of convolutional neural networks and long short term memory (LSTM) networks is explored. A neural network, comprising VGG19 as a feature extractor that is connected to an LSTM cell and a set of dense layers, is used to predict the vertical temperature gradient. The network is found to satisfactorily predict the temperature gradient using inputs extracted from a regional climate model (RCM) forced by reanalysis. Experiments were conducted applying the neural network to climate predictions from the RCM forced by a general circulation model (GCM). The predictions using the GCM as the forcing for the RCM exhibit a disparity to those using reanalysis as the forcing. Analyses of the predictions using future climate data indicate a trend toward more extreme values in the temperature gradient across the valley, but further work is required to assess the statistical significance of this result.

Keywords: Deep learning, climate, atmospheric pollution, Grenoble, convolutional, LSTM, long short term memory, transfer learning

Foreword and Acknowledgements

« *It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers.* »
— Patrick Rothfuss, The Wise Man's Fear

This work would undoubtedly have been impossible for me without the help and guidance from Drs. Mikhail Krinitskiy, Chantal Staquet, and Sara Bacer. Thank you for your time and patience, as well as your company through this journey. Drs. Martin Ménégoz, Hubert Gallée, and Julien Beaumet, thank you for your contributions to the project.

To my family: los quiero y los extraño. Quisiera que pudieran estar conmigo en estos momentos, pero mi esperanza es que algún día será más facil el juntarnos. Gordo, Hermani, Tía, Leo, Bella, Juanca, Mami, Abuelita Inés.

Carlos, Iván, y Pamela: es gracias a ustedes que me aventé a venir acá y sin su apoyo y amistad no hubiera logrado nada de esto. Se los agradezco desde el fondo del alma.

Florence, Aimé, Gabriel, Alexia: merci de m'avoir accueilli; la vie au Rabot ne serait pas aussi belle sans vous.

Almut and Francisco: it was an honor to have been part of the same cohort.

Dr. Achim Wirth: your courses were enlightening and entertaining; if one day I become half the educator that you are, I will consider myself a lucky man.

Laura, Guada, Ricardo, Fernando: están lejos en Taiwan pero cerca en mis pensamientos.

Drs. Lin Kuchin, Songhao Wang, and Jose Leon: it was only with your help that I was able to embark on this journey. Thank you.

Alex, Braulio, Clau, Vic, Saul, Mario, Adris, Boza, Steph, Victor, Javiera, Vernon, Tania, Ivy, Karen, Estefanía, Vi.

This Masters has given me many questions that I expect will fill my thoughts for many years, and there are far too many people I would like to thank and not enough space to do so.

Contents

1 Introduction

There has been a significant amount of research on the effect of air pollution on human health, linking its presence to a number of short and long term health effects (Freire et al. (2010)). These effects include, e.g., adverse reproductive effects, cancer, respiratory disorders, and perhaps even neurocognitive effects (e.g., Suglia et al. (2008); Block and Calderón-Garcidueñas (2009); Jorcano et al. (2019); Daiber et al. (2020)). In light of this, it is not surprising that pollution is a major concern throughout the world (Gulia et al., 2015).

This concern has resulted in numerous studies on atmospheric pollution, some of which link air stagnation to a detrimental effect on air quality (Garrido-Perez et al., 2018). In valleys, this stagnation can manifest as cold-air pools (CAPs), also referred to as temperature inversions, characterized by calm weather conditions and the presence of highly stable layers of air. These CAPs suppress the vertical mixing of pollutants, and the pollutants are thereby trapped under a mass of air over the area affected by the CAPs. The CAPs are generally classified according to the timescale at which they manifest, with nocturnal inversions that are formed and destroyed within the boundary layer diurnal cycle and persistent inversions that can last several days and up to multiple weeks (Largeron and Staquet, 2016a; Lareau et al., 2013). Persistent inversions, the focus of the present work due to their association with a prolonged reduction in air quality, are formed under anticyclonic conditions at the synoptic scale, when a warm mass of air at mid altitudes cause decoupling of the boundary layer in the valley from the free atmosphere.

One region affected by these persistent events is the Grenoble Valley (an urbanized valley in the French Alps), as evidenced by Chemel et al. (2007)'s study on a wintertime pollution episode, which was described as being marked by particularly stable and calm weather conditions trapping pollutants under a well defined temperature inversion. This is further evidenced by Largeron and Staquet (2016b), who develop a method of estimating persistent wintertime inversions by the heat valley deficit or the bulk temperature gradient. There is thus a marked interest in predicting the presence of CAPs, but this remains a challenging task.

(a) clear sky (b) visible cold air pooling

Figure 1 – View of Grenoble from Le Rabot during winter, approximately 100 m above Grenoble.

In numerical weather prediction, general circulation models (GCMs) and more specifically Earth System Models (ESMs) are considered the gold-standard of climate modelling (Watson-Parris, 2021). These models, however, are generally run at relatively low resolutions (e.g., 100 kilometers) (Chantry et al., 2021). Though there are ongoing efforts to increase the resolution of these models, it was estimated by Ringler et al. (2011) that achieving a 4km grid would require over 4000 times the computing power available at the time. Considering CAPs are local phenomena, it is clear that it is not currently possible to study the formation of CAPs in the coming decades by relying solely on ESM simulations.

A higher resolution alternative to ESMs can be found in regional climate models (RCMs), which give results on a finer grid while relying on ESMs to drive the boundary conditions. However, there are important limitations surrounding RCMs, as their relatively high computational load places « a limit on

the number, resolution, or length of RCM runs. » (Rummukainen, 2010). As a result, the data needed to perform a point value prediction for a quantity (e.g., the local vertical temperature gradient) may be unavailable (e.g., because a quantity was not stored during previous runs, the expense to re-run the model may not be justifiable, the cost to run the model for periods longer than a year may be too high, etc.). Additionally, the quantity may not be adequately captured by the RCM (an example of this case is discussed in Section 3.2). One way to address this issue is to use machine learning as a post-processing tool to downscale outputs to a point-specific location (Chantry et al., 2021).

The main objective of this work is thus to develop a neural network capable of predicting the temperature gradient in the Grenoble valley through the use of RCM data. An additional objective of this work is to use the neural network to make predictions on the presence of CAPs in the future and study the effect of climate change on CAPs in Grenoble.

Report Outline

— Section 2 introduces the machine learning concepts required to understand this work. This includes a brief overview of artificial neurons, convolutional neural networks, and recursive neural networks. In addition, the concepts of training, validation, and testing phases are presented along with a brief introduction to hyperparameters governing the behavior of neural networks.

— Section 3 details the data (i.e., the RCM outputs and measurements from weather stations) used in this work, as well as the strategies employed for processing the data.

— Section 4 describes experiments with neural network architectures carried out whilst developing the final neural network. In this section, the hyperparameters of the tested architectures are presented along with the training loss, training mean absolute percentage error (MAPE), validation loss, and validation MAPE. All experiments in this section were carried out using input data extracted from an RCM forced by renalysis, as detailed in Section 3.

— Section 5 details the application of the neural network on input data extracted from an RCM forced by an ESM that simulates past and future climate, as detailed in Section 3. In this section, the predictions of the network on data representative of the past and on data representative of the future are compared.

— Section 6 summarizes the conclusions of this work, and presents a short discussion on future research opportunities.

2 Introduction of Machine Learning Concepts

2.1 Overview

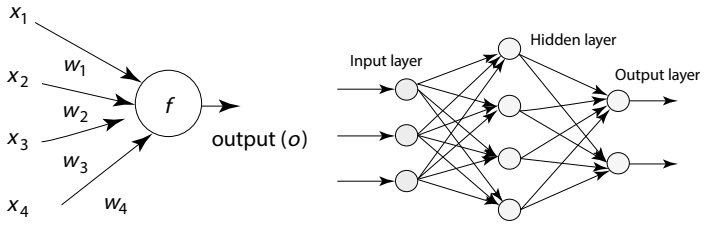
Artificial intelligence and machine learning describe a broad array of techniques. Insofar as relating to neural networks, artificial intelligence can be thought of as conceptually similar to an inverse problem (Kůrková, 2005). In simpler terms, neural networks determine, given a dataset and an answer set associated with the data, a set of "rules" used to make a prediction that is close to the known answer (i.e., the target value). Once the neural network has developed the set of rules (i.e., once the network is trained), a different dataset can be processed using the network and the resulting predictions can be used in a system. In the context of image classification, a neural network may be trained to differentiate between images of dogs and cats using a large set of labeled images. Once trained, the network may be used, e.g., in a phone application that barks when it determines there is a dog in-frame when using the camera. A brief introduction to the concepts required to understand the neural networks used in this work follows.

2.2 Artificial Neurons

Neural networks generally rely on a set of units referred to as artificial neurons, which are often adapted to weigh a set of inputs and to generate an output based on a chosen function, referred to as an activation function. In Figure 2a, a single neuron processes input $\vec{x} = \{x_1, x_2...\}$ to generate output o as follows:

$$o = f\left(\sum_{j=1}^{n}\left(x_j w_j\right) + b\right) \tag{1}$$

where $\vec{w} = \{w_1, w_2...\}$ is a weight vector, b is a bias, and f is an activation function, e.g., those found in Ramachandran et al. (2018).



(a) artificial neuron

(b) multilayered artificial neural network

Figure 2 – Architecture of an artificial neuron and a multilayered neural network. Adapted from Abraham (2005)

Neural networks typically employ a collection of neurons, referred to as a layer, that perceive the same input vector $\vec{x}$ and whose outputs are grouped together as another vector $\vec{o} = \{o_1, o_2...\}$. Generally, neurons in a layer all employ the same activation function f, but each neuron is associated with its own unique weight vector $\vec{w}$. In multilayered neural networks, such as that shown in Figure 2b, a series of layers are typically connected such that the output of one layer serves as the input of another, until a final output layer is reached. The layers in between the input layer and the output layer are referred as hidden layers.

During training of the neural network, the values associated with the output layer are compared to the known answers, and the difference between these values (as calculated via a loss function, e.g. mean square error or binary cross-entropy) is used to determine a new value of each of the weight vectors associated with each neuron (e.g., using the gradient descent method and the back-propagation algorithm).

2.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) provide an effective structure for processing 2D and 3D data (Alom et al., 2018). CNNs generally rely on kernels, which are small matrices that enhance a field in a desirable manner, and pooling layers, which reduce the dimensionality of the data processed by the kernels.

A simplified structure for a CNN is presented in Figure 3, which includes a (5x5) field as the input to the CNN, a single 2x2 kernel, and a 2x2 max-pooling layer. An expression for the field transformed by applying the kernel (i.e., the kernel output) $ko_{i,j}$, $i = \{1, 2, 3, 4\}$, $j = \{1, 2, 3, 4\}$ can be written in terms of the kernel k, the input x, and a chosen activation function f, as shown below.

$$ko_{i,j} = f\left(\sum_{m=1}^{2}\sum_{n=1}^{2} x_{m+i,n+j}\, k_{n,m} + b_{i,j}\right) \tag{2}$$

In the example shown in Figure 3 the kernel activation function f is the identity function and the bias b is set to zero.

In most applications, multiple kernels are be used to transform the field, and the collection of transformed fields are referred to as a feature map. Given the plurality of kernels, said feature maps may be large (i.e., occupy more space in memory) in comparison to the inputs. The pooling layer thus downsamples the kernel output, e.g. via max-pooling (i.e., selecting the maximum value in a predefined grid) or average-pooling (i.e., calculating the average of the values in a predefined grid). In the example given, the max-pooling layer reduces each of the four 2x2 sub-grids to the maximum value within each grid. Usually, the downsampled layer is then connected to fully connected artificial neuron layers similar to those described in Section 2.2. It is worth noting that the values defining the kernels in the convolution layer are learned parameters that are initialized prior to training (e.g., by setting them to a random value) and updated during training (i.e., they are to convolutional layers as weight vectors to artificial neurons).

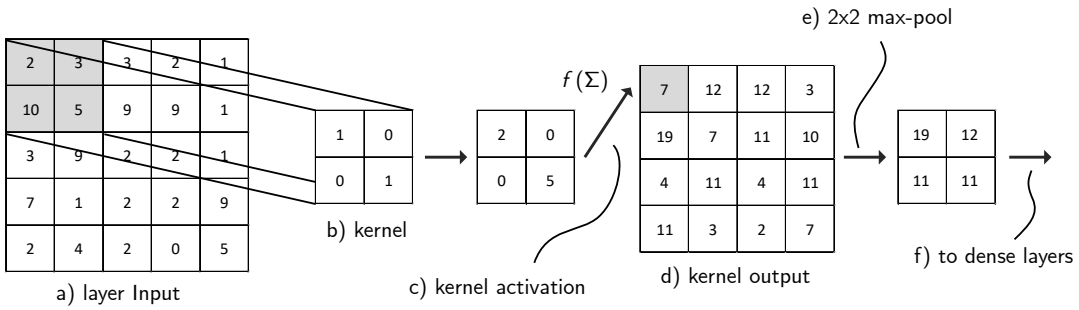


Figure 3 – Illustrative convolutional neural network structure.

2.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) introduce the notion of time to artificial neural networks (Lipton et al., 2015). Considering a time dependent input vector $x(t) = \{x_1(t), x_2(t)...\}$, a neuron can be represented by:

$$o = f\left(\sum_{j=1}^{n}\left(x_j(t)\, wc_j + f(t-1)\, wp_j\right) + b\right) \tag{3}$$

where wc is a weight vector associated with the input vector at the current time step and wp is a weight vector associated with the neuron's output at the previous time step. Using this structure, it is possible to iterate through any number of time steps to generate an output. Long short term memory (LSTM) networks are a class of recurrent neural networks that further include memory cells able to maintain information through several time steps (hence the term "memory"), allowing the RNN to capture longer-term temporal dependencies. More information on LSTM networks can be found in Greff et al. (2017).

2.5 Network Training, Validation, and Testing

Neural networks are generally prepared and evaluated in three phases: the training phase, the validation phase, and the testing phase. During the training phase, the parameters associated with the neural network (e.g., the weight vectors w in Equation (1) and the kernel $k_{n,m}$ values in Equation (2)) are updated so as to increase the quality of the prediction. As mentioned in Section 2.2, this is done by comparing the prediction made by the neural network to the target data using a cost function and an optimization method (e.g., stochastic gradient descent). However, there is a possibility that during this training the network learns spurious patterns in the training data (Eelbode et al., 2020). A different set of data, the validation data, is thus used to evaluate if the model is overfitting or underfitting the data. The performance of the network on the validation dataset is then used to tune the hyperparameters in the network (for more details on network hyperparameters, refer to Section 2.6). Once it is determined that the network is performing as desired on the training and validation datasets, the performance of the network on a third dataset, the testing dataset, serves as an unbiased benchmark of the performance of the network. It is important to highlight that the three datasets generally include similar data for the network to function properly (e.g., the network is trained, validated, and tested using sets of temperature fields with the same dimensions).

2.6 Hyperparameters

Neural networks can be described as a combination of two elements: 1) a model and associated training criterion, and 2) a method of optimizing the training criterion (Bengio, 2012). The characteristics that are used to describe these two elements are referred to as hyperparameters, and these hyperparameters govern the ability of the neural network to learn from a dataset. Furthermore, hyperparameters also determine the network's ability to generalize (i.e., apply what has been learned) from the training dataset to a second dataset (e.g., a validation dataset or a test dataset). Though the hyperparameters themselves can be learned programatically, as is the case in unsupervised learning, these are often set by practitioners that rely on experience and experimentation.

The hyperparameters used to describe the model include, e.g., the number of neurons in a hidden layer, the number of hidden layers, the size and number of kernels in convolution layers, and the number of time steps considered in the time series using RNNs.

The hyperparameters used to describe the optimization method include, e.g., the initial learning rate, the learning rate schedule, the batch size, and the type of optimization method (e.g., Adam, stochastic gradient descent, and Adamax) . Of these, the initial learning rate is perhaps the most important (Bengio, 2012) and serves as a starting point with which the gradient of the cost-function is scaled when updating the parameters (e.g., the weight vectors, kernel values, etc.) associated with the model. The learning rate schedule describes how the learning rate is modified as the neural network iterates through the training data (e.g., reduced at a constant rate, reduced after a predefined interval, decayed exponentially). The batch size refers to how many data points (i.e., combinations of input data and target data) are used to update the parameters in the model (e.g., by averaging the calculated gradient of the entire batch). The optimization method refers to the stochastic optimization strategy employed when searching for a solution to the model (i.e., a set of parameters that satisfactorily predict the target from the inputs), and includes the stochastic gradient descent (SGD) optimizer and the Adam optimizer as two examples within hundreds of available options (Schmidt et al., 2020). The optimizers may themselves include additional hyperparameters (e.g., β and ϵ in Adam) with effects that vary according to the optimizers in consideration.

2.7 Transfer Learning

In transfer learning, neural networks with parameters trained on one set of data are applied to another dataset. This provides certain advantages, including the possibility of training a neural network on large, labeled datasets and using the trained neural network layers to build a second neural network able to work on a comparatively smaller target dataset (Lu et al., 2015). This is inspired on the ability humans exhibit in applying knowledge and techniques learned in one domain to solve problems and perform tasks in other domains more quickly. One application of transfer learning is the use of a pre-trained model as a feature extractor (Shallu and Mehra, 2018), e.g., by freezing the weight vectors and kernel values obtained during training and use the resulting feature maps as inputs to trainable hidden layers.

3 Data

In this chapter, a description of the data used for developing the neural network is disclosed. This includes details on the periods considered and the meteorological data extracted from climate models and observations, as well as the processing and grouping required to develop the network. Notably, the data considered for use with the network was extracted for winter seasons defined as taking place between October 1st and April 30th. This decision was made taking into account that CAPs form mostly during this period, which is consistent with a study of CAPs in the Columbia River basin (McCaffrey et al., 2019).

3.1 Neural Network Input Data

In this work, the meteorological data considered for the neural network inputs (i.e., the data used to make the predictions) are calculated using the Modèle Atmospherique Regional (Gallée and Schayes, 1994) RCM, hereafter shortened to MAR, which was run so as to produce results on the region displayed in Figure 4. As mentioned in Section 1, RCMs are driven using boundary conditions taken from, e.g., an ESM. In this work the following two sources of information were used to drive MAR: ERA5, the fifth generation of European ReAnalysis (Hersbach et al., 2020) developed by the European Centre for Medium-Range Weather Forecasts (ECMWF), used to compare the performance of the neural network to observations in the past; and MPI-ESM1.2 (Gutjahr et al., 2019), henceforth abbreviated to MPI, an ESM developed by the Max Planck Institute for Meteorology that is of interest to this work as it can be used to study climate trends. Data from MAR was provided by the Climat-Cryosphère-Hydrosphère team at the Institut des Géosciences de l'Environnement (IGE).

There is a particular need to highlight that MAR forced by ERA5 predicts variables that are time correspondent with observations recorded in the field. However, when MAR is forced by MPI, the variable values are not correspondent with observations and thus for historical purposes a statistical approach must be used when making comparisons. Data for MAR forced by ERA5 was available for 1980 - 2017, while data for MAR forced by MPI was available for 1981 - 2100.

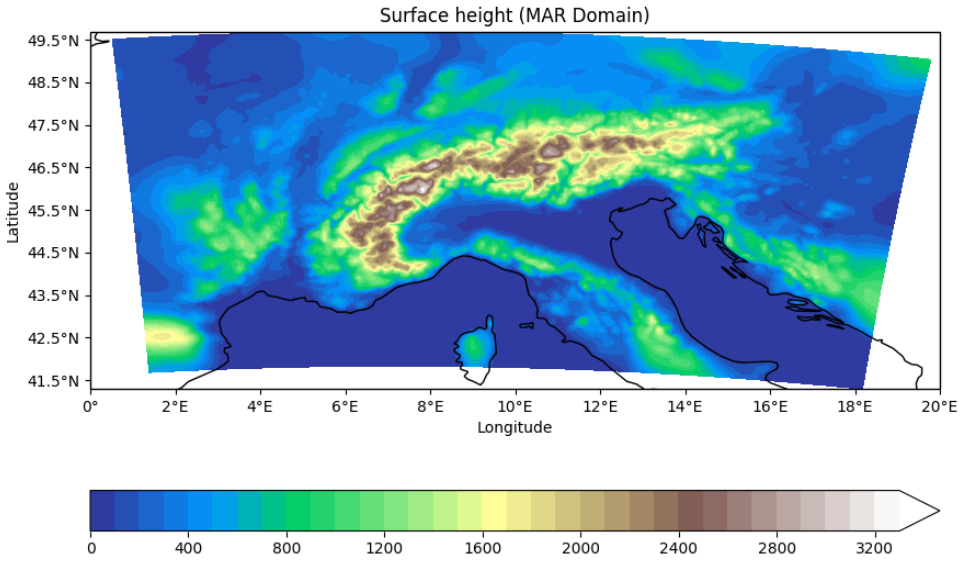


Figure 4 – Topographical map of the MAR RCM over the Alpine Region.

MAR was not run specifically for this project, and there was thus a limitation on the variables that were available for use as inputs. Another important limitation in using MAR was that the variables had a 1-day temporal resolution, compared to the typical hourly measurements recorded at weather stations. The available fields from MAR for use in the project are detailed in Table 1.

ESM	SP	U/V (500 hPa)	T (2/10/50/100 m)	T (500hPa)
ERA5	☒	☐	☒	☐
MPI	☒	☒	☒	☒

Table 1 – Data available from MAR discussed in this work
Surface Pressure SP (hPa), North-South Winds at 500 hPa U (m/s), East-West Winds at 500 hPa V (m/s)
Air Temperature T (K) at 2/10/50/100 m above the ground, Air Temperature at 500 hPa (K)

The sea level pressure (SLP) field was not directly available from MAR. It was determined that sea level pressure was a desirable field for consideration, as in the region under consideration the temporal variation of surface pressure (SP) is relatively small compared to the spatial variation caused by the surface height (SH) as exemplified in Figure 5. Experiments were run using the surface pressure and sea level pressure as inputs, and improvements were found when using sea level pressure over surface pressure. Sea level pressure was calculated using the following expression (Holton, 1973):

$$SLP = SP \cdot exp\left(SH \frac{g}{R\, T_{ref}}\right), \quad T_{ref} = 273\ K, \quad g = 9.81\ m\ s^{-2}, \quad R = 287\ J\ kg^{-1}\ K^{-1} \tag{4}$$

The time averaged sea level pressure field and surface pressure field are plotted in Figure 6. It is noted that the effect of the topography on the sea level pressure field is not completely eliminated using Equation (4), and though other methods for calculating the sea level pressure using the available data were explored, these were ultimately discarded.

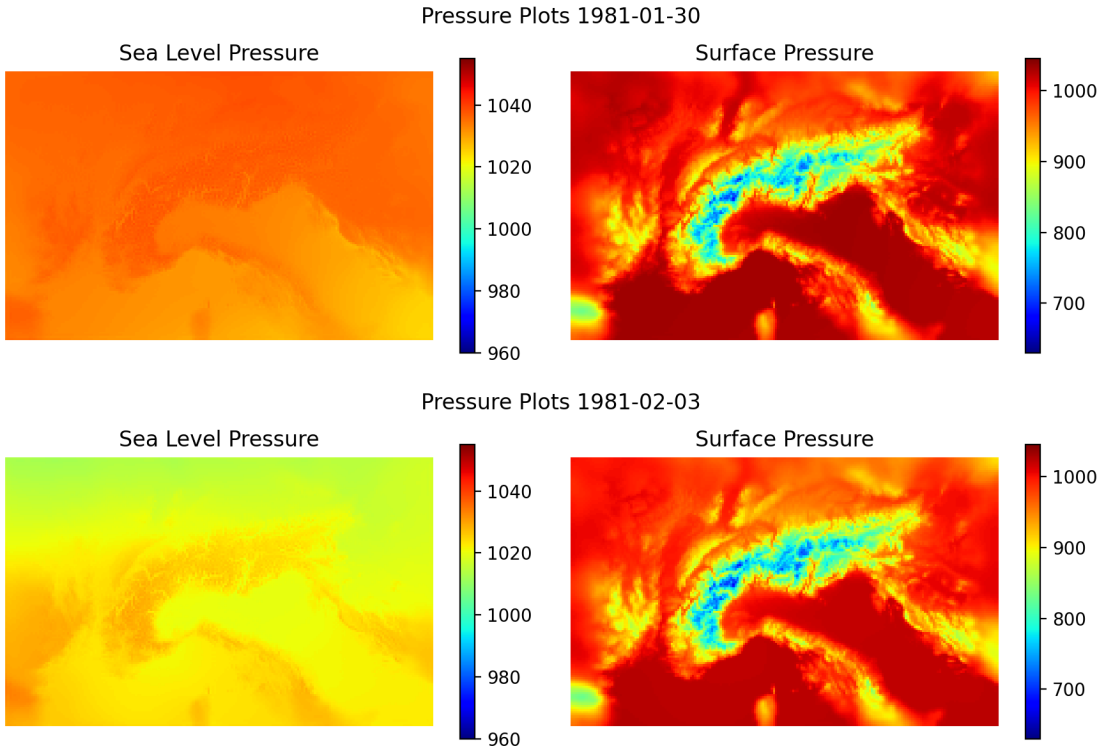


Figure 5 – Comparison of sea level pressure field and surface pressure field (hPa) for two days: January 30th, 1981 and February 3rd, 1981. Data from MAR forced by ERA5.

The winds at altitude (i.e., U and V at 500 hPa) were not available from MAR forced by ERA5, but could be interpolated onto the MAR region using the data from ERA5. Experiments run using the interpolated variable were conducted, but the performance of the neural networks on this data was poor and it was decided to discard the winds as an input to the neural network.

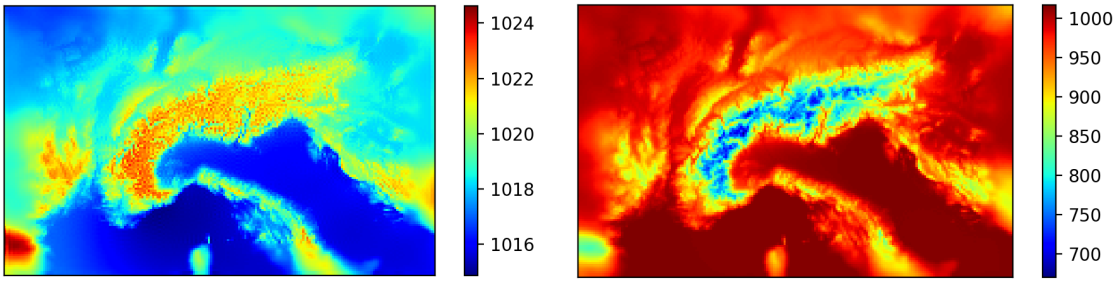


Figure 6 – Left: 30-year, time-averaged wintertime sea level pressure.
Right: 30-year, time-averaged wintertime surface pressure.

It was desired to consider the relative position of the data in the seasonal time series as an input in addition to the datasets extracted from MAR. This value, hereafter the winter depth, was calculated using the following expression:

$$winter\ depth = sin\left(2\pi\ \frac{day\ index}{winter\ length}\right) \tag{5}$$

3.2 Neural Network Target Data

Since an object of the current work is to predict the presence of CAPs in the Grenoble valley based on the bulk vertical temperature gradient surpassing a threshold (e.g., Largeron and Staquet (2016b) discloses a threshold equal to the mean temperature gradient during the 2006-2007 winter), an accurate measure of this quantity was needed to train the network. However, there was a significant limitation in finding long temperature records from which to calculate the vertical temperature gradient. A summary of the available records for this work is presented in Table 2. It is also noted that the availability indicated in Table 2 is not continuous in time (i.e., there are gaps in the data), and that discontinuities are most often observed in older records, as shown in Figure 7. Additionally, in contrast to the neural network input data, the target data was available as an hourly time series.

Station Name	Elevation, Z (m)	Elevation Category	Period Available
Pont de Claix	237	low	1985-01-01 to 2020-12-31
Peuil de Claix	935	mid	1985-01-01 to 2020-12-31
Col de Porte	1325	high	1993-08-01 to 2020-08-01
Le Versoud	220	low	1999-06-01 - 2020-12-21
St. Hilaire	1756	high	1985-01-01 to 2020-31-12

Table 2 – Available weather station data

As mentioned in Section 1, CAPs are associated with a strong temperature inversion. In the Grenoble valley, these inversions generally develop from the ground up to an altitude of approximately 1200m (Largeron and Staquet, 2016a), and thus this study relies on a bulk temperature gradient calculated between low altitude stations situated near the bottom of the valley (typically at around 230 m) and a high altitude station close to the top of the inversion layer. Thus, the temperature gradient that was used as the target was the daily average temperature gradient between a low station (Pont de Claix) and a high station (Col de Porte), calculated using the following expression:

$$\partial_z T = \frac{T_{high\ station} - T_{low\ station}}{Z_{high} - Z_{low}} \tag{6}$$

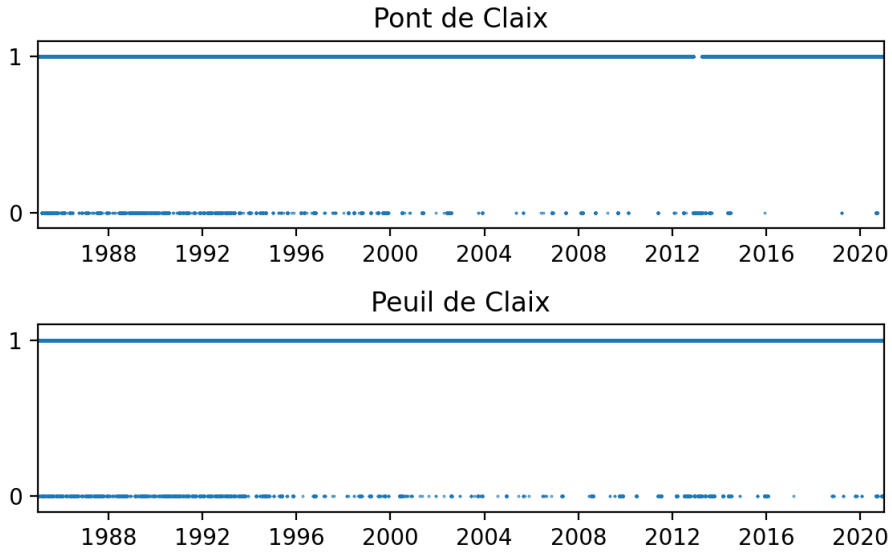


Figure 7 – Visualization of discontinuities in station data.
Each dot is an hourly measurement.
1 indicates measurement exists at that hour.
0 indicates measurement does not exist at that hour.

In view of the discontinuities in the data, re-analysis data from the SAFRAN model (Vernay et al., 2019) was also considered as a source of target data. The analyzed data included temperature values at 300 m, 600 m, 900 m, 1200 m, and 1500 m associated with the following three massifs: Chartreuse, Vercors, and Belledonne. It was however found that though the temperature time series was highly correlated with the in-situ data from Pont and Peuil de Claix (Figures 8, 9a and 9b), the vertical temperature gradient time series extracted from these was poorly correlated with those extracted from the observations (see Figure 9c). Thus, the possibility of using re-analysis data from SAFRAN as the target was discarded.

In addition to using the vertical temperature gradient as a target for the neural network, the use of the daily change in the gradient was explored as a possible target value. Though tested networks exhibited good skill at predicting this value, it was decided to discard it as a value due to the accumulation of error that would result from attempting to predict a time series using an imperfect prediction of daily change in the gradient and an initial condition. Another variable that was considered to be of interest as a target was the potential temperature, as it is a more straightforward indicator of atmospheric stability. However, since surface pressure observation data was not available for this project, the target could not be reliably calculated and was ultimately discarded.

3.3 Data Processing

The above-mentioned datasets were preprocessed in preparation for use in the network. Given that the neural network input data were available at a lower temporal resolution than the target data, the target data needed to be transformed to match the input data. To do this, the time series were forward interpolated, filling up to one missing value. The daily mean of the measurements were then calculated using a function that returned a "not a number" (NaN) value whenever one of the measurements for the day was missing.

Once all the datasets had the same temporal resolution, each dataset was normalized using the following expression:

$$\tilde{x} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{7}$$

where x_{min} and x_{max} were extracted from the data for the longest time period up to 2018. These periods were, e.g., 1980-2017 for the SLP fields from MAR forced by ERA5, and 1993-2018 for the observed

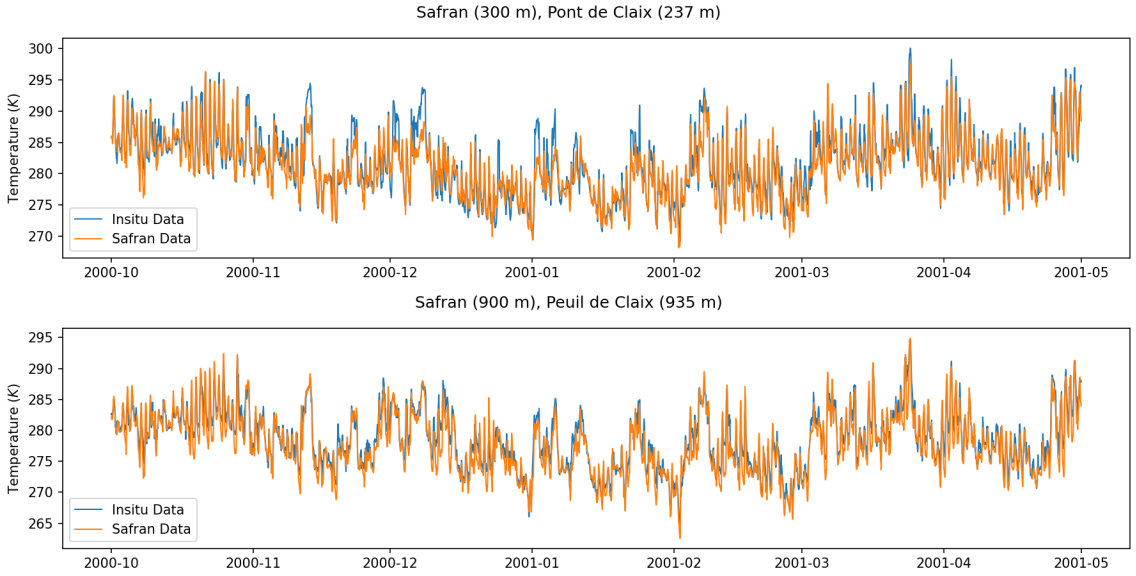


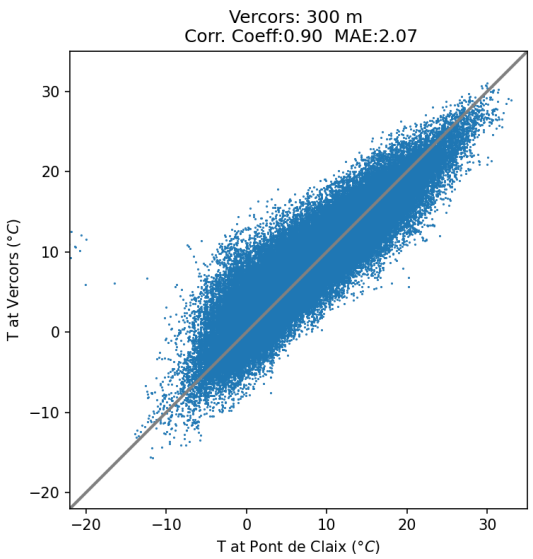
Figure 8 – Temperature time series comparison between
Observations (blue) and SAFRAN (orange).
Top: Temperature time series at low altitude (300 m).
Bottom: Temperature time series at mid altitude (900 m).

vertical temperature gradient between Col de Porte and Pont de Claix. Thus, for values beyond this period the normalized data could include values outside of the [0,1] range. Additionally, the pressure fields extracted from MAR forced by MPI were normalized using SLP_{min} and SLP_{max} extracted from MAR forced by ERA5 in the 1993-2017, and since the minimum and maximum values of MAR forced by MPI do not correspond to those of MAR forced by ERA5, the data includes values outside the [0,1] range. Additional pre-processing was required when relying on transfer learning, the details of which are disclosed in Section 4.2.3

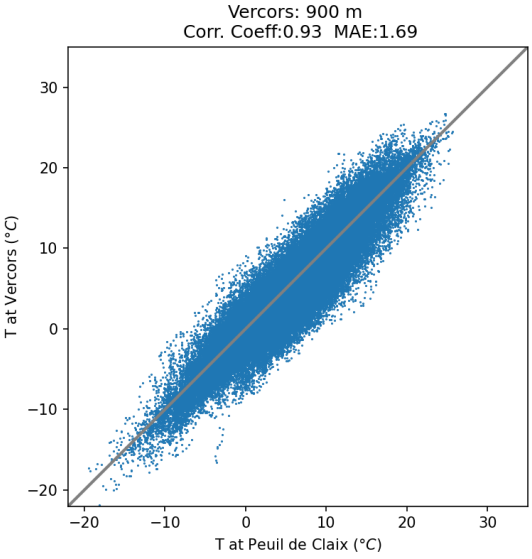
3.4 Data Grouping

As mentioned in Section 2.5, neural networks rely on a training, validation, and test dataset to produce a reliable model. Logically, each of these datasets must include a set of input data and target data. Furthermore, for the neural network to develop an accurate predictive capacity, there must be a meaningful dependency between the two sets of data. Thus, in view of the non-correspondence in time between observation data and data from MAR forced by MPI, the training, validation, and test datasets required the use of inputs extracted from MAR forced by ERA5.

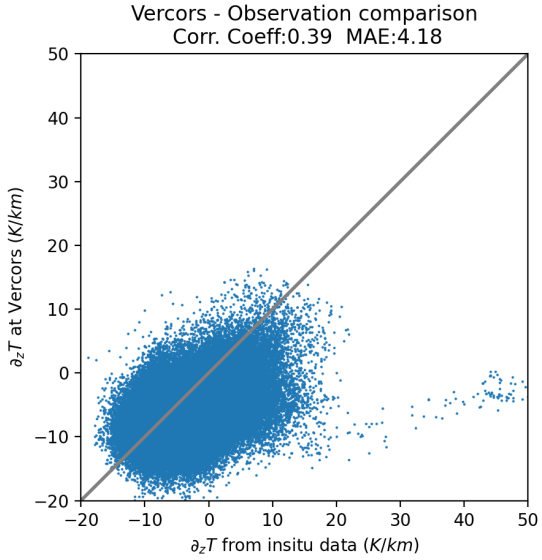
As stated by Eelbode et al. (2020), careful consideration must be taken to select independent data when separating the data to be included in the training, validation, and test sets. It was thus considered that each individual winter season (i.e., the period from Oct-1st of the given year to Apr-30th of the following year) formed a data group independent of other winter seasons, and thereby the training, validation, and training datasets were formed as a set of winters. The testing dataset was selected from the most recent years in the available data. This selection considered the difficulty machine learning systems face with problems of extrapolation (Chantry et al., 2021) and the desire within this work to analyze future climate, reasoning that these seasons would be the most impacted by climate change so far. Once the years for the testing dataset were selected, the remaining available years were randomly selected into the validation dataset until the validation dataset comprised at least 20% of the years available for training and validation.



(a) Low altitude temperature comparison



(b) Mid altitude temperature comparison



(c) Temperature gradient comparison

Figure 9 – SAFRAN (Vercors Massif) and observation comparison plots.
Observations taken from Pont de Claix and Col de Porte.
Pearson product-moment correlation coefficient listed and
Mean Absolute Error (MAE) listed above the plots.
(a) and (b) show Temperature values at low and mid altitudes, respectively
(c) shows the vertical temperature gradient, calculated using Equation (6)

4 Neural Network Design and Analysis

In this chapter, the design of the neural network that is the focus of this work is discussed. The neural network was developed using TensorFlow (Abadi et al., 2015) and Keras (Chollet et al., 2015), and thus relied on the tools made available in this platform whenever possible.

4.1 Optimization Hyperparameters

The experimentation on neural networks in this work rely on exponential decay learning rate schedule. Per the TensorFlow documentation, the learning rate η at each epoch (i.e., at each run through the training data) is calculated using the following expression:

$$\eta = \eta_0 * \lambda^{n/m} \tag{8}$$

where λ is a chosen decay rate, n is the epoch, and m is the number of epochs after which the learning rate should have decayed by a factor of λ.

The optimizer used for training the neural networks discussed in this work is the $\text{Adam}_{\alpha,\beta_1,\beta_2,\epsilon}$ optimizer. Per the TensorFlow documentation, α is the learning rate to be used in the optimizer (and thus depends on the value calculated with the learning rate scheduler), β_1 and β_2 refer to decay rates in moment estimates within the optimizer (typically using a value of 0.9 and 0.999, respectively), and ϵ is a constant used for numerical stability. Though ϵ defaults to a value of 10^{-7}, the documentation notes that this may not be a good general default value. After some experimentation, a value of $\epsilon = 10^{-2}$ was selected.

4.2 Model Hyperparameters

In this section, experiments on the model hyperparameters are presented. For these experiments, the input data comprised the sea level pressure field extracted from MAR forced by ERA5 and the winter depth. As the winter depth is a single value, this was generally used as the input to a neuron in the densely connected layers implemented after the convolution layers. Layer activation functions were set to the default defined by TensorFlow when possible (e.g., in LSTM layers), and to ReLU otherwise.

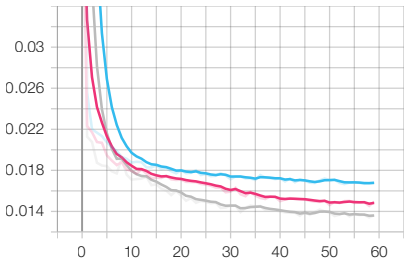
4.2.1 Model depth and width

In this section, experiments exploring the effect of the depth of the model (i.e., the number of hidden layers) and the width of the model (e.g., the number of neurons in densely connected layers and the number of kernels in convolutional layers) are presented. The experiments were carried out using the hyperparameters detailed in Table 3 and run using the neural network architectures presented in Table 4. The resulting learning curves are shown in Figures 10 to 13. In these figures, the mean square error (MSE) and the mean absolute percentage error (MAPE) curves on the training and validation datasets are presented.

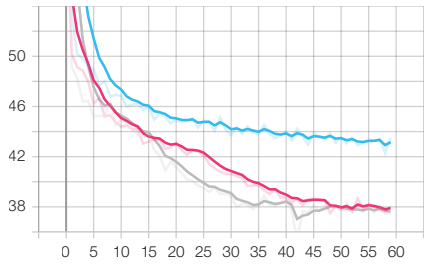
Figure 10 is used as an example to illustrate the interpretation of these figures. In Figures 10a and 10b, the learning curves associated with 3 training runs are presented. For networks that exhibit learning, the errors should decrease as the number of epochs increases due to the adjustment of network parameters to better predict the target. Figures 10c and 10d show the errors on the validation dataset for each training epoch. When comparing the performance of the networks, MAPE is used as the quality metric. The reasoning is that even a relatively low MSE (e.g., the green curve in Figure 10c is lower than the orange curve at all points) can be associated with a higher MAPE (e.g., the green curve in Figure 10d shows a higher MAPE than the orange curve). It is thus observed from Figures 10 to 13 that the best performance, as measured by the lowest achieved MAPE on the validation dataset, is obtained when using Architecture D in Table 4.

4.2.2 LSTM time depth

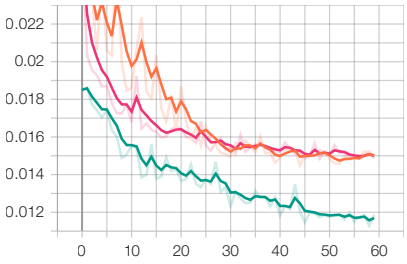
In this section, experiments exploring the effect of the length of the time series used in the LSTM layer of the model (i.e., the number days considered) are presented. The experiments were carried out using the



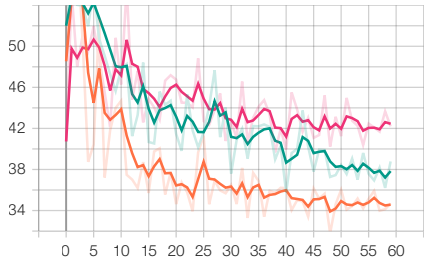
(a) Training Loss
(Mean Square Error)
Calculated on normalized Values



(b) Training Metric
(Mean Absolute Percentage Error)

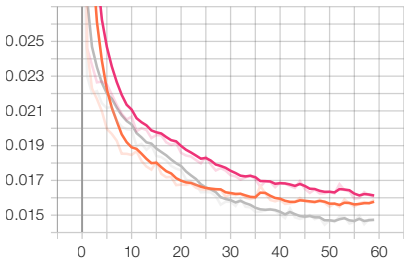


(c) Validation Loss
(Mean Square Error)
Calculated on normalized Values

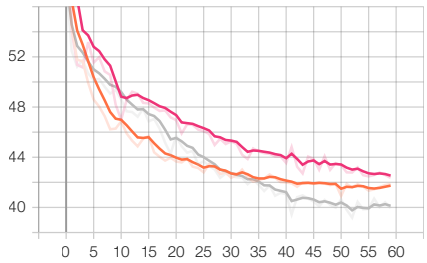


(d) Validation Metrics
(Mean Absolute Percentage Error)

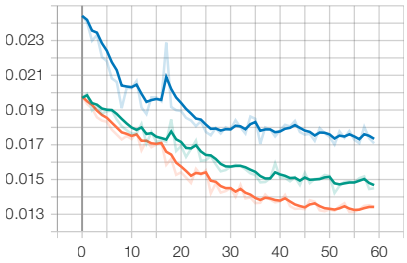
Figure 10 – Architecture A (Shallow, Narrow)
Loss and Metric vs # of Epochs
Opaque: smoothed curves; Translucent: raw data
Colors indicate the same run when comparing losses and metrics.



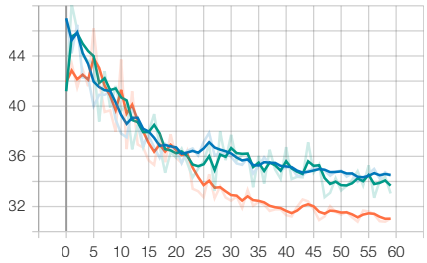
(a) Training Loss
(Mean Square Error)
Calculated on normalized Values



(b) Training Metric
(Mean Absolute Percentage Error)

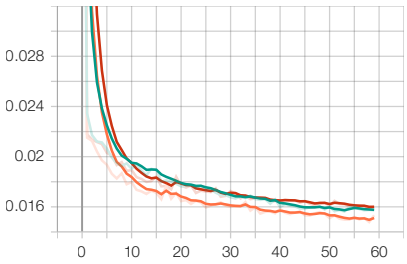


(c) Validation Loss
(Mean Square Error)
Calculated on normalized Values

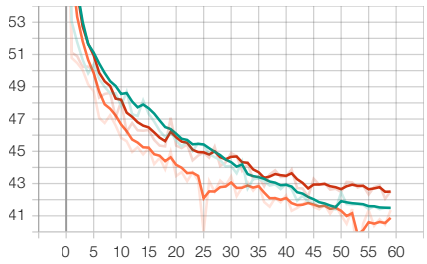


(d) Validation Metrics
(Mean Absolute Percentage Error)

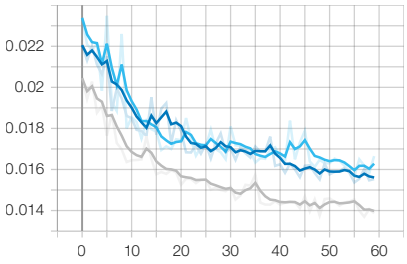
Figure 11 – Architecture B (Deep, Narrow)
Loss and Metric vs # of Epochs
Opaque: smoothed curves; Translucent: raw data
Colors indicate the same run when comparing losses and metrics.



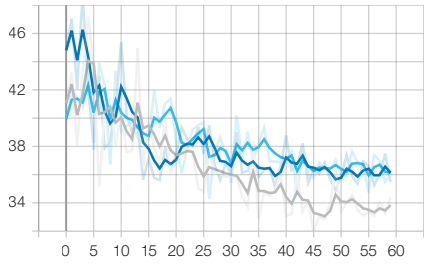
(a) Training Loss
(Mean Square Error)
Calculated on normalized Values



(b) Training Metric
(Mean Absolute Percentage Error)

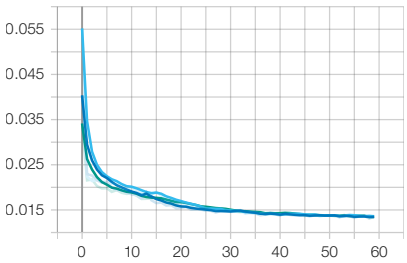


(c) Validation Loss
(Mean Square Error)
Calculated on normalized Values

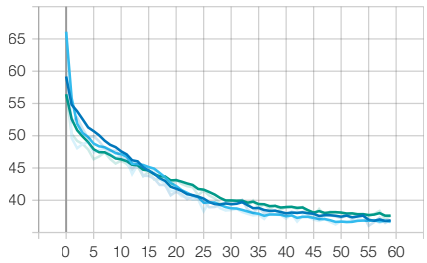


(d) Validation Metrics
(Mean Absolute Percentage Error)

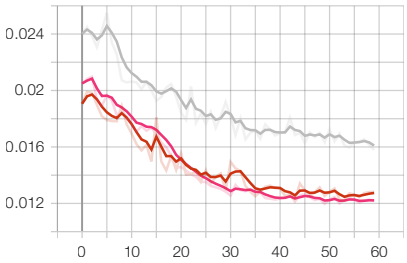
Figure 12 – Architecture C (Shallow, Wide)
Loss and Metric vs # of Epochs
Opaque: smoothed curves; Translucent: raw data
Colors indicate the same run when comparing losses and metrics.



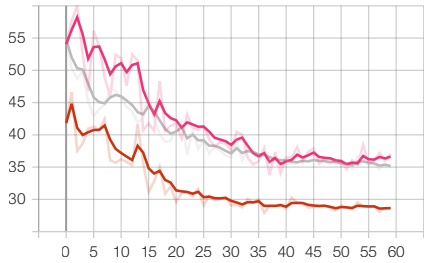
(a) Training Loss
(Mean Square Error)
Calculated on normalized Values



(b) Training Metric
(Mean Absolute Percentage Error)



(c) Validation Loss
(Mean Square Error)
Calculated on normalized Values



(d) Validation Metrics
(Mean Absolute Percentage Error)

Figure 13 – Architecture D (Deep, Wide)
Loss and Metric vs # of Epochs
Opaque: smoothed curves; Translucent: raw data
Colors indicate the same run when comparing losses and metrics.

hyperparameters detailed in Table 3 and run using the neural network architectures presented in Table 5. The resulting learning curves are shown in Figures 14 to 16. From these, it is observed that the best performance, as measured by the lowest mean absolute percentage error on the validation dataset, is achieved when using Architecture F in Table 4. Additionally, the separation between the performance for each run indicates that the performance can be further improved by tweaking hyperparameters.

4.2.3 Transfer Learning with VGG

It was hypothesized that improvements over the networks described in Sections 4.2.1 and 4.2.2 would be achieved by training an LSTM model that used a pre-trained feature extractor instead of training a convolutional architecture section like the one in Table 5. To this end, experiments were run using VGG19 (Simonyan and Zisserman, 2015) as a feature extractor, using the hyperparameters listed in Table 6 architecture detailed in Table 7. As previously mentioned, additional preprocessing is required when relying on transfer learning, as the pretrained portion of the model was trained to extract features from data exhibiting specific characteristics (e.g., number of channels in an image, data range, and shape). The SLP fields were thus scaled to values in the range [0,255] to represent the values in an image with 2-bit color channels, and were duplicated to form grayscale images that TensorFlow's built-in preprocessor for VGG19 were able to process.

The neural network was trained in three phases: a main training phase and secondary training phase during which the VGG model parameters were frozen, and a fine-tuning phase during which the parameters were further adjusted through training. The learning curves for this experiment are shown in Figure 17, where it is observed that the performance of the network is significantly improved compared to the experiments detailed in Sections 4.2.1 and 4.2.2.

4.3 Final Model & Performance Analysis

Since the model with the best performance was obtained when using transfer learning, this model was selected for further analysis. Using the hyperparameters in Table 6 and the architecture in Table 7, the model was trained on the combination of the training and validation datasets. The resulting neural network was then used with the test dataset to analyze its performance, comparing the time series, quantiles, correlation, and probability density function (Figures 18 to 20) between the observations from weather stations (detailed in Section 3.1) and the predictions from the neural network.

It was found that the neural network is able to satisfactorily predict the observation time series using information in the pressure field time series extracted from MAR. However, it is noted that the quantile-quantile plot shows that the network exhibits a consistent tendency to underpredict the target in the test period. Furthermore, it was observed that the predictions were more highly correlated with the

Hyperparameter	Value
optimizer	Adam
η_0	$5 \cdot 10^{-4}$
λ	0.1
m	40
training + validation seasons	1993-2012
validation ratio	≥ 0.2
test dataset* seasons	2013 - 2017
batch size	64

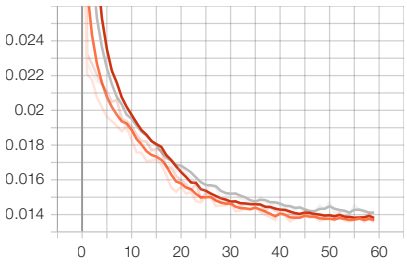
Table 3 – Hyperparameters for depth, width, and LSTM experiments.
*Test seasons not analyzed during experiment(s).

Neural Network Architectures			
A shallow, narrow	B deep, narrow	C shallow, wide	D deep, wide
Sea Level Pressure Input [126 x 201]			
Convolutional_0 (3x3), 64	Convolutional_0 (3x3), 64	Convolutional_0 (3x3), 128	Convolutional_0 (3x3), 128
Max-Pooling_0 (2x2)	Max-Pooling_0 (2x2)	Max-Pooling_0 (2x2)	Max-Pooling_0 (2x2)
Convolutional_1 (3x3), 16	Convolutional_1 (3x3), 16	Convolutional_1 (3x3), 64	Convolutional_1 (3x3), 64
Max-Pooling_1 (2x2)	Max-Pooling_1 (2x2)	Max-Pooling_1 (2x2)	Max-Pooling_1 (2x2)
	Convolutional_2 (3x3), 16		Convolutional_2 (3x3), 64
	Max-Pooling_2 (2x2)		Max-Pooling_2 (2x2)
	Convolutional_3 (3x3), 8		Convolutional_3 (3x3), 32
	Max-Pooling_3 (2x2)		Max-Pooling_3 (2x2)
	Convolutional_4 (3x3), 8		Convolutional_4 (3x3), 32
	Max-Pooling_4 (2x2)		Max-Pooling_4 (2x2)
Concatenation with: Winter Depth Input [1]			
10% Dropout			
Dense, 64			
Output [1]			

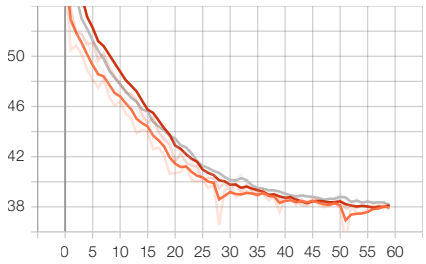
Table 4 – Neural Network Architectures for Experiments on Model Depth and Width.
Windows size and number of units shown as (#_of_rows, #_of_cols), #_of_units.
Input / Output dimensions denoted with square brackets.

Neural Network Architectures		
E Two Day	F Four Day	G Six Day
Sea Level Pressure Input		
[2 x 126 x 201]	[4 x 126 x 201]	[6 x 126 x 201]
Time distributed Convolutional Section		
Convolutional_0 (3x3), 128		
Max-Pooling_0 (2x2)		
Convolutional_1 (3x3), 64		
Max-Pooling_1 (2x2)		
Convolutional_2 (3x3), 64		
Max-Pooling_2 (2x2)		
Convolutional_3 (3x3), 32		
Max-Pooling_3 (2x2)		
Convolutional_4 (3x3), 32		
Max-Pooling_4 (2x2)		
LSTM, 64	LSTM, 64	LSTM, 128
Concatenation with: Winter Depth Input [1]		
10% Dropout		
Dense, 64		
Output [1]		

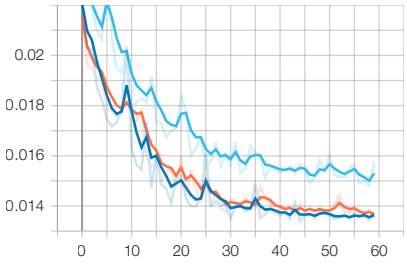
Table 5 – Neural Network Architectures for Experiments on length of LSTM time series.
Windows size and number of units shown as (#_of_rows, #_of_cols), #_of_units.
Input / output dimensions denoted with square brackets.



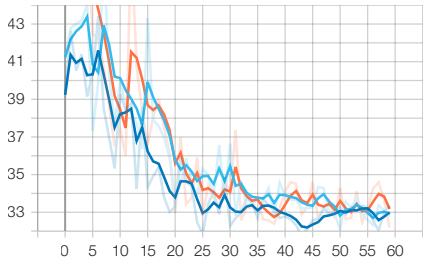
(a) Training Loss
(Mean Square Error)
Calculated on normalized Values



(b) Training Metric
(Mean Absolute Percentage Error)

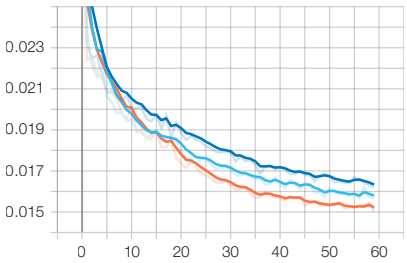


(c) Validation Loss
(Mean Square Error)
Calculated on normalized Values

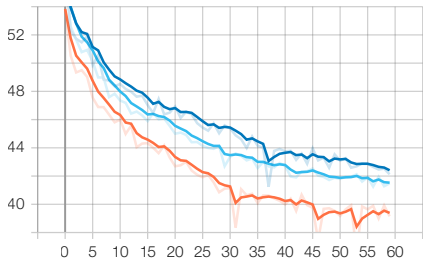


(d) Validation Metrics
(Mean Absolute Percentage Error)

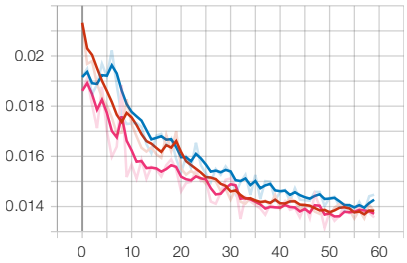
Figure 14 – Architecture E (2 Day Input)
Loss and Metric vs # of Epochs
Opaque: smoothed curves; Translucent: raw data
Colors indicate the same run when comparing losses and metrics.



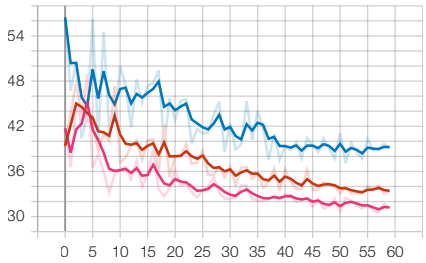
(a) Training Loss
(Mean Square Error)
Calculated on normalized Values



(b) Training Metric
(Mean Absolute Percentage Error)



(c) Validation Loss
(Mean Square Error)
Calculated on normalized Values



(d) Validation Metrics
(Mean Absolute Percentage Error)

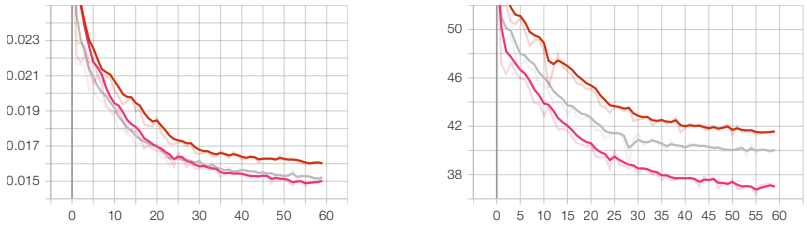
Figure 15 – Architecture F (4 Day Input)
Loss and Metric vs # of Epochs
Opaque: smoothed curves; Translucent: raw data
Colors indicate the same run when comparing losses and metrics.

Global Hyperparameters	
batch size	64
training + validation seasons	1993-2012
validation ratio	≥ 0.2
test seasons*	2013 - 2017
Main Training Phase	
Hyperparameter	Value
optimizer	Adam
# of epochs	400
learning schedule	exponential decay
η_0	$2 \cdot 10^{-4}$
λ	0.05
m	400
Secondary Training Phase	
Hyperparameter	Value
optimizer	SGD with Momentum
momentum	0.9
# of epochs	30
learning schedule	constant
η	$1 \cdot 10^{-5}$
Tuning Phase	
optimizer	Adam
# of epochs	50
learning schedule	constant
η	$4 \cdot 10^{-6}$

Table 6 – Hyperparameters for Transfer Learning Experiment.
*Test seasons not analyzed during experiment(s).

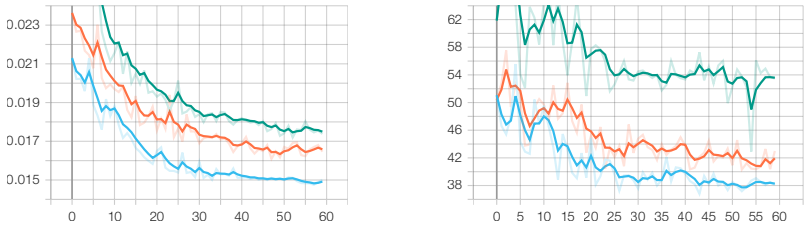
Neural Network Architecture
Sea Level Pressure Input [4 x 126 x 201]
Time distributed VGG19
Convolutional_0 (3x3), 64
Convolutional_1 (3x3), 64
Max-Pooling_0 (2x2), Stride: 2
Convolutional_2 (3x3), 128
Convolutional_3 (3x3), 128
Max-Pooling_1 (2x2), Stride: 2
Convolutional_4 (3x3), 256
Convolutional_5 (3x3), 256
Convolutional_6 (3x3), 256
Convolutional_7 (3x3), 256
Max-Pooling_2 (2x2), Stride: 2
Convolutional_8 (3x3), 512
Convolutional_9 (3x3), 512
Convolutional_10 (3x3), 512
Convolutional_11 (3x3), 512
Max-Pooling_3 (2x2), Stride: 2
Convolutional_12 (3x3), 512
Convolutional_13 (3x3), 512
Convolutional_14 (3x3), 512
Convolutional_15 (3x3), 512
Max-Pooling_4 (2x2), Stride: 2
20% 2D Dropout
LSTM, 1024
Concatenation with: Winter Depth Input [1]
10% Dropout + Dense, 512
10% Dropout + Dense, 256
10% Dropout + Dense, 256
Output [1]

Table 7 – Neural Network Architecture for Transfer Learning Experiment.
Windows size and number of units shown as (#_of_rows, #_of_cols), #_of_units.
Input / Output dimensions denoted with square brackets.



(a) Training Loss
(Mean Square Error)
Calculated on normalized Values

(b) Training Metric
(Mean Absolute Percentage Error)

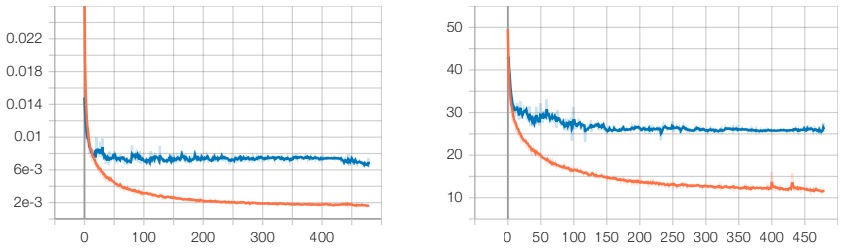


(c) Validation Loss
(Mean Square Error)
Calculated on normalized Values

(d) Validation Metrics
(Mean Absolute Percentage Error)

Figure 16 – Architecture G (6 Day Input)
Loss and Metric vs # of Epochs
Opaque: smoothed curves; Translucent: raw data
Colors indicate the same run when comparing losses and metrics.

observation data than the temperature gradient calculated using SAFRAN (Figure 9c), with Pearson correlation coefficients of 0.89 and 0.39, respectively. Unfortunately, the lack of high altitude temperature data from MAR forced by ERA5 (Table 1) did not allow the comparison between the predictions from the neural network and MAR forced by ERA5.



(a) Loss (Mean Square Error)
Calculated on normalized Values

(b) Metric (Mean Absolute Percentage Error)

Figure 17 – Transfer Learning Experiment
Loss and Metric vs # of Epochs
Training in Orange, Validation in Blue
Opaque: smoothed curves; Translucent: raw data

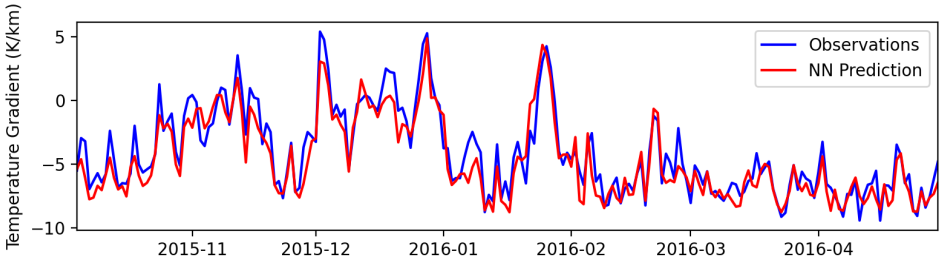


Figure 18 – Sample time series comparison between neural network prediction and observations.
Network input data from MAR forced by ERA5, Winter 2015.

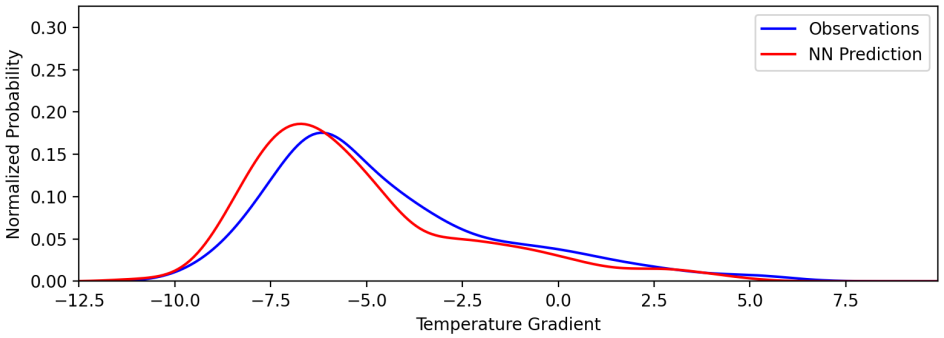


Figure 19 – Normalized probability density function.
Network input data from MAR forced by ERA5, 2013-2017.

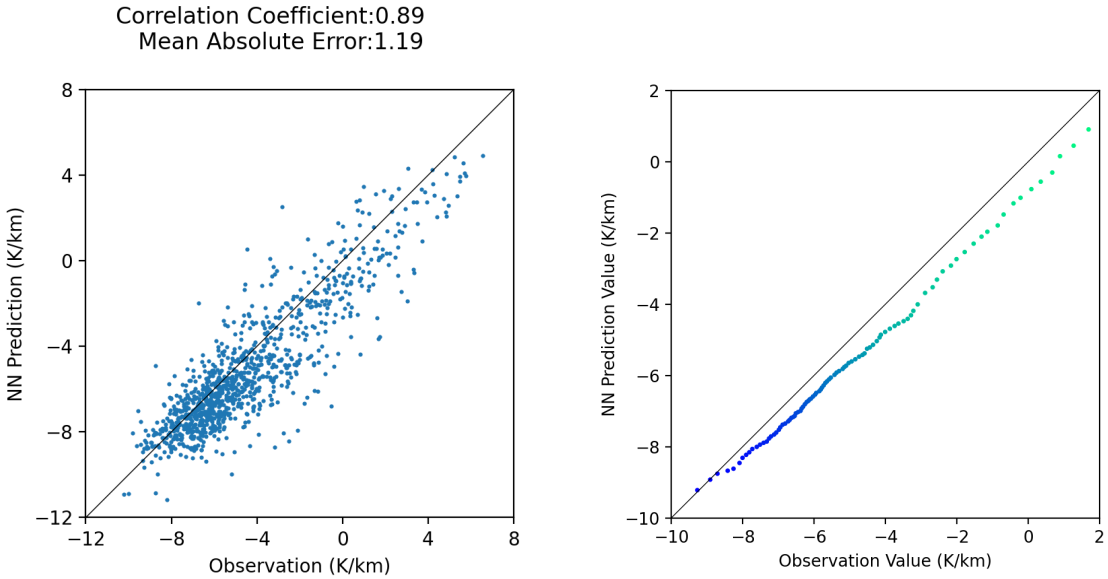


Figure 20 – Left: correlation plot, listing Pearson product-moment
correlation coefficient and mean absolute error above the plot.
Right: quantile-quantile plot (1^{st}-99^{th} quantile).
Blue lightens with increasing quantile.
Network input data from MAR forced by ERA5, 2013-2017.

5 Use of Neural Network with MAR forced by MPI

In this section, the neural network developed in Section 4 is applied to the sea level pressure fields extracted from MAR forced by MPI detailed in Section 3.1. The objective was to obtain predictions in a past period (detailed in Section 5.2) in order to compare the behavior of the neural network with data extracted from MAR forced by MPI with MAR forced by ERA5. After this evaluation, the network was applied to data from climate predictions extracted from MAR forced by MPI in an attempt to study CAP formation in the future.

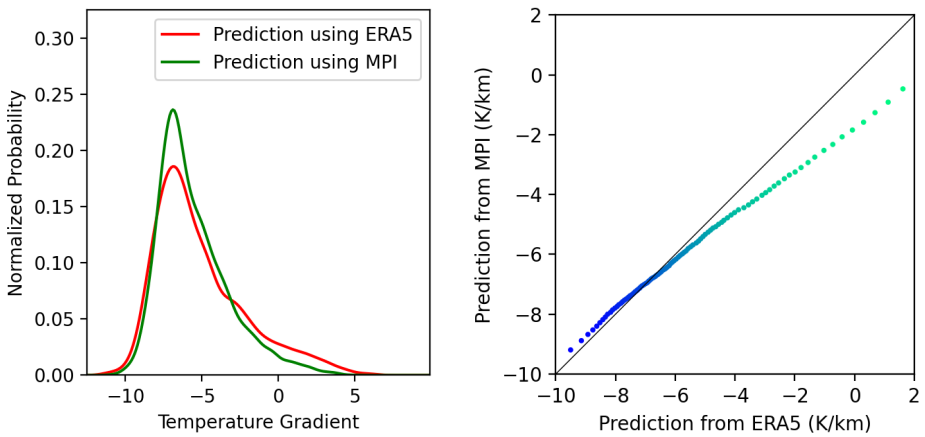
5.1 MAR forced by MPI - Past

Recalling that there is no time correspondence between MAR forced by MPI and observations, it is thus necessary to use statistical analyses when using the neural network developed in Section 4. In view of the satisfactory performance of the neural network on the test period, comparisons were made between the predictions of the neural network using pressure fields extracted from MAR forced by 1) ERA5 and 2) MPI. In an attempt to address multidecadal climate variability, 30 year periods were considered when making statistical comparisons.

Figure 21 shows the plots of the probability density functions (Figure 21a) and the quantile-quantile plot (Figure 21b) for the predictions using the two input datasets. Table 8 further details the median, percentiles, and standard deviations of the predicted values. The results show a disparity in the predictions obtained from the two data sources, e.g., there is about a 2 K/km difference in the 95$^{\text{th}}$ percentile. This divergence is reflected in the quantile-quantile plot as a divergence in quantiles above the median. However, the similarity of the median and 5$^{\text{th}}$ percentiles lead to the hypothesis that comparisons between the predictions based on past data and future data could provide insight into the behavior of the atmosphere in the future.

MAR Forcing	Median	$P_{5\%}$	$P_{95\%}$	σ
ERA5	−5.90	−8.63	1.12	2.97
MPI	−6.09	−8.40	−0.91	2.32

Table 8 – Statistical measures of neural network predictions from ERA5 forced MAR and MPI forced MAR. i-th Percentile $P_{i\%}$, Standard Deviation σ. Quantities in Kelvins per km.



(a) (b)

Figure 21 – Left: probability density function.
Right: quantile-quantile plot (1$^{\text{st}}$-99$^{\text{th}}$ quantile).
Blue lightens with increasing quantile.
Network input data from MAR forced by ERA5 and MAR forced by MPI, 1988-2017.

5.2 MAR forced by MPI - Future

Attention is now turned to the analysis of the predictions made using the neural network on future climate data as modeled with MPI. Though the comparisons in Section 5.1 imply that the network underestimates values at the higher end of the distribution, comparing the predictions based on past data with the predictions based on future data can provide insight into the behavior of the atmosphere. The near-future winter seasons, from 2015 - 2046, are considered to those during which there is the greatest confidence in the predictions. The reasoning is that this period, centered around 2030 and bounding roughly 30 years, provides insight into the future without the input data deviating too far from the data the network was trained on. The 1984 - 2014 period was selected as the past period as it is the set of winter seasons immediately prior to the near-future period.

Figure 22 shows the probability density function plots for the predictions and the quantile-quantile plot for the predictions in the past and near-future. Table 9 further details the median, percentiles, and standard deviations of the predicted values. From these, it is determined that though there are small, observable trends in the values, the predictions for the past and near future are very similar. Additionally, further work is needed to verify the statistical significance of the trends.

Period	Median	$P_{5\%}$	$P_{95\%}$	σ
1984 - 2014	−6.10	−8.38	−0.81	2.33
2015 - 2045	−6.09	−8.47	−0.83	2.38
2046 - 2076	−6.09	−8.34	−0.41	2.43
2069 - 2099	−5.87	−8.38	−0.30	2.51

Table 9 – Statistical measures of neural network predictions from MAR forced by MPI
i-th Percentile $P_{i\%}$, Standard Deviation σ. Quantities in Kelvins per km.

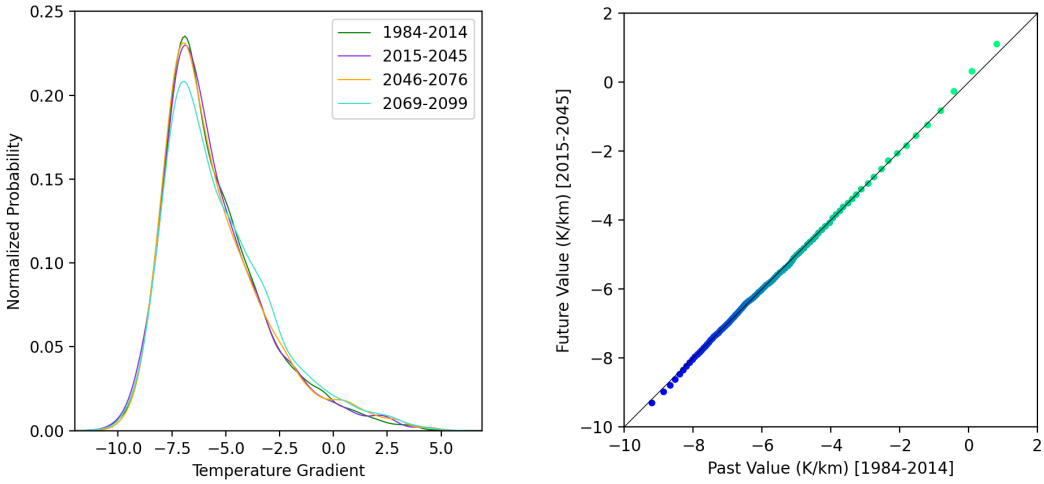


Figure 22 – Left: Normalized probability density function, Neural Network Predictions from MAR forced by MPI.
Right: quantile-quantile plot for neural network predictions in the past [1984-2014] and near future [2015-2045] (1^{st}-99^{th} quantile).
Blue lightens with increasing quantile.

6 Conclusion & Future Works

6.1 Conclusions

In the present work, a deep learning approach to predicting the daily average vertical temperature gradient in the Grenoble valley was developed with the intent of using the trained neural network to analyze trends in the future. Both the convolutional neural network approach and the convolutional LSTM approach were explored, varying hyperparameters associated with these in search of better performance. It was determined that a deep, wide approach with a 4-day input length based on VGG19 exhibited the best predictive ability amongst the models studied.

An exploratory discussion on input data sources and target data sources was presented, and the model was found to exhibit good skill at predicting the daily average temperature gradient in the Grenoble valley using sea level pressure data extracted from MAR forced by ERA5.

The predictions of the model were additionally shown to be a significant improvement on the temperature gradient series simulated using SAFRAN reanalysis. This may be considered further validation of the value in applying deep learning methods to data from climate models.

The neural network was additionally applied to data extracted from MAR forced by MPI. When using this data as input, the network was found to statistically underpredict vertical temperature gradients when compared to those predicted using MAR forced by ERA5. This was most notable in percentiles above the median of the distributions, which are the type of values associated with CAPs.

Finally, the network was applied to data from MAR forced by MPI in both a past period and a future period in order to study the behavior of the temperature gradient in the future. However, it was found that the predictions in the past and near future were statistically quite similar. It was additionally noted that further work is needed to determine the statistical significance of the trends in the predictions.

6.2 Future Work

Future research on the relationship between the learning capacity of the network and the fields used as inputs could extend the understanding of the predictions deep learning models are able to make from particular variables. Of particular interest, research directed to seeking ways of increasing the prediction quality using other variables that can be predicted with higher confidence (e.g., using the daily change in vertical temperature gradient, which exhibited lower error but could not be relied upon due to chaotic behavior stemming from the accumulation of error with the progression of time). It is also hypothesized that the geopotential height field, which was unfortunately unavailable for this project, could result in high quality predictions. Another subject that could constitute the object of future studies is investigation into the advantages and disadvantages of using LSTM models with phenomena that exhibit chaotic behavior.

With regards to analyses, it is noted that there is a need for a benchmark against which to compare the predictions of the network using inputs from MAR forced by MPI to the predictions obtained using inputs from MAR forced by ERA5. With a benchmark that allows an estimate of the quality of the predictions from the neural network, further studies into the trends regarding formation of CAPs in the Grenoble valley and thereby of the associated wintertime pollution episodes would be possible. Additionally, if the neural network predictions exhibit a high enough equality, it would then be possible extract predictions of persistent wintertime inversions from the time series.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abraham, A. (2005). Artificial neural networks. *Handbook of measuring system design*.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Esesn, B. C. V., Awwal, A. A. S., and Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches.

Bengio, Y. (2012). *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg.

Block, M. L. and Calderón-Garcidueñas, L. (2009). Air pollution: mechanisms of neuroinflammation and cns disease. *Trends in Neurosciences*, 32(9):506–516.

Chantry, M., Christensen, H., Dueben, P., and Palmer, T. (2021). Opportunities and challenges for machine learning in weather and climate modelling: hard, medium and soft ai.

Chemel, C., Chaxel, E., and Chollet, J. (2007). On the role of the grenoble valley topography in vertical transport of mass and pollutants. In *International conference on Alpine meteorology (ICAM), Chembery, France*.

Chollet, F. et al. (2015). Keras. https://keras.io.

Daiber, A., Kuntic, M., Hahad, O., Delogu, L. G., Rohrbach, S., Di Lisa, F., Schulz, R., and Münzel, T. (2020). Effects of air pollution particles (ultrafine and fine particulate matter) on mitochondrial function and oxidative stress – implications for cardiovascular and neurodegenerative diseases. *Archives of Biochemistry and Biophysics*, 696:108662.

Eelbode, T., Sinonquel, P., Maes, F., and Bisschops, R. (2020). Pitfalls in training and validation of deep learning systems. *Best Practice & Research Clinical Gastroenterology*, page 101712.

Freire, C., Ramos, R., Puertas, R., Lopez-Espinosa, M.-J., Julvez, J., Aguilera, I., Cruz, F., Fernandez, M.-F., Sunyer, J., and Olea, N. (2010). Association of traffic-related air pollution with cognitive development in children. *Journal of Epidemiology & Community Health*, 64(3):223–228.

Gallée, H. and Schayes, G. (1994). Development of a three-dimensional meso-γ primitive equation model: Katabatic winds simulation in the area of terra nova bay, antarctica. *Monthly Weather Review*, 122(4):671 – 685.

Garrido-Perez, J. M., Ordóñez, C., García-Herrera, R., and Barriopedro, D. (2018). Air stagnation in europe: Spatiotemporal variability and impact on air quality. *Science of The Total Environment*, 645:1238–1252.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

Gulia, S., Shiva Nagendra, S., Khare, M., and Khanna, I. (2015). Urban air quality management-a review. *Atmospheric Pollution Research*, 6(2):286–304.

Gutjahr, O., Putrasahan, D., Lohmann, K., Jungclaus, J. H., von Storch, J.-S., Brüggemann, N., Haak, H., and Stössel, A. (2019). Max planck institute earth system model (mpi-esm1.2) for the high-resolution model intercomparison project (highresmip). *Geoscientific Model Development*, 12(7):3241–3281.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., De Chiara, G., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J.-N. (2020). The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049.

Holton, J. R. (1973). An introduction to dynamic meteorology. *American Journal of Physics*, 41(5):752–754.

Jorcano, A., Lubczyńska, M. J., Pierotti, L., Altug, H., Ballester, F., Cesaroni, G., El Marroun, H., Fernández-Somoano, A., Freire, C., Hanke, W., Hoek, G., Ibarluzea, J., Iñiguez, C., Jansen, P. W., Lepeule, J., Markevych, I., Polańska, K., Porta, D., Schikowski, T., Slama, R., Standl, M., Tardon, A., Vrijkotte, T. G., von Berg, A., Tiemeier, H., Sunyer, J., and Guxens, M. (2019). Prenatal and postnatal exposure to air pollution and emotional and aggressive symptoms in children from 8 european birth cohorts. *Environment International*, 131:104927.

Kůrková, V. (2005). Neural network learning as an inverse problem. *Logic Journal of the IGPL*, 13(5):551–559.

Lareau, N. P., Crosman, E., Whiteman, C. D., Horel, J. D., Hoch, S. W., Brown, W. O. J., and Horst, T. W. (2013). The persistent cold-air pool study. *Bulletin of the American Meteorological Society*, 94(1):51 – 63.

Largeron, Y. and Staquet, C. (2016a). The atmospheric boundary layer during wintertime persistent inversions in the grenoble valleys. *Frontiers in Earth Science*, 4:70.

Largeron, Y. and Staquet, C. (2016b). Persistent inversion dynamics and wintertime pm10 air pollution in alpine valleys. *Atmospheric Environment*, 135:92–108.

Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning.

Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S., and Zhang, G. (2015). Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, 80:14–23. 25th anniversary of Knowledge-Based Systems.

McCaffrey, K., Wilczak, J. M., Bianco, L., Grimit, E., Sharp, J., Banta, R., Friedrich, K., Fernando, H. J. S., Krishnamurthy, R., Leo, L. S., and Muradyan, P. (2019). Identification and characterization of persistent cold pool events from temperature and wind profilers in the columbia river basin. *Journal of Applied Meteorology and Climatology*, 58(12):2533 – 2551.

Ramachandran, P., Zoph, B., and Le, Q. (2018). Searching for activation functions.

Ringler, T. D., Jacobsen, D., Gunzburger, M., Ju, L., Duda, M., and Skamarock, W. (2011). Exploring a multiresolution modeling approach within the shallow-water equations. *Monthly Weather Review*, 139(11):3348 – 3368.

Rummukainen, M. (2010). State-of-the-art with regional climate models. *Wiley Interdisciplinary Reviews: Climate Change*, 1(1):82–96.

Schmidt, R. M., Schneider, F., and Hennig, P. (2020). Descending through a crowded valley–benchmarking deep learning optimizers. *arXiv preprint arXiv:2007.01547*.

Shallu and Mehra, R. (2018). Breast cancer histology images classification: Training from scratch or transfer learning? *ICT Express*, 4(4):247–254.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Suglia, S. F., Gryparis, A., Wright, R. O., Schwartz, J., and Wright, R. J. (2008). Association of black carbon with cognition among children in a prospective birth cohort study. *American journal of epidemiology*, 167(3):280–286.

Vernay, M., Lafaysse, M., Hagenmuller, P., Nheili, R., Verfaillie, D., and Morin, S. (2019). The S2M meteorological and snow cover reanalysis in the french mountainous areas (1958 - present).

Watson-Parris, D. (2021). Machine learning for weather and climate are worlds apart. *Philosophical Transactions of the Royal Society A*, 379(2194):20200098.