Reduction of Temperature Forecast Errors with Deep Neural Networks

Reducering av temperaturprognosfel med djupa neuronnettverk

Robin Isaksson
Reduction of Temperature Forecast Errors with Deep Neural Networks

Reducering av temperaturprognosfel med djupa neuronätverk

Robin Isaksson
The work for this thesis was carried out in cooperation with SMHI.

ISSN 1650-6553
Copyright © Robin Isaksson
Published at Department of Earth Sciences, Uppsala University (www.geo.uu.se), Uppsala, 2018
Abstract

Reduction of Temperature Forecast Errors with Deep Neural Networks
Robin Isaksson

Deep artificial neural networks is a type of machine learning which can be used to find and utilize patterns in data. One of their many applications is as a method for regression analysis. In this thesis, deep artificial neural networks were implemented in the application of estimating the error of surface temperature forecasts as produced by a numerical weather prediction model. An ability to estimate the error of forecasts is synonymous with the ability to reduce forecast errors as the estimated error can be offset from the actual forecast.

Six years of forecast data from the period 2010–2015 produced by the European Centre for Medium-Range Weather Forecasts’ (ECMWF) numerical weather prediction model together with data from fourteen meteorological observational stations were used to train and evaluate error-predicting deep neural networks. The neural networks were able to reduce the forecast errors for all the locations that were tested to a varying extent. The largest reduction in error was by 83.0% of the original error or a 16.7 °C decrease in the mean-square error.

The performance of the neural networks’ error reduction ability was compared with that of a contemporary Kalman filter as implemented by the Swedish Meteorological and Hydrological Institute (SMHI). It was shown that the neural network implementation had superior performance for six out of seven of the evaluated stations where the Kalman filter had marginally better performance at one station.

Keywords: Numerical weather prediction, deep neural networks, machine learning, systematic forecast errors

Degree Project E in Meteorology, 1ME422, 30 credits
Supervisor: Emelie Karlsson
Department of Earth Sciences, Uppsala University, Villavägen 16, SE-752 36 Uppsala (www.geo.uu.se)

ISSN 1650-6553, Examensarbete vid Institutionen för geovetenskaper, No 425, 2018

The whole document is available at www.diva-portal.org
Populärvetenskaplig sammanfattning

Reducering av temperaturprognosfel med djupa neuronätverk
Robin Isaksson

Numeriska vädermodeller används idag i stor utsträckning för att göra meteorologiska prognoser. När atmosfären modelleras så finns det många utmaningar som gör att fel kan uppstå i prognoserna. Eftersom väderprognoser utgör en så stor nytta i samhället så är det intressant att undersöka metoder som kan användas till att förbättra prognoser. En sådan metod är att använda så kallade neuronätverk för att försöka förutsäga vilket fel en prognos har när den görs och sedan justera det.

Neuronätverk är en teknik som används för att finna djupare samband i data. Neuronätverk kan bli tränade att finna önskade samband genom att förse neuronätverken med exempel på vad de önskas producera utifrån en viss typ av indata. Vädermodeller producerar väldigt mycket information som neuronätverken kan försöka hitta användbara samband i.


Detta positiva resultat innebär att maskinlärning är en användbar metod för att minska prognosfel.

Nyckelord: numerisk väderprognos, djupa neuronätverk, maskinlärning, systematiska prognosfel

Examensarbete E i meteorologi, 1ME422, 30 hp
Handledare: Emelie Karlsson
Institutionen för geovetenskaper, Uppsala universitet, Villavägen 16, 752 36 Uppsala (www.geo.uu.se)

ISSN 1650-6553, Examensarbete vid Institutionen för geovetenskaper, Nr 425, 2018

Hela publikationen finns tillgänglig på www.diva-portal.org
# Table of Contents

1. **Introduction** ................................................................. 1

2. **Background** ................................................................. 2
   2.1 Numerical Weather Prediction ........................................... 2
   2.2 Systematic Forecast Errors ................................................. 3
   2.3 Artificial Neural Networks ............................................... 3
      2.3.1 The Principles of Artificial Neural Networks ...................... 3
      2.3.2 The Multilayer-perceptron .......................................... 5
      2.3.3 Training Neural Networks ......................................... 5
      2.3.4 Regularization ...................................................... 6
      2.3.5 Convolutional Neural Network ..................................... 7
      2.3.6 Deep Convolutional Network Structure ......................... 8
   2.4 Kalman Filter ............................................................. 9

3. **Method** ........................................................................ 10
   3.1 Input Data ................................................................. 11
   3.2 Observation Stations Used .............................................. 12
   3.3 Evaluation of Results .................................................... 13

4. **Results** ........................................................................ 14
   4.1 Single Station ............................................................. 14
      4.1.1 Comparison with Kalman Filter ................................... 16
   4.2 Multiple Stations ......................................................... 17

5. **Discussion** ................................................................. 18
   5.1 Performance ............................................................... 18
   5.2 Neural Network Design .................................................. 19

6. **Conclusions** ............................................................... 21

7. **Acknowledgements** ...................................................... 22

8. **References** ............................................................... 23
1 Introduction

Weather forecasts produced by numerical weather prediction (NWP) models have many societal applications. They assist people in everyday life and businesses in making informed decisions. The usefulness of the weather forecasts naturally depend on how well they perform. NWP models are faced with challenges in producing the best possible forecast. This is in part from the weather being a complex and stochastic system and in part from the limitations of numerical modeling architectures. Because of the challenging nature of weather prediction the forecasts will have associated prediction errors.

The usefulness of weather prediction makes it a worthwhile effort to reduce the errors as much as possible. The most direct way would be to improve upon the models making the actual forecasts. This is an on-going effort limited by resources and technology. A different way to improve the forecasts is by post-processing the results, recognizing some of the problems the models have and correcting results after the model has run.

When it comes to the weather some locations are more interesting to the public than others. Cities, airports or wind energy farms are examples of such locations. Kalman filters and model-output statistics are examples of methods which reduce errors for specific locations. They use previously calculated errors to estimate future errors and then mitigate them. A novel alternative is to use deep artificial neural networks. They have been shown to perform well in image processing tasks such as image recognition (He et al. 2016). Such task have similarities to finding error-descriptions in the output of forecast models.

A neural network post-processing method can be particularly useful as it will adjust the forecast to be as similar as possible to actual measurements. Forecasts can in other words be tailored for specific locations. For example factories with temperature-dependent operations can receive forecasts with maximized utility as the forecast will emulate what a factory thermometer measures even though the thermometer readings could have issues from a meteorological standpoint but not from an operational standpoint.

Earlier research investigated a similar post-processing method using smaller networks for a different forecast model with a coarser spatial resolution. The study concluded successful error-reductions of surface temperatures (Marzban 2002).

In this thesis work deep artificial neural networks are explored in the application of reducing systematic forecast errors for the 2-metre temperature. This is achieved by using neural networks to find relations between the data produced by a NWP model and the error the forecast has for certain locations covered by meteorological stations. With a prediction of the error from the forecast the results can be corrected.
2 Background

2.1 Numerical Weather Prediction

Atmospheric forecasting models use dynamical and thermodynamical equations to predict the state and motions of the atmosphere. To put the dynamical equations to practical use in a computer model there are a lot of considerations that needs to be accounted for, which is the subject of NWP.

Given an initial state of the atmosphere the dynamical equations describe the temporal evolution of that state. An atmospheric model requires both an initial state of the atmosphere at all the points which are simulated as well as boundary conditions. Reliable data used to initialize NWP models is important for good performance.

To perform actual computations using the dynamical equations numerical schemes have to be used. This means that all variables used in computations have to be discretized, all quantities will be subject to a finite resolution.

Mainly two methods are used in NWP for representing continuous three-dimensional fields numerically. The more straightforward method is to construct a three-dimensional grid of points where each grid is an approximation of the state of the atmosphere at that location. The alternative method is to represent the fields as sums of continuous functions, for example using Fourier series. These methods are used in grid-point models or spectral models. Each method has its merits.

Both the grid-point model and spectral model have a finite spatial resolution which is limited either by the largest distance between grid-points or by the largest wave number used in the spectral representations. This means that neither method can ever give a true representation of the atmosphere. Values for e.g. temperature are not analogous to point-measurements but instead represent averages taken over volumes dependent on resolution. Important to note is that the limited resolution effectively smooths fields such as topography. Small features could disappear or change considerably.

A method to account for physics on a scale smaller than the model can resolve is to use parameterization. The unresolved process is then approximated using other available quantities. A particular case which illustrate the need of parameterization is turbulence. Turbulent eddies exist within the boundary layer with sizes as small as millimeters (Arya 1988). Running regional operational NWP models which can resolve small features like microscale turbulent eddies is computationally unfeasible. There are many more processes which required parameterization in a NWP model.

All these details mean that the actual environments and situations that the NWP model simulates can be quite different from their model-representation. The impact this has will depend on both the weather and the location.
2.2 Systematic Forecast Errors

NWP models are faced with challenges in producing forecasts with as small errors as possible. It is helpful to divide the errors into two components when discussing the forecast errors. The components are the systematic errors and the random errors.

Systematic errors stem from shortcomings in the models in some way, for example from inadequate representation of physical processes by dynamical equations, shortcomings in physical schemes and parameterization, numerical limitations or from errors in input data from observational systems or land use data. For example numerical models have finite resolution which will impact the forecasts for locations with steep orography as those locations will have a problematic representation by the model. This is one cause of systematic errors (Allen et al. 2006).

Random errors are more elusive and have a probabilistic nature. Lorenz (1963) showed that minute changes in initial conditions for chaotic systems such as weather systems lead to large differences at a later time. Any of a multitude of random error sources will accumulate perturbations which will affect the forecast (Slingo et al. 2011).

2.3 Artificial Neural Networks

In this section a background on how neural networks work is provided. It is by no means comprehensive as artificial neural networks is such a wide subject with many nuances in the implementations.

Artificial Neural Networks (NN) can be described in all manners of romantic and novel ways which derive from their likeness to the human brain. These perspectives are surely useful but in the context of their application in this thesis it would be most straightforward to view NNs as a class of regressional tools. In concept NNs are not that different from least-square regression. They can be used to find relationships between sets of data where the error in the description is sought to be minimized.

NNs have some remarkable properties which however distinguishes them from classical regressional analysis methods. They are non-linear which allows them greater versatility in the description of more complex relationships which contain non-linearity. Their application is very general meaning that they can be applied to a large set of problems. Some examples which illustrate their versatility are their use in image recognition, speech recognition or financial forecasting (Graves et al. 2013; Björklund et al. 2017).

2.3.1 The Principles of Artificial Neural Networks

As previously mentioned NNs are inspired by the human brain in the sense that neurons are used as the fundamental building blocks. Neurons are cells capable of transmitting and receiving signals of varying intensity. Figure 1 shows a NN neuron in a simple context: the neuron having an input signal \( z \) and an activation function \( f(z) \) which produces its output signal \( z' \).
Figure 1. A simple description of a neuron. It receives an input $z$ and produces an output $z'$.

The neuron can receive its input from multiple sources. The input is a weighted sum of the output signals from connected neurons (figure 2). Normally a neuron bias $b$ is included in the neuron model such that the input to the activation function is shifted positively or negatively. Equation 1 shows the relationship inputs, weights and biases have. The inputs $z_i$ are summed according to the weights $w_i$, then a bias $b$ is applied and the activation function $f(z)$ is evaluated to produce the neuron output $z'$.

$$f\left(\sum_i w_i z_i + b\right) = z'$$

The bias can be interpreted as a neuron’s propensity to output certain signals.

Figure 2. Neurons typically have many inputs $z_i$ which stem from other neurons. The total input is a weighted sum of the inputs where the weights $w_i$ depend on which neuron the signal comes from.

The weights and biases in a NN determine the output of any neuron in the network. The process of determining suitable values for these parameters is called training.

The neural activation function $f(z)$ in equation 1 where ($z$ is $\sum_i w_i z_i + b$) can take many forms, a few examples of activation functions follow.

- **Linear**
  $$f(z) = z$$

- **Sigmoid- or logistic**
  $$f(z) = \frac{1}{1 + e^{-z}}$$

- **Rectified Linear Unit (ReLU)**
  $$f(z) = max(0, z)$$

ReLU has been shown to work especially well as an activation function as the training process is more efficient (Krizhevsky et al. 2012). Linear activation are only used in certain situations.
such as in a final output stage which maps to a set of real numbers. A ReLU could for example only provide positive real values in such a case.

The remarkable expressivity of neural networks comes from the many different way neurons can be connected. Networks can be constructed in such a way that information only travels in one direction which is called a feed-forward network or they can be connected such that information can pass through cycles and utilizes a delay in the signal propagation. Such a cyclic network is called a recurrent network. Only feed-forward networks are considered in this thesis and as such any following information concern only such networks.

Figure 3 shows an example of a small feed-forward neural network.

![Feed-forward Network Diagram](image)

**Figure 3.** A feed-forward network. Information travels in only one direction. This network structure is called a multilayer perceptron.

### 2.3.2 The Multilayer-perceptron

Figure 3 shows a NN which is called a multilayer perceptron, the figure shows the characteristic layers a multilayer perceptron has. It has an input layer (shown the four left-most neurons), a hidden layer (shown the two middle neurons) and an output layer (shown the three right-most neurons). A multilayer perceptron has neurons which connect to all neurons in the following layers. Such a neuron structure is sometimes called a dense layer or a fully connected layer.

The universal approximation theorem states that feed-forward multilayer perceptron networks with a single hidden layer can approximate any continuous function on a compact subset of \( \mathbb{R}^n \) given enough neurons. There are some mild restrictions on the activation function. Non-constant, bounded and monotonically increasing continuous functions have been shown to fulfill the criterion (Hornik 1991) as well as unbounded non-polynomial functions such as the popular ReLU (Sonoda et al. 2017). This theorem serves as a good background as to why neural networks can achieve high performance in regressional tasks.

### 2.3.3 Training Neural Networks

Training is the process of iteratively optimizing the neuron weights and biases. A metric is used to quantify and to evaluate the performance of a NN state. This metric is called a loss-function,
cost-function or error-function. The metric can for example be the mean-squared error (MSE)

\[ \text{MSE} = \frac{1}{N} \sum_{i} (y_i - \hat{y}_i)^2 \]  

(2)

where \( y_i \) are observed values, \( \hat{y}_i \) are estimations and \( N \) the number of samples.

When training with a mean-squared error loss-function the optimization algorithms adjust the weights and biases to minimize the mean squared error of the output.

The algorithm used to minimize the loss function is called back-propagation. There are many implementations of the back-propagation algorithm and they can be configured in different ways. For example a parameter called the learning-rate can be set which dictate how much the optimization algorithm modify the weights and biases at each iterative step. Some implementations of the back-propagation algorithm are adaptive in that they change parameters such as the learning-rate while training.

To train and evaluate a NN two datasets are required. A training dataset is used to train the NN and a validation dataset is used to evaluate the NN. A separate validation dataset is used because NNs can become overfitted. If the NN has enough parameters it can actually encode dataset specific information when trained. This means that the network will produce good results for the training data but when new data is tested the performance drops. Such a NN is overfitted. A data set separate from the training data used only for validation gives an insight of how well a network gives a general description which would work for new or unused data.

When training a NN the set of training data is supplied together with a set of the desirable outcomes that the network is sought to reproduce. Training algorithms typically load a subset of this data called a batch at each iteration where the NN parameters are updated. The batch data would optimally be randomly sampled without replacement from the complete training set (Bengio 2012). After the complete training set has been exhausted a so called epoch has elapsed in the training process. After an epoch is completed it is helpful to collect statistics such as how the NN performs with the validation data set. It is beneficial to train for many epochs which means that the complete training dataset is reused. A reasonable criterion for when to stop training is when the loss evaluated for the validation data set has stopped increasing.

2.3.4 Regularization

There are multiple methods of preventing overfitting of NNs. A simple but proven efficient method is called dropout (Srivastava et al. 2014). Dropout is the act of deactivating neurons during training randomly. The deactivated neurons receive no input and transmit no output. Dropout prevents overfitting as the neurons are encouraged to utilize multiple inputs instead of possibly being dependent on few or single neurons output.

A networks with dropout can be viewed as a combination multiple networks where each unique combination of dropped-out neurons is a new network albeit sharing weights with the other possible networks. If each \( n \) neurons has a \( p \) chance of dropping out then there are \( 2^n \)
possible networks. The training can then be viewed as training the \(2^n\) thinned networks where each network having virtually zero-percent chance of being trained. After the training when the network is being used or validated the neurons weights are multiplied with \(p\). This is an approximation of averaging the output from all \(2^n\) possible networks.

### 2.3.5 Convolutional Neural Network

When dealing with spatially correlated data such as in images or the output fields from a NWP forecast a NN architecture called a convolutional neural network (CNN) is especially useful. The previously discussed architectures had layers of neurons where each neuron in a previous layer is connected to all the neurons in the following layer. This is a powerful structure as there are a vast amount of different paths the signals are allowed to propagate and be affected by different values of weights and biases. It does not however always make the most sense to use this type of structure. When the input is spatially correlated it would be a logical conclusion that accounting for the data having a spatial structure in some way could improve upon a NN model.

A neuron in a CNN only receives input from a spatially localized region. You could say that each neuron only ”sees” a part of the input at a time. If the input for instance is a temperature field produced by a NWP forecast, instead of an input neuron receiving all the temperature-values at once (as in a multilayer perceptron) it would receive the temperatures in chunks of localized geographical regions.

The CNNs discussed in the scope of this thesis are designed to process two-dimensional inputs, that is they are two-dimensional CNN layers. The inputs to a CNN layer are structured as multiple two-dimensional images. An image can for example be a temperature field produced by a NWP model. Multiple such images are combined along a third dimension called the depth when passed as an input to a two-dimensional CNN layer.

When a CNN layer receives an input image it uses filters to process the input. A CNN filter is a three-dimensional kernel which contain weights. The kernel is convolved with the input, that is it is placed on the input image at every possible location and at each location a weighted sum is calculated for each two-dimensional image along the depth dimension with the weights in the kernel and the values of the input image. This operation produces a new image with a weighted sum at each possible kernel location. The weighted sums are then each evaluated with an activation function such as ReLU. Figure 4 gives an overview of the convolutional operation. The figure does not illustrate the final step which is the elementwise activation-function evaluation.
As the output from a CNN layer is a new image multiple such layers can follow each other, where the output from a CNN layer serves as the input to a following CNN layer. Stacking layers as such is a characteristic of deep neural networks or deep learning.

One way of interpreting how deep CNN layers work is that the first layer of the CNN network which receives the input-image would detect basic features such as contours. The following layer would then receive an input image which describes the presence or absence of basic features. That layer can then look for combinations of multiple such basic features to represent more complex features. Every additional layer increases the complexity of the features it can describe. As an example a face-recognition algorithm could work in such a manner. An early layer would detect primitive shapes such as different lines and curves and while the output from a neuron in a late layer could represent its confidence that it has detected an eye or some other much more abstract feature (Goodfellow et al. 2016).

2.3.6 Deep Convolutional Network Structure

In image recognition tasks there exist some consensus regarding CNN networks broader structure. Four high performance deep convolutional networks are LeNet (Lecun et al. 1998), ImageNet (Krizhevsky et al. 2012), ZFNet (Zeiler & Fergus 2014) and VGGNet (Simonyan et al. 2014). These network share some structural elements.

A general network structure based upon the mentioned high performance networks is multiple convolutional layers in succession followed by a pooling layer which reduces the image resolution or equivalently the amount of data being propagated through the network. The pooling layer can for example return the maximum value in 2x2 grids but it could just as well do some other image-processing such as taking the average. Multiple convolutional layers followed by pooling layers can be used in succession. A multilayer-perceptrons (a combination
of what is called fully connected layers) is finally connected to the output of a pooling layer. Figure 5 illustrates this typical structure.

![Figure 5. An example of how a deep neural network structure inspired by high-performance image recognition networks could look. The data propagation is from left to right.](image)

### 2.4 Kalman Filter

A Kalman filter is an adaptive post-processing method which recursively estimates a system’s state given uncertain observations. It is an elegant solution to many filtering problems. The Kalman filter requires little resources to implement and it sets few constraints on data (Kalman 1960; Crochet 2004).

A Kalman filter uses uncertain measurements to obtain a generally better estimate than each of the measurements are by themselves. It does this by using statistical relations between variables as well as by using dynamical restraints.

Kalman filters are difficult to explain in a concise way. My approach in making the principles of Kalman filters as understandable as possible is to present an example of estimating an object’s position by measuring both its position $x$ and velocity $v$ as done by a Kalman filter. The example is inspired by Faragher (2012) which offers a more comprehensive description.

If we know the position $x$ and velocity $v$ of an object we know how to predict a future position $x'$ by

$$x' = x + v\Delta t$$  \hspace{1cm} (3)

where $\Delta t$ is the elapsed time. Without knowing any external forces we can estimate the future velocity $v'$ by

$$v' = v$$  \hspace{1cm} (4)

As the future position has a dynamical constraint position and velocity will covary. If we read a large change in position then it is likely that we also read a large value for velocity. The measurements contain information about the other.

Both the measurement of $x$ and $v$ will have some variances $\sigma_x^2$, $\sigma_v^2$ due to the uncertainty of the measurements. These variances will translate over to the estimate of the future position $x'$ and velocity $v'$ and will be $\sigma_{x'}^2$, $\sigma_{v'}^2$ and depend on the dynamical equation. Additional uncertainties could also be added artificially to model external effects which are not included in the dynamical equation.

When some time has passed we’ll receive new measurements of the new current state of the system, $x$ and $v$. The Kalman filters then leverages the covariance of the estimated state of $x'$ and
\( \nu' \) together with the covariance of the immediate measurements \( x \) and \( \nu \) to make a correction. The predicted state and measured state together with their variances will each effectively contain a description of a most likely state, the correction a Kalman filter does is to reconcile these pieces of information. The best guess is a sort of intersection of these likely states. When doing this the Kalman filter has leveraged information contained in a dynamical description to improve upon a noisy series of measurements. This method can be extended in many ways.

A concept of gain exists for Kalman filters which describe how fast the filter adjusts to new states, or how much of a weight the filter puts on new observations. The gain can have a performance impact depending on the situation. A small gain can lead to a filter which has good performance when the state varies little but can lead to errors when the state is transitioning.

## 3 Method

To investigate the feasibility of NN based NWP post-processing methods the performance of different NN models have been evaluated with respect to each other as well as to a contemporary post-processing method using a Kalman filter as implemented by SMHI (the Swedish Meteorological and Hydrological Institute). Information about SMHI’s Kalman filter can be found in Wettring (2015).

Multiple different NN models have been trained using ECMWF (the European Centre for Medium-Range Weather Forecasts) data together with surface observations aggregated by SMHI to predict a corrected 2-metre temperature. The correction was done for the +18 hour forecast. Observations from 14 stations, all with different locations and different characteristic forecast errors, were used.

The Keras API (Chollet et al., 2015) was used to design, train and run the NNs. A Nvidia® Tesla K80 graphical processing unit was used to accelerate the computations.

The NN models have been trained using different ECMWF model input variables as well as different forecast timesteps. Networks with different architectures have been trained and evaluated in an iterative process to reach the final networks which produced the results presented in this project. Parameters which have been modified are the structure of layers, the amount of neurons or convolutional filters, training algorithms, pooling functions, activation functions, and regularization parameters.

The NN post-processing method have been evaluated in two scenarios.

- Correcting errors for a single station for a +18 hour forecast
- Correcting errors for multiple stations for a +18 hour forecast

The first scenario evaluates the potential of a NN post-processing approach while the latter scenarios evaluate potential practical limitations of implementing NN post-processing.
3.1 Input Data

The input data is supplied from the ECMWF NWP model and from in-situ meteorological observations recorded at observational stations. The input data covers six years from 2010 to 2015. The forecasts are issued at 00:00 UTC and 12:00 UTC with three-hour forecast steps.

The ECMWF NWP model is interesting as it is a mature, well known and widely used global model. The resolution is lower than many regional models which mean that some systematic errors are more likely or exaggerated and that there is less detail in the model fields available for use as error predictors.

The NWP data used are 81x141 fields with a 9 km spatial resolution containing meteorological parameters such as temperature, pressure, cloudiness and wind at different pressure heights. More information about the dataset can be found at the ECMWF website (ECMWF, 2017). The input fields which were used in the training and operation of the NNs are presented in table 1. The fields were input to the NNs using multiple forecast times. For an +18 hour error prediction the +15 hour and +21 hour forecast fields were also used as input data.

Each ECMWF fields is normalized to have mean zero and unit standard deviation. This normalization is done for each gridpoint respectively for a time period from 2010 to 2014. The year 2015 is a validation year which is never used in training of networks. The data corresponding to 2015 is normalized using the same mean-shift and standard deviation division as used for 2010-2014. This is done such that no information from the validation year is present in the training data set.

The training data supplied by SMHI’s observation stations is mean hourly 2-metre temperatures. The corresponding NWP prediction is interpolated from the four closest NWP grids using the inverse distance-squared scheme. The difference between the interpolated NWP value and the in-situ observation is calculated and is supplied to the neural network as training data. This means that the neural network is training against the error of the interpolated NWP prediction.
Table 1. The ECMWF fields used for training and operating the neural networks.

<table>
<thead>
<tr>
<th>Normalized NN input fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-metre temperature</td>
</tr>
<tr>
<td>2-metre dewpoint temperature</td>
</tr>
<tr>
<td>Temperature at 850 hPa</td>
</tr>
<tr>
<td>Temperature at 925 hPa</td>
</tr>
<tr>
<td>Sea surface temperature</td>
</tr>
<tr>
<td>Surface pressure</td>
</tr>
<tr>
<td>Solar elevation</td>
</tr>
<tr>
<td>Surface net solar radiation</td>
</tr>
<tr>
<td>10-metre U wind component</td>
</tr>
<tr>
<td>U component of wind at 925 hPa</td>
</tr>
<tr>
<td>10-metre V wind component</td>
</tr>
<tr>
<td>V component of wind at 925 hPa</td>
</tr>
<tr>
<td>Vertical velocity at 925 hPa</td>
</tr>
<tr>
<td>Low cloud cover</td>
</tr>
<tr>
<td>High cloud cover</td>
</tr>
<tr>
<td>Total cloud cover</td>
</tr>
</tbody>
</table>

3.2 Observation Stations Used

Figure 6 gives an overview of the observation stations that were used to supply training- and validation data for the evaluation of NNs. There are 14 stations in total which are positioned at locations with varying meteorological backgrounds. Further the stations are divided into three categories called mountainous-, northern- and southern stations. Grouping similar stations could potentially lead to more efficient neural networks.

In figure 6 the stations are color coded with their corresponding group. The blue stations are mountainous, the red stations are northern and the green stations are southern.

The majority of stations had good data availability during the training- and validation period. Some of the mountainous stations however had periods of missing data, for instance Tromsö was missing approximately 1/3 of the values.
3.3 Evaluation of Results

NNs are evaluated by comparing the mean-squared error with respect to the observations for the validation year 2015. Scatter plots and time-series diagrams are used to give a more comprehensive look for example stations how the NN post-processing corrects errors and how the corrected result looks.

The corrected results using NNs are compared with SMHI’s Kalman filter implementation for the year 2015 using 7 different observation stations. The Kalman filtered values were only available for forecasts issued at 00:00 UTC. Because of this limitation, to make as fair comparison as possible, only NN post-processed values issued at 00:00 UTC were used.

A relative error reduction metric was calculated according to equation 5 for both the NN-corrected data and the Kalman-filtered data and compared to the initial error that the ECMWF forecast had.

\[
\text{Error Reduction} = \frac{(y - \hat{y})}{y} \cdot 100
\]  

(5)

\(y\) denotes the initial error and \(\hat{y}\) the error after a correction.
4 Results

4.1 Single Station

The mean squared error of the best performing NNs are presented in table 2 and compared with the unfiltered ECMWF forecast. Table 2 also shows how the raw ECMWF forecast performs for the stations tested.

The NN post-processing improves the temperature-forecast for all stations listed. The stations which saw the greatest increase were stations with already high mean-square errors. A notable example is Tromsö which had a mean-square error of 20.1 °C\(^2\) which was reduced to 3.43 °C\(^2\) after the NN post-processing.

Table 2. 2-m temperature mean squared error for the interpolated ECMWF forecast and the NN post-processed forecast for the validation year 2015. The error reduction column specifies the relative improvement of the NN post-processing.

<table>
<thead>
<tr>
<th>Station</th>
<th>MSE (°C(^2))</th>
<th>Error Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ECMWF</td>
<td>NN</td>
</tr>
<tr>
<td>Gällivare</td>
<td>6.58</td>
<td>4.94</td>
</tr>
<tr>
<td>Pajala</td>
<td>6.94</td>
<td>4.26</td>
</tr>
<tr>
<td>Östersund-Frösön</td>
<td>3.50</td>
<td>2.05</td>
</tr>
<tr>
<td>Stockholm</td>
<td>2.60</td>
<td>1.09</td>
</tr>
<tr>
<td>Norrköping</td>
<td>1.91</td>
<td>1.31</td>
</tr>
<tr>
<td>Landvetter</td>
<td>1.31</td>
<td>1.07</td>
</tr>
<tr>
<td>Helsingborg</td>
<td>1.13</td>
<td>0.857</td>
</tr>
<tr>
<td>Nordkoster</td>
<td>0.662</td>
<td>0.651</td>
</tr>
<tr>
<td>Ronneby-Bredåkra</td>
<td>1.33</td>
<td>1.20</td>
</tr>
<tr>
<td>Nikkaluokta</td>
<td>22.4</td>
<td>9.60</td>
</tr>
<tr>
<td>Straumsnes</td>
<td>16.3</td>
<td>3.55</td>
</tr>
<tr>
<td>Sylarna</td>
<td>10.1</td>
<td>2.95</td>
</tr>
<tr>
<td>Tarfala</td>
<td>5.30</td>
<td>3.15</td>
</tr>
<tr>
<td>Tromsö</td>
<td>20.1</td>
<td>3.43</td>
</tr>
</tbody>
</table>

Examples of the NN-filtered output is shown in figure 4.1 and figure 8. The left side scatter plot shows for Tromsö that the uncorrected temperature forecast has a negative bias which increases in magnitude towards lower temperatures. This bias is corrected with some negative bias remaining at the lower end. The right side scatterplot shows the same information for Stockholm where there is a negative bias with less temperature dependence which is corrected by the NN.

The time-series in figure 8 shows a time-domain presentation of the temperature evolution for Stockholm during the late half of the validation year. The uncorrected- and corrected datasets
have similar behaviour. One noticeable difference is that the NN produced more intense diurnal temperature oscillation during August. Compared with the observations this behaviour more closely matches that of the observations but is positively biased during the day.

Figure 9 shows the same information as figure 8 except for Tromsø instead of Stockholm. Tromsø had an initially larger bias than Stockholm. The positive effects of the correction is readily observed as a larger separation of the temperature time-series. In general the NN-corrected time-series has a similar behaviour to the unfiltered temperature but the values more closely matches the observations. This gives an indication that the neural network mostly relies on the initial forecast and does not produce radically different results, that the initial forecast will dictate the trends.

Figure 7. Scatter plots for (a) Tromsø and (b) Stockholm showing the distribution of the temperatures pre- and post correction with a NN during the validation year 2015.

Figure 8. Time-series of 2 m temperatures in Stockholm. The NN-corrected forecast is compared to the uncorrected ECMWF forecast and the observed values.
The seasonal distribution of error corrections can be seen in figure 10 where monthly temperature means for the NN-corrected forecast, the unfiltered ECMWF forecast and the observations are compared. The largest monthly biases for the unfiltered ECMWF forecast occur during winter and early spring. The filter noticeably improves performance but also removes some of the seasonal dependence. January was the month with the largest filtered monthly bias but some other winter months such as February and November had relatively low monthly biases compared to the rest of the year. During May–September the initial bias is negative but the NN-corrected bias is positive.

4.1.1 Comparison with Kalman Filter

Some of the stations evaluated had corresponding Kalman filtered values available issued 00:00 UTC. The Kalman filtered error has been compared with the NN post-processed errors for the
year 2015. Only 00:00 UTC issue times are used in the error calculation to make a more direct comparison. The results are shown in table 3.

The NN post-processing methods generally leads to lower mean-squared errors than the Kalman filter. Tromsø is the only tested station where the Kalman filter performs better and there the difference is marginal.

Table 3. A mean-square error comparison between Kalman filtered 2-m temperatures and NN-corrected 2-m temperatures. The errors are calculated for the year 2015 and only for forecasts issued 00:00 UTC and valid 18:00 UTC.

<table>
<thead>
<tr>
<th>Station</th>
<th>MSE ( °C²)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Kalman</td>
</tr>
<tr>
<td>Östersund-Frösön</td>
<td>2.41</td>
</tr>
<tr>
<td>Stockholm</td>
<td>1.81</td>
</tr>
<tr>
<td>Norrköping</td>
<td>1.66</td>
</tr>
<tr>
<td>Landvetter</td>
<td>1.22</td>
</tr>
<tr>
<td>Helsingborg</td>
<td>1.19</td>
</tr>
<tr>
<td>Ronneby-Bredåkra</td>
<td>1.26</td>
</tr>
<tr>
<td>Tromsø</td>
<td>2.83</td>
</tr>
</tbody>
</table>

4.2 Multiple Stations

The best MSE obtained when training the NN using grouped station is presented in table 4 together with the corresponding ECMWF error.

The NN successfully reduced the errors at all the stations. The reduction in mean-square error varies with a wide range depending on the stations. The performance is similar to the single-station NNs but with an increase in error magnitude.
Table 4. Comparison of the raw ECMWF mean-square error and the mean-square error for multiple-station NN post-processing. The error is calculated for the validation year 2015.

<table>
<thead>
<tr>
<th>Station Location</th>
<th>MSE ( °C² )</th>
<th>Error Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ECMWF</td>
<td>NN w.r.t. ECMWF</td>
</tr>
<tr>
<td>Mountainous stations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Straumsnes</td>
<td>16.3</td>
<td>4.03</td>
</tr>
<tr>
<td>Sylarna</td>
<td>10.1</td>
<td>4.21</td>
</tr>
<tr>
<td>Tarfala</td>
<td>5.30</td>
<td>3.70</td>
</tr>
<tr>
<td>Tromsø</td>
<td>20.1</td>
<td>4.30</td>
</tr>
<tr>
<td>Northern stations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gällivare</td>
<td>6.58</td>
<td>4.78</td>
</tr>
<tr>
<td>Pajala</td>
<td>6.94</td>
<td>4.44</td>
</tr>
<tr>
<td>Nikkaluokta</td>
<td>22.4</td>
<td>10.0</td>
</tr>
<tr>
<td>Östersund-Frösön</td>
<td>3.50</td>
<td>2.61</td>
</tr>
<tr>
<td>Southern stations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Helsingborg</td>
<td>1.13</td>
<td>1.08</td>
</tr>
<tr>
<td>Landvetter</td>
<td>1.31</td>
<td>1.11</td>
</tr>
<tr>
<td>Nordkoster</td>
<td>0.662</td>
<td>0.640</td>
</tr>
<tr>
<td>Norrköping</td>
<td>1.91</td>
<td>1.32</td>
</tr>
<tr>
<td>Ronneby-Bredåkra</td>
<td>1.33</td>
<td>1.22</td>
</tr>
<tr>
<td>Stockholm</td>
<td>2.60</td>
<td>1.33</td>
</tr>
</tbody>
</table>

5 Discussion

5.1 Performance

Training against the error of grouped stations is feasible as shown but the results are not as good as when a single station was used. The process of training against multiple stations is more resource efficient but a trade-off in error-reduction has to be considered. With more time spent on optimizing the network architecture it is possible that the results can be further improved.

The case of training the mountainous stations as a group experienced the largest reduction in performance when compared to the single station networks at the corresponding stations. This can be attributed to worse data availability at these stations. The missing data was dealt with naively in this project where no data was used in the training at a time where any of the stations involved had missing data. This lead to a significant reduction in available training data. It is
possible to implement schemes which can tolerate missing data, such as an adaptive loss function, but the effect such schemes has on the results is uncertain without further experimentation.

Figure 4.1 showed that the NN was unable to adjust the bias over the complete temperature range. A possible explanation is that the training-set did not contain enough low-temperature situations. A NN basically learns from examples and for a NN to be able to produce certain outcomes it needs to have been trained with such cases.

Different extents in errors can be expected from meteorological stations depending on their location. Stations at locations with unresolved orography such as in mountainous areas can be expected to have relatively high systematic errors. For example surface effects such as mountain wind systems which impact the boundary layer cannot in such cases be adequately resolved and instead depend on parameterization. The three stations with the largest mean-squared errors where Tromsö, Straumsnes and Sylarna which were grouped as mountainous stations. These stations also saw the largest relative reduction in error which was 83.0%, 78.2% and 70.8% respectively.

The temperature at coastal stations is climatologically coupled with the sea. The sea is a thermodynamical system with a massive thermal inertia and has a stabilizing effect in temperature for nearby land. For example by mesoscale mixing processes such as sea- and land breezes. The proximity to the sea means that coastal stations typically experience a smaller range in temperatures. Their initial error can be expected to be comparatively low, given that the land-sea interface is adequately resolved such that islands and similar features are represented. The stations with coastal-influences were all the ones with the lowest initial mean-square errors. Stockholm had by far the largest reduction in error of those stations. A possible explanation is that the ECMWF model did not account for the extent an urban heat-island effect could have in warming Stockholm. This is in part supported by the temperature distribution presented in figure 4.1 which indicated that the temperature in Stockholm was in large underestimated before any correction.

### 5.2 Neural Network Design

Various NN architectures were evaluated to achieve the results presented. In this section I’ll share some of the experience I have obtained from the process.

The structure inspired by the high-performance image recognition networks performed well for the task of predicting errors in the 2-m temperature. An architecture example of how a deep network which worked well for multiple cases in this study is shown in table 5. The problem benefited from having many filters and especially from having a large number of neurons in the last fully connected layers. The amount given in table 5 has filter- and neuron numbers limited by computational resources. It is possible that even larger amounts could improve performance further. Having more than two fully connected layers before the output did not improve the results.
The more stations which are trained against the more layers benefited the result. When training using a single station the required amount of convolutional layers was lower but the general structure remained. For example when training against the error at Tromsø which had a lower amount of training data available than other stations the best results were obtained when only one or two convolutional layers were used before each pooling layer.

The rate of dropout required for the best result varied. Dropout rates from 15% to 50% produced good results depending on the station.

During training it was found that the Adadelta optimization algorithm (Zeiler 2012) with mean-square error as the loss function worked better than other alternatives that were tried in the Keras API. No significant sensitivity to the training hyperparameters such as the learning rate and learning decay were noticed during training. Standard values performed well.

There are a few mentionable things that were not tried which could improve upon the results or shorten the training process. Batch normalization (Ioffe et al. 2015) is one example. Some work might be required to use batch normalization with ReLU activation functions. Different methods of initializing the weights were not tried. All weights were initialized to random uniform values. Weight regularization is another possible way to improve the network performance. Using weight regularization the magnitude of the weights are penalized in the loss function.

Using a reduced input field size is recommended as it has a direct influence on network size and training performance.
Table 5. An example of how an architecture which performed well for the problem of predicting the error in the ECMWF forecasts. The structure is the same as shown in figure 5 except for the addition of drop-out layers.

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Filters</th>
<th>Activation</th>
<th>Kernel Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D convolution</td>
<td>80</td>
<td>ReLU</td>
<td>3x3</td>
</tr>
<tr>
<td>2D convolution</td>
<td>80</td>
<td>ReLU</td>
<td>3x3</td>
</tr>
<tr>
<td>Average Pooling</td>
<td>2x2</td>
<td></td>
<td>2x2</td>
</tr>
<tr>
<td>2D convolution</td>
<td>160</td>
<td>ReLU</td>
<td>3x3</td>
</tr>
<tr>
<td>2D convolution</td>
<td>160</td>
<td>ReLU</td>
<td>3x3</td>
</tr>
<tr>
<td>Average Pooling</td>
<td>2x2</td>
<td></td>
<td>2x2</td>
</tr>
<tr>
<td>2D convolution</td>
<td>320</td>
<td>ReLU</td>
<td>3x3</td>
</tr>
<tr>
<td>2D convolution</td>
<td>320</td>
<td>ReLU</td>
<td>3x3</td>
</tr>
<tr>
<td>Average Pooling</td>
<td>2x2</td>
<td></td>
<td>2x2</td>
</tr>
<tr>
<td>2D convolution</td>
<td>640</td>
<td>ReLU</td>
<td>3x3</td>
</tr>
<tr>
<td>2D convolution</td>
<td>640</td>
<td>ReLU</td>
<td>3x3</td>
</tr>
<tr>
<td>2D convolution</td>
<td>640</td>
<td>ReLU</td>
<td>3x3</td>
</tr>
<tr>
<td>Max Pooling</td>
<td>2x2</td>
<td></td>
<td>2x2</td>
</tr>
<tr>
<td>Fully Connected</td>
<td>4096</td>
<td>ReLU</td>
<td></td>
</tr>
<tr>
<td>Dropout</td>
<td>0.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fully Connected</td>
<td>4096</td>
<td>ReLU</td>
<td></td>
</tr>
<tr>
<td>Dropout</td>
<td>0.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output, Fully Connected</td>
<td>Linear activation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6 Conclusions

It has been shown that using deep neural networks with 6 years of combined training- and validation data to reduce forecast errors for 2-m temperature works well. All evaluated station locations experienced an error reduction when post-processing the forecast with deep neural networks. The mean-square error of the forecast was reduced by 83% for Tromsø which was an especially erroneous location. The error reduction however varied much across the stations. The lowest reduction in error was 1.7% for Nordkoster which initially had the lowest mean-square error of 0.7 °C². The other stations had a typical error reduction between around 20% to around 70%.

The error reduction is of similar relative magnitudes as those found in Marzban (2002) which used neural networks to reduce forecast temperature errors. In Marzban (2002) the error reductions were however for a different model and with a lower spatial resolution.

The five years of bidaily training data that was used to train the networks is far from a climatological representation of any station. It is very likely that more training data would lead to a further error reduction as noticeable biases were still present after NN post-processing. Time-series analyses showed that the post-processed forecast had the same overall pattern as the
initial forecast which mean that the neural networks are limited in correcting some erroneous forecasts. This could also potentially be remedied by using more training data.

A time-series analysis showed that there were minor behavioural changes in the evolution of the temperature after NN-post processing. This can be important as there can exists constraints which requires that the temperature evolution is consistent with respect to unfiltered model output. The behavioural similarities between the unfiltered model output and the NN-post processed values can also be seen as an indication that the method with 5 years of training data is unlikely to produce radically different forecast patterns, which might be required for a good correction in rare cases.

Deep neural network post-processing was shown to offers advantages over SMHI’s implementation of a Kalman filter for the majority of the stations evaluated for the +18 hour forecast. The neural network approach worked better for all stations except for Tromsö where the performance was marginally worse.

Training using a single station at a time lead to better results than when training multiple stations at once. The results are however still a significant improvement compared to the raw ECMWF forecast. Including multiple stations in a single neural network is more resource efficient but any implementation has to consider a potential reduction in error reduction performance. This is especially important when stations with missing data are included in a group as the missing data has to be accounted for in some way.

7 Acknowledgements

I would like to thank my thesis supervisor Emelie Karlsson. Emelie’s expertise in meteorology and her input on the project has been very valuable and helped the thesis keep a high quality. Her colleague Fredrik Karlsson at SMHI have been of great assistance as well. His input on technical matters have been very helpful in improving upon the results, the design of the neural networks and the associated computer programs.

I would also like to express gratitude to all the friendly people I encountered at SMHI, I felt welcome during my stay there as I was working on the thesis.
8 References


**Internet Resources**

ECMWF. *Datasets* (2015) [https://www.ecmwf.int/en/forecasts/datasets](https://www.ecmwf.int/en/forecasts/datasets) [2018-03-05]

Chollet, F. et al. (2015). *Keras* [https://github.com/keras-team/keras] [2017-11-06]