A comparison between bootstrap and dropout for uncertainty estimates of time series forecast using a convolutional neural network

ELIN VALLBO
A comparison between bootstrap and dropout for uncertainty estimates of time series forecast using a convolutional neural network

ELIN VALLBO
Abstract

Forecasting the future is important in many applications, including forecasting future sales volumes. However, point forecasts without any estimates of the uncertainty are not as useful as if the uncertainty in the forecast is included. Uncertainty estimates allow a range of possible outcomes to be considered, not only the most likely as in the case of point forecasts.

In this study the aim is to produce uncertainty estimations for forecasts made 120 days into the future for three different products. The three products are usually used for several years by the consumers and all three products belong to the same category. For the purpose of estimating the uncertainty in forecasts made with a neural network, two different methods were used, bootstrapping residuals and dropout, respectively. The methods were applied to a convolutional neural network, CNN, with dilated causal convolutions. The two methods were compared based on their coverage probability when constructing an interval which aims for 95% coverage probability. In brief, bootstrapping residual performed better on all three products with a mean coverage probability of 76% compared to 43% for dropout. The study shows that it is possible to estimate the uncertainty in forecasts made with a CNN using bootstrap and dropout. Bootstrap seems more suitable than dropout, however the study was only made on three products in the same product category which makes it hard to draw any general conclusions.
**Sammanfattning**

Att kunna göra prognoser för framtiden är viktigt i många applikationer. Ett exempel på detta är att förutspå hur mycket det kommer säljas i framtiden av en produkt. Endast punktprognoser ger mindre information än om även osäkerheterna i prognoserna kan redovisas. När osäkerheten i prognosen redovisas kan en rad möjliga uttål studeras och inte bara det mest troliga som en punktprognos redovisas.

# Contents

1 Introduction 1

2 Background 4
   2.1 Time Series 4
   2.2 Time series forecasting 5
      2.2.1 Regression Models 5
      2.2.2 Neural Networks 7
   2.3 Prediction Intervals 12
      2.3.1 Dropout 13
      2.3.2 Bootstrap 14
      2.3.3 Delta method 16
      2.3.4 MVE method 16
      2.3.5 Lower upper bound estimation 16
   2.4 Evaluation metrics 17
      2.4.1 Evaluation of forecasts with neural networks 17
      2.4.2 Evaluation of prediction intervals 17
   2.5 Research design 18

3 Methods 21
   3.1 Pre-processing of data 21
   3.2 Time series predictions 22
      3.2.1 Dilated causal convolutional network 22
      3.2.2 Naïve model 23
   3.3 Uncertainty intervals 23
      3.3.1 Bootstrap 25
      3.3.2 Dropout 25
   3.4 Implementation 26
4 Results

4.1 Time series predictions

4.1.1 Predictions using dilated causal convolutional neural network

4.1.2 Naïve model

4.2 Uncertainty intervals

4.2.1 Bootstrap

4.2.2 Dropout

5 Discussion

5.1 Forecasts using dilated causal convolutional neural network

5.2 Uncertainty intervals

5.2.1 Bootstrap

5.2.2 Dropout

5.3 Ethics and environmental impact

5.4 Future work

6 Conclusions

Bibliography
Chapter 1

Introduction

Time series forecast involves the prediction of future values based on the known history in form of a time series. Time series forecast is used in many different areas and have been used for a long time. There are a variety of methods for doing forecasting given a time series, some very simple and some more complex. Maybe the simplest model is to look at the previous time-step to forecast the next, but more complex models are often needed to capture trends, seasonal patterns and holidays or other special events affecting future values. The reason forecasting the future is important is that it gives the possibility to plan. Planning can help businesses to prepare for the future and save both financial and environmental resources.

Time series forecasts have traditionally been made using regression models of different types. Two examples of classical models are Autoregressive integrated moving average (ARIMA) and exponential smoothing while a more modern set of methods for time series analysis is neural networks [1]. Neural networks have received attention in a lot of areas the last years and have proven good performance in, for example, computer vision [2]. There are several different types of neural networks, the two most interesting for time series analysis are long short-term memory networks, LSTM, and convolutional neural networks, CNN. LSTMs are commonly used in sequence-to-sequence learning tasks within natural language processing. A common sequence-to-sequence task is translating a sentence from one language to another. This area of application makes them interesting also for time series analysis since time series have similar temporally ordered nature as sequence-to-sequence tasks. However, LSTMs are quite slow and tricky to train. CNNs are classically considered networks for image analysis but have lately also been explored for sequence learning [3]. When doing time series with CNNs a special type
of convolution is often used called dilated causal convolution. The dilation in the convolution reduces the number of layers necessary to account for data far into the past and a causal convolution makes sure that no future values are used when calculating a step ahead.

Both LSTMs and CNNs produce point forecasts, which is often less useful than a forecast with some kind of uncertainty estimate. Point predictions will only present the most likely outcome of the future while uncertainty estimates provide a range of possible outcomes. Uncertainty estimates therefore help an analyst to better interpret the output from the model and analyse the range of outputs, not only the most likely outcome.

For many of the traditional methods it is possible to construct prediction intervals. Prediction intervals is a form of uncertainty estimates that accounts for both the uncertainty in the model and the noise in the historical data. Prediction intervals are given a confidence level which describes how many of the true values that are expected to lie within the prediction interval, a common confidence level is 95 %. For neural networks, however, it is not possible, in general, to obtain prediction intervals in an analytical way, instead these have to be estimated. Many methods have been proposed to estimate prediction intervals for neural networks, some of them are: dropout [4], bootstrap [5], delta method [5], Mean Variance Estimation (MVE) method [5] and upper lower bound estimation[6].

It is often of interest to create models which can perform forecasts with uncertainty estimates, to either perform the forecast automatically or being a guiding system when humans make predictions. This degree project aims at contributing to a forecasting system which will work as a guiding system for human experts. The system will forecast sales volumes with uncertainty estimates with the help of past sales volumes and the price. The system will be evaluated on three products which, in general, are used for several years by the consumer. All three products belong to the same product group.

**Objective:** To create a system to forecast sales volumes 120 days into the future given past values of both sales volumes and price. The system should not only be able to forecast a value for the future, but it should also provide uncertainty estimates for the values forecasted. To contribute to such a system this degree project will focus on the following research question:

- **When making forecasts for sales volumes with a causal dilated convolutional network which of the following two methods, bootstrapping residuals or dropout, gives uncertainty estimates with highest coverage?**

To be able to answer this question the necessary theory regarding time series,
neural networks and prediction intervals is covered in chapter two. In chapter three the procedure to build the CNN is described and the two ways of estimating the uncertainty. In chapter four the results for the two methods for uncertainty estimations are presented together with the accuracy of the point forecasts of the CNN which is compared against a naïve model. The results are finally discussed and analysed in chapter five and the conclusion is stated in chapter six.
Chapter 2

Background

2.1 Time Series

A time series is data collected successively over a period of time where the data points in the series have a natural order. The distance in time between two data points is often constant, i.e. the frequency is constant. Time series often show dependencies between past and future values. These dependencies may be used when analysing the series [1].

A time series may be divided into three patterns and a remaining part. The three patterns are trend, seasonal and cyclic patterns. Trend is a longer ongoing decrease or increase of the times series while seasonality are patterns which repeat with a fixed frequency, usually, a year, a week or a day. Cyclic patterns are repeating, but do not have a fixed frequency and span over a longer period of time, usually at least two years. The procedure of dividing the time series into patterns is called time series decomposition, it could be the above mentioned patterns or other patterns of choice [1].

When analysing a time series, it can often be useful to decompose it and handle the three parts separately. Some methods, such as ARIMA, requires decomposition. A decomposition can either be additive or multiplicative, the two versions are shown in equation 2.1 and equation 2.2 where \( y_t \) is the time series. \( S_t \) is the seasonal part, \( T_t \) is the trend-cycle part and \( R_t \) is the remaining part. Additive decomposition is used when the magnitude of seasonality or the variance around the trend-cyclic part do not vary with the level of the time series. If they are proportional to the level of the time series a multiplicative decomposition is more suitable. Data where the seasonal part is removed is called deseasonalised data [1].
\begin{align*}
y_t &= S_t + T_t + R_t \quad (2.1) \\
y_t &= S_t \times T_t \times R_t \quad (2.2)
\end{align*}

2.2 Time series forecasting

In time series forecasting, historical values are used to build a model which is later used to predict future values. The model could, for example, be a regression model or a neural network. Sometimes predictive variables, i.e. variables which affect the future but are not part of the time series, are used in addition to the time series. When forecasting sales volumes, the historical sales volumes are the time series and the price or competitors’ price of the forecasted time-step could be used as predictors.

When doing time series prediction, the desired outcome could be either a prediction one time step ahead or multiple time steps ahead. Whether the goal is to predict one step or multiple steps, the uncertainty will increase when the prediction lies further into the future.

2.2.1 Regression Models

Which method should be used to build the prediction models depends on the time series and the forecast conditions. One group of models is regression models. There are some traditional methods such as ARIMA and exponential smoothing [1]. Lately new regression based models such as prophet in 2018 [7] have been presented.

ARIMA

ARIMA stands for Autoregressive integrated moving average and is a combination of an autoregressive model and a moving average model. The autoregressive model of ARIMA uses a combination of past observations to predict the future value. The moving average model uses past forecast errors to predict the future. It is worth noting that the moving average in ARIMA is not a moving average of past values, it is a moving average of the past forecast errors. The integrated part of ARIMA means that the data has been made stationary by differencing the data, i.e. compute the difference between successive observations. This is made because ARIMA requires a stationary time series [1].
An ARIMA model has three parameters, $p, d$ and $q$. The first parameter, $p$, is the order of the autoregressive part, i.e. how many lagged values that are used in the autoregressive model. The second parameter $d$ is the order of differencing and the last parameter, $q$, is the order of the moving average model, i.e. how many past errors that are used. ARIMA in its simplest form cannot handle seasonality, however, it can be extended to handle seasonality [1].

It is also possible to calculate prediction intervals for predictions made using ARIMA. Prediction intervals describes the uncertainty in the forecast made with the model. For multi-step predictions the prediction intervals grow the further into the future the prediction is. However, prediction intervals for ARIMA models tend to be too narrow [1].

**Exponential smoothing**

Exponential smoothing was proposed in the late 50s by Brown (1959), Holt (1957) and Winters (1960) [1]. Exponential smoothing uses a weighted average of past values where the weights decay exponentially the further in the past they are. The model can be extended so that both trend and seasonality can be handled. State-space methods is, the broader, underlying method category to exponential smoothing methods. With exponential smoothing it is also possible to extract prediction intervals [1].

**Prophet forecasting**

Prophet is a newer regression model proposed by Taylor and Letham in 2018. It can be chosen to be either an additive or multiplicative regression model with four main components. The first component is a piecewise linear or logistic trend which automatically detects changes of trends in the data. The two next parts handle seasonality, yearly and weekly, using Fourier series. Finally, the last component is a user-provided list of important holidays, special events or other predictive variables.

Prophet is suitable for high frequency data, such as daily data. Prophet has several advantages such as the observations do not have to be equally spaced as in the case of ARIMA, i.e. prophet is robust against missing values. It is also a flexible model and fast to fit. It is possible to construct uncertainty intervals to describe the uncertainty in predictions made with prophet. Uncertainty intervals are constructed either by sampling trend changes or by full Bayesian sampling. The latter creates more accurate uncertainty intervals but is also computationally heavier [7].
2.2.2 Neural Networks

Neural networks have gained popularity in many areas and have lately also been explored in the area of time series analysis. Neural networks can learn complex non-linear dependencies but often lack any connection between different samples, i.e. some memory, which make time series analysis harder. One advantage of neural networks is that the form of the mapping function between the input and output does not have to be determined in advance, this is what the network learns. However, neural networks also have some disadvantages, for example, they are very data hungry and need large data sets to learn from. Also, neural networks are neither translucent nor easy to interpret.

The following sections will focus on neural networks for time series analysis with supervised learning. The section will cover the pre-processing of data for neural networks and three possible architectures to perform time series forecasting with neural networks, feed forward network, long short-term memory neural network and convolutional neural network.

Pre-processing of data

There are several steps in preparing time series data for a neural network model. Neural networks work best with normalised or standardised data. Transforming the data helps to prevent exploding or vanishing gradients. To improve the performance in the training phase the data is normalised to values between 0 and 1 or standardised to have zero mean and unit variance.

Another frequently discussed part of pre-processing time series data for neural networks is if it should be deseasonalised or not. There are some contradicting findings regarding seasonality and neural networks. Most of the articles on the subject are concerning feed forward networks. Zhang and Qi (2005) presents both result in favour of and against neural networks ability to successfully approximate seasonal patterns.

Training and test data

To perform supervised learning the data must be in samples made of input data and labels, i.e. the desired output values. From these examples the network then learns the function which maps input-values to output-values. In time series forecasting the input is past values and the label is the future values. A common way to divide time series data into such samples is a sliding window, shown in figure 2.1. Let the input dimension be denoted $n$ and the number of outputs $h$. The first input is the $n$ first values in the time series. The next $h$
values will be the corresponding label. This window then slides forward to create more examples. The samples could have an overlap.

The data is often divided into training and test data. This is often done by randomly picking samples to each data set. However, with time series data this method does not work because the network should not be trained on any data that lies further into the future than the test data. The last time step in the training data is called forecast origin. The division may be done in several different ways for time series data, three different ways are shown in figure 2.2 and are explained below.

- Fixed-origin evaluation is shown in figure 2.2a. Here one forecast origin is used and a prediction is made from this origin. This evaluation method is sensitive to the characteristics of the chosen forecast origin. Since only one prediction is made it is hard to measure variance in the predictions [9].

- Rolling-origin evaluation is visualised in the figure 2.2b. In rolling-origin evaluation the forecast origin is moved in each iteration, it may be moved one or several time-steps. In each iteration the same number of points as the origin is moved with is moved from the test data to the training data, making the amount of training data grow for each iteration. There are two versions of rolling-origin, one where the model is retrained every time the forecast origin is moved and one which only uses the moved test data as input. They are called rolling-origin recalibration and rolling-origin update respectively [9].

- The last version is rolling-window which can be found in figure 2.2c. In this method test data is added to the training data but the size of the training data is held constant by removing the oldest observations [9].
Multi-step prediction

When forecasting it could be interesting to predict one or multiple time steps into the future. The number of time steps ahead which the forecast is made for is called the forecast horizon. To perform multi-step predictions with neural networks there are several strategies. Three of the strategies are listed below.

- The recursive strategy means that one prediction is made and then this prediction is used as input for doing the next prediction. This is then continued until the desired forecast horizon is reached. The drawback with this is that errors are accumulated through the forecast [10].

- For the direct strategy the model is built to forecast a value multiple steps ahead. If values at several forecast horizons should be forecasted one model is built for each horizon. This strategy will not have any problem with propagating error but will assume that the different forecasted horizons are conditionally independent [10].

- The MIMO, Multi-Input Multi-Output, strategy has several values as output, one for each forecast horizon that is predicted. MIMO does not require conditional independent forecast horizons, but all horizons
needs to be forecasted with the same model, which reduces the flexibility compared to the direct strategy. In general, MIMO preforms better than the single output approaches [10].

**Feedforward network**

Feedforward neural networks are the simplest form of neural networks. A feedforward network has no circular connections. They have been tried in the area of time series forecasting but have one major drawback, feed forward networks do not have any memory. This leads quickly to a large input dimension which gives a lot of weights to learn. For example, if the network should learn yearly seasonality with daily observations we need to input at least 365 points, the number of input points is multiplied with the dimension of the points. This property makes feed forward networks less suitable for time series analysis and several other structures have been proposed.

**Long short-term memory network**

Long short-term memory networks, LSTMs, are not feedforward networks. LSTMs belong to the family of recurrent networks which have connections that form loops in the network. What distinguishes LSTMs from other recurrent networks is that they have a type of memory. This memory has made them popular in sequence learning, i.e. learning from sequence data. Sequence data is simply data where the values are sequentially ordered, as for example letters in a text or sound in an audio file. The similarity between sequence data and time series data is the reason that LSTMs have been proposed for time series forecasting. An LSTM is built out of cells. A common LSTM have cells with three gates, an input gate, a forget gate and an output gate. The input gate regulates how much new information the cell should take in. The forget gate controls to what extent a value should remain in the cell, i.e. be remembered. Lastly the output cell decides how the values stored in the cell are used to compute the output [11]. This mechanism of being able to remember and forget improves the training of LSTMs compared to classical recurrent network as it reduces the risk for exploding or vanishing gradients. However, they are still quite hard to train [12].

**Convolutional neural network with dilated causal convolution**

Convolutional neural networks, CNNs, are classically used in computer vision, but they have lately also been used for sequence data. In CNNs, a kernel
Figure 2.3: Dilated causal convolution with dilation rate 2. Inspired from [14].

is moved over the data and convolution is applied between the kernel and the data. Several kernels can be used in parallel with each layer to extract multiple features. Kernels are often referred to simply as filters. CNNs have the advantage of being faster to train than LSTMs and do not have the same number of parameters as a feed forward network with the same input size [13].

In a similar manner to a feed forward neural network, past values used for the forecast have to be given as input for each prediction. The reason that CNNs have fewer weights than a feed forward network is because it does not have fully connected layers which process the input. Instead a convolution is performed between the input and each filter.

For time series forecasting a dilated causal convolution is often used [13] [14]. Causal convolutions make sure that no future observation is taken into account when forecasted, i.e. only observations up to and including $x_{t-1}$ are used when predicting $x_t$ [14]. Dilated convolution means that the filter is applied to every $n$:th element of the input values to each layer. In CNNs the dilation typically increases for each layer, for example it can be a series of $[2^0, 2^1, 2^2, 2^3, 2^4, ...]$. This allows for long term dependencies to be taken into account with relatively few layers [13] [14]. The dilated causal convolution is visualised in figure 2.3.

### Residual connections

Neural networks often struggle to learn identity mappings, i.e. mappings where the input goes unchanged through a layer. Residual connections might be a way to allow the network to learn identity mappings, which can improve the performance. A residual connection is a connection which adds the input to layer $n$ with the output of layer $n+t$, see figure 2.4 where $t = 1$. This shortcut makes it possible for the network to learn the residual between the output
and input instead of the output directly.

When using residual connections instead of learning \( H(x) = x \) the network can learn the residual \( R(x) = H(x) - x \) which in the case of an identity mapping is setting all weights in \( R(x) \) equal to zero.

![Residual connections](image)

(a) Two layers without residual connections.  (b) Two layers with residual connections.

Figure 2.4: Residual connections. Inspired by [15].

### 2.3 Prediction Intervals

A point prediction alone is not that useful if nothing about the uncertainty in the prediction is known. Prediction intervals can be one way to provide this uncertainty. A prediction interval is an interval within which a future observation will lie with a certain probability and is usually defined as:

\[
\hat{y}_{T+h|T} \pm c \hat{\sigma}_h
\]

(2.3)

where \( \hat{y} \) is the predicted value, \( T \) is the time up to which observations are available, \( h \) is the time step for which the prediction is made, \( c \) is the multiplier defining the desired coverage probability. \( \sigma_h \) is the standard deviation of the forecast distribution at time \( T+h \). In general, prediction intervals grow wider the further into the future the prediction is made [1].

There are several different methods for computing \( \sigma_h \) depending on the type of forecast that is made. In some cases, it is, under the assumption that
the errors are normally distributed and uncorrelated, possible to calculate the standard derivation directly [1]. When it not possible to assume normally distributed errors, bootstrapping might be used to obtain $\sigma_h$ (see section 2.3.2).

Prediction intervals should not be confused with confidence intervals which quantify the uncertainty between the prediction and the true regression [5]. Confidence intervals only contain the model uncertainty whereas prediction intervals also take data noise variance into account. The uncertainty of prediction intervals can therefore be written as in the equation below.

$$\sigma_y^2 = \sigma_{model}^2 + \sigma_{noise}^2$$ (2.4)

[16] The model uncertainty can be due to several factors, including model misspecification, parameter uncertainty, and training data uncertainty. Model misspecification is the uncertainty of how well the model $\hat{f}(x)$ approximates the true function $f(x)$ under the assumption of optimal parameters and plenty of data. Training data uncertainty tries to capture the uncertainty of how representative the training data is. Parameter uncertainty is the uncertainty regarding if the optimal parameters are used [16]. To be able to create true prediction intervals all these uncertainties need to be captured which is often a challenge. For neural networks there is often no other option than estimating the prediction intervals, i.e. there are no general closed form solution to find prediction intervals. In this section some methods will be described to estimate prediction intervals.

### 2.3.1 Dropout

Dropout as a method for estimating prediction intervals was first proposed by Gal and Ghahramani [4] in 2016. The authors describe it as an approximation for a Bayesian approach. A neural network with dropout applied before each weight layer is mathematically equivalent to an approximation of a probabilistic deep Gaussian process. Gal and Ghahramani [4] show that the distribution consisting of the predictions from a neural network with Monte Carlo dropout minimize the Kullback-Leibler divergence between this approximate distribution and the posterior of a deep Gaussian process [4]. Kullback-Leibler divergence measures the distance between two probability distributions.

Dropout is a method usually used in the training phase of a network to create regularization and in order to avoid overfitting. When dropout is applied to a layer, node weights are set to zero with a probability $p$. Which nodes that are dropped is chosen randomly before each training sample [17].
For estimating the parameter uncertainty with dropout the posterior distribution is estimated using stochastic forward passes. A stochastic forward pass is a prediction where dropout is applied. When stochastic forward passes are used both during training and testing it is referred to as Monte Carlo. When \( B \) forward passes are preformed we obtain \( \{\hat{y}_1^*, \ldots, \hat{y}_B^*\} \). The value of the point prediction is calculated as the mean of the \( B \) stochastic forward passes, as in equation 2.5. The parameter uncertainty is estimated using equation 2.6 [18].

\[
\hat{y}^* = \frac{1}{B} \sum_{b=1}^{B} \hat{y}^*_b
\]  

(2.5)

\[
\text{var}(f(x)) = \frac{1}{B} \sum_{b=1}^{B} (\hat{y}^*_b - \bar{\hat{y}}^*)
\]  

(2.6)

Zhu and Laptev [18] add, in addition to the parameter uncertainty, both the training data uncertainty and inherent noise to form prediction intervals. The inherent noise is estimated with a validation set. The validation set consist of \( V \) samples, \( X' = \{x'_1, \ldots, x'_V\} \) and \( Y' = \{y'_1, \ldots, y'_V\} \). For each sample in the validation set a prediction is made and the noise is estimated as in equation 2.7 below [18].

\[
\hat{\sigma}^2 = \frac{1}{V} \sum_{v=1}^{V} (y'_v - \hat{f}(x'_v))^2
\]  

(2.7)

The last part of the prediction intervals, training data uncertainty, do Zhu and Laptev [18] estimate using an autoencoder which should approximate the distance between the input and the training data in the test phase. This autoencoder is a useful tool for automating the detection of anomalies in the data.

### 2.3.2 Bootstrap

Another method for estimating prediction intervals is bootstrapping. Bootstrapping is method for creating new data sets by resampling. There are several types of bootstrapping, including bootstrapping pairs and bootstrapping residuals [19].

- **Bootstrapping pairs** is a method for creating new data sets where samples, consisting of input and labels, are drawn from the training data. The pairs are sampled with replacement.

- **Bootstrapping residuals** is a method where errors are drawn from an error distribution and added to the labels of data thus creating new data.
sets. As in bootstrapping pairs, the errors are drawn with replacement. In order to create the error distribution a prediction for each training sample is made and the error calculated as the difference between the prediction and the label.

Bootstrapping pairs is more robust if the model is misspecified, i.e. the model is biased or not complex enough. However, when working with data which has a natural order, such as time series, bootstrapping residuals has the advantage of only requiring the assumption that the errors are independent. Bootstrapping pairs is not suitable for data with a natural order as this breaks this order when random drawing pairs of input and output to form the new data set. [19].

When bootstrapping prediction intervals $B$ new data sets, $\{D\}_{b=1}^{B}$, are formed using bootstrapping pairs or bootstrapping residuals. A neural network is trained for each dataset $D_b$. Predictions are then made from all of the $B$ neural networks and the prediction from the bootstrap model is calculated as the mean of these predictions

$$
\hat{y}_i = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_i^b
$$

(2.8)

where $\hat{y}_i^b$ is the prediction of the $i$th sample from model $b$. The parameter uncertainty can then be estimated as

$$
\hat{\sigma}_{\hat{y}_i}^2 = \frac{1}{B-1} \sum_{b=1}^{B} (\hat{y}_i^b - \hat{y}_i)^2.
$$

(2.9)

To be able to form prediction intervals we also need the inherent noise, $\sigma_{\epsilon_i}^2$, defined in equation 2.10 were $t$ is the label. To estimate this variance a neural network with a custom cost function, equation 2.11, can be used. In equation 2.12 the value $r_i^2$ is defined.

$$
\sigma_{\epsilon_i}^2 \simeq E(t - \hat{y}_i)^2 - \hat{\sigma}_{\hat{y}}^2
$$

(2.10)

$$
C_{BS} = \frac{1}{2} \sum \left[ \ln(\sigma_{\epsilon_i}^2) + \frac{r_i^2}{\sigma_{\epsilon_i}^2} \right]
$$

(2.11)

$$
r_i^2 = \max \left( (t_i - \hat{y}_i)^2 - \sigma_{\hat{y}_i}^2, 0 \right).
$$

(2.12)

$$
D_{r^2} = \{(x_i, r_i^2)\}_{i=1}^{n}
$$

(2.13)
The network is trained with a data set which links each \( r_i^2 \) to an input form the original training set. In total, bootstrap requires that \( B+1 \) networks are trained which makes it quite computationally intensive. However, it is easy to run the training in parallel since the networks can be trained independently on the different data sets [5].

### 2.3.3 Delta method

The delta method interprets the neural network as a nonlinear regression model and this allows for the application of asymptotic theories in order to construct prediction intervals. The method consists of calculating prediction intervals form the gradients and Jacobian of the neural network with respect to its parameters. The delta method is quite computationally heavy in the development phase but less so in the predicting phase. However, the delta method gives low quality prediction intervals if the level of the noise in the data is correlated to the magnitude of the true values [5].

### 2.3.4 MVE method

The MVE, mean variance estimation, method is a method for training a network to estimate the variance of the predictions, which is used to calculate prediction intervals. It can either be a separated network that estimates the variance or a network which both estimates the mean and the variance of the forecast [20]. The cost function used, see equation (2.14), is very similar to the the cost function from the bootstrap method previously mentioned in equation (2.11) [5].

The MVE method is a simple method without any complex computations such as the derivatives in the delta method. However, the MVE method has one major drawback, it does not take the misspecification of parameters into consideration and therefore gives too narrow prediction intervals [5].

\[
C_{MVE} = \frac{1}{2} \sum \left[ \ln(\hat{\sigma}_i^2) + \frac{(t_i - \hat{y}_i)^2}{\hat{\sigma}_i^2} \right]
\]  
(2.14)

### 2.3.5 Lower upper bound estimation

In the lower upper bound estimation method a network estimates the lower and upper bound of the prediction intervals instead of a point prediction. The lower and upper are estimated by minimizing CWC, the coverage width-based
2.4 Evaluation metrics

There are several ways to evaluate both the accuracy of time series forecasts and the quality of prediction intervals. In this section some methods will be described.

2.4.1 Evaluation of forecasts with neural networks

There are several evaluation metrics to measure the accuracy of time series forecasting. A selection of available measures is shown in table 2.1. The first two measures in table 2.1, mean square error, MSE, and root mean square error, RMSE, are scale dependent, which means that they cannot be compared over datasets with different scale levels. They are also sensitive to outliers.

The last two measures in table 2.1, mean absolute percentage error, MAPE, and symmetric mean absolute percentage error, sMAPE, are scale-independent. However, they produce a skewed measure close to zero. MAPE also gives infinite values if the true value is zero. The sMAPE can solve this problem to some extent but if the true value is zero the predicted value is also likely to be close to zero which leads to division by a small number.

2.4.2 Evaluation of prediction intervals

There are two properties which are interesting when it comes to prediction intervals, their coverage probability and their width. A prediction interval should be as narrow as possible but still have the desired coverage probability. The coverage probability can be measured with PICP, which stands for *prediction interval coverage probability* and is defined in table 2.2. In words the PICP can be described as the number of points which lies within the interval divided by the total number of points in the evaluation period. The width of the prediction interval can be measured with the MPIW, which stands for *mean
Table 2.1: Some error measures for measuring forecast error. \( y \) denotes the label and \( \hat{y} \) denotes the predicted value.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>( \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 )</td>
</tr>
<tr>
<td>RMSE</td>
<td>( \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} )</td>
</tr>
<tr>
<td>MAPE</td>
<td>( \frac{100%}{n} \sum_{i=1}^{n} \frac{</td>
</tr>
<tr>
<td>sMAPE</td>
<td>( \frac{100%}{n} \sum_{i=1}^{n} \frac{</td>
</tr>
</tbody>
</table>

The last measure in Table 2.2 is CWC, the coverage width-based criterion. This takes both the width and the coverage probability into account. The CWC should be as low as possible to get the narrowest prediction intervals with a coverage probability at the confidence level.

2.5 Research design

In this project a CNN with dilated causal convolutions and skip connections is selected. CNNs used for sequence data have shown promising results worth further investigations, one example of this is the WaveNet [14]. CNNs are also faster and easier to train than LSTMs which is beneficial considering the small amount of data available in this project.

Feedforward networks are not an option due to their lack of memory which leads to large inputs and a lot of parameters to learn. This is a problem because of the limited amount of data available for this project, which would make it difficult to avoid overfitting.

As discussed earlier there is no clear consensus whether data should be deseasonalised or not before being fed to the network. For simplicity the seasonality was not removed in this project.
Table 2.2: Evaluation measures for prediction intervals. $y_i$ denotes the labels, $L_i$ denotes the lower bound of the prediction interval and $U_i$ the upper bound. NMPIW is a normalised MPIW, $\mu$ is the nominal confidence level and is tighter with the hyperparameter $\eta$ controlling the jump of the CWC.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICP</td>
<td>$\frac{1}{n} \sum_{i=1}^{n} c_i$ where $c_i = \begin{cases} 1, &amp; \text{if } y_i \in [L_i, U_i] \ 0, &amp; \text{if } y_i \notin [L_i, U_i] \end{cases}$</td>
</tr>
<tr>
<td>MPIW</td>
<td>$\frac{1}{n_{\text{test}}} \sum (U_i - L_i)$</td>
</tr>
<tr>
<td>CWC</td>
<td>$NMPIW(1 + \gamma(PICP)e^{-\eta(PICP-\mu)})$ where $c_i = \begin{cases} 0, &amp; PICP \geq \mu \ 1, &amp; PICP &lt; \mu \end{cases}$</td>
</tr>
</tbody>
</table>

This degree project also investigates the possibility of obtaining uncertainty estimates for the forecasts. The term prediction intervals will not be used as none of the methods include all uncertainty estimates required for a prediction interval. For constructing uncertainty intervals two methods will be used, bootstrapping residuals and dropout. Bootstrap is chosen due to being a commonly used method to obtain prediction intervals. It is a conceptually simple method and is also used for other forecasting models than neural networks when the prediction intervals cannot be obtained analytically. The choice of bootstrapping residuals instead of bootstrapping pairs is due to the temporal dependencies time series data has, which bootstrapping pairs would destroy.

Dropout is a method which has a strong theoretical framework. It can be shown to be an approximation of the posterior distribution from which we can obtain the prediction intervals. However, dropout is a far simpler estimation of the posterior distribution than creating a Bayesian network and use the full Bayesian framework. The dropout will be implemented in line with the theory above except for the part estimating the training data uncertainty with an autoencoder. The implementation of an autoencoder for a CNN was considered out of scope for this project.

The use of these two methods to create uncertainty estimates using CNN with dilated causal convolution is not a well studied area. This degree project can contribute with an evaluation and a comparison between these two meth-
ods for constructing uncertainty estimations and evaluating their quality.

Bootstrapping residuals has been less investigated than bootstrapping pairs when it comes to constructing prediction intervals for neural networks. But it is interesting in the case of time series data as bootstrapping pairs is less suitable for dependent data.
Chapter 3

Methods

The aim of this project was to implement and compare two methods for estimating uncertainty intervals when a neural network was used as forecasting model. The following chapter provides a description of the necessary steps to achieve this.

3.1 Pre-processing of data

The forecasts are made using daily sales volume data and corresponding product price with a timespan of almost three years. The data set contains three products from the same product category and the products are non-disposable products which usually are used for several years by the consumers. The data contains both yearly and weekly seasonality. There are also some peaks in the data at holidays. For training purposes missing data points in the data set were imputed, for test data they were excluded in the evaluation. The price was first imputed by taking the previous value in the time series. After the sales volumes were imputed employing the following rules.

- If the price is the same as last week, the value from the same day previous week is used.
- If the price is not the same, the value from the previous day is imputed.

The data was then standardised to have standard deviation one and mean zero.

To be able to use the data for training the CNN the data was divided into samples of input data and labels. The input data used was a time series with 512 time-steps, with both price and volumes. For the sales volumes, $v$, 512 values back in time from the forecast origin were used, i.e. $v_{t-512}, \ldots, v_t$ where
Figure 3.1: Data divided into train, validation and test set. The different validations and tests points are then made using rolling origin one each block.

$t$ marks the forecast origin. The price, $p$, also used 512 values but shifted one day ahead so the price at the predicted day also was fed as input, i.e. $p_{t-511}, \ldots, p_{t+1}$. As label the sales volume of predicted day was used, i.e. $v_{t+1}$.

### 3.2 Time series predictions

The time series predictions were evaluated on three different products within the same product category. The predictions were made in a recursive manner to obtain a forecast of 120 days. For each of the three products a hyperparameter search was done on a validation set and then the final performance was measured on a test set (figure 3.1). Both the validation and test were made using rolling origin. The predictions were evaluated with sMAPE, defined in Table 2.1.

#### 3.2.1 Dilated causal convolutional network

The architecture of the CNN is shown in figure 3.2. The input is fed into two branches, one for the sales volumes and one for the price. Each branch consists of a CNN. The two CNNs have nine layers each and a dilated causal convolution is performed with dilation rate $2^i$, where $i = 0, 1, \ldots, 8$. Also in each CNN, residual connections are applied as visualised in figure 3.2. After the nine convolutional layers the two branches are merged together and passed as input to two fully connected layers. The structure of the network is inspired by resNet [15].

The network does predictions one time step ahead given an input sequence. To obtain prediction sequences of the desired length, 120 days, forecasts are performed recursively.
Table 3.1: Values which the parameter search spans.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>epochs</td>
<td>10, 25, 50, 75</td>
</tr>
<tr>
<td>filters</td>
<td>8, 16, 32</td>
</tr>
</tbody>
</table>

When the architecture of the network was decided a search for good hyper-parameters was made on a validation data set. The search was limited to two hyper-parameters, the number of filters for the convolutional layers and the number of epochs which the network should be trained for. The hyper-parameter search was done on five different forecast origins and ten random initialisations for each forecast origin, so in total 50 validation points were used. The values which the hyper-parameter search spanned are presented in Table 3.1. As optimiser, Adam [21] was used with the default parameters in Keras [22].

The final results were produced on a test set with a sliding window of 120 days as prediction period. The forecast origin was moved one day at the time, 100 times, producing 100 test points. For each forecast origin the network was retrained. The test set was the last 220 days in the time series.

### 3.2.2 Naïve model

A naïve model was implemented as a baseline to the predictions from the convolutional neural network. The naïve model was implemented by taking last year’s values and shifting them one day to compensate for the strong weekly seasonality in the data. The idea to this model came from the strong yearly seasonality in the data.

### 3.3 Uncertainty intervals

The uncertainty estimations were performed on the network described in the previous section. They were evaluated on the same three products as the CNN. The number of filters and epochs chosen for each product in the previous section is used in this section when training the networks. In this section two methods are described, bootstrap residuals and dropout. With both methods the goal is to calculate prediction intervals with a 95% confidence level. To evaluate the prediction intervals the PICP and MPIW were calculated.
Figure 3.2: Architecture of the convolutional neural network. All convolutional layers have the same number of dimensions to allow residual connections. The outputs from the two branches are concatenated before being fed into the dense layers. Let $n$ be the batch size and $k$ the number of past values. The dimension of the input to the first convolutional layer is then $n \times k \times 1$. The input to the first dense layer is $n \times k \times 64$ and the input to the second layer is $n \times k \times 128$. 
3.3.1 Bootstrap

For constructing uncertainty intervals with bootstrapping residuals, the number of times bootstrap should be performed was first investigated. This was made using a validation data set. For five forecast origins in the validation set, 200 new data sets were created using bootstrapping residuals. These data sets were then used to train 200 CNNs which later all made a prediction for each of the 5 forecast origins, the small amount of data points was due to the long computational time. These new data sets were also used to estimate the variance of error, by forming a new data set, \( D_{r,2} \) defined in Equation 2.13 at page 15.

From the 200 trained CNNs, 20, 30, 40 and so on up to 200 networks were used to investigate the effect of the number of times bootstrapping was performed. The networks were chosen from the set of 200 instead of retraining the CNNs for each subset to save time in the search for parameters. For each subset the mean and variance of the prediction was calculated and the data set \( D_{r,2} \) was created. The mean of the prediction becomes the prediction of the bootstrap model and the variance the model misspecification. The data set \( D_{r,2} \) was used to train a network with the same architecture as the 200 previously trained networks but with a different cost function, equation 2.11. This network was used to predict the error variance.

The final test was made on 50 test points and bootstrapping 20 times.

3.3.2 Dropout

To preform uncertainty intervals with dropout there were two hyper-parameters which had to be decided, the dropout rate applied and the number of stochastic forward passes. The parameter search spanned the values shown in Table 3.2. The dropout rate is in this project defined as the fraction of nodes for which the weights are set to zero. When searching for hyper-parameters first a CNN, with dropout applied to each of the two dense layers, was trained. Later 1000 stochastic forward passes were made and from these 1000 stochastic forward passes, the first 100 points were used and the mean and variance were calculated according to equation 2.5 and equation 2.6. A held-out validation set was also used to estimate the inherent noise. This procedure was then repeated for the first 200 points, 300 points and so on up to 1000 points. The CNN was then retrained with a different dropout rate, to test the effect of the amount of dropout.
Table 3.2: Values which the parameter search for dropout for uncertainties intervals spans.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>dropout rate</td>
<td>0.1, 0.2, 0.5, 0.8</td>
</tr>
<tr>
<td>number of stochastic forward passed</td>
<td>100, 200, 300, 400, 500, 600, 700, 800, 900, 100</td>
</tr>
</tbody>
</table>

3.4 Implementation

The implementation was made in Python. To build and train the convolutional neural network Keras [22] was used. Additionally, NumPy [23] and Pandas [24] were used in the implementation. To visualise the data and results Matplotlib [25] in Python was used.
Chapter 4

Results

4.1 Time series predictions

In this section the accuracies for both the naïve model and the CNN are presented. The CNN is forecasting 120 days each time, which creates one validation or test point. When several validations or test points should be produced a sliding window of 120 days is moved over the validation or test data. If nothing else is mentioned the sliding window is moved forward one day at the time.

There was no clear difference in the accuracy of the first day and the last day, day 120, of the output from the networks. Therefore, the accuracy of a validation or test point is calculated as a mean of the accuracy over the 120 days. In figure 4.1 an example from each model is shown.

4.1.1 Predictions using dilated causal convolutional neural network

In figure 4.2 the result of the hyper-parameter search is shown. Both the number of filters and the number of epochs for which the network was trained were varied to obtain different models. The variance within each model is quite large for all the three products. The high variance shows that the different random initialisations and time windows give very different results. In the forecast examples given in figure 4.1 it can be seen that models do not succeed in catching the large peaks which gives a higher error for the time windows containing such a peak. The large models with nine convolutional layers and the small amount of training data probably also contribute to a high variance. Which combination of parameters that gives the lowest median sMAPE varies between products. In table 4.1 the chosen parameters for each product are
Figure 4.1: Forecast examples for product A. The missing values in the labels are due to that these are not imputed in the test data.
Table 4.1: Parameters chosen for the three products.

<table>
<thead>
<tr>
<th>product</th>
<th>number of filters</th>
<th>number of epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>B</td>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>C</td>
<td>16</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 4.2: The mean of sMAPE and standard deviation of sMAPE for the CNN. 100 test points were used for each of the three products.

<table>
<thead>
<tr>
<th>product</th>
<th>sMAPE mean (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>16.39 ± 3.28</td>
</tr>
<tr>
<td>B</td>
<td>19.70 ± 3.45</td>
</tr>
<tr>
<td>C</td>
<td>31.06 ± 14.99</td>
</tr>
</tbody>
</table>

In table 4.2, the accuracy on the test set is shown. Here product C has a considerably higher error than the other products. This is due to that product C has a time window where the sales volumes in the test data have a very distinct pattern from the training data. Hence the error becomes very large on the points containing this time window.

4.1.2 Naïve model

In Table 4.3, the mean for sMAPE of the 100 test points is shown together with the standard derivation. Product C has a higher sMAPE than the other two products, as for the CNN. It is interesting to notice how the naïve model performs as good as the CNN.

Table 4.3: The mean of sMAPE and standard deviation of sMAPE for the naïve model. 100 test points were used for each of the three products.

<table>
<thead>
<tr>
<th>Product</th>
<th>sMAPE mean (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>14.9 ± 2.8</td>
</tr>
<tr>
<td>B</td>
<td>18.1 ± 5.9</td>
</tr>
<tr>
<td>C</td>
<td>27.7 ± 8.1</td>
</tr>
</tbody>
</table>
Figure 4.2: Boxplot of the different models in the parameter search for each product. 50 validation points were used for each model. The model with the lowest median is marked with asterisk.
Table 4.4: Results on the test set of 50 points for uncertainty intervals constructed with both parameter uncertainty and inherent noise uncertainty.

<table>
<thead>
<tr>
<th>Product</th>
<th>sMAPE mean</th>
<th>PICP</th>
<th>MPIW</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>19.8 ± 2.9</td>
<td>0.80 ± 0.06</td>
<td>2.0 ± 0.4</td>
</tr>
<tr>
<td>B</td>
<td>19.7 ± 5.1</td>
<td>0.63 ± 0.14</td>
<td>1.9 ± 0.3</td>
</tr>
<tr>
<td>C</td>
<td>25.9 ± 15.5</td>
<td>0.86 ± 0.10</td>
<td>5.9 ± 6.0</td>
</tr>
</tbody>
</table>

### 4.2 Uncertainty intervals

#### 4.2.1 Bootstrap

In figure 4.3 and figure 4.4, both the PICP and MPIW for different number of bootstraps are shown. Neither of them shows a clear trend for any product and from these results no number of bootstrap can be said to differ from the others. Therefore, the number of bootstraps are chosen based on the lowest computational cost, i.e. as few bootstraps as possible. For all products it was chosen to perform 20 bootstraps. The accuracy of the predictions did not vary with the number of bootstraps for any of the products.

Table 4.4 shows the results for the estimated uncertainty intervals using bootstraps. One should note that the time window with higher error for the CNN is not included in the test set for bootstrap and therefore the error of product C lies closer to the other two products. Looking at the examples of the forecasts shown in figure 4.5 it can be seen that the intervals width varies within the time window but seems to fail in covering the high peaks.

#### 4.2.2 Dropout

Figure 4.6 and figure 4.8 show the PICP and the MPIW in the hyper-parameter search. A dropout rate of 0.8 gave the highest PICP for all three products and, not surprisingly, also the broadest uncertainty intervals for all products. It is interesting to notice how such a high dropout rate, 80% of the nodes in the two dense layers are dropped, still does not impact the error of the forecast. The relatively constant forecast error for the different dropout rates can be seen in figure 4.7. The same parameters were chosen for all three products, dropout rate 0.8 and 100 stochastic forward passes. It is convenient to choose a small number of stochastic forward passes due to being computationally lighter.

The results of dropout are shown in table 4.5. Comparing the dropout
Figure 4.3: PICP plotted for different numbers of bootstraps. The validation set consist of five forecast origins.
Figure 4.4: MPIW plotted for different numbers of bootstraps. The validation set five forecast origins.
Figure 4.5: One forecast with uncertainty intervals for each product constructed with bootstrap. The shaded area marks the uncertainty intervals. The sales volumes are standardised. Missing values in the labels are not imputed due to that the prediction is made on test data.
Table 4.5: Result for dropout on 100 test points.

<table>
<thead>
<tr>
<th>Product</th>
<th>sMAPE mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>15.3 ± 3.0</td>
</tr>
<tr>
<td>B</td>
<td>13.3 ± 1.2</td>
</tr>
<tr>
<td>C</td>
<td>24.5 ± 8.3</td>
</tr>
</tbody>
</table>

Table 4.5: Result for dropout on 100 test points.

| Product | PICP 
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.50 ± 0.06</td>
</tr>
<tr>
<td>B</td>
<td>0.40 ± 0.06</td>
</tr>
<tr>
<td>C</td>
<td>0.38 ± 0.20</td>
</tr>
</tbody>
</table>

Table 4.6: Result for dropout with 50 test points

<table>
<thead>
<tr>
<th>Product</th>
<th>sMAPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>17.9 ± 1.8</td>
</tr>
<tr>
<td>B</td>
<td>13.7 ± 1.9</td>
</tr>
<tr>
<td>C</td>
<td>17.7 ± 6.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>PICP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.50 ± 0.04</td>
</tr>
<tr>
<td>B</td>
<td>0.43 ± 0.05</td>
</tr>
<tr>
<td>C</td>
<td>0.51 ± 0.20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>MPIW</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.80 ± 0.11</td>
</tr>
<tr>
<td>B</td>
<td>1.0 ± 0.1</td>
</tr>
<tr>
<td>C</td>
<td>1.8 ± 1.1</td>
</tr>
</tbody>
</table>

accuracy to the accuracy of the CNN with just a point prediction the accuracies are quite similar for product A, but slightly lower and has less variance for the other two products. It can however be seen in figure 4.9 that product C is struggling with the accuracy in the same area as the CNN. Table 4.6 is made on the same test set as the results of bootstrap and is included for a fair comparison between the two methods.

Comparing the PICP and MPIW with bootstrap we see that bootstrap has considerably higher PICP if the inherent noise is considered. The uncertainty intervals constructed with bootstrap also is wider for all products.
Figure 4.6: Boxplot of MPIW for different dropout rates and number of stochastic forward passes. On the x-axis the number of stochastic forward passes is marked and there is one column for each dropout rate. The variance was calculated with 10 points.
Figure 4.7: Plot of sMAPE versus number of stochastic forward passes show for each level of dropout. 10 forecast origins were used.
Figure 4.8: Boxplot of MPIW for different dropout rates and number of stochastic forward passes. On the x-axis the number of stochastic forward passes is marked and there is one column for each dropout rate. The variance was calculated with 10 points.
Figure 4.9: One forecast with uncertainty intervals for each product constructed with dropout. The shaded area marks the uncertainty intervals. The sales volumes are standardised. Missing values in the labels are not imputed due to that the prediction is made on test data.
In this degree project the possibility of creating a system able to make both point forecasts and uncertainty estimates using neural networks was investigated. Two methods for creating uncertainty intervals have been tested, bootstrapping residuals and dropout. When building the models, the effect of their different hyper-parameters was explored. Later the bootstrap method was tested on a test set with 50 points and dropout on a test set with 100 points. Both the methods were tested on three products from the same product category.

5.1 Forecasts using dilated causal convolutional neural network

The first interesting finding is that the accuracy of the CNN, only concerning the point forecast, is no better than the naïve approach of just taking last year’s values. This could be due to that the naïve model captures the high peaks of holidays very well which the CNN often fail to catch, an example of this is shown in figure 4.1. In forecasting the high peaks, the naïve model seems to have an advantage. However, the CNN takes price as input and it could be interesting to investigate how this affects the accuracy. Could the CNN, for example, work as an interactive tool which gives the future sales volumes depending on the price?

Another future possibility to improve the performance of the network is to have some input to the network which tells the network if it is a holiday or some other special day it should forecast. This more direct way of account for seasonality could improve the performance as the three products show strong
seasonality. It is this strong seasonality which favours the naïve approach.

5.2 Uncertainty intervals

Bootstrapping residuals and dropout, respectively, were used to produce forecasts with uncertainty estimates. Both methods consist of varying the model which produces the forecasts such that a distribution of forecasts can be made. From the distribution the variance is calculated to estimate the uncertainty. The Bootstrap method consists of varying the model by training the network 20 times using different data sets and the dropout method by doing 100 stochastic forward passes. Both models also have a part which estimates the inherent noise in the data. Looking at the results from both methods none of them succeeded in creating a 95% prediction interval as was aimed for. Bootstrap had the highest coverage probability of 88% for product C. The highest coverage probability for dropout was 50% for product A when 100 test points were used. When instead only 50 test points were used, the same 50 point as for bootstrap, both products A and C had a coverage probability around 50%. The reason for product C’s high increase in coverage probability is that when only 50 points are included the time window with lowest accuracy is not included. This time window can be seen in figure 4.9c where both the accuracy and the coverage probability are very low.

The example in figure 4.9c shows a property of both methods, if the prediction accuracy is low, the coverage probability of uncertainty intervals for this section also becomes very poor. This can also be seen in other products and for bootstrap. An example of this behaviour can be seen in figure 4.5a around the peak. This might be a drawback with both methods, it is hard to create prediction intervals wider than the flexibility in the network chosen. The inherent noise part helps with this to some extent, but obviously not enough in this setup. However, one should keep in mind when looking at the dropout results is that the autoencoder part of the method which was used by Zhu and Laptev [18] was left out in this project. A similar set up for the CNN might have improved the coverages probability. For example, such an estimate of the uncertainty of how representative the training data is might have given a higher coverage probability for product C were the pattern in test data differs from the training data, shown in figure 4.9c.

Neither of the uncertainty intervals grow with an increasing forecast horizon as one might expect. It is worth noticing that none of the methods have any concept of how far into the future they are constructing uncertainty intervals. However, the most likely reason for the uncertainty estimates not growing with
time, the forecast origin is not that long and the error of the forecast did not vary noticeable from day one of the forecast to day 120. Even if the uncertainty intervals did not vary with time we can see that the methods seem capable of adjusting the uncertainty for different parts of the data, which is seen clearly in figure 4.5a.

Comparing the total computational time for the two methods, bootstrap is the slowest as in this case required 21 networks to be trained. But comparing the time to make a prediction, when all the networks are trained, bootstrap is faster as it only requires 21 predictions compared to 100 for dropout.

## 5.2.1 Bootstrap

Comparing the results produced with bootstrap with the results of Khosravi et al. [5], who created bootstrap prediction intervals using bootstrapping pairs, the PICP of this project is lower. Khosravi et al. [5] presented an average PICP around 0.90-0.95, while the averages over all three products in this project is 0.76. One should note, while comparing these studies that this project aimed at a confidence level of 95% while Khosravi et al. [5] aimed at a confidence level of 90% and that bootstrapping pairs was used and not bootstrapping residuals. Bootstrapping pairs is more robust if the model is misspecified [19] which might lead to more uncertainty being captured than when using bootstrapping residuals.

Regarding the parameter search of bootstrap, it was made using very few validation points, only five forecast origins for each product. This was due to the limited time for the project. However, the pattern is the same for all products. The way the different sets of number of bootstraps were made, i.e. incrementing the number of points with then from 20 to 200, for each product was not optimal either, due to the dependencies which occurs between the different sets.

## 5.2.2 Dropout

Zhu and Laptev [18] presented a mean PICP of 0.95 with the inherent noise, parameter uncertainty and the training data uncertainty included and 0.78 if only stochastic forward passes were used to estimate the uncertainty. This can be compared to the mean PICP of all three products at 0.43 in this project were the inherent noise and parameter uncertainty were included. The considerably lower result, even compared to only stochastic forward passes, can be due to the much lower sMAPE of 5.9%, presented in [18], compared to 18% in mean
for the three products. Lower sMAPE would be beneficial as the uncertainty estimates are very centred around the predictions, so when the prediction is far from the labels the coverage probability becomes low.

The parameter search for dropout showed that the number of stochastic forward passes had no effect. The subsets were created in the same way as for bootstrap method and the same criticism apply, that the different subsets not really are independent.

Increasing the dropout rate had the effect of increasing the PICP of the uncertainty estimates. In this degree project the dropout rate with the highest PICP was chosen. However, one could argue that, for at least product B and C, the difference in PICP between dropout rate 0.5 and 0.8 was very small and the MPIW was higher for dropout rate 0.8. The question is if the small increase in PICP median is worth the increase in MPIW. A dropout probability of 0.8 is also a very high dropout rate and seldom used. There could be several reasons why this high dropout rate works so well and keeps the accuracy of the network. The first is that dropout is applied to the two dense layers and not on the convolutional layers, i.e. there is only a small fraction of the network which is affected by dropout. This is how dropout normally is applied in convolutional neural networks and was therefore chosen. The second reason could be that the network is too large and a high dropout rate works well because the whole network is not needed to fit the data. The slightly better result for product C in terms of sMAPE, seen when comparing table 4.5 and table 4.2 or table 4.6 and table 4.4 indicates that the network is a bit too large and needs some regularisation or decreased size.

5.3 Ethics and environmental impact

To be able to plan and take the risk into consideration helps to optimize the resources. This can reduce the number of unused units produced and thus reduce the environmental impact.

The system in this degree project is intended to work as a guiding system for human analyst and is therefore not considered to replace them. Hence this project will not, in its current form threaten to replace human workers which then would be unemployed. However, this is always a risk as prognosis systems become more efficient fewer human experts are needed to carry out the same amount of work.

To build an intelligent system that in an ethical way it is important with trust, fairness and honesty. This degree project can, in the broader perspective, contribute to trust through its estimation of uncertainty. Algorithms which
are explicit their uncertainty and limitations are important to build algorithms which we can trust. If an algorithm can be uncertain it is, for example, possible for the algorithm to pass on cases it is uncertain about to a human [26].

### 5.4 Future work

For future work it would be interesting to investigate further how the uncertainty intervals perform on other data sets to be able to draw more general conclusions. This could for example be on other products. When doing this it would be interesting to note how individual the hyper-parameter tuning is to each product.

It would also be interesting to extend the dropout method with an estimate of the training data uncertainty as Zhu and Laptev [18] have done with an autoencoder.

In this project the mean of each time-step was fed into the network when doing the prediction for the next time-step for both bootstrap and dropout. If instead a prediction with each point in the distribution created a time-step were made the uncertainty estimation might grow with the time. This could be an interesting future investigation.

In terms of further development of the forecasting network, it would be interesting to try a smaller structure to see if that could speed up the procedure of training and lead to better forecasts. The current size was chosen to span at least a year back in time, in fact it spans 512 days. Maybe so many past values are not necessary in order to obtain good predictions?
Chapter 6

Conclusions

The main focus of this degree project has been to investigate which of bootstrapping residuals and dropout performed best to estimate uncertainty intervals for sales volumes forecasts using a CNN. In addition, the accuracy of the neural network has been evaluated.

The bootstrap method performed better with a coverage probability of 76% compared with 43% for dropout. However, it should be kept in mind the part estimating the training data uncertainty in dropout is not included in this project, which could increase the coverage probability for dropout.

In general, the results indicate that it is possible to estimate the uncertainty in the forecast to some extent with both bootstrap and dropout. However, none of the methods succeeded in creating an interval with a 95% in confidence level. The methods were only tested on one type of network structure and three products from the same category, therefore it is hard to draw any general conclusions regarding if bootstrap or dropout is the more suitable method for estimating uncertainty intervals.

The network generating the predictions seems to have a too large structure for the task and amount of data available. This conclusion is drawn based on that such a high dropout rate still produced a good accuracy. As future work it would be interesting to add the training data uncertainty to the prediction intervals made with dropout using an autoencoder as Zhu and Laptev [18].
Bibliography


[22] François Chollet et al. *Keras*. [https://keras.io](https://keras.io). 2015.


