Aalto University
School of Business

MACHINE LEARNING FOR INVENTORY MANAGEMENT

Forecasting Demand Quantiles of Perishable Products with a Neural Network

Master's Thesis
Philip Ziegler
Aalto University School of Business
Information and Service Management
Fall 2020

Aalto University
School of Business

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Philip Ziegler

Title of thesis Machine Learning for Inventory Management – Forecasting Demand Quantiles of Perishable Products with a Neural Network

Degree Master of Science (MSc)

Degree programme Information and Service Management

Thesis advisor(s) Pekka Malo, Timo Kuosmanen, Andreas Fügener

Year of approval 2020 **Number of pages** 92 **Language** English

Abstract

Accurate demand forecasting is a crucial component in building an efficient supply chain. Forecasting is a major determinant of inventory cost. Several methods and models for forecasting have been studied extensively over the last decades. In recent years, there has been a growing interest in the capabilities of Machine Learning algorithms in forecasting, and specifically in Neural Network models. Despite the expanding research on forecasting with Neural Networks, there have been only few studies focusing on the specific ramifications for forecasting demand of perishable products at the Stock Keeping Unit (SKU) level.

Forecasting SKU-level demand for perishable products is a challenging task: time series for demand are volatile, skewed, subject to external factors, and frequently consist of only a few observations. Furthermore, SKU-level demand forecasts are typically used for inventory management, which imposes additional requirements on the forecasting procedure. This study examines how to design Neural Networks that address the specific ramifications of inventory management for several thousand SKUs.

This work identifies central issues in the field and compiles successful approaches to overcome them. Next, a Neural Network architecture is suggested that takes these special requirements into account, building on insights from the literature. Namely, it learns from multiple hundred time series, incorporates external data into the prediction, and provides quantile forecasts of cumulative demand. In a large-scale experiment, the model forecasted the demand for several hundred SKUs in the fresh product segment of a German wholesale company. These forecasts were subsequently used for simulating the inventory development at the company for three months under close-to-real-life conditions.

This study shows that Neural Networks are a promising approach to deal with large-scale forecasting problems for perishable products. The main finding of this study is that within the experimental setting, the base form of the suggested model for accurate daily demand forecasting yielded superior results to an array of competing baselines. In terms of inventory performance, the results are mixed, but present exciting directions for further research.

Keywords Neural Networks, Demand Forecasting, Inventory Management

Acknowledgements

Without exaggerating I can say that crafting this thesis was probably my most challenging project to date. It would not have succeeded without the contributions of others.

Firstly, I would like to thank the responsible people at Lekkerland SE & Co.KG for their cooperation on this project. In particular, I would like to thank Dr. Maximilian Burkhardt, Florian Kopp, and Nathalie Schneider for supervising my thesis thoroughly and always responding quickly to all my questions and concerns.

Aalto University's Prof. Pekka Malo, Ph.D., and Prof. Timo Kuosmanen, Ph.D. as well as Prof. Dr. Andreas Fügener from the University of Cologne provided academic guidance in a field of research that seemed overwhelming at first.

Finally, I cannot express my gratitude to my dear friends Wen Yap and Florian Ebbing in words; for hours of working together, discussing ideas, proofreading, and emotional support as well, when the finish line seemed far away. I would not have been able to do this on my own. Thank you.

TABLE OF FIGURES

LIST OF ABBREVIATIONS

AIC	Akaike Information Criterion
ARIMA	Autoregressive Integrated Moving Average model
CSL	Cycle Service Level, α-Service Level
EWQR	Exponentially Weighted Quantile Regression
LSTM	Long short-term Memory Network
MAE	Mean Absolute Error
MIMO	Multi-Input Multi-Output strategy to forecasting
ML	Machine Learning
MLP	Multi-Layer Perceptron / feedforward Neural Network
NN	Neural Network
QR	Qauntile Regression
RNN	Recurrent Neural Network
SARIMA	Seasonal Autoregressive Integrated Moving Average model
SARIMAX	SARIMA model with exogeneous variables
Seq2Seq	Sequence-to-sequence modelling
Seq2Quant	Sequence to-Quantile Regression model
SKU	Stock-Keeping Unit
SMAPE	Symmetric Absolute Mean Percentage Error

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Accurate demand forecasting is a crucial component in building an efficient supply chain. Forecasting is a major determinant of inventory cost, service levels, scheduling and staffing efficiency, and forecasting errors contribute to the bullwhip effect (Gardner, 2006). The more accurately a company can predict demand, the less uncertainty it faces, and the better it can adapt to future developments. This especially true in the case of forecasting perishable products, where inaccurate forecasts lead to waste.

Several methods and models for forecasting have been studied extensively over the last decades. In recent years, there has been a growing interest in the capabilities of Machine Learning algorithms (ML) in forecasting, and specifically in Neural Network models. Paraphrasing Mitchel's (1997, as cited in Goodfellow et al., 2016, p. 99) definition, Machine Learning involves a computer learning through experience to perform a set of tasks with increasingly better performance. Though the first research on Machine Learning dates back to the 1950s, the field has seen a surge in popularity over the last decade.

Advancements in computational capacity and data availability have made the training of deep, complex Neural Networks feasible and have led to breakthroughs in, e.g., image recognition and natural language processing. Neural Networks won the ImageNet Competition (Krizhevsky et al., 2012) and beat the world champion in the game of Go (Silver et al., 2016), which was long deemed unimaginable due to the complexity of the game. These breakthroughs have sparked the interest of forecasting researchers and created a need to explore the potential of Neural Networks in time series prediction.

There is a well-established body of literature tackling the problem of forecasting. Many standard procedures rely on constructing a mathematical model of the time series with statistical methods. These statistical models make assumptions about the underlying data-generating process. Other than statistical models, Neural Networks make no *a priori* assumption but infer the underlying process from data. Another attractive property of Neural Network models is their ability to model complex non-linear relationships between variables. In contrast, traditional statistical models usually assume linear relationships between variables.

Numerous papers have investigated the performance of Neural Networks for time series prediction. Though promising in theory, only a few studies found evidence to support the belief that Neural Networks outperform statistical models. The opposite is often the case (c.f. Carbonneau et al., 2008; Makridakis et al., 2018a). Considering the mixed results, the performance of machine learning algorithms in time series prediction is unlike the anticipated breakthrough they had in image recognition or Natural Language Processing. (Makridakis et al., 2018a). On top of that, even when Neural Networks succeed, their performance gains must be weighed against their fundamental drawbacks: They are computationally expensive and provide little insight, as they are black box models. (Carbonneau et al., 2008)

Despite the expanding research on machine learning algorithms in forecasting, there have been few studies focusing on the specific ramifications for forecasting demand of perishable products at the Stock Keeping Unit (SKU) level. This study examines how to design Neural Networks that address the specific ramifications of inventory management for several thousand SKUs.

It seeks to answer the following questions:

1. What are the specific ramifications for forecasting SKU-level demand of perishable products?
2. How can the accuracy of SKU-level demand forecasts of perishable products be improved by using Neural Networks?
3. Do accuracy improvements in forecasting accuracy translate to improved utility values in inventory management under real-life restrictions?

With regards to question 1, this work reviews the literature on SKU-level demand forecasting to identify central issues and compile successful approaches to overcome them. Next, a Neural Network architecture is suggested that takes these special requirements into account, building on insights from the literature. Namely, it learns from multiple hundred time series, incorporates external data into the prediction, and provides quantile forecasts of cumulative demand. A subsequent experiment empirically evaluates the performance of the suggested model; the model forecasts the demand for several hundred SKUs in the fresh product segment for Lekkerland SE & Co. KG, a German wholesale company. To answer question 3, the inventory development of these SKUs will be simulated for three months. The model will be compared against a statistical baseline model to quantify the performance gain of using a more complex model, given there is one.

This thesis is divided into seven chapters: This introductory Chapter 1 has described the need for further investigation on Neural Network models for demand forecasting in wholesale. Chapter 2 provides theoretical foundations on forecasting and Neural Networks. Chapter 3 reviews the literature on Neural Networks forecasting and introduces key issues of SKU-level demand forecasting and approaches to solve them. Building on the existing literature, Chapter 4 introduces a theoretical framework for forecasting under the specific requirements of demand forecasting. For the subsequent empirical part of the

thesis, the data from the case company is described and explored in Chapter 5. This chapter also outlines the process of designing and training the models, generating the forecasts based on the data, and finally, simulating the inventory development. Chapter 6 presents the findings from the simulation and analyses the performance of the model for different product groups and service levels. Chapter 7 discusses the findings and reconciles them with the results of previous studies. It presents their implications for practice, as well as their limitations, and show directions for further research.

2 THEORETICAL FOUNDATIONS

The purpose of this chapter is to introduce central concepts for inventory management, forecasting and Neural Networks. The chapter is structured as follows: Section 2.1 introduces a classic inventory management policy and illustrates the need for forecasting. Section 2.2 defines forecasting as a time series prediction problem and presents an overview of traditional forecasting methods. Chapter 2.3 provides theoretical foundations of Neural Networks.

2.1 INVENTORY MANAGEMENT

Inventory Management is one of the most important and well-developed areas of Operations Management. Inventory management determines the amount and timing of orders for products and materials. The major reasons for keeping inventory are economies of scale, protection against demand fluctuations, and long lead times. (Thonemann, 2010, pp. 194–195)

The two most common inventory management policies are continuous review policies and periodic review policies. This section will focus exclusively on the latter one, which arguably resembles the conditions for inventory management of highly perishable products best. The key idea of periodic review policy is to check demand at regular intervals, and to place orders to fill the gap between the current stock and the pre-defined target stock.

The notation used is as follows:

R	Review Period length, alternatively: the time between two orders
r	Product Lead Time
S	Order-up-to level
I_t	Inventory at the beginning of period t
X_t	Order quantity in period t
PS_t	Pipeline stock of period t, i.e. orders that have not yet arrived
F_{R+r}	Distribution of demand during the replenishment time
μ_{R+r}	Expected demand during the replenishment time
σ_{R+r}	Standard-deviation of demand during the replenishment time
α	Target Cycle Service Level (CSL), i.e. probability of filling the full demand of a period

Under periodic review, the inventory I of a product is checked every *R* periods. Orders are then placed, such that the order-up-to-level *S* is reached. Orders of previous periods that have not yet arrived, and the pipeline stock *PS*, are therefore considered as well. The order quantity of period t is described in Equation 1. (Thonemann, 2010, p. 224)

$$X_t = S - I_t - PS_t \qquad \textit{Eq. 1}$$

The basic assumption of this policy is that demand is stochastic. Given a target cycle service level α, the optimal order-up-to-level S^* covers the demand during the replenishment time with probability α. The replenishment time has the length *R*+*r*. It describes the lead time of a product r plus the time until the next order arrives, which is equal to the review period *R*. $S^*$is found through solving the optimization problem:

$$\begin{aligned} &\min S \\ &s.t. F_{R+r}(S) \geq \alpha \end{aligned} \qquad \textit{Eq. 2}$$

Under the assumption of normally distributed demand with a constant mean and standard deviation, where demand of all periods is independent, *S** can be computed directly. (Thonemann, 2010, p. 232) It consists of the expected demand during the replenishment time μ_{R+r} plus a safety stock, which is computed by the standard deviation

σ_{R+r} times a z-score that corresponds to the value of the cumulative normal distribution function at the target CSL α.

$$S^* = \mu_{R+r} + z * \sigma_{R+r} \quad \textit{Eq. 3}$$

$$z = F^{-1}(a) \quad \textit{Eq. 4}$$

Because of seasonal effects, and exogenous influences, it might be unrealistic to assume a constant demand distribution, let alone assuming normality. An alternative to estimating demand over the replenishment period is to predict the demand via forecasting. (Thonemann, 2010, pp. 251–253)

$$\mu_{R+r} = \sum_{i=1}^{R+r} \hat{y}_{t+i,t} \quad \textit{Eq. 5}$$

The uncertainty of demand is then directly linked to the forecast error, specifically its standard deviation, which can be estimated via the sample variance (Thonemann, 2010, p. 253). For estimating the forecasting error over multiple periods, typically two assumptions are made: forecast errors are normally distributed, and they are stochastically independent. Given these assumptions, the multi-period forecast error can be estimated as:

$$\hat{\sigma}^2 = \frac{1}{T-N} \sum_{t=1}^{T} (y_t - \hat{y_t})^2 \quad \textit{Eq. 6}$$

$$\hat{\sigma}_{R+r}^2 = (R+r)\hat{\sigma}^2 \quad \textit{Eq. 7}$$

2.2 FORECASTING

This chapter will frame forecasting as a time series prediction problem. A forecast $f_{t,k}$ is made in period t for a future period $t+k$. According to Brooks (2008, p. 247), forecasts

are conditional expectations of the value y the time series takes, given all information Ω available at period t. When referring to prediction, this thesis follows Brook's notation:

$$\hat{y}_{t+k,t} = f_{t,k} = E(y_{t+k}|\Omega_t) \qquad \textit{Eq. 8}$$

While qualitative forecasting methods exist, the research has been much more concerned with quantitative forecasting. Quantitative forecasting is applicable if (numerical) information about the past is available, and past patterns of the series are likely to continue. (R.J. Hyndman & Athanasopoulos, 2018)

Brooks (2008, p. 244) also distinguishes between forecasting with structural models and time series models. In a structured model, forecasts are made by relating the dependent variable y_t to one or more independent variables x_t^i. With knowledge or predictions of the independent variables x_{t+k}^i, a forecast for $y_{t+k}$is made. Structural models are popular in econometrics and, according to Brooks (2008, p. 244), often work well in the long run. However, they require a thorough understanding of the system, which may be hard in the first place. (R.J. Hyndman & Athanasopoulos, 2018)

Time series models, on the other hand, try to predict the development of the time series given its historical records. A time series is a row of observations of one or multiple variables at several points in time. (R.J. Hyndman & Athanasopoulos, 2018) For instance, in demand forecasting, observations could be historical records of sales. The distance between single observations determines the time series frequency. Common frequencies are, e.g. annual, monthly or daily observations. With the increasing pace of business cycles and automation, even more granular forecasts are becoming more popular, e.g. intraday-forecasting on an hourly basis. (Bandara et al., 2020).

Time series often exhibit certain patterns in their development over time, namely trends, seasonal and cyclic patterns. The following definitions, adapted from Hyndman & Athanasopoulos (2018), are applied for this thesis:

- A *trend* exists when there is a long-term increase or decrease in the data. It does not have to be linear […]
- A *seasonal* pattern occurs when a time series is affected by seasonal factors such as the time of the year or the day of the week. Seasonality is always of a fixed and known frequency [...]
- A *cycle* occurs when the data exhibit rises and falls that are not of a fixed frequency. These fluctuations are usually due to economic conditions, and are often related to the "business cycle". The duration of these fluctuations is usually at least two years.

In order to come up with an accurate forecast, a model needs to capture these patterns, though these might not suffice in explaining the series' variation on their own.

According to Hyndman & Athanasopoulos (2018), in a time series context, the forecasting horizon refers to how far into the future a forecast is made. The horizon does not not necessarily refer to an absolute measure of time, but to how many steps of the series are forecasted: Models can forecast for a single step or multiple steps ahead. A forecast $f_{t+1|t}$ is a one-step-ahead forecast, whereas a forecast $f_{t+k|t},\ t > 1$ is a multiple-step-ahead forecast. For example, given a time series of daily values, forecasting the demand for the upcoming week could be modelled as a seven-step ahead problem. In general, accuracy decreases with the length of the forecasting horizon (R.J. Hyndman & Athanasopoulos, 2018). In other words, the uncertainty associated with the forecast increases. The following section describes how to measure forecast accuracy.

The parameters and variables of a forecasting model are estimated based on past values of the time series to achieve a good fit, which means tuning the model such that the historical forecasts would be as accurate as possible. When forecasting future values under real-world conditions, this information would not be available.

It is thus common practice to not estimate parameters based on the whole time series sample but only a subset of it, and to evaluate its forecast accuracy on previously unseen data (Brooks, 2008, p. 245). The data is divided into a training and testing set. The training data is used to choose and estimate the parameters of the model. In contrast, the test data is used to evaluate the model's performance on previously unseen data to assess how well it generalizes. This two-step approach is crucial to prevent a model from overfitting. A model, given enough parameters, might achieve perfect accuracy on training data by memorizing the time series perfectly. Such a model would, however, perform poorly on unseen data and hence provide poor forecasts. In other words, it would not generalize well.

Forecasting errors are calculated based on out-of-sample forecasts.

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T} \qquad \textit{Eq. 9}$$

These can be applied both to single-step and multi-step forecasts. Depending on the sample size and the forecasting horizon, the number of forecast errors can be quite high.

Performance metrics are a way to summarize forecast errors in a single metric. Popular scale-dependent metrics are the Mean absolute error (MAE) and the Root Mean Squared Error (RMSE), as defined by the equations in horizon Hyndman & Athanasopoulos (2018):

$$\mathrm{MAE} = \mathrm{mean}(|\mathrm{e_t}|) \qquad \textit{Eq. 10}$$

$$RMSE = \sqrt{mean(e_t^2)} \qquad \textit{Eq. 11}$$

To compare forecasts over multiple time series of different scale, the percentage error p_t is used. The Mean Absolute Percentage Error (MAPE) is given by averaging the absolute percentage errors. This is not possible if the time series contains any zeros. Moreover, they penalize too small forecasts heavier than too high ones. To circumvent these limitations, Armstrong (1978, p. 384) suggested the symmetric MAPE (SMAPE):

$$MAPE = mean\left(\left|\frac{100e_t}{y_t}\right|\right) \qquad \textit{Eq. 12}$$

$$\mathrm{SMAPE} = \mathrm{mean}\left(\frac{200|\mathrm{y_t} - \hat{y}_t|}{\mathrm{y_t} + \hat{y}_t}\right) \qquad \textit{Eq. 13}$$

The SMAPE, unfortunately, also has its limitations when forecasting values close to zero, as the small denominator makes the calculation unstable. (R.J. Hyndman & Athanasopoulos, 2018) Concluding this section, each performance metric comes with strings attached. These limitations need to be accounted for when choosing a performance metric that is suitable for evaluating forecasts. There are several further measures found in the literature that try to overcome these limitations, but this overview is limited to the most popular measures.

2.3 TRADITIONAL FORECASTING METHODS

This subchapter introduces traditional forecasting methods. The term 'traditional' used herein does not refer to commonalities in the models, but rather serves to differentiate them from machine learning methods. The methods discussed here were (and still are) widely popular in research and practice.

2.3.1 Simple Heuristic Forecasting Methods

The methods presented here are widespread. The respective formulas are adapted from Hyndman & Athanasopoulos (2018).

A simple approach to forecasting given no other information is to assume, that there will not be any change in the generating process, i.e. the value of the next period is identical to the current one:

$$y_{t+k|t} = y_t \qquad \textit{Eq. 14}$$

The name of this approach, 'Naïve Forecasting', alludes to its simplicity. Despite the name, Naïve Forecasting is still a popular benchmark for other methods. It serves as a sanity check for whether more sophisticated methods provide actual insight. A variant that accounts for seasonal patterns in the data is the Naïve Seasonal Forecast, that forecasts the value of the same period in the last seasonal cycle.

$$y_{t+k|t} = y_{t+k-m(h+1)} \qquad \textit{Eq. 15}$$

It is self-evident that the naïve approach is prone to fallacies when facing a time series with a lot of variance. For example, a peak in the time series is not necessarily followed by another peak. To smoothen out periodic volatility, one might want to average the demand for past periods. This approach is referred to as the Simple Moving Average.

$$y_{t+k|t} = \frac{1}{n} * \sum_{i=\,t-n}^{t} y_i \qquad \textit{Eq. 16}$$

Averaging can likewise be applied to seasonal forecasting.

The heursitics presented here make no assumptions about the underlying process and thus are easily understandable and simple to implement. The next section describes a more sophisticated framework for time series analysis.

2.3.2 **Statistical Forecasting Methods**

Following Hyndman & Athanasopoulos' (2018) explanation, future values of the time series are unknown until they are observed, and are thus like random variables. The

random variables allow for many potential outcomes. As the true event approaches, the uncertainty about its potential outcomes decreases. This statistical perspective is at the core of the forecasting models introduced in this section. If the entire time series is a row of random variables, it can be understood as the realization of a stochastic process. Statistical forecasting models try to exploit the relationships between the individual random variables in the process, namely their covariance. The covariance of its series values with itself, based on their temporal difference, is called autocovariance.

Box & Jenkins (1970) developed the framework for time series analysis with Auto-regressive Integrated Moving-Average (ARIMA) models. The idea behind ARIMA models is to understand time series as a stochastic process, i.e. a sequence of random variables. ARIMA models find a linear representation of the stochastic process. They comprise Autoregressive and Moving Average parts. The formal definitions in this part are adapted from Hyndman & Athanasopoulos (2018)

And Autoregressive (AR) models explain the value of the time series at point t as a linear combination of lagged values of the time series, i.e. it is a regression model, that explains the value y_t by its past values. The autoregressive model of order p is given by:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t \qquad \textit{Eq. 17}$$

In this equation, c is a constant and ε_t is a white noise process.

Moving Average (MA) models model the time series as a result of past noise, i.e. past forecast errors. The value of y_t is a weighted moving average of past forecast errors. MA models must not be confused with the averaging heuristic. The MA model of order q is given by:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \qquad \textit{Eq. 18}$$

A model that incorporates both AR(p) and MA(q) components is called an ARMA(p,q) model. If differencing is applied to make the original series stationary, the model is an ARIMA(p,d,q), where d is the degree of first differencing applied. Differencing refers to subtracting every value of the series by its predecessor.

$$y'_t = \Delta y_t = y_t - y_{t-1} \qquad \textit{Eq. 19}$$

Combining all the model components, the full ARIMA model is given by:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \qquad \textit{Eq. 20}$$

where y_t is the differenced series.

In their book, Box & Jenkins (1970) developed a three-step approach for fitting ARIMA models systematically.

- Identification: Determining the order of the model via graphical methods as plots of autocorrelation and partial autocorrelation.
- Estimation: The parameters of the chosen model are estimated using either least-squares minimization or maximum likelihood estimation.
- Diagnostic checking: The quality of the model is evaluated, either via residual diagnostics (searching for uncaptured linear dependencies in the residuals) or deliberate overfitting to check if additional components are significant.

While this manual method is widely popular, it does not scale well. When forecasting hundreds of SKUs daily, this manual approach becomes infeasible. Moreover, model choice based on graphical models is somewhat subjective and ambiguous. For this reason, an information criterion can be used to choose a model. Information criteria were originally developed to prevent models from over-parametrization. They strive to balance model accuracy with parsimony in parameters by rewarding explanatory power (a lower residual sum of squares) and penalizing the use of additional variables (a represented by

lower degrees of freedom). Among the most popular ones is the Akaike information criterion AIC. (Akaike, 1974)

$$AIC = ln(\hat{\sigma}^2) + \frac{2k}{T} \qquad \textit{Eq. 21}$$

T is the number of observations, $\hat{\sigma}^2$ is the residual variance (RSS/T) and k is the number of parameters estimated, that corresponds to the model orders ($p + q + 1$). A lower value for each of these implies a better model.

As an extension to this, Seasonal ARIMA models (SARIMA) allow for modelling seasonal data by including additional seasonal terms to the model. A SARIMA model is defined by the order $ARIMA(p, d, q)(P, D, Q)m$, where the part in the second set of brackets is the seasonal part. The seasonal part models seasonal components by applying a backshift of m timesteps. The parameters (P, D, Q) are equivalent to the model components in the non-seasonal ARIMA models.

Models of the ARIMA framework can also incorporate exogenous variables to include information from covariates, e.g. promotional campaigns. These models are called AR(I)MAX models. Through the incorporation of both covariates and lagged values of the time series, they blur the lines between simple time series models and structural models.

2.4 NEURAL NETWORKS

This subchapter introduces the fundamental ideas of Neural Networks. The first section explains the broader paradigm of Machine Learning, under which Neural Networks fall. The remaining sections introduce two types of Neural Neworks. The purpose of this part is to establish a basic understanding of Neural Network, while not covering them

exhaustively. A solid idea of Neural Networks will facilitate understanding of the ideas introduced in later chapters.

2.4.1 **Machine Learning**

While computers effortlessly solve complex, abstract computational tasks, they struggle with tasks that seem intuitive and simple to humans. It appears that performing seemingly simple tasks requires an immense amount of knowledge about the world. (Goodfellow et al., 2016, p. 2) One solution to this would be to hard-code a set of rules that puts knowledge about the world into a formal language. In contrast, another branch of research in Artificial intelligence seeks to enable systems to acquire knowledge by learning from data. This capability is known as machine learning. Mitchell (1997) provided a formal definition of Machine Learning:

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."*(Mitchell, 1997)

When it comes to forms of experience (E), Machine Learning knows three types of learning: supervised learning, unsupervised learning and reinforcement learning. These forms differ with regards to the information that the learning algorithm is presented with during the learning phase.

In a supervised learning setting, the learner has access to a set of input-output pairs. Given an input, it learns a mapping of the input features to the output (Goodfellow et al., 2016, pp. 105–106). Forecasting, for instance, can be framed as a supervised regression problem: Given input data, e.g. historical records, the learner learns to predict the output, which would be the next value in the time series.

Talking about tasks T, within the field of supervised learning, there two most common among them are classification and regression problems (Goodfellow et al., 2016, pp. 100–103). In a classification problem, the output the learner needs to produce is categorical, i.e. there is a finite amount of outputs. These categories could be labels for the cars depicted in a picture. A regression problem, on the other hand, allows for continuous outputs. This would correspond to counting the number of cars in a picture. Most applications of forecasting are regression problem, as exact values of a time series need to be predicted.

Lastly, the performance of a learner is typically measured using some form of performance measure (P). If the performance measure is minimized, e.g. the number of errors, we refer to it as the loss function. Within the supervised learning setting, this loss function is usually minimized using an optimization algorithm. Optimization refers to the task of either minimizing or maximizing some function $f(x)$by altering x.

Putting the parts together and referring to Mitchel's definition, forecasting can be tackled as a machine learning problem. A machine learning model can learn the task of generating forecasts by exposure to historical combinations of input variables and optimal forecasts. The model is supposed to learn a functional form that minimizes the performance measure, which would be for example the forecasting errors.

In supervised learning settings, many of the ideas discussed in section 2.2 about overfitting apply. Given sufficient time and parameters, a machine learning model might find a perfect fit on the data by "memorizing" it. However, such a model would not generalize well and yield poor results on previously unseen data.

2.4.2 **Neural Networks**

Deep Learning is a subdiscipline of machine learning that applies Neural Networks for learning. The name Neural Networks stems from early efforts in the machine learning research to draw inspiration from the biological brain and imitate the structure of synapses. In simple terms, a neural network can be thought of as a function mapping of an input to an output. The classic example of a Neural Network is the Multilayer Perceptron (MLP), also known as feedforward-neural networks. Goodfellow et al. (2016, p. 168) write:

> *"The goal of a feedforward network is to approximate some function f. For example, for a classifier, $y = f*(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = f*(x;\theta)$ and learns the value of the parameters θ that result in the best function approximation."*

The name "networks" stems from the fact that the models are comprised of several simpler functions that are linked together or chained. Neural Networks find a good representation of a complex concept by expressing the problem in terms of simple concepts: They compose highly complex functions sometimes from hundreds of functions. (Goodfellow et al., 2016, p. 168)

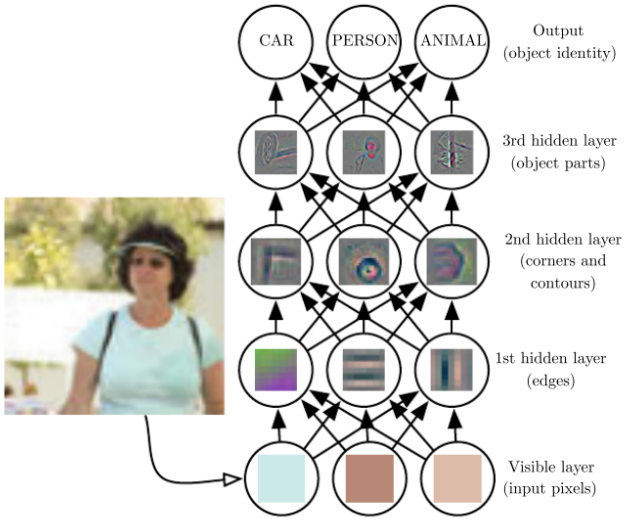

Figure 1: Illustrative Example of a Multi-Layer-Perceptron for Image Classification. Adapted from Goodfellow et al. (2016, p. 6)

Each of this function generates a new representation of their input. These functions can be thought of as units, or neurons. Within one neuron, an input vector x is weighted by multiplying it with a weight vector W and adding a bias c. A non-linear activation function is then applied to the weighted inputs. By applying a non-linear transformation, the model can learn much more complex dependencies and approximate more complex non-linear functions (Goodfellow et al., 2016, pp. 174–175). A single neuron has very limited capabilities for modelling dependencies. To model complex dependencies, several neurons act in parallel, comprising a layer. These layers are then stacked on top of another. An MLP is typically made of an input layer, one or more hidden layers and an output layer. This layer architecture is shown in Figure 1.

As can be seen from the figure, in the input layer, each input feature x_i is propagated forward to the neurons in the first hidden layer. The neurons in the hidden layer perform the computations described above and then propagate their output to each neuron in the

next layer. The final layer is called the output layer. Given the inputs from the final hidden layer, it calculates the final output of the network.

The architecture described above is capable of approximating highly complex functions. In order to do so, it must learn an optimal set of parameters (W,c) to apply to the inputs. Via supervised learning, the network is taught to approximate the supposedly true input-output mapping by being exposed to a set of paired input-output samples. It is worth noting that the training examples do not define the desired output for the hidden layers but only for the output layer. During learning, the algorithm must learn how to use the hidden layers to minimize the loss at the output layer. (Goodfellow et al., 2016, pp. 168–169)

The MLP is also known as a feedforward network, as it propagates information only in one direction. The next section will cover a class of neural networks that allows for feedback connections and thus is capable of detecting temporal dependencies.

2.4.3 **Recurrent Neural Networks**

While it is possible to model time series problems with MLPs, they are not particular suited for processing sequential data. Recurrent Neural Networks (RNN) are a family of NN that introduce a feedback loop mechanism, through which information of past inputs is retained. (Goodfellow et al., 2016, p. 374) After every input step, the network passes a vector through a special hidden layer to the next step. The special hidden layer is known as the hidden state h_t. Through the hidden state, RNNs introduce some form of context for the interpretation of inputs, so they are not modelled in isolation. This way, they can process sequential information. Most recurrent networks can also process sequences of variable length. (Goodfellow et al., 2016, p. 384)

The basic architecture of a classic RNN is outlined on the righthand-side of Figure 2. In order to gain a better understanding of how information is passed over sequential inputs, the left-hand side depicts the network unrolled over time.

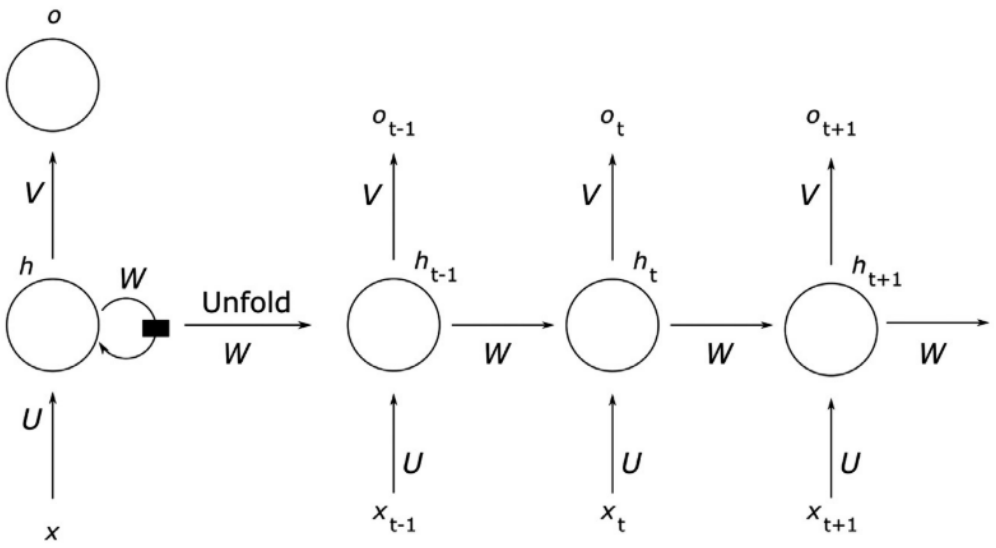

Figure 2: A Recurrent Neural Network, unrolled over time, adapted from Bandera et al. (2020)

The input at time t is x_t, the output is o_t. The hidden state h_t is affected by x_t and the previous hidden state h_{t-1}, weighted by the respective weight matrices U and W. The output is calculated from the current hidden state and the weight matrix V. Mathematically this is expressed as

$$h_t = f_\theta(Ux_t + Wh_{t-1}), \qquad \textit{Eq. 22}$$

$$o_t = f_\alpha(Vh_t), \qquad \textit{Eq. 23}$$

where f_θ, f_α denote non-linear functions as the ReLu-function or the tanh-function. (Bandara et al., 2020)

The hidden state introduces a way to pass information over time in processing sequential data. However, basic RNNs are not very effective at modelling long-term

dependencies. Their central problem is that gradient-based training techniques, as the backpropagation algorithm, determine the influence of a given input by measuring the sensitivity of network parameters on the output. (Goodfellow et al., 2016). For very long sequences, the initial inputs have a very small impact on the output, because they are propagated through many steps.

In response to this problem, Hochreiter & Schmidhuber (1997) proposed the long short-term memory (LSTM) model. This type of RNN is capable of retaining information over much longer periods of time. Moreover, it can learn when to forget information that is no longer relevant. Explaining the inner workings of the LSTM would go beyond the scope of this thesis. Interested readers can referGoodfellow et al. (2016, pp. 410–411)

3 LITERATURE REVIEW

This section reviews the scientific literature on the research questions raised in Chapter 1. It tackles the questions from two perspectives: The first perspective emulates the researcher's view, who is investigating how Neural Networks perform in forecasting in general. Chapter 3.1 covers this perspective by reviewing the research on forecasting with Neural Networks, and trying to identify what contributes to their success or failure.

The second perspective is the practitioner's view: it questions what makes a forecast good in the context of demand forecasting for perishable products. Chapter 3.2 introduces the specific issues of forecasting SKU-level demand, and reviews different approaches to overcome these.

3.1 FORECASTING WITH NEURAL NETWORKS

This section covers previous research on forecasting with Neural Networks, spanning from the early applications in the 90s to recent state-of-the-art solutions. It splits the research into three periods, roughly based on the central insights won during each.

3.1.1 Early Applications

Though the hype in Artificial Intelligence is a recent one, the idea of applying Neural Networks to forecasting problems has been around for decades. In their comprehensive literature review, Zhang et al. (1998) identify the first application of Neural Networks to forecasting back in 1964. Despite this, research was sparse before the conception of the backpropagation algorithm in the 1980s, which enabled the training of deep networks.

This period of early research is characterized by enthusiasm for the new method, which is considered a capable method for modeling non-linear time series (Zhang et al., 1998).

In contrast, Chatfield (1993) regards Neural Networks with a fair share of skepticism, hypothesizing that they may become a 'passing fad'. In an editorial for the Journal of Forecasting, he criticizes that many studies on NNs are lacking comparison against established forecasting methods like the statistical models from the Box-Jenkins framework, which chapter 2.3.2 covered.

Sharda & Patil (1992) conducted some of the first large-scale empirical studies on forecasting with Neural Networks. They compare an MLP network against an automated Box-Jenkins forecasting system on a set of 75 time series. They find that the simple neural network model is on par with the Box-Jenkins forecasting system. In contrast, Foster et al. (1992) find that Neural Networks provide less accurate forecasts compared to traditional methods based on 384 economic and demographic time series. The networks provided worse forecasts than a linear regression model and a weighted average of six simple methods.

Reviewing the literature up until this point, Zhang et al. (1998) conclude that Neural Networks are quite suitable and useful for forecasting tasks and give satisfactory performance. However, despite considerable research, they remain inconclusive on whether Neural Networks outperform classical methods. While they consider them promising alternative approaches to traditional linear models, they list several limitations:

- Since NNs are nonlinear methods, they are less likely to be better at static linear processes.
- They are essentially black-box models
- Since they have many parameters, they tend to overfit and generalize poorly.
- They require a higher amount of data and have higher computational costs
- There is no structured method to determine the model architecture that allows for many designs.

Regarding the last point, the authors note that most researchers seem to follow a trial-and-error-approach in their design. They draw a connection between the lack of a systematic approach and the inconsistent results in the literature. Likewise, doublechecking earlier studies on forecasting business time series with Neural Networks, Adya & Collopy (1998)found flaws in most of them. Out of 48 studies they checked, only eleven implemented and validated the Networks correctly, so these early results should be considered with caution.

Moreover, this early research covers almost exclusively Multi-layer Perceptrons (MLP), not Recurrent Neural Networks. The MLPs are mostly applied to univariate forecasting problems, as if they were just another tool in the box of forecasting (c.f. Chakraborty et al. (1992) for an exception). They show promising results for non-linear prediction problems.

3.1.2 **Research since the 2000s**

The early 2000s saw the biggest forecasting competition of its kind hosted so far: the M3-competition. The purpose of the competition was to develop forecasting algorithms to predict 3003 time series of varying length and frequency from business and economics (Hibon & Makridakis, 2000). The only Neural Network contribution, the Automat ANN algorithm, delivered mediocre performance, and was seldom among the top contenders regardless of industry, time series frequency, and forecasting horizon.

The M3 competition was replicated in 2011 to account for the research progress in Neural Network research. The 'NN3' competition accepted entries from the field of Neural Networks and Computational Intelligence. The models were evaluated based on 111 series of the M3, with the SMAPE metric measuring their accuracy. Among the 59

submissions in the “NN3-competition” (Crone et al., 2011), only one beat the damped trend, a benchmark used in M3 that yielded surprisingly good results considering its simplicity. However, several models were able to beat the Automat ANN model submitted in the original M3 competition, making a case for research progress over the past decade. Furthermore, the NN3 saw several contenders that made use of Non-MLP architectures, e.g., RNNs.

While Neural Networks seemingly fell short of expectations in the forecasting domain, research achieved tremendous breakthroughs in their application in other areas, e.g., in image classification (Krizhevsky et al., 2012) and learning complex games such as GO (Silver et al., 2016). The increasing popularity of Machine Learning, paired with the ubiquity of the term ‘Artificial Intelligence’, further fueled the research on forecasting with Machine Learning models. Makridakis et al. (2018a) conducted another large scale study on the M3 data. Their study empirically evaluated 10 Machine Learning algorithms' performance for time series forecasting on 1045 monthly series and compared them against eight statistical forecasting methods as a baseline. These algorithms included RNNs, and Long Short-term Memory Networks (LSTMs), a variant of Neural Networks that had recently become more popular in forecasting because of its supposed ability to capture long-term dependencies in sequences. All models generated forecasts for up to 18 periods ahead, and different techniques for preprocessing the time series were tested, to facilitate learning for the Machine Learning algorithms.

Once more, the study yielded sobering results for forecasting with Neural Network models. The statistical methods outperformed across all forecasting horizons. Moreover, the authors note that the computational complexity remains an issue with Machine Learning models and that more computational time does not necessarily improve

accuracy. They reinvigorate the idea that Machine Learning models might be beneficial for certain time series types, e.g., when non-linear characteristics are present in the data.

This research period showed that, despite the use of more sophisticated model designs and increasing computational power, Neural Networks struggle to improve upon the performance of statistical baselines.

3.1.3 **Recent Developments and the M4 competition**

In 2018, the M3's successor extended the competition in multiple ways. The M4 forecasting competition (Makridakis et al., 2018b, 2020) tested models on 100,000 time series. It is the most comprehensive competition to date. The 100,000 time series are sampled from several industries and of varying frequency. As with previous competitions, there was an emphasis on business-related time series.

While pure Machine Learning methods performed poorly in general, a hybrid approach by Smyl (2020) of Machine Learning and a statistical method did exceptionally well (Makridakis et al.,2020). This model used a Neural Network to learn to predict seasonality. Interestingly, it did not learn this for each series individually, but from all series combined. This led Makridakes and his co-authors to hypothesize that using information from multiple series to predict individual series works well.

The M4 entailed a series of comments critically discussing whether its conclusions can be generalized. Fry & Brundage (2020) point out that the design of the M4 did not resemble real-life conditions for forecasting properly. They point out that features beyond historical sales are available under real-life conditions, which Machine Learning models can utilize in the prediction process. Moreover, as Barker (2020) points out, the pure ML models entered in the M4 competition did not learn patterns globally, but modeled each

problem individually. The models did not cross-learn from different series as this is not encouraged by the design of the M4, which features diverse series. Smyl's (2020) top-competing submission poses an exception.

Barker (2020) interprets the poor performance of pure machine learning models as a failure to learn from the given data. Machine learning algorithms are interpolation models, but forecasting is an extrapolation problem. In other words, in a forecasting setting, the model must learn to model a situation which it has never seen before. An essential prerequisite for Machine Learning models to succeed is to create dense manifolds, i.e., creating data sets that resemble a wide range of possible situations. For a short time series, this can be achieved by modeling series with similar properties together.

This idea of modeling series together, i.e. building one model for forecasting several series, is referred to as building global models. This approach is most likely to succeed if the forecasting problem involves a large number of related time series, the time series are hierarchical, and there are exogenous features available. (Fry & Brundage, 2020). In univariate forecasting problems, the number of historical observations is typically too small for complex models such as Neural Networks to fit their parameters and avoid overfitting (Bandara et al., 2020). Even for long series, the early observations in the series provide little useful information, as the underlying patterns and relationships are likely to change over time (Rob J. Hyndman, 2016). Other than traditional forecasting techniques, neural networks cannot only model problems globally, but the global modeling approach might fix the problems above, creating a niche for Neural Networks beyond non-linear forecasting.

Only a few empirical studies so far provide evidence for the superiority of global learning. For instance, Bandara et al. (2020) propose a global LSTM model. As an

extension, they apply different clustering algorithms to identify similar time series and suggest training the models only on the clusters. They show that this clustering approach improves prediction. The proposed model yields competitive results on the data of two earlier forecasting competitions. Research in this field seems to be driven by practitioners, as most of the rare instances of global modeling for forecasting in the literature can be traced back to researchers at Amazon (Salinas et al., 2020; Wen et al., 2017) and Uber (Bandara et al., 2020; Zhu & Laptev, 2017).

It appears that the perspective on Neural Networks has shifted towards understanding them as tools for modeling problems that traditional forecasting methods cannot represent. Namely, global modelling and the inclusion of external variables as features.

3.2 DEMAND FORECASTING AT THE SKU LEVEL

Having provided a general overview of forecasting with Neural Networks in the previous section, this section narrows down the scope to SKU-level demand forecasting for perishable products. Typical settings for this type of forecasting problems are the retail and wholesale industry for foods.

The large-scale studies introduced in the previous section have one thing in common. They focus on theoretical performance measures like the ones introduced in Chapter 2.2. These theoretical measures lend themselves to a general evaluation of forecasting accuracy. As explained in Chapter 2.1, forecasting accuracy is a pre-requisite for effective inventory management. Performance metrics provide a good indication for accuracy and are easy to calculate, so many organizations rely on them as ‘a suitable proxy’ (Davydenko & Fildes, 2013).

Demand forecasts for SKUs are primarily used to determine parameters for inventory management. (Arunraj & Ahrens, 2015). For instance, they determine the order-up-to-level in the periodic review policy described in Chapter 2.1. The forecasts are thus not used directly, but to determine the cumulative demand during the replenishment time. While performance metrics are popular in the research community, a practitioner will usually be interested in actual utility values, as the number of stock-outs or the required safety stock directly translates to operational costs. Gardner (2006) points out that forecasting is a primary determinant of inventory cost. Since forecasts are such crucial components in an operating system, he concludes that forecasting methods should be selected based on the benefits for the operating systems that come with them.

In some cases, this can lead to contradictory evaluations: For instance, in his study of intermittent demand forecasting with a Neural Network, Kourentzes (2013) found evaluations based on performance measures and inventory metrics to allow for very different judgments of the model, respectively. Intermittent demand describes the phenomenon of sporadic demand with long periods of no demand at all. While the Neural Network model provided worse forecasts than the Croston method in terms of performance metrics, it achieved consistently higher service levels for the same stock levels in a subsequent inventory simulation.

Apart from that, several other forecast traits that a single metric cannot capture are favorable from an inventory management perspective. For instance, Fry & Brundage (2020) point out that the forecasting literature has paid little attention to measuring the uncertainty of forecasts, which is crucial information in many inventory management applications as safety stock or reorder point calculations. The following sections will discuss some aspects that make demand forecasting perishable products at the SKU level

challenging. They will discuss how previous research efforts have sought to solve these issues.

3.2.1 **Scale**

From a practical perspective, forecasting in a wholesale (or likewise retail) setting is subject to an additional set of challenges. An evident difficulty is the broad scope of the forecasting problem: The product range of wholesalers typically consists of several hundreds of SKUs. Each SKU poses an individual prediction problem or multiple predictions for forecasting several locations. Any solution needs to be scalable to a certain degree. For instance, expert-based adjustments are subject to human resource constraints. Automated solutions might thus be preferable, though they come with a caveat: The available computational resources limit them. Highly sophisticated automated procedures might take just too long to compute daily forecasts of hundreds of products.

Several studies have pointed out the high computational effort of training and fitting Machine Learning models for large-scale forecasting problems (Carbonneau et al., 2008; Makridakis et al., 2018a). At the scale of SKU-level forecasting, the high computational effort for training Neural Networks might be problematic. The global modeling approach mentioned at the end of the last section might be a potential workaround. Salinas et al. (2019) claim that their global demand forecasting model for over 500.000 products at Amazon took only ten hours to train.

3.2.2 **Including external data**

Daily sales of perishable food are usually highly volatile and skewed since they are subjected to several external factors that drive demand, such as holidays and temporary price reductions (Arunraj & Ahrens, 2015).

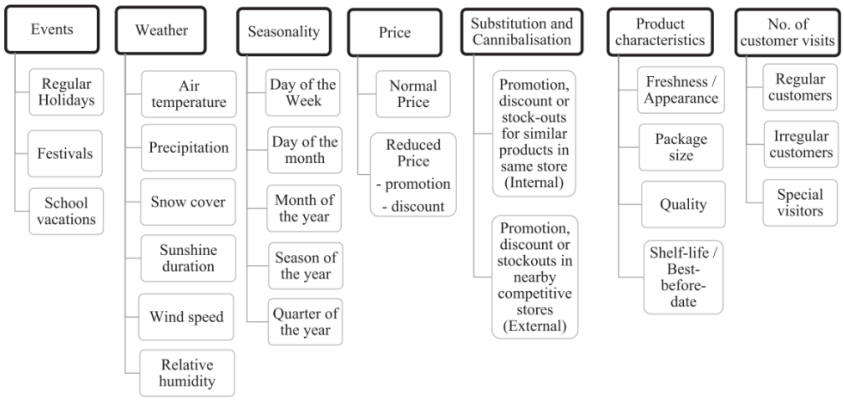

Figure 3: Demand driving factors, adapted from (Arunraj & Ahrens, 2015)

Figure 3 provides an overview of potentially influential factors. To account for these factors, a forecast model must allow for external inputs.Aburto & Weber (2007) investigated a model that, apart from past sales, includes price data and binary variables characterizing special days such as holidays, vacation periods, and employees' payment days. The model has a SARIMA component that models the regular time series. Subsequently, a Neural Network (MLP) is used to model the forecast error of the SARIMA component with both autoregressive features and the additional variables described above. Applied to sales forecasting at a Chilean supermarket, the model yields better forecast results and, consequently, lowers inventory levels and stockouts.

Incorporating weather information can improve sales forecasts, as shown by Steinker et al. (2017). They suggest an ARIMAX model that uses weather forecasts as external variables. They use total daily sunshine hours, the average air temperature, and precipitation as features. Additionally, the model captures non-linear relationships through interactions with dummy variables for seasons and weekends. Apart from using absolute values, they also suggest using relative scores, which are supposed to make the weather comparable over different calendar months. They apply the weather-based

models to forecast sales for a large European e-commerce platform and find forecast errors to decrease by 8.6% to 12.2% on average.

Arunraj & Ahrens (2015) conducted a study on forecasting with the help of several external variables. They utilize both external and autoregressive data to forecast quantiles of perishable food, i.e. the daily sales data of bananas from a typical food retail store situated in lower Bavaria. They fit a seasonal ARIMA model with external data (SARIMAX) and a Neural Network on calendar data, promotional and discount data, as well as the weather. They compare the models against a naïve Baseline and seasonal ARIMA without external data. The Neural Network outperforms both baselines, but the SARIMAX model yields the highest accuracy. Regardless of that, this provides evidence for how external variables can improve food items forecasting at the SKU level.

Apart from external data, sales of related products within a group can serve as a useful predictor of demand. Ali et al. (2009) investigated how SKU-level demand forecasting during promotional periods can be improved. Their study does not use Neural Networks, but another Machine Learning class, Regression Trees. They find that incorporating additional features from sales and promotion time series of related products can significantly improve the forecasting accuracy in promotion periods.

For both promotion- and non-promotion periods, Ma et al. (2016) show models to benefit from capturing cross-effects of promotions within and among categories. They also address the problem of high dimensionality. When dealing with many SKUs, the number of potential predictors is high. High dimensionality makes it impossible to estimate the models because of the limited number of degrees of freedom. Thus, Ma et al. (2016) develop a multi-stage framework for selecting relevant features based on the

regularization techniques Lasso-regression and Principal Component Analysis for dimensionality reduction.

3.2.3 **Uncertainty estimation**

As pointed out at the beginning of this subchapter, daily SKU-level forecasts are typically not used directly in the retail and wholesale industry. Instead, they are translated to inventory policy, i.e. determining the order size based on forecast and safety stock level (Arunraj & Ahrens, 2015). For adequate safety stock calculation, forecasts must come with some form of uncertainty measure. Thus, a strategy to estimate said uncertainty is indispensable. A textbook approach to this is to estimate uncertainty from past forecast errors. Many methods assume normally distributed errors to calculate prediction intervals (c.f. Chapter 2.1).

These normality assumptions often do not hold for skewed, volatile and time-varying time series, like SKU-level demand. For instance, Taylor (2007) showed that methods based on normality assumptions provide inadequate confidence intervals on data from 256 SKUs from a large UK supermarket chain outlet. In response to this, Taylor applied a non-parametric method, exponentially-weighted Quantile Regression (EWQR), to develop robust point-estimates and quantile estimates. In terms of point forecasting, the robust forecasts from using EWQR outperform the level-smoothing methods. For the interval forecasts, EWQR also yields good results.

EWQR is a particular case of the more general Quantile Regression (QR). Quantile Regression seeks to predict quantiles of a dependent variable based on a conditional quantile function of independent variables. (Koenker & Bassett and Jr, 1978) The p-

quantile of a random variable X is defined as the real number x_p, for which $P(X \leq x_p) \geq p$ and $P(x_p \leq X) \geq 1 - p$ (Georgii, 2009).

Quantile Regression avoids the need to make prior assumptions. It is a non-parametric method to estimate uncertainty, which is a desirable property given that the normality assumption often does not hold for SKU-level demand.

In the study by Arunraj & Ahrens (2015) mentioned earlier, the authors subsequently used Quantile Regression to turn their SARIMA model's forecasts into quantile forecasts in a second step. They find their SARIMA-QR model to identify extreme and sparse sales accurately and directly. Using this approach outperformed the confidence-interval estimation with standard Gaussian assumptions.

With the right loss function, Neural Networks can perform Quantile Regression. Wen et al. (2017) propose a framework for general probabilistic multi-step time series regression using a sequence-to-sequence structure, which section 4 will discuss in detail. Their model is a series of LSTMS and MLPs that learns to forecast quantiles of time series. To achieve that, they use the Quantile Loss as the Loss function. Their model falls into the global model framework. The suggested model learns to forecast weekly demand series of around 60,000 sampled products on Amazon. There are two notable shortcomings to this work. The authors compare their model only against ML benchmarks (which they outperform), but not against statistical models. Moreover, their model cannot predict joint quantiles for two cumulative forecasts.

In the operations literature, Quantile Regression has been used to solve the data-driven newsvendor problems. Huber et al. (2019) incorporate Machine Learning-based Quantile Regression into their optimization model for a single-period news vendor model. The

model learns to predict the optimal demand quantile from demand data directly. In classic newsvendor settings, demand prediction and quantile estimation usually constitute two separate stages. Firstly, one method provides a forecast and secondly, based on the forecast errors, a distribution is estimated. Given the critical fractile, the assumed distribution then determines the optimal order quantity. Incorporating both stages into one model avoids the need to make assumptions about forecast-error distribution. Based on point-of-sales data for a large German bakery chain, the authors find their suggested integrated approach to outperform the two-stage approach in terms of cost. Neural Networks performed exceptionally well in this analysis, even more so when trained on multiple time series. The performance of the Neural Network further makes a case for building global models. Like the study by Wen et al. (2017), this work does not cover the multi-period newsvendor problem, i.e., predicting quantiles of cumulative demand.

3.2.4 **Multiple-step Ahead Prediction**

Strijbosch et al. (2011) pointed out that it is common practice to optimize forecasting models based on the one-step-ahead forecasting error, while many operational applications require forecasting for multiple steps ahead. For long lead times, a forecasting procedure needs to provide multiple-step-ahead forecasts. The forecaster needs to develop a strategy to generate forecasts for multiple periods, knowing that the sum of these forecasts is more relevant than the accuracy of individual forecasts.

Within the realm of traditional forecasting techniques, models put out a single scalar value. Coming up with forecasts for multiple steps requires some form of strategy to predict multiple scalars. Commonly, two strategies are applied: The recursive strategy and the direct strategy (Ben Taieb et al., 2010). In the recursive strategy, forecasts are made

iteratively one-step-ahead. Predictions go back into the model between steps as if they were the series' actual value. This leads to the accumulation of forecasting errors over time, resulting in high uncertainty levels for long forecasting horizons (Ben Taieb et al., 2010). The direct strategy avoids the error accumulation by turning the H-step-ahead forecasting into H independent problems: Each model exclusively predicts one of the H steps. Not only is this computationally expensive, but it also ignores the stochastic dependency between two steps (Ben Taieb et al., 2012).

To avoid this tradeoff, Ben Taieb et al. (2010) suggest using models that predict multiple outputs (in a vector) at once. They refer to this as Multi-Input-Multi-Output (MIMO) modelling. The flexibility of Machine Learning models allows for this form of output. Under the MIMO scheme, the model provides forecasts for all periods in the forecasting horizon at once.

One type of modeling approach that uses the MIMO scheme is sequence-to-sequence modeling (Seq2seq). The goal of sequence-to-sequence modeling is to find mappings from one sequence x to another sequence y, where the length of the sequence can vary from each other. (Goodfellow et al., 2016, pp. 396–397).

Cho et al. (2014)and Sutskever et al. (2014) independently proposed an encoder-decoder architecture for Seq2seq-modeling. The idea was initially conceived for a machine translation task to account for the fact that sentences of the same meaning can consist of a different number of words in different languages.

Encoder-decoder architectures are comprised of two RNNs. The job of the encoder RNN is to find a representation of the input sequence that the decoder RNN can read and then process into an output sequence. (Goodfellow et al., 2016, pp. 396–397) In a translation context, the encoder understands what the input sentence, e.g., a French

sentence, says and translates it to an abstract language. The decoder understands the abstract language and translates it to Finnish. In its simplest forms, this works as follows:

1) The encoder RNN gets the input sequence. After processing the whole input sequence, the encoder propagates its final context C_T (or a transformation of it) to the decoder RNN.

2) The decoder RNN takes the context C_T as its initial hidden state, and iteratively generates the output sequence for *n* steps ahead, where *n* is an arbitrary number of steps.

Training of the two RNNs happens simultaneously so that they learn to communicate with one another optimally (Goodfellow et al., 2016, p. 397). Since the idea of encoder-decoder models is relatively young, it has not been discussed thoroughly in the forecasting literature yet. Many of the large-scale global models for time series forecasting referenced in earlier sections make use of the encoder-decoder structure. Salinas et al. (2020) leverage the encoder-decoder structure also to incorporate additional external variables about the known future into their forecasts. These additional variables are used in the decoder part of the model to generate the output sequence. This structure will be described more thoroughly in the upcoming chapter, as it serves as an inspiration for the model suggested in Chapter 4.

4 MODEL DESCRIPTION

This section introduces a neural network demand forecasting model for inventory management: the Seq2Quant model. It lays out the theoretical contribution of this thesis. Building on previous research insights, the Seq2Quant model is a new demand forecasting approach that directly addresses the issues raised in section 3.1. The suggested solution is a global modeling approach that maps multivariate input sequences of historical records and covariates to a sequence of estimated quantiles of cumulative demand. Precisely, the model is:

- A global sequence-to-sequence model
- A multivariate-input model using external variables and covariates
- A quantile estimator of cumulative demand

The following section breaks down the details of the model further.

4.1 GLOBAL SEQUENCE-TO-SEQUENCE MODELING

This section describes the general framework of the model. A significant issue in univariate forecasting with neural networks is the low amount of available data for individual time series. As pointed out by Hyndman (2016), time series are often too short to prevent Neural Networks from overfitting during training. Even if the model is trained on a long time series, data from years ago might not represent the time series' current behavior.

Recent works discussed in chapter 3.1.3 and 3.2.1 suggest using one global forecast model to overcome these limitations. Instead of treating n time series as n independent problems with n underlying distributions, a global model assumes one common underlying distribution for every single time series. The global model understands the n

time series as *n* realizations of one underlying distribution, which can be explained by past observations and covariates.

Smyl's (2020) top-contending entry in the M4 competition demonstrates the potential of global modeling. In a recently proposed framework, Mariet & Kuznetsov (2020) prove that the sequence-to-sequence approach provides superior generalization guarantees compared to modeling forecasts individually, given that

- The number of time series is significantly larger than the number of observations per time series
- The time series are weakly correlated

Note that these guarantees refer to theoretical bounds on the capability of generalization, and are thus not a guarantee for superior performance. As discussed, inventory management typically deals with many SKUs, some of which have only a short history of observations. The fact that the demand for different SKUs is likely to share similar traits can be leveraged to create favorable conditions for Neural Network training, i.e., by creating dense manifolds by modeling series together (c.f. Barker, 2020).

Consequently, the suggested model uses a global sequence-to-sequence structure, as defined in the recently proposed framework by Mariet & Kuznetsov (2020). It seeks to learn a hypothesis to map a sequence of *m* past demand observations to *n* future demand observations.

$$H\colon (y^{t-M}, \dots, y^{t-1}, y^{t}) \rightarrow (\hat{y}^{\,t+1}, \dots, \hat{y}^{t+N-1}, \hat{y}^{\,t+N}) \qquad \textit{Eq. 24}$$

Sequence-to-sequence models are Multi-Input-Multi-Output (MIMO) approaches to forecasting (c.f. Ben Taieb et al., 2012). Hence, they are optimized for multiple-step ahead predictions. During training, each entry in the output sequence contributes equally to the loss value the network seeks to minimize. Equally weighted steps are a desirable trait for

inventory management because the replenishment time typically includes multiple periods, each of which equally contributes to the order-up-to level calculation.

4.2 Exogenous Variables and Covariates

This section describes the architecture of the model. At the core of the suggested model are Recurrent Neural Networks. By design, RNNs can pass information through time (c.f. Chapter 2.4.3), which allows them to learn temporal structures. Moreover, RNNs make it possible to include multivariate-time-series as inputs seamlessly. The inputs are then not individual observations but a vector of multiple observations at each time step t.

Chapter 3.2.2 presented successful examples of including exogenous variables to model external demand drivers in demand. The suggested model structure allows and advocates for including external variables. Commonly suggested external demand drivers in the literature include:

- Price
- Promotional campaigns and discounts
- Holidays and events
- Indicators of seasonality
- Weather
- Data on related SKUs / product groups

The model uses RNNs for both the encoder and the decoder part to capture the temporal aspect of multiple input and output steps. The model is thus fed two-dimensional vectors, consisting of observations of V variables for M past periods and N future periods. The model input is thus a vector of vectors. The model can learn how these variables influence demand, given there are enough observations of them in the data (Wen et al., 2017).

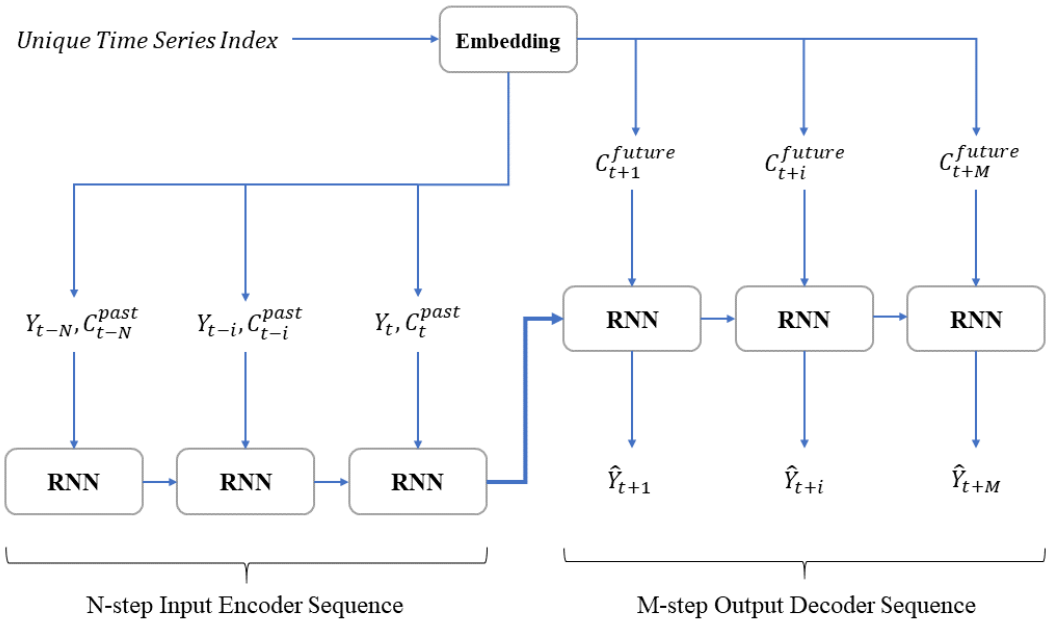

Figure 4: Suggested Seq2seq architecture with exogenous variables and an embedding layer

For events that occur only once a year, learning might be difficult. For instance, Christmas is an annual event. If historical sales for two years are available, there are only two observations in each time series from which the model can learn the exogenous effect of Christmas sales. For a new product introduced in the current year, no observation is available. The global structure facilitates learning because the model is exposed to Christmas's effects in multiple time series and can learn to abstract the effect.

In classical regression approaches, the future values of external factors must be known in order to generate forecasts. For sales of related SKUs, this is impossible at the point of forecast generation. If the future values of external variables are unknown, the model must be provided with forecasts of them. Forecasts introduce an additional source of uncertainty to the model.

The suggested model uses an encoder-decoder structure to avoid this. Figure 4 illustrates this structure. The encoder part encodes all known information about the past

and provides the context to the decoder. The decoder generates forecasts, given the context, and provided with all information about the known future. The structure allows the inclusion of different variables for the encoder and the decoder part of the model, to represent which information is known for sure to generate the forecast.

On top of external variables that change over time, the input includes static features of each time series, referred to as covariates. Covariates include e.g.

- Location
- Product Category
- An index unique to each time series

Including covariates allows the global model to bridge different sets of time series behaviours, e.g., effects of a holiday that only apply at a specific location (Wen et al., 2017). Covariates are input to both the encoder and the decoder part of the model. Through the covariates, the model can learn behaviour that is unique to specific product groups.

Drawing inspiration from Salinas et al. (2019), the model uses an embedding layer through which the model can identify unique time series. The embedding layer enables the model to learn unique behaviours for individual time series. During preprocessing, a unique index (e.g., for every SKU) is assigned to each time series. The embedding layer of the model maps the index to a unique vector of weights. The weights represent the series's unique trait, and they are learned in the training process.

4.3 FORECASTING QUANTILES OF CUMULATIVE DEMAND

The last aspect of the suggested model concerns its output and its training scheme. As established before, the central purpose of demand forecasts under classic inventory management policies is to estimate the demand during the replenishment time. When forecasting for the replenishment time, ultimately, the accuracy of the cumulative forecast is what matters. In this regard, the accuracy of forecasts for individual days is secondary if the cumulative forecasting for the replenishment time is accurate.

For this reason, the Seq2Quant modelling approach reframes the problem of forecasting demand multiple-steps-ahead to a cumulative demand forecasting problem. The output sequence consists of *N* entries, where each entry *n* in the sequence is a forecast of the cumulative demand up until this point. As such, the model seeks to learn a hypothesis H, such that:

$$H\colon (y^{t-M}, \dots, y^{t-1}, y^{t}) \rightarrow \left(\widehat{y^{t+1}}, \widehat{\sum_{\iota=1}^{2} y^{t+\iota}}, \dots, \widehat{\sum_{\iota=1}^{N} y^{t+\iota}}\right) \qquad \textit{Eq. 25}$$

Since demand is always weakly positive, the cumulative demand up until the next period can never be lower than up until the current period. The temporal structure of RNNs lends itself to this framing, as the forecast of cumulative demand for one period directly affects the forecast of the following period. The cumulative forecast is fed back into the network to allow for consistency.

Chapter 2.1 provided an introduction on how to determine the optimal order-up-to level *S*. The optimization problem stated in Eq. 2 corresponds to finding the *p*-quantile of the cumulative demand, where *p* is equal to the target CSL α. The capability of Neural Networks to adopt different loss functions allows the Seq2Quant model to regress for quantiles of cumulative demand directly. The pinball loss (also known as quantile loss)

function is used to obtain quantile forecasts. Citing Smyl (2020), the pinball loss is defined as follows:

$$L_t(y_t, \widehat{y_t}) = \begin{cases} (y_t - \widehat{y_t})\tau, & y_t \geq \widehat{y_t} \\ (\widehat{y_t} - y_t)(1-\tau), & \widehat{y_t} > y_t \end{cases} \quad \textit{Eq. 26}$$

where τ corresponds to the target p-quantile. The pinball loss penalizes over- and underpredictions unequally, so a high τ would incentivize predicting too much over too little. Setting τ to 0.5 would result in a regression to the median. However, the overall loss is zero when the forecast errors $y_t - \widehat{y_t}$ are eradicated, which maintains the overall incentive to forecast as accurately as possible.

Note that this puts the model at the threshold from predictive to prescriptive modelling. The Seq2Quant model does not explicitly forecast future demands but learns to estimate an optimal, risk-adjusted order-up-to level $S^*_{t,t+i}$ for every forecasting horizon i within the following *N* periods.

$$H\colon (y^{t-M}, \dots, y^{t-1}, y^t) \rightarrow \left(S^*_{t,t+1}, S^*_{t,t+2}, \dots, S^*_{t,t+N}\right) \quad \textit{Eq. 27}$$

The goal of prescriptive modelling is not to predict the future but to derive optimal instructions on how to act. The model expands the research of Huber et al. (2019) to the multi-period case.

5 METHODOLOGY

The Seq2Quant's performance will be empirically evaluated in a quantitative experiment using real-world data. The model will forecast the demand for several hundred SKUs in the fresh product segment of a German wholesale company over three months. Subsequently, a simulation gauges the effects on the inventory development of this product segment under close-to-real-life restrictions. The model will be compared against a statistical baseline model to quantify whether a hypothetical gain in performance justifies the additional computational complexity.

5.1 CASE DESCRIPTION

Most of the data for the experiment was provided by Lekkerland SE & Co. KG, a German wholesale company operating in the convenience sector in multiple European countries. Its primary customers Customers include petrol stations, kiosks, convenience stores, bakeries, food retailers and quick service restaurants. The company operates 14 logistic centers at several locations in Germany, five of which serve the role of a central warehouse. The central warehouses deliver both to customers and other logistic centres, which puts them at crucial points in the supply chain. While it provides an array of product segments from tobacco products to beverages, the company is particularly interested in optimizing the forecasting process of its fresh products category. Products in the said category are perishable. The category is furtherly divided into five product segments:

- Dairy products, fruits, and vegetables
- Meat and cold cuts
- Fast food
- Bread
- Sandwiches

Due to the perishable nature of these products, the company tries to keep their warehouse cycle times at a minimum. Many products come with a date of expiry. Upon delivery to customers, the case company guarantees a minimum remaining time range until the expiration date. Likewise, the company has negotiated a minimum remaining time with its suppliers. Thus, there is a natural upper limit to their storage time at the company's warehouse. For products without an expiration date, e.g., many vegetables, excessive storage time makes them unsellable due to outer appearance and health concerns. For some products in the segment dubbed 'ultra-fresh products', the maximum acceptable time in the warehouse is as low as one or two days.

Products exceeding the maximum acceptable time in the warehouse get discarded altogether. From an inventory management perspective, the company is facing overage costs due to waste. On the other hand, due to the market's competitiveness, providing a continuous and full supply of goods for their customers is part of the company's strategy. Sufficiently high stocks are essential for that. Unfulfilled orders are associated with high perceived underage costs. Thorough inventory planning, and accurate forecasting of demand in particular, is vital for balancing this trade-off.

Almost daily replenishment of product stocks is thus necessary. Moreover, the company handles the process with particular caution. For other segments, forecasting and replenishment of many product segments are handled mainly automatically by enterprise software. In contrast, the forecasting process for fresh products is highly manual: A group of material requirement planners determines the order quantity for every individual product daily. These dispatchers estimate the demand for the upcoming days. Order quantities are then calculated based on the expected demand during the product replenishment time, including the supplier lead time and the prespecified review period.

In that sense, the replenishment process follows the forecast-based periodic review policy outlined in Chapter 2.1.

The forecasting process, on the other hand, is only weakly structured. When placing orders, the dispatchers examine daily product sales of the past couple of weeks. They then estimate the demand for the upcoming days, taking past sales and judgmental adjustments into account. Company executives provided me with a description of typical considerations for adjustments:

1. Upcoming holidays that might shift demand
2. Promotional activities at the customer site
3. Increased travel activity due to regional events and school holidays
4. Weather
5. Safety stocks

On top of that, the ordering process is subjected to a set of restrictions imposed by suppliers. These include minimum order quantities, minimum order values, and fixed weekdays for placing orders. According to company executives, the quality of forecasts' varies widely based on the dispatcher’s effort and experience. Learning to forecast takes a long time. Due to the German state's federal structure, public holidays and holiday periods for schools differ across the 16 federal states, which makes taking the local conditions into account a non-trivial task. Moreover, the lack of a structured process makes forecasting hard to scale. In conjunction with the high costs for inaccurate forecasting, the company has a genuine interest in optimizing its forecasting process.

5.2 DATA DESCRIPTION

For this experiment, the company provided comprehensive data on their fresh product segment. The data can be roughly divided into historical data, information on the logistical processes, and lastly, product individual article master data and logistical properties.

The historical data covers the time of the 01st March 2018 to 31st March 2020, which was the maximum time range retrievable from the system at that point. The data is from the five central warehouses the company operates in Germany, serving both customers and other intermediate warehouses in the supply chain. Three data sets were provided, the first of which reports daily sales and the number of discarded products. The second data set reports customer orders for products and the respective degree of fulfilment. The third historical data set reports past promotional activities for the product.

Information on the logistical processes come in the form of location-specific replenishment schedules. Each schedule provides information on the delivery lead time, possible order days, and order restrictions for each listed supplier. Another dataset contains a list of ZIP code areas to which the warehouses supply.

Finally, the company provided several data sets containing general information and information on every SKU's logistical properties. The master data set contains each product's IDs, its supplier, and a product description text. Some products are sold in bundles, meaning a sales unit consists of a fixed number of product units. This information was extracted from the article description text using regular expressions as filters. On top of that, for every product with a date of expiry, the maximum acceptable storage time is specified. Information on product weight, price, and size of some products round up the data set.

External data supplemented the company-internal data. Company executives suggested that the weather might influence sales. As discussed in chapter 3.2.2, including weather information has been shown to improve forecast quality. Steinker et al. (2017) have shown that while the weather within Germany varies a lot between different locations, the weather within a federal state is relatively homogenous. Records at a single location can thus represent the weather in each state. To model weather, historical observations from 16 weather stations were obtained from the German weather service DWD. The 16 weather stations correspond to one federal state and are the ones used in Steinker et al.'s model.

It is worth noting that for this experiment, historical records of weather are used rather than historical weather forecasts. In practice, weather records would not be available when the forecast is made, but only weather forecasts. Using records might bias the analysis, as the quality of weather forecasting typically depletes with the forecasting horizon. The ultimate decision to use historical records instead of forecasts was made based on the following rationale: The required forecasting horizon for most products is less than a week. Within a week, weather forecasts are relatively accurate. Moreover, obtaining historical forecasts is a non-trivial task and would increase the complexity of the data. If the inclusion of actual weather data proves to improve the forecasting accuracy, a subsequent experiment with weather forecasts instead of records is possible.

Table 1: Overview of data used for the experiment

Data Set	Content
Historical sales	Daily sales, discarded units, inventory differences per product by location
Customer orders	Daily orders placed per products by location, degree of order fulfillment
Promotional Activities	Promotional activities for the fresh products segment
Logistical schedule	Supplier specific order days, lead time and order restrictions for every warehouse location
Postal Code Mapping	Mapping of head warehouse to warehouse, list of postal codes of respective customers
Article Master Data	Product specific Product ID, article description, supplier, max time in warehouse, weights
Product Price	Price in € for selected products
Product Size Information	Units per palette/box for selected products
Weather data	Historical records for average temperature, hours of sunshine and amount precipitation from 16 weather stations
School holiday periods	List of school holiday periods by German Federal State for 2018 to 2020
Holidays	List of Holidays by German Federal State

The full data set provides observations for 14 metrics at every station. To avoid problems with high dimensionality, only three of these are used in the final model: daily average temperature, hours of sunshine, and the total amount of precipitation. The python package 'Holidays' provided a list of federal state-specific holidays in Germany. School holiday periods for every federal state in the years 2018 to 2020 were taken from a web service and collected in an Excel file. **Error! Reference source not found.** provides an overview of all the data used for this experiment.

5.3 DATA CLEANING

The data cleaning process was two-fold: On one hand, it involved making the data machine-readable. The original data came in mainly unstructured formats in CSV and Excel files. The data format was standardized to pickled data frame objects of the python library pandas, by applying appropriate formatting and a standard naming convention to features.

An initial visualization of sales data revealed outliers in the data that would distort the analysis, implying the data needs to undergo some preliminary filtering. A rolling window filter identified outliers in the sales records using z-scores. The sample mean $\bar{x}$, and the sample standard deviation S were calculated for a rolling window of 20 steps. The z-score for an individual data point x was given by:

$$z = \frac{x - \bar{x}}{S} \qquad \textit{Eq. 28}$$

A z-score of 4 marks the threshold for data points to be an outlier. Applying the filter resulted in a list of 45 data points, which a dispatcher doublechecked. The dispatcher confirmed the majority of them to be mistakes in the data. These faulty data points are likely the result of human error during data acquisition. The identified faulty data points were eliminated from the data set.

Furthermore, during data cleaning the data underwent some preliminary filtering, the goal of which was to drop the time series which had insufficient information to be used in the experiment. In the historical sales data, sales for 1029 SKUs were recorded at up to five locations over two years. This amounts to a total of 3882 unique time series, i.e., unique combinations of SKUs and locations. Not every product is delivered from every logistic centre, thus the discrepancy. Of the articles, 146 were identified as drop shipment articles by their product description. The company asked to exclude these from the experiment upfront. A total of 3408 unique time series remained.

The time range from 12.01.2019 to 31.03.2020 constitutes the testing phase, for which the model is supposed to forecast demand. This way, the model makes out-of-sample forecasts for about 20% of the available historical records. Moreover, due to the German holiday season around Christmas and New Year, many sales anomalies occur in this time,

making it a suitable period to evaluate whether the model is capable of taking these into account. The remainder of the historical data serves as training data to fit and optimize models.

The product range varies over time as some products are added to and dropped from the product range. In the testing set, there are 1438 series more than ten events of sales. Only these are kept from the original data set to exclude ultra-short time series. Moreover, to fit the model, a minimum amount of observations during the training phase is necessary. The threshold for this minimum amount is at 20 observations. Finally, for a small number of the products remaining in the testing set, no reliable information on best before date or logistical properties was available. They were excluded from the analysis as well.

In total, for the final experiment, 1318 unique time series were considered. While this might seem like only a small subset of the roughly 3400 series in the original data set, the considered time series make up about 94% of the sales in the segment during the testing periods. Thus, they still constitute a representative sample of the overall sales.

5.4 PRE-PROCESSING

The data underwent several pre-processing steps to make the input suitable for the Neural Network. This section describes all the steps taken.

First, in order to make the observations in the time series equidistant, data was resampled to include one observation on every business day (Monday to Friday). The rare case of demand on Saturdays (due to delivery shifts after holidays) is treated as if the demand had occurred on Friday.

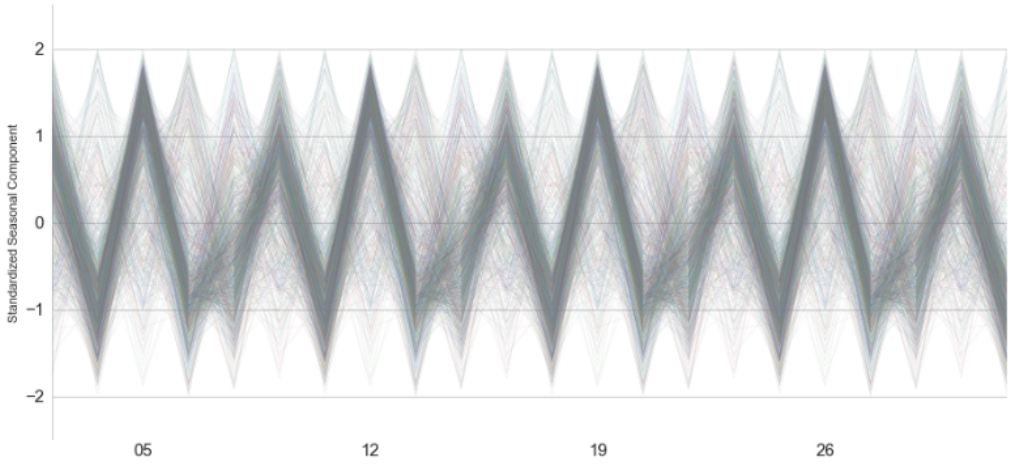

Figure 5: Standardized seasonality component for every individual time series in October 2019.

For prediction, the model uses engineered features and exogenous features. Apart from the actual sales of the product, several categorical variables are passed to the model as covariates. These categorical variables include the product segment, the location where the product is sold, and an index unique to every series made up of the location name and the product ID, from which the time series can be identified.

The forecasting literature suggests that Neural Networks provide better forecasts when the original time series is deseasonalized (Bandara et al., 2020). Figure 5 plots the standardized seasonality component, as extracted from trend-seasonality decomposition, for every single series. It shows that the overall seasonality pattern is quite regular among the series. Following a suggestion by (Barker, 2020), the model is thus supposed to learn the seasonality with the help of date features. Considering that most of the time series exhibit a similar weakly seasonal pattern, the model is more likely to succeed here than in forecasting series that exhibit varying seasonality, which Bandara et al., (2020) are dealing with. For this reason, additional features for the year, month, day of month and weekday were created.

In order to leverage the hierarchical structure of the sales data, sales of related products were included at different levels of aggregation. For every time series, apart from historical sales, the following observations were included: Total sales of the SKU at every location, total sales of the product segment, and total sales at the location. Apart from daily records of these values, biweekly means for every series were provided.

In a traditional regression model, the inclusion of such features would be complicated, as future values are unknown. To incorporate future values into the model, they would have to be forecasted themselves, introducing additional sources of error to the forecast. The Seq2seq-structure of the model allows for including different features for the past and future. Thus, future values of related time series need not be known for them to be included in the model.

The exogenous features from the weather observations and the holiday data were added to the model. An overview of all the features used by the Neural Networks is given by Table 2.

Table 2: Overview of features used by the Neural Networks

Feature	Description	Used in
Time Series Observation	Absolute Sales per SKU per Location	Encoder
Categorical Data	Product Segment, Location and Index	Encoder & Decoder
Date Features	Year, Month, Day, Weekday	Encoder & Decoder
Related Product Sales	Daily and two-week-average of corresponding SKU, product segment, and location	Encoder
Promotional Activities	Binary variable: promotional campaign active?	Encoder & Decoder
Weather Data	Sunshine hours, average temperature and precipitation recorded at 16 weather stations	Encoder & Decoder
Holiday	Sixteen binary variables: holiday?	Encoder & Decoder
School Holiday Period	Sixteen binary variables: school holiday period?	Encoder & Decoder

Categorical variables with a low amount of categories per feature were encoded using one-hot-encoding (OHE). OHE turns a categorical feature into a vector of binary variables with one entry for every possible category. The vector consists of zeros and ones, where one indicates that an observation falls into a specific category. OHE encoded features include the location (five categories), the product segment (five categories), the year (three categories), and the weekday (five categories) of the observation.

OHE-encoding the time series index (1318 categories) would have drastically increased the model complexity; thus, the index was passed to an embedding layer, as described in chapter 4.2. The month and day of the month were encoded as combinations of sine and cosine values. This approach signals to the model that January and December fall in the same season, even though their numerical encoding (1 and 12) suggests they are far apart.

All non-encoded additional variables, namely the engineered features for corresponding product groups and the weather data, were standardized to zero mean and unit standard deviation, as suggested by Wen et al. (2017). For the actual values of the time series, (Salinas et al., 2020) suggest letting the model learn the scaling itself, which means that the model learns to scale the time series at the input layer and inverses the scaling at the output layer. Learning the scaling complicates the training process, but preliminary testing results solidified the notion that scaling should not be handled as part of the pre-processing. While preliminary scaling accelerated the convergence in training time significantly, the resulting model yielded poor results on validation data.

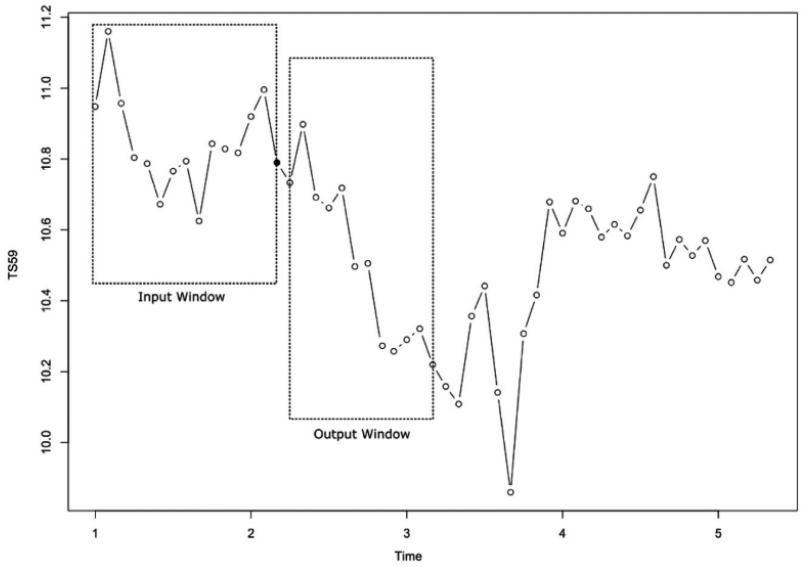

Figure 6: An example of applying the moving window approach, adapted from Bandara et al. (2020)

In order to train the models, each time series in the original training set was turned into multiple samples by applying the moving window approach. (Bandara et al., 2020; Salinas et al., 2020). While keeping the original order of observations, two fixed-length windows are used to sample input and output sequences from the original time series. Figure 6 illustrates this process. The windows are moved one step ahead until the output window reaches the last date in the time series. Applying this approach to the 1318 time series yielded 446,862 training samples.

The full pre-processing process is documented in Appendix A.

5.5 IMPLEMENTATION

The following section describes how the models were implemented.

5.5.1 Neural Network Models

In total, four Neural Networks were trained:

- A Seq2seq model for forecasting period-specific demand
- A Seq2quant model for forecasting the medium cumulative demand
- A Seq2quant model for forecasting 0.8-quantiles of cumulative demand
- A Seq2quant model for forecasting 0.9-quantiles of cumulative demand

The model consists of an encoder and a decoder part, each of which consists of stacked LSTMs. The model takes three sets of inputs: The first input — the time series index — is passed to an embedding layer that maps the index to a multidimensional vector. The encoder input is a sequence of past observations and additional features for 15 periods back. The decoder input is a sequence of deterministic future features for 15 periods forward. Additionally, both the encoder and the decoder process the embedding layer's output vector at every time step. The final layer is a time-distributed dense layer that applies a ReLu-activation to the output. The activation ensures that the output is non-negative, and allows the network to predict the demand as precisely zero.

During preliminary testing, several model architectures were tested and evaluated on the validation set. The final model was supposed to predict 15 business days ahead in time, as some products' orders are placed only once a week, sometimes covering the replenishment time for two weeks, plus the lead time. Bandara et al. (2020) recommend choosing the input sequence length as 1.25 times the output sequence length, or at least as long as the output.

Table 3: Hyperparameters for the implemented models

encoder length	15 periods
Decoder length	15 periods
# LSTM layers	3 encoder layers, 3 decoder layers
# LSTM nodes	40
# of training samples	446,862
# of validation samples	9,125
Batch size	64
Optimizer	Adam optimizer with a learning rate of 1e-3

To date, there is however no dedicated analysis of the optimal input-to-output length ratio. In preliminary testing, an input sequence of 15 steps yielded the best results. Table 3 summarizes the ultimate choice for the hyperparameters. It is partly inspired by a similar setup that has been proposed by Salinas et al. (2019) in an experiment with an equally large number of time series and training samples. The suggested architecture and the hyperparameters are applied to all four networks. Figure 7 on the following page illustrates the model structure.

Models were trained for a maximum of 20 epochs (i.e. iterations over the training set) with mini-batch learning. Early stopping was applied, which means that the training process is stopped when the loss on the validation set does not improve for two consequent epochs. A dropout layer with a dropout probability of 0.2. in both the encoder and the decoder part of each model prevented overfitting.

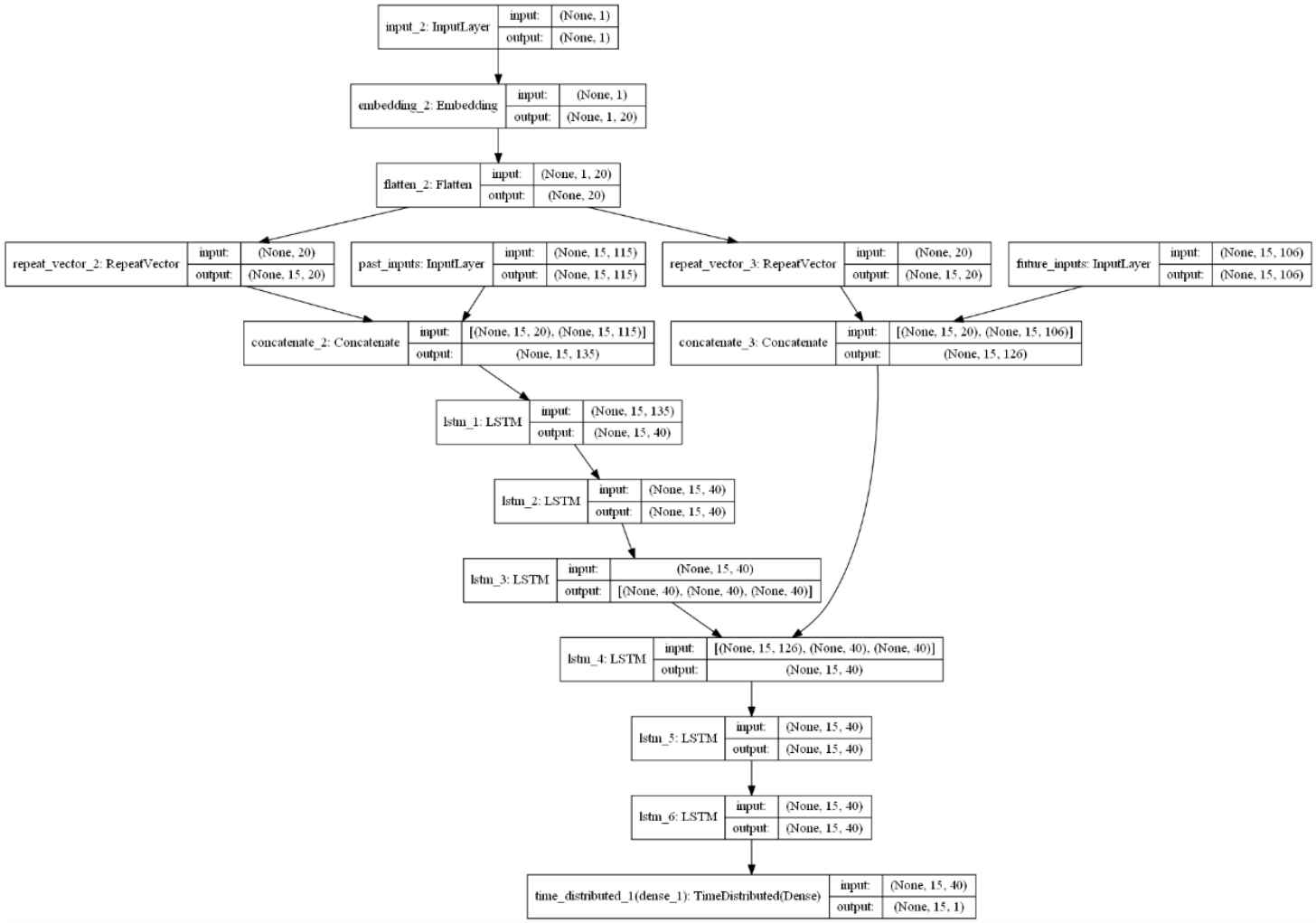

Figure 7: Tensorflow implementation of the Seq2Quant model

In an initial test, the Seq2Quant models failed to converge. To facilitate learning, each model reused the weights learned by its predecessor for initiation ('warm-starting'). This training scheme reduces the complexity of the problem to learn for each model and accelerates the training process. The Seq2seq model needs to learn to predict the future demand. The Seq2Quant_Median model learns to add up demands over periods. The remaining models learn to adapt these cumulative demand forecasts to overpredict demand on average.

The Neural Network Models were implemented in Python 3.7 using TensorFlow 2.1.0, using the Keras library (version 2.3.1). All models were trained on an HP laptop running on Windows 10 and using an eight-core Intel i5-8265u CPU. With this set-up, training the first two models took approximately 24 hours, whereas the other two took 14 hours and 7 hours due to warm-starting. The code for the implementation can be found in Appendix B.

5.5.2 **Baselines**

The experiment compares the performance of the Neural Networks against the following baselines:

- Moving Average Forecast
- Naïve Seasonal Forecast
- Naïve Seasonal Forecast with Averaging over four periods
- ARIMA
- SARIMAX

The Heuristic models approximate the current practice of manual forecasting by the company employees, i.e. the Naïve models for time series with steady demand and the Moving Average approach for when demand is erratic and follows no pattern. The ARIMA models provide a state-of-the-art statistical baseline. The SARIMAX model is an

extension with exogenous variables to show how external demand drivers can be accounted for in the statistical forecasting framework.

The simple heuristics, i.e., the Moving Average and Seasonal Nãive forecasting approaches, were implemented as Python function. Their code can be found in the Appendix. The ARIMA and the SARIMAX models were implemented via the objects provided by statsmodels-library (v.0.10.1). The models are fitted with the help of the pmdarima library (v.1.6.1), a wrapper for the statsmodel implementation mimicking the behaviour of the famous R-function auto.arima from the forecast package (Hyndman, Rob J. and Khandakar, 2008). The function fits several models with different orders and chooses the model that minimizes the AIC (c.f. Chapter 2.3.2) for the training period. Stepwise search is applied, with a maximum order of $ARIMA(10, d, 10)$ and $ARIMA(10, d, 10)(4, D, 4)m$ respecitvely. The m component (seasonality) of the SARIMA models was fixed as 5, as the time series come in business day frequency. The function also applies appropriate differencing

Furthermore, the SARIMAX models took the following features as exogenous inputs: weather data, holidays, vacation periods, and promotional activities. The high dimensionality of the exogenous covariates would have made fitting the model series with little observation impossible, as there would be fewer observations than variables, hence no degrees-of-freedom. To prevent this, the SARIMAX model uses only a subset of the data. This subset includes only observation from federal states that are relevant for the location the forecast is made for. The relevant federal states for each location were identified from ZIP codes of each central warehouse's clients.

After fitting on the training period, every method produced forecasts for 15 steps ahead for every date in the test period, having access to all the observations available up until this point. The code for the implementation can be found in Appendix C.

5.6 INVENTORY SIMULATION

In response to the call for taking utility values from inventory management into account for evaluation (c.f. Chapter 3.2), the experiment involves a close-to-reality simulation of inventory development. The simulation covers the test period time from December 2019 to March 2020 and uses actual sales data. However, orders and stocks are calculated based on forecasts generated by the models described in the previous section.

The inventory management in this simulation follows a periodic review policy, as outlined in Chapter 2.1. Demand occurs every business day and is served immediately. To approximate the case company's real-world conditions, the experiment settings impose two forms of restrictions on the orders: order date restrictions and order quantity restrictions.

Though the company monitors stock development daily, daily orders are not always possible due to order date restrictions. Some suppliers only deliver products on certain weekdays. Thus, the review period differs between suppliers, and sometimes even varies over time for different suppliers. As a first step, the simulation algorithm calculates every possible order date and the corresponding replenishment time the order needs to cover.

Subsequently, on every possible order day, the order-up-to levels for every product are determined. The order-up-to level is based on forecasts for the demand during the replenishment time, plus safety stock. For the baselines, the safety stocks are obtained from the forecast error, which was estimated from their performance during the month

before the test period (baselines) or their fitting residuals (ARIMA and SARIMAX). The Seq2Quant models incorporate the safety stock in their forecast, hence no additional calculation is necessary. The ordered amount is calculated by subtracting the current stock and the pipeline stock. Orders are rounded to sales unit level (i.e., the bundle of units the SKU is sold in). Orders placed arrive after a supplier-specific lead time, and in the meantime are considered part of the pipeline stock.

For some suppliers, orders are subject to additional restrictions concerning the order quantity. In the data set provided, three forms of order quantity restrictions were present: Minimum order values, minimum order weights, and minimum order volumes. Upon calculating all products' optimal orders for a supplier, the simulation algorithm checks whether the ordered amount is sufficient to fulfil the restrictions. If not, the algorithm extends these products forecasting horizon so that it covers the demand for the following replenishment period as well. Consequently, orders for subsequent replenishment periods are brought forward. This process is repeated as often as necessary, but never beyond a forecasting horizon of three weeks.

A crucial aspect of the evaluation is the amount of waste. Waste occurs when products exceed the maximum acceptable time in the warehouse, and are thus unsellable. During the simulation, the algorithm tracks incoming order batches per SKU independently. The expiration date is set to the arrival date, plus the maximum acceptable time in the warehouse. For products with no expiration date, the maximum acceptable time in the warehouse was defined manually, based on company employees' estimations. Sales are first served from the oldest order batch, following a first-in-first-out approach. At the beginning of each period, all remaining stocks of batches exceeding their expiration dates are discarded.

The simulation tracks the stock level at the beginning and end of the period, the number of discarded products, and the demand fill-rate on a product level. The simulation runs independently for each of the forecasting models introduced in the previous section. In total, the simulation is repeated three times, for the target service levels of 50%, 80%, and 90%. The full code for the simulation algorithm can be found in Appendix D.

6 RESULTS

This chapter summarizes the results of the experiment. The first section evaluates the models based on performance metrics, the second section based on the results of the inventory simulation.

6.1 DEMAND FORECASTING

This section compares the results for forecasts of exact demand. The Seq2Quant-models forecasts cumulative demand; thus, this analysis excludes them. For each day in the training period, the models produced 15-day-ahead forecasts. Table 4**Error! Reference source not found.** summarizes the results for two performance metrics: The MAE and the SMAPE, as introduced in Chapter 2.2. The metrics were calculated across all forecasting dates and forecasting horizons for every unique time series individually. The Average Value describes the mean of the metric over all products and locations. The Average Rank is the mean rank a model achieved within a ranking of models for time series.

Table 4: Mean Performance Metrics for 15 step-ahead forecasts

Model	MAE		SMAPE	
	Average Value	Average Rank	Average Value	Average Rank
MA	21.74 (6)	4.57 (6)	80.38 (6)	4.45 (6)
Naïve	19.97 (5)	4.54 (5)	70.44(3)	4.34 (5)
Naïve A.	18.42 (4)	3.11 (4)	69.19 (2)	3.24 (3)
ARIMAX	18.09 (2)	2.95 (2)	76.87 (4)	3.17 (2)
SARIMAX	18.33 (3)	2.73 (1)	79.68 (5)	3.60 (4)
Seq2Seq	17.44 (1)	3.08 (3)	66.91 (1)	2.21 (1)

In terms of average values over all models, the suggested Seq2seq model is the most accurate. Based on the SMAPE rank, it also seems to be among the best models for most series. Interestingly, judgment based on the MAE favours the use of statistical methods, whereas the SMAPE favours the Naïve Methods. The Average SMAPE weights all the time series equally, as the metric is standardized between 0 and 200. The MAE, on the other hand, is not standardized; thus, single series with large volumes have a higher impact on the overall results. Intuitively, the statistical models might perform better on single series with high, regular demand, skewing the metric in their favour. In terms of all metrics, the simple moving average approach is the worst method.

Judging from the ranked values, there is no single dominant model. Even the Moving Average, which ranked last in every metric, was found to yield the best forecasts for a small group of products. The relatively good performance of the Seq2seq model can be attributed to its ability to model multiple behaviours at once, due to its embedding layer. Nevertheless, the question remains whether the machine learning model would perform better on more homogenous sets of demand.

Figure 8 shows the average SMAPE of all models for forecasting *n* steps ahead. As expected, the forecast accuracy decreases with the forecasting horizon. Seq2seq dominates for the majority of forecasting horizons. For short-term forecasts of less than a week ahead, the Seasonal Naïve method achieves a slightly better mean SMAPE. However, the accuracy of the method deteriorates for forecast horizons of more than a week.

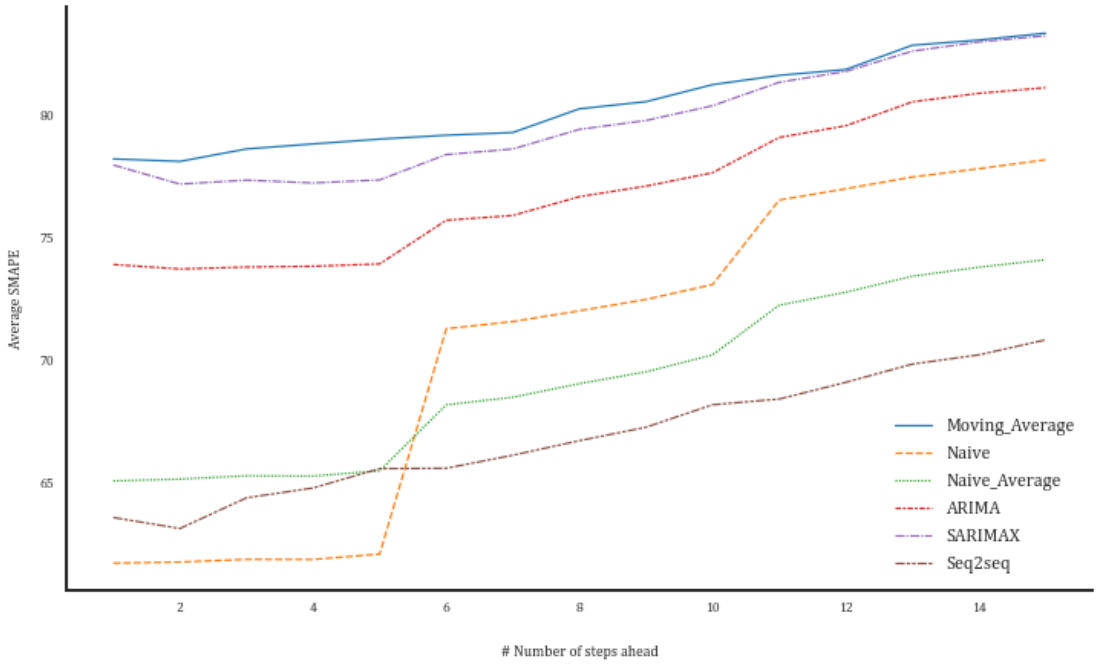

Figure 8: Mean SMAPE per method over the forecasting horizon

Note the apparent stepwise curve of the Naïve method (in orange). Averaging over multiple weeks smoothens out the curve (as in Naïve_Average, in green), but sacrifices short-term accuracy. Like the Näive forecasts, the ARIMA and Naïve Average methods show step-wise behaviour, with spikes after every seasonal cycle. The curve for Seq2seq resembles the Moving Average curve best.

6.2 INVENTORY SIMULATION

Table 5 and Table 6 summarize the results of the inventory development simulation. In the first set, where a target service level of 50% is applied, effectively no adjustments are made. Interestingly, all of the models reach comparatively high service levels, even without any adjustment.

Table 5: Results of the Inventory Simulation for different target cycle service levels, part 1

Model	Target CSL					
	50%		80%		90%	
	α	β	α	β	α	β
Moving_Average	0.76	0.84	0.89	0.92	0.9	0.92
Naive	0.76	0.83	0.88	0.91	0.9	0.91
Naive_Average	0.76	0.84	0.87	0.91	0.89	0.91
ARIMA	0.78	0.86	0.9	0.92	0.91	0.92
SARIMAX	0.79	0.86	0.89	0.91	0.9	0.92
Seq2seq	0.82	0.87				
Seq2Quant_50	0.81	0.87				
Seq2Quant_80			0.84	0.89		
Seq2Quant_90					0.86	0.9

Table 6: Results of the Inventory Simulation for different target cycle service levels, part 2

Model	Target CSL					
	50%		80%		90%	
	Lost Sales	Waste	Lost Sales	Waste	Lost Sales	Waste
Moving_Average	449.121	141.894	237.561	300.458	225.457	359.917
Naive	594.331	135.588	270.947	301.462	248.413	367.197
Naive_Average	513.302	154.614	305.859	297.726	287.172	353.004
ARIMA	529.092	136.521	315.452	234.564	288.009	273.137
SARIMAX	449.015	157.706	262.073	281.310	245.329	330.413
Seq2seq	398.193	149.700				
Seq2Quant_50	434.608	153.627				
Seq2Quant_80			400.090	193.934		
Seq2Quant_90					368.901	215.396

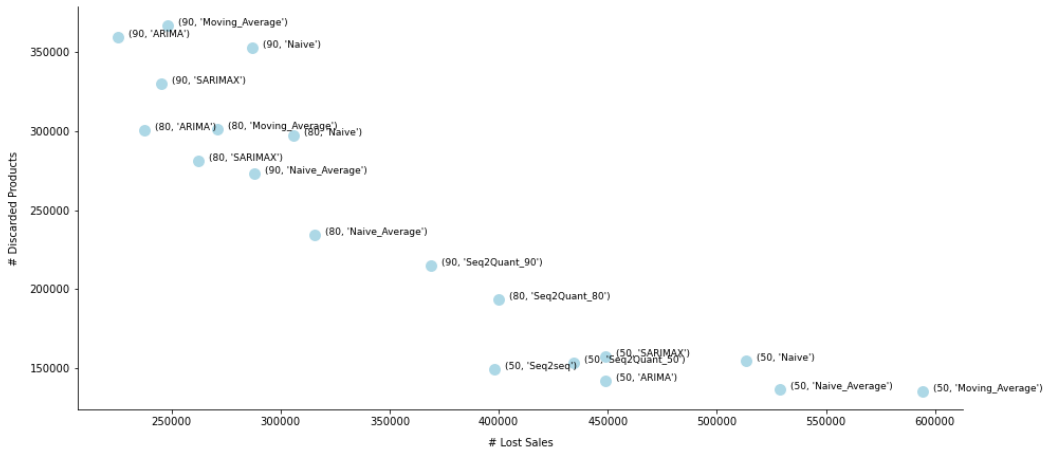

Figure 9: Ratio of lost sales nnumber of overall discarded Products for (Target CSL, Model Name)

Note that the four Neural Networks are listed independently, as they are technically four models. The model did indeed adapt forecasts to account for the increase in target service level, which is reflected by the increase in the realized service level alpha. For the target CSL of 80%, the SeqQuant_80 is the most conservative method, reaching the lowest CSL but also the lowest level of waste. The baselines, however, massively overpredict and overshoot the 80% target by a margin. This implies that the calculation based on the normality assumption of forecasts error is inadequate.

However, the Seq2Quant_90 fails as only one of two models to reach the target CSL of 90%. In terms of the fill rate, or beta level, all models are on par. It appears that the Quantile Loss function did not sufficiently incentivize the model to adjust the prediction.

Figure 9 visualizes the results of Tables 5 and 6 by plotting the number of lost sales against the number of discarded products over the testing period. This visualization allows for checking for efficiency. A method is efficient if there is no other method that achieves better results for both utility values at the same time.

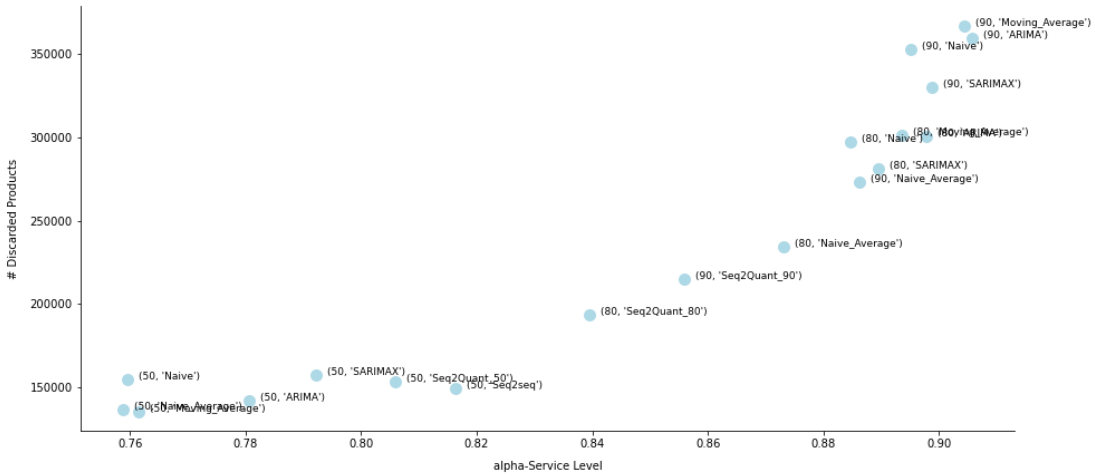

Figure 10: Ratio of realized CSL to number of overall discarded products for (Target CSL, Model Name)

Since both values need to be as low as possible, the Seq2Quant_80, Seq2Quant_90, and the Seq2seq model are all part of the efficient frontier, i.e., the set of efficient methods. Among the models without safety stock adjustments, the Seq2seq model dominates the Seq2Quant_50, the SARIMAX and the Naïve model. This is somewhat counterintuitive, as the Seq2Quant_50 was explicitly trained to forecast cumulative demand. The Seq2seq model also achieves better lost sales values at only a slightly higher level of waste than the ARIMA, Naïve Average and Moving Average models.

Judging from the shape of the efficient frontier, it appears, however, that the Seq2Quant_80 and Seq2Quant_90 are part of it, but have slightly worse ratios than other efficient methods. The comparatively simple seasonal Naïve Average models are part of the frontier too. Plotting the product waste against the CSL (Figure 10) reveals a similar image. As before, the Seq2seq-model dominates or nearly dominates other solutions with the target-service level of 50%.

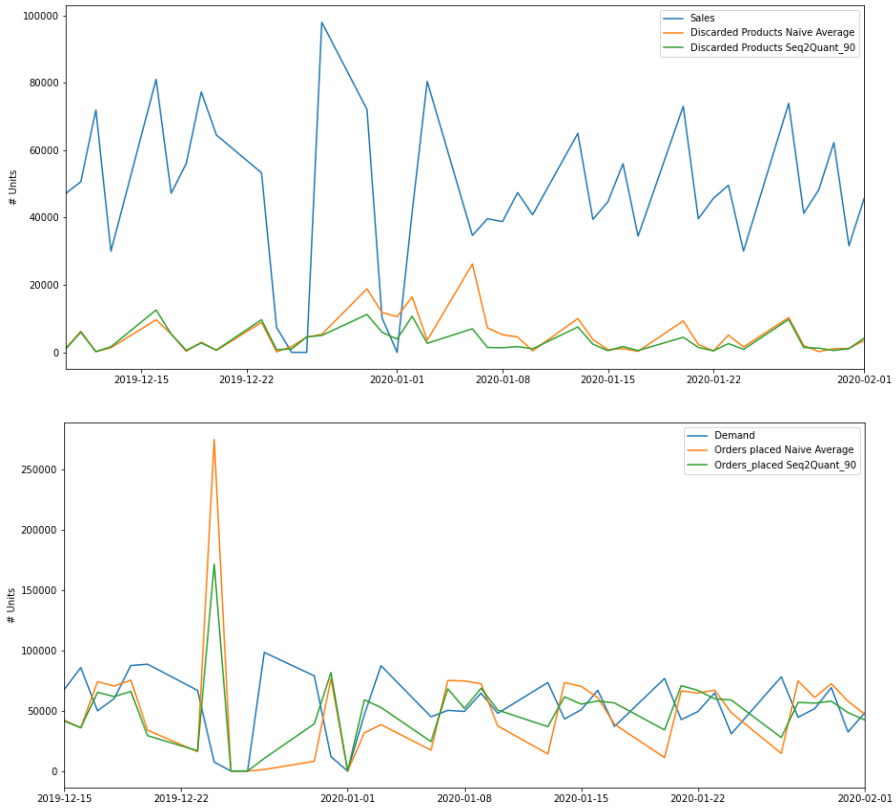

Figure 11: Exemplary Plot of Sales, Orders and Discarded Products for two models from the 90% target CSL simulation

The primary value of the Seq2Quant models in this setup is that they reached tradeoffs between product availability and waste that traditional methods did not. The question remains whether the baseline models could achieve similar trade-offs with more accurate methods to determine the safety stocks.

Finally, the results provide some evidence for the value of intelligent models or models that are augmentable by external features. The upper plot of Figure 11 displays the development of the daily number of overall waste for two methods: The Naïve Average (90) and the Seq2Quant 90. The blue line plots total sales. It is added for visualization purposes because it shows the massive irregularities in demand during the holiday season.

The Naïve Average method has no way to model the extraordinary case of holidays. Thus the irregularities in the sales skew its forecasts, leading to high amounts of waste. For the Seq2Quant model, on the other hand, the amount of discarded products remains relatively stable over time, even during the holiday period.

The lower part of the figure plots the orders placed. Note that due to the different scale, the blue line (sales) appears squished. Under the Naïve Average model, the simulation algorithm placed higher orders before Christmas. It likely misinterpreted the elevated sales in the pre-Christmas week and thus placed higher orders than necessary, which ultimately leads to products wasted. On the other hand, this implies that the Neural Network learned successfully to interpret the demand irregularities during the holidays.

7 DISCUSSION AND CONCLUSION

This study shows that Neural Networks are a promising approach to deal with large-scale forecasting problems for perishable products. Forecasting SKU-level demand for perishable products is a challenging task: time series for demand are volatile, skewed, subject to external factors, and frequently consist of only a few observations. Furthermore, SKU-level demand forecasts are typically used for inventory management, which imposes additional requirements on the forecasting procedure, namely scalability, proper uncertainty estimation, multi-step-ahead accuracy and the inclusion of external demand driving factors.

The suggested Seq2Quant model seeks to address these issues directly by allowing multiple types of external variables as inputs, and by predicting demand quantiles directly. To avoid common problems of forecasting with Neural Networks, the model learns globally from all the available time series. A large-scale experiment on the fresh products segment of a German wholesaler tested the model's performance on real-world data.

The main finding is that based on forecasts for over 1300 unique time series of demand at the SKU-level, the base form of the suggested model for accurate daily demand forecasting (Seq2seq) yielded superior results to all of its competitor baselines in terms of SMAPE and MAE. It outperformed all other methods for forecast horizons of more than a week, demonstrating the success of forecasting methods optimized for multiple-step-ahead predictions.

In terms of inventory performance, the results are mixed. The performance of the basic Seq2seq model without Quantile Regression translated directly to superior trade-offs for product availability and product waste over other baseline methods without safety-stock

estimations. There is, however, little evidence to prove that the Seq2Quant approach yields superior forecasts of demand quantiles. Totalled across all locations, the Seq2Quant model failed to reach the highest of the pre-defined target cycle service levels, 90%. On the other hand, Seq2Quant Networks found a fair tradeoff between product availability and product waste. None of them was dominated by a baseline model in terms of service level to discarded products ratio. Furthermore, the simulation provided evidence for the capability of the network to handle exogenous demand drivers appropriately.

These findings provide further evidence for the promising forecasting capabilities of global sequence-to-sequence models. They lend support to, for example, Salinas et al. (2019) and Wen et al. (2017), who showed that global models improve forecasting for SKU-level demand. Moreover, the findings extend these studies by comparing the model against statistical baselines, and show that Machine Learning models perform best when additional features and sufficiently large data sets are available, as suggested by Barker (2020).

In contrast to Huber et al. (2019), this study found the integrated quantile regression approach not to yield superior inventory performance. Note, however, that Huber et al. studied a comparatively more straightforward one-period-newsvendor problem. The Seq2Quant model in this study dealt with several multi-period prediction problems at once. Hence, the findings of this study do not necessarily contradict earlier findings on Quantile Regression, but may imply that the problem was too complex for the model to learn.

The question of why the Seq2Quant model did not reach the target service level of 90% remains. In this setting, the learning algorithm had to be optimized for 15 quantiles at the

same time. Non-linear programming is, in general, not easy, so shortening the output sequence to reduce the complexity of the problem might be a first step.

The practical implications of the study are two-fold: First, the results seem to indicate that global Neural Network models are a promising approach to deal with large scale demand forecsting problems of food products at the SKU-level. However, this study finds no sufficient support for integrating the safety-stock estimation into the prediction process.

On the other hand, though theoretically appealing, well-known drawbacks of Neural Networks remain a problem: namely their poor interpretability due to their black-box-nature, and their high computational complexity. From a practical perspective, the poor interpretability is secondary; daily forecasts for several hundred SKUs leave little room for in-depth evaluation, so from the practitioners' perspective accuracy trumps interpretability. Nevertheless, the poor interpretability might be problematic in two ways: First, if the model provides poor forecasts, it may be hard to pinpoint the reason for the subpar performance. Second, as a direct consequence of this, building trust into this type of complex Neural Network is hard, which may prove to be a burden in fully automating such a crucial (and potentially expensive) aspect of inventory management.

Another burden for adaptation in practice is the effort for setting up and maintaining the network. This is mainly the result of the high computational effort for training the model. The global modelling approach reduced the computational effort immensely, making it possible to train a large-scale forecasting network for several hundred time series on a regular laptop within a day. Still, in order to set up the network, relevant data had to be collected, cleaned and pre-processed. Preliminary testing to determine the most suitable structure and the set of hyperparameters was time-consuming as well. In total, the process took several months. Putting a network into production would require a high

degree of automation of the above-mentioned steps. Supply Chain Managers must weigh these additional efforts against the potential accuracy gain that comes with the large-scale and long-term implementation of a Neural Network. This study has provided some evidence for the competitiveness of Neural Networks, but the gains over the competing forecasting methods were overall not substantial.

However, this study is exploratory in many regards. Global modelling is a relatively young phenomenon in forecasting. Moreover, to the author's best knowledge, this is the first sequence-to-sequence model that forecasts the quantiles of demand simultaneously. This project was subject to time and computational constraints. Learning algorithms for Neural Networks are somewhat stochastic; thus it is usually worthwhile to train several versions of the model while altering the hyperparameters. The search process for the hyperparameters in the present study was not extensive and lacked a systematic structure, which leaves room for potential improvements.

Likewise, the baseline methods were fit by automatic algorithms. The author tried to ensure a fair comparison against proper baselines. They were, however, not the primary focus of this work, so it cannot be ruled out that another established forecasting method might achieve better results on the data. Another limitation was that this experiment used historical weather records when they were technically not yet available. Further analysis should seek to replicate these results using historical weather forecasts instead.

The limitations of the present study are closely linked to suggestions for further research. Obviously, the expansion of the input data set comes to mind. This refers to both the number of features, and the length of the historical records. The model was trained on only a bit more than one-and-a-half years of data. Many annual events like Christmas thus only appeared once in the data. Moreover, the model was trained over-proportionally on

data from summer months, which have higher sales on average. Extending the number of observations is desirable, as Neural Networks are Machine Learning models after all.

The findings of this study imply, that there is no single dominant forecasting method for all products. Having one global model for one time series is convenient, but as mentioned in the literature review, Bandara et al. (2020) found evidence for the idea that segmentation improves the forecasts of global Neural Network models. A forecasting method should thus first seek to identify clusters of similar time series in the data, and then model multiple global models on these clusters. Applied to this case, this could involve clustering time series by mean sales or velocity, and fitting (smaller) models on them individually.

In the operations research context, some of the ideas used in the Seq2Quant network can be applied in research on data-driven newsvendor problems. This study demonstrated how Neural Networks could forecast sequences of (cumulative) demand quantiles. This idea lends itself to solving multi-period newsvendor problems. Possible further studies could focus on applying this approach to inventory management problems to other industries.

In conclusion, Neural Networks for demand forecasting remain an exciting field of research. Despite decades of prior work, researchers keep uncovering new application domains for Neural Networks. This study has shown that operations research has not yet utilized their full potential for forecasting extensively. It will be exciting to witness where the current advancements in Machine Learning will take us over the next years.

APPENDIX

This thesis is supplemented by an electronical Appendix. It consists of parts of the code for the experiemnt, namely:

A: Pre-processing for Neural Networks

B: Neural Network Training

C: Baseline Training

D: Inventory Simulation.

The electronic Appendix that can be obtained by contacting the author of this thesis.

BIBLIOGRAPHY

Aburto, L., & Weber, R. (2007). Improved supply chain management based on hybrid demand forecasts. *Applied Soft Computing Journal*, *7*(1), 136–144. https://doi.org/10.1016/j.asoc.2005.06.001

Adya, M., & Collopy, F. (1998). How effective are neural networks at forecasting and prediction? A review and evaluation. *Journal of Forecasting*, *17*(5–6), 481–495. https://doi.org/10.1002/(sici)1099-131x(1998090)17:5/6<481::aid-for709>3.0.co;2-q

Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, *19*(6), 716–723. https://doi.org/10.1109/TAC.1974.1100705

Ali, Ö. G., Sayin, S., van Woensel, T., & Fransoo, J. (2009). SKU demand forecasting in the presence of promotions. *Expert Systems with Applications*, *36*(10), 12340–12348. https://doi.org/10.1016/j.eswa.2009.04.052

Armstrong, J. S. (1978). *Long-range forecasting: From crystal ball to computer*. John Wiley & Sons.

Arunraj, N. S., & Ahrens, D. (2015). A hybrid seasonal autoregressive integrated moving average and quantile regression for daily food sales forecasting. *International Journal of Production Economics*, *170*, 321–335. https://doi.org/10.1016/j.ijpe.2015.09.039

Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, *140*. https://doi.org/10.1016/j.eswa.2019.112896

Barker, J. (2020). Machine learning in M4: What makes a good unstructured model?

International Journal of Forecasting, *36*(1), 150–155. https://doi.org/10.1016/j.ijforecast.2019.06.001

Ben Taieb, S., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, *39*(8), 7067–7083. https://doi.org/10.1016/j.eswa.2012.01.039

Ben Taieb, S., Sorjamaa, A., & Bontempi, G. (2010). Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing*, *73*(10–12), 1950–1957. https://doi.org/10.1016/j.neucom.2009.11.030

Box, G. E. P., & Jenkins, G. M. (1970). *Time series analysis: Forecasting and control*. Holden-Day.

Brooks, C. (2008). *Introductory econometrics for finance* (2. ed.). Cambridge Univ. Press.

Carbonneau, R., Laframboise, K., & Vahidov, R. (2008). Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, *184*(3), 1140–1154. https://doi.org/10.1016/j.ejor.2006.12.004

Chakraborty, K., Mehrotra, K., Mohan, C. K., & Ranka, S. (1992). Forecasting the behavior of multivariate time series using neural networks. *Neural Networks*, *5*(6), 961–970. https://doi.org/10.1016/S0893-6080(05)80092-9

Chatfield, C. (1993). Neural networks: Forecasting breakthrough or passing fad? *International Journal of Forecasting*, *9*(1), 1–3. https://doi.org/10.1016/0169-2070(93)90043-M

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMNLP 2014 - 2014 Conference on Empirical*

Methods in Natural Language Processing, Proceedings of the Conference, 1724–1734. https://doi.org/10.3115/v1/d14-1179

Crone, S. F., Hibon, M., & Nikolopoulos, K. (2011). Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting*, *27*(3), 635–660. https://doi.org/10.1016/j.ijforecast.2011.04.001

Davydenko, A., & Fildes, R. (2013). Measuring Forecasting Accuracy: The Case Of Judgmental Adjustments To Sku-Level Demand Forecasts. *International Journal of Forecasting*, *29*(3), 510–522. https://doi.org/10.1016/j.ijforecast.2012.09.002

Foster, W. R., Collopy, F., & Ungar, L. H. (1992). Neural network forecasting of short, noisy time series. *Computers and Chemi- Cal Engineering*, *16*(4), 293–297.

Fry, C., & Brundage, M. (2020). The M4 forecasting competition – A practitioner ' s view. *International Journal of Forecasting*, *36*(1), 156–160. https://doi.org/10.1016/j.ijforecast.2019.02.013

Gardner, E. S. J. (2006). *Exponential smoothing : The state of the art — Part II*. *22*, 637–666. https://doi.org/10.1016/j.ijforecast.2006.03.005

Georgii, H.-O. (2009). *Stochastik. Einführung in die Wahrscheinlichkeitstheorie und Statistik* (4. Auflage). Walter de Gruyter. https://doi.org/10.1515/9783110215274.

Goodfellow, I., Yoshua, B., & Courville, A. (2016). *Deep Learning* (Vol. 1). MIT press.

Hibon, M., & Makridakis, S. (2000). The M3-Competition: results, conclusions and implications `. *International Journal of Forecasting*, *16*, 451–476.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huber, J., Müller, S., Fleischmann, M., & Stuckenschmidt, H. (2019). A data-driven

newsvendor problem: From data to decision. *European Journal of Operational Research*, *278*(3), 904–915. https://doi.org/10.1016/j.ejor.2019.04.043

Hyndman, Rob J. and Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, *26*(3).

Hyndman, R.J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd ed.). OTexts. OTexts.com/fpp2.

Hyndman, Rob J. (2016). *Q&A time*. Hyndsight. https://robjhyndman.com/hyndsight/qa-time/

Koenker, R., & Bassett and Jr, G. (1978). Regression Quantiles. *Econometrica*, *46*(1), 33–50. https://www.jstor.org/stable/1913643

Kourentzes, N. (2013). Intermittent demand forecasts with neural networks. *Intern. Journal of Production Economics*, *143*(1), 198–206. https://doi.org/10.1016/j.ijpe.2013.01.009

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *2*, 1097–1105.

Ma, S., Fildes, R., & Huang, T. (2016). Demand forecasting with high dimensional data: The case of SKU retail sales forecasting with intra- and inter-category promotional information. *European Journal of Operational Research*, *249*(1), 245–257. https://doi.org/10.1016/j.ejor.2015.08.029

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018a). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE*, *13*(3), 1–26. https://doi.org/10.1371/journal.pone.0194889

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018b). The M4 Competition:

Results, findings, conclusion and way forward. *International Journal of Forecasting*, *34*(4), 802–808. https://doi.org/10.1016/j.ijforecast.2018.06.001

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, *36*(1), 54–74. https://doi.org/10.1016/j.ijforecast.2019.04.014

Mariet, Z., & Kuznetsov, V. (2020). Foundations of sequence-to-sequence modeling for time series. *AISTATS 2019 - 22nd International Conference on Artificial Intelligence and Statistics*, *89*.

Mitchell, T. M. (1997). *Machine learning*. McGraw Hill.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001

Sharda, R., & Patil, R. B. (1992). Connectionist approach to time series prediction: an empirical test. *Journal of Intelligent Manufacturing*, *3*(5), 317–323. https://doi.org/10.1007/BF01577272

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. Van Den, Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., & Kavukcuoglu, K. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7585), 484–489. https://doi.org/10.1038/nature16961

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, *36*(1), 75–85. https://doi.org/10.1016/j.ijforecast.2019.03.017

Steinker, S., Hoberg, K., & Thonemann, U. W. (2017). The Value of Weather Information

for E-Commerce. *Production and Operations Management*, *26*(10), 1854–1874. https://doi.org/10.1111/poms.12721

Strijbosch, L. W. G., Syntetos, A. A., Boylan, J. E., & Janssen, E. (2011). On the interaction between forecasting and stock control: The case of non-stationary demand. *International Journal of Production Economics*, *133*(1), 470–480. https://doi.org/10.1016/j.ijpe.2009.10.032

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, *4*(January), 3104–3112.

Taylor, J. W. (2007). Forecasting daily supermarket sales using exponentially weighted quantile regression. *European Journal of Operational Research*, *178*(1), 154–167. https://doi.org/10.1016/j.ejor.2006.02.006

Thonemann, U. (2010). *Operations Management: Konzepte, Methoden und Anwendungen*. Pearson Deutschland GmbH.

Wen, R., Torkkola, K., Narayanaswamy, B., & Madeka, D. (2017). A Multi-Horizon Quantile Recurrent Forecaster. *@ 31st Conference on Neural Information Processing Systems (NIPS 2017)*, *Nips 2017*. http://arxiv.org/abs/1711.11053

Zhang, G., Eddy Patuwo, B., & Y. Hu, M. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*(1), 35–62. https://doi.org/10.1016/S0169-2070(97)00044-7

Zhu, L., & Laptev, N. (2017). Deep and Confident Prediction for Time Series at Uber. *IEEE International Conference on Data Mining Workshops, ICDMW*, *2017-Novem*, 103–110. https://doi.org/10.1109/ICDMW.2017.19