Production Plan Forecasting on Limited Dataset

Utilizing Hierarchical Model Architectures to Circumvent Irregularities in Limited Time Series Datasets.

Master's thesis in Complex Adaptive Systems

Pontus Kjellström
Niklas Lundblad

DEPARTMENT OF PHYSICS

Chalmers University of Technology
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Production Plan Forecasting on Limited Dataset

Utilizing Hierarchical Model Architectures to Circumvent Irregularities in Limited Time Series Datasets

PONTUS KJELLSTRÖM
NIKLAS LUNDBLAD

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of physics
Chalmers University of Technology
Gothenburg, Sweden 2025

Production Plan Forecasting on Limited Dataset
Utilizing Hierarchical Model Architectures to Circumvent Irregularities in Limited Time Series Datasets

PONTUS KJELLSTRÖM
NIKLAS LUNDBLAD

Supervisor: Di Xue, Data Team at Plejd
Examiner: Giovanni Volpe, Departement of Physics, University of Gothenburg

Master's Thesis 2025
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 10 00

Typeset in LaTeX
Gothenburg, Sweden 2025

Abstract

Forecasting production plans with limited data is a significant challenge, especially for smaller firms or new products with short production histories. This thesis aims to predict actual production outcomes based on historical data and to understand the relationships between and connect planned and executed production volumes. By exploring various forecasting approaches, including ARIMA and Long Short-Term Memory (LSTM) networks, the study focuses on methods designed to perform well with smaller datasets. The research employs a hierarchical model architecture that decomposes the forecasting task into three components: production forecasting, quarterly mapping, and unfolding to a monthly plan estimate. The models are evaluated against baseline strategies, including naive predictions and basic LSTM and ARIMA models. Results show that the proposed hierarchical models outperform baseline models, capturing the general behavior of the data more effectively. The deep learning-based model excels at capturing extremes in the time series, while the regression-based model provides stable and accurate forecasts. However, the models struggle with highly erratic production plan patterns, indicating the need for further refinement. This thesis contributes to more robust and scalable production planning solutions for data-constrained environments, offering valuable insights for both academic research and practical applications.

Keywords: Production Planning, Time series Forecasting, Hierarchical Model, Deep Learning, ARIMA, MLP, Time Series, Limited Data, hybrid model, LSTM

Acknowledgements

We would like to express our sincere gratitude to the company PLEJD for providing the real historical data that formed the foundation of this thesis. Their support and collaboration have been invaluable throughout the project.

We are especially thankful to our supervisor, Di Xue, for their continuous guidance, insightful feedback, and encouragement during every stage of this work. Their expertise and mentorship have been instrumental in shaping the direction and quality of this thesis.

Finally, we extend our appreciation to everyone who supported us throughout this journey, academically, professionally, and personally.

Pontus Kjellstöm, Niklas Lundblad, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis, listed in alphabetical order:

ADAM	Adaptive Movement Estimation
AR	AutoRegression
ARIMA	AutoRegressive Integrated Moving Average
ES	Exponential Smoothing
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi Layer Perceptron
MA	Moving Average
MSE	Mean Square Error
RF	Random Forest
RNN	Recurrent Neural Network
ReLu	Rectified Linear Unit
STL-Decomposition	Seasonal-Trend Decomposition using LOESS
VIMs	Variable Importance Measures
rMSE	root Mean Square Error
sMAPE	symmetric Mean Average Percentage Error

Contents

1 Introduction **14**

1.1 Ethical & Sustainability considerations 15

2 Literature review **15**

2.1 Time series forecasting 15

2.2 Extracting Information from Time Series Data 16

2.2.1 Time Series Clustering 16

2.3 Statistical or ML-methods 17

2.4 Hybrid models 18

2.5 Hierarchical Production Planning 18

2.6 Summary of Findings 19

3 Available Dataset **20**

3.1 Production Plan 21

3.2 Production 21

3.3 Installation 22

3.4 Stock 23

4 Theory **23**

4.1 AutoRegressive Integrated Moving Average (ARIMA) 23

4.1.1 Autoregression (AR) models 23

4.1.2 Integrated (I) component 23

4.1.3 Moving average (MA) models 24

4.1.4 ARIMA-model 24

4.2 ARIMAX . . . 25
4.2.1 Random Forest . . . 26
4.3 Long Short-Term Memory (LSTM) . . . 26
4.4 Multi-Layer Perceptron . . . 28
4.5 Clustering method . . . 29
4.5.1 Product Category clustering . . . 29
4.5.2 K-means clustering . . . 30
4.5.3 Quarterly Peak Clustering . . . 31

5 Method **32**
5.1 General Approach . . . 33
5.2 Model Structure . . . 34
5.2.1 Implementation . . . 34
5.3 Model Evaluation . . . 35

6 Deep Learning-based Model **36**
6.1 Step 1: Production Forecasting . . . 36
6.1.1 Training Procedure . . . 37
6.1.2 Output . . . 37
6.2 Step 2: Quarterly Mapping . . . 38
6.2.1 Training Procedure . . . 39
6.2.2 Output . . . 39
6.3 Step 3. Unfolding . . . 40

7 Regression-based model **41**

7.1 Step 1. Production Forecasting 41
7.1.1 Feature Selection 41
7.2 Step 2. Quarterly Mapping 43
7.2.1 Training Procedure 43
7.2.2 Output 44
7.3 Step 3. Unfolding 44
7.3.1 Training Procedure 45

8 Results **46**

9 Discussion **49**

10 Conclusion **50**

11 Future Work **51**

12 Appendix **56**

List of Figures

1 Time series behavior of the three central datasets, illustrated for the article *DIM-01.* 21

2 STL decomposition of installation data illustrating the contrast between well-structured and irregular time series. 22

3 Illustration of the internal structure of an LSTM cell, including its three gates. [28] 27

4 Illustration of the different cluster categories. 32

5 Overview of the proposed hierarchical model architecture. 35

6 Production forecast using Multi-Layer Perceptron architecture (12x32, 32x32, 32x32, 32x12). 37

7 Example output of the deep learning-based production forecast compared to ground truth. 38

8 Mapping of production to quarterly planning data using a fully connected neural network (4x32, 32x32, 32x32, 32x1). 38

9 Examples of mapping from forecast production to quarterly production plan for three separate articles. 39

10 Comparison of seasonal encoding and production plan unfolding processes. 40

11 Example of predicting productions twelve months ahead 43

12 Mapping of Production to Quarterly plan (5x3,3x3,3x3,3x1) 44

13 Example of the output from step 2 of the regression based model . . . 44

14 Mapping of production to quarterly production plan (5x4,4x3,3x2,2x3,3x4,4x12) 45

15 Final Predictions Compared to the Ground Truth for the article WRT-01-SW 46

16 Performance metrics for the proposed models compared to baseline for WRT-01-SW. Lower is better. 47

17 Final Predictions Compared to the Ground Truth of the article REL-02. 47

19 Average Performance Across all 26 Articles in Test Dataset. 48

18 Performance metrics for the proposed models compared to baseline for REL-02. Lower is better. 48

List of Tables

1 Summary of Forecasting Models and Surveys (Part 2) 20

2 Cluster Categories and Associated Articles 30

3 Cluster from K-means Clustering 31

4 Cluster from Quarterly Peak Clustering method 32

5 Predictions for all articles in test set 56

6 Predictions for all articles in test set 57

7 Predictions for all articles in test set 58

1 Introduction

Effective production planning is a cornerstone of success in the manufacturing industry. An efficient approach to this task can significantly reduce inventory costs, mitigate supply chain disruptions, and enhance overall operational efficiency, ultimately contributing to increased profitability and a healthier work environment. However, as companies scale, the traditional manual methods of developing production plans become increasingly complex and unsustainable. Automating this process is therefore essential for enabling long-term growth and agility.

This thesis focuses on the challenge of forecasting production plans in scenarios where historical data is limited, a common issue for smaller firms or newly introduced products with short production histories. The goal is twofold: to predict the actual production outcomes based on historical data, and to understand the underlying relationships between planned and executed production volumes and connect them. These relationships are often obscured by external factors such as market dynamics, supply chain interruptions, or internal strategic shifts, making the problem particularly complex.

At its core, this is a time series forecasting problem—where historical production data is used to predict future trends. Traditional statistical models like AutoRegressive Integrated Moving Average (ARIMA) and Exponential Smoothing (ES) have long been employed for such tasks. More recently, machine learning (ML) techniques have gained traction, benefiting from advances in computational power and offering new ways to model complex patterns in data. However, research shows that neither approach is universally superior, and that their effectiveness varies depending on the nature and structure of the dataset.

In response, hybrid models, that combine statistical and machine learning methods, have emerged as a promising direction. These models can leverage the strengths of both paradigms and have shown impressive results on specific forecasting tasks.

Given the constraint of limited data, this thesis explores a range of forecasting approaches, with a particular emphasis on methods designed to perform well with smaller datasets. Our aim is to identify and develop models that can accurately capture the trends in executed production while uncovering the nuanced interplay between planning and actual outcomes. By doing so, we hope to contribute to more robust and scalable production planning solutions for data-constrained environments.

1.1 Ethical & Sustainability considerations

As with all types of machine learning, there is a heavy demand for annotated data in the time-series forecasting field. This process is riddled with ethical concerns [1]. Time-series data is, however, different in this aspect. The data consists of historical data points and is thereby inherently annotated: the correct label is the data itself; thus, the time-series data may, in large, circumvent these ethical issues. This is especially true for the data used in this project.

Training deep networks for any type of application is associated with the expenditure of computational resources, contributing to a high energy consumption [2]. As the networks studied in this report are very small this impact is minimal as of right now, but with the potential of scaling the project this could very well become concerning.

The forecasting models developed in this thesis aim to improve production efficiency, the implications of this are generally positive. An accurate prediction of the production pace may lead to less energy consumption. Optimizing the prediction entails producing as few articles as possible, while still meeting the demands. Thereby, fewer articles need warehouse storage, cutting down on the necessary space required. In addition, optimal prediction minimizes the amount of waste produced [3].

A critical aspect when automating previously manual tasks lies in the responsibility and accountability for the result. A clear outline regarding who holds responsibility for the consequences, in all respects, is crucial to guarantee a sustainable and ethical use of the models.

2 Literature review

2.1 Time series forecasting

Time series forecasting refers to the process of predicting future values of a variable, based on collected historical data points. It is often seen as a fundamental task in various industrial areas, such as finance, economics, and meteorology[4, 5].

The data in a time series could take many forms, and the time between observations could vary between different time series data. It could often exhibit complex patterns such as trend, seasonality, and other sudden shifts. Carefully choosing an appropriate model allows us to, in an optimal way, capture as many of these patterns

as possible [4, 5].

Historically, statistical models such as AutoRegressive Integrated Moving Average (ARIMA) models, and Exponential Smoothing (ES) models have dominated time series forecasting. These make use of data properties such as mean, variance and autocorrelation of the data at hand to sample and predict future values. More recently, machine learning methods such as Long Short Term Memory (LSTM) networks and other deep learning networks have seen an increase in popularity. This is mainly due to their ability to capture complex patterns and structures over time. [6]

2.2 Extracting Information from Time Series Data

Time series data often consist of three main components:

- Seasonal component: Patterns that recur at regular intervals.
- Trend component: Representing the direction in which the time series move over time, when excluding seasonality and other irregularities.
- Residual component: After accounting for the seasonal and the trend component of the time series, the residual component is what is left.

There are several ways to decompose a time series into these components. Seasonal trend decomposition using LOESS (STL-decomposition) consists of two recursive procedures: an inner loop nested inside an outer loop. Each pass of the inner loop consists of a seasonal smoothing, that computes the new values for the seasonal component, followed by a trend smoothing, which computes the updated value for the trend component. After removing both the seasonal and the trend component at a time step in the data series, what is left is the residual component. The outer loop then computes the robustness weights, which describes how extreme the residual component is at a specific observed time point. An outlier in the data will result in a very small robustness weight. This process is then repeated for all datapoints in the time series. [7]

2.2.1 Time Series Clustering

Clustering serves as a pivotal tool in time series forecasting. By grouping similar time series, clustering enables extraction of shared patterns and characteristics, which can be leveraged to improve forecasting accuracy [8].

In practical applications, clustering can also support model scalability and generalization. For instance, in large-scale forecasting tasks involving hundreds or thousands of time series, clustering allows practitioners to reduce computational complexity by training a smaller number of representative models, one per cluster, rather than a unique model for each series. This approach has been shown to improve forecasting performance, especially when the clusters are well-aligned with the intrinsic characteristics of the data. Furthermore, clustering can aid in anomaly detection, seasonality profiling, and the identification of regime shifts, all of which are critical for adaptive forecasting systems. As such, clustering serves not only as a preprocessing step but also as a strategic component in the broader forecasting pipeline[9].

2.3 Statistical or ML-methods

Different types of ARIMA models have been widely applied in the field of time series forecasting for over half a century due to their mathematical simplicity and flexibility in application. These, together with other statistical models, have dominated time series forecasting. Recent advances in the development and efficient deployment of machine learning models and techniques have shown promise and are rapidly changing the time series forecasting landscape [10].

[11] compares eight different forecasting methods on 35 smaller time series with the objective of seeing which performs the best given a limited data set. One of their findings is that the statistical models in the study (SMA, ARIMA) perform significantly better than the models that are based on neural networks. They explain the poor performance of the neural network models by stating that the limited amount of data prohibits the neural network models from identifying robust patterns, ultimately causing the advanced network to fail to learn anything useful for predicting future values of the variables in their dataset.

Other studies show examples of machine learning models repeatedly outperforming statistical models. [12] shows comparisons between several statistical models and machine learning models for demand prediction within various different retail segments. The findings show that the machine learning methods, for almost all categories, outperform the statistical models quite comfortably. The main strength of the ML methods, in comparison to statistical models, is that instead of describing the data generating process, these allow for relationships and patterns in the data to be identified and estimated automatically. The fact that the ML models make no assumptions about the data generation, their performance relies heavily on an adequate amount of data. [13]

[14] provides an extensive comparison of multiple statistical and machine learning models across more than 90 datasets, evaluated using several performance metrics. Consistent with earlier studies mentioned above, their findings clearly indicate that no single model consistently outperforms the others. Instead, the dataset characteristics play a significant role in influencing the performance of different models.

2.4 Hybrid models

In parallel with the advancements in ML-models for time forecasting, there has also been an increase in research for hybrid models, which combine both machine learning models and statistical models in a directed way tailored to specific datasets and forecasting tasks. These have, for several different datasets, proven to extract the best properties of each model type to actually yield better results than either of the two model types individually [15].

Further evidence of this can be seen in the M4 forecasting competition [16], in which a high number of different models are developed to best try to solve the forecasting problems in a rather large dataset. The competition is highly regarded in the forecasting community, and a good driver for targeted further research. Both the first and the second placed contributions in this competition were hybrid models. In [16] it is further stated that the most accurate models in many future applications will require careful consideration of how to combine the available tools in order to build custom solutions to match the unique characteristics of the time series that is to be forecast.

2.5 Hierarchical Production Planning

In recent years, hierarchical forecasting has emerged as a powerful approach for tackling complex production planning problems, especially in environments where data is fragmented across multiple levels of the supply chain. Hierarchical forecasting refers to the process of generating forecasts at different levels of aggregation, such as product, category, or regional levels, and reconciling them to ensure consistency across the hierarchy[17].

[18] proposes a data-driven methodology for hierarchical production planning using an LSTM-Q network. Their model integrates long-term and short-term planning by combining deep learning with optimization techniques, demonstrating improved accuracy in demand forecasting across multiple planning levels.

[19] introduced a three-stage hierarchical model tailored for the production of

perishable goods. Their approach begins with demand pattern identification using clustering techniques to group similar time series. This is followed by the application of a Nonlinear AutoRegressive model with eXogenous inputs (NARX) for forecasting within each cluster. Finally, the model incorporates a reconciliation step to ensure consistency across hierarchical levels. This modular structure allows the model to capture both global and local demand dynamics, improving interpretability and robustness in volatile production environments.

These studies reinforce the value of hierarchical modeling in production planning, particularly when dealing with noisy, irregular, or limited datasets. By structuring the forecasting process into modular components, hierarchical models can better capture the dependencies and constraints inherent in real-world manufacturing systems.

2.6 Summary of Findings

Table 1 summarizes the important findings of the literature review. The reviewed literature highlights the strengths and limitations of various forecasting models in data-constrained environments. Simple statistical models like ARIMA often outperform deep learning methods on short or irregular time series, while machine learning approaches tend to excel on more complex datasets. No single model consistently dominates, emphasizing the importance of tailoring model choice to dataset characteristics. Hybrid models, which combine statistical rigor with machine learning flexibility, consistently achieve strong performance, particularly in competitions like M4. Additionally, recent work demonstrates that simpler architectures such as MLPs can outperform more complex models like Transformers in real-world forecasting tasks. Hierarchical and hybrid models, especially those incorporating clustering and seasonality encoding, have shown promise in production planning contexts.

Ref.	Authors / Title	Type	Models	Key Insights
[6]	Kong et al. (2025), *Deep Learning for Time Series Forecasting*	Survey	RNN, LSTM, GRU, CNN, Transformer, Hybrid	Taxonomy of DL models; discusses feature extraction, architectures, and challenges like overfitting.
[10]	Kim et al. (2025), *Survey of DL for Time Series*	Survey	MLP, CNN, RNN, GNN, Transformer, Mamba, Diffusion	Finds linear models can outperform Transformers; trends include foundation models, hybridization.
[11]	Cruz-Nájera et al. (2022), *Short Time Series Forecasting*	Comparative	SMA, HW, ARIMA, RF, GB, Bagging	Simple models often outperform ML on short series. Emphasizes seasonal adjustments.
[12]	Kandananond (2012), *Forecasting Autocorrelated Series*	Comparative	ARIMA, ANN, SVM	SVM outperforms others on autocorrelated demand; data characteristics affect model choice.
[14]	Parmezan et al. (2019), *Statistical vs. ML Models*	Comparative	ARIMA, ETS, RF, SVM, XGBoost, LSTM	ML often wins on complex data; no model dominates overall.
[15]	Kontopoulou et al. (2023), *ARIMA vs. ML Forecasting*	Review	ARIMA, LSTM, RF, Hybrid	Hybrids outperform solo models; combining ARIMA with ML boosts robustness.
[16]	Makridakis et al. (2019), *M4 Competition Summary*	Competition	Hybrid, Stat + ML	Top entries were hybrid models; blends statistical rigor with ML flexibility.
[18]	Luo et al. (2025), *LSTM-Q for Planning*	Specific Model	LSTM-Q (LSTM + Q-learn)	Integrates short/long-term forecasts with optimization; improves volatile demand planning.
[19]	Chen et al. (2019), *Forecasting for Perishables*	Specific Model	NARX, Clustering, Hierarchical	3-stage model improves perishable forecasts; cuts profit loss.
[20]	Chen et al. (2023), *TSMixer: All-MLP*	Specific Model	TSMixer (MLP)	MLPs with time/feature mixing outperform Transformers; efficient and simple.

Table 1: Summary of Forecasting Models and Surveys (Part 2)

3 Available Dataset

This project makes use of four primary datasets, each representing different aspects of the production pipeline. For each dataset, we consider a pool of 35 unique articles. The general characteristics and challenges associated with these datasets are described in the following subsections.

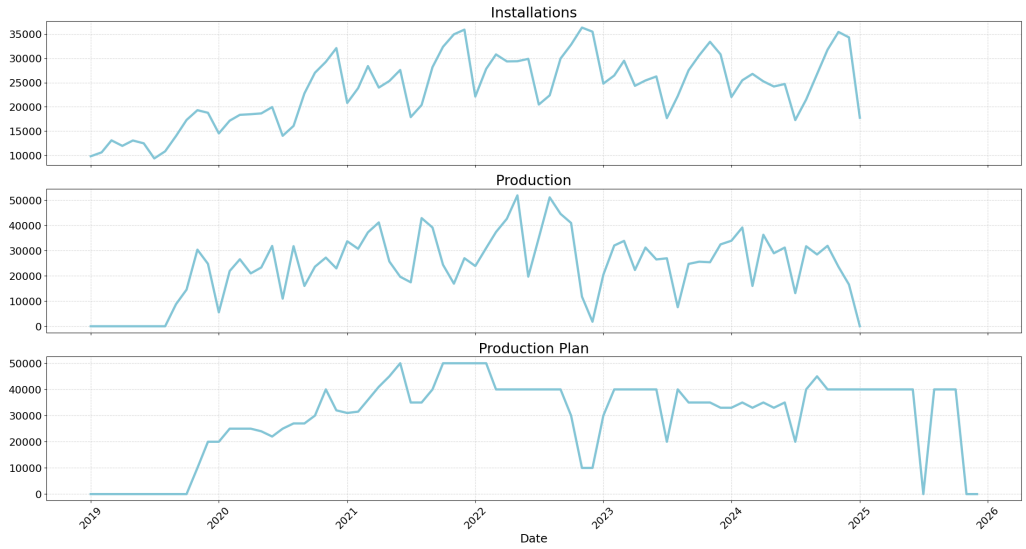


Figure 1: Time series behavior of the three central datasets, illustrated for the article *DIM-01*.

3.1 Production Plan

The production plan dataset, referred to as *Plan* throughout this work, represents the ground truth for our forecasting efforts. It contains monthly targets for the number of units to be produced per article and per country. Notably, these plans have been manually created by a domain expert, incorporating tacit knowledge and market intuition rather than purely data-driven processes.

As illustrated in Figure 1 (bottom plot), the *Plan* data displays irregular and often abrupt fluctuations, including sharp peaks, drops to zero, and intermittent stability. Such patterns suggest a strong influence from external factors, including regional market dynamics, promotional campaigns, supply chain issues, and strategic business decisions. The absence of a consistent temporal structure presents challenges for conventional sequence-to-sequence forecasting models, which typically rely on recurrent patterns to learn effective mappings.

3.2 Production

The production dataset records the actual monthly production quantities for each article. The middle plot in Figure 1 shows the historical production values for an article called DIM-01. Although the general timing of peaks and troughs tends to align with the plan data, there are significant discrepancies in magnitude and volatility. This suggests that while production may be loosely guided by the plan, it is also subject to dynamic factors such as manufacturing constraints, real-time demand fluctuations, and inventory considerations.

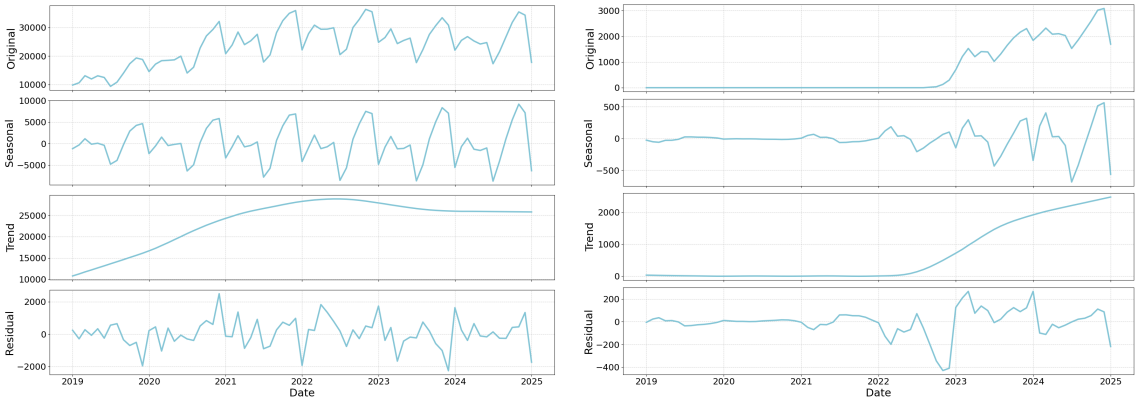
3.3 Installation

Installation data tracks the number of units installed per article and serves as an indirect indicator of product deployment. The top plot in Figure 1 illustrates the installation behavior for DIM-01, which generally follows the production trend but with less direct correlation to the planning data.

To further explore the structure of this data, STL-decomposition was performed, as shown in Figure 2. Two contrasting examples are provided:

- Figure 2a (left) presents a well-behaved time series for DIM-01, exhibiting strong and regular 12-month seasonality. This predictable structure aligns well with the corresponding plan data and presents an opportunity for improved forecasting.
- Figure 2b (right) displays the STL decomposition for LED-75, a product with a short production history and little discernible seasonal structure. This type of behavior illustrates two core challenges in our dataset: (1) the absence of coherent temporal patterns, and (2) limited data availability due to short production runs.

Overall, the dataset averages just 2.46 full 12-month sequences per article, which, combined with high variability across the 35 articles, introduces substantial complexity in modeling and forecasting.



(a) Decomposition of Historical Installations of DIM-01

(b) Decomposition of Historical Installations of LED-75

Figure 2: STL decomposition of installation data illustrating the contrast between well-structured and irregular time series.

3.4 Stock

The final dataset pertains to inventory, or stock levels. This data includes the number of produced articles that remain unsold or uninstalled at any given time. Stock levels are reported monthly and aggregated across the company's own warehouses and those of third-party retailers. This dataset offers valuable insight into supply-demand imbalances and is a key input for inferring future production needs based on existing circulation.

4 Theory

4.1 AutoRegressive Integrated Moving Average (ARIMA)

A common statistical method used for univariate time series forecasting is the AutoRegressive Integrated Moving Average (ARIMA). An ARIMA model is labeled as ARIMA(p,d,q), where p is the number of autoregressive terms, d is the number of differences, and q is the number of moving averages. The ARIMA model comprises three parts, namely an AutoRegressive (AR) part, an Integrated part (I) and a Moving Average (MA) part.

4.1.1 Autoregression (AR) models

Autoregressive models assume that Y_t is a linear combination of the earlier observations. It is given by

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t$$

where:

- $\phi_1, \ldots, \phi_p$ are the autoregressive coefficients,
- ϵ_t is errors at timestep t

4.1.2 Integrated (I) component

A common concept in time series data is stationarity. A stationary time series exhibits constant mean and variance over time. If the variance or expected values

are dependent on time, the series is said to be non-stationary. Both AR and MA models assume stationarity. The I-component of the ARIMA model is used to make the time series stationary. The first difference is:

$$Y'_t = Y_t - Y_{t-1}$$

In general, differencing of order d is defined recursively as:

$$\nabla^d Y_t = (1 - B)^d Y_t$$

where B is the backshift operator ($BY_t = Y_{t-1}$). This operator is used to shift back the time series by d time steps. [21]

4.1.3 Moving average (MA) models

The MA(q) model expresses the current value of the series as a linear function of past forecast errors:

$$Y_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

where:

- $\theta_1, \ldots, \theta_q$ are the moving average coefficients,
- ϵ_t is the errors at timestep t.

4.1.4 ARIMA-model

The composition of the above yields the following expression for the ARIMA model.

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t \tag{1}$$

where:

- Y_t is the dependent variable (time series observation at time t).
- c is the constant term.

- ϕ_i represents the autoregressive (AR) coefficients, where p is the number of lagged values.
- θ_j denotes the moving average (MA) coefficients, where q is the number of past error terms included in the model.
- ϵ_t is the error term (white noise).
- β_k represents the regression coefficients for exogenous variables.
- $x_{k,t}$ is the external regressor affecting y_t.
- m is the number of external regressors.

The (p,d,q) parameters are said to be the order of the ARIMA model. There are several metrics that could aid in choosing the optimal values for these parameters. Some of the most popular techniques used for this include visually studying the Partial Autocorrelation function (PACF) and the Autocorrelation function (ACF) plots. The use of these in parameter estimation is, however, subjective, and different analysts may arrive at different conclusions. A more analytical method for parameter estimation is maximum likelihood estimation (MLE).[22] [23]

4.2 ARIMAX

While ARIMA models are effective in many environments, they assume that the data is self-contained and solely affected by its own historical values. For more complex datasets, with dependability between different variables, an original ARIMA model may struggle to accurately forecast future values. The ARIMAX is an extension of the original ARIMA model, which allows for the integration of external variables. This could provide a more suitable framework for predicting future values in comparison to the original ARIMA model. [23]

Mathematically, the ARIMAX model is formulated as follows:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{k=1}^{m} \beta_k x_{k,t} + \epsilon_t \tag{2}$$

where:

- Y_t is the dependent variable at time t.
- ϕ_i represents the coefficients of the autoregressive (AR) terms.

- θ_j represents the coefficients of the moving average (MA) terms.
- β_k represents the influence of external regressors $x_{k,t}$.
- ϵ_t is the error term.

4.2.1 Random Forest

Random Forest (RF) is a widely used machine learning technique that constructs an ensemble of decision trees to improve predictive performance. By aggregating the outputs of multiple trees, RF reduces the variance associated with individual decision trees, resulting in more stable and robust predictions [24]. In the context of regression tasks, the model produces final predictions by averaging the outputs of all individual trees.

A key advantage of RF is its capacity to assess the relative importance of input variables through Variable Importance Measures (VIMs). These measures facilitate feature selection and provide insight into the underlying data structure. The two most commonly used VIMs are impurity importance and permutation importance [25]. Impurity importance, often referred to as Gini importance, tends to be biased toward categorical and linearly correlated features. Permutation importance, although more computationally intensive, offers a less biased alternative and is therefore often preferred in practice [26].

In this thesis, Random Forest has been employed to identify the most relevant features for the ARIMAX model. Feature selection was conducted separately for each article, allowing the model to tailor its inputs based on the unique characteristics of the individual time series.

4.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are among the most widely adopted deep learning models for time series prediction and forecasting. Originally introduced in 1997 [27], LSTMs were designed to address the issue of vanishing gradients, which impeded learning in earlier recurrent neural network (RNN) architectures. By truncating the gradient and maintaining a constant error flow through internal memory cells, LSTMs provided a powerful and efficient solution for modeling long-term dependencies in sequential data that were previously unmanageable by standard RNNs.

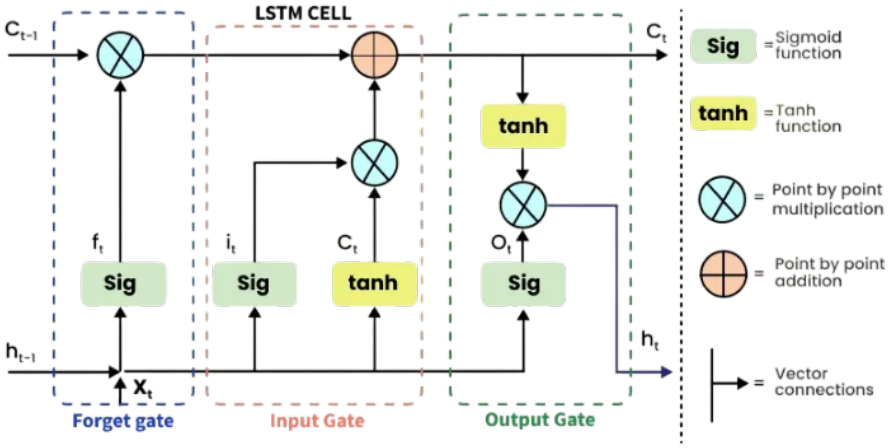


Figure 3: Illustration of the internal structure of an LSTM cell, including its three gates. [28]

LSTM networks build upon the structure of RNNs, which are composed of repeating modules that generate a hidden state for each time step t. In standard RNNs, each module typically consists of a single layer using a hyperbolic tangent (tanh) activation function. The primary difference between an RNN and an LSTM lies in the internal structure of this module, referred to as a *cell* in LSTM terminology.

The LSTM cell is more complex, comprising four neural network layers structured into three distinct components, or *gates:* the forget gate, the input gate, and the output gate, as illustrated in Figure 3.

Forget Gate: The forget gate determines which information should be retained or discarded from the cell state. It uses a sigmoid activation function to produce values between 0 and 1, where 0 indicates complete forgetting and 1 indicates complete retention. It takes as input the previous hidden state h_{t-1} and the current input x_t:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

Here, σ denotes the sigmoid function, W_f is the weight matrix, and b_f is the bias term for the forget gate.

Input Gate: The input gate regulates what new information will be stored in the cell state. It has two components:
First, a sigmoid layer that decides which values will be updated:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4}$$

Second, a tanh layer that creates candidate values $\hat{C}_t$ for the cell state:

$$\hat{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{5}$$

These components are combined to update the cell state:

$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t \tag{6}$$

where $\odot$ denotes element-wise multiplication.

Output Gate: The output gate determines the output of the LSTM cell, which also serves as the hidden state for the next time step. It is computed as:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t \odot \tanh(C_t) \tag{8}$$

This gated architecture allows the LSTM to maintain long-term dependencies by preserving relevant information and discarding irrelevant data across many time steps.

The LSTM will be used as a comparative performance baseline for the final results of this thesis.

4.4 Multi-Layer Perceptron

One of the most popular models for time series forecasting in recent years has been the transformer model. [29] discusses the relevancy of this and offers an alternative. In their paper, [29] provides an extremely simple linear neural network, that, with the right composition, can outperform these much more advanced transformer models on time-series forecasting tasks. This proves the validity of these more novel architectures for time-series forecasting tasks.

MLPs are some of the most basic architectures among modern neural networks. They generally consist of three different types of layers: Input, hidden, and output. A key to the historical success of MLPs is the usage of non-linear activation functions. This allows the networks to perform well with datasets that are not linearly separable.

The application of MLPs for time series forecasting tasks is uncommon, but not unheard of. Building upon the findings of [29],[20] proposes a time series forecasting model built entirely on MLPs, specifically utilizing the more modern approach of MLP-mixers. This model outperforms many state-of-the-art methods on the M5 dataset [30], a dataset created for the explicit purpose of comparing time series forecasting models, highlighting the relevancy of not closing the door on simpler model architectures for forecasting applications.

4.5 Clustering method

The most common usage for time series clustering, in regards to forecasting, is in dividing the data into smaller groups with similar characteristics to train models separately for each cluster. In this project, this approach is inhibited by the small dataset, leading to immediate overfitting when training on smaller groups of data.

Instead, the clustering component of the model serves to identify similarities among different articles, thereby enabling the encoding of general temporal patterns as input features. Capturing the overall shape and dynamics of the time series is essential for several stages of the model pipeline, and clustering is one of the primary strategies employed for this purpose. An additional method for incorporating this type of information is described in a later section.

Three distinct clustering approaches were explored. The first is an intuitive method based on the company's existing categorizations of articles. The second is a data-driven approach using K-Means clustering, implemented through the `Tslearn` library [31], which utilizes temporal similarity measures. The third approach is a custom hybrid method that combines elements of both the intuitive and data-driven techniques to leverage domain knowledge while preserving the objectivity of algorithmic clustering.

4.5.1 Product Category clustering

The product category clustering divides the articles into six separate clusters based on the function of that specific product. This leads to a somewhat unbalanced clustering, with two categories containing only one product each. However, the clusters created here worked well when implemented in the model. Table 2 below shows the different clusters, what category they correspond to, and what articles are in that specific category.

Cluster Category	Articles
1. Pucks & Drivers	DIM-01, DIM-01-2P, DIM-02, CTR-01, VRI-01, VRI-02, VRI-03, REL-01-2P, REL-02, LED-75, JAL-01, DAL-01, EXT-01, LED-10
2. Luminaires	DWN-01, DWN-01-B, DWN-01-W, DWN-02, CCL-01, CCL-01-B, CCL-01-W, LST-01, OUT-01, OUT-02, OUT-01-G, OUT-01-B, OUT-01-W, DWN-02-W, DWN-02-B, DWM-01-W, DWM-01-B
3. Thermostat	TRM-01
4. Remote Control	GWY-01
5. Wireless Control	WMS-01, WIN-01, WPH-01, WRT-01, WPH-01-SW, WPH-01-EB, WPH-01-EW, WPH-01-SA, WRT-01-RB, WRT-01-RW, WRT-01-EW, WRT-01-SW, WRT-01-EB, WRT-01-SA
6. Accessories	SPR-01, RTR-01, BAT-01, MNT-01, MNT-02, SPR-01-BL

Table 2: Cluster Categories and Associated Articles

4.5.2 K-means clustering

K-Means clustering is an unsupervised algorithm that groups data into K clusters based on similarity. For time series, this means grouping sequences with similar patterns. The algorithm starts by initializing K centroids. Each time series is then assigned to the nearest centroid using a distance metric such as Euclidean distance or Dynamic Time Warping (DTW). The centroids are updated as the mean of the assigned series, and the process repeats until the assignments stabilize. In this project, we used the `TimeSeriesKMeans` implementation from the `tslearn` library [31], which supports time series-specific distances and handles sequences of equal length. Table 3 details the clusters achieved using K-Means.

Cluster	Articles
1	DWN-01-B, DWN-02-B, DWN-02-W, JAL-01, SPR-01-BL, WIN-01, WRT-01-RB, WRT-01-RW
2	BAT-01, CTR-01, DIM-01, DIM-02, GWY-01, LED-10, REL-02, WRT-01-SA
3	REL-01-2P, WPH-01-EW, WPH-01-SA, WPH-01-SW
4	DAL-01, DWN-01-W, WMS-01
5	WPH-01, WPH-01-EB, WRT-01-EB, WRT-01-EW, WRT-01-SW
6	CCL-01-B, CCL-01-W, EXT-01, LED-75, OUT-01-B, OUT-01-G, OUT-01-W

Table 3: Cluster from K-means Clustering

4.5.3 Quarterly Peak Clustering

This clustering method aims to identify the seasonal production peak for each product. Products are grouped based on the quarter in which their production typically reaches its highest level—Q1, Q2, Q3, or Q4. Those that do not exhibit a clear quarterly peak are assigned to a separate category, designated as cluster 0.

$$x_i = \arg\max_q y_i q \tag{9}$$

where

$$q \in 1, 2, 3, 4 \tag{10}$$

corresponds to the four quarters in a year, and

$$i \in 1, ..., N \tag{11}$$

corresponds to the the years that a certain product has been in production.

$$X = \begin{cases} \text{round}\left(\frac{\Sigma_{i=1}^{N} x_i}{N}\right), & |x| < 0.8N \\ 0, & |x| \geq 0.8N \end{cases} \tag{12}$$

where the rounding is done to the closest integer.

Figure 4 shows an intuitive illustration of the clustering process, where the line symbolizes the general movement of the time series corresponding to that specific cluster.

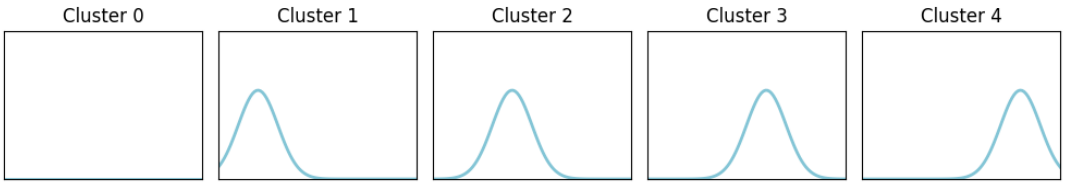


Figure 4: Illustration of the different cluster categories.

The quarterly peak clusters can are shown in table 4.

Cluster	**Articles**
0	EXT-01, OUT-01-B, WMS-01, DWN-01-W, DWN-01-B
1	OUT-01-G, OUT-01-W, WPH-01, SPR-01-BL, DWN-02-B, WRT-01-SW, WRT-01-EW, DWN-02-W
2	DIM-02, LED-75, WPH-01-SW, DAL-01, LED-10, WRT-01-SA, WPH-01-EB, CTR-01, WRT-01-EB, REL-01-2P
3	DIM-01, BAT-01, GWY-01, REL-02, WPH-01-SA
4	No articles in cluster 4

Table 4: Cluster from Quarterly Peak Clustering method

5 Method

Based on the insights gained from the exploratory data analysis, this thesis investigates a non-standard approach to the time series forecasting problem. The plan data, characterized by irregular patterns, manual construction, and a lack of consistent seasonality, presents challenges for direct modeling. These characteristics motivated the exploration of alternative strategies.

Instead of using the production plan directly as an input, we propose an indirect forecasting method. The approach is structured around a hierarchical model that incorporates auxiliary datasets such as production, installation, and stock levels to predict future production plans. By doing so, we aim to mitigate the limitations associated with the plan data and to exploit more structured and informative signals

present in the surrounding data.

The proposed method is evaluated against more conventional forecasting techniques. Specifically, we implement and compare a deep learning model based on Long Short-Term Memory (LSTM) networks and a regression-based model using AutoRegressive Integrated Moving Average (ARIMA).

5.1 General Approach

The model development process began with a thorough phase of data exploration and familiarization. This initial phase was essential for identifying patterns, inconsistencies, and potential predictive relationships across the available datasets, as discussed in Section 3. Given the complexity of the data and the absence of an established modeling path, we adopted an iterative and experimental strategy.

During early experimentation, we tested a range of predictive tasks and configurations in order to assess the utility of different features and modeling approaches. The outcomes of these tests guided key design choices and helped to shape the final model structure. Emphasis was placed on identifying the most effective strategies for capturing the underlying dynamics in the data.

To ensure a comprehensive evaluation, two modeling pipelines were developed in parallel:

- A regression-based pipeline based on ARIMAX models, designed to capture temporal dependencies and incorporate relevant external variables. The subsequent layers comprise deep, but narrow Multi-Layer Perceptron (MLP) architectures.
- A deep learning pipeline utilizing an MLP architecture, intended to model more complex and potentially nonlinear relationships between inputs and future production plans. The forecasting step is followed by less deep, wider MLPs.

Each pipeline follows the same overall structure, which consists of three stages: production forecasting, quarterly plan mapping, and unfolding to a monthly plan estimate. The implementation details of these components are presented in the subsequent sections.

5.2 Model Structure

To address the challenges posed by the production plan, which lacks regularity and is not suitable for conventional time series forecasting, we propose a hierarchical model architecture that decomposes the task into three sequential components. An overview of this structure is shown in Figure 5.

The first component forecasts the production data for the 12 months corresponding to the target plan. Production data is used because it is closely related to the planning data; although the two series do not align perfectly, production volumes are inherently influenced by the planning process. The goal of this stage is to estimate the general trend or shape of the monthly production distribution rather than to predict exact values. Since the total annual plan is fixed, a correctly shaped distribution can subsequently be scaled to match the annual total.

The second component carries out what we call quarterly mapping. In this step, the forecasted monthly production values are aggregated into four pools of quarterly data, which are then mapped to quarterly plan estimates. This mapping incorporates additional contextual features, which differ between the regression-based and deep learning-based pipelines and are described in detail in Sections 6 and 7.

The third component, referred to as unfolding, converts the quarterly plan estimates into a full 12-month forecast. This step applies a softmax activation function to produce a relative distribution over the months and then multiplies that distribution by the specified *Total Goal Plan*, which is the predetermined annual production target for each article.

Figure 5 illustrates two model pipelines built on this common framework. The first pipeline is regression-based and uses an ARIMAX model for the production forecasting stage [23]. The second pipeline is deep learning-based, employing a Multi-Layer Perceptron (MLP) to forecast production. Although both pipelines share the same three-stage design, their implementation details and feature sets differ; these are elaborated in Sections 6 (deep learning model) and 7 (regression-based model).

5.2.1 Implementation

All models and code developed for this project were implemented in `Python`. The relevant libraries and packages are referenced throughout this report in their respective

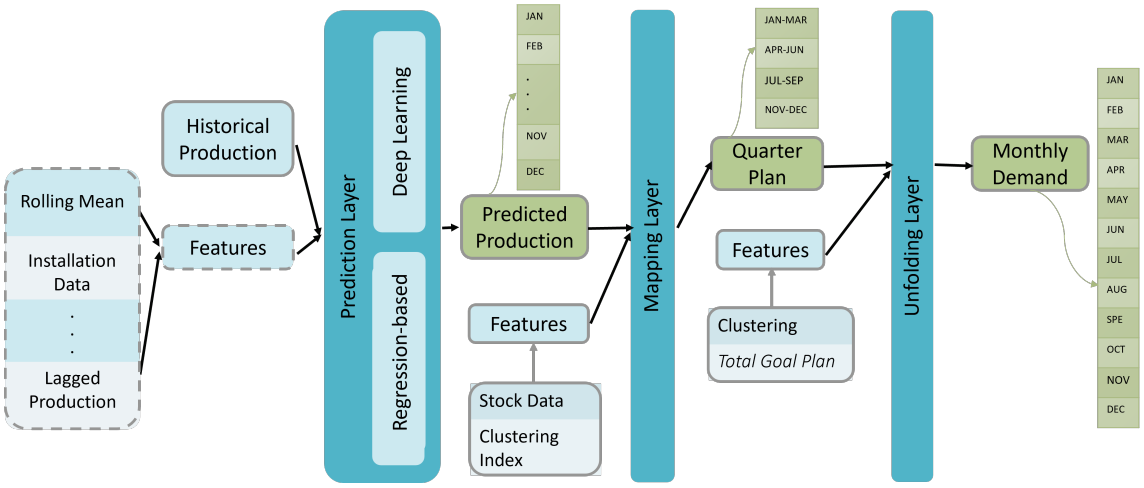


Figure 5: Overview of the proposed hierarchical model architecture.

contexts. For the development of all deep learning models, the `PyTorch` framework was employed due to its flexibility and strong support for dynamic computation graphs, which are particularly useful in time series modeling.

5.3 Model Evaluation

Due to the nature of our dataset, using a regular, pure performance-focused evaluation metric such as Mean Squared Error (MSE) or symmetric Mean Absolute Percentage Error (sMAPE) to base the choices made at each step was not an alternative. While these metrics remain essential for the final assessment, the primary goal in the Forecasting and Unfolding stages is to accurately capture the overall shape of the ground truth rather than precise numerical accuracy. To gauge this, the comparison between different models was made through a visual inspection of the output compared to the ground truth.

In the result section, production plan forecasts obtained from our two model implementations, the Deep Learning-based model and the Regression-based model, will be presented. Their performance is compared against three baseline forecasting strategies. The goal is to evaluate the effectiveness of our hierarchical modeling approach in forecasting the monthly production plan under the constraint of limited and irregular data.

The three baseline models used for comparison are:

- **Previous Year Production Plan Replication:** A naive forecast where the plan for the upcoming year is assumed to be identical to the previous year.

This serves as a simple benchmark to assess whether more complex models provide added value.

- **Basic LSTM Model:** A standard Long Short-Term Memory (LSTM) network trained directly on the production plan data. This model represents a conventional deep learning approach to time series forecasting, without the hierarchical structure or auxiliary data sources used in our proposed models.
- **Basic ARIMA Model:** A classical statistical forecasting model applied directly to the production plan data. This model serves as a baseline for evaluating the performance of our regression-based ARIMAX model, which incorporates external variables.

Each model's performance is evaluated using standard forecasting metrics, and visual comparisons are provided for selected products to illustrate the differences in predictive behavior. The results are discussed in terms of performance, robustness, and the ability to generalize across different product categories. To quantitatively evaluate the model performance, we will use two widely adopted forecasting metrics:

- **Root Mean squared Error (rMSE)**: The square root of the average squared distances between prediction and ground truth.
- **Symmetric Mean Absolute Percentage Error (sMAPE)**: A normalized metric that expresses forecast accuarcy in terms of a percentage, making it suitable to compare results of different scales of the plan.

6 Deep Learning-based Model

This section will detail the three different steps as shown in figure 5 for the Deep learning-based model.

6.1 Step 1: Production Forecasting

The initial stage of the hierarchical model architecture involves forecasting monthly production volumes. For the deep learning-based approach, an MLP architecture was selected due to its simplicity and effectiveness in capturing non-linear relationships. The architecture of the MLP is illustrated in Figure 6. The model accepts a 12-month sequence of historical production data as input and processes it through a series of fully connected layers, each employing the Rectified Linear Unit (ReLU) activation function. The output is a 12-month forecast of production values. Prior to training, the input data is normalized using the `MinMaxScaler` from the

`scikit-learn` library [32] to ensure numerical stability and improve convergence during optimization.

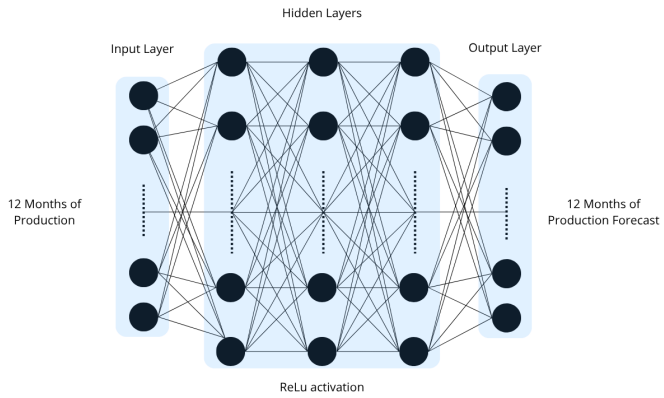


Figure 6: Production forecast using Multi-Layer Perceptron architecture (12x32, 32x32, 32x32, 32x12).

6.1.1 Training Procedure

The training process for the production forecasting model follows a standard supervised learning pipeline. To simulate a realistic forecasting scenario, the final year of available production data (corresponding to 2025) is reserved for testing. The remaining historical data is used for training and validation. Prior to model training, any year in which the total production volume is zero, typically corresponding to pre-launch or post-discontinuation periods, is excluded from the dataset.

Normalization is performed using the `MinMaxScaler` from `scikit-learn` [32], which scales the input features to the range [0, 1]. To prevent data leakage, the scaler is fitted exclusively on the training data and subsequently applied to both the training and validation sets.

The model is trained for 50 epochs using MSE loss, as implemented in the `PyTorch` library [33]. This loss function penalizes large deviations between predicted and actual values, thereby encouraging the model to learn the underlying structure of the production time series.

6.1.2 Output

The objective of this forecasting step is not to replicate the exact monthly production values, but rather to approximate the overall temporal pattern and distribution of production across the forecast horizon. As shown in Figure 7, the model captures

the general trend and seasonality of the ground truth data, even if minor deviations in magnitude are present.

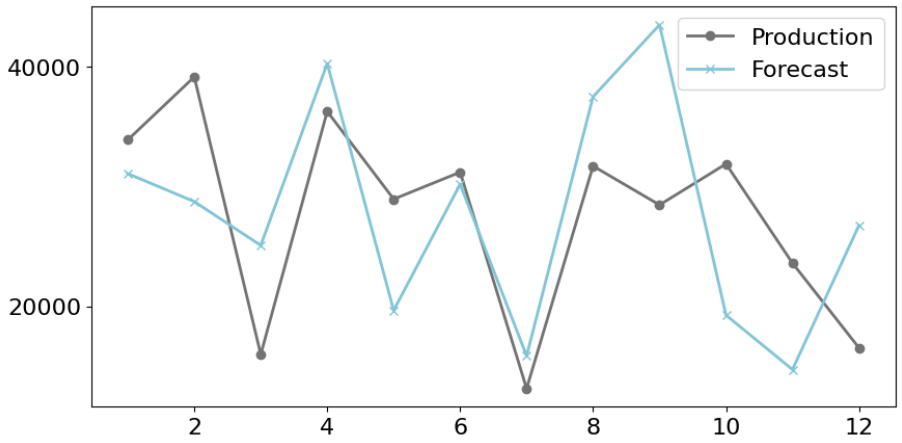


Figure 7: Example output of the deep learning-based production forecast compared to ground truth.

6.2 Step 2: Quarterly Mapping

The second stage of the hierarchical forecasting model serves to bridge the gap between the predicted monthly production data and the target quarterly plan values. In the deep learning pipeline, this is achieved by aggregating three consecutive months of forecast production data and combining it with the stock level of the corresponding product from the month preceding the quarter. This composite input is then passed through a sequence of fully connected layers, as illustrated in Figure 8.

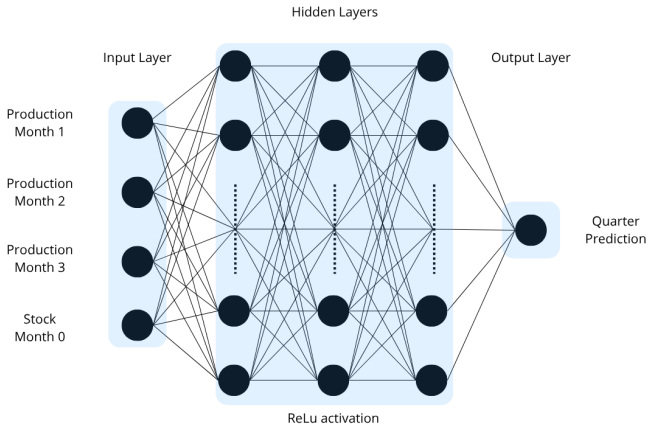


Figure 8: Mapping of production to quarterly planning data using a fully connected neural network (4x32, 32x32, 32x32, 32x1).

This component of the model is designed to learn the relationship between short-

term production behavior and quarterly plan outcomes. By incorporating stock levels as an additional input feature, the model is better equipped to account for inventory dynamics that may influence production plan realization.

6.2.1 Training Procedure

The training of the quarterly mapping model is conducted in two distinct phases. The first phase involves training the model on historical data, where actual production values are mapped directly to the corresponding quarterly plan figures. This phase constitutes a complete training cycle and is executed over 50 epochs.

In the second phase, the model undergoes fine-tuning using forecast production data generated by the production forecasting model described in Section 6.1. Specifically, the trained production forecasting model is applied to all available sequences of historical production data to generate synthetic forecasts. These forecasts, along with the corresponding stock levels, form the input dataset for fine-tuning the quarterly mapping model. This two-stage training process ensures that the model is both grounded in historical patterns and adapted to the characteristics of the forecast inputs it will encounter during inference.

6.2.2 Output

As shown in the three example outputs in Figure 9, the model manages to match the quarterly production plan very well, in almost all cases showing the same general pattern as the ground truth.

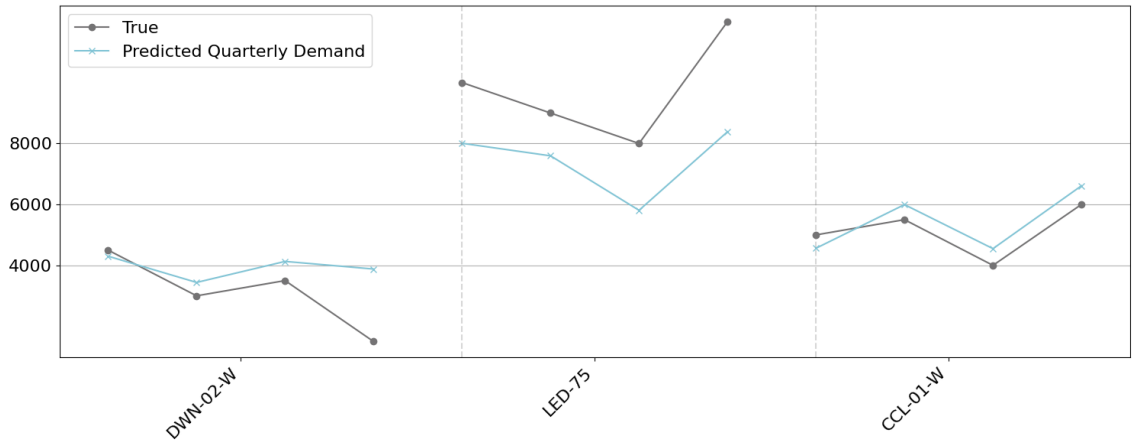
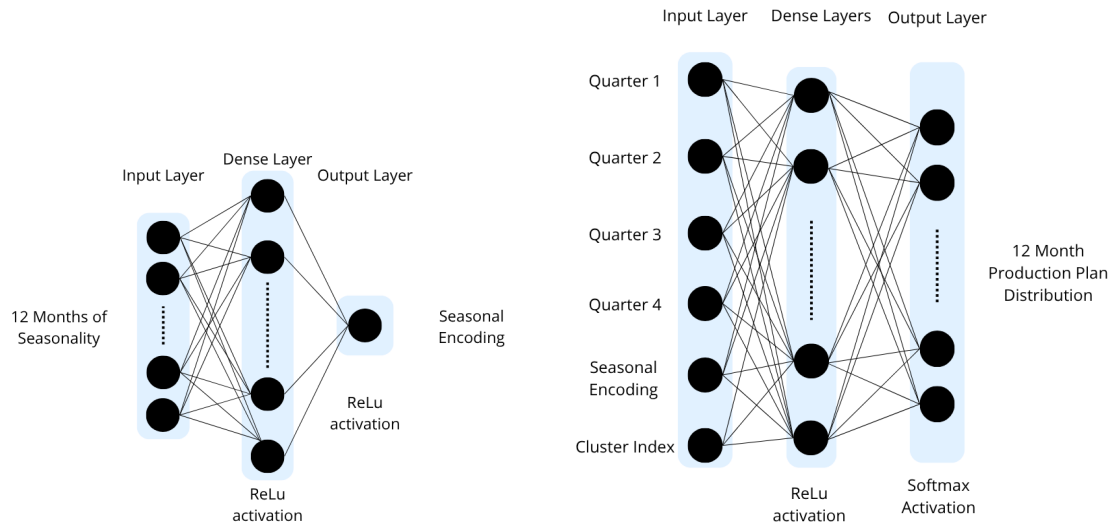


Figure 9: Examples of mapping from forecast production to quarterly production plan for three separate articles.

6.3 Step 3. Unfolding



(a) Encoding of seasonal component of installation data.
(12×32, 32×1)

(b) Unfolding of quarterly plan data to monthly plan data.
(6×32, 32×12)

Figure 10: Comparison of seasonal encoding and production plan unfolding processes.

The final stage of the hierarchical forecasting model involves disaggregation, or *Unfolding* of the predicted quarterly plan into a monthly distribution. This unfolding process consists of two primary components. The first component incorporates seasonal and categorical information to guide the shape of the monthly distribution. The second component transforms the quarterly plan into a normalized monthly profile.

To capture the general distributional characteristics of the target plan, two complementary strategies are applied. The first strategy uses a clustering-based approach to encode prior knowledge about product behavior. Specifically, the intuitive clustering method, which is based on broad product categories, was found to produce the most consistent results and is therefore used in the final model. This categorical information provides a proxy for typical production plan patterns associated with different product types.

The second strategy introduces a seasonal encoder to extract temporal structure from the installation data. Among the available datasets, the installation data exhibited the most pronounced seasonality. The encoder processes the most recent 12 months of installation data for a given product through an MLP and outputs a single latent feature. This feature is intended to represent the seasonal characteristics

of the product's deployment and is used to inform the unfolding process.

The unfolding model itself is implemented as an MLP that takes as input the four predicted quarterly plan values, the cluster index, and the seasonal encoding. The network outputs a 12-element vector representing the relative monthly distribution of the production plan. A softmax activation function is applied to the final layer to ensure that the output forms a valid probability distribution. This normalized output is then scaled by the *Total Goal Plan*, which is a predefined annual production target for each product. During training, this target is set to the sum of the ground truth monthly plan for the corresponding year.

7 Regression-based model

7.1 Step 1. Production Forecasting

The first step of the regression-based model makes use of an ARIMAX model, which is implemented using the Python libraries `statsmodels` [34] and `pmdarima` [35], to predict production for the following 12 months. The latter, inspired by the auto-ARIMA functionality in R, is used to automatically select the optimal model order (p,d,q) based on the Akaike Information Criterion (AIC). Once the optimal parameters are identified, the ARIMAX model is fitted to historical production data for a chosen product, enabling it to capture temporal patterns. These patterns are then leveraged to generate monthly production forecasts for the upcoming 12 months.

7.1.1 Feature Selection

The selection of external variables for the ARIMAX model is conducted through a feature analysis. This is key to fully utilizing the advantages of the ARIMAX model compared to the original ARIMA model.

The feature selection itself is mainly done with a RF-regressor. The number of trees was selected as 100, which is standard in `sci-kit` learn [36] that was used to implement the RF-regressor. Since the RF-regressor played a relatively small part in this project, we deemed it outside the scope of the report to perform any sort of parameter tuning on this.

The classical 0.05 significance threshold is used for the choice of external variables to be used when forecasting the production. If there are no external features which scores exceed this threshold, the production to production forecasting will be done by an ordinary ARIMA model.

The random forest regressor and the permutation importance were implemented using `scikit learn` in Python.[37]

The external variables available for feature selection are as follows:

- Installation data
- Stock level data
- Rolling variance - The rolling variance of the production data with a window size of 3.
- Rolling mean - The rolling mean of the production data with a window size of 3.
- Trend component - The trend component of the production data obtained from the STL-decomposition.
- Seasonal component - The seasonal component of the production data obtained from the STL-decomposition.
- Lagged production - The production lagged by 1,2, and 3 time-steps.

Similar to the discussion regarding step one of the deep learning model, the objective of this step is not to obtain a perfect match for the prediction, but rather for the model to be able to capture structural patterns. As can be seen in 11, the model manages to capture the general structure and pattern to some extent.

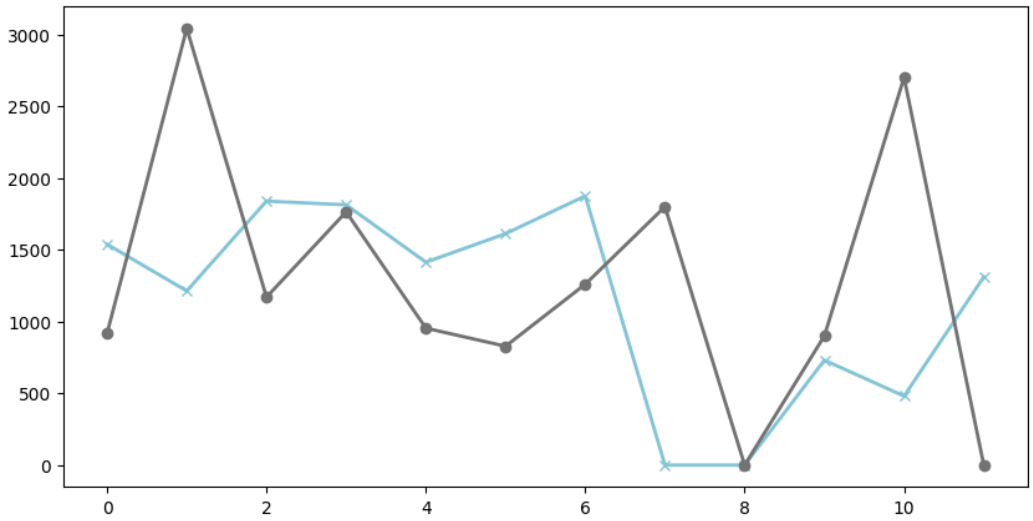


Figure 11: Example of predicting productions twelve months ahead

7.2 Step 2. Quarterly Mapping

The output of the first step in this model pipeline is a 1x12 vector corresponding to the predicted 12 months of production data. In order to be able to achieve results in the required plan format, this vector needs to be converted from being a prediction of production to a prediction of the production plan.

The regression based model performs the quarterly mapping by letting the produced twelve months, as well as a stock feature and a cluster feature, pass through a MLP. This process is done quarterly, meaning that the input vector to the MLP is of the shape 1x5, where the first three columns correspond to each month in a quarter, the fourth column is the stock level going into the first month of that certain quarter, and finally the fifth column is a cluster index. The MLP has a sequence of hidden layers, with LeakyReLU activation function between each layer. The network has one output neuron, corresponding to the predicted quarterly production plan. The MLP architecture is visualized in Figure 12

7.2.1 Training Procedure

The training for this step was divided in two batches. Similar to the quarterly mapping component of the deep learning model, the network was first trained on a straight mapping of the historical production data to the quarterly production plan for 75 epochs. Fine tuning is then, similar to the deep learning approach, performed on the forecasted production data for another 15 epochs. The training was performed using ADAM-optimizer [38], and an MSE-loss function.

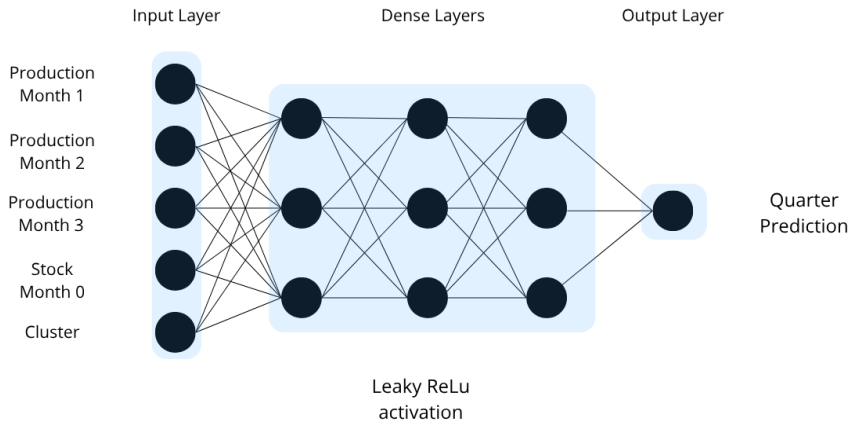


Figure 12: Mapping of Production to Quarterly plan (5x3,3x3,3x3,3x1)

7.2.2 Output

After doing this for all twelve months on the output from the production prediction component, we obtain results for the production plan for an entire year, but on a quarterly basis. An example of this is shown in Figure 13. Once again we note that the objective is not for the model to perfectly match the actual data, but rather to capture the overall pattern and distribution between the quarters.

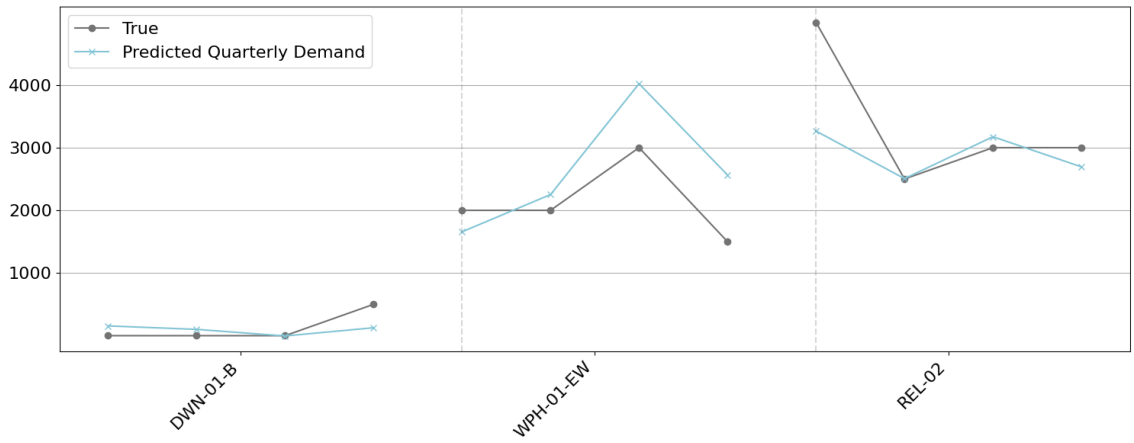


Figure 13: Example of the output from step 2 of the regression based model

7.3 Step 3. Unfolding

The third and final part of the regression-based model is another MLP, which takes as input a 1x5 vector, where the five columns correspond to the predicted quarterly plan from step two for every quarter in a year, as well as a cluster index based on the production data. The clustering method used for this method is the quarterly peak clustering. The architecture of this MLP is visualized in Figure 14. The 1x12

output vector here is a prediction of the monthly distribution of the total yearly plan for one product. This vector is then scaled by the *Total Goal plan*, meaning that the resulting 1x12 vector represents the monthly production plan one year ahead.

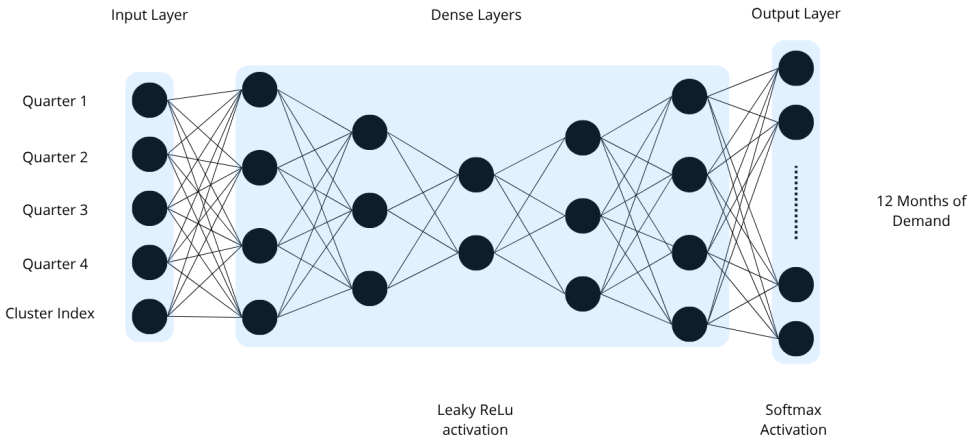


Figure 14: Mapping of production to quarterly production plan (5x4,4x3,3x2,2x3,3x4,4x12)

7.3.1 Training Procedure

The unfolding component of the regression based model is trained on the output from the second component of the model, and the ground truth production plan. The model was trained for 50 epochs, and given the limited amount of data, no fine-tuning was performed in this step. Similar to the second component of this model, MSE was used for the loss function, and the ADAM optimizer was used to train the weights in our model.

8 Results

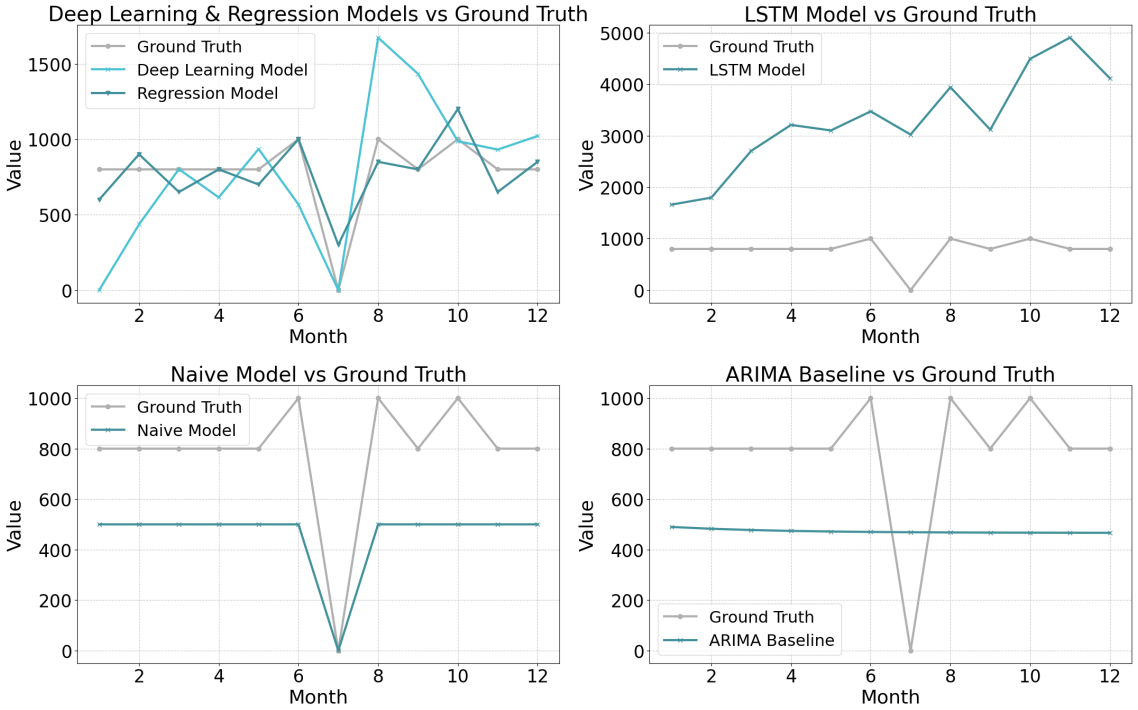
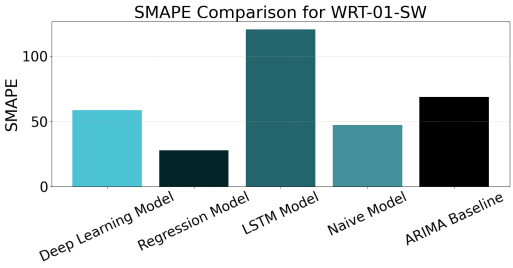
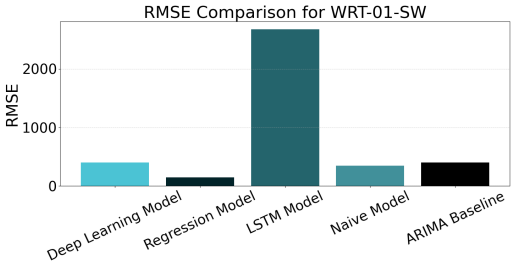


Figure 15: Final Predictions Compared to the Ground Truth for the article WRT-01-SW

Figure 15 shows the final predictions of our models. The top left shows the performance of both proposed hierarchical architectures; the other three plots show a comparative baseline for LSTM (top right), a Naive prediction (bottom left) and an ARIMA-prediction (bottom right). These comparisons highlight the strengths of our models, capturing the general behavior of the data very well. The Regression-based model is a near perfect fit for this article, displaying a very stable prediction, without any major discrepancies from the ground truth. The Deep Learning-based model displays a more erratic forecast that struggles to stay close to the ground truth but still manages to capture the important features of the production plan. The baseline models really struggle with this prediction. The LSTM application is not close either in terms of the general shape of the time series or in terms of the number of units forecast. The naive prediction is naturally similar in shape but does not capture the volume of the ground truth.



Symmetric Mean Average Percentage Error (sMAPE) for all compared models for WRT-01-SW.



Root Mean Squared Error (rMSE) for all compared models for WRT-01-SW.

Figure 16: Performance metrics for the proposed models compared to baseline for WRT-01-SW. Lower is better.

Figure 16 compares a quantified performance measure for our models compared to the baseline performances on the WRT-01-SW article. The metrics indicate that the regression-based model is the best fit for this product, having achieved the lowest score in both metrics.

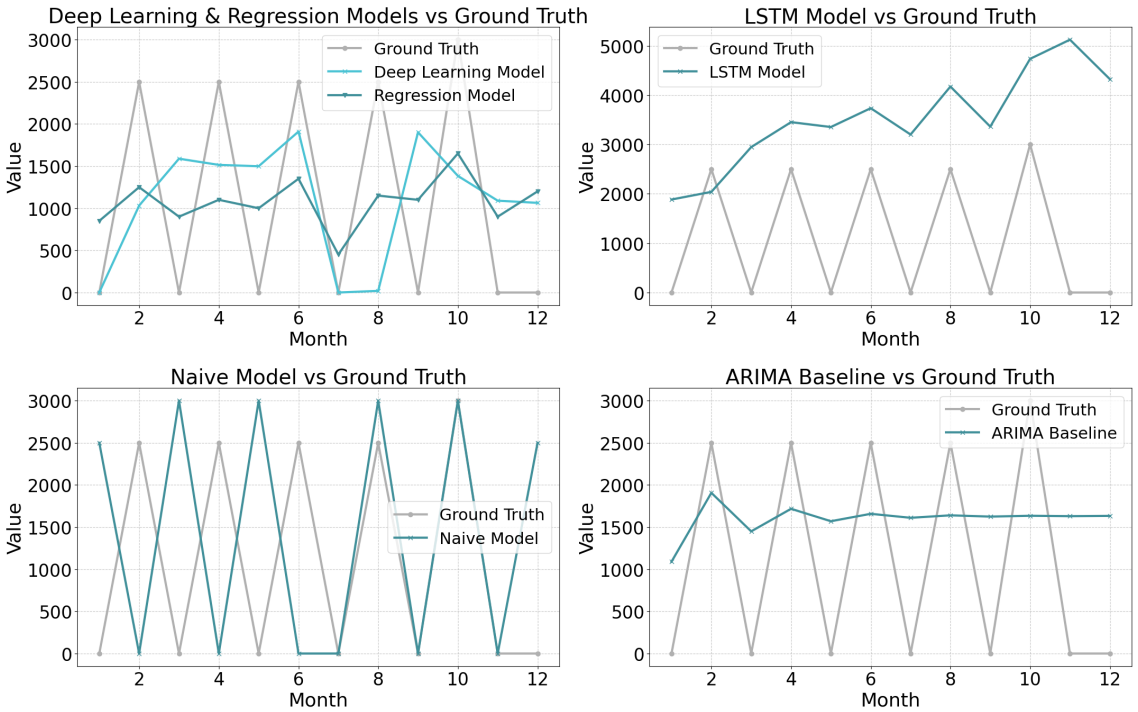


Figure 17: Final Predictions Compared to the Ground Truth of the article REL-02.

Figure 17 shows the final predictions of our models and the baseline comparisons together with the ground truth for the product REL-02. This product exemplifies one of the types of products that the proposed model struggles significantly with.

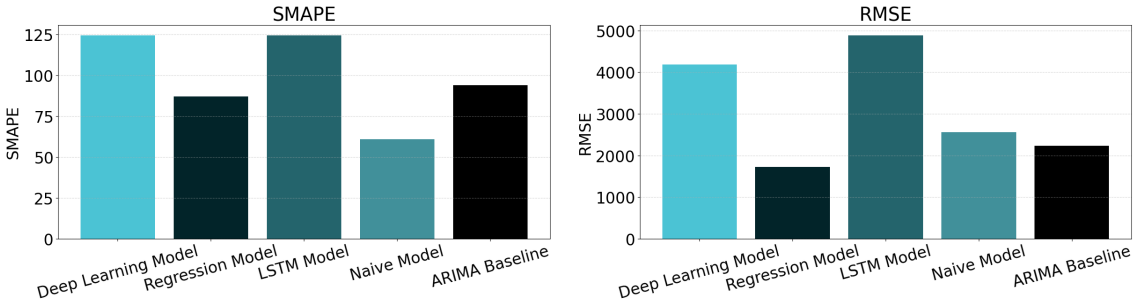
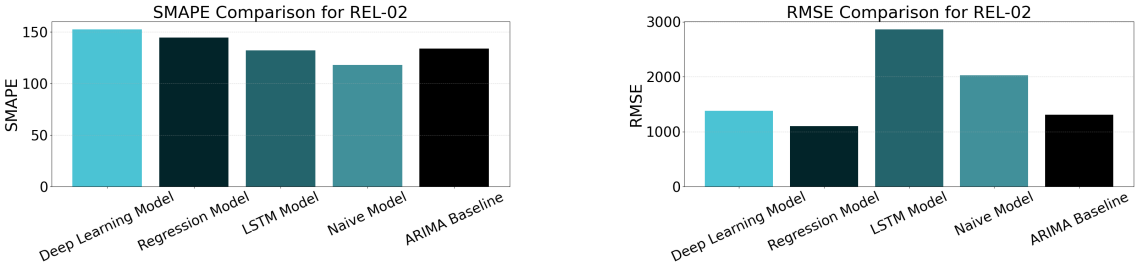


Figure 19: Average Performance Across all 26 Articles in Test Dataset.

This type of erratic production plan is not captured by the proposed models, which struggle to find the highs and lows. The naive prediction is the closest to this data, again being able to capture the general shape of the data very well, the main difference being a one-month shift for the first half of the year.



(c) Third image

(b) Second image

Figure 18: Performance metrics for the proposed models compared to baseline for REL-02. Lower is better.

Figure 18 shows the quantified performance for the models corresponding to the forecasts displayed in 17. Since the models performed poorly across the board, the metrics do not point to a specific model as the outright best model for this type of article.

Figure 19 shows the average performance of the models compared to baseline predictions. The regression model scores well in both metrics, while the deep learning model performs poorly. The naive prediction receives a consistently decent score.

All final forecasts produced by the proposed models can be found in tables 5, 6, and 7 in the appendix.

9 Discussion

The evaluation of the proposed hierarchical forecasting models reveals several important insights. Both the deep learning-based and regression-based approaches outperform the implemented baseline models, validating the effectiveness of decomposing the forecasting task into structured stages. By first predicting production, then mapping to the quarterly and finally monthly production plan, the models are generally able to circumvent the irregularities and noise present in the handcrafted production plan.

When comparing the performance metrics with the specific forecasts for each model, specifically in Figure 16, it is interesting to note that the ARIMA baseline scores very similarly to the deep learning approach, despite the deep learning model showing an intuitively better fit to the data. This highlights the difficulty in relying on these types of metrics during our model selection and training process; they simply do not give a fair indication of the performance of the models.

The average performance metrics for the test data, as shown in Figure 19, indicate that our proposed regression-based model is best suited for the task, outperforming the baseline models as well as our proposed deep learning approach. However, the naive approach scores better in the sMAPE metric. The Deep Learning model shows poor average results, which is not surprising due to the erratic nature of the forecasts it generates.

The deep learning model excels at capturing the extremes of the time series, particularly for products that have shown consistent seasonal patterns. This highlights the importance of the clustering and seasonality encoding for capturing the general behavior of the data when possible. The model does struggle when the time series becomes very erratic, as shown in Figure 17. This is most likely due to multiple factors, most notably the dataset bias. The dataset is not particularly balanced with regards to the general patterns. Articles with patterns similar to what is shown in Figure 15 occur quite frequently, and tend to have been in production for a longer time. Articles exhibiting more erratic patterns tend to have shorter production runs and show larger dissimilarities among each other. Despite our clustering and seasonal encoding, these patterns have been very difficult to generalize and capture.

The stand-out characteristic of the regression-based model is the stable nature of the generated forecasts. For most predictions, with the exception of the more erratic articles mentioned above, the model captures the pattern well, seldom straying far from the ground truth. The main drawback of the regression-based model is the strength of the deep learning model. It struggles in capturing the extremes of the dataset, both highs and lows, predicting a more generalized structure.

The apparent success of the regression-based model compared to our deep model implementation may lie in the use of a deeper, narrower step 3. implementation, being able to capture the production plan more accurately. Despite the deep model performing seemingly better in the first two steps, it falls short in this regard. This indicates that a deeper, narrow model architecture is more suitable for this task.

The comparison with the naive model highlights one of the main issues in viewing this problem as a time series forecasting task. The naive prediction, i.e. using the previous year's production plan as the prediction, produces relatively good results across the board. Naturally, it often keeps the same shape as the goal plan, with the main differences being the growth of planned production numbers. However, this is the case due to the nature of this specific dataset and does not undermine the value of the hierarchical method proposed in this dataset. Companies with a production plan that utilize a *Just-In-Time* strategy may benefit more from this type of specific, month-to-month prediction.

10 Conclusion

This thesis presents a hierarchical approach to production plan forecasting that attempts to circumvent inherent issues that are caused by multiple factors, such as irregular patterns in the production plan, limited data, and unique behavior for many of the 35 articles utilized for this thesis. The proposed model architecture outperforms both the baseline comparison models for this task, and often succeeds in connecting the historical production data to the corresponding production plan.

Even though the models proposed in this thesis are not immediately applicable and ready to use in a real world production planning automation, they serve as proof of concept. As the company continues to grow, and more data is collected, this type of hierarchical model architecture will continue to improve in performance.

It is shown that, by utilizing quite simple model architectures, such as an MLP, in an efficient way, relevant information can be extracted and learned even in exceptionally small and varied datasets.

11 Future Work

Due to the characteristics of the two proposed models, often complementing each other in terms of strengths and weaknesses, future work may want to look into a way of combining or merging the two predictions. In [39], a strategy for this specific purpose is proposed, AFTER. Applying this proposed algorithm may solve some of the most glaring issues of our current model implementations. This approach may work well for the forecasts where both of the proposed implementations lie in close proximity to the ground truth, but for the more erratic articles, it is unlikely to see drastic improvements.

A general change in approach to the problem may also be useful for further work on this specific project. As discussed, in many cases, the naive prediction has a general structure that is similar to the ground truth. Utilizing this by focusing on making accurate predictions on the total amount of expected articles to be produced may yield a more strategically useful result.

The development of a custom loss function that emphasizes general structure similarity rather than pure distance-based metrics would also be an interesting topic to look into when delving further into this problem. This could help the training procedure for the first two steps, that should find the general pattern of the data rather than the exact fit.

References

[1] Chris Dulhanty. *Issues in Computer Vision Data Collection: Bias, Consent, and Label Taxonomy*. Sept. 2020. URL: http://hdl.handle.net/10012/16414.

[2] Emma Strubell, Ananya Ganesh, and Andrew McCallum. *Energy and Policy Considerations for Deep Learning in NLP*. 2019. arXiv: 1906.02243 [cs.CL]. URL: https://arxiv.org/abs/1906.02243.

[3] Daniel Keleperas. *Exploring the Importance of Demand Forecasting*. Bachelor of Business Administration, International Business. Nov. 2023.

[4] Stephen Haben, Marcus Voss, and William Holderbaum. "Time Series Forecasting: Core Concepts and Definitions". In: *Core Concepts and Methods in Load Forecasting*. Springer, 2023, pp. 55–66. DOI: 10.1007/978-3-031-27852-5_5. URL: https://link.springer.com/chapter/10.1007/978-3-031-27852-5_5.

[5] OpenStax. *5: Time Series and Forecasting*. Accessed via Engineering LibreTexts. 2023. URL: https://eng.libretexts.org/Bookshelves/Data_Science/Principles_of_Data_Science_%28OpenStax%29/05%3A_Time_Series_and_Forecasting.

[6] Xiangjie Kong et al. "Deep learning for time series forecasting: a survey". In: *International Journal of Machine Learning and Cybernetics* (Feb. 2025). ISSN: 1868-808X. DOI: 10.1007/s13042-025-02560-w. URL: http://dx.doi.org/10.1007/s13042-025-02560-w.

[7] Robert B. Cleveland, William S. Cleveland, and Irma Terpenning. "STL: A Seasonal-Trend Decomposition Procedure Based on Loess". English. In: *Journal of Official Statistics* 6.1 (Mar. 1990). Copyright - Copyright Statistics Sweden (SCB) Mar 1990; Last updated - 2023-11-28, p. 3. URL: http://proxy.lib.chalmers.se/login?url=https://www.proquest.com/scholarly-journals/stl-seasonal-trend-decomposition-procedure-based/docview/1266805989/se-2.

[8] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering – A decade review". In: *Information Systems* 53 (2015), pp. 16–38. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2015.04.007. URL: https://www.sciencedirect.com/science/article/pii/S0306437915000733.

[9] T. Warren Liao. "Clustering of time series data—a survey". In: *Pattern Recognition* 38.11 (2005), pp. 1857–1874. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2005.01.025. URL: https://www.sciencedirect.com/science/article/pii/S0031320305001305.

[10] Jongseon Kim et al. *A Comprehensive Survey of Deep Learning for Time Series Forecasting: Architectural Diversity and Open Challenges*. 2025. arXiv: 2411.05793 [cs.LG]. URL: https://arxiv.org/abs/2411.05793.

[11] Mariel Abigail Cruz-Nájera et al. “Short Time Series Forecasting: Recommended Methods and Techniques”. In: *Symmetry* 14.6 (2022), p. 1231. DOI: 10.3390/sym14061231. URL: https://www.mdpi.com/2073-8994/14/6/1231.

[12] K. Kandananond. “A Comparison of Various Forecasting Methods for Autocorrelated Time Series”. In: *International Journal of Engineering Business Management* 4 (2012). DOI: 10.5772/51088. URL: https://journals.sagepub.com/doi/full/10.5772/51088.

[13] Jocelyn Barker. “Machine learning in M4: What makes a good unstructured model?” In: *International Journal of Forecasting* 36.1 (2020). M4 Competition, pp. 150–155. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2019.06.001. URL: https://www.sciencedirect.com/science/article/pii/S0169207019301463.

[14] Antonio Rafael Sabino Parmezan, Vinicius M.A. Souza, and Gustavo E.A.P.A. Batista. “Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model”. In: *Information Sciences* 484 (2019), pp. 302–337. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.01.076. URL: https://www.sciencedirect.com/science/article/pii/S0020025519300945.

[15] Vaia I. Kontopoulou et al. “A Review of ARIMA vs. Machine Learning Approaches for Time Series Forecasting in Data Driven Networks”. In: *Future Internet* 15.8 (2023), p. 255. DOI: 10.3390/fi15080255. URL: https://www.mdpi.com/1999-5903/15/8/255.

[16] Vassilios Assimakopoulos Spyros Makridakis Evangelos Spiliotis. “The accuracy of machine learning forecasting methods versus statistical ones”. In: *International Journal of Forecasting* 36.1 (2019), pp. 217–227. DOI: 10.1016/j.ijforecast.2019.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0169207019301189.

[17] H Bakhshi-Khaniki and SMT Fatemi Ghomi. “Integrated dynamic cellular manufacturing systems and hierarchical production planning with worker assignment and stochastic demand”. In: *International Journal of Engineering* 36.2 (2023), pp. 348–359.

[18] Dan Luo et al. “A Data-Driven Methodology for Hierarchical Production Planning with LSTM-Q Network-Based Demand Forecast”. In: *Symmetry* 17.5 (2025). ISSN: 2073-8994. DOI: 10.3390/sym17050655. URL: https://www.mdpi.com/2073-8994/17/5/655.

[19] Can Chen et al. “Hierarchical Demand Forecasting for Factory Production of Perishable Goods”. In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 188–193. DOI: 10.1109/BigData47090.2019.9006161.

[20] Si-An Chen et al. *TSMixer: An All-MLP Architecture for Time Series Forecasting*. 2023. arXiv: 2303.06053 [cs.LG]. URL: https://arxiv.org/abs/2303.06053.

[21] Jamal Fattah et al. “Forecasting of demand using ARIMA model”. In: *International Journal of Engineering Business Management* 10 (2018), pp. 1–9. DOI: 10.1177/1847979018808673.

[22] Robert H. Shumway and David S. Stoffer. “ARIMA Models”. In: *Time Series Analysis and Its Applications: With R Examples*. Cham: Springer International Publishing, 2017, pp. 75–163. ISBN: 978-3-319-52452-8. DOI: 10.1007/978-3-319-52452-8_3. URL: https://doi.org/10.1007/978-3-319-52452-8_3.

[23] Marcin Majka. “ARIMAX Time Series Forecasting with External Variables”. In: *ResearchGate* (Sept. 2024). URL: https://www.researchgate.net/publication/384196976_ARIMAX_Time_Series_Forecasting_with_External_Variables.

[24] Philipp Probst and Anne-Laure Boulesteix. “To Tune or Not to Tune the Number of Trees in Random Forest”. In: *Journal of Machine Learning Research* 18 (2018), pp. 1–18. URL: https://jmlr.org/papers/volume18/17-269/17-269.pdf.

[25] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[26] Romain Couronné, Pierre Probst, and Gilles Simon. “Random forests for high-dimensional variable selection”. In: *Bioinformatics* 34.21 (2018), pp. 3711–3718. DOI: 10.1093/bioinformatics/bty373. URL: https://academic.oup.com/bioinformatics/article/34/21/3711/4994791.

[27] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[28] GeeksforGeeks. *What is LSTM - Long Short Term Memory?* Accessed: 2025-05-24. 2025. URL: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/.

[29] Ailing Zeng et al. “Are Transformers Effective for Time Series Forecasting?” In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.9 (June 2023), pp. 11121–11128. DOI: 10.1609/aaai.v37i9.26317. URL: https://ojs.aaai.org/index.php/AAAI/article/view/26317.

[30] Addison Howard et al. *M5 Forecasting - Accuracy*. https://kaggle.com/competitions/m5-forecasting-accuracy. Kaggle. 2020.

[31] tslearn developers. *tslearn.clustering.TimeSeriesKMeans*. Accessed: May 21, 2025. tslearn. 2025. URL: https://tslearn.readthedocs.io/en/stable/gen_modules/clustering/tslearn.clustering.TimeSeriesKMeans.html.

[32] scikit-learn developers. *sklearn.preprocessing.MinMaxScaler*. Version 1.6.1. scikit-learn. 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

[33] PyTorch Developers. *torch.nn.MSELoss*. Accessed: May 21, 2025. PyTorch. 2025. URL: https://docs.pytorch.org/docs/stable/generated/torch.nn.MSELoss.html.

[34] Skipper Seabold and Josef Perktold. "Statsmodels: Econometric and statistical modeling with python". In: *Proceedings of the 9th Python in Science Conference* 57 (2010), p. 61.

[35] Taylor G. Smith et al. *pmdarima: ARIMA estimators for Python*. [Online; accessed 2025-06-02]. 2017–. URL: http://www.alkaline-ml.com/pmdarima.

[36] Scikit-learn Developers. *Scikit-learn: Ensemble Methods - Forests*. Accessed: 2025-06-02. 2025. URL: https://scikit-learn.org/stable/modules/ensemble.html#forest.

[37] Scikit-learn Developers. *RandomForestRegressor — scikit-learn 1.6.1 documentation*. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[38] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[39] Hui Zou and Yuhong Yang. "Combining time series models for forecasting". In: *International Journal of Forecasting* 20.1 (2004), pp. 69–84. ISSN: 0169-2070. DOI: https://doi.org/10.1016/S0169-2070(03)00004-9. URL: https://www.sciencedirect.com/science/article/pii/S0169207003000049.

12 Appendix

Article	Model	Forecast
BAT-01	Deep Learning model:	0, 27, 0, 0, 0, 53972, 0, 0, 0, 0, 0, 0
	Regression Based model:	3550, 5250, 3950, 4650, 4200, 5650, 1900, 4900, 4650, 6500, 3900, 4850
	Ground Truth:	6000, 6000, 6000, 6000, 6000, 6000, 0, 6000, 6000, 6000, 0, 0
CTR-01	Deep Learning model:	0, 6, 0, 0, 0, 105994, 0, 0, 0, 0, 0, 0
	Regression Based model:	7050, 10300, 7900, 9150, 8350, 11050, 3800, 9650, 9150, 12350, 7800, 9500
	Ground Truth:	24000, 19000, 9000, 9000, 9000, 9000, 0, 9000, 9000, 9000, 0, 0
DAL-01	Deep Learning model:	0, 1987, 3047, 2888, 2819, 3669, 0, 24, 3396, 2514, 1846, 1809
	Regression Based model:	1550, 2300, 1750, 2050, 1850, 2500, 850, 2150, 2050, 2950, 1750, 2200
	Ground Truth:	0, 4000, 0, 5000, 0, 5000, 0, 5000, 0, 5000, 0, 0
DIM-01	Deep Learning model:	0, 47092, 53012, 40040, 45926, 50031, 0, 148, 43747, 35854, 22793, 21358
	Regression Based model:	30100, 35850, 32950, 32650, 32100, 34150, 14700, 34100, 31900, 25200, 29700, 26600
	Ground Truth:	40000, 40000, 40000, 40000, 40000, 40000, 0, 40000, 40000, 40000, 0, 0
DIM-02	Deep Learning model:	0, 6071, 10553, 6768, 7038, 13701, 0, 64, 23306, 17294, 16272, 15933
	Regression Based model:	8100, 11450, 9000, 10200, 9450, 12050, 4300, 10750, 10200, 12600, 8750, 10150
	Ground Truth:	13000, 12000, 13000, 12000, 12000, 13000, 0, 14000, 14000, 14000, 0, 0
DWN-01-W	Deep Learning model:	0, 1485, 2312, 2480, 2477, 2699, 0, 1049, 3093, 1994, 1731, 1680
	Regression Based model:	1350, 2050, 1500, 1800, 1600, 2200, 700, 1900, 1800, 2700, 1500, 1900
	Ground Truth:	2000, 2000, 2000, 2000, 2000, 2000, 0, 3000, 3000, 3000, 0, 0
EXT-01	Deep Learning model:	0, 1082, 1677, 1557, 1557, 2014, 0, 20, 2082, 1531, 1253, 1227
	Regression Based model:	900, 1350, 1000, 1200, 1050, 1450, 500, 1250, 1200, 1800, 1000, 1300
	Ground Truth:	0, 4000, 0, 3000, 0, 4000, 0, 3000, 0, 0, 0, 0
GWY-01	Deep Learning model:	0, 2797, 775, 5579, 1500, 5482, 0, 0, 107, 195, 11, 52
	Regression Based model:	1050, 1600, 1150, 1400, 1250, 1750, 550, 1500, 1400, 2100, 1150, 1500
	Ground Truth:	1500, 1500, 1500, 1500, 1500, 1500, 0, 1500, 1500, 1500, 1500, 1500
LED-10	Deep Learning model:	0, 4440, 6418, 5804, 5986, 7310, 0, 48, 7111, 5418, 4289, 4176
	Regression Based model:	3300, 4950, 3700, 4350, 3950, 5350, 1800, 4600, 4400, 6300, 3650, 4650
	Ground Truth:	8000, 7000, 7000, 7000, 7000, 7000, 0, 8000, 0, 0, 0, 0
LED-75	Deep Learning model:	0, 2840, 4530, 4148, 4137, 5534, 0, 58, 5966, 4389, 3726, 3672
	Regression Based model:	2550, 3800, 2850, 3350, 3050, 4100, 1400, 3550, 3350, 4750, 2850, 3500
	Ground Truth:	3000, 4000, 3000, 3000, 3000, 3000, 0, 4000, 4000, 4000, 4000, 4000

Table 5: Predictions for all articles in test set

Article	Model	Forecast
OUT-01-B	Deep Learning model:	0, 2292, 447, 2881, 461, 6255, 0, 0, 38, 107, 2, 17
	Regression Based model:	800, 1200, 900, 1050, 950, 1300, 450, 1100, 1100, 1600, 900, 1150
	Ground Truth:	1000, 1000, 1000, 1000, 1000, 1000, 0, 1000, 1500, 1500, 1500, 1000
OUT-01-G	Deep Learning model:	0, 316, 432, 542, 553, 482, 0, 37, 397, 279, 233, 228
	Regression Based model:	200, 350, 250, 300, 250, 350, 100, 300, 300, 450, 250, 300
	Ground Truth:	0, 0, 300, 300, 300, 300, 0, 500, 500, 500, 500, 300
OUT-01-W	Deep Learning model:	0, 846, 919, 1290, 1399, 1013, 0, 7, 609, 476, 372, 370
	Regression Based model:	450, 700, 500, 600, 550, 750, 250, 650, 650, 950, 500, 650
	Ground Truth:	700, 700, 500, 500, 500, 700, 0, 500, 800, 800, 800, 800
REL-01-2P	Deep Learning model:	0, 1770, 826, 1846, 860, 2985, 0, 0, 224, 327, 51, 110
	Regression Based model:	550, 850, 650, 750, 700, 950, 300, 800, 800, 1200, 650, 850
	Ground Truth:	1000, 1000, 1000, 1000, 1000, 1000, 0, 1000, 1000, 1000, 0, 0
REL-02	Deep Learning model:	0, 1031, 1588, 1515, 1498, 1910, 0, 20, 1901, 1384, 1090, 1064
	Regression Based model:	850, 1250, 900, 1100, 1000, 1350, 450, 1150, 1100, 1650, 900, 1200
	Ground Truth:	0, 2500, 0, 2500, 0, 2500, 0, 2500, 0, 3000, 0, 0
SPR-01-BL	Deep Learning model:	0, 330, 663, 462, 873, 355, 0, 1921, 1190, 872, 840, 995
	Regression Based model:	550, 800, 600, 700, 650, 900, 300, 750, 750, 1100, 600, 800
	Ground Truth:	1000, 500, 500, 500, 500, 500, 0, 1000, 2000, 2000, 0, 0
WMS-01	Deep Learning model:	1, 1363, 2533, 1932, 2935, 1802, 0, 5404, 4597, 3163, 2988, 3283
	Regression Based model:	1950, 2900, 2150, 2550, 2300, 3150, 1050, 2700, 2600, 3750, 2150, 2750
	Ground Truth:	3000, 3000, 3000, 3000, 3000, 3000, 0, 4000, 4000, 4000, 0, 0
WPH-01	Deep Learning model:	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
	Regression Based model:	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
	Ground Truth:	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
WPH-01-EB	Deep Learning model:	0, 233, 427, 328, 499, 303, 0, 886, 762, 525, 495, 542
	Regression Based model:	300, 450, 350, 400, 400, 500, 200, 450, 450, 650, 400, 500
	Ground Truth:	500, 0, 500, 0, 1000, 0, 0, 1000, 0, 1000, 0, 1000
WPH-01-EW	Deep Learning model:	0, 400, 752, 569, 874, 528, 0, 1654, 1377, 950, 899, 994
	Regression Based model:	550, 850, 650, 750, 700, 950, 300, 800, 800, 1150, 650, 800
	Ground Truth:	1000, 0, 1000, 1000, 1000, 1000, 0, 0, 1000, 1000, 1000, 1000

Table 6: Predictions for all articles in test set

Article	Model	Forecast
WPH-01-SA	Deep Learning model:	0, 369, 681, 521, 794, 482, 0, 1429, 1220, 841, 793, 870
	Regression Based model:	500, 750, 550, 700, 600, 850, 250, 700, 700, 1050, 550, 750
	Ground Truth:	0, 1500, 0, 1500, 0, 1000, 0, 1000, 0, 1500, 0, 1500
WPH-01-SW	Deep Learning model:	0, 13583, 2319, 29295, 4772, 32256, 0, 0, 195, 479, 11, 91
	Regression Based model:	5700, 8100, 6350, 7200, 6650, 8550, 3000, 7600, 7200, 9150, 6200, 7300
	Ground Truth:	8000, 7000, 7000, 7000, 7000, 7000, 0, 8000, 8000, 8000, 8000, 8000
WRT-01-EB	Deep Learning model:	0, 23, 43, 33, 50, 30, 0, 88, 76, 52, 49, 54
	Regression Based model:	50, 50, 50, 50, 50, 50, 0, 50, 50, 50, 50, 50
	Ground Truth:	0, 0, 0, 0, 0, 0, 0, 500, 0, 0, 0, 0
WRT-01-EW	Deep Learning model:	0, 112, 122, 175, 207, 121, 0, 15, 82, 63, 50, 52
	Regression Based model:	50, 100, 50, 100, 50, 100, 50, 100, 100, 150, 100, 100
	Ground Truth:	0, 0, 0, 0, 0, 500, 0, 0, 0, 500, 0, 0
WRT-01-SA	Deep Learning model:	0, 32, 60, 46, 69, 42, 0, 124, 107, 74, 69, 76
	Regression Based model:	50, 50, 50, 50, 50, 50, 50, 50, 50, 100, 100, 100
	Ground Truth:	0, 0, 200, 0, 0, 0, 0, 0, 0, 500, 0, 0
WRT-01-SW	Deep Learning model:	0, 436, 802, 614, 935, 568, 0, 1673, 1433, 987, 931, 1021
	Regression Based model:	600, 900, 650, 800, 700, 1000, 300, 850, 800, 1200, 650, 850
	Ground Truth:	800, 800, 800, 800, 800, 1000, 0, 1000, 800, 1000, 800, 800

Table 7: Predictions for all articles in test set

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY