Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 277 (2026) 739–748

Procedia Computer Science

www.elsevier.com/locate/procedia

7th International Conference on Industry of the Future and Smart Manufacturing (former International Conference on Industry 4.0 and Smart Manufacturing)

The Influence of Preprocessing on Time Series Prediction

Florian Grimm[a,*], Günter Bitsch[a], Clemens van Dinther[b]

[a]Reutlingen University, Alteburgstr. 150, 72762 Reutlingen, Germany
[b]Karlsruhe Institute of Technology, Kaiserstr. 89-93, 76133 Karlsruhe, Germany

Abstract

This study examines the influence of preprocessing techniques in relation to time series forecasting using the example of demand prediction. By conducting 80 different experiments with various preprocessing methods like transformation (Log, Box-Cox, Yeo-Johnson), detrending (differencing), and scaling (standardization, min-max and robust scaling), insights about the behavior and correlation between the preprocessing step and a forecasting task are gathered. Using the M5 dataset with real-world sales data from Walmart and two different neural network types, one Multi-Layer-Perceptron and one Long Short-Term Memory network, the impact of preprocessing on a model's performance can be shown. The results emphasize the crucial role of preprocessing in improving model accuracy and form the basis for future research on automatic preprocessing selection based on time series characteristics.

Keywords: Preprocessing; Time Series Preparation; Time Series Forecasting; Deep Learning; Artificial Intelligence;

1. Introduction

The application of artificial intelligence (AI) algorithms is more and more common in the daily live of everyone. About two third of the earths population use AI on a regular basis [1] and over 80% believe that AI brings benefits to their lives. Within the corporate environment the situation differs enormously. According to a recent study of McKinsey, only 1% have the opinion, that AI is integrated in a beneficial way inside their company structure [2]. 78% of global companies use AI [3], compared to only 39% of small and medium sized enterprises [4]. On the other hand, over 90% of the companies plan to increase their AI investments over the next couple of years, and according to another study conducted by ICONIQ Capital, enterprises that include AI from the very start are scaling faster than those which are incorporating AI over time [5]. This shows the enormous potential behind AI and its importance within the corporate world and industry.

* Corresponding author. Tel.: +49 (0) 7121 271 1498
E-mail address: florian.grimm@reutlingen-university.de

But contrary to what many people believe, you can't simply solve every problem with AI. Besides the circumstances that AI models have to be finetuned to specific problems, also the preprocessing plays an important role. For example, the same AI networks show different performance levels when fed with the same dataset that has been preprocessed using different methods [6].

There are different preprocessing algorithms, that can be applied to data. In the case of images for example, the preprocessing mostly is straight-forward: One uses a mapping algorithm, that scales all values from the input image to a value between 0-1. Images usually have a range between 0-255 (in case of 8-bit images), the mapping is therefore self explanatory (0 to 0, 255 to 1). This also applies on 16-bit images for example, as the maximum of a pixel value just increases to 65,535, but is fixed to a certain value (depending on the bit format). Due to the infinite numbers between two integer values, the value behind is fixed to a certain float value. There are several more image preprocessing algorithms, that can be applied, an overview can be found in [7], [8] or [9].

In case of time series, this is more complicated. Time series have complete different characteristics. For example, temperature data in degree Celsius can either be negative or positive. When using Kelvin, only positive values are possible. Within typical data like stock or sales, only positive numbers are possible. Most also have in common, that there is no defined maximum, and in some cases even no minimum. This makes it very difficult to approach a proper preprocessing of the time series. If one applies a mapping to 0-1, there might easily be new values, that would be mapped to a number larger than 1. Also values might be negative after the mapping, which makes it difficult for models to deal with it [10]. Several research has been performed on the field of time series preprocessing, such as [11], [12] or [13]. With this study, the research will be extended by an additional set of experiments for time series preprocessing.

1.1. Problem Description

Since preprocessing seems to be important before applying AI algorithms, as it can be seen in the image processing case, the question arises: ***How should time series properly be preprocessed and how strong is the impact of choosing the proper preprocessing approach?*** The training of a forecasting network is highly time consuming. With the variety of different existing preprocessing algorithms, a brute-force approach of permuting all considerable preprocessing methods in combination with the selected prediction algorithm would cost an immense amount of time and is therefore not feasible. But on the other hand, if the impact is noticeable in both, the performance and accuracy of the network, the choice of correct data handling might be crucial. To investigate, whether the preprocessing of time series has a considerable influence, an experiment in the domain of demand prediction is conducted: A dataset containing sales data is taken and preprocessed in different ways, including a transformation, detrending and a scaling procedure. Using a generic deep learning architecture as in [14] as forecasting algorithm, a comparison is performed to determine, what influence the chosen preprocessing mode has on the prediction results.

1.2. Structure of this Study

This paper is divided into three major parts. In Section 2: **Approach and Methodology**, the used dataset, the different transformation and preprocessing steps as well as the architecture of the used neural networks and the training parameters are described. A detailed overview of the experiment design is illustrated as well. Section 3: **Results and Discussion** shows results of the different experiments and discusses and interprets the findings. Various plots and tables illustrate tendencies and relationships. A summary, possible limitations and an outlook of further investigations is outlined in Section 4: **Conclusion and Outlook**.

The study contributes to the field of data preprocessing for deep learning algorithms in the context of time series prediction. It tries to investigate the impact of different preprocessing algorithms for specific forecasting tasks and creates the basis for a comprehensive study in the ongoing research.

2. Approach and Methodology

To investigate the stated research questions, an experiment is conducted with different preprocessing methods for deep learning based time series prediction. Two neural network architectures are selected to cover both, generic AI and

specific forecasting models. In this section, information about the used dataset, the different preprocessing algorithms, the structure of the neural networks as well as the experiment design is provided.

2.1. Used Datasets

To make this study reproducible, a freely available dataset is used in the experiment. Since the researchers domain is demand prediction, the main focus is set on sales data, as they occur in a wide variety of areas (business-to-business retailing, production planning, and many more).

Within the problem of correct sales forecasting, the M5 dataset [15], which consists of sales data from the Walmart Group, is particularly well known. It covers 42,840 time series that represent the demand of 30,490 products. The data is gathered over a period of 5.5 years over different states and stores in the U.S.A. and is available on a daily basis - which corresponds to 1,941 entries per time series.

To limit the run-time in this first investigation, the data is categorized accordingly to Syntetos et al. [16] into the four categories lumpy, intermittent, erratic and smooth. For each category, 200 time series are sampled, resulting in 800 time series in total. The last 28 days are taken as testing set, to match the M5 competition settings [15]. The training set is created by a rolling window procedure, taking 28 days as training vector and the 29th day as label and then shift the window one by one forward, resulting in 1,885 training vectors per time series.

2.2. Preprocessing Steps

Preprocessing is key to many areas of data analysis and deep learning. It is crucial for the performance of neural networks and helps to stabilize variances and supports convergence during model optimization. There are different methods, that usually are applied to the training data. For the conducted experiments, the procedures in Figure 1 are considered, which are described in the following subsections.

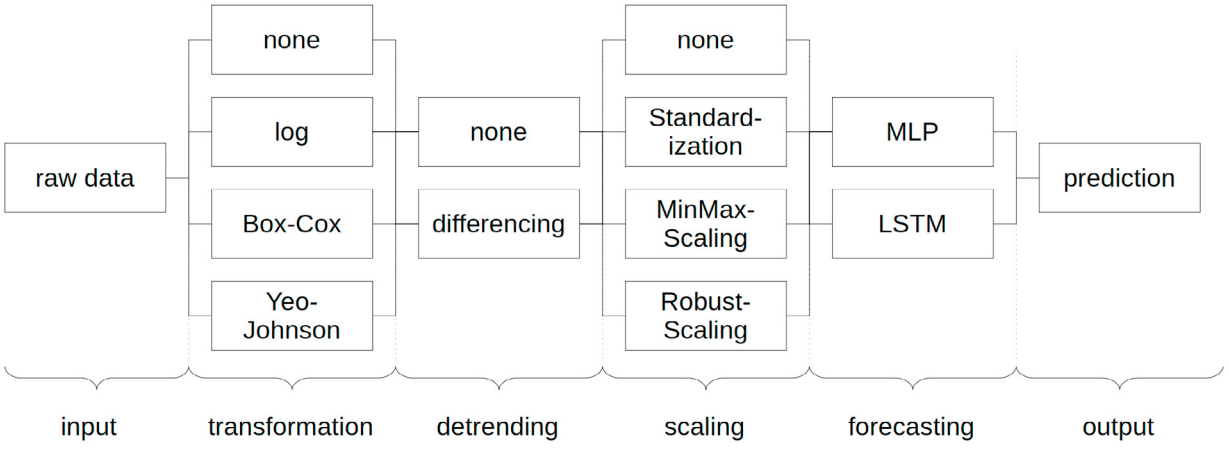

Fig. 1. Design of the different steps of the conducted experiments.

2.2.1. Distribution of Time Series

Forecasting models can benefit from the application of transformation methods like Box-Cox Transformation or Log Transformation. These methods are applied to stabilize variance and that the data gets more normally distributed. Especially sales data, that shows exponential trends, can be linearized, which makes it easier for models to handle. Both algorithms assume that the data consists of positive values.

Log Transformation. Log Transformation is a procedure that replaces a feature value with its logarithm. In most cases, the natural logarithm or the decadic logarithm is applied.

$$x' = \log(x) \tag{1}$$

where:

x = is the original feature value

Box-Cox Transformation. The Box-Cox Transformation is a power transformation of the given feature.

$$x' = \begin{cases} \frac{x^{\lambda}-1}{\lambda} & \text{for } \lambda \neq 0 \\ \log(x) & \text{for } \lambda = 0 \end{cases} \tag{2}$$

where:

x = is the original feature value
λ = is a parameter that determines the power to which all data should be raised

The λ parameter is typically estimated from the data using a maximum likelihood estimation. The goal is to find a value, so that the transformed data is as close to normal as possible.

Yeo-Johnson Transformation. The Yeo-Johnson Transformation is a modification of the Box-Cox Transformation, that can handle negative values.

$$x' = \begin{cases} \frac{(x+1)^{\lambda}-1}{\lambda} & \text{for } x \geq 0, \lambda \neq 0 \\ \log(x+1) & \text{for } x \geq 0, \lambda = 0 \\ -\frac{(-x+1)^{2-\lambda}-1}{2-\lambda} & \text{for } x < 0, \lambda \neq 2 \\ -\log(-x+1) & \text{for } x < 0, \lambda = 2 \end{cases} \tag{3}$$

where:

x = is the original feature value
λ = is a parameter that determines the power to which all data should be raised

2.2.2. Trends inside Time Series

Trends in time series introduce non-stationarity, which tends to confuse neural networks. To prevent this behavior, detrending can be used to make time series stationary. Stationary means, that statistical properties like mean and variance do not change over time. Especially model architectures, like Long Short-Term Memory, can benefit from stationary input data, as it helps these models to learn underlying dynamics rather than memorizing the trend itself. As detrending method, the procedure of differencing is chosen, where the previous observation is subtracted from the current observation.

$$x'_t = x_t - x_{t-1} \tag{4}$$

where:

x = is the original feature value
t = is the current time step

2.2.3. Scaling of Time Series

There are several common feature scaling algorithms that are used as preprocessing for AI algorithms. Scaling is a crucial step before applying deep learning on data. It ensures a faster convergence for gradient descent procedures and helps to avoid exploding or vanishing gradients.

Standardization. Standardization, also called Z-Score Normalization, is the process of transforming features or data so that they have an average equals 0 and a standard deviation equals 1. It is the preferred procedure when data is approximately normally distributed.

$$z = \frac{x - \mu}{\sigma} \tag{5}$$

where:

x = is the original feature value
μ = is the mean of the feature
σ = is the standard deviation of the feature

Normalization. Normalization, or more specific min-max normalization or min-max scaling, describes the process of rescaling all feature values to a certain interval, mostly [-1, 1] or [0, 1]. This procedure works best, when the data range is bounded (as in images usually) and when there are no outliers.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{6}$$

where:

x = is the original feature value
$\min(x)$ = is the minimum value of the features in the training data
$\max(x)$ = is the maximum value of the features in the training data

Robust Scaling. Robust Scaling, sometimes also called Standardization with median and interquartile range, is applied, when data contains many outliers. It is not necessary, when the data is well balanced or normally distributed.

$$x' = \frac{x - Q_2(x)}{Q_3(x) - Q_1(x)} \tag{7}$$

where:

x = is the original feature value
$Q_i(x)$ = are the three quartiles (1 = 25th, 2 = 50th, 3 = 75th)

There are several more algorithms, such as *MaxAbs Scaling* or *Unit Vector Normalization*, but since the most common applied ones are the previous described scaling methods, other methods are not considered in this study. For all scaling algorithms, the Python library scikit-learn is used [17].

2.3. Model Architectures and Training

Two different model architectures are tested using the different combinations of the preprocessing section (see subsection 2.2). One architecture uses a Multi-Layer-Perceptron (MLP), consisting of multiple hidden fully connected layers with several dropout layers to prevent overfitting and Rectified Linear Unit (ReLU) as activation function (see Figure A.3). This architecture is chosen due to two reasons: Firstly, it showed good results on forecasting the M5 dataset in [14] and secondly it is a widely applicable model type which can also be used for classification, regression tasks and many more. As second architecture, a Long Short-Term Memory (LSTM) network is chosen, specifically designed for prediction purposes [18]. It uses one LSTM layer to capture time related dependency features and a fully connected layer to map the learned temporal features to the output (see Figure A.3). To enhance the amount of data, both architectures are used in a transfer learning approach: All available data is taken to pre-train both models and then finetune it to the individual prediction tasks. Both models are implemented using the deep learning framework PyTorch [19]. Information about the model parameters can be found in Table B.2.

2.4. Experiment Design

The experimental design for the proposed study consists of all permutations of the presented preprocessing steps, followed by the application of both deep learning architectures to perform the prediction. Here, the order of the individual steps is taken into account - first the transformation to a normal distribution, then the removal of the trend and last but not least the scaling. All methods are applied to the training set in order to avoid data leakage to the neural networks. Finally, the classic error metric Mean Squared Error (MSE) is used for evaluation. An illustration of the experimental design can be seen in Figure 1.

3. Results and Discussion

In this section, first results about the impact of the different time series preprocessing settings on the prediction performance are shown and discussed. All computations are performed in Python, using PyTorch [19] as deep learning framework and scikit-learn [17] for the error calculation and preprocessing methods. Plots are made using the visualization library Plotly [20]. As hardware, a system with a AMD EPYC 9745 as CPU, five NVIDIA L40S and two NVIDIA H200, and 1.5TB of RAM is used.

The permutation of all experiments is listed in Table 1. For all experiments, the MSE is calculated and an evaluation of which permutation of preprocessing methods performs best is done. In Figure 2, the results of the best performing permutations can be seen. Three experiments clearly stand out: 011, 029, 051. With over 200 times best performance and lowest MSE, they combined make up about $\sim$ 42% of the used data. The experiments 011 and 051 have the same preprocessing steps in common: Log as transformation and no detrending or scaling. Experiment 029 instead uses Box-Cox, difference as detrending and min-max scaling with [-1, 1] as feature range. When considering relatively good performing results (marked as yellow), it can be observed that some of them have the same settings as experiment 029: The experiments 028, 068 and 069 all use Box-Cox as transformation, difference as detrending and min-max scaling (with varying feature ranges). As third top performer, the combination of Yeo-Johnson as transformation, difference as detrending and min-max scaling with feature range [-1, 1] offers good results. This is interesting, since in contrast to images, the range of values for demand prediction is not bound to a maximum value. It is very likely

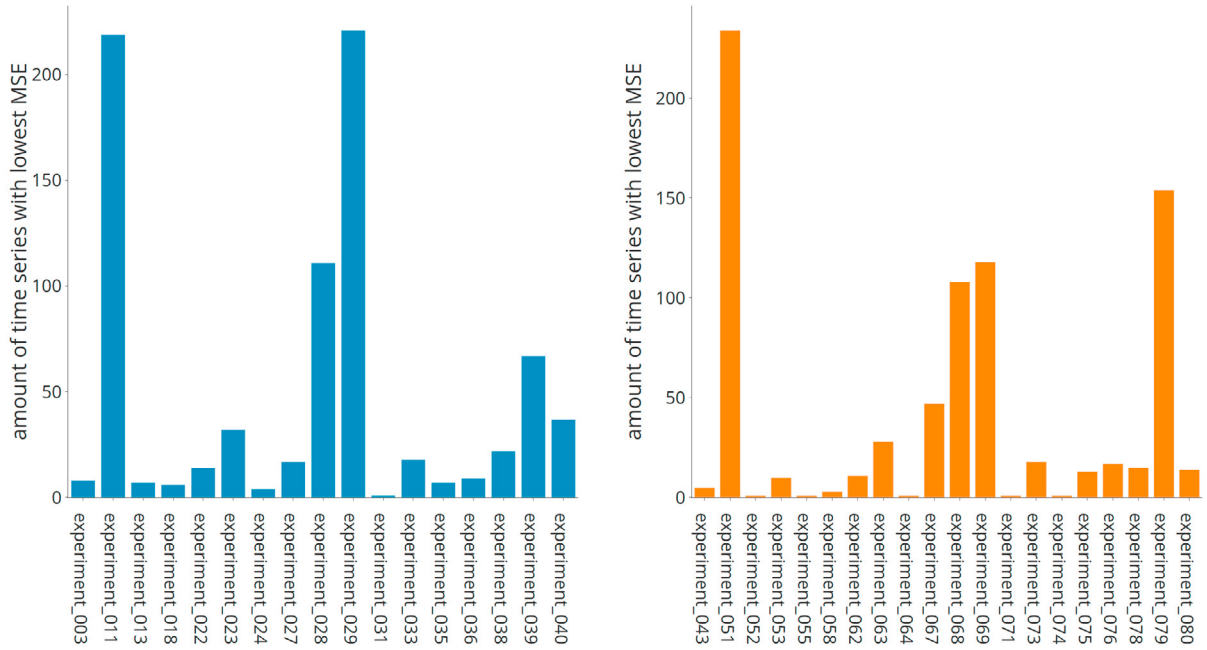

Fig. 2. Bar chart showing the frequency of experiments with the best performance. For each time series and model, the experiment with the lowest MSE is determined and its occurrence is then added up. The MLP results are shown on the left, and LSTM is shown on the right.

that demand can increase over time and exceed previous maxima, so that the forecasting algorithm has to deal with unknown values. At the same time, it is evident that neural networks are functioning better when the vectors are either normalized or standardized [21]. Therefore, a standardization algorithm would have been more expected as part of the best preprocessing permutations.

In the experiments with the lowest performance, no clear correlation with specific permutation settings can be established. One could say, that using Yeo-Johnson and Box-Cox as transformations resulted in the fewest non-occurring or worst experiments, or that using min-max scaling offers the highest chances of success when it comes to learning something, but compared to the experiments with the best performance, these indications are not clear.

Therefore, the first conclusion that can be drawn is that different algorithms for preparing time series undoubtedly have an enormous influence on the prediction results. Trends can be observed, that the use of a Log Transformation without further adjustments or a difference detrending and min-max scaling with either Box-Cox or Yeo-Johnson as the transformation, lead to significantly better results than other preprocessing combinations.

Considering only the best performing experiments per time series, with a calculated MSE of 0.171 for the MLP architecture and 0.190 for the LSTM network, it can be said that the correct preprocessing per time series reduces the prediction error significantly. A general recommendation about what algorithm to use best for each time series can not be drawn yet unfortunately, since a correlation between the standard statistics like average, median and standard deviation of the time series or more complex characteristics like the Syntetos-Boylan categorization in relation to the conducted experiments has not been detected.

4. Conclusion and Outlook

In this study, the influence of different preprocessing methods for time series forecasting is explored by conducting different experiments and comparing the MSE as well as observing the frequency of the best performing experiments. Two neural networks are chosen to predict the preprocessed data, which is the result of 80 different combinations of preprocessing algorithms, belonging to the groups of transformation, detrending and scaling procedures. With a

Table 1. Overview of all conducted experiments. The best performing experiments are highlighted in green, the lowest performers are colored in red. Yellow marked experiments are those, which are relatively good performing compared to the rest. Gray cells indicate, that the experiments did not occur in the list of the best performers.

MLP Experiments	LSTM Experiments	Transformation	Detrending	Scaling	Feature Range
001	041	none	none	none	
002	042	none	none	standardization	
003	043	none	none	minmax_scaling	(0, 1)
004	044	none	none	minmax_scaling	(-1, 1)
005	045	none	none	robust_scaling	
006	046	none	difference	none	
007	047	none	difference	standardization	
008	048	none	difference	minmax_scaling	(0, 1)
009	049	none	difference	minmax_scaling	(-1, 1)
010	050	none	difference	robust_scaling	
011	051	log	none	none	
012	052	log	none	standardization	
013	053	log	none	minmax_scaling	(0, 1)
014	054	log	none	minmax_scaling	(-1, 1)
015	055	log	none	robust_scaling	
016	056	log	difference	none	
017	057	log	difference	standardization	
018	058	log	difference	minmax_scaling	(0, 1)
019	059	log	difference	minmax_scaling	(-1, 1)
020	060	log	difference	robust_scaling	
021	061	boxcox	none	none	
022	062	boxcox	none	standardization	
023	063	boxcox	none	minmax_scaling	(0, 1)
024	064	boxcox	none	minmax_scaling	(-1, 1)
025	065	boxcox	none	robust_scaling	
026	066	boxcox	difference	none	
027	067	boxcox	difference	standardization	
028	068	boxcox	difference	minmax_scaling	(0, 1)
029	069	boxcox	difference	minmax_scaling	(-1, 1)
030	070	boxcox	difference	robust_scaling	
031	071	yeojohnson	none	none	
032	072	yeojohnson	none	standardization	
033	073	yeojohnson	none	minmax_scaling	(0, 1)
034	074	yeojohnson	none	minmax_scaling	(-1, 1)
035	075	yeojohnson	none	robust_scaling	
036	076	yeojohnson	difference	none	
037	077	yeojohnson	difference	standardization	
038	078	yeojohnson	difference	minmax_scaling	(0, 1)
039	079	yeojohnson	difference	minmax_scaling	(-1, 1)
040	080	yeojohnson	difference	robust_scaling	

strong indication, that Log Transformation with no further adjustments or a detrending and min-max scaling with either Box-Cox or Yeo-Johnson Transformation perform best, and the significant low occurence of other experiment permutations, it can be said, that there is a strong impact of the correct choice of preprocessing algorithms in terms of

the prediction accuracy. This study laid the foundation for a large-scale investigation of the various influencing factors and correlations in time series preprocessing and prediction.

Although first insights have been gained about the preprocessing procedures in the demand prediction context, there are still some limitations that have to be investigated. The study is limited to one dataset so far, whereas the field of time series forecasting has a huge variety of different forecasting tasks and data types. Other datasets, such as weather or production data, should be explored to verify findings and determine additional observations on the behavior of preprocessing methods in combination with prediction algorithms. Besides that, an extensive guide on how to preprocess which time series is still missing. As initial investigations into the correlation of time series statistics and the experiments carried out could not be determined, additional statistics and methods (such as time series statistic classification with experiments as label) should be considered. In future research, a classification method for time series will be examined, that automatically determines the appropriate preprocessing algorithms and subsequently applies them. Corresponding studies will start soon.

Appendices

Appendix A. Model Architectures

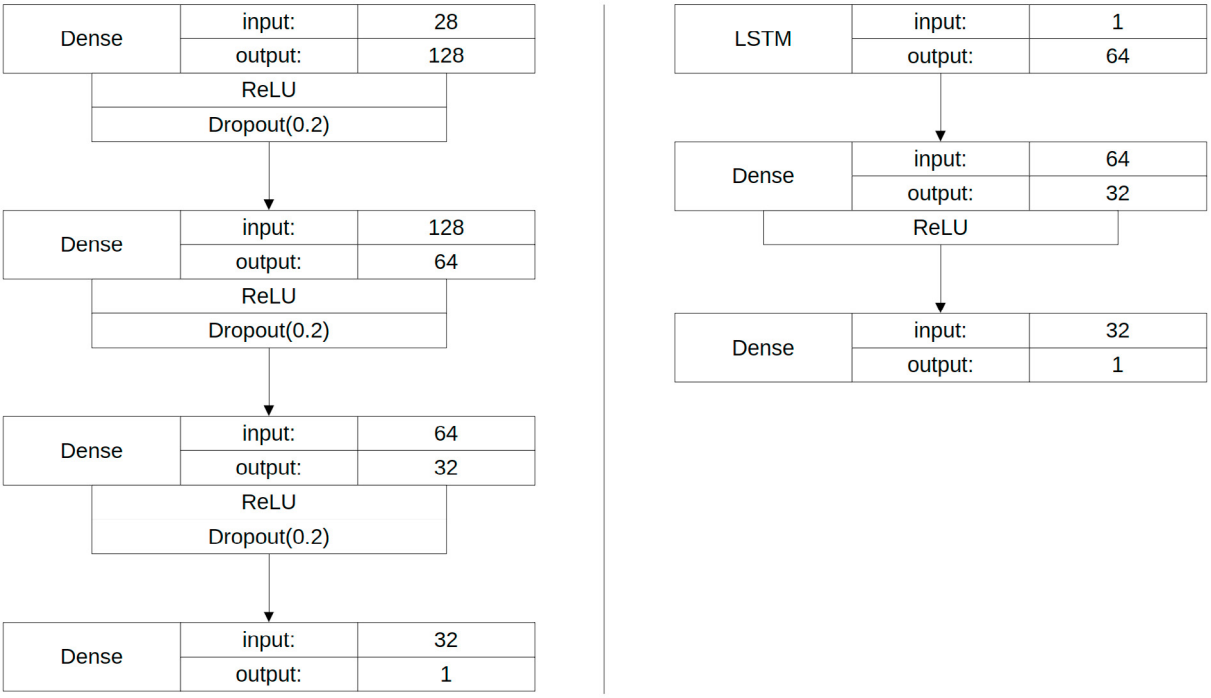

Fig. A.3. Architectures of the used Deep Learning networks. On the left, the MLP network can be seen, on the right, the LSTM network is depicted.

Appendix B. Model Parameters

In Table B.2, further information about the model training parameters are listed.

Table B.2. Details about the used model parameters for deep learning algorithm training.

parameters	value
optimizer	adam
learning rate	0.001
loss	MSE
activation functions	relu
dropout	0.2
validation split	0.2
batch size	32
epochs	300
early stopping	30
shuffle	false
random seed	42

References

[1] Nicole Gillespie, Steven Lockey, Tabi Ward, A Macdade, and G Hassed. Trust, attitudes and use of artificial intelligence. 2025.

[2] L Yee, M Chui, and R Roberts. Superagency in the workplace: Empowering people to unlock ai's full potential. mckinsey & company report, january 28, 2025.

[3] Nestor Maslej, Loredana Fattorini, Raymond Perrault, Yolanda Gil, Vanessa Parli, Njenga Kariuki, Emily Capstick, Anka Reuel, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Juan Carlos Niebles, Yoav Shoham, Russell Wald, Tobi Walsh, Armin Hamrah, Lapo Santarlasci, Julia Betts Lotufo, Alexandra Rome, Andrew Shi, and Sukrut Oak. Artificial intelligence index report 2025, 2025.

[4] M Bianchini and M Lasheras Sancho. Sme digitalisation for competitiveness: The 2025 oecd d4sme survey, 2025.

[5] ICONIQ Capital. 2025 state of ai report: The builder's playbook. https://www.iconiqcapital.com/growth/reports/2025-state-of-ai, 2025. Accessed: 2025-06-30.

[6] Everleen Wanyonyi and Newton Masinde. The impact of data preprocessing on machine learning model performance: A comprehensive examination. *Available at SSRN 5319101*, 2025.

[7] Kuntal Kumar Pal and KS Sudeep. Preprocessing for image classification by convolutional neural networks. In *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 1778–1781. IEEE, 2016.

[8] Puneet Kumar and Dalwinder Singh. Pre-processing techniques for enhancing cnn performance. In *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pages 1–5. IEEE, 2024.

[9] Pegah Dehbozorgi, Oleg Ryabchykov, and Thomas Bocklitz. A systematic investigation of image pre-processing on image classification. *IEEE Access*, 12:64913–64926, 2024.

[10] Kelsy Cabello-Solorzano, Isabela Ortigosa de Araujo, Marco Peña, Luís Correia, and Antonio J. Tallón-Ballesteros. The impact of data normalization on the accuracy of machine learning algorithms: a comparative analysis. In *International conference on soft computing models in industrial and environmental applications*, pages 344–353. Springer, 2023.

[11] SC Nayak, Bijan B Misra, and Himansu Sekhar Behera. Impact of data normalization on stock index forecasting. *International Journal of Computer Information Systems and Industrial Management Applications*, 6:13–13, 2014.

[12] Samit Bhanja and Abhishek Das. Impact of data normalization on deep neural network for time series forecasting. *arXiv preprint arXiv:1812.05519*, 2018.

[13] Felipe Tomazelli Lima and Vinicius MA Souza. A large comparison of normalization methods on time series. *Big Data Research*, 34:100407, 2023.

[14] Florian Grimm, Clemens van Dinther, Tim Straub, and Günter Bitsch. On the relevance of demand pattern categorization. In *Proceedings of the 58th Hawai'i International Conference on System Sciences (HICSS): 7-10 January 2025, Hawai'i*, pages 1473–1481. University of Hawai'i at Manoa, 2025.

[15] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m5 competition: Background, organization, and implementation. *International Journal of Forecasting*, 38(4):1325–1336, 2022.

[16] Aris A Syntetos, John E Boylan, and JD Croston. On the categorization of demand patterns. *Journal of the operational research society*, 56(5):495–503, 2005.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] A Paszke. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[20] Plotly Technologies Inc. Collaborative data science, 2015.

[21] Jorge Sola and Joaquin Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on nuclear science*, 44(3):1464–1468, 1997.