Customer behaviour forecast using deep learning networks

Agris Nikitenko
Department of Artificial Intelligence and System Engineering, Riga Technical University, Riga, Latvia
Agris.Nikitenko@rtu.lv

Ilze Andersone
Department of Artificial Intelligence and System Engineering , Riga Technical University, Riga, Latvia
Ilze.Andersone@rtu.lv

Andrejs Zujevs
Department of Artificial Intelligence and System Engineering , Riga Technical University, Riga, Latvia
Andrejs.Zujevs@rtu.lv

Elina Gaile-Sarkane
Faculty of Engineering Economics and Management , Riga Technical University, Riga, Latvia
Elina.Gaile-Sarkane@rtu.lv

Valdis Bergs
Administrative department, SIA Mobilly, Riga, Latvia
Valdis.Bergs@mobilly.lv

ABSTRACT

The paper presents a case study on customer behaviour forecast within a parking products domain. In particular, authors compare different LSTM-based DL networks with and without data preprocessing to decide which of the selected architectures and hyperparameter combinations provide the least square error estimate. Unfortunately, well-known forecast methods like regression and ARIMA did not deliver the needs forecast reliability, which brought the authors to the application of DL models. While the paper does not offer novel DL models or architectures, it provides a convenient application insight enabling a better understanding of the application of the selected models. Data has been collected for several months and offers a good study backbone.

CCS CONCEPTS

• **Machine learning**;

KEYWORDS

Customer behaviour forecast, time series, LSTM, Deep learning networks

ACM Reference Format:
Agris Nikitenko, Ilze Andersone, Andrejs Zujevs, Elina Gaile-Sarkane, and Valdis Bergs. 2022. Customer behaviour forecast using deep learning networks. In *2022 7th International Conference on Machine Learning Technologies (ICMLT) (ICMLT 2022), March 11–13, 2022, Rome, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3529399.3529431

1 INTRODUCTION

For decades, consumption has been based on consumer choice theory, which creates a fundamental of microeconomics and is related to consumer preferences. Consequently, consumer behaviour studies emerged like a discipline intending to study the behaviour of individuals and groups and organizations. For more than 70 years, consumer behavioural science has been developing in different directions, becoming more inter- and cross-disciplinary by blending up economics and phycology, sociology, marketing, and now overlapping with natural sciences like mathematics, computer science, etc. An individual will consider all possible choices by rationally evaluating benefits and costs and assigning weight to different attributes depending on their importance. The individual then selects the best possible choice based upon available information, costs, benefits and the probability of possible risks. [1] [2] [3] This research is based on a holistic approach to the study of consumer behaviour, focusing more on the nature of consumption experience than on purchasing [4] and the Engel Kollat Blackwell Model (EKB) of Consumer Behaviour, which is one of most examined consumer utility models.

EKB is based upon prior work in educational philosophy by John Dewey (1910/1978) and proposes a sequential process of decision making consisting of 1) problem recognition, 2) information search, 3) evaluation of alternatives, 4) purchase, and 5) post-purchase evaluation [5] [6] . According to EKB, consumer behaviour is based on events where information plays the dominant role in decision making. To follow and verify the EKB main statements, within our effort, we examined several data sets of the selected company of its customer behaviour in a context of particular product use.

Customer behaviour forecasts and analysis are among the most interesting business application prospectives, enabling building marketing and product strategies. However, since, in most cases, a particular customer's behaviour is expressed as a sequence of events – time series, it brings a certain complexity and uncertainty into the analysis, with appropriate implications on the model of the behaviour.

Time series in today's data analytics appear in very different contexts and applications, for instance, medical data analysis for disease evolvement forecast [7] [8] [9], sensor data measurement forecast to reduce sampling rate and, as a consequence, reduce power consumption or other control actions [10] [11] [12], workload forecast for single and multi-server systems for proper load balancing policies [13] [14] [15], financial market dynamics forecast [16] [17] [18] [19].

In many cases, well-understood approaches like autoregressive moving average (ARIMA) [20] or its extension for seasonal data SARIMA [21] are applied [10] [14] [15], which sometimes are hybridized with other methods like deep learning networks, providing

ICMLT 2022, March 11–13, 2022, Rome, Italy

ACM ISBN 978-1-4503-9574-8/22/03.
https://doi.org/10.1145/3529399.3529431

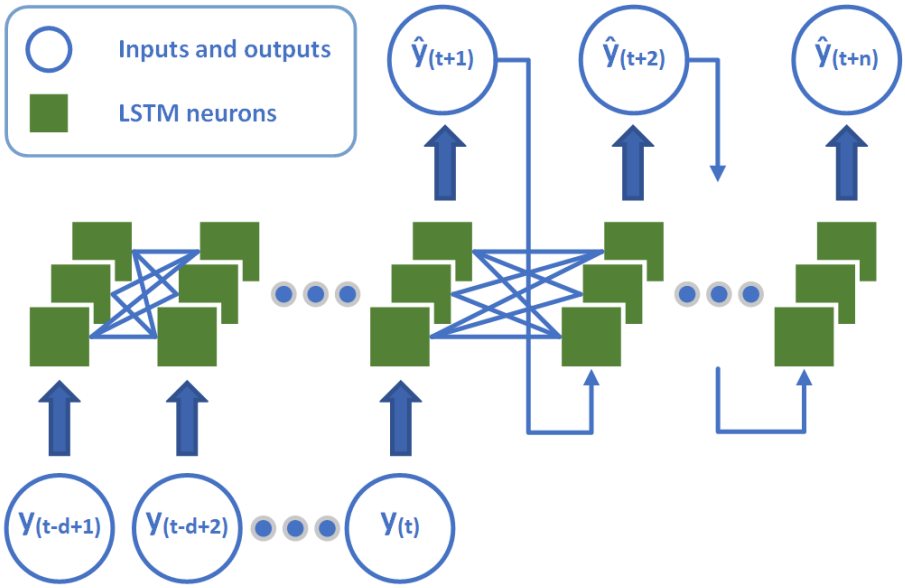

Figure 1: Schema of a single-layer LSTM network

higher performance in terms of prediction precision and higher resistance to outlier influence [16]. Deep learning is among the most used in the context time series forecast – as the backbone or supplementary method in different configurations. Due to the application versatility and flexibility of the deep learning paradigm, there are attempts to use even image processing methods for time series forecast transforming time series plots into multi-channel input images [18], enabling to capture patterns that are hard to model numerically or analytically. In some cases, for better performance, methods like LSTM (long short term memory) are combined with other deep learning network architectures as stacked layers, like autoencoders [19], enabling to capture of some bold features of the time series. While attempts to compare deep learning methods with other methods like ARIMA for a general case [22], each technique still has its drawbacks and strengths, enabling many flexibilities for different applications. However, it does not always provide a positive contribution since it is unclear which method is better to use. It is necessary to note that ARIMA did not provide the needed accuracy on the training data set.

2 MOTIVATION AND GOAL

According to recent research reports, one of the most promising and widely used approaches is to use deep learning networks (DLN) and recurrent neural networks (RNN) in particular, which in the case of LSTM (long short term memory) provides promising results and applications potential [23]. According to comparative analysis of different neural network architectures, including feed-forward and recurrent architectures, under conditions of having enough data for training, LSTM architecture provides higher accuracy and higher robustness under very challenging forecast conditions [24] [25]. In addition, LSTM allowed solving vanishing and exploding gradient problems that are still issues in feed-forward recurrent neural networks [26].

Having a time series Y consisting of time steps y(t), the problem of forecasting n steps ahead (forecast horizon) of the current one requires to find a predictor that accepts as an input of d last samples or lags:

$$Input = \{y_{t-d+1},\ y_{t-d+2}, \cdot s, y_{t-1}, y_t\} \tag{1}$$

The output of the predictor is a vector of length n:

$$Output\ pred. = \{\hat{y}_{t+1}, \hat{y}_{t+2}, \cdot s, \hat{y}_{t+n}\} \tag{2}$$

The output corresponds to actual observations of Y:

$$Output = \{y_{t+1},\ y_{t+2}, \cdot s, y_{t+n}\} \tag{3}$$

The forecast accuracy depends on input length, and it must be long enough. A feasible d is said to allow embedding the dataset in nonlinear time series analysis, and the minimum d is called the dataset embedding dimension. A good practice is selecting d at least twice the length of n, enabling to model of even very complex time series [24]. However, if the d is too large, the prediction accuracy worsens, implying that the length is one of the main parameters to be determined. The main advantage of the LSTMs over other types of the RNNs is its essence allows them to explicitly take into account the sequence of events (samples), manage error backflow using dedicated so-called gates (input, output, and forget) and provide enriched status representation by using two different internal states – hidden and cell state [24] [27] [28].

Thereby instead of using regular feed-forward neurons, LSTM uses LSTM cells (see Fig. 1). The cell state is responsible for maintaining the needed information from the past inputs – long term memory, while the hidden state combines the cell state, input and the previous hidden state. The gates control the amount of information taken into account for the update of the cell state – input and forget gates, and the amount of information passed to the output – the output gate [24] [28].

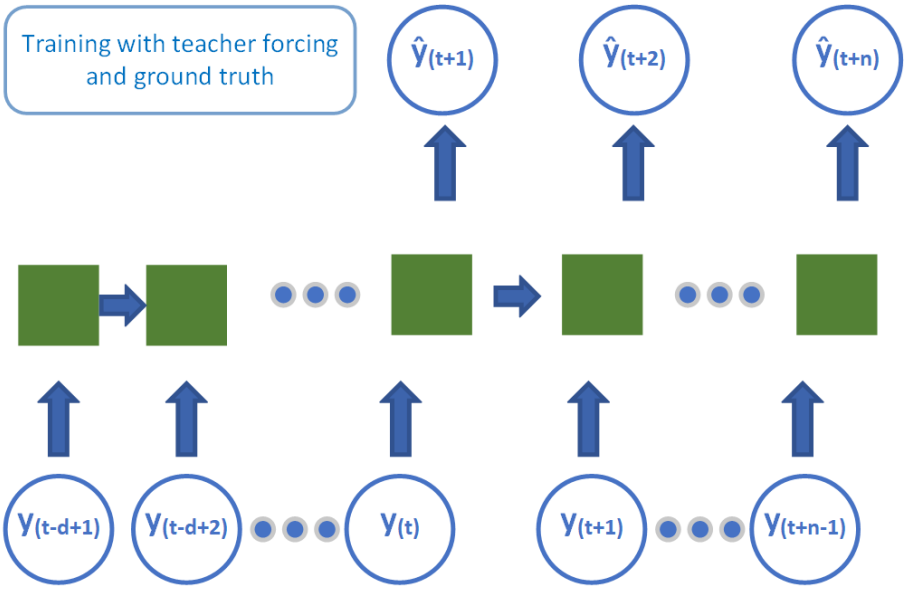

Figure 2: LSTM training with teacher forcing

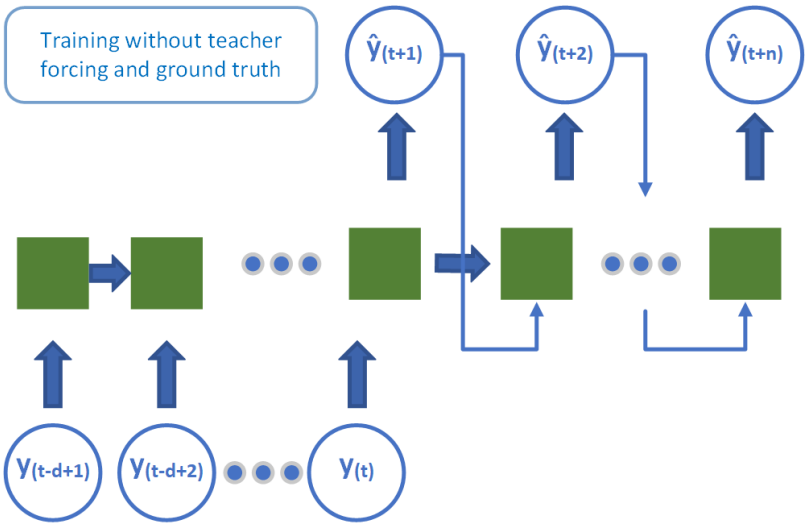

Figure 3: LSTM training without teacher forcing

The gates control the amount of information taken into account for updating the cell state – input and forget gates- and the amount of data passed to the output gate [24] [28].

As with other problem domains, it is possible to use both training methods with teacher forcing [29] or without teacher forcing in time series analysis. The teacher forcing has the advantage to provide a ground truth of length n, which allows a direct error estimation between output and ground truth (Fig. 2). In training, without the teacher forcing, the output is fed back to the network as the unknown input of the next step (Fig. 3)

Depending on the inference settings, it might be necessary to predict a single future state and propagate the forecast like a sliding average time window be feeding back the output as the unknown inputs of future states. However, it is possible to predict a whole vector of the future state of the length n.

In the case of a single future state prediction, the inference introduces a difference between training and inference steps, which leads to error accumulation. This is because during the training with the teacher forcing predictions do not affect other future predictions – ground truth allows considering them. At the same time, in inference with a single output step, the earlier forecast provides a significant contribution to the error of the later ones [24] [30]. Therefore it is necessary to consider and align training and inference phases.

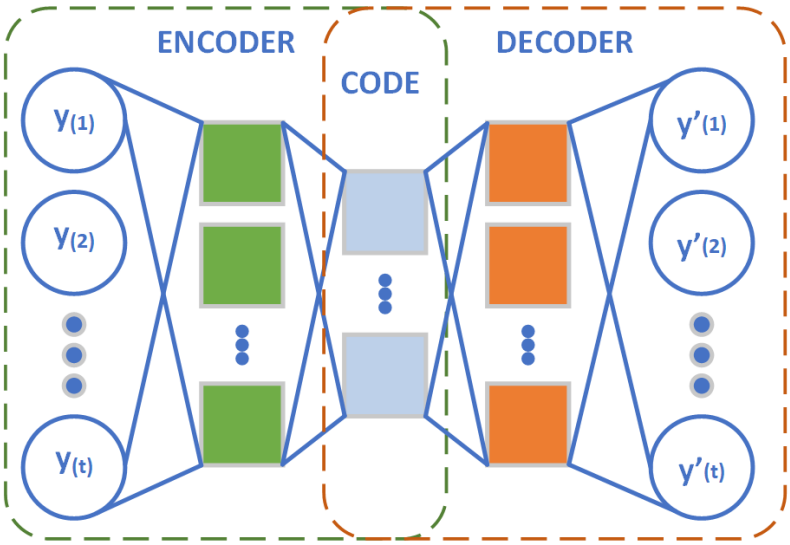

Figure 4: Two-layer autoencoder network

Additional layers are used in highly challenging domains where it is necessary to extract some complex or hidden behaviour of the input sample like asset health monitoring of failure prognostics, besides one or several LSTM layers [31]. To learn abstract unlabeled features of a given time series, the input sample is passed to the Restricted Boltzmann Machine (RBM) layer, which intensifies potentially significant regions of the input before it is passed to supervised LSTM layers [31] [32]. The output of the LSTM layers (one or several) is fed in a fully connected feed-forward layer, which enables to map, classify or do other final tasks before the output. Instead of an unsupervised RBM layer, a supervised convolutional neuron network (CNN) layer might be used to provide the power to extract significant features from the raw input, which are passed then to LSTM layers [33]. In the context of our study, only a one-dimensional convolution layer is needed, which performs a feature mapping applying the convolution operation:

$$o(t) = (y * k)(t) = \sum_{h} \left(\; y_{(t-h)} \quad k_h \; \right) \tag{4}$$

where
$y_{(t)}$ - input vector of the time series, with time step t;
k – kernel function of a length h;
o(t) – the output or feature map;
While the CNN layer extracts low-level input features, LSTM layers extract more complex ones [33]. Besides CNNs to extract relevant features, auto-encoders (Fig. 4) are another option to boost feature extraction [34] [35].

Autoencoder (AE) network maps inputs to their output, which usually are of the same size, enabling learning effective coding of the input data [36]. AE is a feed-forward network with fully connected layers in its most straightforward implementation. Encoder layers reduce the dimensionality of the input vector, producing as its output the code. Thereby encoder enables to learning of valuable properties of the input data. The decoder part transforms the code into the output by adding dimensions. Thus, AE is trained to keep useful features and omit the unnecessary ones. However, AE can learn the input or minor features like noise [37]. To diminish the mentioned risks of capable AE, a designer could change the hidden layer and code size to be smaller than the input or output size (see Fig. 4) – undercomplete AE, or use other properties like sparsity, the smallness of the derivative of the representation and robustness to missing values and noise [37]. Some recent research reports combine CNNs and AEs to build convolutional autoencoders CAEs, which Rumerhalt et al. [38] proposed. Later much larger attention was pulled by Vincent et al., which showed CAE's ability to cope with the noisy data [39] [40]. The main difference between AE and CAE is the convolutional function as a signal transfer method from layer to layer, providing higher extraction effectiveness over the regular feed-forward approach.

CAEs have another advantage over AE since they have a good performance on complex time series like ECG data enabling them to extract essential features effectively [41]. Despite extensive research efforts provided by the scientific community, selecting particular network architecture for the specific problem is difficult, depending on seasonality, level of noise, length of the time series, and the particular architecture of the model being used. Therefore our goal is to provide an insight into a specific forecast problem through applying different model architectures and different data set pro-processing.

3 OUR APPROACH AND BENCHMARK CRITERIA

Despite extensive research efforts provided by the scientific community, selecting particular network architecture for the specific problem is a difficult task, which depends on seasonality, level of noise, length of the time series and particular architecture of the model being used. Therefore our goal is to provide an insight into a specific problem of the forecast through applying different model architectures and different data set preprocessing applied.

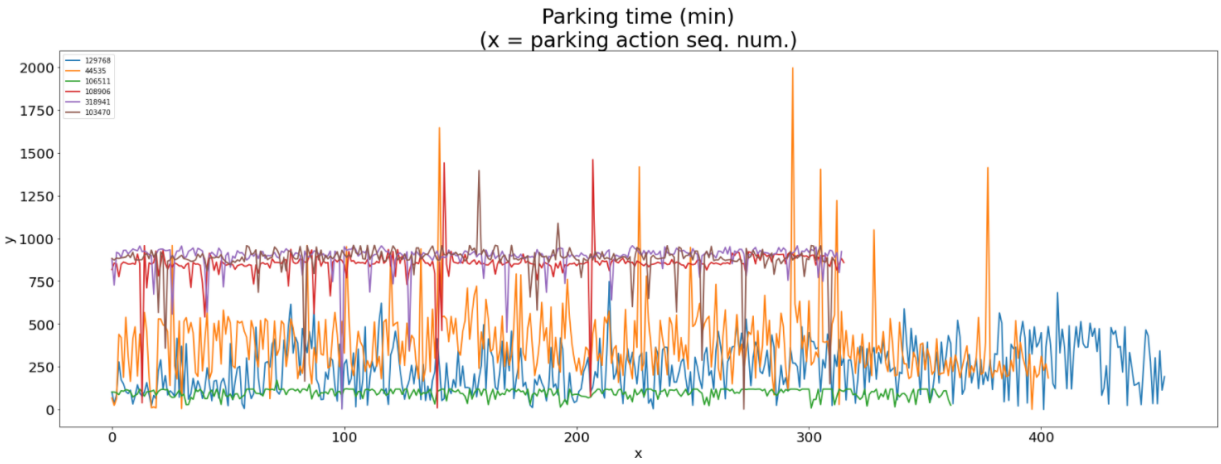

Figure 5: Selected customer behaviour. X – parking sequence number, Y – Parking time in minutes

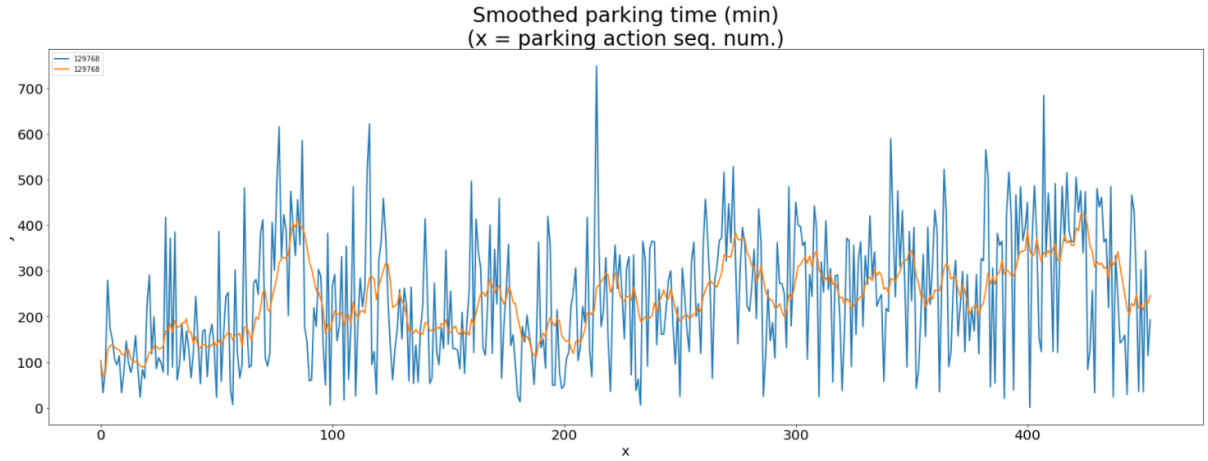

Figure 6: Smoothed data (orange) of one of the selected customers using moving average

3.1 The dataset

The initial data set consists of approximately 200K customer behaviour logs, representing parking time, i.e. daily customer behaviour and should reflect regularities if any. We intentionally selected six customers, where three represent similar behaviour while the remaining three represent a different one (see Fig. 5). Each customer is marked with an anonymous identifier, and their behaviour data is collected within the same season, making them relatively comparable. We also used smoothed time series of the same data for comparison reasons of different models (see Fig. 6).

We took into account different preferences of the customers in terms of parking time interval and how often customers select a particular parking time, which might be seen in the following value distribution plots (See Fig. 7 and Fig. 8).

As shown in figure 5, figure 6, figures 7 and 8, the value and frequency distribution are somewhat different, allowing for examining different forecast models for different behaviour. Following the smoothed time series in Fig.6, one can notice that certain seasonality is reflected in the time series, which provides a ground to believe that effective methods on seasonal data might be practical here.

We also tried to use downsampled datasets, but unfortunately, the forecast models results were feeble and did not provide any meaningful contribution to our study goals. The main reason was a significant drop in quality of reflecting the actual behaviour of the customers and loss of dataset length, which is important for proper training of the model.

3.2 Used models

Our main intention was to compare different deep learning network architectures that were proven effective and reported by other referenced studies. As a core architecture, we used LSTMs, which form the backbone of the models. According to the references studies [24] [26] [27], both single or multiple layer LSTMs are effective predictors enabling to reveal of complex regularities.

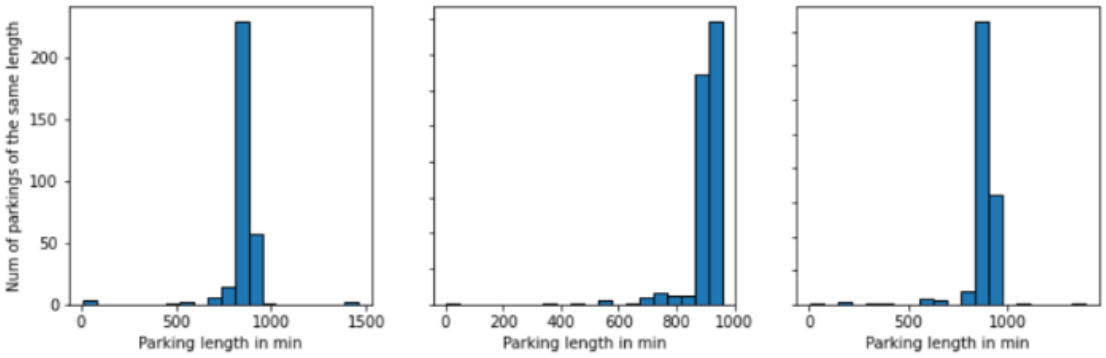

Figure 7: Customers of similar behaviour (frequency of parking time in minutes)

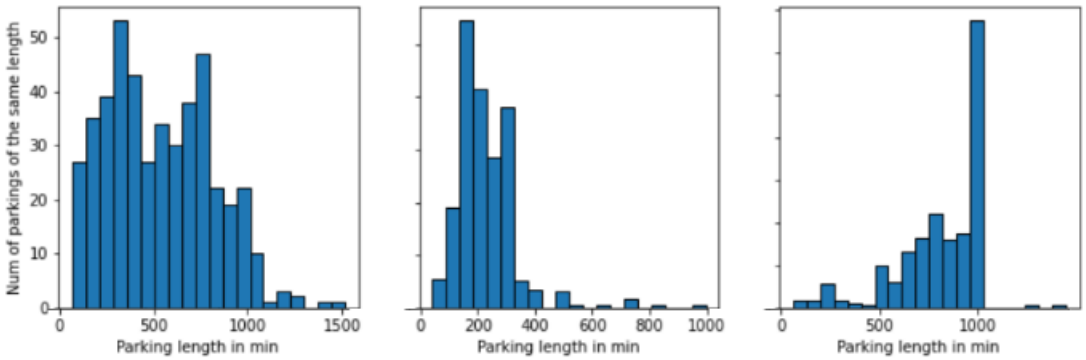

Figure 8: Customers of different behaviour (frequency of parking time in minutes)

Table 1: Configuration parameters

Parameter	Parameter values and comments
Forecast horizon	Three steps represent three coming days, enough to provide additional value for the customer through tailored offers or discounts.
Input sample size	Fourteen steps, which represent two whole weeks.
Hidden layer size	30 and 100 neurons for all architectures
Output	Entire horizon (3 steps) or propagated using a single output for all architectures. The horizon value is selected to provide ground for tailored product offers (discounts etc..). Therefore the horizon is relatively short.
Loss function	Mean Square Error (MSE) for all architectures
Optimizer	ADAM [42] for all architectures
Learning ratio (LR)	Set empirically to 0.0005, which provided the best results during the experiments. The experimental setup was tested on values within the interval of [0.001; 0.0001]. The selected value provided an efficient balance between training speed and forecast error.
Training data	Random slices of input (14 steps) and output (3 steps) pairs were shuffled during the training. Each sample represents a specific behaviour of a given customer for the last two weeks.
Epochs	We used 3000 epochs, which was selected empirically following the loss dynamics.

To follow the trends of recent studies [39] [40] [41], we tested the effects of additional layers of AE and CAE at the input stage of the network. The additional configuration parameters include the size of hidden layer 30 or 100 neurons and the way how the input is produced: providing the output vector of the needed forecast horizon at once or providing a single output and propagating it to reach the required horizon by feeding the output back as a part of the input of the next step. Summary of the architectures and configuration parameters are listed below in Table 1

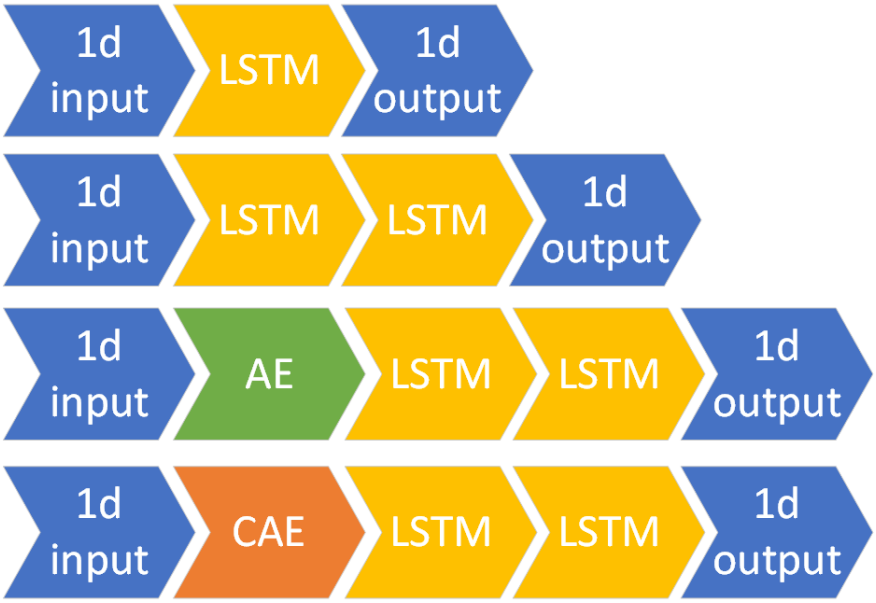

Figure 9: Different architectures of input stages

As depicted in Fig.9, additional input stage layers were used only in conjunction with a two-layer LSTM backbone. This is because we noticed a significant drop in the forecast error when using a two-layer LSTM backbone. We do not report CAE results for the opposite reasons since the forecast errors of different parameter configurations were significantly worse than other alternatives. Therefore this particular architecture is not reported since they do not provide a meaningful value for further study.

The hidden layer size was selected from a range of 30 – 300, where we did not observe significant differences between 100 and 300. Therefore we settled on the two sizes to show the impact of the parameter on the forecast error. Since our study does not intend to provide the optimal set of parameters but tendencies instead, the selected layer size values are suitable for further optimization purposes if needed.

We selected the ADAM optimizer because, as experimentally shown in [43], the ADAM optimizer not necessarily on all occasions is better than other alternatives like Stochastic gradient descent or Root mean square propagation [44] [45]. However, the method is as good as the other alternatives under various parameter settings [43].

For implementation purposes, we used Pytorch [45], which provided a relatively easy way to control the architecture on the one hand and enough abstraction of training complexity on the other hand.

3.3 Benchmark

We used randomly selected time slices of inputs (14 steps) and outputs (3 steps) for model training purposes, which were shuffled during the training. Due to differences in customer behaviour, each model was trained separately on every customer's data. Thereby it was possible to follow to what extent the models find the regularities of the customer behaviours.

As the main comparison criterion, we selected the mean square error (MSE) between the forecast of the entire horizon and the observed customer behaviour. The value of the MSE provides an easy to interpret estimate.

However, since the models are tested and trained on several customers' mean of separate MSEs, their standard deviation was monitored to see to what extent the model provides relatively similar results for all selected customers. This estimate is indicative and includes information on have general the models might be for different customers.

4 RESULTS

The collected results are provided in tables II and III, where the best results are bolded. We used a relatively long training time – 3000 epochs. We picked the best-found result, which provided a rather good ground to conclude the performance of particular architectures and parameter sets. In the case of goth data sets, raw and smoothed slightly better results are for two-layer LSTM with hidden layer size 100, while in some cases, an AE-LSTM architecture provides the best results, while in others is the next best architecture. During the experiments, a single layer LSTM has never offered the best result and, in most cases, is by several orders of magnitude behind the two-layer case. Another interesting finding is full forecast horizon v.s. the propagated one, where significantly better performs the propagated output architecture.

From the figure Fig. 10, it might be noticed that models on raw data sets perform slightly better than on smoothed datasets. Standard deviation also drops accordingly, but that is related to the drop of the absolute values. If the standard deviation relative to the mean value is examined, which shows the relative amount of

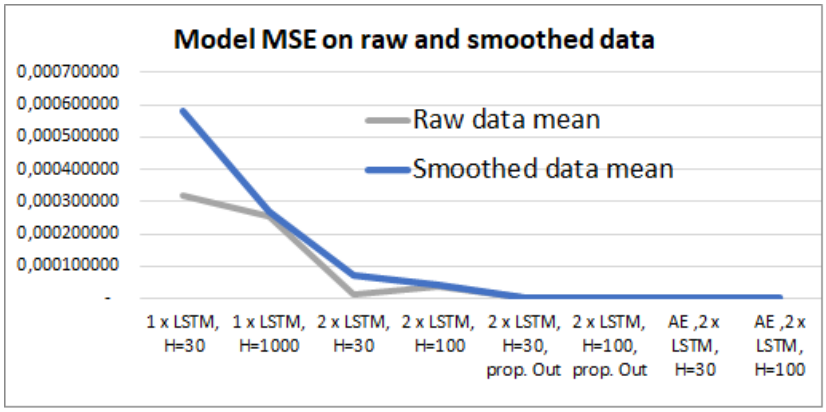

Figure 10: MSE plot of different models for raw and smoothed data

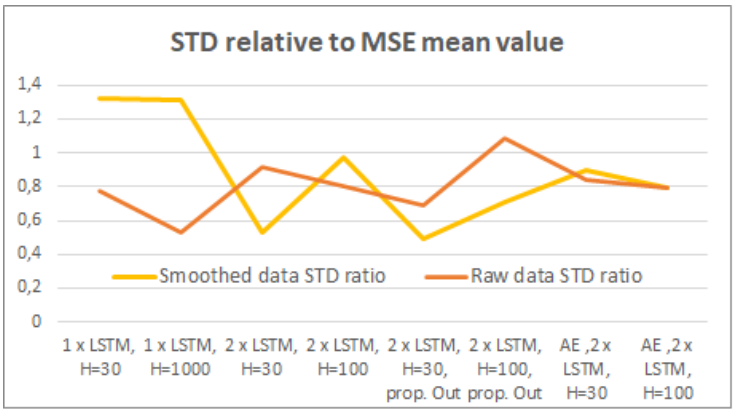

Figure 11: STD relative to MSE mean values

Table 2: Results Details on Smoothed Data Sets

Layers	Single hidden layer		Two hidden layers		Two hidden layers - prop. Output		Two hidden layers - AE input	
Cust. #	H = 30	H = 100	H = 30	H = 100	H = 30	H = 100	H = 30	H = 100
44535	0.00106222	0.000168498	0.000000106	0.0000000645	0.000000029	0.000000017	0.0000000399	0.000000031
129768	0.00127532	0.00043749	0.000000927	0.0000004103	0.000000362	0.000000013	0.0000002706	0.000000144
106511	0.00051815	0.00063149	0.000001285	0.0001015138	0.000024514	0.000000015	0.0000021718	0.000001193
108906	0.00037099	0.000195994	0.000047179	0.0000000182	0.000000308	0.000002291	0.0000000819	0.000004047
318941	0.00014588	0.000018759	0.000005478	0.0000747771	0.000000018	0.000000098	0.0000033345	0.000001139
103470	0.00011918	0.000155712	0.000366080	0.0000687687	0.000001277	0.000001299	0.0000007273	0.000000132
mean	0.000581957	0.000267990	0.000070176	0.000040925	0.000004418	0.000000622	0.0000011043	0.000001114
std	0.000440513	0.000204396	0.000133370	0.000041985	0.000008997	0.000000878	0.0000012335	0.000001396

distribution, then a clear trend is not evident (see Fig. 11). Collected particular accuracy on smoothed data, raw data and other statistics are represented in tables Table 2 un Table 3

5 CONCLUSIONS AND FUTURE WORK

While the customer-wise MSE varies from model to model, the overall trend is rather apparent and suggests that a model with a single layer LSTM backbone provides a significantly lower forecast precision in comparison with a two-layer LSTM backbone.

Our study shows that a propagated output outperforms an entire horizon forecast approach, which we relate to day-by-day dependencies of the customer behaviour, meaning that the particular days parking needs are more related to previous day experience than on

Table 3: Results Details on Raw Data Sets

Layers	Single hidden layer		Two hidden layers		Two hidden layers - prop. Output		Two hidden layers - AE input	
Cust. #	H = 30	H = 100	H = 30	H = 100	H = 30	H = 100	H = 30	H = 100
44535	0.000000068	0.000002084	0.000001585	0.0000001196	0.0000006628	0.000000076	0.000000160	0.0000000660
129768	0.000985636	0.000016715	0.000003156	0.0000004011	0.0000002312	0.000000314	0.000000276	0.0000003865
106511	0.000799875	0.000143428	0.000006044	0.0001277570	0.0000000663	0.000000055	0.000001062	0.0000000610
108906	0.000041584	0.000015891	0.000031567	0.0000268392	0.0000015081	0.000000031	0.000002665	0.0000000545
318941	0.000067547	0.000032035	0.000001276	0.0000001183	0.0000132396	0.000000271	0.000008510	0.0000008256
103470	0.000016435	0.001324469	0.000033323	0.0000808191	0.0000033857	0.000000023	0.000001724	0.0000000032
mean	0.000318523	0.000255770	0.000012825	0.000039342	0.000003182	0.000000128	0.000002400	0.0000002328
std	0.000410102	0.000480250	0.000013968	0.000048797	0.000004632	0.000000118	0.000002863	0.0000002935

the entire last week. Unfortunately, this thesis is not directly supported by our study and, therefore, should be studied more closely in future efforts.

Our study also provides evidence that more complicated architectures like CAE-LSTM or AE-LSTM do not add significant value to MSE reduction for a particular task. However, due to the considerable limitations of our study, we have to be careful not to overgeneralize our results.

The most important caveat is our intentional distance from architecture optimization and hyperparameter tuning, favouring possible alternatives and guiding potential developers instead of optimizing one or a few viable options to get the highest possible output.

On average raw dataset provides somewhat better results over the smoothed data set, which suggests that using the raw dataset allows considering some internal causalities that are not obvious.

Having the results of the study, we could suggest the following: appliers should use at least double layer LSTMS and run them on raw data instead of heavily preprocessed with smoothing or downsampling. If there is space for parameter and architecture optimization, appliers should consider using more complex architectures with preprocessing stages to extract significant features of the input datasets. We also suggest using hidden layers of considerably large size instead of preserving smaller ones and losing forecast precision.

The overall results provide enough evidence to conclude that our initial goal of verifying EKB statements has proven true for the selected product applying modern forecast methods of deep learning, which corresponds to the current practice of behaviour forecast problems.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the project "Competence Centre of Information and Communication Technologies" of EU Structural funds, contract No. 1.2.1.1/18/A/003 signed between IT Competence Centre and Central Finance and Contracting Agency, Research No. 1.3 “Research, development prototyping of financial analysis tool based on document management system”.

REFERENCES

[1] Bettman, J. R., Luce, M. F., & Payne, J. W. (1998). Constructive consumer choice processes. Journal of Consumer Research, 25(3), 187-217.

[2] Monroe, K. R., & Maher, K. H. (1995). Psychology and rational actor theory. Political Psychology, 16(1), 1-21.

[3] Holland, J. (2019). Navigating uncertainty: Tourists’ perceptions of risk in ocean cruising [online]. Research Gate Web site [accessed 13 march 2021]. Available at: https://www.researchgate.net/figure/The-Engel-Kollat-Blackwell-complete-model-ofconsumer-decision-making-Engel-et-al-1968_fig1_339513025

[4] Abey F., 2021, Approaches to Studying Consumer Behaviour [online]. MBA Knowledge Base [accessed 17 July, 2021]. Available at: https://www.mbaknol.com/marketing-management/approaches-to-studying-consumer-behaviour

[5] Engel, J.F., Blackwell, R.D., & Miniard, P.W. (1995). Consumer Behaviour, 8th edition. Fort Worth, TX: The Dryden Press Harcourt Brace College Publishers

[6] Ashman R., Solomon M. R., Wolny J., An old model for a new age: Consumer decision making in participatory digital culture, 2015. Journal of Customer Behaviour 14(2):127-146, DOI: 10.1362/147539215X14373846805743Available from: https://www.researchgate.net/publication/282350425_An_old_model_for_a_new_age_Consumer_decision_making_in_participatory_digital_cultureIn the appendix section, three levels of Appendix headings are available.

[7] Y. Matsubara, Y. Sakurai, W. G. van Panhuis, and C. Faloutsos, "FUNNEL," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug. 2014, pp. 105–114, doi: 10.1145/2623330.2623624.

[8] E. Yang *et al.*, "A Simulation-Based Study on the Comparison of Statistical and Time Series Forecasting Methods for Early Detection of Infectious Disease Outbreaks," *Int. J. Environ. Res. Public Health*, vol. 15, no. 5, p. 966, May 2018, doi: 10.3390/ijerph15050966.

[9] M. Maleki, M. R. Mahmoudi, D. Wraith, and K.-H. Pho, "Time series modelling to forecast the confirmed and recovered cases of COVID-19," *Travel Med. Infect. Dis.*, vol. 37, p. 101742, Sep. 2020, doi: 10.1016/j.tmaid.2020.101742.

[10] S. Bhandari, N. Bergmann, R. Jurdak, and B. Kusy, "Time Series Data Analysis of Wireless Sensor Network Measurements of Temperature," *Sensors*, vol. 17, no. 6, p. 1221, May 2017, doi: 10.3390/s17061221.

[11] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "Geoman: Multi-level attention networks for geo-sensory time series prediction," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2018-July, pp. 3428–3434, 2018, doi: 10.24963/ijcai.2018/476.

[12] S. Papadimitriou and P. Yu, "Optimal multi-scale patterns in time series streams," *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 647–658, 2006, doi: 10.1145/1142473.1142545.

[13] L. Ruan, Y. Bai, S. Li, S. He, and L. Xiao, "Workload time series prediction in storage systems: a deep learning based approach," *Cluster Comput.*, 2021, doi: 10.1007/s10586-020-03214-y.

[14] J. Kumar and A. K. Singh, "Performance Assessment of Time Series Forecasting Models for Cloud Datacenter Networks’ Workload Prediction," *Wirel. Pers. Commun.*, vol. 116, no. 3, pp. 1949–1969, 2021, doi: 10.1007/s11277-020-07773-6.

[15] A. S. Higginson, M. Dediu, O. Arsene, N. W. Paton, and S. M. Embury, "Database Workload Capacity Planning using Time Series Analysis and Machine Learning," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, Jun. 2020, pp. 769–783, doi: 10.1145/3318464.3386140.

[16] A. H. Bukhari, M. A. Z. Raja, M. Sulaiman, S. Islam, M. Shoaib, and P. Kumam, "Fractional neuro-sequential ARFIMA-LSTM for financial market forecasting," *IEEE Access*, vol. 8, pp. 71326–71338, 2020, doi: 10.1109/ACCESS.2020.2985763.

[17] J. Cao, Z. Li, and J. Li, "Financial time series forecasting model based on CEEMDAN and LSTM," *Phys. A Stat. Mech. its Appl.*, vol. 519, pp. 127–139, Apr. 2019, doi: 10.1016/j.physa.2018.11.061.

[18] S. Barra, S. M. Carta, A. Corriga, A. S. Podda, and D. R. Recupero, "Deep learning and time series-to-image encoding for financial forecasting," *IEEE/CAA J. Autom.*

Sin., vol. 7, no. 3, pp. 683–692, May 2020, doi: 10.1109/JAS.2020.1003132.

[19] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLoS One*, vol. 12, no. 7, p. e0180944, Jul. 2017, doi: 10.1371/journal.pone.0180944.

[20] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2018, pp. 1394–1401, doi: 10.1109/ICMLA.2018.00227.

[21] M. H. Pesaran, "Time Series and Panel Data Econometrics," *Time Ser. Panel Data Econom.*, 2016, doi: 10.1093/acprof:oso/9780198736912.001.0001.

[22] I. M. Chakravarti, G. E. P. Box, and G. M. Jenkins, "Time Series Analysis Forecasting and Control.," *J. Am. Stat. Assoc.*, vol. 68, no. 342, p. 493, 1973, doi: 10.2307/2284112.

[23] R. DiPietro and G. D. Hager, "Deep learning: RNNs and LSTM," in *Handbook of Medical Image Computing and Computer Assisted Intervention*, Elsevier, 2020, pp. 503–519.

[24] M. Sangiorgio and F. Dercole, "Robustness of LSTM neural networks for multi-step forecasting of chaotic time series," *Chaos, Solitons & Fractals*, vol. 139, p. 110045, Oct. 2020, doi: 10.1016/j.chaos.2020.110045.

[25] M. F. Rabby, Y. Tu, M. I. Hossen, I. Lee, A. S. Maida, and X. Hei, "Stacked LSTM based deep recurrent neural network with kalman smoothing for blood glucose prediction," *BMC Med. Inform. Decis. Mak.*, vol. 21, no. 1, 2021, doi: 10.1186/s12911-021-01462-5.

[26] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994, doi: 10.1109/72.279181.

[27] O. M. Yamashita, J. R. Betoni, S. C. Guimarães, and M. M. Espinosa, "Deep Learning (Adaptive Computation and Machine Learning series)," *Sci. For. Sci.*, no. 84, p. 355, 2009.

[28] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989, doi: 10.1162/neco.1989.1.2.270.

[29] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 2015-January, pp. 1171–1179, 2015.

[30] A. Listou Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov, and H. Zhang, "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture," *Reliab. Eng. Syst. Saf.*, vol. 183, pp. 240–251, 2019, doi: 10.1016/j.ress.2018.11.027.

[31] Y. Freund and D. Haussler, "Unsupervised learning of distributions on binary vectors using two layer networks," in *NIPS'91: Proceedings of the 4th International Conference on Neural Information Processing Systems*, 1991, pp. 912–919.

[32] A. L. Ellefsen, S. Ushakov, V. Aesoy, and H. Zhang, "Validation of Data-Driven Labeling Approaches Using a Novel Deep Network Structure for Remaining Useful Life Predictions," *IEEE Access*, vol. 7, pp. 71563–71575, 2019, doi: 10.1109/ACCESS.2019.2920297.

[33] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–27, 2009, doi: 10.1561/2200000006

[34] W. Wei, H. Wu, and H. Ma, "An autoencoder and LSTM-based traffic flow prediction method," *Sensors (Switzerland)*, vol. 19, no. 13, 2019, doi: 10.3390/s19132946.

[35] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE J.*, vol. 37, no. 2, pp. 233–243, Feb. 1991, doi: 10.1002/aic.690370209.

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.

[38] P.-A. M. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010

[39] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6791 LNCS, no. PART 1, pp. 52–59, doi: 10.1007/978-3-642-21735-7_7.

[40] O. Yildirim, R. S. Tan, and U. R. Acharya, "An efficient compression of ECG signals using deep convolutional autoencoders," *Cogn. Syst. Res.*, vol. 52, pp. 198–211, Dec. 2018, doi: 10.1016/j.cogsys.2018.07.004.

[41] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, 2015.

[42] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On Empirical Comparisons of Optimizers for Deep Learning," 2019, [Online]. Available: http://arxiv.org/abs/1910.05446.

[43] N. Ketkar, "Stochastic Gradient Descent," in *Deep Learning with Python*, Berkeley, CA: Apress, 2017, pp. 113–132.

[44] G. Hinton and T. Tieleman, "RMSPROP: Divide the Gradient by a Running Average of its Recent Magnitude," *Coursera Neural Networks Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012, [Online]. Available: https://www.coursera.org/learn/neural-networks/lecture/YQHki/rmsprop-divide-the-gradient-by-a-running-average-of-its-recent-magnitude.

[45] A. Paszke *et al.*, "PyTorch: An Imperative Style , High-Performance Deep Learning Library," Adv. Neural Inf. Process. Syst., no. NeurIPS, pp. 8024–8035, 2019.