Kernel Corrector LSTM *

 $\begin{array}{c} {\rm Rodrigo\ Tuna^{1}}_{[0009-0009-0047-2052]},\ Yassine\ Baghoussi^{1,4}}_{[0000-0002-1943-1471]}, \\ {\rm Carlos\ Soares^{1,2,3}}_{[0000-0003-4549-8917]},\ {\rm and\ João\ Mendes-Moreira^{1,4}}_{[0000-0002-2471-2833]} \end{array}$

 ¹ Faculdade de Engenharia, Universidade do Porto up201904967@edu.fe.up.pt, {baghoussi,csoares,jmoreira}@fe.up.pt
² Artificial Intelligence & Computer Science Lab.(LIACC - member of LASI LA), Universidade do Porto
³ Fraunhofer AICOS Portugal

⁴ INESC TEC, Portugal

Abstract. Forecasting methods are affected by data quality issues in two ways: 1. they are hard to predict, and 2. they may affect the model negatively when it is updated with new data. The latter issue is usually addressed by pre-processing the data to remove those issues. An alternative approach has recently been proposed, Corrector LSTM (cLSTM), which is a Read & Write Machine Learning (RW-ML) algorithm that changes the data while learning to improve its predictions. Despite promising results being reported, cLSTM is computationally expensive, as it uses a meta-learner to monitor the hidden states of the LSTM. We propose a new RW-ML algorithm, Kernel Corrector LSTM (KcLSTM), that replaces the meta-learner of cLSTM with a simpler method: Kernel Smoothing. We empirically evaluate the forecasting accuracy and the training time of the new algorithm and compare it with cLSTM and LSTM. Results indicate that it is able to decrease the training time while maintaining a competitive forecasting accuracy.

Keywords: Time series for ecasting \cdot Recurrent Neural Networks \cdot Data-Centric AI

1 Introduction

In many fields, including energy, healthcare, management, and climate research, time series forecasting is a crucial task that can be accomplished using machine

^{*} This work was partially funded by projects AISym4Med (101095387) supported by Horizon Europe Cluster 1: Health, ConnectedHealth (n.^o 46858), supported by Competitiveness and Internationalisation Operational Programme (POCI) and Lisbon Regional Operational Programme (LISBOA 2020), under the PORTU-GAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF); NextGenAI - Center for Responsible AI (2022-C05i0102-02), supported by IAPMEI, and also by FCT plurianual funding for 2020-2023 of LIACC (UIDB/00027/2020_UIDP/00027/2020) and SONAE IM Labs@FEUP.

learning or statistical methods [8]. As data becomes widely available, more precise forecasting models are expected. However, data quality issues like outliers, missing values, and changes in the underlying data generation process might impact predictive techniques.

Traditional machine learning (ML) models are often considered read-only models, capable of learning from data but neglecting the feedback loop for correcting the data during the learning process. This approach, while efficient in many cases, lacks proper adaptation of preprocessing techniques and the ML model itself, as the model's feedback is often overlooked.

To address this limitation, the concept of Read-Write Machine Learning (RW-ML) has emerged. RW-ML models, such as Corrector LSTM (cLSTM) [1], not only learn from data but also have the capability to change the data during the learning process. cLSTM is a time series forecasting method designed to improve forecasting accuracy by dynamically adjusting the data. It utilizes a meta-model of the Hidden State Dynamics obtained with SARIMA to detect data quality issues and employs a greedy heuristic to correct them. cLSTM has demonstrated superior predictive performance compared to traditional LSTM models. However, the computational cost associated with the meta-learning component of cLSTM is significant.

In this paper, we propose a computationally less expensive variant of cLSTM, named Kernel Corrector LSTM (KcLSTM), which replaces the meta-learner with a simpler method: Kernel Smoothing. We empirically compare KcLSTM with both cLSTM and LSTM models. Results reveal that KcLSTM achieves better predictive performance than LSTM and cLSTM, while also being faster than cLSTM, although the computational efficiency improvement is not as substantial as expected.

The main contributions of this paper are:

- Introducing a variant of cLSTM, KcLSTM, which is computationally less expensive while maintaining high predictive accuracy.
- An empirical study comparing KcLSTM with LSTM and cLSTM in terms of predictive performance and training time.

This paper is structured as follows: we first provide an overview of the stateof-the-art forecasting method, LSTM. Then, we delve into the concept of RW-ML and its significance in time series forecasting. Next, we introduce the proposed algorithm, KcLSTM. Finally, we describe the experimental setup, present the results, and discuss their implications.

2 Related Work

In this section, we first present the Long Short-Term Memory. The algorithm that our proposed algorithm is built on and the one it will be compared to. We then define Data-Centric AI and provide examples of Data-Centric models built for time series forecasting. 2.1 LSTM



Fig. 1. Cell unit of the LSTM recurent neural network [6].

The Long Short-Term Memory (LSTM) [16] is a Recurrent Neural Network (RNN), that can capture long-term dependencies in the input and it is used for processing sequential data. RNNs differ from feed-forward networks through recurrent connections, allowing them to learn from sequential data. Back Propagation is applied to RNNs by taking advantage of the fact that for every recurrent network, there exists an equivalent feed-forward network with identical behavior for a finite number of steps [21], training it using Back Propagation Through Time (BPTT) [23].

RNNs have some well-known limitations. First, they have problems capturing long-term dependencies, being limited to only bridge between 5-10 steps [23]. This occurs because RNNs are sensible to the exploding/vanishing gradient problem [15]. The LSTM solves this problem through the use of a gating mechanism.

An LSTM network consists of blocks, with each block containing an input gate, forget gate, output gate, and memory cell (eqs. (1b) to (1d)). The input gate controls which inputs are relevant; the forget gate learns which information should be kept in memory; and the output gate controls which information should be passed to the next block. The information is retained through the use of two states called the cell state, (eq. (1e)), and the hidden state, (eq. (1f)). The forward pass concatenates the input with the hidden state from the last block, while the backward pass derives the error and updates the gates using

the chain rule of derivatives. The gate derivatives are multiplied by the hidden output to obtain the gradient deltas that update the gates.

$$i_t = \sigma(W_i \cdot h_{t-1} + V_i \cdot x_t + b_i) \tag{1a}$$

$$o_t = \sigma(W_o \cdot h_{t-1} + V_o \cdot x_t + b_o) \tag{1b}$$

$$f_t = \sigma(W_f \cdot h_{t-1} + V_f \cdot x_t + b_f) \tag{1c}$$

$$\hat{C}_t = tanh(W_c \cdot h_{t-1} + V_c \cdot x_t + b_c) \tag{1d}$$

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

$$h_t = o_t \cdot tanh(C_t) \tag{1f}$$

$$z_t = h_t \tag{1g}$$

2.2 Data-Centric Time Series Forecasting

Anomalies, including outliers, missing values, and changes in the underlying data generation process can impact predictive tasks. This affects the predictions of such methods, hindering their performance [17], conversely to traditional machine learning methods, that build models using a fixed dataset. In Data-Centric AI [25], the focus is on the data. Data quality is increased to improve the performance of AI models.

The exploration into machine learning models capable of learning and correcting data has been a topic of interest in various studies. Both [22] and [20] delve into this concept, with [22] focusing on the potential of ML models to memorize sensitive information while [20] emphasize the importance of model interpretability and safety. Additionally, authors in [4] further underscore the significance of data quality in enhancing model performance, advocating for a data-centric approach. Providing a broader perspective, the work in [14] discusses the role of probabilistic modeling in understanding learning and uncertainty in machine learning.

Moving to neural network models, authors in [11] discuss highly interconnected networks for associative memory and optimization, with a focus on learning and adaptation. Moreover, [10] propose a model for neural networks that learn temporal sequences through selection, employing synaptic triads and a local Hebbian learning rule. Furthermore, [9] introduce predictive-corrective networks for action detection in videos, which utilize top-down predictions and bottomup observations for adaptive computation and simplified learning. These models collectively demonstrate the potential of neural networks to learn and correct data across various applications.

Similarly, recurrent neural network (RNN) models have been developed to address the challenge of learning and correcting data. In [13], an attempt is made to introduce a learning algorithm for the recurrent random network model, employing gradient descent of a quadratic error function. Later, authors in [2] propose a recurrent network architecture for modeling dynamical systems, which can learn from multiple temporal patterns and cope with sparse data. More recently, the research in [5] demonstrates that tree-structured recursive neural networks can learn logical semantics, including entailment and contradiction.

In the context of time series forecasting, some data-centric approaches have been employed. In dLSTM [19], the authors train the model on non-anomalous data and use the predictive errors to detect anomalies. The deviation from the normal state is measured through delayed prediction errors. The normal state can then be restored from several candidate values. Following the idea of using the prediction errors to improve the quality of the data Pastprop was introduced [3]. The responsibility for the training error is shared between the model parameters and the training data. The backpropagation of the derivatives is applied to the input, indicating the part of the input that caused the training error.

2.3 Corrector LSTM

cLSTM [1] is an architecture that improves its predictive performance by reconstructing the data of the model. The architecture of the algorithm is based on the LSTM and a data correction component. This data correction component uses a meta-learner, SARIMA, to identify problems in the hidden states of an LSTM model. This is achieved by predicting the hidden states using SARIMA and if the difference between the predicted and the real hidden states is over a certain threshold they are considered anomalous. The anomalies detected in the hidden states are assumed to be caused by the data which is then reconstructed. The reconstruction of the data points is such that the difference between the predicted and the real hidden states falls under a certain threshold. The authors showed that analyzing the Hidden State Dynamics [24] of an LSTM can be used to detect anomalies in the training data and consequently improve the forecasting performance of the model. However, the data correction relies on a meta-learner which makes the algorithm computationally expensive.

3 Kernel Corrector LSTM

The architecture of the Kernel Corrector LSTM (KcLSTM) is the same as the cLSTM architecture, and the meta-learner used to detect problems in the learning is substituted by a simpler approach, kernel smoothing.

3.1 Training

The KcLSTM utilizes the hidden states learned during the training process to find and correct data points of the series. The training of the KcLSTM is divided into three distinct phases. The first phase consists of training the data on a standard LSTM. This allows the hidden states to capture the information of the time series and to be indicative of problems in the data. We then perform the correction, which is comprised of a detection and a correction component. These two components find and correct errors in the data respectively, this phase is thoroughly explained in section 3.2. Finally, the LSTM is trained on the new

data, learning a corrected time series, that can improve the predictions of the model.

3.2 Data Correction

The Data Correction phase of the algorithm is divided into two different components: the correction and the detection. These two components aim to find data points that worsen the learning of the model and change the data so that the learning process is improved and a better model is obtained. Each phase has a threshold δ_d and δ_c . The hidden states of the last iteration of the first training phase, $H = h_0, ..., h_n$ are used to find errors in the training data. cLSTM uses a meta-model that is computationally expensive to compute; our goal is to assess if a simpler method can obtain competitive results with less cost; the method selected for this purpose is Kernel Smoothing because states are estimated rather than predicted which makes it computationally. A new set of estimated hidden states $H' = h'_0, ..., h'_n$ is calculated using Gaussian Kernel Smoothing of H as described in eq. (2).

$$h'_{i} = \frac{\sum_{j \in [i-W/2, i+W/2], i \neq j} h_{j} * K(h_{i}, h_{j})}{\sum_{j \in [i-W/2, i+W/2], i \neq j} K(h_{i}, h_{j})}$$
(2)

Where $K(h_i, h_j)$ is:

$$K(h_i, h_j) = e^{\frac{\|h_i - h_j\|^2}{2\sigma^2}}$$
(3)



Fig. 2. Gaussian Smoothing of the Hidden States, represented as a series.

The goal of error detection is to discriminate between data points that need reconstruction and those that do not. A point needs reconstruction if the Dynamic Time Warp similarity between the hidden state from which the point originated h_i and the corresponding estimated hidden state h'_i is greater than a given threshold δ_d . This relation is depicted in eq. (4).

$$DTW(h'_i, h_i) > \delta_d \tag{4}$$

In the error correction, we reconstruct the detected points such that the Dynamic Time Warp similarity of the hidden state and estimated hidden state is less or equal to a given threshold δ_c eq. (5). Early stopping is employed, and if a maximum number of iterations is reached, the original value for the point is restored.

$$DTW(h'_i, h_i) \le \delta_c \tag{5}$$

4 Experimental Setup

The goals of the empirical validation are to investigate if the proposed algorithm is faster than the original one, without a significant decrease in forecasting accuracy.

A straightforward holdout method was used to estimate forecasting performance, used when there is a temporal dependency in the dataset [7]. The model is trained on the first s samples and assessed on the succeeding n - s samples. The data used for evaluation is always the original one. Using corrected data for the evaluation would likely lead to inadequate optimistic estimates of the forecasting performance of the corresponding method.

The hyperparameters of LSTM and KcLSTM are chosen using hyperparameter tuning using grid search. The learning rate was varied between: 0.0001, 0.001, 0.01, 0.1; and the batch size was varied between: 1, 2, 4, 8. For cLSTM we do not perform hyper-parameter tuning due to the high computational cost. Instead, we use the results described in the original paper [1]. The thresholds for KcLSTM are fixed with values of 0.6 for the detection and 0.5 for the correction. The thresholds for cLSTM are described in the original paper [1], 0.6 for the detection and 0.2 for the correction. We chose to maintain the same detection threshold and increased the correction threshold. The kernel estimates of the hidden states are smoother; thus, a small correction threshold would significantly alter the hidden states, and the information learned in the previous phase would be lost.

4.1 Datasets

	Monthly
Timeseries	200
Average Length	366
Mean	4222
Standard Deviation	1160

Table 1. Statistical description of the dataset.

We have used the M4 Competition Dataset [18] comprising six subsets. From one subset, Monthly, we evaluate the performance of the algorithms on the first

199 time series. To evaluate the time taken to train the models, we use the first 20 time series of that subset.

4.2 Evaluation Metrics

This study focuses on both the predictive performance of the algorithm as well as its training time. To quantify the error of forecasts, we focus on the Mean Absolute Scaled Error (MASE) in Eq 6 because it allows for the averaging of results across different time series as opposed to the Rooted Mean Squared Error (RMSE). The MASE measures the appropriateness of a forecast against the naive forecast of predicting the previous value. To assess if the differences are statistically different, we use the Mariano-Diebold Test [12].

$$MASE = \frac{\frac{1}{n-s}\sum_{i=s+1}^{n}\hat{y}_i - y_i}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|}$$
(6)

The execution time is measured in seconds and the experiments were run on an Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz processor.

5 Results

investigate if the proposed algorithm is faster than the original one, without significantly decreasing forecasting accuracy. To illustrate the usefulness of the proposed algorithm, we first evaluate its forecasting capabilities, comparing it with LSTM section 5.2 and cLSTM section 5.1.

5.1 Comparison with cLSTM

Results of MASE for the algorithms presented in table 2 indicate KcLSTM outperforms cLSTM, but this may be explained by the hyper-parameter tuning that was performed for the KcLSTM and not the cLSTM. As such comparison between these two algorithms, can not be performed directly.

The Mariano-Diebold Test for cLSTM and KcLSTM resulted in 40 wins for cLSTM, 111 wins for KcLSTM, and 48 draws. This shows an improvement in forecasting accuracy by substituting the meta-learner with Kernel Smoothing. Again the uneven conditions do not allow us to reach clear conclusions about these two methods.

Results for the training time presented in table 2 indicate that KcLSTM is indeed faster than cLSTM, significantly. Nonetheless, the gain is not as large as would be expected. Estimating the states with Kernel Smoothing is less computationally expensive than predicting the states with SARIMA. However, this is a cruder method that results in estimated states that are farther away than from the original states when compared with. Consequently, more points are considered anomalies that will be corrected, which will cause the training time to increase.

9

Fabl	le 2	. C	omparison	of	each	a	lgorit	hm.
------	------	-----	-----------	----	------	---	--------	-----

	Mean	Median	Standard Deviation	Average time (s)
LSTM	3.48	0.74	6.07	20.47
cLSTM	8.77	1.04	36.96	56.15
KcLSTM	4.64	0.83	11.96	48.77

However, when performing the Mariano-Diebold test to compare LSTM and KcLSTM at the significance level of $\alpha = 0.05$ we get 102 wins for KcLSTM, 50 wins for LSTM, and 47 draws. We can conclude that KcLSTM is superior to LSTM as it wins more often, although when it loses it is by a greater margin. This is confirmed by the values of the standard deviation of MASE for LSTM and KcLSTM and explains the (apparent) superiority of LSTM when analyzing only the MASE. We see examples of a series with clear outliers that KcLSTM is able to correct and as such increase their predictions in fig. 3. Conversely, an example of a series without outliers made KcLSTM wrongfully alter the data which results in disastrous predictions in fig. 4. These two examples reflect the different behaviors mentioned before.



Fig. 3. Example where data reconstruction was successful.

5.2 Comparison with LSTM

Results of MASE for the algorithms presented in table 2 indicate that LSTM has an overall better performance than KcLSTM with lower values for both the median and the mean for the MASE. However, when performing the Mariano-Diebold test to compare LSTM and KcLSTM at the significance level of $\alpha = 0.05$ we get 102 wins for KcLSTM, 50 wins for LSTM, and 47 draws. We can conclude that KcLSTM is superior to LSTM as it wins more often, although when it loses it is by a greater margin. This is confirmed by the values of the standard deviation of MASE for LSTM and KcLSTM and explains the (apparent) superiority of



Fig. 4. Example where the data reconstruction destroyed the data which caused the model not to capture the information of the series.

LSTM when analyzing only the MASE. We see examples of a series with clear outliers that KcLSTM is able to correct and as such increase their predictions in fig. 3. Conversely, an example of a series without outliers made KcLSTM wrongfully alter the data which results in worse predictions in fig. 4. These two examples reflect the different behaviors mentioned before.

Results for the training time presented in table 2 indicate that KcLSTM is significantly slower than LSTM. This to be expected as KcLSTM has a data correction component that is responsible for most of the execution time of the algorithm.

6 Conclusion

The goal of our work is to create a forecasting algorithm that reconstructs data that is faster than current solutions in the literature. We present a new algorithm: Kernel Corrector LSTM (KcLSTM). This algorithm alters the training data to improve its forecasting accuracy. Like in cLSTM, this is done by analyzing the Hidden States Dynamics and finding anomalies in hidden states to detect anomalies in data points and consequently correct them. However, the metalearner of cLSTM was replaced by the Gaussian Kernel Smoothing of the hidden states to decrease the training time of cLSTM.

We empirically compare our algorithm with LSTM and cLSTM both in terms of predictive performance and training time. Results show that KcLSTM obtains a competitive forecasting accuracy surpassing both the LSTM and cLSTM in the number of statistically significant wins. However, KcLSTM is more sensitive to its training data and more prone to making worse forecasts than the baseline, which caused the average MASE of LSTM to be inferior to the average MASE of KcLSTM. The measured training times also show that KcLSTM indeed improves on cLSTM in terms of computational cost, but the margin is smaller than expected because KcLSTM detects more points as anomalies than cLSTM. The estimated hidden states by KcLSTM are more distant from the real hidden states than the predicted states of cLSTM. The empirical study showed that KcLSTM is a faster algorithm that corrects its training data than cLSTM and that those corrections improve the forecasts by being superior to LSTM. Future work comprises the possibility of implementing the algorithm with different estimators.

References

- Baghoussi, Y., Soares, C., Mendes-Moreira, J.: Corrector lstm: Built-in training data correction for improved time series forecasting. In: Proceedings of the 8th SIGKDD International Workshop on Mining and Learning from Time Series–Deep Forecasting: Models, Interpretability, and Applications. pp. 1–8. ACM, Washington DC, USA (August 2022)
- Bailer-Jones, C., MacKay, D.J.C., Withers, P.J.A.: A recurrent neural network for modelling dynamical systems. Network 9 4, 531-47 (1998), https://api. semanticscholar.org/CorpusID:653765
- 3. Baptista, A., Baghoussi, Y., Soares, C., Mendes-Moreira, J., Arantes, M.: Pastproprnn: improved predictions of the future by correcting the past (2021)
- Bhowmik, P., Partha, A.S.: A data-centric approach to improve machine learning model's performance in production. International Journal of Engineering and Advanced Technology (2021), https://api.semanticscholar.org/CorpusID: 240328155
- Bowman, S.R., Potts, C., Manning, C.D.: Recursive neural networks can learn logical semantics. In: Workshop on Continuous Vector Space Models and their Compositionality (2014), https://api.semanticscholar.org/CorpusID:15618372
- Castro, J., Achanccaray Diaz, P., Sanches, I., Cue La Rosa, L., Nigri Happ, P., Feitosa, R.: Evaluation of recurrent neural networks for crop recognition from multitemporal remote sensing images (11 2017)
- Cerqueira, V., Torgo, L., Mozetič, I.: Evaluating time series forecasting models: an empirical study on performance estimation methods. Machine Learning 109(11), 1997–2028 (oct 2020). https://doi.org/10.1007/s10994-020-05910-7, https://doi.org/10.1007%2Fs10994-020-05910-7
- 8. Cerqueira, V., Torgo, L., Soares, C.: Machine learning vs statistical methods for time series forecasting: Size matters (2019)
- Dave, A., Russakovsky, O., Ramanan, D.: Predictive-corrective networks for action detection. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2067-2076 (2017), https://api.semanticscholar.org/CorpusID: 2466592
- Dehaene, S., Changeux, J.P., Nadal, J.P.: Neural networks that learn temporal sequences by selection. Proceedings of the National Academy of Sciences of the United States of America 84 9, 2727-31 (1987), https://api.semanticscholar. org/CorpusID:7423734
- Denker, J.S.: Neural network models of learning and adaptation. Physica D: Nonlinear Phenomena 2, 216-232 (1986), https://api.semanticscholar.org/ CorpusID:119988262
- Diebold, F., Mariano, R.: Comparing predictive accuracy. Journal of Business & Economic Statistics 13(3), 253-63 (1995), https://EconPapers.repec.org/ RePEc:bes:jnlbes:v:13:y:1995:i:3:p:253-63
- 13. Gelenbe, E.: Learning in the recurrent random neural network. Neural Computation 5, 154-164 (1992), https://api.semanticscholar.org/CorpusID:38667978

- 12 R. Tuna et al.
- Ghahramani, Z.: Probabilistic machine learning and artificial intelligence. Nature 521, 452-459 (2015), https://api.semanticscholar.org/CorpusID:216356
- Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6, 107–116 (04 1998). https://doi.org/10.1142/S0218488598000094
- Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9, 1735–80 (12 1997). https://doi.org/10.1162/neco.1997.9.8.1735
- Kanarachos, S., Christopoulos, S.R.G., Chroneos, A., Fitzpatrick, M.E.: Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and hilbert transform. Expert Systems with Applications 85, 292– 304 (2017). https://doi.org/https://doi.org/10.1016/j.eswa.2017.04.028, https:// www.sciencedirect.com/science/article/pii/S0957417417302737
- Makridakis, S., Spiliotis, E., Assimakopoulos, V.: The m4 competition: Results, findings, conclusion and way forward. International Journal of Forecasting 34(4), 802-808 (2018). https://doi.org/https://doi.org/10.1016/j.ijforecast.2018.06.001, https://www.sciencedirect.com/science/article/pii/S0169207018300785
- Maya, S., Ueno, K., Nishikawa, T.: dlstm: a new approach for anomaly detection using deep learning with delayed prediction. International Journal of Data Science and Analytics 8 (09 2019). https://doi.org/10.1007/s41060-019-00186-0
- 20. Otte, C.: Safe and interpretable machine learning: A methodological review (2013), https://api.semanticscholar.org/CorpusID:56899177
- Sherstinsky, A.: Fundamentals of recurrent neural network (rnn) and long shortterm memory (lstm) network. Physica D: Nonlinear Phenomena 404, 132306 (2020). https://doi.org/https://doi.org/10.1016/j.physd.2019.132306, https:// www.sciencedirect.com/science/article/pii/S0167278919305974
- Song, C., Ristenpart, T., Shmatikov, V.: Machine learning models that remember too much. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (2017), https://api.semanticscholar.org/CorpusID: 2904063
- 23. Staudemeyer, R.C., Morris, E.R.: Understanding lstm a tutorial into long short-term memory recurrent neural networks (2019)
- 24. Strobelt, H., Gehrmann, S., Pfister, H., Rush, A.M.: Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks (2017)
- Zha, D., Bhat, Z.P., Lai, K.H., Yang, F., Jiang, Z., Zhong, S., Hu, X.: Data-centric artificial intelligence: A survey (2023)