

Enhanced transformer long short-term memory framework for datastream prediction

Nada Adel Dief, Mofreh Mohamed Salem, Asmaa Hamdy Rabie, Ali Ibrahim El-Desouky

Department of Computer and Control Systems Engineering, Faculty of Engineering, Mansoura University, Cairo, Egypt

Article Info

Article history:

Received Aug 15, 2023

Revised Sep 26, 2023

Accepted Oct 9, 2023

Keywords:

Datastream

Long short-term memory

Machine learning

Multiprocessing pool

Parallel processing

Prediction accuracy

Transformer

ABSTRACT

In machine learning, datastream prediction is a challenging issue, particularly when dealing with enormous amounts of continuous data. The dynamic nature of data makes it difficult for traditional models to handle and sustain real-time prediction accuracy. This research uses a multi-processor long short-term memory (MPLSTM) architecture to present a unique framework for datastream regression. By employing several central processing units (CPUs) to divide the datastream into multiple parallel chunks, the MPLSTM framework illustrates the intrinsic parallelism of long short-term memory (LSTM) networks. The MPLSTM framework ensures accurate predictions by skillfully learning and adapting to changing data distributions. Extensive experimental assessments on real-world datasets have demonstrated the clear superiority of the MPLSTM architecture over previous methods. This study uses the transformer, the most recent deep learning breakthrough technology, to demonstrate how well it can handle challenging tasks and emphasizes its critical role as a cutting-edge approach to raising the bar for machine learning.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Nada Adel Dief

Department of Computer and Control Systems Engineering, Faculty of Engineering, Mansoura University

Mansoura, 35511, Egypt

Email: nadadief@mans.edu.eg

1. INTRODUCTION

In the era of big data, traditional machine learning methods can be computationally burdensome and complex, making them unsuitable for processing such large-scale datasets. To achieve accurate predictions for data, traditional machine learning techniques often struggle to handle the challenges posed by big data, including its sheer volume, complexity, and high-dimensional nature [1]. On the other hand, data-driven methods utilizing deep learning have attracted interest due to their capacity to perform statistical analysis and information extraction automatically and successfully on large-scale, multi-source, and high-dimensional data, thereby overcoming the limitations of traditional prediction methods [2]. Recurrent neural networks (RNNs) are a kind of neural network that is particularly good at handling sequential data. They have a feedback connection, in contrast to conventional feedforward neural networks, which enables them to keep an internal memory of prior inputs. This memory enables RNNs to effectively capture temporal dependencies and patterns in sequential data. However, traditional RNNs experience the “vanishing gradient” problem, where the gradient signal weakens with time and makes it difficult to adequately capture long-term dependencies. To overcome this restriction, variants like long short-term memory (LSTM) and gated recurrent unit (GRU) were developed. Incorporating gating mechanisms that selectively remember or forget information, these models are better able to capture and spread pertinent information over longer sequences [3]. Additionally, in neural networks (NNs), the pre-assignment of

parameters defines the network's topology and has an impact on how computationally intensive training and prediction are. Therefore, optimizing the parameters is crucial for achieving excellent performance. However, like other deep learning (DL) networks [4], LSTM also faces the challenge of parameter selection, which often requires hand-engineered adjustments. Manual parameter adjustment is difficult, particularly when dealing with vast amounts of data and very deep network structures. To address this issue, a grid search (GS) is employed to look for the ideal settings for multi-processor long short-term memory (MPLSTM), leading to predicting the datastream flow. This approach aims to build a suitable model structure and increase the MPLSTM's prediction accuracy. A multi-processor LSTM framework for real-time data stream processing is the main goal of this study. It conducts a comprehensive analysis of MPLSTM using a real-life dataset, offering valuable insights into monitoring the parallel approach.

2. THE PROPOSED DATASTREAM MULTIPROCESSING LSTM FRAMEWORK

In this section, a framework for real datastream analysis is presented that harnesses the strengths of MPLSTM along with other techniques to attain superior accuracy in real-time data processing. Figure 1 demonstrates the framework's overall architecture and gives a visual representation of the intricate details that underlie its operational procedures. The framework is made up of several parts, including an output layer, a hidden layer, and a layer for data input.

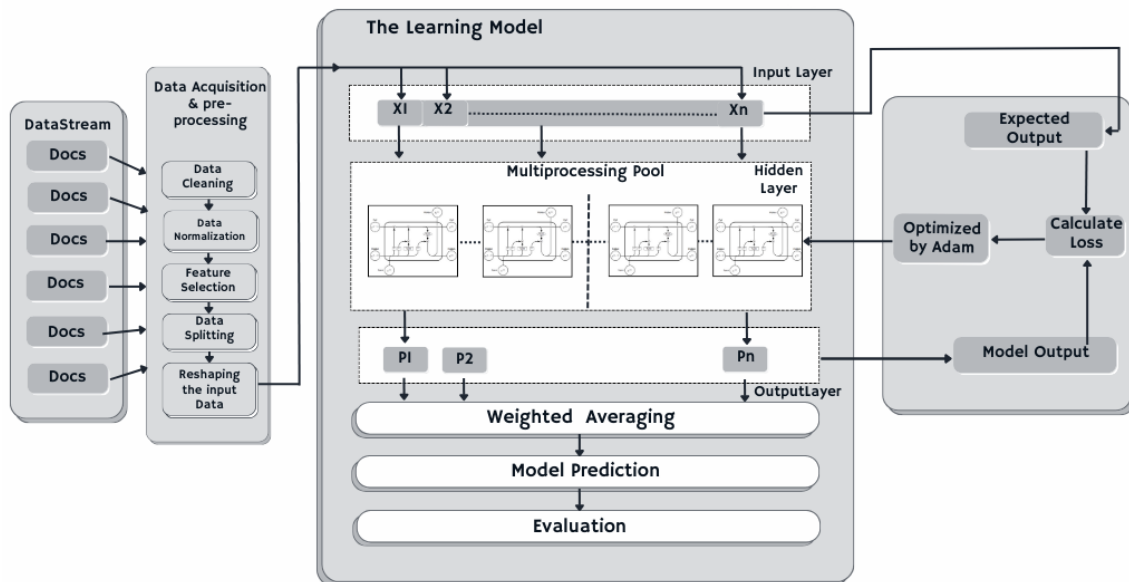


Figure 1. The details of the proposed MPLSTM framework

2.1. Data acquisition and preprocessing

The pre-processing stage is essential for getting the data ready for input into MPLSTM. This phase involves several steps, including data cleaning, normalization, splitting, and reshaping [5]. Firstly, data cleaning is performed to remove any missing or inconsistent data that can adversely affect the model's performance. Secondly, normalization, splitting, and reshaping the input data into a form that is compatible with the LSTM unit to scale the data and bring it within a specified range to avoid bias in the model's performance [6].

2.2. The learning model

After splitting the dataset into train and test, the training set is divided into chunks and processed in parallel using the multiprocessing pool. The pool manages a pool of worker processes, automatically assigning tasks to available workers and handling the communication between the main process and the worker processes. After that, the proposed MPLSTM framework is trained with various hyperparameters adjusted through multiple experiments until reaching a stable state, optimizing the weights with a grid search algorithm for best performance.

2.2.1. Parallelization using multiprocessing pool

Parallelizing LSTM-based models using multiprocessing [7] enables faster processing of input sequences, efficient resource utilization, scalability, and flexibility in model design and optimization. It can be particularly useful in scenarios where large sequences or computationally intensive models need to be processed within a reasonable time frame. To apply multiprocessing, there are some steps [8] that must be followed: i) split the data into smaller chunks that can be processed independently, ii) create a function that will be executed in parallel by multiple processes, iii) this function will take a data chunk as input and perform LSTM processing on that chunk, iv) inside the function, create an instance of the LSTM model and train or predict on the input chunk, v) set up a multiprocessing pool, vi) the pool manages a group of worker processes that will execute the parallel processing function, vii) specify the number of worker processes to utilize, typically based on the available hardware resources, viii) collect the results from the parallel processes, and ix) however, it is important to note that the level of parallelism achievable depends on factors such as the number of available CPU cores, memory capacity, and the size of the input sequence. Parallel execution can be increased potentially by having a higher number of CPU cores, allowing for the execution of more processes in parallel. Similarly, the presence of sufficient memory capacity is crucial to accommodate the running of data and processes in parallel without being constrained by memory-related issues.

2.2.2. Dropout layer

To increase the network's speed and sturdiness, dropout regularization has been incorporated [9]. A regularization method is frequently employed in neural networks. It randomly deactivates a fraction of the neurons in the previous layer during each training iteration. This dropout of neurons helps prevent overfitting [10] by reducing the reliance of the network on specific neurons and encourages the learning of more robust and generalizable representations. During inference, the dropout layer is typically turned off, and the full network is used for making predictions. By incorporating dropout layers, the network becomes more resilient to overfitting and can improve its generalization performance on unseen data [11], [12].

2.2.3. Dense layer

A fully connected layer is a fundamental component of a neural network. It consists of multiple nodes, or neurons, where every neuron is linked to every other neuron in the layer below. An activation function is applied to the weighted sum of the inputs from the layer before to determine each neuron's output in a dense layer [13]. To maximize the network's performance on the specified task, these weights and biases are learned throughout the training phase [14].

The model consists of several dense layers that are fully connected to all the activations in the former layer. These dense layers combine the complicated feature maps to produce a feature vector that is flattened. The occurrences are then categorized using the softmax [15] output probabilities produced by the last dense layer.

2.2.4. Adam optimizer

This section highlights the importance of parameter optimization in improving the model's performance. MPLSTM was trained using the Adam optimizer algorithm. A well-liked optimization technique frequently employed in deep learning is the Adam optimizer [16]. It combines the advantages of both the adaptive gradient algorithm (AdaGrad) and root mean square propagation (RMSProp) algorithms by adapting the learning rate for each parameter individually. This adaptive learning rate helps in achieving faster convergence as well as improved performance while training. The Adam optimizer is used to reduce the categorical cross-entropy loss function with a learning rate of 10^{-4} . By using the Adam optimizer, MPLSTM by successfully updating its weights and biases, can reduce the loss function and boost its accuracy overall.

2.3. Prediction

In the prediction phase, the input data is fed forward through the network, The softmax function is applied in the output layer to create a probability distribution over the classes [17]. The anticipated class label is normally determined by the class with the highest probability. Softmax makes sure that the projected probabilities are restricted between 0 and 1 and add up to 1. Because of this, it is appropriate for multi-class classification problems in which each instance belongs to a single class.

$$P_i = \frac{\exp(Z_i)}{\sum_{j=1}^n \exp(Z_j)} \quad (1)$$

where n is the total number of classes, Z_i is the raw output value for class i . By applying softmax, the neural network can provide a probability-based prediction, allowing for decision-making based on the highest probability class.

2.4. Evaluation

Datastreams often exhibit changes in the class distribution of incoming instances regularly. These metrics provide a more comprehensive assessment of the model's performance, considering the evolving nature of the data stream and allowing for timely adaptation and monitoring. To evaluate the results, MPLSTM uses the following evaluation metrics: classification accuracy: it compares the predictions of MPLSTM with the actual target values from the dataset [18].

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total Number of predictions}} \quad (2)$$

Where *Number of correct predictions*: This refers to the count of instances where MPLSTM correctly predicts the target value, *Total number of predictions*: This is the total number of instances for which predictions were made by the MPLSTM. The result will be a value between 0 and 1, representing the percentage of correct predictions made by MPLSTM.

Then, three error metrics mean square error (MSE), root mean squared error (RMSE), and mean absolute error (MAE) Loss are used to assess the model's performance. These measures are employed to evaluate several facets of the model's precision and prognostication [19].

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_t - \bar{y}_t)^2 \quad (3)$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_t - \bar{y}_t)^2} \quad (4)$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_t - \bar{y}_t| \quad (5)$$

Where n is the number of samples, y_t , \bar{y}_t the predicted and actual values respectively. RMSE measures the average magnitude of the prediction errors by taking the square root of the mean squared difference between the predicted y_t and actual values \bar{y}_t . It indicates how accurately the model predicts the desired variable. While MAE measures the average absolute difference between y_t , \bar{y}_t .

3. LSTM enhancement

Inspired by the transformer model's innovations [20], we enhance LSTM by incorporating transformer principles. This fusion includes self-attention and cross-attention mechanisms [21] similar to transformers, improving LSTM's ability to capture complex data dependencies, especially in large datasets. The resulting TransLSTM architecture combines LSTM and transformer strengths, making it adaptable and powerful for real-world applications and predictions. Figure 2 illustrates TransLSTM: input encoding converts tokens to continuous vectors, positional encodings provide context and positional information, transformer encoder blocks process sequences with multi-head self-attention and feedforward networks, LSTM integration captures sequential dependencies, attention mechanism combines information from both sources, and the output layer produces final predictions.

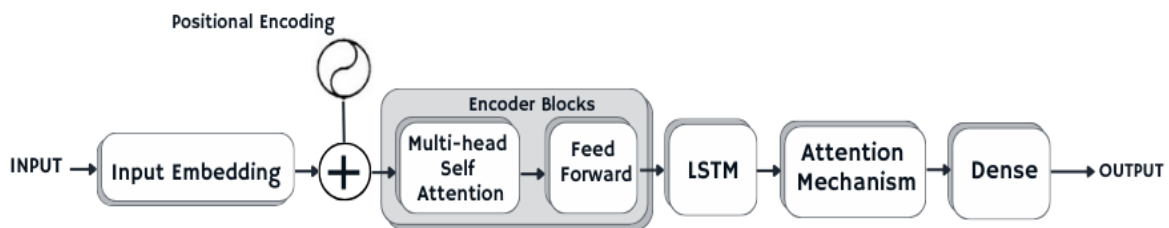


Figure 2. TransLSTM architecture

4. RESULTS AND DISCUSSION

This section outlines the comparative study conducted to assess the performance of MPLSTM. The experimental procedure employs statistical analysis to evaluate the results obtained across all datasets, comparing MPLSTM with several state-of-the-art algorithms for data stream classification. The results of this study provide important new information about the performance of the proposed MPLSTM framework and its competitive position in the field of data stream classification techniques.

4.1. Dataset

A total of 29 different time-series datasets were used in this study and came from the UCR repository, which is accessible to the public [22]. Stream clustering [23], anomaly detection [24], and data stream density estimation are just a few of the applications for which these datasets have been used in research in the past. Each dataset comprises of instances of a one-dimensional time series with a built-in grid structure.

The IMDB dataset [25], introduced by Maas *et al.* [26], is a prominent benchmark for sentiment classification. It comprises 25,000 reviews in both the training and test sets, each limited to 30 reviews per movie for diversity. This balanced dataset contains an equal number of positive and negative reviews, establishing a 50% accuracy baseline if predictions were random.

4.2. Case study 1

The study encompassed a comprehensive exploration of various established techniques, aiming to encompass all algorithm families proposed in the literature for the given problem. Table 1 provides an overview of the evaluated classifiers, organized by their respective families, and includes the abbreviations used throughout this paper [27], [28]. The results obtained from the conducted experiments are presented and discussed. Additionally, the processing time on each dataset is analyzed, considering the significance of speed-up in a data streaming scenario.

Table 1. Utilized models for case study1

Classifier	Abbreviation	Family
Naive Bayes	NaivBy	Bayesian classifiers
Adaptive Size H T	AdptSHOFT	Decision tree
Stochastic gradient-descent	StoGrdD	Function classifiers
Single classifier drift	SnglCDrft	Drift classifiers
Leveraging bagging	LvrgBag	Meta classifiers bagging
Adaptive random forest	AdptRnF	Meta classifiers bagging
Boosting using adwin	BoAdwin	Meta classifiers boosting
Multi-layer perception	MLPrecept	Neural networks

Hyperparameter selection typically involves using rule-of-thumb parameters or proven combinations from previous studies. However, a systematic approach like grid search (GS) [29] is employed for meticulous hyperparameter selection. Grid search is favored due to its simplicity, parallelizability, and effectiveness in low-dimensional spaces. It entails discretizing hyperparameter value ranges and systematically testing all possible combinations. This approach explores diverse model configurations. Before training the final models, a validation run optimizes hyperparameters based on accuracy assessments. The training process ends when the maximum epoch limit is reached. MPLSTM configuration details are summarized in Table 2. In this paper, the Adam optimizer is chosen post-validation for its computational efficiency and slightly superior test results. A batch size of 32 is used for all models, and the sparse categorical cross entropy as a loss function [30] is employed. This loss function calculates the negative logarithm of the predicted probability for the true class index when applied to class indices, showing the model's level of assurance in the accuracy of its class prediction.

Table 2. Hyperparameters used in tuning MPLSTM framework

Network Parameter	Configuration
Dense	10
Epochs number	200
Optimization function	ADAM optimizer
Size of a batch	32
Learning rate	0.001
Activation function	Softmax
Loss function	Sparse categorical cross entropy

$$\text{Sparse Categorical CrossEntropy} = -\sum_{i=1}^n y_i \log(\hat{y}_i) \quad (6)$$

where n represents the classes' number, y represents the true label or target value of the i th class, and \hat{y} represents the predicted probability for the corresponding class.

4.3. Performance evaluation

Batch sizes above 30 exhibit stable accuracy and processing times regardless of the number of batches, offering flexibility in parameter selection. However, batch sizes below 30 significantly degrade performance, hindering model adaptability to evolving data streams. Very small batch sizes overly focus on individual examples, preventing learning of overall data distribution changes. MPLSTM achieves high accuracy across various datasets, showcasing LSTM's suitability for time-series data streaming.

Convergence of training and validation loss lines during model training is a positive indicator, signifying learning progress. MPLSTM reduces processing time significantly through parallel processing, enhancing accuracy and predictive capabilities. The trade-off with increased computational time should be considered based on application requirements. Figure 3 demonstrates parallel processing consistently outperforming sequential processing across 29 datasets, ensuring MPLSTM's effectiveness.

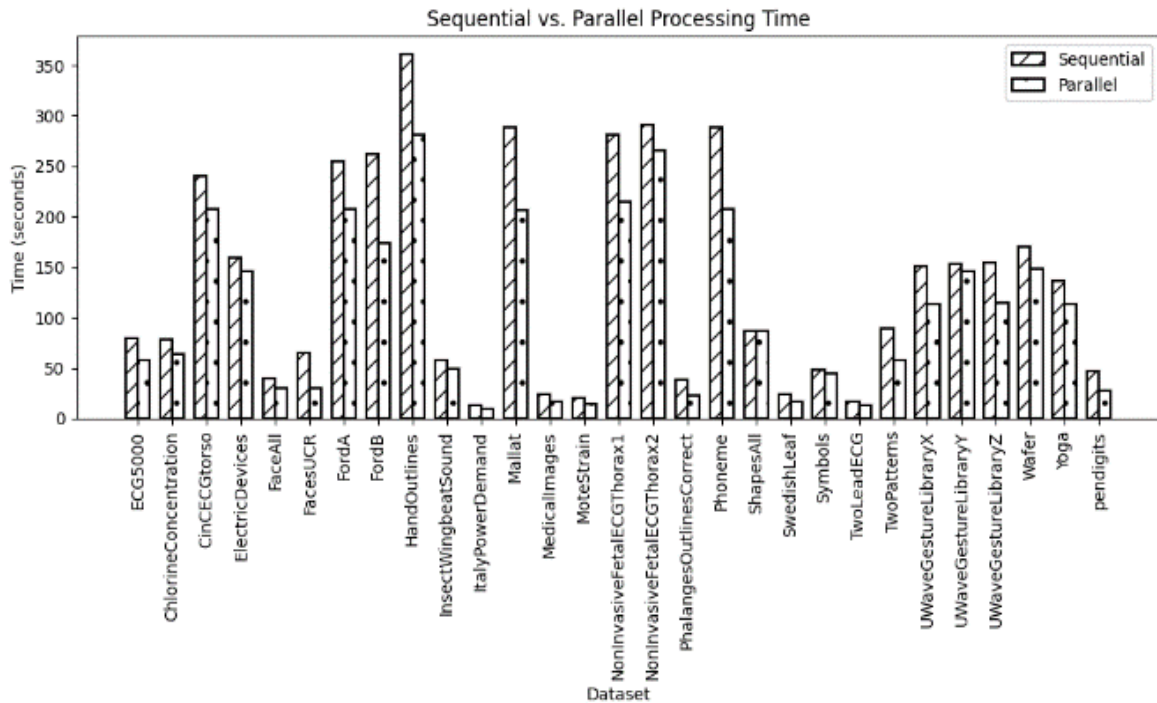


Figure 3. A comparison between sequential and parallel execution processing times

The FacesUCR dataset exhibits a speedup of 2 times when processed in parallel, indicating a significant improvement in processing time compared to sequential processing. Similarly, for the Pendigits dataset, the parallel model achieves a speedup of 1.7, further demonstrating the efficiency of parallel processing over the sequential approach in this case. Several other datasets, such as PhalangesOutlinesCorrect and TwoPatterns, also exhibit notable speedups larger than 1.5 times when processed in parallel. These findings further emphasize the effectiveness of MPLSTM in reducing processing time. The observed speedup across multiple datasets as shown in Figure 4 underscores the model's ability to leverage parallelism efficiently, resulting in faster dataset processing.

By harnessing parallel processing, the MPLSTM demonstrates its capability to significantly improve performance and expedite data analysis tasks. Upon closer analysis, the processing time of individual datasets, such as the ECG5000, demonstrates notable improvements in processing time as shown in Figure 5, although the magnitude of the speedup may not be extremely high. However, when examining the Pendigits dataset, with its larger size and increased complexity, the benefits of parallel processing become increasingly pronounced. Consequently, the speedup achieved becomes substantially larger.

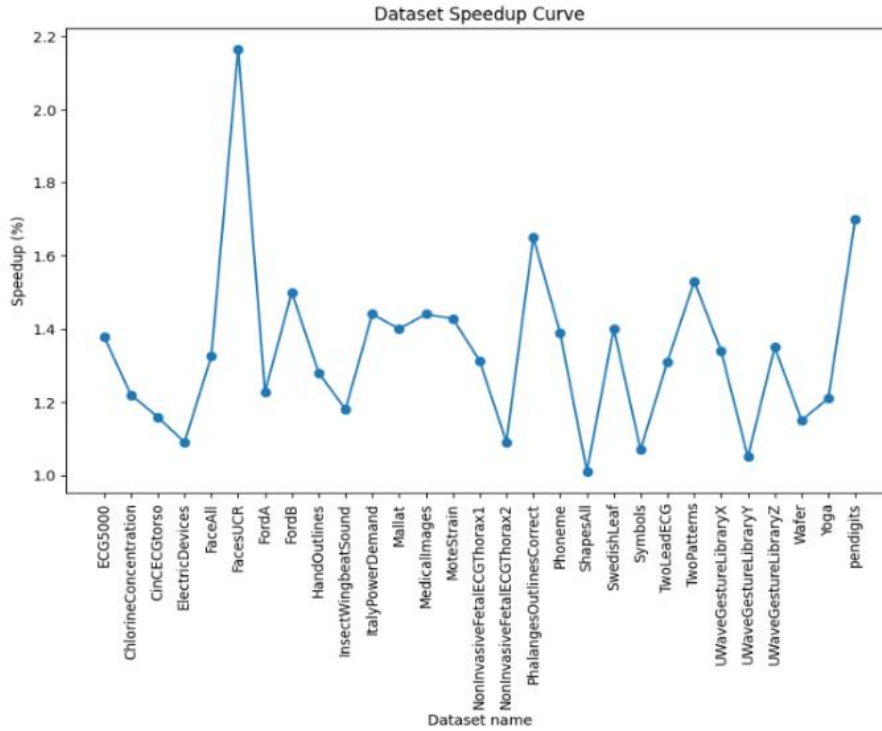


Figure 4. Dataset speedup curve when applying the MPLSTM framework

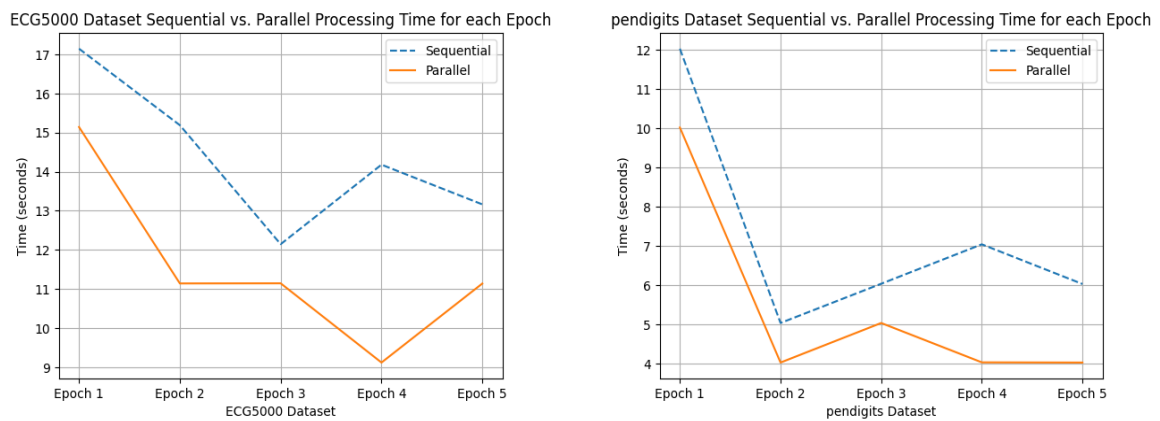


Figure 5. A Comparison of the processing time of sequential and parallel execution of two datasets ECG5000 and Pendigits

Figure 6 displays the learning curves, which depict the accuracy improvement across each epoch for two different datasets ECG5000 and Pendigits. These curves visually demonstrate how the model's accuracy increases as the training progresses. The learning curves for the two datasets show how well the model was trained and how well it could learn from the data. The consistent improvement in accuracy over the course of the epochs implies that the model is not just remembering the training data but also generalizing well to new data. This is encouraging for the model's ability to predict outcomes using fresh data.

Similarly, Figure 7 illustrates the decrease in loss across each epoch for the same mentioned datasets. All these learning curves provide crucial insights into the model's performance and offer valuable guidance for enhancing its architecture and training procedure. The learning curves provide valuable insights into the model's performance and offer guidance for optimizing its architecture and training process. By addressing overfitting and considering early stopping, the model's accuracy can be further improved while maintaining good generalization capabilities.

In Table 3, The proposed MPLSTM framework's effectiveness was assessed using MSE, RMSE, and MAE as evaluation metrics. It is desirable to have low values for these metrics as they indicate better performance. In this study, the framework yielded promising results with low values for example, it gives MSE=0.237 for the ECG5000, RMSE=0.583 for the PhalangesOutlinesCorrect, and MAE=0.074 for the pendigits. This implies that the predictions made by the MPLSTM model were close to the actual values. The model performed well because it was able to learn the patterns and relationships in the data. This led to accurate predictions and shows that the MPLSTM framework is an effective way to address this problem. Table 4, The accuracy table showcases the performance evaluation results for MPLSTM on the UCR dataset. It provides a comprehensive overview of the accuracy achieved by the framework in predicting the target variable. The table offers a detailed breakdown of the accuracy scores across different metrics or experimental configurations, allowing for a comprehensive analysis of the framework's performance. Researchers and practitioners can refer to this table to assess the effectiveness and reliability of MPLSTM in accurately predicting the target variable on the UCR dataset.

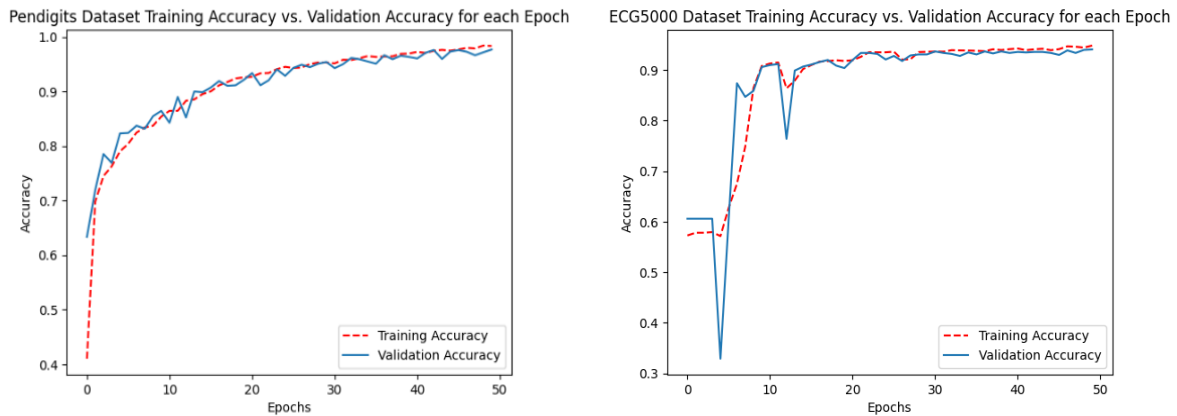


Figure 6. The Accuracy curves of training and validation sets in two datasets ECG5000 and Pendigits

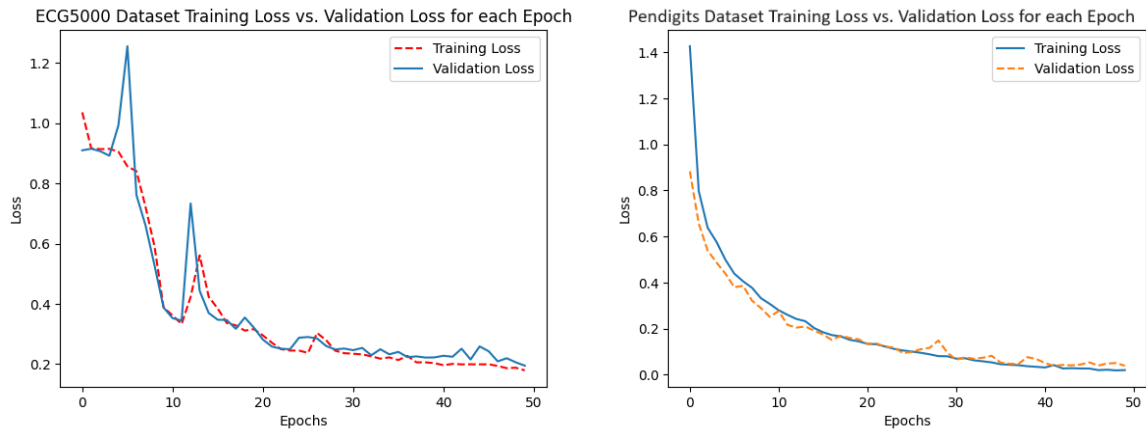


Figure 7. The loss curves of training and validation sets of two datasets ECG5000 and Pendigits

Table 3. Performance of MPLSTM in terms of MSE, RMSE, and MAE

Dataset	MSE	RMSE	MAE
Wafer	0.449	0.670	0.2247
Pendigits	0.390	0.625	0.074
ECG5000	0.237	0.478	0.113
HandOutlines	0.361	0.601	0.361
PhalangesOutlinesCorrect	0.340	0.583	0.340
ChlorineConcentration	1.11	1.0536	0.340

Table 4. Accuracy of the top 7 classifiers for the UCR datasets compared with the MPLSTM framework

Dataset	Proposed DPMLSTM	AdaptRnf	MLPrecept	NaivBy	SnglCDrft	AdaptSHOFT	LvrgBag	BoAdwin	StoGrdD
Wafer	0.100	0.982	0.991	0.194	0.192	0.356	0.963	0.960	0.542
Pendigits	0.976	0.950	0.938	0.824	0.784	0.850	0.867	0.909	0.800
ECG5000	0.941	0.856	0.877	0.750	0.772	0.750	0.752	0.843	0.833
ElectricDevices	0.85	0.526	0.526	0.456	0.456	0.456	0.468	0.457	0.194
HandOutlines	0.638	0.720	0.634	0.533	0.533	0.533	0.530	-0.084	0.475
PhalangesOutlinesCorrect	0.659	0.377	0.060	0.134	0.134	0.133	0.245	0.277	0.072
ChlorineConcentration	0.533	0.149	0.082	0.122	0.122	0.001	0.063	0.001	0.099

4.5. Case study 2

In this study, LSTM is integrated and the Transformer model to create TransLSTM, a novel architecture. TransLSTM leverages the Transformer's success in handling sequential data and capturing long-range dependencies. This fusion enhances LSTM's ability to model complex relationships and temporal dependencies in sequential data by incorporating self-attention and cross-attention mechanisms from the Transformer. The investigation demonstrates how TransLSTM can address LSTM's limitations, potentially leading to more accurate and efficient predictions. This case study highlights the innovative potential of combining diverse neural architectures for enhanced predictive capabilities.

4.6. TransLSTM evaluation

The training history curves in the provided case study offer insights into the performance of two different models, LSTM and TransLSTM, across multiple epochs. In Figure 8, the first and the third curves represent training loss for LSTM and TransLSTM, while the second and fourth curves represent validation loss for LSTM and TransLSTM, respectively. These curves depict the evolution of training loss over epochs, showing a decreasing trend, and indicating learning from the training data. TransLSTM consistently achieves lower training loss and outperforms LSTM in validation loss, indicating better generalization to new data. Similarly, the other figure displays training and validation accuracy curves, with the first and the third curves representing training accuracy for LSTM and TransLSTM, and the second and the fourth curves representing validation accuracy. Both models exhibit an increasing trend in training accuracy, demonstrating efficient learning from the training data as well as the capacity to generalize to new, untried data. TransLSTM achieves higher training and validation accuracy, highlighting its superior data modeling capabilities.

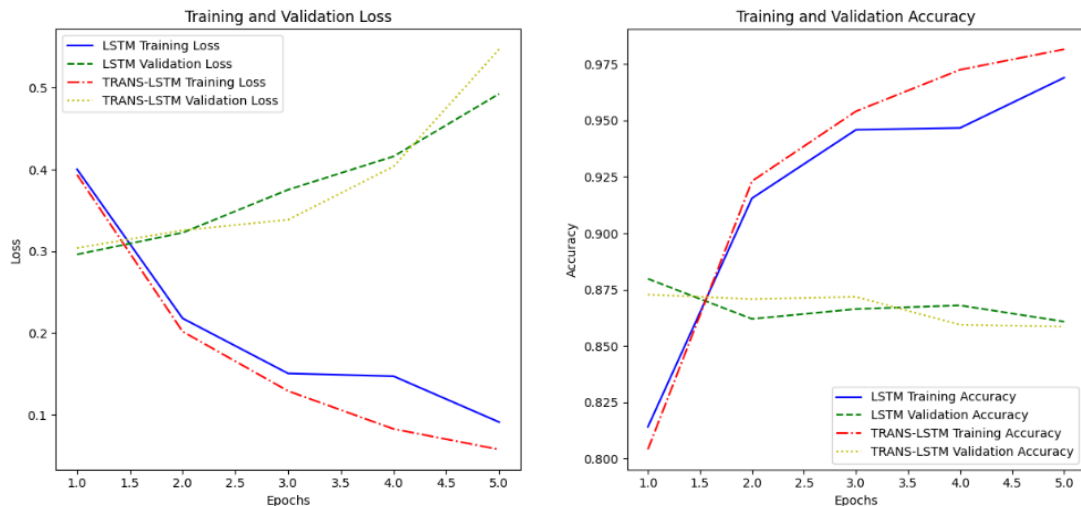


Figure 8. Comparison between LSTM and TransLSTM training and validation loss and accuracy

5. CONCLUSION




In conclusion, this paper presents a novel framework for datastream regression, referred to as MPLSTM. The proposed framework effectively addresses the challenges associated with handling continuous and large-scale data in real-time prediction scenarios. By leveraging the inherent parallelism of LSTM networks, MPLSTM achieves a remarkable balance between high prediction accuracy and

computational efficiency. Experimental evaluations, conducted on real-world datasets including the UCR dataset, validate the superior performance of MPLSTM compared to traditional regression models. The framework's ability to capture temporal dependencies and long-term patterns in streaming data is demonstrated through accurate predictions, as evidenced by accuracy measures and loss calculations. MPLSTM emerges as a promising approach for datastream prediction, showcasing improved performance and outperforming existing results in terms of accuracy and loss.




REFERENCES

- [1] S. Bharany *et al.*, "A comprehensive review on big data challenges," in *2023 International Conference on Business Analytics for Technology and Security (ICBATS)*, Mar. 2023, pp. 1–7, doi: 10.1109/ICBATS57792.2023.10111375.
- [2] S. Homayoun and M. Ahmadzadeh, "A review on data stream classification approaches," *Journal of Advanced Computer Science and Technology*, vol. 5, no. 1, Feb. 2016, doi: 10.14419/jacst.v5i1.5225.
- [3] S. Ray, "A quick review of machine learning algorithms," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Feb. 2019, pp. 35–39, doi: 10.1109/COMITCon.2019.8862451.
- [4] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018, doi: 10.1109/ACCESS.2017.2779939.
- [5] S. Smyl and K. Kuber, "Data preprocessing and augmentation for multiple short time series forecasting with recurrent neural networks," *36th International Symposium on Forecasting*, 2016.
- [6] I. O. Muraina, "Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts," in *7th International Mardin Artuklu Scientific Researches Conference*, 2022, pp. 496–504.
- [7] J. Hunt, "Multiprocessing," in *Advanced Guide to Python 3 Programming*, Springer International Publishing, 2019, pp. 363–376.
- [8] Z. A. Aziz, D. Naseradeen Abdulqader, A. B. Sallow, and H. Khalid Omer, "Python parallel processing and multiprocessing: a rivew," *Academic Journal of Nawroz University*, vol. 10, no. 3, pp. 345–354, Aug. 2021, doi: 10.25007/ajnu.v10n3a1145.
- [9] X. Liang *et al.*, "R-Drop: regularized dropout for neural networks," *Advances in Neural Information Processing Systems*, vol. 13, pp. 10890–10905, 2021.
- [10] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, no. 2, Feb. 2019, doi: 10.1088/1742-6596/1168/2/022022.
- [11] N. Watt and M. C. du Plessis, "Dropout for recurrent neural networks," in *Proceedings of the International Neural Networks Society*, Springer International Publishing, 2020, pp. 38–47.
- [12] A. Zunino, S. A. Bargal, P. Morerio, J. Zhang, S. Sclaroff, and V. Murino, "Excitation dropout: encouraging plasticity in deep neural networks," *International Journal of Computer Vision*, vol. 129, no. 4, pp. 1139–1152, Jan. 2021, doi: 10.1007/s11263-020-01422-y.
- [13] D. Jha, A. Yazidi, M. A. Riegler, D. Johansen, H. D. Johansen, and P. Halvorsen, "LightLayers: parameter efficient dense and convolutional layers for image classification," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12606, Springer International Publishing, 2021, pp. 285–296.
- [14] P. Lara-Benítez, M. Carranza-García, D. Gutiérrez-Avilés, and J. C. Riquelme, "Data streams classification using deep learning under different speeds and drifts," *Logic Journal of the IGPL*, vol. 31, no. 4, pp. 688–700, Jul. 2023, doi: 10.1093/jigpal/jzac033.
- [15] O. Du, Y. Zhang, X. Li, J. Zhu, T. Zheng, and Y. Li, "Multi-view heterogeneous network embedding," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13369 LNAI, Springer International Publishing, 2022, pp. 3–15.
- [16] Z. Zhang, "Improved Adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, Jun. 2018, pp. 1–2, doi: 10.1109/IWQoS.2018.8624183.
- [17] Y. Gao, W. Liu, and F. Lombardi, "Design and implementation of an approximate softmax layer for deep neural networks," *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, Seville, Spain, 2020, pp. 1–5, doi: 10.1109/iscas45731.2020.9180870.
- [18] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, no. 3, pp. 317–346, Oct. 2013, doi: 10.1007/s10994-012-5320-9.
- [19] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature," *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247–1250, Jun. 2014, doi: 10.5194/gmd-7-1247-2014.
- [20] A. Vaswani *et al.*, "Attention is all you need," *arXiv:1706.03762*, Jun. 2017.
- [21] Z. Huang, P. Xu, D. Liang, A. Mishra, and B. Xiang, "TRANS-BLSTM: transformer with bidirectional LSTM for language understanding," *arXiv:2003.07000*, Mar. 2020.
- [22] Y. Chen *et al.*, "The UCR time series classification archive," *NSF*, Jul. 2015, https://www.cs.ucr.edu/~eamonn/time_series_data/ (accessed Jul. 13, 2023)
- [23] A. Bifet, G. De Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2015, pp. 59–68, doi: 10.1145/2783258.2783372.
- [24] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, May 2012, doi: 10.1109/TKDE.2010.235.
- [25] IMDb, "Internet movie database," *IMDb datasets*. <https://datasets.imdbws.com/> (accessed Jul. 13, 2023).
- [26] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, vol. 1, pp. 142–150.
- [27] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.
- [28] P. Fabian *et al.*, "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [29] B. H. Shekar and G. Dagnew, "Grid search-based hyperparameter tuning and classification of microarray cancer data," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, Feb. 2019, pp. 1–8, doi: 10.1109/ICACCP.2019.8882943.
- [30] S. Mannor, D. Peleg, and R. Rubinstein, "The cross entropy method for classification," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 561–568, doi: 10.1145/1102351.1102422.




BIOGRAPHIES OF AUTHORS

Nada Adel Dief    is a computer engineer interested in big data, deep learning, machine learning, and text mining. She received her master of science degree from Mansoura University in 2016 and is currently working as a teaching assistant for the Faculty of Engineering Computer and System Department, Department of Computer Engineering and Systems, Faculty of Engineering, Mansoura University, Mansoura, Egypt. She can be contacted at email: nadadief@mans.edu.eg.






Mofreh Mohamed Salem    received his Ph.D. degree from Strathclyde University, U.K., in 1985. He was the director of the Software Engineering Unit, Faculty of Engineering, from 2001 to 2006. He was the head of the Computers Engineering and Control Department, Faculty of Engineering, Mansoura University, Egypt, from 2004 to 2008, where he is currently a member of the Computer Center Council. He was the Dean of the High Institute for Computers in Mansoura, from 2008 to 2011. He has published 92 scientific articles in international journals periodicals and conferences of computer engineering. His current research interests include software engineering, computer systems design, parallel processing, computer networks, cloud computing, and big data. Department of Computer Engineering and Systems, Faculty of Engineering, Mansoura University, Mansoura, Egypt. He can be contacted at email: dr_mofreh@mans.edu.eg.



Asmaa Hamdy Rabie    received a B.Sc. in computers and systems engineering, with a general grade of excellent with class honors in 2013. She got her master's degree in the area of load forecasting using data mining techniques in 2016 at the Computers Engineering and System Department, Mansoura University, Egypt. She got her Ph.D. degree in load forecasting using data mining techniques in 2020 at Computers Engineering and System Department, Mansoura University, Egypt. Her interests (programming languages, classification, big data, data mining, healthcare systems, and the internet of things), she is currently a lecturer in the faculty of Engineering, at Mansoura University, Egypt. She can be contacted at email: asmaa91hamdy@yahoo.com.



Ali Ibrahim EL-Desouky    holds received his MSc and Ph.D. degrees from the University of Glasgow, USA. He is currently a full professor with the Computers Engineering and Systems Department, Faculty of Engineering, Mansoura University, Egypt. He is also a visiting part-time professor with MET Academy. He also teaches at American and Mansoura universities and has taken over many positions of leadership and supervision of many scientific articles. He has published hundreds of articles in well-known international journals. Department of Computer Engineering and Systems, Faculty of Engineering, Mansoura University, Mansoura, Egypt. Email: adesoky@mans.edu.eg.