An Implementation of Traffic Volume Forecasting System using Distributed Deep Learning

Kim-Chi Le[*,†], Trong-Hop Do[*,†], Quang Dieu Tran[‡], Dinh-Thuan Do[§], Hong-Anh Le[¶], Nhu-Ngoc Dao[‖]

[*] University of Information Technology, Ho Chi Minh City, Vietnam.
[†] Vietnam National University, Ho Chi Minh City, Vietnam.
[‡] Ho Chi Minh National Academy of Politics, Hanoi, Vietnam.
[§] Electrical Engineering Department, University of Colorado Denver, Denver, CO 80204, USA.
[¶] Faculty of Information Technology, Hanoi University of Mining and Geology, Vietnam.
[‖] Department of Computer Science and Engineering, Sejong University, Seoul, South Korea.

***Abstract*—The increase of means of transport, on the one hand, makes traveling easy and convenient, and unintentionally makes traffic congestion more and more serious, especially in big cities. Traffic congestion has a direct and indirect impact on a country's economy and its dwellers' health. Traffic congestion effects on individual level as well. Time loss, especially during peak hours, mental stress, and the added pollution to the global warming are also some important factors caused due to traffic congestion. In recent years, traffic volume prediction has received lots of interest from researchers. However, there are still many challenges that needs to be resolved. Models for traffic volume prediction needs not only to be accurate but also be able to handle the huge volume of input traffic data. This study present an implement of a traffic volume forecasting system using distributed deep learning. A state of the art deep learning model, that is LSTM, is used to provide accurate forecasting result. Especially, the model is trained in a distributed manner using BigDL library. Thank to the distributed training, the model now is able to receive a huge volume of traffic data to provide the most accurate results.**

***Index Terms*—Traffic Volume, Time Series, Distributed Computing, Deep learning**

I. Introduction

Traffic Volume Forecasting is the process of creating predictive data about traffic density in one or more areas in the future, serving to make the result for traffic density. There are many forecasting methods used to estimate future vehicle traffic forecast, from purely subjective judgments to complex methods, which require large amounts of data, and are expensive.

The problem of predicting traffic density at specific road frames and time frames will contribute to minimizing congestion and making the most of resources. Traffic users can rely on the expected traffic density to choose the appropriate travel frame, avoid crowded places and move along the route with fewer vehicles. If traffic density is predictable in advance, traffic dispatchers will also be allocated locations more efficiently. Points predicted to be heavily congested will receive more resources than others. This problem will be based on predicted information to make recommendations for traffic users to make the most of the infrastructure and reduce the time affected by traffic jams and congestion to the lowest level.

There are two major obstacles when solving the Traffic Volume Forecasting problem. To begin, the time-series forecasting is a arduous problem in Data Science [1], traditional machine learning based method often fail to achieve high result due to the time series property in the data [2]. Furthermore, the Traffic Volume Forecasting problem needs to be perform on a large scale dataset. To illustrate, the forecasting system will need to make predictions for thousands of locations on the map of a city simultaneously. For each prediction, a large number of data need to be processed. Current research concentrate only on building a predictive model at a certain location, not on designing a model that can be applied on a large scale dataset

In this research, a Traffic Volume Forecasting system that is based on distributed deep learning method is proposed to tackle the above difficulties. The deep learning model used in the study is LSTM which is commonly utilized to solve the time series forecasting problem with high accuracy [3]. The special thing in the proposed system is that the LSTM model is implemented using distributed computing technique so the system can perform well even the amount of input data increases. Hence, the proposed system can give high accuracy an can be implemented on a large-scale dataset.

II. Fundamental and related works

A. Algorithms for time series forecasting

1) Machine learning based: A variety of industries employ autoregressive integrated moving average (ARIMA) models [4]. It is frequently applied to demand forecasting, such as when predicting future demand for the production of food. This is so that managers have solid rules to follow when making decisions on supply chains. On the basis of previous prices, ARIMA models can also be used to forecast the future price of your stocks. A wide class of models called ARIMA models is employed to forecast time series data. The standard abbreviation for ARIMA models is ARIMA (p,d,q), where p denotes the order of the moving-average model, d the degree of differencing, and q the order of the autoregressive model. ARIMA models transform a non-stationary time series into a stationary one via differencing and then extrapolate future values from the past. In order to predict future values, these

models employ "auto" correlations and moving averages over residual errors in the data.

In this research, we did not choose the ARIMA model because of two reasons. First, as we mentioned in the Introduction section, the Traffic Volume Forecasting problem requires a large-scale dataset. Therefore, ARIMA model is not an ideal choice for this problem. Second, before implementing the ARIMA model, we need to calculate three parameters p, d, and q then provide them to ARIMA model while LSTM does not require calculating such parameters. As a result of the above reasons, we prefer to choose LSTM model as the solution for the Traffic Volume Forecasting problem.

2) Deep learning based: Today, different Deep Learning-based algorithms are used to handle different types of data. One of the most difficult types of data to handle and forecast is sequential data. Sequential data is different from other types of data in the sense that while all the features of a typical dataset can be assumed to be order-independent, this cannot be assumed for a sequential dataset [5]. To handle such type of data, the concept of Recurrent Neural Networks (RNN) was conceived. Long Short-Term Memory (LSTM) [6] networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information [7]. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying on the basis of time-series data.

Logic gates in a computer are maybe the design inspiration for LSTM. The LSTM introduces a memory cell, or cell for short, that is designed to store additional information and has the same shape as the hidden state (some literatures refer to the memory cell as a particular type of the hidden state). We require a number of gates in order to regulate the memory cell. To read out the entries from the cell, one gate is required. The output gate will be the name we use for this. To determine when to read data into the cell, a second gate is required. This is known as the input gate. The cell's content needs to be reset, and this method must be controlled by a forget gate.

As shown in Fig. 1, the input at the current time step and the concealed state of the previous time step are the data entering into the LSTM gates. To determine the values of the input, forget, and output gates, three completely connected layers with sigmoid activation functions process them. As a result, the three gates' values fall between (0,1).

Assume mathematically that there are ***h*** hidden units, ***n*** is the batch size, and ***d*** is the total number of inputs. Thus, the hidden state from the previous time step is $H_{t-1} \in \mathbb{R}^{n\times h}$, and the input is $X_t \in \mathbb{R}^{n\times h}$. Accordingly, the input gate is designated as $I_t \in \mathbb{R}^{n\times h}$, the forget gate as $F_t \in \mathbb{R}^{n\times h}$, and the output gate as $O_t \in \mathbb{R}^{n\times h}$ at time step t. The value of F_t, I_t, and O_t are calculated as in Eq. 1, 2, 3:

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \tag{1}$$

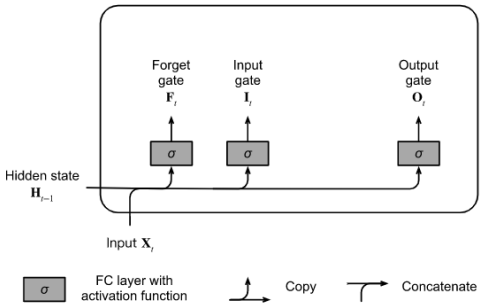

Fig. 1: Computing the input gate, the forget gate, and the output gate in an LSTM model [6]

$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \tag{2}$$

$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \tag{3}$$

In the equations above, the weight parameters are $W_{xf}, W_{xi}, W_{xo} \in \mathbb{R}^{d\times h}$ and $W_{hf}, W_{hi}, W_{ho} \in \mathbb{R}^{h\times h}$, the bias parameters are $b_f, b_i, b_o \in \mathbb{R}^{1\times h}$.

Next, we will take a look at the candidate memory cell in LSTM model, it is illustrated in Fig. 2. The candidate

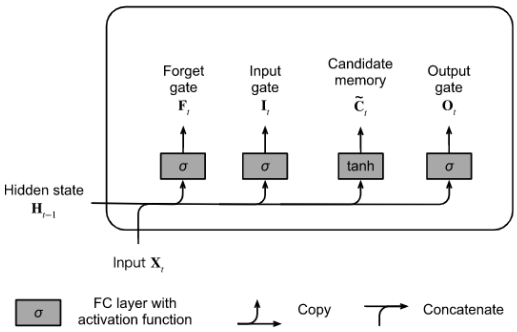

Fig. 2: Computing the candidate memory cell in an LSTM model [6].

memory cell $\tilde{C} \in \mathbb{R}^{n\times h}$ is then introduced. Its computation is comparable to that of the three gates discussed above, but instead uses a tanh function with an activation function that has a value range of (1,1) instead. The value of $\tilde{C}$ is calculated as in Eq. 4:

$$\tilde{C} = tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \tag{4}$$

The weight parameters are $W_{xc} \in \mathbb{R}^{d\times h}$ and $W_{hc} \in \mathbb{R}^{h\times h}$, $b_c \in \mathbb{R}^{1\times h}$ is bias parameter.

The illustration of the computing the memory cell in an LSTM model is shown in Fig. 3. We have two specific gates in LSTMs for these uses: the entrance gate It controls how much of the old memory cell content—Ct1RnH—we reserve via Ct, and the forget gate Ft controls how much of it we forget. We

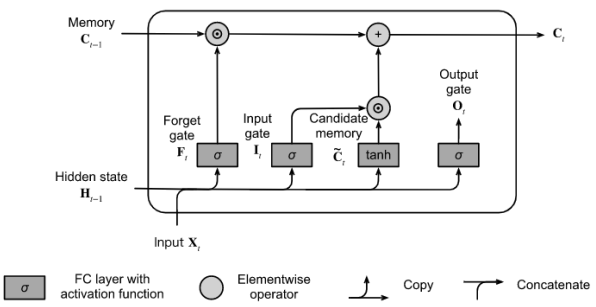

Fig. 3: Computing the memory cell in an LSTM model [6].

arrive at the following update equation as the equation 5 shown below by employing the pointwise multiplication.

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C} \tag{5}$$

The prior memory cells C_{t-1} will be saved over time and sent to the current time step if the forget gate is always roughly 1 and the input gate is always roughly 0. The vanishing gradient issue is addressed by this approach, which also enhances the ability to identify long-range connections within sequences.

Last, we dissect the hidden state $H_t \in \mathbb{R}^{n \times h}$ of a LSTM model (as shown in Fig. 4). The output gate is used at this time. It is merely a gated variation of the memory cell's *tanh* in LSTM. This guarantees that the values of H_t are always within the range (-1,1). We can defined the hidden state as the

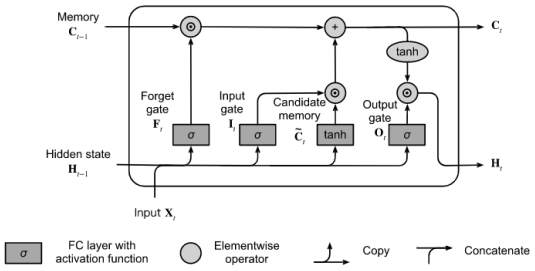

Fig. 4: Computing the hidden state in an LSTM model [6].

equation 6:

$$H_t = O_t \odot tanh(C_t) \tag{6}$$

When the output gate is close to 1, we essentially transfer all memory data to the predictor; however, when it is close to 0, we only keep the data inside the memory cell and don't process it further.

B. Distributed learning

Thanks to the help from Apache Spark, we tackled the large-scale data problem. When used alone or in conjunction with other distributed computing tools, Apache Spark is a data processing framework that can quickly conduct operations on very large data sets and distribute operations across several machines. These two characteristics are essential to the fields of big data and machine learning, which call for the mobilization of enormous computer power to process vast data warehouses. With an intuitive API that abstracts away most of the tedious labor of distributed computing and big data processing, Spark also relieves developers of some of the programming responsibilities associated with these activities [8].

A driver transforms user code into many tasks that can be distributed across worker nodes, while executors operate on those nodes and carry out the tasks assigned to them. These two major parts make up an Apache Spark application at its most basic level. To act as a liaison between the two, some kind of cluster manager is required.

Apache Spark offers us a framework in the BigDL library, Chronos, which is an application framework for building large-scale time series analysis applications.Along with several data processing and feature engineering tools, Chronos includes a number of built-in deep learning and machine learning models for time series forecasting [9], detection, and simulation. To get the greatest flexibility, users can directly call standalone algorithms and models (Forecasters, Detectors, Simulators), or they can use a highly integrated, scalable, and automated process for time series (AutoTS).

The architecture of Chronos framework is illustrated in Fig. 5, which is on top of BigDL and Ray (Ray is a framework for creating cutting-edge AI applications with straightforward and common APIs). Ray Tune serves as a hyper-parameter search engine for the AutoTS framework (running on top of RayOnSpark). The search engine chooses the optimum lookback value for a prediction task during automatic data processing. The search engine chooses the optimal subset of features from a set of features that are automatically generated by various feature creation tools for auto feature engineering (e.g., tsfresh). The search engine looks for hyper-parameters like hidden dim, learning rate, etc. for auto modeling.

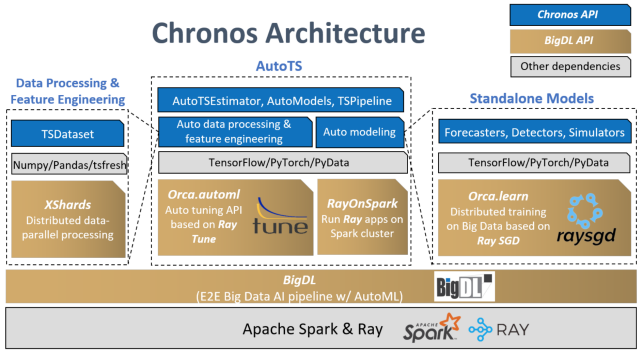

Fig. 5: Chronos architecture.

III. Distributed Learning

The proposed method for this study is illustrated in Fig. 6. We begin with preparing the dataset like in the "Dataset description" section. Before training the LSTM model, we will apply standard scaling to the dataset to make it has zero-mean and unit variance. Doing so will help increase the speed and

efficiency of the training process. After scaling the dataset, we will feed the LSTM model with the scaled data and start the training process. We will train the model with 6 different epochs; 5, 10, 25, 50, 100, and 1000, for each epoch we will choose the sequence length of 10. When the training process finishes, we will then compare the result to find out with which epoch the model will give us the best results.

Fig. 6: Proposed system.

Furthermore, the dataset contains numerous amount of data points, and it is getting bigger and bigger day by day. Therefore, to handle that problem, we applied BigDL in Apache Spark to help us process and train the model when the dataset becomes big. In this work, we will use Chronos framework of Orca library in BigDL to build a large-scale time series analysis model. Chronos offers AutoTSEstimator as a highly integrated time series forecasting solution with auto feature selection, auto hyperparameter tuning, and auto preprocessing. we will use the dataset that was prepared in the "Dataset Description" section as input data of the AutoTSEstimator. A TSPipeline containing the best model and pre/post data processing will be returned for further development of deployment by building an AutoTSEstimator and running "fit" function on the data.

IV. Experiment

A. Experimental procedure and environment

The experimental procedure (as shown in Fig. 7) that was used to solve the Traffic Volume Forecasting is proposed in this section. First of all, we downloaded the dataset from the TII website. The dataset originally included 11 features, but only three features are informative, which are "Date", "Time", and "All", so in the data preprocessing stage, we decided to delete other features that are uninformative. Subsequently, in the data

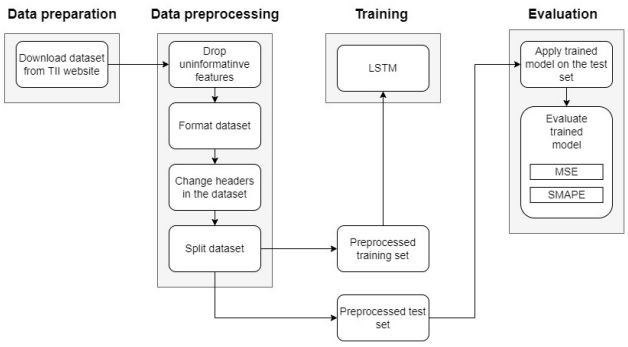

Fig. 7: Traffic Volume Experimental Procedure.

preprocessing stage, we format the dataset by concatenating the "Date" feature with the "Time" feature to create a new feature named "timestamp" and changing its data type to date time format. Then, we change the "All" feature to the "value" feature. After that, we split the dataset into a training set and a test set for the next stages.

Next, we trained the LSTM model in the training stage for predicting traffic volume. After finishing the training process, we applied the test set to the trained LSTM model, and then we used MSE and SMAPE metrics to evaluate the model. The Mean Squared Error (MSE) [10] is arguably the most straightforward and typical loss function. The MSE is calculated by taking the difference between the predictions made by your model and the actual data, squaring it, and averaging it over the entire dataset. Since we are always squaring the errors, the MSE can never be negative. Equation 7 provides the definition of the MSE:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{7}$$

where N is the number of samples, y_i and $\hat{y}_i$ are the exact and predicted results, respectively.

A traditional evaluation metric for "predicted value and actual value" is called symmetric mean absolute percentage error (SMAPE) [11]. The value of SMAPE is calculated as in Eq. 8.

$$SMAPE = \frac{1}{n} \times \sum_{t=1}^{n} \frac{|F_t - A_t|}{(A_t + F_t)/2}, \tag{8}$$

where A_t denotes the current value and F_t denotes the predicted value. The sum of the absolute values of the actual value A_t and the predicted value F_t is divided by half to determine the absolute difference between A_t and F_t. For each fitted point t, the value of this computation is added, then divided by the total number of fitted points n.

We trained the LSTM model on Google Colab virtual machine having CPU and GPU specifications as shown in the table 1 and table 2.

Parmeter	Specification
CPU model name	Intel(R) Xeon(R)
CPU freq	2.30GHz
No. CPU cores	2
CPU family	Haswell
Available RAM	12GB
Disk space	25GB

TABLE I: Google Colab VM's CPU specification [12]

Parmeter	Specification
GPU	Nvidia K80 / T4
GPU Memory	12GB / 16GB
GPU Memory Clock	0.82GHz / 1.59GHz
Performance	4.1 TFLOPS / 8.1 TFLOPS
No. CPU Cores	2
Available RAM	12GB

TABLE II: Google Colab VM's GPU specification [12]

B. Data set

It is known that Ireland Governments turn to Advanced Traffic Management Systems to solve traffic congestions and

adopt new transport management plans and utilize transport resources. Unfortunately, major cities are still waiting for traffic to be resolved, where Santry is ranked number one city in Ireland with major problems in it. City Departments are interested in improving traffic situation and therefore adopt information from popular navigation platforms to understand and analyze current situation. The selected dataset is a dataset containing traffic congestion data that is aggregated daily and shared publicly. The data set contains information about the time and the total number of vehicles on the M50 motorway statistics from time to time.

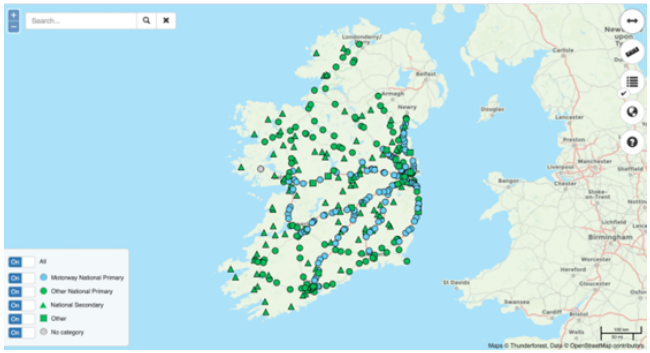

Fig. 8: Transport Infrastructure Ireland Traffic Data website.

In this study, we collected data from Transport Infrastructure Ireland Traffic Data website [13] shown in Fig. 8. The TII Traffic Data website presents data collected from the TII traffic counters located on the road network. The Website uses a dynamic mapping interface to allow the User to access data in a variety of report, graph, and data export format.

The dataset that was collected for this work is about the traffic count at M50 motorway, which is shown in Fig. 9 in Ireland from 1/1/2022 to 5/7/2022, and on each date, the traffic count was reported one time every 1 hour.

Fig. 9: M50 motorway in Ireland.

The dataset originally included 11 features: Date, Time, All, N, S, Southbound 1, Southbound 2, Southbound 3, Northbound 1, Northbound 2, and Northbound 3 (shown in figure 3). However, we just keep the informative features which are "Date", "Time", and "All", then we delete other features. In addition, we change the type of "Date" feature to date format (dd/mm/yyyy) and concatenate the "Date" column with the "Time" column and change the header to "timestamp", we change the header of "All" column to "value" as well.

C. Experimental result

We trained the model over 6 epochs (5, 10, 25, 50, 100, and 1000), the prediction result of the LSTM model for each epoch is shown in Fig. 10

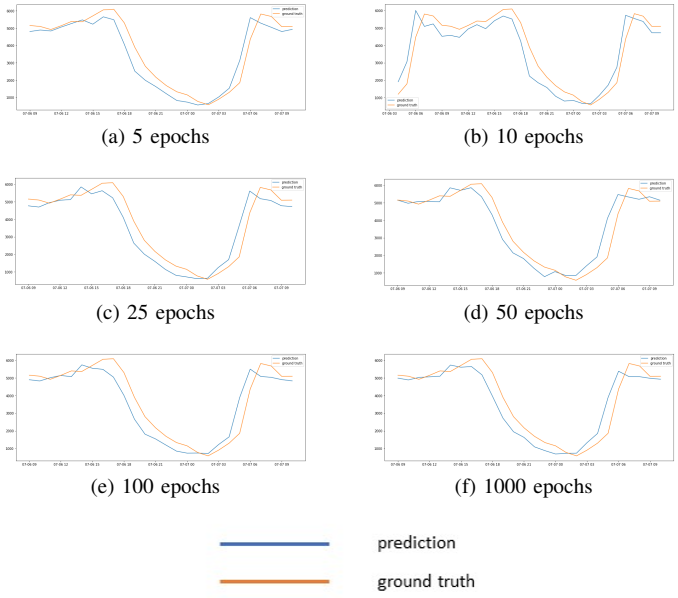

Fig. 10: Forecasting result for epoch 5, 10, 25, 50, 100 and 1000.

As figure 6 shown, there was no significant difference in the result from the 25th epoch to the 1000th epoch. In addition, for each epoch, the model can learn the trend of the dataset, it can predict at what time the traffic count is highest or lowest. We also use 2 metrics which are MSE and SMAPE to evaluate the model. As shown in table 3, it seems that at the 100th epoch, the model reached the best result, since the value of MSE and SMAPE are lowest at the 100th epoch in comparison with the others.

Epoch	MSE	SMAPE
5	167068.56	5.48
10	141097.06	5.81
25	110991.89	4.43
50	101280.75	4.31
100	**94263.12**	**3.86**
1000	96700.38	4.06

TABLE III: MSE and SMAPE for each epoch.

We have just trained the model with a univariate dataset. However, we will research and add more features to the dataset such as weather, location, etc., to make it become a multivariate dataset. Hence, the model can learn more information and give us a better result. Moreover, our model just only forecasts the traffic count for one area, so we will

develop the model further in the future to make it able to predict the number of vehicles for more locations.

V. Conclusion

Traffic volume forecasting is a essential part of an intelligent transport system which facilitates traffic management and helps reduce the negative impact of traffic congestion on social and economy. There are three major challenges when building a traffic volume forecasting system. First, the model used in the system must have the ability to learn complex features of time series data to provide accurate prediction of traffic volume. Second, the model need to be trainable with huge amount of traffic data. Third, the model need to be integrable to big data framework so that the system can process a huge amount of data required for this type of application. To deal with these challenges, this study present an implementation of a traffic volume forecasting system using deep learning and big data technology. LSTM, which is a state of the art deep learning model for time series prediction is used to provide accurate result. The LSTM model is trained in a distributed way using BigDL framework. Thanks to BigDL, the model can be trained with huge amount of traning data. Furthermore, the trained model can be integrated into Spark, which is a Big Data framework to give the system the capability of processing huge amount of data required for traffic volume forecasting problem.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1G1A1008105).

References

[1] H. Yatish and S. Swamy, "Recent trends in time series forecasting–a survey," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 04, pp. 5623–5628, 2020.

[2] J. Shi, M. Jain, and G. Narasimhan, "Time series forecasting (tsf) using various deep learning models," *arXiv preprint arXiv:2204.11115*, 2022.

[3] S. Elsworth and S. Güttel, "Time series forecasting using lstm networks: A symbolic approach," *arXiv preprint arXiv:2003.05672*, 2020.

[4] "Understanding arima models for machine learning." [Online]. Available: https://www.capitalone.com/tech/machine-learning/understanding-arima-models/

[5] R. Adhikari and R. K. Agrawal, "An introductory study on time series modeling and forecasting," *arXiv preprint arXiv:1302.6613*, 2013.

[6] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.

[7] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, no. 3. Makuhari, 2010, pp. 1045–1048.

[8] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. USA: USENIX Association, 2010, p. 10.

[9] W. Du, J. Deng, K. Huang, S. Yu, and S. Huang, "From ray to chronos: Build end-to-end ai use cases using bigdl on top of ray," Apr 2022. [Online]. Available: https://www.codeproject.com/Articles/5330192/From-Ray-to-Chronos-Build-end-to-end-AI-use-cases

[10] G. Seif, "Understanding the 3 most common loss functions for machine learning regression," Feb 2022. [Online]. Available: https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3

[11] "Symmetric mean absolute percentage error," Dec 2020. [Online]. Available: https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error

[12] "How to do deep learning research with absolutely no gpus - part 2." [Online]. Available: https://kazemnejad.com/blog/how_to_do_deep_learning_research_with_absolutely_no_gpus_part_2/

[13] [Online]. Available: https://trafficdata.tii.ie/publicmultinodemap.asp