DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

# DLSTM-MSF: Distributed LSTM Models for Multimedia Streaming Workload Forecasting Based on Kafka Environment

Saja Dheyaa Khudhur[1], Hassan Awheed Jeiad[2]
[1,2]*Computer Engineering Department, University of Technology, Baghdad, Iraq*
[1]*saja.d.khudhur@uotechnology.edu.iq*, [2]*hassan.a.jeiad@uotechnology.edu.iq*

***Abstract***— *This paper introduces DLSTM-MSF, a distributed approach designed to address the challenge of demand forecasting in multimedia streaming workloads. DLSTM-MSF leverages the power of multi-LSTM networks, each tailored to predict data demand for a specific type of multimedia streaming workload. The central problem addressed in this research is the accurate prediction of workload demand in a dynamic and diverse multimedia streaming environment. To achieve specialization, the training time series set for each LSTM network comprises examples with targets belonging exclusively to the workload type it is designed to predict. This specialization ensures that each LSTM network becomes proficient at capturing the unique demand patterns associated with its designated workload category. The methodology of the proposed approach is based on building the best forecasting model for each multimedia streaming workload type by exploring various combinations of LSTM hyper-parameters using the grid search method. This enables the proposed approach to effectively capture nonlinear patterns in time series data. Furthermore, the implementation of DLSTM-MSF incorporates Apache Kafka for online demand prediction, utilizing the best-developed model for each workload type. Experimental evaluations of DLSTM-MSF compare the performance of two ensemble-learning LSTM models (Ensemble V1 and Ensemble V2) with a single LSTM model. The results unequivocally highlight the superiority of Ensemble V1, with reductions of 71.85% and 74.88% in RMSE and MAE values, respectively, compared to the single LSTM model.*

***Index Terms***— *Multimedia streaming, LSTM, Ensemble learning, Forecasting, Workload Demand, Big data.*

## I. INTRODUCTION

All sizes of businesses now employ cloud computing extensively. Although there is no denying its advantages, cloud resources can be expensive, especially when they are provisioned with a sufficient safety buffer to ensure system availability during unplanned situations. Severe costs can be entailed due to the unrestricted scaling of the cloud resources; businesses are over budget on the public cloud by 13%, according to Flexera 2022 State of the Cloud Report, and they anticipate a further rise in cloud spending of 29% over the following 12 months that suggests understanding forecasting and cost optimization is more important than ever. Access to auto-scaling components integrated into cloud platforms is available from all major cloud computing providers, including Amazon (Amazon Web Services), Google (Google Cloud Platform), and Microsoft (Azure) There

*Received 15/August/2023; Accepted 09/October/2023*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

are also commercial, cloud provider-agnostic applications created to maximize the use of cloud resources. Similarly, there are other researches on this subject that can be found in the literature [1]–[3]. The typical strategy for resource optimization [4], [5] concentrates on formalizing the system model and attempting to classify system load. This can lead to poor efficiency when dealing with real-world data that contains random anomalies such as big data streaming [6], [7]. The workload demand for multimedia streaming is one of the more crucial issues in big data streaming. It imposes the dealing with a multi-non-linear time series to provide an accurate demand forecasting impacting the resource utilization.

All planning activities heavily rely on demand forecasts, making demand prediction a crucial aspect of predictive analytics for anticipating future demand. Accurate demand forecasting is essential to ensure appropriate supply chain management [8]. However, traditional demand forecasting techniques have faced challenges in accurately predicting demand due to intense competition across industries.

The development of workload demand forecasting based on multimedia streaming meaning the deal with the big data streaming characteristics. The main distinctive characteristic of big data in this context is Variety, which means that data is collected in various formats, such as blog entries, videos, text, images, and audio, often referred to as unstructured data. This type of data poses additional requirements that traditional solutions may not adequately address [8].

Furthermore, different workloads, areas, or domains require various resources. Cloud resources vary in terms of memory size, processor speed, etc. Therefore, resource allocation algorithms need to efficiently utilize these resources while adhering to service-level agreements (SLAs). To ensure the availability of the services, no machine should be overloaded or underloaded. It is essential to balance the load to optimize the performance parameters of the allocation process. Two types of resource allocation implementations exist: static and dynamic. However, due to the stochastic nature of the big data environment and the need for heterogeneous resources, static resource allocation has limitations [9].

Dynamic resource provisioning is a challenging problem in the scheduling of big data applications. However, workload prediction plays a crucial role in this dynamic resource allocation process. Therefore, the prediction of workload and job estimation has garnered significant attention from researchers and data scientists [10]–[12]. Moreover, much of the existing research on workload forecasting relies on historical data. In the context of big data streaming, the characteristics of incoming data streams are unpredictable due to the stochastic nature of the data sources, and the user cannot determine the requirements in advance. This unpredictability makes it challenging to predict upcoming workload patterns in real-time and forecast the demand to allocate appropriate resources. These challenges serve as the primary motivation for the research presented in this paper.

The main objectives of the present work are as follows:

A- Propose multi-forecasting models that can effectively address the heterogeneity issues of multimedia streaming. These models will automatically forecast the workload demand based on the elementary types of multimedia, such as image, video, audio, and text, using the LSTM network.

B- Prepare the time series data that is generated from the deployment of the LSDStrategy [13]. This data will be used for training and evaluating the proposed multi-forecasting models.

C- Optimize the hyper-parameters of the multi-LSTM networks and select the most suitable configuration for each workload type. This step aims to overcome the challenges of obtaining accurate forecasting models for nonlinear and non-stationary demand time series data.

Received 15/August/2023; Accepted 09/October/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

D- Deploy the developed models in a distributed environment to ensure scalability and reliability in handling data streaming.

The present work achieved the following contributions:

- It introduces a distributed approach called DLSTM-MS, which consists of multi-forecasting models capable of accurately predicting workload patterns based on different types of multimedia streaming.
- It presents data preprocessing techniques for the time series data generated from [13], which is essential for training the forecasting models.
- It fine-tunes the multi-LSTM networks by selecting hyper-parameters that strike a balance between accuracy and model complexity.
- The work deploys four workers, ensuring scalability and availability while achieving impressive RMSE and MAE values of 71.85% and 74.88% lower than that of the simple LSTM networks respectively. This deployment addresses challenges related to centralized deployment when dealing with real-world data, such as high memory consumption, long processing and analysis times, computational performance degradation, and significant I/O overhead.

## II. BACKGROUND RESEARCH

In this section, a comprehensive review of the recent literature surveyed the current state of the art for workload demand forecasting.

For applications based on forecasting, most earlier techniques based on computational intelligence and statistical techniques are broadly accepted.

*Fig.* **1** shows the most common time series forecasting methods. These models come in various forms, with each model specialized in representing a particular type of nonlinearity. Consequently, finding an appropriate model for time series forecasting becomes more challenging due to the diverse range of nonlinear patterns [14]. Although it is cost-effective, aggressive optimization degrades the Quality of Service (QoS) [15], [16].

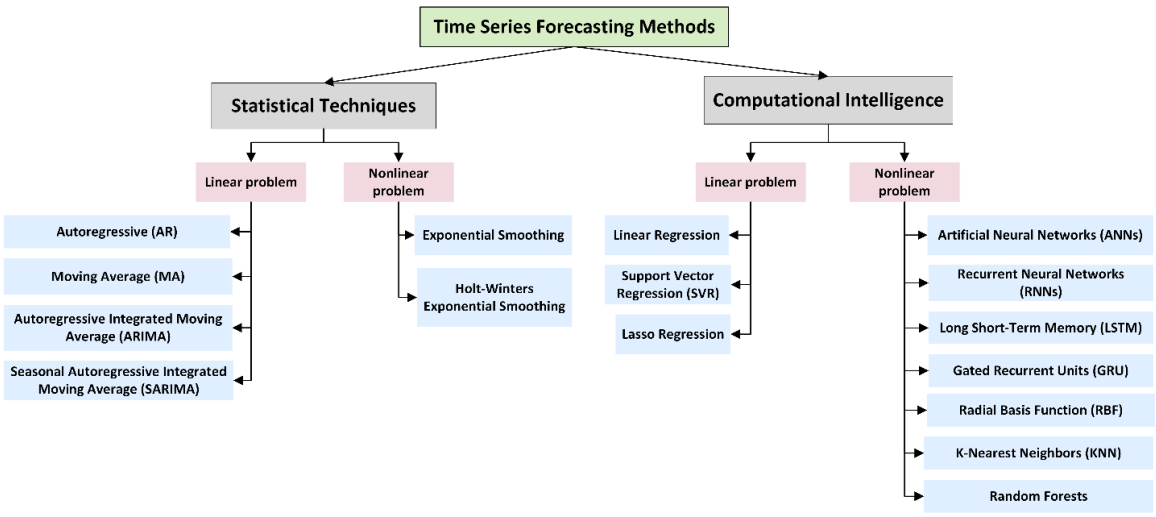


FIG. 1. THE CLASSIFICATION OF THE TIME SERIES FORECASTING METHODS.

The widely used statistical methods are ARIMA, Seasonal ARIMA (SARIMA), moving average and exponential smoothing. [17]–[21].

In [17], [21] the authors proposed a method that categorizes the Big Data (BD) stream based on its variety. Subsequently, the volume and velocity of the stream are predicted

Received 15/August/2023; Accepted 09/October/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

using KF by considering this variety. To achieve this, they identified the workload types using a statistical approach. Their approaches heavily depended on semantic parsing of the workload, which was considered a limited approach due to its reliance on a multi-file identification tool to generalize the work for various types. Unfortunately, there is no such tool capable of handling all file types effectively. Additionally, any method relying on file signatures for type identification becomes ineffective in the event of data corruption. These limitations highlighted the need for alternative approaches to overcome these challenges in file type identification and semantic parsing.

In addition to the high exploitation of statistical methods by researchers, many studies related to time-series data forecasting based on computational intelligence methods have emerged [22]–[27]. The authors in [27] proposed a workload prediction adaptive NNs model for average workload forecasting over consecutive prediction intervals proactively. The proposed model learned workload patterns for specific prediction intervals from historical data using the proposed novel Auto Adaptive Differential Evolution (AADE) algorithm. The performance of the proposed model was evaluated on NASA and Saskatchewan HTTP traces.

In [14], the authors presented an demand forecasting method based on multi-layer LSTM networks, specifically designed to handle highly fluctuating demand data. Their approach selected the best forecasting model by exploring different hyperparameters' combinations of LSTM.

In [28], the authors introduced an ensemble learning-based workload prediction technique that leverages extreme learning machines, and their predictions are combined using a voting engine with weighted inputs. Google Cluster Trace, which constitute resource utilization metrics, and PlanetLab traces were used to evaluate the accuracy of their approach.

In [29], the authors conducted an investigation into the use of Bayesian Neural Networks and DL models for predicting workload distribution. They evaluated these models in the context of time series forecasting for CPU and memory workloads across 8 clusters in the Google Cloud data center, which involved resource utilization metrics. The experiments revealed that the proposed models were effective in providing accurate demand predictions and improved estimations of resource usage bounds. The models successfully reduced overprediction and the total predicted resources, while also avoiding underprediction, demonstrating their potential in optimizing resource allocation and management in cloud environments.

In [30], an approach based on DL was presented, employing a combination of neural networks to analyze historical workload data and predict future workloads. The model's performance was assessed using real datasets obtained from real-time resource utilization and performance metrics The proposed algorithm successfully provided precise predictions of future workloads, demonstrating its effectiveness in workload forecasting.

All the mentioned research, except [17], [21], employ performance metrics, resource utilization metrics, or system logs as historical data rather than historical workload patterns, which are utilized in studies [17], [21]. In the context of BD streaming, historical workload patterns hold particular significance. Streaming involves real-time data ingestion, making workload patterns crucial for understanding variations. While resource utilization and performance metrics are also relevant, workload patterns address the dynamic nature of streaming. They play a vital role in guiding real-time resource provisioning decisions by revealing workload fluctuations and patterns.

table I summarizes the recent studies that relied on the time series forecasting approach. The most prominent feature of our work with the literature referred to in table I is the deployment of it in a virtual machine composing the possibility of distributed analysis

Received 15/August/2023; Accepted 09/October/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

and forecasting with RMSE and MAE values of 71.85% and 74.88% lower than that of the LSTM respectively.

TABLE I. THE RECENT WORKLOAD DEMAND FORECASTING STUDIES

| Contributors | year | Forecasting Method | Methodology | Type of historical data | Dataset Name |
|---|---|---|---|---|---|
| Kaur and Sood [17] | 2017 | statistical techniques | KF | Workload Pattern | Synthetic data |
| Murray et al. [18] | 2018 | | ARIMA | real-life time series data | historical transaction data (costumer segmentation) |
| Sarica et al. [19] | 2018 | | AR–ANFIS | real-life time series data | real-life time series data |
| Parmezan et al. [20] | 2019 | | Eleven predictors (seven parametric and four non-parametric) | - | 95 datasets |
| Kaur et al. [21] | 2020 | | KF | Workload Pattern | Synthetic data |
| Martinez et al. [22] | 2018 | computational intelligence | K-NN | real-life time series data | NN5 competition time series. |
| Kumar et al. [23] | 2018 | | LSTM | System logs | Three benchmark datasets consisting of web server logs |
| Martinez et al. [26] | 2019 | | K-NN | real-life time series data | NN3 competition time series. |
| Shi et al. [25] | 2019 | | LSTM | real-life time series data | gyroscope shell temperature data |
| Sagheer and Kotb [24] | 2019 | | DLSTM | real-life time series data | petroleum time series datasets |
| Kumar and Singh [31] | 2019 | | Differential Evolution | Resource utilization metrics | Google's real-world trace |
| Saxena and Singh [27] | 2020 | | AADE | Resource utilization metrics | NASA and Saskatchewan HTTP traces. |
| Abbasimehr et al. [14] | 2020 | | LSTM | real-life time series data | sale data of a furniture company |
| Kumar et al. [28] | 2020 | | Ensemble learning-based model | Resource utilization metrics | CPU utilization of Google clusters and PlanetLab traces. |
| Rossi et al. [29] | 2022 | | Bayesian Neural Networks and DL models | Resource utilization metrics | Google cloud clusters trace |
| Bansal and Kumar [30] | 2023 | | ANN | resource utilization and performance metrics | real-time dataset |
| DLSTM-MSF | - | | Ensemble LSTM | Workload Pattern | Time series of LSDStrategy |

## III. PROBLEM STATEMENT

The efficient and timely delivery of resources in cloud computing presents a major challenge. Some cloud providers still allocate resources statically based on peak demands, leading to high costs for customers and low resource utilization. Dynamic resource allocation is a more effective approach, but resource demands fluctuate continuously. To ensure a positive user experience, proactive resource provision is crucial, but overestimating can result in wasteful over-provisioning, while underestimating leads to unmet demands. Accurately forecasting future resource demands is the key challenge [32].

Numerous techniques and studies have addressed workload forecasting for resource allocation. In stochastic environments like big data steaming, shallow or short-term algorithms may impact prediction accuracy, leading to over- or under-provisioning. DL models, such as LSTM, excel at recognizing inherent patterns, making them superior in extracting characteristics from workload patterns compared to shallow models.

Received 15/August/2023; Accepted 09/October/2023

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

Our work focuses on streaming big data applications with unpredictable workload characteristics, which heavily influences resource provisioning. Predicting the type of streaming data also impacts the choice of cloud resources for processing or storing the data.

In this paper, we employ DLSTM-MSF to forecast multimedia streaming workload demand using timeseries data obtained from our previous work [13] that utilizes the content based analysis [33] in the prediction and generating the time series. The timeseries dataset comprises the workload request types at specific time instances, predicted through content-based analysis. DLSTM-MSF is deployed in a distributed environment for reliable and scalable data streaming capabilities. Accurate workload demand prediction can significantly enhance the availability and reliability of multimedia streaming systems.

## IV. THEORETICAL BACKGROUND

### A. LSTM

LSTM is an extension of RNNs that has a strong capability in forecasting time series data. The RNNs have some weaknesses, such as the exploding gradient and vanishing problem, which make training challenging [20]. To overcome these issues, researchers have introduced gated designs like LSTM, which can capture longer-range timing information and are more effective in handling time series data The main difference between an RNN and LSTM is that it is structured of three gates: a forget gate, an input gate (cell memory), and an output gate which controls the flow of information [14], [34]. That structure provides it with the capability of storing long-range time dependency information and the suitability of mapping between input and output data.

Fig. **2** illustrated the LSTM network structure where the notations used in LSTM learning process equations is summarized in Table II. The status updating process of the memory unit are as follows [14]:

1. The forget gate removes data from the memory unit in accordance with the input ($x(t_i)$) of the present time and the prior time output $h(t_{i-1})$ using sigmoid activation function (σ). This is computed using Eq. 1.

$$f(t_i) = \sigma\left(w_f x(t_i) + w_{hf} h(t_{i-1}) + b_f\right) \quad (1)$$

2. The information to be stored in the memory unit is determined by the input and forgetting gates. This is computed using Eq. 2 and 3.
3. 

$$a(t_i) = \sigma(w_a x(t_i) + w_{ha} h(t_{i-1}) + b_a) \quad (2)$$

$$c(t_i) = f_t \times c(t_{i-1}) + a_t \times tanh(w_c x(t_i) + w_{hc} h(t_{i-1}) + b_c) \quad (3)$$

4. The output gate commits regulation on the output of an LSTM cell and finally is updated by considering cell state. This is computed using Eq. 4 and 5.

$$o(t_i) = \sigma(w_o x(t_i) + w_{ho} h(t_{i-1}) + b_o) \quad (4)$$

$$h(t_i) = o(t_i) \times tanh\left(c(t_i)\right) \quad (5)$$

Received 15/August/2023; Accepted 09/October/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

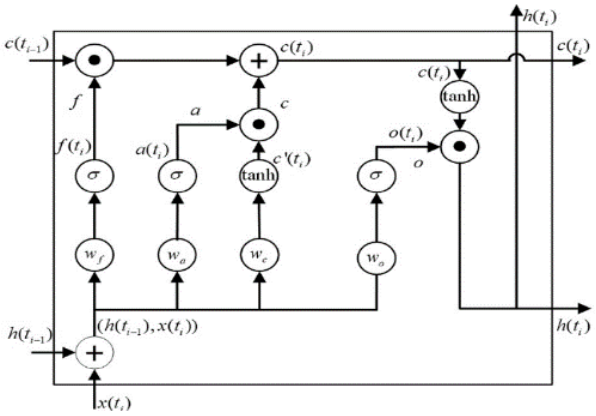


FIG. 2. LSTM NETWORK STRUCTURE [14].

Overall, the learning process steps for the LSTM network done as follows:

*Step_1:* Calculate the value of the output gate for the LSTM network, the forward learning, using Eq. (1)-(5).

*Step_2*: Calculate the error value that resulted from the difference between the input value and the output value for each layer of the network.

*Step_3*: Spread the error value computed in Step_2 in reverse for the three gates: input, cell, and forget.

*Step_4*: Update the weight for each of the above gates depending on the error value using an optimization algorithm.

TABLE II. NOTATIONS BASED LSTM LEARNING PROCESS

| Notation | Brief Meaning |
|---|---|
| $a(t_i)$ | The input gate |
| $f(t_i)$ | The forget gate |
| $c(t_i)$ | The cell state (cell memory) |
| $o(t_i)$ | The output gate |
| $x(t_i)$ | The current input value |
| $h(t_i)$ and $h(t_{i-1})$ | The current and previous output values |
| $c(t_i)$ and $c(t_{i-1})$ | The current and previous state of the cell (cell memory) |
| $\overrightarrow{W_1} = \{ w_a, w_f, w_c, w_o \}$ | The weight matrix for the input gate, forget gate, cell state and output gate. |
| $\overrightarrow{W_2} = \{ w_{ha}, w_{hf}, w_{hc}, w_{ho} \}$ | The recurrent weights matrix for the input gate, forget gate, cell state and output gate. |
| $b = \{b_a, b_f, b_c, b_o\}$ | The biases value for the input gate, forget gate, cell state and output gate. |
| $\vec{a} = \{ a(t_i), f(t_i), c(t_i), o(t_i) \}$ | The output matrix for the input gate, forget gate, cell state and output gate |
| $\sigma$ | Sigmoid activation function |
| $\times$ | point-wise multiplication |

## B. LSTM Hyperparameter Tuning

The LSTM network has demonstrated satisfactory performance when applied to sequence data. However, achieving good results with LSTM networks is a complex endeavor due to the need for optimizing multiple hyperparameters. The selection of appropriate hyperparameters is crucial for improving the model's performance. Noteworthy hyperparameters for time series forecasting problems include lag size (the number of past observations, layers number, learning rate, number of neurons, activation function, number of epoch, and Batch size [35].

However, the impact of these hyperparameters can vary depending on the dataset and the specific time series forecasting task. Therefore, a systematic hyperparameter search, like grid search or random search, combined with cross-validation, is recommended to find the best combination of hyperparameters for the LSTM model.

Received 15/August/2023; Accepted 09/October/2023

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

**C. Dataset Analysis**

The dataset utilized here is the time series dataset was generated from the deployment of the LSDStrategy [13]. The LSDStrategy deployment is designed for a streaming file upload use case, where file requests are broken down into smaller chunks. This strategy is particularly effective when dealing with distributed file systems, distributed databases, distributed computing, and limited Internet capability. The use of a specific sliding window size allows for the collection of these smaller chunks over a period of time. From these chunks, features are extracted, which were previously selected in the Feature Selection stage of the LSDStrategy pipeline. The data set used in this pipeline was based on content analysis [33]. The primary objective of the LSDStrategy was to estimate diversity in multimedia streams within big data. The estimation results take the form of time series data, which includes the timestamp of request reception, its type, and the number of requests. The variables of the used dataset are:

- *Timestamp of request reception*: It indicates the time at which a particular request type is received. This variable represents the temporal aspect of the data points.
- *Request type*: It represents the data type of the request, which is related to the file content being uploaded including four types: image, video content, audio, and text.
- *Number of requests*: It indicates the count of requests of a particular request type, which is received at that timestamp.

Therefore, there are three variables being recorded over time, so this time series is considered multivariant time series data. A multivariate time series data set is data that contains multiple variables or features, where each variable's value is recorded over time. That's mean each timestamp is associated with the two variables' values.

**D. Data Preparation**

Time Series data contains a lot of information; however, it is typically hidden. So, any time-series data, especially real-time data, needs strict preprocessing. Unordered timestamps, missing values, and data noise are the most frequent issues with time series. The handling of missing values is the most challenging of the aforementioned issues. This section composes the data preprocessing technique including *structuring, imputation, normalization,* and *transformation*.

In most cases, time series data is acquired in unstructured formats, e.g., timestamps often be mixed up and improperly arranged. Therefore, the *structuring* process involves several operations to form the data in a structure that is amenable to time-based, including data parsing, and temporal aggregation.

Data parsing revolves around representing the timestamps of the data in a structure that is amenable to time-based slicing and dicing (transformed into an appropriate date-time data type). Moreover, temporal aggregation is employed to transform time series data by consolidating data points and adjusting the data's granularity by clearing data and resampling.

Data Cleaning involves consolidating data points that share the same timestamp and summing their values. While data resampling concerns changing the time series data's granularity by converting it to a different time frequency.

The *imputation* process is concerned with compensating for missing data or gaps in time series timestamps, which in turn may negatively affect the quality of the data and thus the forecasting because the order in which values are received is important. That makes the conventional imputation approaches unsuitable for the time-series data [36].

Moreover, the *normalization* technique standardizes the time series data to a uniform range, such as Min-Max process. This type of works by subtracting the minimum value from each data point and then the result by the difference between the maximum and minimum values. This results in a new range of values between 0 and 1, where 0 represents the minimum value, and 1 represents the maximum value [37]. In addition, *transformation* is one of the most data preparation processes. It concerns

Received 15/August/2023; Accepted 09/October/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

reshaping the dataset instance from a sequence of points to a pair of input/output points, enabling its compatibility with supervised learning model.

## V. METHODOLOGY OF DLSTM-MSF MODEL

A DLSTM-MSF has been proposed to forecast the workload patterns of multimedia streaming in a distributed environment. This approach utilizes LSTM networks, which are ideal for capturing long-range dependencies in time series data. DLSTM-MSF involves constructing and evaluating four forecasting models using LSTM networks, each specializing in forecasting the workload demand for a specific media type: image, audio, video, and text.

The deployment of DLSTM-MSF utilizes Apache Kafka, a distributed messaging system that enables scalable and reliable data streaming. As well as Apache Kafka plays a crucial role in ensuring the delivery and ordering of data streams, making it suitable for latency-critical applications like real-time workload demand prediction.

In this setup, a stimulus streamer is developed to stream the time-series-based workload demand generated from the deployment of [13]. Moreover, four workers are employed, which each is responsible for forecasting a specific media type. This division of forecasting problems enables efficient parallel processing, considering the unique characteristics of each media type. The architecture of DLSTM-MSF is depicted in *Fig. 3.*

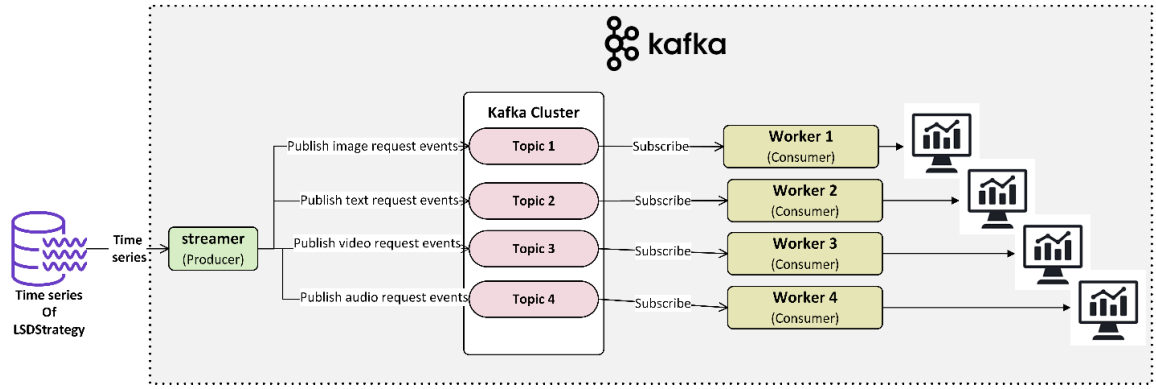


FIG. 3. DLSTM-MSF ARCHITECTURE.

The streamer streams the historical time series data of [13] in JSON format. Each tuple in the schema consists of three parameters: timestamp, media type, and the number of requests.

To implement this at the technical level, the Kafka Producer API is utilized by the streamer to publish the streaming data to four Kafka topics. Each topic is corresponding to a specific media type of request. In addition, four workers are developed using the Consumer API. Where each is subscribing to a specific topic.

The processing stages of each worker are illustrated in

Received 15/August/2023; Accepted 09/October/2023

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

Fig. **4** including data collecting, data preparation, hyperparameters tuning, transformation, model training, and model evaluating. These stages focus on preparing the time series data and optimizing the hyperparameters of the LSTM network to overcome the challenges of obtaining an accurate forecasting model for nonlinear and non-stationary demand time series.

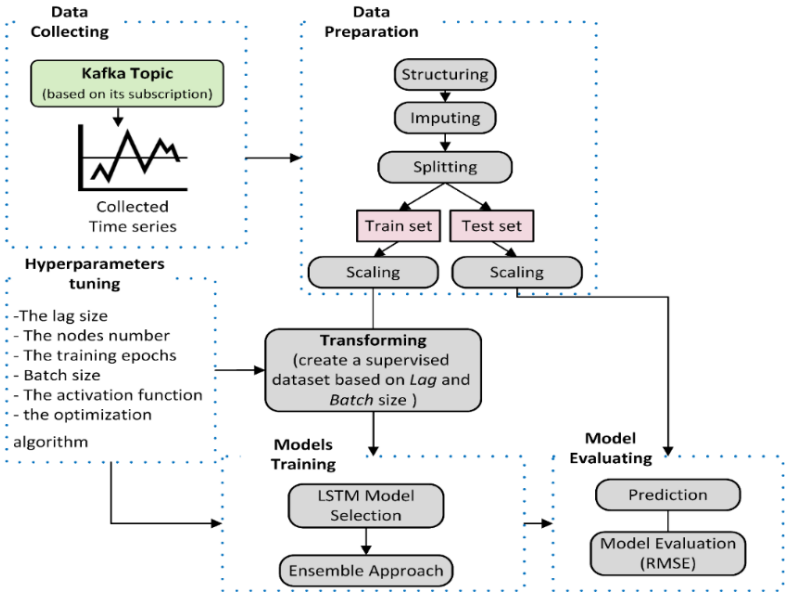


FIG. 4. WORKER BASED PROCESSING STAGES.

## A. Data Collecting

Each worker in the proposed DLSTM-MSF plays a crucial role in collecting time series data that is related to the specific topic it subscribes to. As mentioned, the historical time series of the [13] is a multivariant time series data. Thus, to convert it to univariant and enabling the distributed processing, the streamer publishes those data into four specific topics. Upon subscription of each worker to a specific topic, the workers continuously listen for new data records generated by the corresponding media type, such as image, audio, video, or text. By adhering to their designated topics and leveraging the capabilities of the Kafka Consumer API, each worker in the DLSTM-MSF system efficiently and accurately collects the data required for forecasting workload patterns in multimedia streaming. As data streams arrive, the workers extract relevant information from each tuple that is required to train the forecasting model, which are *timestamp*, and the *number of requests*. These data points are vital for understanding the temporal characteristics and demand patterns specific to the media type under consideration. Furthermore, the workers store the collected data in a structured format for further analysis and forecasting. To ensure real-time data processing, the workers are designed to efficiently handle incoming data as it becomes available. This enables the workers to stay synchronized with the data flow and capture the latest workload demands for their respective media types. *Fig.* 5 shows the original observation and trends component of the collected media time series data by all workers.

## B. Data Preparation

Due to the reality of each worker ingesting only the data related to a specific topic, the collected time series of each worker may encourage a time gap. In addition, the collected data were streamed as a string data time type that means several data preprocessing techniques are required including:

Received 15/August/2023; Accepted 09/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

- ***Structuring****:* To make the collected time series amenable to forecast, several processing steps are performed, including parsing, cleaning, and resampling. *Parsing* involves converting the timestamp of the collected data to a time-based data type. Additionally, the *cleaning* process deals with duplicated values, where multiple instances have the same time value. Since time series forecasting requires unique time instances, all duplicated values for the same time instance are aggregated. Finally, the *resampling* process involves downsampling the data by aggregating the time frequency from a lower level to an upper level to increase the data's worth. The time frequency of the collected data is summarized from seconds to hours.
- ***Imputing***: Due to the context of each worker ingesting only the time series related to its subscription, often some time gaps occur. These gaps represent the absence of a specific event (request for a specific media time). Therefore, zero imputation is utilized, which is a straightforward method to replace the missing values with zeros, indicating the absence of a certain measurement during those time points.
- ***Splitting***: The collected time series is split into a training and testing set, with a thirty-day training set and the last day as the testing set. The LSTM models are trained and optimized using the training set and evaluated using the testing set.
- ***Scaling***: Since most forecasting techniques show more promising forecasting ability with normalized data, in this stage, the min-max normalization algorithm is utilized.

**C. Hyperparameter Tuning**

In this work, LSTM networks by Keras are adopted. However, achieving high performance with these networks is not a straightforward task. To optimize this model, we need to search for high-impact parameters to configure. We employ the widely-used grid search method, which allows us to explore various combinations of hyperparameters comprehensively. Seven different hyperparameters were adjusted during our experiments, as illustrated in Table III. Depending on the number of values for each hyperparameters, a list of parameter combinations is generated and utilized for data transformation along with model configuration and training.

Received 15/August/2023; Accepted 09/October/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

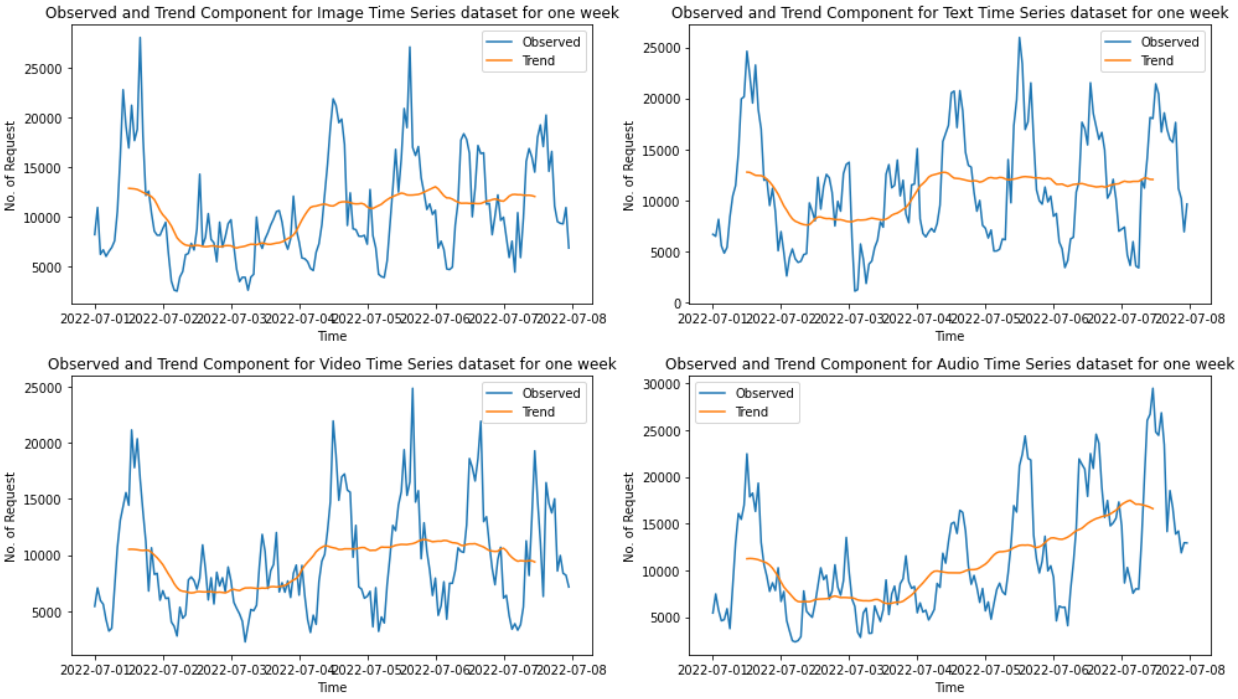


FIG. 5. THE OBSERVED AND TREND COMPONENT OF THE IMAGE, TEXT, VIDEO, AUDIO REQUEST FOR ONE WEEK.

TABLE III. HYPERPARAMETERS SEARCH SPACE

| Hyperparameter Name | Values |
|---|---|
| Lag size | [3, 4, 5, 6] |
| The number of nodes | [50, 100, 150, 200] |
| Epochs number | [25, 50, 100, 125] |
| Batch size | [2, 3, 4] |
| Activation function | [tanh, relu, sigmoid] |
| Learning rate | [0.001, 0.01, 0.1] |
| Number of layers | [1, 2] |

## D. Transforming

Based on the selected lag and batch sizes observed in the previous stage, the time series data is re-formed into a structure of input/output points. By this form, the time series data be suitable for supervised learning, making it ready for training the LSTM network.

## E. Models Training

The proposed DLSTM-MSF approach is mainly designed to improve the performance of the LSTM model. Initially, a single LSTM network is designed and trained using the best set of selected hyperparameters. The LSTM network employs the Adam algorithm as the optimizer, and the Mean Squared Error (MSE) as the loss function. Subsequently, two ensembled versions of the LSTM network are created named Ensemble V1 and Ensemble V2. In Ensemble V1, two LSTM models are independently trained, each utilizing a distinct subset of the training set. Similarly, in Ensemble V2, three LSTM models are trained independently on different subsets of the data. The ensemble LSTM models are implemented using the Bagging technique [38]. By training multiple models on diverse subsets, the ensemble can effectively capture various patterns in the data and mitigate overfitting.

## F. Model Evaluation

*Received 15/August/2023; Accepted 09/October/2023*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

Two performance metrics including root mean square error (RMSE) and mean absolute error (MAE) are applied to evaluate the efficiency of all the LSTM models, the single LSTM mode, and the two versions of the ensemble LSTM model. The average magnitude of error is computed by RMSE and average magnitude of that error irrespective to its direction is calculated by MAE [39]. They computed using the Eq. 6 and 7 respectively.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (6)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_{i_i} - y_i| \quad (7)$$

In both equations, $n$ depicts the data observations, while $y_i$ and $\hat{y}_i$ denoted to the original and forecasted value at time point $i$ respectively. The testing set is used in this stage to evaluate the three models and find which provide the minimum RMSE and MAE.

## VI. EXPERIMENTAL RESULTS AND DISCUSSION

This section discusses the results of implementing the DLSTM-MSF. The Kafka server (v. 3.2.0) is utilized which composed of one producer (streamer) and four consumers (workers). Virtual machines with Ubuntu are utilized to build our cluster. Table IV illustrates the cluster nodes characteristics.

TABLE IV. CLUSTER NODES CHARACTERISTICS

| Parameter | Streamer | Worker |
|---|---|---|
| Operating System | Ubuntu 20.04.4 LTS | Ubuntu 20.04.4 LTS |
| Processor | 2.00 GHz Intel(R) Core (TM) i7 | 2.00 GHz Intel(R) Core (TM) i7 |
| Memory | 12 GB | 12 GB |

The implementation used the Python programming language (3.9.5) with the necessary data manipulation and analysis libraries to prepare the data, and the Keras and Tensorflow packages to design the LSTM models.

Each worker subscribes to a specific topic under which the corresponding data is streamed. Then the collected data is prepared to be transformed into supervised data. The Keras 'TimeseriesGenerator' class is utilized to re-form the collected time series into a structure of input/output points. This class automatically reshapes the time series data into a format suitable for supervised learning, making it ready for training the LSTM network.

The transformation process is affected by the number of lag and batch size hyperparameters that are selected. Table V illustrates the optimized value of hyperparameters obtained from the grid search process of each worker. These hyperparameters are selected based on their ability to minimize the RMSE.

TABLE V. THE BEST SET OF THE SEARCHED LSTM NETWORKS HYPERPARAMETERS FOR DIFFERENT TIME SERIES

| Hyperparameter Name | Hyperparameter Value | | | |
|---|---|---|---|---|
| | Worker (1) | Worker (2) | Worker (3) | Worker (4) |

Received 15/August/2023; Accepted 09/October/2023
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

| | ***(Image workload time series)*** | ***(Text workload time series)*** | ***(Video workload time series)*** | ***(Audio workload time series)*** |
|---|---|---|---|---|
| Lag size | 4 | 4 | 3 | 4 |
| The number of nodes | 150 | 100 | 50 | 150 |
| Epochs number | 50 | 50 | 50 | 50 |
| Batch size | 3 | 3 | 2 | 4 |
| Activation function | Relu | Relu | Relu | Relu |
| Learning rate | 0.001 | 0.001 | 0.001 | 0.001 |
| Number of layers | 2 | 2 | 1 | 1 |

Based on the selected lag size and batch size, the 'TimeseriesGenerator' class generates overlapping sequences of length equal to lag size from the original time series data. For example, with a lag size of 4, each time series points sequence of length 4 is used as input to predict the subsequent time step, which becomes the output. Furthermore, with a batch size of 3, the parameters of the model will be updated after three instances. A subset of the created input/output pairs for the four time series data are illustrated in

Table **VI**, VII, and VIII. These input/output pairs create a supervised learning dataset, which is suitable to learn the LSTM network. The $t_i$ in these tables refers to the time series point in $i$ time.

TABLE VI. A SUBSET OF INPUT/OUTPUT PAIRS INSTANCE FOR IMAGE AND TEXT WORKLOAD TIME SERIES (FOR LAG=4, BATCH SIZE = 3)

| **Batch No.** | **Input** | **Output** |
|---|---|---|
| **1st** | $\{t_1, t_2, t_3, t_4\}$ | $\{t_5\}$ |
| | $\{t_2, t_3, t_4, t_5\}$ | $\{t_6\}$ |
| | $\{t_3, t_4, t_5, t_6\}$ | $\{t_7\}$ |
| **2nd** | $\{t_4, t_5, t_6, t_7\}$ | $\{t_8\}$ |
| | $\{t_5, t_6, t_7, t_8\}$ | $\{t_9\}$ |
| | $\{t_6, t_7, t_8, t_9\}$ | $\{t_{10}\}$ |

TABLE VII. A SUBSET OF INPUT/OUTPUT PAIRS INSTANCE FOR VIDEO (FOR LAG=3, BATCH SIZE = 2)

| **Batch No.** | **Input** | **Output** |
|---|---|---|
| **1st** | $\{t_1, t_2, t_3\}$ | $\{t_4\}$ |
| | $\{t_2, t_3, t_4\}$ | $\{t_5\}$ |
| **2nd** | $\{t_3, t_4, t_5\}$ | $\{t_6\}$ |
| | $\{t_4, t_5, t_6\}$ | $\{t_7\}$ |

TABLE VIII. A SUBSET OF INPUT/OUTPUT PAIRS INSTANCE FOR AUDIO WORKLOAD TIME SERIES (FOR LAG=4, BATCH SIZE = 4)

| **Batch No.** | **Input** | **Output** |
|---|---|---|
| **1st** | $\{t_1, t_2, t_3, t_4\}$ | $\{t_5\}$ |
| | $\{t_2, t_3, t_4, t_5\}$ | $\{t_6\}$ |
| | $\{t_3, t_4, t_5, t_6\}$ | $\{t_7\}$ |
| | $\{t_4, t_5, t_6, t_7\}$ | $\{t_8\}$ |
| **2nd** | $\{t_5, t_6, t_7, t_8\}$ | $\{t_9\}$ |
| | $\{t_6, t_7, t_8, t_9\}$ | $\{t_{10}\}$ |
| | $\{t_7, t_8, t_9, t_{10}\}$ | $\{t_{11}\}$ |
| | $\{t_8, t_9, t_{10}, t_{11}\}$ | $\{t_{12}\}$ |

Through the utilization of the best-selected hyperparameter and the transformed data, the LSTM models are designed and trained. A comparative study is conducted among the three models using RMSE and MAE as an evaluation metric. To make a fair evaluation, all the designed models from all

*Received 15/August/2023; Accepted 09/October/2023*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

workers are trained with thirty-day time points and tested by one-day time points of the collected dataset.

*Fig. 6* shows the evaluation report for the adopted three models by all workers. All the designed models by each worker are tested using the test set of the corresponding media time series.

The evaluation results highlighted the superiority of the Ensemble V1 model across all datasets. The values of RMSE and MAE for this model in Image and Text demand forecasting are largely consistent, measuring 0.09 and 0.077, respectively. Meanwhile, for Video demand forecasting, these values are 0.067 and 0.052, respectively. However, despite the Ensemble V1 outperforming other models in Audio demand forecasting, its performance isn't as impeccable as it is in the case of the other three media demand forecasts. This discrepancy is due to the inherent fluctuations present in the Audio time series data. *Fig. 7* depicts the performance of the three models' training and test fit over the corresponding time series.

The primary objective of the proposed DLST-MSF is to establish a distributed environment ready for forecasting multimedia streaming's workload demand. Historical data form the foundation of this endeavor, drawn from the workload pattern of requests. These historical data originate from a previous deployment that predicts multimedia data types through content analysis [13]. This utilization employed a machine learning (ML) algorithm, which addresses the limitations of semantic and non-semantic parsing (e.g., non-ML) methods by leveraging ML's statistical classification capabilities. This pioneering analysis marks the first instance of streaming data analysis.

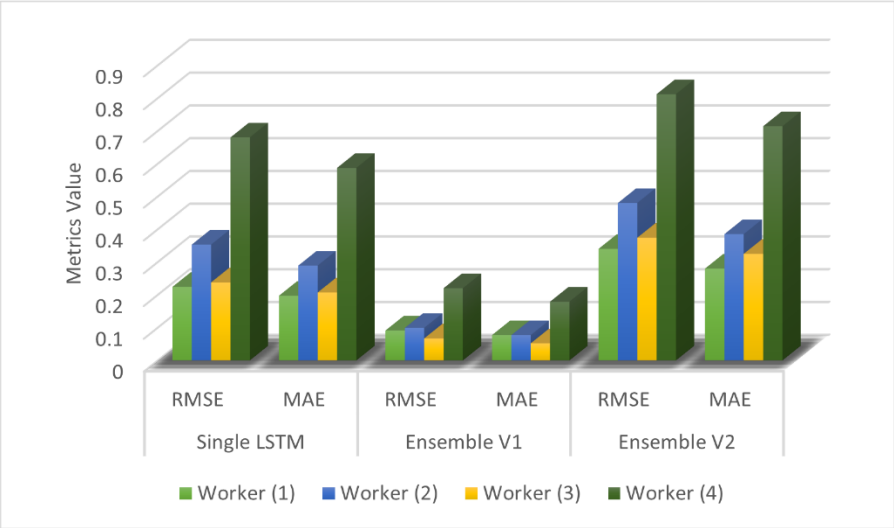


FIG. 6. THE EVALUATION REPORT OF THE THREE MODELS BASED ON EACH WORKER: TIME-SERIES PAIR.

Furthermore, the proposed DLST-MSF's validity in workload demand forecasting is substantiated through a comparative analysis with recent approaches. In our survey, all studies employed performance metrics, resource utilization metrics, or system logs as historical data except [17], [21], which were employed historical workload patterns. Within the realm of Big Data (BD) streaming, historical workload patterns assume particular significance. As streaming involves real-time data ingestion, these patterns become critical for comprehending variations. While resource utilization and performance metrics maintain relevance, workload patterns address the dynamic nature of streaming. They play a pivotal role in guiding real-time resource provisioning decisions by unveiling workload fluctuations and patterns.

The authors in [17], [21] employ historical data extracted based on identifying workload via examination of the magic numbers (file extensions) in data request blocks. To accomplish this, they

Received 15/August/2023; Accepted 09/October/2023

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

utilize a statistical approach to ascertain workload types. Their approaches heavily depend on semantic parsing of the workload, which is deemed limited due to its reliance on a multi-file identification tool that struggles to generalize across various types of work. Unfortunately, no such tool is universally effective across all file types. Additionally, any method hinging on file signatures for type identification loses effectiveness in cases of data corruption. Implementing this method becomes challenging when confronting big data streams in practical scenarios. Moreover, they utilize the Kalman Filter (KF) for forecasting workload demand, dealing with linear dynamics and Gaussian noise data. However, in the realm of big data streaming, such as multimedia streaming, non-linear and stochastic data prevail. As such, these limitations underscore the need for alternative approaches to surmount the challenges and provide media workload pattern data that significantly influence the workload demand forecasting process. Hence, the aforementioned non-predictive time series approaches and those founded on performance metrics and resource utilization metrics exhibit evident limitations in comparison to our proposal.

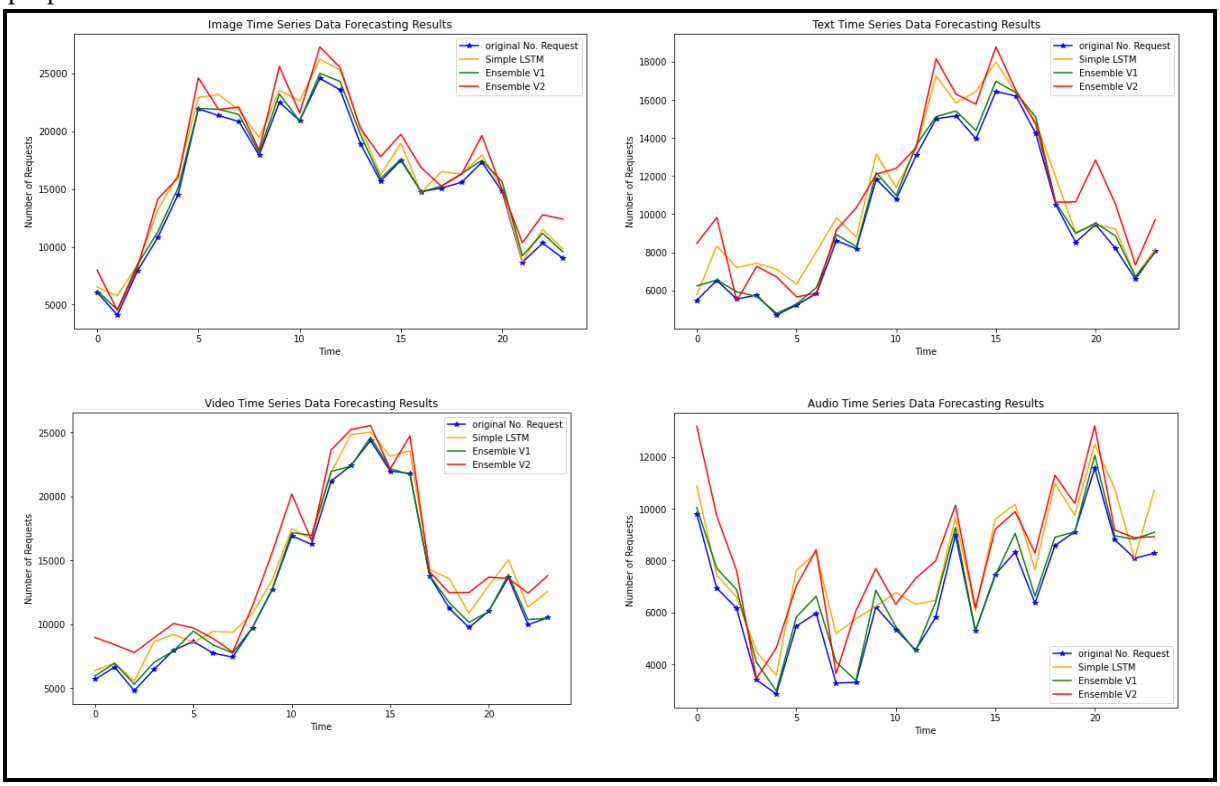


FIG. 7. FORECASTING RESULTS FROM THREE MODELS OVER THE FOUR MEDIA TIME SERIES DATA.

## VII. CONCLUSIONS AND FUTURE WORK DIRECTIONS

The focal point of LSTM network models lies in their potential for forecasting workload demand, thereby enhancing the resource manager's decision-making. To this end, we proposed a DLSTM-MSF approach, leveraging both a straightforward LSTM model and an ensemble LSTM model (a composite of LSTM models). The implementation of DLSTM-MSF was executed on the Apache Kafka server. Renowned for its capability to ensure data stream delivery and ordering, Apache Kafka proves ideal for latency-critical applications like real-time workload demand prediction.

The hyperparameters of the LSTM model were calibrated using the adopted time series data. A stimulus streamer was developed to stream time-series data encompassing the count of requests for four distinct types of media streaming over the course of a month. For this context, four workers were developed, each tasked with the preparation and forecasting of a specific media time series—Image, Text, Video, or Audio.

*Received 15/August/2023; Accepted 09/October/2023*

*ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

The core responsibility of each worker involved determining the optimal hyperparameters for the LSTM network relevant to its time series data following its preparation. Subsequently, each worker devised and trained three network models utilizing the LSTM network, employing the optimal set of hyperparameters. The evaluation results underscored the ensemble model's superior performance, based on two LSTM networks, which outperformed other designed models. This superiority was evident in its capacity to minimize error rates when forecasting unseen data for the upcoming day.

In future work, we hope to do the following:

- Aggregating the forecasting outcomes for the four-time series to derive statistical characteristics pertinent to the workload. This endeavor will provide valuable insights into the workload's dynamics, influencing the resource manager's decision-making process.
- Develop an optimization methodology tailored for the resource manager by leveraging the identified workload characteristics. This method will guide the allocation and provisioning of resources. The goal is to enhance resource utilization, thereby optimizing operational efficiency.

## REFERENCES

[1] A. Evangelinou, M. Ciavotta, D. Ardagna, A. Kopaneli, G. Kousiouris, and T. Varvarigou, "Enterprise applications cloud rightsizing through a joint benchmarking and optimization approach," *Future Gener. Comput. Syst.*, vol. 78, pp. 102–114, Jan. 2018, doi: 10.1016/j.future.2016.11.002. Available: http://dx.doi.org/10.1016/j.future.2016.11.002.

[2] A. Naseri and N. Jafari Navimipour, "A new agent-based method for QoS-aware cloud service composition using particle swarm optimization algorithm," *J. Ambient Intell. Humaniz. Comput.*, vol. 10, no. 5, pp. 1851–1864, May 2019, doi: 10.1007/s12652-018-0773-8. Available: http://dx.doi.org/10.1007/s12652-018-0773-8.

[3] B. Sniezynski, P. Nawrocki, M. Wilk, M. Jarzab, and K. Zielinski, "VM reservation plan adaptation using machine learning in cloud computing," *J. Grid Comput.*, vol. 17, no. 4, pp. 797–812, Dec. 2019, doi: 10.1007/s10723-019-09487-x. Available: http://dx.doi.org/10.1007/s10723-019-09487-x.

[4] M. Adhikari and T. Amgoth, "Multi-Objective Accelerated Particle Swarm Optimization Technique for Scientific workflows in IaaS cloud," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, Sep. 2018. doi: 10.1109/icacci.2018.8554584. Available: http://dx.doi.org/10.1109/icacci.2018.8554584.

[5] J. Yang, W. Xiao, C. Jiang, M. S. Hossain, G. Muhammad, and S. U. Amin, "AI-powered green cloud and data center," *IEEE Access*, vol. 7, pp. 4195–4203, 2019, doi: 10.1109/access.2018.2888976. Available: http://dx.doi.org/10.1109/access.2018.2888976.

[6] P. Nawrocki and P. Osypanka, "Cloud resource demand prediction using machine learning in the context of QoS parameters," *J. Grid Comput.*, vol. 19, no. 2, Jun. 2021, doi: 10.1007/s10723-021-09561-3. Available: http://dx.doi.org/10.1007/s10723-021-09561-3.

[7] H. M. Fadhil, M. N. Abdullah, and M. I. Younis, "A framework for predicting airfare prices using machine learning," *Iraqi Journal of Computer, Communication, Control and System Engineering*, vol. 22, no. 3, pp. 81–96, Sep. 2022, doi: 10.33103/uot.ijccce.22.3.8. Available: http://dx.doi.org/10.33103/uot.ijccce.22.3.8.

[8] A. Kumar, R. Shankar, and N. R. Aljohani, "A big data driven framework for demand-driven forecasting with effects of marketing-mix variables," *Ind. Mark. Manag.*, Jun. 2019, doi: 10.1016/j.indmarman.2019.05.003. Available: http://dx.doi.org/10.1016/j.indmarman.2019.05.003.

[9] Sukhpreet Kaur, Yogesh Kumar, and Sushil Kumar, "Soft Computing Techniques for Energy Consumption and Resource Aware Allocation on Cloud: A Progress and Systematic Review," in *Advanced Soft Computing Techniques in Data Science, IoT and Cloud Computing*, Sujata Dash , Subhendu Kumar Pani , Ajith Abraham and Yulan Liang, Ed., New York, NY: Springer, 2021, pp. 191–213.

[10] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Gener. Comput. Syst.*, vol. 81, pp. 41–52, Apr. 2018, doi: 10.1016/j.future.2017.10.047. Available: http://dx.doi.org/10.1016/j.future.2017.10.047.

[11] J. Gao, H. Wang, and H. Shen, "Machine learning based workload prediction in cloud computing," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, IEEE, Aug. 2020. doi: 10.1109/icccn49398.2020.9209730. Available: http://dx.doi.org/10.1109/icccn49398.2020.9209730.

[12] M. Kulkarni, P. Deshpande, S. Nalbalwar, and A. Nandgaonkar, "Cloud computing based workload prediction using cluster machine learning approach," in *Applied Computational Technologies*, in Smart innovation, systems and technologies. Singapore: Springer Nature Singapore, 2022, pp. 591–601. doi: 10.1007/978-981-19-2719-5_56. Available: http://dx.doi.org/10.1007/978-981-19-2719-5_56.

*Received 15/August/2023; Accepted 09/October/2023*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

[13] S. D. Khudhur and H. A. Jeiad, "LSDStrategy: A lightweight software-driven strategy for addressing big data variety of multimedia streaming," *IEEE Access*, vol. 10, pp. 111794–111810, 2022, doi: 10.1109/access.2022.3215531. Available: http://dx.doi.org/10.1109/access.2022.3215531.

[14] H. Abbasimehr, M. Shabani, and M. Yousefi, "An optimized model using LSTM network for demand forecasting," *Comput. Ind. Eng.*, vol. 143, no. 106435, p. 106435, May 2020, doi: 10.1016/j.cie.2020.106435. Available: http://dx.doi.org/10.1016/j.cie.2020.106435.

[15] T. Mehmood, S. Latif, and S. Malik, "Prediction of cloud computing resource utilization," in *2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT)*, IEEE, Oct. 2018. doi: 10.1109/honet.2018.8551339. Available: http://dx.doi.org/10.1109/honet.2018.8551339.

[16] Y. Yu, V. Jindal, F. Bastani, F. Li, and I.-L. Yen, "Improving the smartness of cloud management via machine learning based workload prediction," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, Jul. 2018. doi: 10.1109/compsac.2018.10200. Available: http://dx.doi.org/10.1109/compsac.2018.10200.

[17] N. Kaur and S. K. Sood, "Efficient Resource Management System Based on 4Vs of Big Data Streams," *Big Data Research*, vol. 9, pp. 98–106, 2017, doi: 10.1016/j.bdr.2017.02.002. Available: https://www.sciencedirect.com/science/article/pii/S2214579616300909.

[18] P. W. Murray, B. Agard, and M. A. Barajas, "Forecast of individual customer's demand from a large and noisy dataset," *Comput. Ind. Eng.*, vol. 118, pp. 33–43, Apr. 2018, doi: 10.1016/j.cie.2018.02.007. Available: http://dx.doi.org/10.1016/j.cie.2018.02.007.

[19] B. Sarıca, E. Eğrioğlu, and B. Aşıkgil, "A new hybrid method for time series forecasting: AR–ANFIS," *Neural Comput. Appl.*, vol. 29, no. 3, pp. 749–760, Feb. 2018, doi: 10.1007/s00521-016-2475-5. Available: http://dx.doi.org/10.1007/s00521-016-2475-5.

[20] A. R. S. Parmezan, V. M. A. Souza, and G. E. A. P. A. Batista, "Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model," *Inf. Sci. (Ny)*, vol. 484, pp. 302–337, May 2019, doi: 10.1016/j.ins.2019.01.076. Available: http://dx.doi.org/10.1016/j.ins.2019.01.076.

[21] N. Kaur, S. K. Sood, and P. Verma, "Cloud resource management using 3Vs of Internet of Big data streams," *Computing*, vol. 102, no. 6, pp. 1463–1485, Jun. 2020, doi: 10.1007/s00607-019-00732-5. Available: http://dx.doi.org/10.1007/s00607-019-00732-5.

[22] F. Martínez, M. P. Frías, M. D. Pérez-Godoy, and A. J. Rivera, "Dealing with seasonality by narrowing the training set in time series forecasting with k NN," *Expert Syst. Appl.*, vol. 103, pp. 38–48, Aug. 2018, doi: 10.1016/j.eswa.2018.03.005. Available: http://dx.doi.org/10.1016/j.eswa.2018.03.005.

[23] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters," *Procedia Comput. Sci.*, vol. 125, pp. 676–682, 2018, doi: 10.1016/j.procs.2017.12.087. Available: http://dx.doi.org/10.1016/j.procs.2017.12.087.

[24] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, Jan. 2019, doi: 10.1016/j.neucom.2018.09.082. Available: http://dx.doi.org/10.1016/j.neucom.2018.09.082.

[25] H. Shi, S. Hu, and J. Zhang, "LSTM based prediction algorithm and abnormal change detection for temperature in aerospace gyroscope shell," *Int. J. Intell. Comput. Cybern.*, vol. 12, no. 2, pp. 274–291, Jun. 2019, doi: 10.1108/ijicc-11-2018-0152. Available: http://dx.doi.org/10.1108/ijicc-11-2018-0152.

[26] F. Martínez, M. P. Frías, M. D. Pérez, and A. J. Rivera, "A methodology for applying k-nearest neighbor to time series forecasting," *Artif. Intell. Rev.*, vol. 52, no. 3, pp. 2019–2037, Oct. 2019, doi: 10.1007/s10462-017-9593-z. Available: http://dx.doi.org/10.1007/s10462-017-9593-z.

[27] D. Saxena and A. K. Singh, "Auto-adaptive learning-based workload forecasting in dynamic cloud environment," *Int. J. Comput. Appl.*, pp. 1–11, Oct. 2020, doi: 10.1080/1206212x.2020.1830245. Available: http://dx.doi.org/10.1080/1206212x.2020.1830245.

[28] J. Kumar, A. K. Singh, and R. Buyya, "Ensemble learning based predictive framework for virtual machine resource request prediction," *Neurocomputing*, vol. 397, pp. 20–30, Jul. 2020, doi: 10.1016/j.neucom.2020.02.014. Available: http://dx.doi.org/10.1016/j.neucom.2020.02.014.

[29] A. Rossi, A. Visentin, S. Prestwich, and K. N. Brown, "Bayesian uncertainty modelling for cloud workload prediction," in *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, IEEE, Jul. 2022. doi: 10.1109/cloud55607.2022.00018. Available: http://dx.doi.org/10.1109/cloud55607.2022.00018.

[30] S. Bansal and M. Kumar, "Deep learning-based workload prediction in cloud computing to enhance the performance," in *2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC)*, IEEE, May 2023. doi: 10.1109/icsccc58608.2023.10176790. Available: http://dx.doi.org/10.1109/icsccc58608.2023.10176790.

*Received 15/August/2023; Accepted 09/October/2023*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.24.1.7

[31] J. Kumar and A. K. Singh, "Cloud Resource Demand Prediction using Differential Evolution based Learning," in *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, IEEE, Jun. 2019. doi: 10.1109/icscc.2019.8843680. Available: http://dx.doi.org/10.1109/icscc.2019.8843680.

[32] J. Chen and Y. Wang, "A resource demand prediction method based on EEMD in cloud computing," *Procedia Comput. Sci.*, vol. 131, pp. 116–123, 2018, doi: 10.1016/j.procs.2018.04.193. Available: http://dx.doi.org/10.1016/j.procs.2018.04.193.

[33] S. D. Khudhur and H. A. Jeiad, "A Content-based File Identification Dataset: collection, construction, and evaluation," *Karbala int. j. mod. sci.*, vol. 8, no. 2, pp. 63–70, May 2022, doi: 10.33640/2405-609x.3222. Available: http://dx.doi.org/10.33640/2405-609x.3222.

[34] A. Q. Albayati, A. S. Al-Araji, and S. H. Ameen, "A Method of Deep Learning Tackles Sentiment Analysis Problem in Arabic Texts," *IJCCCE*, vol. 20, no. 4, Oct. 2020, doi: 10.33103/uot.ijccce.20.4.2. Available: https://www.uotechnology.edu.iq/ijccce/issues/2020/vol20/no.04/full-text/02.pdf.

[35] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks," *arXiv [cs.CL]*, Jul. 21, 2017. Available: http://arxiv.org/abs/1707.06799.

[36] B. S. Panda, A. Misra, and S. S. Gantayat, "Methods and concepts of data mining techniques to impute missing data information," *Far East J. Electron. Commun.*, vol. 20, no. 1, pp. 41–54, Apr. 2019, doi: 10.17654/ec020010041. Available: http://dx.doi.org/10.17654/ec020010041.

[37] S. Agarwal, "Data Mining: Data Mining Concepts and Techniques," in *2013 International Conference on Machine Intelligence and Research Advancement*, IEEE, Dec. 2013. doi: 10.1109/icmira.2013.45. Available: http://dx.doi.org/10.1109/icmira.2013.45.

[38] T. Wang, Y. Li, W. Chang, and S. Zhou, "A bagging ensemble learning traffic demand prediction model based on improved LSTM and transformer," in *Third International Conference on Computer Science and Communication Technology (ICCSCT 2022)*, Y. Lu and C. Cheng, Eds., SPIE, Dec. 2022. doi: 10.1117/12.2662894. Available: http://dx.doi.org/10.1117/12.2662894.

[39] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?," Feb. 28, 2014. doi: 10.5194/gmdd-7-1525-2014. Available: http://dx.doi.org/10.5194/gmdd-7-1525-2014.

*Received 15/August/2023; Accepted 09/October/2023*
 *ISSN (Print) 1811-9212* *ISSN (Online) 2617-3352*