**DEEP RNN-BASED TRAFFIC ANALYSIS SCHEME FOR DETECTING TARGET APPLICATIONS**

### AMINORPROJECTREPORT

#### Submittedby

**DIBYAJYOTI GANGULY [RA2011026010122]**

**TARUN NEGI [RA2011026010132]**

*Undertheguidanceof*

**Dr Maivizhi R**

(*Assistant Professor*)

#### inpartialfulfillment fortheawardofthedegree of

### BACHELOROFTECHNOLOGY

in

**COMPUTERSCIENCE &ENGINEERING**

of

### FACULTYOF ENGINEERINGANDTECHNOLOGY



**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR– 603 203**

**NOVEMBER 2023**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGYKATTANKULATHUR–603 203

**BONAFIDECERTIFICATE**

##### Certifiedthat18CSP107Lminorprojectreporttitled “**DEEP RNN-BASED TRAFFIC ANALYSIS SCHEME FOR DETECTING TARGET APPLICATIONS**” is the bonafide work of “**DIBYAJYOTI GANGULY[RA2011026010122], TARUN NEGI[RA2011026010132]”**who carried out theminor projectwork under my supervision. Certified further, that to the best of myknowledge the work reported herein does not form any other project report ordissertation on the basis of which a degree or award was conferred on an earlieroccasiononthisoranyother candidate.

###### SIGNATURE

Dr. Maivizhi R

###### GUIDE

Assistant Professor

Department of Computational Intelligence, School of Computing, SRMIST, KTR

###### SIGNATURE

Dr. Annie Uthra

###### PROFESSOR AND HEAD

Department ofComputational Intelligence,

School of Computing, SRMIST, KTR

**SIGNATURE**

Dr. Hariharan B

**PANEL HEAD**

Associate Professor

Dept.ofComputational Intelligence,

School of Computing, SRMIST, KTR

# ABSTRACT

##### Network security and performance optimization face major problems due to the development of various applications and developing attack vectors, which in turn drive the growing complexity of network traffic. To handle this issue, a new approach to categorization learning is proposed, in which two-dimensional graphics of traffic packets and target applications are used to represent input features and output labels, respectively. The proposed method classifies network packets into preset application categories by analyzing them in real-time and utilizing the capabilities of recurrent neural networks and deep learning. The plan uses a commercial deep long short-term memory system to analyze traffic quickly and precisely. Experiments based on simulations show that the minimal complexity and great accuracy of the system reach 99.82%. Tools like Wireshark, which are used for packet analysis and network environment monitoring, could benefit from this research.

**ACKNOWLEDGEMENT**

During the course of the project, we received a lot of help from several people who gave their precious time and guidance which helped us to successfully complete this project, so we would like to thank them all. We would like to take this opportunity to thank **Dr. Annie Uthra**, Professor and Head, Department of Computational Intelligence, for allowing us to present our project and report on "Deep RNN-based traffic analysis scheme for detecting target applications".We would like to convey our gratitude to our project guide, **Dr. Maivizhi R**, Assistant Professor, Department of Computational Intelligence, and panel head, **Dr. Hariharan B**, Associate Professor, Department of Computational Intelligence for their guidance, valuable advice, and cooperation without which this projectwould nothave been possible.

**Tarun Negi DibyajyotiGanguly**

**RA2011026010132 RA2011026010122**

**TABLEOFCONTENTS**

**Chapter No. Title PageNo.**

[ABSTRACT](#_bookmark0) iii

ACKNOWLEDGEMENT iv

[TABLEOF CONTENTS v](#_TOC_250001)

[LIST OFFIGURES](#_bookmark1) vi

[ABBREVIATIONS](#_bookmark2)vii

|  |  |  |  |
| --- | --- | --- | --- |
|  | [**1 INTRODUCTION**](#_bookmark3) | | **1** |
| 1.1Motivation | | 2 |
| 1.2SoftwareRequirementsSpecification | | 3 |
| 1.3 Organization of the report | | 3 |
|  | |  |
| **2 LITERATURESURVEY** | | **4** |
| **3 SYSTEMARCHITECTUREANDDESIGN** | | **6** |
| 3.1 Proposed Architecture Diagram | | 6 |
| 3.2 DesignofModules | | 6 |
|  |  | |  |
|  | **4 METHODOLOGY** | | **9** |
|  | **5 CODINGANDTESTING** | | **10** |
|  |  | |  |
|  | **6RESULTSANDDISCUSSIONS** | | **13** |
|  |  | |  |
| **7** | | **CONCLUSIONANDFUTUREENHANCEMENT** | **16** |
|  | |  |  |
|  | | **REFERENCES** | **17** |
| **APPENDIX** | |  |  |
|  | | **PLAGIARISMREPORT** |  |

**LISTOFFIGURES**

**FigureNo. FigureName PageNo.**

###### 3.1 Architecture Diagram for Website Traffic Forecasting 6

###### 6.1 Traffic Prediction using LSTM for 24 hours 13

###### 6.2 Comparison of LSTM with other models 14

# ABBREVIATIONS

RNNRecurrent Neural Network

LSTM Long Short-Term Memory

SRS Software Requirements Specification

GRU Gated Recurrent Unit

IoT Internet of Things

FRC Fraudulent Resource Consumption

MSE Mean Squared Error

MAE Mean Absolute Error

RMSE Root Mean Squared Error

## CHAPTER 1

**INTRODUCTION**

In a time when the internet is a vital part of our everyday existence, network management, cybersecurity, and application optimization all depend on being able to comprehend and track network traffic. In order to guarantee that networks operate effectively, identify security risks, and improve service quality, it is now crucial to identify and analyze network traffic due to the quick growth of internet-based services and applications.

Traditionally, network traffic analysis has involved analyzing data packets to determine the type, destination, and source of traffic. Although helpful, this method has limitations when it comes to identifying the types of applications and services that are driving the traffic. The increase in encrypted and obfuscated traffic makes traffic analysis an even more difficult task. In addition, new services and applications are constantly being added to contemporary networks, which are dynamic and ever-evolving. It becomes extremely difficult to identify and categorize these applications, especially given their ongoing updates and shifting communication patterns.

An ever-growing dependence on digital applications and data-driven decision-making have made precise and efficient network traffic monitoring crucial. Numerous applications, such as network management, security, and quality of service optimization, depend on an understanding of the traffic composition of the network.

This necessitates the use of cutting-edge methods, and the Deep Recurrent Neural Network (RNN)-based traffic analysis technique is one such novel way. Recognized for their capacity to represent sequential data and grasp temporal connections, RNNs have become more important in the artificial intelligence community. Deep RNN-based traffic analysis enables us to better understand network behavior, pinpoint specific applications, and improve our network security and management tactics. This method offers a significant advance in the field of traffic analysis since it provides precise identification as well as adaptability to the constantly changing environment of network applications.

1

The objective of the proposed method is to develop and deploy a deep neural network architecture that can identify target applications by analyzing network traffic sequences. A wide range of network traffic, including both well-known and recently developed applications, will be used to train the model.By assessing the Deep RNN-based approach's accuracy, mean squared error in identifying target applications across a range of network scenarios, its efficacy will be thoroughly assessed. There will be comparison studies done with conventional traffic analysis methods.

The proposed method will investigate scenarios of real-world applications, such as network management, quality of service optimization, and security. Network administrators can prioritize network resources, better understand and regulate application use, and spot possible security risks by identifying target application

The proposed method focuses on applying deep learning methods to analyze network traffic data in order to detect applications. The necessity for effective traffic analysis and management has grown due to the growth of various traffic data and the rising demand for network services. This paper presents a novel approach to classification learning that uses deep learning—more particularly, Long Short-Term Memory (LSTM) networks—to discover target applications and categorize network traffic with high accuracy.

**1.1 Motivation**

* 1. **Complexity of Network Traffic:**
  2. Modern networks carry diverse types of traffic from various applications. Traditional methods that rely on simple heuristics or shallow models struggle to accurately identify and classify this complex traffic
  3. **Dynamic Nature of Applications:**
  4. Applications are constantly evolving, and new applications with different communication patterns emerge regularly. Deep RNNs, with their ability to capture temporal dependencies and patterns, can adapt to these dynamic changes and learn from evolving traffic behavior.

2

**Encrypted Traffic:**

With the increasing use of encryption protocols, a significant portion of network traffic is now encrypted. Deep learning techniques, such as deep RNNs, can be effective in analyzing encrypted traffic patterns and making predictions based on the learned representations.

**Improved Accuracy:**

Deep learning models, particularly those with recurrent structures, have shown superior performance in capturing intricate patterns and dependencies in sequential data. This can lead to higher accuracy in classifying and detecting target applications within network traffic.

**Anomaly Detection:**

Deep RNNs can be trained to recognize normal patterns of network behavior, allowing them to detect anomalies or deviations from the expected traffic. This is crucial for identifying potential security threats or abnormal activities.

**1.2 Software Requirements Specification**

The entirety of this project has been coded and implemented on Jupyter Notebook, aweb-based interactive computing platform.

**1.3 Organization of the Report**

The report is organized as follows:

**Chapter 1** consists of a general introduction to the study and problem statement.

**Chapter 2** includes the Literature survey of the various methodologies used in this project.

**Chapter 3**includes the System design which includes the proposed architecture of this project.

**Chapter 4** consists of a detailed description of the proposed methodology.

**Chapter 5** includes the coding and implementation partof the project.

**Chapter 6** consists of the results and discussions of the proposed methodology.

**Chapter 7** lists the conclusions drawn and identifies the future enhancements possible for the methodology conducted.

## CHAPTER 2

**LITERATURE SURVEY**

In 2022, the paper authored by Xianbin Wang and Yongqin Fu and based on [1] "Traffic Prediction-Enabled Energy-Efficient Dynamic Computing Resource Allocation in CRAN Based on Deep Learning" was published. The suggested methodology has a number of benefits. First, it uses a new two-dimensional CNN LSTM model with temporal aggregation to predict wireless traffic, which improves prediction accuracy by identifying temporal dependencies and spatial correlations. This methodology may have some drawbacks despite its advantages. First off, the deep learning-based method for traffic prediction might need a lot of training data and processing power, which might make it less appropriate for smaller-scale implementations.

In 2021, the paper authored by Prajwal Kaushal, related toa traffic analysis scheme based on [2] deep learning and Long Short Term Memory (LSTM) networks was published. In order to achieve high accuracy in classifying traffic into target applications, network traffic data was preprocessed and converted into a format that was appropriate for LSTM learning. The suggested system classified network traffic with high accuracy (99.82%). It effectively used LSTM to process sequential data, making it appropriate for real-time traffic analysis. The necessity of precise traffic classification in contemporary networks is discussed in the paper. Depending on the payload size and number of packets per flow, the accuracy may change.

On May 18, 2022, the paper authored by ImtiazUllah and Qusay H. Mahmoud, and titled [3] "Design and Development of RNN Anomaly Detection Model for IoT Networks" was published.The proposed methodology offers a number of recurrent neural network (RNN)-based anomaly detection approaches for Internet of Things (IoT) networks, including Long Short Term Memory (LSTM), BiLSTM, and Gated Recurrent Unit (GRU) approaches.The models' capacity to identify anomalies is improved by the incorporation of deep learning techniques, such as LSTM, BiLSTM, and GRU, which enable the models to extract intricate patterns from IoT network data.The proposed work examines several RNN architectures, such as GRU andBiLSTM. These architectures are all appropriate for various kinds of sequential data analysis because of their unique features.

In 2019, the paper "Machine Learning Based Web-Traffic Analysis for Detection of [4] Fraudulent Resource Consumption Attack in Cloud" authoredby Samant Saurabh andAyush Prasad was published. By segmenting web pages into quantiles, the authors suggested a novel method for identifying Fraudulent Resource Consumption (FRC) attacks. Even for low attack percentages, the method efficiently and accurately detects FRC attacks, which qualifies it for cloud security. The technique depends on a number of popularity assumptions for the page, and it necessitates careful tuning when choosing parameters like the number of quantiles.The suggested method, according to the paper, can effectively and precisely identify FRC attacks, even in situations where the proportion of such attacks is low.

The paper "A Hybrid Approach for Web Traffic Prediction Using Deep Learning Algorithms" [5] was written in 2022 by AnupamaPrasanth, Bindhya Thomas, Densy John, and Priyanka Surendran. The authors suggested a hybrid model for predicting web traffic that combines ensemble learning to combine Long Short Term Memory (LSTM) and Radial Basis Functional Networks (RBFNs). By combining LSTM and RBFN, the hybrid model improves web traffic prediction accuracy. Because it makes use of both algorithms' advantages, it can be applied in practical settings. Although the paper presents a promising hybrid model, it is important to remember that it might not offer a thorough analysis of the drawbacks and difficulties that could arise when implementing the model in practical settings. These might include deep learning models like LSTM, which frequently need large amounts of training data.

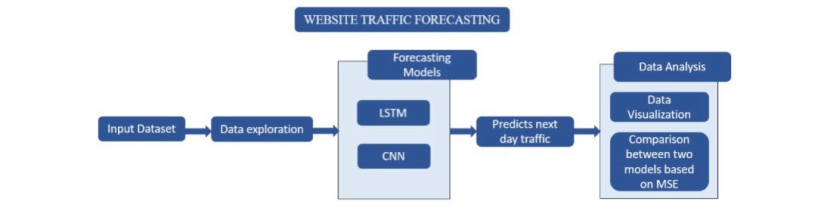
In 2023, M. Ramesh andKusumaKatragadda have proposed Website Traffic Forecasting Using Deep Learning Techniques [6]. Based on previous data, the authors used Long Short-Term Memory (LSTM) models and Convolutional Neural Networks (CNN) to predict website traffic. They contrasted how well these models predicted web traffic. The proposed methodology showed that CNN and LSTM models are good at accurately predicting website traffic. A smaller mean squared error was attained by LSTM, suggesting improved performance. The methodology doesn't explore how these models might be applied outside of the dataset that was employed. The issues of scalability and practical deployment were not fully covered. The generalizability of the models are also not thoroughly examined, which is one of the limitations.

## 

## CHAPTER 3

**ARCHITECTURE AND DESIGN**

**Proposed Architecture**



###### Figure 3.1:Architecture Diagram for Website Traffic Forecasting

Figure 3.1 displays the architecture diagram for web traffic forecasting. Initially, a dataset is considered that includes the features Session and Hour Index. Then, the dataset is analyzed to find the trends. Afterwards,the data is used to train the models and project the traffic for the next day.

3.1 Input dataset

3.2 Data exploraration

3.3

## CHAPTER 4

**PROPOSED METHODOLOGY**

**Dataset Description**

The dataset used in this project is taken from Kaggle. The web traffic dataset contains two columns – Hour Index and Sessions. There are 4896 rows in the dataset. The first column in the table lists the hours. The second column in the table is Session is the volume of traffic at an hourly level. It is a data set with a six-month series.

For recurrent neural networks (RNNs) to identify long-term dependencies in data, a particular type of architecture is needed. An RNN cell's performance and that of an LSTM are quite comparable. Whether or not the data from the previous timestamp should be remembered is determined in the first section.

**Algorithm Description**

**Require**: webtraffic.csv file containing session data

**Ensure**: Comparison plot of true values and predicted values using LSTM, Simple RNN, GRU, and Conv1D models

**for**model\_type in model\_types:

# **Build model**

model = build\_model(model\_type)

# **Train the model**

train\_model(model, X\_train, y\_train, X\_test, y\_test)

**# Load the best weights for the model**

model.load\_weights('best\_model.hdf5')

**# Evaluate the model on the training data**

mse = evaluate\_model(model, X\_train, y\_train)

print(f"{model\_type} - Mean Square Error: {mse}")

**# Forecast future values using the trained model**

y\_pred = forecast(model, X\_test, no\_of\_pred, ind)

**# Plot true vs predicted value**s

plot\_comparison(y\_test[ind:ind+no\_of\_pred], y\_pred, model\_type=model\_type)

## 9

## CHAPTER 5

**CODING AND TESTING**

import pandas aspd

importnumpyas np

importmatplotlib.pyplotasplt

fromsklearn.preprocessingimportStandardScaler

fromtensorflowimportkeras

fromkeras.modelsimport Sequential

fromkeras.layersimport LSTM,SimpleRNN, GRU, Dense, Conv1D

fromkeras.callbacksimportModelCheckpoint

data=pd.read\_csv("webtraffic.csv")

data.shape

data.head()

sessions=data['Sessions'].values

ar=np.arange(len(sessions))

plt.figure(figsize=(22,10))

plt.plot(ar,sessions,'r')

plt.show()

sample=sessions[:168]

ar=np.arange(len(sample))

plt.figure(figsize=(22,10))

plt.plot(ar,sample,'r')

plt.show()

defprepare\_data(seq,num):

    x=[]

    y=[]

    foriinrange(0,(len(seq)-num),1):

        input\_=seq[i:i+num]

        output=seq[i+num]

        x.append(input\_)

        y.append(output)

    returnnp.array(x),np.array(y)

num=168

x,y=prepare\_data(sessions,num)

print(len(x))

ind=int(0.9\*len(x))

# prepare training and test data

X\_train=x[:ind]

y\_train=y[:ind]

X\_test=x[ind:]

y\_test=y[ind:]

X\_scaler=StandardScaler()

X\_train=X\_scaler.fit\_transform(X\_train)

X\_test=X\_scaler.fit\_transform(X\_test)

y\_train=y\_train.reshape(len(y\_train),1)

y\_test=y\_test.reshape(len(y\_test),1)

y\_scaler=StandardScaler()

y\_train=y\_scaler.fit\_transform(y\_train)[:,0]

y\_test=y\_scaler.fit\_transform(y\_test)[:,0]

X\_train=X\_train.reshape(X\_train.shape[0],X\_train.shape[1],1)

X\_test=X\_test.reshape(X\_test.shape[0],X\_test.shape[1],1)

print(X\_train.shape)

model= Sequential()

model.add(LSTM(128,input\_shape=(168,1)))

model.add(Dense(64,activation='relu'))

model.add(Dense(1,activation='linear'))

model.summary()

model.compile(loss='mse',optimizer='adam')

mc=ModelCheckpoint('best\_model.hdf5',monitor='val\_loss',verbose=1,save\_best\_only=True,mode=min)

# train the model

history=model.fit(X\_train,y\_train,epochs=30,batch\_size=32,

                   validation\_data=(X\_test,y\_test),callbacks=[mc])

model.load\_weights('best\_model.hdf5')

mse=model.evaluate(X\_train,y\_train)

print("Mean Square Error:",mse)

defForecast(X\_test,no\_of\_pred,ind):

    predictions=[]

    temp=X\_test[ind]

    foriinrange(no\_of\_pred):

        pred=model.predict(temp.reshape(1,-1,1))[0][0]

        temp=np.insert(temp,len(temp),pred)

        predictions.append(pred)

        temp=temp[1:]

    returnpredictions

no\_of\_pred=24

ind=72

y\_pred=Forecast(X\_test,no\_of\_pred,ind)

y\_true=y\_test[ind:ind+(no\_of\_pred)]

defplot(y\_true,y\_pred):

  ar=np.arange(len(y\_true))

  plt.figure(figsize=(22,10))

  plt.plot(ar,y\_true,'r')

  plt.plot(ar,y\_pred,'y')

  plt.show()

plot(y\_true,y\_pred)

# Create and compile the Simple RNN model

model\_rnn=Sequential()

model\_rnn.add(SimpleRNN(128,input\_shape=(168,1)))

model\_rnn.add(Dense(64,activation='relu'))

model\_rnn.add(Dense(1,activation='linear'))

model\_rnn.compile(loss='mse',optimizer='adam')

# Train the Simple RNN model

history\_rnn=model\_rnn.fit(X\_train,y\_train,epochs=30,batch\_size=32,

                            validation\_data=(X\_test,y\_test),callbacks=[mc])

model\_rnn.load\_weights('best\_model.hdf5')

# Create and compile the GRU model

model\_gru=Sequential()

model\_gru.add(GRU(128,input\_shape=(168,1)))  # Add the missing closing parenthesis here

model\_gru.add(Dense(64,activation='relu'))

model\_gru.add(Dense(1,activation='linear'))

model\_gru.compile(loss='mse',optimizer='adam')

# Train the GRU model

history\_gru=model\_gru.fit(X\_train,y\_train,epochs=30,batch\_size=32,

                          validation\_data=(X\_test,y\_test),callbacks=[mc])

model\_gru.load\_weights('best\_model.hdf5')

# Create and compile the Conv1D model

model\_conv1d=Sequential()

model\_conv1d.add(Conv1D(128,kernel\_size=3,activation='relu',input\_shape=(168,1)))

model\_conv1d.add(Flatten())

model\_conv1d.add(Dense(64,activation='relu'))

model\_conv1d.add(Dense(1,activation='linear'))

model\_conv1d.compile(loss='mse',optimizer='adam')

# Train the Conv1D model

history\_conv1d=model\_conv1d.fit(X\_train,y\_train,epochs=30,batch\_size=32,

                                  validation\_data=(X\_test,y\_test),callbacks=[mc])

model\_conv1d.load\_weights('best\_model.hdf5')

defforecast(model,X\_test,no\_of\_pred,ind):

    predictions=[]

    temp=X\_test[ind]

    foriinrange(no\_of\_pred):

        pred=model.predict(temp.reshape(1,-1,1))[0][0]

        temp=np.insert(temp,len(temp),pred)

        predictions.append(pred)

        temp=temp[1:]

    returnpredictions

y\_pred\_lstm=forecast(model,X\_test,no\_of\_pred,ind)

y\_pred\_rnn=forecast(model\_rnn,X\_test,no\_of\_pred,ind)

y\_pred\_gru=forecast(model\_gru,X\_test,no\_of\_pred,ind)

y\_pred\_conv1d=forecast(model\_conv1d,X\_test,no\_of\_pred,ind)

defplot\_comparison(y\_true,y\_pred\_lstm,y\_pred\_rnn,y\_pred\_gru,y\_pred\_conv1d):

    ar=np.arange(len(y\_true))

    plt.figure(figsize=(22,10))

    plt.plot(ar,y\_true,'r',label="True")

    plt.plot(ar,y\_pred\_lstm,'y',label="LSTM")

    plt.plot(ar,y\_pred\_rnn,'g',label="Simple RNN")

    plt.plot(ar,y\_pred\_gru,'b',label="GRU")

    plt.plot(ar,y\_pred\_conv1d,'c',label="Conv1D")

    plt.legend()

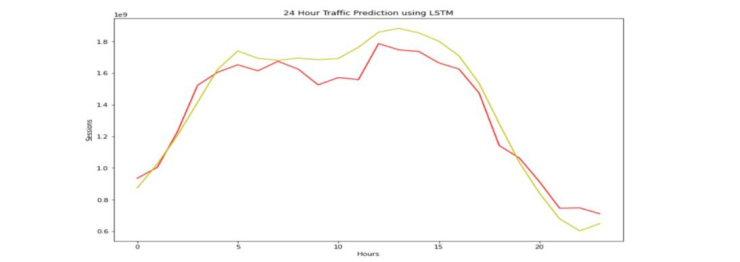
    plt.show()

plot\_comparison(y\_true,y\_pred\_lstm,y\_pred\_rnn,y\_pred\_gru,y\_pred\_conv1d)

## CHAPTER 6

**RESULTS AND DISCUSSIONS**

A popular metric for regression applications, like time series forecasting, is mean square error (MSE). It measures the discrepancy between a dataset's actual (ground truth) values and predicted values. The Mean Squared Error (MSE) computes the mean of the squared deviations between the expected and observed values. In mathematics, it is frequently expressed as: \* Σ(actual - predicted)^2 \* MSE.



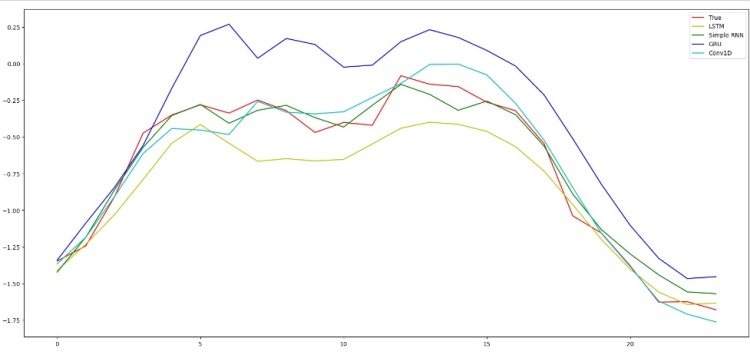
**Fig. 6.1 Traffic Prediction Using LSTM for 24 hours**

The mean squared error for the validation data is only 0.014, for the LSTM model. We use this mean squared error to measure the model's performance. MSE is a metric that quantifies how well the model's predictions match the real data. A lower mean square error (MSE) denotes a higher predictive accuracy of the model as its predictions are more closely aligned with the actual values. A 0.014 MSE is a comparatively low value. Lower values are preferable in the context of MSE because they show that the model's predictions and actual values are fairly close. This suggests that the web traffic forecasting performance of the LSTM model is good.

MSE is frequently used to evaluate and compare various models during the model-development process. To determine which LSTM model or other forecasting method works best for your particular web traffic forecasting task, we can assess several models and compare their MSE values.

Mean Squared Error (MSE) is a predictive accuracy metric, but it has restrictions. The squaring operation penalizes large prediction errors more severely. In some situations, to obtain a more thorough picture of the model's performance, we might also wish to take into account additional assessment metrics, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or domain-specific metrics.

An LSTM model performs well in predicting web traffic if its MSE value for the validation data is 0.014, which shows that the model's predictions closely match the real data. When interpreting and comparing MSE values, it's crucial to take the context and domain-specific requirements into account.



**Fig. 6.2. Comparison of LSTM with other models**

LSTM can capture both short-term and long-term dependencies in time series data effectively. It is less prone to vanishing gradient problems compared to Simple RNN. Suitable for tasks with complex temporal patterns and long-range dependencies. Simple RNNs are limited by their ability to capture only short-term dependencies.They are more likely to suffer from vanishing gradient problems when dealing with long sequences.Suitable for tasks where short-term patterns dominate, and computational resources are limited.LSTM is likely to outperform Simple RNN when the time series data has significant long-term dependencies or complex temporal patterns. It can capture both short and long-term trends effectively.

LSTM is suitable for tasks with complex temporal dependencies and long-range dependencies. It can capture both short-term and long-term patterns. However, it may require more parameters and careful tuning.

GRU is a balance between complexity and efficiency.It performs well on tasks with moderate-length dependencies.It is computationally more efficient and often easier to train than LSTM.The choice between LSTM and GRU depends on the specific characteristics of the data and available computational resources. LSTM is more powerful for capturing long-range dependencies, but GRU may be preferred when a balance between accuracy and efficiency is required.

LSTM models are capable of capturing both local and global patterns in time series data. They are effective at handling sequences with varying lengths and complex temporal relationships. Conv1D models are focused on capturing local patterns within a sequence. They excel at recognizing short-term, local dependencies in data.LSTM can outperform Conv1D in scenarios where the time series data contains significant long-range dependencies and complex temporal relationships. Conv1D may be more suitable for tasks where the emphasis is on local, short-term patterns.

In summary, LSTM is a powerful choice for time series forecasting tasks, especially when the data exhibits both short-term and long-term dependencies. It excels at capturing complex temporal patterns. However, the choice of model should always be based on the specific characteristics of the data and the requirements of the task, and it may vary depending on the dataset and the computational resources available.

## CHAPTER 7

**CONCLUSION AND FUTURE ENHANCEMENTS**

The proposed methodology shows how well deep learning—in particular, LSTM—performs in traffic analysis and application identification. Due to its high classification accuracy, DR-TAS is a viable method for use in practical applications. Subsequent investigations could concentrate on applying this methodology in real networks and tackling issues associated with categorizing novel, unseen packets.

The main goal of our research is to create a forecasting algorithm that can precisely predict the future volume of traffic to Wikipedia pages. In online traffic time series, Long Short-Term Memory Recurrent Neural Networks exhibit superior accuracy and efficiency. Using this data, we trained the model to predict future online traffic for pages based on metrics such as hours and views, or sessions. It is possible to predict how many visitors the website will get in the future.

The improvement will be visible as more user data is added to the system. All websites can use the service to enhance business analysis and load control of online traffic. The usage of LSTM and RNN makes our system extremely beneficial. Time series forecasting is one of the least researched fields of forecasting, and several models are tested to improve forecast precision. The primary objective of the proposed work is to predict future web traffic in order to improve congestion control via decision-making. In forecasting future values, historical data is considered.

Lastly, further investigations regarding multivariate time series suggestions for streamlining instantaneous decision-making can be conducted.

**REFERENCES**

[1] Tondale, S., Yadav, S., &Ahir, S. (2020). Web traffic time series forecasting using ARIMA and LSTM RNN [Review of Web traffic time series forecasting using ARIMA and LSTM RNN].

[2] Golait, A., Pardeshi, S., &Javeri, R. (n.d.). Web Traffic Time Series Forecasting Using ARIMA Model [Review of Web Traffic Time Series Forecasting Using ARIMA Model].

[3] Zheng, J., & Huang, M. (n.d.). Traffic Flow Forecast Through Time Series Analysis Based on Deep Learning [Review of Traffic Flow Forecast Through Time Series Analysis Based on Deep Learning].

[4] Zhuo, Q., Yan, H., & Qi, Y. (2019). A network traffic prediction method based on LSTM. [Review of A network traffic prediction method based on LSTM.].

[5] Liu, H., Xu, H., & Zhang, X. (2022). Network Traffic Prediction Based on LSTM and Transfer Learning. [Review of Network Traffic Prediction Based on LSTM and Transfer Learning.].

[6] Sharma, S., Parashar, L., & Singh, U. (2020). Network traffic prediction using long short-term memory. [Review of Network traffic prediction using long short-term memory.].

[7] Yang, F. (2018). Research on network traffic prediction based on long short-term memory neural network. [Review of Research on network traffic prediction based on long short-term memory neural network.].

[8] Petluri , N., & Al-Masri, E. (2018). Wikipedia Page Traffic Prediction [Review of Wikipedia Page Traffic Prediction].

[9] Aggelos, &Prasanna., V. (2019). An LSTM framework for modeling network traffic [Review of An LSTM framework for modeling network traffic].

[10]Zheng, J., & Huang, M. (2020). Traffic Flow Forecasting Using Deep Learning and Time Series Analysis [Review of Traffic Flow Forecasting Using Deep Learning and Time Series Analysis].

[11] Shuvo, M. A. R., Zubair, M., Purnota, A. T., & Hossain, S. (2021). Traffic Forecasting Using Time-Series Analysis [Review of Traffic Forecasting Using Time-Series Analysis].

[12] Wunnava, & Praveen, V. (2020). Exploration of Wikipedia traffic data to analyze the relationship between multiple pages [Review of Exploration of Wikipedia traffic data to analyze the relationship between multiple pages].

[13] Miyaguchi, A., Chakrabarti, S., & Garcia, N. (2019). ForecastingWikipedia Page Views with Graph Embeddings [Review of ForecastingWikipedia Page Views with Graph Embeddings].

[14] Chakraborty, A., J.S. Banerjee, and A. Chattopadhyay*.*Non-uniform quantized data fusion rule alleviating control channel overhead for cooperative spectrum sensing in cognitive radio networks*in*2017 IEEE 7th International Advance Computing Conference (IACC)2017. IEEE.

[15] Chattopadhyay, A. (n.d.). Non-uniform quantized data fusion rule for data rate saving and reducing control channel overhead for cooperative spectrum sensing in cognitive radio networks. [Review of Non-uniform quantized data fusion rule for data rate saving and reducing control channel overhead for cooperative spectrum sensing in cognitive radio networks.].

[16] Rueda, *A.*A survey of traffic characterization techniques in telecommunication networks*in*Proceedings of 1996 Canadian Conference on Electrical and Computer Engineering1996. IEEE

[17] Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy [Review of Intrusion detection systems: A survey and taxonomy].

[18] Mukkamala, S., Janoski, G., & Sung, A. (2002). Intrusion detection: support vector machines and neural networks [Review of Intrusion detection: support vector machines and neural networks].

[19] Shafiq, M. (2016). Network traffic classification techniques and comparative analysis using machine learning algorithms [Review of Network traffic classification techniques and comparative analysis using machine learning algorithms].

[20] Zamani, M. (2009). A DDoS-aware IDS model based on danger theory and mobile agents [Review of A DDoS-aware IDS model based on danger theory and mobile agents].

[21] Zhou, W. (2011). Internet traffic classification using feed-forward neural network [Review of Internet traffic classification using feed-forward neural network].

**APPENDIX**

**PLAGIARISM REPORT**