Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Networked Systems and Services

PROJECT LABORATORY REPORT

Student Name : Emen Hadj Sassi

Neptun Code : K82ESY

Academic Year : 2023-2024

Abstract

The aim of this document is to introduce and explain the workshops that belong to project laboratory classes for the second term of the 2023-2024 academic year.

Table of Contents

List of Figures

1 Introduction

Traffic prediction is a fundamental research problem in the realm of transportation studies.
Over the past decades, significant strides have been made in both theoretical approaches and empirical applications in order to reduce traffic congestion since it has become an increasingly issue in many urban environments, posing challenges in areas such as fuel consumption, emission of pollutants, and overall travel time.
Thus, accurate prediction of traffic flow is essential for effective traffic management and planning.

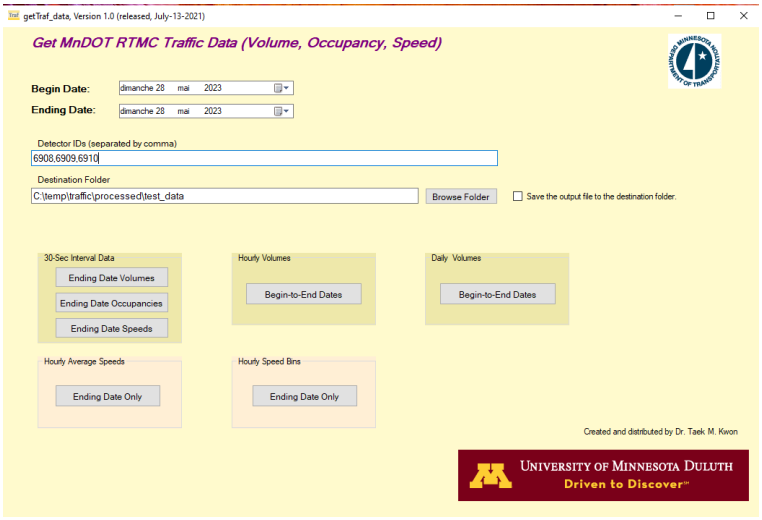
2 Methodology:

This section will discuss the procedures and techniques of the application used in predicting traffic patterns.
Our methodology is divided into several critical components : data collection, data preprocessing, model development, results and evaluation.
Each of these components plays a vital role in the accuracy and reliability of our traffic prediction outcomes.

2.1 Data Collection

We shall start with an in-depth exploration of the data collection process, which forms the foundation of our empirical analysis.
We utilized data collected from the Regional Transportation Management Center (RTMC), a division of the Minnesota Department of Transportation (MnDOT).
This data was gathered via two types of sensors: inductive loop detectors and Wavetronix radar detectors, strategically placed on the freeway network of Minneapolis and St. Paul Cities. Each lane had a designated sensor, with each detector identified by a unique ID number. The sensors collected a variety of data every 30 seconds, which included parameters like volumes, occupancies and speeds.
The data was available in multiple formats, such as 30-second, hourly, and daily metrics. To access this information, a software called "GetTrafdata" was used.

The software user interface consists of the top portion for setting up the begin/end dates and detector list; and the bottom portion for buttons on retrieving different types of traffic data.



'Figure 1 : GetTrafdata Software Interface'

Data retrieval is simple and done by three steps.
Step 1: Select the begin/end dates.
Step 2: Enter detector IDs.
Step 3: Press the button that matches with the data type you wish to retrieve.

All outputs are generated in CSV file.
This figure shows an hourly volume data for '6908', '6909', '6910' and the total.

	A	B	C	D	E	F
1	date	hou	6908	6909	6910	Total Vol
2	01/01/2022	0	259	288	83	630
3	01/01/2022	1	223	296	77	596
4	01/01/2022	2	135	243	84	462
5	01/01/2022	3	103	175	46	324
6	01/01/2022	4	96	95	7	198
7	01/01/2022	5	69	111	17	197
8	01/01/2022	6	116	138	17	271
9	01/01/2022	7	181	205	50	436
10	01/01/2022	8	237	274	72	583

'Figure 2 : CSV file'

2.2 Data Preprocessing

Following the collection of raw traffic data, our next step in the methodology involved data preprocessing. This crucial stage often determines the quality of the results, as it involves cleaning, organizing, and transforming the raw data into a suitable format for analysis.

2.2.1 Missing Values

First, we dealt with missing values by implementing a mean imputation approach. This allowed us to replace missing values with the mean of the respective variable, thus preserving the overall distribution and avoiding potential biases that could result from deletion.
In some cases, where data points formed a logical sequence, we also utilized forward and backward fill methods to replace missing values, making the assumption that they are similar to their adjacent data points.

2.2.2 Splitting Data

Once we ensured the completeness of the dataset, we proceeded with the data splitting process. This step divided our dataset into separate training and testing sets where training data is the subset of original data, whereas testing data is used to check the accuracy of the model.

2.2.3 Feature Scaling

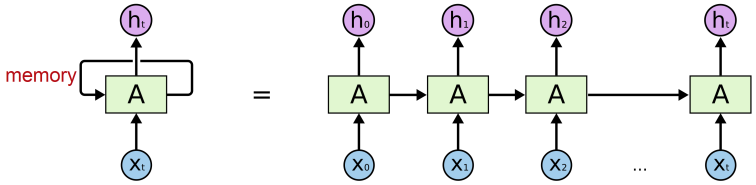
Feature scaling ensures that no particular feature dominates others due to differences in measurements or units.
We utilized two primary techniques: **Normalization** and **Standardization**.
Normalization transforms features to a scale ranging from 0 to 1.
This method ensures that our features have a similar scale, preventing any one feature from disproportionately influencing the model due to its numerical magnitude.
On the other hand, **Standardization** adjusts the features to have a mean of 0 and a standard deviation of 1.
Unlike normalization, standardization does not confine values to a specific range, which can be useful for certain algorithms that require the preservation of outlier information.

2.3 Model Architecture

In the model development stage, we opted to employ two types of deep learning algorithms to forecast traffic:
Recurrent Neural Networks (RNN) and **Long Short-Term Memory (LSTM)** networks.

2.3.1 Recurrent Neural Networks

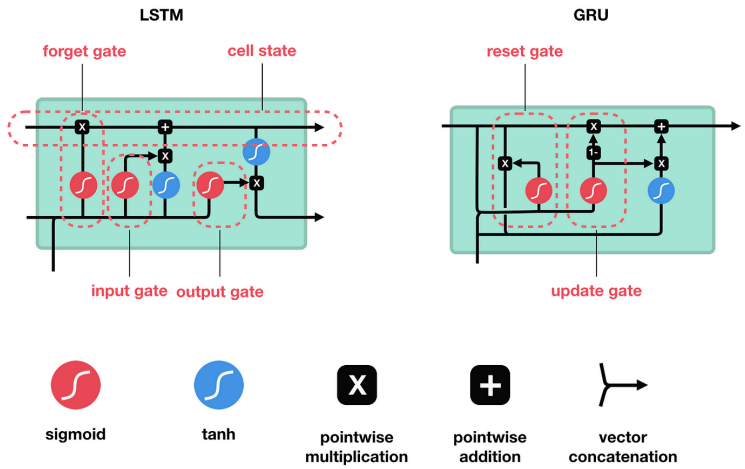
We initiated our process with RNN, a type of neural network particularly adept at handling sequential data, such as our time-series traffic data.
The key feature of RNN is its ability to use its internal memory to process sequences of inputs, allowing it to make use of temporal information. However, traditional RNNs have a shortcoming: as the gap between the relevant information and the point where it is needed grows, RNNs lose their ability to connect the information, also known as the vanishing gradient problem.



'Figure 3: Recurrent Neural Network'

2.3.2 Long Short-Term Memory

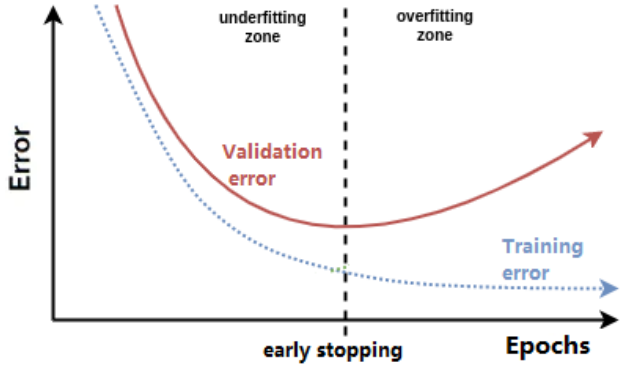
To circumvent this limitation of RNNs, we employed LSTM networks.
LSTMs, a special kind of RNN, are capable of learning long-term dependencies, making them highly suitable for our time-series traffic data. It is achieved through a series of 'gates' (input, forget, and output gates) which selectively control the information flow into and out of the memory cell.
This unique structure enables LSTM to remember or forget information over long periods, addressing the vanishing gradient problem inherent in standard RNNs.



'Figure 4: Long Short-Term Memory'

2.3.3 Early Stopping

Early stopping is a form of regularization used to prevent overfitting during the training phase of our models.
Overfitting typically occurs when a model learns not only from the underlying patterns of the training data but also from its noise, leading to poor performance on new, unseen data.
In our case, we set up early stopping rules to cease training when the validation loss did not decrease for a predetermined number of epochs, often called the 'patience'.



'Figure 5: Early stopping'

3 Evaluate the performance

In the process of assessing the predictive performance of our models, we leaned on two essential statistical metrics: **Mean Squared Error (MSE)** and **Mean Absolute Percentage Error (MAPE)**.

3.1 Mean Squared Error

We used MSE as our primary measure of model performance. This tool computes the average squared difference between the actual and predicted values, highlighting the magnitude of errors produced by the model.

```
# Calculate MSE for training and test sets

mse_test = np.mean((y_test - y_test_pred.flatten())**2)
mse_train = np.mean((y_train - y_train_pred.flatten())**2)
print(f'Mean Squared Error train: {mse_test}')
print(f'Mean Squared Error test: {mse_train}')
```

```
Mean Squared Error train: 2.0613081040669656
Mean Squared Error test: 1.9995541846912748
```

'Figure 6: Mean Squared Error'

3.2 Mean Absolute Percentage Error

In tandem with MSE, we utilized MAPE to offer a different perspective on model accuracy. MAPE calculates the mean absolute error as a percentage of the actual value, providing a relative measure of accuracy. This metric offers a straightforward, intuitive understanding of the error rate, which can be especially useful when comparing the performance across various datasets or different models.

```
# Calculate MAPE for training and test sets
mape_train = mean_absolute_percentage_error(y_train, y_train_pred)
mape_test = mean_absolute_percentage_error(y_test, y_test_pred)

print(f'Training MAPE: {mape_train}%')
print(f'Test MAPE: {mape_test}%')
```

```
Training MAPE: 2.6603522927608365%
Test MAPE: 1.9339970744402364%
```

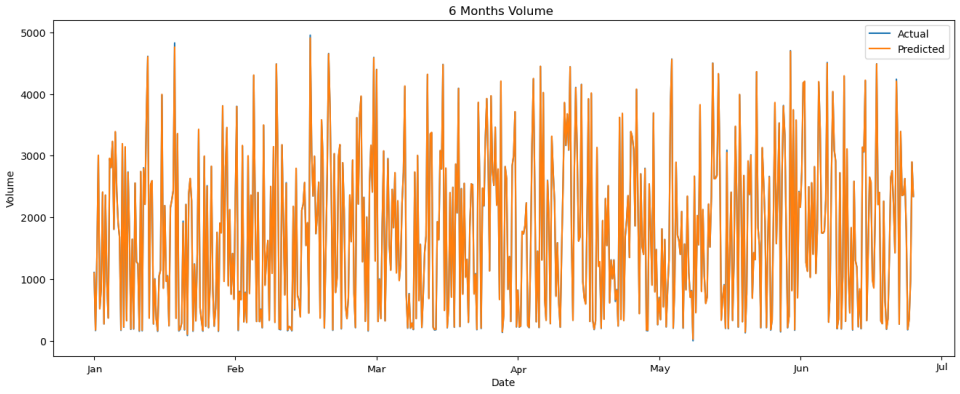
'Figure 7: Mean Absolute Percentage Error'

4 Results

4.1 The effects of RNN and LSTM

After applying both methods on our data, we can say that LSTM model exhibited lower MSE and MAPE values compared to the RNN model, indicating a superior ability to handle long-term dependencies in our time-series traffic data. The more sophisticated memory mechanism of LSTM effectively captured temporal dynamics, contributing to higher predictive accuracy.
The RNN model, while not performing as well as LSTM, still showed a reasonable ability to predict traffic patterns. Although its performance was limited by short-term memory, it was able to capture some level of temporal dependencies and yielded acceptable results.

4.2 Plot of the Original and Predicted Values



'Figure 8: Original and Predicted Values Plot During 6 Months'

This plot indicates high degree of accuracy in our predictions. The close alignment of the predicted values with the actual ones suggested that our model, particularly LSTM, was able to capture and predict the traffic patterns effectively.

5 Conclusion

In wrapping up this project, it's clear that our exploration into traffic prediction using advanced deep learning models has been successful. It underscores the significant potential of machine learning, specifically deep learning, in the area of traffic prediction.
The successful implementation and tangible results of this project suggest promising prospects for its application in real-world traffic management and urban planning scenarios.
As we move forward, these tools can become increasingly integral to driving more efficient, intelligent responses to traffic challenges.