uc3m | Universidad Carlos III de Madrid

Master Degree in Big Data Analytics

2021-2022

Master Thesis

Deep Learning Models to Predict the Traffic Intensity in Madrid City

BELCE KAYA

Supervisor: Mario Muñoz Organero

Madrid, Spain - September, 2022

SUMMARY

As the human population in the world increases and technology develops, the increase in car usage has witnessed a sharp rise from the past to the present. Therefore, nowadays, due to this increase, traffic has become one of the most important problems in urbanized settlements. For more livable city life, it is necessary to control vehicle traffic by determining the current density and taking the necessary measures for future congestion. To reduce the time spent in traffic, various applications have been developed that show people the fastest and most comfortable route from one point to another. Although these applications are suitable for short-term traffic congestion estimation in the traffic flow, the aim of this thesis is to provide a different approach to forecasting the traffic intensity on a regional basis by using sensor data from the sensors placed in URB and M30 road types in 21 districts of Madrid. Sensor data and zone areas of each sensor were obtained from the Madrid open data portal. In this paper, first, the related works similar to this problem have been studied and the approaches of these studies have been placed in the literature review part. After broad research, the success of Convolutional neural networks (CNN) and Long-short term Memory (LSTM) which is a special type of recurrent neural network (RNN) has been witnessed in the state of the art. In the proposed model, CNN and LSTM model combinations have been implemented in sequence to sequence modeling approach. Before training the models, the sensor and zone data have been analyzed in detail and various data manipulations have been made to prepare the data for deep learning models. Apart from numeric sensor data, another experiment has been made with the traffic intensity map images. The images have been produced by using the geographical information in the sensor data set. First, the pre-processed numeric data has been trained with 24-minute time steps which correspond to a 6-hour period and the regional traffic intensity has been estimated for the next 4 time steps which equate to a 1-hour period. Then as a second experiment, the regional traffic intensity images have been forecasted with the same defined looking back and forecasting time step periods. This experiment has proved that the deep learning models learning capability showed a good forecasting performance and the models could learn the 15-minute interval traffic intensity flow from the images produced. The final results have shown that even though the experiment of training the model with traffic intensity images could perform more than the baseline model and could learn the traffic intensity flow information, the models' performance with the numerical data has over-performed.

Keywords: Deep Learning, Traffic Intensity Forecasting, Convolutional Neural Network, Long Short-Term Memory, Recurrent Neural Network, Image Processing

DEDICATION

I would like to express my endless thanks to my esteemed professor Prof. Dr. Mario Muñoz Organero, who did not spare his interest and support in the planning and research of this thesis, and benefiting from his knowledge and experience. Besides, I am grateful to my dear family, who has never lost their moral and financial support throughout my academic life.

Belce Kaya

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Nowadays, with the evolution of human life, industrialization, and trade, the importance of transportation is increasing day by day. Especially with the concepts of time and comfort, the number of vehicle owners in various societies is increasing from the past to the present, and this causes an increase in the traffic density of the cities. When this is the case, it is quite crucial to observe, control, analyze and make predictions of this enormous flow by using traffic sensors. Therefore, to increase the efficiency of traffic management and to inform people about the traffic intensity of the areas in a city, smart transportation systems have been started to implement in many cities, and nowadays, many cities are in the concept of "smart city", and Madrid city is one of those.

In this paper, the historical traffic sensor data from the open data portal of Madrid city was used to generate short-term regional basis traffic intensity predictions. In the study, sequence-to-sequence deep learning-based traffic intensity prediction models were experimented. Two types of input data representation were used, numerical traffic data and its regional static traffic intensity map images. As the data had also geographical information, the traffic data wanted to be visualized as time-distributed static intensity maps. In the Literature Review section, the related studies were made according to the scope of the thesis have been reviewed. In the section of Data Acquisition and Description, first, the data sources used have been described, and in the rest of this section, as the data sources were in the website of the open data portal, the designed data pipeline to do web scraping has explained in detail. In the following chapter called Exploratory Data Analysis, a comprehensive data analysis was carried out to detect the missing traffic sensor data, to analyze the traffic information both based on the district and sensor type, and to get crucial insights from Madrid traffic life. After that, in the section of Experiments, the proposed system to forecast the short-term traffic intensity of the regions of Madrid have been explained. Finally, in the last part of the paper which is the section of Conclusions and Future Improvements, all the studies carried out were briefly summarized and some hypotheses have been mentioned which might increase the current model performance in this study under the name of future studies.

2. LITERATURE REVIEW

Nowadays, due to the increasing population density and people's preference for their private vehicles, there is a very high traffic density in big cities. To track, manage and estimate the traffic volume, a lot of research has been done in the field of traffic volume and speed, traffic congestion, and traffic flow estimation, covering a wide variety of statistical, machine learning, and deep learning techniques. As traffic data represents a time series, in the topic of time series forecasting, in the past, many researchers mainly were using traditional statistical time series model approaches such as Autoregressive Integrated Moving Average Model (ARIMA). [1] However, a large number of studies have found that the traffic flow data are random and nonlinear. The ARIMA algorithm is not a good method to analyze the nonlinear traffic data because it is based on a linear relationship. [1] Moreover, some machine learning and deep learning methods have also been proposed, such as Support vector machines (SVM), K-Nearest Neighbors (KNN), Convolutional neural networks (CNN), Recurrent neural networks (RNN), and so on. [2] Furthermore, today, as computer vision has gained great importance in recent years due to the increasing availability of geographic data, various studies related to the estimation of traffic data by using traffic images could be seen in the literature. Images captured by satellites and sensors are the most used ones in the literature. [3] Moreover, density images such as heat-maps [4] and Time-Space Diagrams [5] are typically seen in literature.

When the studies carried out for the estimation of traffic in the literature are examined, In the study of Y. Duan et al. [6], traffic speed data were collected at 30-second intervals from more than 15000 sensors located in different states in California highways. The speed data obtained were pre-processed at 5-minute intervals, and models were created to predict future speed information for 15, 30, 45, and 60 minutes with deep learning algorithms. The proposed model was evaluated with mean absolute error (MAE), mean relative error (MRE) and root mean square error (RMSE) metrics. The MAPE value was calculated as the percentage of 34.1 when 15 minutes were predicted. Furthermore, in the study of W. Jin et al. [7], Spatiotemporal Iterative Convolutional Networks, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) methods were combined. It was stated that the obtained method gives better results than classical time series and other deep learning methods in short-term traffic density estimation.

In the study of N. Ranjanm et al. [8] which is about the estimation of traffic congestion according to the past observation of traffic congestion data, hybrid deep learning model has been implemented, combining CNN, LSTM and Transpose CNN. In the study, the images have been provided by the snapshot of traffic congestion maps from an open-source online web service. The traffic congestion images had three congestion levels (Jam, Slow, and Free State). Red colour was representing the Jam class, yellow and green colors were displaying the Slow and Free states, respectively. Those images have been

passed through a pre-processing pipeline which contains masking operation to extract the congestion level from the image, "bitwise and" to generate the image with the only road network. The results of this process shown on the below figure.

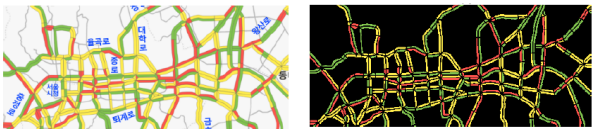


Fig. 2.1. Example of Traffic Image Masking Process

In the experiment of Z. Zhao et al. [9], an approach that can predict short-term traffic flow by using LSTM architecture has been presented. The model proposed in the study was applied to data collected from more than 500 observation stations by the Beijing Traffic Management Bureau. 25 million data from each observation station for 6 months with attributes such as vehicle density values, lane occupancy rates, and average speed have been used. It has been observed that when the traffic forecast was 15 minutes, the forecast results were close to the original data. The Mean Relative Error (MRE) in this study was found at 6.41.

In another research made by X. Ma et al. [10], minute Spatio-temporal traffic dynamics were transformed into time-space images [11] for each day as in the image below.

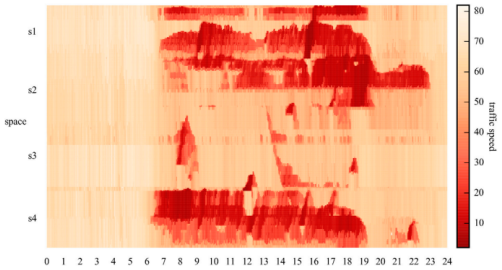


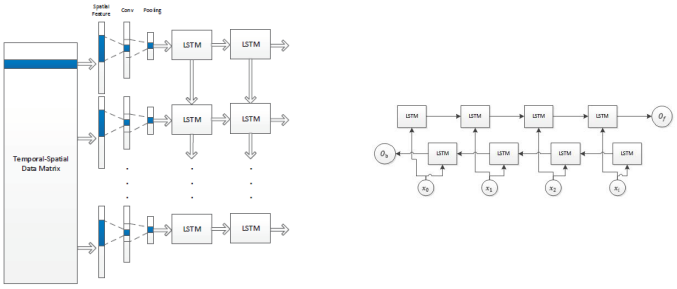
Fig. 2.2. Traffic speed time-space matrices

In total 37 days of data were used to predict the two different network roads' traffic speeds. The CNN-based model has been implemented with two famous architectures, AlexNet and LeNet. Convolutional filter size has been chosen (3,3) and max pooling has been defined (2,2). According to the depth of CNN, the model prediction mean squared error has been compared and the best results have been achieved with the deepest tried CNN model structure which was 256->128->64 and fully connected convolutional layers. In the study to test the model performance and do bench-marking, three other types of deep learning-based models and four statistical algorithms have been implemented, these were Artificial Neural Network (ANN), Recurrent Neural Network (RNN) and Long short term Memory Neural Network (LSTM NN) with three hidden layers with 1000 hidden units per layer for deep learning and random forest (RF) with 10 decision trees, OLS basic regression, K-nearest neighbors (KNN) with 10 nearest points, and Stacked auto-

encoder (SAE) with three hidden layers with 3000, 2500, 2000 hidden units [12]. Among all tried algorithms, CNN has achieved the least MSE value in the research.

In the study of predicting the traffic flow carried out by Y. Liu et al. [2], CNN and LSTM algorithms have been combined. In the proposed system, first, one dimensional convolutional and LSTM have been combined and Conv-LSTM module has been generated to extract the spatial-temporal features and for the second part, Bi-LSTM layer has been added to extract the period features of traffic data. Finally, the features extracted by CNN and LSTM are combined to obtain the traffic flow data prediction. Thanks to the Conv-LSTM module, the temporal and spatial characteristics of the traffic data could be fully fused. By using convolutional and LSTM, data pre-processing and feature extraction steps could be done automatically.

On the figure below, the Conv-LSTM structure of the proposed system could be seen.



(a) The Conv-LSTM structure

(b) The Bi-LSTM structure

Fig. 2.3. The model structure of the CNN-LSTM proposed system

Each column vector in the input matrix of Conv-LSTM represents the traffic data of each adjacent region at time t, whereas each row represents the same time for each region. As in the study, the traffic flow between spots of each region was considered, then converting the traffic data into a one-dimensional matrix form helped to get the spatial correlation of traffic flow between spots in the region. After the convolutional process, the output of spatial correlations was sent to the LSTM as seen in the first figure above. After LSTM layers, the output was sent to the Bi-LSTM. The structure of the Bi-LSTM has been shown in the second above visual. The meaning of x_i is the input of LSTM, O_f denotes the output of forwarding LSTM and O_b demonstrates the output of back LSTM. As the traffic data had periodic aspects, then some similarities have been found for the same time of the day. In the study, traffic flow information at the same time on the last day and at the same time of the last week has been considered in order to extract the periodic feature of the traffic. In this LSTM, the output value of the traffic data was only related to the information of the previous time. In the model performance evaluation part, the study has shown that the combination of the Conv-LSTM module performed better than the CNN-LSTM structure. Moreover, the Conv-LSTM module has been trained with and without the Bi-LSTM module, and the result has been found that the model with Bi-LSTM performed the best among all tried models.

Moreover, in the study of X. Xu et al. [4], by using the taxi information which contains the taxi pick-up and drop-off times and locations, the traffic volume has been predicted with the model structure of CNN-LSTM. In the study, the traffic data has been visualized with a matrix thermodynamic diagram as the input of the two-dimensional convolutional layers. In the CNN module, the spatial matrix information has been extracted, and then in the LSTM part, the sequence prediction has been made by using certain time steps and assigning weights to different time steps. As the time step, 5 time-steps have been used to predict the next time step. In the paper, for the training period, they considered only the traffic volume of the weekdays, from Monday to Friday, of 4 weeks which is account for 5 days. For the prediction period, they predicted on the 5^{th} week of November, Monday to Wednesday traffic volume data. The thermodynamic diagrams used to train the model could be seen in the below visual.

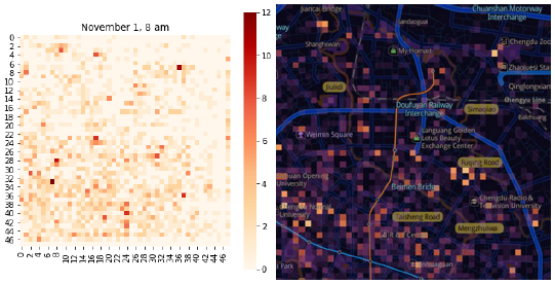
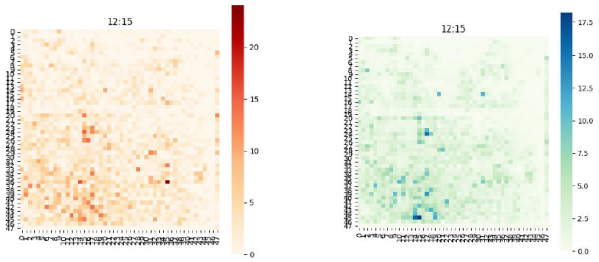


Fig. 2.4. The heat-map of Traffic Volume in grids

In the study, they considered one examined area on the map and they divided the area into 48*48 grids, each grid refers to the latitude and longitude coordinates within a certain range. After training the CNN-LSTM model with these generated images, they obtained heat-map prediction images. One example could be seen in the below figure.



(a) Actual Traffic Volume at 12:15

(b) Predicted Traffic Volume at 12:15

Fig. 2.5. CNN-LSTM Model Input and Output Heat-maps in grids

For the evaluation of actual and predicted results, the entropy difference between the predicted image has been calculated and the difference has been found less, this result has been indicated that the model was relatively robust.

3. DATA ACQUISITION AND DESCRIPTION

In this study, the traffic sensors data which were provided by the government of Madrid city has been used. The monthly data sets could be reached on the Open data portal of the Madrid City Council. [13]

On the open data portal of the Madrid City Council, two types of data have been used, the formerly used data source has the monthly sensor data of the traffic every fifteen minutes and the latter used data source has the information of the sensors, their placed districts and the geographic information of the sensors.

3.1. The Traffic Sensor Data Source

On the website of the open data portal of Madrid city council, there are monthly traffic data sets from January 2013 to June 2022. In these data sets, there are several variables measured in 15 minutes intervals. Each measured variable meaning could be seen in the below visual.

Name	Type	Description
idelem	Integer	Unique identification of the Measuring Point in the control systems of the traffic of the Madrid City Council.
date	Date	Official Madrid date and time in yyyy-mm-dd hh:mi:ss format
id	Text	Measurement Point Identifier in Traffic Systems (provided by backward compatibility).
elem_type	Text	Name of Type of Measuring Point: Urban or M30.
Intensity	Integer	Measuring Point Intensity in the 15-minute period (vehicles/hour). A negative value implies no data.
occupation	Integer	Measuring Point Occupancy Time in the 15 minute period (%). A negative value implies no data.
charge	Integer	Vehicle load in the period of 15 minutes. Parameter that has in The value of a negative value implies the absence of data.
vmed	Integer	Average speed of vehicles in the period of 15 minutes (km/h). Only for M30 interurban measurement points. A negative value implies the absence of data.
error	Text	Indication of whether there has been at least one incorrect or substituted sample in the 15-minute period. N: no errors or substitutions E: the quality parameters of some of the integrated samples are not optimal. S: some of the samples received were totally erroneous and have not been integrated.
integration_period	Integer	Number of samples received and considered for the period of integration.

Fig. 3.1. The Description of Traffic Sensor Data sets

3.2. Location of traffic measurement points

The data set contains the location information of each traffic measurement zone (each sensor ID in the first data set) in Madrid city for January 2022. [14] The data has a common ID and measuring point(elem_type) column with the other data source. In below table the first rows of the data source could be seen.

	Elem_type	District	ID	Cod_cent	Name	utm_x	utm_y	longtitude	latitude
0	URB	4.0	3840	1001	Jose Ortega y Gasset E-O - Pº Castellana-Serrano	441615.3433	4475767.942	-3.688323275	40.43050187
1	URB	4.0	3841	1002	Jose Ortega y Gasset O-E - Serrano-Pº Castellana	441705.8823	4475769.687	-3.687256103	40.43052394
2	URB	1.0	3842	1003	Pº Recoletos N-S - Almirante-Prim	441319.3713	4474841.154	-3.691726845	40.42213209
3	URB	4.0	3843	1004	Pº Recoletos S-N - Pl. Cibeles- Recoletos	441301.633	4474763.728	-3.691928782	40.42143334
4	URB	4.0	3844	1005	(AFOROS) Pº Castellana S-N - Eduardo Dato - Pl.Emilio Castelar	441605.7651	4476132.139	-3.68846965	40.43378206

TABLE 3.1. THE LOCATION OF TRAFFIC MEASUREMENT DATA SOURCE

This data contains 4,576 rows and in the data set, there is the name of each id and their district zone in which area they belong. Moreover, in the data, there are utm_x and utm_y columns which are the x and y coordinates of the centroid of the polygon representation of the measurement points. The last two columns which are longitude and latitude represent the points place geographically.

3.3. ETL (Extract, Transform and Load) with Web Scraping

Web scraping, also known as Data Scraping, is the process of extracting data from Web sites. [15] As all the data sources are on the open data portal of Madrid city, ETL with Web scraping has been implemented. Therefore, each *zip* file *href URL* has been accessed first, then *csv* files inside each zip file have been extracted. As the data file sizes were very high which is around 11 million, then each *csv* file has been read by using the parallel computing method. To do that pandas *read_csv* function has been used with 50000 chunk size and the data has been loaded onto Google Drive storage. The *href URLs* have been accessed by using the REST API GET protocol. The *BeautifulSoup* package has been imported to get all the files ending with *.zip*, and all the *href URLs* could have been written onto a text file later to be used.

To gather the monthly data sets, the first monthly data sources with minute frequency have been combined. Due to huge file sizes, to understand the data properties, the last 6 months' data sets that belong to the 2022 year period have been aggregated. The size of each file could be seen in the below table.

File Name	Number of Rows	Execution Time
06-2022.csv	11,399,709	69.2815
05-2022.csv	11,732,565	56.1781
04-2022.csv	11,323,906	55.4675
03-2022.csv	11,796,900	56.1046
02-2022.csv	11,399,709	50.5870
01-2022.csv	12,429,813	57.9841

TABLE 3.2. DATA SIZE AND EXECUTION TIME OF EACH COMBINED FILE

As a result, all available data sets that have been recorded in 2022 could have been successfully combined and the combined data had 69,477,674 rows and a 2.89 GB file

size. Later obtained data set has been analyzed by merging with the location of the sensors data set in Apache Spark.[16] Before starting to do data analysis, as the data size was really big, to analyze the data more efficiently, the Spark session has been initialized with 40GB of driver and executor memory, and 4 driver and executor cores. After initializing the Spark, the *shema* which identifies the properties of data such as the data type of each column and null-able information has been defined due to high execution time while reading the CSV file, because when the schema is defined, Spark does not spend time to infer a most proper schema for the data set. After defining the schema of the data, the big data set could have been read in 0.13 seconds in Spark.

4. EXPLORATORY DATA ANALYSIS

To understand the data sets more and get important insights, Exploratory Data Analysis (EDA) approach has been used to analyze two-type of datasets and to summarize their key features by using visual methods.

4.1. Analysis of Missing Values

First, in both data sets, missing values have been examined. In the below visual, the number of missing values in each column of both data sets could be seen. In the data which has six-month fifteen-minute intervals sensor data, missing values have been observed in occupation, vmed, and error columns, while on the second location data set, missing values have been found in latitude, longitude, utm_x, utm_y, name, cod_cent and district.

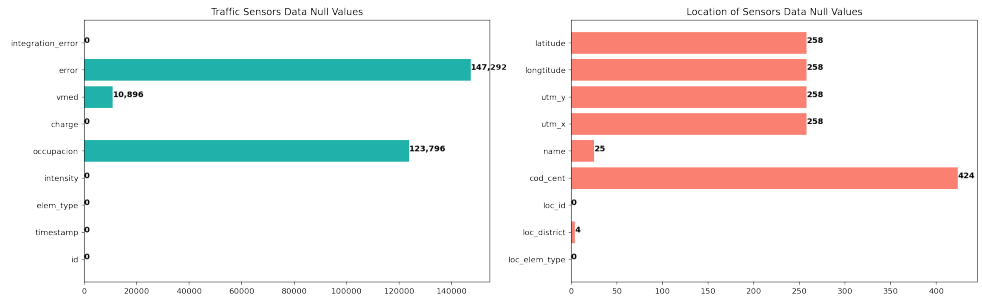


Fig. 4.1. The number of Missing Values

As seen in the first horizontal bar graph, the error variable contains the most null values, while the occupation and vmed variables also contain null values, although not that much. Here, the most important point is that id, timestamp, intensity, and elem_type variables do not have any missing values. As these variables will be used in the project, it is very important for this study that these variables do not contain null values, because firstly, the project aims to estimate the traffic density according to the district information, and secondly, the traffic sensor data will be combined with the location data while combining them over id. Also, elem_type is an indicator of the measuring point which is a good identifier. Moreover, when looking at the missing value bar graph of the data set containing location information, missing data is seen for many variables. In this study, since traffic density map images will be created for each district area, then latitude, longitude, and district type information are quite important. Since the latitude and longitude data of each sensor will be averaged on a district basis while creating the traffic density images, the missing data in these variables will be examined in detail.

4.1.1. Missing Values in the Monthly Traffic Sensors Data

When the null values in the vmed variable in the traffic sensor data set were analyzed in detail, it was found that this information was missing in 170 sensors' data located at the M30 point. At the same time, in the absence of vmed information, the traffic intensity variable contains 10,676 zero values. As can be seen in the first bar graph above, there are 10,896 missing data. In cases where vmed information is lost, it can be said that the traffic intensity variable is generally zero. The average traffic density of 170 sensors in the absence of vmed information was found to be 0.49. It is quite normal for some sensors to have an intensity of zero for some times. When the traffic intensity is measured as zero, this case is mostly seen during the night times. To conclude, it would not be the right decision to drop rows that have vmed variable having null values as intensity measure is the most important one and does not have any null values.

Moreover, vmed variable has a lot of zero values in both M30 and URB types sensors,65,717,283 values in URB, and 201,104 in M30 type of sensors. Also, as in this project, the district-based traffic intensity will be studied, the results of average intensity and car speeds can be seen in the below table.

District	Average Traffic Intensity	Average Car Speed
1	406.515	3.023
2	483.795	10.252
3	483.889	5.636
4	471.665	1.656
5	365.661	1.426
6	325.224	**0.0**
7	501.925	**0.0**
8	219.103	1.379
9	384.341	6.239
10	305.308	2.731
11	271.245	3.098
12	346.833	2.892
13	321.902	4.171
14	348.383	9.085
15	289.544	1.926
16	233.273	0.639
17	229.907	**0.0**
18	168.461	**0.0**
19	226.683	**0.0**
20	243.616	**0.0**
21	215.754	**0.0**

TABLE 4.1. THE AVERAGE INTENSITY AND CAR SPEED OF EACH DISTRICT

As seen from the results the car speed (vmed) measure is missing or is zero for many of the sensors and in districts 6, 7, 17, 18, 19, 20, and 21, the average value is zero. Therefore, even though this variable could give some important insights related to traffic intensity, vmed variable will not be used in this study.

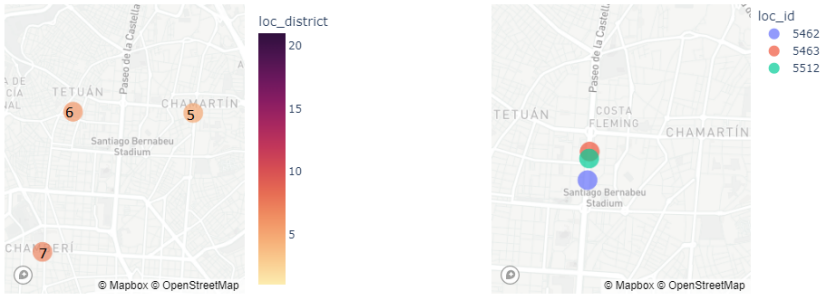
4.1.2. Missing Values in the Location of Traffic Sensors Data

As a result of the detailed analysis, it was seen that the sensors in the case where the latitude and longitude variables are null belong to the URB measuring point only and these missing values consist of 258 sensors' data. Therefore, for 258 sensors that are in URB, the location information is missing to generate the traffic intensity images. On the other hand, in the data, there is no missing data regarding the location of sensors of M30. As it is important to have the longitude and latitude information of the sensors in order to generate the traffic intensity images, these sensors have been checked in the traffic sensors data whether they have information or not. Indeed, only 2 sensors out of 258 sensors are not included in the traffic sensor data set, which means that 256 sensors have information. After calculating the average traffic intensity of each sensor in the data, high-intensity values have been found. Therefore, it would be a mistake to eliminate those rows from the data. Even though these sensors' location information is missing, other sensors' location information in the same district could be used to calculate the average latitude and longitude of each district area. Therefore, in order to not miss these sensors' intensity measures, while plotting the traffic intensity images, each district should be placed according to the average latitude and longitude available and then the intensity of each district should be averaged. However, only for ID 6637, both district and location variables' values are missing. Therefore, unfortunately, this sensor could not be included and will be eliminated. To conclude, missing values of latitude and longitude of 257 sensors should remain and 6637 sensor data should be eliminated in the location of sensors data before merging with the traffic sensors data set.

Moreover, the district information is missing in 4 rows and these rows have 6637, 5462, 5463, and 5512 sensors in the URB measuring point. Additionally, as mentioned before, for the 6637 sensor, latitude and longitude values are missing too. When these sensors' intensity values have been checked from the traffic sensors data set, they could have been found at 180.59, 1028.88, 1119.32, and 1015.61, respectively, and their average vmed was found zero. Therefore, for these sensors, there is an intensity value, however, there is no correct vmed value. As for sensor 6637, the latitude and longitude information are missing, this row should be dropped, however, for the other three sensors, these sensors could be tried to place in the nearest district area after finding each district zone's average latitude and longitude values and plotting them on the map.

4.1.3. Finding the District of Sensors without District Information

As a result of the analysis in the previous section, it was seen that for 3 sensors, there was location information, but there was not any district information. In this section, using the location information of these sensors, sensors have been displayed on the map and their distances to the nearest possible region have been calculated.



(a) The closest possible District Locations

(b) Sensors that District Information Missing

Fig. 4.2. Assigning sensors that do not have district information to the closest district

For this, the square of the difference of latitudes and the square of the longitudes were added and the square root of the result was taken.

Sensor IDs	Dist-5	Dist-6	Dist-7	Closest District Found
ID-5462	0.0134	0.0093	0.0215	District-6
ID-5463	0.0128	0.0089	0.0237	District-6
ID-5512	0.0129	0.0090	0.0232	District-6

TABLE 4.2. THE DISTANCE OF EACH EXAMINED SENSOR TO THE NEAREST DISTRICT ZONES

According to the results, all 3 sensors were closer to district zone 6, therefore, the null values of district measure of these 3 sensors were numbered as 6.

4.2. Analysis of Sensors by Measuring Points and Districts

As mentioned before in the part of Data Acquisition and Description, the traffic sensor data sets have two measuring points, URBAN (URB) and M30 roads. In the below pie chart, it could be seen the percentage of the average intensity in 2022 from January to June for the monthly traffic sensors data.

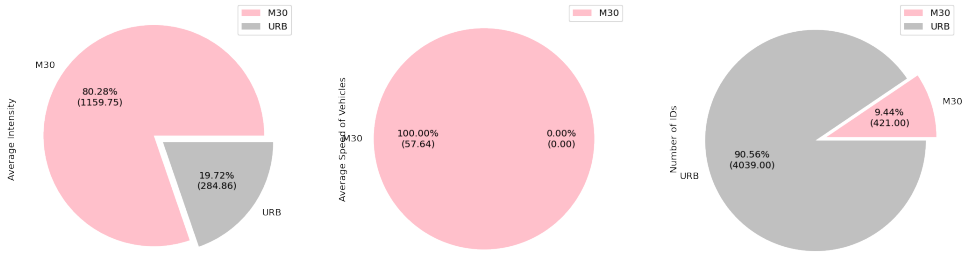


Fig. 4.3. The Percentage of Average Intensity of each Measuring Point

The average intensity has been calculated at 284.86 for URB and 1159.75 for M30 measuring points. The M30 measuring point is quite busier than URB. Moreover, the average speed of the vehicles is also calculated but as seen in the visual that the value is zero, therefore, for URB, there is no speed of vehicles (vmed) information in the traffic sensors data set.

Moreover, on the below donut chart, the percentage of the number of distinct sensor IDs with their unique districts could be seen. In total, there are 4152 and 424 sensors and 21 and 14 district areas for URB and M30, respectively. The district zones included by the M30 and URB points are shown as long text in the image below.

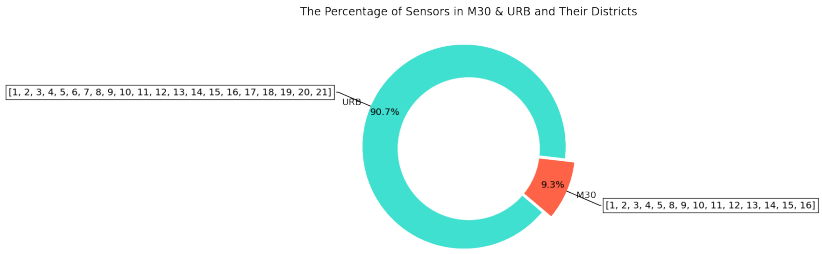


Fig. 4.4. The Percentage of Sensors placed in M30 & URB and Districts

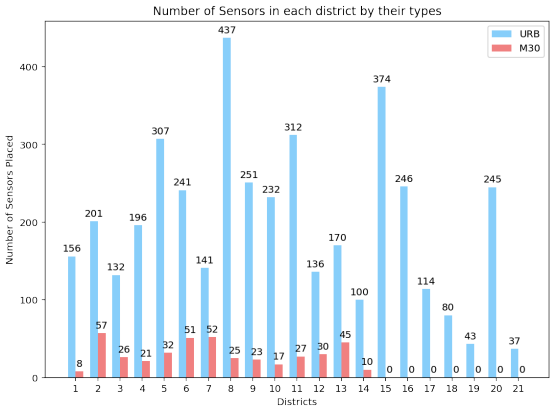


Fig. 4.5. The Number of Sensors in each District in the Location Data set

In addition, on the left bar graph, the number of sensors by the type for each district in the location data could be seen. In total there are 4576 sensors available. As seen in the image, districts 8, 15, and 12 contain the most sensors. While there are many URB-type sensors, the sensors located along the M30 path are fewer. District 2 contains the most M30-type sensors.

The image below shows the time series from January to June during the year 2022, obtained from the traffic density information of the sensors located on the M30 and URB roads of each region. The red color corresponds to M30, whereas the blue represents the URB data.

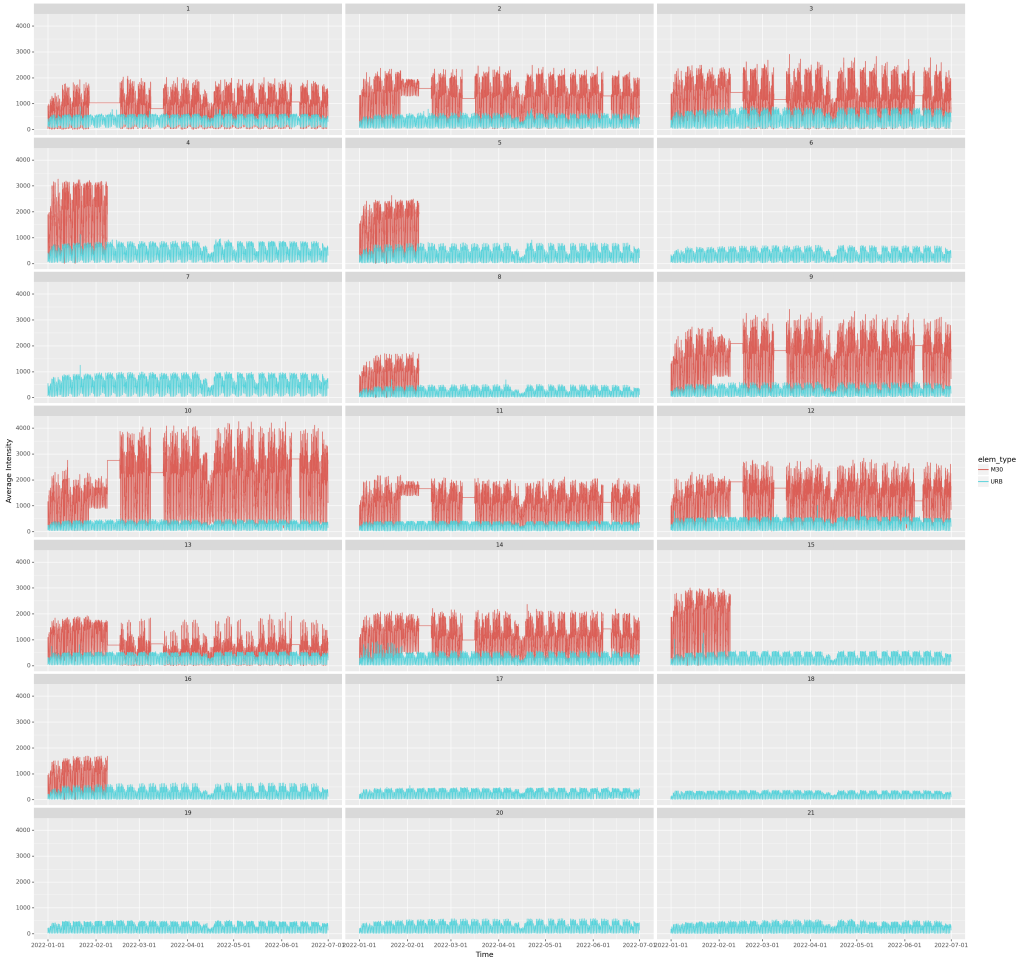
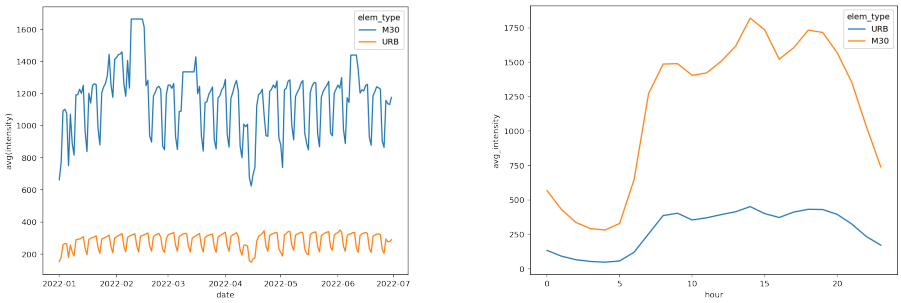


Fig. 4.6. The Percentage of Sensors placed in M30 & URB and Districts

When the traffic intensity flow of each type of sensor in the image is examined for each district, it is seen that the time series of the sensors located on the M30 road is not continuous as URB sensors. For instance, for district numbers 1,2,3,9,10,11,12,13, and 14, the data of M30 type sensors do not change in some periods. The flows do not follow the pattern of the time series. Moreover, in some districts such as 4, 5, 8, 15, and 16, the traffic intensity values have been observed only at the beginning of the examined period. The data for the last months are missing. On the other hand, the traffic intensity data obtained from URB has a continuous flow for each district, and it is proper to analyze. As seen in the visual and detailed analysis, traffic intensity of type M30 is not suitable for forecasting analysis as it has either the same value or there is no data for some periods. For this reason, traffic density information of URB sensors on a regional basis will be used in the deep learning model.

In addition, hourly and daily average density analysis were performed on the basis of

the type of sensors and the following images were obtained.



(a) Daily Average Traffic Intensity (b) Hourly Average Traffic Intensity

Fig. 4.7. Intensity Flow of M30 and URB sensors by day and hour

As seen in the image, according to the daily traffic intensity information obtained from the sensors in all regions, in line with the information recorded by the M30 sensors, an increasing trend was observed from January to mid-February, while an increase was experienced in the first weeks of March after mid-February. Afterward, a constant flow appears in general. However, while there was an increase in traffic density in March and a decrease in mid-April, there was an increase in early June. However, it should be noted that, as mentioned in the previous section, there is inconsistency in M30 data for some periods. The sharp increases and decreases appearing on the daily chart may be due to this reason. In addition, the daily traffic intensity time series of the URB sensors showed a smooth flow in general. However, there was a decrease in mid-April. On the other hand, according to the hourly traffic density information in the second graph, the hourly graph of the M30 and URB sensors shows a similar pattern. In general, while the traffic is the least at night, the traffic density has increased in the morning and afternoon hours. Again, the density decreases after 8 pm.

4.3. Daily, Hourly and Minutely Traffic Density of Sensors

Since the traffic sensor data includes date and time, and it is recorded at intervals of every 15 minutes, the traffic sensor data shows a time series attribute. In this section, the time series characteristics will be examined in detail such as its trend, seasonality, and residuals. Moreover, traffic intensity information will be re-sampled over an hour, per day, and on weekdays. To do that, the traffic intensity of sensors placed in district three has been used to get some meaningful insights from the data.

In the below visual, it could be seen the traffic intensity of chosen district at 15-minute intervals.

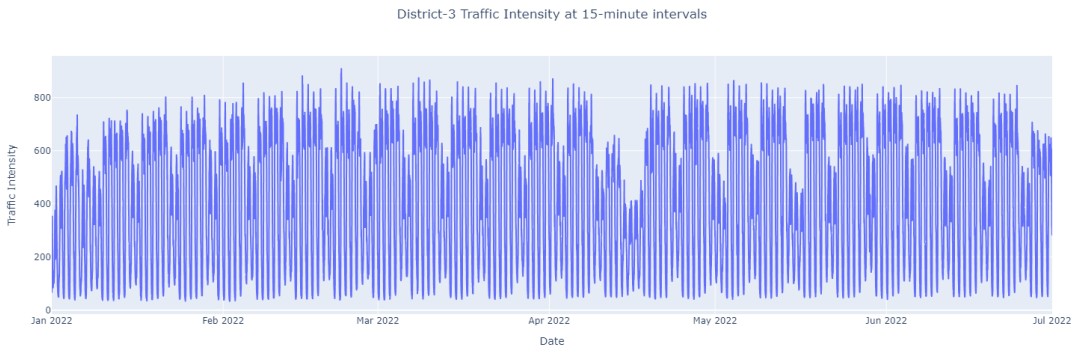


Fig. 4.8. Minutely Traffic Intensity from January 2022 to June 2022

In the image below, traffic intensity information for different periods is shown. When the day and week charts on the left are examined, the day chart showed an increase from January to April, and then a sharp decrease was experienced in mid-April. According to the weekly graph, it could be seen that this decrease occurred in the third week of April. Moreover, in the weekly graph, there was an increase from the first week to the last week throughout January. Although the weeks of February and March generally follow a constant flow, there is an increase with fluctuations from the beginning of February to the beginning of April. During the 3rd week of April, even though the traffic density decreased sharply, afterward, the intensity increased and showed a generally stable flow until the end of June.

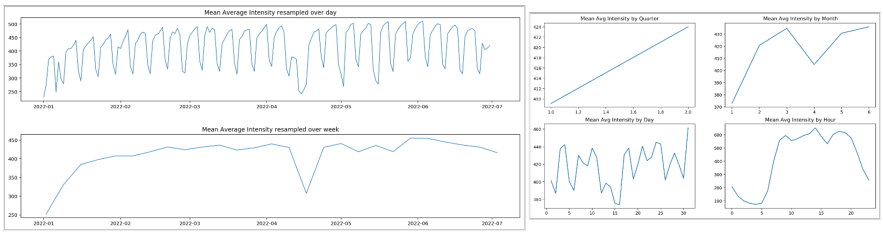


Fig. 4.9. The pattern of the Traffic Intensity

When the above visual that is consisting of four sections on the right is examined, quarterly, monthly, daily, and hourly patterns could be seen. Since the period from January to the end of June is analyzed in the data, there is data for 2 quarters. January, February, and March symbolize the first quarter, and April, May, and June symbolize the second quarter. As can be seen from the graph, traffic density increased from the first quarter to the second quarter. Looking at the average traffic density on a monthly basis, it is seen that the months with the highest traffic are March and June, the month with the lowest traffic is January with an average of 372 density, and the other month with the lower density is April with an average of 404. March and June have similar traffic densities. Looking at the chart on a day-by-day basis, the average information of traffic density by day is shown. In the hourly chart, it can be interpreted that the traffic intensity is at its lowest level during the night hours, and on the contrary, the traffic is very active from 5 in

the morning until 8 in the evening.

The traffic density in the third region consists of regular time series patterns that follow each other. These patterns are boxed in the image below. For the sake of detailed data in the image, only the 15-minute traffic density history of the days of February is given. Looking at this data, traffic density information generally shows a very similar flow from Monday to Sunday.

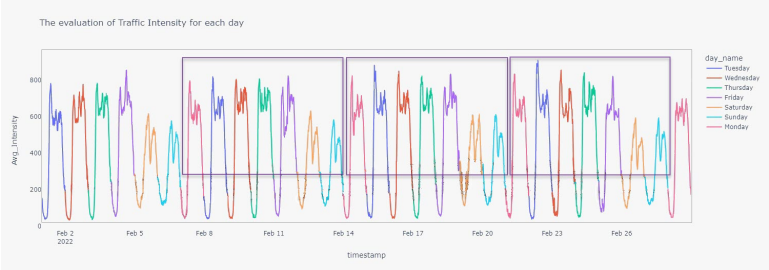
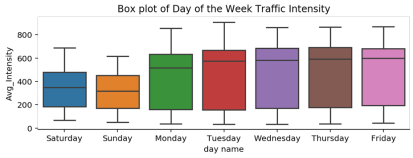
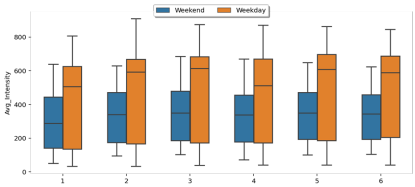


Fig. 4.10. Hourly Traffic Intensity from January 2022 to June 2022

In the box-plots below, more detailed analyzes are positioned according to the days of the week and weekends. The traffic intensity information obtained according to the days of the week is shown on the left side, whereas traffic data every month for weekdays and weekends is demonstrated on the right image below.



(a) Traffic of the Day of the week



(b) Weekday and Weekend of the Traffic of each Month

Fig. 4.11. Traffic Intensity by the Day of the Week

On weekdays, as seen in the first box-plot, the medians of weekdays are much larger than the medians of weekend days, and the intensities are generally similar on weekdays. Looking at the density box plot of weekdays and weekends every month in the second box chart, similar traffic density medians were calculated for working days. While the median value is lower in January and April, it is higher for other months. Considering the intensity of the weekend of each month, the month with the highest density in the 3rd district was observed as March.

5. EXPERIMENTS

In this section, the algorithms used in the models that predict the traffic intensity recorded at 15-minute intervals on a district basis will be explained within the scope of the used deep learning methods, model architecture, and model estimation results. In the determination of algorithms, studies made in this field in the literature were considered. The algorithms used in this thesis are CNN and RNN-based neural networks, respectively. RNNs are widely used and they are superior to traditional NNs in time series problems. To increase the success of RNNs in estimation problems, LSTM networks have been developed by adding a memory unit. In this study, LSTM networks have been used too. In the following parts, first, all the used deep learning methods will be explained in detail and later the models implemented for the estimation of traffic intensity in a district base will be explained step by step.

5.1. Deep Learning Methods Used

5.1.1. Convolutional Neural Network

Convolutional neural networks (CNN or ConvNet) are in the group of multi-layered, feed-forward artificial neural networks that have been proven to be very effective in areas such as image recognition and computer vision. [17] There are various studies that also use the time series data and obtain successful results inspired by the success of the Convolutional Neural Network in image recognition. [18] CNNs have the ability to identify faces, individuals, street signs, traffic flows, and many other aspects of visual and also numeric data.

Architecturally, CNNs are inspired by multi-layer perceptrons. CNN takes advantage of local spatial correlation by applying local connectivity constraints between neurons of adjacent network layers [19]. The main focus of CNN is the processing of data through the convolution operation. By converting any signal to another signal, a third signal is produced that reveals more about the signal than the original signal itself. In a mathematical structure, CNN typically consists of three types of layers, these layers are convolution, pooling, and fully connected layers. [20] The convolution and pooling layers perform the feature extraction, while the fully connected layers send the extracted features to the final output for prediction. Convolutions are necessary for a neural network to interpret pixels in an image or time series as numerical values. [20] When working with images, the function of convolution layers is to convert the image into numerical values that the neural network can interpret and then extract the relevant patterns. In the convolution layer, filters are passed over the input image to get the attributes of the image. In addition, more than one convolutional layer could be used in a CNN architecture for obtaining

multiple features. Moreover, each convolutional layer has an activation function. The most commonly used functions are ReLu, tanh, and sigmoid. In this study in order to produce positive output values, sigmoid and ReLu functions have been used. More detailed information will be given in the model development part.

In the literature, there are many popular CNN architectures such as LeNet, AlexNet, GoogleNet, and VGGNet. [20] In the section of Literature Review, some studies about forecasting traffic by using AlexNet and LeNet structures have been mentioned. In the thesis, VGGNet pre-trained model with 16 convolutional layers has been implemented to extract the features of traffic intensity map images in the sub-section of Traffic Intensity Prediction with Traffic Intensity Map Images.

5.1.2. Long Short Term Memory

Long Short-Term Memory networks (LSTM) are a special type of Recurrent Neural network (RNN). [21] Unlike traditional neural networks, RNNs have a mechanism that remembers past information. In this type of neural network, information is transferred in different steps and due to this reason, they are used to process data over time such as traffic intensity at 15-minute intervals. However, RNN has one disadvantage that is since RNN has a short-term memory, it has been found in the literature that it does not work well in problems where some long historical information is given as input data. [22] Due to that reason, LSTM is designed to memorize the information better and eliminate the short-term memory problem of basic RNN.

The LSTM neural network has a complex structure. [23] Unlike the RNN architecture, it has four interactive layers in a chain with a unique method of communication which are between the input gate, forget gate, output gate, and cell state. The three gates control the flow of information through the cell and neural network. The Forget Gate is the part that decides which information should be forgotten or retained from the previous steps according to the activation function throughout the chain. Then the input gate looks for the current step and decides which information is important to add to the cell state. Finally, the output gate decides the next cell's entry and is also used for prediction. [24]

Additionally, another powerful LSTM type which is Bidirectional LSTM, also called Bi-LSTM has been used in the thesis model implementation. Bi-LSTM is an enhanced version of traditional unidirectional LSTM networks. [25]. Bi-LSTM networks are also like LSTM networks, but unlike them, there are two different LSTM networks to be trained in the structure. The series given as input in the first LSTM is used as they are, then, the second LSTM network is trained using the inverted versions of these series. [26] In the literature, there are various studies have been made to forecast traffic congestion and speed prediction by using both Uni-LSTM and Bi-LSTM and according to the studies, compared to Uni-LSTM, they used Bi-LSTM to improve the model performance. [27] [28]

5.2. Development of the Deep Learning Models

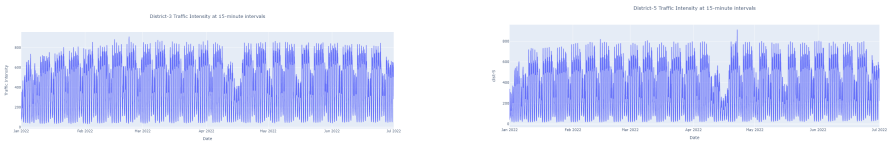
In the thesis, first, one-dimensional deep learning models were created using numerical traffic intensity information. Afterward, since there was latitude and longitude location information in the data, images were generated on the map of each region at 15-minute time intervals and the deep learning models used in the first part were converted into a two-dimensional structure. This time, traffic density information in each zone was tried to learn with images produced.

5.2.1. Traffic Intensity Prediction with Numerical Traffic Intensity Series

In this section, firstly, two different models in a one-dimensional structure were tested by using the average traffic intensity information obtained for each region at 15-minute intervals. While training the model, 24-time steps were used, and then it was tried to estimate after 4 steps. Since a time step in the data consists of 15 minutes, 24-time steps correspond to 6 hours, while 4 steps correspond to 1 hour of data.

Pre-Modelling - Pivoting the Raw Data

In the pre-modeling step, the pre-processed data set which had the average traffic intensity values for each district by observation time has been pivoted. The reason for making the raw data into a pivot table is that in the data there was the time distributed traffic intensity information of 21 districts. Therefore, the raw data has been pivoted so that the columns were districts and the rows were the observed time. Therefore, as the time observed was from January to June 2022 at 15-minute intervals, therefore, a data matrix of 17,376 rows and 21 columns could have been obtained. For the beginning, only two districts' data have been used, districts 3 and 5. The chosen districts' traffic intensity flows could be seen in the below visual.



(a) District-3 (b) District-5

Fig. 5.1. Chosen Districts' Traffic Intensity Levels

It can be seen that the traffic behavior collected from URB sensors in these two districts has a very similar series. After building the models and increasing their performance, all traffic zones were used as the final step after the models were implemented and tested in these two districts in the section of Model Traffic Intensity Predictions of All Districts in Madrid.

Before moving further to the pre-modeling steps, each district series stationary aspect has been tested by using the Augmented Dicky-Fuller Test in Python. This test is used for checking the time-distributed data whether it has the stationary attribute or not. [29] Basically, a time series that is stationary keeps very similar mean, and variance values through examined time period and it does not include seasonality and trend. Stationarity is quite an important point to consider for time series forecasting. For instance, for statistical models such as ARIMA, the model performance could be affected very much if the data is not stationary. However, in the thesis, LSTM-type neural networks were used, and LSTMs could learn even with the non-stationary series (nonlinear). Even though LSTMs are not quite affected by the stationarity, the chosen districts' traffic series have been checked and according to the test results, the p-values have been found less than 0.5. It indicated that they do not have the non-stationary attribute. Even though the series was not stationary, as mentioned before, this situation would not affect the very much LSTM network, however, if the series is stationary, this might increase the model performance. [30]

Train-Test Split

After pivoting the raw data and analyzing some important aspects, the data matrix is divided into train and test sections for the modeling process, and the information of the new samples could be seen in the table below.

Samples	Time Duration	Days	Times per 15-Minute
Train	2022-01-01 00:00:00 to 2022-05-31 23:45:00	151	14,496
Test	2022-06-01 00:00:00 to 2022-06-30 23:45:00	30	2,880

TABLE 5.1. TIME DURATION AND NUMBER OF OBSERVATIONS OF TRAIN-TEST SETS

The time series of traffic sensors have been divided since the first recorded traffic information on June 6^{th} which was at 00:00:00. In that way, for train and test splits, 151 and 30 days' minutely data have been used. In other words, with the first 5 months' data, the model has been trained and the June data has been tried to forecast and evaluate.

Standardization

Furthermore, the train and test samples have been normalized within the range of zero and one. In optimization, feature standardization is a common procedure that has been shown to reduce convergence rates. [31] Therefore, for the deep learning methods used in the thesis, standardization is an essential step. As there were traffic intensity values with widely different scales, therefore, without standardization, the neural network could not weigh the features equally. This could cause a false prioritization of some features over others in the learning process. [32]

After having the train and test sets scaled, the numerical time series data for each district have been rearranged according to the multi-step forecasting structure. To do that, looking back and forecasting periods have been assigned in a function that produces consecutive sequence time series to train and predict. This is the structure of LSTM with time steps. As mentioned at the beginning of this section, the 24-time steps have been used for training and then the following 4-time steps have been predicted. The idea of how the written function works could be simplified by thinking of a 5-time step sequence data. For instance, for the first generated time series, if the look back and forecasting parameters are defined as 3 and 1 in the function, then the first generated samples will have the first 3 series data for training and the 4^{th} data for testing. Then the second generated sample will have 2,3 and 4^{th} rows in the data and for prediction, will have the 5^{th} row. According to this simplified idea, 14469 samples with 24-time steps and 2 features that presents the two districts have been produced for training, and 2853 samples with 4-time steps and 2 features have been obtained for the testing set.

Modeling

In this part, the model structures will be examined. In the thesis, sequence-to-sequence (seq2seq) models have been used with LSTM and CNN encoders and LSTM decoders.

1. Sequence-to-Sequence Model with One LSTM Encoder-Decoder Layers

The model structure in which the encoder and decoder layers are LSTMs could be seen in the image below. In the model, the input layer starts with an LSTM encoder. As in the example architecture, more than one LSTM layer could be used as an encoder and these layers could be assigned with different sizes. In that way, the model may be able to understand better the complexity of the time series by using the stacking LSTMs, in other words, by using more hidden layers in the neural network architecture. [33]

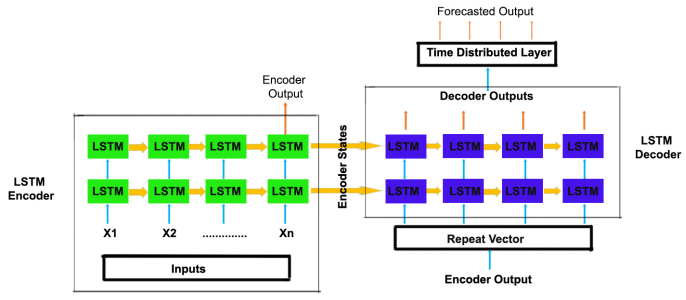


Fig. 5.2. General Structure of the Sequence-to-Sequence Model with One LSTM Encoder-Decoder Layers

In the encoding part of the model, basically, the given input is interpreted and the feature vector is extracted. Then, the output feature vector of the encoder is given to the

decoder LSTM part to be performed forecasting. As seen in the visual, there is also a part called Repeat Vector. This repeat vector layer should be placed between the encoder and decoder layers as a bridge between encoder and decoder parts because there are multiple created time series samples in sequence. Therefore, thanks to the repeat vector layer, the encoding processes could be repeated as many times as the number of future forecasting steps and the feature vector could be replicated in that way. [34] Moreover, after the Decoder part, the time distributed layer is added in order to get the final output, and the output is duplicated according to the number of features.

While creating the first model, inspiration was taken from the model structure briefly mentioned above. The stages of the first created one-to-one encoder-decoder LSTM model and the dimensions of the vectors in each stage are shown in the image below figure.

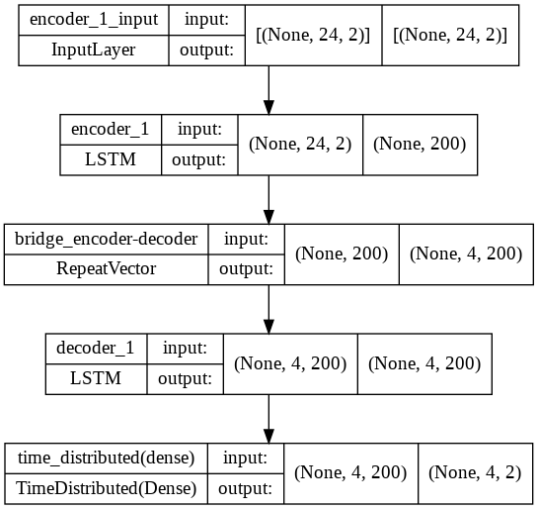


Fig. 5.3. Implemented Sequence-to-Sequence LSTM Encoder-Decoder Model

To summarize briefly, the input of the first LSTM encoder layer was given with the shape of (None, 24, 2), which means the input data had 24-time steps and 2 features. The features were defined 2 for the beginning of the modeling process as only 2 zones in Madrid city have been taken to forecast for testing. Therefore, the output of the model will be the predictions of two districts. In the model structure as seen in the figure steps, in the encoding part in which the feature vector has been created, one LSTM layer has been implemented with 200 units with ReLu activation function and in the decoding part, one LSTM layer has been defined with 200 units with ReLu activation function and the return sequences parameter of LSTM has been set as True. In that way, each time step emits the sequences of output. If it was defined as False, only the last cell in each time step could emit an output. Moreover, between encoder and decoder parts, a repeat vector layer has been defined four times repeating as the forecasting range was defined four while preparing the time series samples. To collect the predictions, a Dense layer with two features has been defined with the Time Distributed layer.

2. Sequence-to-Sequence Model with CNN encoder and LSTM decoder layers

As mentioned before, CNNs are commonly used for extracting features and they perform quite well in the literature. A detailed explanation of CNNs could be found in the *Convolutional Neural Network* part.

As first the numeric time series data has been used to forecast each district's traffic intensity, therefore, for feature extraction, the model structure has been started with two one-dimensional convolutional layers. Each layer's filter size has been set to 64. After two convolutional layers, the maximum pooling layer has been used with 2 pool sizes. This pooling layer is a layer that is often added between consecutive convolutional layers. The task of this layer is to reduce the displacement size of the representation and the number of parameters and calculations within the network. In this way, incompatibility in the network is checked. There are many pooling operations in CNN, [35] but the most popular one is max pooling. After the pooling step, Flatten layer has been used to prepare the data at the input of the time distributed fully connected layer. After flattening, the data in this neural network was the one-dimensional array of matrices from the Convolutional and Pooling layers. Then as there were multi-time series and forecasting steps that have been produced according to the defined time steps, therefore, a repeat vector has been used to encode the time series up to the number of the forecasting time steps. In the decoding part, one LSTM layer has been used with 200 units and the deep learning model has been finalized with the two-unit dense layer which two represent the number of districts to predict.

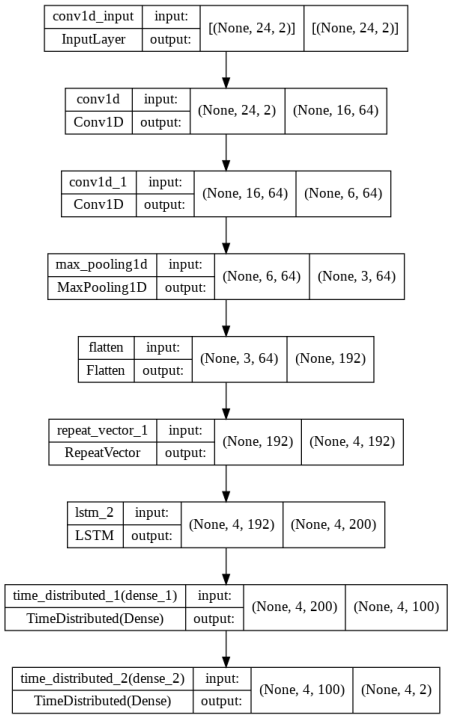


Fig. 5.4. Sequence-to-Sequence Model with CNN encoder and LSTM decoder layers

Defining Callback Functions

In deep learning models, the training is processed by using an epoch size. In many cases, training the models with higher epoch sizes is important for the model to learn more. However, while high epoch size is defined, there are various drawbacks. For instance, there is a high possibility of having an over-fitting situation as the model learns more even though the validation loss is not improving anymore. Another problem could be consuming hours to train the model and not being able to modify some parameters between each epoch. Therefore, it is quite crucial to automate some tasks after every training epoch which could help to control the training process. Due to that reason, before compiling the models in the thesis, some callback functions of Keras have been defined to regularize the model fitting. To eliminate the over-fitting situation, the early stopping callback API has been used. [36] With the early stopping function, the minimum validation loss has been observed with 10 patience. Basically, thanks to this callback function, the model training could be stopped if there are no improvements in the validation loss within 10 epochs. Another used callback function was Model Checkpoint [37] which is used for saving the model(weights) at certain frequencies. The last callback function has been implemented was Reduce LR On Plateau [38] which was used to control the learning rate. In the function, according to the defined metric to look, the learning rate is decided to decrease or remained at the minimum defined learning rate. As the minimum learning rate, 0.001 value has been assigned with patience parameter 3.

Implementing a Base Model to Evaluate the Model Performance

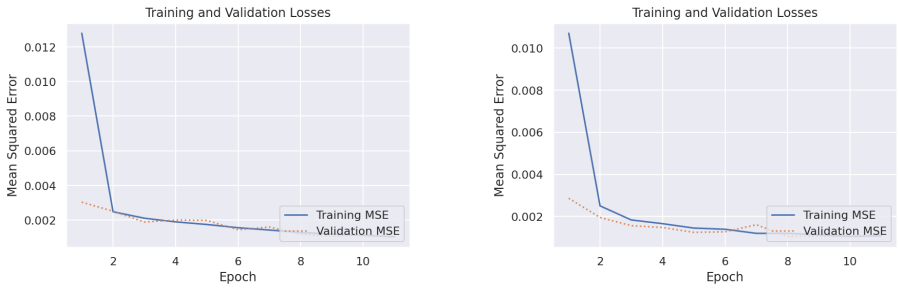
In the application part, a baseline model has been implemented with the same day-time split of train and test sets to compare the performance of the model. (The section of *Train-Test Split* could be read to see the split process in more detail.) To create the predictions and calculate the evaluation metrics, the average of the previous time for each district has been assigned to the prediction of the next time. So with the experiment of the baseline model, it has been thought that the worst model could be the one that predicts the mean value of the previous time. Therefore, if the model performance of the built deep learning models is worse than the baseline model, it could be interpreted that the implemented model is performing even worse than the baseline that it has the mean value predictions, and it is not a good model for forecasting the traffic intensity values of the next times of each district. The evaluation metric results could be seen in the table below.

MAE	MSE	RMSE	MAPE
83.17	12143.41	110.19	26.17

TABLE 5.2. BASELINE MODEL EVALUATION METRIC RESULTS

Running the Deep Learning Models and Evaluating the Performance

The explained two models (one LSTM Encoder-Decoder and a second CNN-LSTM model) have been fitted with the defined callback functions, 50 epoch size, 32 batch size, and 0.1 validation set, and their training and validation losses have been updated for each epoch. The evaluation of the losses through each training could be seen in the below visuals.



(a) LSTM Encoder-Decoder Model (b) CNN-LSTM Model

Fig. 5.5. The Training and Validation Losses

In these model trials, both models were trained with 50 epochs and the training was stopped automatically at 11 epochs due to callback functions. While the training and validation losses in the last epoch were calculated as 0.0010333 and 0.001025 for the first model, the last recorded epoch loss values in the second model were observed as 0.001077 and 0.001126. At the same time, the training times of the models were calculated as 411.25 and 147.27 seconds, respectively. These results showed that the training time of the model was completed faster in the model where CNN was used as an encoder.

Moreover, the results of the evaluation metrics of each model prediction could be seen in the below table. The evaluation metrics have been calculated with the inverse transformation of the scaled traffic intensity values, in other words, the intensity values have been changed to their normal range.

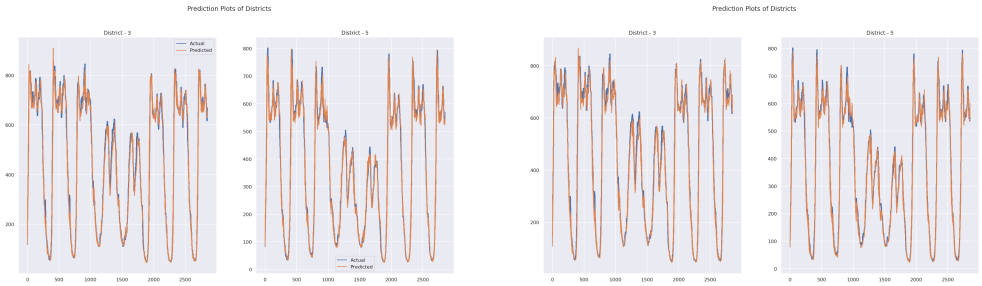
Models	MAE	MSE	RMSE	MAPE
LSTM Encoder-Decoder	19.07	708.83	26.62	6.65
CNN Encoder-LSTM Decoder	20.46	885.08	29.75	6.29

TABLE 5.3. MODEL EVALUATION METRIC RESULTS OF ONE-DIMENSIONAL ENCODER-DECODER MODELS

According to the obtained results, both models performed much better than the baseline model. While according to the MAE, MSE, and RMSE error values obtained from the forecasting of the test sample, the model with one encoder LSTM and one decoder LSTM model performed slightly better than the other built model, on the other hand, the MAPE result has been found slightly less for the CNN-LSTM model, but as such, both models showed very similar performance on the test data. However, it should be noted that

model performances may vary when both models are trained with different depths, units, filter sizes, activation functions, and batch size parameters. In the section of *Model Traffic Intensity Predictions of All Districts in Madrid*, the predictions with the best-performed models with all the districts of Madrid will be shown and how the hyper-parameter tuning has been implemented in the study to improve the model performance will be explained in detail.

For both implemented models, the actual values of the test set which has the whole June traffic intensities could be seen with the predicted values in the figures below for the districts 3 and 5:



(a) Model-1 (b) Model-2

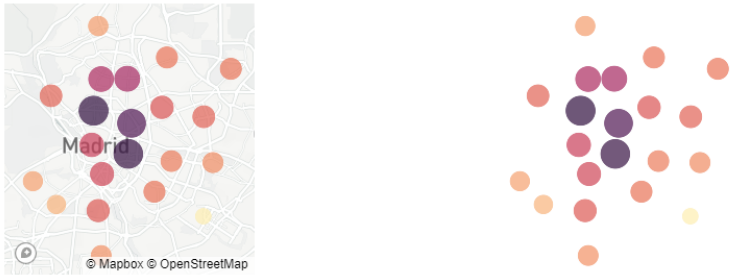
Fig. 5.6. Actual and Predicted Values of District-3 and District-5

5.2.2. Traffic Intensity Prediction with Traffic Intensity Map Images

In this section, the traffic density information for each district was visualized with color and bubble size on a map by using the latitude and longitude information included in the data. With the models created in this section, the traffic density information of the regions in the city of Madrid was trained using maps and the estimation of the density information was studied.

Generating the Image Database

Firstly, the traffic intensity values observed at 15-minute intervals with the location information have been plotted by using the mapbox's API [39]. After that, the produced images were saved in a folder. Later, to only keep the necessary information on each map, the text, street view, and mapbox logo on the background of the images were removed from images by making the background of the map white color. The figures below show an example of this process. Each produced image had 400 height and 400 width sizes.



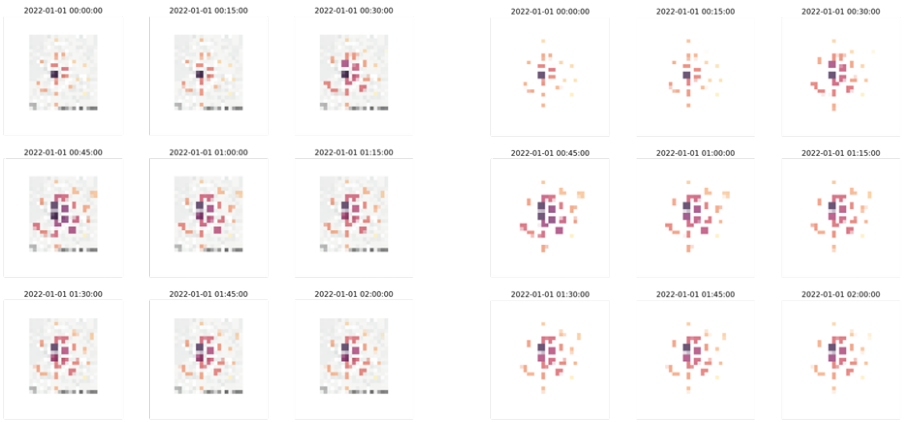
(a) The Traffic Intensity Map on April 1 at 15:15

(b) The Traffic Intensity Map with White background on April 1 at 15:15

Fig. 5.7. The Traffic Intensity Map Image Example

Then, the produced images have been uploaded by using Tensorflow's loading image function. While loading the images, each image has been resized to 33x33 pixels to simplify the high number of pixels, and the images were converted to an array. Every image array representation was appended into a list that had all the images.

As in the previous models, this obtained list of the traffic intensity maps' arrays has been split into train and test samples by 2022-06-01 00:00:00 day-time value as in the first model experiments with numerical data. Therefore, for train and test sets, 14,496 and 2880 images could have been obtained. In that way, train and test images could have been produced with the size of (14496, 33, 33, 3) and (2880, 33, 33, 3). Here, the last parameter of 3 indicates the RGB (Red, Green, Blue) value because the images are colored. If they were loaded with the grayscale, then this last parameter could be 1. In the left figure below, resized (33,33,3) images could be seen for the first 9 times of the data, and in the right visual, the same images with white backgrounds could be seen.



(a) The Traffic Intensity Maps

(b) The Traffic Intensity Maps with White background

Fig. 5.8. The Traffic Intensity Images

Preparing the data for Deep Learning Models

Each pixel value in an image ranges from 0 to 255. Actually, it ranges to 256, because each color channel in RGB is 8 bits and therefore there are $2^8 = 256$ possibilities. [40] As the range starts from zero, then the possible maximum value would be 255. Therefore, dividing all the values by 255 would convert each value between 0 and 1. In that way, before the modeling part, by dividing each pixel value by 255, the pixels of the images could be normalized. For the output of the model, the same districts (districts 3 and 5) have been used as in the first model experiments. Therefore, the steps taken for producing the output values were the same. Moreover, the pixel arrays for each image shape have been rearranged according to the 24-time steps.

Implemented Deep Learning Models

In total, two models have experimented with the image database. The first model built had the same structure as the one-dimensional CNN-LSTM model created in the previous section with numerical data. In the model, only the layers used in the encoding part were changed to two-dimensional layers. In total, 999,598 parameters have been trained in the model. On the other hand, the second implemented model had a very different structure. For feature extraction, VGG16 Net pre-trained model has been used. The VGG16 architecture is shown in the visual below [41]:

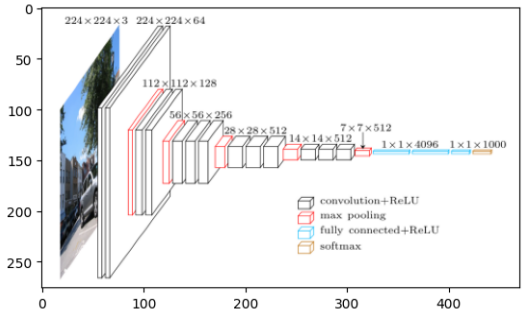


Fig. 5.9. VGG16 Model Architecture

Model Fitting, Predictions and Model Performance Evaluation

The second to last layer of the VGG16 has been used to keep the extracted features from the images. In that way, the shape of (14496, 4096) and (2880, 4096) train and test matrices could be produced. Therefore, thanks to this model structure, 4096 features have been extracted for each image at 15-minute intervals. After obtaining this feature matrix, the matrix was reshaped according to the 24-time steps to be used in the sequence-to-sequence model.

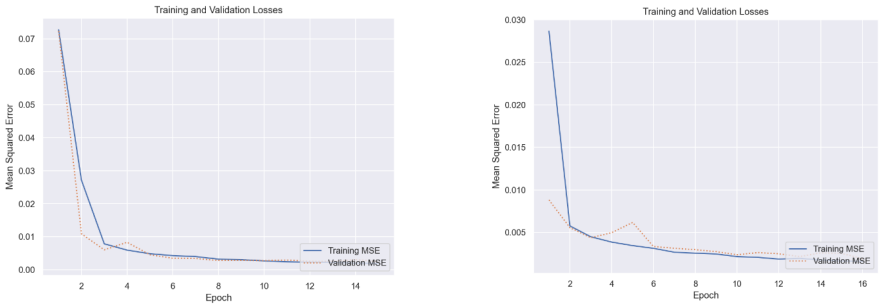
After obtaining the feature matrix with 4096 features thank to the VGG16 pre-trained CNN model, the input of (None, 24, 4096) was given to a Bi-LSTM encoder layer with 100 units and ReLu activation function. As there is a forecasting time step which is the next 4 time steps, then the repeat vector layer was used after the Bi-LSTM encoder. As this repeat vector is a bridge between encoder and decoder parts, after the repeat vector layer, another Bidirectional LSTM layer has been used by returning the sequences. After this layer, the time distributed fully connected dense layer has been implemented to produce the predictions of two district traffic intensity values. The model has been compiled with ADAM optimizer and MSE loss metric. In total as seen in the below figure, 3,598,802 parameters have been trained.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling_2 (Rescaling)      (None, 24, 4096)          0

bidirectional_3 (Bidirectio  (None, 200)               3357600
nal)

repeat_vector_1 (RepeatVect  (None, 4, 200)            0
or)

bidirectional_4 (Bidirectio  (None, 4, 200)            240800
nal)

time_distributed (TimeDistr  (None, 4, 2)              402
ibuted)

=================================================================
Total params: 3,598,802
Trainable params: 3,598,802
Non-trainable params: 0
_________________________________________________________________
```

Fig. 5.10. Bi-LSTM Encoder-Decoder Model Summary

After fitting the model with the same callback functions and epoch, batch size, and validation percentage as in the previous model which was 50, 32 and 0.1 respectively,

the evaluation of training and validation losses could be seen in the figure below for both models:

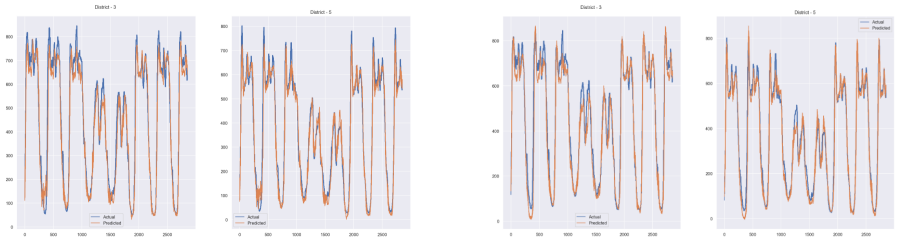


(a) Two-dimensional CNN-LSTM Model

(b) VGG16 + Bi-LSTM Encoder-Decoder Model

Fig. 5.11. Training and Validation Losses of Models trained by Traffic Images

Moreover, on the below visuals, the predictions with actual June values comparison graph is shown:



(a) Two-dimensional CNN-LSTM Model

(b) VGG16 + Bi-LSTM Encoder-Decoder Model

Fig. 5.12. Actual Values of June and Model Predictions

Moreover, the results of the evaluation metrics in the test data of each model trained using the created visuals can be seen in the table below. Evaluation metrics were calculated with the inverse of the scaled traffic intensity values, in other words, the density values were changed to their normal ranges.

Models	MAE	MSE	RMSE	MAPE
Two-dimensional CNN-LSTM	38.09	2645.06	51.43	13.26
VGG16 + Bi-LSTM Encoder-Decoder	35.59	2273.52	47.68	13.41

TABLE 5.4. MODEL EVALUATION METRIC RESULTS OF TWO-DIMENSIONAL ENCODER-DECODER MODELS

According to the results, the combination of VGG16 and Bi-LSTM structure has performed better than the CNN-LSTM model architecture. However, compared to the one-dimensional models which were trained by the numeric data, the model performances

showed worse results in the test set predictions. Although according to the model performance in the test set, the models trained with numerical data have performed better, it has also been proven that the implemented models could learn the traffic density information in a visual form. Thanks to the model experiment with image processing, it has been proven that numerical data can be expressed with images and these images can teach traffic density information to the model. It paved the way for the hypothesis that different visuals could be created and could be experimented with different deep learning models.

5.2.3. Model Traffic Intensity Predictions of All Districts in Madrid

In this section, the model performance and prediction results of all districts in Madrid will be given.

Traffic Intensity Predictions of All Districts in Madrid

To predict all regions' traffic intensity levels, models whose performance was tested the best in the previous sections for the selected 3^{rd} and 5^{th} districts have been chosen. To be trained with the numeric traffic intensity series, the LSTM Encoder-Decoder model has been run again with all the districts. Therefore, in the deep learning model, only the unit parameter has been replaced with 21 which presents the number of districts to estimate. To be trained with the traffic intensity images, the model with the combination of VGG16 and Bi-LSTM encoder-decoder model has been rerun with all the region data. Here, it should be noted again that these two selected models were run with the same model parameters whichever parameters were used in the previous experiments. The only difference was that the models predicted all 21 regions, not just two regions. The overall evaluation metric results of each model could be seen in the below table:

Models	Districts	MAE	MSE	RMSE
LSTM Encoder-Decoder	3,5	19.07	708.83	26.62
LSTM Encoder-Decoder	All	15.67	516.38	22.72
VGG16+Bi-LSTM Encoder-Decoder	3,5	35.59	2273.52	47.68
VGG16+Bi-LSTM Encoder-Decoder	All	34.25	2244	47.37

TABLE 5.5. OVERALL MODEL PERFORMANCES WITH TRAFFIC INTENSITY SERIES AND IMAGES

According to the results obtained, it can be interpreted that for both models, when all the districts in the data have been used to train the models, both models' test prediction performance experienced an increase. This would not be a surprise, because neighboring regions show similar traffic density information as seen in the traffic density images. This means that if the traffic density in an area is high at the measured time, the traffic density in the adjacent region will likely be high as well. This actually explains the traffic flow in

the city of Madrid. Therefore, the use of all regions' data affected the model performance positively.

Since it would be more accurate to examine the model performances according to their performance in each region, the actual and predicted traffic intensity values of each district with their MAPE scores could be seen in the figures below. The first and second figures are the outputs of the first and second models, respectively.

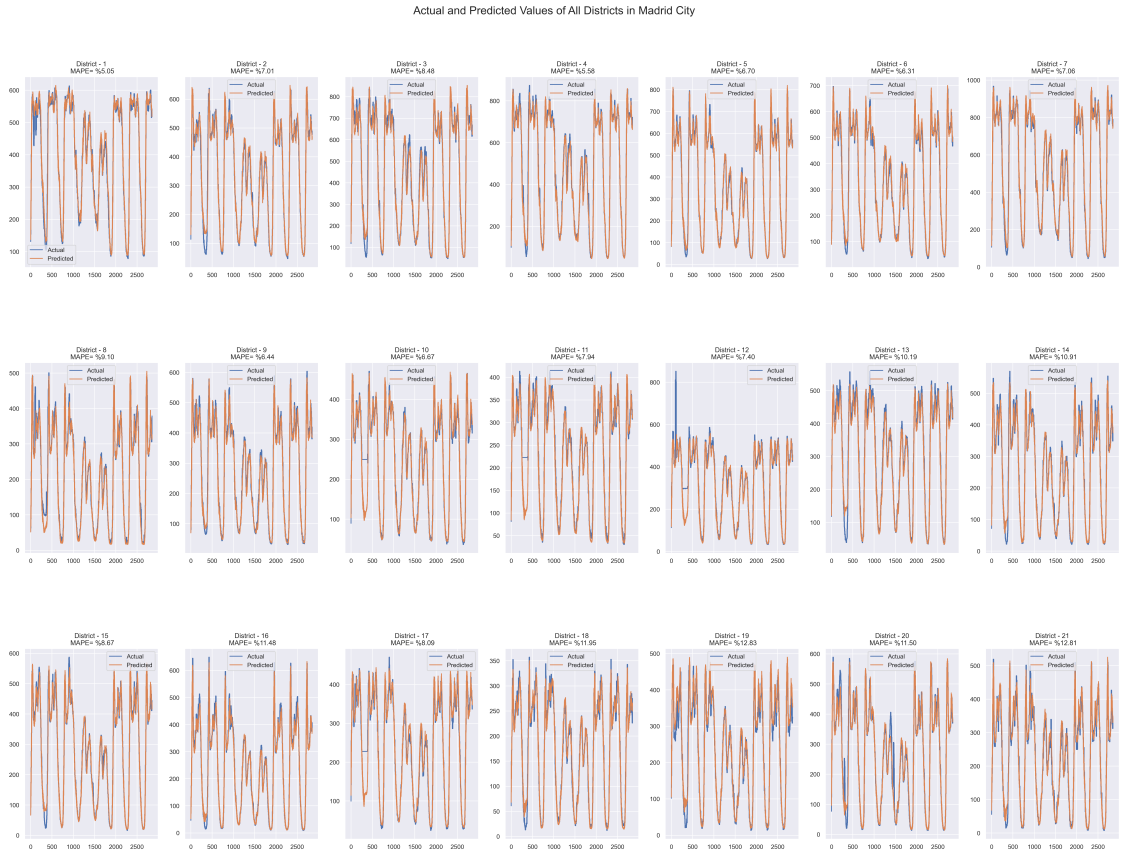


Fig. 5.13. Actual and Predicted Traffic Intensity of All Districts with LSTM Encoder-Decoder Model and MAPE Scores

In the first figure predictions, it could be seen that the model works quite well for each district. However, of course, there are some districts that the model could predict with higher accuracy. Also, it should be noticed that the neighbor districts had similar prediction errors in general. The district that had the least error for this model was 1^{st} district with %5 MAPE rate.

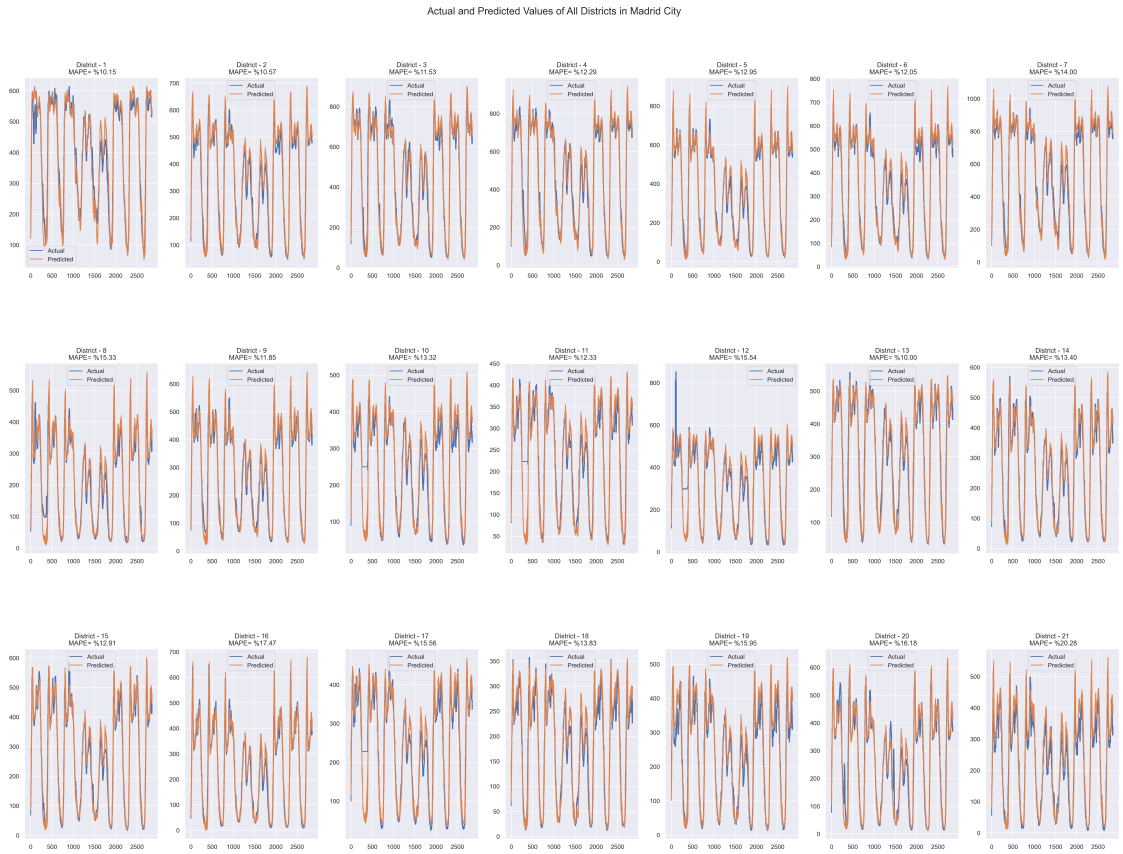


Fig. 5.14. Actual and Predicted Traffic Intensity of All Districts with VGG16 + Bi-LSTM Encoder-Decoder Model and MAPE Scores

In the second figure predictions, it could be seen that model predictions are slightly worse than the previous model results, however, it is still a good model which proves that the implemented model could learn with the generated traffic images. In this figure also it can be seen that the neighbor districts had similar prediction errors. The district that had the least error for this model was 13^{th} district with %10 MAPE rate.

Hyper-parameter Tuning to Enhance the Model Performance

As with the methods in machine learning, deep learning methods are established by specifying a lot of different parameters such as the filter size in convolutional layers, unit size in LSTM and Bi-LSTM, activation functions, batch size, dropout, rate, etc. For this reason, finding the most accurate parameter affects the model performance considerably. In this part, the most optimal model has been tried to be found by making some parameter trials on the model that gives the best result. As seen in the previous results, the model in which LSTM is used as both an encoder and a decoder gave the best results among the methods tried in this study. Therefore, in this part, this model's hyper-parameters and its architecture will be focused on. The hyper-parameter optimization process has been done by using the Keras Tuner package in Python. [42] In the Keras tuner package, there are many options to do hyper-parameter optimization such as grid search, random search,

and bayesian search. In this study, to have less computational time, a random search approach has been implemented as a parameter search algorithm. The parameters tried to be optimized were, unit size in LSTM layers, dropout rate, activation function choice in the time distributed dense layer, batch size while fitting the model, and the validation split parameter, respectively. The range of the unit size for LSTM has been defined as between 40 and 400 at 20 steps, the dropout parameter search range has been set from 0 to 0.5 at 0.1 increase, and the activation functions to be tried have been defined ReLu and sigmoid as the output was aiming to be positive. Lastly, for the model fitting parameters, batch size has been searched from 8 to 64 with 16 steps and the validation split has been experienced with 0.1 or 0.2 as a choice. Moreover, as in the literature, Bi-LSTM has over-performed compared to basic LSTM, therefore, in the model, the unidirectional LSTM layer has been replaced by the bidirectional LSTM to enhance the model performance.

In the hyper-parameter optimization study, a total of 3 trials were tried in a random search. In the image below, the output of the first trial result can be seen as an example.

```
Trial 1 Complete [00h 05m 15s]
mse: 0.0017367934342473745

Best mse So Far: 0.0017367934342473745
Total elapsed time: 00h 05m 15s

Search: Running Trial #2

Value             |Best Value So Far |Hyperparameter
320               |60                |units
0.2               |0.4               |Dropout_rate
sigmoid           |sigmoid           |dense_activation
8                 |32                |batch_size
0.2               |0.1               |validation

Epoch 1/50
1447/1447 [==============================] - ETA: 0s - loss: 0.0043 - mse: 0.0043INFO:tensorflow:Assets written to: path_to_che
1447/1447 [==============================] - 274s 189ms/step - loss: 0.0043 - mse: 0.0043 - val_loss: 0.0020 - val_mse: 0.0020
Epoch 2/50
 266/1447 [====>.........................] - ETA: 3:26 - loss: 0.0018 - mse: 0.0018
```

Fig. 5.15. The First Trial Result in Random Search with Keras Tuner

As seen after each trial, the tuner returns the best values so far as indicated in the figure above. Then, the random search objective function looks for the parameters that minimize the mean squared error value in every trial. Finally, each parameter trial range and the best found optimal parameters with the model structure summary could be seen in the figure below:

```
Search space summary
Default search space size: 5
units (Int)
{'default': None, 'conditions': [], 'min_value': 40, 'max_value': 400, 'step': 20, 'sampling': None}
Dropout_rate (Float)
{'default': 0.0, 'conditions': [], 'min_value': 0.0, 'max_value': 0.5, 'step': 0.1, 'sampling': None}
dense_activation (Choice)
{'default': 'relu', 'conditions': [], 'values': ['relu', 'sigmoid'], 'ordered': False}
batch_size (Int)
{'default': 32, 'conditions': [], 'min_value': 8, 'max_value': 64, 'step': 16, 'sampling': None}
validation (Choice)
{'default': 0.1, 'conditions': [], 'values': [0.1, 0.2], 'ordered': True}
INFO:tensorflow:Oracle triggered exit
{'units': 320, 'Dropout_rate': 0.2, 'dense_activation': 'sigmoid', 'batch_size': 8, 'validation': 0.2}
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional (Bidirectional (None, 640)               875520
_________________________________________________________________
repeat_vector (RepeatVector) (None, 4, 640)            0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 4, 640)            2460160
_________________________________________________________________
dropout (Dropout)            (None, 4, 640)            0
_________________________________________________________________
time_distributed (TimeDistri (None, 4, 21)             13461
=================================================================
Total params: 3,349,141
Trainable params: 3,349,141
Non-trainable params: 0
```

Fig. 5.16. Random Search Search Space and
The best chosen Model

Afterward, with these best parameters, the new model has been trained and evaluated in the test set and the evaluation metric results were found as follows:

Model	MAE	MSE	RMSE
Optimized Bi-LSTM Encoder-Decoder	13.67	498.87	19.32

TABLE 5.6. OVERALL MODEL PERFORMANCE OF BI-LSTM ENCODER-DECODER MODEL WITH OPTIMIZED PARAMETERS

Compared to the previous results, it can be seen that there has been a clear performance increase in model error metrics according to the predictions in the test traffic intensity series.

6. CONCLUSIONS AND FUTURE IMPROVEMENTS

In this thesis, the traffic intensity information recorded by the sensors placed at URB and M-30 roads at 15-minute intervals from January 2022 to June 2022 was first examined on a regional basis. After doing comprehensive data analysis and getting important insights from the traffic data, only URB type of sensors decided to use due to having a lot of missing records of M-30 type of sensors data in the examined period.

Then, the pre-processed regional traffic intensity data set was given to the model in the form of numerical time-distributed series and traffic intensity map images which were visualized by using the latitude and longitude information of the traffic density information. To estimate the traffic intensity level of each district, various deep learning models were created using CNN and LSTM structures. In the study, encoder-decoder models have been implemented for sequence-to-sequence prediction. As in the sequence-to-sequence model architecture, there is a parameter called time-step, which presents the main time series in subsequent series, that have been generated. In the experiments, the input traffic intensity series have been re-sampled according to the 24-time steps looking back parameter and 4-time steps forecasting range. In the data, as each single time step equals 15-minute intervals, therefore, 24 and 4 time-steps correspond to six and one-hour periods, respectively. Therefore, deep learning models that predict 1 hour ahead were created by training the patterns of the regional traffic density data of the previous 6 hours.

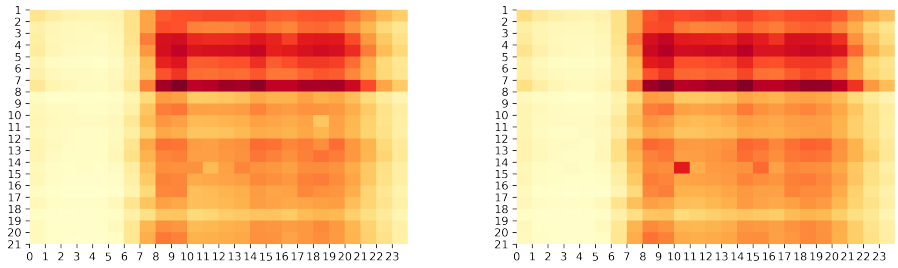
In the experiments, first, to test the implemented deep learning models, only districts 3 and 5 were tried to forecast due to having a huge computational time when running the models with all the districts in Madrid. After the model training, the predictions were obtained in the test set, and the estimations were compared with the real data of districts 3 and 5 by calculating the MSE, RMSE, MAE, and MAPE error metric values. The results showed that the all models tried were performing way better than the implemented baseline model, therefore, the best performing models among all the models were rerun with all the districts in the final section of the thesis. It was observed that the performance of the models trained with only the 3^{rd} and 5^{th} districts was lower than the models trained with all districts. This proved that as neighboring regions show similar traffic density characteristics, the model could make more accurate predictions by using this spatial correlation. For example, the RMSE value of the trained LSTM encoder-decoder model decreased from 26.62 to 22.72 after training with all districts. In the last step, in order to increase the performance of the best model, the model hyper-parameters were tried to be optimized using the Keras Tuner package and the performance of the LSTM encoder-decoder model was increased to a higher level by implementing a Bi-LSTM encoder-decoder model with optimized parameters. In that way, the prediction accuracy of the June traffic data was reduced to 19.32 based on the RMSE value. Apart from the overall error rate, when the estimation error rates were examined for each district, the MAPE

values were calculated in the range of %5-12 in the districts.

Moreover, the experiments showed that the models that learned the traffic intensity data in the numerical form reached a higher accuracy in the test data than the models that learned the traffic information from the visual intensity maps. On the other hand, it should be noted that all the tested models performed much better than the base model. This means that the performance of the models trained with traffic intensity map images was also very good and the traffic density information could have been learned using the generated visuals. This result proves that the traffic density information could be visualized using different visualization methods and different experiments could be performed using powerful deep learning architectures such as CNN.

For future improvements, models could be trained with more days as in the open data portal of Madrid city, there are monthly traffic data from sensors since 2013. After increasing the training samples, the prediction period could be extended and long-term predictions could be made. In addition to this, as in this study, 24-time step looking back and 4-time step forecasting range parameters have been defined for sequence-to-sequence deep learning models, these parameters could be changed to produce different forecasting periods and the training performance could be examined by trying different looking back time step sizes.

Moreover, in the part of visualizing the traffic intensity information as images and training the model with these produced images, different types of images could be used to try to train the model more. As mentioned before in the literature review part, there were various studies about estimating the traffic data by learning from the images. More visuals made and used in this area could be examined, minute traffic density information could be illustrated with different visuals, and the created models could be trained with these visuals again. The possible improvement could be visualizing each day of the traffic data as seen in the figure below, and producing long-term forecasts to estimate the next days' average traffic intensity level of each district in Madrid city.



(a) 18^{th} of January,2022 (b) 19^{th} of January,2022

Fig. 6.1. Traffic density at all hours of the day in the form of heat-map images

BIBLIOGRAPHY

[1] P. Purnawansyah, H. Haviluddin, R. Alfred, and A. F. O. Gaffar, "Network traffic time series performance analysis using statistical methods," *Knowledge Engineering and Data Science*, vol. 1, no. 1, pp. 1–7, 2017.

[2] Y. Liu, H. Zheng, X. Feng, and Z. Chen, "Short-term traffic flow prediction with conv-lstm," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, IEEE, 2017, pp. 1–6.

[3] M. R. McCord, Y. Yang, Z. Jiang, B. Coifman, and P. K. Goel, "Estimating annual average daily traffic from satellite imagery and air photos: Empirical results," *Transportation Research Record*, vol. 1855, no. 1, pp. 136–142, 2003.

[4] X. Xu, C. Liu, Y. Zhao, X. Yu, and X. Wu, "Short-term traffic volume forecast method based on cnn-lstm-at," 2021.

[5] M. Khajeh Hosseini and A. Talebpour, "Traffic prediction using time-space diagram: A convolutional neural network approach," *Transportation Research Record*, vol. 2673, no. 7, pp. 425–435, 2019.

[6] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2014.

[7] W. Jin, Y. Lin, Z. Wu, and H. Wan, "Spatio-temporal recurrent convolutional networks for citywide short-term crowd flows prediction," in *Proceedings of the 2nd International Conference on Compute and Data Analysis*, 2018, pp. 28–35.

[8] N. Ranjan, S. Bhandari, H. P. Zhao, H. Kim, and P. Khan, "City-wide traffic congestion prediction based on cnn, lstm and transpose cnn," *IEEE Access*, vol. 8, pp. 81 606–81 620, 2020.

[9] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "Lstm network: A deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.

[10] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[11] P. G. Gallery. "Heatmap for timeseries." (), [Online]. Available: https://www.python-graph-gallery.com/heatmap-for-timeseries-matplotlib.

[12] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015. DOI: 10.1109/TITS.2014.2345663.

[13] O. D. P. of The Madrid City Council. "Historical traffic data of the madrid city." (2022), [Online]. Available: https://datos.madrid.es/sites/v/index.jsp?vgnextoid=33cb30c367e78410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD.

[14] D. P. of The Madrid City Council. "Location of the traffic measurement points." (2022), [Online]. Available: https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=ee941ce6ba6d3410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD.

[15] parsehub. "What is web scraping?" (), [Online]. Available: https://www.parsehub.com/blog/what-is-web-scraping/.

[16] A. Spark. "Apache spark offical website." (), [Online]. Available: https://spark.apache.org/.

[17] I. A. Basheer and M. Hajmeer, "Artificial neural networks: Fundamentals, computing, design, and application," *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[18] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

[19] S. Pattanayak, "Introduction to deep-learning concepts and tensorflow," in *Pro deep learning with TensorFlow*, Springer, 2017, pp. 89–152.

[20] L. Alzubaidi *et al.*, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[21] N. K. Manaswi, "Rnn and lstm," in *Deep Learning with Applications Using Python*, Springer, 2018, pp. 115–126.

[22] Y. D. Prabowo, H. L. H. S. Warnars, W. Budiharto, A. I. Kistijantoro, Y. Heryadi, *et al.*, "Lstm and simple rnn comparison in the problem of sequence to sequence on conversation data using bahasa indonesia," in *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, IEEE, 2018, pp. 51–56.

[23] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The performance of lstm and bilstm in forecasting time series," in *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, 2019, pp. 3285–3292.

[24] K. Khalil, B. Dey, A. Kumar, and M. Bayoumi, "A reversible-logic based architecture for long short-term memory (lstm) network," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2021, pp. 1–5.

[25] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[26] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in *International conference on artificial neural networks*, Springer, 2005, pp. 799–804.

[27] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," *arXiv preprint arXiv:1801.02143*, 2018.

[28] R. L. Abduljabbar, H. Dia, and P.-W. Tsai, "Unidirectional and bidirectional lstm models for short-term traffic prediction," *Journal of Advanced Transportation*, vol. 2021, 2021.

[29] R. Mushtaq, "Augmented dickey fuller test," 2011.

[30] V. Flovik. "How (not) to use machine learning for time series forecasting: Avoiding the pitfalls." (), [Online]. Available: https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424.

[31] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, pp. 2657–2661.

[32] L. Hou, J. Zhu, J. Kwok, F. Gao, T. Qin, and T.-y. Liu, "Normalization helps training of quantized lstm," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[33] A. Vidhya. "Multivariate multi-step time series forecasting using stacked lstm sequence to sequence autoencoder." (), [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/multivariate-multi-step-time-series-forecasting-using-stacked-lstm-sequence-to-sequence-autoencoder-in-tensorflow-2-0-keras/.

[34] S. Mehtab, J. Sen, and A. Dutta, "Stock price prediction using machine learning and lstm-based deep learning models," in *Symposium on Machine Learning and Metaheuristics Algorithms, and Applications*, Springer, 2020, pp. 88–106.

[35] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, "Learning pooling for convolutional neural network," *Neurocomputing*, vol. 224, pp. 96–104, 2017.

[36] K. C. API. "Early stopping." (), [Online]. Available: https://keras.io/api/callbacks/early_stopping/.

[37] ——, "Model check point." (), [Online]. Available: https://keras.io/api/callbacks/model_checkpoint/.

[38] ——, "Reduce lr on plateau." (), [Online]. Available: https://keras.io/api/callbacks/reduce_lr_on_plateau/.

[39] plotly. "Scatter mapbox." (), [Online]. Available: https://plotly.com/python/scattermapbox/.

[40] Wikipedia. “Rgb color model.” (), [Online]. Available: https://en.wikipedia.org/wiki/RGB_color_model.

[41] W. Nash, T. Drummond, and N. Birbilis, “A review of deep learning in the study of materials degradation,” *npj Materials Degradation*, vol. 2, no. 1, pp. 1–12, 2018.

[42] K. Tuner. “Keras tuner.” (), [Online]. Available: https://keras.io/keras_tuner/.