Cairo University-Faculty of Engineering

Electronics & Electrical Communication Engineering Department

Graduation project 2020

Traffic Congestion Prediction and Recommendation System

TrCPRS team:

Yomna Sayed Mohamed -Mariam Mohamed Farouk - Rana Hussein Ali

Mohamed Hazem Labib -Omar Wael Mashaal -Omar Essam Mohamed

Under Supervision of:

Prof. Amin Nassar

Prof. Hassan Mostafa

Advisors:

Prof. Samah El-Shafiey El-Tantawi

Prof. Hossam Abd El-Gawad

Table of Content

Table of Figures

Acknowledgment

First and foremost, this thesis not only describes the results of our work but also reflects the assistance and guidance of many generous and stimulating people.

Huge thanks to our beloved families, **father, mother, brothers, and sisters**, to those how supports us and believed in us, to supervisors, **Dr. Amin Nassar, Dr. Hassan Mostafa,** and our advisors **Dr. Samah El-Shafiey El-Tantawi**. Without their assistance and dedicated involvement in every step throughout the process, this project would have never been accomplished. our Sincere, gratitude, and appreciation for their support, understanding, excellent guidance, caring, patience, and immense help in planning and executing this project promptly.

Their tender love and support have always been the cementing force for building what we achieved.

Finally, we would also like to show gratitude to **Dr. Hossam Abd El-Gawad**, who was always there for any questions, providing precious advice and sharing his time and experience, He raised many precious points in our discussion and we hope that we have managed to address several of them here.

To all we extend our sincere thanks from:

Yomna Mariam Rana

Mohamed Omar Essam Omar Wael

Chapter 1

Introduction

The increases of population density and the relative amount of car owners makes traffic jam a serious problem in the whole world.
Traffic jam is a major source for discomforting drivers, and also the cause of an increasing number of traffic accidents.

1.1 Traffic Congestion Impacts

The greatest impact of traffic congestion is on economics, Table1 shows the impact of traffic congestion on some countries, also causes serious health issues.

Area	Loss in Billion	Note
USA	$305 [11]	[12]
UK	$52.01	[13]
NYC	$33.7	
LA	$19.2	[14]
Manila	$18.615	[15]
Bangladesh	$11.4	[16]
SF	$10.6	
Atlanta	$7.1	
Jakarta	$5	[17]
Dhaka	$4.463	[18]
GTHA	$3.3	[19]

Figure1.1 traffic congestion impact

1.2 Problem Statement

Traffic congestion has a lot of causes and impacts, solving it is a great challenge for all governments and countries worldwide.

Accidents: Even simple accidents cause traffic congestions because of curiosity leads to low speed.

Income: Higher-income can cause more traffic congestion leading to less social and economic development because it leads to an increment in the card numbers.

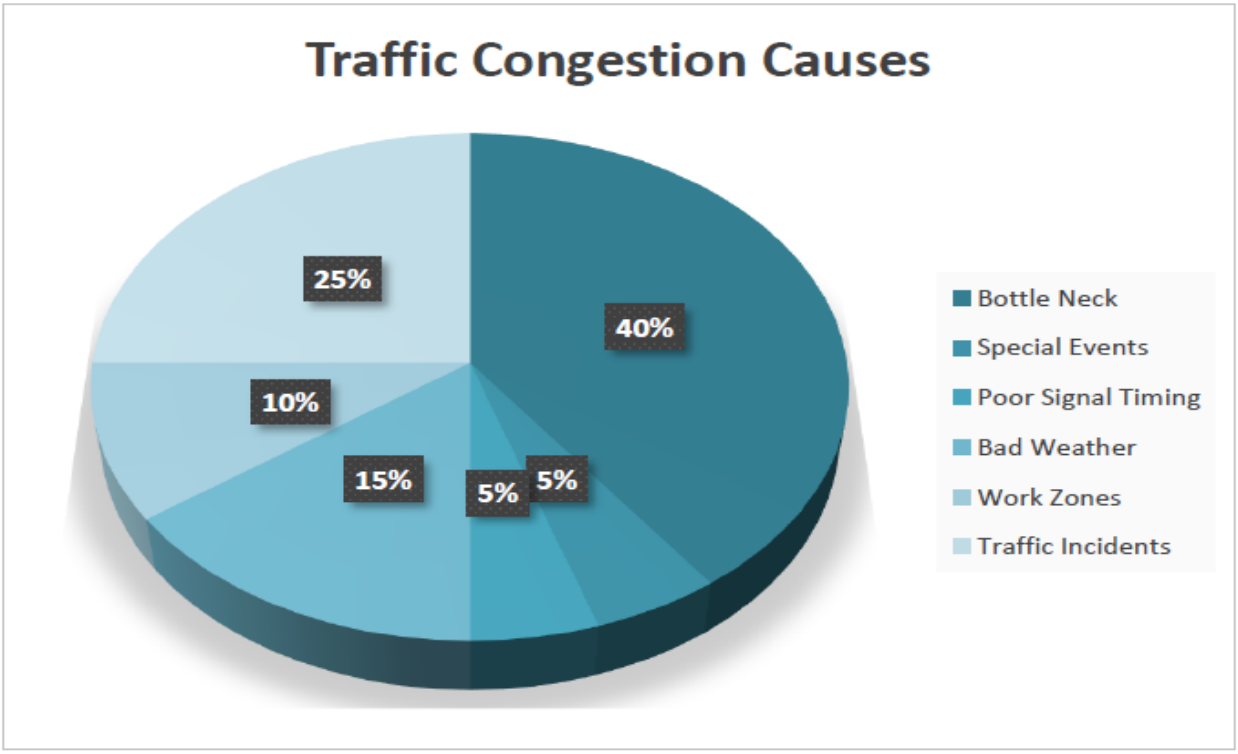


Figure1.2 traffic congestion causes

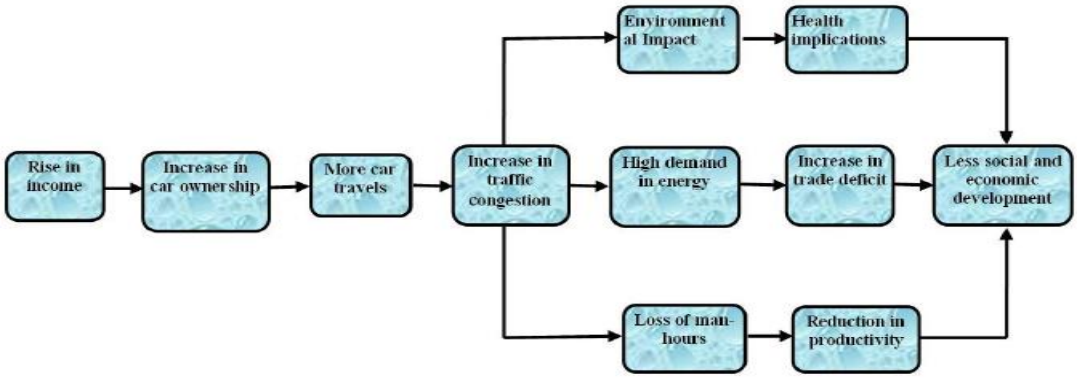


Figure1.3 impact graph

1.3 Solution approach

1.3.1 Former Solutions:

To minimize the severe effect of traffic congestion are carelessness and accidents, these strategic policies were used:

1. The strict application of traffic rules and regulation.
2. Awareness campaign.
3. Strict licensing rules
4. Better car inspection tests.
5. Encouraging mass transport.
6. Improving infrastructure.
7. Improving monitoring systems.

1.3.2 Our Approach:

By using machine learning algorithms and supervised learning models like RNN (Recurrent Neural Network), traffic congestion in a given route can be predicted. Following the next parameters equation:

Rush hours + gathering facilities + school time + weather + number of cars +

the capacity of the road + accidents occurrence + stadiums and matches time

A website and android application which take your origin and destination to predict your delay time, suggest other routes to prevent traffic jam, and lastly will suggest the best position for the routes and transportation modes barking that will be set by the government.

1.4 Benefits

1.4.1 User benefits:

- Our system will lead to faster trips.
- Solving traffic congestion would increase the safety of the routes.
- Offer another transportation mode which will solve the congestion in the long-time.

- Solve a lot of health issues where congestion causes mental and physiological effects.

1.4.2 Industrial benefits:

- Automotive cars because it will make safer routes and reduce the complex situations that will probably lead to more failures of the self-driving cars, with companies like Valeo.
- Implement the system we will make in any touchpad at any car with one, as it's a Linux kernel system with companies like Valeo.
- Transportation modes services like Swvl, Careem, Uber, etc.
- Provide a booking system for the companies, which support some transportation modes like buses or bikes.

1.4.3 Governmental benefits:

- Collected data will in modifying the traffic system in our specific region.
- Our suggestion system will guide in-charge facilities to start new routes and parking in the most effective way to enhance traffic health in our country.
- Effectively will increase the economy in Egypt as transportation will be easier.
- Improve the total traffic infrastructure.
- Improve tourism as the traffic jams are an annoying cause for the tourists.

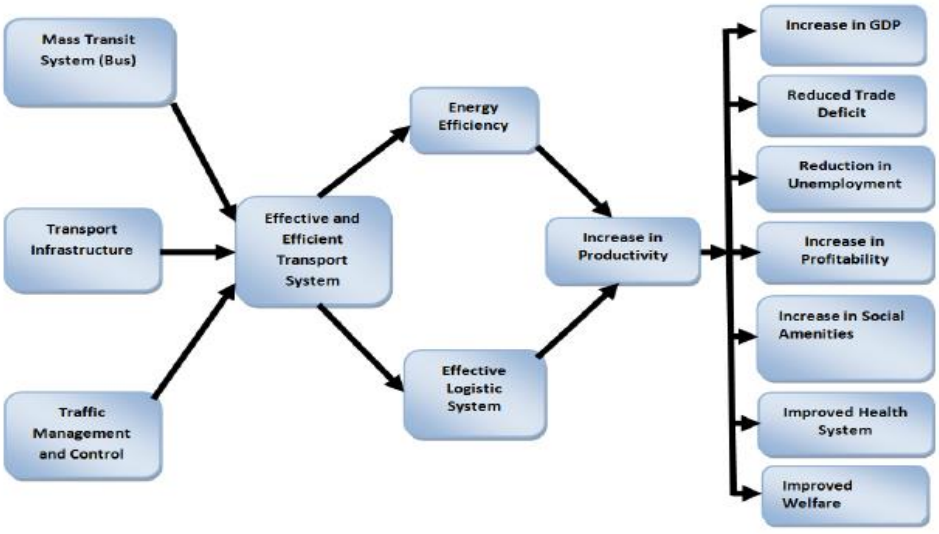


Figure1.4 outcome of our approach on society

Chapter 2
Data Collecting

"More data beats clever algorithms, but better data beats more data."
-Peter Norvig

Data is the engine of every AI projects around us starting from a simple graduation project to a large scale of effect projects like Google's self-driving car, amazon's suggesting systems…etc.

When it comes to the data used in your project, it's a fatal decision to make as your data will decide which path your model will take and the end of it too.

2.1 Region of interest

This part will describe the process of choosing the area that will be studied and try to extract as much traffic data, to use it further to train the model that will predict the traffic state in each street in the few next minutes.

Few criteria will be studied to decide which area to choose, but first, let us see the recommended areas, that can be chosen.

- **One of the USA states**: the motivation beyond this option is the USA is one of the most countries that have daily online traffic data providers that are very accurate, free, and minutely refreshed.

- **The Giza Governorate**: the motivation beyond this option is the Giza governorate is a place we all see and can measure by real trips the efficiency of the data and how much it is real and true and also, it's a governorate that is full of variations and random pattern along the day.

- **The Alexandria Governorate**: the motivation beyond this option is The Alexandria Governorate has more variations in weather and events and more seasons variation like in summer there is higher population than the rest of the year so it could be more suited for the variations of events and weather.

Now we have three different regions to start to study the traffic states in it, let us discuss the criteria that will define which region will be used:

- Must have unpredictable Traffic states like sudden incidents or traffic jams in non-rush hours
- Must suffer from events and high distribution of facilities
- Must be able to well-define the traffic states for most of the regions in Egypt
- Must have an online real-time data for the traffic states in it, in any online traffic data providers.
- Must have a normal weather state like the other regions of Egypt.

Regarding these criteria, **The Giza Governorate** is the most proper choice and can be used as a source for daily traffic data that will be used for training and testing the model.

2.2 Streets Network and Features

After choosing the most suitable region (**The Giza Governorate**), Now time to extract Giza network and the streets features to help to collect traffic data after, in this field, there are two possible chooses **Google Maps** and **Open Street Map (OSM),** as OSM is a free and refreshed source to extract Giza network of streets and its features, so it was the best platform to use.

OSM is a simple metadata source for maps with different layers and transportation modes, after choosing the transportation layer with care mode to identify all the streets that are suitable to travel in with cars, we get a **.osm** file**,** that is consists of two parts, nodes **Figure2.1** that has features and identify one latitude

and longitude couple, and ways **Figure2.2** that consists of a group of nodes and identify a route from its first node to its last node then using Java to Transfer it into .csv file **Figure2.3** consists of 31 feature and 103901 street.

After downloading data from OSM for Giza Governate it looks like **Figure2.4**

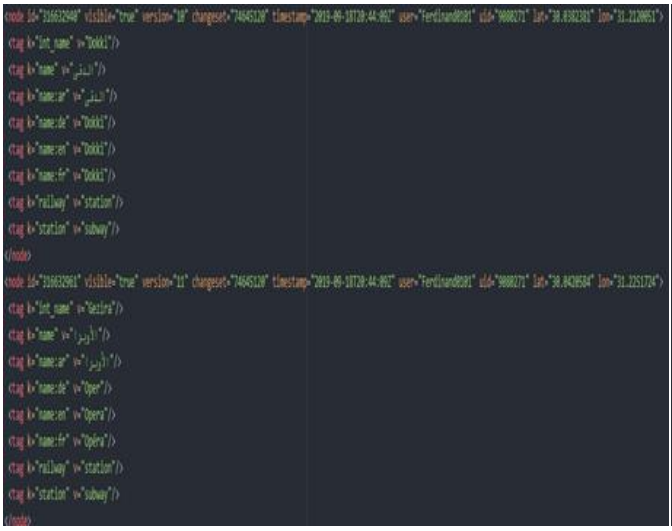


Figure 2.1 Nodes in OSM file example

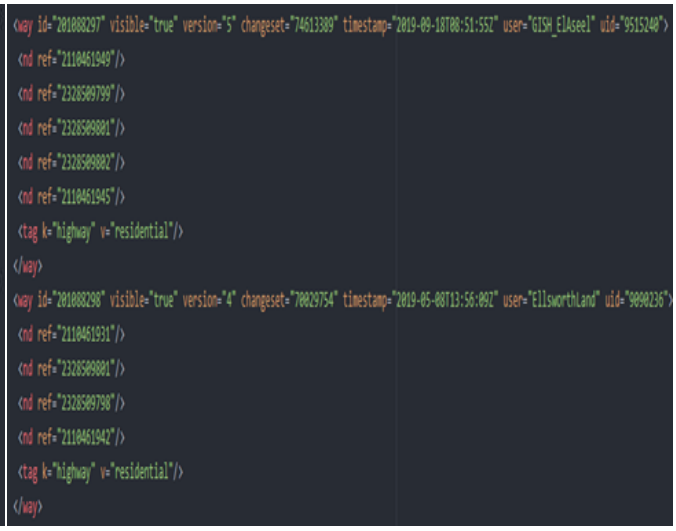


Figure 2.2 Ways in OSM file example

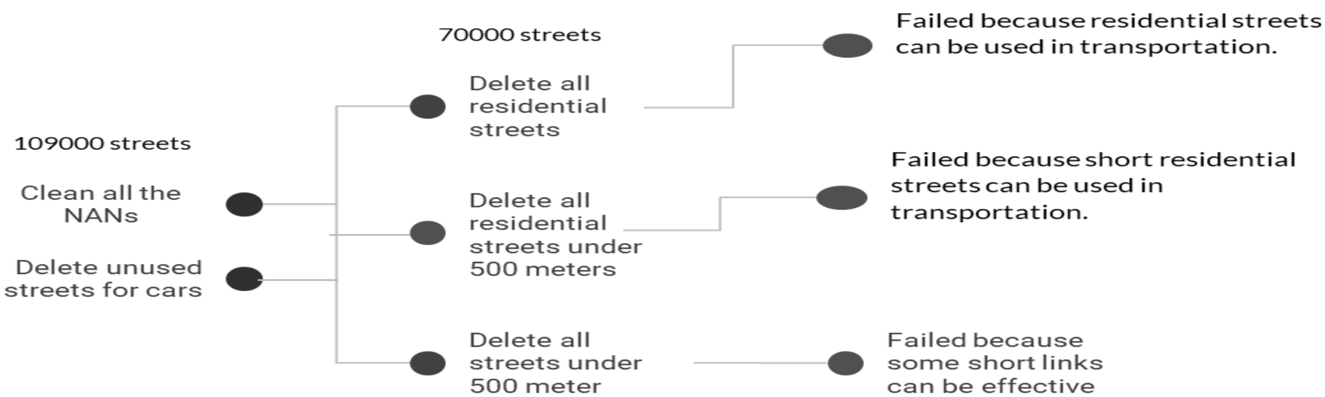
	id	longf	latf	longL	latL	name	name_en	name_ar	type	oneway	landuse	surface	bridge	layer	bicycle	foot	tunnel	service	construct	access	amenity	horse	width	alt_name	alt_name	lanes	roof_sha	addr_pos	crossing	history	junction
2	4987193	31.22066	30.04421	31.22685	30.04348	ش طوسون	Mahmoud	ش طوسون	residenti	yes																					
3	5091220	31.1009	30.01473	31.09625	30.01433				residential																						
4	5091224	31.09849	30.01194	31.09941	30.01008				residential																						
5	5096132	31.20695	30.06256	31.21088	30.06683	ش عمر طو	Omar Tou	ش عمر طو	residential													2									
6	5096152	31.20811	30.06176	31.1968	30.06961	ش احمد عل	Ahmed Ok	ش احمد عل	primary	yes																					
7	22947883	31.19652	30.0694	31.19861	30.06787	ش احمد عل	Ahmed Ok	ش احمد عل	secondar	yes			yes	1																	
8	22948103	31.21178	30.04722	31.21192	30.04738				trunk_lin	yes		paved	yes	1																	
9	22948117	31.21192	30.04738	31.24745	30.06227	ش احمد عل	6th Octob	ش احمد عل	trunk	yes		paved	yes	1																	
10	24574603	31.22701	30.04674	31.22011	30.04624	ش طوسون	Al Borg st	ش طوسون	residenti	yes		paved																			
11	24668816	31.13306	29.98356	31.13313	29.98194				service																						
12	24668876	31.1402	29.97502	31.13283	29.97754				service																						
13	24749289	31.22686	30.04338	31.22173	30.0405	ش طوسون	Al Tahrir	ش طوسون	primary	yes		paved																			
14	24749354	31.22173	30.0405	31.22032	30.04018	ش احمد عل	Al Gala' E	ش احمد عل	primary	yes		paved	yes	1																	
15	24749394	31.21994	30.03999	31.21994	30.03999				primary	yes																		roundabout			
16	24749403	31.21944	30.04008	31.19825	30.03507	ش طوسون	Al Tahrir	ش طوسون	primary	yes		paved								Al Behou	ش طوسون										
17	24764150	31.21615	30.02848	31.21091	30.02787	ش احمد عل	Nahda Mi	ش احمد عل	primary	yes																					
18	24764157	31.21827	30.02859	31.21615	30.02848	ش طلعت ح	Midan El	ش طلعت ح	primary	yes																					
19	24764172	31.21826	30.0285	31.22343	30.02836	ش احمد عل	Aljamiea	ش احمد عل	primary	yes			yes	1																	
20	24764228	31.22343	30.02836	31.23002	30.02902	ش طوسون	Al Saraya	ش طوسون	primary	yes																					
21	24764295	31.09845	30.01364	31.09837	30.0136				residential																						
22	24853225	31.22013	30.21187	31.20315	30.27668	شارع طلعت	Cairo	شارع طلعت	trunk	yes					asphalt											3						90
23	24869131	31.21394	30.06655	31.22516	30.09159	ش طوسون	Kornish Al	ش طوسون	primary	yes										Gamal Ab	ش طوسون										

Figure2.3 CSV file Example

Figure2.4 OSM network for Giza

2.3 Streets Dataset Cleaning

After collecting the streets data, we know need to optimize the number of streets so that we can finally work with 1000 streets because we are limited to a maximum number of requests per day, we know will follow a sequence of trials to achieve this aim and how we solved it.



Graph 2.5 Trials to choose streets sample

2.3.1 As in **Graph2.5** We started with removing all the NAN values from the dataset because these samples don’t contain the required information, we want like **Figure2.6**, it’s obvious that two last samples missing their type so they won't be useful at all.

72	25012962	31.09978	30.01418	31.0993	30.01506				path			
73	25048758	30.90587	30.11889	30.93624	30.11119							asphalt
74	25048803	30.91088	30.13234	30.91945	30.10325							asphalt

Figure 2.6 Missing values example

2.3.2 After deleting all the missing values we then deleted all the streets that cannot be used in a transportation task like:

- Construction
- Unclassified
- Footway
- Service

These streets are used for interconnections inside a building or a bikes way, so they cannot be included in our task.

2.3.3 After those successes filtration we settle with 70000 streets and give three long shots trying to optimize more but they were not acceptable as we try to filter under 500-meter streets with the attempt of decreasing the least effective streets but we find out that those streets still can be effective on the transportation tasks so we still need to find out another way to solve this dilemma.

2.3.4 Solution of this optimization problem can be achieved by separating streets into three categories: residential, non-links, links, then using a sigmoid-like function **equ2.1** that try to take a 0-10 number of streets from a cluster depends on the average street length on this cluster, as the average length increases the number of taken streets from this specific cluster increases as we see in **Figure2.7,** now we have 1000 streets that represent the total population.

$$\frac{1}{1+ab^{x}} \qquad \textbf{Equ2.1}$$

Where a and b are constants calculated by trial and error and x is the average group of street length.

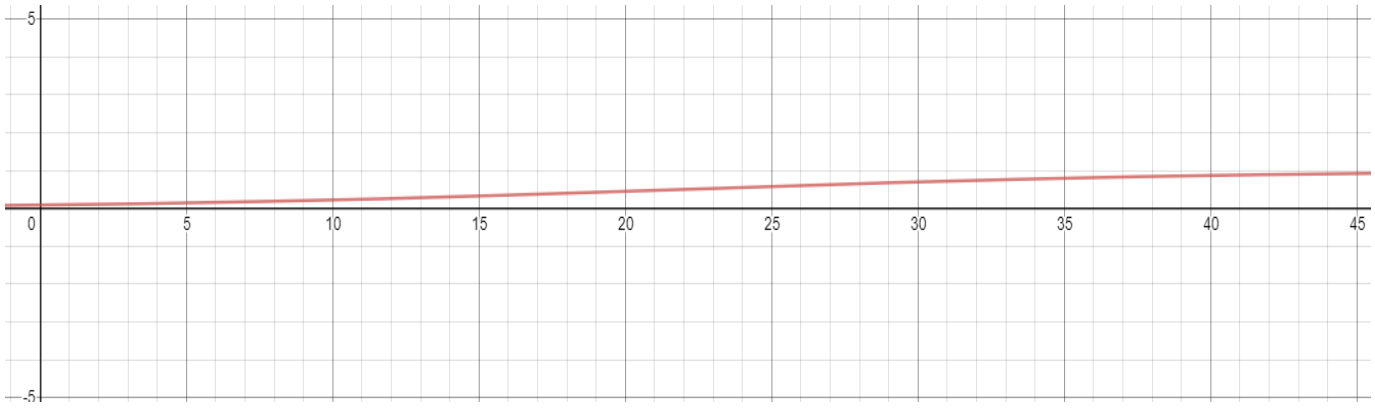
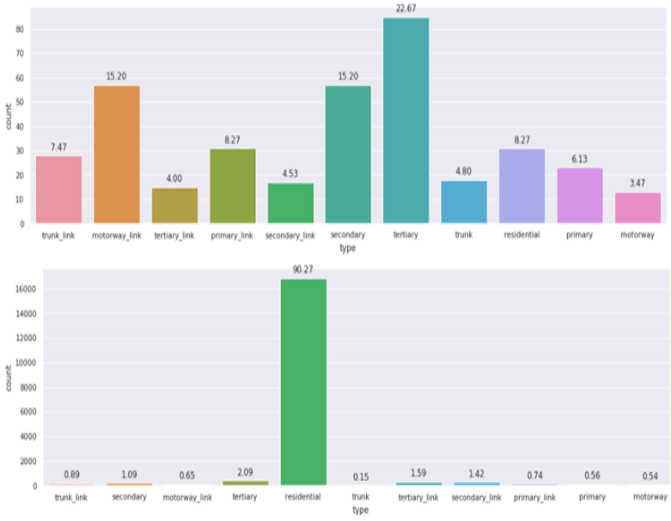


Figure2.7 Filtration function for decreasing streets number

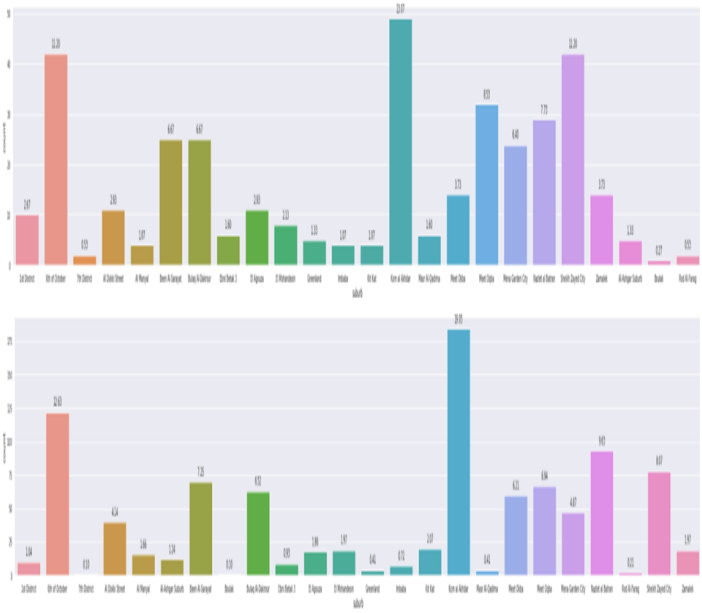
2.4 Cleaning Results

In data science we should intensely study the dataset after cleaning, to observe the effect of the cleaning technique on our dataset, because sometimes the dataset affected badly and the cleaning process manipulate the dataset distribution, so we will see some graphs to show the effect of our cleaning technique.



Comment: Before cleaning we see in the lower graph the distribution is much more unbalanced as residential streets are 90.27% of the data, and after cleaning the distribution we can observe the important streets that are mainly used in transportation is the scope of the modified dataset.

Figure 2.8 Streets types distribution after and before cleaning



Comment: The distribution of streets in the different suburbs should not change dramatically as we want to have the same focus on important suburbs after the cleaning too, and that what happened exactly.

Figure 2.9 Distribution of streets over the different suburbs in Giza

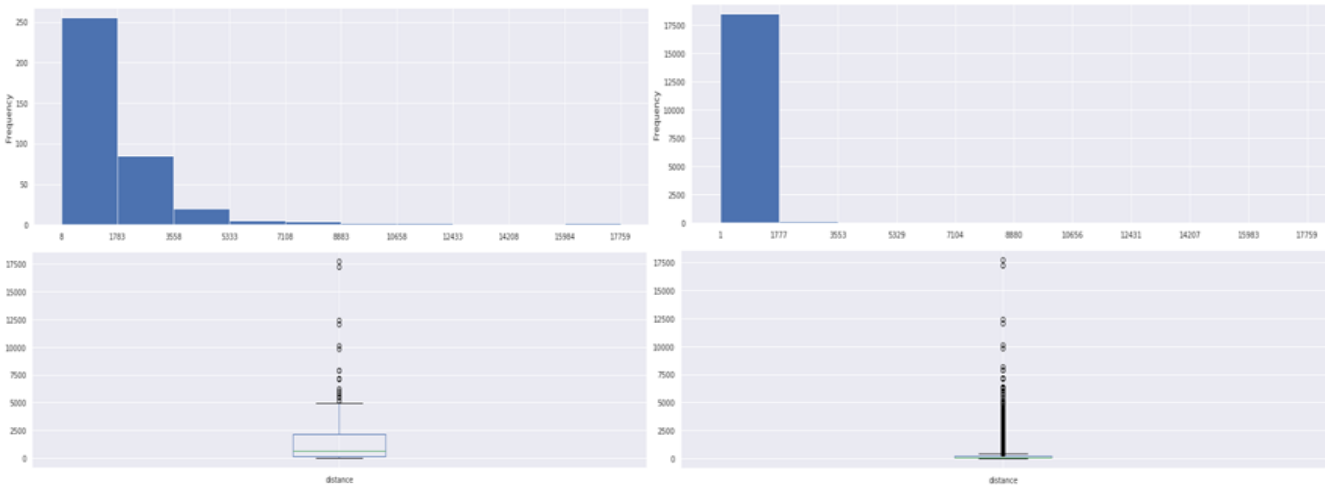


Figure 2.10 Histogram and boxplot for streets lengths after and before cleaning

Comment: As we see the cleaning process clears most of the outliers and make a fine one-sided Gaussian distribution for the street's lengths, and increases the mean to become more expressive dataset with a higher variety and larger overall entropy coefficient.

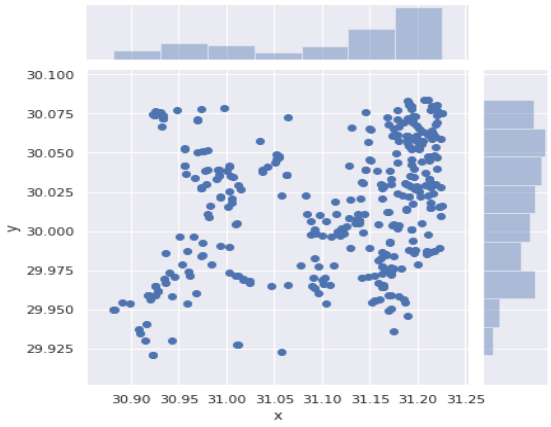


Figure 2.11 Streets distribution over the area of Giza

2.5 API service providers:

2.5.1 Google Places API:

Provides information about the available facilities in the area, opening and closing times. Like schools, malls, universities, shops, government services … etc.

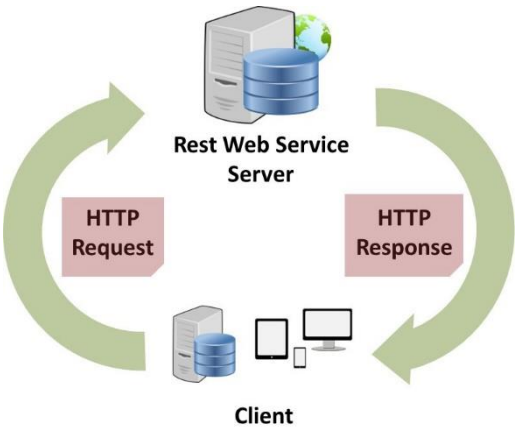
2.5.2 TomTom Traffic services (Flow, Incidents):

- TomTom's API provides us with very important traffic information.
- Free service includes 2500 requests/day.
- Is updated every minute with very latest traffic speed information.
- Is based on the zooming level of different road categories that are displayed.
- Provides traffic speed information with an option to use the absolute or relative speed information.
- Returns detailed information about traffic speed, like Current speed, Free flow speed, quality indicator.
- Is updated every minute with the very latest traffic incident and delay information.

- Returns detailed information about traffic jams and traffic-related incidents. Details include start-location, end-location, road-name, type of delay, length (in time) of the delay, significance, and distance.

2.5.3 Open Weather Maps:

- Provides current real-time weather data, which is frequently updated based on global models and data from more than 40,000 weather stations.
- Free service includes 60 requests/hour.



Now after we had our information sources, we need to get all Giza streets. And here came the OSM website "OpenStreetMap". OSM provides all information about the streets, like longitude, latitude, shops, facilities, street links and nodes, type of street, street name, one-way Boolean, lanes, and a lot more. OSM also gives each street a unique ID called osm id.

OSM lets you choose an area manually by drawing a boundary box around the area you want to get street information from. After that, you can export the osm file for this area.

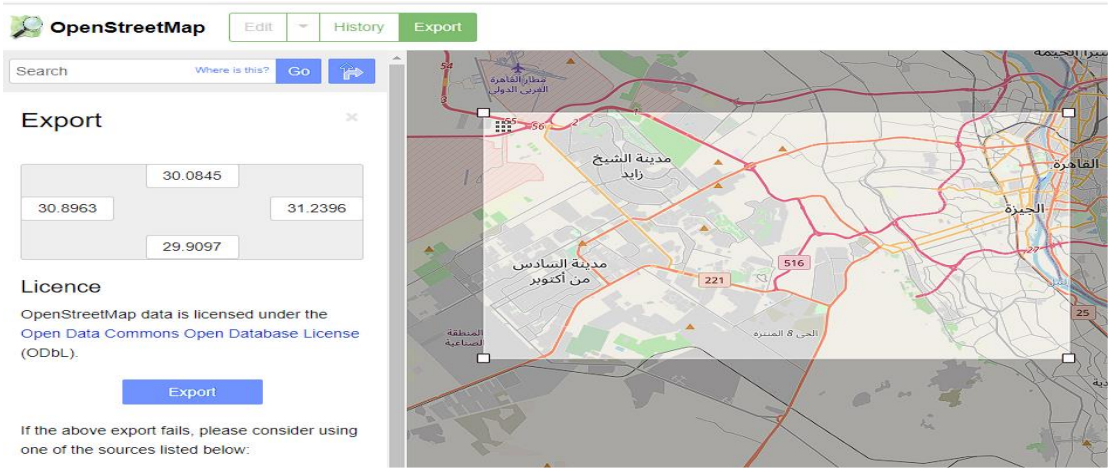


Figure 2.12 Choosing custom area from OpenStreetMap website

2.6 Data Preprocessing and Collecting

After we were finally able to choose 1000 streets from the original 102000 streets, now we can preprocess those streets to prepare them for the collecting phase.

Before preceding to the preprocessing part, we need first to discuss what kind of data we want to collect, after some intensive searching we found out there are two types of reasons that will lead to congestion and they are stochastic reasons or we can call it user effect, this part depends on the user behavior itself, and patterned reasons, they are four main predictable parameters that can affect the traffic state:

- Facilities.
- Incidents.
- Weather.
- Events.
- And our label speed & distance.

So, we won't be able to predict these stochastic effective reasons but we will be able to understand the other four.

Now we are in the phase of searching from where we can get this type of data for our 1000 streets and track it by some rate at a day, and after that search, we settle with these sources and their limitations:

- Google Places API: 300-dollar free requests for each month
- TomTom Traffic Incidents API: 2500 request free per day
- TomTom Traffic Flow API: 2500 request free per day
- Open Weather Maps: 30 requests per minute, but with update state each one hour
- Open Streets Map: Open source for maps and its features metadata
- Web Scrapping: Unlimited

Now we can collect our data and the label with some rate at a day.

2.6.1 Clusters Level Preprocessing

Out of the five datasets, we need to collect there are two of them need to work in a cluster level not the streets level as they providers need the request to contain a cluster of data, not a specific street, so now we need to cluster the streets we see in **Figure2.13** to be able to fit for the request needs.

The first cluster-based dataset is Incidents dataset collected by TomTom Incidents API, that it needs a rectangle shape to express the streets incident state inside it, we can see in Figure2.12 that we used K-means clustering algorithm to be able to create those cluster and we choose them to be 10 clusters as this number contain the best number of streets per each and also not over-focused as that will make the requests a lot and useless.

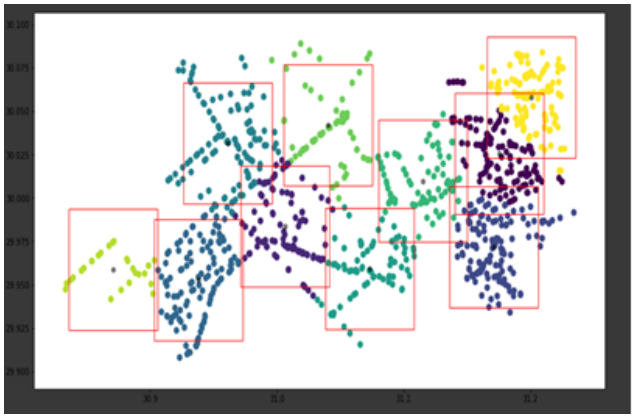


Figure 2.13 Incident cluster level

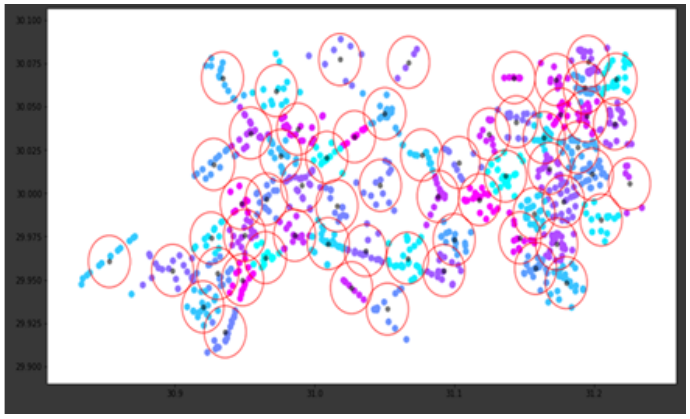


Figure 2.14 Places cluster level

The second cluster-based dataset is places cluster it consists of 60 clusters clustered by K-means, with circle shape with a 1.5 km radius collected by Google Maps API.

2.6.2 Street Level Preprocessing

The Other two main files which are speed & distance that contain the label of our dataset, and the second file is weather dataset, and they both all they need is the center point from the street which can be collected from OSM data we already gathered.

Note: Events are ignored due to Covid-19

2.6.3 Data Collecting Rates

In this phase we were trying to estimate and decide the rate for collecting each dataset file from the four above files, in this area we were squeezed by two limits, upper limit of the maximum number of requests we can make in one day, and the lower limit is the phenomena rate itself as each file has its changing rate, for example, the weathering rate can be tens of minutes as the weather cannot change faster than that and catching faster rates will be useless.

So, we decided we will go for this rate:

- Google Places API every 3 hours for **all clusters.**
- TomTom: Traffic Incidents API every 20 minutes for **all clusters.**
- Open Weather Map every hour for **each street.**
- Web scraping for events periodically **every week.**
- TomTom: Traffic flow API every 20 minutes for **each street.**

By that, we make sure we won't miss any details, and also, we won't exceed the maximum number of requests per day.

After deciding and hard-coding those API routines we now can upload our data to a server that will run it for days and hours without missing any time or dates,

We use PythonAnyWhere.com as it's only cost us 5$ per account and we reserve three accounts to make sure every code runs without any problems, and so on we collected 1 million and 100 thousand samples by the duration of 3 months.

Chapter 3

Deep Learning Introduction

Deep Learning is the building block of any AI system, as it will decide what the machine will virtually-understand and be able to process, just like the human, his brain is the main source of actions, thoughts and decision making.

3.1 Background and techniques:

3.1.1 Neural network

Training a neural network revolves around the following:

- Layers, which are combined into a network (or model).
- The input data and corresponding targets.
- The loss function, which defines the feedback signal used for learning.
- The optimizer, which determines how learning proceeds.

The network, composed of layers that are chained together, maps the input data to predictions. The loss function then compares these predictions to the targets, producing a loss value: a measure of how well the network's predictions match what was expected. The optimizer uses this loss value to update the network's weights.

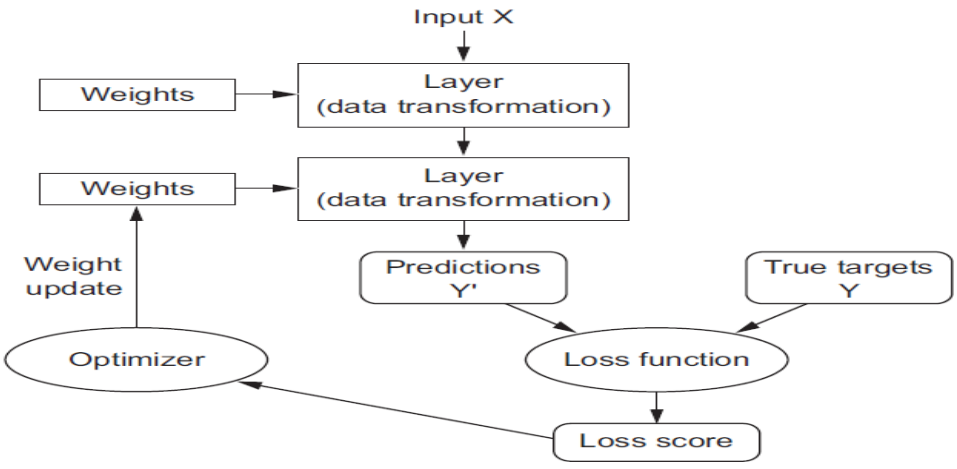


Figure 3.1 Relationship between the network, layers, loss function, and optimizer

A layer is a data-processing module that takes as input one or more tensors and that outputs one or more tensors. Some layers are stateless, but more frequently layers have a state: the layer's weights, one or several tensors learned with stochastic gradient descent, which together contains the network's knowledge.

Once the network architecture is defined, two more things:

- Loss function (objective function)—The quantity that will be minimized during
  training. It represents a measure of success for the task at hand.
- Optimizer—Determines how the network will be updated based on the loss function.

It implements a specific variant of stochastic gradient descent (SGD).

3.1.2 Convolutional neural network

This chapter introduces convolutional neural networks, also known as conv-nets, a type of deep-learning model almost universally used in computer vision applications. It is most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series. CNNs are regularized versions of multilayer perceptron. Multilayer perceptron usually means fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function.

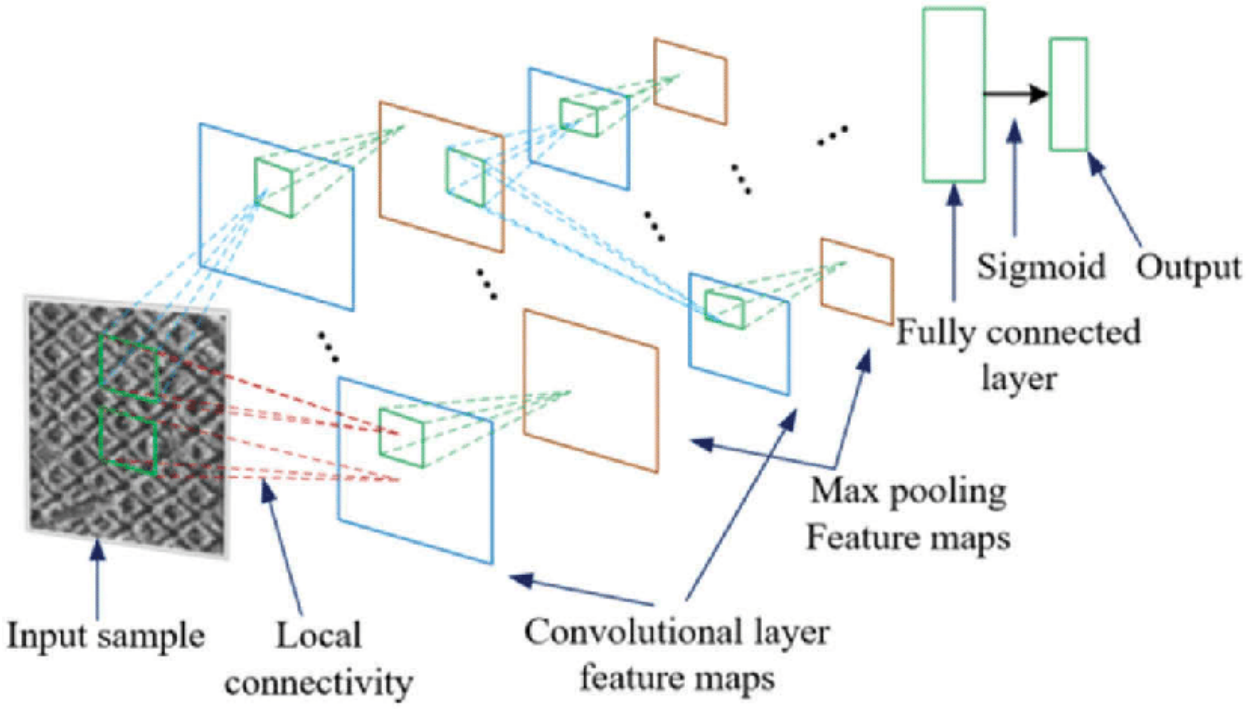


Figure 3.2 CNN architecture

It consists of:

- A convolutional layer that extracts features from a source image.
- A pooling layer that down-samples each feature to reduce its dimensionality and focus on the most important elements.
- A fully connected layer that flattens the features identified in the previous layers into a vector, and predicts probabilities that the image belongs to each one of several possible labels.

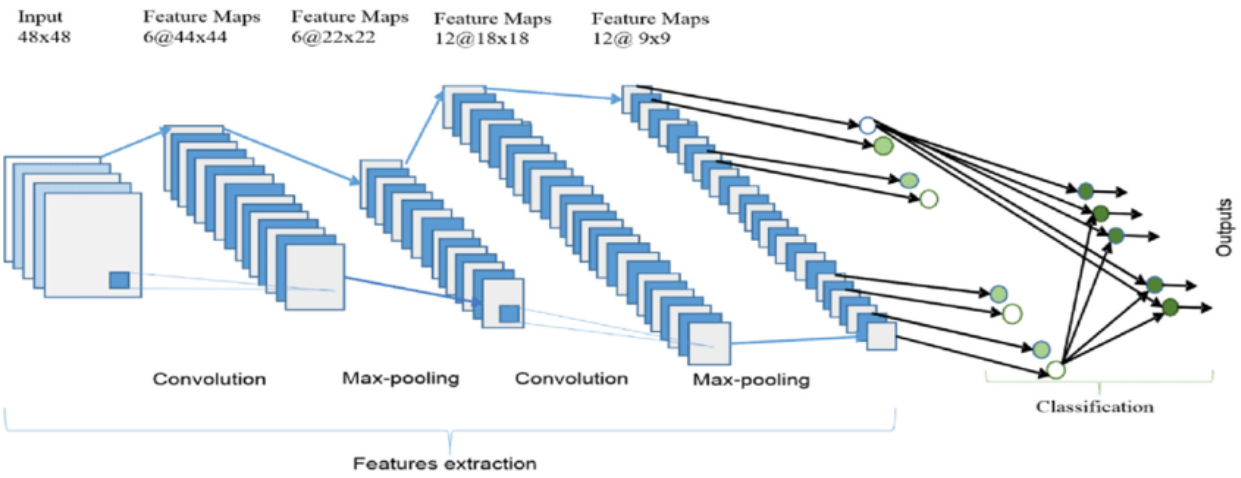


Figure 3.3 CNN layers

Conv2d Explained

A basic conv-net looks like. It's a stack of Conv2D and MaxPooling2D layers. 2D convolutional layers take a three-dimensional input, typically an image with three color channels. They pass a filter, also called a convolution kernel, over the image, inspecting a small window of pixels at a time, for example, 3×3 or 5×5 pixels in size, and moving the window until they have scanned the entire image. The convolution operation calculates the dot product of the pixel values in the current filter window with the weights defined in the filter.

Convolutions operate over 3D tensors, called feature maps, with two spatial axes (height and width) as well as a depth axis (also called the channels axis). For an RGB image, the dimension of the depth axis is 3, because the image has three color channels: red, green, and blue. For a black-and-white picture, like the MNIST digits, the depth is 1 (levels of gray). The convolution operation extracts patches from its input feature map and applies the same transformation to all of these patches, producing an output feature map.

This output feature map is still a 3D tensor: it has a width and a height. Its depth can be arbitrary because the output depth is a parameter of the layer, different channels in that depth axis no longer stand for specific colors as in RGB input; rather, they stand for filters. Filters encode specific aspects of the input data: at a high level.

A 2D convolution layer means that the input of the convolution operation is three-dimensional, for example, a color image that has a value for each pixel across three layers: red, blue, and green. However, it is called a "2D convolution" because the movement of the filter across the image happens in two dimensions. The filter is run across the image three times, once for each of the three layers.

After the convolution ends, the features are down-sampled, and then the same convolutional structure repeats. At first, the convolution identifies features in the original image (for example in a cat, the body, legs, tail, head), then it identifies sub-features within smaller parts of the image (for example, within the head, the ears, whiskers, eyes). Eventually, this process is meant to identify the essential features

that can help classify the image. Learn more in our guide to Convolutional Neural Networks (CNN).

In Conv2D layers, padding is configurable via the padding argument, which takes two values: "valid", which means no padding (only valid window locations will be used); and "same", which means “pad in such a way as to have an output with the same width and height as the input.” The padding argument defaults to "valid".

Convolution Stride Explained

The other factor that can influence output size is the notion of strides. The description of convolution so far has assumed that the center tiles of the convolution windows are all contiguous. But the distance between two successive windows is a parameter of the convolution, called its stride, which defaults to 1. It’s possible to have stride convolutions: convolutions with a stride higher than 1. In figure 5.7, you can see the patches extracted by a 3 × 3 convolution with stride 2 over a 5 × 5 input (without padding).

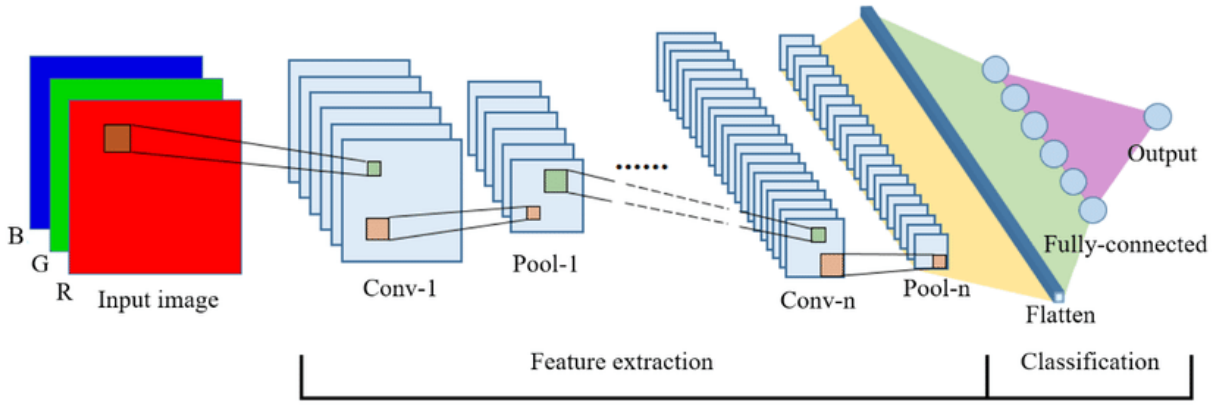


Figure 3.4 2D-convnet

3.1.3 Recurrent Neural Network

A major characteristic of all neural networks densely connected networks and convnets, is that they have no memory. Each input shown to them is processed independently, with no state kept in between inputs. With such networks, to process a sequence or a temporal series of data points, must show the entire sequence to the network at once: turn it into a single data point which is not practical.

A recurrent neural network (RNN) processes sequences by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far. In effect, an RNN is a type of

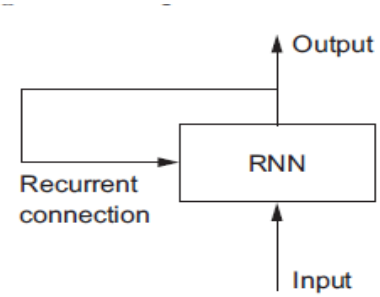


Figure3.5 RNN structure

neural network that has an internal loop the state of the RNN is reset between processing two different, independent sequences, so it still considers one sequence a single data point: a single input to the network. What changes is that this data point is no longer processed in a single step; rather, the network internally loops over sequence elements.

Easy enough: in summary, an RNN is a for loop that reuses quantities computed

during the previous iteration of the loop, nothing more. RNNs are characterized by their step function, such as the following function in this case:

output-t = np.tanh(np.dot(W, input-t) + np.dot(U, state-t) + b)

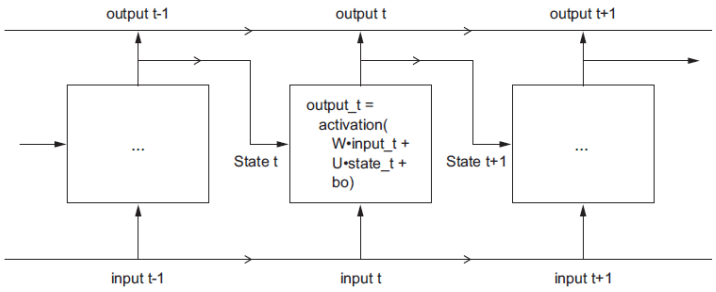


Figure 3.6 RNN states

Each time step t in the output tensor contains information about time steps 0 to t in the input sequence—about the entire past that's why no need for this full sequence of outputs; you just need the last output (output_t at the end of the loop) because it already contains information about the entire sequence.

3.1.4 LSTM

Simple RNN is generally too simplistic to be of real use. Simple RNN has a major issue: although it should theoretically be able to retain at time t information about inputs seen many time steps before, in practice, such long-term dependencies are impossible to learn. This is due to the vanishing gradient problem, an effect that is similar to what is observed with non-recurrent networks (feedforward networks)

that are many layers deep: by keep adding layers to a network, the network eventually becomes untrainable. The theoretical reasons for this effect were studied by Hochreiter, Schmidhuber, and Bengio in the early 1990s.2 The LSTM is designed to solve this problem.

Long Short-Term Memory (LSTM) algorithm was developed by Hochreiter and Schmidhuber in 1997;3 it was the culmination of their research on the vanishing gradient problem. as; it adds a way to carry information across many time steps. it saves information for later, thus preventing older signals from gradually vanishing during processing.

An additional data flow that carries information across time steps. Call its values at different time steps Ct, where C stands for carrying. This information will have the following impact on the cell: it will be combined with the input connection and the recurrent connection (via a dense transformation: a dot product with a weight matrix followed by a bias add and the application of an activation function), and it will affect the state being sent to the next time step (via an activation function a multiplication operation). Conceptually, the carry dataflow is a way to modulate the next output and the next state.

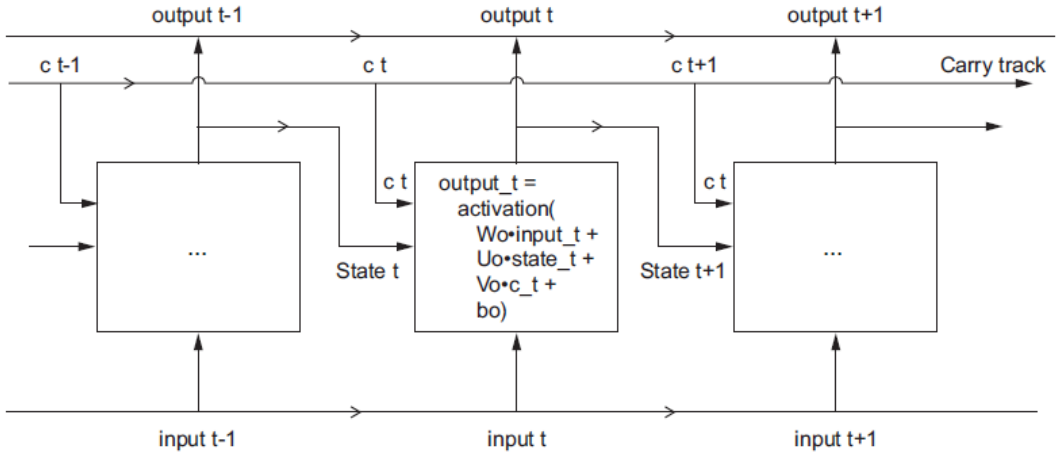


Figure 3.7 output states

More explanation of what each of these operations is meant to do. multiplying c_t and f_t is a way to deliberately forget irrelevant information in the carry dataflow. Meanwhile, i_t and k_t provides information about the present, updating the carry track with new information. But at the end of the day, these interpretations don’t

mean much, because what these operations do is determined by the contents of the weights parameterizing them; and the weights are learned in an end-to-end fashion, starting over with each training round, making it impossible to credit this or that operation with a specific purpose. The specification of an RNN cell (as just described) determines the hypothesis space but it doesn't determine what the cell does; that is up to the cell weights.

The same cell with different weights can be doing very different things. So, the combination of operations making up an RNN cell is better interpreted as a set of search constraints, not as a design in an engineering sense.

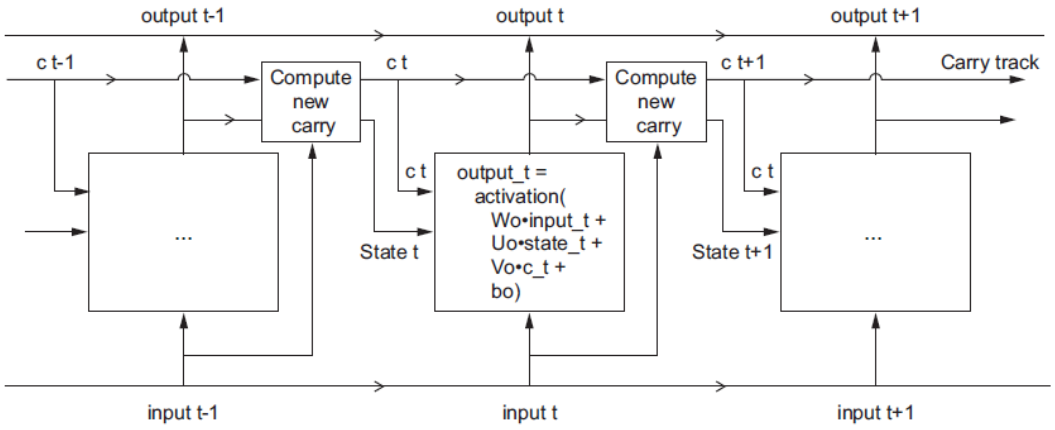


Figure 3.8 anatomy of LSTM

3.2 Optimizing Techniques and Tuning:

3.2.1 Over-sampling and Under-sampling

Both over-sampling and under-sampling involve introducing a bias to select more samples from one class than from another, to compensate for an imbalance that is either already present in the data, or likely to develop if a purely random sample were taken.

Imbalanced Data: A classification data set with skewed class proportions is called imbalanced. Classes that make up a large proportion of the data set are called majority classes. Those that make up a smaller proportion are minority classes.

Degree of imbalance	The proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

Down-sampling and Upweighting

An effective way to handle imbalanced data is to down-sample and up to weight the majority class.

Down-sampling: training on a disproportionately low subset of the majority class examples.

Upweighting: adding an example weight to the down-sampled class equal to the factor by which you down-sampled.

Step 1: Down-sample the majority class. Consider again our example of the fraud data set, with 1 positive to 200 negatives. We can down-sample by a factor of 20, taking 1/10 negatives. Now about 10% of our data is positive, which will be much better for training our model.

Step 2: Up weight the down-sampled class: The last step is to add example weights to the down-sampled class. Since we down-sampled by a factor of 20, the example weight should be 20.

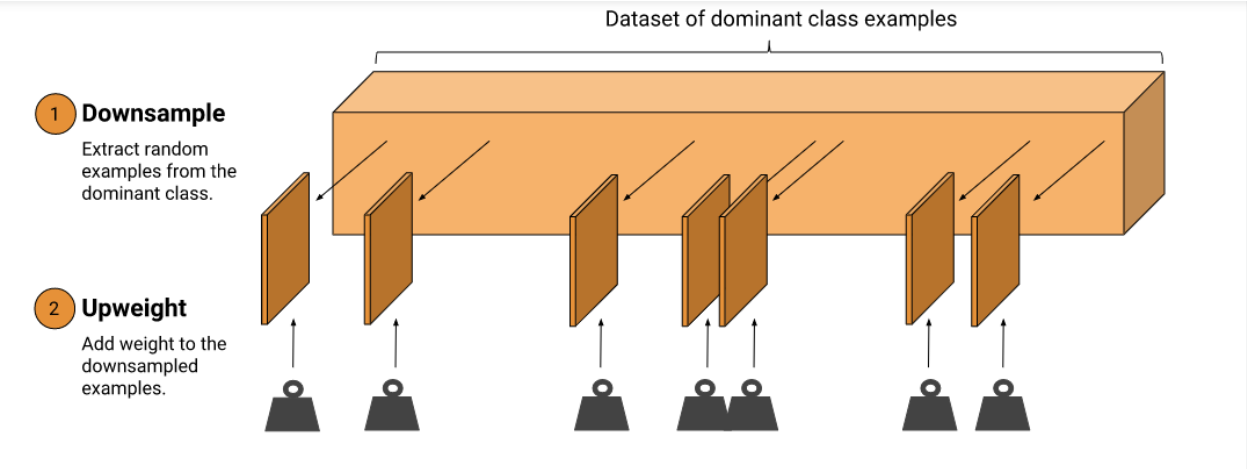


Figure 3.9 counting an individual example more importantly

Example weights, which means counting an individual example more importantly during training. An example weight of 10 means the model treats the example as 10 times as important (when computing loss) as it would an example of weight 1.

The weight should be equal to the factor you used to down-sample:

{example weight} = {original example weight} × {down-sampling factor}

Why Down-sample and Upweight?

It may seem odd to add example weights after down-sampling. We were trying to make our model improve on the minority class -- why would we up weight the majority? These are the resulting changes:

- Faster convergence: During training, we see the minority class more often, which will help the model converge faster.
- Disk space: By consolidating the majority class into fewer examples with larger weights, we spend less disk space storing them. This savings allows more disk space for the minority class, so we can collect a greater number and a wider range of examples from that class.
- Calibration: Upweighting ensures our model is still calibrated; the outputs can still be interpreted as probabilities.
- positive and false-negative results.

For these reasons, one will typically cleanse only as much data as is needed to answer a question with reasonable statistical confidence (see Sample Size), but not more than that.

Normalization

The goal of normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model.

Four common normalization techniques may be useful:

- scaling to a range
- clipping
- log scaling
- z-score

The following charts show the effect of each normalization technique on the distribution of the raw feature (price) on the left. The charts are based on the data set from 1985 Ward's Automotive Yearbook that is part of the UCI Machine Learning Repository under Automobile Data Set.

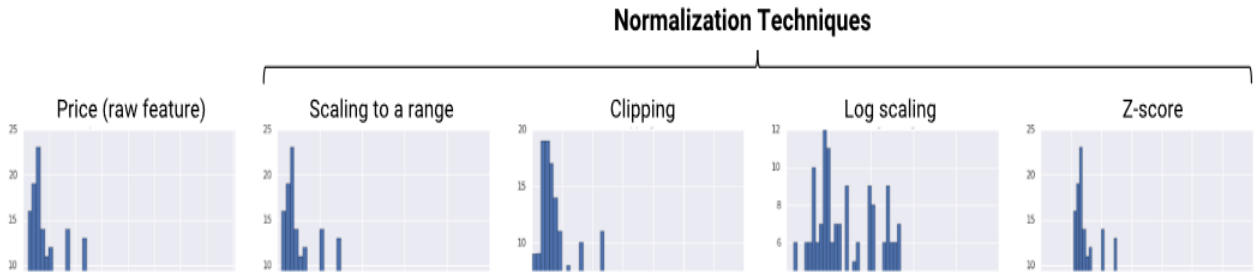


Figure 3.10 Normalization Tech

Scaling

Scaling means converting floating-point feature values from their natural range (for example, 100 to 900) into a standard range—usually 0 and 1 (or sometimes -1 to +1). Use the following simple formula to scale to a range:

$$x' = (x - x\text{-min}) / (x\text{-max} - x\text{-min}) \qquad \text{equ 3.1}$$

Scaling to a range is a good choice when both of the following conditions are met:

- know an approximate upper and lower bound on your data with few or no outliers.
- data is approximately uniformly distributed across that range.

A good example is an age. Most age values fall between 0 and 90, and every part of the range has a substantial number of people.

In contrast, scaling on income, because only a few people have very high incomes. The upper bound of the linear scale for income would be very high, and most people would be squeezed into a small part of the scale.

Feature Clipping

If your data set contains extreme outliers, try feature clipping, which caps all feature values above (or below) a certain value to fixed value. For example, clip all temperature values above 40 to be exactly 40.

You may apply feature clipping before or after other normalizations.

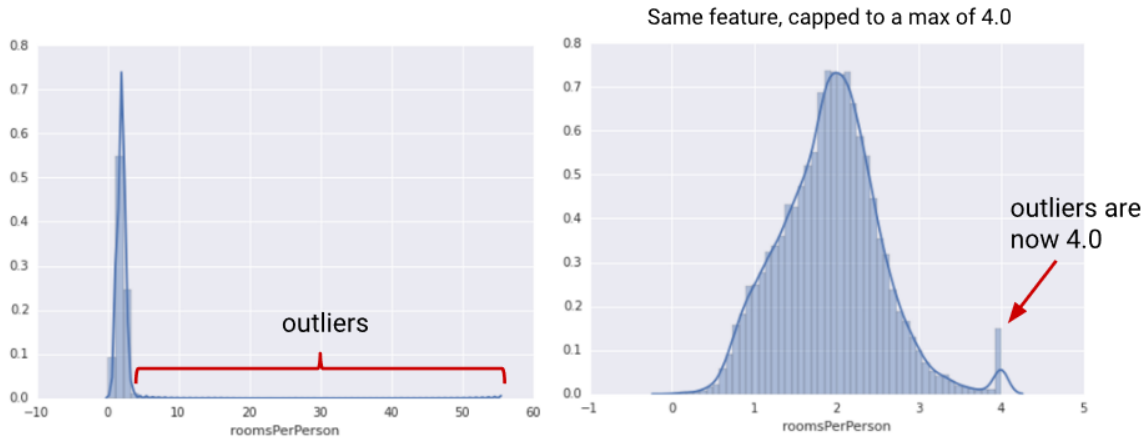


Figure 3.11 comparing a raw distribution to its clipped version

Another simple clipping strategy is to clip by z-score to +-Nσ (for example, limit to +-3σ). Note that σ is the standard deviation.

Log Scaling

Log scaling computes the log of values to compress a wide range to a narrow range.

$$x' = \log(x) \qquad \text{equ 3.2}$$

Log scaling is helpful when handful values have many points, while most other values have few points. This data distribution is known as the power-law distribution. Movie ratings are a good example. In the chart below, most movies have very few

ratings (the data in the tail), while a few have lots of ratings (the data in the head). Log scaling changes the distribution, helping to improve linear model performance.

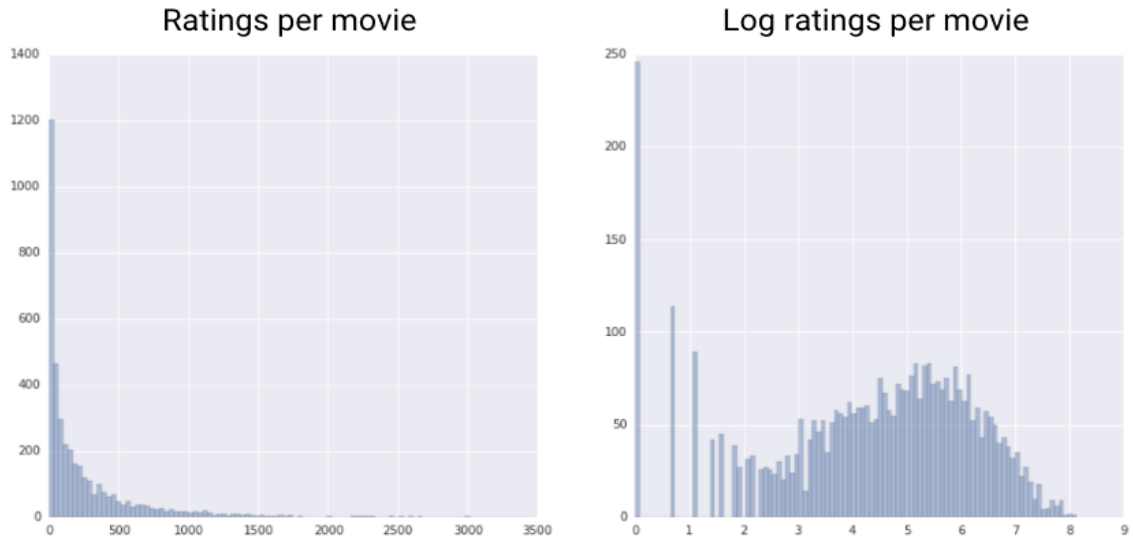


Figure 3.12 comparing a raw distribution to its log

Z-score is a variation of scaling that represents the number of standard deviations away from the mean. use z-score to ensure that the feature distributions have mean = 0 and std = 1. It's useful when there are a few outliers, but not so extreme that you need clipping.

The formula for calculating the z-score of a point, x, is as follows:

$$x' = (x - \mu)/\sigma \qquad \text{equ 3.3}$$

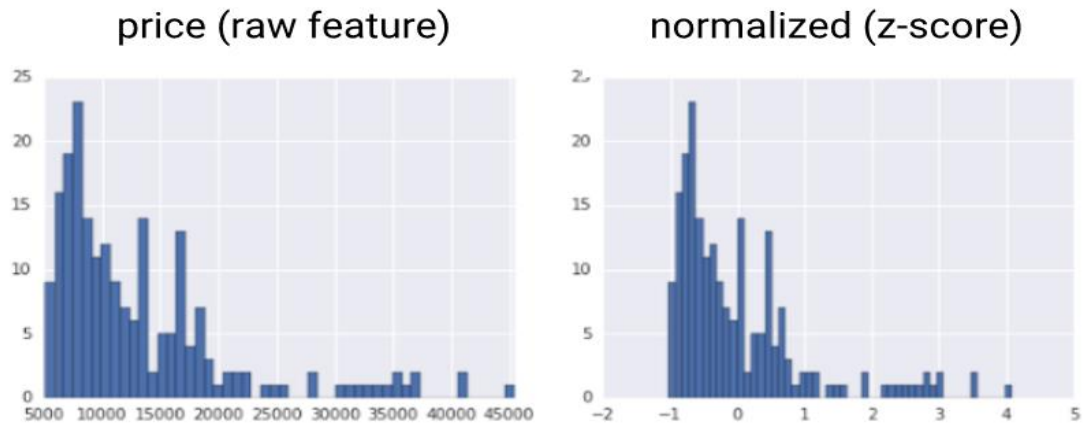


Figure 3.13 comparing a raw distribution to its z-score distribution

Note: μ is the mean and σ is the standard deviation.

Chapter 4

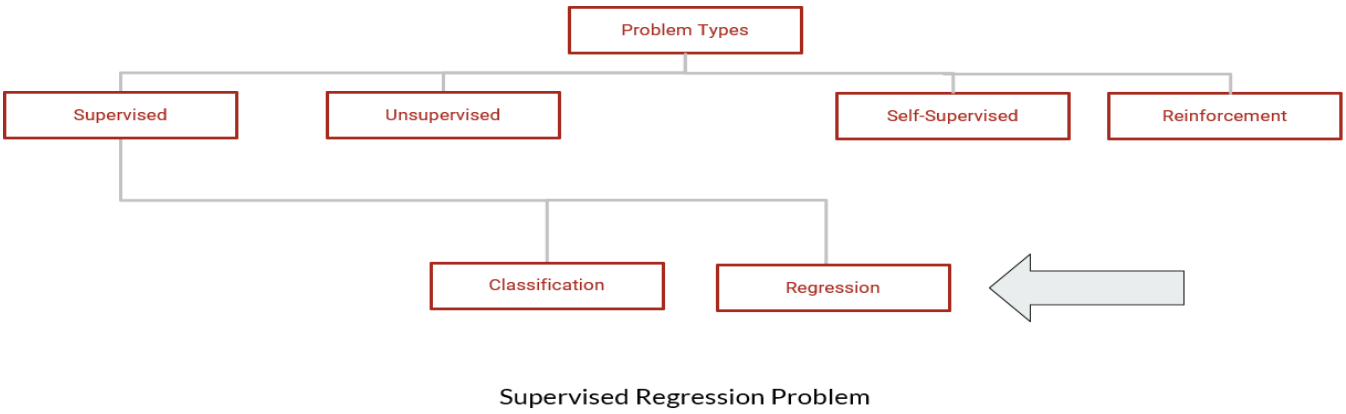
Deep Learning-First Approach

"When an algorithm or neural network inherits the flaws of its human creator."

- Clyde DeSouza

In our project after the last chapter underlines from the process of collecting data and processing it, the process of building models be able to understand and extract features from this data, so then it will be able to process and manipulate new data to help to predict new labels that will help us avoiding traffic congestion.

Before starting any progress, we have made in this area we need first to study our problem from the machine learning point of view, in **Graph3.1** the problem types in the prospection of machine learning.



Graph 4.1 Problem categorization

Supervised: The dataset used in the learning process has labels on it to identify if the model is good enough or not

Regression: The task of the model is to follow some variable that we do not know its function or its definite parameters it depends on.

After categorizing the problem we need to dive together in the model building process and how we cooperate to conquer this hard-deep learning problem.

As our problem has no former solutions, we need to cover a wider area in the model building strategy and try to use a large number of models and combinations with a state of art techniques, to achieve this aim, we will

Split into two teams, each one has its own unique and different criteria and then we can reach the best solution in our problem.

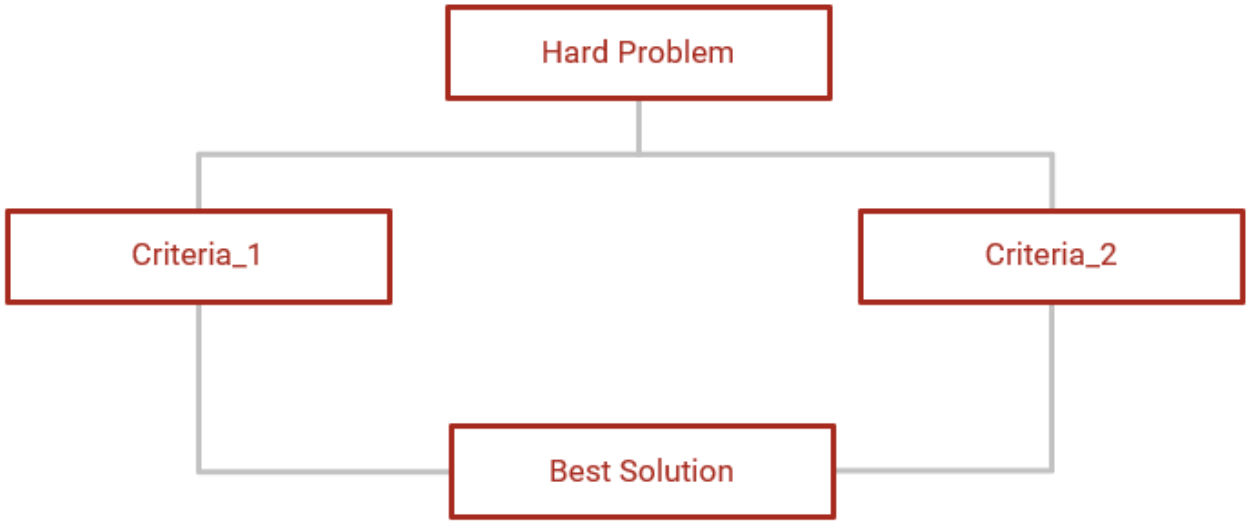


Figure4.2 Divide and conquer

4.1 System Preparing

4.1.1 Criteria

- Work with all streets.
- Intense feature extraction methods.
- Deep networks with large parameter number.
- Time series-based data.

By this criterion, we will be able to proceed in the deep learning task and implement our solution.

4.1.2 Data Preprocess

In this problem we have 4 main datasets we already discussed before they are:

1. Incidents
2. Speed and distance
3. Weather
4. Places

And we add five more data files to provide criteria two that heavy feature extracting, and they are:

1. Houses
2. Amenities
3. Buildings
4. Intersections
5. Shops

Those five files contain static data about the streets, describe the number of facilities, buildings and houses in a small cluster contain this street, and also provide data describing how much this street is important by the number of intersections made by it and the types of those intersections.

4.2 Data Preprocessing

After preparing the nine files we then will intensively extract the features in it, to create a 187 feature raw data, consists of dynamic and static parts.

And then to create a time series we will take the last three samples and stack them together and concatenate the output with the static part, to be able to predict the current sample.

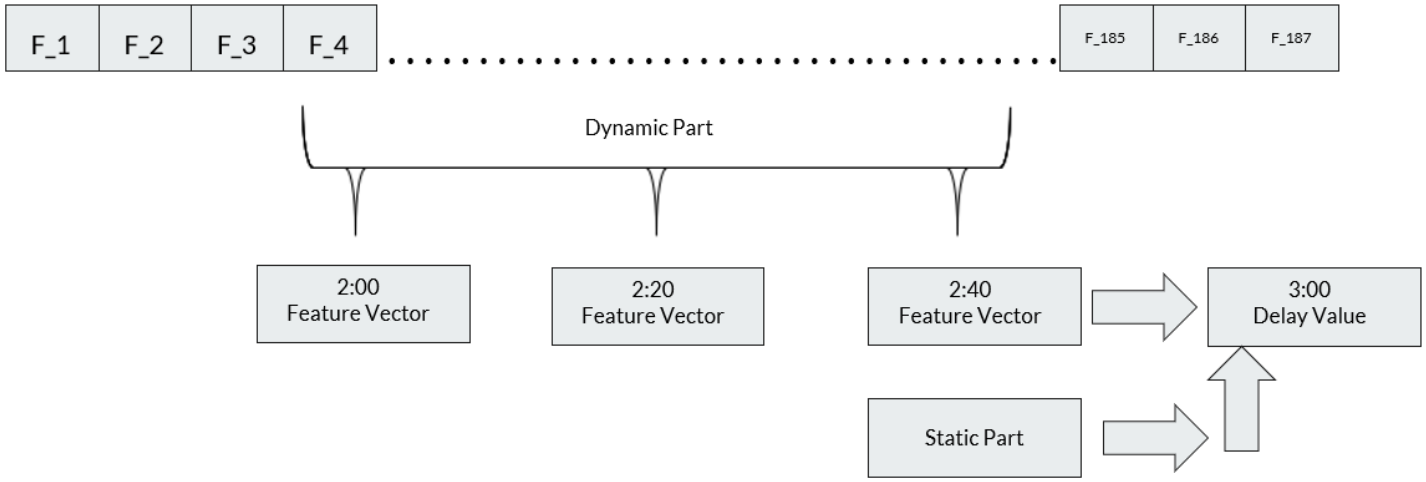


Figure 3.3 data preprocess

4.3 Model building process

4.3.1 Trial-1

As we already stated that we will work with a time series data and see previously how we create it, so in our first trial it's obvious that we will lean on RNN model,

As the most famous RNN layer is LSTM we will build our first model with it, as in Figure 3.4 we can see that the model consists of a dynamic part analyses and static concatenated with it.

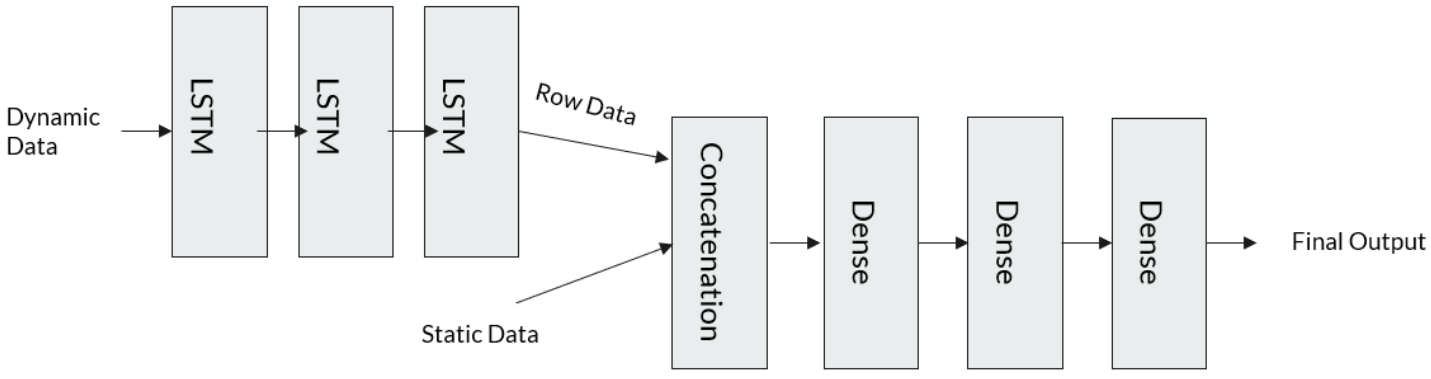


Figure 3.4 LSTM model-based

Problems in Trial-1

Problem	Solution
Not able to calculate MAE metrics	Min-Max Normalization
Not able to train	Downsampling

After building the model we tried to train it on 1 million samples, but the model was calculating a nan MAE value and that was because the different range of the input features, so we used min-max normalization to normalize the samples values and modify the range such that it will be between 0-1 so we can kill that problem.

Another problem that appears after the model was able to learn is that the MAE does not decrease, and that was because the labels were unbalanced as 90.12% is zeros so the model was not able to train, to solve this problem we used Down-sampling technique to modify the data distribution.

After all of that, the model was able to learn and after some SGD fine-tuning, we were able to get

Test Error = 3-min

Train Error = 4.1-min

Trial Observation

“LSTM Network is not good enough alone to process and train the model with this data”

4.3.2 Trial-2

After trying LSTM, we saw that another simpler and more trusted network will be better, so we go to CNN to provide a more common network and easier to observe and understand.

But now we need to prepare each time series, so that it can be feed to convolution layer, to solve this dilemma we use the simplest way, by reshaping each time series sample so that it will represent a 187x3x1 image that can be feed into Conv2D layer, as in figure 3.5

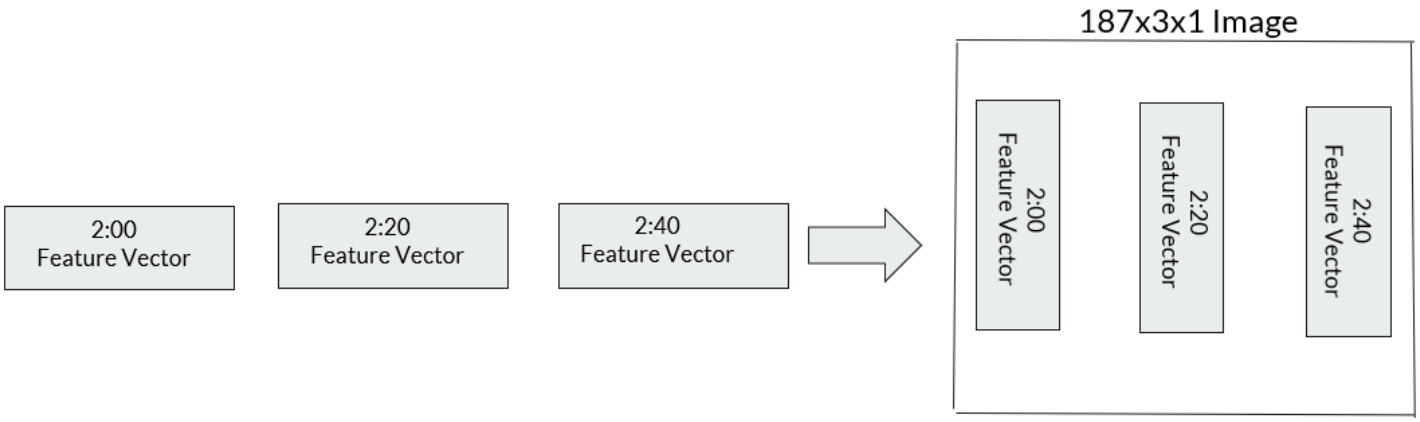


Figure 3.5 Data preparation for Convolution network

Now we can build our model as in figure 3.6 we can see that we used CNN for process dynamic part of the data and then concatenate with the static part.

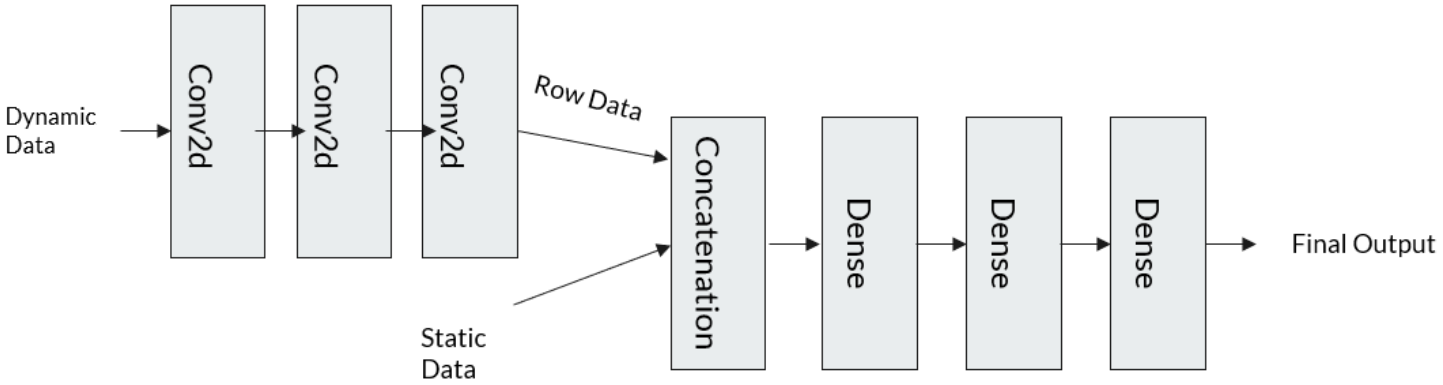


Figure 3.6 Conv-based model

And after some SGD fine-tuning, we can prove our hypothesis that conv models will behave better than RNN models in our dataset, By training and testing error much lower.

Test Error = 92-sec

Train Error = 130-sec

Trial Observation

"CNN is good enough as a feature extractor to process and train the model with this data"

Trial-3

In a process of building a state of art model, we were in the right path of trying and error so we decide to build a hyper model that will include Conv and LSTM layers both.

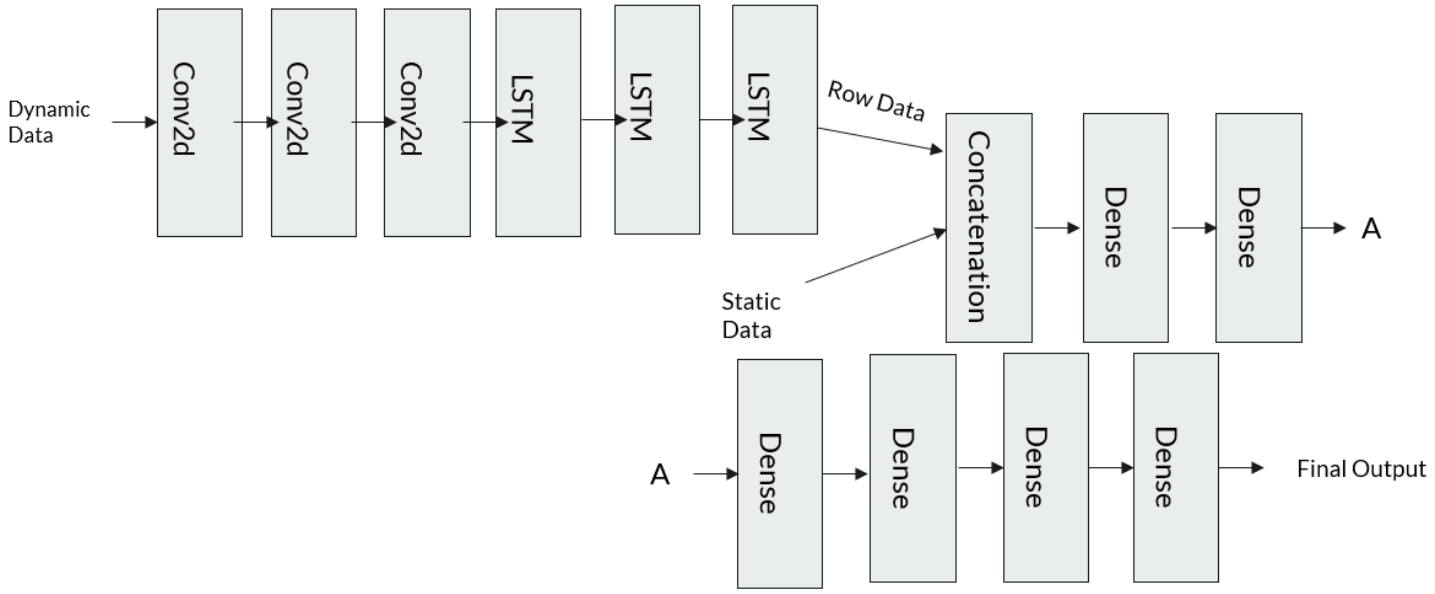


Figure 3.7 Hyper model components

Test Error = 62-Sec

Train Error = 58-sec

Chapter 5

Deep Learning-Second Approach

Deep Learning techniques are used based on the problem, where the most basic type of deep learning techniques is Vanilla Neural Network (VNN) or known as the multi-layer perceptron. VNN is used in the basic machine learning problems where there is no time dependency in the data or no spatial features needed to be detected. The second type of neural networks used is Convolutional Neural Networks (CNN), where CNN is used in detecting spatial features in the data e.g. In a cat vs dog classification problem, it doesn't matter where the ears of the cat are at in the picture, the important thing is to detect the cat ears anywhere in the picture, VNN is sensitive to the ears place while CNN is not.

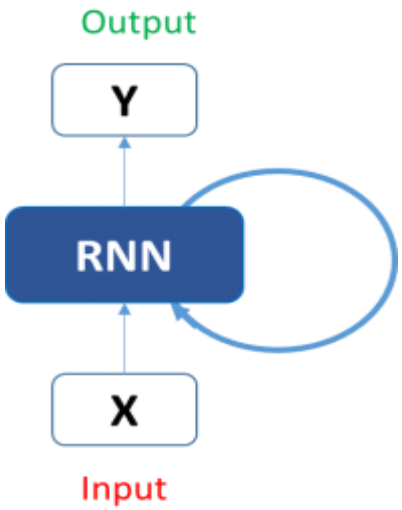


Figure 5.1 RNN unit

The third type is Recurrent Neural Networks (RNN) in figure 5.1, RNN is used when the current output is dependent on the previous output and the current input. RNN is used in many fields such as text mining, speech recognition, time series problems, and much more.

5.1 Problem Description

In TrCPRS it's required to predict the next time step delay based on multi-features input. This problem is considered a real-time prediction. In real-time problems, the past time steps are crucial in predicting the next time step.

5.2 Solution approach

RNN is used to solve time series problems as it uses its memory to remember previous states as well as the current state. RNN is expected to get the best results but before using RNN one must try VNN first as it has much lower computational power

than RNN. VNN is used to check the feasibility and whether or not it can be solved using deep learning. After checking the feasibility of the problem, LSTMs will be used which is a special kind of RNN.

5.2.1 LSTM

Long Short-term memory networks were invented in 1997 and set accuracy records in multiple application domains. LSTMS gives better performance as it fights the vanishing gradient problem.

LSTM fights back vanishing gradient problem which gives it the ability to process thousands or millions of back steps with multiple LSTM layers.

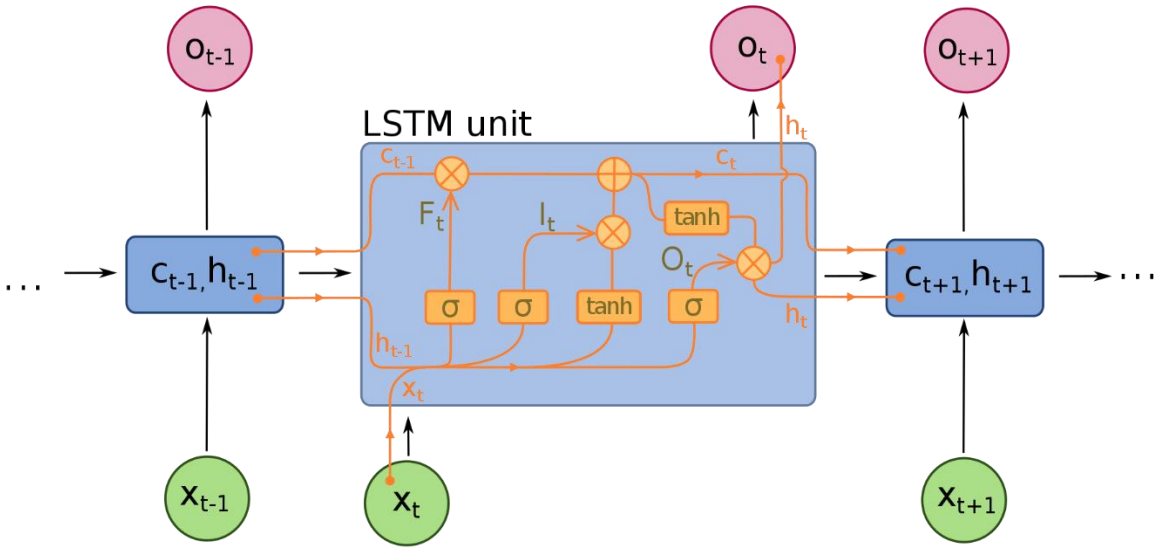


Figure 5.2 LSTM unit

5.3 Pre-Processing

Good data pre-processing is the key to a good model. Pre-processing is done for many reasons:

- Turning all data to numerical data as machine learning can't work with non-numerical data.
- Pre-processing categorical data.
- Normalizing Data.
- Removing outliers.

5.3.1 Turning data to numerical

Machine learning models are all based on numbers and differentiation problems which can't be done using string or any other data rather than numerical data. Any other type of data must be converted to numerical data before starting and must be converted in a certain way to ensure that the information included is not lost. E.g. if the data is 'Good', 'Moderate', and 'Bad' it can be turned easily to 2, 1, and 0 respectively, while if the data is 'Giza', 'Cairo' and 'Alex.' It can't be turned in the same manner as they all share the same weight and none of them is preferable over the other where this data is called categorical data.

5.3.2 Categorical data

As mentioned earlier, categorical data must be dealt with particularly to ensure that non-data loss takes place. Machine learning depends on numbers, if 'Cairo' was given number 2 and 'Alex' is given number 1, the model understands that 'Cairo' is more important where they share the same importance. There are many different ways to deal with categorical data, the one used in the project is called One Hot Encoding method where each data input is considered a one zeros vector with one in the required place. E.g. in processing fruits data as 'Apple', 'Orange' and 'Bananas' and Orange representation is required, it will be represented in the vector [0 1 0] which means it's not apple nor bananas but orange, where Apple and Bananas can be represented as [1 0 0] and [0 0 1] respectively.

5.4 Model Trials

As mentioned earlier, the first step in building machine learning models is to build a VNN. This step ensures that the problem is solvable. In the following section, VNN trials will be listed and discussed.

5.4.1 VNN

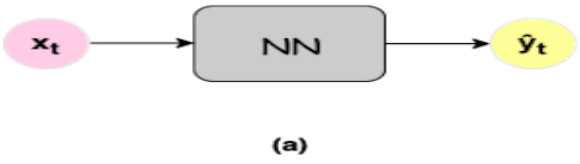


Figure 5.3 VNN unit

VNN or multi-layer perceptron is considered the most basic neural network. In TrCPRS, it's used to check the feasibility of the problem. Figure 5.3 shows the architecture of VNN. Through different VNN trials, a brief description of the input will be discussed.

5.4.1.1 VNN-1

While constructing a model, it's important to recognize that it resembles the human mind. If a human start thinking about understanding a cluster level, one might think of it as a group of streets where incidents occur repeatedly more than once, if a cluster was watched over for a long time, the number of streets where incidents occur will converge. E.g. if a cluster has 10 streets in total, you can think of it that almost 3 streets have repeated incidents. Watching the 10 streets in a long interval will eventually get you maybe 6 streets where incidents occur.

In this way, the cluster level data dependents on how unique streets affect their cluster over some time.

Pros:

- Model starts learning

Cons:

This model has a serious problem, a feature space explosion! If the cluster data collected has 300 unique streets, there's one hot encoding for each street in the feature space with 300 lengths. All other features follow one hot encoding method which means if there are 4 different causes for incidents and delay magnitudes, there are 4*300 columns feature space for causes and 4*300 columns feature space for delay magnitudes as for each street there must be 4 columns feature space indicating the cause and 4 indicating delay magnitude. Most of this is 0 while 1's is only included for the present street. This problem led to a large error in the model as in machine learning there must be data rows 10x more than data columns for good learning procedure.

5.4.1.2 VNN-2

This model had great improvement, eliminating street names and replacing them by types, i.e. instead of listing all street names, only their types matter. E.g. if a new street in the cluster had an accident, no new columns added as, in the end, it has one of 6 types discussed in the data frame. In this way, the feature space size reduces a lot! Feature space size reduced to 1/10 at least of version 1.

Pros:

- Model Learns
- Average-Good error

Cons:

Due to in-availability of the time step number in the input, the model error doesn't decrease much

5.4.1.3 VNN-Final

In this model, two new data columns were added to street level, specific street id, and time step number. The previous cases where street names where used and caused relatively large errors were due to Incidents Street name at the cluster level. Adding specific street id isn't the same, as there's a constant number of streets in street level but on a cluster level, each day new streets may have new incidents.

This addition increases the size by only the number of unique streets in speed and distance data frame.

Adding the number of time steps within the day helped the model to understand when the congestion occurs on average.

Pros:

- Low error.
- Can generalize to future time steps.

Cons:

- Street-level specific.

VNN-Final Architecture

```
[39] if(train):
       modelt.summary()

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 82230, 32)         9152
_________________________________________________________________
dropout (Dropout)            (None, 82230, 32)         0
_________________________________________________________________
dense_1 (Dense)              (None, 82230, 64)         2112
_________________________________________________________________
dropout_1 (Dropout)          (None, 82230, 64)         0
_________________________________________________________________
dense_2 (Dense)              (None, 82230, 32)         2080
_________________________________________________________________
dropout_2 (Dropout)          (None, 82230, 32)         0
_________________________________________________________________
dense_3 (Dense)              (None, 82230, 16)         528
_________________________________________________________________
dropout_3 (Dropout)          (None, 82230, 16)         0
_________________________________________________________________
dense_4 (Dense)              (None, 82230, 8)          136
_________________________________________________________________
dense_5 (Dense)              (None, 82230, 1)          9
=================================================================
Total params: 14,017
Trainable params: 14,017
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.4 VNN-final architecture

VNN-Final Results

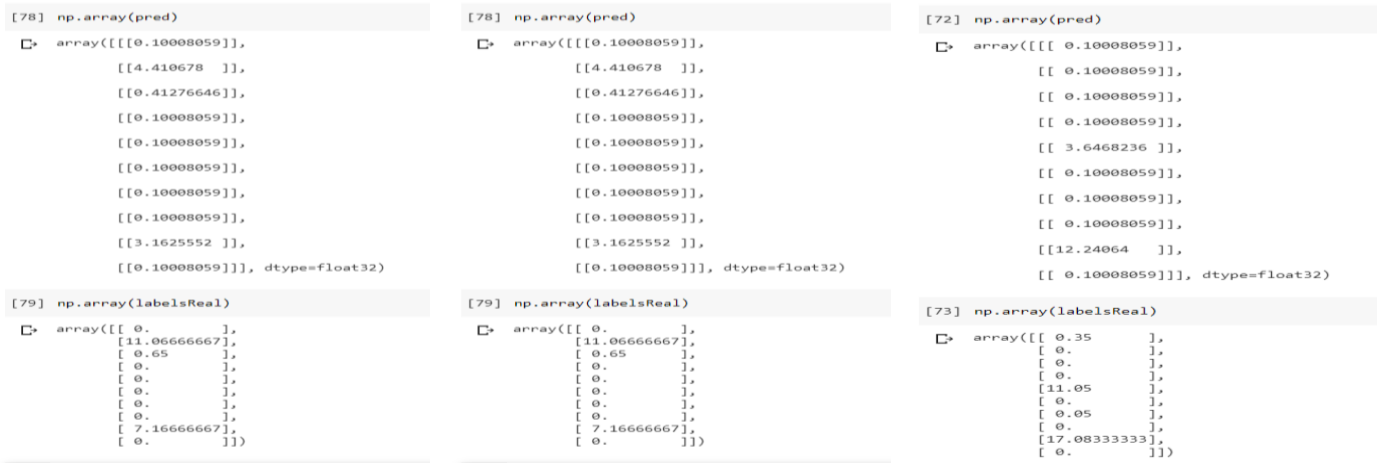


Figure 5.5 VNN-final results

After checking the problem feasibility using VNN, time to use RNN. RNNs are better than VNNs in a real-time application where future time steps depend on the current and previous steps. Each street has a pattern that resembles that in figure 5.5. LSTM tries to catch that pattern and find a relation between other features and the change in the pattern. All street specific data has been removed in both cluster and street levels. The first layer in the model is the masking layer which was discussed briefly in section 5.6 in this chapter.

Pros:

- Low error.
- Non-street specific.
- Can generalize well to future time steps.

Cons:

- Even with low error, better results can be achieved but due to several changes in the data patterns in reflect changes in government curfew times, people commitment and more, data became non-consistent which affected the model greatly.

LSTM Architecture

```
[ ]  if(trainNow):
       modelx.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
masking (Masking)            (None, 3, 137)            0
_________________________________________________________________
lstm (LSTM)                  (None, 3, 32)             21760
_________________________________________________________________
dropout (Dropout)            (None, 3, 32)             0
_________________________________________________________________
lstm_1 (LSTM)                (None, 3, 64)             24832
_________________________________________________________________
dropout_1 (Dropout)          (None, 3, 64)             0
_________________________________________________________________
lstm_2 (LSTM)                (None, 32)                12416
_________________________________________________________________
dense (Dense)                (None, 32)                1056
_________________________________________________________________
dense_1 (Dense)              (None, 16)                528
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17
=================================================================
Total params: 60,609
Trainable params: 60,609
Non-trainable params: 0
```

Figure 5.6 LSTM architecture

Chapter 6

Routing

One of the most common tools for divide traffic costs on all the roads dynamic traffic assignment is a tool used to divide traffic costs o not one route to making less congestion between specific origin and destination.

In a dynamic traffic assignment, each traveler is an agent choosing a route; the joint actions of all travelers result in congestion patterns throughout the network, which determine the travel time each traveler faces. At the equilibrium solution, no traveler can reduce their travel time by switching to another route, so Travel forecasting models are the target.

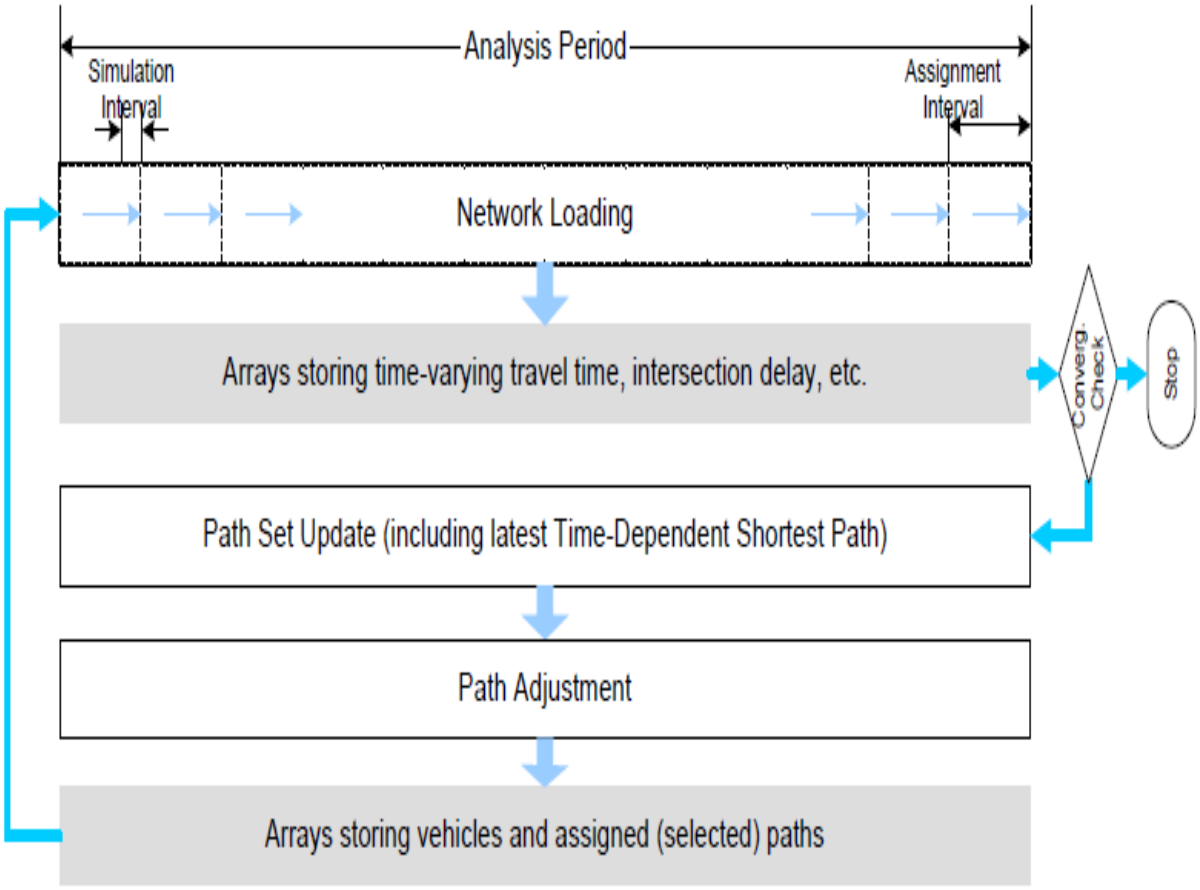


Figure 6.1 General DTA algorithmic procedure

6.1 Routing Procedure

Travel forecasting models are used in transportation planning to evaluate the impact of future changes in demographics, land use, or transportation facilities on the performance of a region's transportation system. Traveler behavior is introduced into these forecasting models through sequences of modeling steps. The traditional four-step process, for example, results in travel choices made by groups of homogeneous travelers in aggregate trip-based models. More advanced activity-based processes seek to represent travel choices made by individual travelers. Cost and time of travel are key components of all travel models throughout the entire sequence of model steps. For example, a household's choice of the number of personal vehicles to own is often forecast subject to aggregate measures of the accessibility of the household. The less accessible a household is, the more likely it is to own automobiles. An accessibility measure is then some representation of the travel time and costs from the residence to workplaces or shopping places. Likewise, time and cost are significant factors in other choices made, including residential, workplace, and discretionary activity locations, as well as factors in deciding which transportation services to use and which routes to follow when engaging in travel. From a travel forecasting perspective, the time and cost of travel are critical factors. Those measures are also critical in quantifying impacts on a regional scale to inform policy decisions. Travel-time and cost measures determined using static network analysis (assignment) procedures use variables of interest that are time-invariant. It has become increasingly evident that these procedures are inadequate as explanations of influences on travel choices and as measures used to evaluate impacts when deciding how to develop policies for managing transportation systems, how to fund transportation system improvements, and how to measure environmental impacts related to system-wide travel.

Dynamic network analysis models seek to provide another, more detailed means to represent the interaction between travel choices, traffic flows, and time and cost measures in a temporally coherent manner (e.g., further improve upon the existing time-of-day static assignment approach). More specifically, Dynamic Traffic Assignment (DTA) models aim to describe such time-varying network and demand

interaction using a behaviorally sound approach. The DTA model analysis results can be used to evaluate many more meaningful measures related to individual travel time and cost, as well as system-wide network measures for regional planning purposes. Now we can describe the many advantages DTA models have over static network assignment models that make them better suited for use in travel forecasting models and transportation planning studies.

Traffic engineers increasingly rely on traffic analysis tools(matsim-dynest-visim-sumo) to analyze and evaluate the current and future performance of transportation facilities for various modes of transport. There are a variety of analytical procedures and methodologies available that support different aspects of traffic and transportation analyses. Nowadays, most traffic analysts rely on either analytical/deterministic tools or microscopic simulation modeling to assess the performance of transportation systems of interest.

Newer microscopic models are route-based, meaning vehicles select a route at departure and follow that route with or without further updates along the journey during the simulation.

Most microscopic simulation models provide various ways by which a vehicle's route at departure or enroot is selected or updated. Each approach is linked to a distinct route choice behavior and, while such flexibility can be of great convenience to the modeling work at hand, one needs to realize the underlying route choice behavior assumption associated with each method, as well as the impact of analysis outcomes depending upon which of the different available mechanisms is chosen.

For example, the "one-shot" (non-iterative) assignment-simulation approach is commonly used in some micro-simulators, in which vehicles departing at different times are given a route that is periodically updated in simulation-based on instantaneous travel-times – snapshot travel-time measured at the time that the routes are generated without considering congestion during subsequent periods. Such an

assignment can be regarded as if travelers strictly follow some types of “pre-trip” route guidance.

Some micro-simulation models allow the enroot vehicles to update their routes based on the updated shortest route generated at a later time. This feature also implies a route choice behavior that strictly follows the enroot route guidance. While these two route choice behaviors exist in reality, it is important to realize that the majority of travelers may choose a route that leads to the minimally experienced travel-time instead of minimal instantaneous travel-time. The experienced travel-time needs to be evaluated “after the fact”, by which point the traffic condition along the entire journey is revealed and experienced. In other words, choosing a minimal experienced travel-time route at departure involves anticipation of future traffic conditions along the journey. This anticipation is usually formed by learning from prior experience (e.g., try different routes).

To account for this “learning” process, an iterative algorithmic process is needed.

Such an iterative process reflects the learning and adjustment in route choice from one iteration to the next until the traveler cannot find a route with a shorter experienced travel-time. The equilibrium-seeking DTA methods are based on iterative algorithmic procedures that are particularly aimed at describing such an individual route/departure time choice adjustment as well as at relating such changes to the network-level performance through simulation.

These models apply iterative procedures involving the interplay of vehicular traffic loading and assignment algorithm to adjust the traveler route assignment for travelers departing at different times to select the respective minimal experienced travel-time route.

Many simulation-based DTA models adopt more computationally efficient traffic simulation logic (at the price of simplifying some simulation fidelity or detail) to be able to describe a corridor/region-wide traffic flow shift at a larger geographical scope (from a corridor up to a region) and over a longer period (from peak hours to 24 hours), compared with microscopic models. Most of these simulation methods are generally defined as mesoscopic simulation sharing common characteristics with

microscopic models – individual vehicles are represented and vehicle dynamic states are simulated through simplified car-following or traffic flow theories without describing detailed inter-vehicle interactions (e.g. lane changing or gap acceptance).

6.2 STA Vs DTA

In a model defined on a relatively long time-of-day period, such as the peak period, the congestion properties of each link are described by a link-time/performance function that expresses the average or steady-state travel-time on a link as a function of the volume of traffic on the link. Such models are called "static". The volume of traffic on the link is determined directly from the loading of the Origin-Departure (O-D) matrix to links via routes. The travel times of each link on a route are added together to determine the route travel time. This approach has some limitations as far as the realism with which it represents the actual process (taking place on the road) that gives rise to congestion and increased travel time. In a static model, inflow to a link is always equal to the outflow: the travel time simply increases as the inflow and outflow ("volume") increases. The volume on a link may increase indefinitely, and exceed the physical capacity (in vehicles/hour) of the link, as represented by a volume/capacity (v/c) ratio > 11. Since the link volume does not conform to the traffic flow limit that results from the physical characteristics of the roadway, the assigned link volume can be considered as demand – trips desired to traverse the link – instead of the actual flow. V/C > 1.0 means that the demand exceeds the capacity and subsequently, congestion will occur. The drawback of using V/C is that it does not directly correlate with any physical measure describing congestion (e.g., speed, density or queue, etc.) In dynamic models, as in reality, explicit modeling of traffic flow dynamics ensures direct linkage between travel-time and congestion. If link outflow is lower than link inflow, link density (or concentration) will increase (congestion), and speed will decrease (fundamental speed-density relationship), and therefore link travel time will increase.

Outflow from a link may be reduced, and thus be potentially less than the inflow, for various reasons, such as:

- Merging two lanes into one (e.g., at a freeway on-ramp) effectively reduces the capacity of each of the two merging lanes;
- Weaving (lane change maneuvers that cross over each other) also reduces link capacity;
- On arterial streets, traffic signals reduce the outflow capacity of links;
- On both freeways and arterial streets, significant over-saturation for one exiting movement from a link can result in reduced flow rates on the other exiting movements, due to a local choke-off effect.

Traffic initially becomes congested (e.g., queuing occurs) at the end of a link because link inflow is greater than link outflow (put another way, a congested traffic state arises at the end of the link under these conditions). According to the basic tenets of traffic flow theory – upon which dynamic models are based – for a given value of outflow, there is a corresponding value of density and speed under congested conditions. This is best thought of in the case of a freeway, where the outflow is roughly constant, as opposed to a signalized road where the outflow is constantly fluctuating. For the moment – for purposes of this discussion, we assume that the outflow is constant. The longer this condition (inflow > outflow) persists, the more vehicles accumulate on the link, and the portion of the link covered by the congested traffic grows in the upstream direction until it reaches the link entrance. At this point, the inflow is reduced. It is equal to the outflow, and the link is in a steady-state condition, meaning that speed, density, and flow are essentially constant at all positions (in space) along with the link. The speed and density on the link corresponding to the flow (inflow and outflow, which are equal) in a well-defined mathematical way, called the "fundamental diagram" of traffic flow. In a dynamic model, each link may be defined by its fundamental diagram, if desired. This is sometimes thought of as the dynamic analogy to the static VDF, but this analogy is loose as the two mathematical relationships perform very different functions in the contexts of their respective models. In a static model, the VDF represents the congested condition, while in a dynamic model, the fundamental diagram describes how congestion at the exit node (reduced link outflow) is propagated upstream through the link until it "spills back" onto the next upstream links. This phenomenon brings forth the question of congestion spill-back, which is not represented in a static

model. At the moment that the link inflow becomes equal to the outflow (as described above), the congestion then continues to spread upstream into whichever upstream links are feeding traffic into the congested link. The outflows of these links are thus reduced, and the process repeats as described above. This queue spillback process also describes how a long queue (congested traffic) can be represented over a sequence of links in a dynamic traffic model. There is also the question of link FIFO (first-in-first-out). Static models, and even some dynamic models that are based on fluid mechanics, enforce the link FIFO rule. In a static model, this means that all vehicles traveling on the link experience the same travel time. In a dynamic model with FIFO, this means that all vehicles entering the link at a given point in time experience the same travel time. What this implies is that there is no overtaking between vehicles and, in particular, this means no overtaking between vehicles that exit the link by different turning movements. In reality, it is quite obvious that if there are two turning movements for exiting a link and if one is oversaturated and the other is not, then the vehicles in the queue for the over-saturated movement can be overtaken by the other vehicles (assuming the link has more than one lane), and that the latter vehicles can have significantly lower travel times than the former. Models that move individual vehicles on discrete lanes of the roadway can model non-FIFO conditions realistically, and thus do not need employing the FIFO assumption. Further, if the turn-bay queue spills back to the through lane, the resulting capacity reduction also needs to be properly accounted for through appropriate traffic modeling. Lastly, it is worth noting that, as there is no explicit representation of individual lanes in static models, there can be no distinction between the traffic conditions on different lanes of the same link.

There is no way to represent the fact, for example, that the outside lane of a freeway is at a crawl due to an oversaturated off-ramp, while the other lanes are moving at a higher speed. In summary, the limitations of static models due to their use of VDFs include:

- Using VDFs, a link may have a volume/capacity (V/C) ratio greater than 1.0; the V/C ratio does not have intuitive traffic meaning.
- VDFs assume link FIFO, and therefore no overtaking.

- VDFs do not distinguish between different lanes on a roadway.
- VDFs are based on a single value of link flow (or volume), implying that inflow is equal to outflow, and hence there is no accumulation of traffic on the link. As a result, there is no representation of the phenomenon of congestion spillback, i.e., where congested traffic spans a sequence of two or more links due to a downstream bottleneck.

Beyond the issues related directly to the use of the VDF and how travel time is determined in static models, other limitations include, for example, modeling of signal synchronization, modeling of lane-based effects, such as High Occupancy Vehicle (HOV) or High Occupancy Toll (HOT) lanes, as they require representing the special lane as a parallel link. Most intelligent transportation systems (ITS)-related applications, such as traveler information systems and advanced network control schemes (e.g., adaptive control and ramp metering), are beyond the modeling capabilities of static assignment models. Notwithstanding the above critique, this document does not intentionally overlook the merit of static models. The widely recognized advantages of static models, including the ability to solve large-scale problems, to converge to precise equilibria and to provide consistency of solutions (if a proper algorithm is used with a sufficient number of iterations) have been aiding policy/project decision-making for agencies for decades. The critique is of benefit in demonstrating the contrast with dynamic models, but the contributions and merits of the static models should not be understated.

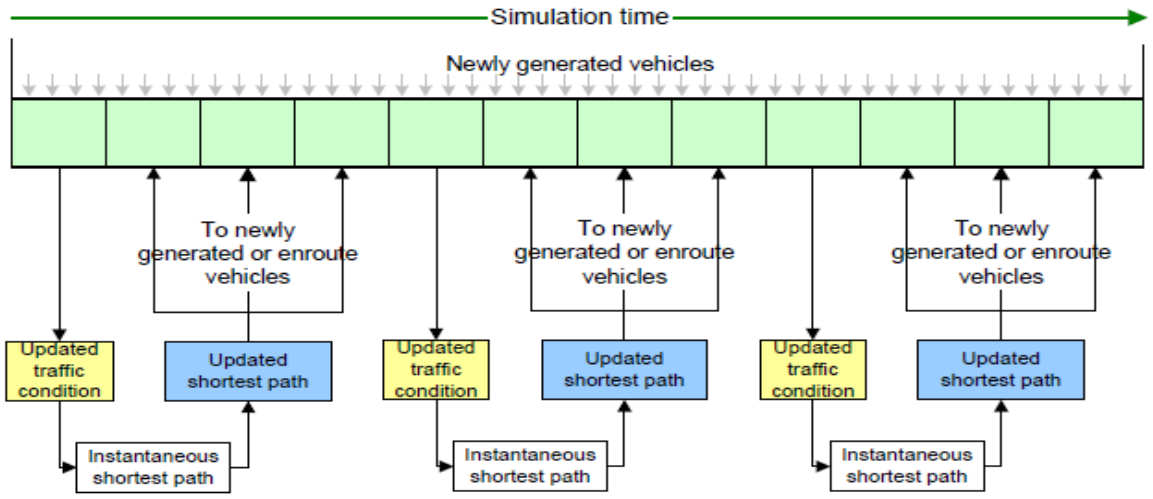


Figure 6.2 Dynamic assignment (with feedback) in a one-shot simulation

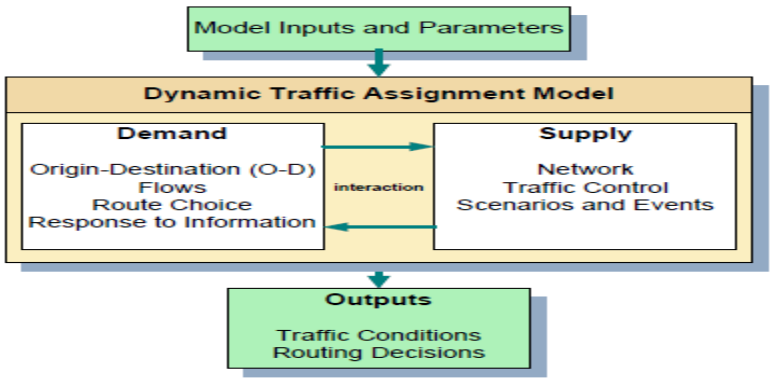


Figure 6.3 Structure of a generic DTA model

6.3 equilibrium solution

The equilibrium solution is the interdependence between different travelers' route choices and travel times. If all travelers were to shift to the shortest routes found in the previous step, those routes would become highly congested and would no longer be shortest. Therefore, only some travelers' route choices should be adjusted, avoid overcorrecting. Generally, this step involves finding which routes in the set need to be increased with assignment flow/vehicles and which to be decreased, and by how much. Normally, the newly found TDSP along with several other good routes (with close to minimal travel time) are among those to be increased with flows. Underperforming routes (long travel time) are decreased with the flow. It is also noteworthy that at this step, not all vehicles will select (or be assigned) a new route. The adjustment made is only what is necessary to achieve equal travel among all routers in the current set. After performing Path Assignment Adjustment, the algorithm returns to the route evaluation step to determine the traffic pattern that would result from the new route choices (route flows). Thus, the three steps work sequentially: the output of the network loading provides the input for Path Set Update; the output of Path Set Update provides the input for Path Assignment Adjustment, and the output of Path Assignment Adjustment provides the input for Network Loading. These three steps are repeated until a stopping criterion is met. The algorithmic structure is illustrated in Figure 2.1. The stopping criterion is typically computed at the end of the network loading step. Older DTA solution algorithms applied a solution approach called the Method of Successive Averages (MSA). MSA imposes a pre-determined fixed amount of flow adjustment at each

iteration, implying a slower convergence, but most recent algorithms employ the notion of the relative gap as the stopping criterion.

6.4 Defining Quality of DTA Model Outputs

- Convergence

a DTA algorithm can test for convergence by calculating various metrics that measure the deviations in flow patterns or congestion indexes (such as experienced travel-times) between successive iterations and checking to determine whether they are less than a pre-specified tolerance level.

- Solution Sensitivity and Stability

Sensitivity and stability are two notions that relate to how a problem's solution varies as a function of parameters that characterize the problem. It describes the expected behavior of DTA problem solutions as the problem itself is changed. For example, given a DTA solution for a particular O-D pattern and network, what might be expected of the solution for a modified problem in which a link is added or changed.

time-dependent route choices and link flows and times change as the algorithm progresses towards an equilibrium solution.

- The realism of Traffic Dynamics

6.5 Tools to make routing be done

6.5.1 Original plan: MAT-Sim

MATSim: Multi-agent-transport-simulation.

MATSim is an activity-based, extendable, multi-agent simulation framework implemented in Java. It is open-source. It needs minimally 3 files (network file- population file- configuration file).

The framework is designed for large-scale scenarios, meaning that all models' features are stripped down to efficiently handle the targeted functionality; parallelization has also been very important

Using MAT-Sim through the GUI to provide Transportation Analysis and Simulation System as well as bringing together expertise in traffic flow, large-scale computation, choice modeling and CAS (Complex Adaptive Systems):

- Microscopic modeling of traffic: MATSim performs an integral microscopic simulation of resulting traffic flows and the congestion they produce.
- Microscopic behavioral modeling of demand/agent-based modeling: MATSim uses a microscopic description of demand by tracing the daily schedule and the synthetic travelers' decisions. In
- retrospect, this can be called "agent-based".
- Computational physics: MATSim performs fast microscopic simulations with 107 or more "particles".
- Complex adaptive systems/co-evolutionary algorithms: MATSim optimizes the experienced utilities of the whole schedule through the co-evolutionary search for the resulting equilibrium or steady state.

MATSim is based on the co-evolutionary principle. Every agent repeatedly optimizes its daily activity schedule while in competition for space-time slots with all other agents on the transportation infrastructure. This is somewhat similar to the route assignment iterative cycle but goes beyond route assignment by incorporating other choice dimensions like time choice, mode choice, or destination choice into the iterative loop.

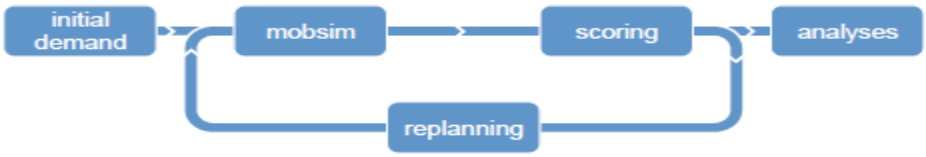


Figure 6.4 MATSim loop, (MATSim cycle)

6.5.1.1 The simulation is done by

A MATSim run contains a configurable number of iterations, represented by the loop. It starts with an initial **demand** arising from the study area population's

daily activity chains. The modeled persons are called **agents** in MATSim. Activity chains are usually derived from empirical data through sampling or discrete choice modeling. A variety of approaches is suitable, as evidenced in the scenarios. During iterations, this initial demand is optimized individually by each agent. Every agent possesses a memory containing a fixed number of day plans, where each plan is composed of a daily activity chain and an associated score. The score can be interpreted as an econometric utility.

Scenario Data: population.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE population SYSTEM "http://www.matsim.org/files/dtd/population_v6.dtd">
<population>
  <person id="1">
    <attributes>
      <attribute name="employed" class="java.lang.Boolean" >true</attribute>
    </attributes>
    <plan score="140.00982053869416" selected="yes">
      <activity type="home" link="1112" x="890.0" y="685.0" end_time="07:43:56" />
      <leg mode="car" dep_time="07:18:16" trav_time="00:01:40">
        <route type="links" start_link="1112" end_link="2223" trav_time="00:31:40" distance="14403.0">1112 1222 2223</r
      </leg>
      <activity type="work" link="2223" x="1802.0" y="2782.0" end_time="17:09:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="2223" end_link="1312" trav_time="00:27:32" distance="12573.0">2223 2313 1312</r
      </leg>
      <activity type="shopping" link="1312" x="1802.0" y="2782.0" end_time="17:42:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="1312" end_link="1112" trav_time="00:04:15" distance="3127.0">1312 1211 1112</ro
      </leg>
      <activity type="home" link="1112" x="890.0" y="685.0" />
    </plan>
    <plan ...>
      ...
  </person>
  ...
</population>
```

Figure 6.5 population file

In every iteration, before the simulation of the network loading with the MATSim mobsim (mobility simulation) each agent selects a plan from its memory. This selection is dependent on the plan scores, which are computed after each mobsim run, based on the executed plans' performances. A certain share of the agents (often 10 %) is allowed to clone the selected plan and modify this clone (planning).

For the network loading step, multiple mobsims are available and configurable.

Plan modification is performed by the planning modules. Four dimensions are usually considered for MATSim at this time: departure time (and, implicitly, activity duration), route, mode, and destination.

Further dimensions, such as activity adding or dropping, or parking and group choices are currently under development and only available experimentally. MATSim planning offers different strategies to adapt plans, ranging from random mutation to approximate suggestions, to best-response answers where, in every iteration, the currently optimal choice is searched. For example, routing often is a best-response modification, while time and mode planning are random mutations.

toynetwork.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE network SYSTEM "http://www.matsim.org/files/dtd/network_v1.dtd">
<network name=" moza network ">
<nodes>
<node id= "1" x=" -100.0" y="1000.0 "/>
<node id= "2" x=" 100.0 " y=" 1000.0 "/>
<node id= "3" x=" 200.0 " y=" 0.0 "/>
<node id= "4" x=" 100.0" y="-1000.0 "/>
<node id= "5" x=" -100.0 " y=" -1000.0 "/>
<node id= "6" x=" -200.0 " y=" 0.0 "/>
<node id= "7" x=" 400.0" y="-2000.0 "/>
<node id= "8" x=" 1000.0 " y=" 0.0 "/>
<node id= "9" x=" 400.0 " y=" 2000.0 "/>
<node id= "10" x=" -400.0" y="2000.0 "/>
<node id= "11" x=" -1000.0 " y=" 0.0 "/>
<node id= "12" x=" -400.0 " y=" -2000.0 "/>
</nodes>
<links>
<link id= "1" from= "1" to= "2" length= " 200.00 " capacity= " 3600 "freespeed= " 27.78 " permlanes= "2" modes= " car " />
<link id= "2" from= "2" to= "3" length= " 1004.987562  " capacity= " 3600 "freespeed= " 27.78 " permlanes= "2" modes= " car " />
<link id= "3" from= "3" to= "4" length= " 1004.987562  " capacity= " 3600 "freespeed= " 27.78 " permlanes= "2" modes= " car " />
<link id= "4" from= "4" to= "5" length= " 200.00 " capacity= " 3600 "freespeed= " 27.78 " permlanes= "2" modes= " car " />
<link id= "5" from= "5" to= "6" length= " 1004.987562 " capacity= " 3600 "freespeed= " 27.78 " permlanes= "2" modes= " car " />
<link id= "6" from= "6" to= "1" length= " 1004.987562 " capacity= " 3600 "freespeed= " 27.78 " permlanes= "2" modes= " car " />

<link id= "7" from= "2" to= "9" length= " 1004.987562 " capacity= " 3000 "freespeed= " 40.5 " permlanes= "2" modes= " car " />
<link id= "8" from= "3" to= "8" length= " 800 " capacity= " 3000 "freespeed= " 40.5 " permlanes= "2" modes= " car " />
<link id= "9" from= "4" to= "7" length= " 1004.987562 " capacity= " 3000 "freespeed= " 40.5 " permlanes= "2" modes= " car " />
<link id= "10" from= "5" to= "12" length= " 1004.987562  " capacity= " 3000 "freespeed= " 40.5 " permlanes= "2" modes= " car " />
<link id= "11" from= "6" to= "11" length= " 800 " capacity= " 3000 "freespeed= " 40.5 " permlanes= "2" modes= " car " />
<link id= "12" from= "1" to= "10" length= " 1004.987562 " capacity= " 3000 "freespeed= " 40.5 " permlanes= "2" modes= " car " />

<link id= "13" from= "10" to= "9" length= " 800 " capacity= " 4000 "freespeed= " 25.5 " permlanes= "2" modes= " car " />
<link id= "14" from= "9" to= "8" length= " 2154.065923 " capacity= " 4000 "freespeed= " 25.5 " permlanes= "2" modes= " car " />
<link id= "15" from= "8" to= "7" length= " 2154.065923 " capacity= " 4000 "freespeed= " 25.5 " permlanes= "2" modes= " car " />
<link id= "16" from= "7" to= "12" length= " 800 " capacity= " 4000 "freespeed= " 25.5 " permlanes= "2" modes= " car " />
<link id= "17" from= "12" to= "11" length= " 2154.065923 " capacity= " 4000 "freespeed= " 25.5 " permlanes= "2" modes= " car " />
<link id= "18" from= "10" to= "9" length= " 2154.065923 " capacity= " 4000 "freespeed= " 25.5 " permlanes= "2" modes= " car " />

</links>
</network>
```

eXtensible Markup Language file | length : 2,991 | lines : 41 | Ln : 39 | Col : 1 | Sel : 0 | 0 | Windows (CR LF) | UTF-8

Figure 6.6 network file

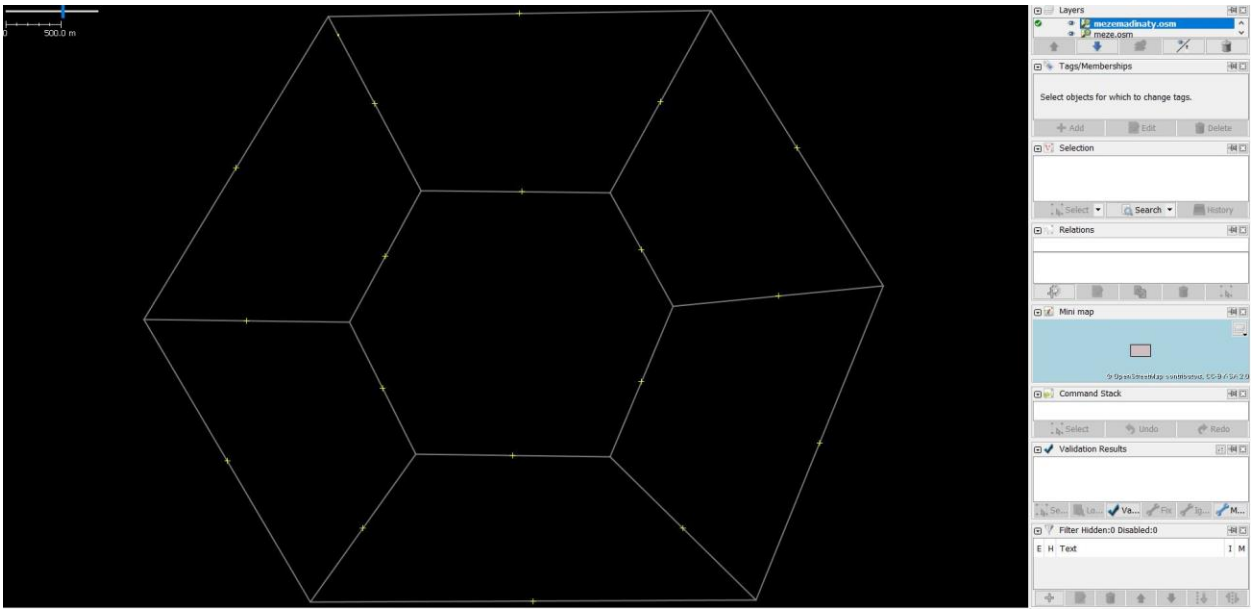


Figure 6.7 toy network

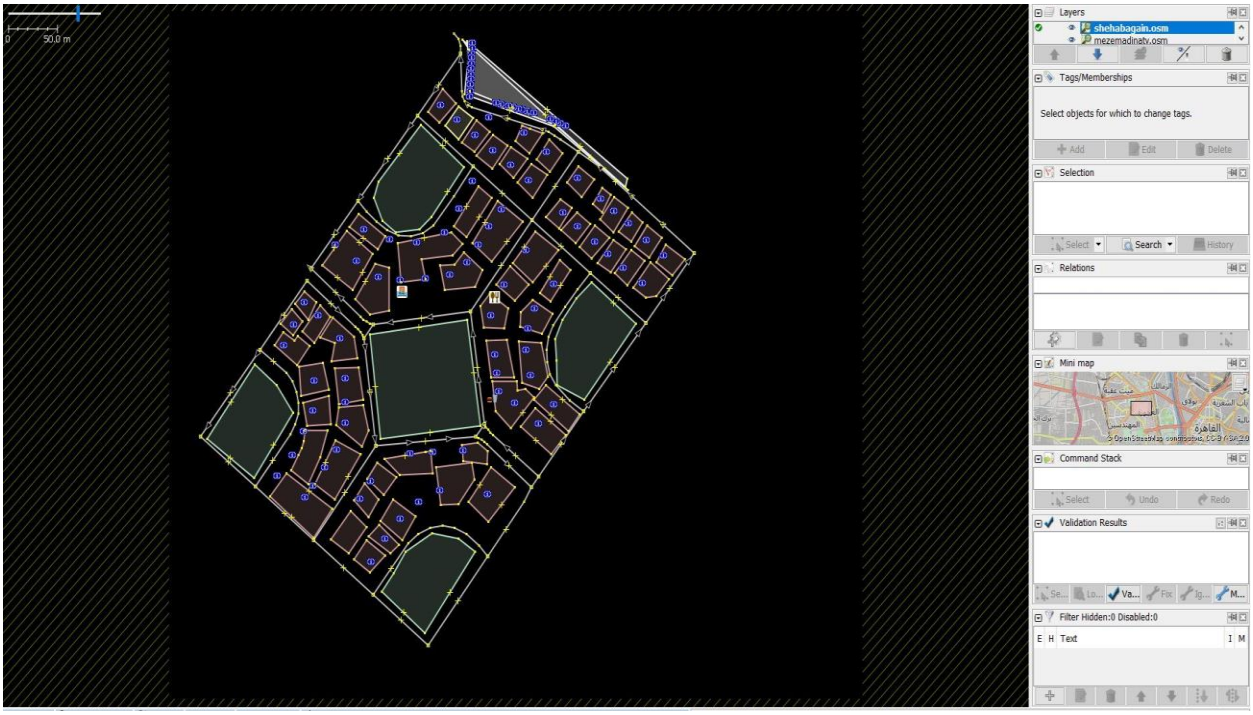


Figure 6.8 real network

Initial day chains do not have to be very carefully defined for the planning dimensions included in the optimization process. Plausible values just speed up the optimization process.

If an agent ends up with too many plans (configurable), the plan with the lowest score (configurable) is removed from the agent's memory. Agents that have not undergone planning select between existing plans. The selection model is configurable; in many MATSim investigations, a model generating a logit distribution for plan selection is used.

An iteration is completed by evaluating the agents' experiences with the selected day plans (*scoring*).

The iterative process is repeated until the average population score stabilizes. The typical score development curve takes the form of evolutionary optimization progress. Since the simulations are stochastic, one cannot use convergence criteria appropriate for deterministic algorithms; for a discussion of possible approaches for the MATSim situation.

```
    <module name="global">
        <param name="randomSeed" value="4711" />
        <param name="coordinateSystem" value="Atlantis" /> <!--WGS84/EPSG/ESRI      // three co-ordinate systems -->
    </module>
    <module name="network">
        <param name="inputNetworkFile" value="network.xml" />
    </module>
    <module name="facilities">
        <param name="inputFacilitiesFile" value="facilities.xml" />
        <param name="facilitiesSource" value="fromFile"/>
    </module>
    <module name="plans">
        <param name="inputPlansFile" value="plan11.xml" />
    </module>
    <module name="controler">
        <param name="outputDirectory" value="./output" />
        <param name="firstIteration" value="0" />
        <param name="lastIteration" value="10" />
    </module>
    <module name="qsim">
        <!-- "start/endTime" of MobSim (00:00:00 == take earliest activity time/ run as long as active vehicles exist) -->
        <param name="startTime" value="00:00:00" />
        <param name="endTime" value="00:00:00" />
            <param name = "snapshotperiod"  value = "00:00:00"/> <!-- 00:00:00 means NO snapshot writing -->
    </module>
    <module name="planCalcScore">
        <param name="learningRate" value="1.0" />
        <param name="BrainExpBeta" value="2.0" />
        <param name="lateArrival" value="-18" />
        <param name="earlyDeparture" value="-0" />
        <param name="performing" value="+6" />
        <param name="traveling" value="-6" />
        <param name="waiting" value="-0" />
        <param name="activityType_0"            value="h" /> <!-- home -->
        <param name="activityPriority_0"        value="1" />
        <param name="activityTypicalDuration_0" value="12:00:00" />
        <param name="activityMinimalDuration_0" value="08:00:00" />
        <param name="activityType_1"            value="w" /> <!-- work -->
        <param name="activityPriority_1"        value="1" />
        <param name="activityTypicalDuration_1" value="08:00:00" />
        <param name="activityMinimalDuration_1" value="06:00:00" />
        <param name="activityOpeningTime_1"     value="07:00:00" />
        <param name="activityLatestStartTime_1" value="09:00:00" />
        <param name="activityEarliestEndTime_1" value="" />
        <param name="activityClosingTime_1"     value="18:00:00" />
    </module>
        <module name="strategy">
        <param name="maxAgentPlanMemorySize" value="5" /> <!-- 0 means unlimited -->
        <param name="ModuleProbability_1" value="0.9" />
        <param name="Module_1" value="BestScore" />
        <param name="ModuleProbability_2" value="0.1" />
        <param name="Module_2" value="ReRoute" />
    </module>
</config>
```

Figure 6.9 configuration file

6.5.1.2 Drawbacks and challenges

Our main target is to apply the machine learning algorithm outputs (delay) on routes that will control the routing between an origin and a destination.

MATSim could not be configured as We have tried to change the plans of every agent to give different scores to the needed routes while the simulated program already has a plan to calculate score function which gives the score according to the implemented algorithm which cannot be changed.so we need to go to an alternative plan.

6.5.2 Alternative plan: Google developer platform

Google APIs:

is a set of application programming interfaces (APIs) developed by Google which allows communication with Google Services and their integration to other services.

Examples of these include Search, Gmail, Translate, or Google Maps. Third-party apps can use these APIs to take advantage of or extend the functionality of the existing services.

The APIs provide functionality like analytics, machine learning as a service (the Prediction API), or access to user data (when permission to read the data is given). Another important example is an embedded Google map on a website, which can be achieved using the Static Maps API, Places API, or Google Earth API.

Authentication and Authorization

Usage of all of the APIs requires Authentication and Authorization using the OAuth 2.0 protocol. Oauth 2.0 is a simple protocol. To start, it is necessary to obtain credentials from the Developers Console. Then the client app can request an Access Token from the Google Authorization Server and uses that Token for authorization when accessing a Google API service.

Client libraries

There are client libraries in various languages that allow developers to use Google APIs from within their code, including Java, JavaScript, Ruby, .NET, Objective-C, PHP, and Python. [5]

The Google Loader is a JavaScript library that allows web developers to easily load other JavaScript API provided by Google and other developers of popular libraries. Google Loader provides a JavaScript method for loading a specific API (also called a module), in which additional settings can be specified such as API version, language, location, selected packages, load callback (computer programming), and other parameters specific to a particular API. Dynamic loading or auto-loading is also supported to enhance the performance of the application using the loaded APIs.

Google Apps Script

Google Apps Script is a cloud-based JavaScript platform that allows developers to write scripts the only owner can manipulate API services such as Calendar, Docs,

Drive, Gmail, and Sheets and easily create Add-Ons for these services with chromium-based applications.

the algorithm of Google Maps

maps and navigation systems have many algorithms including:

- Spatial indexing algorithms and algorithms of computational geometry to organize the map data and retrieve it efficiently.
- Algorithms to draw maps (e.g. project latitude and longitude coordinates, fill the polygons, place names for streets, cities, businesses, parks, …).
- Algorithms to understand queries from users (NLU or NLP)
- Algorithms to process GPS signals (on which I am much less expert than on other topics here)
- Algorithms to perform what is called geocoding, converting addresses to points (or polygons) on a map.
- Algorithms to perform reverse geocoding, converting points to addresses using point in polygon algorithms (another example of using computational geometry).

I'll get to the route calculation algorithms, but let me mention more of the other complex aspects of maps and navigation especially those relevant to computing routes.

- Gathering and organizing the data: it can come from many different sources. Just one example of a very complex problem that arises from that is the very messy geometry that results when the map coordinates from two sources (latitude, longitude) are in slight disagreement – especially polygons and lines. Besides data about roads, data about addresses, businesses, parks, malls, and institutions are needed.
- Iddo mentions "costs". Yes, that's important and not at all trivial for various reasons. The costs are the "weights" on the graph's edges to use the terminology of graph theory, but computing those weights accurately is complex. Here are some complex parts of determining costs:
    - Collecting and using traffic data so that slow roads are less likely to be chosen in the route calculation.
    - Estimating future costs when the route is long and is driven in the future.
    - Time to travel through intersections of roads. All navigation products take into account the differences in time needed to make different turns at an intersection because the times can vary significantly – left turns usually take much longer than right turns but not always. And the many

and complex varieties of configurations of intersections make this a challenging problem.

- Additional parameters to consider are fuel costs and the complexity of the route. If a route with a few turns is only slightly longer than a route with many turns, it is likely to be a better choice for the driver.

- I mentioned intersections. The diagram below might provide some idea of how difficult it is to deal with intersections because not every turn is allowed. The restrictions shown must be encoded into the graph used by the route calculation algorithm.

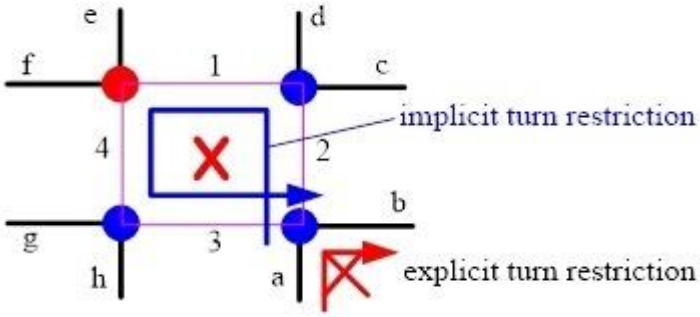


Figure 6.10 restriction

Okay, I'll discuss route calculation algorithms themselves but remember it's only one component of many in the map and navigation technology. As some have mentioned computing a route a long way across a large map can be expensive and slow. So, optimizations of the Dijkstra are needed:

- A*. A* has been a very important concept in traditional AI, and many route calculation algorithms include it. But, in my experience in developing the algorithms, A* offers only a minor improvement in performance – maybe 30%. For other applications in AI, the graphs might be very different in such a way the A* provides a larger performance advantage, but not much for road networks.
- Bi-directional. This is a somewhat more effective optimization and almost all route calculation algorithms in the industry use it but not primarily for the performance advantage it brings. It means that the route is computed both forward from the origin (with Dijkstra, A*, reach-based routing, highway hierarchies, contraction hierarchies, or other methods) and backward from the destination. Like A*, the technique reduces the area searched, but more importantly, some more important algorithms (ones using road hierarchies) require it to work. But bi-directional route computations bring their complications – the algorithm must determine when the two searches have met

at a point on an optimum or near optimum route and that's more complex than it sounds.

- Using the road network hierarchy. This is the most important performance optimization and was used long ago at companies like Etak in a practical but not academically formal way. My invention and paper on reach-based routing brought the concept into academia and spawned further research producing algorithms like Highway Hierarchies and Contraction Hierarchies. Since the lead author of the papers on Highway Hierarchies, Peter Sanders, gave a talk at Google, it has been speculated that Google has used Highway Hierarchies:
- Shortcuts. This concept replaces sequences of edges with single edges in the preparation of the graph. The concept was used in the industry before I published my paper and I regret not having space to mention it under Further Work in my paper. The folks at Karlsruhe in Germany took full advantage of shortcuts in Highway Hierarchies and Contraction Hierarchies.
- Other issues affecting the algorithms:
- Even if we ignore performance, Dijkstra does not solve all problems. One problem is time-dependency. Sometimes the algorithms must allow the costs on the edges to change depending on the time that the route brings the user to a particular point on the roads.
- For example, the road might be closed at a particular time.
- Or traffic might be predicted to become worse.
- Most of the fast algorithms require processing on the map data before any routes are computed, but some of the data affecting that "pre-processing" changes quickly, e.g. traffic data. Determining what part of the problem is solved in pre-processing and what part is solved during the route calculation and how it is solved is very difficult.

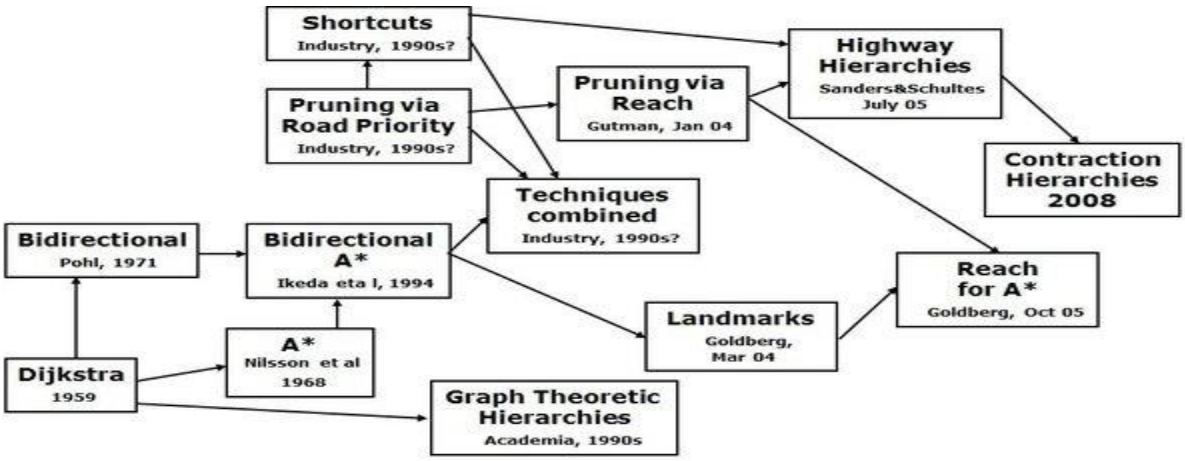


Figure 6.11 algorithm history

One final list of complexities that map and navigation technologies face:

- Guidance: how is the user instructed on the route and when?
- Detecting when the driver has gone off the route and needs a new route.
- Special requirements depending on the type of transportation
- Bicycles cannot legally use all roads.
- Electric vehicles needing to be charged along the route.
- Use of trains and buses
- Trips that combine different modes of transportation including walking
- Where will the vehicle park? That's not trivial in city centers like San Francisco.
- What if the navigation device (e.g. cell phone) can only hold map data for nearby areas but the driver is on a long trip that sometimes travels out of the cell network.
- Keeping map data up to date on cell phones.
- Differences between states and countries that affect
- Rules for using roads.
- Language and addressing systems supported by geocoding.
- 3D map data – organizing and displaying it. Complex 3D graphics algorithms here.

Google Maps javascript API

Build customized, agile experiences that bring the real world to your users with static and dynamic maps, Street View imagery, and 360° views.

The API provides:

- An interactive map to your website. Customize it with your content and imagery.
- Waypoints specify an array of DirectionsWaypoints. Waypoints alter a route by routing it through the specified location(s).
- ProvideRouteAlternatives when set to true specifies that the Directions service may provide more than one route alternative in the response. Note that providing route alternatives may increase the response time from the server. This is only available for requests without intermediate waypoints.
- You can calculate directions (using a variety of methods of transportation) by using the DirectionsService object. This object communicates with the Google Maps API Directions Service which receives direction requests and returns an efficient path. Travel time is the primary factor that is optimized, but other factors such as distance, number of turns, and many more may be taken into account. You may either handle these directions results yourself or use the DirectionsRenderer object to render these results.

Using Java Script

Java-Script is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved. It is the third layer of the layer cake of standard web technologies, two of which (HTML and CSS) we have covered in much more detail in other parts of the Learning Area.

- HTML is the markup language that we use to structure and give meaning to our web content, for example defining paragraphs, headings, and data tables, or embedding images and videos on the page.
- CSS is a language of style rules that we use to apply styling to our HTML content, for example setting background colors and fonts, and laying out our content in multiple columns.
- JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. (Okay, not everything, but it is amazing what you can achieve with a few lines of JavaScript code.)

The core client-side JavaScript language consists of some common programming features that allow you to do things like:

- Store useful values inside variables. In the above example, for instance, we ask for a new name to be entered then store that name in a variable called name.
- Operations on pieces of text (known as "strings" in programming). In the above example, we take the string "Player 1: " and join it to the name variable to create the complete text label, e.g. "Player 1: Chris".
- Running code in response to certain events occurring on a web page. We used a click event in our example above to detect when the button is clicked and then run the code that updates the text label.
- And much more!

What is even more exciting however is the functionality built on top of the client-side JavaScript language. So-called Application Programming Interfaces (APIs) provide you with extra superpowers to use in your JavaScript code.

APIs are ready-made sets of code building blocks that allow a developer to implement programs that would otherwise be hard or impossible to implement. They do the same thing for programming that ready-made furniture kits do for home building — it is much easier to take ready-cut panels and screw them together to make a bookshelf than it is to work out the design yourself, go and find the correct

wood, cut all the panels to the right size and shape, find the correct-sized screws, and then put them together to make a bookshelf.

6.6 Outputs

We have thankfully reached all the possible routes between an origin and destination given the waypoint of each route which is in the form of latitude and longitude. each two-way point represents a route that has a cost of delay.

the chosen route is the route which has the least accumulative cost.

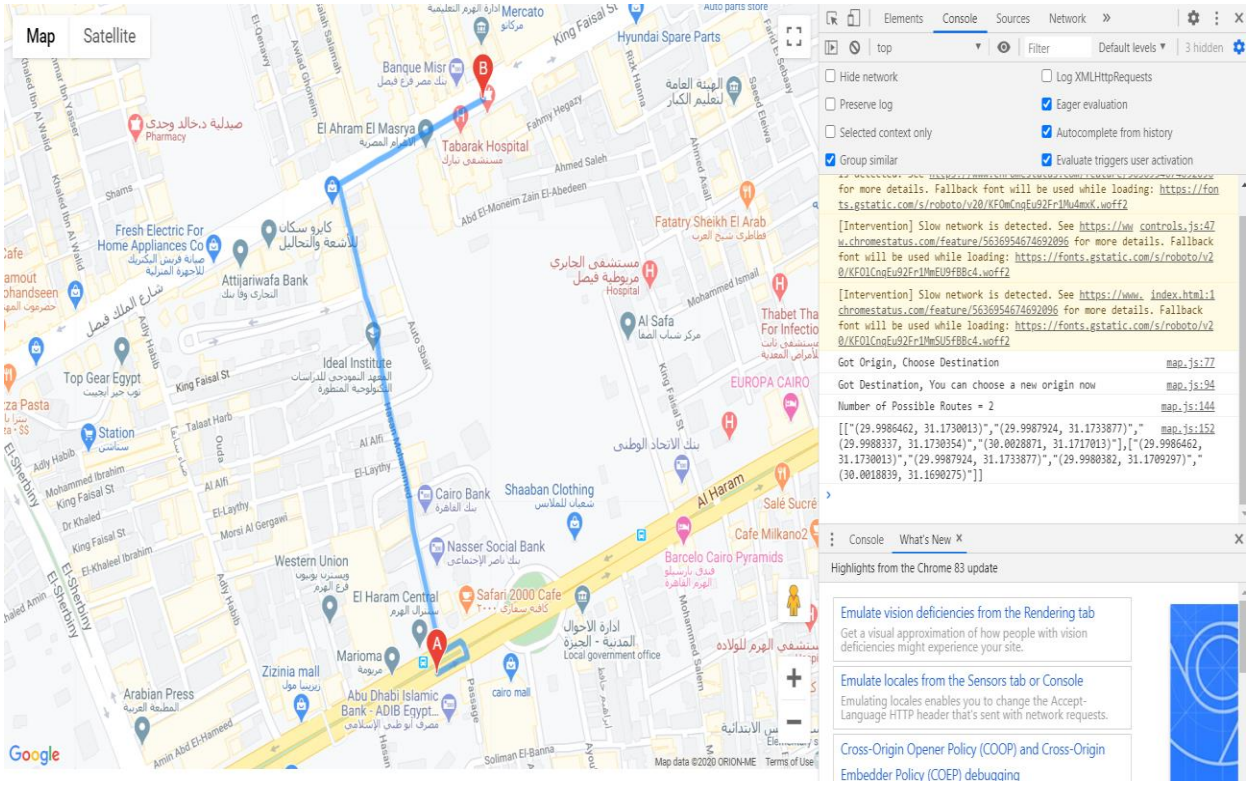


Figure 6.12 Output displayed on a map

Chapter 7

User Data Gathering Backend

Now, we try to connect what we built before all together to get the final output on our USI which is the website, we try to handle all the input data and output of the DL model and present them all on the map that the user access it.

7.1 procedure

- implementation of the map from google java-script as we discussed in the chapter before.
- using a flask (python library) to get all the possible routes from the java-script map as a JSON request.
- returning this data (all the possible routes) to the map on the website to be available to the USI user.
- By using an online server(pythonanywhere) to host URL which has the needed data to be sent to the original website.

7.2 Handling JSON

- This route will receive some JSON, parse the data, do some validation, and return a new JSON response.
- Posting data to the server so passing the method's argument to the @app. route() decorator, along with the HTTP methods allowed for this route.
- To gain access to request object in Flask import it from the Flask library to use it in any view functions

7.2.1 Receiving JSON from an HTTP Request

To handle the incoming JSON. Flask provides the handy request.get_json () the method, which parses any incoming JSON data into a Python dictionary.

7.3 Functions

7.3.1 Flask

Is a micro web framework written in Python? It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

In any web app, you'll have to process incoming request data from users. Flask, like any other web framework, allows you to access the requested data easily.

one of the extensions is Cross-Origin Resource Sharing (CORS), making cross-origin AJAX possible.

7.3.2 Using JSON with CORS

When using JSON cross origin, browsers will issue a pre-flight OPTIONS request for POST requests. For browsers to allow POST requests with a JSON content type, you must allow the Content-Type header. The simplest way to do this is to simply set the CORS_HEADERS configuration value on your application

7.3.3 The Request Object

The request object is an instance of a Request subclass and provides all of the attributes Werkzeug defines

To access the incoming data in Flask, you have to use the request object. The request object holds all incoming data from the request, which includes the mimetype, referrer, IP address, raw data, HTTP method, and headers, among other things. Although all the information the request object holds can be useful.

Jsonfiy

Creates a Response with the JSON representation of the given arguments with an application/JSON mimetype. The arguments to this function are the same as the dict constructor. For security reasons, only objects are supported top level. For more information about this, have a look at JSON Security.

JSON. loads ()

Takes in a string and returns a JSON object.

JSON. dumps ()

Takes in a JSON object and returns a string.

Reading the incoming JSON data

Assignment everything from the JSON object into a variable using request.get_json (). which converts the JSON object into Python data for us. Let's assign the incoming request data to variables and return them by making the following changes to our JSON-example route.

Method

The request method. (For example, 'GET' or 'POST')

- GET: Sends data in unencrypted form to the server. Most common method.
- POST: Used to send HTML form data to the server. Data received by the POST method is not cached by the server.

By default, the Flask route responds to the GET requests. However, this preference can be altered by providing methods argument to route () decorator.

To demonstrate the use of the POST method in URL routing, first, let us create an HTML form and use the POST method to send form data to a URL.

Chapter 8

Model Cloud Deployment

After Building a strong backend algorithm using machine learning different algorithms and routing algorithms, the model deployment is critical as it plays many different roles i.e.

- Friendly UI with the users
- The connection between routing and machine learning algorithms

The model deployment contains many parts such as:

- Amazon SageMaker hosting
- Website Building

8.1 Amazon SageMaker

Amazon SageMaker is a cloud machine-learning platform that was launched in November 2017. SageMaker enables developers to create, train, and deploy machine-learning models in the cloud. SageMaker also enables developers to deploy ML models on embedded systems and edge-devices.

Using SageMaker, one can bind the model to a website to interface with the machine learning model.

8.2 Cloud Computing

Cloud computing can be easily interpreted as transforming an IT product to an IT service. We're dealing with cloud computing daily using many applications like Google Drive, iCloud, Microsoft OneDrive, and many more.

E.g. using a hard drive (a product) to store images can be transformed to using Google Drive (a service), this is why cloud computing is a process of transforming an IT product into IT service.

Using cloud computing can be helpful in many ways, your data is portable and easy to access on any device and surely has a lower cost.

Cloud storage is one of many examples of cloud computing, there are also cloud applications, databases, virtual machines, and many other cloud services.

8.2.1 Why Cloud Computing

Cloud computing is proven to be efficient and dependable in many ways. Startups with IT needs to find it better to start with cloud services as its implementation cost and running time is very low in comparison to implementing a whole new system for the corporate as well as upgrading/downgrading the system is made easily based on the corporate needs and with less financial risks.

8.2.2 Pros and Cons of Cloud computing

Cloud computing has proven to be beneficial with various benefits such as:

- Reduced Investments and Proportional Costs (providing cost reduction)
- Increased Scalability (providing simplified capacity planning)
- Increased Availability and Reliability (providing organizational agility)
  Although it has many pros, it comes with some risks:
- (Potential) Increase in Security Vulnerabilities
- Reduced Operational Governance Control (over cloud resources)
- Limited Portability Between Cloud Providers
- Multi-regional Compliance and Legal Issues

8.3 Deployment to Production

Deployment to Production is a process of introducing a machine learning model into a running environment where the model takes actions or give results based on system inputs.

To deploy a model, many ways can be used but the most three used ways are:

- Python model is recoded into the programming language of the production environment.

- Model is coded in Predictive Model Markup Language (PMML) or Portable Format Analytics (PFA).
- Python model is converted into a format that can be used in the production environment.

8.3.1 Recoding Model into Programming Language of Production Environment

The first method involves recording the Python model into the language of the production environment, often Java or C++. This method is rarely used anymore because it takes time to recode, test, and validate the model that provides the same predictions as the original.

8.3.2 Model is coded in PMML or PFA

The second method is to code the model in Predictive Model Markup Language (PMML) or Portable Format for Analytics (PFA), which are two complementary standards that simplify moving predictive models to deployment into a production environment. The Data Mining Group developed both PMML and PFA to provide vendor-neutral executable model specifications for certain predictive models used by data mining and machine learning. Certain analytic software allows for the direct import of PMML including but not limited to IBM SPSS, R, SAS Base & Enterprise Miner, Apache Spark, Teradata Warehouse Miner, and TIBCO Spotfire. The third method is to build a Python model

8.3.3 Model is converted into Format that's used in the Production Environment

The third method to build a Python model and use libraries and methods that convert the model into code that can be used in the production environment. Specifically, most popular machine learning software frameworks, like PyTorch, TensorFlow, SciKit-Learn, have methods that will convert Python models into an intermediate standard format, like ONNX (Open Neural Network Exchange Format). This intermediate standard format then can be converted into the software native to the production environment.

This is the easiest and fastest way to move a Python model from modeling directly to deployment.

- Moving forward this is typically the way models are moved into the production environment.
- Technologies like containers, endpoints, and APIs (Application Programming Interfaces) also help ease the work required for deploying a model into the production environment.

8.3.4 Endpoint

An endpoint in the deployment process is the interface to the model, this endpoint facilitates the communication between the model and the application where this interface:

- Allows the application to send user data to the model and
- Receives predictions back from the model-based upon that user data.
  One way to think of the endpoint that acts as this interface is to think of a Python program where:
- The endpoint itself is like a function call.
- The function itself would be the model.
- The Python program is the application.

E.g. Figure 1 shows a simple example of an Application, Model, and Endpoint connection, where:

- The endpoint: line 8 function call to ml_model
- The model: beginning on line 14 function definition for ml_model
- The application: Python program web_app.py

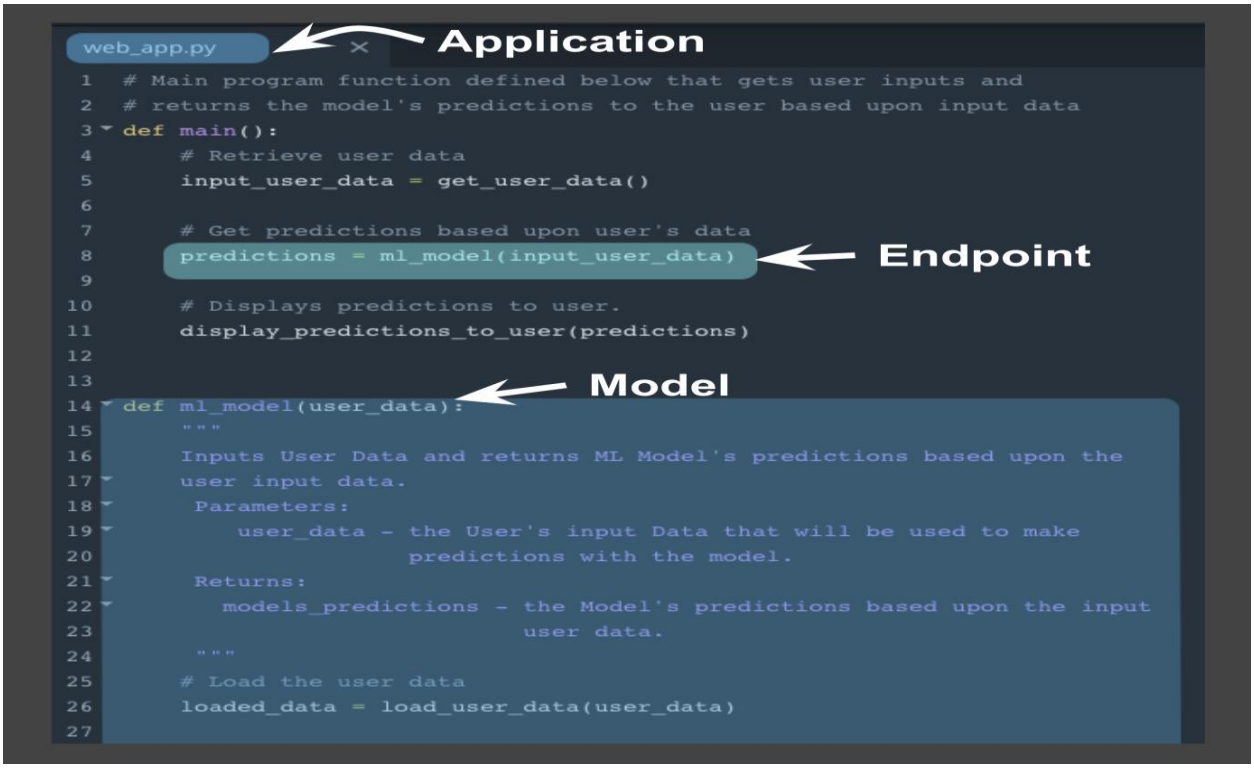


Figure 8.1 App, model, and endpoint

8.3.5 How Endpoint Works

The interface between application and model takes place through endpoint where the endpoint is considered an application programming interface (API)

- An API is thought of as a set of rules used to set the interface between the model and the application.
- Here, REST API is used, as it uses HTTP requests and responses to handle the communication between the application and the model through an endpoint.
- HTTP request and response are the communication between the model and the application.

8.3.5.1 HTTP Request

The HTTP request sent from the application to model is composed of 4 crucial parts:

1. Endpoint

This endpoint will be in the form of a URL, Uniform Resource Locator, which is commonly known as a web address.

2. HTTP Method

HTTP has many methods; POST is used in deployment.

3. HTTP Headers

The headers will contain additional information, like the format of the data within the message, that's passed to the receiving program.

4. Message

The final part is the message (data or body); for deployment will contain the user's data which is input into the model.

8.3.5.2 HTTP Response

The HTTP response has 3 parts:

1. HTTP status code

A status whether the request is successful or not, start with 2 if successful

2. HTTP Headers

The headers will contain additional information, like the format of the data within the message, that's passed to the receiving program.

3. Message

What's returned as the data within the message is the prediction that's provided by the model.

This prediction can be represented by a developed UI by the developer on the application.

8.3.6 Application Responsibilities

The application is mainly responsible for:

- To format the user's data in a way that can be easily put into the HTTP request message and used by the model.

- To translate the predictions from the HTTP response message in a way that's easy for the application users to understand.

8.4 Deployment Characteristics

Model deployment has many characteristics, which include:

Model Versioning

With each update to the model, its version must be saved to the model's metadata in the database, this will make it easier to build, monitor, and maintain model versions.

Model Monitoring

The ease of model monitoring post-deployment is important to make sure that the model still meets its intended needs otherwise it need update.

Model updating and routing

The ease of updating a deployed model is another important part of the model deployment, the model update is needed when the model performance starts to decline. One of the most important reasons for updating the model is that the data it gets became different than the data it was trained on.

The deployment platform should support routing differing proportions of user requests to the deployed models; to allow comparison of performance between the deployed model variants.

Routing in this way allows for a test of model performance as compared to other model variants.

Model Predictions

There are two main types of model deployment:

- On-Demand Predictions

- online

- real-time
- synchronous

With On-Demand predictions, one can expect:

- Low latency of response to each prediction request
- But allows for the possibility of high variability in request volume

Predictions are returned in the response from the request. Often these requests and responses are done through an API using JSON or XML formatted strings.

Each prediction request from the user can contain one or many requests for predictions. Noting that many are limited based upon the size of the data sent as the request. Common cloud platforms on-demand prediction request size limits can range from 1.5(ML Engine) to 5 Megabytes (SageMaker).

On-demand predictions are commonly used to provide customers, users, or employees with real-time, online responses based upon a deployed model.

Batch Predictions

- Asynchronous
- batch-based predictions
  With these types of predictions, one expects:
- the high volume of requests with more periodic submissions
- So, latency won't be an issue.

Each batch request will point to a specifically formatted data file of requests and will return the predictions to a file. Cloud services require these files will be stored in the cloud provider's cloud.

Cloud services typically have limits to how much data they can process with each batch request based upon limits they impose on the size of the file you can store in their cloud storage service. For example, Amazon's SageMaker limits batch predictions requests to the size limit they enforce on an object in their S3 storage service.

Batch predictions are commonly used to help make business decisions.

8.5 Deployment on AWS

After model building and verifying, time to deploy it. Amongst many service providers, AWS provides a great service with high reliability and viability. In this section, deployment on Amazon SageMaker will be discussed in detail.

Amazon SageMaker gives the ability to work on your machine learning projects using Jupiter notebooks which makes it easier to write, share, and build machine learning programs alongside data preparations.

After creating an AWS account and entering AWS SageMaker from the console, one can go to notebook instances and create a new instance to start working on machine learning projects using Jupiter notebooks.

On creating a notebook instance, one must choose carefully an IAM role, this role decides which services are bind to the notebook instance. For instance, create a new role with all options enabled.

Start by data exploration ad preparation like any other machine learning project. After data preparation, it's preferable to save your pre-processed data on AWS servers which will guarantee less data preparation time for the next time you use the notebook.

E.g.

```
In [61]: data_dir = '../data/folder' # The folder we will use for storing data
         if not os.path.exists(data_dir): # Make sure that the folder exists
             os.makedirs(data_dir)

In [62]: with open(os.path.join(data_dir, 'preProcesed.pkl'), "wb") as f:
             pickle.dump(preProcesed, f)
```

Figure 8.2 saving a pre-processed list to a pickle file

```
In [67]: import pandas as pd

         pd.concat([pd.DataFrame(train_y), pd.DataFrame(train_X_len), pd.DataFrame(train_X)], axis=1) \
                 .to_csv(os.path.join(data_dir, 'train.csv'), header=False, index=False)
```

Figure 8.3 saving the train data and its labels to a .csv file

Next, we need to upload the training data to the SageMaker default S3 bucket so that we can provide access to it while training our model.

```
In [68]: import sagemaker

         sagemaker_session = sagemaker.Session()

         bucket = sagemaker_session.default_bucket()
         prefix = 'sagemaker/sentiment_rnn'

         role = sagemaker.get_execution_role()

In [69]: input_data = sagemaker_session.upload_data(path=data_dir, bucket=bucket, key_prefix=prefix)
```

Figure 8.4 Creating a SagaMaker session to upload out files to s3 bucket

Note that S3 Bucket is considered as SageMaker's storehouse.

8.5.1 Train on SageMaker

When a model is constructed in SageMaker, an entry point must be specified. This is the Python file which will be executed when the model is trained. Let the training file located at train/train.py

Starting a training job on SageMaker is easy with the following two steps assuming Pytorch use.

```
In [100]: from sagemaker.pytorch import PyTorch

          estimator = PyTorch(entry_point="train.py",
                              source_dir="train",
                              role=role,
                              framework_version='0.4.0',
                              train_instance_count=1,
                              train_instance_type='ml.p2.xlarge',
                              hyperparameters={
                                  'epochs': 10,
                                  'hidden_dim': 200,
                              })

In [101]: estimator.fit({'training': input_data})
```

Figure 8.5 starting a training job on SageMaker

The folder where the .py file is in is called a train, the file where the model is built is train.py.

The role in Pytorch function is the role specified when creating the notebook instance and can be found using the line of code in figure 8.5.

8.5.2 Model Deploying

Once the model is built on SageMaker, it's necessary to test it to check how good the results are on test data. To test the model, it has to be deployed first.

NOTE: When deploying a model, you are asking SageMaker to launch a compute instance that will wait for data to be sent to it. As a result, this compute instance will continue to run until you shut it down. This is important to know since the cost of a deployed endpoint depends on how long it has been running for.

```
In [102]: # TODO: Deploy the trained model
          predictor = estimator.deploy(initial_instance_count = 1, instance_type = 'ml.m4.xlarge')
          ----------------!
```

Figure 8.6 deploying a model in SageMaker

And then to predict, you have to preprocess your test data the same way you did to input data and call

Predictor.predict() function.

Once we've deployed an endpoint it continues to run until we tell it to shut down. Since we are done using our endpoint, for now, we can delete it.

```
In [111]: estimator.delete_endpoint()
```

Figure 8.7 deleting an endpoint

8.5.3 Linking Model to Website

In the upper steps, the model has been deployed and used using predictor function, in this section, it will be shown how to link this endpoint to be used through a website.

Figure 8.8 The link between a web app and a model

The web app can be easily interpreted as an attractive UI to the users, the model is the decision-maker which is the core of the project.

In the sections earlier it was shown why using a REST API in the connection between the model and the app is crucial, but what is Lambda?

Lambda is a straightforward Python function that can be executed whenever a specified event occurs. We will give this function permission to send and receive data from a SageMaker endpoint.

Lastly, the method we will use to execute the Lambda function is a new endpoint that we will create using API Gateway. This endpoint will be a URL that listens for data to be sent to it. Once it gets some data it will pass that data on to the Lambda function and then return whatever the Lambda function returns. Essentially it will act as an interface that lets our web app communicate with the Lambda function.

8.5.3.1 Lambda Function

The Lambda function is the function that will be executed whenever it receives data from the API gateway, then it will send the given data to the SageMaker endpoint and return with the results.

To create a Lambda function on SageMaker:

Create an IAM Role for the Lambda function

Lambda function needs permissions to be able to call SageMaker endpoint, it's important to create an IAM Role that will be given to the Lambda function later.

Using the AWS Console, navigate to the IAM page and click on Roles. Then, click on Create role. Make sure that the AWS service is the type of trusted entity selected and choose Lambda as the service that will use this role, then click Next: Permissions.

In the search box type sagemaker and select the checkbox next to the AmazonSageMakerFullAccess policy. Then, click on Next: Review.

Lastly, give this role a name. Make sure you use a name that you will remember later on, for example, LambdaSageMakerRole. Then, click on Create role.

Creating the Lambda function

Time to build the Lambda function itself and assign to it the IAM Role.

Using the AWS Console, navigate to the AWS Lambda Page and click on Create a function. When you get to the next page, make sure that the Author from scratch is selected. Now, name your Lambda function, using a name that you will remember later on, for example, first_func. Make sure that the Python 3.6 runtime is selected and then choose the role that you created in the previous part. Then, click on Create Function.

8.5.3.2 Setting API Gateway

After setting up the Lambda function, it needs an API Gateway to trigger it.

Using AWS Console, navigate to Amazon API Gateway and then click on Get started.

On the next page, make sure that New API is selected and give the new API a name, for example, first_api. Then, click on Create API.

After creating an API, time to bind it with Lambda function, click on Actions, and then Create Method. A new blank method will be created, select its dropdown menu, and select POST, then click on the checkmark beside it.

For the integration point, make sure that Lambda Function is selected and click on the Use Lambda Proxy integration. This option makes sure that the data that is sent to the API is then sent directly to the Lambda function with no processing. It also means that the return value must be a proper response object as it will also not be processed by API Gateway.

Type the name of the Lambda function you created earlier into the Lambda Function text entry box and then click on Save. Click on OK in the pop-up box that then appears, permitting API Gateway to invoke the Lambda function you created.

The last step in creating the API Gateway is to select the Actions dropdown and click on Deploy API. You will need to create a new Deployment stage and name it anything you like, for example, prod.

After completing the previous steps, the API is successfully set up and working, copy the API URL which will be used to invoke the API.

This URL will be used later on the website in a form to send and receive data.

Chapter 9

Website and User Interface

Building a strong UI is crucial to any IT service, as it contacts directly with the user.

In TrCPRS, the website is important where it's responsible for:

- Friendly UI with the user.
- Receiving the user data (Location and Destination).
- Linking between Routing algorithms and Machine learning model.
- Integrated with Google Maps.
- Acts as the web app in the deployment process.

Building the website needs to meet a certain specification to be able to work well as a friendly UI i.e.

- A well-built structure using HTML.
- Friendly UI using CSS and SASS.
- Dynamic website using JavaScript and jQuery.
- Responsive across multi-platform using Bootstrap.

9.1 HTML Structure

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It's considered as the basic structure for building any web app. The TrCPRS website is built from scratch to ensure it meets the project specifications. The website has many sections such as a header which contains a project slogan, an 'about' section which contains information about the project as well as its vision and mission. The most important

part of the website is the google maps integration part which is responsible for taking the user's location and destination and finally draw the best route on the map.

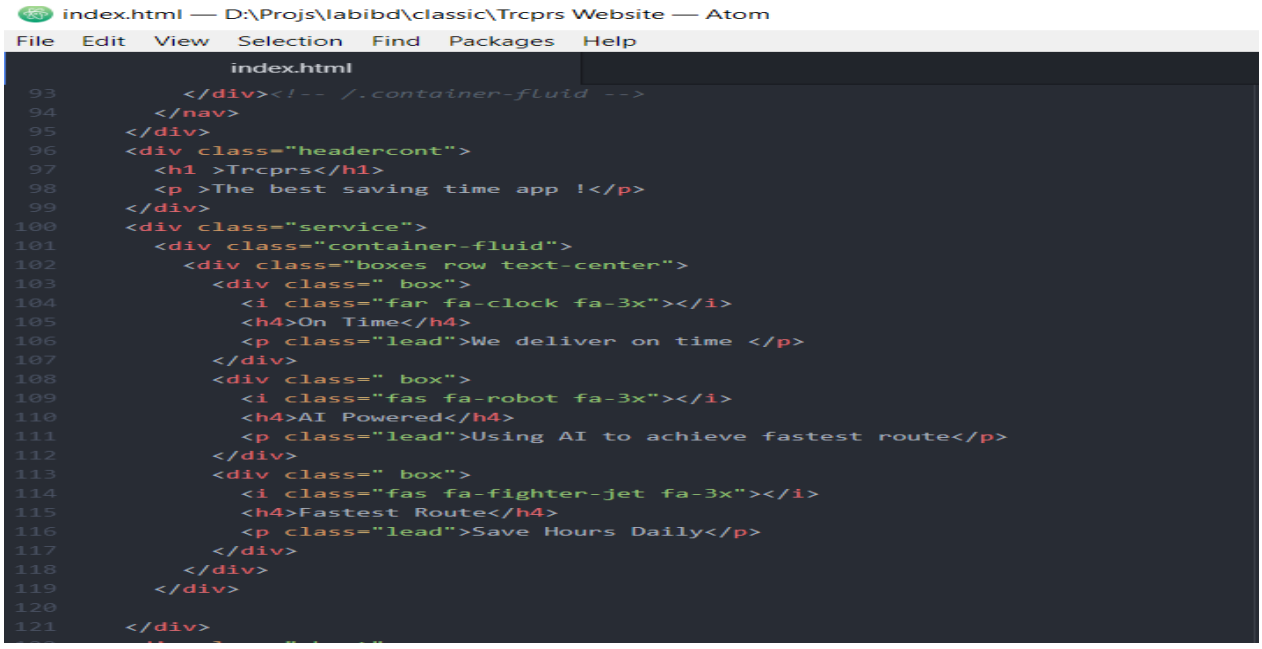


Figure 9.1 Header HTML Structure

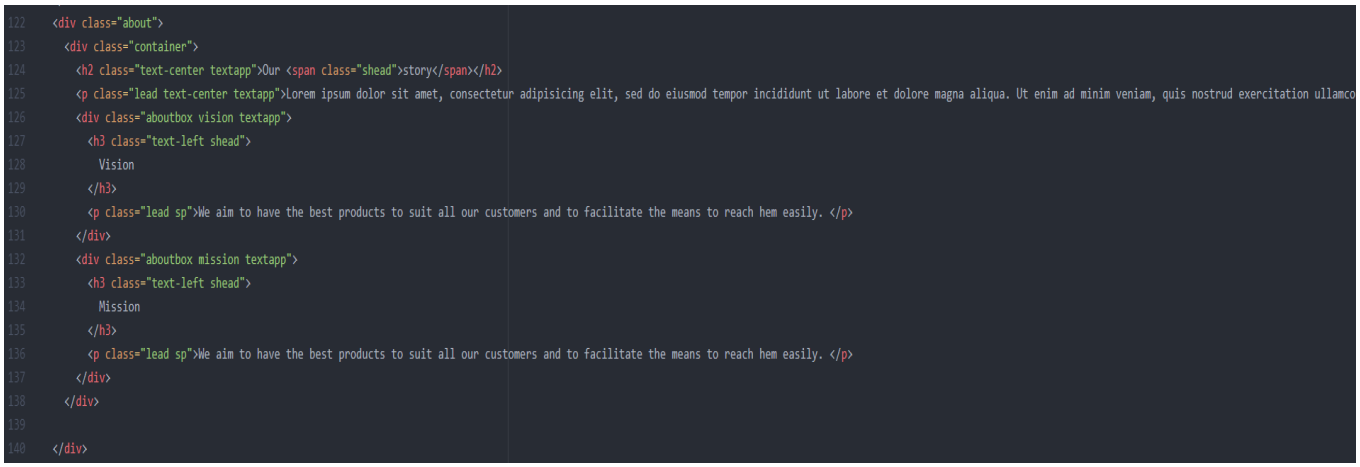


Figure 9.2 About HTML Structure

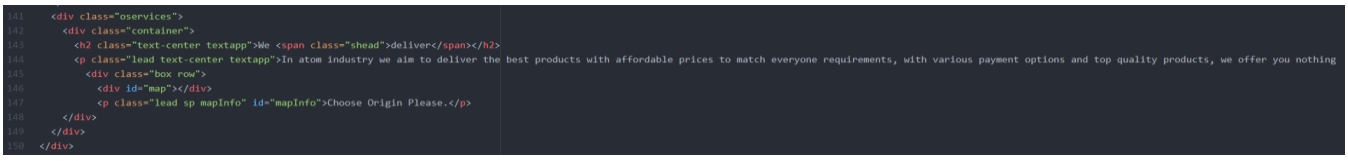


Figure 9.3 Map HTML Structure

9.2 CSS Designs

HTML can be assisted by technologies such as Cascading Style Sheets (CSS) where CSS can be used to add colors, shapes, positions, or any added designs to an HTML Structure.

9.3 Map Design

One of the most crucial parts of TrCPRS is the routing process, where it outputs all the possible routes between two given points. This routing process is done through Google Maps JavaScript API.

In the website, there's a map provided for such purpose where it first asks for user location to center the map on the user's location, then it waits for both the starting and ending points.

In the backend, the maps.js file on the website handles this process where it calls the API using an API key, works on the routing process, and finally print all the possible routes between two points.

Finally, this route is sent as a request to be pre-processed and worked through the machine learning model, the result of this request is the best route given by the model which will be displayed using renderer tools on the map.

The map design is initialized by google maps and is handled by map.js which is called on index.html file as in the next figure.