THE HONG KONG POLYTECHNIC UNIVERSITY

Final Year Project Poster

By Azat ALIMKHANOV (17086198d)

Transfer Learning to Estimate Intersection Delays

Supervisor: Prof CHUNG Chin-Shin Edward

Abstract

Machine learning started gaining popularity only in recent decades with one reason of this being more data available for processing and resulting in better performance of models. Ability to solve complex problems allows to apply these techniques in transportation tasks as well. However, a dataset with significant number of real transportation data cannot be easily obtained as it would require high amount of labor and/or equipment involved. In this project, a process to transfer machine learning model, shortly known as a transfer learning, will be proposed to address this problem.

Objectives

1. Analyze intersection data and make assumptions about it
2. Create the input volumes data that follows conservation of flows
3. Obtain intersection delays for base model (SIDRA INTERSECTION 9.0)
4. Train base machine learning model
5. Create simulation mimicking real traffic
6. Create real data
7. Conduct transfer learning

Background

Previously, the intersection delay was calculated by mathematical methods:

1. Webster [1] method

$$d = \frac{C(1-\lambda)^2}{2(1-\lambda x)} + \frac{x^2}{2q(1-x)} - 0.65\left(C/q^2\right)^{\frac{1}{3}} x^{2+5\lambda}$$

d = vehicle average delay; C = cycle time; q = traffic volume;
λ = effective green to cycle time ratio; s = saturation flow;
g = green time;x = degree of saturation

2. Akcelik [2] method

$$d = \frac{c(1-u)^2}{2(1-y)} + \frac{T_f}{4}\left(z + \sqrt{z^2 + \frac{12(x+x_0)}{QT_f}}\right)$$

d = vehicle average delay; Q = capacity; T_f = period of flow;
z = degree of saturation – 1; z_0 = degree of saturation when queue overflow ~0
c = cycle time; q = traffic flow; y = flow to design ratio;

3. HCM [3] method

$$d = d_1(PF) + d_2 + d_3$$

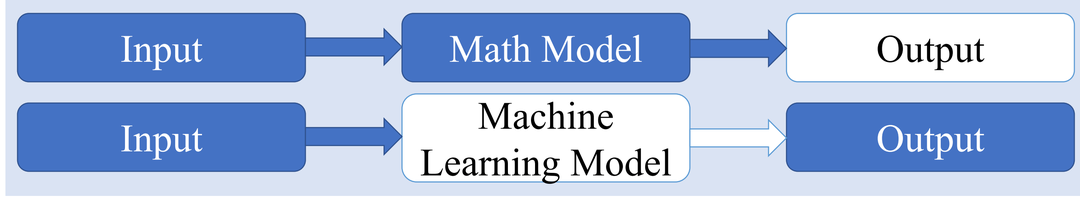
$$d_1 = \frac{0{,}5c\left(1-\frac{q}{c}\right)^2}{1-[\min(1,x)\frac{g}{c}]}$$ is uniform arrival delay

$$d_2 = 900T[9(x-1) + \sqrt{(x-1)^2 + \frac{8klx}{cT}}]$$ is non−uniform arrival delay

$$PF = \frac{(1-P)f_{PA}}{1-\frac{g}{c}}$$ is progression adjustment factor

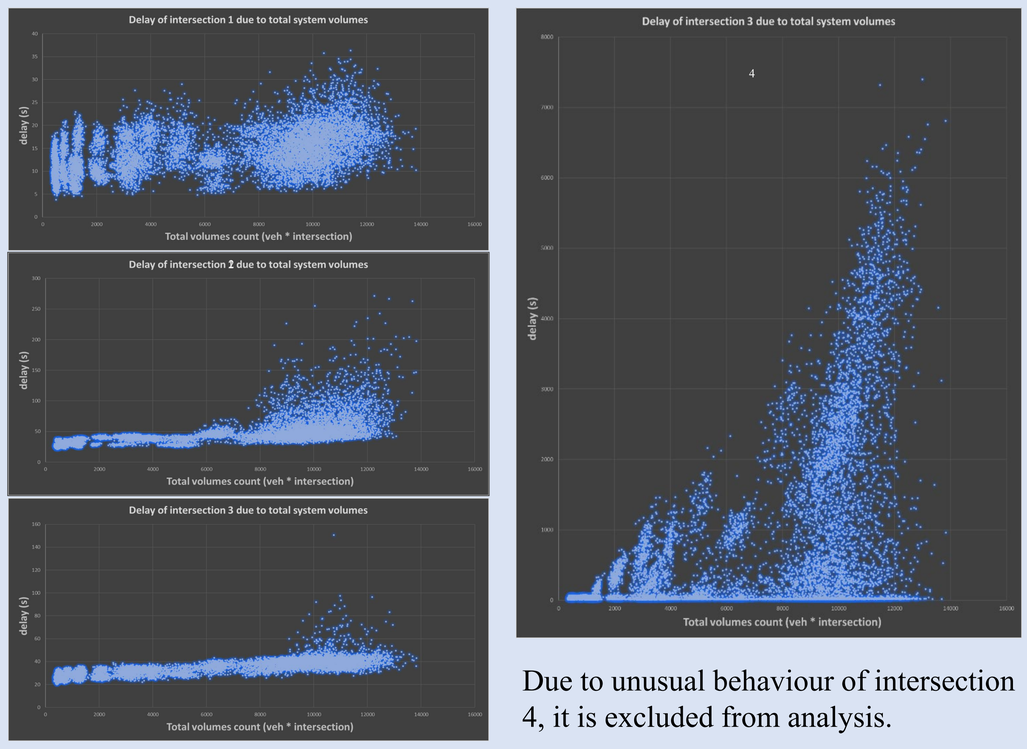
d_3 is an initial delay

The above models are able to provide reasonable estimations only for DoS < 1.0. Murat and Baskan [6] have compared performance of the above formulas and found that Artificial Neural Networks (ANN) perform better, specifically for saturation flow ~ 1.0. For this reason, ANN approach should be preferred over mathematical models. It also assists to find complex relationships for input and output data available.



Methodology

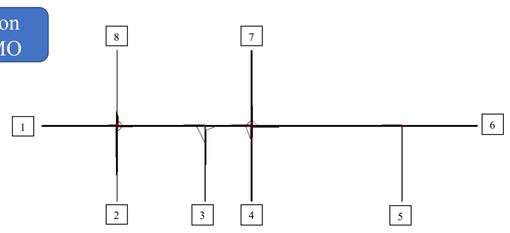
The network delays due to total count of volumes



Due to unusual behaviour of intersection 4, it is excluded from analysis.

- ➔ Initial data included real sensors data, intersection design from government and Google Maps.
- ➔ Based on that data, artificial data and simulation environments have been created. Statistical and math approaches implanted on Python 3 were used to create varying volumes.
- ➔ Volumes have been adjusted to meet criteria of flow conservation and non-zero flow
- ➔ SIDRA API and Python 3 were used to obtain intersections performance parameters
- ➔ Python 3 and tensorflow.keras were used to create and train CNN model.
- ➔ OD matrices have been created from volumes
- ➔ SUMO Transportation network was created with NetEdit
- ➔ SUMO simulation failed to compile with different phase timings
- ➔ Consequently, transfer learning has not been conducted

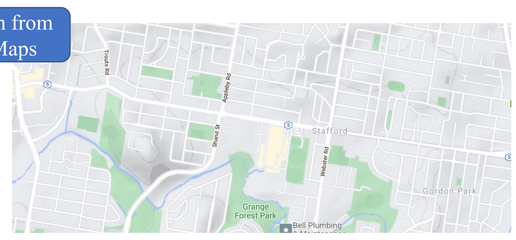
Intersection with SUMO



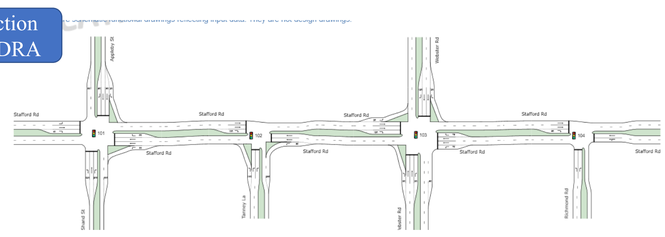
Origin-Destination matrix with SUMO

	1	2	3	4	5	6	7	8
1	1	15	3	4	2	17	3	5
2	9	1	2	3	2	8	2	77
3	6	3	1	3	2	12	3	2
4	3	2	2	1	2	13	37	2
5	4	2	2	2	1	7	2	2
6	38	14	7	11	6	1	14	9
7	8	4	3	27	2	4	1	3
8	6	23	2	2	2	4	2	1

Intersection from Google Maps



Intersection with SIDRA

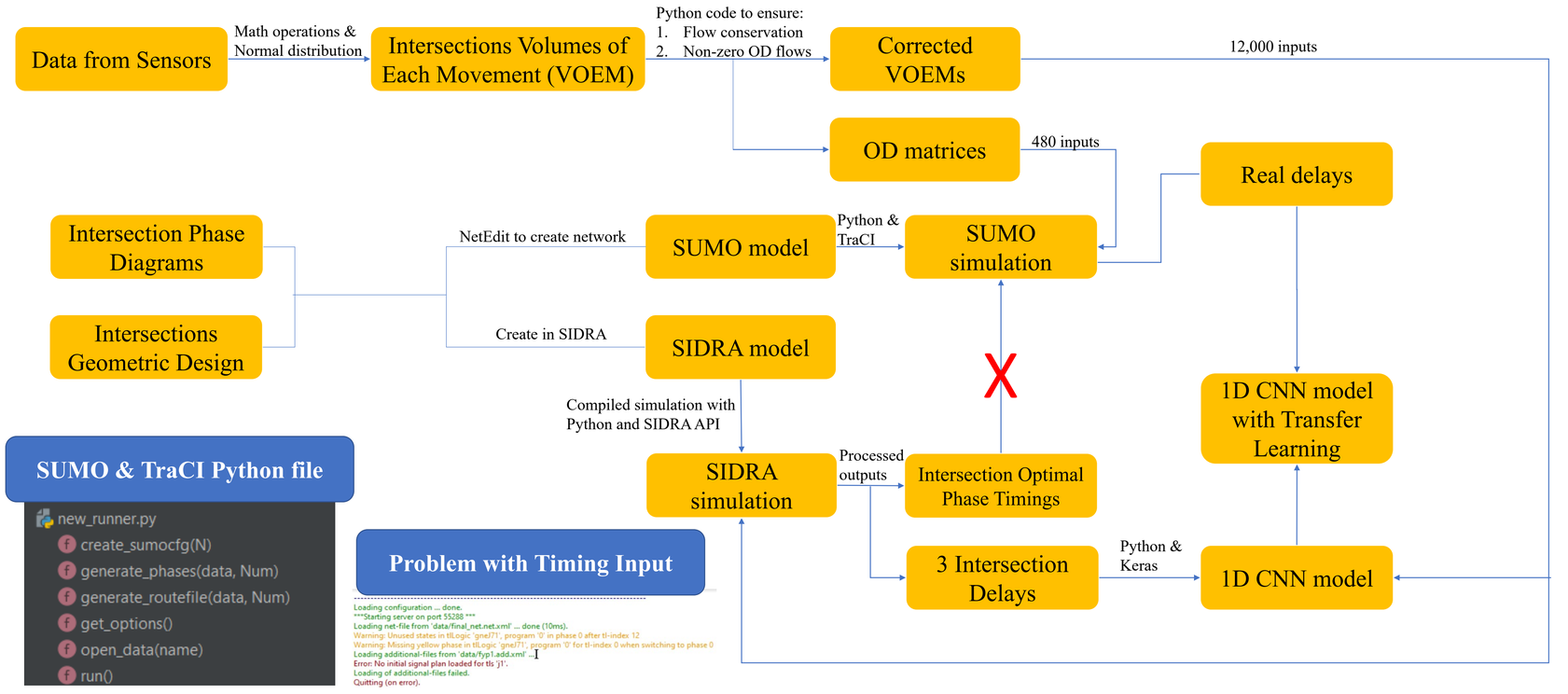


ML Model

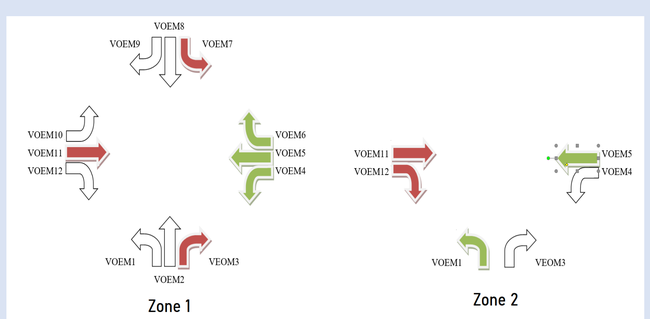
Layer (type)	Output Shape	Param #
lambda (Lambda)	(None, 67, 1)	0
conv1d (Conv1D)	(None, 65, 32)	128
max_pooling1d (MaxPooling1D)	(None, 32, 32)	0
conv1d_1 (Conv1D)	(None, 30, 32)	3104
max_pooling1d_1 (MaxPooling1D)	(None, 15, 32)	0
flatten (Flatten)	(None, 480)	0
dense (Dense)	(None, 144)	69264
dense_1 (Dense)	(None, 64)	9280
dense_2 (Dense)	(None, 288)	18720
dense_3 (Dense)	(None, 3)	867

Total params: 101,363
Trainable params: 101,363
Non-trainable params: 0

	Mean Absolute Error	Quantity (samples)	Mean Absolute Error in Percentage (%)
Train data	1.2628	9600	4.006
Validation data	1.3095	1200	4.154
Test data	1.3433	1200	4.262



Flow Conservation

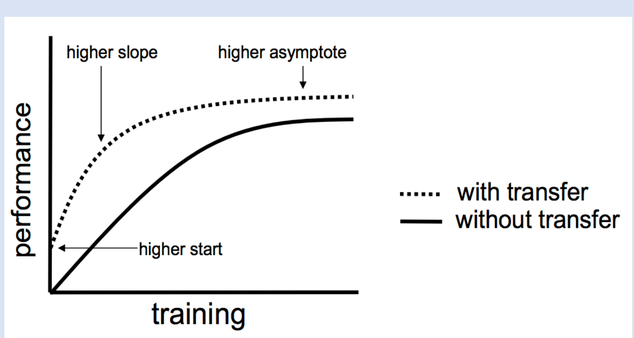


Formulas:

$$\sum_{zone\ 1}^{Green} VOEM = \sum_{zone\ 2}^{Green} VOEM$$
$$\sum_{zone\ 1}^{Red} VOEM = \sum_{zone\ 2}^{Red} VOEM$$

Conclusion

Although the outcome has not been achieved, it can be assumed that the results should look like the following illustration:



Discussion & References

- ➔ This project was an illustration of how input from traffic sensors can be used to obtain one of the traffic parameters without high costs and labor. Different parameters to estimate can be used in this project.
- ➔ This topic can be evolved to use of IoT devices in transportation systems to have some estimation of traffic real-time data.
- ➔ The project also investigated use of 1-D CNN. From my perspective, 2-D CNN can perform better, but they need more traffic parameters, as well as complicated NN model used.

References:

[1] F. V. Webster, Traffic Signal Setting. London: Her Majesty Stationary Office, no. 39, pp. 75-88, 1958.

[2] R. Akcelik, *Traffic signals: capacity and timing analysis*, pp 8-23, 1981.

[3] TRB, *Special Report 209: Highway Capacity Manual*. Washington D C: TRB Press, pp. 365-377, 1965.