# Image-based Vehicle Traffic Monitoring For Estimating Carbon Emissions: Traffic flow prediction model report

Sven van Loon 1009567

Supervisor: Francisco Benita

December 13, 2024

**Abstract**

This report presents a hybrid LSTM-PINN approach for traffic flow prediction, validated using the UTD19 dataset due to limitations in Singaporean data. The physical constraint come from the flow conservation equation. Results demonstrate the model's potential to predict traffic flow under physical constraints, showcasing acceptable results for areas lacking data.

## Contents

## 1 Introduction

This iUROP project involves collecting traffic data every 5 minutes, such as density and velocity, from various locations in Singapore. Using computer vision models, images retrieved from LTA cameras are processed to estimate the number of vehicles, and the velocity is estimated using the Google API by extracting speed and travel duration, within a 1 km radius of the cameras to correlate with observed traffic patterns. Such data could be very useful for predicting traffic flow. That is why as part of my contribution to this project, I was tasked with training a Physically Informed Neural Network (PINN) to predict future traffic flow values, both for existing locations and for areas where data is currently unavailable. Why is it relevant for this project? Let us rewind the 3 goals of this iUROP project:

1. Monitor traffic conditions in Singapore by analyzing images retrieved from speed cameras.
2. Predict carbon emissions generated by vehicles.
3. Reconstruct historical traffic patterns over the past six years.

A model capable of predicting future traffic flow, including locations with no prior data, would greatly support goal number 2. It enables forecasting of near-future carbon emissions and extends those predictions to data-sparse areas.

Currently the traffic flow predicitons methods are model-driven, data-driven or hybrid of both. PINN falls under the hybrid category, it is a regular neural network, which is physically informed. That way a PINN tries to satisfy both fitting the data and at the same time respecting the physical rules. This dual capability helps connect data-driven methods, which may lack clear explanations, and model-driven methods, which often have difficulty with incomplete or noisy data. By using a PINN, we can make sure predictions respect real-world physical rules while still adapting to the data. The physical constraints are usually encoded into the loss function. A common applications of PINNs are solving PDEs, solving inverse problems or predicting physical systems [1]. In this study, we use PINNs to predict the traffic flow by incorporating conservation laws into the model.

## 2 Literature review

This section examines the usage of Physics-Informed Neural Networks (PINNs) in traffic dynamics modeling. Researchers have looked deeply into traffic flow prediction by using different approaches such as model-driven, data-driven, and hybrid of both. Here I will focus on the hybrid approach, where usually a neural network is employed with integrated physics information. One study predicted the density at different points of a network with 3 roads [2]. They trained a different PINN for each road. They simulated traffic data for each road, it produced vehicle trajectory data: speed, flow and density for random points on the road. The physical constraint here was the LWR model, which is also based on the continuity equation [3].Their data is nearly continuous, covering both the entire spatial extent of each link and the full time period of interest. This means they have access to detailed, granular information at every point along the road links and at every moment within the simulated timeframe. In contrast, our data is considerably sparser:

- We only have data available at specific points—namely, at the places where cameras are.
- Data is recorded at discrete time intervals, specifically every 5 minutes, rather than continuously.

Another paper used Beijing's third ring road traffic dataset, which is a sparse dataset [4]. It is a dataset from one road, so it is not a network, and they are predicting the traffic flow in different points in time on one road. They trained a Self-adaptive equation embedded PINN (SAEE-PINN), which unlike a traditional PINN incorporates an adaptive mechanism that modifies the physical equations themselves during training. They are solving an inverse problem, they are not using the network to predict different traffic states, they are using the network to find the equation that describes the traffic flow and using that equation to predict the traffic flow.

I have not found any papers that would use a Physically Informed LSTM in traffic flow prediction. Existing works either use standard PINNs for physical modeling or LSTMs for time-series data (without incorporating physics).

## 3 Methodology

My methodology combines Long Short-Term Memory (LSTM) networks with Physics-Informed Neural Networks (PINNs) to predict traffic flow at multiple locations using one-hour traffic data as input. This hybrid approach uses the LSTM network's ability to capture temporal dependencies in the traffic data and the PINN's capability to incorporate physical constraints and domain knowledge into the learning process. Below, I detail the dataset preparation, model architecture, training procedure, and evaluation methodology.

## Dataset

The initial goal of this project was to train the model using data collected from cameras in Singapore. However, the dataset lacked sufficient location diversity to validate predictions for areas without existing data. Essentially in the Singaporean data, there is no intersection where I have all the flows going into the intersection and all the flows going out of the intersection available, without it, it is impossible to build the model with the physical flow conservation constraints and evaluate it. As an example take the intersection from Figure 1. I have all the flows going in or out and from or to the north camera (red indicates data available in 2 directions) and the same with the west, however, there is a missing flow for the camera in the east (green indicates data available in one direction). To address this limitation, I decided to use data from a different city. I selected London

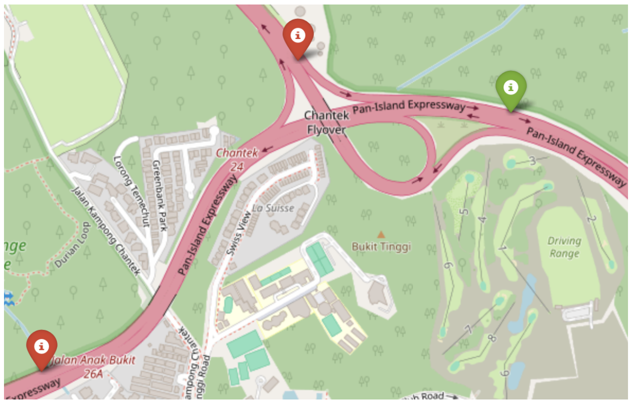


Figure 1: Snippet of the Singaporean dataset.

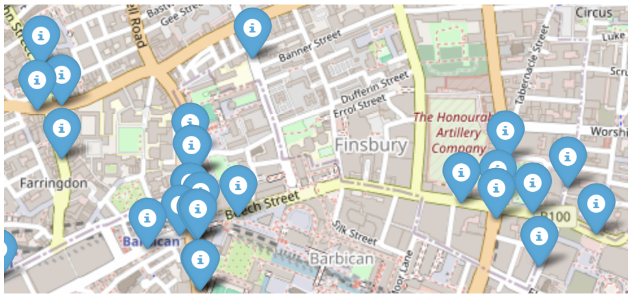
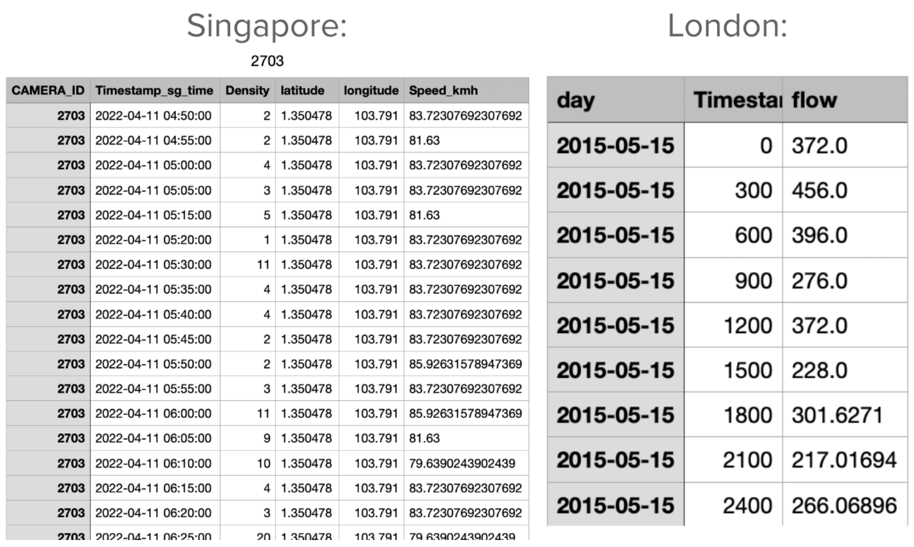


Figure 2: Snippet of the London dataset.

and used the UTD19[5] dataset, as it is closely similar to the data available from Singapore. Let's discuss those similarities. As can be seen in Figure 1 for Singapore and Figure 2 for London, both datasets have cameras/detectors in different locations around the city. Both datasets have data on timestamps every 5 minutes. The difference is, as can be seen in Figure 3 that the London dataset captures the flow, while the Singaporean dataset captures the density and velocity of certain locations. However the Singaporean dataset can be easily converted to also have flows using


Singapore:

2703

| CAMERA_ID | Timestamp_sg_time | Density | latitude | longitude | Speed_kmh |
|---|---|---|---|---|---|
| 2703 | 2022-04-11 04:50:00 | 2 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 04:55:00 | 2 | 1.350478 | 103.791 | 81.63 |
| 2703 | 2022-04-11 05:00:00 | 4 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 05:05:00 | 3 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 05:15:00 | 5 | 1.350478 | 103.791 | 81.63 |
| 2703 | 2022-04-11 05:20:00 | 1 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 05:30:00 | 11 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 05:35:00 | 4 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 05:40:00 | 4 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 05:45:00 | 2 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 05:50:00 | 2 | 1.350478 | 103.791 | 85.92631578947369 |
| 2703 | 2022-04-11 05:55:00 | 3 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 06:00:00 | 11 | 1.350478 | 103.791 | 85.92631578947369 |
| 2703 | 2022-04-11 06:05:00 | 9 | 1.350478 | 103.791 | 81.63 |
| 2703 | 2022-04-11 06:10:00 | 10 | 1.350478 | 103.791 | 79.6390243902439 |
| 2703 | 2022-04-11 06:15:00 | 4 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 06:20:00 | 3 | 1.350478 | 103.791 | 83.72307692307692 |
| 2703 | 2022-04-11 06:25:00 | 20 | 1.350478 | 103.791 | 79.6390243902439 |

London:

| day | Timestai | flow |
|---|---|---|
| 2015-05-15 | 0 | 372.0 |
| 2015-05-15 | 300 | 456.0 |
| 2015-05-15 | 600 | 396.0 |
| 2015-05-15 | 900 | 276.0 |
| 2015-05-15 | 1200 | 372.0 |
| 2015-05-15 | 1500 | 228.0 |
| 2015-05-15 | 1800 | 301.6271 |
| 2015-05-15 | 2100 | 217.01694 |
| 2015-05-15 | 2400 | 266.06896 |

Figure 3: Data comparison: Singapore vs. London

the formula:

$$q = k \cdot v$$

**Where:**

- $q$ = Flow (vehicles/hour)
- $v$ = Speed (miles/hour, kilometers/hour)
- $k$ = Density (vehicles/mile, vehicles/kilometer)

As flow is the product of Speed and Density[6]. Therefore the model presented in this report could be used for the Singapore dataset (or any dataset that has flows and corresponding timestamps) if it had a sufficient amount of cameras in crucial locations.

I decided to use 9 days of data from the London dataset, resulting in 2540 data points.

## Network

I decided to train a model using a simple flow network as it is easier to understand the principles used for building the network. The detectors used for this can be seen in Figure 4 (notice that Parliament Square Street and Great George Street are one-directional), and this map, for a clearer view of the network, can be converted into a flow network shown in Figure 5. It is a simple flow network with 4 flows. The model will try to predict the next timestamp flow (for each location) based on 12 previous timestamps (each 5 minutes apart, covering 1 hour). However, it will have no information of $f_{11}$.

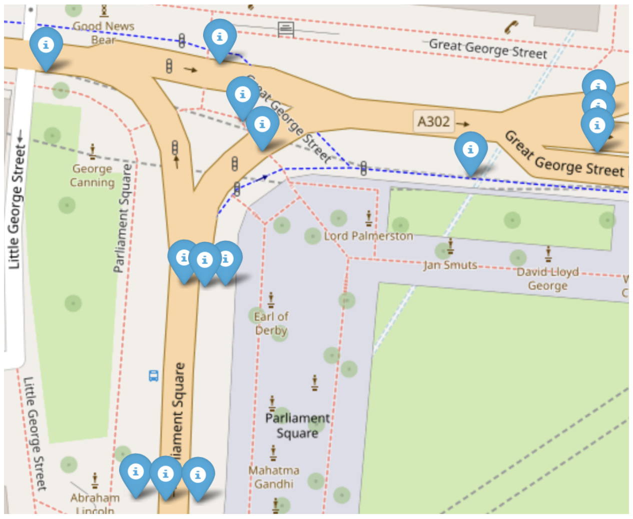


Figure 4: Detectors used for the network.

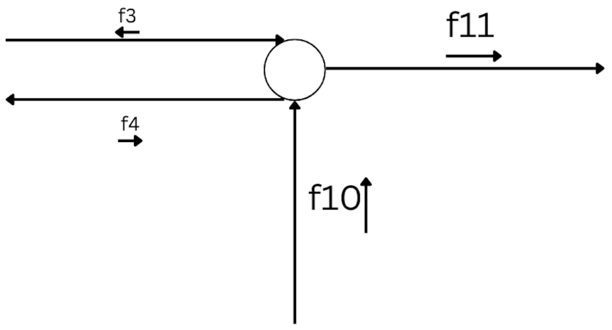


Figure 5: The flow network.

## Preprocessing

### Missing values

The UTD19[5] dataset contains no missing values for London in the days used for training and testing this model.

### Normalization

To normalize the data, Min-Max scaling was applied to all the values in the dataset. Min-Max scaler scales all the values in the dataset to the range 0 to 1, where 1 is the highest value in the dataset and 0 is the lowest. The formula is as following:

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

**Where:**

- $x$ = Original value
- $x'$ = Scaled value
- $x_{\text{min}}$ = Minimum value in the dataset
- $x_{\text{max}}$ = Maximum value in the dataset

This ensures that streets with vastly different ranges of flows do not disproportionately influence the model, as larger numbers could otherwise receive more importance. Min-Max scaling effectively addresses this issue.

### Train-test split

The data includes flow records from May 15, 2015, to May 23, 2015, 9 days in total. The test set is 12% and the train set is 88% of all the the datapoint. The following is the explanation of how were both sets obtained:

**Test set** The test set is created by selecting data points from specific time ranges (morning, mid-day, and evening) on different days. Morning includes data points between midnight and 8:00 AM for a specified day. Mid-Day includes data points between 8:00 AM and 4:00 PM for another specified day. Evening includes data points between 4:00 PM and midnight for a different specified day. The test set consists of morning on 2015-05-19, mid-day on 2015-05-16 (this is a Saturday) and evening on 2015-05-20.

By including a Saturday for mid-day traffic, the test set also accounts for weekday and weekend dynamics, providing a good evaluation of the model's ability to generalize to various scenarios.

**Train set** The train set consists of all data points excluding those in the test set.

### Sliding window

The LSTM expects the input data to have a three-dimensional shape, defined as follows:

- **Samples:** The total number of sliding windows extracted from the data.
- **Time Steps:** The number of timestamps in each window (12 in this case).
- **Features:** The number of variables per timestamp (e.g., 3 in this case for 3 flow values).

This shape allows the LSTM to process each sequence as a time series, while treating each timestamp as a vector of features.

To prepare the data for an LSTM, a sliding window approach was implemented. This method divides the time-series data into overlapping windows of past flows and the corresponding target flows, enabling the extraction of sequential patterns that can be used to predict future values. Both the train and test split were divided into sliding windows. Each window consists of 12 sequences, and each sequence consist of 3 flows per timestamp. One for $f_3$, $f_4$, $f_{10}$ (keep in mind we do not mention $f_{11}$ here, as this is the flow we want to predict without giving the model any information about it). Given that the flow values are recorded at 5-minute intervals, this results in a total window duration of 1 hour (12 × 5 minutes). For each sliding window (12 timestamps), there is a corresponding output is a flow vector with 4 entries: the flow value of each location at the 13th timestamp. That way we get data we can feed into out model. An example sliding window and its corresponding target can be seen in Figure 6. There are 3 columns, each column corresponding to $f_3$, $f_4$ and $f_{10}$. There are 12 columns in the sliding window, each one corresponding to one timestamp with the first one representing $t - 55$ and the last one corresponding to $t$. At the bottom the target output can be seen, a vector with 4 entries, each entry corresponding to the ground truth flow value for one of the flows at timestamp $t + 5$.

### LSTM

Unlike traditional neural networks, which treat inputs independently, LSTMs incorporate mechanisms to learn temporal relationships, making them ideal for time-series predictions, like traffic flow predictions where conditions at a given time depend on prior states.

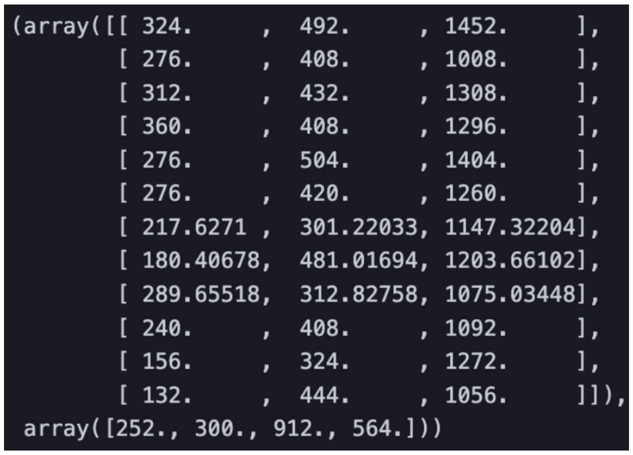

```
(array([[ 324.      ,  492.      , 1452.      ],
       [ 276.      ,  408.      , 1008.      ],
       [ 312.      ,  432.      , 1308.      ],
       [ 360.      ,  408.      , 1296.      ],
       [ 276.      ,  504.      , 1404.      ],
       [ 276.      ,  420.      , 1260.      ],
       [ 217.6271 ,  301.22033, 1147.32204],
       [ 180.40678,  481.01694, 1203.66102],
       [ 289.65518,  312.82758, 1075.03448],
       [ 240.      ,  408.      , 1092.      ],
       [ 156.      ,  324.      , 1272.      ],
       [ 132.      ,  444.      , 1056.      ]]),
 array([252., 300., 912., 564.]))
```

Figure 6: Exmaple sliding window.

**Architecture** The architecture of the LSTM model was designed to capture temporal dependencies in traffic data while preventing overfitting.

Key Components of the Model:

1. **Input Layer**: The model accepts sequential input data with a shape of the sliding window described earlier.

2. **LSTM Layers**:

    - The first LSTM layer contains 200 units.
    - A second LSTM layer with 150 units.

3. **Dropout Layers**:

    - Dropout layers with a rate of 0.4 are placed after each LSTM layer to reduce overfitting by randomly deactivating neurons during training.

4. **Dense Layers**:

    - Two dense layers are included, with 100 and 4 units respectively, to transform the extracted features into the final output.

5. **Output Layer**:

    - The output layer uses a linear activation function, making it suitable for regression tasks.

6. **Optimization and Compilation**:

- The model is optimized using the Adam optimizer with a learning rate of 0.01.
- A custom loss function will be described in the next sections.

### Physical Constraint

For the physical constraints I used the flow conservation equation (assuming both the location of $v_1$ and the location of $v_2$ have the same amount of lanes)[7]:

$$v_1 = v_2$$

**Where:**

- $v_1$ = Flow in region 1
- $v_2$ = Flow in region 2

To better explain this equation and how it can be used for our network, I will first provide an example on a smaller network. The example network can be seen in Figure 7. It can be observed that $f_1$ is being split into $f_2$ and $f_3$, a physical constraint for that network would look like this:

$$f_1 = f_2 + f_3$$

Now let us define the physical constraint for the network presented in Figure 5. We can see that $f_{11}$ "receives" flows from $f_{10}$ and $f_4$, however $f_{10}$ can either go to $f_{11}$ or $f_3$. When we combine it we end up with this equation:

$$f_{11} = f_4 + (f_{10} - f_3) \tag{1}$$

The model will learn to respect this equation. The output of $f_{11}$ is entirely dependent on this formula, while the other flows are also influenced by the error between the prediction and the ground truth.

### Loss function

It is a custom loss function that combines MSE and the physical constraints.

**MSE loss** The loss function calculates the Mean Squared Error (MSE) for three predicted flows:

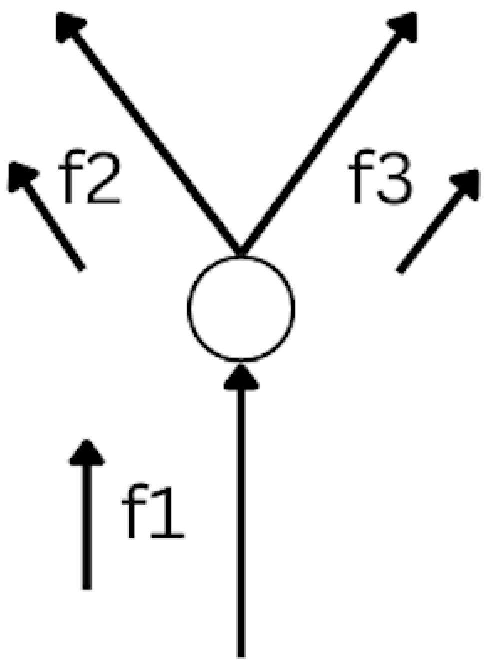


Figure 7: Small example network to explain the physical constraints.

- $mse_{f3}$ computes the MSE for $f_3$.
- $mse_{f4}$ computes the MSE for $f_4$.
- $mse_f 10$ computes the MSE for $f_{10}$.

These terms penalize differences between the true traffic flow values ($y_{\text{true}}$) and predicted values ($y_{\text{pred}}$) for individual locations.

**Physical loss** In order to use equation 1 in a loss function we need to transform it by subtracting $f_{11}$ from the LHS, we end up with this physical loss equation:

$$0 = f_4 + (f_1 0 - f_3) - f_1 1$$

The final physical loss consists of calculating the Mean Squared Error:

$$mse_p = f_4 + (f_1 0 - f_3) - f_1 1$$

That way the model can minimize the RHS by getting as close to 0 as possible (meaning it respects the physical constraints).

**Total loss** The total loss is combining the two components:

$$loss = mse_{f3} + mse_{f4} + mse_{f10} + mse_p$$

The model needs to learn to minimize each part of the total loss in order to minimize it.

### Metric

The Mean Absolute Error (MAE) was selected as the evaluation metric due to its simplicity and interpretability. MAE measures prediction errors in the same unit as traffic flow, making it easier to understand results. MAE is robust to outliers, ensuring occasional large deviations do not disproportionately affect the overall performance evaluation. Unlike other metrics, such as Mean Squared Error (MSE), MAE does not over-penalize large errors.

## 4 Results

The performance of the proposed traffic flow prediction model was evaluated using the test dataset described in the methodology section. This section presents the key results. The results are analyzed in terms of both numerical metrics and visualizations of predicted versus actual traffic flow.

### Model learning

As can be seen in Figure 8, the training and validation loss curves converge and remain close to each other after a few epochs, suggesting that the model is neither overfitting nor underfitting.

### Prediction Accuracy

The primary evaluation metric for the model was the Mean Absolute Error (MAE). The model MAE for the four flow variables on the test dataset can be seen in Table 1.

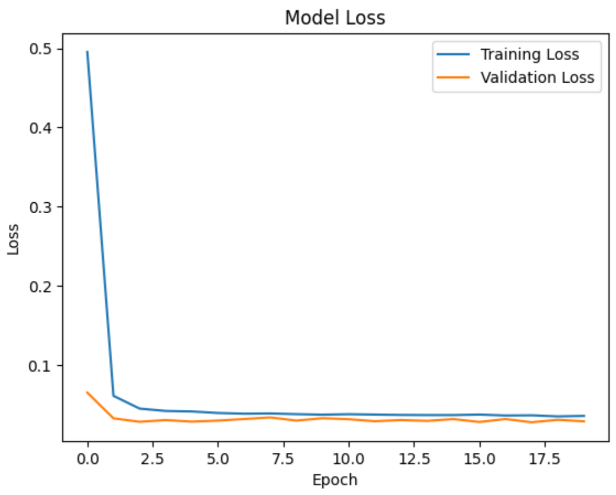


Figure 8: Epoch vs. Loss graph.

| Flow Variable | MAE (vehicles / 5 min) |
|---|---|
| $f_3$ | 5.195 |
| $f_4$ | 6.253 |
| $f_{10}$ | 11.175 |
| $f_{11}$ | 12.706 |

Table 1: Mean Absolute Error (MAE) for each flow variable.

These results indicate that the model predicts traffic flow decently, given that the flow values range from 0 to 1500. As expected $f_{11}$ has the highest error, as the model does not have any information provided about that location. However, it is still able to learn the physical constraint and quite accurately predict the future flow for that location.

## 4.1 Visual Comparison of Predictions

To better understand the model's performance, the predicted and actual traffic flows were plotted for several time windows. An example of the predictions for $f_{11}$ is shown in Figure 9.

We can see that for the morning of 19-05-2019 presented in Figure 9 the model generally predicts the traffic pattern, however fails to forecast the sudden increase at 7 am. For the mid-day of 16-05-2019 (Figure 10) the model in general predicts a lower than true flow value. For the evening of 20-05-2019, we can see that the data is noisier than for other dates, and the model fails to learn that noise.

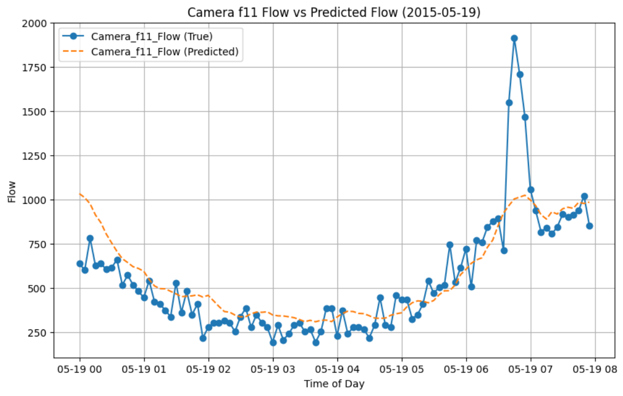


Figure 9: Predicted vs. actual traffic flow for Flow 11 for morning 19-05-2019.

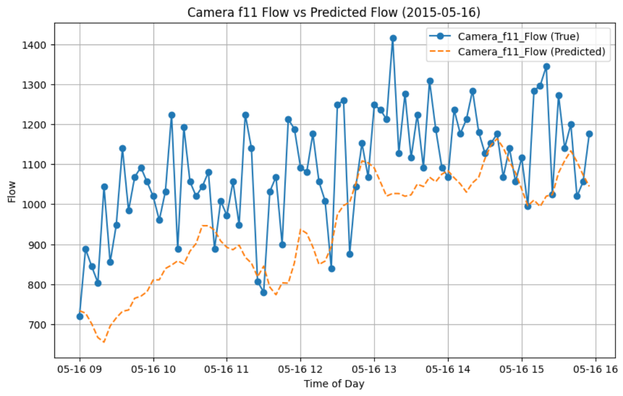


Figure 10: Predicted vs. actual traffic flow for Flow 11 for mid-day 16-05-2019.

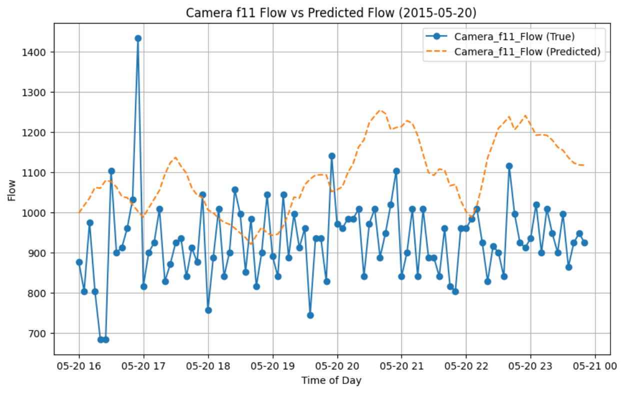


Figure 11: Predicted vs. actual traffic flow for Flow 11 for evening 20-05-2019.

The results for other locations can be seen in the appendix section.

## 5 Analysis

The model successfully learned the flow dynamics of the locations it was trained and did decently well on the location with no prior information. In this section I will discuss why did the model perform poorer for $f11$ and what can be done to potentially improve those results.

### Model underperformance for $f_{11}$

There are several potential reasons for model underperformance:

#### The Flow Conservation Equation

The conservation equation assumes a strict relationship between the input and output flows at a given intersection. While this principle is theoretically valid in ideal conditions, real-world traffic dynamics have complexities that might violate that simple relationship. The model assumes uniform flow distributions (when a flow is split into more flows, each flow has the same number of lanes), which might not hold in scenarios where traffic is unevenly split across multiple directions. For instance, in the London dataset, I lack access to images from the detectors to verify flow consistency (I only have the data). However, such verifications could be performed using the Singapore data, where images from each camera are available for inspection. Addressing these challenges in future iterations might involve relaxing the strict conservation assumptions (by reducing their weight in the loss function).

#### Noisy Sensors

Sensors may give incorrect readings due to hardware issues or environmental factors like rain (which happens quite often in UK) or glare. Noise can hide real patterns in the data, causing the model to perform poorer. If sensor noise is present, increasing the quantity of training data can help the model generalize better to unseen data.

#### Overfitting

The low training loss observed during model training could indicate overfitting, where the model learns patterns specific to the training data rather than to a general solution. A potential solution would be to monitor validation loss during training and stopping training when it stops improving, such a technique is called early stopping.

### Model usability

This model could be used to estimate the future flow values in both locations with data and locations lacking data. Potentially the predictions from locations lacking data could be used to expand the network, by using those predictions to make more predictions in a different unknown area and using those new predictions to make new predictions in a different unknown area, by iteratively applying this technique, this network could be expanded. The issue here is that such a network can not be tested as there is no ground truth for the unknown locations (the issue of Singapore data).

## 6 Conclusion

This report presents an approach to predicting traffic flow using a hybrid Long Short-Term Memory (LSTM) network combined with Physics-Informed Neural Networks (PINNs). Due to limitations in Singapore data, the method was evaluated on the UTD19 dataset. This method combines the LSTM's ability to learn time-based patterns with PINNs' use of physical rules to improve traffic predictions, even in areas with less data. The model achievied reasonable accuracy for traffic flow at known locations and acceptable predictions for a location with no prior data. The integration of the flow conservation equation into the loss function proved effective in enforcing physical consistency across predictions. This approach shows promise for scenarios where direct measurements are unavailable.

Future research could explore applying this model to datasets from other cities to validate its generalizability. Future improvements could involve adding other physical constraints. This approach shows promise for scenarios where direct measurements are unavailable.

## References

[1] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Nov. 2018.

DOI: 10.1016/j.jcp.2018.10.045. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999118307125 (visited on 12/13/2024).

[2] M. Usama, R. Ma, J. Hart, and M. Wojcik, “Physics-informed neural networks (pinns)-based traffic state estimation: An application to traffic network,” *Algorithms*, vol. 15, pp. 447–447, Nov. 2022. DOI: 10.3390/a15120447. [Online]. Available: https://www.mdpi.com/1999-4893/15/12/447 (visited on 12/13/2024).

[3] *Traffic flow dynamics 6. the lighthill-whitham-richards (lwr) model*. [Online]. Available: https://mtreiber.de/Vkmod_Skript/Lecture06_Macro_LWR.pdf.

[4] Y.-B. Su, X. Lü, S.-K. Li, L.-X. Yang, and Z. Gao, “Self-adaptive equation embedded neural networks for traffic flow state estimation with sparse data,” *Physics of Fluids*, vol. 36, Oct. 2024. DOI: 10.1063/5.0230757. [Online]. Available: https://pubs.aip.org/aip/pof/article/36/10/104127/3317738 (visited on 12/13/2024).

[5] *Utd19 - largest multi-city traffic dataset publically available*, Ethz.ch, 2017. [Online]. Available: https://utd19.ethz.ch/ (visited on 12/11/2024).

[6] C. to, *Fundamental diagram*, Wikipedia.org, Feb. 2007. [Online]. Available: https://en.wikipedia.org/wiki/Fundamental_diagram_of_traffic_flow#:~:text=All%20the%20graphs%20are%20related,points%20a%20best%20fit%20curve. (visited on 12/11/2024).

[7] L. Sheridan, *Fluids fluid dynamics*. [Online]. Available: http://nebula2.deanza.edu/~lanasheridan/4C/Phys4C-Lecture3-san.pdf.

## Appendix A: Additional Figures and Data

### Results for $f_3$, $f_4$, $f_{10}$

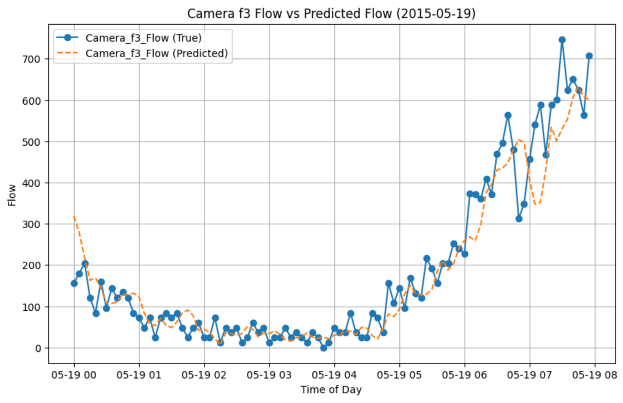


Figure 12: Predicted vs. actual traffic flow for Flow 3 for morning 19-05-2019.

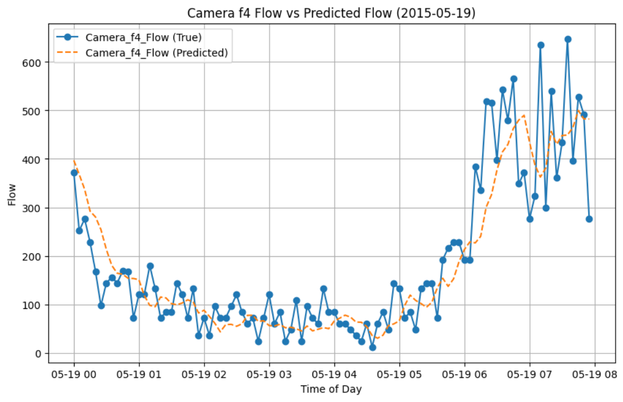


Figure 13: Predicted vs. actual traffic flow for Flow 4 for morning 19-05-2019.

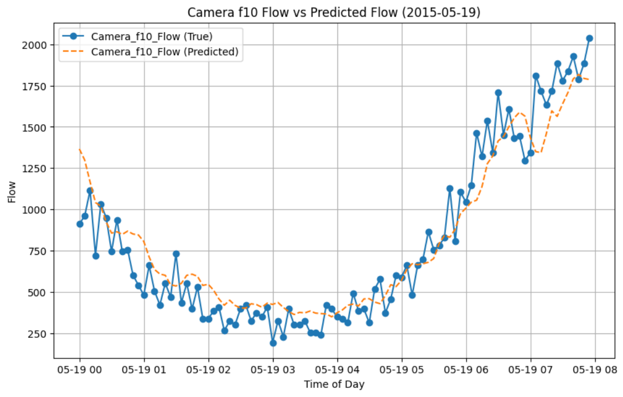


Figure 14: Predicted vs. actual traffic flow for Flow 10 for morning 19-05-2019.

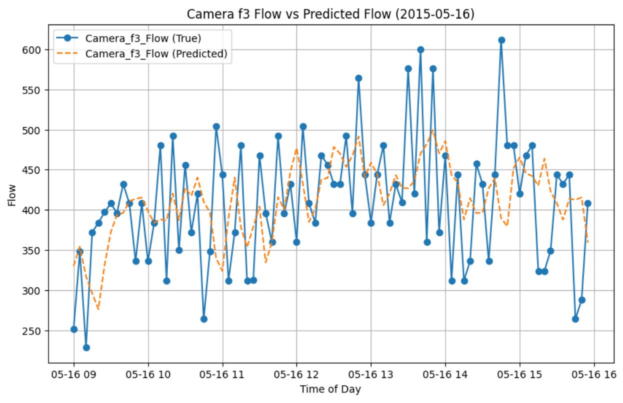


Figure 15: Predicted vs. actual traffic flow for Flow 3 for mid-day 16-05-2019.

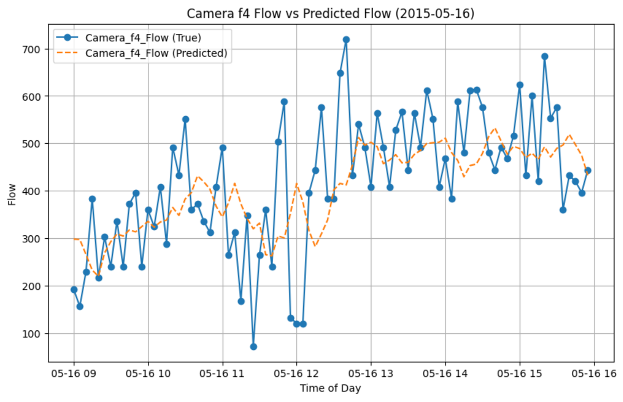


Figure 16: Predicted vs. actual traffic flow for Flow 4 for mid-day 16-05-2019.

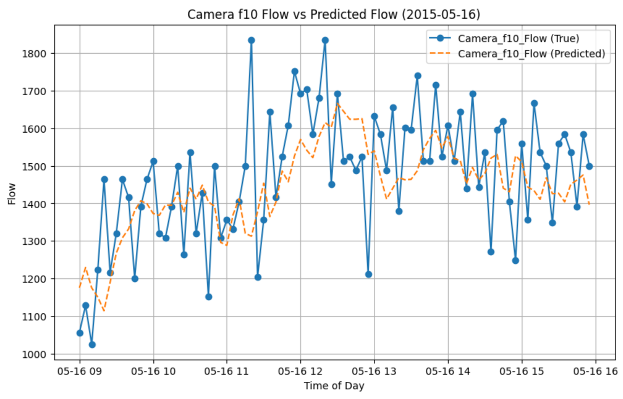


Figure 17: Predicted vs. actual traffic flow for Flow 10 for mid-day 16-05-2019.

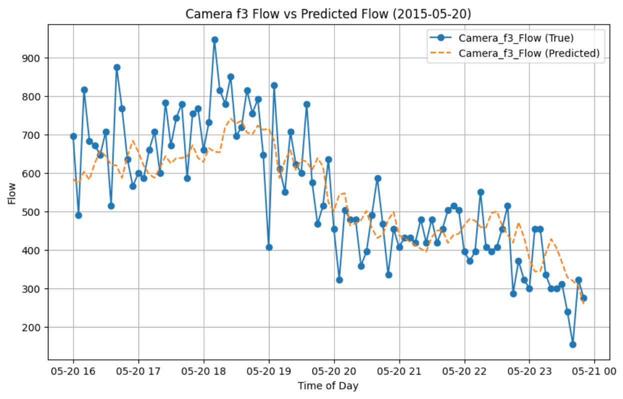


Figure 18: Predicted vs. actual traffic flow for Flow 3 for evening 20-05-2019.

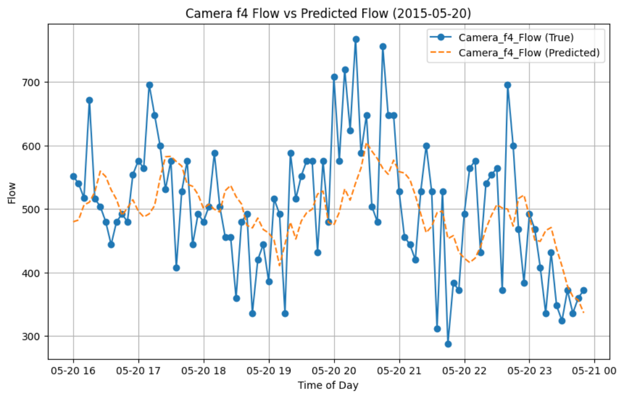


Figure 19: Predicted vs. actual traffic flow for Flow 4 for evening 20-05-2019.

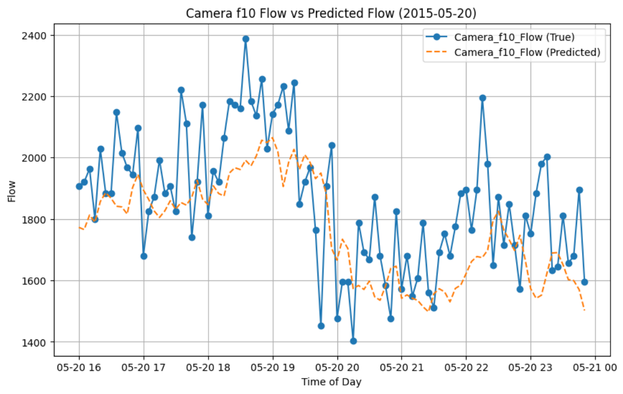


Figure 20: Predicted vs. actual traffic flow for Flow 10 for evening 20-05-2019.