Road Traffic Flow Prediction Using Aerial Imagery

by

Simran K. Pabla

S.B., Computer Science and Engineering, Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

Author . . . .
Department of Electrical Engineering and Computer Science
May 20, 2021

Certified by . . . .
Hamsa Balakrishnan
William E. Leonhard (1940) Professor
Associate Department Head of Aeronautics & Astronautics
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Road Traffic Flow Prediction Using Aerial Imagery

by

Simran K. Pabla

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Technological advancements have increased the potential and feasibility of widespread drone networks. Among other tasks, monitoring road traffic flow is a task well-suited for such networks. While real-time traffic flow estimation systems have been explored at length and exist as commercial services, these systems have limited spatial reasoning and suffer in accuracy when predicting future traffic conditions. To that end, graph neural networks can account for spatial patterns, and can more effectively capture the impact of a region's current traffic conditions on neighboring regions in the future. Our work builds on prior graph neural network architectures for traffic flow prediction. While current traffic prediction models are trained on ground-based data with limited features, we propose leveraging aerial traffic data to train spatiotemporal models with richer feature spaces. Our research makes contributions towards assembling a dataset from aerial footage and predicting traffic across a road network given aerial images from a small set of drones.

Thesis Supervisor: Hamsa Balakrishnan
Title: William E. Leonhard (1940) Professor
Associate Department Head of Aeronautics & Astronautics

Acknowledgments

I would like to thank my thesis advisor, Professor Hamsa Balakrishnan, for being an invaluable source of inspiration and wisdom throughout the course of this project. She has always encouraged me to pursue the problems that most excite me and generously offered her time and resources to support my learning and growth. I truly look up to her and the way she leads the DINaMo group.

I would also like to thank the DINaMo group for welcoming me into its warm, knowledge-driven culture. I am especially appreciative of Karthik Gopalakrishnan and Max Li for their mentorship when I first joined the group and their continued accessibility throughout my research journey. I can't thank Karthik enough for guiding me through the process of framing my project and sharing his accumulated domain knowledge without reservation. I owe a huge thanks to Siddharth Nayak as well for his valuable insights and support.

I couldn't be where I am today without the support of my friends and family. I'm grateful to my friends for always inspiring me, offering me sage advice, and bringing me so much joy and laughter. Through every Zoom call and countless late-night work sessions, they've helped me inch closer towards my goals. Special thanks to Stuti Vishwabhan, Shana Mathew, and Anj Fayemi for fueling my motivation and making this unique MEng experience so rewarding and memorable.

Above all, I'm grateful to my family for their unwavering support and for always instilling confidence within me. They've cheered me on through every accomplishment and setback, acted as my sounding board, and revitalized me with their contagious exuberance and boundless love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 2012, the Federal Aviation Administration forecasted that 30,000 drones would be operating in the US airspace by 2020 [1]. As of January 2021, the US government has registered over 1.7 million drones, blowing these predictions out of the water [2]. Remarkably, the market for commercial and recreational drones is continuing to grow at a rapid pace. Commercially, drones have become indispensable in the streamlining and advancement of operations across many sectors, including entertainment, agriculture, and delivery. In 2017, drone activity generated $1 billion in revenue in the United States, and it is estimated to achieve annual contributions of $31-$46 billion to the nation's GDP by 2026 [14].

It is clear that commercial and consumer interest in drones is soaring, and as the market heads towards the prospect of mass adoption, this technology offers greater potential for impact. Recognizing this, researchers at MIT and elsewhere have pursued the development of shared mobility platforms, intended to connect and establish a robust network between drones and their consumers. Pursuing a future where drones operate in coordinated fleets to provide various services, the primary objective for these platforms is to match drone hardware to clientele in need of such services. Leveraging the excess capacity that currently exists in the form of over 1.2 million, often idle, registered recreational drones, such platforms would offer the ability to coordinate routes and tasks with this finite resource network.

Over the last few years, significant progress has been made along several facets of the problem, including scheduling [5], as well as data collection and analysis for various drone sensing tasks. In this thesis, we explore drone-based traffic flow prediction as a promising application for a shared mobility platform. The primary objective for this work is to frame and lay the foundations for a project aiming to leverage aerial traffic footage and powerful spatiotemporal forecasting models to more accurately predict traffic conditions.

We first study how optical flow can be utilized to implement an enhanced speed estimation algorithm and curate a ground truth dataset. We then define and model an initial task that aims to predict traffic conditions across a road network given traffic data from a subset of its nodes, effectively simulating the constraints of traffic flow prediction across a network with a limited number of drones.

Chapter 2

Background

Traffic forecasting plays a key role in traffic patterns and management. Tracking metrics like average speed within a region on a daily, hourly, or even more granular basis can inform city planning and maintenance decisions and more effectively reduce congestion. This data also informs drivers in selecting efficient routes to reach their destinations.

Until recently, traffic forecasting services predominantly provided estimates based on traffic conditions collected via GPS and various ground-based sensor systems. With limited perception of future traffic conditions, these services offered limited accuracy when predicting traffic along routes and calculating estimated times of arrival. In the last decade, however, advancements in machine learning and technology have enabled significant progress in the traffic forecasting domain. With the availability of larger datasets and an explosive increase in computing power, studies have shifted attention towards more sophisticated learning methods.

2.1 Temporal Forecasting Methods

Initial advancements in traffic forecasting focused on enhancing models with temporal reasoning. As Figure 2-1 suggests, traffic patterns display strong periodicity [17]. This is not only limited to daily and weekly patterns; rather, we often capture oscillating behavior on the order of minutes as well, and this can be attributed to

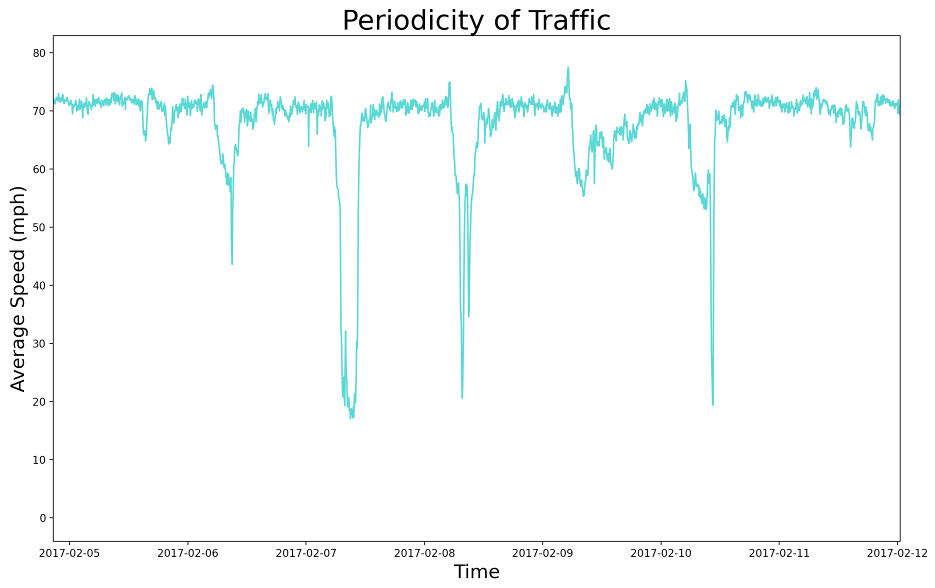

Figure 2-1: Average traffic speed shows periodic behavior on a daily and weekly basis.

the periodic ebbs and flows of traffic caused by traffic signals. Services like Google Maps have amassed years of traffic data and leveraged these temporal signals in their data to enable their predictive models to characterize dynamic traffic conditions more effectively [15].

Prior to the resurgence of deep learning, statistical and machine learning methods like the Autoregressive Integrated Moving Average (ARIMA) model [3, 4] and support vector regression [22] were leading methods for establishing temporal reasoning. In recent years, deep learning models for traffic forecasting have surpassed many of these approaches. Variants of the recurrent neural network, including long short-term memory networks and gated recurrent units, have especially shown great potential [20, 21].

Though these methods are powerful in capturing the overarching traffic patterns that we expect on a daily or weekly basis, they largely lack adaptability to real-time changes in traffic [13]. These methods are often limited in the event of unanticipated short-term occurrences like road closures and accidents, as well as long-term pattern shifts caused by factors like COVID-19. For increased adaptability, models should ideally account for spatial dependence as well.

2.2 Spatiotemporal Forecasting Methods

Spatial reasoning is valuable in capturing the propagation effect that occurs as traffic shifts upstream or downstream in a road network [23]. The impact of congestion is rarely isolated to one region, so identifying a representation that captures the connectivity between regions would be beneficial for a quicker response to evolving traffic behavior.

2.2.1 Graph Neural Networks

To characterize this spatial dependence effectively, several recent studies have proposed graph neural network (GNN) architectures [23, 6, 18]. Traffic-sensing regions and the connections between them can be represented as a graph $G = (V, E)$, where nodes $v_i \in V$ represent said regions and edges $e_{ij} \in E$ represent the road segments connecting them. For simplicity, we can define G as an undirected graph, as the impact of congestion typically propagates in both directions. G is accompanied by an associated feature matrix $X \in R^{N \times D}$ where N represents the number of nodes and D represents the number of features attributed to each node.

These studies then primarily define the problem as learning the function f that maps the traffic network G and feature matrix X to traffic metrics over the next T time steps:

$$[X_{t+1}, \cdots, X_{t+T}] = f(G; (X_{t-n}, \cdots, X_{t-1}, X_t)) \tag{2.1}$$

where n represents the length of the historical time series and T represents the length of the time series to be forecasted.

2.2.2 Graph Convolutional Networks

The core component underlying these models' spatial understanding is the graph convolutional network (GCN) [19]. GCNs take inspiration from convolutional neural networks, their image equivalents; however, graphs pose additional complexities, mostly stemming from the variability of their structures. Notably, GCNs are designed to handle graphs consisting of unordered nodes with varying degrees - complexities that are not pertinent to image convolution.

Making use of the node features and graph structure (i.e., the adjacency matrix), GCNs pass a filter over the nodes and their first-order neighbors to capture spatial relationships between the nodes. A single layer of the model can be expressed as

$$f(G; X_t) = D^{-\frac{1}{2}} G' D^{-\frac{1}{2}} X_t W \tag{2.2}$$

where D is the degree matrix, G' represents the adjacency matrix G with self-loops included for each node, and W represents the weight matrix for the layer. Stacking k GCN layers allows for information to be accumulated for the k-hop neighbors of each node.

Various GCN-based architectures have achieved remarkably accurate predictions of average speeds, with accuracies exceeding 90% and error margins in the range of approximately two to six miles per hour [23, 6, 18]. Architectures that combine GCN layers with temporal layers have been shown to perform particularly well.

2.3 Aerial Imagery Using Drones

The evolution of technology has a role to play in the improvement of forecasting methods as well. Previously proposed traffic forecasting models have been predominantly trained upon a small selection of popular traffic speed or volume datasets [16, 7, 21]. These datasets have been curated using inductive-loop vehicle detectors, taxi trajectory data, and GPS probe data, and as such, the subsequent feature matrices for these datasets are limited to the traffic metrics and time-based features, such as the day or month, for each sample. Recent studies have attempted to learn representations that can capture features such as number of lanes and road conditions [18]; however, no dataset directly captures these features.

The rise of commercial and consumer drone technologies presents us with the novel opportunity to improve traffic flow prediction models by expanding their feature spaces. Aerial footage can offer a visual dimension to traffic datasets and directly capture number of lanes, road quality and closures, road type, and other spatial characteristics.

2.4 Objectives

The vision for this project is to develop a traffic flow prediction model that can forecast traffic given current and past traffic data collected in the form of drone footage. Breaking down this idea into current and future objectives, we aim to

1. Leverage drone footage to build a dataset that can enable a richer feature space.
2. Predict traffic flow across all N nodes in the network given eyes (i.e., drones) that observe a subset K of those nodes.
3. Calculate the optimal placement of the K drones at each time step to maximize traffic flow information and reduce uncertainty within the network.

This thesis makes contributions towards the first two objectives and lays a foundation for future efforts towards the third objective.

Chapter 3

Data

3.1 Drone Footage

In prior work, a framework was devised and implemented for collection of video-based traffic data across a road network. Eleven locations were selected in the Boston-Cambridge area based on their proximity to four handpicked intersections, as shown in Figure 3-1. Selecting points on or near these intersections allowed for the possibility of studying how the impact of traffic signals on traffic flow propagates across the network.

Six drones were utilized to gather approximately one hour each of footage across the eleven different locations. The drone flights were synchronized to enable accurate comparison and analysis of traffic patterns in the collected footage. Distance per pixel was also tracked to account for variations in the height at which the footage was collected at each of these locations.

3.2 Additional Sources

Though useful for the framing of this project, one hour of footage across a traffic network was too limited for the prediction tasks we intended to pursue. To avoid being bottlenecked by the need for further drone-based data collection, we identified additional traffic datasets that could be utilized for the early stages of our prediction

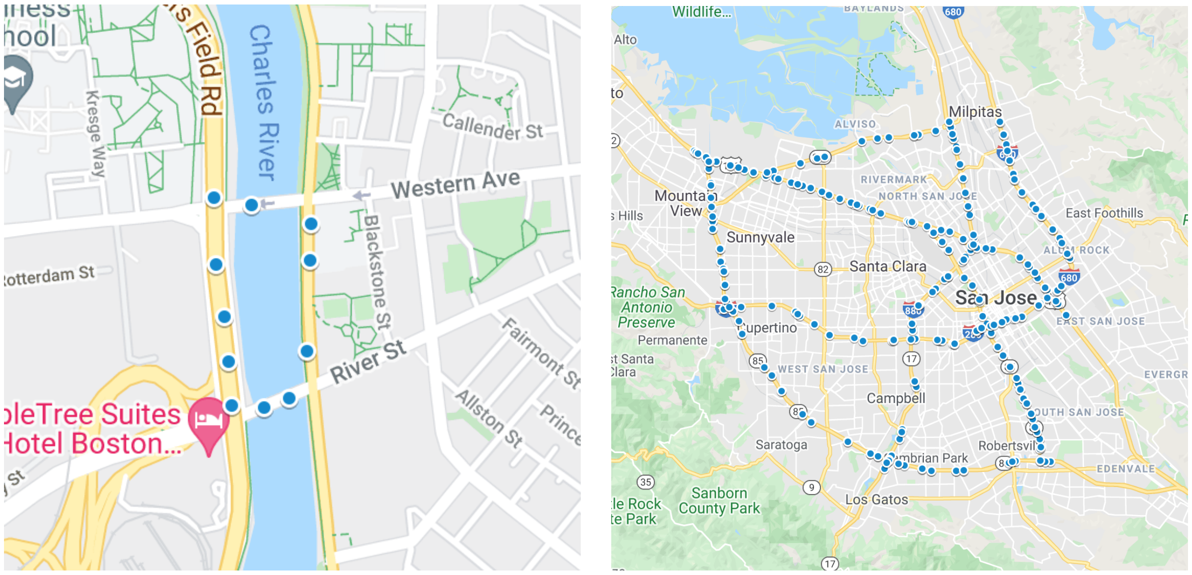

Figure 3-1: Traffic-sensing locations in drone footage (left) and PeMS-BAY (right) datasets.

efforts. We ultimately selected the publicly available PeMS-BAY dataset based on thoroughness and cleanliness of data [16, 17].

The PeMS-BAY dataset, provided by the California Department of Transportation, contains traffic data collected from January 1st, 2017, to May 31st, 2017, by 325 sensors positioned across the freeway system in the Bay Area. The data is collected in 5-minute intervals.

Chapter 4

Estimating Speed via Optical Flow

To assemble a complete, usable dataset from the collected drone footage, we first sought out a method for establishing ground truth, as the raw footage had no associated labels measuring traffic flow. Selecting average speed as our traffic metric of choice, we landed upon object detection and tracking as a means for estimating the speed within the recorded field of view.

4.1 Detection and Tracking

Vehicle detection and tracking is a common method for estimation of speed, density, and other traffic metrics. Upon detection, the vehicle is tracked until it is no longer in view. By tracking the trajectory of each vehicle C_i in the n-length list of currently tracked vehicles C, we have sufficient information to reasonably estimate its speed by approximating the distance D it travels and dividing this value by the time T taken to do so. The average speed a for the region is then calculated as follows:

$$D(C_i) = \sqrt{(e_x - s_x)^2 + (e_y - s_y)^2} * p_d \tag{4.1a}$$

$$T(C_i) = \frac{1}{r} * (f_e - f_s) \tag{4.1b}$$

$$a = \frac{1}{n} * \sum_{i=1}^{n} \frac{dist(C_i)}{time(C_i)} \tag{4.1c}$$

where $s = (s_x, s_y)$ and $e = (e_x, e_y)$ are the tracked starting and ending coordinates of vehicle C_i as it passes through the field of view, p_d represents distance per pixel, r represents video frame rate, and $f_e - f_s$ represents the length of C_i's trajectory (i.e. the number of frames in which vehicle C_i is present and tracked). It should be noted that this estimation simplifies all paths to a straight line between the starting and ending coordinates.

While variations of this speed estimation method are commonly utilized for video-based traffic surveillance, it is best suited for footage collected by fixed surveillance cameras. A key limitation of traditional object detection and tracking is the assumption of a static background. In the case of footage collected by drones and other mobile camera platforms, we expect background movement caused by planned platform movement as well as air turbulence. Developing vehicle detectors and trackers that subtract this noise is essential for accuracy of estimation.

4.2 Sparse vs. Dense Optical Flow

Determining this relative movement between the vehicles and the camera is the fundamental idea behind optical flow [10]. Leveraging optical flow, we can generate vector fields expressing the movement of points between consecutive frames.

The two primary optical flow methods are sparse and dense optical flow. As the name suggests, sparse optical flow generates flow vectors for a subset of pixels. These pixels are selected based on distinguishing factors. For example, state-of-the-art sparse optical flow methods select these pixels based on maximum difference in pixel intensity. Dense optical flow, on the other hand, generates flow vectors for the entire frame. Consequently, it achieves higher accuracy at the cost of being more computationally expensive than sparse optical flow.

Based on empirical analysis and further research, we concluded that dense optical flow is more applicable for our use case. The ideal method would distinguish stationary and moving features, enabling us to calculate and remove the stationary speed, which constitutes the speed contributed by drone movement. As evident in Figure

A-1, selecting features based on maximum difference in pixel intensity causes a mix of stationary and moving objects to be selected, especially in more complex fields of view that contain buildings and other prominent objects in addition to vehicles. The noise introduced by this reduces the utility of sparse optical flow for our problem space.

When using dense optical flow, we produce information for every pixel and can more effectively differentiate stationary and moving objects by running a clustering algorithm over the full-frame flow field. Dense optical flow is additionally promising because it allows for separation of features by flow direction. As seen in Figures B-2 and B-3, this allowed us to distinguish lanes, which should prove useful for future efforts to expand the feature space of our predictive models.

4.3 Dense Optical Flow Estimator

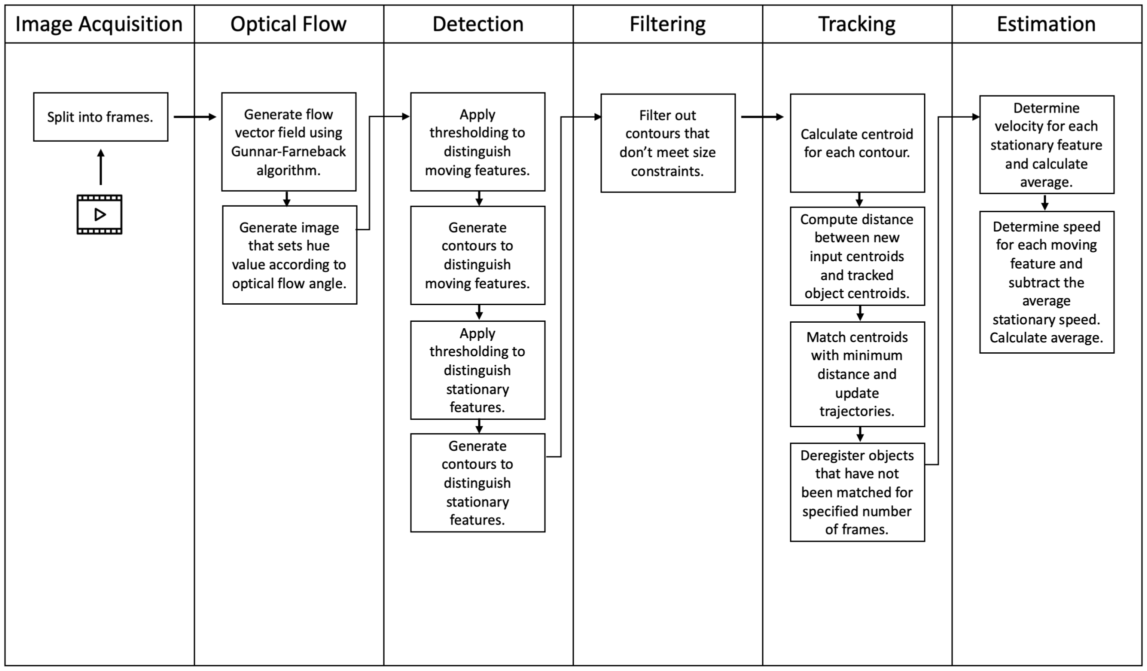
Image Acquisition
Optical Flow
Detection
Filtering
Tracking
Estimation
Split into frames.
Generate flow vector field using Gunnar-Farneback algorithm.
Generate image that sets hue value according to optical flow angle.
Apply thresholding to distinguish moving features.
Generate contours to distinguish moving features.
Apply thresholding to distinguish stationary features.
Generate contours to distinguish stationary features.
Filter out contours that don't meet size constraints.
Calculate centroid for each contour.
Compute distance between new input centroids and tracked object centroids.
Match centroids with minimum distance and update trajectories.
Deregister objects that have not been matched for specified number of frames.
Determine velocity for each stationary feature and calculate average.
Determine speed for each moving feature and subtract the average stationary speed. Calculate average.

Figure 4-1: Flow diagram for optical flow speed estimation algorithm.

Figure 4-1 depicts the optical flow pipeline that takes the drone footage as input and outputs the average speeds for a sequence of video segments.

4.3.1 Image Acquisition

The estimator takes in a traffic surveillance video as input as well as the interval at which average speed should be calculated. For our data, we elected to calculate the average speed for every 10-second segment in the video. Additional metadata that the estimator requires are the video's frame rate and the distance spanned per pixel.

We loop over the frames in the video stream and run them through the optical flow, detection, filtering, tracking, and estimation modules. Sample frames can be seen in Figure B-1.

4.3.2 Optical Flow

We apply the Gunnar-Farnebäck dense optical flow algorithm[1] to generate flow vector fields for the incoming frames. The algorithm takes two consecutive frames as input and computes the magnitude and direction of optical flow. We then visualize the magnitude and direction of flow using the value and hue, respectively, in the HSV color representation. Saturation is set to the maximum 255 value for maximum color intensity. Figure B-2 in Appendix B offers an example of this visualized flow vector field.

4.3.3 Detection

In the detection step, we independently detect the moving and stationary features in each frame that we intend to track and estimate the speed of. These features can be distinguished based on the magnitude and direction of flow, which are encoded in the HSV-frames. Moving features typically have greater magnitude of flow and variable angles that are defined by the lanes, while stationary features can be identified by a lower magnitude and likely a shared, fixed angle of flow within any given video segment. Leveraging these differences, we apply thresholding to the frames to isolate the two categories of features.

[1]See Section A.2 for background.

Beginning with the moving features, we first empirically establish a binary threshold above which pixels are set to white and below which pixels are set to black; hence, we distinguish the moving features and categorize everything else as *background*. Next, to extract the moving features, we apply a contouring algorithm to each frame. In doing so, we identify the boundaries separating white pixels from black and detect the location of each moving object in the frame. The example depicted in Figure B-3 highlights the results of this process.

We repeat the same steps for detecting stationary features, beginning with empirically establishing a threshold and then generating contours to detect locations.

4.3.4 Filtering

Because optical flow is not a noiseless method, we factor in a simple filtration step to only store the contours that are likely to represent moving or stationary features. This filtering occurs based on the size of the generated contours. We estimate size using bounding boxes for each contour and leverage width, height, and area to classify contours as moving, stationary, or error. The size constraints are similarly determined through empirical means. Stationary features are projected to be smaller on average, as the majority of detected stationary features are lane markings. As vehicles are typically of similar size, they can be distinguished with reasonable accuracy.

4.3.5 Tracking

Upon detection, we transition into the tracking module. All newly detected moving and stationary contours are stored as bounding boxes in two separate lists. We begin by using the bounding box coordinates to derive the centroid of each object in each list.

We pass each of these lists through an association algorithm that matches these new input centroids to the existing centroids. The algorithm computes the distance between every pair of the new and existing centroids and matches them based on minimal distance, as it is expected that the objects will not stray far from their current trajectories from frame to frame. After these centroids are matched, the new centroid is appended to its associated object's stored trajectory.

Any remaining input centroids are registered as new objects with newly initiated trajectories. Similarly, any existing centroids that were not matched are marked as having disappeared. Since an object may be momentarily out of sight (e.g., as a vehicle passes under a bridge) or incorrectly unmatched by the algorithm, we set a buffer corresponding to the number of frames during which an object can hold the *disappeared* status before it is officially de-registered. If the object reappears, we reset its status and count.

This procedure is run for both the moving and stationary features to update tracked data about the objects in view.

4.3.6 Estimation

When the specified number of frames for a segment has been processed (i.e., frames spanning ten seconds in our case), we estimate the speed over that segment.

$$a_s = \frac{1}{m} * \sum_{i=1}^{m} d_{S_i} * \frac{dist(S_i)}{time(S_i)} \tag{4.2a}$$

$$a' = \frac{1}{n} * \sum_{i=1}^{n} \left(\frac{dist(C_i)}{time(C_i)} + q * a_s \right) \tag{4.2b}$$

$$\text{where } q = \begin{cases} 1 & d_{S_i} \neq d_{C_i} \\ -1 & d_{S_i} = d_{C_i} \end{cases} \tag{4.2c}$$

The average speed a' for the optical flow estimator is derived from the average speed calculation for traditional detection and tracking methods as detailed in Equation 4.1c. Rather than calculating speed based on the vehicles alone, we now factor in the m-length list of currently tracked stationary features S. We first calculate the average velocity of the stationary features using the same framework set up in Equation 4.1c, but with the multiplicative factor d_{S_i} representing direction of motion for each stationary feature S_i. We use the stored trajectory for each object S_i to determine its direction d_{S_i}. The resulting average velocity of the stationary features, described in Equation 4.2a, represents the average velocity of the drone.

We then calculate the speed for each moving feature and adjust for the average stationary velocity by adding or subtracting based on the direction of motion. This is formalized in Equations 4.2b and 4.2c.

The resulting average across all moving features should theoretically hold little to no error caused by drone movement and should accordingly be more accurate than traditional methods.

4.4 Comparison of Methods

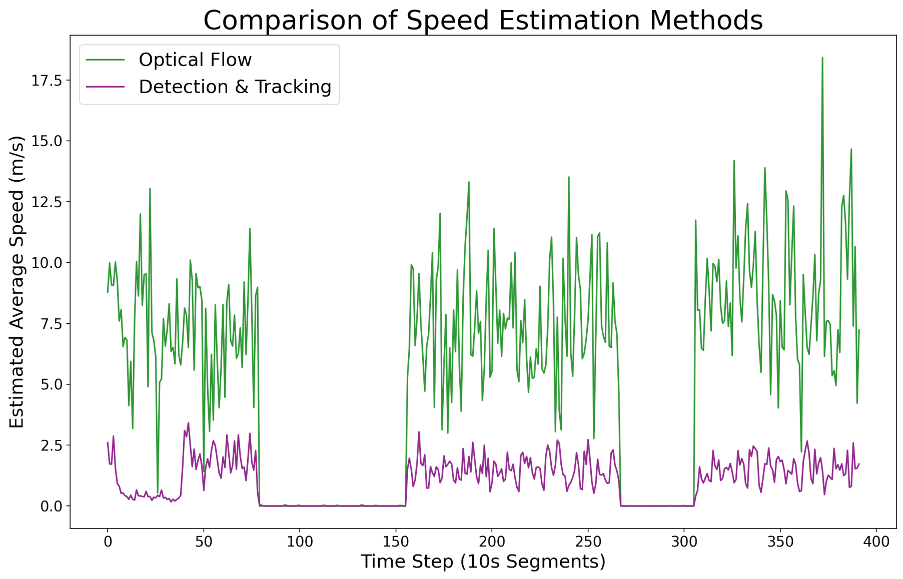

Figure 4-2: A comparison between our optical flow estimator and the traditional detection and tracking algorithm.

To evaluate the merit of the optical flow speed estimator, we analyze its performance against comparable methods. Because our aerial footage lacks ground truth, we compare the optical flow method's results against the previously implemented detection and tracking algorithm's results. This detection and tracking algorithm is described in Section 4.1. After passing an hour of drone footage through both estimators, we obtain the results shown in Figure 4-2.

The optical flow estimator consistently approximates higher average speeds than the previous method. Despite this disparity, it is promising to note that both estimators share similar peaks and troughs, and both accurately identify the segments during which the drone is charging and not monitoring any region, as evidenced by the time steps during which both estimators output an average speed of 0 meters per second.

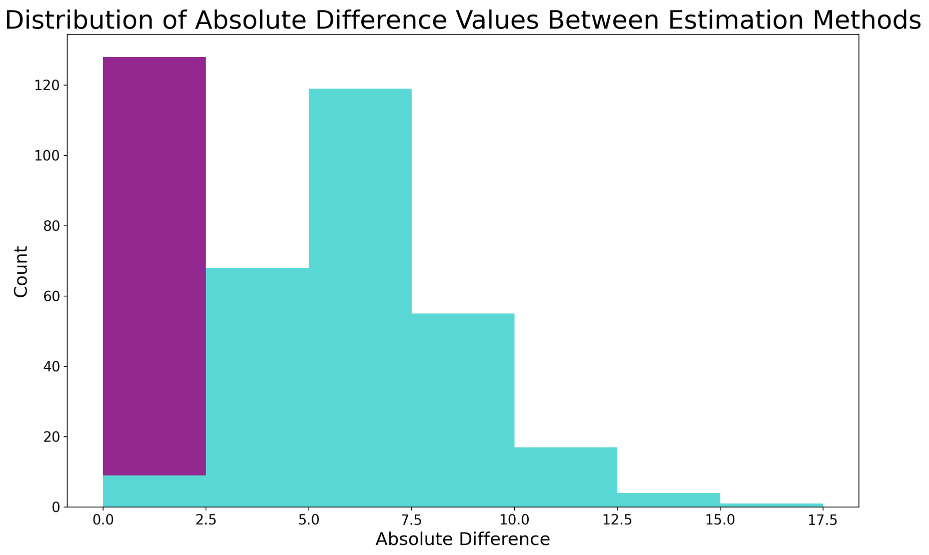

Figure 4-3: The distribution of values for the absolute difference between the optical flow algorithm and detection and tracking algorithm approximations. The purple bar represents the additional sample count that comes from the segments during which the drone is charging.

We calculate the absolute difference between approximations generated by both estimators and find that 31.38% of the optical flow estimates are within 2.5 meters per second of their detection and tracking counterparts, as seen in Figure 4-3. Noting that many of the samples that fall under this bin came from time segments during which the drone was charging and the average speed was 0, we re-calculate without these samples and find that the number of values in the bin drops to less than 2.5%.

This analysis suggests that the optical flow algorithm likely accurately identifies increases and decreases in average speed within a region over time. Without a reliable ground truth, speculating on the accuracy beyond a high-level trend comparison is ineffective. A deeper investigation into the discrepancies between the two estimators is warranted; regardless, we can likely rely on this dataset to equip a spatiotemporal model with useful signals on traffic dynamics, with the caveat that values may need to be scaled up or down.

Chapter 5

Prediction

In this chapter, we describe our contributions towards the second objective: traffic flow prediction across a road network. As mentioned in Section 2.2, several studies have explored the effectiveness of graph neural networks (GNNs) in predicting traffic flow. These studies design prediction tasks that leverage sensors at every location to forecast traffic conditions in those regions at future time steps, as described in Equation 2.1.

Unlike data collected by ground-based sensor networks, data collected by drone fleets is likely to be less comprehensive, especially if executed as envisioned on shared mobility platforms. A limited supply of drones may not be able to monitor every location within a road network at all times. One approach to resolving this is modifying the definition of a traffic sensing location such that a network consists of fewer regions; however, doing so introduces permanent uncertainty into the network, as we would lose granularity for each region.

We instead explore a dynamic setup where drones relocate to different locations at different time steps. Given N total locations, we place drones at a subset K of these locations and expect to relocate them in accordance with a preset scheme. Similar to the alternative approach, we face uncertainty at the unmonitored $N - K$ locations; however, we can reposition drones and distribute uncertainty across all locations over a given time period. With the optimal scheme, we can coordinate this repositioning to reduce uncertainty and effectively fill in the gaps in our knowledge.

Various imputation methods can be utilized to fill in our lack of knowledge at the $N - K$ unmonitored locations at any time step t. Most of these make use of data from past time steps, while some, such as the k-nearest neighbors method, incorporate some spatial reasoning by accounting for traffic conditions in neighboring regions.

Taking inspiration from the spatiotemporal methods applied to traffic forecasting, we leverage GNNs to predict traffic conditions at the unmonitored $N - K$ regions given data for the remaining K regions.

5.1 Methodology

Formalizing our prediction task, we establish a graph G to represent our road network, as described in Section 2.2.1. The associated feature matrix $X \in R^{N \times D}$ holds the features attributed to each node. Given that we only have data for K rows in this matrix, we fill the rows for the unlabeled $N - K$ nodes with a reserve value. We define the simplest problem as learning the function that maps the network G and incomplete feature matrix X'_t to the complete feature matrix with predicted values for those previously unlabeled $N - K$ nodes.

$$X_t = f(G; X'_t) \tag{5.1}$$

If we factor in temporal reasoning as well, we extend the definition to include data for past time steps as input.

$$X_t = f(G; (X_{t-n}, \cdots, X_{t-1}, X'_t)) \tag{5.2}$$

5.2 Experimental Framework

5.2.1 Data

In this study, we primarily utilize the PeMS-BAY dataset due to the scarcity of collected drone data, as explained in Section 3.2. The original dataset consists of samples collected by a 325-node sensor network across a freeway system in 5-minute intervals.

Utilizing this data, we build a 325-node graph G and associated feature matrices X'_t for every time step t in the dataset. At each time step, we randomly select the K nodes to be labeled, simulating drone movement across time steps.

5.2.2 Evaluation Metrics

To evaluate the prediction performance of our models, we use three metrics that quantify the difference between the prediction vector $\hat{Y}_t \in R^N$ and the target vector $Y_t \in R^N$, which contain the average speeds at the N nodes that make up the road network:

1. Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} (Y_t - \hat{Y}_t)^2} \tag{5.3}$$

2. Accuracy

$$Accuracy = 1 - \frac{\|Y_t - \hat{Y}_t\|}{\|Y_t\|} \tag{5.4}$$

3. Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{N} * \sum_{i=1}^{n} \left| \frac{Y_t - \hat{Y}_t}{Y_t} \right| \tag{5.5}$$

The training process minimizes error by utilizing mean squared error (MSE) as its loss function. We leverage the accuracy and MAPE metrics to compare results across experiments. Furthermore, these metrics are used by several previously cited traffic flow prediction efforts [23, 6, 18], which enables effective comparison across studies.

Parameter	Values
Batch Size	64 - 256
Epochs	300 - 1500
Learning Rate	0.0005 - 0.002
Hidden Layer Size	4 - 256
Normalization	True/False
Masked Loss	True/False

Table 5.1: Valid values for model parameters.

Parameter	Values
Window Size	30 min
	120 min
Data Sampling Frequency	5 min
	10 min
	30 min
	60 min

Table 5.2: Additional parameters for spatiotemporal models.

5.2.3 Model Configuration

The hyperparameters for the evaluated suite of spatial and spatiotemporal models include batch size, training epochs, learning rate, and the size of hidden layers. Additionally, we assess the value of data normalization and any benefits to masking the values of the K labeled nodes in the output.

The valid values we test for these parameters are listed in Table 5.1. Additional parameters that are specific to spatiotemporal models are listed in Table 5.2.

To identify the optimal settings for these parameters, we run a Bayesian search [9]. This optimization method models the relationship between select parameters and a chosen metric with the goal of selecting configurations with the highest probability of improvement over the current setting. These updates at each step are done according to Bayes' rule. For our experiments, we optimize for validation accuracy using the metric described in Equation 5.4.

Based on our search, we set the learning rate to 0.001, batch size to 64, and the hidden layer size to 256. These results, as shown in Figure 5-1, fall in line with what we expect; we land upon a low batch size, as larger batch sizes tend to lead to poor generalization [12]. Analyzing the feature importance values for each hyperparameter, we find that the hidden layer size is the most important parameter for the spatial and spatiotemporal models in our suite. This finding is corroborated by previous studies on spatiotemporal traffic models [23].

Through our experiments, we additionally find that network training converges faster with the support of data normalization. We set the lower and upper bounds

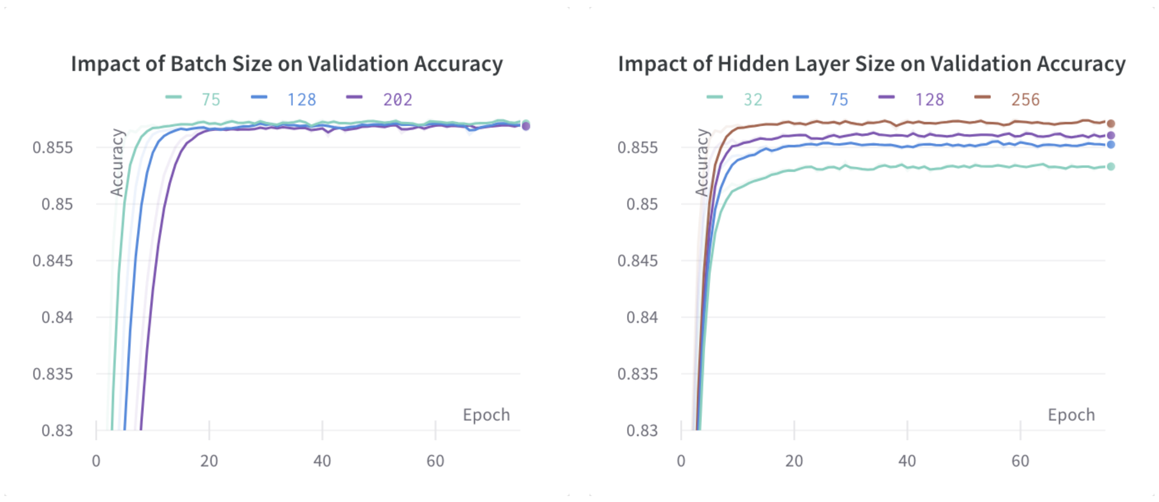

Figure 5-1: Comparison of prediction performance under different batch sizes (left) and hidden layer sizes (right).

for our dataset to 0 and 100 miles per hour, respectively, and utilize these bounds to apply min-max normalization and rescale the input and labels to the range [0, 1]. As Figure 5-2 illustrates, the model converges in fewer epochs when the data is normalized. On average, 50-100 epochs of training is sufficient for spatial models if the data is normalized, while the raw data requires 400-500 epochs. Similarly, spatiotemporal variants require 300-500 epochs if the data is normalized and 800-1000 epochs if not. As such, normalization offers a significant boost in training efficiency.

Finally, we find that masking the K labeled nodes in the output is critical for model accuracy. Training our models to both carry forward the average speeds at the K labeled nodes and predict the average speeds at the unlabeled $N - K$ nodes introduces unnecessary noise. We can avoid this by only running the predicted values for the $N - K$ unlabeled nodes through the loss function and subsequent learning pipeline. We can see in Figure 5-3 that the model fails to learn without masking; there is a significant reduction in error once this noise is eliminated.

5.2.4 Baseline Methods

We compare the performance of the spatiotemporal models with a GCN model as described in Section 2.2.2 and by Equation 5.1. Applying a GCN to our problem

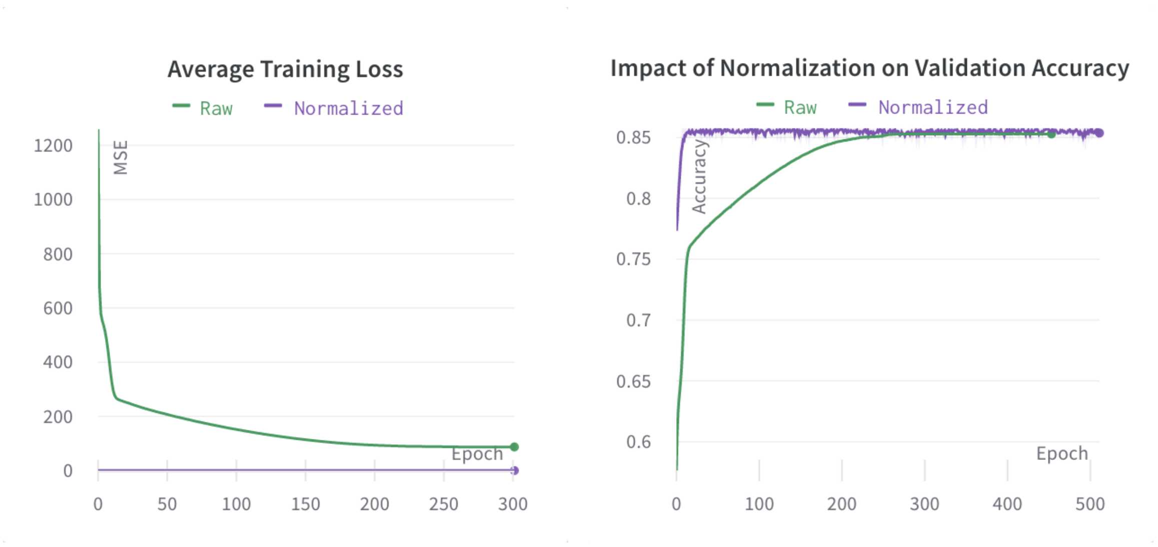

Figure 5-2: Data normalization contributes to network training efficiency.

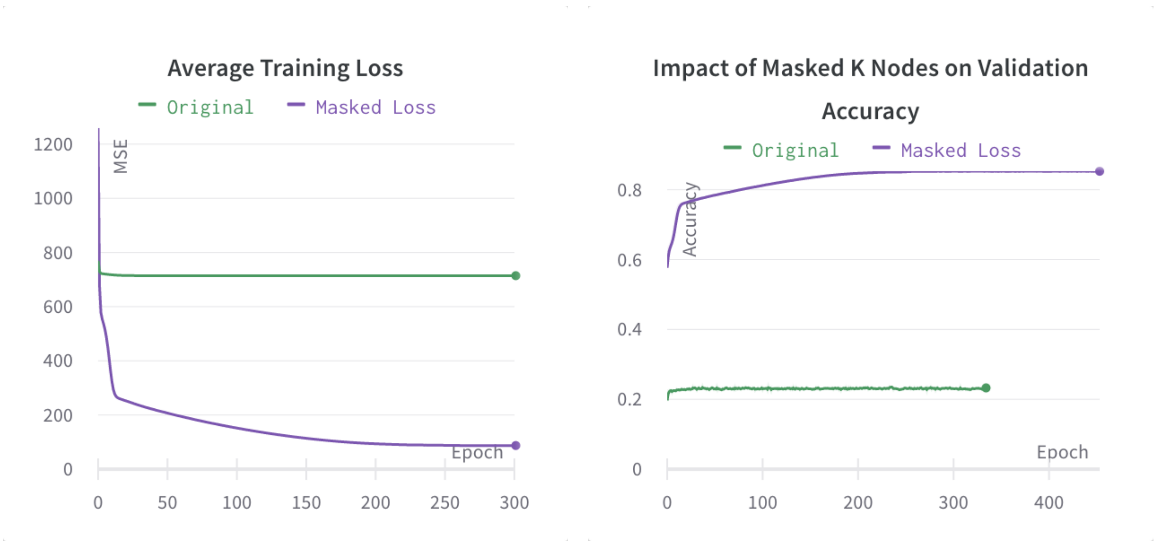

Figure 5-3: Masking the labeled nodes in the loss function is critical to model learning.

space helps us to confirm the merit of spatial reasoning in capturing traffic flow trends in real-time, and comparing it against spatiotemporal models enables us to additionally identify the value of equipping traffic flow prediction models with both spatial and temporal reasoning. The GCN model used for the experiments in this study is a 2-layer model.

Additional benchmarks we utilize for each spatiotemporal model applied to our unmonitored node prediction problem are the performance metrics of its forecasting equivalent. Each of the model architectures selected for our experiments has been previously applied to the traffic forecasting problem defined by Equation 2.1 and trained on comparable datasets [23, 24]. We can analyze the results of our experiments by determining whether or not they are within a reasonable range of the results generated for the forecasting problem. We can extend our benchmarks to include other comparable traffic flow prediction models as well.

5.2.5 Temporal Graph Convolutional Network

The core spatiotemporal model we build our analysis upon is the temporal graph convolutional network (T-GCN) [23]. The T-GCN architecture consists of a GCN for spatial reasoning and gated recurrent unit (GRU) for temporal reasoning. Beginning with Equation 2.2 for the graph convolution process, the T-GCN operations are defined as follows:

$$u_t = \sigma(W_u[f(G, X_t), h_{t-1}] + b_u) \tag{5.6}$$

$$r_t = \sigma(W_r[f(G, X_t), h_{t-1}] + b_r) \tag{5.7}$$

$$c_t = \tanh(W_c[f(G, X_t), (r_t * h_{t-1})] + b_c) \tag{5.8}$$

$$h_t = u_t * h_{t-1} + (1 - u_t) * c_t \tag{5.9}$$

In summary, u_t and r_t are the update and reset gates, h_{t-1} refers to the output at time $t-1$, and W and b represent the weights and biases.

5.2.6 Attention Temporal Graph Convolutional Network

We also apply a variant of the T-GCN model, the attention temporal graph convolutional network (A3T-GCN) to the unmonitored node prediction problem. The A3T-GCN architecture simply channels the hidden states produced by the T-GCN operations defined in Equations 5.6-5.9 through an attention model. The attention mechanism is intended to capture global trends about the traffic state to supplement the local trends captured by the GRU.

5.3 Results

5.3.1 T-GCN Experiments

We first compare the performance of the T-GCN model with varying window sizes and data sampling frequencies, as defined in Table 5.2. Varying the window size enables us to analyze the impact of including short-term or long-term input signals. We vary the data sampling frequency to analyze the optimal granularity of data; the original PeMS-BAY dataset is collected at 5-minute intervals, and we resample this data at three additional intervals to study this effect.

Sampling Frequency (min)	Window (min)	RMSE	Accuracy (%)	MAPE (%)
5	30	6.73	89.37	9.08
5	120	6.17	90.27	8.31
10	30	6.74	89.39	9.10
10	120	6.34	90.01	8.55
30	30	7.60	88.04	10.25
30	120	6.92	89.08	9.68
60	60	7.89	87.65	10.75
60	120	7.60	88.07	10.58

Table 5.3: Prediction results produced by the T-GCN model with different sampling frequency and window size configurations.

Table 5.3 shows the results of these experiments. It can be seen that the model produces the best results when a 2-hour window is used in conjunction with a 5-minute data sampling rate. In other words, larger quantities of more granular data are optimal for learning. The positive results generated by lower sampling frequencies suggest that variations in traffic on the order of minutes offer valuable signals. Traffic light systems may be a contributing factor to this. Additionally, the larger window size consistently leads to better performance. Equipping the model with more data may be especially effective in capturing long-term traffic trends such as rush hour. The optimal settings would likely differ for a dataset with more nodes at local streets.

5.3.2 Comparison Against Baseline Models

Model	RMSE	Accuracy (%)	MAPE (%)
GCN	9.06	85.71	13.50
T-GCN	6.17	90.27	8.31
A3T-GCN	6.11	90.29	8.30
GRU*	4.00 - 5.22	72.49 - 91.09	N/A
GCN*	5.66 - 7.79	61.07 - 86.73	N/A
T-GCN*	3.92 - 5.13	73.06 - 91.27	N/A
A3T-GCN*	3.90 - 5.09	73.18 - 91.33	N/A
DRCNN*	2.95 - 4.74	N/A	2.90 - 4.90
GaAN*	5.24 - 7.65	N/A	6.99 - 10.62

Table 5.4: Prediction results generated by our suite of models and other baseline methods. Models marked with * are trained to solve the traffic flow prediction problem detailed in Section 2.2.1 and by Equation 2.1. We provide performance ranges for models not trained on the PeMS-BAY dataset to offer a high-level comparison.

Table 5.4 contains the results of our suite of models and the baseline models identified in Section 5.2.4. The T-GCN and A3T-GCN models visibly outperform the GCN model under all evaluation metrics, and this showcases the importance of modeling temporal reasoning in addition to spatial reasoning.

When comparing these models to results from past studies that addressed the original traffic forecasting problem defined in Equation 2.1, we compare against performance ranges for the models from past studies. These models have been tested on multiple data sources, and the varying results generated by each of those sources form the ranges. We choose to compare our evaluation metrics to ranges due to the disparities caused by our models being trained on a different data source.

The spatiotemporal models deviate from the target value by approximately six miles per hour on average. This error can be further reduced if K is selected more intentionally, which will be the case in a practical setting. Rather than selecting K random nodes at each time step, we can update our dataset to select the nodes to be monitored in a more realistic fashion; by selecting randomly within the k-hop neighbors of each monitored node, we mimic the limitations on range of movement for the drones within a given time step. We can further build upon this by tracking prediction uncertainty at nodes to inform optimal drone placement; by more densely monitoring regions with or near greater uncertainty, we can develop a more accurate understanding of conditions within the road network.

A 90% accuracy rate with clear scope for further improvement is an extremely promising result and showcases the potential these models possess to surpass the traffic flow prediction methods underlying most commercial real-time estimation systems.

Chapter 6

Prediction Using Aerial Imagery

In the previous section, we establish the effectiveness of spatiotemporal graph network architectures in filling in missing traffic data across a road network. We can capture the variable nature of traffic flow and predict traffic conditions at the unmonitored $N - K$ locations with remarkable accuracy.

While data scarcity prompted us to leverage the PeMS-BAY dataset for our analysis, we ultimately intend to train our models on aerial footage. As described in Section 2.3, aerial footage offers the opportunity to expand the feature space we equip our models with.

As a final experiment, we utilize the dataset assembled via our optical flow speed estimator to train the T-GCN model. The footage collected at eleven different locations is labeled by the optical flow algorithm, and the resulting dataset is preprocessed to be channeled through the T-GCN training pipeline.

As described in Section 3.1, six drones are used to capture data at these eleven locations, and as a result, we have incomplete knowledge of traffic patterns across the road network at any given time. For the purpose of running a controlled experiment with credible evaluation procedures, we mask the labels of an additional monitored node and define a scaled down task of predicting traffic at this fixed node given data from a combination of the remaining ten nodes.

There are some clear limitations to the aerial footage dataset. While the PeMS-BAY dataset consists of over 52,000 datapoints, collected at consistent intervals over a wide network, the aerial footage produces 400 datapoints that represent a much smaller road network. As such, it is critical to reduce the complexity of the prediction task. We do so by fixing $N - K$ instead of masking nodes randomly at each time step, as well as adjusting some of the hyperparameters. We set batch size to 32 and hidden layer size to 16. After analyzing the minimum and maximum speeds in the aerial data, which is not a uniform, freeway system dataset, we also modify the maximum speed value to 50 miles per hour, which impacts the normalization operation. We acknowledge that there are multiple sources of noise within the dataset. The filmed footage consists of certain patches where the camera is not pointed at the region of interest. Additionally, the optical flow algorithm introduces some noise when generating our ground truth.

Sampling Frequency (min)	Window (min)	Accuracy (%)	MAPE (%)
0.17	1	61.05	38.12
0.17	5	53.55	66.20
0.17	10	45.20	55.60

Table 6.1: Prediction results produced by the T-GCN model when trained on aerial footage.

Despite these limitations, the results are promising. Interestingly, smaller window sizes contribute to higher accuracy, unlike the results produced by the PeMS-BAY dataset. This may be attributed to a few different reasons. First and foremost, the six drones that generated the aerial footage had to recharge after every 12-20 minutes of flight time. It is highly likely that a larger window size would result in more datapoints where a significant portion of the drones were out of commission, and consequently, most nodes were unmonitored. A narrow window size, on the other hand, may have produced more meaningful data. Another point to consider is the difference between the road network captured in the aerial footage and the freeway system captured in the PeMS-BAY dataset. Freeway traffic may be more heavily influenced by long-term traffic trends than traffic within internal roads, which is likely dictated by the short-term, cyclical patterns of traffic light systems. A larger window size may introduce noise and prevent the model from clearly identifying some of these short-term signals. Finally, a larger window size reduces the number of valid datapoints. A ten-minute window size reduces the dataset by 60 samples – a non-trivial 15% of the original dataset.

More aerial data and a cleaner framework for collecting it would address most of the limitations listed above and improve the model's ability to learn and predict. Additionally, tapping into the additional features that can be extracted from the aerial footage may significantly improve performance. These are some of the many potential next steps that we outline for this research in Section 7.2.

Chapter 7

Conclusions

In this work, we made progress towards multiple facets of the drone-based traffic flow prediction problem. Revisiting the objectives defined in Section 2.4, the overarching purpose of this project is to build a richer traffic flow dataset using drone footage, develop a model that can predict traffic conditions across a network given data at a subset of its nodes, and devise an approach for calculating the optimal placement of drones across the network at each time step such that uncertainty is minimized.

To address the first objective, we developed an optical flow-based speed estimation algorithm to assemble a dataset from the previously collected aerial footage. We identified potential features that could be extracted from this data to create a richer feature space for our prediction models, and leveraging optical flow, we made some progress towards lane detection.

Significant contributions were made towards the second objective as well. After exploring past efforts to apply spatiotemporal models to traffic forecasting, we defined a new prediction task: predicting traffic conditions at the unmonitored $N-K$ locations given data from drones at the remaining K locations. We implemented a suite of spatial and spatiotemporal models, identified optimal hyperparameter settings for these models, and compared their performance to several baseline models. On the whole, we achieved model performance surpassing 90% accuracy, and we identified next steps that will likely further reduce the error percentage.

Finally, we initiated efforts to combine the results of the first two objectives by configuring a spatiotemporal model to be trained on the newly assembled aerial traffic dataset. Though data limitations prevented us from achieving performance on par with the models trained on the PeMS-BAY dataset and models from past studies, we produced valuable insights on the data collection framework and potential sources of error.

These efforts laid out the foundation for this research and suggested promising directions for further investigation.

7.1 Future Work

We have several recommendations regarding next steps for each of the three primary objectives.

To build upon our contributions to the first objective, we suggest further evaluation of the optical flow speed estimation algorithm. Securing a reliable ground truth dataset for evaluation would help to confirm the performance of our estimator. Additionally, the estimator should be tested on a diverse range of terrains and road network types. The current estimator may not generalize, especially due to its reliance on specific features like lane markings. Although lane markings are likely to be present in most footage, additional indicators could be incorporated for generalizability.

There is also potential for further refinement of the implementation of our estimator. It is currently bottlenecked by the Gunnar-Farnebäck computation step, which alone takes 1.5 seconds per frame. Two potential solutions include optimizing the implementation of this algorithm or exploring how to run the estimator on a subset of the input video's frames while retaining sufficient information. The Gunnar-Farnebäck algorithm is inherently computationally expensive because it is a dense optical flow algorithm. While sparse optical flow proved to produce unfavorable results in this study, some recent work has had more success [11]. Investigating this work and experimenting with parameter tuning for such algorithms may be beneficial.

The second objective would be primarily advanced by significant efforts to grow the aerial dataset. Data collection using a refined data collection framework and data augmentation can be conducted to collectively assemble a dataset comparable in size to PeMS-BAY and other datasets assembled using ground-based sensors. Once an adequate amount of data has been collected for model learning, we suggest dedicating efforts towards expansion of the feature set. Features such as number of lanes, road quality and closures, and road types can be extracted from the aerial footage for this purpose. This will enable the model to effectively leverage the rich information encoded within video data. Finally, more realistic selection of the K nodes at every time step would be useful in mimicking the practical setup for drone-based traffic sensing. Instead of random selection, constraints should be placed on the range of movement for drones at every time step.

Finally, the contributions in this work are intended to serve as a foundation for future research towards the coordinated planning of drone locations for aerial imagery. With optimal drone placement, the resource limitations of this technology can be accounted for, and the full potential of shared mobility platforms can be realized.

Appendix A

Optical Flow Methods

A.1 Lucas-Kanade Method

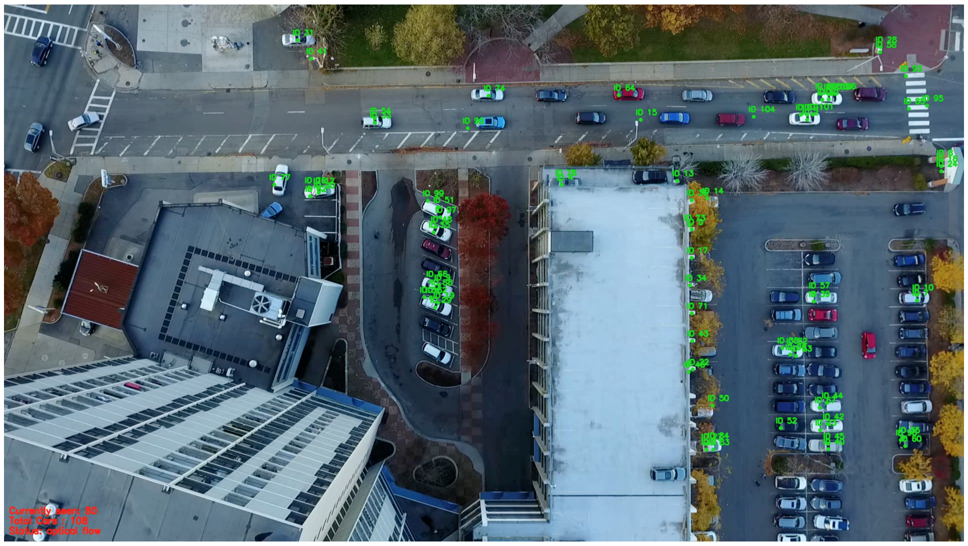

Figure A-1: Feature detection using sparse optical flow.

The Lucas-Kanade method is a sparse optical flow algorithm that estimates motion between consecutive frames. This method assumes that objects aren't displaced drastically between consecutive frames and that the pixel intensities of objects won't change from frame to frame. Relying on these assumptions, it builds windows around detected features of interest and assumes all pixels within these windows share the same magnitude and direction of flow. These pixels can be used to generate a system of equations to solve for the displacement.

Figure A-1 shows an example of the drawbacks of the Lucas-Kanade method; due to its reliance on selecting features based on maximum difference in pixel intensity, both moving and stationary features are selected and challenging to differentiate.

A.2 Gunnar-Farnebäck Method

The Gunnar-Farnebäck method is a dense optical flow algorithm. Unlike Lucas-Kanade and other sparse optical flow methods, which only track certain points of interest, dense optical flow considers every pixel and compares each of them from frame to frame. Similar to Lucas-Kanade, this method generates windows around each pixel. It approximates these windows through polynomial expansion and then estimates flow vectors from the coefficients of this expansion. Specific operations can be found in Farnebäck's original paper [8].

Appendix B

Optical Flow on Aerial Data

The following visuals depict the results of different modules within the dense optical flow speed estimation pipeline.

B.1 Image Acquisition

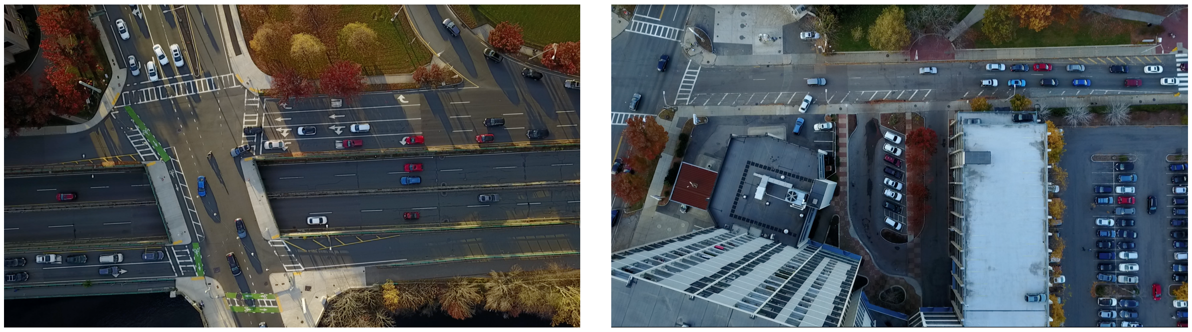

Figure B-1: Sample frames from aerial dataset.

B.2 Optical Flow

Because the flow field highlights objects in motion, it leaves out parked cars and other stationary objects when the camera platform is steady. In the case where the drone is in motion due to turbulence or planned flight, stationary objects are detected in the flow field; however, their magnitude of flow is distinctly smaller than their moving counterparts.

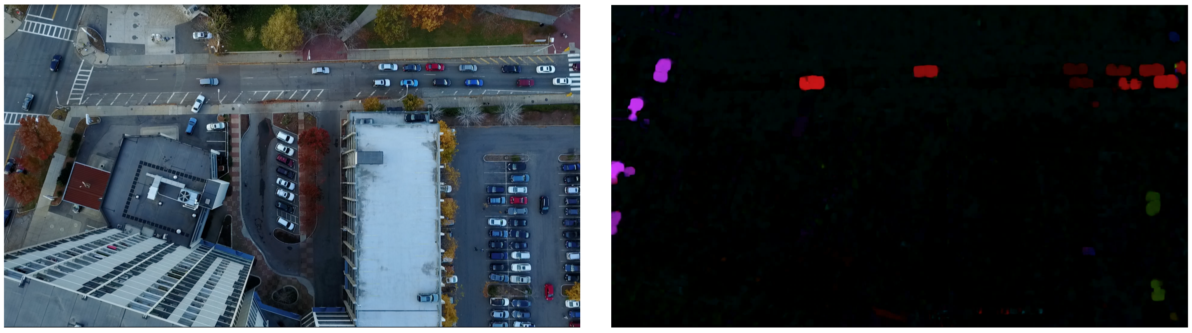

Figure B-2: RGB representation of flow vector field generated by Gunnar-Farnebäck method.

B.3 Detection

We leverage differences in magnitude and direction of flow to isolate the moving and stationary features. Thresholding and contouring are applied to the frames to generate the bounding boxes as shown in Figure B-3. We also see in Figure B-3 that there are cases where multiple moving objects are registered as one mass, largely due to noise in the flow field generated by the optical flow function.

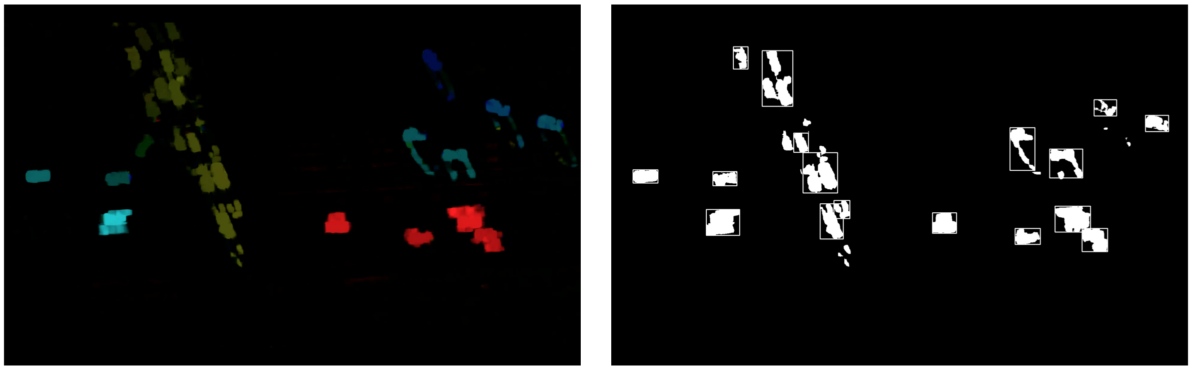

Figure B-3: Bounding boxes are generated around moving features via thresholding and contouring.

Bibliography

[1] FAA Modernization and Reform Act of 2012, February 2012.

[2] Federal Aviation Administration. UAS by the Numbers. Retrieved online from https://www.faa.gov/uas/resources/by_the_numbers/.

[3] M. S. Ahmed and A. R. Cook. Analysis of Freeway Traffic Time-Series Data by Using Box-Jenkins Techniques. *Transp. Res. Rec.* no. 722, pp. 1–9, 1979.

[4] M. M. Hamed, H. R. Al-Masaeid, and Z. M. B. Said. Short-term Prediction of Traffic Volume in Urban Arterials. *J. Transp. Eng.* vol. 121, no. 3, pp. 249–254, 1995.

[5] A. Balasingam, K. Gopalakrishnan, R. Mittal, V. Arun, A. Saeed M. Alizadeh, H. Balakrishnan and H. Balakrishnan. Throughput-fairness tradeoffs in mobility platforms. To Appear, Mobisys, 2021.

[6] Z. Cui, K. Henrickson, R. Ke, and Y. Wang. High-Order Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. *CoRR*, abs/1802.07007, 2018.

[7] Z. Cui, K. Henrickson, R. Ke, and Y. Wang. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[8] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. volume 2749, pages 363–370, 06 2003.

[9] Peter I. Frazier. A tutorial on bayesian optimization, 2018.

[10] A. Mordvintsev A. K. Optical Flow. Retrieved online from https://opencv-python-tutroals.readthedocs.io/, 2013.

[11] Ruimin Ke. Advanced framework for microscopic and lane-level macroscopic traffic parameters estimation from uav video. *IET Intelligent Transport Systems*, 14:724–734(10), July 2020.

[12] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *ICLR*, 2017.

[13] O. Lange and L. Perez. Traffic prediction with advanced graph neural networks. Retrieved online from https://deepmind.com/blog/, September 2020.

[14] P. Cohn, A. Green, M. Langstaff, and M. Roller. Commercial drones are here: The future of unmanned aerial systems. Retrieved online from McKinsey.com, December 2017.

[15] J. Lau. Google Maps 101: How AI helps predict traffic and determine routes. Retrieved online from https://blog.google/products/maps/google-maps-101-how-ai-helps-predict-traffic-and-determine-routes/, September 2020.

[16] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations (ICLR '18)*, 2018.

[17] California Department of Transportation. Caltrans PeMS. Retrieved online from http://pems.dot.ca.gov/.

[18] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu. Traffic Flow Prediction via Spatial Temporal Graph Neural Network. In *Proceedings of The Web Conference 2020*, WWW '20, page 1082–1092, New York, NY, USA, 2020. Association for Computing Machinery.

[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019.

[20] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transp. Res. Part C Emerg. Technol.* vol. 54, pp. 187–197, 2015.

[21] R. Ke Z. Cui and Y. Wang. Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. in 6th International Workshop on Urban Computing (UrbComp 2017), 2016.

[22] C. F. Shao, Z. S. Yao, and Y. L. Gao. Research on Methods of Short-term Traffic Forecasting based on Support Vector Regression. *Journal of Beijing Jiaotong University.* vol. 30, no. 3, pp. 19–22, 2006.

[23] L. Zhao, Y. Song, M. Deng, and H. Li. Temporal Graph Convolutional Network for Urban Traffic Flow Prediction Method. *CoRR*, abs/1811.05320, 2018.

[24] J. Zhu, Y. Song, L. Zhao, and H. Li. A3t-gcn: Attention temporal graph convolutional network for traffic forecasting, 2020.