# Multi-purpose, Multi-step Deep Learning Framework for Network-Level Traffic Flow Prediction

Maged Shoman
PhD Civil Engineering
Department of Civil and Environmental Engineering
University of Missouri
W1024 Lafferre Hall, Columbia MO 65211
ORCID number: 0000-0002-4265-071X
e-mail: mas5nh@mail.missouri.edu

Mark Amo-Boateng
Postdoc Civil Engineering
Department of Civil and Environmental Engineering
University of Missouri
W1024 Lafferre Hall, Columbia MO 65211
ORCID number: 0000-0002-8564-6307
e-mail: marbz@missouri.edu

Yaw Adu-Gyamfi
Assistant Professor
Department of Civil and Environmental Engineering
University of Missouri
W1024 Lafferre Hall, Columbia MO 65211
Tel (+1 573 882-4375)
ORCID number: 0000-0002-1924-9792
e-mail: adugyamfiy@missouri.edu

## ABSTRACT
This study proposes a data fusion and deep learning (DL) framework that learns high-level traffic features from network-level images to predict large-scale, multi-route, speed and volume of connected vehicles (CVs). We present a scalable and parallel method of processing statewide CVs trajectory data that leads to real-time insights on the micro-scale in time and space (2D arrays) on GPUs using the Nvidia Rapids framework and Dask parallel cluster, which provided a 50x speed-up in the data extraction, transform and load (ETL). A UNet model is then applied to perform feature extraction and multi-route speed and volume channels over a multi-step prediction horizon. The accuracy and robustness of the proposed model are evaluated by taking different road types, times of day and image snippets and comparing the model to benchmarks: Convolutional Long-Short-Term Memory (ConvLSTM) and a historical average (HA). The results show that the proposed model outperforms benchmarks with an average improvement of 15% over ConvLSTM and 65% over the historical average (HA). Comparing the image snippets from each prediction model to the actual image shows that image textures were highly similar in UNet to the benchmark models used. UNet's dominance in performing image predictions was also evident in multi-step forecasting, where the increase in errors was relatively minimal over longer prediction horizons.
**Keywords:** data fusion, CUDA, GPU, traffic prediction, deep learning, connected vehicles

## 1. INTRODUCTION
Traffic congestion costs cities billions of dollars every year when factors such as accidents, pollution and delays are factored in. According to a recent report published by the Texas Transportation Institute [40] all 494 metropolitan areas in the United States experienced 8.7 billion vehicle-hours of delay in 2019; resulting in 3.5 billion gallons of wasted fuel and $190 billion in lost productivity, or about 0.15 percent of the nation's GDP. These costs drive the need for a data-driven strategy to solving these issues. When traffic demand approaches or exceeds the traffic system's available capacity, traffic congestion occurs. Many studies [40, 41, 42] have shown that traffic datasets can be used to predict traffic congestion, allowing drivers to avoid congested areas (e.g., through traffic flow forecasting navigation systems), policymakers to decide on changes to traffic regulations (e.g., replacing a normal lane with a toll lane), urban planners to design better pathways (e.g., adding or removing a road lane), and transportation engineers to better plan for the timing of construction activities. Traffic forecasting is a critical component of advanced traffic management systems that can help transportation planners in planning for volatile events ahead, by taking early actions and arrangements, which contributes to better traffic management and service quality. It may not only serve as a valuable reference for increasing the efficiency of limited traffic management resources, but it can also assist passengers in making arrangements ahead of time to minimize traffic congestion. Long-term projections are more likely than short-term forecasts to reduce travelers' average trip time [43]. Common forecasted traffic parameters include: traffic flow [1], traffic speed [2], and traffic time [3]. The increasing availability of large-scale traffic data, which can be looked at from a temporal and spatial lens, has paved the way to develop prediction models that are robust to capture the underlying driving mechanism of traffic volatilities, especially the random (unforeseen) components.

Temporally, majority of prior studies have focused on single-step traffic flow forecast for a single road section with a time interval of less than 30 minutes. For some applications in intelligent transportation systems (ITS), such as traffic planning, it can be insufficient. Another issue is the increased frequency of collected (input) data which allowed the value of long time horizon predictions to supersede shorter term. As a result, multi-step traffic flow prediction is gaining popularity. Multi-step traffic flow prediction uses the same methodologies as single-step traffic flow prediction, however, the prediction performance rapidly degrades as the number of steps grow. Developing a practical multi-step prediction model is, thus, more important than a single-step prediction task because it provides valuable insights over longer time horizons which allows for better positioning of traffic management strategies.

In addition, many studies only focused on the spatial component by predicting traffic on a single-route or a specific connection or crossing. The development of an ITS demands the need to explore multi-route predictions on a larger scale by considering the complex spatial dynamics of a network [5]. While prior

knowledge of the distance or travel time between regions can aid in capturing spatial correlation, there are still some hidden time-varying traffic patterns that data-driven methods must uncover. The challenge is resolving the intricate spatiotemporal dependencies, which refer to traffic information (e.g., speed or volume) at a certain location in space and moment in time. With the emergence of deep learning models, this research aims to solve the question of how to construct appropriate deep learning models to cope with large-scale complex network-wide traffic data.

Large-scale network traffic prediction demands an intelligent and efficient prediction methodology to forecast traffic on longer horizons and reflect the flow propagation. Numerous variables affect a region's future traffic state, including historical observations of traffic, correlation with other regions, and external factors (holidays and special events). The technique used to fuse muti-purpose variables such as traffic speed and volume, is a challenge for the current generation of prediction models. The interrelationships between regions are intricate and complex which adds to the challenges in developing a prediction model. As a result, more research into how to create an accurate and reliable network-wide (by exploring multi-routes), multi-purpose (such as speed and volume), multi-step (longer prediction horizon) prediction model is required.

Reliability of the estimates obtained from the developed models is another issue since it greatly depends on the data source. Data used for traffic forecasting has two main issues: availability, size of data, and the overreliance on probe data. When qualified traffic data is unavailable, the trained model's performance degrades since performance correlates with the quality of input data. While we can collect more traffic data due to transportation infrastructure modernization, the data is frequently of poor quality, with noise and critical features missing. Currently, the amount of qualified traffic data available for analysis is insufficient. To our knowledge, most prior studies [7-9] used probe traffic data that was less than a year old and, in some cases, as recent as one or two months [10]. Probe data cannot capture the live travel time or volume on road segments, and using it for traffic forecasting is likely to yield unreliable estimates. Therefore, there is a need to use a more reliable data source that can provide microscopic live travel information to improve the reliability of traffic predictions along road segments.

The projected growth of CV will provide an alternative way of collecting real-time data for traffic forecasting. The future of ITS is shifting towards big real-time data from CV as automobile makers rush to incorporate CV technology in novel and current vehicles for numerous apparent advantages, which include vehicle autonomy and navigation, vehicle sensor and driver monitoring, live over-the-air updates, advanced road warnings, and improved battery and fuel efficiency. Government and state institutions that create, maintain and manage road infrastructure may take advantage of the CV data available to know what is happening on the road and make informed decisions on traffic flow and road pavement infrastructure. Thus, it is critical to effectively process all CV data on a state level for statewide transportation infrastructure management. This study's connected vehicle (CV) data is from wego technologies. The data was collected and transmitted every 3 seconds. The study estimated travel times on arterials and freeways by analyzing data from connected vehicles, including the vehicle's speed, acceleration, GPS location, and “brake press”. Additionally, the current study advances the state-of-the-art by developing a traffic forecasting model using UNet architecture. Figure 1 presents the framework for the network-wide predictions using the image outputs from each phase.

The significant contributions of the paper are summarized below;

1. Propose a pipeline for processing and learning from large-scale spatiotemporal data by leveraging distributed GPU clusters.
2. Propose a data fusion technique that enables state-of-the-art machine learning (ML) models to learn from multisource data, by leveraging GPU computing through Nvidia Rapids and Dask framework.
3. Design a DL framework for simultaneous, pixel-level, dense prediction of traffic flow variables (speed and volume) while considering the network traffic temporal evolutions and spatial dependencies using a UNet model that learns traffic data through 3-dimensional matrices.

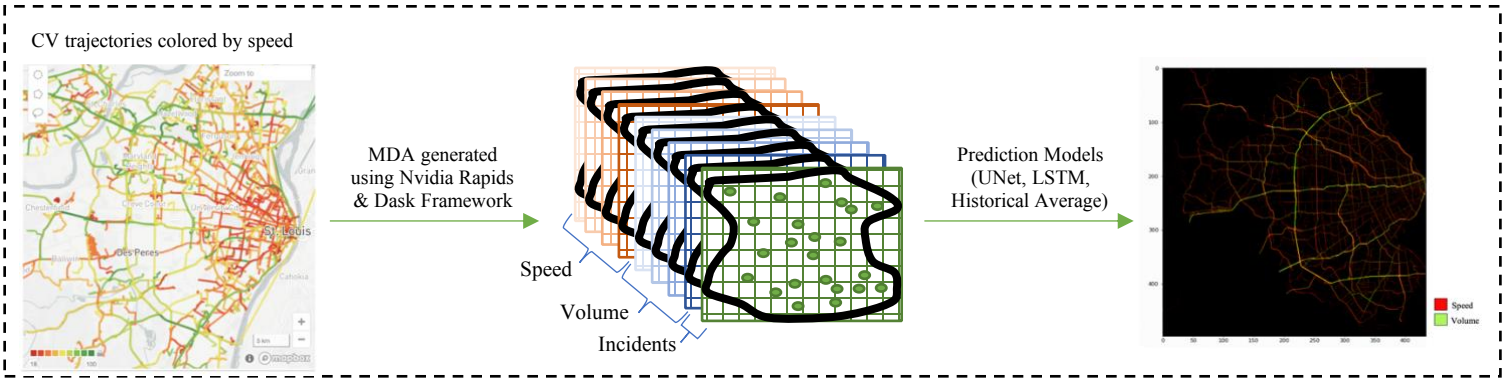



**Figure 1: Framework for network-wide traffic predictions**

## 2. LITERATURE REVIEW

Developing an Intelligent Transportation System (ITS) is a promising solution to provide more accurate travelling information based on future predictions to transportation users and developers. The techniques used to predict traffic across the literature is summarized into the following categories: statistical, light machine learning and deep learning. Statistical mainly uses time series analysis models such as historical and moving averages which can be helpful with short-term predictions on static data. However, they fall short with multiple time step predictions into the future on the continuously varying traffic data. Standard machine learning models include Artificial Neural Network (ANN), Support Vector Machine (SVM) and K-Nearest Neighbor (kNN), which are generally better performing than statistical models because of their architecture's capability in capturing more features. However, extracting the complex and dynamic patterns in the spatio-temporal dynamic traffic data adds to the model limitations. The rise in faster Graphics Processing Units (GPU) paved the way for the increased use of deep learning models to perform predictions. Due to their superior ability to capture complex traffic patterns, various deep learning-based methods for traffic prediction have recently been successfully applied to traffic forecasting. Table 1 presents a sample of the recent use of deep learning models for traffic predictions. The table presents the authors, used prediction model, predicting variables, road-type, prediction horizon and results.

### *2.1. Single and Multi-step Forecasting*

Short-term traffic forecasts restrict many existing approaches, and there are few successful methods existing for predicting long-term traffic status. Short-term is also referred to as single-step  We define short term predictions as any predictions that fall in the range of 5 to 15 minutes into the future. Medium-term predictions are 15 minutes to one hour, and long term predictions are beyond one hour. Long-term or multi-step forecasting is more difficult than short-term prediction because of the sensitivity of error propagation [44]. Real-time traffic control is where short-term forecasting is most useful. Long-term forecasting that is accurate and timely may assist managers in making early judgments, actions, and overall arrangements, which can help improve traffic management and service quality. It can not only serve as a valuable reference for increasing the efficiency of limited traffic management resources, but it can also assist passengers in planning ahead of time to avoid traffic congestion [2]. Sequence-to-sequence (Seq2Seq) is a frequently used technique in multi-step forecasting [18–20]. It is also common to capture temporal dependency using recurrent neural networks (RNNs) and temporal convolutional networks (TCNs) [13–15].  Long Short Term Memory (LSTM) is a widespread technique that Chen et al. (2021) and Cui et al. (2018) used in their studies to predict the short term changes in traffic flow and speed, respectively. Several authors used graph neural networks by focusing attention on different space and time features. Yu (2021) performed short, medium and long term predictions of traffic speed on urban roads while other authors (Li et al. (2021), Yin et al. (2021), Zhao et al. (2020)) used freeway segments. It is worth noting that Zhao et al. (2020) attention of

temporal changes in their model performed much better than Li et al. (2021) on the same road type and prediction horizon.

### *2.2. Multi-purpose Forecasting*

A single-purpose forecasting technique is focused on modeling traffic condition using one variable (such as speed or flow or occupancy), whereas the multi-purpose approach is based on constructing a model that takes into account more than one variable. These models, unlike single-purpose models, are capable of capturing travel characteristics from multiple dimensions of a transportation network over time. Multilinear regression models were used by [45] to forecast bus arrival time using multi-purpose attributes such as: distance, number of passengers at stops, stop numbers, and weather conditions. The performance of regression models will degrade as the dimension of the data rises, because the attributes in transportation services are frequently not independent but connected with one another. Complex interactions and noisy data demand the use of machine learning algorithms. [46] proposed a data clustering and genetic programming technique for predicting highway trip time. Two of the most extensively used machine learning models in multi-purpose bus travel time prediction are artificial neural networks (ANN), and support vector machines (SVM), [47]. Kalman Filtering models, which use both historical and real-time data, have been widely used to estimate bus arrival times [48, 49, 50]. Previous research in this field has mostly focused on constructing models for anticipating delay as a self-contained single-purpose prediction process.

### *2.3. Multi-routes Forecasting*

Mulit-routes is defined as the collection of many different road types and routes. Numerous GNNs are used to extract spatial dependency from traffic networks [5, 8, 11, 12]. Chen et al. (2021) filtered freeway segments from the traffic network and achieved good results. The entire network was used by Cui et al. (2018). However, most models lacked the use of a network-wide dataset and longer-term predictions. Image segmentation and classification has been widely successful using UNet [13]. U-Net is a CNN based on a fully convolutional neural network where its architecture is altered and expanded to work with fewer training images to obtain significantly precise segmentation results. Choi (2020) achieved strong results using a UNet model for predicting traffic speed and volume for multiple routes in the short and medium term. Although GCN-based techniques may learn more hidden aspects of traffic networks than CNNs, they are ineffective at capturing dynamic spatial traffic dependency. The term "Mulit-routes" present a challenge due to the fact that the relationship between two static places can change over time. For example, during morning and evening peak hours, the spatial links between residential and commercial districts are more important than at other times. Since the majority of previously published works fall short of accurately maintaining spatial information while simultaneously making good predictions [5, 21, 22], this study seeks to adopt an approach that sought to maintain spatial information. Numerous publications [12, 16, 23] have described the development of an adaptive matrix for data-driven spatial correlation discovery using spatial correlation data. When the data is insufficient or noisy, the efficiency of data-driven methods is limited, and accurate prior knowledge may help the model perform better in these situations. Most studies focus exclusively on predefined correlations or data-driven correlations for a prediction. Also, most prediction models suffer from information dilution, observed in other multi-step prediction models [6, 18, 19]. The original data from each input step has been diluted several times by both the encoder and decoder cells before reaching a specific output step in the sequence. When there is sufficient data, the dilution effect can be mitigated; however, insufficient data can exaggerate the effect, resulting in decreased prediction performance. To eliminate the issues mentioned earlier, we use connected vehicle data in this study for traffic forecasting.

| AUTHORS | MODEL | PREDICTING | ROAD TYPE | PREDICTION HORIZON | RESULTS |
|---|---|---|---|---|---|
| **CHEN ET AL. (2021)** | LSTM + Ensemble Empirical Model Decomposition (EEMD) | Traffic flow | Freeway | Short (5-15min) | RMSE: 0.79 |
| **LI ET AL. (2021)** | Graph Convolution Network (GCN) | Traffic flow | Freeway | Short (5-15min) and medium (15min – 1hr) | RMSE: 15_min = 32.17 30_min = 32.96 45_min = 33.68 60_min = 34.53 |
| **YU (2021)** | Generative Adversarial Graph Attention Network | Traffic Speed | Urban | Short (5-15min), medium (15min – 1hr) and long (1 - 4hr) | MAPE: Short: 6.1% Medium: 8.3% Long: 12.6% |
| **YIN ET AL. (2021)** | Multi-stage Attention Spatial-Temporal Graph Network (MASTGN) | Traffic Flow & Speed | Freeway | Short (5-15min) | RMSE: 17.73 |
| **ZHAO ET AL. (2020)** | Temporal Graph Convolutional Network (TGCN) | Traffic Speed | Freeway | Short (5-15min) and medium (15min – 1hr) | RMSE: 15_min = 4.53 30_min = 5.01 45_min = 5.35 60_min = 5.64 |
| **CHOI (2020)** | UNet | Traffic Speed and Volume | Network-wide | Short (5-15min), medium (15min – 1hr) | MSE: 0.0016 |
| **YAO ET AL. (2019)** | CNN + LSTM | Traffic flow & volume | Network-wide | Medium (15min – 1hr) | RMSE: 24.10 |
| **CUI ET AL. (2018)** | Deep stacked bidirectional and unidirectional LSTM | Traffic Speed | Nework-wide | Short (5-15min) | MAPE: 5.6% |
| **WU ET AL. (2018)** | CNN + Recurrent Neural Network (RNN) | Traffic flow | Freeway | Short (5-15min) and medium (15min – 1hr) | RMSE: 15_min = 32.16 30_min = 34.29 45_min = 36.08 |
| **MA ET AL. (2015)** | LSTM Neural Network (LSTM NN) | Traffic Speed | Freeway | Short (5-15min) | MAPE: 5_min = 3.78% 10_min = 3.78% 15_min = 3.78% |

**Table 1: Comparison of recent use of deep learning models for traffic predictions**

Each experimental traffic feature (for example, traffic speed and flow) has both spatial and temporal attributes (i.e., its observation location and time). Generally, studies [9,10, 24, 27] extracted spatiotemporal patterns solely from traffic features without fully exploiting those traffic features' spatiotemporal attributes. By providing additional information, these attributes, on the other hand, can directly aid the model in identifying spatiotemporal correlations between traffic states. Apart from that, they can augment existing spatiotemporal information when sufficient feature data is unavailable. Furthermore, versatile and

extendable transportation data integration frameworks are critical for modern transportation analysis and management. Data Fusion is the challenge of merging data from several sources and giving consumers with a consistent representation of that data [51]. Data integration system design is a critical step in a wide range of real-world applications, particularly in Intelligent Transportation Systems (ITS). Other common challenges in traffic prediction, such as planning issues and traffic estimation, are similarly involved with multi-source fusion [52]. In both research and practice, transportation data integration frameworks and tools have been devised and deployed for a variety of applications. To address the challenges mentioned earlier and limitations, we employed a large-scale GPU cluster-based data processing framework to fuse large-scale datasets and then leveraged the UNet architecture for multi-step forecasting by combining the volatile traffic features on a network-wide level to augment the spatiotemporal information contained in the model input.

## 3. METHODOLOGY

### 3.1. Problem formulation and overview

Many existing studies ignore the use of large-scale data to develop traffic prediction models and thus, disregard the complex topological structure of road networks and temporal patterns by using a single-route, single-step and single-purpose predictions. Such approaches are motivated by faster computations and reporting high accuracies. However, in practice, the applications of such techniques is very limited since it only captures the instantaneous and steady-state interactions among traffic variables, therefore, a multi-step, multi-purpose traffic prediction framework for multiple routes should be developed.

Specifically, let $x_i = (T * W * H * C)$ represent the input data tensor for a specific day from the training data ($i$) with $C$ number of channels or purpose, ($W * H$) is the 2D array width and height and $T$ is the time bins per day with each time step aggregated by five minute intervals. The goal is to predict $x_p$ on test data ($p$) using only one hour from the test dataset ($T+1, T+2,..., T+11$) to predict the remaining hours in the same day ($T+12, T+13,..., T+287$). Compared to exisiting approaches where $C$ is usually one (i.e, 3D tensor instead of 4D) and predictions are usually short or medium-termed (i.e, $T+12, T+13,..., T+23$), the current framework proposed in this study addresses the multi-purpose, multi-step large-scale traffic forecasting challenge at a network level. Figure 2 presents the framework of the proposed methodology. Firstly, CV data was collected for one month in Saint Louis County. The data provides attributes such as the vehicle's location, heading, speed, volume and flow, etc. This work uses histrocial traffic speed, volume, and incidents to forecast future speed and volume. The data then goes through a pre-processing stage to make it feasible in our prediction models. Data cleaning is then performed to clean the data from missing values and anomalies. Cleaned data is then formatted by grouping different headings and time bins together. Data fusion is then performed on the different datasets along with the same spatial and temporal bins. MDAs or images are then generated to efficiently leverage smaller size compacted data layers as an input to the proposed prediction model. In the prediction model, UNet is used as our Convolutional Neural Network (CNN) as it is designed to learn from the MDA matrices and make predictions. The accuracy and robustness of the UNet model are compared to the conventional Convolutional Long Short Term Memory (ConvLSTM) model and a statistical historical average model. The following section discusses data fusion and MDA generation in detail.

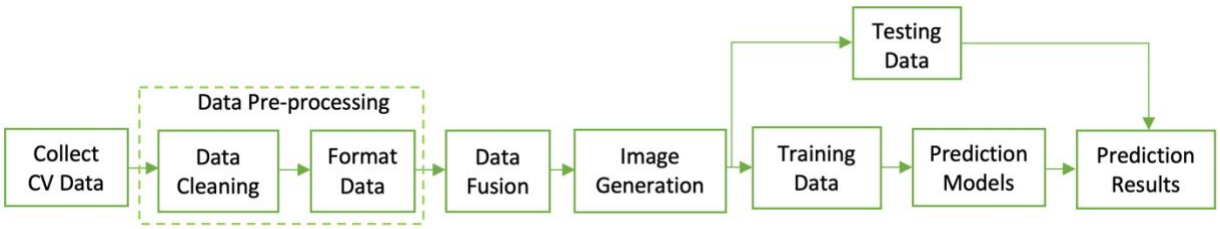



**Figure 2: Framework of the proposed methodology**

### 3.2. Input Data

Saint Louis county consists of around 30 cities and spans an area of 523mi$^2$. The location is in the eastern-central portion of Missouri state. We collect CV data through Wejo (wejo.com) for May 2021 (31 days). Wejo provides highly granular vehicle point data about live traffic conditions at a frequency of three seconds and an accuracy of 3 meters. Table 2 presents a sample of the data, followed by an explanation of each attribute. Two primary datasets were used to develop our prediction model. CV data provided connected vehicles traffic speed and volume, and Waze (waze.com) data provided incidents data. Both datasets are considered to be dynamic because they vary over time. Data was initially collected in a CSV format and converted to a Multidimensional Array (MDA) - H5 image format.

H5 MDA format collects datasets and groups for efficiently storing raw images. MDAs are created by temporally aggregating data in 5-minute bins and spatially in 495*436 (2D image size) spatial bins. Figure 3 presents an image example of the spatial and temporal variation of CV data for a few time bins for two channels with different colours for each channel. A detailed description of the data structuring process is presented in the following section.

| Data_point_id | Journey_id | Timestamp | latitude | longitude | Postal_code | speed | heading | squishvin | Ignition_status |
|---|---|---|---|---|---|---|---|---|---|
| C3b34-kr5r... | 33456rd | 2021-05-09 03:48:42 | 37.664087 | -92.6546 | 65536 | 105.98 | 33 | 1G11A5SLFW | MID_Journey |
| A4b33-g5e... | 31224tf | 2021-05-09 03:49:42 | 37.667707 | -92.6490 | 65536 | 0 | 53 | 1H11Fg5LHF | MID_Journey |
| D3b64-k6te... | 22124fs | 2021-05-09 03:49:49 | 37.690978 | -92.6490 | 65536 | 48.38 | 33 | 4561A5SLFQ | MID_Journey |

**Data_point_id:** unique identifier of the row point data collected.
**Journey_id:** unique identifier of the trip performed by a vehicle from start to finish.
**Timestamp:** date and time when the point data was collected
**Latitude & Longitude:** coordinates of the collected point data.

**Postal_code:** unique identifier of a particular region.
**Speed:** speed of the vehicle in km/hr.
**Heading:** direction or bearing of the vehicle.
**Squishvin:** first 11 digits of the vehicle identification number.
**Ignition_status:** engine operation status of the vehicle.

**Table 2: Sample of collected CV data**

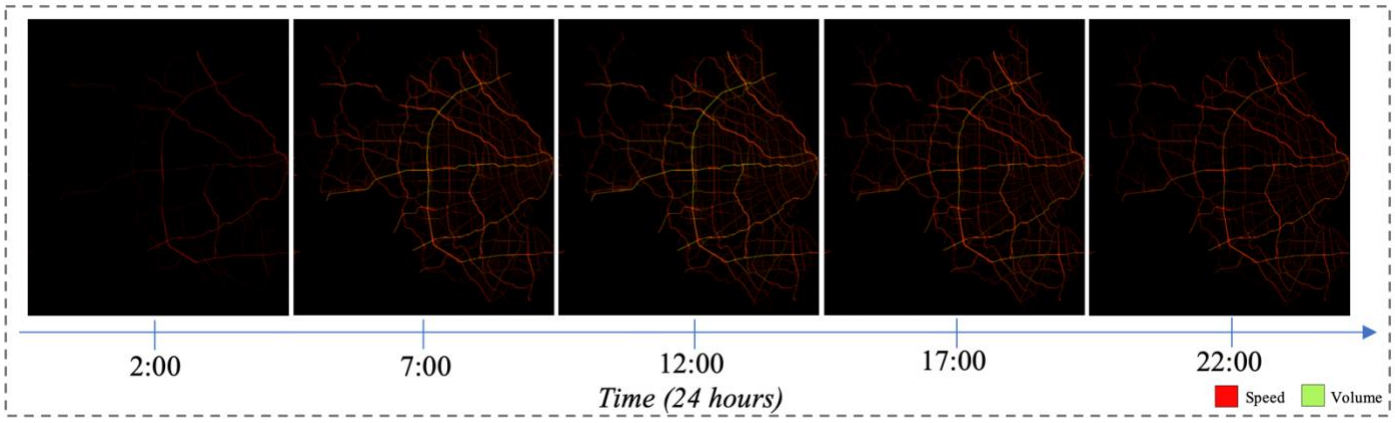


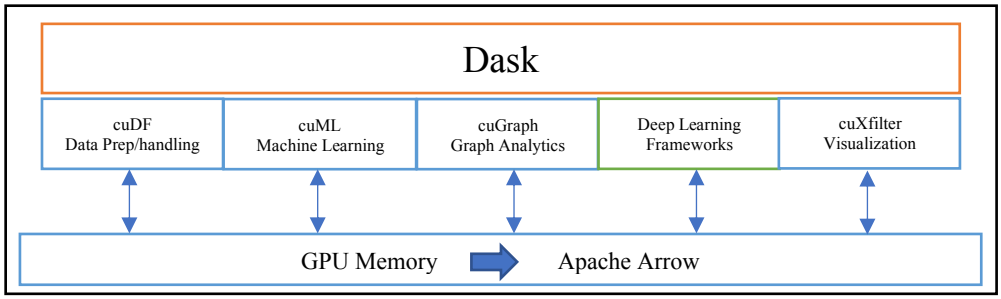

**Figure 3: Speed and volume variations visualized by space and time**

### 3.3. Input Data Structuring

To accelerate the processing of big CV data in our study, we use Nividia Rapids and Dask framework. Nvidia Rapids is an open-source suite of software libraries for end-to-end data science and analytics pipelines on GPUs. Rapids is built on top of Nvidia CUDA for accelerated computing and Apache Arrow for GPU in-memory computing, see Figure 4 (a) and includes several libraries across the data science toolchain. Rapids is the GPU implementation of conventional data science libraries and natively scales from workstations to clusters to cloud systems with the help of Dask libraries. A comparison of Rapids library with popular data science libraries is shown in Figure 4 (b).

Dask natively scales Python data frames (CPU and GPU) across several nodes and partitions. Dask also offers advanced parallelism and data processing pipelines that enable large-scale analytics by using a directed acyclic graph (DAG) lazy execution framework, which ensures that computational work is scheduled, rebalanced, and optimized before the data is needed. This allows for fast prototyping and experimenting even on massive cluster systems. Dask integration with Rapids allows for large-scale GPU cluster-based data processing.

(a)




(b)

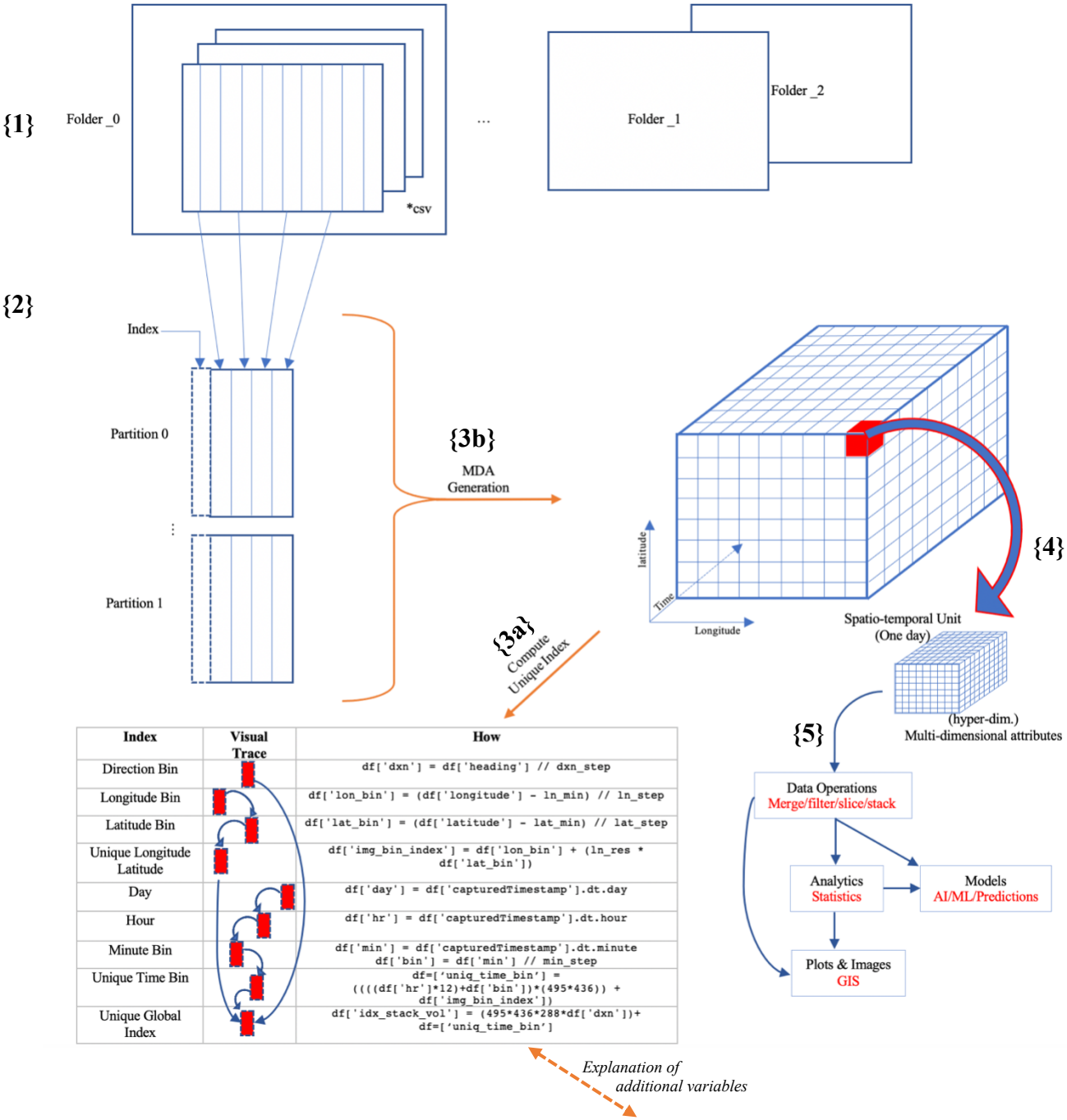
| | CPU | GPU/RAPIDS |
|---|---|---|
| *Data handling* | Pandas | cuDF |
| *Machine Learning* | Scikit-learn | cuML |
| *Graph analytics* | NetowrkX | cuGraph |

| | CPU | GPU/RAPIDS |
|---|---|---|
| *Viz* | Bokeh/ Datashader | cuXfilter |
| *Geospatial* | GeoPandas/ SciPy.spatial | cuSpatial |
| *Signals* | NetowrkX | cuSignal |
| *Cyber* | cyberpandas | CLX |

**Figure 4: (a) Nvidia Rapids Framework, (b) Comparison of Rapids to popular libraries**

The experimental setup for the project was on AWS GPU virtual machines with Intel Xeon Platinum 8259CL 48 core vCPUs @ 2.50GHz, 192 GB of RAM and 4xT4 GPUs with 16 GB vRAM each. The virtual machines were running AWS optimized Ubuntu 18.04 LTS operating system software. The software stack installed included CUDA 10.2 with driver version 440.33.01. Additional software includes Docker CE v18.03.1-ce and Nvidia Docker2 software for GPU containerized setup. The DLI RAPIDS Course – Base Environment container image v1.0.0 available at the Nvidia Container Catalogue (NGC) was used to launch a Python Jupyter Lab environment for this experiment on the AWS virtual machine. The NGC DLI RAPIDS container image already comes with preinstalled software including Rapids, Conda, Graphiz, cuDF, cuPy, etc., simplifying the experimental setup. In addition, the pull and launch of the container image expose internal ports to the container and allow for global internet access to the Jupyter Lab environment outside the localhost environment. To benchmark the experimental setup, we used the in-built Python ***timeit()*** function with ***repeat()*** method to run each algorithm a couple of times. Our results show a 50x speed-up in the ETL of the CV data for an entire day for all the unique CV journeys, reducing the processing time from

48 hours to 15 minutes. The algorithm and overview of the data structuring approach for processing the CV and sensor data fusion are presented in Figure 5, with each step numbered in curly brackets. The main reason behind structuring the data in such a format is because MDAs can store and organize large amounts of data better than CSV, which allows for more efficient processing of files [28]. One CSV file size 16GB can be structured into a 20MB MDA. Our approach was to query the data from several CV data files across several folders and drives into a giant temporary in-memory database and then transform it into a Spatio-temporal 3D lattice with unique attributes that can be further used for attribute-based hyper-dimensional analysis. To achieve this, we used the Dask framework for massively large distributed data processing and filtering with the GPU backend on Nvidia Rapids. After setting up a local cluster, the Dask framework was used to read all the CV and sensor data files {1} and filtered on interest columns into a giant in-memory data lake {2}. A new unique index was computed for the data, and the data was repartitioned to reduce the number of Dask workers and optimize performance while at {2}. In order to translate the data into a 3D Spatio-temporal matrix, unique indices of each data row were computed using the procedure in {3a}. This began with the computation of the unique spatial discretized bins for longitude and latitude, and each data row longitude and latitude were used to compute the spatial positional index and placed in the appropriate bin. The same procedure created a discretized temporal bin based on the day, hour and minute. Using the unique spatial, temporal, and directional indices, unique unrolled positional global indices were computed for each data point which was then used to translate the in-memory database into the 3D spatial-time lattice {3b}. Each spatial-time lattice cube {4} contained all data entries with the same index as well as other attributes such as speed and direction, which could then be used in hyper-dimensional data operations based on the data attributes {5}. After filtering and stacking based on attributes, other analytics based on speed, data counts and direction were performed and used in this study.




| Index | Visual Trace | How |
|---|---|---|
| Direction Bin | | `df['dxn'] = df['heading'] // dxn_step` |
| Longitude Bin | | `df['lon_bin'] = (df['longitude'] - ln_min) // ln_step` |
| Latitude Bin | | `df['lat_bin'] = (df['latitude'] - lat_min) // lat_step` |
| Unique Longitude Latitude | | `df['img_bin_index'] = df['lon_bin'] + (ln_res * df['lat_bin'])` |
| Day | | `df['day'] = df['capturedTimestamp'].dt.day` |
| Hour | | `df['hr'] = df['capturedTimestamp'].dt.hour` |
| Minute Bin | | `df['min'] = df['capturedTimestamp'].dt.minute`<br>`df['bin'] = df['min'] // min_step` |
| Unique Time Bin | | `df=['uniq_time_bin'] = ((((df['hr']*12)+df['bin'])*(495*436)) + df['img_bin_index'])` |
| Unique Global Index | | `df['idx_stack_vol'] = (495*436*288*df['dxn'])+ df=['uniq_time_bin']` |

*Explanation of additional variables*

| Variable | Explanation | Value |
|---|---|---|
| `min_step` | bin function for minutes | 5 |
| `dxn_step` | bin function for direction | 90 |
| `lat_res` | latitude resolution | 495 |
| `ln_res` | longitude resolution | 436 |
| `lat_min` | latitude minimum (bounding box [bottom left] for study area) | 38.40274 |
| `lat_max` | latitude maximum (bounding box [top right] for study area) | 38.86540 |
| `ln_min` | longitude minimum (bounding box [bottom left] for study area) | -90.73262 |
| `ln_max` | longitude maximum(bounding box [top right] for study area) | -90.18459 |
| `lat_step` | Y-image resolution<br>`(lat_max - lat_min) / lat_res` | |
| `ln_step` | X-image resolution<br>`(ln_max - ln_min) / ln_res` | |

**Figure 5: Overview of data structuring approach**

Thirty-one unique dates (days) are available for our data. Twelve separations are exported from hourly CSV files since 12 (five-minute) time bins per hour (60 minutes per hour/5-minute bins). Each separate file then goes through the splitting channels process where separations are made for the unique columns/channels: incidents, speed and volume. The direction column is estimated and added based on the bearing information provided by the speed and volume channels. Four main heading quadrants are used in the estimation: Northeast (NE), Southeast (SE), Southwest (SW), Northwest (NW). However, the incidents channel did not provide a bearing column, so we could not split its directions. To create an MDA for each CSV exported in the previous step, we first created an empty raster with latitude and longitude coordinates for the study area, scaled at a height and width of 495 and 436, accordingly. The coordinates used in spatial bins are fixed throughout all MDAs to ensure that they are all developed at the same scale, as presented in Figure 6. We use the mean of values within a temporal (frame) and spatial (bin) for the speed channel to get the average speed. We use the sum of values within a temporal (frame) and spatial (bin) for each volume and incident channel to get the total count. Each array created at this step has the shape [495*436].

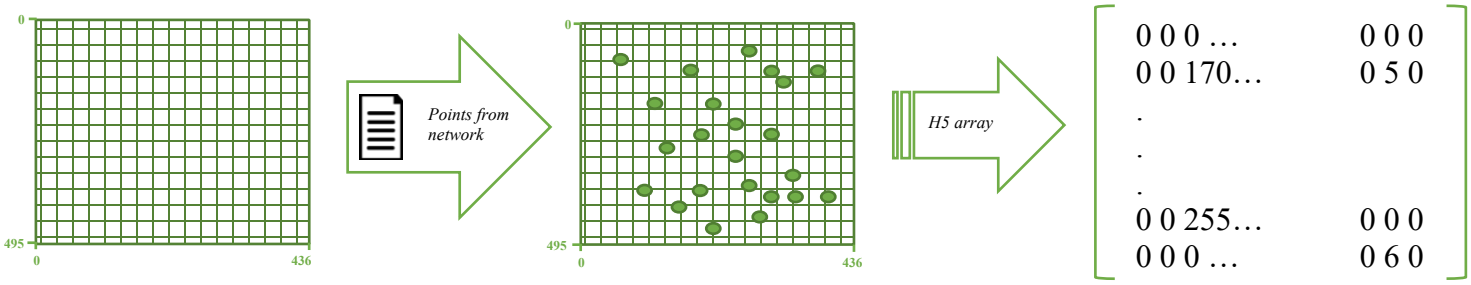



**Figure 6: Spatial bins created for a consistent scaling of H5 arrays**

MDA for speed along with four directions, volume along with four directions, and incidents are then stacked together to form a stacked array of the shape [495*436*9]. The channels are stacked along the third axis/dimension. Time bins along each hour are then stacked together along all channels to form another stacked array of the shape [12*495*436*9]. Time bins are stacked along the fourth axis/dimension. Each day, hourly bins are stacked together along stacked time bins and channels to form a further stacked array of the shape [288*495*436*9]. Hourly bins are stacked along the fourth axis/dimension. This process is performed for each unique date, so at the end, we can have an array of shape [288*495*436*9] for each day. Figure 7 presents a visualized shape of the temporal aggregation of channels and bins of an MDA.

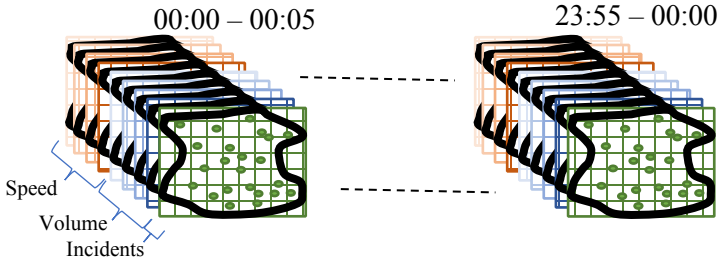



**Figure 7: Temporal aggregation of a MDA per day**

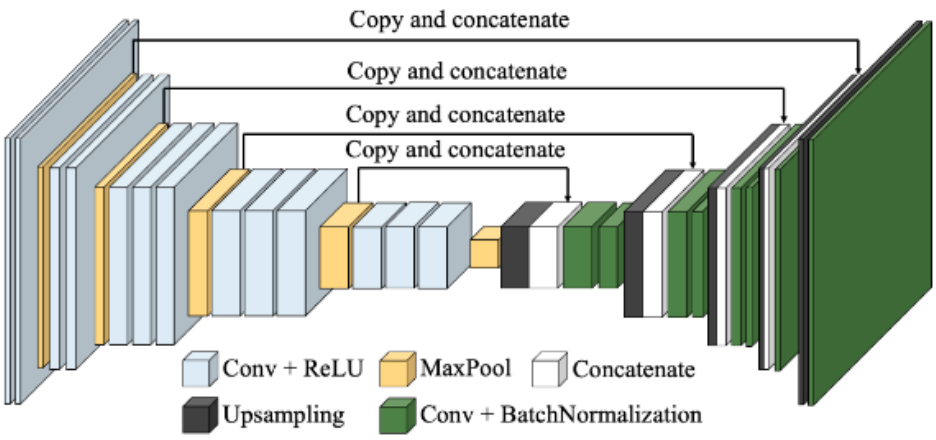
### 3.4. Data Normalization

Normalizing arrays or pixels is essential to give minor differences in one variable the same attention as huge variances in another. The model only sees a collection of numbers and does not know predictors, factors, variables, observations, or anything else. Values across volume and incidents channels are normalized between 0 and 255. However, values across speed channels are normalized between 1 and 255.

### 3.5. UNet Model

Image segmentation and classification has been widely successful using UNet [13]. U-Net is a CNN based on a fully convolutional neural network where its architecture is altered and expanded to work with fewer training images to obtain significantly precise segmentation results. While training on an NVIDIA GTX 1080 Ti GPU, the segmentation of a 495 × 436 image took less than a second. As shown in Figure 8, UNet's architecture consists of a contracting path to absorb context and an expansive symmetric path to facilitate precise localization. The contracting path follows the typical convolutional network with multiple convolutions accompanied by Rectified Linear Unit (ReLU) and max-pooling operation.

Similarly, the contracting part reduces spatial information and increment in features information. However, the expansive path integrates spatial and feature information using upconvolutions with feature information from the contracting path. In our model, the convolution layer was heavily connected to the average pooling layer and then decoded using one deconvolution layer trailed by one convolution layer. We decided to use average pooling because of its ability to retain features and give smooth arrays. The learning rate is 3e-4 and was configured/lowered to improve the model performance. Adam optimizer was used as the optimization algorithm, and mean squared error was used to measure how well each model performed.




| | Output Shape |
|---|---|
| DenseBlock-1 | (495, 436, 64) |
| AvgPool | (248, 218, 64) |
| DenseBlock-2 | (248, 218, 96) |
| AvgPool | (124, 109, 96) |
| DenseBlock-3 | (124, 109, 128) |
| … | |
| DenseBlock-7 | (8, 7, 128) |
| AvgPool | (4, 4, 128) |
| DenseBlock-8 | (4, 4, 128) |
| Conv Layer | (4, 4, 128) |
| DeconvBlock-1 | (8, 7, 128) |
| DeconvBlock-2 | (16, 14, 128) |
| … | |
| DeconvBlock-7 | (495, 436, 128) |
| ConvLayer | (495, 436, 96) |

**Figure 8: Designed UNet architecture with output shape per block**

Table 3 presents the UNet input parameters used in our model and explains how each value was extracted/calculated.

| Input Parameter | Value | Explanation |
|---|---|---|
| No. of training files | 24 | First 24 days |
| No. of validation files | 3 | Three random days |
| No. of testing files | 3 | Last three days |
| No. of frames/day | 288 | (60mins per hour / 5mins time bin) * 24 hours/day |
| No. of frames before | 12 | Previous hour time frames |
| No. of frame sequence | 24 | Used time frames (12) + Frames to predict (12) |
| No. of frames output | 12 | Subset to predict |
| Height | 495 | Image height |
| Width | 436 | Image width |
| No. of channels | 9 | Speed (4 directions) + Volume (4 directions) + Incidents |
| No. of channels output | 8 | Speed (4 directions) + Volume (4 directions) |
| Visual input channels | 108 | [channels (9) * Used time frames (12)] |
| Visual output channels | 96 | No. of channels output (8) * No. of frames output (12) |
| Batch size | 2 | No. of samples processed |
| Learning rate | 3e-4 | The amount that the weights are updated during training |
| Number of epochs | 20 | No. of complete passes through the training dataset |

**Table 3: Model input parameters**

The input to the training model is the MDAs generated from the study area with spatial and temporal characteristics, which can be defined as:

$$x_j^i = [v_i, v_{i+1}, \dots, v_{i+O-1}], i\epsilon[1, L - I - F + 1] \quad (1)$$

Where,

- $i$ is the image index;
- $j$ is the channel index;
- $v_i$ is a column vector representing the traffic variable (speed/volume);
- $O$ is the span of output intervals;
- $I$ am the span of input intervals and
- $L$ is the period intervals.

The input image goes through convolution and pooling to extract the significant image features, which is the principal phase of the UNet model where the output size gets smaller in dimension. The output from this phase can be defined as:

$$O_m^k = P\left(\sigma\left(W_m^k x_m^k + b_m^k\right)\right), k\epsilon[1, c_1] \quad (2)$$

Where,

- $P$ is the pooling procedure;
- $\sigma$ is the activation function;
- $(W_m^k, b_m^k)$ is the parameters of the $m$th layer and
- $k$ is the convolutional filter channel index.

The output from the preceding convolutional layer is max-pooled in the succeeding block, and then the identical architecture is applied again. Max pooling is applied to downsample the size of the image (pixels), reducing the number of used parameters. The joining of layers together is done in the concatenation phase.

### 3.6. ConvLSTM Model

The first benchmark model used to validate the accuracy and robustness of our proposed UNet model is ConvLSTM. LSTM is a Recurrent Neural Network (RNN) that focuses on learning long-term dependencies. A series of memory blocks make up the LSTM architecture. Each block has one or more self-contained memory cells, as well as three gates: input, forget, and output. The input gate receives new data from the outside and processes it. The forget gate determines when to forget the initial state and, as a result, the input sequence's ideal time lag. The output gate is responsible for generating output for the LSTM cell by combining all the computed results. ConvLSTM is a recurrent layer, except convolution operations are used instead of internal matrix multiplications. As a consequence, instead of being a 1D vector containing features, the data that travels through the ConvLSTM cells retains the input dimension (3D in our case). ConvLSTM has been proven in recent literature that it is capable of handling the spatial temporal dependence in traffic data, however, due to its complex structure it has a longer training time.

MDA (Images) is used as the model input. Figure 9 presents the model architecture, and Table 4 presents the input parameters. The shape of data is presented in the following format: (samples, frames, channels, rows, cols). The final input format is when the frames are limited to 1000 per sample, and the image is an eight-channel 495x436 pixel picture (samples, 288, 8, 495, 436). The number of available trailers for training is referred to as samples. `return_sequences` is set to True, which means the output should be (samples, frames, categories), but because the model has eight separate outputs, the result should be (categories, samples, frames, 1), implying (8, samples, 1000, 1). Return sequences have the effect of classifying each frame into several categories. The model architecture begins with two ConvLSTM layers, each with a BatchNormalization and a MaxPooling layer in between. It breaks into branches in order, one for each category. All branches start with one ConvLSTM layer and then a MaxPooling layer. The output is then linked to a Dense network that is completely connected. Finally, the final layer is a Dense single-cell.

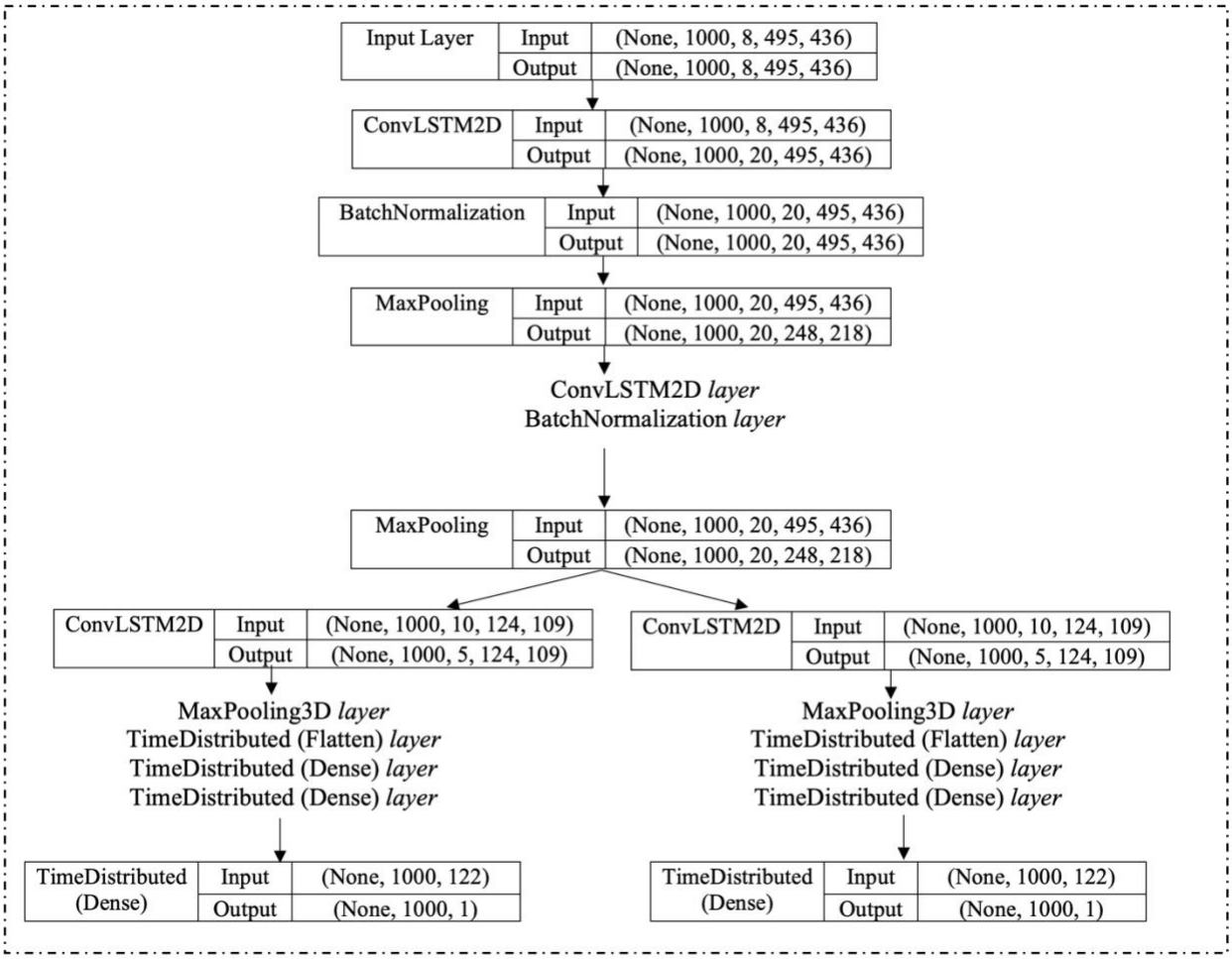



**Figure 9: ConvLSTM architecture for two catgeories**

| | | |
|---|---|---|
| No. of frame sequence | 24 | Used time frames (12) + Frames to predict (12) |
| No. of frames output | 12 | Subset to predict |
| Height | 495 | Image height |
| Width | 436 | Image width |
| No. of channels | 8 | Speed (4 directions) + Volume (4 directions) |
| No. of channels output | 8 | Speed (4 directions) + Volume (4 directions) |
| Batch size | 2 | No. of samples processed |
| Number of epochs | 20 | No. of complete passes through the training dataset |
| Activation | Relu | Linear Function |
| Padding | same | The output will have the same size as the input |

**Table 4: LSTM Model input parameters**

### 3.7. Historical Average (HA) Model

The second benchmark model used to validate the accuracy and robustness of our proposed UNet model is a simple historical average model. HA simply uses the average of historical variables as predictions. We calculated the average at a spatial and temporal level for each variable/channel, meaning, data was filtered along each pixel and time bin for each day and then the average is calculated along all days. The formula used can be defined as:

$$x_{(T,j,k,z)} = \sum_{i=1}^{d} \frac{[T, W_j, H_k, C_z]}{d} \quad (5)$$

Where,

- $x_{(T,j,k,z)}$ represent the predicted pixel along a specific time-step ($T$);
- $W_j$ and $H_k$ are the pixel index along the tensor width and height, respectively;
- $C_z$ is the channel index and
- $d$ is the number of days used in the training model.

### 3.8. Training Model

The prediction model uses the previous hour (12 frames) to predict the future hour (12 frames). The output file is a tensor of the shape (12, 495, 436, 8). The first dimension of six represents the future 12 time-bins: 5min, 10min, 15min, 20min, 25min, 30min, 35min, 40min, 45min, 50min, 55min and 60min. The width of an image is 495, and the height is 436. Our main task is to forecast traffic conditions so the first eight channels (speed and volume, for each of the four headings) are forecasted. The ratio of data used for training, validation and testing is (0.8:0.1:0.1).

### 3.9. Testing Dataset

For the testing dataset, we perform the recursive multi-step forecast, selecting the last three days of available data to test the reliability of the forecasting model. We perform forecasting throughout all hours of the day, using an hour of actual data to predict the future hour and then using every new predicted hour for a newer prediction, as presented in Figure 10. The main prediction task is to test the UNet algorithm in predicting network-wide traffic speed and volume. Eventually, we forecast the traffic flow propagation throughout the day by performing a multi-step prediction. The previous hour (12 steps) of observed data is fed into the trained model to predict the next hour (12 steps), and then every new predicted hour is an updated input bin to predict the next hour.

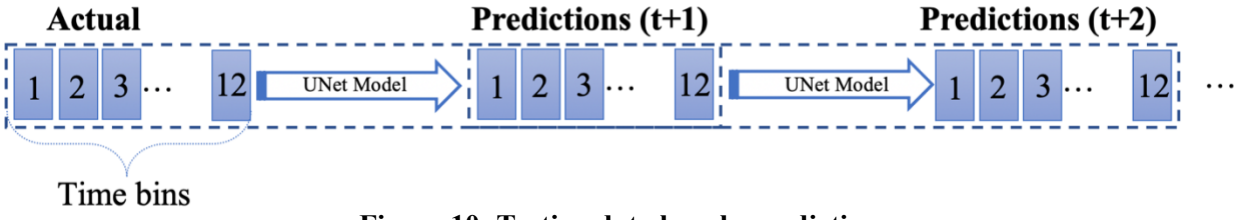



**Figure 10: Testing data hourly predictions**

## 4. RESULTS

### 4.1. Description of evaluation metrics

This section evaluates the performance of the trained UNet model against a test dataset which consisted of the last three days of data from the data collected for one month. In order to test the performance of the proposed algorithm, statistical and deep learning-based algorithms are chosen for comparison. HA and ConvLSTM neural network is used, an extension of Recurent Neural Network (RNN), which is more popular due to its capability to deal with longer-term memories and evade fading gradient problems that conventional RNNs suffer from [30].

First, we will present the general results for the UNet model performance compared to benchmark models: HA and ConvLSTM, followed by a visual comparison of a few images exported from the results of each model and a deeper dive into the UNet model results. While forecasting CV speed and volume, errors from

the models are calculated from the observed CV speed and volume and shall be used to justify forecasting results. Figure 11 summarizes the model performance along:

- Peak hours which represents only pixels with data (excluding zeros) and hours from 6-9 and 16-20,
- Non-peak hours also represent only pixels with data (excluding zeros) and excluding peak hours.

Root Mean Squared Error (RMSE) is the performance metric we use in evaluating our model because of its very intuitive statistic interpretation in terms of having the same measurement unit as the variable predicted, with smaller RMSE values indicating higher model accuracy. The formula can be defined as:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \quad (3)$$

Where,

- $\hat{y}_i - y_i$ represents the difference between actual and predicted values and
- $n$ represents the number of samples

SSIM is also used when comparing images exported from each model since it is a more indicative metric that can reflect perceived structural similarity by taking image texture into account [29]. Structural similarity refers to the assumption that pixels have many interdependencies, especially when close together. SSIM values closer to (1) indicate higher similarity, while (-1) indicate lower similarity.

$$SSIM\ (x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (4)$$

Where,

- $\mu_x$ is the mean of *x;*
- $\mu_y$ is the mean of *y;*
- $\sigma_x^2$ is the variance of *x;*
- $\sigma_y^2$ is the variance of *y;*
- $\sigma_{xy}$ is the covariance of *x* and *y;*
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ are two variables that stabilize the division*;*
- $L$ is the dynamic range of pixel-values and
- $k_1$ and $k_2$ are 0.01 and 0.03, respectively, by default.

### 4.2. General results across prediction models

In terms of RMSE, the performance of models across all subsets can be seen in Figure 11 and ranked: UNet, ConvLSTM and HA, where UNet saw an average improvement of 65% over HA model and 15% over ConvLSTM. UNet significantly outperforms the other models because it applies a considerable amount of kernels to each image to perform the dense predictions at a pixel level. Ultimately, this leads to a lower RMSE across volume and speed channels, too, though the significance of error varies enormously (Speed - UNet peak NZ: 7 kph, Volume - UNet peak NZ: 1 vehicle). The reason for this is relatively simple: speed channels are normalized from 1 to 255 while volume is normalized from 0 to 255. As a result, incorrect speed forecasts are more likely to be penalized (for example, volume, which is usually close to zero for most pixels). Generally, both channels' forecasts along arterials were better than freeways, reasoned by the higher density of data points (pixels) on arterials than freeways. RMSE peaks occur during peak hours (bin

72: hour 6:00) and (bin 192: hour 16:00), reflecting the model's challenging task with higher volume around peak hours.

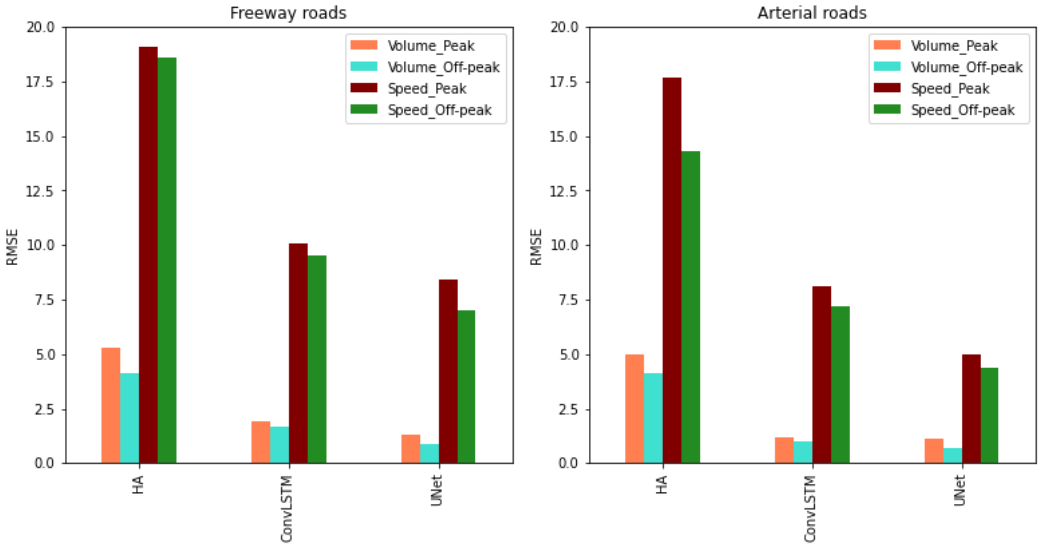



**Figure 11: RMSE results across various models: (a) freeway roads and (b) arterial roads**

### 4.3. Evaluation of predicted images

Figure 12 presents a few images exported from the forecasting results of the models. The forecasting snippet is for hours: 5:00, 6:40, 17:00 and 20:00. Structural Similarity Index (SSIM) and RMSE are presented above each image exported from the model and calculated concerning the observed image. The count of non-zero pixels for each image is presented below each image to analyze performance concerning spatial granularity. In terms of results, the forecasting models need to decide the non-zero positions through a map with 215,820 spaces, which is a challenging assignment because the model input state of traffic could be reduced or expanded spatially. The performance of the UNet model was dominant in predicting closer non-zero pixels, higher SSIM and lower RMSE, followed by ConvLSTM and HA models. UNet exhibits an excellent learning ability in comprehending images because of its locally linked layers which means that output neurons are linked to local adjacent input layers, rather than all input neurons in fully-linked layers. The pooling mechanism in the UNet model also enhances the model to retain the essential image features while efficiently reducing the number of used parameters.

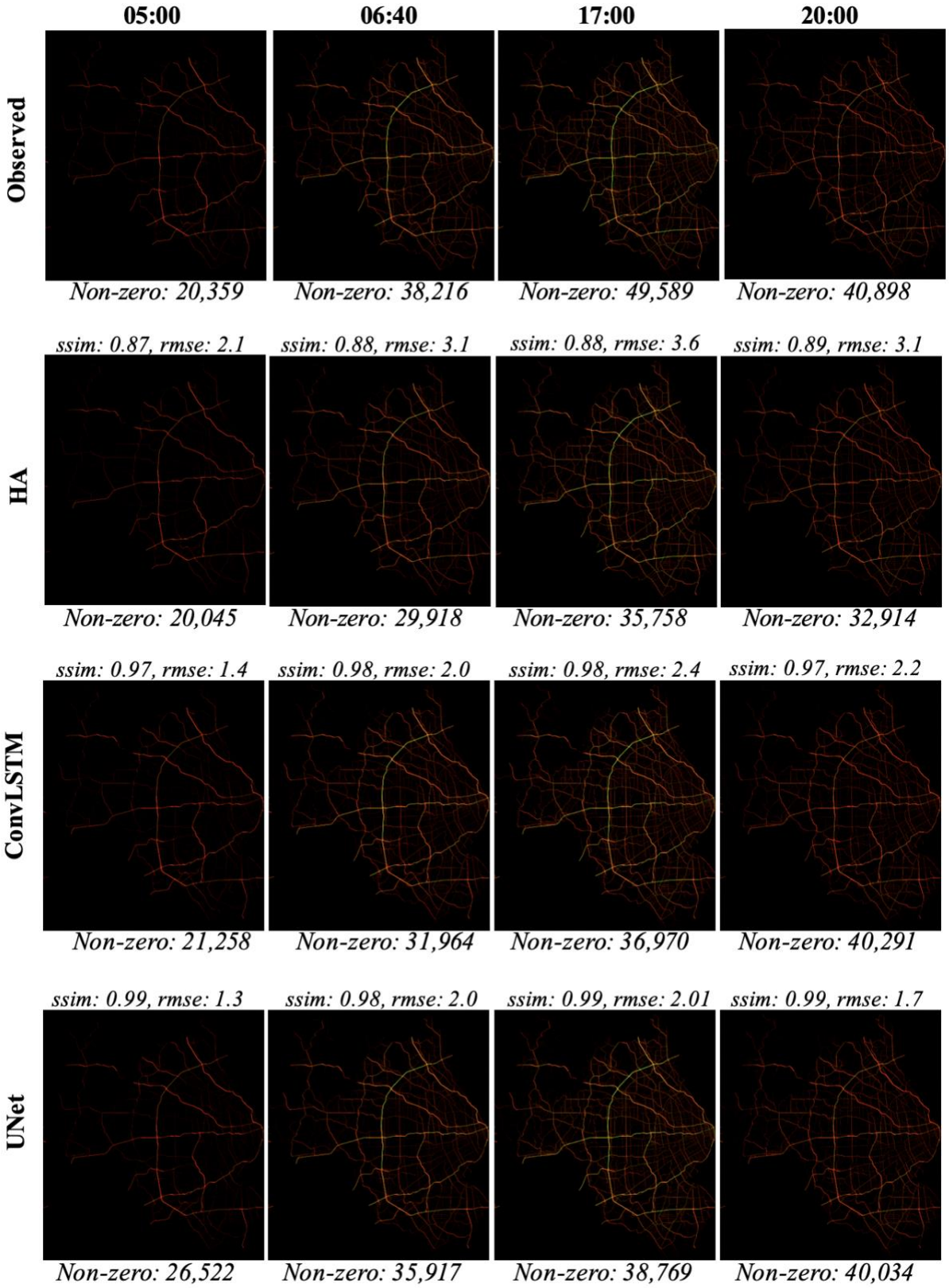



**Figure 12: Forecasted snippets from prediction algorithms**

### 4.4. Influence of forecasting horizon

To understand the influence of the length of the forecasting period and road type on our proposed forecasting mUNet model, we present Figure 13 as a box plot analysis of the change in RMSE along 12 future time steps averaged for the entire day forecast. Box plots provide a standardized way of interpreting the distribution of errors based on the minimum, maximum, median, 25$^{th}$ and 75$^{th}$ percentiles and the outliers. RMSEs for all plots increase over the length of prediction time steps, indicating a positive association between prediction errors and the span of prediction length. The median of RMSE on Arterials is very close for volume and speed channels with minimal deviations. For Freeways, the number of time steps is larger than 6, RMSE deviations start increasing and are much larger than other cases. The number of prediction horizon time-steps tend to influence performance in such a case. It is worth noting that generally, the errors and range of errors throughout the forecasting period was pretty stable with insignificant increases, which implies that the proposed model was robust in learning temporal features achieving the most accurate forecasts in all circumstances.

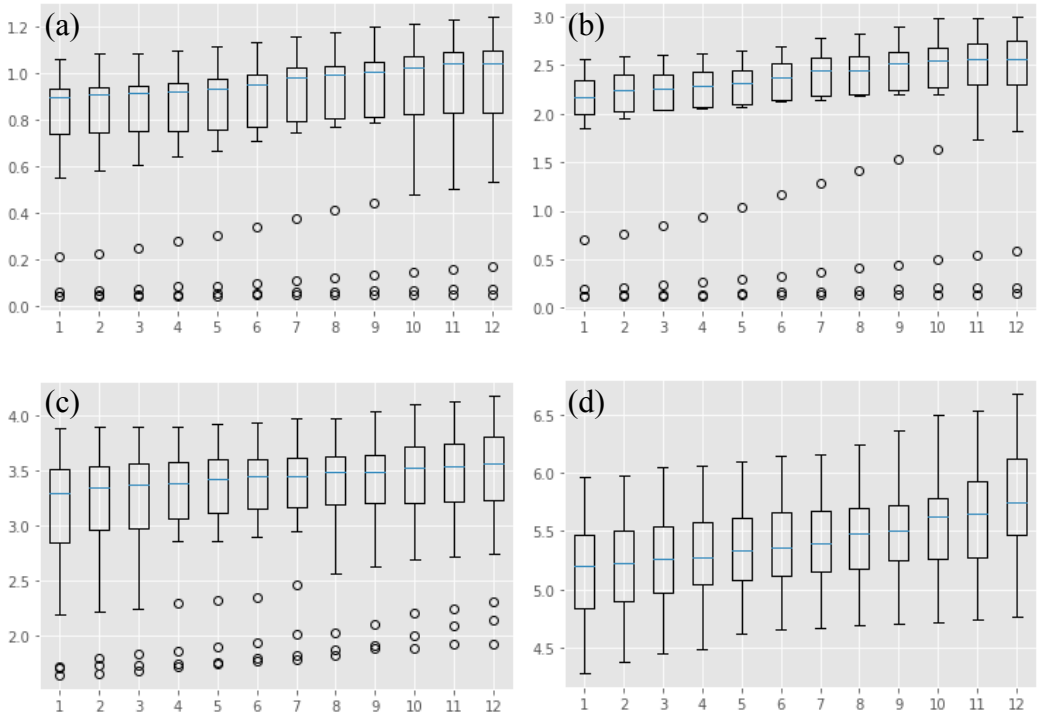



**Figure 13: RMSE Box plots along 12 future time steps: (a) volume channel on aterials, (b) volume channel on freeways, (c) speed channel on arterials, (d) speed channel on freeways**

### *Additional Experiments*

We looked into additional possibilities that may help us make more accurate forecasts such as using an additional input channel (incidents) for forecasting the main channels (speed and volume). Additional channels or traffic variables could provide useful information about a particular place and aid in the development of a more realistic model. Initially, we experimented with using all three channels (speed, volume and incidents) as input features versus solely using two channels only (speed and volume). However, the performance benefit from the incidents channel appears to be negligible. We decide to incorporate it into the input feature anyway because it does not appear to harm performance, but the value in terms of the final performance appears to be extremely limited in our testing.

We also experimented with different encoder and decoder structures. Instead of using average pooling in the encoder, we implemented a convolution pooling layer in two other models. We added a linear interpolation layer in parallel path to one of the additional models in addition to the deconvolution layer, which may be thought of as the inverse of the average pooling layer. The decoder block also includes tightly coupled convolution layers. We cannot assert that the additional trials are superior to the finally implemented model based on test set assessment scores alone. Performance varies per training iteration, but there is no noticeable difference in terms of performance between them in general.

## 5. CONCLUSION

Prediction of traffic flow has seen a rich use of deep learning methods, which yielded satisfactory results. These approaches can perform dense predictions and portray more non-linear functions than other neural networks [31,32]. However, most of these studies address a single step, channel or route prediction. A multipurpose, multi-step, spatiotemporal forecasting is necessary to improve the accuracy of predictions and provide a longer prediction length into the future.

This study proposed a data fusion technique and an image-based forecasting model to perform multipurpose, multi-step predictions. This enabled state-of-the-art ML models to efficiently learn from large-scale multisource data while considering the network traffic temporal evolutions and spatial dependencies. The first procedure involves processing statewide CV traffic and sensor data on GPUs that leads to real-time insights on the micro-scale in temporal and spatial dimensions on GPUs using the Nvidia Rapids framework and Dask parallel cluster in Python. The features of the network are preserved because adjacent roads are also adjacent in the MDA. Our results show a 50x speed-up in the ETL of the CV data for an entire day for all the unique CV journeys, reducing the processing time from 48 hours to 15 minutes. The second procedure was to perform predictions using our UNet model, which successfully achieved image-learning assignments. In the scope of this study, the UNet model has the following properties: (a) space and time features can be extracted automatically because of the implementation of convolutional and max-pooling layers; and (b) represents speed, volume and incident features on a pixel-level dense traffic network that are then used to create traffic speed and volume predictions on all routes. The testing model used one hour of actual data to forecast all future hours. To test the applicability of the proposed model and its performance, the comparison to HA statistical method and ConvLSTM saw an average improvement of 65% and 15%, respectively. The image snippets from each prediction model to the actual image showed that image textures were more similar in UNet than the benchmark models used. UNet's dominance in performing image predictions was also evident in multi-step forecasting, where the increase in errors was relatively minimal over longer prediction spans.

A possible extension can include using higher penetration rates for CV data over longer periods of time and using/predicting extra traffic features such as weather and incidents. Higher penetration rates can aid the prediction model in performing better especially during peak hours demand due to the availability of higher amounts of data. Another exciting extension can use data augmentation which can allow the prediction models to make the model more robust by adding variables that the model would encounter in the actual world. While the model can be scaled to other datasets, transfer learning on a different unseen study area using the weights of convolution layers and pooling mechanism can provide opportunities to test the model on a spatiotemporal domain shift and leverage knowledge from the current trained model which inturn enhances efficiency and saves computational resources.

The generated model not only produced encouraging forecast accuracy, but the visualization of the input data structure and model architecture challenges the common perception of neural networks in the transportation area, which is that they are solely a "black-box" model. Most existing traffic flow prediction research, to our knowledge, focuses on finding models with higher prediction accuracy; however, this work not only provides a long-term prediction model with trustworthy accuracy, but also examines the underlying process of structuring network-wide data. It provides a different way of thinking about structuring large-scale point data to forecast high-level traffic features.

## REFERENCES


[1] P. Xie, T. Li, J. Liu, S. Du, X. Yang, and J. Zhang, "Urban flow prediction from spatiotemporal data using machine learning: A survey," Information Fusion, 2020.
[2] 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," Transportation Research Part C: Emerging Technologies, vol. 54, pp. 187–197, 2015.
[3] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? estimating travel time based on deep neural networks," in AAAI, 2018.
[4] J. Ye, J. Zhao, K. Ye, and C. Xu, "How to build a graph-based deep learning architecture in traffic domain: A survey," IEEE Transactions on Intelligent Transportation Systems, pp. 1–21, 2020.
[5] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial temporal graph convolutional networks for traffic flow forecasting," in AAAI, pp. 922–929, 2019.
[6] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in ICLR, 2018.
[7] X. Zhou, Y. Shen, and L. Huang, "Revisiting flow information for traffic prediction," arXiv:1906.00560, 2019.
[8] L. Ge, H. Li, J. Liu, and A. Zhou, "Temporal graph convolutional networks for traffic speed prediction considering external factors," in MDM, pp. 234–242, 2019.
[9] J. Sun, J. Zhang, Q. Li, X. Yi, and Y. Zheng, "Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks," IEEE Transactions on Knowledge and Data Engineering, 2020.
[10] S. Fang, Q. Zhang, G. Meng, S. Xiang, and C. Pan, "Gstnet: Global spatial-temporal network for traffic flow prediction," in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, pp. 2286–2293, 2019.
[11] J. J. Q. Yu and J. Gu, "Real-time traffic speed estimation with graph convolutional generative autoencoder," IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 10, pp. 3940–3951, 2019.
[12] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, "Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting," in AAAI, pp. 890–897, 2019.
[13] B. Yu, H. Yin, and Z. Zhu, "St-unet: A spatio-temporal u-network for graph-structured time series modeling," arXiv:1903.05631, 2019.
[14] C. Chen, K. Li, S. G. Teo, X. Zou, K. Wang, J. Wang, and Z. Zeng,"Gated residual recurrent graph neural networks for traffic prediction," in AAAI, pp. 485–492, 2019.
[15] J. Li, Z. Han, H. Cheng, J. Su, P. Wang, J. Zhang, and L. Pan, "Predicting path failure in time-evolving graphs," in KDD, pp. 1279–1289, 2019.
[16] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in IJCAI, 2019.
[17] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in IJCAI, pp. 3634–3640, 2018.
[18] Y. Zhang, S. Wang, B. Chen, and J. Cao, "GCGAN: generative adversarial nets with graph CNN for network-scale traffic prediction," in IJCNN, pp. 1–8, 2019.
[19] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in AAAI, 2020.
[20] Z. Zhang, M. Li, X. Lin, Y. Wang, and F. He, "Multi-step speed prediction on traffic networks: A deep learning approach considering spatiotemporal dependencies," Transportation Research Part C: Emerging Technologies, vol. 105, pp. 297–322, 2019.
[21] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in AAAI, vol. 33, pp. 3656–3663, 2019.
[22] K. Guo, Y. Hu, Z. Qian, H. Liu, and e. Zhang, "Optimized graph convolution recurrent neural network for traffic prediction," IEEE Transactions on Intelligent Transportation Systems, pp. 1–12, 2020.

[23] J. Hu, C. Guo, B. Yang, and C. S. Jensen, “Stochastic weight completion for road networks using graph convolutional networks,” in ICDE, pp. 1274–1285, 2019.
[24] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D. Yeung, “Gaan: Gated attention networks for learning on large and spatiotemporal graphs,” in UAI, pp. 339–349, 2018.
[25] B. Yu, M. Li, J. Zhang, and Z. Zhu, “3d graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting”, 2019.
[26] C. Zhang, J. J. Q. Yu, and Y. Liu, “Spatial-temporal graph attention networks: A deep learning approach for traffic forecasting,” IEEE Access, vol. 7, pp. 166 246–166 256, 2019.
[27] J. Ye, J. Zhao, K. Ye, and C. Xu, “Multi-signet: A graph convolution based spatial-temporal framework for subway passenger flow forecasting,” in 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8, 2018.
[28] Retrieved from: https://support.hdfgroup.org/HDF5/doc/H5.intro.html
[29] Retrieved from: https://scikit-image.org/docs/dev/auto_examples/transform/plot_ssim.html
[30] Ma, X.; Tao, Z.; Wang, Y.; Yu, H.; Wang, Y. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transp. Res. Part C Emerg. Technol. 54, 187–197, 2015.
[31] Ma, X.; Tao, Z.; Wang, Y.; Yu, H.; Wang, Y. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transp. Res. Part C Emerg. Technol. 54, 187–197, 2015.
[32] Lv, Y.; Duan, Y.; Kang, W.; Li, Z.; Wang, F.-Y. Traffic flow prediction with big data: A deep learning approach. IEEE Trans. Intell. Transp. Syst. 16, 865–873, 2015.
[33] Chen, X., Chen, H., Yang, Y., Wu, H., Zhang, W., Zhao, J., & Xiong, Y. Traffic flow prediction by an ensemble framework with data denoising and deep learning model. Physica A: Statistical Mechanics and its Applications, 565, Article 125574, 2021.
[34] Li, W., Wang, X., Zhang, Y., & Wu, Q. Traffic flow prediction over muti-sensor data correlation with graph convolution network. Neurocomputing, 427, 50–63, 2021.
[35] Yu, J. J. Citywide traffic speed prediction: A geometric deep learning approach. Knowledge-Based Systems, 212, Article 106592, 2021.
[36] Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H., & Yin, B. Multi-stage attention spatial–temporal graph networks for traffic prediction. Neurocomputing, 428, 42–53, 2021.
[37] Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., & Li, H. T-GCN: A temporal graph convolutional network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems, 21, 3848–3858, 2020.
[38] Cui, Z., Ke, R., Pu, Z., & Wang, Y. Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction, 2018.
[39] Choi, S. Utilizing UNet for the future traffic map prediction tast: Traffic4cast challenge 2020’, 2020.
[40] M. Miller and C. Gupta, “Mining traffic incidents to forecast impact,” ser. UrbComp. New York, NY, USA: ACM, pp. 33–40, 2012.
[41] B. Pan, U. Demiryurek, and C. Shahabi, “Utilizing real-world transportation data for accurate traffic prediction,” ser. ICDM ’12. Washington, DC, USA: IEEE Computer Society, pp. 595–604, 2012.
[42] B. Pan, U. Demiryurek, C. Shahabi, and C. Gupta, “Forecasting spatiotemporal impact of traffic incidents on road networks,” in ICDM, pp. 587–596, 2013.
[43] A. Phusittrakool, C. Jeenanunta, and P. Prathombutr, “Evaluation of network performance under provision of short predictive traffic information,” Walailak J. Sci. Technol. (WJST), vol. 13, no. 6, pp. 433–450, 2015.
[44] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, “Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks,” Sensors, vol. 17, no. 7, p. 1501, 2017.
[45] J. Patnaik, S. Chien, and A. Bladikas, “Estimation of bus arrival times using apc data,” Journal of public transportation, vol. 7, no. 1, p. 1, 2004.

[46] M. Elhenawy, H. Chen, and H. A. Rakha, “Dynamic travel time prediction using data clustering and genetic programming,” Transportation Research Part C: Emerging Technologies, vol. 42, pp. 82–98, 2014.
[47] B. Yu, W. H. Lam, and M. L. Tam, “Bus arrival time prediction at bus stop with multiple routes,” Transportation Research Part C: Emerging Technologies, vol. 19, no. 6, pp. 1157–1170, 2011.
[48] F. Sun, Y. Pan, J. White, and A. Dubey, “Real-time and predictive analytics for smart public transportation decision support system,” in 2nd IEEE International Conference on Smart Computing, 2016.
[49] C. Bai, Z.-R. Peng, Q.-C. Lu, and J. Sun, “Dynamic bus travel time prediction models on road with multiple bus routes,” Computational intelligence and neuroscience, vol. p. 63, 2015.
[50] S. Shekhar, S. Pradhan, F. Sun, A. Dubey, and A. Gokhale, “Empowering the next generation city-scale smart systems,” IEEE 22nd International Conference on High Performance Computing Workshops (HiPCW), 2015.
[51] Lenzerini, Maurizio. “Data integration: A theoretical perspective.” In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, pp. 233–246 (cit. on pp. 5, 155), 2015.
[52] El Faouzi, Nour-Eddin, Henry Leung, and Ajeesh Kurian. “Data fusion in intelligent transportation systems: Progress and challenges–A survey.” In: Information Fusion 12.1, pp. 4–10 (cit. on pp. 4, 5, 154, 156), 2011.
[53] Yuankai Wu, Huachun Tan, Lingqiao Qin, Bin Ran, and Zhuxi Jiang. 2018. A hybrid deep learning based trafc fow prediction method and its understanding. Transportation Research Part C: Emerging Technologies 90, 166–180, 2018.
[54] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. Revisiting spatial-temporal similarity: A deep learning framework for trafc prediction. In AAAI. 5668–5675, 2019.
[55] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, Yunpeng Wang, Long short-term memory neural network for traffic speed prediction using remote microwave sensor data, Transportation Research Part C: Emerging Technologies, Volume 54, Pages 187-197, ISSN 0968-090X, 2015.