# DeepFlow: Abnormal Traffic Flow Detection Using Siamese Networks

Sepehr Sabour, Sanjeev Rao and Majid Ghaderi
Department of Computer Science, University of Calgary
{sepehr.sabour, sanjeev.rao, mghaderi}@ucalgary.ca

***Abstract*—Nowadays, many cities are equipped with surveillance systems and traffic control centers to monitor vehicular traffic for road safety and efficiency. The monitoring process is mostly done manually which is inefficient and expensive. In recent years, several data-driven solutions have been proposed in the literature to automatically analyze traffic flow data using machine learning techniques. However, existing solutions require large and comprehensive datasets for training which are not readily available, thus limiting their application. In this paper, we develop a traffic anomaly detection system, referred to as DeepFlow, based on Siamese neural networks, which are suitable in scenarios where only small datasets are available for training. Our model can detect abnormal traffic flows by analyzing the trajectory data collected from the vehicles in a fleet. To evaluate DeepFlow, we use realistic vehicular traffic simulations in SUMO. Our results show that DeepFlow detects abnormal traffic patterns with an $F_1$ score of $78\%$, while outperforming other existing approaches including: Dynamic Time Warping (DTW), Global Alignment Kernels (GAK), and iForest.**

## I. Introduction

**Motivation.** Driving safety continues to be a challenging problem in city management. Based on a road safety plan published by *Canadian Council of Motor Transport Administrators*, about 2000 people are killed, and $165,000$ are injured in car accidents annually in Canada [1]. Distracted driving leads to unusual actions such as sudden accelerations, decelerations, and lane changes that other vehicles cannot predict, which can result in collisions. Technology is expected to play a significant role in road safety to lead the transportation system toward zero fatal accidents [1]. An Intelligent Transportation System (ITS) makes use of technologies such as vehicular networks, cloud computing, and artificial intelligence to solve traffic flow problems. For example, ITSs employ Vehicle to Everything (V2X) communications [2] to collect information related to vehicles, pedestrians, and road conditions. This data can be used to analyze traffic flows and driver behavior in order to detect abnormal driving patterns. Detecting abnormal behavior in vehicular traffic, apart from improving transportation safety, significantly impacts evaluation of driving skills, including that of autonomous vehicles. For instance, insurance companies can base their premiums on one's driving behavior. Also, understanding the misbehavior of autonomous vehicles can help analyze the risks involved in detaching human drivers from vehicles.

This work was supported by Wedge Networks Inc., Alberta Innovates and Natural Sciences and Engineering Research Council of Canada.

The emergence of Machine Learning (ML) has paved the way for more efficient solutions for many of the challenges faced in various fields such as fraud detection, cyberattack prevention and anomaly detection. ML solutions can help reduce the system dependence on human-in-the-loop processes in order to boost performance and reduce cost. Traffic management in cities can benefit from this idea too. A modern traffic control center is equipped with several display devices to monitor daily traffic flows in a city. The operators in these centers continuously watch for abnormal events on the roads to make sure traffic flows are steady and safe. In addition, these centers provide helpful information to the emergency units in case of accidents. However, manually checking traffic cameras in a city is inefficient and expensive. An automated system to check for traffic anomalies is essential for continuous and real-time analysis of vehicular traffic.

Existing solutions such as [3]–[7] use a dataset of normal driving patterns, and mark any unseen pattern as an anomaly. However, several factors like weather condition, road side constructions and traffic load can change the behavior of vehicles. Therefore, a huge dataset of normal patterns is required, which is not easy to acquire. Other solutions rely on finding outliers in traffic flows [8], [9]. These approaches are based on two assumptions. First, the driving data (*e.g.* trajectory, speed, and acceleration) of abnormal vehicles diverges from normal ones. Second, normal vehicles form the majority of a given traffic flow. However, the behavior of normal vehicles in a fleet varies over time, which makes it challenging for such approaches to distinguish between normal and abnormal patterns. For example, drivers may usually drive within $\pm 10\%$ of the speed limit and still be considered to be driving normally.

**Our Work.** In this work, we introduce DeepFlow, an anomaly detection system which detects abnormal traffic flows by *analyzing vehicle trajectories in a fleet*. We use a small set of normal cases to train our model, and test it with a dataset containing previously unseen patterns. We show that DeepFlow can address the challenges faced by existing approaches. The model learns the similarity between vehicles, assigns a score based on that, and classifies flows based on this similarity score. Our realistic experiments show DeepFlow is effective *even when a comprehensive dataset is not available for training*.

The main idea behind our solution is that vehicles in a normal traffic flow have similar trajectory data, so the presence

of any abnormal cases can be detected by measuring the average similarity between vehicles. For example, when a car drives with a higher speed and acceleration than other ones or abnormally changes lanes, its trajectory data is different than others. To measure this similarity, we use neural networks to compress the data-series from each vehicle into a latent vector, and we measure the distances between them.

Our main contributions in this paper are:

- We present the design of DeepFlow, a traffic anomaly detection system, and describe its various components including data collection, anomaly detection and application.
- We design, implement and evaluate a semi-supervised Siamese network to measure the abnormality score of traffic flows within a fleet of vehicles.
- We use realistic traffic simulations in SUMO to obtain datasets for training and testing DeepFlow. Our results show that DeepFlow detects abnormal traffic patterns with 78% $F_1$ score and outperforms other existing approaches including: Dynamic Time Warping (DTW), Fast Global Alignment Kernels (GAK) and iForest.

**Paper Organization.** We review recent work on traffic anomaly detection in *Section II*. The design of DeepFlow is discussed in *Section III*. The anomaly detection engine used in DeepFlow is presented in *Section IV*. In *Section V*, we describe the simulation process and our datasets. Evaluation results are presented in *Section VI*, while *Section VII* concludes the paper.

## II. Related Works

In the following, we briefly review several representative papers on traffic anomaly detection that are most relevant to our work. We categorize available works into **history-based** and **outlier-detection** approaches.

**History-Based Approaches.** In this approach, the behavioral history of vehicles is used to check for the presence of anomalies in real-time. For instance, SafeDrive [3] is a driving anomaly detection approach that uses historical data to generate a state graph in which states represent the value (or its range) of sensor data, and weighted edges show the likelihood of transitions between the states. At the beginning of the driving path, the vehicle is in the starting state, and by receiving the real-time driving events, the state changes. One can measure the driver's anomaly score by aggregating the weight of the traveled edges. Similarly, authors in [4] employ a graph-based approach and reinforcement learning techniques to detect abnormal trajectories.

Neural networks such as autoencoders [10] and LSTMs [11] have also been used for history-based anomaly detection. For example, a technique for detecting anomalies using autoencoders is proposed in [5]. Similarly, authors in [6] propose an anomaly management system which uses autoencoders to find abnormal drivers in a collaborating transportation system. Driving behavior prediction is another approach to identify anomalies. To give an example, authors in [7] apply two different solutions containing a recurrent neural network

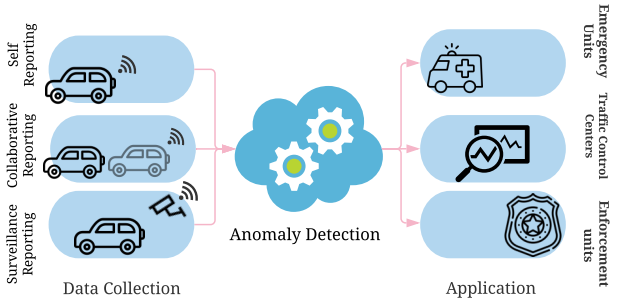


Fig. 1: High-level design of DeepFlow.

(RNN) and a long short-term memory (LSTM) to predict driver's actions, and mark behaviors that are varying from the predicted ones.

**Outlier-Detection Approaches.** These approaches compare the behavior of vehicles and mark outliers. For instance, in [8], three ML algorithms, namely Support Vector Machine (SVM), Isolation Forest (iForest), and K-Nearest Neighbors (K-NN), are used to detect outlier drivers. Also, the authors of [9] present a reckless driver detection framework which uses vehicular collaboration to collect data and then apply support vector machine (SVM) and decision-tree models to measure every vehicle's driving performance.

In general, history-based approaches require a large dataset. On the other hand, SVM and K-NN are supervised approaches, which need a dataset containing abnormal cases for training. However, DeepFlow can operate with a small dataset, and because it is a semi-supervised method, no datasets containing abnormal cases are required for training.

## III. DeepFlow Design

Fig. 1 shows the high-level design of DeepFlow. As shown in the figure, DeepFlow contains three components including data collection, anomaly detection and application. In the following, we describe each of these components.

### A. Data Collection

We feed DeepFlow with time-series data collected from a group of vehicles; this contains information such as speed, location and steering angle. The data can be gathered using one or a combination of the following approaches:

**Self Reporting.** Most modern vehicles are equipped with sensors and onboard communication devices due to the industry-wide push towards more automated vehicles. These sensors measure the speed, location, and gap between a vehicle and the surrounding objects. Vehicles can report their data to the server for anomaly detection. The self-reported data to the server is more accurate than other approaches. Nevertheless, it demands that all vehicles have the required hardware. Also, an abnormal vehicle can manipulate or avoid sending data to stay hidden from the anomaly detection system.

**Collaborative Reporting.** In this approach, each vehicle measures the state of the adjacent vehicles and reports it to the server. The information inferred by the other neighbors

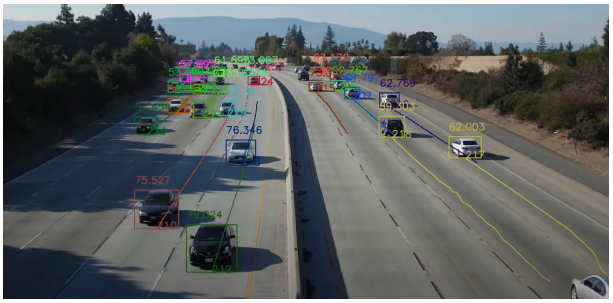


Fig. 2: Tracking vehicles using a surveillance camera [12], [13].

may be inaccurate; however, participating vehicles can reach a consensus on its validity using vehicle to vehicle (V2V) communication. Therefore, the system understands the correct state of the traffic flow even if an adversarial vehicle blocks or changes the data. However, like any other wireless network solution, jamming and corrupting the communication signals is possible.

**Surveillance Reporting.** Surveillance systems such as traffic cameras and road sensors can capture information like the speed and location of the vehicles. Image processing techniques can be used to process the collected video feeds and extract traffic flow information. For example, Fig. 2 shows a demo of such a vehicle tracking system [12], [13]. Despite the processing overhead of this approach, it does not require any hardware installed on the vehicles. Also, since there is a direct connection between the surveillance devices and the cloud server, it has a lower security risk.

### *B. Anomaly Detection*

At the core of DeepFlow is a Siamese network, as depicted in Fig. 3. A Siamese network contains two identical neural networks with the same weights. This model is used to measure the similarity of two vectors by feeding them to the twin networks and comparing their outputs [14]. Siamese networks are very useful in applications where no comprehensive dataset exists for training. The anomaly detection component applies a pre-trained machine learning model on each vehicle's trajectory data and outputs an anomaly score for it. The detector can function on a cloud or an edge server. For the sake of privacy requirements and decreasing the cost of storage requirements, the server can eliminate all the processed data after making the required computations. In *Section IV*, we delve into the structure of the applied machine learning model in this component.

### *C. Application*

The output of the anomaly detection can be used in multiple applications as described below.

**Traffic Flow Analysis.** By employing this system we can measure the average abnormality score of traffic flows in streets, junctions and highways, which can help optimize the vehicular movements in these areas.

**Enforcement of Traffic Laws.** Enforcement units make use of this system to detect aggressive and distracted drivers. The detected abnormal drivers can be penalized based on their anomaly scores.

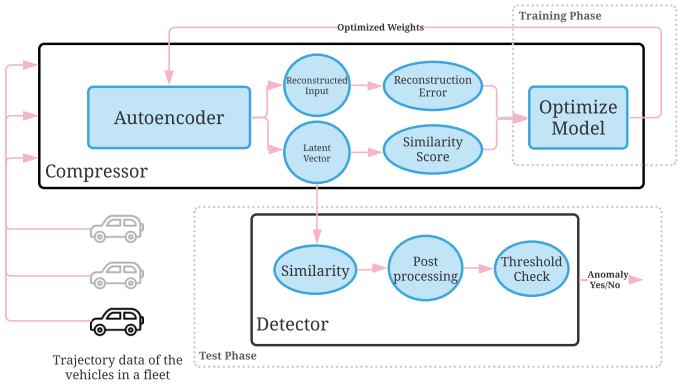


Fig. 3: Structure of DeepFlow's Siamese network.

**Emergency Situation Detection.** Traffic accidents can affect the abnormality score of a flow, which makes DeepFlow able to mark sudden changes in scores and notify nearby emergency units.

This paper mainly focuses on implementation and evaluation of the anomaly detection component. Therefore, elaboration of the data collection and application components is out of the scope of this article.

## IV. Anomaly Detection Engine

DeepFlow employs a semi-supervised machine learning approach based on a Siamese network to detect abnormal traffic patterns. Fig. 3 shows the architecture of this network. The input of the network is a set of trajectory data-series collected from vehicles driving in a fleet. At the first step, the *Compressor* converts the data from each vehicle to a latent vector. Then, the *Detector* measures the distance between the vectors, specifies a similarity score for each one, and finally measures the abnormality score of the flow. In the following sub-sections, we explain the architecture, objectives, and functionality of each component in detail.

### *A. Compressor*

The compressor is intended to shrink the input into a compressed "latent representation", which is accomplished by using an autoencoder (AE). An autoencoder is a type of artificial neural network trained to compress the input and then decompress it with minimal distortion. These neural networks are composed of two parts, an **encoder** that imposes a bottleneck and compresses the data, followed by a **decoder** which reconstructs the input from the compressed representation [10].

Usually, the only objective of an AE is to minimize the reconstruction error. In our system, we use the Mean Squared Error (MSE) to measure the error. Also, to simplify the learning process, we use a $\tanh$ function to limit the error value in a range between 0 and 1. Thus, the reconstruction loss function in DeepFlow can be expressed as follows:

$$\mathrm{RLoss}(Y, \widehat{Y}) = \tanh\left(\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2\right), \qquad (1)$$

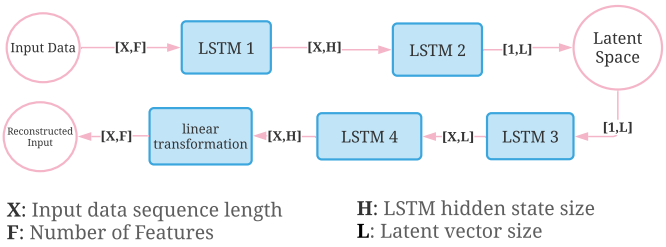


Fig. 4: Structure of DeepFlow autoencoder. The arrows show the flow of data and indicate the input and output dimensions at each point in the network.

where $Y_i$ and $\widehat{Y}_i$ are the $i$th actual and the $i$th reconstructed values of the input vector with size $n$, respectively.

To use the latent representation for outlier detection, we consider the compressed representation of an outlier pattern to be different from the regular patterns. In the training phase, we use a dataset containing only normal trajectories to train our model to maximize the similarity of latent vectors. Hence, we add a second error function, similar to the first one, to measure the distance between latent spaces:

$$\mathrm{Sim}(L_i, L_j) = \tanh\left(\frac{1}{n}\sum_{k=1}^{n}(L_i[k] - L_j[k])^2\right), \qquad (2)$$

where $L_i$ and $L_j$ are the latent vectors, created for vehicles $i$ and $j$, respectively. In DeepFlow, we calculate the aggregated loss in each training iteration and optimize the neural networks to minimize this value. We use the following expression to determine the aggregated loss:

$$\begin{aligned}\mathrm{Loss} =& \frac{1}{m}\sum_{i=1}^{m}\mathrm{RLoss}(Y_i, \widehat{Y}_i) \\ &+ \frac{2\lambda}{m(m-1)}\sum_{i=1}^{m}\sum_{j=i+1}^{m}\mathrm{Sim}(L_i, Lj),\end{aligned} \qquad (3)$$

where $Y_i$, $\widehat{Y}_i$ and $L_i$ are the input, reconstructed input, and latent representation of vehicle $i$, respectively. Also, $m$ is the number of vehicles in the fleet, and $\lambda$ indicates the importance of reconstruction accuracy over the similarity score.

As shown in Fig. 4, DeepFlow's autoencoder consists of four LSTMs and one linear neural network. LSTMs use a shared state between the nodes to remember the changes in the input over time. In each step, a sequence of mathematical processes decides which part of the data should be remembered through the shared state and which part should be eliminated [11]. The first two networks in our autoencoder change the dimension of the input data and convert it to the latent vector. Then, the next two LSTMs increase the size of the vector; finally, the linear neural network unit reconstructs the input. The compressor unit has two outputs: the latent representation and the reconstructed input. In the training phase, we use both outputs to optimize the model using (3). However, we only use the first output in the testing phase and pass the latent vector to the detector unit.

### B. Detector

After training the compressor, the system can be used to detect anomalies. The Detector compares the latent space of each time-series data using an MSE function which is similar to the loss function used in the training phase. We use the following expression to calculate the abnormality score of traffic flows:

$$\mathrm{AS} = 1 - \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j=i+1}^{m} Sim(L_i, Lj)\,. \qquad (4)$$

After calculating the score, we compare it with a threshold to decide whether the traffic is normal or not. A high score suggests low similarity between the behavior of the vehicles, which indicates an anomaly is present. Therefore, finding an optimal threshold is critical. In *Section IV*, we study two approaches for finding a suitable threshold.

### C. Implementation

The model is implemented in Python using PyTorch as explained below.

**PyTorch.** To implement the model, we use *Python3* programming language and *PyTorch* [15]. *PyTorch* is an open source deep learning library which facilitates building ML models by providing basic machine learning modules such as LSTMs and linear neural networks. We use the predefined models in PyTorch to construct the Siamese network [1] in DeepFlow.

**PyTorch Lightning.** This library [16] allows us to run neural network models on any hardware (CPU, GPU, TPU) with no changes required in the source code. This feature is helpful as our development is done on a desktop computer, while model training is conducted on an Ubuntu Linux server with two V100 GPUs.

**Weights & Biases.** One of the main challenges in this work was finding the proper number of epochs for training our model. This is important because, first, we try to find the minimal loss value which is important to prevent the model from over-fitting the training dataset. Second, we should monitor the value of the two loss functions. At the beginning of the training phase, the value of the reconstruction error is significantly higher than the similarity error. After several iterations, the model learns to reduce the first loss value and proceeds to minimize the second one. Therefore, we need to monitor these values to stop the training after a suitable number of iterations. For this, we used *Weights & Biases (WandB)* [17] to monitor the training process.

## V. Traffic Flow Dataset

The input of DeepFlow is trajectory data collected from a group of vehicles in a fleet. Due to the difficulty in obtaining a dataset from a group of vehicles in an actual scenario, we generated the necessary datasets using simulation. The datasets used in our evaluations are generated using the road traffic simulator software *Simulation of Urban Mobility (SUMO)* [18],

[1] Our source code is available at: https://github.com/pesehr/DeepFlow

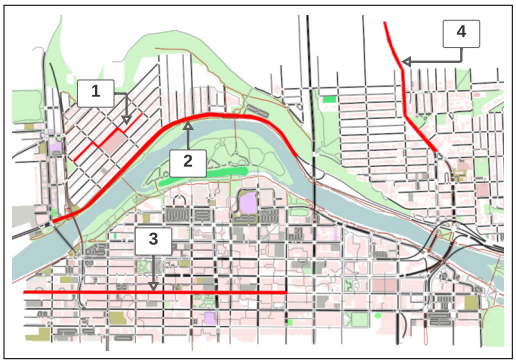


Fig. 5: Simulated City of Calgary traffic flow using SUMO. The highlighted paths are used for generating the test dataset.

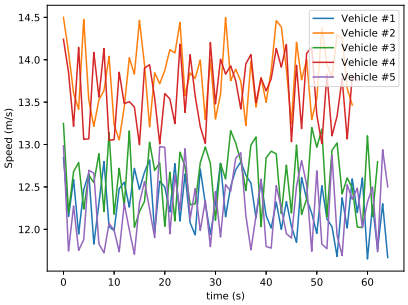


Fig. 6: Constant speed limit.

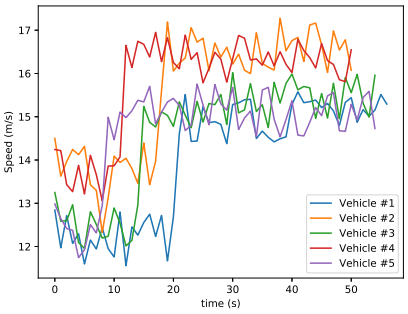


Fig. 7: Speed limit raise.

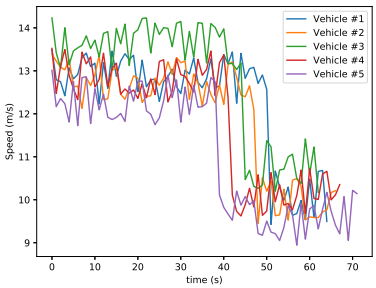


Fig. 8: Speed limit decline.

Fig. 9: Driving scenarios simulated for the training dataset.

which is an open-source traffic simulator capable of simulating different components involved in a traffic scenario such as roads, vehicles and pedestrians.

### A. Traffic Simulation

In order to use this software, the following elements should be specified:

- **Network File:** The location and shape of each road, junction, and sidewalk. Also, the network file indicates the traffic rules such as direction, priority, and speed limit of each path.
- **Traffic Demand File:** Determines how many vehicles are in the system and describes their behavior. Also, each driver's arrival and departure time, and the path taken by them should be defined in the traffic demand file.

SUMO includes several tools to help with the simulation process. We use the following tools in our work:

**OSMWebWizard.** A Python script implemented to work with *OpenStreetMap* [19] which extracts network data from the actual street map. In this work, we use it to simulate downtown Calgary and the surrounding regions. We select this area due to the variety of available streets. Fig. 5 shows the area of the city which is simulated.

**TraCI.** An interface implemented by SUMO to get data values from simulated vehicles and control their behavior. It is available as a Python library and employs a TCP-based client-server architecture to access the simulator [20]. We use *TraCI* to manipulate the speed of the vehicles and create the scenarios that are used in our training dataset.

### B. Collected Data

We generated two different datasets for the training and testing process. The training dataset contains trajectory data of 6660 normal vehicles (across 1332 groups of cars) driving in a straight street. The initial assigned speed of each vehicle is distributed randomly according to a Gaussian distribution (parameters are in Table I), and it changes based on one of the three scenarios in Fig. 9.

In the first scenario, we simulate a group of vehicles driving with a constant speed limit; the second and third involve an increase and decrease of the speed limit in a road. We use these scenarios to help our model learn how a group of normal vehicles changes their behavior when it is necessary.

We also use simulated traffic in four streets (highlighted in Fig. 5) with different shapes, number of lanes and speed limits in the city to test our model. Path #1 is located in a residential area and contains one lane with a speed limit of $40\,\mathrm{km/h}$. Paths #2 and #3 are two main streets with two lanes, with maximum permitted driving speed of $50\,\mathrm{km/h}$. Also, vehicles can drive up to $80\,\mathrm{km/h}$ in the three lanes of path #3.

This dataset contains the trajectory data of 16320 vehicles with 1020 abnormal cases. Each abnormal case consists of a vehicle either over-speeding or under-speeding (as shown in Table I). Similar to normal cases, the speed of each abnormal case is chosen randomly. Table I shows the parameters of the Gaussian distribution used for each speed class. The numbers in the table are multiplied by the speed limit of a street, for example the speed of a normal vehicle cannot exceed the street speed limit by more than 10%.

TABLE I: Speed classes for each vehicle.

| SpeedClass | $\sigma$ | $\mu$ | min | max |
|---|---|---|---|---|
| $\times$ Speed Limit of streets | | | | |
| Normal | 1.0 | 0.1 | 0.9 | 1.1 |
| Over Speed | 1.25 | 0.1 | 1.2 | 1.3 |
| Under Speed | 0.75 | 0.1 | 0.7 | 0.8 |

## VI. Evaluation Results

In this section, we evaluate the anomaly detection performance of DeepFlow and compare it with three baseline methods: Dynamic Time Warping (DTW) [21], Global Alignment Kernels (GAK) [22], and iForest [23].

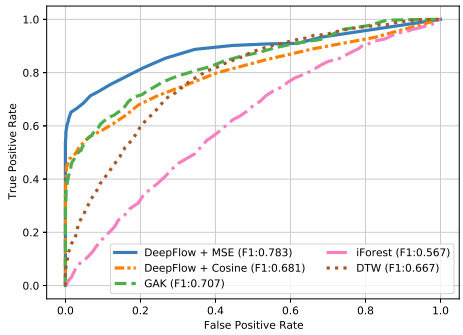


Fig. 10: ROC curve for DeepFlow, DTW, GAK, and iForest.

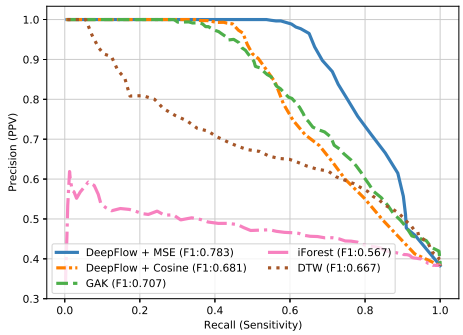


Fig. 11: Precision-Recall curve for DeepFlow, DTW, GAK, and iForest.

### A. Implemented Approaches

DTW and GAK are two techniques to compare two or more time series with unequal length, which find an optimal alignment between the points on the two sequences. We use a Python package called *TsLearn* [24] which provides machine learning tools for the analysis of time series to implement these two methods. Additionally, iForest is an unsupervised anomaly detection technique that works based on isolating the outlier cases. We implement it using *Sklearn* [25], an open source machine learning library which contains various tools for supervised and unsupervised learning.

Additionally, we use a Cosine similarity function instead of (2) and study its performance compared to MSE. To calculate the similarity of two latent vectors $A$ and $B$ using the Cosine similarity, we use the following expression:

$$\cos(A, B) = \frac{\sum_n^{i=1} A_i B_i}{\sqrt{\sum_n^{i=1} A_i^2}\sqrt{\sum_n^{i=1} B_i^2}} \quad (5)$$

### B. Performance Metrics

Undetected abnormal vehicles in the system are a threat to traffic safety; on the other hand, marking normal vehicles as abnormal is not desirable either. Therefore, finding a proper threshold has a significant impact on the True Positive and False Positive rate of the system. We employ a **Receiver Operating Characteristic (ROC)** curve to show DeepFlow's performance compared to other solutions; this displays the True Positive rate (TPR) versus False Negative rate (FNR) when we change the anomaly score threshold. TPR shows the proportion of abnormal cases which are detected, and FNR indicates the proportion of normal cases which are marked incorrectly.

TABLE II: Performance evaluation.

| Method | Recall | Precision | F1 | Time |
|---|---|---|---|---|
| **DeepFlow (MSE)** | 71.27% | **86.96%** | **78.34%** | $96ms$ |
| DeepFlow (Cosine) | 67.84% | 68.38% | 68.11% | |
| GAK | 70.88% | 70.54% | 70.71% | **33**$ms$ |
| DTW | 78.92% | 41.40% | 66.75% | $136ms$ |
| iForest | **89.90%** | 57.83% | 56.69% | $154ms$ |

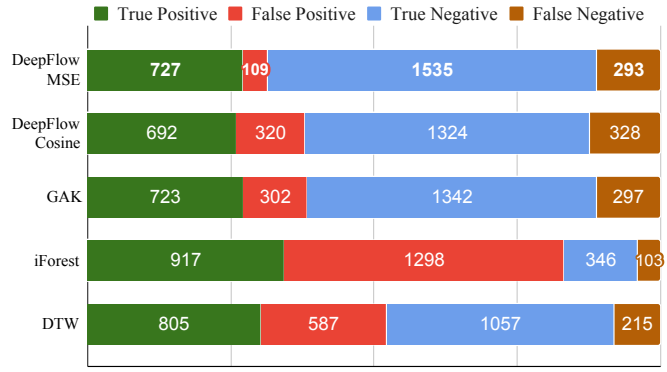


Fig. 12: TP, FP, TN, and FN for DeepFlow, DTW, GAK, and iForest.

We also use the **Precision-Recall** metric to evaluate the quality of our detector. Precision-Recall is an important measure when we use an imbalanced dataset. Precision is defined as the ratio of between the number of true positives and the number of detected anomalies, while recall determines the proportion of the actual abnormal cases that are identified correctly. A high value in both metrics is a sign of good performance for the detector. To consider Precision and Recall metrics in our evaluations, we use $F_1$ Score as it places equal weights on both precision and recall; this can be expressed as follows:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (6)$$

### C. Results and Discussion

**Performance Comparison.** We use the dataset that is described in *Section V*. The results, shown in Fig. 10, Fig. 11 and Table II, indicate that DeepFlow (with MSE) outperforms other solutions.

Based on Fig. 10, there is a threshold where DeepFlow detects 70% of the abnormal cases or a 6% FN rate. However, achieving the same TP rate using GAK or DTW will result in an FN rate of 18% and 26%, respectively. Likewise, the Precision-Recall curve in Fig. 11 shows that DeepFlow can achieve both greater precision and recall than other methods. In addition, Table II shows that DeepFlow can reach the highest $F_1$ score among all the evaluated approaches. Also, we show the average anomaly detection time for each traffic flow in the table.

**Microscopic Behavior.** As Fig. 12 shows, DeepFlow achieved the lowest number of false positives and detected more than 700 abnormal cases. Although iForest and DTW detected more abnormal cases than DeepFlow, a large number of normal cases are incorrectly marked by these approaches. The reason is that the random distribution of the vehicles speeds makes it

TABLE III: DeepFlow $F_1$ score for streets shown in Fig. 5.

| Path | Indiv. | Com. | Feature |
|---|---|---|---|
| 1 | 81.4% | 74.9% | Includes turnings |
| 2 | 82.5% | 60.0% | Curved and longest |
| 3 | 95.4% | 86.4% | Straight (similar to training) |
| 4 | 78.0% | 76.4% | North to south |

difficult for these classifiers to distinguish between the normal and abnormal cases, thus resulting in high false positive rates. However, DeepFlow learns the pattern of similarity between the normal behaviors, which help it detect anomalies more accurately.

**Anomaly Threshold.** Finding a proper threshold value can affect the accuracy of the system noticeably. This value can be selected for each street individually or a common value can be used. Table III shows that finding a threshold specifically for each street increases the $F_1$ score. However, it requires providing a dataset containing abnormal cases for each path and calculating the corresponding threshold value. Therefore, we suggest using a common threshold for the majority of the paths in cities and only assign individual thresholds for streets with greater importance, *e.g.* highways.

**Road Characteristics.** From Table III, we can observe that the performance of DeepFlow is dependent on the shape and length of the street that is being monitored. For example, street #2 is the longest simulated path which results in our model being unable to detect abnormal cases as accurately as in other paths. Also, street #3 has the most similar structure to the training dataset among other paths, so our model is able to detect anomalies with a higher $F_1$ score on this path.

**Parameter Tuning.** We analyze the result of changing the value of two different training parameters on our model behavior. First, we investigate the effect of the $\lambda$ value in Exp. 3 on the training performance. The results show that best performance is when the importance of the two loss functions in the training phase are equal, which can be achieved by setting the $\lambda$ value to 1. Second, we searched for the best latent vector size. Our experiment shows that we can, on average, get the best result from compressing the input value by 60%.

## VII. Conclusion

In this paper, we presented DeepFlow for detecting abnormal traffic flows. We showed that DeepFlow performs well even when it is trained with a dataset that is collected in a different environment. This feature indicates that DeepFlow can be employed for roads with different shapes, number of lanes and speed limits, with no retraining required. Even though that DeepFlow can work with any number of inputs (*e.g.* speed, acceleration and steering angle), we used trajectory data to detect anomalies as this type of data can be collected easily using existing surveillance camera infrastructure at no additional cost. Despite the excellent performance and easy implementation, our model suffers from two disadvantages. First, this model only works for a group of vehicles and cannot detect abnormal behavior of an individual car. The second problem is, we assume abnormal vehicles form the minority of target traffic flow, so our model only works as long as this assumption holds.

Future works include determining the effect of using more variables (such as speed, acceleration, and vehicle angle) and considering a wider range of anomalies.

## References


[1] Canadian Council of Motor Transportation Administrators, "Towards Zero: The safest roads in the world," 2016. [Online]. Available: https://roadsafetystrategy.ca/web/road-safety-strategy/files/public/docs/RSS-2025-Report-January-2016-with%20cover.pdf

[2] "Intelligent Transport Systems (ITS); vehicular communications; basic set of applications; definitions," European Telecommunications Standards Institute, Tech. Rep., 2009.

[3] M. Zhang, C. Chen, T. Wo *et al.*, "SafeDrive: Online driving anomaly detection from large-scale vehicle data," *IEEE Trans. on Ind. Inform.*, vol. 13, no. 4, 2017.

[4] H. Wu, W. Sun, and B. Zheng, "A fast trajectory outlier detection approach via driving behavior modeling," in *Proc. ACM Conf. on Inf. and Knowl. Manage.*, 2017.

[5] H. Oikawa, T. Nishida, R. Sakamoto *et al.*, "Fast semi-supervised anomaly detection of drivers behavior using online sequential extreme learning machine," in *IEEE ITSC*, 2020.

[6] S. Ucar, C. Patnayak, P. Oza *et al.*, "Management of anomalous driving behavior," in *IEEE VNC*, 2019.

[7] M. Matousek, M. EL-Zohairy, A. Al-Momani *et al.*, "Detecting anomalous driving behavior using neural networks," in *IEEE IV Symp*, 2019.

[8] M. Matousek, M. Yassin, A. Al-Momani *et al.*, "Robust detection of anomalous driving behavior," in *IEEE 87th VTC*, 2018.

[9] L. Zhang, L. Yan, Y. Fang *et al.*, "A machine learning-based defensive alerting system against reckless driving in vehicular networks," *IEEE Trans. on Veh. Technol.*, vol. 68, no. 12, 2019.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[11] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual prediction with LSTM," *Neural Computation*, vol. 2, 2000.

[12] Z. Tang, G. Wang, H. Xiao *et al.*, "Single-camera and inter-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features," in *Proc. CVPR Workshops*, 2018.

[13] M. Naphade, M.-C. Chang, A. Sharma *et al.*, "The 2018 NVIDIA AI city challenge," in *Proc. CVPR Workshops*, 2018.

[14] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," 2015.

[15] A. Paszke, S. Gross, F. Massa *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Inf. Process. Syst.*, 2019.

[16] W. Falcon *et al.*, "PyTorch Lightning," 2019. [Online]. Available: https://github.com/PyTorchLightning/pytorch-lightning

[17] L. Biewald, "Experiment tracking with Weights and Biases," 2020. [Online]. Available: https://www.wandb.com/

[18] P. A. Lopez, M. Behrisch, L. Bieker-Walz *et al.*, "Microscopic traffic simulation using SUMO," in *IEEE ITSC*, 2018.

[19] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org," 2017. [Online]. Available: https://www.openstreetmap.org

[20] A. Wegener, M. Piórkowski, M. Raya *et al.*, "TraCI: An interface for coupling road traffic and network simulators," in *Proc. Commun. and Netw. Simul. Symp.*, 2008.

[21] M. Mller, *Dynamic Time Warping*. Springer Berlin Heidelberg, 2007, pp. 69–84.

[22] M. Cuturi, "Fast global alignment kernels," in *Proc. Int. Conf. on Machine Learning*, 2011.

[23] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *IEEE Int. Conf. on Data mining*, 2008.

[24] R. Tavenard, J. Faouzi, G. Vandewiele *et al.*, "Tslearn, A machine learning toolkit for time series data," *Journal of Machine Learning Research*, vol. 21, no. 118, 2020.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, Michel *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, 2011.