Vehicle Trajectory-Based Intersection Classification Using Deep Transfer Learning

Journal:	*IEEE Access*
Manuscript ID	Access-2023-19531
Manuscript Type:	Regular Manuscript
Date Submitted by the Author:	05-Jul-2023
Complete List of Authors:	Alharbi, Fares; Shaqra University, Department of Computer Science, College of Computing and IT Kased, Abanoub; Minia University Faculty of Engineering Rabee, Rana; Minia University Faculty of Engineering Fahmy, Akram; Minia University Faculty of Engineering Mohamed, Hussein; Minia University Faculty of Engineering Yacoub, Marco; Minia University Faculty of Engineering Elhenawy, Mohammed; Queensland University of Technology Centre for Accident Research and Road Safety Queensland Ashqar, Huthaifa I.; Arab American University Masoud, Mahmoud; King Fahd University of Petroleum & Minerals, Department of Information Systems & Operations Management; King Fahd University of Petroleum & Minerals, Center for Smart Mobility and Logistics Hassan, Abdallah A.; Minia University Faculty of Engineering
Subject Category Please select at least two subject categories that best reflect the scope of your manuscript:	Intelligent transportation systems, Vehicular and wireless technologies
Keywords: <b>Please choose keywords carefully as they help us find the most suitable Editor to review</b>:	Connected vehicles, Convolutional neural networks, Transfer learning, Trajectory
Additional Manuscript Keywords:	

AUTHOR RESPONSES TO IEEE ACCESS SUBMISSION QUESTIONS	
Author chosen manuscript type:	Research Article
Author explanation /justification for choosing this manuscript type:	In this papaer, the problem in networks of taransportation is identified and the new model is developed to solve the problem using CNN algorithm. Then, the results are compared to state of art and proven the performance of the new model.Â
Author description of how this manuscript fits within the scope of IEEE Access:	This paper discuss problems relate to intellegent transportation system and using convolutional deep neural network which directly are subject of topics in IEEE access.
Author description detailing the unique contribution of the manuscript related to existing literature:	1-Â Â Â Â Â developing new model for intelligent transportation systems of to solve intersection classification using trajectory data. 2-Â Â Â Â Â Using five different CNN algorithms are used differently to build efficient classifier to detect road intersection types. 3-Â Â Â Â Â The algorithms used real world vehicle trajectory datasets and shows the better accuracy.

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Vehicle Trajectory-Based Intersection Classification Using Deep Transfer Learning

Abanoub Kased[1], Rana Rabee[1], Akram Fahmy[1], Hussein Mohamed[1], Marco Yacoub[1], Mohammed Elhenawy[2], Huthaifa I. Ashqar[3], Mahmoud Masoud [45], Abdallah A. Hassan[1] , and Fares Alharbi[6]

[1]Minia University, Faculty of Engineering, Computers and Systems Department, Minia, Egypt;
abanoub.attia354@eng.s-mu.edu.eg; rana.rabiee660@eng.s-mu.edu.eg; akram.nassif368@eng.s-mu.edu.eg; hussein.suleeman641@eng.s-mu.edu.eg; marco.abed357@eng.s-my.edu.eg; abdallah@mu.edu.eg
[2]Centre for Accident Research and Road Safety, Queensland (CARRS-Q); mohammed.elhenawy@qut.edu.au
[3]Civil Engineering Department, Arab American University, Jenin, Palestine; huthaifa.ashqar@aaup.edu
[4]Department of Information Systems & Operations Management, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.
[5]Center for Smart Mobility and Logistics, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.
[6]Department of Computer Science, College of Computing and IT, Shaqra University, Shaqra 15526, Saudi Arabia.
Corresponding author Fares Alharbi (e-mail: faalhrbi@su.edu.sa).

Abstract: Accurate and up-to-date road network maps are required for connected and automated vehicles to keep up with the world's continually expanding road networks. The traditional approaches for identifying road networks, i.e., satellite images and field surveys, require significant time and effort and are labor-intensive. With the wide usage of GPS-embedded devices, a massive amount of trajectory data is generated, which provides a new opportunity to update road maps with less effort. In this paper, a classifier is built to identify intersection types from vehicle trajectories, which can be the first step in the generation of road maps. This classifier is based on five different convolutional deep neural network architectures that are utilized to detect road intersection types. The experimental results using three different real-world vehicle trajectory datasets show that the different models achieve accuracy levels ranging between 84% and 91 % using only less than 10% of the pictures needed to achieve these accuracy levels without features used in this paper.

INDEX TERMS Convolutional Deep Neural Networks, GPS, map generation, road intersections, transfer learning, vehicle trajectories.

I. INTRODUCTION

Intersection classification is essential for map generation as the rules for creating intersection maps depend on the type of intersection. By classifying intersections according to their geometry, it is possible to generate road networks that include appropriate junction types in the right locations. There are several benefits of using intersection classification for map generation [1-4]. First, improved traffic flow management; different types of intersections can have different traffic characteristics and requirements, such as traffic lights, roundabouts, or stop signs. By accurately reflecting these patterns in a generated map, it becomes possible to plan and manage traffic flow more effectively. By using intersection classification to generate maps with appropriate junction types, it becomes possible to prioritize safety in urban planning and road design. Third, accurately modeling real-world road networks requires a nuanced understanding of the types of intersections that are common in different regions and urban environments. By using intersection classification to generate maps, it becomes possible to create more realistic representations of real-world cities and towns. Fourth, by generating road networks with appropriate intersection types, it becomes possible to reduce congestion, improve traffic flow, and optimize transportation networks.

Intersections are a pivotal part of road networks, as most traffic fatalities take place at or close to intersections. The Federal Highway Administration reported 10,180 traffic fatalities involving an intersection in 2019 [5], so managing and updating them is the key to keeping the traffic in urban areas under control. Road intersections are places where two or more roads cross each other, and they are often the locations of crashes and collisions between vehicles, pedestrians, and cyclists.

Existing methods of intersection map inference/generation can be categorized into two categories based on the data sources used:

(1) methods relying on image-based sources like remote sensing images (RS images), and aerial images[6-8]

(2) methods relying on trajectories [9-14]

RS and aerial images are quickly advancing and provide much more information compared to trajectories. However, they are susceptible to errors caused by natural factors like rain and clouds or by the city's topology, like trees or tall buildings, so it faces different challenges than trajectory data. With the rise of GPS-embedded devices, large quantities of trajectory data are being provided and updated daily, making using trajectory data a good choice for map generation. But with large amounts of data comes a price, which is the degraded quality of the data and the amount of noise in the data. Besides, most GPS devices have a low sampling rate which makes it difficult as the car might have passed multiple intersections before an update has taken place. Therefore, the classification of intersections needs a large set of observed trajectory data points to identify the intersection type. Consequently, the specific map inference model is chosen to estimate the intersection characteristics such as the center point, the stop lines, and the number of lanes.

Traditional map inference involves collecting data through surveys or remote sensing techniques, such as satellite imagery or light detection and imaging (LiDAR). While these methods can provide accurate information, they are often expensive and time-consuming. For example, a detailed topographical survey of an area could take weeks or even months to complete, and the cost of equipment and personnel can be significant. Crowdsourced data, on the other hand, offers a potential solution to this problem. By leveraging the vast amounts of data generated by mobile devices and other sources, researchers can potentially create and update maps more quickly and at a lower cost. One type of crowdsourced data that can be used for map inference is trajectory data.

Trajectories refer to the paths that individuals or objects take as they move through space. This data can be collected through GPS-enabled devices such as smartphones, fitness trackers, or vehicles equipped with tracking systems, including automated and connected vehicles (CAVs). By analyzing trajectories, researchers can gain insights into patterns of movement and activity within a particular area. This information can then be used to infer the locations of roads, buildings, and other features on a map. For example, if a large number of trajectories converge on a particular point, it may indicate the presence of a popular landmark or attraction. However, there are challenges to using crowdsourced data for map inference. For example, the quality and accuracy of the data can vary significantly, and there may be issues with privacy and data ownership. Additionally, crowdsourced data may not be available or limited to minor streets where traffic is light, leading to biases in the resulting maps. To overcome these challenges, more research is needed on how to effectively collect, process, and analyze crowdsourced data for map inference. This includes developing new algorithms and tools for data analysis, as well as addressing ethical and legal considerations related to data ownership and privacy. Ultimately, the use of crowdsourced data has the potential to revolutionize the way maps are generated and updated, making them more accurate, timely, and accessible to a wider range of users.

II. RELATED WORK.

Intersection classification is used to classify the types of intersections or road junctions that appear in a road network. The classification system is typically based on the number of ingress and egress approaches at each intersection, and the angles between them. The purpose of intersection classification is to identify the intersection type and consequently select the suitable map inference model to generate a realistic intersection map that accurately reflects the patterns and behaviors of real-world traffic flows, as shown in **Error! Reference source not found.**. To the best of the knowledge of the authors, this paper is the first to address this classification problem, previous research work focused only on the intersection map inference and assumed that the type of the intersection is known (i.e., four legs intersection, roundabout, etc.). This section briefly discusses the different categories of methods found in the literature that have been employed to generate maps at intersections.

A. Methods relying on image-based sources

As this is considered a map inference problem, Convolutional Neural Networks (CNNs) are often the tool used since they have proved their accuracy and reliability in this task, especially Deep Convolutional Neural Networks (DCNNs) [2]. The big number of layers allows room for more accuracy. Thus, several research efforts proposed the use of different DCNN architectures for the task of map generation with the increasing popularity of RS images and aerial images. He et al. [3] proposed an architecture called ResNet, with several variations discussed in their paper such as ResNet-18, ResNet-34, and ResNet-50, with the numbers corresponding to the number of layers. It was tested on the CIFAR-10 [4] dataset which is a dataset of colored images with ten labeled classes. Ronneberger et al. [1] proposed an architecture called U-Net which was used in image segmentation in the Biomedical field. Zhang et al. [15] adapted both ResNet and U-Net architectures and proposed ResUNet to be optimal for the task of road extraction using aerial images of roads in Massachusetts [6]. ResUNet has 15 Convolutional layers compared to 23 layers of U-Net and performs better in the task of road extraction. LinkNet was proposed by Chaurasia et al. [16], it uses ResNet-18 as its pre-trained encoder, and it was used in semantic segmentation. Zhou et al. [7] used LinkNet architecture but with a different pre-trained encoder ResNet-34 and named it D-LinkNet, it was used on aerial images for road extraction and it achieved better performance than LinkNet for this task. The previous methods relied on only one source of data whether it was RS images or satellite images, however, both sources have their flaws as discussed earlier. Li et al. [8] proposed to combine trajectory data with RS images to get the best results and to overcome the challenges of using either of these sources alone. The U-Net architecture was utilized, and it showed far better results than RS images alone.

B. Methods relying on trajectories

The methods used in literature that fall under this category can be classified into three categories: trace merging, clustering, and rasterization. Trace merging is the process of adding new traces and merging them with the old traces like in Cao et al [9], and Niehöfer et al [10]. The method proposed by Cao is considered the standard trace merging method. Afterward, Niehöfer improved this method by removing or adjusting the existing road network after merging a new trace using its weights. Clustering is a method of grouping data points into similar groups, there are several methods of clustering trajectories like Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [11], K-means [12], Kernel Density Estimation (KDE) [13]. Edelkamp et al. [12] proposed the first K-means algorithm for generating road networks. Clustering and Trace merging methods do not perform well on noisy data [14]. Rasterization is the process of converting an image into raster data. In Guo et al. [17], rasterization with an Otsu algorithm was used. The rasterization method performs the best in comparison to the other two methods in the face of noise as it treats each trace individually [14].

C. Methods using extracted features

The two previous categories of methods only rely on the raw data and then try to make it more efficient or less noisy. Still, some researchers used trajectory data to extract input features, including speed, acceleration, heading, or density. Nguyen et al. proposed extracting Histogram-based features using GPS Trajectories [18]. The mapped space containing the trajectory was divided into equal-sized cell grids. Therefore, the GPS trajectory can be encoded into a matrix; its elements at location (i, j) are the number of GPS Points in the grid cell (i, j). Zhang et al. proposed converting the trajectories into images by counting the trajectory points inside each cell [19]. However, they suggested resampling the trajectory at even intervals to avoid short intervals that lead to a long stay (i.e., many points inside the same cell). Elhenawy et al. proposed extracting features using Recurrence Quantification Analysis (RQA), which provided extensive temporal behavior of the obtained signal (e.g. speed and acceleration) [20]. Consequently, the extracted features using RQA were used to create images (called RQA images) that could then be used as inputs in a classification algorithm instead of using many numerical features. Daley et al. segmented 10 Hz trajectory data into a 1-second window that was transformed into a 3-channel image. The three channels represented the Gramian Angular Summation/Difference Field (GASF/GADF) and the Markov and Transition Field (MTF) [21]. Guo et al. [17] proposed using a grid of equal-sized cells and a Gaussian kernel to calculate the contribution of each GPS point in the 2D plan to the intensity of the cell. Li et al. used the grid idea to calculate the density graph/image and the speed graph/image, which are considered static and dynamic features, respectively [8]. Ruan et al. divided the map region of interest into tiles and each tile into a grid with $N \times M$ cells [22]. Then for each cell, two types of features are extracted, namely, the spatial and the transition features. The spatial features are extracted from the trajectory data inside the cell. Spatial features include the density of the GPS points, average speed, number of line segments inside the cell, and the normalized histogram of the regular direction of movement between two consecutive points. On the other hand, the transition features of a grid cell are useful for low-frequency trajectories. The transition features at cell i are two binary matrices; the first represents all the direct origins of trajectories landing in cell i, and the other matrix represents all the direct destinations from cell i. Prabowo et al. [23] divided the area of interest into a 1mx1m grid and used this grid to build a 3-channel image. The first channel is directly constructed such that a cell is assigned a one if it has at least one GPS point and zeroes otherwise. The other two channels need the computation of the speed vectors in the x and y directions by extrapolating the speed and heading of each GPS point in the trajectory. Then, each cell in the second and the third channels will have the aggregated and normalized speed projection in the x and y directions, respectively. Eftelioglu et al. proposed creating consecutive pairwise lines using the GPS points and using these lines to create the image [24].

In this study, the use of transfer learning and pre-trained DCNNs is proposed to classify/recognize the location type at which a subset of trajectory data is observed. Two approaches are investigated to convert the trajectories (i.e., time series) into an image that can be used as an input for the DCNNs. The first approach creates images by scattering the trajectory location information in the 2-D plan. The second approach converts the trajectory data into multi-channel images by dividing the 2-D plan into a grid of equal-area cells and extracting different features at each cell, including density and speed. For the sake of testing the robustness of the idea and getting good classification results, five different DCNNs (NasNet, ResNet, MobileNet, EfficientNet, and DenseNet) are trained and tested on three datasets from the following cities: Porto in Portugal, San Francisco in The United States, and Beijing in China.

In the first set of experiments (i.e., the baseline), simulated data is created and used in the first training round. Consequently, real data is used to fine-tune the models in the second training round. Then the trained models are tested using real data only. In the second set of experiments, features are extracted from the dataset using two grid sizes

20x20 and 30x30, and only real data is used for training and testing the models. The extracted features are density, speed, acceleration, and heading. Moreover, different models are trained using different subsets of features, and their performances are compared. Finally, a bootstrap-based heuristic is used to construct a good ensemble using a subset of the trained models.

III. METHODOLOGY

This study aims to propose a novel framework that can classify the type of intersection which can be used for map generation. For this reason, two sets of experiments are designed. For the first set, namely, the baseline, trajectory data is used from three different datasets of real-world setups, and five pre-trained DCNNs are applied using transfer learning. The trajectory data of the three datasets are converted into images by scattering the individual movements of vehicles over the 2-D space. The resulting images are static and show the trajectory data point distribution. To increase the number of images used for training the different algorithms, simulated datasets of 10,000 images are created that mimic the real ones and are only used for training purposes. This set of experiments aims to explore the different datasets and the behavior of different algorithms and to set a baseline for the next set of experiments.

The second set of experiments includes multiple steps. First, it is important to ensure the framework can be generalized across various scenarios by testing it on multiple datasets. In other terms, instead of testing the model on just unseen instances of the same data used for training, multiple datasets are used for testing. Hence, the performance of the model is better understood, and potential weaknesses or biases are identified. This helps to ensure that the model will be useful in real-world scenarios and not just in the training data. This will also be beneficial in choosing the best datasets for the following steps, which include extracting more features and creating multi-channel images to train the algorithms. Nonetheless, to ensure the usefulness of the model, we cross-tested the five models with the three different real-world datasets. In this phase, the best dataset among the three is chosen for the next steps.

Second, to take advantage of the trajectory data and increase the accuracy, density, speed, heading, and acceleration features are extracted. Extracting features from trajectories involved quantifying the movement patterns of vehicles in terms of various attributes such as speed, heading, acceleration, and density. These features can be derived from the raw trajectory data. This can be achieved by dividing the space into a grid of equal cells and using the trajectory data points inside each cell to estimate the feature we are interested in, as explained in section 3.4.

Multi-channel images are visual representations of the trajectory features, where each channel corresponds to a particular feature and its values are mapped to pixel intensities or color values. The images are typically created by stacking multiple channels on top of each other to form a composite image. For example, a multi-channel image for a set of trajectories of a vehicle might have the following channels:

- Speed channel, which represents the speed of each car at each point in time, with faster speeds shown in brighter colors
- Heading channel, which represents the direction that each car is traveling, with different colors for different directions.
- Acceleration channel, which represents the rate of change of speed of each car, with positive acceleration shown in one color and negative acceleration shown in a different color.

By overlaying these channels, a multi-channel image can provide a more comprehensive view of the movement patterns and interactions between cars, which can be useful in understanding and analyzing traffic flow dynamics. No specific maximum number of channels can be used in generating multi-channel images from trajectory data, as the number of channels depends on the number of features extracted and visualized. In practice, the number of channels used in these images is usually determined by the number of relevant features and the amount of information that needs to be conveyed. However, it is important to keep in mind that as the number of channels increases, the complexity of the image also increases, which can make it more difficult to interpret and analyze. Therefore, it is important to strike a balance between the number of channels used and the clarity and interpretability of the resulting image. Hence, the maximum number of channels used in this work is determined to be three channels. As such, a fourteen combination of the four features, (i.e., $4_{C_1} + 4_{C_2} + 4_{C_3}$), can create images for training. With five different DNN algorithms, there are about 70 models that can classify intersections.

Third, a bootstrapping-based heuristic is employed to choose the best-performing models. Bootstrapping refers to a family of statistical methods that involve resampling from a dataset to estimate the uncertainty and variability of a statistical model or an estimator. In bootstrapping, multiple resamples are created from the original dataset by randomly sampling observations with replacement. Each resample is used to fit the statistical model or estimate the parameter of interest. This process is repeated many times, generating multiple bootstrap models or estimates. The variability and uncertainty of the original model or estimator can be estimated by analyzing the distribution of the bootstrap models or estimates. This can provide insight into the stability and robustness of the original model or estimator and can help identify potential sources of bias or overfitting. Bootstrapping can be used in various statistical applications, including regression analysis, hypothesis testing, and machine learning. It is important to stress that this paper focuses on the first three steps of the map inference problem illustrated in **Error! Reference source not found.**.

A. Deep Transfer Learning Algorithms

The following steps are followed for the proposed classifiers using pre-trained models and transfer learning:

1. Loading the model without including the top layer
2. Adding a classification layer with four units and a softmax activation function
3. Freezing the base model layer to prevent the weights in the base model from being updated during the training of the classification layer.
4. Training the network on the simulated dataset examples
5. Fine-tuning by unfreezing some of the higher layers of the base model
6. Training the network again but on the real dataset examples, because the first few layers learn simple features that generalize to all types of images while the higher layers learn the features that are more specific to the dataset used for training.

Five different well-known models are used. The used models are presented in **Error! Reference source not found.**.

B. Real-World Datasets

Trajectory data of three different datasets is used to create four classes that correspond to the following intersection types including cross-intersection, roundabout, T-intersection, and non-intersection (i.e., highway). Creating real examples includes making a bound box for the longitude and latitude of each intersection, then selecting a random sample of the vehicle trajectories within that box and plotting longitude and latitude points, as shown in Fig. 2. The three datasets used to create the examples are presented in Table 2. The T-drive dataset [29, 30] has a total distance of the trajectories that reaches nine million kilometers, an average sampling interval of about 177 seconds, and a distance of about 623 meters. The cab-spotting dataset [31] contains mobility traces of taxi cabs in San Francisco, USA. However, ECML-PKDD 15 [32] dataset contains the trajectories of taxis of about 1.7 million instances with each data sample corresponding to one complete trip.

C. Simulated Dataset

The more the simulated examples are close to real road intersection examples, the more variability will be. To create a simulated example, several factors are considered including sampling rate, independent noise for X and Y coordinates achieved by assuming that the data is normally distributed with a different mean and variance for each direction, the density of trips on each road is different than other roads in the same road intersections, and an OD matrix is used with a random number of trips for source-destination. An example of an intersection resulting from the simulation is shown in **Error! Reference source not found.**. The flow chart for creating the simulated data is presented in **Error! Reference source not found.**.

D. Feature Extraction

In addition to the density, three other features are extracted including acceleration, speed, and heading. The PTRAIL library [33] is used, which uses trajectory data to extract features using the NumPy library and parallel computing which makes it relatively faster in computations. Extracting features is performed through the following steps.

First, the type of data frame is converted to a PTRAIL Data Frame using the ptrail.core.TrajectoryDF module.

Second, using the ptrail.features.kinematic_features.KinematicFeatures module, three features are extracted by applying three corresponding methods; create_acceleration_column for acceleration, create_speed_column for speed, and create_bearing_column for heading.

Third, trajectories are calculated from a bounding box which typically involves analyzing the movement of an object within a given space. A bounding box is a rectangular shape that is used to enclose a region of interest (i.e., intersection). In the context of trajectory calculation, a bounding box is used to define the position of the vehicle within the defined region.

Finally, The ptrail. preprocessing. filters module is used to apply filters to the dataset of trajectories before any analysis or visualization. Filters can help to remove noise, reduce data redundancy, and improve the quality of the data. These filters can help to improve the accuracy and reliability of the trajectory data and to make it easier to analyze and visualize the data.

To extract the density feature using a Gaussian kernel, a grid of cells that covers the spatial extent of the trajectory data is defined. For each trajectory point, its contribution to the density of each cell is calculated using a Gaussian kernel. The Gaussian kernel has a bell-shaped curve that assigns higher weights to nearby points and lower weights to farther points. The formula for a 2D Gaussian kernel is:

$$K(X,Y) = \frac{1}{\sqrt{2 * \pi * \sigma}} * e^{\frac{-((x_1 - x_0)^2 + (y_1 - y_0)^2)}{2 * \sigma^2}} \quad (1)$$

where σ is the standard deviation of the Gaussian kernel, x, and y are the distances from the trajectory point to the center of the cell.

Then, the sum of contributions of all trajectory points to each cell is calculated to obtain the cell density. Cell densities are normalized by dividing by the total number of trajectory points.

To use the extracted features as inputs to the model, they need to be represented as images. The features are placed in the RGB channels of the photo, which means there can only be a maximum of three combinations in a single image. As mentioned earlier, four features are extracted, with fourteen possible unique feature combinations including density (D), speed (S), acceleration (A), heading (H), density-speed (DS), density-acceleration (DA), density-heading (DH), speed-acceleration (SA), speed-heading (SH), acceleration-heading (AH), density-speed-acceleration (DSA), density-speed-heading (DSH), density-acceleration-heading (DAH), and speed-acceleration-heading (SAH). The following steps are used to create the grid, illustrated in Fig. 5:

1. Determining the minimum and maximum of the longitude and latitude,
2. Choosing the appropriate grid size (e.g., 20x20 and 30x30),
3. Calculating the longitude and latitude step using (2):

$$step = \frac{|\max - min|}{grid\ length * 2} \quad (2)$$

4. Creating a 3D matrix with the first two dimensions equal to the grid dimensions and the third dimension equal to two which corresponds to the longitude and latitude of the grid cell center, the matrix is initialized with zeros,
5. Calculating the center points of the grid cell using (3) and (4):

$$matrix[x, y, 0] = \max longitude - (1 + 2 * x) * step\ longitude \quad (3)$$

$$matrix[x, y, 1] = \min latitude + (1 + 2 * y) * step\ latitude \quad (4)$$

6. Initializing four grids with the same dimensions of the grid for each feature with zeros,
7. Determining the weights of the Gaussian Kernel using (1), where the distance between the two points is less than or equal to three times the sigma value as follows:

$$Distance = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \quad (5)$$

,

8. Extracting the required features, where the distance between every two points does not exceed the distance limit,
9. Completing the features grid with the result of weights of the Gaussian kernel matrix by calculating the dot product with the feature matrix,
10. Transposing and flipping the matrices of features, so they look identical to the original intersection photo,
11. Applying the min-max scaling on the matrix so it can fit in the range of the RGB channel
12. Placing the feature grids in the RGB channel of the images

E. Ensemble Models

Ensemble models, also known as ensemble learning or meta-algorithms, are machine learning techniques that combine multiple models to improve the accuracy and robustness of predictions. The idea behind ensemble models is to leverage the diversity of different models to reduce bias and variance and to produce more accurate and reliable predictions. There are several types of ensemble models, but the most common one is based on bootstrapping the training dataset. It involves training multiple models on different subsets of the training data, where each subset is randomly sampled with replacement from the original training data. The final prediction is obtained by averaging or voting the predictions of all models. Ensemble models can be applied to a wide range of machine learning tasks, including classification (i.e., as in this work), regression, and clustering. They are particularly effective in situations where individual models may have limited accuracy or reliability due to overfitting, noisy data, or bias. In this step, an ensemble is built using the models trained with fourteen possible feature combinations and five different models. Therefore, 70 different models are trained independently. To select a good subset of models that yield an accurate ensemble, a stepwise regression approach (i.e., a greedy approach) is used. Stepwise regression needs the models to be ordered using some metric. Using the accuracies of the stand-alone models to order them for the stepwise regression is not a good choice because it does not reflect how well a model performs in an ensemble. Therefore, the marginal accuracies of the models are initialized using the algorithm described in the next subsection.

Estimating the marginal accuracy of a model in an ensemble

This algorithm estimates the models' marginal accuracies based on randomly forming ensembles. The whole experiment is repeated (NE) times. Each time a randomly chosen subset of models is used. The number of models in each subset can be any random number from two to a maximum length (NF). This estimation method consists of the following four steps:

The first step is ensemble generation. NE ensembles are generated randomly during this step, but model duplication is not allowed within the same subset.

The second step is evaluation, where each ensemble is evaluated using the testing dataset. After the evaluation, each ensemble has its classification percentage.

The third step is updating the marginal accuracy of individual models. The marginal accuracy of each model is the average of the ensemble's classification accuracy, which contains this model.

The final step is ordering the models in descending order according to their marginal accuracy.

Fig. 6 illustrates the marginal accuracy estimation process. To calculate the marginal accuracy of every model, initialize a vector with the accuracies of the models and a counter equal to 1 then run a loop for a thousand iterations. In each iteration, a number of models between 2 and 10 is chosen randomly, and the chosen models are ensembled together, the resulting accuracy is added to every model accuracy in the initialized vector. Within the loop, the randomly chosen combination with the highest accuracy is saved. The models are ranked depending on the highest marginal accuracies and performed stepwise on the ranked models, starting with the 1st ranked model then ensemble with it the 2nd rated model, and continuing to ensemble the 70 models together.

IV. Results

A. Baseline Experiment

The five classifiers selected for this study are tested on three different datasets including T-Drive, Cab-spotting, and ECML-PKDD 15. In each dataset, the same 10,000 images of simulated data are used for the first phase of the training and then the model is fine-tuned using a real-world training dataset. Consequently, 500 images of each real-world dataset are used for testing. Based on the results shown in Fig. 7, DenseNet has the best F1 score compared to the other classifiers, it scored 96% in T-drive, 96% in cab-spotting, and %93 in ECML-PKDD 15. This is likely due to its high number of parameters combined with the low number of layers, compared to the other classifiers. The next best-performing classifiers are ResNet and EfficientNet on T-drive with a score of 94%, MobileNet, and EfficientNet on cab-spotting with a score of 93%, and ResNet on ECML-PKDD 15 with a score of 92%. ResNet is the second-best performing classifier after DenseNet, this also matches up with the assumption that a smaller number of layers combined with a bigger number of parameters is what performs best as ResNet parameters are close to DenseNet parameters, but ResNet has an even smaller number of layers. The least-performing classifier is NasNet. This is likely due to its large number of layers compared to the other two classifiers that have the same number of parameters (i.e., MobileNet and EfficientNet) and the fewer number of parameters compared to the best-performing classifiers (i.e., DenseNet and ResNet).

B. Cross-testing models

The main goal of this step is to ensure that the framework can be generalized by testing it on a separate set of data (i.e., test set) to ensure that it can make accurate predictions on new, unseen data. The trained models on each dataset are saved individually and then tested on the datasets they were not initially trained on. To visualize the results, the Kruskal test is used to compare the distribution of performance metrics of the model when tested using two different datasets and assuming a cut-off p-value of 0.05. The Kruskal test is used because the results are not uniformly scattered. Fig. 8 shows the results on T-drive. From Fig. 8, it can be seen that DenseNet, which has the highest F1 score on the T-drive dataset, generalizes well on the other two datasets. ResNet, which has the second-best score, does not generalize well and is rejected on both datasets. However, EfficientNet, which has the same score as ResNet is accepted on cab-spotting and rejected on ECML-PKDD 15 which is likely due to the large noise in the dataset as discussed earlier. NasNet and MobileNet are rejected on ECML-PKDD 15 and failed to reject on cab-spotting as they have the same F1 score.

Fig. 9 shows the results of cab spotting. From Fig. 9, it can be seen that DenseNet still generalizes well on the other two datasets. It can also be seen that even though MobileNet and EfficientNet are the second-best performing models, MobileNet is rejected from both datasets while EfficientNet failed to reject from both. It can also be seen that even though ResNet has a lower score than MobileNet, it failed to reject both. And while NasNet performs the worst here with an F1 score of 81%, it still fails to reject from the T-drive dataset.

Fig. 10 shows the results on ECML-PKDD 15. From Fig. 10, it can be seen that ResNet, which has the second-best score, is also rejected from both datasets, while EfficientNet and NasNet are not rejected from both datasets. Even though NasNet has the lowest score out of all the five classifiers. MobileNet is also rejected from T-drive similar to DenseNet.

C. Feature extraction results

The images resulting from feature extraction are tested on two different sizes of grids, namely, 20x20 and 30x30, using the ECML-PKDD 15 dataset to see if the size of the grid affects the classification accuracy. The same five models are used. However, this time, 1,000 images are created from the real-world dataset, and models are trained on 900 images and then tested on the remaining 100 images. The results of using a 20x20 grid size are shown in Fig. 11, it can be seen that NasNet is the best-performing model on half of the combinations. A close second is ResNet, which has the best scores on 6 out of 14 combinations, MobileNet scores the highest in only two combinations, while EfficientNet has the lowest scores out of all models in all of the combinations. It is also noticed that speed, acceleration, and speed-acceleration combinations have the lowest accuracy results. Results of using a 30x30 grid size are shown in Fig. 12, the accuracy of models ResNet and EfficientNet improved in all combinations, while MobileNet and NasNet did not improve that much. It is also noticed that DenseNet is no longer performing the best overall as was the case without the features discussed in the previous section. It can still be seen that the speed, acceleration, and speed-acceleration combinations have the lowest scores, while density-speed-heading has the highest scores on both grid results.

D. Ensemble model results

Fig. 13 shows the marginal accuracy results in descending order. ResNet-Heading (RN-H) has the highest accuracy while EfficientNet-Acceleration (EN-A) has the lowest accuracy. Fig. 14 shows the frequency of each of the chosen models, RN-SA and DN-D are the models chosen most (about 55 times), while some models are not chosen a lot such as EfficientNet-Heading (EN-H) (less than 30 times). It is found that MobileNet-SAH, NasNet-AH, MobileNet-SH, Dense-Net-DA, EfficientNet-DA, ResNet-DAH, ResNet-DSH, EfficientNet-DSA outperformed the other models and can be generalized with about 91% classification accuracy.

V. Conclusion

The problem of intersection classification using trajectory data is tackled in this paper. Five CNNs are used: (NasNet, ResNet, MobileNet, EfficientNet, and DenseNet), using three different datasets: T-drive, cab-spotting, and ECML PKDD 15.

First, a dataset of images is created using longitude and latitude data from the trajectory data. Two sets of images are constructed: the first set of images is formed from simulated data, while the other set of images is formed from real data. The same 10,000 simulated photos are used to train all models which are then tested on 500 photos of real data from the dataset. The achieved F1 scores range from 80% to 94%. The best scoring model in this task is the DenseNet with F1 scores of (.96 - .96 - .93) on the three datasets. The lowest accuracy comes from the ECML PKDD 15 dataset which is likely due to the large amount of noise in it, unlike the simulated data.

To test how well the trained models generalize, the same models trained on the three datasets are tested on the other two datasets that are not used in training. The Kruskal test with a p-value of .05 is used to visualize the outputs. The models that generalize well on the other datasets are the models trained on cab-spotting and ECML PKDD 15, whereas the models trained on T-drive do not generalize as well.

Later, the PTRAIL library is used to extract features from the raw trajectory data. The following features are chosen: density, speed, acceleration, and heading. A total of 14 combinations of the selected features are used, 1000 images of each combination are created, 900 images for training, and the remaining 100 images for testing. Two sizes of grids are experimented with; 20x20 and 30x30. With the 20x20 grid, the achieved accuracy ranges between 39 – 85%, while the 30x30 grid significantly improved the results with an accuracy range of 55 – 90%.

Finally, the 70 models that have been trained on the various combinations are ensembled based on their accuracies. The achieved accuracy levels range between 84 – 91% using only 900 photos for training compared to the use of 10,000 photos from the scatter plots. Results with the scatter plot range from 80 – 93% accuracy, and Results with the features range from 50 – 90 % accuracy without the ensemble and 84 % - 91% with the ensemble.

ACKNOWLEDGMENT

The authors would like to thank the Deanship of Scientific Research at Shaqra University for supporting this work.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," ArXiv, vol. abs/1505.04597, 2015.
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770-778.
[4] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.
[5] F. H. W. Administration. "Intersection Safety." https://safety.fhwa.dot.gov/intersection/about/
[6] V. Mnih and G. E. Hinton, "Learning to detect roads in high-resolution aerial images," in European conference on computer vision, 2010: Springer, pp. 210-223.
[7] L. Zhou, C. Zhang, and M. Wu, "D-LinkNet: LinkNet with Pretrained Encoder and Dilated Convolution for High-Resolution Satellite Imagery Road Extraction," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 192-1924, 2018.
[8] Y. Li, L. Xiang, C. Zhang, and H. Wu, "Fusing Taxi Trajectories and RS Images to Build Road Map via DCNN," IEEE Access, vol. 7, pp. 161487-161498, 2019.
[9] L. Cao and J. Krumm, "From GPS traces to a routable road map," Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2009.
[10] B. Niehöfer, R. Burda, C. Wietfeld, F. Bauer, and O. Lueert, "GPS Community Map Generation for Enhanced Routing Methods Based on Trace-Collection by Mobile Phones," 2009 First International Conference on Advances in Satellite and Space Communications, pp. 156-161, 2009.
[11] J. Qiu and R. Wang, "Road Map Inference: A Segmentation and Grouping Framework," ISPRS Int. J. Geo Inf., vol. 5, p. 130, 2016.
[12] S. Edelkamp and S. Schrödl, "Route Planning and Map Inference with Global Positioning Traces," in Computer Science in Perspective, 2003.
[13] J. J. Davies, A. R. Beresford, and A. Hopper, "Scalable, Distributed, Real-Time Map Generation," IEEE Pervasive Computing, vol. 5, pp. 47-54, 2006.
[14] J. Biagioni and J. Eriksson, "Inferring road maps from global positioning system traces Survey and comparative evaluation," Transportation research record, vol. 2291, no. 1, pp. 61-71, 2012.
[15] Z. Zhang, Q. Liu, and Y. Wang, "Road Extraction by Deep Residual U-Net," IEEE Geoscience and Remote Sensing Letters, vol. 15, pp. 749-753, 2018.
[16] A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," 2017 IEEE Visual Communications and Image Processing (VCIP), pp. 1-4, 2017.

[17] Y. Guo, B.-j. Li, Z. Lu, and J. Zhou, "A novel method for road network mining from floating car data," Geo-spatial Information Science, vol. 25, pp. 197 - 211, 2022.
[18] C. Nguyen, T. Dinh, V.-H. Nguyen, N. Tran, and A. Le, "Histogram-based Feature Extraction for GPS Trajectory Clustering," EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, vol. 7, no. 22, 2020.
[19] L. Zhang, G. Zhang, Z. Liang, and E. F. Ozioko, "Multi-features taxi destination prediction with frequency domain processing," PloS one, vol. 13, no. 3, p. e0194629, 2018.
[20] M. Elhenawy, H. I. Ashqar, M. Masoud, M. H. Almannaa, A. Rakotonirainy, and H. A. Rakha, "Deep transfer learning for vulnerable road users detection using smartphone sensors data," Remote Sensing, vol. 12, no. 21, p. 3508, 2020.
[21] M. Daley, M. Elhenawy, M. Masoud, S. Glaser, and A. Rakotonirainy, "Detecting road user mode of transportation using deep learning to enhance VRU safety in the C-ITS environment," in Proceedings of the 2021 Australasian Road Safety Conference, 2021: Australasian College of Road Safety (ACRS), pp. 450-452.
[22] S. Ruan et al., "Learning to generate maps from trajectories," in Proceedings of the AAAI conference on artificial intelligence, 2020, vol. 34, no. 01, pp. 890-897.
[23] A. Prabowo, P. Koniusz, W. Shao, and F. D. Salim, "Coltrane: Convolutional trajectory network for deep map inference," in Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, 2019, pp. 21-30.
[24] E. Eftelioglu, R. Garg, V. Kango, C. Gohil, and A. R. Chowdhury, "RING-Net: road inference from GPS trajectories using a deep segmentation network," in Proceedings of the 10th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, 2022, pp. 17-26.
[25] A. Howard et al., "Searching for mobilenetv3," in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1314-1324.
[26] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in International conference on machine learning, 2019: PMLR, pp. 6105-6114.
[27] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697-8710.
[28] G. Huang, Z. Liu, K. Weinberger, and L. van der Maaten, "Densely connected convolutional networks. CVPR 2017. arXiv 2016," arXiv preprint arXiv:1608.06993.
[29] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 2011, pp. 316-324.
[30] J. Yuan et al., "T-drive: driving directions based on taxi trajectories," in Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems, 2010, pp. 99-108.
[31] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD data set epfl/mobility (v. 2009-02-24)," ed, 2009.

[32] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi–passenger demand using streaming data," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 3, pp. 1393-1402, 2013.
[33] Y. Haranwala. "PTRAIL library." https://github.com/YakshHaranwala/PTRAIL

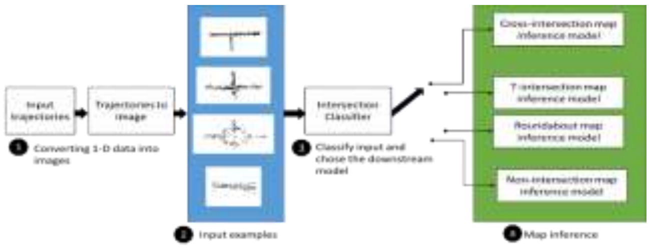

Input trajectories
Trajectories to image
1 Converting 1-D data into images
2 Input examples
Intersection Classifier
3 Classify input and chose the downstream model
Cross-intersection map inference model
T-intersection map inference model
Roundabout map inference model
Non-intersection map inference model
4 Map inference

Figure 1 The map inference steps

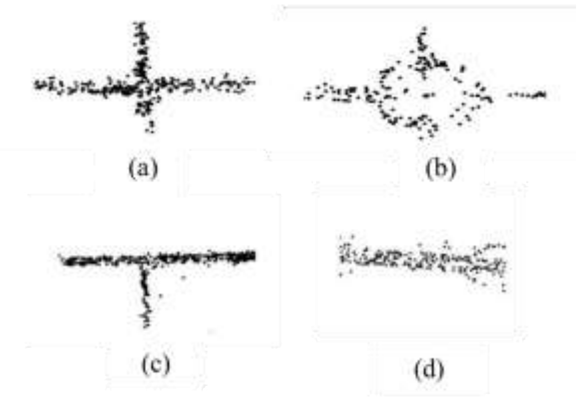


Figure 2 Image example of real-world data (a) Cross-intersection; (b) Roundabout; (c) T-intersection; (d) Non-intersection

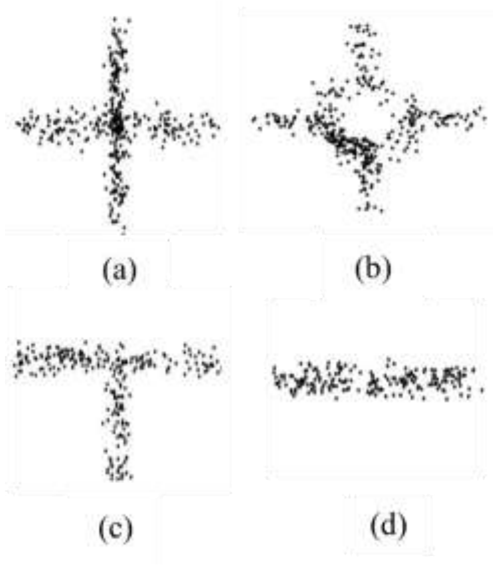


Figure 3 An image of a simulated example of (a) Cross-intersection; (b) Roundabout; (c) T-intersection; (d) Non-intersection.

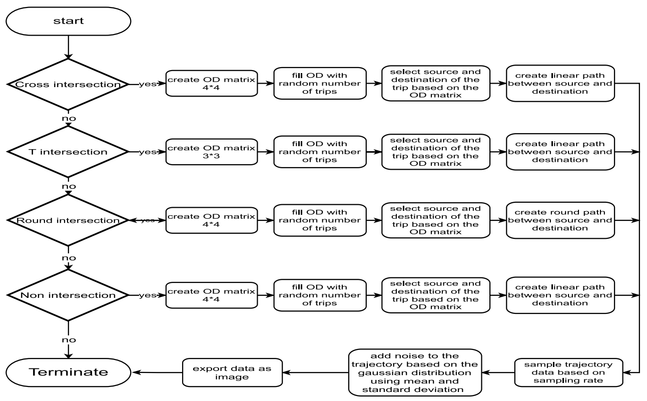

start
Cross intersection
yes
create OD matrix 4*4
fill OD with random number of trips
select source and destination of the trip based on the OD matrix
create linear path between source and destination
no
T intersection
yes
create OD matrix 3*3
fill OD with random number of trips
select source and destination of the trip based on the OD matrix
create linear path between source and destination
no
Round intersection
yes
create OD matrix 4*4
fill OD with random number of trips
select source and destination of the trip based on the OD matrix
create round path between source and destination
no
Non intersection
yes
create OD matrix 4*4
fill OD with random number of trips
select source and destination of the trip based on the OD matrix
create linear path between source and destination
no
Terminate
export data as image
add noise to the trajectory based on the gaussian distribution using mean and standard deviation
sample trajectory data based on sampling rate

Figure 4 Steps of simulating data for different types of intersections.

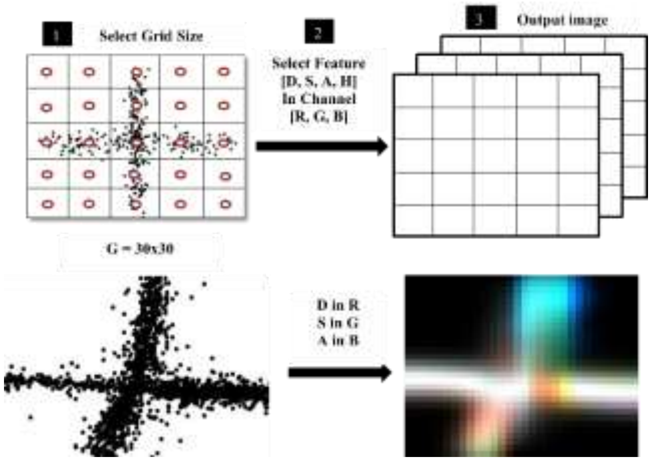


Figure 5 Illustrating the method of generating features.

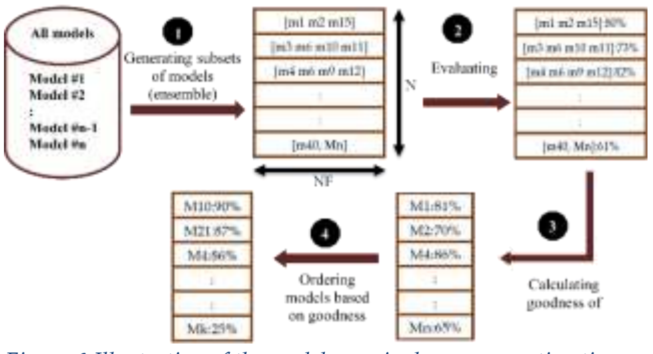
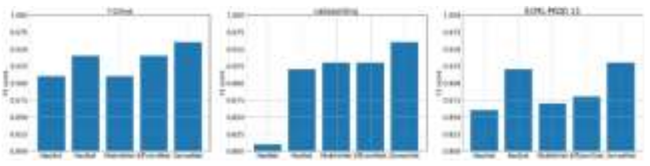


Figure 6 Illustration of the model marginal accuracy estimation



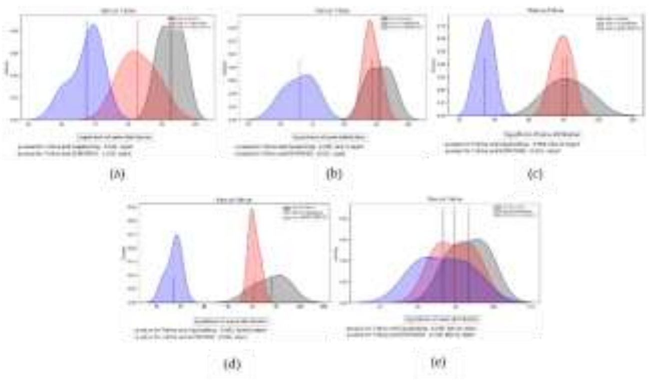


Figure 7 F1 scores for the five classifiers applied to the three datasets.

Figure 8 Kruskal Test on T-drive. (a) ResNet; (b) NasNet; (c) MobileNet; (d) EfficientNet; (e) DenseNet.

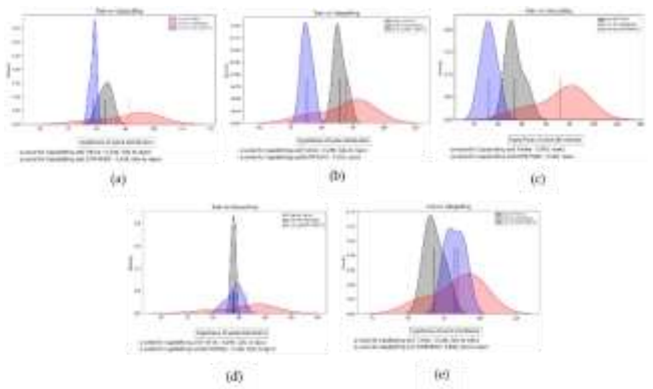


Figure 9 Kruskal Test on cab-spotting. (a) ResNet; (b) NasNet; (c) MobileNet; (d) EfficientNet; (e) DenseNet.

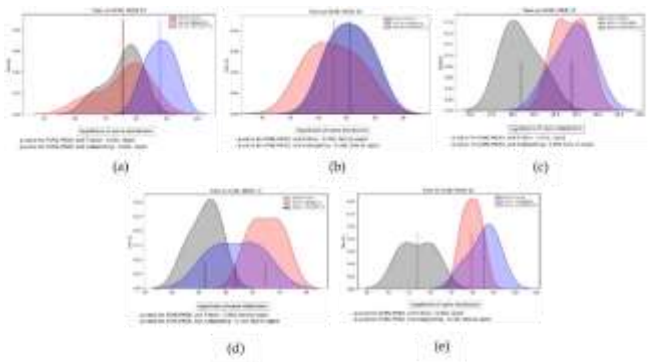


Figure 10 Kruskal Test on ECML-PKDD 15. (a) ResNet; (b) NasNet; (c) MobileNet; (d) EfficientNet; (e) DenseNet.

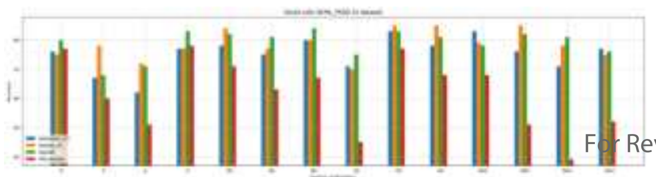


Figure 11 Classification accuracy of the five models with the 20x20 grid from the ECML PKDD 15 dataset.

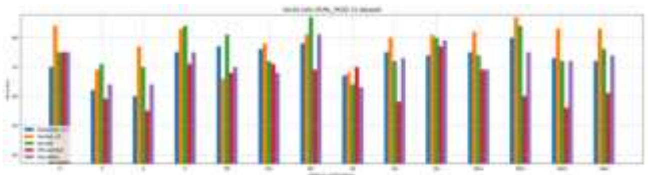


Figure 12 Classification accuracy of the five models with the 30x30 grid from the ECML PKDD 15 dataset.

Figure 13 Classification accuracy results of the five models with the ensemble on the ECML PKDD 15 dataset.

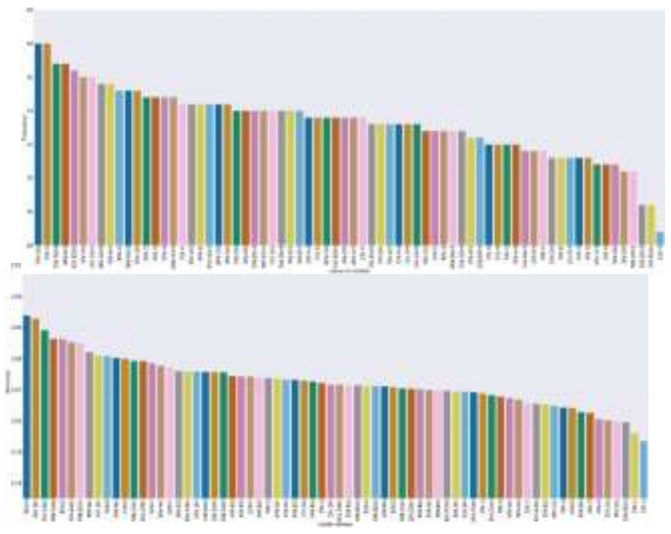


Figure 14 Frequency of the chosen models' ensemble applied on the ECML PKDD 15 dataset

Table I Features of the used models.

Model	Label	Number of Parameters	Number of Layers	Reference
MobileNet-V3-Large	MN	4.2 M	269	[25]
EfficientNet-B0	EN	4 M	238	[26]
NASNetMobile	NN	4.2 M	769	[27]
ResNet-50	RN	23.5 M	175	[3]
DenseNet-121	DN	29 M	427	[28]

Table II Datasets features.

Dataset	Location	Period	Number of Users	Number of Events	Reference
T-Drive	Beijing, China	1 week	10,357	15 million	[29, 30]
Cab-spotting	San Francisco, USA	1 month	536	11 million	[31]
ECML-PKDD 15	Porto, Portugal	1 year	442	83 million	[32]

Abanoub Kased was born on November 8th, 1999, in Minia, Egypt. He received his Bachelor of Engineering (BE) in computer and system engineering from Minia university, Egypt, in 2023.

Rana Rabee was born July 26th, 2000, in Minia, Egypt. She received her Bachelor of Engineering (BE), in the department of computer and systems, from university of Minia, Egypt, in 2023.

Akram Fahmy was born on September 12th, 2000, in Minia, Egypt. He received his Bachelor of Engineering (BE) in computer and system engineering from Minia university, Egypt, in 2023.

Hussein Mohamed was born on July 2nd, 2000, in Minia, Egypt. He received his Bachelor of Engineering (BE) in computer and system engineering from Minia university, Egypt, in 2023.

Marco Yacoub was born on April 8th, 2000, in Minia, Egypt. He received his Bachelor of Engineering (BE) in computer and system engineering from Minia university, Egypt, in 2023.

MOHAMMED ELHENAWY received the Ph.D. degree in computer engineering from Virginia Tech (VT). He worked, for three years, as a Postdoctoral Research at the Virginia Tech Transportation Institute (VTTI), Blacksburg, VA, USA. He is currently a Research Fellow with the Center for Accident Research and Road Safety - Queensland (CARRS-Q), Queensland University of Technology. He has authored or coauthored over 40 ITS related articles. His research interests include machine learning, statistical learning, game theory, and their application in intelligent transportation systems (ITS) and cooperative intelligent transportation systems (C-ITS).

HUTHAIFA I. ASHQAR received the B.Sc. degree (Hons.) in civil engineering from An-Najah National University, Nablus, Palestine, in 2013, the M.Sc. degree in road infrastructure from the University of Minho, Braga, Portugal, in 2015, and the Ph.D. degree in civil engineering from Virginia Tech, VA, USA, in 2018. He also received two graduate certificates in data science and economic development from Virginia Tech, in 2018 and 2021, respectively. He is currently a Senior Transportation Engineer at Precision Systems Inc. in DC and an Adjunct Professor at the University of Maryland Baltimore County. He has authored/coauthored one book chapter and over 30 refereed publications in the areas of innovative mobility, ITS, advanced transportation and energy technologies, AI, and data analytics. His experience includes being a Technical Advisor for programs with over $50 million value in advanced transportation and energy technologies in the U.S. DOE's ARPA-E.

MAHMOUD MASOUD received the M.Phil. degree in applied mathematics operations research and decision support systems from Cairo University (CU) and the Ph.D. degree in operations research and mathematical sciences from the School of Mathematical Sciences, Queensland University of Technology (QUT). He is currently a Research Associate with king Fahd university of petroleum and minerals, Saudia Arabia. He has extensive experience, as a Research Associate, in many industrial projects as a part of effective teamwork at the Centre for Tropical Crops and Bio-commodities (CTCB), School of the Mathematical Sciences, and CARRS-Q, QUT, Brisbane. This team constructed industrial linkages with big industrial organizations, such as EY and MLA (beef supply chain projects), owners of the Australian Miles (Biomass and Bioenergy assessment sugarcane projects), and Brisbane Royal Hospital (health system project). He has a wide range of experience in academic research and industrial projects with more than 22 refereed journal and conference papers and industrial reports.

Abdallah Hassan is an assistant professor of computer engineering at Minia University (Egypt) since 2017. The research interests for Dr. Hassan include intelligent systems and machine learning. He received his PhD in Computer Engineering from Virginia Tech (USA) in 2016. Before that, Dr. Hassan received his master’s degree in computer engineering from the University of Texas at El Paso (USA), and his bachelor’s degree in computer engineering from Cairo University.

Fares Alharbi received the Ph.D. degree in Computer Engineering from Queensland University of Technology, Brisbane QLD, Australia, in 2019. He is currently an Assistant Professor with the Department of Computer Science and

Head of Department of IT in College of Computing & IT, Shaqra University, Shaqra, Saudi Arabia. His research interests include big data computing, cyber security, cloud computing, computer networks, blockchain and privacy artificial intelligence and machine learning, large-scale optimization, intelligent systems, and smart cities.