Feature Selection and Mobile Implementation for Pre-Trip Driver Identification from In-Vehicle Data

Kai Kang[1]

Abstract— **This paper continues the work of a previous paper written by Kar et. al [1] which explores the minimal data set necessary to effectively differentiate drivers using data from existing in-vehicle sensors. The existing system interfaces to the vehicle bus and executes supervised or unsupervised driver differentiation techniques on in-vehicle sensor information. Our work includes extracting new features from pre-trip in-vehicle data and developing Android application of driver classification. The results show that new features improves the accuracy by approximately 10% comparing to the old features.**

I. INTRODUCTION

For years, we have been conducting new sensors into modern automobiles. This provide us the potential to program our cars, to require data from internal of the vehicle and to build applications on top of that. There are hundreds of sensors and actuators that exchange data on internal buses. We have easy access to a small part of all data by taking advantage of OBD-II standard. This small part of data is sufficient of building usefully and powerful applications. We only briefly discuss some of the vehicle applications that might want to implement driver-specific functions. Applications listed below are only some of the representative apps. With increasing access to the in-vehicle data, we are expecting a grate number of useful applications in the future.

Detecting Unauthorized User. Modern vehicles are being connected to the internet with broadband technologies. If the vehicle has a built-in system that could differentiate drivers, it is possible that the vehicle could identify unauthorized drivers. When detected an unauthorized driver, the vehicle could immediately notify the owner. Furthermore, if the system has sufficient confidence of the detection, the vehicle may able to notify other authorities.

Targeted Advertisement. Advertising revenues have had a significant impact on the Internet and mobile economy. Websites are already applying user identification or clustering system in order to provide targeted advertisement. Since vehicles are becoming more and more programmable and connected, we are expecting that targeted advertisement is integrated into in-vehicle applications. Driver differentiation is no doubt the foundation of these kinds of application.

In-Car Personalization. Some modern vehicles are already capable of personalizing setting according to drivers preferences. Some vehicles provide two keys with different electronic IDs and encourage drivers to consistently use the same key. Others use physical buttons for drivers to choose their personalized settings. Personal setting may include radio station, temperature, dashboard display brightness and navigation view. These applications still depend on physical or additional hardware. Software based driver differentiation system will provide better in-car personalization.

All the applications listed above requires or benefits from the driver differentiation system. A low-cost system which could distinguish different drivers will truly provides developers and engineers the opportunity of designing and implementing the applications with customized user experiences.

Even though with the smart car in development, it would still be taking some time for smart cars to be available to the majority. We seek a effective solution for utilize the vehicle data. Former papers have already proved that a minimal set of in-vehicle sensors data can be used to distinguish drivers without additional hardware cost. The former proposed solution differs from other existing solutions. It usually takes a token for product solutions to distinguish drivers. The token could be a form of a smart key fob that the driver needs to carry. This lacks the flexibility of distinguish multiple drivers sharing the same key fob or even without any additional hardware. Other works have also discussed the possibility of using mobile devices in vehicles to determine the identification of the driver. Yet they tend to yield the conclusion that the device owner is the driver.

Our work focus on improving the former system proposed by [1] and implementing the classification system on Android. Further more, we applied the Neural Network pattern recognition toolbox in MATLAB to analyze new data set. In summary, the notable contributions of this work are the following:

- defining and extracting new features from pre-trip in-vehicle data and adding new indicators into the driver classification algorithm. Experiments show an approximately 10% increase on classification accuracy after new indicators are applied.
- applying Neural Network Pattern Recognition algorithm to differentiating drivers. Classification result shows higher accuracy than former conducted Support Vector Machine algorithm.
- developing Android application which simulates the in-vehicle data collection, extracts features, applies classification algorithm and makes driver detection decisions.

*This work was supported by Winlab, Rutgers University, North Brunswick, NJ, USA

*Special thanks to Gorkem Kar[2] as the adviser on this project.

[1]Kai Kang is a graduate student of Electrical and Computer Engineering, Rutgers, the State University of New Jersey, New Brunswick, NJ, USA `kk769@scarletmail.rutgers.edu`

[2]Gorkem Kar is with Winlab, Rutgers University, North Brunswick, NJ, USA `gkar87@winlab.rutgers.edu`

II. RELATED WORK

There are several papers that have established solid work in the area of driver identification. Enev et. al [7] managed to identify 15 drivers with 100% accuracy using drivers' longitudinal behaviors. It takes at least 15 minutes to train the proposed system for each driver. The training includes braking patterns, acceleration, vehicle speed and throttle position etc. Choi et. al [3] discussed modeling driver behavior using steering angle, acceleration status , brake status and vehicle's speed that are collected from vehicle's CAN bus. The identification idea in their paper is quite similar to our approach. Yet the accuracy of driver identification is less than 35% with their system mainly because they are only using longitudinal data.

There are also some discussion about the privacy of the collected data. Many research proposes to use speed and location coordinates for privacy preservation. Yet Krumm et.al [15] showed that we can use location traces to identify different drivers. Also Gao et. al [8] proved that the speed is enough to track a driver and it is not a privacy preserving parameter.

As a matter of fact, many researchers have been exploring ways to maintain driver privacy and anonymity by modifying the data used for identifying. Zan et. al [23] used a zone-aware path cloaking scheme to provide anonymity at a high degree. In another work, Hoh et. al [11] have proposed using virtual trip lines to maintain driver privacy. Another approach is using a Trusted Third Party to perform cryptographic operations. Details can be found in research [4] , [5] , [18] and [19].

The work by Pentland et.al [17] investigated the modeling and predicting human behavior. For determine when a car will be passing another, turning or following the previous cars in next couple of seconds, their approach achieved 95% accuracy at predicting vehicle drivers actions based on their initial preparatory movements.

Rather than driver monitoring both in commercial companies and research facilities, there are also many work study the systems for traffic monitoring. Researches leverage GPS units on cars(OnStar [24] system) to track the vehicles movements and analysis can be done at the server.

Other papers have discussed using accelerometer and GPS sensors of the mobile phones. The Nericell [16] project concentrates on the road topology and detection of potholes, bumps and braking. In another paper [20], researchers showed how drivers behavior under critical circumstances(sudden breaks, extreme steering angle rotations) varies compared to regular times. Differ from the former papers, our approach uses vehicle sensors which we believe is more reliable for driver identification system.

III. BACKGROUND AND PREVIOUS WORK

In this section, we will first discuss the background of driver identification. Then the author will describe the design of the former system proposed by [1] which is also the foundation of our newly improved driver classification system.

A. Background

Nowadays vehicles are equipped with lots of Electronic Control Units (ECUs). These ECUs controls and monitors different vehicle modules, such as the engine, HVAC, doors or power seats. Most of these ECUs are connected to the Controller Area Network (CAN) bus [6], which is a standardized vehicle data bus that allows those ECUs to communicate with each other. Many sensor data fields are broadcast on this bus since some vehicle functions require sensor data from other ECUs. Also for diagnostic purposes, transmitted data allow us to diagnose vehicle malfunctions using repair technicians' devices.

Since 1996, all vehicles sold in the United States mandatorily support the On-Board Diagnostics II (OBD II) port1. It is a standard interface for vehicles to provide self-diagnostics and data reporting capabilities. Yet the extent to which this data is accessible through the OBD II varies. Even though the port is usually directly connected to the CAN bus, we only have access to a small subset of fields which relevant for government permission testing and basic troubleshooting. The majority of in-vehicle data uses proprietary encoding and we have no directly access to those data through the OBD II standard. Some of the inaccessible fields of data include individual cylinder misfires, door status, seat belt status, ignition voltage and turn signals status. The exact fields that are included in such proprietary data streams also depends on the exact vehicle model and the sensors that are built into the vehicle. For instance, only if the vehicle has a built-in GPS receiver will the GPS data be available on the internal bus. The bus architecture and start-up procedure really vary across different vehicle models.

However, there exist a great number of opportunities to exploit the proprietary in-vehicle data for other applications. For example, OpenXC [2] provides access to select proprietary in-vehicle data fields through a special OBD-II adapter. This adapter is a OBD-to-Bluetooth device and it relays CAN messages to Bluetooth equipped mobile devices (e.g., smart phones). Similar to other standard OBD-II adapters, it also decodes additional proprietary fields through a custom firmware. Car makers could also remotely access and process vehicle data because of the increasing availability of broadband connectivity in vehicles.

B. Previous Work

The idea was to establish a system that could distinguish drivers within a small period of time. The proposed system are able to identify a time-sequence of in-vehicle data which contains several events that closely tied to driver habits so that they will varies among different drivers. At the meantime, these behaviors are little affected by the traffic participants and other driving conditions. In order to accomplish that, pre-trip in-vehicle data was analyzed and several pre-trip was extracted.

1) Pre-trip Events Selection: Only do the in-vehicle data be available in a number of different vehicle models can the system be general. Fortunately, there are 14 fields of in-vehicle data that are available in many vehicle models and these values always changes in the early stage of a trip, which providing the system the potential to identify drivers within a small amount of time. Since the vehicle generates data even before it starts moving, the system could collect data of several fields that are directly correspond to driver actions. Six distinct events happen before the vehicle begins to move: door opening (DO), door closing (DC), starting the ignition (ISU), seat belt fastening (SF), shifting gear (SU) and releasing the brake pedal (RB). After the vehicle begins to move, there are some other driving data streams indicating driving habits such as steering wheel angle (SWA), engine revolutions per minute (RPM), vehicle speed (V), and acceleration (AP) values.

The six pre-trip events happened before the vehicle starts moving are particularly useful for driver detection because they are dependent on driving habit greatly and are unaffected by the road and traffic conditions in most of the cases. Plus they all happen within a short period of time starting from the door open which is the very first event of a trip except the unlock of the car. The number and type of events will not differ across drivers yet the timing of these events will differ across drivers according to different driving habit. As for indicators like steering wheel angle, the degree the driver turns the steering wheel when the vehicle first start moving is really depends on the road condition, whether the car is leaving a parking lot or a garage. Fortunately, the relative timing of the six pre-trip events should not depend significantly on external factors. Therefore, their specific order and their timing interval should be helpful in building driver identification model.

To gain a better understanding of the timing and order of pre-trip events, the former research conducted a preliminary experiment with eight drivers. They used a 2008 Cadillac CTS vehicle (test vehicle), an LG Nexus 5 phone and a custom OBD scanner (dongle) to extract data from the vehicle. The custom dongle provides access to a richer set of vehicle data streams, compared to standard dongles, providing grater potential to build more sophisticated system. The smart phone is located inside the vehicle and the dongle is plugged into the OBD-II port during all the experiments. Though it should be expected that the smart phone carried by the driver in real environment, the smart phone stays in the car so far so that drivers could do the trails conveniently. Note that they conducted an IRB approved study, and used coded data for drivers instead of their actual identities. All participants are 18 years old or older and have a valid drivers license in United States.

The drivers were instructed to enter the test vehicle, complete a loop in the parking lot, and return to the same spot. Each driver repeated the experiment 10 times. For this preliminary experiment, drivers were asked to consistently follow their regular habits, to reduce some variance in the data and reveal possible distinct patterns across drivers. The results showed that the relative timing is quite distinct across drivers, even when drivers started the vehicle in the same controlled test situation and it can be observed that all pre-trip events occur within 20 seconds after opening the door.

2) Feature Vector: As discussed before, we use six events and two additional values to generate the pre-trip feature which we believe is crucial in distinguishing one driver from others. To be specific, the pre-trip feature vector f_{pt}, is defined as follows.

$$f_{pt} = [\Delta t_{DC},\ \Delta t_{ISU},\ \Delta t_{SF},\ \Delta t_{SU},\ \Delta t_{RB}] \qquad (1)$$

Here, Δt_{DC} represents the time difference between the occurrence of the (driver) Door Close (DC) event and the occurrence of a reference starting event, which marks the beginning of the vehicle data stream for the new trip.

The Door Open (DO) event was used as the reference event because it is the first observable event on the vehicles used in experiments. All other events in the feature vector happen in some order yet definitely after the Door Open event. According to that, Δt_{ISU} , Δt_{SF} , Δt_{SU} and Δt_{RB} represent the time difference to the ignition switch usage (ISU), Seat belt Fastened (SF), Shift Usage (SU) and Release of the Brake pedal (RB).

IV. REFINED FEATURES AND CHOICE OF ALGORITHMS

The key to successful identification of the drivers is the feature vector which currently contains five indicators. The difference of feature vectors represents the difference between drivers. The reason we use multiple indicators is that we are hoping at least one of the indicator difference is big enough for separating two drivers. For example, if the driving habit of two drivers are too similar that the first four indicators cannot show the difference, then we are rely on the last indicator. The more indicators could show the difference, the more chance we have on differentiating drivers. Therefore, we try to improve the accuracy of classification by adding new indicators into the system.

A. New Indicators

Besides of the six pre-trip events which all have specific timing, we also have access to the brake pedal usage. It is represented by Double value and indicates whether the driver is pushing the brake pedal and how hard the pushing is. Nearly all modern vehicles requires the driver to push the brake pedal while the driver is shifting the gear from P (park mode) to D (driving mode). We acquire this brake pedal usage value, noting as pushing brake (PB), and use it as another indicator. As will shown in later sections, adding this indicator will yield reasonable increase of accuracy for driver identification compared with former five-indicator implementation. Another two more indicators that we added are speed peak timing (SPT) and speed peak value (SPV) where the speed peak is the first peak that vehicle speed reaches when it starts moving. Relatively, the speed peak timing is the time stamp of speed peak event and the speed peak value is the actual speed of the vehicle at that time. These two indicators reflex the driving habit at some point. A driver having small SPT and high SPV are likely drive faster than others having large SPT and small SPV. These two new indicators helped us achieve better accuracy among some drivers.

B. Refined Feature Vector

We now redefine the feature vector with newly added indicators. The new definition of pre-trip feature vector f_{pt} is shown as follows.

$$f_{pt} = [\Delta t_{DC},\, \Delta t_{ISU},\, \Delta t_{SF},\, \Delta t_{SU},\, \Delta t_{RB},\, \Delta t_{SP},\, pb,\, sp] \quad (2)$$

Similar to the first five indicators, the Δt_{SP} represents the time difference between the occurrence of the release brake (RB) event and the occurrence of the speed peak (SP) event.

In general, the order of these events are yet not limited as described as follows. The driver Door Open and driver Door Close events are usually due to the driver entering the car. The ISU event represents starting the engine. The Seat belt Fastened event occurs of course when the driver fastened their seat belt. Some drivers tend to fasten their seat belt before they start the engine. Since we are calculating the Δt_{SF} by using the time stamp of the SF event minus the time stamp of the ISU event as defined in (2), we are expecting the Δt_{SF} to be positive for some drivers and negative for others.

$$\Delta t_{SF} = t_{SF} - t_{ISU} \quad (3)$$

The Shift usage refers to changing the setting on an automatic transmission from park mode to drive mode. At the meantime, the driver must be pushing the brake pedal as he or she changing the transmission. We acquire the brake pedal value at this time as the pb indicator. The Release Brake event marks the instance when the brake pedal was released to start driving. The Speed Peak refers to the vehicle reaches the first speed peak of this trip. The peak usually happens within several seconds after the Release Brake event. Indicator sp represent the speed value of the peak.

C. Classification Algorithms

1) Support Vector Machine: Support Vector Machine is a supervised classification algorithm. The training data is formulated by putting together the feature vectors and class labels. Each class is representing a driver in our implementation. The rows of the training data are the feature vectors. They contains the pre-trip indicators as mentioned before. All the driver data is anonymized so we use numbers to store and refer to the data. Therefore, the drivers ID number is the corresponding class label. The trained model is then used to classify each incoming feature vector into one of multiple classes.

2) Neural Network Pattern Recognition: We also use the MATLAB built in Neural Network Pattern Recognition toolbox to carry out the classification of drivers. The input data of this algorithm is a huge matrix which contains all the ten driver feature vectors. Similar to SVM, the target is the corresponding class label. MATLAB splits the input data into three parts: Training, Validation and Testing. Training data are presented to the network during training, and the network is adjusted according to its error. Validation data are used to measure network generalization, and to halt training when generalization stops improving. Testing data have no effect on training and so provide an independent measure of network performance during and after training.

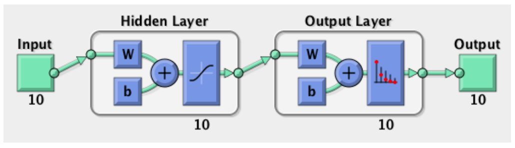


Fig. 1. Neural Network Architecture

MATLAB allow us to modify the portion of each part of the data and the default setting is using 70% of the data as Training and 15% of the data as both the Validation and Testing.

Another configuration of the neural network algorithm is the number of neurons in the pattern recognition network's hidden layer. Figure 1 shows the architecture of neural network with configuration of 10 neurons. The number will affect the performance of the algorithm as will shown in later sections.

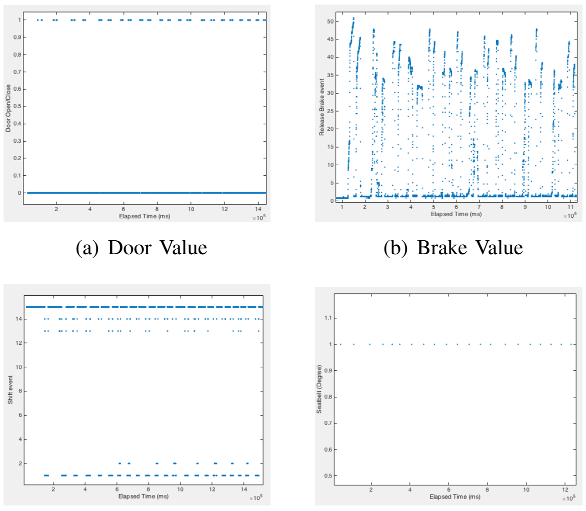
V. IMPLEMENTATION

In this section, we will discuss the detail of our system including how to extract all the features from the raw data and implementing the system in MATLAB as well as in Android.

A. Feature Extraction

Most of the data we acquired from OBD-II dongle are time continuously. We have pre-trip in-vehicle data of ten drivers. They separated into five groups. Each group has two drivers. The two drivers in each group was doing trails one by one. Yet the OBD data was continuously. So the very first thing of data process is separating each trails.

Figure 2 shows the plots of four PID values corresponding to the time stamp. As we can see in the figures, only seat belt value shows specific distinct events. Each point in Figure 2(d) represents a Seat belt Fastening action. However, we can not use the seat belt value to separate each trails. When we look real close into the seat belt values, we found that when some drivers did the fasten seat belt action, some how they tried twice before the seat belt was actually fasten. Therefore, there are some corrupted data points in the seat belt value. We decided to use the shift value to separate all the trails. The shift value data we acquired from the OBD is clear and well patterned. Every sudden drop of the shift value indicates the changing of transmission from park mode to driving mode. As for each trail, there is and only is one shift event happened. Therefore, the first feature we extracted is the timing for all shift event.



(a) Door Value (b) Brake Value

(c) Shift Value (d) Seatbelt Value

Fig. 2. Pre-trip in-vehicle data

Based on the shift event, we are now able to extract other features relatively. Here, we only discuss extracting door events in detail. The extraction of other events are similar. As shown in Figure 2(a), high level of door value indicates that the door is open and the low level value represents that the door is closed. Therefore, an increase of door value shows the changing state of the door from closed to open and vice versa. However, for each trail, the driver has to open and close the door before the trail. He or she also has to open and close the door one more time after completing the trail. So we are expecting two open door events and two close door events for each trail. Since we have already extracted the shift events, we now can identify the door open and door close events based on that. Obviously, the door events that happened before the shift event are due to the driver entering the vehicle. Yet the door events that happened after the shift event are caused by the driver leaving the vehicle which we are not using in our system.

Algorithm 1 Door Events Extraction

```
 1: procedure DOOR EVENTS RECOGNITION
 2: function Extract(data d)
 3:     for each data point point in d do
 4:
 5:         if time of point less than time of shift event AND
            value of point increases then
 6:             Update door open timing
 7:
 8:         if time of point less than time of shift event AND
            value of point decreases then
 9:             Update door close timing
10:
11:         if time of point exceeds time of shift event then
12:             move to next shift event
13:             store the door events
```

Algorithm 1 describes the procedure to extract door open and door close event. The algorithm iterates through all the door values data points and extracts the door events which are closest to shift events. By doing so, the number of door events will equal to the number of shift events, which is exact what we are hoping for. After we extract all the features/indicators, the feature vector can be generated.

B. Android and MATLAB Implementation

1) MATLAB: The pre-trip in-vehicle data for experiments are stored in large text files. MATLAB has really useful features for importing data from text file and processing data as matrices. We used MATLAB to plot data and so that we could have a better understanding of it. To decide which indicator should be added into the driver identification system and how to extract all the features needed requires clear and deep understanding of the data stream.

Another reason we choose MATLAB is that it provide us well functioned built-in toolbox for classification algorithms. So far we have applied multi-class SVM and Neural Network Pattern recognition classification algorithms as discussed before. Differ from [1], our system analyze six pre-trip fields. In the feature extraction phase, our system not only extracts timing for considered events but extract two more PID values at specific time as well.

After the MATLAB program generated the feature vector for all ten drivers, we write the feature vectors for each drivers into text file for further experiments. Classification program reads specific feature vector data files as the way it needs. Experiments and evaluations will be discussed in later sections in detail.

2) Android Application: Our Android Application is part of a larger project. Our Android Application so far contains two activities. The main activity simulates the activity which captures pre-trip in-vehicle data from OBD-II and passes the data stream to the second activity. The data is passed between activities as a costumed parcelable object. The main activity simulates the data capturing by reading the data from a text file and writing the data into the costumed parcelable object.

The second activity receives the parcelable object along with the data inside. The parcelable object has well structured API so that our program could easily read and process the data inside. The feature extraction algorithm we implemented in Android application is similar to the algorithm we discussed in former section. Yet in Android Application the program only need to process data for one trip. Therefore instead of separating each trips, the Android program only needs to look for the specific mark for each events. After extracting the features, the program calculates the timing difference of each event and generates the feature vector for the trip. Since the classification model is already trained by the MATLAB, making the classification decision only takes a few calculations, which provide us the efficient identification. The Android application has a simple UI showing the identification result which is the driver's ID.

VI. Experiment and Evaluation

We conducted several experiments, aiming to address the following questions:

- How does multi-class SVM classification algorithm compare to Neural Network Pattern Recognition algorithm?
- How is accuracy affected with the increasing number of drivers?
- How do different features/indicators affect the driver differentiation performance?

To evaluate our system, we carried out the following experiments.

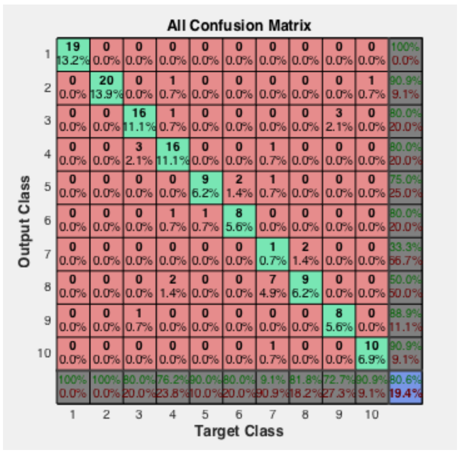
A. Experimental Setup

1) Data Collection: The way we collected new data set used in our experiments is similar to which in the previous work discussed in former section. Instead of the 2008 Cadillac CTS test vehicle, we used a GMC vehicle and all test drivers are actually GMC employees. There are total ten drivers and each driver completed ten to twenty trips during the test. Specifically, we collected data for 144 trips total. Driver 1, 2, 3 and 4 completed about twenty trips whereas other drivers completed about ten trips.

2) Metrics: In our work, we evaluate the performance of driver detection algorithms in terms of accuracy which is the ratio of correctly identified number of trails to the total number of trails. Note that when applying Neural Network Pattern Recognition algorithm in MATLAB, the program will calculate the accuracy of classification automatically and it is using the same metrics as us.

B. SVM and Neural Network Pattern Recognition

We compare the accuracy of SVM and Neural Network classification algorithm using the same data of ten drivers total. The data contains all eight indicators for each driver: DC, ISU, SF, SU, RB, PB, SPT and SPV. The neural network is using 90% data for training, 5% for validation and 5% for testing. The number of hidden neurons are set to 10 as default.


All Confusion Matrix

	1	2	3	4	5	6	7	8	9	10	
1	19 13.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	20 13.9%	0 0.0%	1 0.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.7%	90.9% 9.1%
3	0 0.0%	0 0.0%	16 11.1%	1 0.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 2.1%	0 0.0%	80.0% 20.0%
4	0 0.0%	0 0.0%	3 2.1%	16 11.1%	0 0.0%	0 0.0%	1 0.7%	0 0.0%	0 0.0%	0 0.0%	80.0% 20.0%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	9 6.2%	2 1.4%	1 0.7%	0 0.0%	0 0.0%	0 0.0%	75.0% 25.0%
6	0 0.0%	0 0.0%	0 0.0%	1 0.7%	1 0.7%	8 5.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	80.0% 20.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.7%	2 1.4%	0 0.0%	0 0.0%	33.3% 66.7%
8	0 0.0%	0 0.0%	0 0.0%	2 1.4%	0 0.0%	0 0.0%	7 4.9%	9 6.2%	0 0.0%	0 0.0%	50.0% 50.0%
9	0 0.0%	0 0.0%	1 0.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	8 5.6%	0 0.0%	88.9% 11.1%
10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.7%	0 0.0%	0 0.0%	10 6.9%	90.9% 9.1%
	100% 0.0%	100% 0.0%	80.0% 20.0%	76.2% 23.8%	90.0% 10.0%	80.0% 20.0%	9.1% 90.9%	81.8% 18.2%	72.7% 27.3%	90.9% 9.1%	80.6% 19.4%

Fig. 3. Confusion Matrix of Neural Network Pattern Recognition

MATLAB could plot confusion matrices for training, testing, and validation, and the three kinds of data combined. Figure 3 shows the confusion matrix for all data combined. As we can see in the figure, the network outputs are quite accurate with high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies which is 80.6% in this experiment. Each column of the matrix represents the classification result for each driver of the target class number. Each row of the matrix shows all the pre-trip data that are identified as the corresponding output class, also marked as the row number.

Note that training the neural network multiple times will generate different results due to different initial conditions and sampling. Therefore we trained the model ten times and got the accuracy of 72.5% on average.

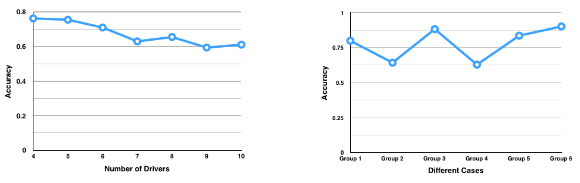
Comparing to the high accuracy of neural network result, the SVM algorithm yields a low accuracy of 59.0% over the same data using the same indicators. Differ from the neural network configuration, the SVM test each trail independently. For example, the SVM uses all the data except only one testing data to train the model. Then it classifies the one testing data using the trained model and yields a result. We use a for loop to let the algorithm eventually tests all the data points. We count the total number of correct identification. The accuracy of SVM classification is the ratio of number of correct cases to total number of data points.

Group	Driver IDs
Group1	1, 2, 3, 4
Group2	5, 6, 7, 8
Group3	1, 3, 5, 7
Group4	2, 4, 6, 8
Group5	1, 2, 9, 10
Group6	1, 3, 8, 10

TABLE I
Drivers of each group

C. Different Drivers

We seek to observe the effect of number of drivers on the classifier performance. We used data set of different drivers to test our algorithm. The indicators used in the experiments in this subsection are DO, SF, ISU, SU, RB, PB and SPV. The SVM classification accuracy of data set with different number of drivers is shown in Figure 4(a). We started testing with four drivers data (with driver ID 1, 2, 3 and 4). Then we continuously added more drivers into the data set and run the same test. Eventually, we added all ten drivers into the data set. In general, as shown in Figure 4(a), the accuracy decreases as more drivers added into the data set. Note that, when driver 8 was added into the data set, the accuracy increases 2.5%. This indicates that some drivers are standing out among other drivers. It is reasonably to say that these drivers have driving habit that are quite different from others.



(a) Different Number of Drivers (b) Different Drivers

Fig. 4. SVM Classification Accuracy of data set with different drivers

We conducted another experiment to understand the difference between driving habits further. Using the same indicators, we tested the algorithm on different groups of drivers. All of these groups have four drivers. The specific driver IDs of each group are shown in Table I.

The SVM classification accuracy of each groups are shown in Figure 4(b). As expected, the fluctuation in the plot indicates the difference between each driver varies. When two drivers have the similar driving habit, their feature vectors are also similar to each other. Therefore it is harder for our algorithm to differentiate some drivers while others might be easier to be identified.

D. *Different Features*

We also analyzed the importance of individual indicators. Figure 5 shows the accuracy obtained using only one indicator at a time from each trace, for three cases: Driver Group 1, Driver Group 2 and all ten drivers. Shown clearly in the Figure, some indicators are more influential than others in differentiating drivers. However, it is hard to say which indicator is most influential in classification because one indicator might be efficient identifying some drivers while having low performance differentiating others. For example, the BV indicator yields high accuracy when differentiating Driver Group 1 and has bad performance for Group 2.

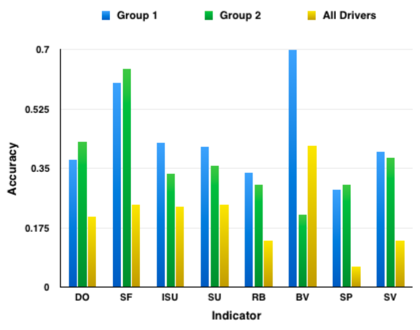


Fig. 5. SVM Classification Accuracy of single indicators

This result inspires us to investigate further of the system performance when using different combination of indicators. Therefore we carried out experiments across different Groups using different combinations of indicators. The SVM classification accuracy results are shown in Figure 6. The first five indicators are DO, SF, ISU, SU and RB which are used in [1]. Each line in the figure represents one combinations of indicators used, across five Groups of drivers and all ten drivers data set. As shown in the Figure, there is no global winner at this time.

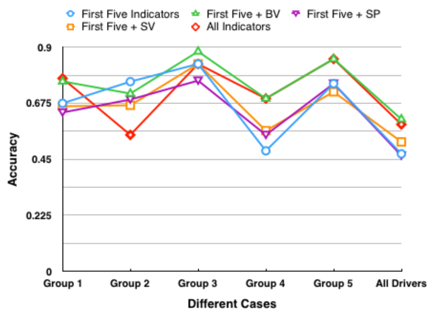


Fig. 6. SVM Classification Accuracy using different indicators

The combination of six indicators consisting of DO, SF, ISU, SU, RB and BV, represented by the green line has the better results comparing to other combinations. Note that, adding the correct indicator will improve the performance of the system. For example, adding the BV indicator to the first five indicator raises the accuracy level from the blue line to the green line. On the contrary, a bad indicator will lower the accuracy level represented by the line shifting downward.

VII. FUTURE WORK

Results show adding new feature could increase the classification accuracy in general. In-vehicle sensors provide much richer data than what our system could use so far. Analyzing and extracting new indicators from data could reasonably increase the accuracy of classification algorithm.

As discussed before, different combinations of indicators have different performance over Driver Groups. There might be a more smart and sophisticated way of training the SVM classifier so that it will have a global optimized solution.

VIII. CONCLUSIONS

We have shown that adding new features could improve the accuracy by approximately 10% comparing to the old features in general. Some new indicator like Speed Peak Value could only improve the performance of the system over some specific sub-dataset. The accuracy of the SVM classification algorithm can be further increased by adding more indicators extracted from in-vehicle data. The Android implementation of system is able to pass pre-trip in-vehicle data from one activity to another. The latter activity can extract all the indicators system needs and apply a simple version of classification model. The identification results will be displayed on the screen. One can expect that, with newer vehicles and next generation of mobile devices, our driver identification system will have better performance and user experience.

ACKNOWLEDGMENT

The author would like to thank Professor Gruteser for his instruction and support. The author would also like to thank Gorkem Kar for working with me and providing his great help on this project. In addition, the author would like to thank everyone who took the road test and collected the in-vehicle data. All of our experiments are carried out upon the data they provided.

References

[1] G. Kar, S. Jain, J. Chen, M. McCartney, F. Bai, M. Gruteser, R. Govindan. Pre-Trip Driver Identification from In-Vehicle Data.

[2] The openxc platform. Accessed: 2015-12-5.

[3] S. Choi, J. Kim, D. Kwak, P. Angkititrakul, and J. H. Hansen. Analysis and classication of driver behavior using in-vehicle can-bus information. In Biennial Workshop on DSP for In-Vehicle and Mobile Systems, pages 17-19, 2007.

[4] T. Churches and P. Christen. Blind data linkage using n-gram similarity comparisons. In Advances in Knowledge Discovery and Data Mining, pages 121-126. Springer, 2004.

[5] T. Churches and P. Christen. Some methods for blindfolded record linkage. BMC Medical Informatics and Decision Making, 4(1):9, 2004.

[6] S. Corrigan. Introduction to the controller area network (can). Application Report, 2008.

[7] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno. Automobile driver ngerprinting. Proceedings on Privacy Enhancing Technologies, 2016(1):34-50, 2016.

[8] X. Gao, B. Firner, S. Sugrim, V. Kaiser-Pendergrast, Y. Yang, and J. Lindqvist. Elastic pathing: Your speed is enough to track you. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '14, pages 975-986, New York, NY, USA, 2014. ACM.

[9] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In Pervasive computing, pages 390-397. Springer, 2009.

[10] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in gps traces via density-aware path cloaking. Proceedings of CCSaA Z07, 2007.

[11] B. Hoh, T. Iwuchukwu, Q. Jacobson, D. Work, A. Bayen, R. Herring, J.-C. Herrera, M. Gruteser, M. Annavaram, and J. Ban. Enhancing privacy and accuracy in probe vehicle-based trac monitoring via virtual trip lines. Mobile Computing, IEEE Transactions on, 11(5):849-864, May 2012.

[12] M. A. W. J. A. Hartigan. Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1):100-108, 1979.

[13] Y. Jiang, H. Qiu, M. McCartney, W. G. J. Halfond, F. Bai, D. Grimm, and R. Govindan. Carlog: A platform for exible and ecient automotive sensing. In Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, SenSys '14, pages 221-235, New York, NY, USA, 2014. ACM.

[14] C. Karatas, L. Liu, H. Li, J. Liu, Y. Wang, J. Yang, Y. Chen, M. Gruteser, and R. Martin. Leveraging wearables for steering and driver tracking,. In IEEE International Conference on Computer Communications (Infocom) 2016. ACM, 2016.

[15] J. Krumm. Inference attacks on location tracks. In Proceedings of the 5th International Conference on Pervasive Computing, PERVASIVE'07, pages 127-143, Berlin, Heidelberg, 2007. Springer-Verlag.

[16] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: Rich monitoring of road and trac conditions using mobile smartphones. In Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, SenSys '08, pages 323-336, New York, NY, USA, 2008. ACM.

[17] A. Pentland and A. Lin. Modeling and prediction of human behavior. Neural Computation, 11:229-242, 1995.

[18] M. Scannapieco, I. Figotin, E. Bertino, and A. K. Elmagarmid. Privacy preserving schema and data matching. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 653-664. ACM, 2007.

[19] R. Schnell, T. Bachteler, and J. Reiher. Privacy-preserving record linkage using bloom lters. BMC medical informatics and decision making, 9(1):41, 2009.

[20] H. Summala. Automatization, automation, and modeling of driver's behavior. In Recherche - Transports - SAl'curitAl', pages 35-45. Elsevier, 2000.

[21] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin. Sensing vehicle dynamics for determining driver phone use. In Proceeding of the 11th annual international conference on Mobile systems, applications, and services, pages 41-54. ACM, 2013.

[22] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. P. Martin. Detecting driver phone use leveraging car speakers. In Proceedings of the 17th annual international conference on Mobile computing and networking, pages 97-108. ACM, 2011.

[23] B. Zan, P. Hao, M. Gruteser, and X. Ban. Vtl zone-aware path cloaking algorithm. In Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on, pages 1525-1530, Oct 2011.

[24] OnStar by GM. Accessed: 2015-11-10.