Collaborative Cloud-Edge Computation for Personalized Driving Behavior Modeling

Xingzhou Zhang*
Institute of Computing Technology, University of Chinese Academy of Sciences
Beijing, China
Wayne State University
MI, USA

Mu Qiao
IBM Research - Almaden
San Jose, CA, USA

Liangkai Liu
Wayne State University
Detroit, MI, USA

Yunfei Xu
DENSO International America Inc
San Jose, CA, USA

Weisong Shi
Wayne State University
Detroit, MI, USA

ABSTRACT

Driving behavior modeling is an essential component of Advanced Driver Assistance Systems (ADAS). Existing methods usually analyze driving behaviors based on generic driving data, which do not consider personalization and user privacy. In this paper, we propose pBEAM, a collaborative cloud-edge computation system for personalized driving behavior modeling. The driving behavior model is built on top of Generative Adversarial Recurrent Neural Networks (GARNN), which adapts to the dynamic change of normal driving. Transfer learning from cloud to edge improves the model performance and robustness on the edge. We prune the deep neural networks in the cloud in order to minimize the model transferring load while maximally preserve the original model performance. A personalized edge model is trained on top of the pruned model using CGARNN-Edge (Conditional GARNN), which considers drivers' personal or contextual information as additional conditions. User privacy is well protected as no personal data needs to be uploaded to the cloud. Experimental results on driving data from both real world and driving simulator show that the proposed CGARNN-Edge achieves the best performance among all the methods.

CCS CONCEPTS

• **Applied computing** → **Transportation**; • **Computing methodologies** → **Cooperation and coordination**.

KEYWORDS

driving behavior model, anomaly detection, personalization, generative adversarial networks, edge computing, transfer learning

ACM Reference Format:
Xingzhou Zhang, Mu Qiao, Liangkai Liu, Yunfei Xu, and Weisong Shi. 2019. Collaborative Cloud-Edge Computation for Personalized Driving Behavior Modeling. In *SEC '19: ACM/IEEE Symposium on Edge Computing, November 7–9, 2019, Arlington, VA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3318216.3363310

1 INTRODUCTION

The past decades have seen significant improvements in road safety. However, numerous traffic accidents are still occurring every day. It is estimated that over 1.2 million people worldwide die in road crashes each year, with millions more sustaining serious injuries and living with long-term adverse health consequences [25]. Most traffic accidents are caused by human errors, such as speeding, distracted driving, drowsy driving, and drunk driving.

Advanced driver assistance system (ADAS) are developed to automate, adapt and enhance vehicle systems to increase road safety. ADAS provides various safety features including blind spot monitoring, lane change assistance, and forward collision warnings. Driving behavior modeling is an essential component in ADAS to detect abnormal driving behaviors, send early alerts, and therefore reduce accidents [38]. Driving behavior modeling can also be used by insurance companies to determine the vehicle insurance premium [3, 24].

Many methods have been proposed to model driving behaviors and detect anomalies. For example, the driver's facial features, such as the state of eyes and mouth, are used in classification methods to detect whether the driver is drowsy or distracted [16]. These methods usually require drivers to wear glasses or leverage in-car cameras. Driving data, such as vehicle velocity, angular velocity, and acceleration, is also used in driving behavior analysis [20, 36]. These data are often collected through on-board diagnostic (OBD) sensors or smart phones. While many research efforts have been devoted to driving behavior analysis, several open challenges still remain when deploying the driving behavior models in real practice:

*This work was done when the author was a visiting student at Wayne State University.

SEC '19, November 7–9, 2019, Arlington, VA, USA

ACM ISBN 978-1-4503-6733-2/19/11…$15.00
https://doi.org/10.1145/3318216.3363310

(1) Personalization. Existing driving behavior models are usually trained on generic datasets, which do not consider driving contextual information, such as drivers' individual difference, location, weather, and traffic conditions. However, driving is significantly influenced by these factors, which lead to personalized driving behaviors. Drivers may have distinct driving behaviors because of their individual difference, such as age group, gender, and driving experience. For example, the driving behavior model that is built on datasets from experienced drivers cannot appropriately capture the behavior of novice drivers. The same driver may have different behaviors under various situations. For example, the driving speed of 60 miles per hour on US highways is normal in sunny days. However, it is dangerous in blizzard. If the driving behavior model is only trained on data under normal weather conditions, it will not be able to detect abnormal behaviors under severe weather conditions.

(2) Latency. Real-time performance is a stringent requirement for ADAS. For example, fatigue driving or other abnormal driving behaviors should be detected immediately, in order to avoid accidents or minimize damage. There are two types of methods to run the driving behavior model: cloud-based and edge-based methods. For the cloud-based method, the computing unit sends the driving behavior data to the model in the cloud and then receives decisions over the network [37]. This method suffers greatly from the stability and latency of the network. For the edge-based method, ADAS analyzes the driving behavior on board, which can minimize the network latency. However, the computing unit on vehicle usually has limited computation and storage resources, which are not sufficient to run a large driving behavior model.

(3) Privacy. As General Data Protection Regulation (GDPR) [28] has already taken effect, the collection of driving data as well as the modeling of driving behavior need to consider various privacy regulations. However, most existing approaches usually collect users' driving data, send them directly to the cloud, build the model and then conduct analysis. This process may put the user privacy at risk.

(4) Integration. The driving data is being collected by various applications for their own driving behavior analysis. For example, the insurance company analyzes driving data to dynamically adjust the insured's premium. The ride sharing company detects abnormal driving behaviors, in order to increase ride safety. It is desired to have an integrated driving behavior model that can provide a universal service to various applications. The efforts on data collection and computational analysis can therefore be significantly reduced.

To address these challenges, we propose pBEAM, a collaborative cloud-edge computation system for personalized driving behavior modeling. Comparing with the traditional cloud-based approaches which upload all the data to a centralized cloud service, edge computing can reduce or minimize network latency [29, 30] and address privacy concerns [21]. Vehicles are important players in edge computing, which are suitable to perform certain computational tasks [33] when building driving behavior models. Figure 1 shows the high level work flow of our method. Specifically, we first train a common baseline model in the cloud using all the driving data, such as velocity, orientation, and acceleration over time. There data are anonymized and integrated from users who would like to share

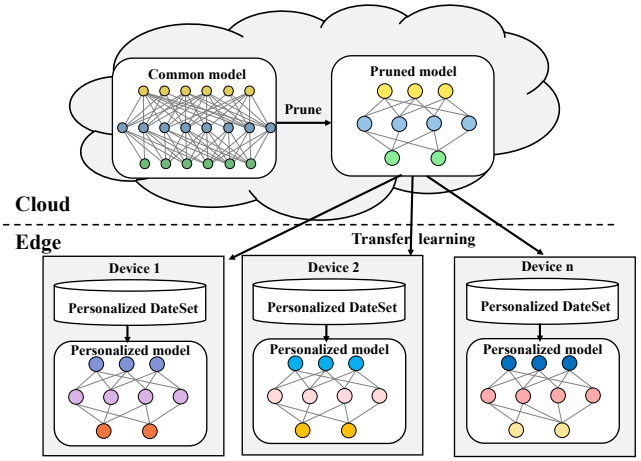

Figure 1: The collaborative cloud-edge method

their data in the cloud. The cloud model is based on Generative Adversarial Recurrent Neural Networks (GARNN), which adapts to the dynamic change of users' normal driving. This deep neural network model is further pruned or compressed to a smaller model, and transferred to the edge device on each vehicle. After compressing or pruning, the computational cost of running the deep learning model can be reduced, thereby satisfying the computing power restrictions of the on-board hardware. We then train a personalized model on top of the pruned model through transfer learning, considering the specific driving condition or context information.

There are several advantages of our method. First, the Generative Adversarial Recurrent Neural Networks (GARNN) are adaptive to the dynamic change of normal driving. Instead of only training a binary classifier to identify abnormal driving behaviors, GARNN learns both a generator that tries to produce true driving data as well as a discriminator that tries to distinguish between generated data and true driving data. Essentially, GARNN learns what the true driving data should look like. Any abnormal driving behaviors that do not follow the distribution of true driving data will be considered as anomalies. The labeling of anomalous data is not needed when training the model, and is only used for the model performance evaluation. Second, the transfer learning from cloud to edge improves the model performance and robustness. The pruning of deep neural networks minimizes the transferring load of the model while maximally preserves its original performance. Third, the model trained on edge devices outperforms the original cloud model, since personal and contextual information are considered in the training. Last but not the least, users' privacy is well protected because they do not need to upload personal data to the cloud.

We summarize the contributions of this paper as follows:

- A personal driving behavior modeling system, pBEAM, is presented. pBEAM can improve the overall performance of detecting abnormal driving behaviors and reduce detection latency, while ensure the privacy of drivers. A RESTful engine in pBEAM provides universal modeling services to different applications.

- A deep learning model, CGARNN-Edge, is developed to model the driving behavior. Experimental results on driving data from both real world and a driving simulator show that the proposed CGARNN-Edge achieves the best performance among all the methods.
- A collaborative cloud-edge computation method that trains and prunes common models in the cloud and conducts transferring learning on the edge is developed. This method is general and can be used in other cloud-edge applications.

The rest of the paper is organized as follows. We introduce the related work in Section 2. The system design and implement are presented in Section 3 and Section 4, respectively. The performance evaluation and experimental results are reported in Section 5. We finally conclude and discuss future work in Section 6.

2 RELATED WORK

Driving behavior modeling has enjoyed wide applications in automotive engineering, for example, to increase road safety, to optimize battery consumption in electrical vehicles, to detect driver drowsiness from facial expressions, and to assess driving risk on certain routes. In this work, we are focused on developing deep learning methods for driving behavior modeling.

A lot of research efforts have been devoted to abnormal driving behavior detection, which are roughly divided into two categories: detection based on drivers data and detection based on driving data. For example, [10] uses driver facing cameras to record a large scale in-vehicle videos. They analyze drivers' facial expressions to detect drivers' drowsiness and distraction. Models based on driving data generally collect the driving trajectories and driving related features. For example, by analyzing the GPS trajectory data, [32] develops a Peer and Temporal-Aware Representation Learning based framework for driving behavior analysis and detects the driving operations and states of each driver. [14] proposes a smartphone-based sensing platform to detect aggressive driving behaviors.

Generative adversarial networks (GAN) is proposed to generate various types of data via adversarial training, where a generative neural network and a discriminative neural network are trained in tandem [7, 11, 15]. The generator tries to produce samples that look like true data while the discriminator tries to discriminate between generated samples and true data. In this work, we use GAN to detect abnormal driving behaviors. Specifically, the true data is the normal driving data. The generator and discriminator are built on top of Recurrent Neural Networks (RNN), which model the distribution of sequential driving data. The generator tries to produce data that match normal driving data as much as possible. The discriminator distinguishes between generated data and normal driving data. If there are abnormal driving behaviors, the corresponding driving data will be different from normal driving data. Therefore, we can use the finally trained discriminator to detect abnormal driving behaviors. Most recent work [18] applies GAN to imitate driving behaviors. However, it is used to learn how to drive via generative adversarial imitation learning (GAIL), which is fundamentally different from our problem. Our work is inspired by C-RNN-GAN [23], a recurrent neural network trained to generate continuous musical sequences via generative adversarial training. However, our final application is focused on discrimination, rather than generation. Conditional GAN is proposed by [22] to add additional conditions to both the generator and discriminator. In the same spirit, we apply conditional Generative Adversarial Recurrent Neural Network (CGARNN) to develop personalized driving behavior model, considering personal or contextual information in the model training.

Edge-based AI has attracted a lot of research attention recently. [31] propose a distributed deep neural networks over cloud, edge and end devices. [27] propose an edge cloud collaborative framework for video analysis, where a small model is implemented on the edge while a large model is implemented on the cloud. Privacy is an important issue in driving behavior modeling, which has not been well studied in the past. Edge computing enables us to train personalized driving model on edge devices, respecting data privacy. In this work, we propose a collaborative cloud-edge computing method for personalized driving behavior modeling. Instead of training a personalized model on the edge from scratch, we first train a common baseline model in the cloud and then transfer it to the edge device for personalized training or transfer learning.

However, it is non-trivial to transfer a large machine learning model from cloud to edge, as the storage and computing resources on edge are limited. Various deep neural network pruning or compression methods have been developed. [12] propose a simple but effective pruning method, where the importance of neural network connections are determined by their absolute weight values. Sparsity constraints are used in CNN pruning [34], where group sparsity is applied to find unimportant parameters and guide the pruning. [2] prune the model by iteratively deleting unimportant filters. [8] propose a filter-wise pruning pipeline to remove redundant filters from a trained CNN. In this work, we apply pruning to remove unimportant parameters from the cloud model to reduce the model size while preserve the model performance as much as possible.

Distributed machine learning and federated learning [4][17] also train models through the collaboration among multiple computing devices. The goal of distributed machine learning is to train a final and universal model on the decentralized data. Federated learning belongs to the general category of distributed machine learning, but focuses more on privacy protection. It enables edge devices to collaboratively train a shared machine learning model while keeping all the personal data on device. While the goal of our proposed method is to train a personalized model on each edge device, the model is not trained through a distributed approach, but rather transfer learned from a pruned common model in the cloud.

3 SYSTEM DESIGN

3.1 System Overview

In this section, we introduce pBEAM, a collaborative cloud-edge computation system for personalized driving behavior modeling. The traditional cloud-based methods require the uploading of driving data to the cloud, which may cause privacy issues as the driving data usually contain personal information. In addition, it is not efficient or even feasible to store all the data and perform model training on edge devices as the storage and computing resources are limited.

To address these challenges, we propose pBEAM in this paper. As shown in Figure 2, it has four stages to build pBEAM: (1) build

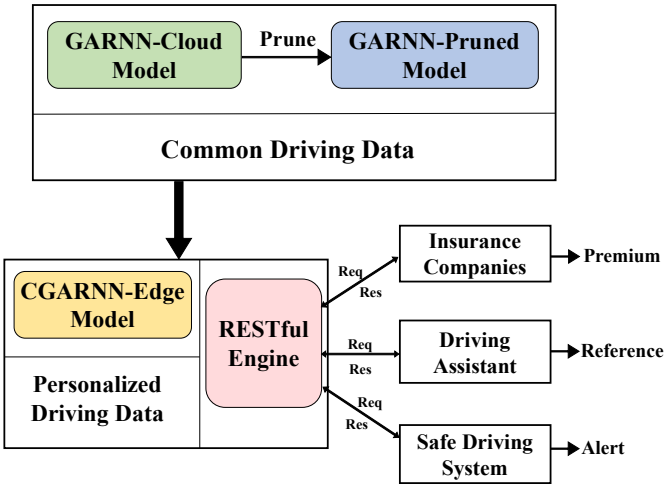

Figure 2: The overview of pBEAM. pBEAM includes four parts: GARNN-Cloud, GARNN-Pruned, CGARNN-Edge, and RESTful Engine.

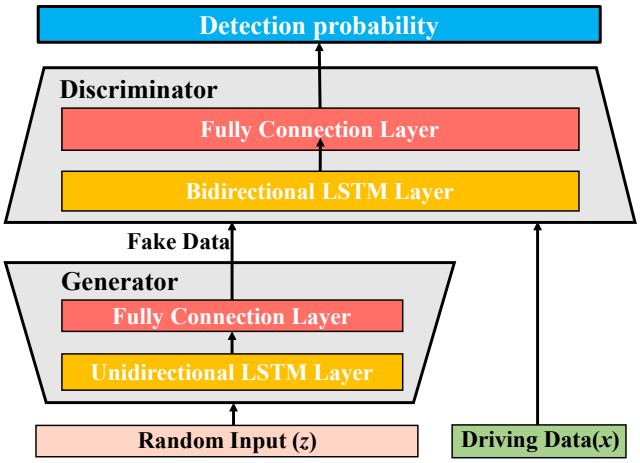

Figure 3: The architecture of GARNN-Cloud

a common baseline model in the cloud using all the driving data, denoted by GARNN-Cloud, (2) prune the baseline cloud model to reduce the total number of parameters and model size, denoted by GARNN-Pruned, and (3) transfer the pruned model to the edge device and retrain an edge model by considering conditional or contextual information, denoted by CGARNN-Edge, and (4) provide RESTful web services for third-party applications development, denoted by RESTful Engine. RESTful Engine is designed on top of CGARNN-Edge and provides normalized APIs to developers. For example, the CGARNN-Edge model can be used in the driving assistance systems to detect abnormal driving behaviors. In practice, GARNN-Cloud and GARNN-Pruned are located in the cloud servers in order to leverage the large-scale storage and computation infrastructure. The pruning of GARNN-Cloud helps to reduce the cost of training CGARNN-Edge on the edge. CGARNN-Edge and RESTful engine are deployed on the vehicle computing unit to train on personalized data and reduce the amount of data to be uploaded to the cloud. We will present the details of GARNN-Cloud, GARNN-Pruned, CGARNN-Edge, and RESTful Engine in the following sections.

3.2 GARNN-Cloud

To model the common driving patterns, we first build a baseline cloud model using Generative Adversarial Recurrent Neural Networks (GARNN) which is inspired by C-RNN-GAN [23]. GARNN is a continuous recurrent neural network with adversarial training. The overall architecture is shown in Figure 3. GARNN consists of two components: generator (G) and discriminator (D), which are built based on LSTM (Long Short-Term Memory networks), a type of recurrent neural networks [13]. The generator is trained to generate data that is indistinguishable from real normal driving data, while the discriminator is trained to identify whether the generated data is real or not. Note that since the driving data is time series, we use unidirectional LSTM in the generator to capture the temporal direction. In the discriminator, we instead use a bidirectional LSTM as the goal is to classify the driving data without the constrain of a particular sequential order.

The input of discriminator includes the fake data generated by generator and the real driving data. The loss functions of generator and discriminator, L_D and L_G, are defined as follows:

$$L_G = \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z^i))) ,$$

$$L_D = \frac{1}{m}\sum_{i=1}^{m}\left[-\log D(x^i) - (\log(1 - D(G(z^i))))\right] ,$$

where $z^{(i)}$ is a time series of uniform random vectors in $[0,1]^k$, k is the feature size, and $x^{(i)}$ is the time series of true driving data, normalized between 0 and 1 .

3.3 GARNN-Pruned

When the cloud model is finished training, we use pruning to reduce the total model size so that the model can be transferred to the edge device which has limited storage and computing resources. As shown in Figure 4, we prune the connections with low-absolute values. All connections with weights below a threshold are removed from the network, therefore converting a dense network into a sparse one. The threshold is a hyper-parameter that depends on the trade-off curve between compression ratio and prediction accuracy. We use the mask mechanism to implement model pruning; weights below the threshold are masked by zeros, while those above are masked by ones. By taking the dot product between the original weight tensor and the mask tensor, the connections with weights smaller than the threshold are set to zero, i.e., pruned.

We apply an automated gradual pruning algorithm [40] to cut the unimportant connections. Over a span of n pruning steps, the sparsity is increased from an initial sparsity value s_i to a final sparsity value s_f. The sparsity at time t is defined as follows:

$$s_t = s_f + \left(s_i - s_f\right)\left(1 - \frac{t - t_0}{n\Delta t}\right)^3$$

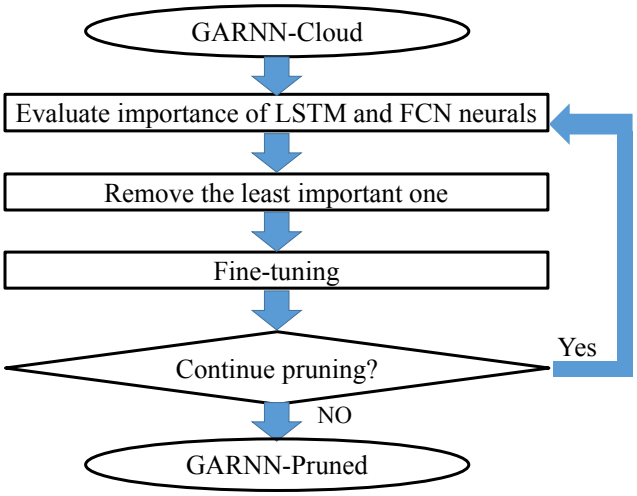

Figure 4: The process of model pruning

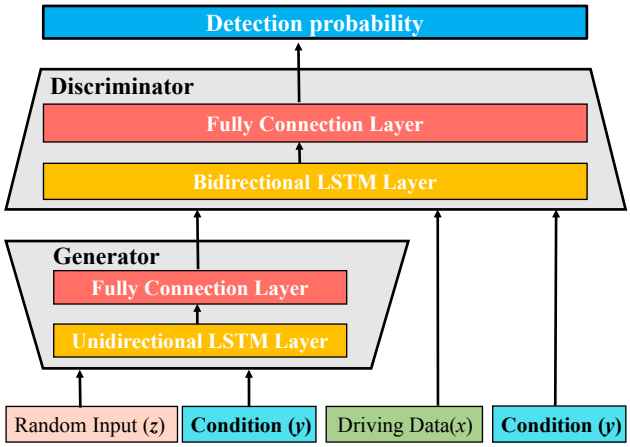

Figure 5: The architecture of CGARNN-Edge

where t_0 is the start step and Δt is the pruning frequency. In this paper, we start pruning the model at the first step and empirically set the pruning frequency to be 10. The initial sparsity s_i is 0 and the target sparsity s_f is set to 0.5. As the sparsity of the network increases, the weight masks are updated. Once the model's sparsity s_t achieves the target sparsity, the weight masks are no longer updated and the pruning process is completed. Comparing with the original cloud model, the pruned model may suffer an accuracy loss, but achieve a considerable reduction in the model size.

3.4 CGARNN-Edge

After the pruned model is transferred to the edge device, we will train a personalized model on top of GARNN-Pruned through transfer learning, considering the contextual information. Specifically, GARNN-Pruned is used as the initialization of the personalized model. We propose to use CGARNN as the personalized model, where both the generator and discriminator are conditioned on the context information $\boldsymbol{y}$. $\boldsymbol{y}$ can be any kind of contextual information, such as drivers' personal data, location, weather condition, and traffic situation. We perform the conditioning by feeding $\boldsymbol{y}$ into both discriminator and generator as additional input. Figure 5 shows the architecture of CGARNN-Edge.

The new loss functions of generator and discriminator, L_D and L_G, are defined as follows:

$$L_G = \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z^i|\boldsymbol{y}))) ,$$

$$L_D = \frac{1}{m}\sum_{i=1}^{m}\left[-\log D(x^i|\boldsymbol{y}) - (\log(1 - D(G(z^i|\boldsymbol{y}))))\right] ,$$

where the random input noise $\boldsymbol{z}$ and $\boldsymbol{y}$ are combined as a joint hidden representation in the generator, while $\boldsymbol{x}$ and $\boldsymbol{y}$ are combined as new input to the discriminator.

As shown in Figure 5, CGARNN-Edge will be deployed on the computing platform of the vehicle and trained on the personalized data. After the model training, the discriminator of CGARNN-Edge can be used to detect whether the driving behavior is normal or not. The input is the real-time driving behavior and the output is the probability of being an anomaly.

Recent research work is devoted to train the machine learning models on the edge based on the local data, along the same line as CGARNN-Edge [1]. However, one of the most significant challenges of training model on the edge is training data labeling [9]. For example, it is difficult to train an object detection model on the vehicle with images generated from the on-board camera, which requires image labelling in real time. CGARNN-Edge does not need the labeled data at the training stage as it only uses the driver's normal driving behavior as the input. The drivers' personal data (such as age group and gender) is the one-time input and CGARNN-Edge can obtain it at the beginning. CGARNN-Edge can acquire the information of location, weather condition, and traffic situation by using web services, such as the weather report services and the Google maps API.

3.5 RESTful Engine

After CGARNN-Edge is trained, the RESTful Engine will call for it and provide web service for the third-party applications. Figure 2 lists three potential services: (1) insurance companies leverage the driving behavior analysis for credit rating and premium adjustment, (2) driving assistant provides driving reference, and (3) ridesharing companies can detect abnormal driving behaviors to increase ride safety [19] and send alarms.

Figure 6 lists two RESTful APIs and their purposes when using GET operations. CGARNN-Edge is represented by a URI which consists of four fields. The first field is the IP address and port number of the vehicle. The second field represents the particular driving behavior model, i.e., CGARNN-Edge. The third field indicates the request type, such as real-time and batch. The fourth field is the arguments which will be sent to the RESTful Engine. For example, if the driving assistant system needs to obtain the driving behavior in the real time manner for the sake of safety, it should follow the URI shown in *Example 1* in Figure 6 and send the timestamp argument. The response is the detection probability that the driving

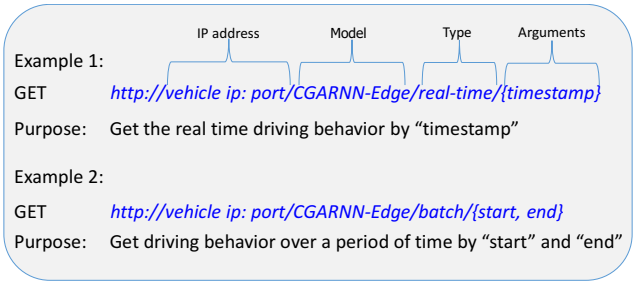

Figure 6: The examples of RESTful Engine

behavior is normal or not at that moment. If an insurance company wants to know the driving performance of a customer in the last month, it needs to follow the URI shown in *Example 2* and sends the starting and ending timestamp. The response is the probability of the driving behavior being abnormal during that period.

The design of RESTful Engine can address the challenge of integration. In this case, there is no need to build a driving behavior model for each application separately. The integration function of the RESTful Engine and the privacy-preserving goal of the proposed system do not conflict. The original driving data and the trained CGARNN-Edge will be kept on the vehicle and not be transferred. The third-party application can only call the model via the RESTful API. Moreover, the access permission of the RESTful Engine is managed by the driver, which will protect user privacy.

4 SYSTEM IMPLEMENT

We will present the implementation and hardware/software configuration of the entire system in this section. Figure 7 shows the overview of the system implementation. The GARNN-Cloud and GARNN-Pruned models are trained on the cloud server and CGARNN-Edge model is trained on the edge, i.e., vehicle computing unit. When the training process of CGARNN-Edge is completed, it will predict the driving behavior based on the real-time driving data collected by the on-board sensors.

4.1 Data collection and processing

There are many options to collect the driving data, such as On-board diagnostics (OBD), Inertial Measurement Unit (IMU), and even GPS sensors in the mobile phone. In this work, we adopt Xsens MTi-G-700 [35] to collect the real-time driving data. The MTi-G is an integrated GPS and IMU with a Navigation and Attitude and Heading Reference System (AHRS) processor. The sampling frequency is 100ms. As shown in Figure 8 [6], the data collected by Xsens MTi-G-700 has three coordinates, x, y, and z. It consists of 12 dimensions: orientation$_x$, orientation$_y$, orientation$_z$, velocity$_x$, velocity$_y$, velocity$_z$, angularrate$_x$, angularrate$_y$, angularrate$_z$, latitude, longitude and altitude.

The driving data is essentially multivariate time series data. To analyze the driving data and model driving behaviors, we divide the time series into overlapping sliding windows. Each sliding window consists of driving data over five seconds, such as 0.0-5.0s, 1.0-6.0s, and 2.0-7.0s. Since the collection frequency is 100 ms, each sliding window has 50 samples. Each sliding window is considered as an input of the driving behavior models, including GARNN-Cloud, GARNN-Pruned, and CGARNN-Edge.

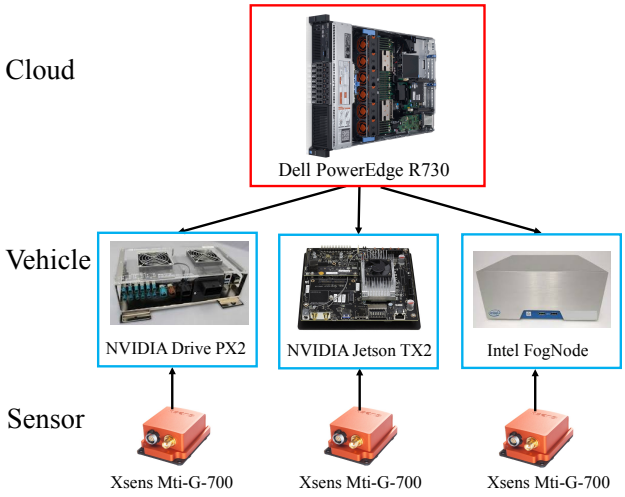

Figure 7: System implementation

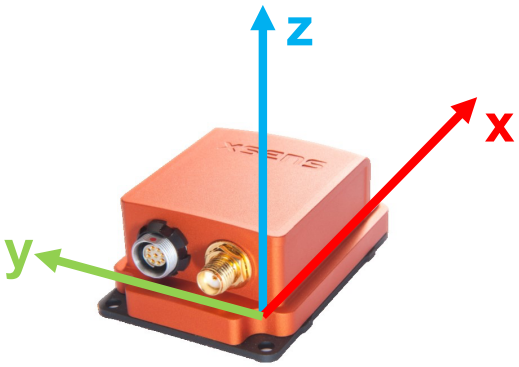

Figure 8: The Xsens Mti-G-700 sensor [6]. x positive to the East (E), y positive to the North (N), and z positive when pointing up (U).

4.2 Hardware Configuration

The GARNN-Cloud and GARNN-Pruned models are trained on a high performance server, Dell PowerEdge R730, which has two NVIDIA Tesla P100 GPUs and two Intel Xeon E5-2640 CPUs. The CGARNN-Edge model is trained on the edge, i.e., the vehicle computing unit.

Since the same model will have different system performance when running on different hardware architectures, we choose NVIDIA Drive PX2 (AutoChauffeur), NVIDIA Jetson TX2, and Intel FogNode to simulate the heterogeneous vehicle computing units. NVIDIA Drive PX2 and Jetson TX2 are GPU-based hardware platform and Intel FogNode only provides CPU in this paper. The configurations of the hardware are listed in Table 1.

Table 1: Hardware Setup

Hardware	CPU	GPU	CPU Frequency	Cores	Memory	OS
Dell PowerEdge R730	Intel Xeon E5-2640	2x Tesla P100	2.4GHz	40	128GB	Linux 4.4.0-128-generic
NVIDIA Drive PX2	Denver 8, Cortex A57	2x 512-core Pascal	2GHz	12	16GB	Linux 4.9.80-tegra
NVIDIA Jetson TX2	Denver 2, Cortex A57	256-core Pascal	2GHz	6	8GB	Linux 4.4.38-tegra
Intel FogNode	Intel Xeon E3-1275	Null	3.6GHz	4	32GB	Linux 4.13.0-32-generic

NVIDIA Drive PX2 is designed as an high-performance and energy-efficient autonomous vehicle computing platform for functionally safe AI-powered self-driving, which is also suitable for the ADAS development. NVIDIA Jetson TX2 is a fast, power-efficient embedded AI computing device which is built around the NVIDIA Pascal-family GPU. It is also one of the most popular edge devices. Jetson TX2 is appropriate to be used as a vehicle computing unit. Intel FogNode is a versatile reference design in a self-contained enclosed chassis for testing and demonstration of edge use cases. Although the computing power cannot compare with the high-performance computer on the cloud, it is better than a normal personal computer. It can be used as the CPU based computing unit on the vehicle.

4.3 Software Configuration

The performance of deep learning libraries varies on heterogeneous hardware platforms [39]. Based on the popularity and the overall performance, we choose TensorFlow as the deep learning library and use *Python* API to build driving behavior models. The entire system is implemented in *Python* 2.7 programming language. For the GPU based hardware platform, CUDA library is leveraged to improve the performance. Table 2 shows the software setup in the system.

Table 2: Software Setup

Hardware	Python	CUDA version	TensorFlow
PowerEdge	2.7.12	9.0.176	GPU, 1.7
Drive PX2	2.7.11	9.2.78	GPU, 1.7
Jetson TX2	2.7.12	9.0.252	GPU, 1.7
FogNode	2.7.15	Null	CPU, 1.7

To provide RESTful service, we build a web server based on the event-driven networking engine *twisted*. Twisted accepts the request with the argument from the third-party applications and returns the driving behavior. A built-in database is used to store the historical driving behavior.

5 PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed method and compare it with several anomaly detection baselines. We first describe the data sets and experimental setup, and then present the performance of different methods from both system and algorithm perspectives.

5.1 Dataset

Two datasets are used in our experiment: one is from the real-world driving environment and the other is from a driving simulator.

As is described in Section 4.1, the real-world driving data is collected through Xsens Mti-G-700 at a frequency of 100 ms. We divide the time series driving data into overlapping sliding windows. The collection frequency is 100 ms, so each sliding window has 50 samples. This vehicle is driven in the Bay area of California. The real-world driving data is divided into two sets. The first set is used to build the generic GARNN-Cloud model. It includes sixteen driving trajectories from eight drivers. The total driving time is 432 minutes and the total sample size is 259,205. We construct a dataset based on the first set. Since the real-world driving data only includes normal driving behaviors, we simulate several different kinds of abnormal driving data, which will be introduced later. The real-world normal driving data is labelled as 1 while the simulated abnormal driving data is labelled as 0. The ratio of normal and abnormal data is one to one. The dataset is split into training and test based on the 80%-20% principle. Note that the simulated abnormal driving data is not needed when we train the GAN-based model, and is only used in the model performance evaluation.

The second set of the real-world driving data is carefully designed to train CGARNN-Edge which considers different driving conditions. Table 3 summarizes the details of the data for building CGARNN-Edge. There are three drivers A, B, and C. Driver A has three driving trajectories while the other two have one trajectory, respectively. These five trajectories are collected at different time and under different weather. Note that all the driving trajectories follow the same driving routes, i.e., the same start and end points. The eight drivers in the first set of the data also include the three drivers in the second set. However, the sixteen driving trajectories in the first set do not include the five trajectories in the second set.

Table 3: Real world driving data for CGARNN-Edge

driver	time	weather	sample size
driver A_1	2 pm	cloudy	15255
driver A_2	5 pm	rain	19143
driver A_3	5 pm	cloudy	15402
driver B	2 pm	cloudy	18993
driver C	2 pm	sunny	15751

The second dataset was obtained from driving simulators in a controlled experiment [26]. The driving data was collected at a frequency of 1.0 second. Similar to the real-world data, we also divide the driving simulator data into overlapping sliding windows. The

data has 5 samples in each sliding window, as its collection frequency is 1.0 second. The driving simulator data contains data from 68 volunteers who drove on the same highway under four different conditions: no distraction, cognitive distraction, emotional distraction, and sensorimotor distraction. The drivers include male and female, with age group divided into young and old. The detailed sample size are summarized in Table 4. This data has six features related to the driving behavior: speed, acceleration, brake force, steering, lane offset, and lane position signal. For this data set, we also use 80% for training and the remaining 20% for test.

Table 4: Driving simulator data

driver	no distraction	cognitive	emotional	sensorimotor
Young Male	785	826	795	827
Young Female	794	797	766	793
Old Male	791	768	697	793
Old Female	856	831	822	1026

5.2 Experiment Setup

5.2.1 Baselines. We compare the performance of the proposed methods with three anomaly detection baselines: Local Outlier Factor (LOF), LSTM, and Bi-directional LSTM (Bi-LSTM). LOF is an unsupervised distance based method, which measures the deviation of a data point with respect to its density in the neighborhood formed by the k nearest neighbors [5]. We empirically set the numbers of neighbors to 200. LSTM is widely used to model and classify sequential data. Both bidirectional and unidirectional LSTMs are used as baselines. We use the many-to-one LSTM architecture, followed by a fully connected layer and a softmax classifier, which outputs the probability of a sequence being abnormal or not. All the deep learning models are implemented using TensorFlow 1.7 and LOF is from the sklearn python package.

5.2.2 Abnormal driving behaviors. Since real-world driving data only includes normal driving behaviors, we simulate four different kinds of abnormal driving behaviors as follows:

- Anomalous Driving 1: Speeding is one of the most dangerous driving behaviors. We multiply velocity_x and velocity_y by 1.3 to simulate speeding.
- Anomalous Driving 2: When the driver is distracted, driving trajectory may show abnormal or unexpected turn. We simulate such scenario by adding Gaussian noise (mean 1.0, variance 1.0) to each angularrate_x and angularrate_y.
- Anomalous Driving 3: Lots of accidents are caused by drowsy and fatigue driving, where the exhibited driving orientation is abnormal. We simulate such scenario by adding Gaussian noise (mean 1.0, variance 1.0) to orientation_x and orientation_y.
- Anomalous Driving 4: In certain emergent scenario, the speed, orientation and angular rate can all become abnormal. We add Gaussian noise to all the dimensions to simulate this scenario. The Gaussian mean and variance are set to 0.2, a relatively smaller value than the previous three scenarios, since there are changes in every dimension.

5.2.3 Hyper-parameters. For all the deep learning methods, we use the RMSProp optimizer, as it has shown good performance in modeling sequential data. The learning rate is set to be 10^{-5}. The batch size is 10 and the number of epochs is 100. In Generative Adversarial Recurrent Neural Networks, the number of LSTM layer in both generator and discriminator is 1. Each LSTM cell has 32 neurons.

5.3 Experiment Results

We evaluate the proposed method and baselines on two aspects: system performance and algorithm performance. As a core component in ADAS, the time requirement of abnormal driving behavior detection is stringent. Therefore, the model inference time is an important metric. When the model is transferred from cloud to edge, the model size and the memory footprint are also important for the edge device. Therefore, in our evaluation, the system performance includes inference time, model size, and memory footprint, while the algorithm performance includes AUC (area under curve), Precision, Recall, and F1.

Table 5: System performance on NVIDIA Jetson TX2

Model	Time (ms)	Size (KB)	Memory (MB)
LOF	1321.0	NA	76.9
LSTM	119.2	1153.0	530.1
Bi-LSTM	214.9	2364.0	546.9
GARNN-Cloud	260.2	9405.0	611.3
GARNN-Pruned	220.8	7403.0	521.6
CGARNN-Edge	222.1	7403.0	524.5

Table 6: System performance of CGARNN-Edge

Hardware	Time (ms)	Memory Footprint (MB)
NVIDIA Drive PX2	10.2	819.3
NVIDIA Jetson TX2	96.3	820.1
Intel FogNode	44.25	251.8

5.3.1 System Performance. As shown in Table 5, on the edge device NVIDIA Jetson TX2, we evaluate the system performance of our proposed methods and the baselines on the real-world data. For the deep learning models, we calculate their average inference time for 10 times. When LOF is used to detect the abnormal driving behavior, it has to read all the data and calculate the distances among all the data points, which is time consuming. As a comparison, the inference time of deep learning models is much smaller. On average, LOF takes 5 times longer than the deep learning models when detecting an anomaly. Because LOF has the simplest architecture and the smallest number of parameters, its memory footprint is the least, which is only about 1/7 - 1/8 of the other models. When GARNN-Cloud gets pruned, its inference time, size, and memory are reduced by 15.38%, 21.29%, and 14.49%, respectively.

Since the proposed CGARNN-Edge will be deployed on heterogeneous computing platforms, we evaluate the performance of CGARNN-Edge on Drive PX2, Jetson TX2, and FogNode. Table 6 shows the inference time and memory footprint of CGARNN-Edge. We run the

Table 7: Performance on real-world data with simulated anomalies.

	Models	Anomaly1	Anomaly2	Anomaly3	Anomaly4
AUC	LSTM	0.762	0.707	**0.901**	0.835
	Bi-LSTM	0.821	0.727	0.878	0.849
	GARNN-Cloud	**0.892**	**0.757**	0.897	**0.891**
	GARNN-Pruned	0.831	0.712	0.832	0.772
P	LOF	0.782	0.784	0.89	0.863
	LSTM	0.730	0.723	**0.940**	0.843
	Bi-LSTM	0.793	0.795	0.932	0.864
	GARNN-Cloud	**0.832**	**0.798**	**0.940**	**0.883**
	GARNN-Pruned	0.792	0.705	0.920	0.801
R	LOF	0.820	0.789	0.890	0.853
	LSTM	0.793	0.732	0.860	0.845
	Bi-LSTM	0.79	0.785	0.870	0.872
	GARNN-Cloud	**0.943**	**0.801**	**0.892**	**0.892**
	GARNN-Pruned	0.734	0.692	0.872	0.865
F1	LOF	0.801	0.786	0.890	0.858
	LSTM	0.760	0.727	0.898	0.844
	Bi-LSTM	0.791	0.790	0.900	0.868
	GARNN-Cloud	**0.884**	**0.799**	**0.915**	**0.887**
	GARNN-Pruned	0.762	0.698	0.895	0.832

CGARNN-Edge model 10 times and perform 100 inferences each time. We then calculate the average inference time and its standard deviation. The average inference time of CGARNN-Edge on Drive PX2, Jetson TX2, and FogNode are 10.2ms, 96.3ms, and 44,25ms, with standard deviations 3.92, 7.21, and 4.35, respectively. The inference time of CGARNN-Edge is all less than 100ms, which meets the real-time requirement. Drive PX2 spends a shorter time than Jetson TX2 and FogNode since Drive PX2 has a powerful GPU. The memory footprint of Drive PX2 and Jetson TX2 are similar and they are more than 3x of FogNode.

Summary: The system side experimental results indicate that the proposed system can meet the real-time requirement on heterogeneous hardware platforms.

5.3.2 Algorithm Performance. For all the deep learning models, we use AUC (area under curve), Precision, Recall, and F1 to evaluate their algorithm performance. Note that AUC is not available in the performance evaluation of LOF. Since the algorithm performance is independent of the hardware platform, this paper only presents the experimental results on NVIDIA Jetson TX2. Table 7 shows the performance of all the methods on the real-world driving dataset under four different types of anomaly simulations. We abbreviate Precision and Recall as "P" and "R" in all the tables and figures. To demonstrate the overall performance of the methods, in Figure 9, we show their average AUC, Precision, Recall, and F1 over four anomaly simulations on the real-world dataset. As we can see, GARNN-Cloud achieves the best performance on all the metrics. When GARNN-Cloud is pruned (i.e., the GARNN-Pruned method), all the performance metrics are decreased. For anomaly 1, anomaly 2, and anomaly 4, the AUC, Precision, Recall, and F1 of GARNN-Cloud are much higher than that of the other models. For anomaly 3, LSTM has the highest AUC and Precision, which are 0.901 and 0.940, respectively. However, the Recall of LSTM is not as good as GARNN-Cloud, which leads to a lower F1 than GARNN-Cloud. The Precision of LOF is lower than Bi-LSTM on all the anomalies while the Recall of LOF is better than Bi-LSTM on anomaly 1, anomaly 2, and anomaly 3. Overall, LOF has similar F1 as Bi-LSTM, but lower than GARNN-Cloud. Through the above comparison, we can see that the performance of GARNN-Pruned is better than the other models on the real-world data.

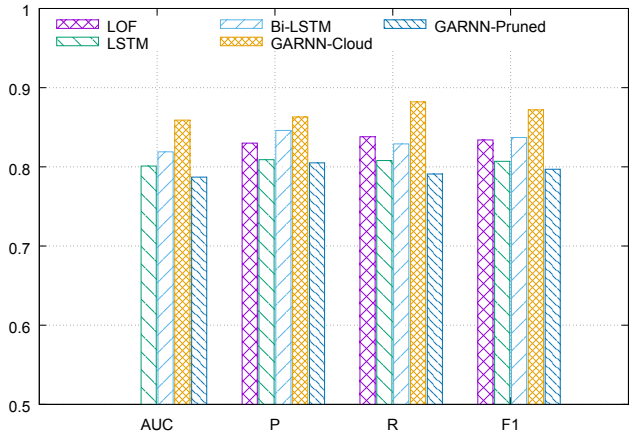

Figure 9: The average AUC, F1, P, and R on real-world data.

Table 8 shows the performance of all the methods on the driving simulator data with three types of abnormal driving behaviors i.e., cognitive distraction, emotional distraction, and sensorimotor distraction. For the cognitive and emotional distraction, GARNN-Cloud achieves the highest performance among all the methods. For the sensorimotor distraction, Bi-LSTM and LSTM have similar performance, outperform GARNN-Cloud. LOF has worse performance than LSTM and Bi-LSTM on all the metrics for cognitive and sensorimotor distractions, but performs much better than LSTM and Bi-LSTM for emotional distraction. Similar to the results on real-world data, after model pruning, GARNN-Pruned performs worse than GARNN-Cloud. Figure 10 shows the average AUC, Precision, Recall, and F1 of all the methods on the driving simulator data over three abnormal driving distractions. As we can see, GARNN-Cloud has the best overall performance on all the metrics.

Comparing GARNN-Cloud and GARNN-Pruned on both real-world data and driving simulator data, we can see that the model pruning process decreases the original performance of the cloud model significantly. The overall performance of GARNN-Cloud is the best among all the methods.

Comparing Figure 9 and Figure 10, we can find that the detection performance of these models on the real-world data is better than the driving simulator data. For the real-world data, it is dangerous and difficult to collect true abnormal driving data, so we simulate four different kinds of abnormal driving behaviors. These artificially simulated anomalies are relatively easier to be detected, comparing with the driving simulator data, where the abnormal driving trajectories are actually from human volunteers, closer to real world driving scenarios.

Table 8: Performance on driving simulator data

	Models	Cognitive	Emotional	Sensorimotor
AUC	LSTM	0.730	0.606	0.786
	Bi-LSTM	0.783	0.650	**0.789**
	GARNN-Cloud	**0.821**	**0.700**	0.720
	GARNN-Pruned	0.785	0.690	0.720
P	LOF	0.705	0.692	0.754
	LSTM	0.740	0.625	0.792
	Bi-LSTM	0.793	0.638	**0.793**
	GARNN-Cloud	**0.823**	**0.690**	0.720
	GARNN-Pruned	0.790	0.655	0.705
R	LOF	0.700	0.690	0.750
	LSTM	0.730	0.645	**0.784**
	Bi-LSTM	0.763	0.682	**0.784**
	GARNN-Cloud	**0.820**	**0.700**	0.740
	GARNN-Pruned	0.780	0.682	0.715
F1	LOF	0.702	0.691	0.752
	LSTM	0.735	0.635	**0.788**
	Bi-LSTM	0.778	0.659	**0.788**
	GARNN-Cloud	**0.821**	**0.695**	0.730
	GARNN-Pruned	0.785	0.668	0.710

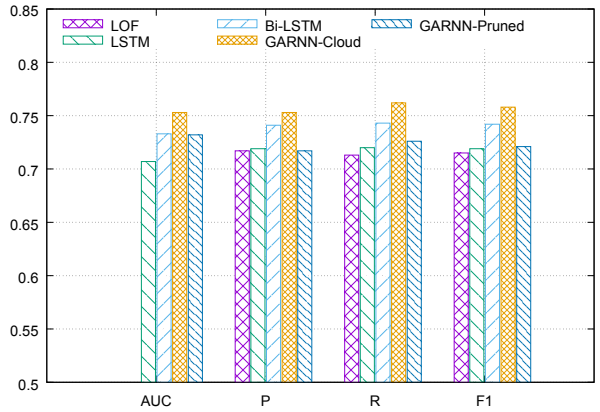

Figure 10: The average AUC, F1, P, and R on driving simulator data.

Table 9: Drivers' information on real-world data

	Age group	Gender	Driving experience
driver *A*	30-40	Male	at least 15 years
dirver *B*	30-40	Male	at least 15 years
driver *C*	40-50	Male	at least 20 years

5.3.3 Personalized Model. When GARNN-Pruned is transferred to the edge devices, we add personalized or contextual feature as conditional input and train a new model on top of GARNN-Pruned, that is, perform transfer learning. We denote the new model as CGARNN-Edge. Specifically, GARNN-Pruned serves as the initialized training model for CGARNN-Edge.

We consider the personalization on two levels: The first level is the drivers' individual difference and the second level is the driving environment or context for the same driver. For the first level, we use the driving data of driver *B* and driver *C* from the real-world data. As shown in Table 9, driver *B* and driver *C* are of different ages and driving experience. They drove along the same trajectory, at the same time of the day, and under the same weather. However, they had different driving behaviors: driver *B* used 1899.3s to finish the drive while driver *C* only used 1575.1s. Driver *C* drove significantly faster than Driver *B*. Therefore, we consider these two datasets having individual difference. For the second level, we consider two conditions: weather and traffic. As shown in Table 3, driver A_2 and driver A_3 are the same driver who drove at the same time, but under different weathers, cloudy and rainy, which may lead to different driving behaviors. Meanwhile, driver A_1 and driver A_3 are the same driver who drove under the same weather, but at different traffic time, 2pm (off-peak) and 5pm (peak). For the driving simulator data, as shown in Table 4, each data is labeled with the drivers' age group and gender, which present their individual difference. However, the driving simulator data does not provide the environment information, such as weather and traffic condition. Therefore, we only consider the first level personalization for the driving simulator.

Table 10: AUC and F1 on real-world data with personalization

	Individual		Weather		Traffic	
Metrics	AUC	F1	AUC	F1	AUC	F1
LOF	NA	0.875	NA	0.871	NA	0.872
LSTM	0.741	0.770	0.729	0.770	0.731	0.77
Bi-LSTM	0.819	0.859	0.803	0.858	0.824	0.829
PGARNN	0.860	0.893	0.848	0.899	0.814	0.826
GARNN-Cloud	0.876	**0.895**	0.845	0.896	0.834	0.894
GARNN-Pruned	0.755	0.765	0.753	0.789	0.753	0.792
CGARNN-Edge	**0.880**	0.893	**0.852**	**0.921**	**0.901**	**0.932**

Table 11: AUC and F1 on driving simulator data with personalization

	Age group		Gender	
Metrics	AUC	F1	AUC	F1
LOF	NA	0.752	NA	0.754
LSTM	0.650	0.680	0.671	0.680
Bi-LSTM	0.700	0.690	0.710	0.696
PGARNN	0.770	0.780	**0.790**	0.790
GARNN-Cloud	0.760	0.760	0.762	0.770
GARNN-Pruned	0.664	0.760	0.640	0.766
CGARNN-Edge	**0.780**	**0.784**	**0.790**	**0.804**

Based on the two level personalization definition, we construct three personalized datasets from the real-world data to evaluate the model performance. These datasets are denoted by: individual, weather, and traffic. For driving simulator data, we construct two

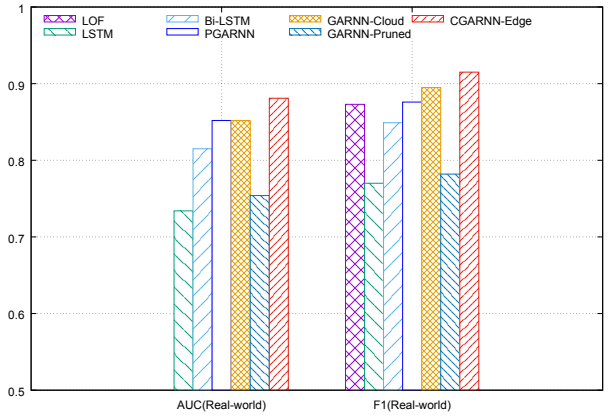

Figure 11: Average AUC and F1 with personalization on real-world data

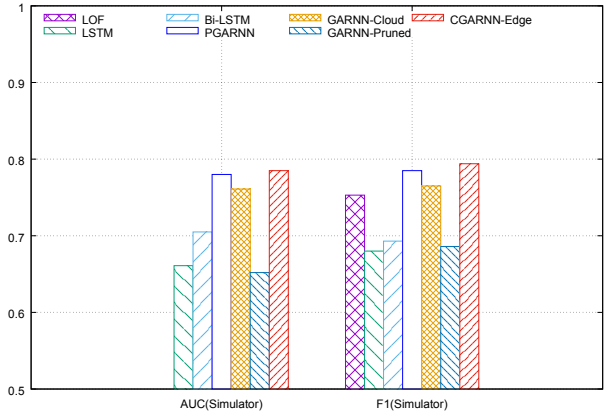

Figure 12: Average AUC and F1 with personalization on driving simulator data

personalized datasets, which are denoted by: age group and gender. For each dataset, we train a CGARNN-Edge model, which is built on top of GARNN-Pruned and takes the personalization information as the condition input. To further compare with CGARNN-Edge, we build a new baseline, PGARNN, which has the same deep learning architecture as GARNN-Edge. PGARNN is trained from scratch on edge devices using personalized data while CGARNN-Edge is trained through transfer learning from GARNN-Pruned. For the real-world data with personalization, as shown in Table 10, GARNN-Edge achieves the highest F1 and AUC on all the datasets, except for F1 on the individual data, which is very close to the best performer GARNN-Cloud (0.895 and 0.893). For the driving simulator data, as shown in Table 11, GARNN-Edge has the best performance among all the methods. Since F1 can be regarded as the harmonic mean of Precision and Recall and due to the space limitation, we only show the results of F1 and AUC. We have a similar conclusion on the Precision and Recall. The results that GARNN-Edge outperforms PGARNN indicate that our proposed transfer learning process improves the model performance. Through transfer learning, GARNN-Edge is also able to learn the patterns from the baseline cloud model, which is trained using all the driving data. On the other hand, PGARNN is trained using only the personal driving data on the edge. Therefore, GARNN-Edge tends to be more robust and adaptive to the potential data distribution change.

We also calculate the average AUC and F1 of all the methods over three personal conditions on real-world data and two personal conditions on driving simulator, data which are summarized in Figure 11 and 12. As we can see, on the real-world data, the AUC and F1 of PGARNN is similar to that of GARNN-Cloud, CGARNN-Edge has the best performance. The AUC and F1 of CGARNN-Edge is 0.881 and 0.915 while the AUC and F1 of BiLSTM is 0.815 and 0.849. On the driving simulator data, PGARNN and GARNN-Edge have better performance than the other models. The performance of GARNN-Pruned suffers due to pruning. CGARNN-Edge achieves the best performance among all the methods, the AUC and F1 is 0.785 and 0.794.

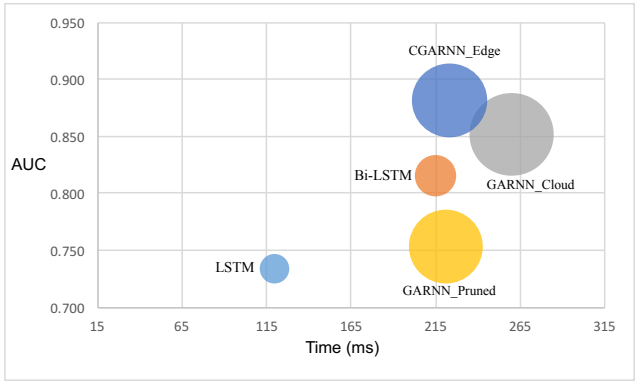

Figure 13: Overall performance of all the methods on real-world data

5.3.4 Performance Summary. Figure 13 integrates the system performance and algorithm performance of all the methods on real-world data. As a core component of ADAS, near real-time detection is one of the most important requirements. Therefore, we choose inference time for the system performance. AUC on real-world data is used for the algorithm performance. The size of bubble shows the model size. As we can see, LSTM has the smallest inference time and model size due to its simple architecture. However, its algorithm performance, i.e., AUC, is also the lowest. On the other hand, CGARNN-Edge has the best AUC. The inference time of CGARNN-Edge is also ranked in the middle, less than 230 ms. GARNN-Cloud has the largest model size and largest inference time, since it has the most number of parameters. We have similar observations on the driving simulator data.

The experimental results from the system side and algorithm side indicate that the proposed system can meet the real-time requirement while achieving promising performance.

6 CONCLUSIONS AND FUTURE WORK

We propose a collaborative cloud-edge computation method for personalized driving behavior modeling. This method models driving behaviors using Generative Adversarial Recurrent Neural Networks, which is adaptive to the dynamic change of normal driving. To address the challenges of personalization and privacy, the proposed collaborative cloud-edge computation method first trains a common baseline model in the cloud and then trains a personalized model on the edge through transfer learning, considering personal and contextual information as additional conditions. Model pruning is applied to minimize the transferring load as well as to maximally preserve the original cloud model performance. The proposed CGARNN-Edge model achieves the best performance among all the models.

In the future, we plan to further investigate how to enhance the collaboration between cloud and edge. For example, it is interesting to study how the collected data and trained model on the edge can improve the common baseline model in the cloud without violating user privacy. As the driving condition can be changed, it is non-trivial to update the personalized driving behavior model dynamically and efficiently. We would like to investigate how incremental learning can be applied to solve such problems in a cloud-edge collaborative framework. In addition, we find that the GARNN-Pruned model usually has a worse performance than the original GARNN-Cloud model. Most recently, more sophisticated pruning methods for deep neural networks have been developed. We would like to explore if such pruning methods can further improve the model performance.

7 ACKNOWLEDGMENTS

This work is supported in part by National Science Foundation (NSF) grant IIS-1724227.

REFERENCES

[1] 2019. Collaborative Learning on the Edges: A Case Study on Connected Vehicles. In *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*. USENIX Association, Renton, WA. https://www.usenix.org/conference/hotedge19/presentation/lu

[2] Reza Abbasi-Asl and Bin Yu. 2017. Structural Compression of Convolutional Neural Networks Based on Greedy Filter Pruning. *arXiv preprint arXiv:1705.07356* (2017).

[3] Benjamin F Bowne, Nicholas R Baker, Duane Lee Marzinzik, Matthew Eric Riley, Nick U Christopulos, Brian Mark Fields, J Lynn Wilson, Bryan T Wilkerson, David W Thurber, et al. 2013. Methods to Determine a Vehicle Insurance Premium Based on Vehicle Operation Data Collected Via a Mobile Device. US Patent App. 13/763,231.

[4] Daniel Ramage Brendan McMahan. 2017. *Federated Learning: Collaborative Machine Learning without Centralized Training Data.* Retrieved Sep 20, 2019 from https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[5] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *ACM Sigmod Record*, Vol. 29. ACM, ACM, New York, NY, USA, 93–104.

[6] Xsens Technologies B.V. 2009. *Xsens Mti-G-700 user manual and technical documentation.* Retrieved Sep 20, 2019 from https://projects.asl.ethz.ch/datasets/lib/exe/fetch.php?media=hardware:tiltinglaser:mti-g_user_manual_and_technical_documentation.pdf

[7] Emily L Denton, Soumith Chintala, Rob Fergus, et al. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*. 1486–1494.

[8] Xiaohan Ding, Guiguang Ding, Jungong Han, and Sheng Tang. 2018. Auto-balanced filter pruning for efficient convolutional neural networks. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

[9] Ziqiang Feng, Shilpa George, Jan Harkes, Padmanabhan Pillai, Roberta Klatzky, and Mahadev Satyanarayanan. 2018. Edge-Based Discovery of Training Data for Machine Learning. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, IEEE, 145–158.

[10] Lex Fridman, Daniel E. Brown, Michael Glazer, et al. 2017. MIT Autonomous Vehicle Technology Study: Large-Scale Deep Learning Based Analysis of Driver Behavior and Interaction with Automation. *CoRR* abs/1711.06976 (2017). arXiv:1711.06976 http://arxiv.org/abs/1711.06976

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*. 2672–2680.

[12] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*. 1135–1143.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[14] Jin-Hyuk Hong, Ben Margines, and Anind K Dey. 2014. A smartphone-based sensing platform to model aggressive driving behaviors. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems (CHI)*. ACM, ACM, New York, NY, USA, 4047–4056.

[15] Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. 2016. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110* (2016).

[16] Sinan Kaplan, Mehmet Amac Guvensan, Ali Gokhan Yavuz, and Yasin Karalurt. 2015. Driver behavior analysis for safe driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* 16, 6 (2015), 3017–3032.

[17] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[18] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer. 2017. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. 204–211.

[19] Liangkai Liu, Xingzhou Zhang, Mu Qiao, and Weisong Shi. 2018. Safeshareride: Edge-based attack detection in ridesharing services. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 17–29.

[20] Chunmei Ma, Xili Dai, Jinqi Zhu, Nianbo Liu, Huazhi Sun, and Ming Liu. 2017. DrivingSense: Dangerous Driving Behavior Identification Based on Smartphone Autocalibration. *Mobile Information Systems* 2017 (2017).

[21] Yunlong Mao, Shanhe Yi, Qun Li, Jinghao Feng, Fengyuan Xu, and Sheng Zhong. 2018. Learning from differentially private neural activations with edge computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, IEEE, 90–102.

[22] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[23] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904* (2016).

[24] Naic. 2018. *Usage-based insurance and telematics, https://www.naic.org.* Retrieved Sep 3, 2018 from https://www.naic.org/cipr_topics/topic_usage_based_insurance.htm

[25] World Health Organization. 2015. *Global status report on road safety 2015.* World Health Organization.

[26] I Pavlidis, M Dcosta, S Taamneh, M Manser, T Ferris, R Wunderlich, E Akleman, and P Tsiamyrtzis. 2016. Dissecting driver behaviors under cognitive, emotional, sensorimotor, and mixed stressors. *Scientific reports* 6 (2016), 25651.

[27] Xukan Ran, Haoliang Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. 2018. DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics. In *IEEE International Conference on Computer Communications (INFOCOM)*.

[28] Protection Regulation. 2016. General data protection regulation. *Official Journal of the European Union* 59 (2016), 1–88.

[29] Mahadev Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017), 30–39.

[30] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.

[31] Surat Teerapittayanon, Bradley McDanel, and HT Kung. 2017. Distributed deep neural networks over the cloud, the edge and end devices. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 328–339.

[32] Pengyang Wang, Yanjie Fu, Jiawei Zhang, Pengfei Wang, Yu Zheng, and Charu Aggarwal. 2018. You are how you drive: Peer and temporal-aware representation learning for driving behavior analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, ACM, New York, NY, USA, 2457–2466.

[33] Yifan Wang, Shaoshan Liu, Xiaopei Wu, and Weisong Shi. 2018. CAVBench: A Benchmark Suite for Connected and Autonomous Vehicles. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, IEEE, 30–42.

[34] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 2074–2082.

[35] Xsens. 2018. *Xsens North America Inc, https://www.xsens.com/products/mti-g-710/.* Retrieved Jun 4, 2018 from https://www.xsens.com/products/mti-g-710/

[36] Chuang-Wen You, Nicholas D Lane, Fanglin Chen, Rui Wang, Zhenyu Chen, Thomas J Bao, Martha Montes-de Oca, Yuting Cheng, Mu Lin, Lorenzo Torresani, et al. 2013. Carsafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services.* ACM, ACM, New York, NY, USA, 13–26.

[37] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, ACM, New York, NY, USA, 316–324.

[38] Qingyang Zhang, Yifan Wang, Xingzhou Zhang, Liangkai Liu, Xiaopei Wu, Weisong Shi, and Hong Zhong. 2018. OpenVDAP: An Open Vehicular Data Analytics Platform for CAVs. In *Distributed Computing Systems (ICDCS), 2018 IEEE 38th International Conference on. IEEE.* IEEE, 1310–1320.

[39] Xingzhou Zhang, Yifan Wang, and Weisong Shi. 2018. pCAMP: Performance Comparison of Machine Learning Packages on the Edges. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18).* USENIX Association, Boston, MA. https://www.usenix.org/conference/hotedge18/presentation/zhang

[40] Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878* (2017).