DarNet: A Deep Learning Solution for Distracted Driving Detection

Christopher Streiffer
Duke University
Durham, NC 27705, USA

Ramya Raghavendra
IBM Research
Yorktown Heights, NY 10598, USA

Theophilus Benson
Brown University
Providence, RI 02912, USA

Mudhakar Srivatsa
IBM Research
Yorktown Heights, NY 10598, USA

Abstract

Distracted driving is known to be the leading cause of motor vehicle accidents. With the increase in the number of IoT devices available within vehicles, there exists an abundance of data for monitoring driver behavior. However, designing a system around this goal presents two key challenges - how to concurrently collect data spanning multiple IoT devices, and how to jointly analyze this multimodal input. To that end, we present a unified data collection and analysis framework, *DarNet*, capable of detecting and classifying distracted driving behavior. *DarNet* consists of two primary components: a data collection system and an analytics engine. Our system takes advantage of advances in machine learning (ML) to classify driving behavior based on input sensor data. In our system implementation, we collect image data from an inward facing camera, and Inertial Measurement Unit (IMU) data from a mobile device, both located within the vehicle. Using deep learning techniques, we show that *DarNet* achieves a Top-1 classification percentage of 87.02% on our collected dataset, significantly outperforming our baseline model of 73.88%. Additionally, we address the privacy concerns associated with collecting image data by presenting an alternative framework designed to operate on down-sampled data which produces a Top-1 classification percentage of 80.00%.

ACM Reference format:
Christopher Streiffer, Ramya Raghavendra, Theophilus Benson, and Mudhakar Srivatsa. 2017. DarNet: A Deep Learning Solution for Distracted Driving Detection. In *Proceedings of Middleware Industry '17: Proceedings of the Industrial Track of the 18th International Middleware Conference, Las Vegas, NV, USA, December 11–15, 2017 (Middleware Industry '17),* 7 pages.
DOI: 10.1145/3154448.3154452

1 Introduction

Distracted driving, wherein a driver is engaged in another activity that takes their attention away from operating the vehicle, is identified to be a significant cause of fatal motor vehicle crashes and injuries. According to a 2014 study conducted by the National Highway Traffic Safety Administration (NHTSA), 3,179 drivers were killed and 431,000 drivers were injured in accidents involving distracted driving[12]. The most typical distracted driving behaviors include but are not limited to cell phone use, talking to passengers, grooming, and adjusting the radio and navigation systems [14]. These behaviors require both cognitive and visual attention and increase the chance of an accident by nearly a factor of 3, according to a 2013 report conducted by the NHTSA [19].

Given the severity of the problem, detecting distracted driving has received significant attention from the research community, industry and government agencies [1, 3, 4]. Commercial solutions that collect onboard data e.g. fuel consumption and emergency alerts, and incentivize good driving are currently being offered [8, 9]. There have been research efforts to detect cellphone usage based on acoustic or cellphone sensing [16, 34, 37], alcohol impairment using supervised and semi-supervised techniques [27, 28], and drowsy driving by tracking driver eye gaze [29]. Our system advances prior work by creating a mobile middleware system comprising a data collection framework which in turn relies on an inbuilt ensemble of deep learning techniques to enable us to detect a richer set of distracted driver classes and to better disambiguate between these classes by combining various sensing modalities.

We take a combined *Internet of Things* (IoT) and *deep learning* based approach to counter the problem of distracted driving. This is made possible by the significant proliferation of IoT devices e.g. embedded mobile sensors, that are capable of collecting a variety of information ranging from vehicle speed, acceleration and braking, to driver's heart rate and dashboard images. Each of these inputs provides richer contextual information that allows machine learning models to better disambiguate human behavior and enable fine-grained analysis. For instance, the image of a driver sending a text message can be cross-validated by checking the acceleration of the mobile device from the embedded accelerometer. Further, applications such as Waze [13] turn off usage in driving mode regardless of who is using the phone, whereas image analysis can discern whether it is actually the driver or passenger using the device. Such instances motivate us to explore multimodal analysis to improve classification accuracy without deploying expensive sensors. Further, with the advancements in Big Data and improvements in computational power, analyzing IoT data using deep learning allows for more powerful and accurate methods for performing this analysis.

Detecting distraction is critical for retroactively analyzing accident scenes, providing variable insurance rates, and providing real-time alerts to drivers and fleet managers. However, building such a system requires overcoming practical challenges to realize a working solution. First, *how* do we extract data being generated by multiple sources and at varying time scales such that they can be meaningfully analyzed? Second, *where* should the processing be performed - at the edge, remote server, or split between the two.

Middleware Industry '17, Las Vegas, NV, USA

DOI: 10.1145/3154448.3154452

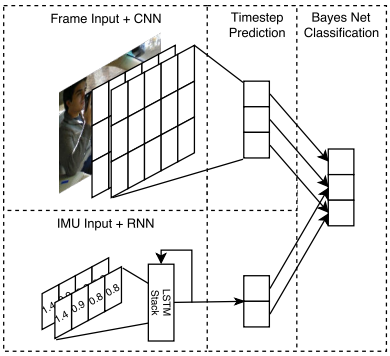


Figure 1. Diagram depicting the classification system. The probability distributions of the two networks are combined using a bayesian network to produce probabilities for each class, and the overall classification.

To make this decision, the system must have a sense of processing capability, network bandwidth and latency. Finally, the system must provide certain *privacy* guarantees to the user when collecting IoT data, without sacrificing classification accuracy.

To this end, we present *DarNet*, a multi-modal data collection and analysis system designed to detect and classify distracted driving behavior. The system is composed of a data collection framework which collects streaming input from multiple sensors and an analytics engine responsible for combining the different modalities into a cogent output. The data collection framework can be further decomposed into *data collection agents* which run on each IoT device and a *centralized controller* designed to communicate with the agents. The *analytics engine* utilizes advances in deep learning to process the data received from the controller.

DarNet operates on two different data modalities - IMU data collected from the driver's mobile device and image data received from a dashboard camera mounted to face the driver. The system is currently configured to operate in a remote configuration where all data is shipped to a remote server for classification. In the remote configuration, the user has the option to down-sample the image data which serves as a basis for privacy, while also saving network bandwidth and diminishing latency.

The analytics engine builds on two advanced deep learning techniques for spatial and temporal data analysis, specifically Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). The system employs a CNN for processing the image data received from the camera and a RNN for processing the sensor data (IMU) received from the mobile device. An ensemble-based learning approach is employed to strategically combine the output of the two models to produce the overall classification. Our system is designed to make classifications at each time-step from the data, making it amenable to near real-time detection.

Placing a camera inside a car imposes major privacy concerns that might deter individuals from using the application, despite a surge in their popularity [1]. To mitigate these concerns, we explore the feasibility and tradeoffs involved with minimizing the risk of potentially compromising data leaving the vehicle. We implement a methodology that allows for a distortion filter to be applied to the video data before processing. This design allows for a user to specify the level of privacy associated with their data, with the tradeoff of altering the classifications.

We make the following contributions in this paper:

- We build a data collection framework to collect and align data across multiple IoT devices and modalities. This is open sourced and can be useful for quickly collecting, aggregating and labeling data for data processing tasks.
- We demonstrate how to use deep learning across different IoT modalities - CNN for video analysis, and RNN for IMU time-series analysis. Further, we explored several techniques to *combine* these modalities and to the best of our knowledge, we present a novel Bayesian Network combiner approach to strengthen driver distraction detection.
- Because we are working with sensitive data, we take the first steps towards developing an unsupervised learning methodology for adding different levels of privacy to the system.

We believe this is an initial step towards developing middleware solutions to processing IoT sensor data in a scalable, useful and private manner. By making the software and learning models available to the general research community, we hope to facilitate future work in this field of important and emerging technology.

2 Related Work

Detecting distracted driving is receiving significant amount of attention from the research community and the industry. There is a wealth of data that has been has been made available to the research community to study this problem [1, 3, 4].

There have been efforts that recognize driver distraction using data extracted from wearables, cellphone and onboard diagnostics to obtain sensing information such as accelerometer, gyroscope, etc. to model phone usage while driving. Additionally, using this data to collect unsafe lane changes [16, 28, 34, 37] is an active area of research. A detailed review that discusses various driver condition monitoring techniques can be found in recent survey papers [5, 17].

By integrating powerful sensors into top-end cars (e.g., cameras, radar, and ultrasonic sensors), manufacturers are bringing similar forms of driver behavior monitoring to the consumer market [2, 6, 11]. While safety technology is getting cheaper, few safety features are available in entry-level vehicles. In contrast, smartphone solutions such as *DarNet* can be used with little additional expense in all cars today. Commercial solutions such as Automatic [9] and LDWS [7] provide tracking and emergency services using OBD and GPS data but require special hardware. Progressive Insurance offers a service, Snapshot [8] that uses OBD-II data from a mounted device to detect driving behavior. To the best of our knowledge, there is no commercial solution that combines video and sensor data to analyze driving behavior.

Recognizing distracted driving behavior (i.e., driver drowsiness, lane departure, and following distance) using fixed vehicle-mounted devices is an active area of research. Computer Vision techniques and learning-based methods [27, 36, 38] are used to extract features corresponding to eye-tracking, driver postural stability, cellphone usage etc. out of driving images or video. Our work builds on prior work by combining the use of the powerful Convolutional Neural Networks (CNN) [26, 32, 33] on image data with the use of a Recurrent Neural Network (RNN) for temporal data. These networks have been shown to be effective for analyzing sequential information specially for NLP tasks, and are recently gaining popularity in other

[1] http://www.ceoutlook.com/2016/08/10/the-strange-emerging-dvr-market/

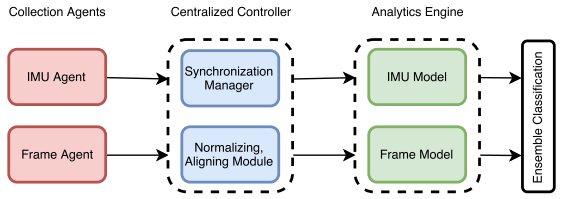


Figure 2. Diagram depicting the system. The collection agents run embedded within IoT devices. The controller collects and prepares the data for the analytics engine which performs the ensemble classification.

applications [18, 21, 23, 30]. To the best of our knowledge, our work is the first to use an ensemble neural network to combine video and sensor data to produce accurate classifications for distracted driving. We believe applying deep neural network ensembles on temporal and spatial data to distracted driver detection problem opens up several exciting possibilities, enabling us to easily learn non-linear relationships between sensors and driver behavior in a scalable fashion (e.g., centripetal acceleration to detect phone usage).

Next, *we build a unified neural network framework to analyze multimodal sensing data* obtained from dashboard videos and IMU sensors, and without the need for adding expensive sensors (such as hi-res cameras). Combining modalities helps us easily identify contexts that a single modality does not provide e.g. *is the driver or passenger using the phone?*. Multimodal deep learning approach has received recent attention [31] for being able to identify nonintuitive features largely from cross-sensor correlations which can result in a more accurate estimation, with prior work taking this approach to detect human activity recognition through sensor data from wearable devices.

Recent work by Kim et.al. looks at the issue of sharing of private dashcam videos under anonymity [25]. Privacy of driving data, even before it leaves the car, is a critical issue going forward and one that has not received much attention. We take a step towards anonymizing *dashcam video data* while still ensuring classification can be performed. We explore heterogeneous learning architectures wherein we train a CNN to reproduce the feature activations learned by the deep model. We expand upon this approach by training a deep CNN to reconstruct features from down sampled images using a similar error minimization methodology. The problem of multimodal data anonymity will be the subject of our future work.

3 System Design

DarNet aims to tackle the task of detecting distracted driving by providing a well connected framework for performing data collection and analysis. Integral to the design of *DarNet* is the proliferation of available IoT sensor devices. We therefore design our system to allow for new devices to be easily incorporated into the existing framework. The more data available for processing and training, the better the overall performance of the system.

3.1 Collection Agents

The *collection agent* is designed as an abstraction to allow for easy-integration into a broad range of IoT devices. The responsibilities of the agent include periodically polling the device's sensor, maintaining an internal clock for timestamping the data, and transmitting the data to the centralized controller at a specified frequency. The periodicity at which the agent polls the sensor should be determined based on the operating frequency of the sensor itself, while the transmission frequency should be determined based on the latency and bandwidth between the agent and the controller. The implementation of each agent is specific to the system and sensors in which it is embedded.

3.2 Centralized Controller

The *centralized controller* has several responsibilities which include: aggregating, smoothing, and aligning data received from the agents, maintaining clock synchronization between the agents, and deciding how and where the data should be processed by the analytics engine.

Data Normalization: The controller receives periodic updates from each agent at different frequencies based on the agent's configuration, network latency, and the order in which the operating system interleaves the received network packets. As such, the agent relies on the timestamp associated with each tuple to determine the ordering of the collected data. Because the timestamps for data received from different agents will not align exactly, the controller uses interpolation to fill in the gaps, and to aggregate the data at consistent intervals. Additionally, the controller performs a smoothing operation on the data by maintaining a sliding moving average. Because the system runs on commodity sensor hardware, we expect the data measurements to fall within a bounded range of error, therefore the smoothing average normalizes the data against any aberrations or irregularities.

Clock Synchronization: We have implemented a protocol between the agents and controller to maintain clock synchronization. In this protocol, the controller maintains a *system time* and pushes this time to each agent. Whenever an agent receives an updated system time, the agent will update its own clock to reflect that of the controller's, plus an additional constant to account for network latency. This protocol is set to run periodically in order to account for internal clock drift of each agent.

Processing Decision: In determining where the data should be processed, the controller can choose between a local and remote configuration. A remote server would have a greater amount of processing power, and hence is a computationally better option over the local configuration that runs on the device. However, under poor network conditions, the controller has the option of processing all data locally, albeit slower. When remote configuration is selected, the user has the option of specifying the *degree of privacy* at which the image data is transmitted to the analytics engine. Privacy is added to the system by down-sampling the image data to varying sizes. The advantage of this technique is that it not only obfuscates the user's identifying information, but also improves bandwidth by transmitting less data.

3.3 Analytics Engine

The *analytics engine* is responsible for processing the data streams and producing a classification. The engine is designed to be entirely modular – the system maintains a 1-to-1 relationship between device data-streams and machine learning models. Rather than training a single model to process the data, we train individual models and combine the results at a later stage. This design provides two benefits: 1. New devices can be incorporated into the network without requiring the existing models to be retrained. 2.

Each machine learning model can be specified based on the type of data streamed from the device i.e. spatial models for image data, temporal models for IMU data. The engine combines the results of each model to produce the overall classification.

4 System Implementation

We have designed and deployed an implementation of this system which collects data from an inward facing camera and mobile device. *DarNet* collects and processes the data from these devices to perform driver behavior classification.

4.1 Data Streaming Framework

We have developed two Android modules — i) the *collection agent* which interfaces with IMU data sensors and the camera hardware, and ii) the *centralized controller* which aggregates data from the agents, maintains clock synchronization, and formats the data for processing. We have made the source code for these modules available to the public [2].

Data Collection Agent: In our setup, we use a Nexus 7 tablet running Android 6.0.1 (API v23) to emulate the dashcam, and a Nexus S device running Android 4.1.2 (API v16) acting as the driver's cellphone. Each *collection agent* operates by registering *listeners* to receive updates from the device's embedded sensors. Each listener receives updates every 25 ms through Android's *sensor manager*.

The agent running on the Tablet is responsible for interfacing with the embedded camera, while the agent running on the mobile device interfaces with the *accelerometer, gyroscope, gravity, and rotation sensors*. Each agent is designed to communicate with the controller over Bluetooth or 802.11 point to point wireless.

Centralized Controller: The controller runs on the Nexus 7 tablet. Upon startup, the controller will open a two-way communication channel with each collection agent registered by the system. Since we are collecting data with multiple modalities, synchronizing the timestamps across the devices is important for further analysis. We implement a *timestamp manager* in our application with master-slave architecture for clock distribution with the controller acting as the master and distributing its UTC timestamp to the agents over their communication channel. The agent sets its own clock to the master's UTC, plus the empirically measured network delay. Because the system clock is highly susceptible to drift, this synchronization process is repeated every 5 seconds.

We have implemented *DarNet* to perform all data processing on a remote server, in both the normal and privacy-preserving settings. When the controller receives an update from an agent, it will perform the aforementioned aligning and normalization techniques, and store the data in a time-series database e.g. statsd. The data is transferred and processed in an offline manner.

4.2 Analytics Engine

Within our system, we make use of the advances in deep learning to perform our distracted driver analysis. We utilize two different types of neural networks – a Convolutional Neural Network (CNN) for performing classification on a per-frame basis, and a Recurrent Neural Network (RNN) for classifying the time-series data. CNNs have proven to perform exceptionally well at image classification tasks, while RNNs excel at analyzing time-series data by discovering non-linear dependencies which are not visible to other learning methodologies.

Frame-Sequence Architecture: We use the Inception-V3 [33] architecture for performing frame classification. We made the decision to use this model because of its exceptional performance on the ILSVRC 2012 classification task [33]. The design of the model exploits the "hebbian principle" [15] which proposes that spatially close neurons should activate together, and is a refinement over GoogleNet [32], focusing on maximizing computation efficiency while allowing for additional expansion of layers. Due to the large amount of time required for training deep networks, we take a fine-tuning approach by initializing our model using the weights of a pre-trained model. We use the model made available through the Tensorflow source code which has been trained to a 78.7% Top-1 classification percentage on the ILSVRC 2012 [10] dataset as our initialization point. We modify the final fully connected layer of this network, such that the number of outputs corresponds to the number of driving classes.

IMU-Sequence Architecture: We use a deep bidirectional LSTM network for performing the IMU-sequence classification [22, 39]. The bidirectional aspect of the model refers to each LSTM cell propagating its output forward and backward through time. The deep aspect refers to the use of multiple LSTM cells, where the output of one is used as input to the next, before the final softmax classification layer. We made the decision to use this type of network over support vector machines because of the RNNs ability to capture long-term dependencies between input features while diminishing the vanishing gradient problem inherent to these networks. The architecture for the RNN consists of 2 bidirectional LSTM cells containing 64 hidden units. Because we use a sampling frequency of 4Hz and a time window of 5 seconds, the network is trained and evaluated on a sliding window of 20 data points.

Ensemble Learning: Because the RNN and CNN output probability distributions for a different set of classes, we implement a Bayesian Network (BN) [20] to combine the outputs into a single inference. Each class is assigned its own BN consisting of two parent nodes and a child node. We compute the conditional probability tables (CPTs) for each class based on the number of true-positive observations from the training data presented to the system.

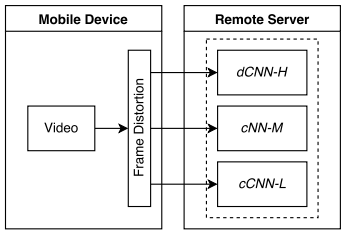


Figure 3. The privacy implementation for the system. The paths between the mobile device and server correspond to the different levels of downsampling – 100x100, 50x50, 25x25.

4.3 Privacy Preserving Analytics

In Figure 3, we present the privacy design for processing the down-sampled video data. This novel architecture introduces two unique components: the distortion module which down-samples the video data, and a set of deep learning models for performing feature extraction. We design this portion of the system to be minimally

[2] https://github.com/cstreiffer/sensordataclient
https://github.com/cstreiffer/driverdatarecorder

Figure 4. From right to left: the undistorted image; the image distorted to the size of 100x100 pixels; the image distorted to the size of 50x50 pixels; the image distorted to the size of 25x25 pixels.

invasive — each model is trained using unsupervised techniques which do not require any knowledge of the original dataset.

System Implementation: In our design, the distortion module down samples the video according to user-specified preference and tags the video with the down-sampling rate. At the remote server, the analytics engine picks the appropriate classifier for performing feature extraction on the distorted video. In *DarNet*, we provide three levels of distortion – low, medium, and high, thus explaining the different paths to the remote server.

The image data is distorted using nearest neighbor down sampling to the sizes of 100x100 (dCNN-L), 50x50 (dCNN-M), and 25x25 (dCNN-H) pixels. Being able to reduce the image from 300x300 pixels these sizes represents approximately a 9x, 25x, and 144x decrease in amount of data required for transmission. This is ideal for situations where a user desires a high amount of privacy, or where network conditions are poor and lower resolution frames must be transmitted to the remote server. Examples of the down-sampled frames can be observed in Figure 4.

Training Methodology: The motivation behind the training methodology stems from the success exhibited by de-noising auto-encoders [24, 35]. In our work, we train an additional CNN to reproduce the classifications results of the original model. We reuse the Inception-V3 architecture and initialize the weights using the CNN trained on the driving dataset. Because the goal of the training is for the dCNN model to mimic the behavior of the CNN model, we believe that this initialization methodology provides a good starting point for the weights. This technique allows for the training to be completely unsupervised, and allows for new data to be incorporated into the system.

The methodology is described as follows: 1. The image is passed through the original CNN and the output of the final layer is recorded. Note: In this design, the original image does not have to leave the device to perform this action. 2. The image is down-sampled and sent to the remote server with the appropriately marked tag. 3. The server aggregates the data, the distortion level, and the prediction results observed from passing the image through the original CNN. 4. Finally, the dCNN model is trained by passing the down-sampled image through the model, and comparing the observed output against the recorded output. The loss-function computes the L2 euclidean distance between these two vectors, and the model is trained using stochastic gradient descent as the optimization technique. This has the effect of training the dCNN model to perform a de-noising operation on the image to produce accurate predictions.

5 Evaluation

We evaluate our framework by running *DarNet* on a dataset collected using the data collection component. In our evaluation, we look to answer the question *how can different modalities be incorporated together to strengthen classifications?* We present results showing how an ensemble approach produces stronger classifications than a single modality, and how deep learning can strengthen these results. Additionally, we present results showing the efficacy of *DarNet* to operate on the down-sampled frames in our privacy configurations.

Class	Description	Data Types	Frame Count
1	Normal Driving	Image, IMU	5,286
2	Talking	Image, IMU	10,352
3	Texting	Image, IMU	9,422
4	Eating/Drinking	Image, —	9,463
5	Hair and Makeup	Image, —	4,848
6	Reaching	Image, —	17,709

Table 1. Driver behavior classes collecting using the data collection component. The Data Types corresponds to the type of data that was collected for the corresponding behavior. Classes 4, 5, and 6 do not require cellphone use and thus are considered as "Normal Driving" for the IMU sequence data. The frame count corresponds to the number of data points collected for that category.

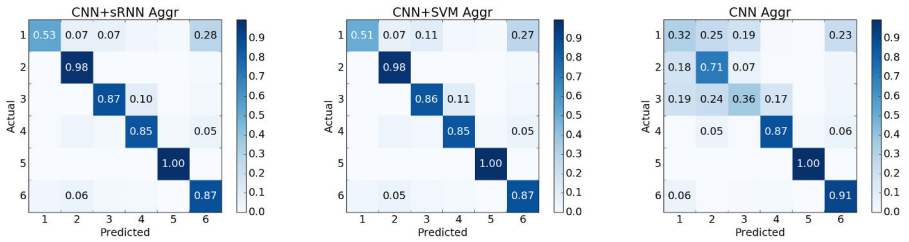
5.1 Evaluation Setup

Data Collection: To perform our analysis, we collect datasets from 5 drivers performing the driving tasks listed in Table 1. Each driver was given the same route to drive, used the same vehicle, and drove under varying degrees of lighting. Each driver was instructed (by the passenger, in real time) to perform a scripted set of "distractions" for a duration of 15 seconds and the entire script was repeated 10 times for each driver. Each video was verified at a later point in time for the accuracy of the given driving test. We divide the collected dataset into an 80/20 partition for running the training and evaluation on the ML models.

The complete list of distraction classes we consider and the corresponding number of collected data points are outlined in Table 1. The data type column refers to the types of data we were able to collect for the given task. For instance, for the eating/drinking category, while we were able to collect image data, the mobile device was placed in the "Normal Driving" position, and thus, we were not able to collect any IMU sequence data specific to the distraction task. The classes are reflective of the distraction classes recognized by Statefarm insurance company [3].

In this paper, we take the first step towards demonstrating the strength of our classifier by positioning the client mobile device in one of five varying orientations during data collection. The Texting orientation corresponds to the driver holding the phone between waist and eye level in either the left or right hand, while the Talking orientation corresponds to the driver holding the phone to either ear. The final orientation corresponds to all remaining classes and consists of the device being positioned horizontally in the drivers front-right pocket.

[3] www.kaggle.com/c/statefarmdistracteddriverdetection#evaluation



(a) (b) (c)

Figure 5. Confusion matrices for the collected datasets for architectures: (a) CNN+RNN (*DarNet*) (b) CNN+SVM (c) CNN (frame data only).

Model	Hit@1
CNN+RNN	87.02%
CNN+SVM	86.23%
CNN	73.88%

Table 2. Ensemble model Top-1 classification results for the collected dataset.

5.2 Evaluation Results

We evaluate the performance of *DarNet* against two additional architectures. In the first architecture, we only consider the CNN portion of the system operating on the frame data. In the second architecture, we combine the CNN frame architecture with a support vector machine (SVM) trained to classify the IMU sequence data. The Top-1 classification results for the three architectures can be found in Table 2, while the resulting confusion matrices can be observed in Figure 5.

The results show that the top performing model is the selected architecture for *DarNet* which consists of the CNN for frame classification and RNN for IMU sequence classification. While each method produces strong classifications results, Figure 5 shows that all three models output a high number of false positives when predicting normal driving. From a safety perspective, classifying normal driving as distracted driving is acceptable in that the overall safety of the driver is not diminished by the incorrect classification. However, from a usability perspective, a high false positive rate for distracted driving would diminish the user experience. Because of the small size of the dataset, we believe that these false positives can be reduced by training across data collected from a larger participant study.

Out of the three architectures considered, both ensemble approaches (CNN+RNN, CNN+SVM) significantly outperform the single modality approach (CNN) with Top-1 classification percentages of 87.02% and 86.23% compared to 73.88%. These results show that adding the IMU sequence data to the frame data has the effect of improving the accuracy of the system by >16.7%.

For the per-class results, the biggest improvement when using multiple modalities is the system's ability to discern between texting, talking, and normal driving. For instance, the CNN model produces a classification accuracy of 36.0% for texting, whereas the CNN+RNN produces an accuracy of 87.0%. Similar improvements can be observed for normal driving and talking. Overall, the ensemble approach is able to eliminate the majority of the noise between these three groups as observed in Figure 5c. Among the classes which did not have corresponding IMU sequence data, there was an observable drop in accuracy for eating/drinking and reaching, of 2% and 4% respectively. Whereas the CNN model misclassifies reaching as normal driving, both ensemble models produce a higher talking misclassification rate for this category of ~5%. An explanation for this result is that the movement that occurs when reaching for an object adds enough noise to the IMU data to produce a talking classification. The complete confusion matrix for each class and model can be observed in Figure 5.

Between the ensemble models, using the RNN architecture over the SVM corresponds to a ~1% increase in performance. When considering the amount of time an individual spends in their vehicle during the day, a 1% increase in the amount of accurately classified driving behaviors could be the difference in deterring an accident. When running the RNN and SVM models on the IMU sequence dataset alone, the RNN produces a classification accuracy of 97.44% compared to 95.37% for the SVM. Because the RNN architecture better captures non-linear dependencies between the data, we believe that the CNN+RNN architecture is best suited for this system.

5.3 Denoising-CNN Results

The dCNNs were trained and evaluated on a previously collected distracted driver dataset. The dataset consists of 18 classes, and was collected from a total of 10 drivers. All video was recorded using a GoPro Hero 3 camera at a rate of 30 fps.

Model	Hit@1
CNN	78.87%
dCNN-L	80.00%
dCNN-M	77.78%
dCNN-H	63.13%

Table 3. CNN and dCNN Top-1 classification percentages for the alternative distracted driving dataset.

As can be viewed in Table 3, the most surprising result of the evaluation was the dCNN-L outperforming the other networks by achieving a Top-1 classification percentage of 80.00%. In comparison, the baseline CNN achieved a Top-1 classification percentage of 78.87%.

While the dCNN-L performed the best, the most promising results from the evaluation were the performance of the dCNN-M and dCNN-L models. As can be viewed in Table 3, the dCNN-M achieved a Top-1 classification percentage within <1% of the baseline CNN, while the dCNN-H was able to produce the moderately-accurate Top-1 classification results of 63.13%. As can be viewed in Figure 4, the distortion levels for dCNN-M and dCNN-H render the image almost unidentifiable. Future work is still required to determine how effective these distortion techniques are for preventing adversarial networks from performing classification tasks e.g. facial recognition.

Why does the dCNN-L outperform the baseline CNN? The biggest anomaly in our results was the improved performance of

the dCNN-L over the baseline CNN. A potential explanation for this phenomenon is that the baseline CNN displays effects of overfitting accrued during training. The loss of information from the image data mitigates these effects, to a degree. As can be observed by the results of dCNN-H, too much loss of information results in significantly diminished performance, leading us to suspect that there is a local distortion maxima that results in the optimal performance. Although we are fascinated by these results, we leave further investigation for future work.

6 Conclusion

Driver distraction is identified as a significant contributor to fatal motor vehicle crashes and injuries. At the same time, there has been a proliferation of IoT-enabled sensors in the car that provide real-time information on the driver's performance and road conditions. In this work, we take the attempt at combining the multiple sensor modalities, and apply deep learning techniques to detect driver distraction thus providing a richer set of classifications and improving accuracy over previous approaches. To this end, we build DarNet, a combined convolutional and recurrent neural network that can analyze driving image and IMU sensor data to learn to detect up to 6 classes of driving behaviors with increased accuracy. We also enable feature extraction at the edge, enabling preserving driver privacy. While our ensemble learning approach is extensible to adding more modalities, we believe that building such systems in the light of privacy concerns and edge device constraints provides a rich area for future research.

7 Acknowledgements

Research was sponsored by US Army Research laboratory and the UK Ministry of Defense and was accomplished under Agreement Number W911NF-15-R-0003.

References

[1] 1996. VIRGINIA TECH TRANSPORTATION INSTITUTE. http://www.vtti.vt.edu/. (1996).
[2] 2008. Mercedes E-Class No Doze. http://autoweek.com/article/car-news/no-doze-mercedes-e-class-alerts-drowsy-drivers. (2008).
[3] 2008. Official US Government Website for Distracted Driving. https://www.fhwa.dot.gov/goshrp2/About. (2008).
[4] 2008. SHRP2 Solutions. https://www.fhwa.dot.gov/goshrp2/About. (2008).
[5] 2012. *An Evaluation of Emerging Driver Fatigue Detection Measures and Technologies.* Technical Report. Technical Report, US Department of Transportation.
[6] 2012. Bosch Drowsy Driver Alert System. http://www.dougnewcomb.com/2012/05/07/bosch-debuts-low-cost-drowsy-driver-alert-system. (2012).
[7] 2012. LDWS. "http://www.mazda.com/en/innovation/technology/safety/active_safety/ldws/". (2012).
[8] 2014. Progressive snapshot. https://www.progressive.com/auto/snapshot/. (2014).
[9] 2015. Automatic. https://www.automatic.com/. (2015).
[10] 2015. Inception-V3 Source Code. "https://github.com/tensorflow/models/tree/master/inception". (2015).
[11] 2015. Volvo. https://www.cars.com/articles/volvo-car-seat-concept-could-cut-distracted-driving-by-parents-1420682901604. (2015).
[12] 2016. Research Note: Distracted Driving 2014. (Apr 2016). https://crashstats.nhtsa.dot.gov/api/public/viewpublication/812260
[13] 2016. Waze. http://www.waze.com. (2016).
[14] 2104. Distracted Driving. http://www.distraction.gov/stats-research-laws/facts-and-statistics.html. (2104).
[15] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. 2014. Provable Bounds for Learning Some Deep Representations.. In *ICML*. 584–592.
[16] Cheng Bo, Xuesi Jian, Xiang-Yang Li, Xufei Mao, Yu Wang, and Fan Li. 2013. You'Re Driving and Texting: Detecting Drivers Using Personal Smart Phones by Leveraging Inertial Sensors. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking*.
[17] J. Culp, M El Gindy, and A Haque. 2008. Driver Alertness Monitoring Techniques: A Literature Review. *International Journal of Heavy Vehicle Systems* 15, 2/3/4 (Sept 2008).
[18] Li Deng and John C Platt. 2014. Ensemble deep learning for speech recognition.. In *INTERSPEECH*. 1915–1919.
[19] Gregory M Fitch, Susan A Soccolich, Feng Guo, Julie McClafferty, Youjia Fang, and others. 2013. *The impact of hand-held and hands-free cell phone use on driving performance and safety-critical event risk.* Technical Report.
[20] Nir Friedman, Dan Geiger, and Moises Goldszmidt. 1997. Bayesian Network Classifiers. *Mach. Learn.* 29, 2-3 (Nov. 1997), 131–163.
[21] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. *CoRR* abs/1303.5778 (2013).
[22] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 6645–6649.
[23] Michiel Hermans and Benjamin Schrauwen. 2013. Training and Analysing Deep Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). 190–198.
[24] Viren Jain and Sebastian Seung. 2009. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems*. 769–776.
[25] Minho Kim, Jaemin Lim, Hyunwoo Yu, Kiyeon Kim, Younghoon Kim, and Suk-Bok Lee. 2017. ViewMap: Sharing Private In-Vehicle Dashcam Videos. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA.
[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). 1097–1105.
[27] John Lee, Timothy Brown, Dary Fiorentino, James Fell, Eric Traube, and Eric Nadler. 2011. Using Vehicle-Based Sensors of Driver Behavior to Detect Alcohol Impairment. In *22nd International Technical Conference on the Enhanced Safety of Vehicles (ESV)*.
[28] Luyang Liu, Cagdas Karatas, Hongyu Li, Sheng Tan, Marco Gruteser, Jie Yang, Yingying Chen, and Richard P. Martin. 2015. Toward Detection of Unsafe Driving with Wearables. In *Proceedings of the 2015 Workshop on Wearable Systems and Applications*.
[29] Tianchi Liu, Yan Yang, Guang-Bin Huang, and Zhiping Lin. 2015. *Detection of Drivers' Distraction Using Semi-Supervised Extreme Learning Machine.*
[30] Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to Construct Deep Recurrent Neural Networks. *CoRR* abs/1312.6026 (2013).
[31] Valentin Radu, Nicholas D. Lane, Sourav Bhattacharya, Cecilia Mascolo, Mahesh K. Marina, and Fahim Kawsar. 2016. Towards Multimodal Deep Learning for Activity Recognition on Mobile Devices. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '16)*. ACM, New York, NY, USA, 185–188. DOI: https://doi.org/10.1145/2968219.2971461
[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.
[33] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567* (2015).
[34] Yan Wang, Jie Yang, Hongbo Liu, Yingying Chen, Marco Gruteser, and Richard P. Martin. 2013. Sensing Vehicle Dynamics for Determining Driver Phone Use. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*.
[35] Junyuan Xie, Linli Xu, and Enhong Chen. 2012. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems*. 341–349.
[36] Xuehan-Xiong and Fernando De la Torre. 2013. Supervised Descent Method and its Application to Face Alignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[37] Jie Yang, Simon Sidhom, Gayathri Chandrasekaran, Tam Vu, Hongbo Liu, Nicolae Cecan, Yingying Chen, Marco Gruteser, and Richard P. Martin. 2011. Detecting Driver Phone Use Leveraging Car Speakers. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*.
[38] 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, and Andrew T. Campbell. 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*.
[39] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. 2015. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4694–4702.