Drive-Net: Convolutional Network for Driver Distraction Detection

Mohammed S. Majdi*, Sundaresh Ram†, Jonathan T Gill*, Jeffrey J Rodriguez*
*Electrical and Computer Engineering,
University of Arizona, AZ, Tucson, USA
†School of Electrical and Computer Engineering,
Cornell University, NY, Ithaca, USA

Email: mohammadsmajdi@arizona.edu (Mohammed S. Majdi)
Email: sr2255@cornel.edu (Sundaresh Ram)
Email: jtgill@arizona.edu (Jonathan T Gill)
Email: jjrodrig@arizona.edu (Jeffrey J Rodriguez)

***Abstract*—To help prevent motor vehicle accidents, there has been significant interest in finding an automated method to recognize signs of driver distraction, such as talking to passengers, fixing hair and makeup, eating and drinking, and using a cell phone. In this paper, we present an automated supervised learning method called Drive-Net for driver distraction detection. Drive-Net uses a combination of a convolutional neural network (CNN) and a random decision forest to classify images of a driver. We compare the performance of our proposed Drive-Net to two other popular machine learning approaches: a recurrent neural network (RNN) and a multilayer perceptron (MLP). We tested the methods on a publicly available database of images acquired in a controlled environment that contained about 22425 images manually annotated by an expert. The results show that Drive-Net achieves a detection accuracy of 95%, which is 2% more than the best results obtained in the same database using other methods.**

***Index Terms*—Image classification, convolutional neural networks, random forest, driver distraction**

1. Introduction

Distracted driving is a major cause of motor vehicle accidents. Each day in the United States, approximately 9 people are killed and more than 1000 are injured in crashes that involve a distracted driver. [**?**] It is estimated that about 25% of fatalities in motor vehicle accidents are due to distracted driving. [**?**] A study of American motor vehicle fatalities [**?**] reveals the top 10 causes of distracted driving:

1) Generally distracted or "lost in thought" –- 62%
2) Cell phone use — 12%
3) Outside person, object, or event – 7%.
4) Other occupants -– 5%.
5) Using or reaching for a device brought into the car (e.g., phone) – 2%.
6) Eating or drinking -– 2%.
7) Adjusting audio or climate controls — 2%.
8) Using devices to operate the vehicle (e.g., adjusting mirrors or seatbelts) — 1%.
9) Moving objects (e.g., insects or pets) -– 1%
10) Smoking related — 1%.

Therefore, there is interest in using dashboard camera image analysis to automatically detect drivers engaged in distracting behavior. A dataset of such dashboard camera images, observing various activities of drivers, has been compiled and used for the Kaggle competition regarding automated detection of driver distraction. [**?**] Figure **??** shows examples of some images from the dashboard camera manually annotated as different activities while driving.

Object detection and human behavior detection are well-researched topics in the computer vision literature. [**?**] Machine learning (esp. Deep Learning) techniques can often learn complex models and achieve high accuracy, so many researchers have started to apply such techniques to solve computer vision problems, including object detection and human behavior detection. For example, the Inception-v4 model proposed by Szegedy [**?**] is a supervised learning model made up of deep convolutional residual networks (ResNet) that have more than 75 trainable layers, and achieves 96.92% accuracy in the ImageNet data set. Girshick [**?**] introduced a very powerful method for object detection and segmentation using a region-based convolutional neural network (CNN). This method divides the human behavior detection problem into two problems. First, they apply an object detection algorithm to detect the regions of interest (ROIs) where people are present within an image. Next, each ROI is fed to a CNN to identify the type of behavior exhibited in the given ROI. Adding other traditional machine learning methods, such as ensemble learning (i.e. bagging) and K nearest neighbors (KNN), to the CNN model is a way to improve the accuracy of the already existing model. [**?**]

One of the main drawbacks of CNN is that training the network using a large data set can lead to overfitting the model. To avoid this, ensemble methods such as random

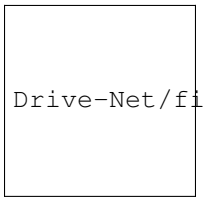

Figure 1. Representative dashboard camera images of drivers being distracted. From left to right: drinking while driving; safe driving; reaching behind while driving; talking on the phone (left hand) while driving. [**?**]

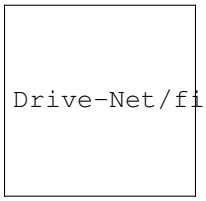

Figure 2. An overview of the proposed Drive-Net. Our proposed CNN architecture (shown on the left side) consists of two convolution layers (conv), each followed by a maxpooling layer (pool), and a final ReLU layer, the output of which is regularized using dropouts to obtain a fully connected layer (FC). The FC layer is fed as input to the random forest classifier (on the right side), which predicts the final class label.

decision forests can be effective. With this in mind, we propose a new supervised learning algorithm called Drive-Net that combines a CNN and a random forest in a cascading fashion for application to the problem of driver distraction detection using dashboard camera images. We compare our proposed Drive-Net to two other neural network methods: a residual neural network (RNN) and a multilayer perceptron (MLP). We show that Drive-Net achieves better classification accuracy than the driver distraction detection algorithms that were proposed in the Kaggle competition. [**?**]

2. Methods

Our proposed method, Drive-Net, is a cascaded classifier consisting of two stages: a CNN as the first stage, whose output layer is fed as input to a random decision forest to predict the final class label. We define each stage in detail below.

2.1. Convolutional Neural Network Configurations

We adopted the U-Net architecture [**?**] as the basis for our CNN. The motivation behind this architecture is that the contracting path captures the context around the objects to provide a better representation of the object compared to architectures such as AlexNet [**?**] and VGGNet [**?**] Very large networks like AlexNet and VGGNet require learning a massive number of parameters and are very difficult to train in general, needing significant computational time. Thus, we empirically modified the U-Net architecture in this work to suit our application.

To construct our CNN, we discard U-Net's layers of up-convolution and the last two layers of down-sampling and replace them with a 1×1 convolution instead to obtain a fully connected layer. We use the rectifier activation function [**?**] for our CNN as the constant gradient of rectified linear units (ReLU) results in faster learning and reduces the problem of vanishing gradient compared to the hyperbolic tangent ($tanh(\bullet)$). We implement a maximum-pooling layer instead of average-pooling in the subsampling layer. We observed that the performance is better when a ReLU layer was configured with the maximum pool layer, resulting in higher classification accuracy after 50 epochs. We use the 1×1 convolutional filter for the Adam [**?**] optimizer. All other parameters, such as the number of layers, the size of the convolutional kernel, the training algorithm, and the number of neurons in the final dense layer, were experimentally determined for our application.

To keep the training time small, we reduced the size of the dashboard camera images by a factor of 10 by making them 64×48 in size and feed them as input to our CNN. We do not zero-pad the image patches, as the ROI of human activity is located toward the center of the image. Two consecutive convolutional layers are used in the network. The first convolutional layer consists of 32 kernels of size $5 \times 5 \times 1$. The second convolutional layer consists of 64 kernels of size $5 \times 5 \times 32$. The subsampling layer is set as the maximum value in nonoverlapping windows of size 2×2 (stride of 2). This reduces the size of the output of each convolutional layer by half. After the two convolutional and subsampling layers, we use a ReLU layer, where the activation y for a given input x is obtained as

$$y = f\,(x) = \max(0, x) \tag{1}$$

A graphical representation of the architecture of the proposed CNN model is shown in Figure **??** (see the left side).

2.2. Random Decision Forest

A random forest classifier consists of a collection of decision tree classifiers combined to predict the class label, where each tree is grown in a randomized fashion. Each

decision tree classifier consists of decision (or split) nodes and prediction (or leaf) nodes. The prediction nodes of each tree in the random forest classifier are labeled by the posterior distribution over the image classes. [**?**] Each decision node contains a test that best splits the space of the data to be classified. An image is classified by sending it down the decision tree and aggregating the posterior distributions that are reached. Most of the time, randomness is added to training at two points: when subsampling the training data and when choosing node tests. Each tree within the random forest classifier is binary and grows top-down. At each node, we select the binary test to maximize the information gain obtained by partitioning the training set Q of image patches into two sets Q_i according to the test.

$$\Delta E = -\sum_i \frac{(Q_i|}{(Q|} E\left(Q_i\right) \tag{2}$$

Here $E\left(\cdot\right)$is the entropy of the set and $|\cdot|$is the size of the set. We repeat this selection process for each decision node until it reaches a certain depth. Many implementations of random forests [**?**], [**?**] use simple pixel-level tests at the nodes because it results in faster tree convergence. As we are interested in features that encode shape and appearance, we are interested in spatial correspondence between pixels. Therefore, we use a simple test proposed by Bosch [**?**] — as a linear classifier on the characteristic vector — at each decision node.

Suppose that T is the set of all trees, C is the set of all classes, and L is the set of all leaves for a given tree. During training, posterior probabilities $P_{t,l}\left(Y\left(I\right) = c\right)$ for each class $c \in C$ are found at each leaf node $l \in L$ for each tree $t \in T$. These probabilities are calculated as the ratio of the number of images I of class c that reach a leaf node l to the total number of images that reach that leaf node l. $Y\left(I\right)$ is the class label of image I. During test time, we pass a new image through every decision tree until it reaches a prediction (or leaf) node, average all the posterior probabilities, and classify the image as

$$\widehat{Y}\left(I\right) = \underset{c}{\operatorname{argmax}} \left\{ \frac{1}{|T|} \sum_{t=1}^{|T|} P_{t,l}\left(Y\left(I\right) = c\right) \right\} \tag{3}$$

where l is the leaf node reached by the image I in the tree t. A graphical representation of the proposed random forest classifier is shown in Figure **??** (see the right side).

3. Experiments and Results

3.1. Data set

The Kaggle competition [**?**] for driver distraction has provided 22425 images for training and 79727 for tests. Since we did not have access to the test labels, our experiments were done solely on the training images. However, the quality and conditions of the training and testing images are similar; the only difference is that none of the drivers used in the training data set appear in the images in the test data set. The images are of size 640×480, and for our experiments we converted them from color to grayscale.

Ten classes are provided, related to the ones listed in Section I. Each class includes almost tens of the data, so that we have a uniform distribution of sample data.

- c0: safe driving
- c1: texting (right hand)
- c2: talking on the phone (right hand)
- c3: texting (left hand)
- c4: talking on the phone (left hand)
- c5: operating the radio
- c6: drinking
- c7: reaching behind
- c8: hair and makeup
- c9: talking to a passenger

3.2. Algorithm Parameters

The convolutional neural random forest classifier is implemented using TensorFlow, [**?**] and runs on an NVIDIA GeForce GTX TITAN X GPU with 16 GB of memory. The classifier was trained using the Adam stochastic gradient descent algorithm [**?**] to efficiently optimize CNN weights. The weights were normalized using initialization as proposed in Kingma [**?**] and updated in a mini-batch scheme of 128 candidates. The biases were initialized with zero and the learning rate was set at $\alpha = 0.001$. The exponential decay rates for the estimates for the first and second moments were set as $\beta_1 = 0.9$ and $\beta_2 = 0.99$, respectively. We used $\epsilon = 10^{-8}$ to prevent division by zero. A dropout rate of 0.5 was implemented as regularization, applied to the output of the last convolutional layer and the dense layer to avoid overfitting. Finally, we used an epoch size of 50. SoftMax loss (cross-entropy error loss) was used to measure the error loss. We used 100 estimators and a keep rate of $\gamma = 10^{-4}$ for the random forest algorithm.

3.3. Performance Evaluation

We tested the algorithm performance by performing k-fold cross-validation on the entire dataset. For our experiments, we varied the values of $k \in [2,\ 5]$ and found that the results were consistent enough to indicate that the network is not overfitting. Therefore, we chose $k = 5$. First, we randomly choose the order of the driver images within the data set. For each fold of the k-fold cross-validation, we chose 80% of the 22425 images as the training data set and tested the trained model on the remaining 20% of the images. We ensured that the images from the entire dataset appeared in the test dataset only once in all k-folds, thereby allowing each image to be classified as a test image exactly once.

We compared our proposed Drive-Net with two other neural network classifiers: an RNN classifier, [**?**] and an MLP classifier [**?**]. We report the classification accuracy, which is defined as the percentage of correct predictions

and the number of false positives (a.k.a. false detections) for each class, as the figures of merit for comparing the algorithms. For classification precision, we present the results of seven other methods based on support vector machines (SVMs), dimensionality reduction techniques such as principal component analysis (PCA), feature extraction techniques such as histogram of oriented gradients (HOG), very deep convolutional nets such as VGG-16, VGG-GAP, and an ensemble of these two as reported by Zhang [**?**] using the same Kaggle dataset of 22425 images.

Table **??** shows the mean classification accuracy of the different classifiers as reported by Zhang [**?**] and that of the three neural network classifiers that we implemented. From Table **??**, we observe that Drive-Net achieves a classification accuracy of 4.8% points higher than the VGG-16 classifier, 3.7% points higher than a VGG-GAP classifier, 2.4% points higher than an ensemble of VGG-16 and VGG-GAP classifiers, 3.3% points higher than the RNN classifier and 13% points higher than the MLP classifier.

TABLE 1. MEAN CLASSIFICATION ACCURACY OF AUTOMATED METHODS

Method	Accuracy
Methods from Zhang [?]	
Pixel SVC	18.3%
SVC + HOG	28.2%
SVC + PCA	34.8%
SVC + BBox + PCA	40.7%
VGG-16	90.2%
VGG-GAP	91.3%
Ensemble VGG-16 and VGG-GAP	92.6%
Methods we implemented:	
MLP	82%
RNN	91.7%
Drive-Net	95%

Table **??** shows the number of false classifications for our Drive-Net and for RNN and MLP. From Table **??**, we observe that our Drive-Net can identify the classes c6 (drinking) and c3 (texting with left hand) with minimum false detections, while the RNN and MLP classifiers have a hard time distinguishing these classes with many false detections, usually higher than the number of false detections in the other classes of these methods. Also, the total number of false detections for our Drive-Net is an order of magnitude smaller than that of the MLP classifier and slightly smaller compared to the RNN classifier.

TABLE 2. CLASS'S ERROR COUNT FOR NEURAL NETWORK METHODS

Class	Drive-Net	MLP	RNN
c0	35	356	48
c1	17	199	34
c2	14	158	31
c3	09	116	47
c4	34	252	30
c5	15	108	14
c6	08	263	18
c7	21	117	10
c8	29	181	46
c9	26	268	46
All Classes	208	2018	324

4. Conclusion

Distracted driving is one of the main causes of motor vehicle accidents. Therefore, there is a significant interest in finding automated methods to recognize signs of driver distraction from dashboard camera images installed in vehicles. We propose a solution to this problem using a supervised learning framework. Our method, named Drive-Net, combines a CNN and a random forest classifier to recognize the various categories of driver distraction in images. We apply our Drive-Net to a publicly available dataset of images used in a Kaggle competition and show that our Drive-Net achieves better accuracy than the driver-distraction algorithms reported in the competition. We also compared Drive-Net to two other neural network algorithms: an RNN and an MLP algorithm, using the same data set. The results show that DriveNet achieves a better detection accuracy compared to the other two algorithms.