https://doi.org/10.2352/ISSN.2470-1173.2017.10.IMAWM-162

Distracted Driver Detection: Deep Learning vs Handcrafted Features

Murtadha D Hssayeni, Sagar Saxena, Raymond Ptucha, Andreas Savakis; Rochester Institute of Technology, Rochester, NY US

Abstract

According to the National Highway Traffic Safety Administration, one in ten fatal crashes and two in ten injury crashes were reported as distracted driver accidents in the United State during 2014. In an attempt to mitigate these alarming statistics, this paper explores using a dashboard camera along with computer vision and machine learning to automatically detect distracted drivers. We consider a dataset that incorporates drivers engaging in seven different distracting behaviors using left and/or right hands. Traditional handcrafted features paired with a Support Vector Machine classifier are contrasted with deep Convolutional Neural Networks. The traditional features include a blend of Histogram of Oriented Gradients and Scale-Invariant Feature Transform descriptors used to create Bags of Words. The deep convolutional methods use transfer learning on AlexNet, VGG-16, and ResNet-152. The results yield 85% accuracy with ResNet and 82.5% accuracy with VGG-16, which outperformed AlexNet by almost 10%. Replacing the fully connected layers by a Support Vector Machine classifier did not improve the classification accuracy. The traditional features yielded much lower accuracy than the deep convolutional networks.

Introduction

According to the last National Highway Traffic Safety Administration (NHTSA) report, one in ten fatal crashes and two in ten injury crashes were reported as distracted driver crashes in the United State in 2014 [1]. Sadly, this translates to an estimated 3,179 people killed and 431,000 people injured [1]. The largest proportion of the crashes were caused by teen drivers who are 15-19 years old. The AAA Foundation for Traffic Safety reviewed moderate-to-severe crashes that involve teen drivers and found that six out of ten drivers were distracted [2]. In an attempt to mitigate this problem, different campaigns and programs have been conducted to educate drivers about the problem, the risks, and how to avoid it.

Recent developments have paved the way for real-time approaches to detect the distraction and then assist and alert the distracted driver. Any activity that takes the driver's attention away from the critical task of driving safely is considered a distraction, e.g. interacting with a passenger, using a cellphone, reaching to adjust the radio, etc. Distractions are classified into three types: visual, manual and cognitive [3] depending on what the driver is looking, doing or thinking. Previous approaches to detecting distractions were based on observing the driver's perception [4] or estimating the driver's distracted behaviors using motion sensors or a camera [5, 6]. The method in [7] used head tracking data (position and rotation) to analyze the driver's steering and lane keeping behavior to detect if the driver is distracted without considering specific behavior. A complementary detection method in [6] increases accuracy by reducing the false triggers to the lane-keeping assistance when the driver intentionally changes lane. Thus, the lane-keeping assistant triggers only in the case of vehicle drift and driver distraction.

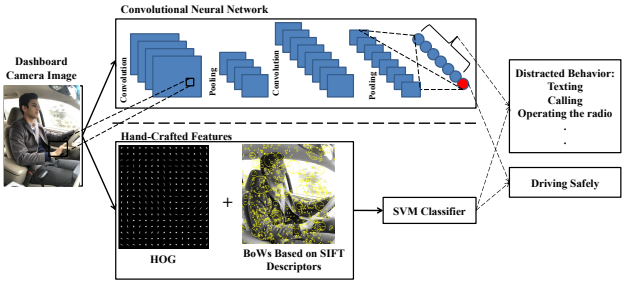

Figure 1: Diagram illustrating the methods considered.

In April 2016, State Farm started a competition on Kaggle.com by submitting a dataset of dashboard camera images that showed drivers either engaging in distracted behaviors or driving safely [8]. The goal of this competition was the detection of distracted drivers based on activity recognition. The most effective techniques used multiple Convolutional Neural Networks (CCN) models to obtain very high accuracies. Many submissions preprocessed the images by cropping the driver's region after applying skin, body, face, head, limbs, or/and joints detection algorithms. One common problem while fine-tuning the models was overfitting the training set. Participants who managed to tackle this issue got improved results. However, no single CNN model or method stood out as the best.

Traditional computer vision techniques pair hand crafted low level features such as Scale Invariant Feature Transform (SIFT) [9], Speeded-Up Robust Features SURF [10], or Histogram of Oriented Gradients (HOG) [11] along with complimentary classifiers such as support vector machines (SVM) or neural networks. LeCun et al. [12] introduced CNNs, computer vision oriented deep feed forward networks based upon a hierarchy of abstract layers. CNNs are end-to-end models, learning the low level features and classifier simultaneously in a supervised fashion, giving advantage over methods using independent vision features and classifiers.

In this paper, we compare the performance of CNNs with traditional features fed into a Support Vector Machine (SVM) classifier for automatically detecting distracted drivers using data from a dashboard camera. Figure 1 outlines the two types of approaches considered. Transfer learning is applied to fine-tune three pre-trained CNN models. The traditional methods use handcrafted features, specifically HOG and clustered SIFT descriptors using Bag of Words (BoWs) [13] passed into a tuned SVM classifier.

Background

Traditional Computer Vision Techniques

The traditional hand-crafted features considered in this paper include HOG [11] and SIFT descriptors [9]. HOG features and clustered SIFT features using Bag of Words [13] are passed to an SVM [14] classifier. HOG counts the occurrences of gradient orientations on a dense grid of uniformly spaced cells. SIFT uses difference of Gaussians to localize interesting features at varying resolutions. The SIFT features capture the gradient structure and are tolerant to modest amounts of geometric transformations. The Bag of Words technique tabulates SIFT features across a set of quantized buckets, irrespective of location in the image.

Deep Convolutional Neural Networks

Deep CNNs are a type of Artificial Neural Network (ANN) inspired by the mammalian visual cortex. The main components of CNNs include convolutional filters, pooling, non-linear activation layers, fully-connected layers (FC) and finally the objective function loss layer. CNNs have been used in a wide range of applications in the tasks of object and activity recognition, object detection, computational photography, and natural language processing.

In 2012, Krizhevsky and Hinton [15] beat the nearest competitor by 10% in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [16]. They used a seven layer deep CNN, known as AlexNet, that leveraged the advantage of a simple, yet effective activation function called Rectified Linear Unit (ReLU), as well as a powerful regularization scheme called dropout [17]. Their methodologies were so effective, that all subsequent winners in this prestigious ILSVRC competition have used CNNs. AlexNet [15], VGG [18] and ResNet [19] were the models that won ILSVRC2012, ILSVRC2014, and ILSVRC2015, respectively.

Recent progress in classification accuracy can be attributed to advances in building deeper architectures and improved regularization methods [17, 20–22]. Zeiler & Fergus [23] improved classification results by introducing random crops on training samples and improved parameter tuning methodologies. Simonyan and Zisserman [18] investigated the effects of network depth and Szegedy et al. [24] used banks of smaller convolutional filters to simultaneously improve accuracy while decreasing the number of parameters.

Advances were also attributed to the use of new non-linear activation functions [21, 25–28] such as Rectifier Linear Units (ReLU). He et al. [29] used a parameterized version of ReLU to simultaneously learn slope parameters along with weight hyper parameters during backpropagation. Most recently, He et al. [19] introduced skip connections in a residual learning framework to enable learning of very deep networks.

AlexNet Model

The AlexNet architecture [15] is shown in the first column of Table 1. The input image size for the model is $227 \times 227 \times 3$. This network consists of five convolutional layers and three FC layers. The receptive field sizes are 11×11, and 5×5 for the first and second convolutional layers, respectively, and 3×3 for the last three convolutional layers. In addition, it consists of local response normalization layers, ReLU layers, and overlapping max-pooling layers. To reduce overfitting on the training data, the following methods are applied during the training stage: data augmentation by cropping 224×224 patches and flipping them horizontally, dropout technique [17] with probability 0.5 which is applied to the first two FC layers, using batch normalization with a size of 128, and then applying Stochastic Gradient Descent Learning (SGD) with momentum (m) update and weight decay (λ). An error rate (ε) of 15.3% was achieved in ILSVRC2012 after averaging seven AlexNet models.

VGG Model

Simonyan *et al.* [18] investigated the benefits of CNN depth on image recognition accuracy in ILSVRC2014 by adding more convolutional layers. What makes training these deep CNNs feasible is using 3×3 filters for all the convolutional layers. The size of input RGB image for this model is 224×224. The image is passed through a sequence of convolutional layers (the number of layers are 8, 10, 13, and 16 for different architectures) and three FC layers. All architectures have ReLU layers, and max-pooling layers. Training these models generally follows the AlexNet training procedure [15]. The 19-weight layers architecture was the winning architecture in ILSVRC2014 after averaging seven models and the best single model was the 22-weight layers. The 19-weight layers architecture is shown in Table 1.

ResNet Model

He *et al.* [19] presented a new training framework called residual learning to facilitate very deep CNN training. In this framework, a shortcut connection was added to each building block (two or three consecutive convolutional layers). Because of this new training framework, they were able to train very deep networks, with 18, 34, 50, 101, and 152-weight layers, without encountering the degradation problem. The input image size for these networks is 224×224. Only one FC layer is used without using dropout layers. For ILSVRC2015, they trained the networks using SGD with batch size $B = 256$, momentum $m = 9 \times 10^{-1}$, and weight decay $\lambda = 1 \times 10^{-4}$. The 152-weight layers architecture that is shown in Table 1 was the winner in this competition.

Experimental Methods

Two approaches were explored in this work. The first is based on traditional handcrafted features (HOG and BoWs) along with SVM, while the second is based on deep convolutional neural networks, specifically AlexNet, VGG-16, and ResNet-152.

Method 1: Traditional Handcrafted Features

In this approach, features based on HOG and the clustered SIFT descriptors using BoWs were extracted and concatenated in a single vector. The images were first down-scaled to 227×227 to make a consistent comparison with deep CNN methods that use comparable input image size. The MATLAB function "extractHOGFeatures" was used to extract the HOG features. Multiple cell sizes were examined, namely $8 \times 8, 16 \times 16, 24 \times 24$, and 32×32. All other parameters were set to the default values. The VL-Feat library wes used to extract the SIFT descriptors [30]. The descriptors from the training images were clustered using k-means to obtain BoW vocabularies. Different numbers of words (500 to 2000) were examined to find the best BoW vocabulary for this classification problem.

Before image classification using SVM classifier, Principal

Table 1: 8-weight Layers architecture for AlexNet [15], 19-weight layers architectures for VGG [18], and 152-weight layers for ResNet [19]. Convolutional layers notation is following the format: conv(filter size)-(number of filters) [18]. The shared layers between the models are depicted as rows across the table. In addition, the horizontally corresponding blocks between the models result in the same output dimensions.

8-Weight Layers	19-Weight Layers	152-Weight Layers
Input Image		
conv11-96	conv3-64 conv3-64	conv7-64
Max-pooling Layer		
conv5-256	conv3-128 conv3-128	
Max-pooling Layer		
conv3-384 conv3-384 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256	3 Blocks of [conv1-64 conv3-64 conv1-256]
	Max-pooling Layer	A block of [conv1-128 conv3-128 conv1-512] with a stride of 2
	conv3-512 conv3-512 conv3-512 conv3-512	7 Blocks of [conv1-128 conv3-128 conv1-512]
	Max-pooling Layer	A block of [conv1-256 conv3-256 conv1-1024] with a stride of 2
	conv3-512 conv3-512 conv3-512 conv3-512	35 Blocks of [conv1-256 conv3-256 conv1-1024]
Max-pooling Layer		A Blocks of [conv1-512 conv3-512 conv1-2048] with a stride of 2
		2 Blocks of [conv1-512 conv3-512 conv1-2048]
Fully-connected Layer-4096		Average-pooling Layer
Fully-connected Layer-4096		
Fully-connected Layer-1000		

Component Analysis (PCA) [31] was applied on the extracted feature vectors. After that, the components that captured 99% of the total energy in the features were used as input to the SVM classifier. The MATLAB function "pca" is used to perform PCA and the LibSVM library [32] was used to train and test the SVM classifier with linear kernel. Grid search was conducted on about 25% of the dataset to choose the optimal cost parameter (c) for the linear kernel.

Method 2: Deep CNN

Because of the relatively small size of the training dataset, transfer learning was used to fine-tune three deep CNN models that were pre-trained on ImageNet ILSVRC challenge. These models are AlexNet, VGG-16, and ResNet-152. Another reason to use transfer learning is that it initializes the weights of the convolutional layers using the pre-trained models and the first layers of these pretrained models detect edges, blobs and textures that are essential for most of the detection and classification methods. The first step was to resize the input images to 227×227 for AlexNet, and to 224×224 for VGG-16 and ResNet-152. After resizing, the RGB mean of the training images was subtracted from all the images.

To fine-tune AlexNet and VGG-16, the last two FC layers were initialized with random weights, whereas the rest of the layers were initialized using the weights of the pretrained AlexNet and VGG-16 models. The output depth of last FC layer was changed to 10, which is the number of class labels in the dataset. To fine-tune ResNet-152, the FC layer was changed to a depth of 10 and initialized with random weights, whereas the rest of the layers were initialized using the weights of the pre-trained ResNet-152 model. Fine-tuning AlexNet and VGG-16 was performed using SimpleNN wrapper in Matconvnet library [33]. The DagNN wrapper in the same library was used to fine-tune ResNet-152. During training, we allowed the convolutional layer weights that were initialize using transfer learning to be fine-tuned as required in distracted driver behavior classification.

All models were trained using mini-batch SGD, with momentum (m) and weight decay (λ). The details of the parameters optimization are mentioned in the results section. Dropout was used for the first two FC layers in AlexNet and VGG-16 architecture.

In addition to use the FC layers as a classifier, the SVM classifier was explored. Features were extracted from the first and second FC layers of the best fine-tuned AlexNet and VGG-16, and from the average pooling layer of the best fine-tuned ResNet-152. These features were passed into a SVM classifiers (linear vs. radial basis function (RBF) kernel), where the c and gamma parameters were tuned using grid search based on about 25% of the dataset.

Results and Discussions

An online dataset was used in this work that includes around twenty thousand dashboard camera images for multiple drivers. These images were captured while the drivers were safely driving or were engaging in seven distracting behaviors (e.g. using a cellphone for texting or calling, drinking, operating car accessories, etc.). The drivers used right or left hands for some of the activities, therefore, the total number of labels for this dataset was ten. For the purpose of validating the methods, images of 80% of the

drivers were used for training and the images of the other 20% of the drivers were used for testing. Data augmentation was not performed. The images were down-scaled to fit the required image size for each model.

For the first method, each of the handcrafted features was first evaluated separately using linear SVM to find their optimal parameters. For HOG, a cell size of 24×24 was showed to have the highest accuracy after trying multiple cell sizes as shown in Table 2. Other parameters were 2×2 block size, half of the block size as the overlap between adjacent blocks, and nine bins for the orientation histogram. For BoWs, a 1750-word vocabulary was selected because it yielded the highest accuracy, as shown in Table 2. Finally, a combination of HOG and BoWs with the previously selected parameters was used to classify each image. PCA was applied on the combined features, so that 98% of their energy was captured, resulting in 1287 selected features from 3204 extracted features. An accuracy of 27.7% was obtained from this method. Using 99% of the features energy yielded lower accuracy. The accuracy of using the combined features was lower than the accuracy of using only HOG features; this means the extracted vocabularies for BoWs were not informative.

Table 2: The accuracy of passing each of the handcrafted features individually with different parameters' values to a linear SVM.

HOG Features				
Cell Size	8×8	16×16	24×24	32×32
Accuracy (%)	27.2	30.5	**33.2**	29.9
SIFT/BoW Features				
# of Words	900	1150	1450	1750
Accuracy (%)	19.3	18.3	17.2	**21.5**

For the deep CNN approach, the following hyper parameters were adjusted for each model (AlexNet, VGG-16, and ResNet-152): learning rate (η), weight decay (λ) and batch size (B), whereas the momentum was fixed to 9×10^{-1}. For Alexnet, the following starting values for the learning rate schedule were examined : $\eta = 5 \times 10^{-4}$, 9×10^{-4}, and 1×10^{-3}. The weight decay and the batch size were $\lambda = 1 \times 10^{-5}$ and $B = 50$, respectively. The main issue with fine-tuning this model was overfitting the training data in the second epoch as shown in Figure 2 which made the testing error rate (ε) dropped slightly after this epoch as shown in Figure 3.

Increasing the batch size and also increasing the dropout probability to 0.6 didn't help with generalizing and improving the testing accuracy. The highest accuracy was 72.6% for the following parameters: the learning rate schedule was $\eta = 9 \times 10^{-4}$, 7×10^{-4}, and 5×10^{-4} for the first, second and third 10 epochs, respectively, and the weight decay and the batch size were $\lambda = 1 \times 10^{-5}$ and $B = 50$, respectively.

For VGG-16, the following starting values for the learning rate schedule were examined : $\eta = 1 \times 10^{-7}$, and 5×10^{-5}. The batch size was fixed to 2 because the required GPUs to train the model were not available at the time of the experiments. The learning rate schedule associated with the best accuracy was $\eta = 5 \times 10^{-6}$, 3×10^{-5}, and 1×10^{-5} for (30,15, and 10) epochs, respectively. After selecting the previous learning rate schedule, two values for the weight decay were examined which were $\lambda = 1 \times 10^{-5}$ and 5×10^{-5}. The weight decay $\lambda = 5 \times 10^{-5}$

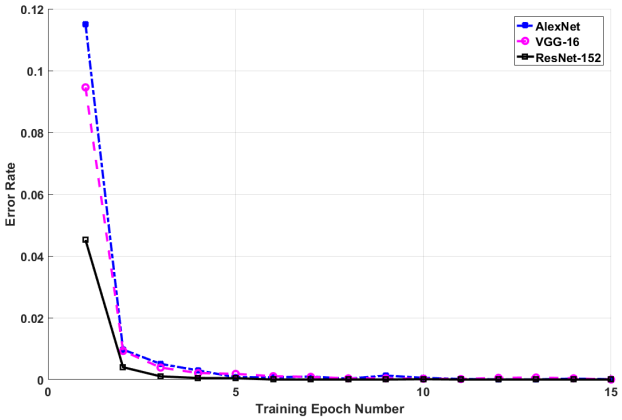

Figure 2: Training curves for all three models that were fine-tuned for the first 15 epochs. Training error rate continued to be approximately zero after these epochs. AlexNet used SGD with η starting at 9×10^{-4}, $\lambda = 1 \times 10^{-5}$, $m = 9 \times 10^{-1}$, and mini-batch size of 50. VGG-16 and ResNet-152 used SGD with $\lambda = 5 \times 10^{-5}$, $m = 9 \times 10^{-1}$ and mini-batch size of 2. The learning rate started at $\eta = 5 \times 10^{-5}$ for VGG-16 and at $\eta = 1 \times 10^{-3}$ for ResNet-152.

gave a better driver behavior prediction with 82.5% accuracy. The fine-tuning curves for the training and validation are shown in Figure 2 and 3.

For ResNet-152, the following starting values for the learning rate schedule were examined : $\eta = 8 \times 10^{-4}$, 1×10^{-3}, 5×10^{-3}, and 1×10^{-2}, with different batch size values which were B =2, 4, and 16. Three trials for a batch size of 16 with different learning rate were performed and the results are shown in Figure 4. Reducing the batch size to 4 yielded slightly higher accuracy; after that, the weight decay was reduced from 1×10^{-4} to 5×10^{-5} which added about 2% to the accuracy as shown in Figure 5. The last trial was reducing the batch size to 2 which added 1% to the accuracy to be 85%. Figure 5 shows the testing curves for the previous trials. The learning rate schedule with lowest error rate was $\eta = 1 \times 10^{-3}$, 5×10^{-4}, 1×10^{-4}, and 5×10^{-5} for (10,10,5, and 5) epochs for batch size of 2. Increasing the batch size as a regularization approach did not reduce the error rate. It

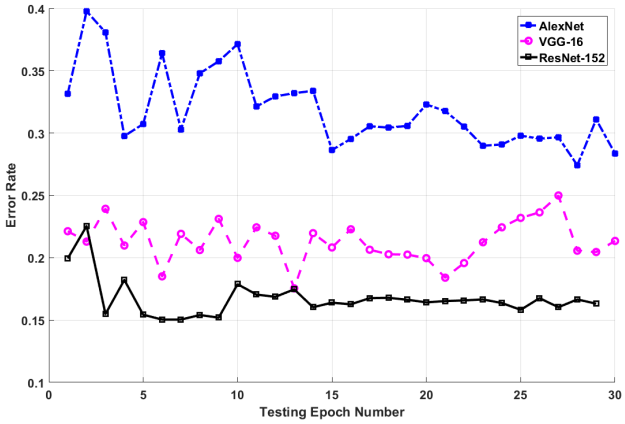

Figure 3: The testing curves for the three fine-tuned models used with 20% of the drivers.

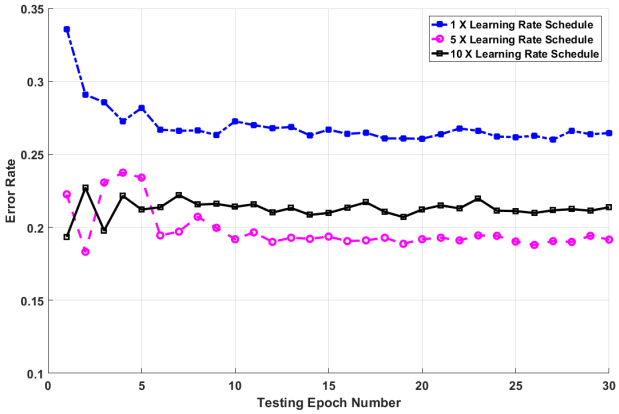

Figure 4: The testing curves for three trials to fine-tune ResNet-152. For all the trails, SGD was used with $\lambda = 1 \times 10^{-4}$, $m = 9 \times 10^{-1}$ and $B = 16$. The learning rate schedule for the first trial was $\eta = 1 \times 10^{-3}$, 5×10^{-4}, 1×10^{-4}, and 5×10^{-5} for (10, 10, 5 and 5) epochs, respectively. The learning rate schedule was multiplied by a factor of 5 and 10 for the second and third trials, respectively.

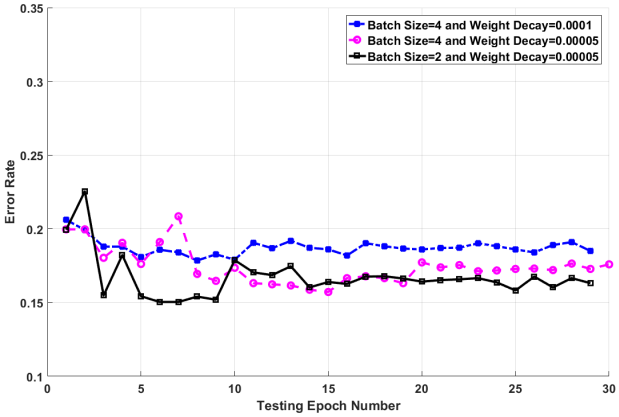

Figure 5: The testing curves for three trials to fine-tune ResNet-152. For first and second trails, SGD was used with $m = 9 \times 10^{-1}$, $B = 4$, and a learning rate schedule that started at $\eta = 1 \times 10^{-3}$. The weight decay was reduced from 1×10^{-4} to 5×10^{-5} for these two trials. For the third trial, the batch size was reduced to 2, and the rest of the hyper parameters were kept the same as in trail 2.

just made the model need more iterations to fit the training data. After choosing the learning rate and the batch size, the following values for the weight decay were examined: $\lambda = 5 \times 10^{-5}$, and 1×10^{-4}. $\lambda = 5 \times 10^{-5}$ was shown to produce better accuracy. Fine-tuning for the ResNet-152 model converged faster to zero error rate using the training data than the other two models (i.e. deeper models fitted the training data faster) as shown in Figure 2. This model also resulted in lower error rate using the testing data. The best validation curve for this model is shown in Figure 3.

To compare SVM classification with Softmax layer for CNN models, the extracted features from the best fine-tuned models were passed to SVM classifier. For AlexNet and VGG-16, the features were extracted from (FC6, ReLu6, FC7, and ReLU7), whereas for ResNet-152, the feautres were extracted from the average-pooling layer. Table 3 shows the results for this experiment with two SVM kernels which are linear and RBF. The best accuracy for AlexNet was 70% with Linear kernel using the features extracted from FC7. Whereas, the best results for VGG-16 and ResNet-152 was 78.5%, and 84.6%, respectively, using RBF kernel. In addition, the classification accuracy for the features that were extracted from ReLU6 for both AlexNet and VGG-16 were better than the classification accuracy for the features that were extracted from FC6. This illustrates the advantage of using ReLu rather than linear activation layers.

The final results for all the models are shown in Table 4. The highest accuracy of 85% was recorded for ResNet-152 that was slightly higher that VGG-16 accuracy by 2.5%. VGG-16 outperformed AlexNet by about 10%. In addition, the Softmax layer as a classifier performed better than SVM for all the models. The handcrafted features resulted in a low accuracy of 27.7%.

Table 3: The accuracy for each model after applying SVM classifier with linear and RBF kernels. The features were extracted from a different layer of each network for each trail.

Model	**Features**	**Linear SVM**	**RBF SVM**
AlexNet	FC6	66.4	66.4
	ReLu6	68.5	68.6
	FC7	**70**	68.6
	Relu7	69.1	66.9
VGG-16	FC6	77.8	76.4
	ReLu6	78.2	**78.5**
	FC7	77.9	77.9
	Relu7	78	77.9
ResNet-152	Average Pooling Layer	84.2	**84.6**

Table 4: Final results for all the methods.

Features	**Classifier**	
	Softmax (%)	SVM (%)
Handcrafted Features	N/A	27.7
AlexNet	72.6	70
VGG-16	82.5	78.5
ResNet -152	**85**	84.6

Conclusion

Three well-known CNN models have been compared to traditional handcrafted features for automatically detecting if the drivers are engaging in distracting behaviors based on images from a dashboard camera. ResNet-152 yielded the highest accuracy of 85% using Softmax layer as a classifier which is better than VGG-16 accuracy and much better than AlexNet. In addition, using SVM classifier on the extracted features for the last layers of the CNN models did not increase the accuracy. On the other hand, traditional handcrafted features yielded much lower accuracy than deep CNN models. The main issue with fine-tuning the CNN models was overfitting the training data. This issue can be addressed by collecting more training data, and performing data augmentation to get higher accuracy. As a conclusion, current deep CNN models can be used to automatically detect distracted drivers and can be integrated with different systems to warn the driver. However, the main constrain for distracted driver detection using a dashboard camera from the drivers' viewpoint is the driver's privacy. The future work is first investigating if CNN models are able to recognize distracted behaviors using images from a dashboard cameras that captures the drivers from different angles, and second combining distracted driver detection with drowsy driver detection using one classification model .

References

[1] National Center for Statistics and Analysis, *Distracted Driving 2014 (Traffic Safety Facts Research Note)*. No. DOT HS 812 260, National Highway Traffic Safety Administration (NHTSA), April 2016.

[2] C. Carney, D. McGehee, K. Harland, M. Weiss, and M. Raby, *Using Naturalistic Driving Data to Assess the Prevalence of Environmental Factors and Driver Behaviors in Teen Driver Crashes*. AAA Foundation for Traffic Safety, March 2015).

[3] Centers for Disease Control and Prevention, "Distracted driving," November 2016. `http://www.cdc.gov/motorvehiclesafety/distracted_driving/`.

[4] F. Tango and M. Botta, "Real-time detection system of driver distraction using machine learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 894–905, 2013.

[5] Y. Yun, I. Y. Gu, M. Bolbat, and Z. H. Khan, "Video-based detection and analysis of driver distraction and inattention," in *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*, pp. 190–195, IEEE, 2014.

[6] J. Pohl, W. Birk, and L. Westervall, "A driver-distraction-based lane-keeping assistance system," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 221, no. 4, pp. 541–552, 2007.

[7] C. Blaschke, F. Breyer, B. Färber, J. Freyer, and R. Limbacher, "Driver distraction based lane-keeping assistance," *Transportation research part F: traffic psychology and behaviour*, vol. 12, no. 4, pp. 288–299, 2009.

[8] State Farm Corporate, "State farm distracted driver detection," April 2016. Competition website is Kaggle.com, `https://www.kaggle.com/c/state-farm-distracted-driver-detection`.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, pp. 1–2, Prague, 2004.

[14] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "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.), pp. 1097–1105, Curran Associates, Inc., 2012.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Iclr*, pp. 1–14, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[20] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[21] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[22] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1058–1066, 2013.

[23] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer vision–ECCV 2014*, pp. 818–833, Springer, 2014.

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[25] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc.*

ICML, vol. 30, p. 1, 2013.

[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

[27] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[28] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber, "Compete to compute," in *Advances in Neural Information Processing Systems*, pp. 2310–2318, 2013.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

[30] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms." `http://www.vlfeat.org/`, 2008.

[31] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[32] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[33] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," in *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

Author Biography

Murtadha Hssayeni is a graduate student in Computer Engineering at Rochester Institute of Technology (expected 2017). He received his BS in the same field from University of Technology-Baghdad (2012). Currently, he is developing a method to automatically assess the medication states of patients with Parkinson's disease using wearable sensors. His research interests are in the area of signal processing and machine intelligence.

Sagar Saxena received his Bachelor of technology (B.Tech) in Electronics and Communication Engineering from Jaypee Institute of Information Technology, India (2015) and is now pursuing his Masters in Computer Engineering at Rochester Institute of Technology, New York (expected 2017). His research interests mainly lie in ASIC / VLSI design with a particular focus on design and verification of modern System on Chips (SoC) along with aspects of Computer Vision.

Raymond Ptucha is an Assistant Professor in Computer Engineering and Director of the Machine Intelligence Laboratory at Rochester Institute of Technology. His research specializes in machine learning, computer vision, and robotics. Ray was a research scientist with Eastman Kodak Company where he worked on computational imaging algorithms and was awarded 30 U.S. patents with another 19 applications on file. He graduated from SUNY/Buffalo with a B.S. in Computer Science and a B.S. in Electrical Engineering. He earned a M.S. in Image Science and a Ph.D. in Computer Science, both from RIT. Ray is a passionate supporter of STEM education and is an active member of his local IEEE chapter and FIRST robotics organizations.

Andreas Savakis is Professor of Computer Engineering at Rochester Institute of Technology. He received the BS and MS degrees from Old Dominion University and the PhD from North Carolina State University, all in Electrical Engineering. Before joining RIT he was with the Eastman Kodak Research Labs. His research interests include object detection, visual tracking, activity and expression recognition, scene understanding and deep learning.