Machine Learning Techniques for Distracted Drivers Detection

Shubham Nawani, Shubham Kumar, Shubham Tripathi and Ankita Gupta

Maharaja Agrasen Institute of Technology, Delhi, India

Abstract

Distracted driving causes a large number of traffic accident fatalities and is becoming an increasingly important issue in recent research on traffic safety. Gesture patterns are less distinguishable in vehicles due to in-vehicle physical constraints and body occlusions from the drivers. However, by capitalizing on modern camera technology, convolutional neural network (CNN) can be used for visual analysis. In this paper, we present a multitude of machine learning and deep learning models to detect the behaviors of distracted drivers and demonstrate the comparison between them. We use Decision Trees, Random Forests, Support Vector Machines and AdaBoost as our choice of machine learning algorithms and ResNet101, VggNet16 and InceptionV3 as our choices for deep learning models. We extract the driver behavior features based on transfer learning and apply augmentation to make it robust for unseen data. We apply extra layers to learn more complex features and use dropout to prevent the deep learning models from overfitting to the training data. The experimental results show that from machine learning models, Random Forest achieves the best classification accuracy of 99.89% and from deep learning models VGGnet16 achieves an accuracy of 90.65% when detecting distracted driving behaviors, demonstrating that it can potentially help drivers maintain safe driving habits.

I. Introduction

According to the report from World Health Organization (WHO) [1] in 2020, approximately 1.35 million people worldwide have died from traffic accidents each year. Losses from road traffic collisions are equal to approximately 3% of the GDP in most countries.

Driving a car is a complex task, and it requires complete attention. Distracted driving is any activity that takes away the driver's attention from the road. Several studies have identified three main types of distraction: visual distractions (driver's eyes off the road), manual distractions (driver's hands off the wheel) and cognitive distractions (driver's mind off the driving task). Research shows that, more than 90% of traffic accidents are attributable to driver error [3], including the dominating factors such as fatigue, distraction, and drunkenness. Among these, distracted driving has become an increasingly important cause of traffic accidents in recent years. For example, distracted driving can cause wrong lane changes, which will lead to serious traffic accidents [4], [5]. According to the preliminary definition of the International Organization for Standardization (ISO) [6], distracted driving is "attention given to a non-driving related activity, typically to the detriment of driving performance" The NHTSA has defined distracted driving as "an activity that could divert a person's attention away from the primary task of driving". [7]. The current popularity of on-board electronics such as navigation systems and smart phones has introduced additional factors that induce distracted driving behavior by drivers. Therefore, it is necessary to carry out in-depth research on distracted driving behaviors, determine their occurrence mechanisms, and propose corresponding solutions. By doing so, we can reduce the frequency of distracted driving behaviors and improve driving safety. Many states now have laws against texting, talking on a cell phone, and other distractions while driving. We believe that computer vision can augment the efforts of the governments to prevent accidents caused by distracted driving.

To detect distracted driver behaviors, several approaches have been proposed based on physiological parameters, vehicle driving state and vision [8]. There are two requirements for a driving behavior detection system. First, regardless of the detection approach adopted, drivers should not be affected by the measuring instruments. Although approaches based on physiological parameters can achieve high accuracy, such measurements require numerous instruments that may disturb the driver. Second, current detection methods for driving behaviors consistently lag occurrences. We need to warn the drivers just at the point when they start to get distracted— not warn after their driving behavior has already become abnormal. Computer vision approaches meet the above two requirements. Based on the improvements in computing hardware, camera technology and neural networks, features can be fully extracted from complex images.

Machine learning models such as Support Vector Machines (SVMs), Decision Trees, AdaBoost and Random Forests are widely used to establish a baseline for image detection tasks. These models train faster than their deep learning counterparts but are more prone to being overfit. Deep Learning models such as Convolutional neural networks (CNNs) are widely used for complex image processing tasks. In recent years, various methods, such as AlexNet [9], VGGNet [10], Inception [11], ResNet [12], and DenseNet [13], have been proposed to address problems such as image classification and recognition. These methods can extract features from images and use them to perform classification. However, recognizing distracted drivers' behavior using only one pretrained model is prone to overfitting, which causes detection failures in practice.

In this paper we demonstrate the comparison between the machine learning models – SVMs, Decision Trees, Random Forests & AdaBoost and deep learning models – ResNet101, InceptionV3 and VGGNet16.

First, the images would be pre-processed according to the model specifics. This would ensure that the models learn the maximum features available.

Second, in the feature module, the deep behavior features extracted by the CNN models are deeply fused into a set of one-dimensional vectors. These fused features contain high-level semantics and lay the foundation for the subsequent image classification.

Third, in the feature classification module, the neurons in the fully connected layer of the models capture the key elements of the fused feature vectors and use them as basis for classification. The fused features are classified into different distracted driving behaviors.

Our main contributions are summarized below.

(1) We compare the machine learning techniques with deep learning techniques and compare the two on the basis of training accuracy, validation accuracy and log loss. The best model can then be used to adapt the confined space in the vehicle without disturbing the driver and make accurate decisions in real time.

(2) We implement image augmentation and dropout techniques to make our models robust to unseen data and less prone to being overfit respectively.

(3) The experimental results show that, the random forest classifier does a great job and is the best Machine Learning model with a classification accuracy of 99.89% and a log loss of 0.04 While, the VGG model is the best Deep Learning model with a classification accuracy of 90.65% and a log loss of 0.28.

The remainder of this paper is organized as follows. In Section 2, we introduce prior related works. The detailed design of the all the models for the detection of distracted drivers is provided in Section 3. Model evaluation and optimization is described in Section 4. Our experimental results and analysis are reported in Section 5, and the final section presents conclusions and the outlook for future works.

II. Related Works

This section summarizes the academic research on distracted driving detection in done recent years. Mobile phones are one of the major causes of driver distraction. Beck and Park [14] compared the impact of editing text messages on traditional mobile phones and smartphones on road traffic safety in a simulated driving environment. The results showed that smartphones increase driving risks as measured by key performance parameters. Esfahani et al. [15] used a driving simulator method to study the impact of reading and texting on driving performance. Studies have shown that such behaviors result in significant reductions in driving safety and should be prohibited. Hoang et al. [16] proposed an automatic system that determined the driver's actions through a dashboard-mounted camera. The observed features are input into a hidden conditional random field (HCRF) model. This method is able to effectively recognize mobile phone usage by drivers.

Tran et al. [17] used pretrained models to extract distracted drivers features and adopted a support vector machine (SVM) as a classifier. They compared the performances of CNNs for extracting the features input into the SVM to detect distracted driver behaviors. Chawan et al. [18] used VGG16, VGG19 and Inception models to classify distracted drivers. Baheti et al. [19] proposed a modified VGG model that used regularization techniques, their approach achieved a classification accuracy of 95.54%.

Yang et al. [20], [21] proposed a driving-related recognition system based on the deep CNN model. They used Kinect cameras to collect images of distracted drivers and the raw images are processed with a GMM-based segmentation algorithm. Then CNN models are used as a binary classifier which achieves 91% accuracy.

Yan [22] proposed a CNN-based model that used unsupervised learning and transfer learning to classify

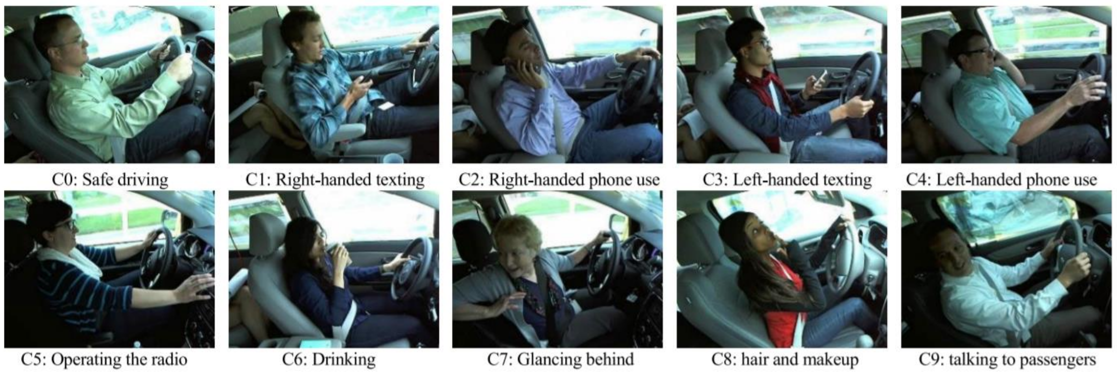


FIGURE 1. Ten common driver behaviors while driving.

activities of normal driving, answering a cellphone call, eating, and smoking. Ramzan et al. [23] proposed using a hierarchical classification approach and treating driving behavior in spatio-temporal reference frame terms rather than as a static image. The overall prediction accuracy for this method reached 89.62%. Shahverdy et al. [24] proposed a deep learning-based method for acquiring distracted driving data and constructed an analysis system called DarNet, which achieved a classification accuracy of 87.02% on their collected dataset.

Kaggle organized a competition to classify driver behavior images [25] because early distracted driving datasets included only a few categories that covered common distracted driving behaviors. In contrast, the Kaggle dataset contains ten different distracted driver behaviors while driving (see Fig. 1). Majdi et al. [26] proposed Drive-net, a method that uses a combination of a CNN and a random decision forest to classify driver images. Driver-net achieved a detection accuracy of 95% on the Kaggle dataset. Ou and Karray [27] proposed a driver distraction recognition system that used generative adversarial networks (GANs) and demonstrated that generative models can generate images of drivers in different driving scenarios. By using these images to augment training, they improved the system’s image classification performance by 11.45%.

Currently, computer vision techniques are widely used to extract key features from images. Such classification tasks are completed by various deep neutral network models running on high-performance computers to achieve better recognition accuracy.

III. Research Methodology

A. PRE-PROCESSING

Image pre-preprocessing is done as follows:

(1) Images in the dataset have very high resolutions (640×480×3), and in order to improve the computational efficiency for the machine learning models (SVMs, Random Forest, AdaBoost and Decision Trees), we will preprocess the images by resizing them to (64×64×3), followed by flattening the high dimensional image matrix to image vectors as the input to train the classifiers.

(2) Before inputting the training image to the Deep Learning models, we need to preprocess the original images (640×480×3) to satisfy the input requirements of ResNet101 (224×224×3), InceptionV3 (299×299×3) and VGG-16 (224x224x3),

B. MACHINE LEARNING MODELS

In this section we introduce all the machine learning models that we will use for comparison.

1) Support Vector Machines (SVMs)

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.

The Linear SVM classifier is as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max\big(0, f(x_i,W)_j - f(x_i,W)_{y_i} + 1\big) + \lambda||W||_2^2$$

The flattened image features are passed to the Linear SVM classifier. Different learning rates and regularization strengths were used to help us find the rates that would give us the best validation accuracy during cross-validation.

2) Decision Trees

A decision tree (also referred to as a classification tree or a reduction tree) is a predictive model which is a mapping from observations about an item to conclusions about its target value. In the tree structures, leaves represent classifications (labels), non-leaf nodes are features, and branches represent conjunctions of features that lead to the classifications. If the dataset consists of N attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. To solve such problem, we use the following criteria:

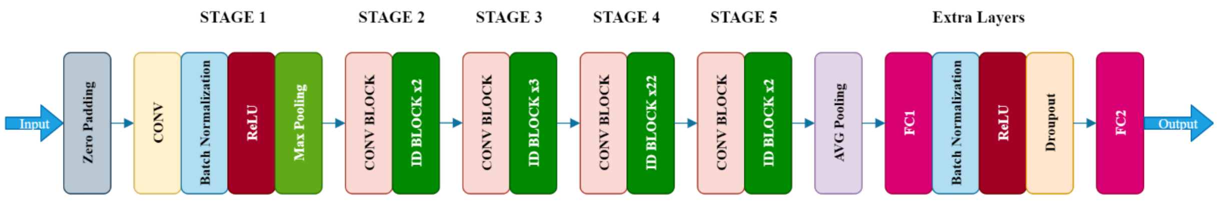


FIGURE 2. **The architecture of ResNet101**

i) Entropy: is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. It is represented as:

$$E(S) = \sum_{i=1}^{c} -p_i log_2\, p_i$$

ii) Information gain: is a statistical property that measures how well a given attribute separates the training examples according to their target classification. It is represented as:

$$Information\ Gain\ = Entropy(before) - \sum_{j=1}^{K} Entropy(j, after)$$

iii) Gini index: It is calculated by subtracting the sum of the squared probabilities of each class from one. It favors larger partitions and easy to implement whereas information gain favors smaller partitions with distinct values. It is represented as:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

3) AdaBoost

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm which is used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. We have combined AdaBoost with the Decision tree algorithm. The performance of each stump is given as:

$$Performance\ of\ Stump = \frac{\frac{1}{2}\ln(1 - TE)}{TE}$$

And new sample weight is given as:

$$New\ Weight = Sample\ Weight * e^{performance}$$

4) Random Forests

A random forest is an ensemble method that consists of multiple decision trees. A single decision tree has overfitting problems as a tree grows deeper, Random forests address this issue by constructing multiple decision trees. Each decision tree uses a randomly selected subset of training data and features. The output is calculated by averaging the individual decision tree predictions.

Decisions trees are very sensitive to the data they are trained on — small changes to the training set can result in significantly different tree structures. Random forest takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees.

C. DEEP LEARNING MODELS

In this section we introduce all the deep learning models that we will use for comparison.

1) ResNet101 Model

The ResNet101 model effectively solves the problem of the increase in training errors caused by increasing the number of layers in a neural network. ResNet101 includes 5 main stages and another stage with extra layers. Fig. 3 shows the architecture of the ResNet101 model.

The steps of the training procedure for ResNet101 are as follows:

First, in stage 1, a convolutional kernel of 7 × 7 is used to extract image features. Batch normalization, activation and max pooling are completed, and the output of stage 1 is a 55 × 55 × 64 feature map in the CNN.

Second, from stages 2–5, two types of residual blocks are added to the CNN. A CONV block is a scaled residual block. Each CONV Block includes three convolutional kernels of 1×1 or 3×3. An ID block represents a residual block that does not change size. The sizes of the output feature maps from stages 2–5 are 55×55×256, 28×28×512, 14×14×1024 and 7 × 7 × 2048, respectively.

Third, after the five stages, a global average pooling layer (AVG Pooling) with a kernel size of 7×7 is used to convert the output feature map into a 2,048-dimensional feature vector. This feature vector is input into the extra layers, which consists of a fully connected dense layer of 1,024 features followed by batch normalization and ReLU activation layer.

Finally, dropout technology is adopted to reduce overfitting on the training data, followed by another fully connected layer of 512 features. For classification we use the softmax classifier [31] to produce a one-hot vector consistent with the probabilities of the ten types of districted driving behaviors in the Kaggle dataset. After which the classification results of the ten types of distracted driving behaviors are obtained.

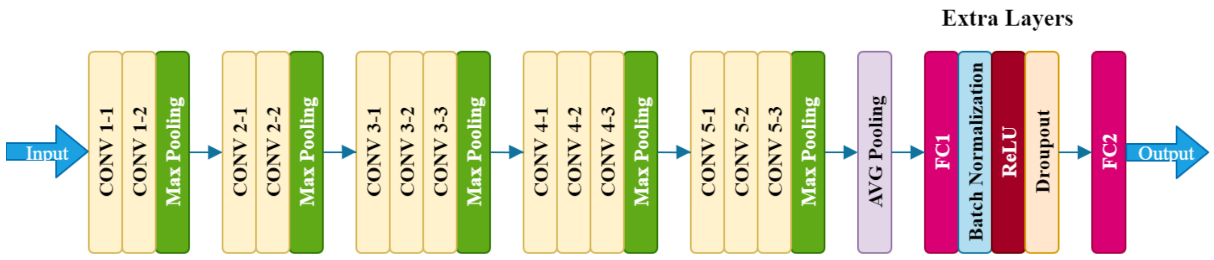


FIGURE 3. The architecture of VGG16

2) VGG16 Model

The VGG16 model is an improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. Figure 3 show the VGG16 architecture with extra layers.

The training procedure steps for VGG16 are as follows:

First, input image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3. The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers. Max-pooling is performed over a 2×2 pixel window, with stride 2. A feature map of 7 × 7 × 512 can be obtained.

Second, a global average pooling layer with a kernel size of 7×7 is used to convert the output feature map into a 512 -dimensional feature vector.

Finally, Two Fully-Connected (FC) layers follow with dropout follow this output feature map having 1,024 and 512 channels each. For classification we use the softmax classifier after which the classification results of the ten types of distracted driving behaviors are obtained.

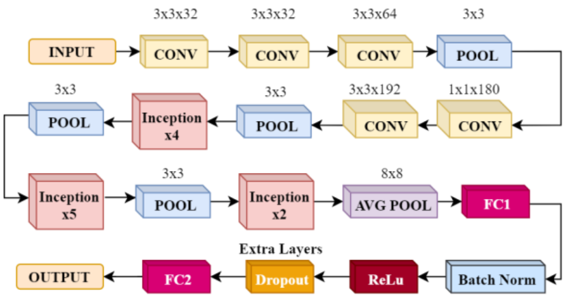


FIGURE 4. The architecture of InceptionV3

3) InceptionV3 Model

Inception V3 clusters the sparse convolution kernel structure into multiple dense sub-convolution kernel combinations. Fig. 4 shows the architecture of Inception V3. The core units (the pink blocks in Fig. 4) of Inception V3 are shown in Fig. 5. Convolutional kernels of different sizes, (e.g., 1×1, 3×3 and 5 × 5) are used to obtain different receptive fields.

The training procedure steps for Inception V3 are as follows:

First, for the input image, 3 convolution layers with kernels of 3 × 3 and one max pooling layer are used to extract low latitude features of the image. Then two convolution layers with kernels of 1×1 and 3×3 and one max pooling layer are used to further extract features. A feature map of 35×35×192 can be obtained.

Second, the 11 core units of Inception V3 and 3 pooling layers are used to extract high-dimensional features. A feature map of 8 × 8 × 2048 can be obtained.

Finally, we obtain a 2,048-dimensional feature vector from the feature map. This feature vector which is followed by our extra fully connected layers and followed by the classification module.

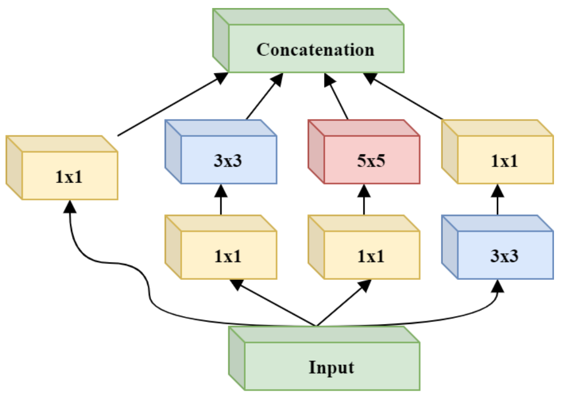


FIGURE 5. Core unit of InceptionV3

IV. Model Evaluation and Optimization

We use the loss function to evaluate the performance of all the models during training. The loss function evaluates the error between the output of the models and the given target value. Meanwhile, to speed up training, we use the Adam algorithm [11].

A. LOSS FUNCTION FOR PERFORMANCE EVALUATION

We generally evaluate the performance of deep learning CNN models by the numerical magnitude of the loss function.

To improve model generalizability, regularization terms are added to the loss function $\tilde{J}(\theta)$ as follows:

$$\tilde{J}(\theta) = J(\theta) + a\phi(\theta)$$

where θ is the measurable state. The difference between the real value and the output value is represented as J(θ), which is the standard loss function. φ(θ) is the regularization term, a ∈ [0,∞] is the regularization term coefficient, and a is used to measure the weight of the regularization term in the loss function $\tilde{J}(\theta)$. The larger a is, the greater the weight of the regularization term is.

The commonly used loss functions are listed below:

1) THE MEAN SQUARED LOSS FUNCTION

$$J = \frac{1}{2}\|y - \hat{y}\|_2^2$$

where y is the real value, $\hat{y}$ is the actual output value of the model and J is the mean squared loss function.

2) CROSS ENTROPY LOSS FUNCTION

In binary classification, the number of classes M equals 2, and the cross-entropy loss function can be described as follows:

$$J = \frac{1}{N}\sum_{i=1} -y_i \,.\, log(p_i) + (1 - y_i).\, log(1 - p_i)$$

where N is the number of samples, y_i is the real value of sample *i*, $\hat{y}$ is the output value of sample *i*, and p_i is the probability that sample *i* is positive.

When M > 2 (i.e. multiclass classification), we calculate a separate loss for each class label and sum the result:

$$J = \frac{1}{N}\sum_{i} -\sum_{c=1}^{M} y_{ic} \log(p_{ic})$$

where M is the number of classes, and yic is the binary indicator (0 or 1). If class c is the correct classification for observation *i*, p_{ic} refers to the predicted probability observation *i* of class c.

3) EXPONENTIAL LOSS FUNCTION

The exponential loss function is:

$$J = \frac{1}{N}\sum_{i=1}^{N} e^{-y_i\hat{y}_i}$$

4) ABSOLUTE LOSS FUNCTION

$$J = |y - \hat{y}|$$

These functions have different metrics for the target deep learning method. When the partial derivative is small, the parameter updating rate of the mean squared loss function becomes slow. Then, we can choose the cross-entropy loss function to evaluate the performance of the proposed models.

B. OPTIMIZER FOR PERFORMANCE EVALUATION

We generally evaluate the performance of deep learning CNN models by the numerical magnitude of the optimizers.

1) GRADIENT DESCENT

Gradient Descent is the most basic but most used optimization algorithm. It's used heavily in linear regression and classification algorithms. Backpropagation in neural networks also uses a gradient descent algorithm.

The update step is given as:

$$\theta = \theta - \alpha \cdot \nabla J(\theta)$$

Gradient descent is a first-order optimization algorithm which is dependent on the first order derivative of a loss function. It calculates that which way the weights should be altered so that the function can reach a minima. Through backpropagation, the loss is transferred from one layer to another and the model's parameters also known as weights are modified depending on the losses so that the loss can be minimized.

2) STOCHASTIC GRADIENT DESCENT

It's a variant of Gradient Descent. It tries to update the model's parameters more frequently. In this, the model parameters are altered after computation of loss on each training example.

The update step is given as:

$$\theta = \theta - \alpha \cdot \nabla J(\theta; x(i); y(i)) ,$$

where {x(i) ,y(i)} are the training examples.

As the model parameters are frequently updated parameters have high variance and fluctuations in loss functions at different intensities.

3) ADAM

Adam (Adaptive Moment Estimation) works with momentums of first and second order. The intuition behind the Adam is that we don't want to roll so fast just because we can jump over the minimum, we want to decrease the velocity a little bit for a careful search. Adam also keeps an exponentially decaying average of past gradients M(t).

M(t) and V(t) are values of the first moment which is the Mean and the second moment which is the uncentered variance of the gradients respectively.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}.$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}.$$

Here, we are taking mean of M(t) and V(t) so that E[m(t)] can be equal to E[g(t)] where, E[f(x)] is an expected value of f(x).

To update the parameter:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t.$$

The values for β1 is 0.9, 0.999 for β2, and 10^{-8} for ∈ We are choosing Adam as our choice of optimizer for our deep learning models.

V. Experimental Results and Analysis

A. THE EXPERIMENTAL ENVIRONMENT

1) DATA COLLECTION AND EXPLORING THE DATASET

The dataset of distracted driving behaviors comes from the State Farm insurance company for a Kaggle challenge, which includes 22,424 images for training and 79,726 images for testing. Because the testing set images are unlabeled, we perform our experiments on the training set. The size of each image is 640 × 480. The dataset includes the images of 26 drivers, and the distracted driving behaviors in the dataset are divided into ten classes.

The ten classes of distracted driving behaviors include: safe driving, right-handed texting, right-handed phone use, lefthanded texting, left-handed phone use, operating the radio, drinking, glancing behind, hair and makeup, and talking to passengers. We label these ten classes of distracted driving behaviors as c0–c9, the number of samples for each driver, and the number of distracted driving behavior samples in each class are shown in Figs. 7 and 8, respectively. The number of distracted driving behaviors samples for each class and the number of samples for each driver are roughly even, which is beneficial for our research. We can train the models with this dataset to achieve more accurate detection of distracted driving behaviors.

We selected images of some of the drivers (P015, P016, P022, and P050) as a verification set (total: 4,320 samples), and the other images are included in the training set (total: 18,104 samples). The ratio of images in the training set and the validation set was kept at approximately 8:2.

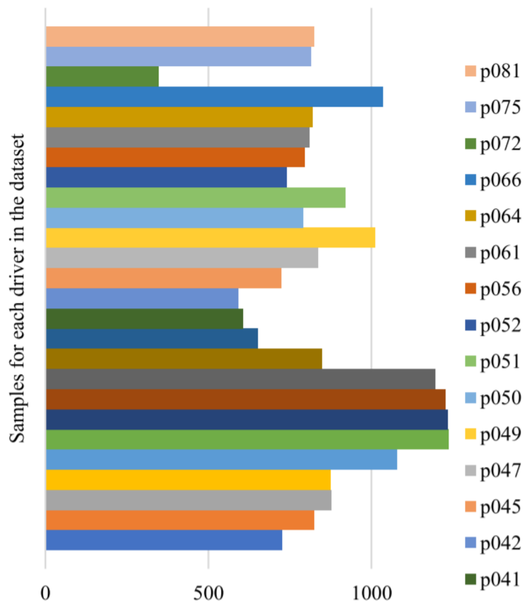


FIGURE 6. Number of samples associated with each driver in the dataset.

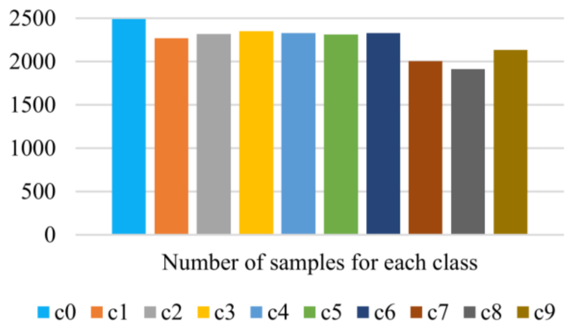


FIGURE 7. Number of samples of distracted driving behavior for each class in the dataset

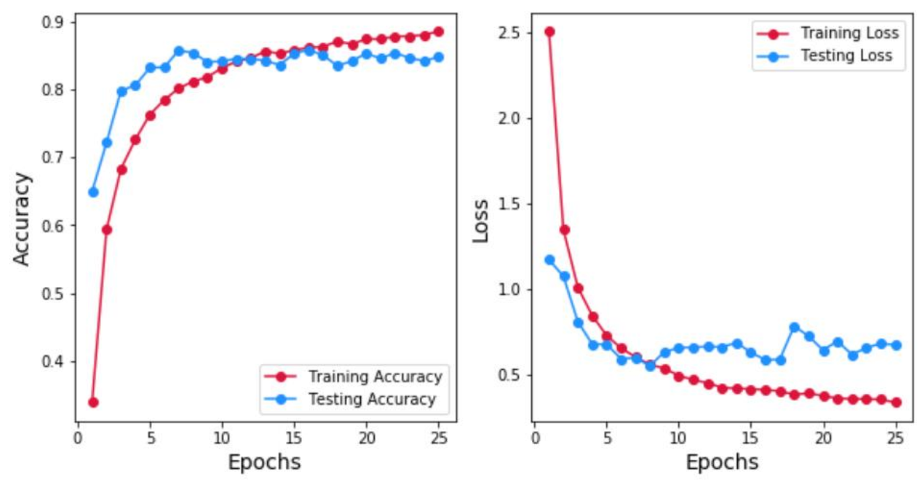
2) EXPERIMENTAL SETUP

The models were tested on Google colaboratory with Intel(R) Xeon(R) CPU @ 2.20GHz, 12.72 GB of RAM, and a Tesla T4 GPU running the Ubuntu 18.04 operating system. The code was mainly implemented in the Python language. For machine learning we used the SciKit Learn library and for deep learning we adopted the deep learning Keras and TensorFlow frameworks [32].

B. TRAINING STRATEGY

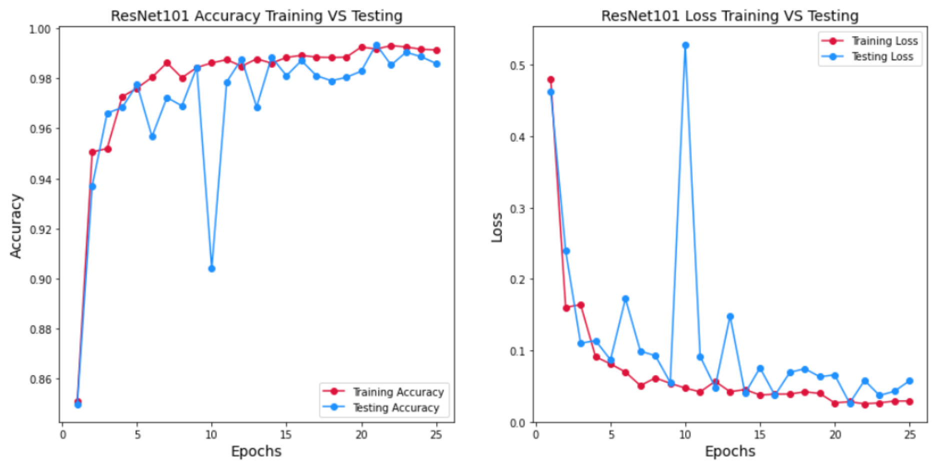
1) MACHINE LEARNING MODELS

We used Scikit learn to implement the machine learning models of Decision Trees, Random Forest, and AdaBoost with Linear SVC implemented by us for a baseline model.



(a) Accuracy with varying epochs (b) Loss with varying epochs

FIGURE 8. The classification results using VGG16



(a) Accuracy with varying epochs (b) Loss with varying epochs

FIGURE 9. The classification results using ResNet101

Cross-validation was done to ensure the results are fair and accurate. Thus, we computed the validation accuracy, cross validation mean accuracy, log loss and time taken for all the models. Max features for Decision Trees, Random Forest and AdaBoost was set to auto with KFold splits set to 5 and the max_iters for Linear SVC was set to 1500.

2) DEEP LEARNING MODELS

The parameters of the ResNet101, Inception V3 and VGG16 models were pretrained on ImageNet, and we transferred the parameters directly to our distracted behavior detection task. However, the features of the State Farm dataset are different from those of the ImageNet dataset.

Thus, these pretrained models must be fine-tuned to adapt them to the State Farm dataset.

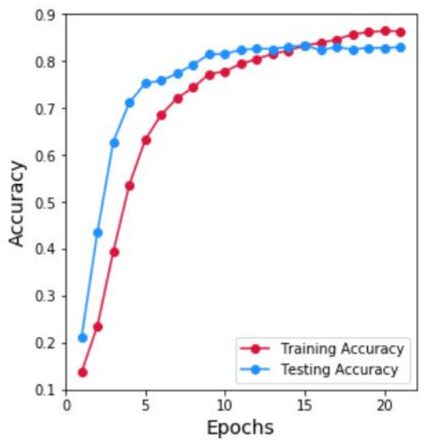
The strategy for training is as follows:

(1) ResNet101: For the pretrained ResNet101, we froze the weights from layers 0 to 262 and trained only the weights from layer 262 to the top layer.

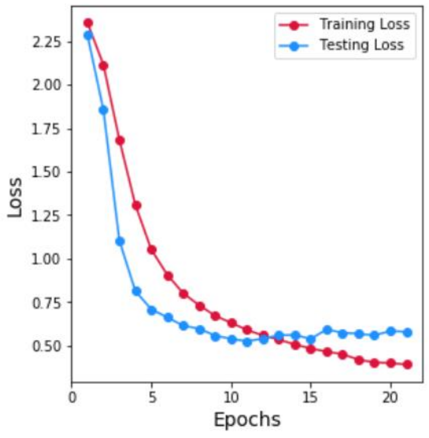
(2) Inception V3: We froze the weights of the pretrained Inception V3 layers 0 to 171 and trained only the weights from layer 172 to the top layer.

(3) VGG16: We froze the weights of the pretrained VGG16 from layers 0 to 17 and trained the weights from layer 17 to the top layer.

During training, we used the Adam algorithm and set the initial learning rate to 0.001. Image Augmentation was used to make the models more robust to unseen testing data. The image batch size was set to 64, and each model was trained for 25 epochs on the GPU. We applied dropout to reduce overfitting. A softmax classifier is used to complete the feature classification, and the final output is the



(a) Accuracy with varying epochs



(b) Loss with varying epochs

FIGURE 10. The classification results using InceptionV3

probability corresponding to the ten classes of distracted driving behaviors in the State Farm dataset.

C. COMPARISION OF MODELS

1) THE LOSS AND ACCURACY

Among the four machine learning models, Random Forests achieves the highest accuracy and the lowest loss. Decision Trees and AdaBoost are extremely time efficient but fall short on their log losses.

For the task of distracted behavior detection, Decision Trees achieves a Cross Validation Mean Accuracy of 82.77% on the training set and Validation Accuracy of 84.48% on the validation set. The test set accuracy for the selected drivers is 96.53% but the log loss on this model is 1.2. For LinearSVC, we get a training accuracy of 72.82% and validation accuracy of 71.39% with test accuracy of 58.04%

For AdaBoost, these two accuracy rates are 83.08% and 84.88%, respectively with test accuracy as 96.37% and log loss as 1.25

For our Random Forest classifier, the two accuracy rates are 99.13% and 99.28% with the test accuracy as 99.89% and log loss as 0.04

Among the four deep learning models, VGG achieves the highest test accuracy and the lowest log loss. ResNet is faster to train than the InceptionV3 model.

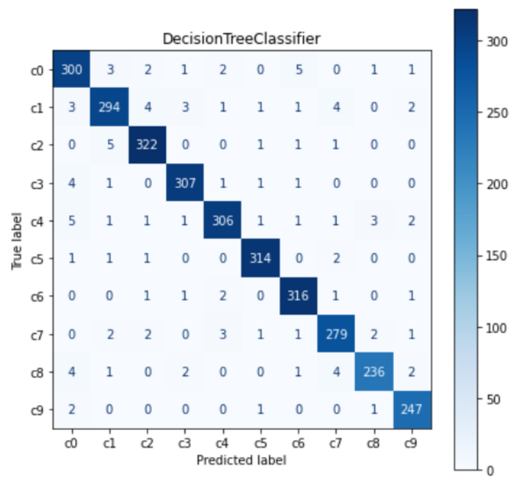
For the task of distracted behavior detection, ResNet101 achieves an accuracy of 98.72% on the train set and 98.59% on the validation set with an accuracy of 89.46% on the test set and log loss of 0.39. For InceptionV3, these accuracy rates are 88.76% and 83.65%, respectively, with test accuracy as 87.9% and log loss of 0.56 and for VGG16, they are 83.59% and 83.31%, respectively, with testing accuracy as 90.65% and log loss as 0.28

Thus, the Random Forest classifier and the deep learning models are able to recognize distracted driving behaviors in the State Farm dataset. Furthermore, the extremely low loss and high accuracy of Random Forest suggest that the model is overfitted to this particular dataset and with no augmentation done during the training phase for this model. It works exceptionally well on the test set from the same dataset.

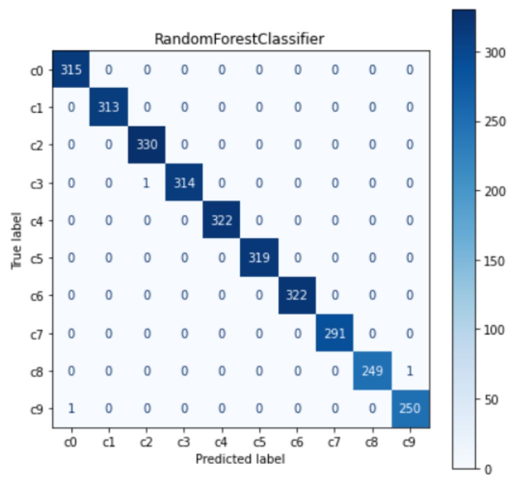
The Deep learning models on the other hand are more robust to unseen data, thus performing better than most of the Machine learning algorithms, giving high accuracies and low loss.

2) THE CONFUSION MATRIX

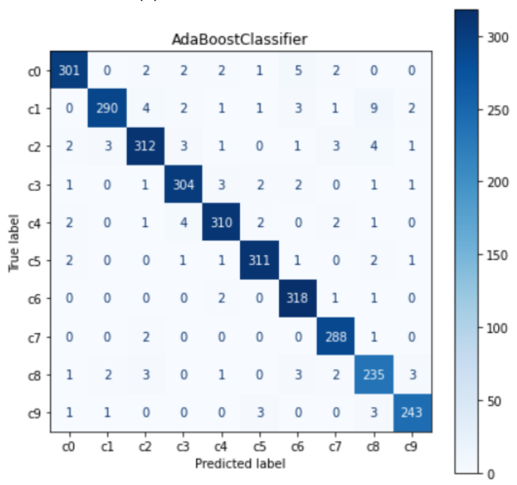
Fig. 13 illustrates the confusion matrixes for ten classes of distracted driving behaviors using the machine learning and deep learning models. The dark blue diagonal shows the percentage of correctly detected cases for each class. The leftmost column shows the actual label of each class, and the top row shows the output label. From the confusion matrixes in Fig. 11, it can be observed that for a test set derived from training data, machine learning models do exceedingly well to predict the correct classes, it is only after seeing the log loss, we could see these models are tending to overfit. Additionally, it shows that the deep learning models (as shown in Fig 12) are more prone to mis-classify distracted behaviors as safe driving, but in practical use with inconsistent data set, since trained on augmented data, these models are more likely to predict correctly than their machine learning counterparts. The Random Forest classifier never mistook a distracted behavior as ‘‘Safe Driving’’; thus, it is more reliable in practice, but can face issues in unseen data.



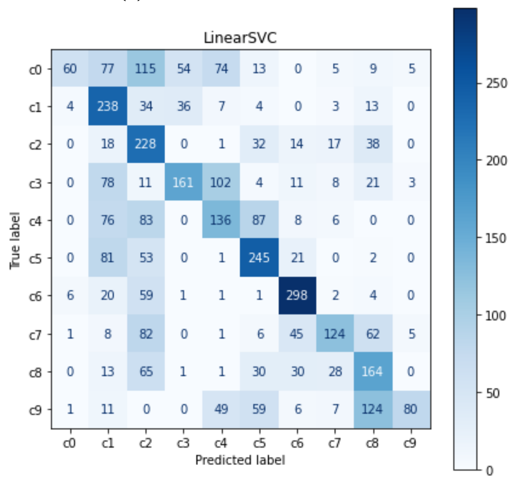
(a) Decision Tree Classifier



(c) Random Forest Classifier



(b) AdaBoost Classifier



(d) Linear SVC

FIGURE 11. Confusion matrix of Machine Learning models

```
[399,   0,   0,   1,   0,   0,   0,   9,   0,   3]
[  4, 127,   0,   3,   0,  15,   6,  37,  80,  67]
[  0,   0, 308,   0,   0,   1,   1,  20,   3,  15]
[  0,   0,   0, 397,   2,   0,   0,   0,   0,   0]
[  1,   0,   0,   0, 395,   0,   0,   0,   0,   0]
[  3,   0,   0,   0,   0, 400,   0,   0,   0,   0]
[  0,   0,   0,   0,   2,   5, 338,   0,   1,  46]
[  0,   0,   0,   0,   0,   0,   0, 334,   0,   0]
[  0,   0,   0,   0,   0,   0,   2,  21, 278,  31]
[  0,   0,   0,   0,   0,   1,   0,   3,   6, 327]
```

(a) ResNet101 Model

```
[[395,   7,   0,   0,   2,   0,   0,   0,   0,   8],
 [  2, 337,   0,   0,   0,   0,   0,   0,   0,   0],
 [  0,  83, 222,   0,   0,   0,  37,   0,   0,   6],
 [  1,  72,   0, 326,   0,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0, 348,   0,  48,   0,   0,   0],
 [  0,   0,   0,   0,  11, 311,  79,   1,   1,   0],
 [  0,   0,   0,   0,   0,   0, 392,   0,   0,   0],
 [  0,  50,   0,   0,   0,   0,   0, 284,   0,   0],
 [  0,   6,   0,   0,   0,   0,   5,   3, 315,   3],
 [  4,   3,   0,   0,   0,   0,  10,   0,   3, 317]]
```

(c) InceptionV3 Model

```
             VGG16 Model
[[404   0   0   0   0   7   0   0   0   1]
 [  0 248   0   0   0   0  87   0   4   0]
 [  0   0 302   0   0   7  19   1  19   0]
 [ 10   0   0 384   0   5   0   0   0   0]
 [  3   0   0   1 342  50   0   0   0   0]
 [  1   0   0   0   0 402   0   0   0   0]
 [  1   0   0   0   2  17 357   0  15   0]
 [  0   0   0   0   3   0   0 331   0   0]
 [  1   0   0   0   2   2   0  18 289  20]
 [ 14   0   0   0   0  30   0   4   1 288]]
```

(b) VGG16 Model

FIGURE 12. Confusion matrix of Deep Learning models

D. DISCUSSION

1) IMPACT OF DATASET DIVISION

First, we randomly selected 80% of the dataset as the training set and used the remaining 20% as the validation set. This approach can introduce overfitting on the validation set because the dataset consists of a sequence of frames drawn from the video, and randomly dividing the dataset into an 8:2 ratio causes the driver images in the validation set and the training set to be highly similar. Fig. 13 shows two different images of one driver in the State Farm dataset. Clearly, these images could easily be confused with each other.

	Model	Training accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)	Log Loss
Machine Learning Models	Linear SVM	72.82	71.39	58.04	15.02
	Decision Tree	82.77	84.48	96.53	1.2
	AdaBoost	83.08	84.88	96.37	1.25
	Random Forest	99.13	99.28	99.89	0.04
Deep Learning Models	ResNet101	98.72	98.59	89.46	0.39
	VGG16	83.59	83.31	90.65	0.28
	InceptionV3	88.76	83.65	87.9	0.28

TABLE 1. Accuracy and Loss of different machine learning and deep learning models

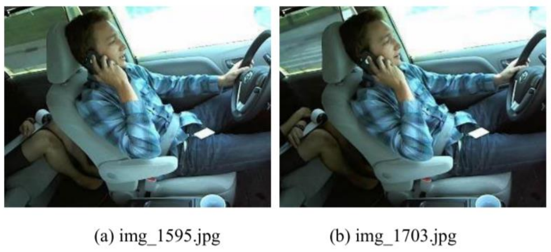


FIGURE 13. Different images of one driver in the dataset.

Hence, we adopted the images numbered P015, P016, P022, and P050 as the verification set (total: 4,320) and included the other images in the training set (total: 18,104). In this way, the images in the training set and validation set are not as apt to be confused with each other.

2) IMPACT OF TRAINING STRATEGY

We used transfer learning to fine-tune the three pretrained deep CNN models. In transfer learning, the finetuning mainly focuses on tuning only a few layers while preserving the main characteristics of the pretrained convolutional layers. In these experiments, we tested different numbers of frozen layers to find the best training strategy.

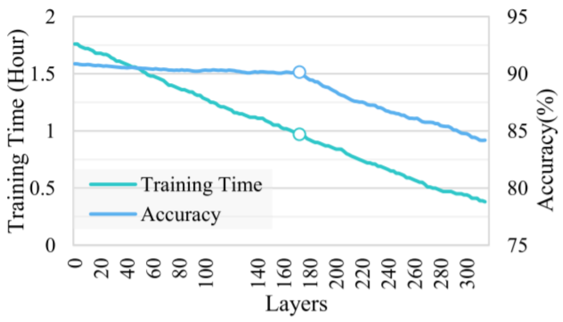


FIGURE 14. Training strategy of Inception V3.

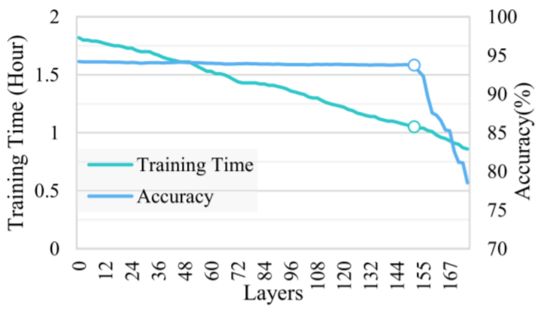


FIGURE 15. Training strategy of ResNet101.

The experimental results are shown in Figs. 14–15. When the number of frozen layers is small, training requires more time. However, the performance of the single pretrained models do not improve as the training time increases. Conversely, when the number of frozen layers is large, the accuracy decreases.

To consider the balance between the training time and the accuracy, we chose 302, 171 and 17 as the number of frozen layers for ResNet101, Inception V3 and VGG16, respectively.

VI. Conclusion and Future Work

Distracted driving behaviors are a primary cause of traffic accidents. Hence, it is necessary to find methods to effectively identify distracted driving behaviors. In this paper, we compared the machine learning and deep learning ideologies to recognize distracted driver behaviors. Features are extracted at different scales for the models. Subsequently, we add additional layers to extract more features with a fully connected layer to classify each distracted driving behavior.

During the training procedure, we apply dropout technology to prevent the training model from overfitting to the training data. The results show that the Random Forest classifier achieves the best performance for recognizing distracted driver behaviors, reaching a classification accuracy of 99.89%.

The dataset in this paper only provides the images from the right-hand side. However, the performance of the even the best machine learning model may degrade when the camera is set in different places in the vehicle. In addition, the models are greatly influenced by the light, which indicates that the models cannot detect the distracted driving behavior accurately at night.

The deep learning models even though not as accurate as the random classifier do show some promise, since they are trained on augmented data, which would mean, for not optimal data, these models have a higher chance to find success in classifying the classes correctly.

Considering the above limitations, in the future, we will analyze the driving behavior from different camera angles and work towards reducing the computation time and the number of parameters. Additionally, the goal is to not only to recognize distracted driving behaviors but also to prevent them.

REFERENCES

[1] World Health Organization. (2020). Road Traffic Injuries. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-trafficinjuries

[2] W.-C. Yang, Z.-X. Chen, M. Zhang, G.-G. Guo, and W.-B. Zhang, ‘‘The analysis of road condition causes of traffic accidents on mountainous highway and its safety countermeasures,’’ in Proc. CICTP, Jul. 2019, pp. 3784–3796.

[3] M. Chen, Y. Tian, G. Fortino, J. Zhang, and I. Humar, ‘‘Cognitive Internet of vehicles,’’ Comput. Commun., vol. 120, pp. 58–70, May 2018.

[4] Y. Xing, C. Lv, H. Wang, D. Cao, and E. Velenis, ‘‘An ensemble deep learning approach for driver lane change intention inference,’’ Transp. Res. C, Emerg. Technol., vol. 115, Jun. 2020, Art. no. 102615.

[5] Y. Xing, C. Lv, H. Wang, H. Wang, Y. Ai, D. Cao, E. Velenis, and F.-Y. Wang, ‘‘Driver lane change intention inference for intelligent vehicles: Framework, survey, and challenges,’’ IEEE Trans. Veh. Technol., vol. 68, no. 5, pp. 4377–4390, May 2019.

[6] A. D. McDonald, H. Alambeigi, J. Engström, G. Markkula, T. Vogelpohl, J. Dunne, and N. Yuma, ‘‘Toward computational simulations of behavior during automated driving takeovers: A review of the empirical and modeling literatures,’’ Hum. Factors, J. Hum. Factors Ergonom. Soc., vol. 61, no. 4, pp. 642–688, Jun. 2019.

[7] National Highway Traffc Safety Administration Traffc Safety Facts. Accessed: Dec. 19, 2019. [Online]. Available: https://www.nhtsa.gov/risky-driving/distracted-driving

[8] H. Mårtensson, O. Keelan, and C. Ahlström, ‘‘Driver sleepiness classification based on physiological data and driving performance from real road driving,’’ IEEE Trans. Intell. Transp. Syst., vol. 20, no. 2, pp. 421–430, Feb. 2019.

[9] X. Zhang, X. Zhou, M. Lin, and J. Sun, ‘‘ShuffleNet: An extremely efficient convolutional neural network for mobile devices,’’ in Proc. CVPR, Jun. 2018, pp. 6848–6856.

[10] J. Su, D. V. Vargas, and K. Sakurai, ‘‘One pixel attack for fooling deep neural networks,’’ IEEE Trans. Evol. Comput., vol. 23, no. 5, pp. 828–841, Oct. 2019.

[11] X. Yuan, P. He, Q. Zhu, and X. Li, ‘‘Adversarial examples: Attacks and defenses for deep learning,’’ IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[12] W. Li, W. Tao, J. Qiu, X. Liu, X. Zhou, and Z. Pan, ‘‘Densely connected convolutional networks with attention LSTM for crowd flows prediction,’’ IEEE Access, vol. 7, pp. 140488–140498, 2019.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, ‘‘A comprehensive survey on graph neural networks,’’ IEEE Trans. Neural Netw. Learn. Syst., early access, Mar. 24, 2020, doi: 10.1109/TNNLS.2020.2978386.

[14] D. Beck and W. Park, ‘‘Perceived importance of automotive HUD information items: A study with experienced HUD users,’’ IEEE Access, vol. 6, pp. 21901–21909, 2018.

[15] H. N. Esfahani et al., ‘‘Prevalence of cell phone use while driving and its impact on driving performance, focusing on near-crash risk: A survey study in Tehran,’’ J. Transp. Saf. Secur., vol. 1, no. 1, pp. 1–21, 2019.

[16] T. H. N. Le, Y. Zheng, C. Zhu, K. Luu, and M. Savvides, ‘‘Multiple scale faster-RCNN approach to driver's cell-phone usage and hands on steering wheel detection,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2016, pp. 46–53.

[17] D. Tran, H. M. Do, W. Sheng, H. Bai, and G. Chowdhary, ‘‘Real-time detection of distracted driving based on deep learning,’’ IET Intell. Transp. Syst., vol. 12, no. 10, pp. 1210–1219, Dec. 2018.

[18] P. P. M. Chawan, S. Satardekar, D. Shah, R. Badugu, and A. Pawar, ‘‘Distracted driver

detection and classification,'' J. Eng. Res. Appl., vol. 8, no. 4, pp. 60–64, 2018.

[19] B. Baheti, S. Gajre, and S. Talbar, ''Detection of distracted driver using convolutional neural network,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2018, pp. 1032–1038.

[20] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis, and F.-Y. Wang, ''Driver activity recognition for intelligent vehicles: A deep learning approach,'' IEEE Trans. Veh. Technol., vol. 68, no. 6, pp. 5379–5390, Jun. 2019.

[21] Y. Xing, C. Lv, Z. Zhang, H. Wang, X. Na, D. Cao, E. Velenis, and F.-Y. Wang, ''Identification and analysis of driver postures for in-vehicle driving activities and secondary tasks recognition,'' IEEE Trans. Comput. Social Syst., vol. 5, no. 1, pp. 95–108, Mar. 2018.

[22] C. Yan, ''Vision-based driver behaviour analysis,'' Univ. Liverpool, Liverpool, U.K., Tech. Rep. 706848, 2016.

[23] M. Ramzan, H. U. Khan, S. M. Awan, A. Ismail, M. Ilyas, and A. Mahmood, ''A survey on state-of-the-art drowsiness detection techniques,'' IEEE Access, vol. 7, pp. 61904–61919, 2019.

[24] M. Shahverdy, M. Fathy, R. Berangi, and M. Sabokrou, ''Driver behavior detection and classification using deep convolutional neural networks,'' Expert Syst. Appl., vol. 149, Jul. 2020, Art. no. 113240.

[25] State Farm Distracted Driver Detection. Accessed: Aug. 2, 2016. [Online]. Available: https://www.kaggle.com/c/state-farm-distracteddriver-detection

[26] M. S. Majdi, S. Ram, J. T. Gill, and J. J. Rodríguez, ''Drive-Net: Convolutional network for driver distraction detection,'' in Proc. IEEE SSIAI, Apr. 2018, pp. 1–4.

[27] C. Ou and F. Karray, ''Enhancing driver distraction recognition using generative adversarial networks,'' in Proc. IEEE Trans. Intell. Vehicles, early access, Dec. 19, 2019, doi: 10.1109/TIV.2019.2960930.

[28] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, ''Semantic understanding of scenes through the ADE20K dataset,'' Int. J. Comput. Vis., vol. 127, no. 3, pp. 302–321, Mar. 2019.

[29] Y. Luo and N. Mesgarani, ''Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation,'' IEEE/ACM Trans. Audio, Speech, Language Process., vol. 27, no. 8, pp. 1256–1266, Aug. 2019.

[30] C. Cangea, P. Veličković, and P. Lio, ''XFlow: Cross-modal deep neural networks for audiovisual classification,'' IEEE Trans. Neural Netw. Learn. Syst., early access, Nov. 8, 2019, doi: 10.1109/TNNLS.2019.2945992.

[31] Y. Luo, Y. Wong, M. Kankanhalli, and Q. Zhao, ''G-softmax: Improving intraclass compactness and interclass separability of features,'' IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 2, pp. 685–699, Feb. 2020.

[32] M. Raissi, P. Perdikaris, and G. E. Karniadakis, ''Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,'' J. Comput. Phys., vol. 378, pp. 686–707, Feb. 2019.