Distracted Driver Detection

Gurpreet Singh
2018csb1092@iitrpr.ac.in
Paras Goyal
2018csb1111@iitrpr.ac.in
Indian Institute of Technology Ropar
Rupnagar, Punjab, India

Abstract

Computer Vision and Deep Learning have wide ranging applications. In this project, we will try to deploy these two fields to ensure the safety of the drivers. This project is based on a Kaggle competetion posted four years ago. More details about the challenge can be found on the challenge page. According to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year.[1]

1 Project Details

1.1 Dataset

The dataset consists of training(22.4k) and testing images(79.7k). The images are are taken from a camera put on the dashboard of the car. Three Channel (RGB) Images are provided for the challenge. The images depict the driver being involved in certain activities. Total Size of the dataset is around 4 GB.

1.2 Objective

The primary objective is to test whether dashboard cameras can automatically detect drivers engaging in distracting activities. We are given driver images, each of which captures the driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc). The goal is to predict the likelihood of what the driver is doing in each picture. The images needs to be classified into the below classes:

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

The least multi-class loss achieved on Kaggle is 8.739%. Apart from three submissions, all others have more than 10% multi-class loss. We aim to attain the best performance possible given the likelihood that we might have to truncate the dataset.

[1] https://www.kaggle.com/c/state-farm-distracted-driver-detection

1.3 Literature Review

There are various methods for distracted driver distraction[14]:
Thresholding : The simplest method for detecting of a distracted driver is by thresholding on a specific preset image feature. This method has been most clearly described in reference [15].
Traditional Machine Learning : Various Traditional Machine Learning methods have been employed to solve this problem. Support Vector Machine (SVM) was implemented in [10] using eye movements as features. In [8], Vision and Lane tracking data was used as features for SVM. A hidden hidden Markov based method was used by [9] which requires the detection of driver's face and the left arm. The use of Bayesian Methods for the problem is highlighted in [6] [5].
Deep Learning : Deep Learning models have gained more attention attributed to the power of GPUs. In State Farm distracted driver detection challenge on Kaggle, Convolutional Neural Networks have attained highest accuracy as compared to Traditional Machine Learning. A VGG-16 Model attains an accuracy of 80% [3]. AlexNet and Region based CNNs were used by [13], [11] respectively. Abouelang *et al.* proposed a solution using a combination of five AlexNet and five GoogleNet models [1]. It attains an accuracy of 95.98%. A deep learning approach based on eye gaze was proposed by Choi *et al.* in [2]. The driver's face is recorded and a Haar feature-based face detector is combined with a correlation filter based minimising the output sum of square error tracker for face tracking. Then the AlexNet model is utilised to classify the nine gaze zones. Various techniques combining traditional machine learning to preprocess images for generating good features are used. These features can be then fed to a CNN to obtain a high classification accuracy.
After thoroughly studying various submissions on Kaggle we plan to use a CNN with transfer learning and also use Machine Learning Methods like SVM and GNB for classification of images.

1.4 Data set exploration

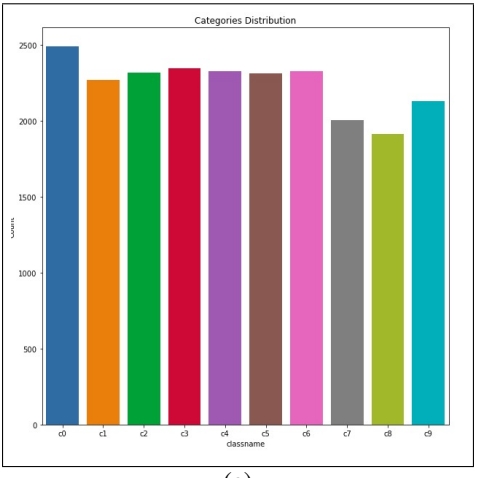
There are a total of 22424 images in the dataset which were available for training. These images were provided with labels. There were another 79728 images given for testing and they were without labels and meant for creating a submission at the kaggle challenge. We used the training images for training as well as validation set. The classwise distribution of each class can be seen in Figure 1(a). From this histogram it can be safely concluded that the dataset is roughly balanced and we need not worry about the issues pertaining to imbalanced datasets.

These images were taken in simulated environment using 26 different subjects performing various tasks. The Distributions of images with respect to each driver is shown in the Figure 1(b). We can see that there is some imbalance. But since our main focus is activity recognition this imbalance would not cause much problems.

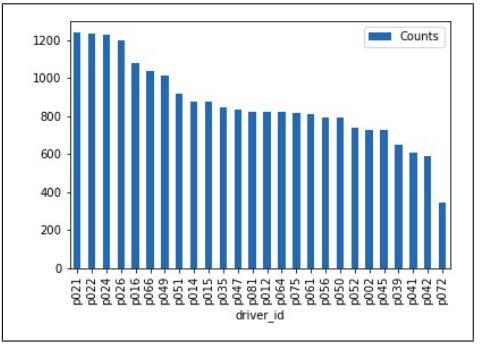
Some of sample images are shown in Figure 2. These images belong to a single subject with driverId=p021.

1.5 Machine Learning Implementation

1. The dataset had 22.4k images. Firstly, all the images were converted into grayscale and the sizes were reduced to 64x64. This was done to make the computations faster.



(a)



(b)

Figure 1: (a) Class wise distribution of the dataset and (b) Driver Wise distribution of the dataset.

(a) (b) (c)

Figure 2: Some Sample Images from the Dataset. (a) Texting - Right hand. (b) Operating the radio. (c) Drinking

2. For feature extraction, HOG (Histogram of Oriented Gradients) feature descriptor had been used. Feature descriptors retain only the most important information about images and discard the rest. HOG counts the occurrences of gradient orientation in a localized portion of the image. Orientations are classified into some bins. Then magnitude of gradient is distributed over bins according to the orientation. Each image gives rise to an array of features, which can be processed further.

Two important parameters of HOG descriptor are pixels per cell and cells per block. By varying these two, different HOG images are obtained. The clarity and sharpness of the images obtained decreases a lot when resolution is decreased. Using default values of the parameters, 2916 features were obtained. The information obtained was stored in a dataframe of size 22424x2916. Further operations were performed on this dataframe.

3. After that, a new dataset was created by choosing 4000 random images out of the total 22424. This was done to analyze the variation of explained variance with the number of principal components obtained using PCA (Principal Component Analysis).

Further data exploration was done to see distribution of labels in the data set. PCA was applied to reduce dimensionality to 2. Though, two principal components (PC1 and PC2) could explain just 22% of the total variation, a good insight was obtained on how different

Table 1: %age Variance explained by PCA

S.No.	Principal Components	Variance Explained
1	500	0.58
2	1000	0.72
3	1500	0.82
4	2000	0.89

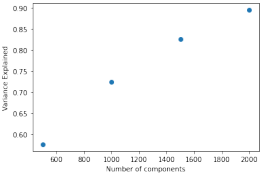


Figure 3: Scatter plot corresponding to Table 1.

classes are scattered. It can be easily observed from both the plots that data points of each class are closely grouped as clusters.

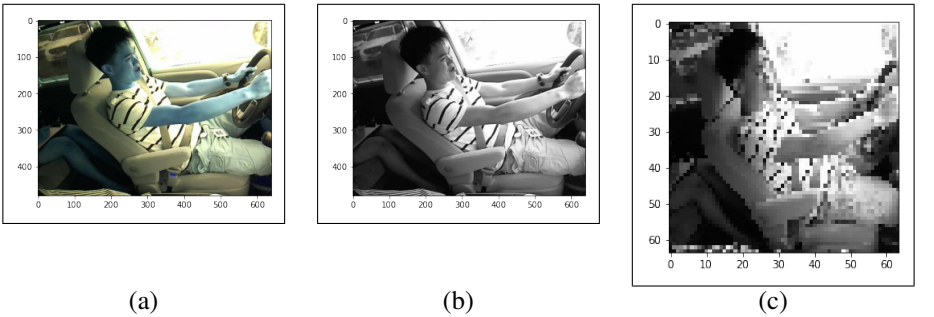
Classification Algorithms:

Traditional machine learning classification algorithms were applied. All of them were supervised learning algorithms. Firstly, the data was split into train and test data. The train data had 70% of the total images while the test data had 30% (6728 images). The classification report of each algorithm was also obtained.

SVM: Support vector machine is used to find a decision boundary which separates the points of different classes with maximum margin. The maximum margin hyperplane reduces the chances of points in the test data to be misclassified. It is a very popular technique in image classification too.

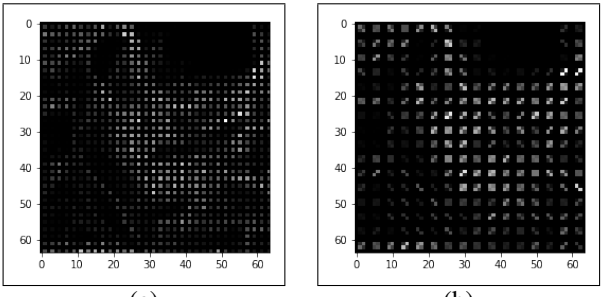
The decision boundary can be linear or non-linear. Non-linear boundary can be obtained by changing the function of the kernel. Here, we had used both linear and 'rbf' kernels. The performance of 'rbf' kernel by varying the parameter 'c' had also been studied. When the value of C was decreased from 1 to 0.1 and then to 0.01, the F1-score also decreased from 0.98 to 0.96 to 0.92.

GNB: Gaussian Naive Bayes is a generative classifier which learns the joint probability distribution of classes and labels. It becomes less efficient when the number of features or



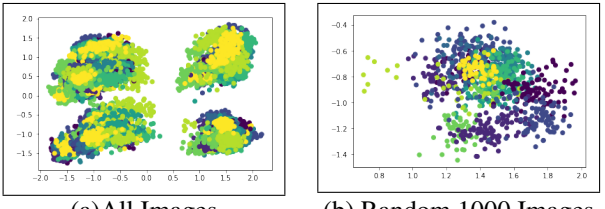
(a) (b) (c)

Figure 4: (a) Original image (b) Original image converted to grayscale (c) Resized image



(a) (b)

Figure 5: HOG image of Fig 5(c) when pixels per cells and cells per block are (a) less (b) more



(a)All Images (b) Random 1000 Images

Figure 6: Scatter Plot after doing the PCA on HOG.

classes are large. Its performance was the poorest of all and gave about 70% accuracy.
Logistic Regression: Logistic Regression is used to classify data into categorical labels. L2 regularization changed some class specific metrics but had no impact on the overall F1 score.
K-Nearest Neighbours Classifier: A point is classified according to its k nearest neighbours. The class which is most densely present in its k nearest neighbours is assigned to that point. Ten was set as the value of 'k' for this task. Reducing or increasing its value did not have much impact.

1.6 CNN based Implementation

We implemented several Convolutional Neural Networks to choose the best performing of them. For each model we rescaled the images to 64x64 and converted them to Grayscale for faster and memory efficient computations. We tried using Vanilla CNN with 4 convolutional and 2 dense layers and then we increased its depth by introducing more convolutional layers. Then we tried Data Augmentation in a bid to improve the validation accuracy of the model. After that we used Transfer learning using VGG16[7] and MobileNetV2[12] CNNs both of which are very deep ConvNets and latter of them employ skip connections to prevent the loss of information as it trickles down the Network.
Vanilla CNN V1: This model has 3 2D convolutional layers with 64, 128, 256 and 518 kernels respectively and 2D Max Pooling Layers in order to reduce the height and width by 2. Then we used 2 Dense layers with 500 and 10 neurons. Layers to Flatten and Dropout are also used in this model.

This model is shown in Figure 7. We used 19343 images (80 percent of dataset) as a train set for this and 4481 images(20 percent training images) as the validation set. We obtained

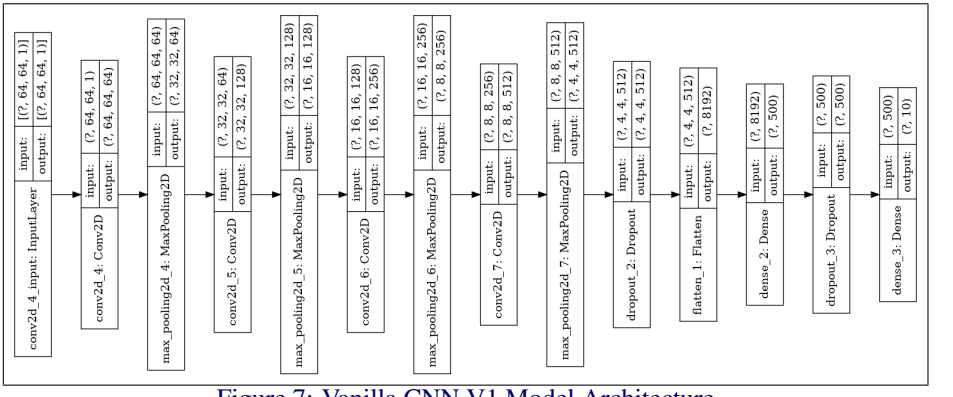


Figure 7: Vanilla CNN V1 Model Architecture

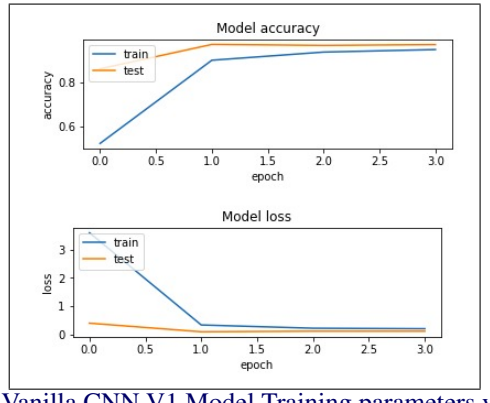


Figure 8: Vanilla CNN V1 Model Training parameters vs epochs.

an accuracy of 94.8% with a multi class cross entropy loss of 0.2211 on train set and 97.1% accuracy on validation set with a loss of 0.1121. The model could only be trained for 4 epochs as we used Early Stopping to monitor the reduction of validation loss which did not decrease after 4 epochs.

Vanilla CNN V2 : We increased the number of convolutional layers in V1 Model from 4 to 6. We also introduced the Batch Normalization layers in the network. This Model trained for 9 epochs with RMSProp Optimizer. We obtained an accuracy of 97% on train Set and an accuracy of 98% on Validation Set. The Model Architecture is shown in Figure 9 and training characteristics are shown in Figure 10.
We show some Model Outputs in Figure 11.

The training characteristics are shown in Figure 10 for this model. We retrained this model with image augmentation using random shear,zoom and horizontal flipping. This data augmentation didn't increase the accuracy much and they remained the same as model trained without using Augmentation. This performance on augmentated dataset can be attributed to the fact that the images were taken in a controlled enirоment and the location of person in each image was more or less the same.

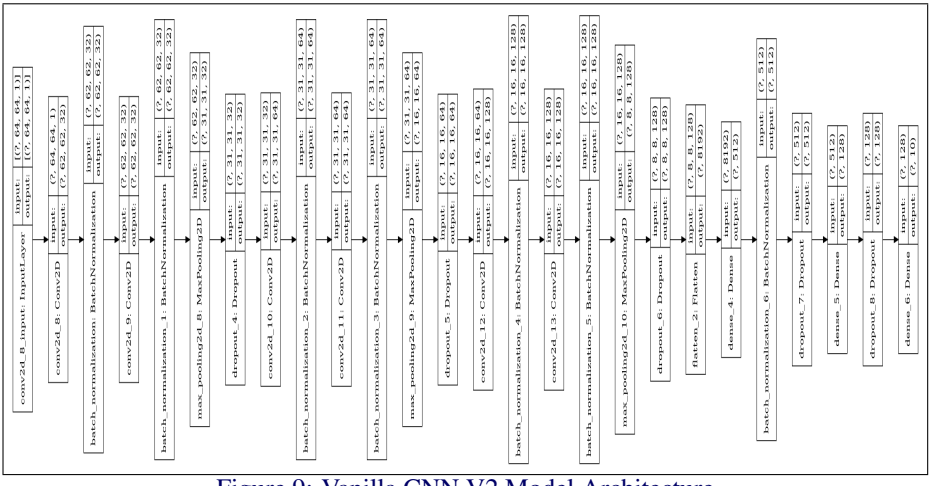


Figure 9: Vanilla CNN V2 Model Architecture

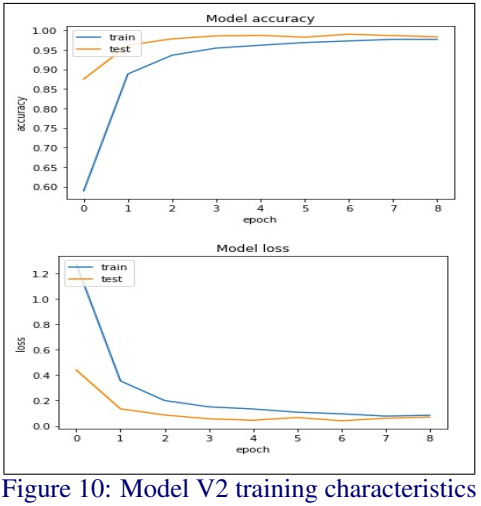
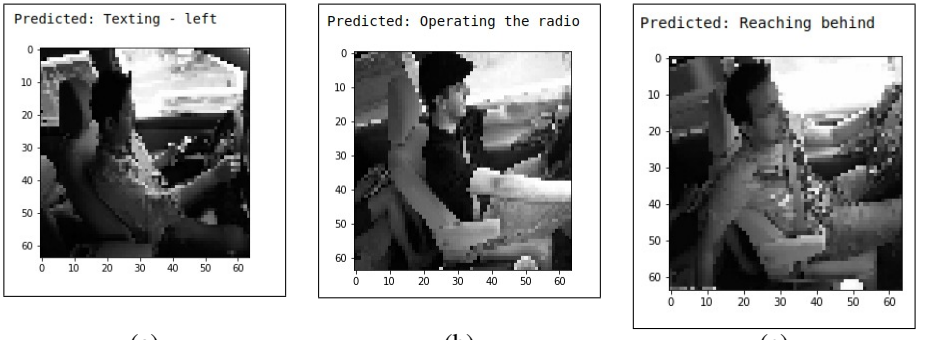


Figure 10: Model V2 training characteristics

Transfer Learning with VGG16 : We used transfer learning[16] with VGG16 Model using pretrained weights of imagenet Dataset[9]. We froze all the layers of this model so that it acts only like a feature generator. We added a trainable Dense classification layer to it with 1024 neurons and another with only 10 neurons. The model is shown in the Figure 12. We trained this model using RMSprop Optimizer , batch Size of 40 and Image Size as (64,64,1). We obtained a training accuracy of 81.72% and training loss of 0.51. The validation accuracy and validation loss were 78% and 0.5 respectively.The model was trained for 8 epochs using Early Stopping on Validation Accuracy.

Transfer Learning using MobileNetV2: MobileNet V2 is a deep CNN with skip connections so that information is not lost during the image pass from the network. We did transfer learning similar to the VGG16 network and trained this model for 20 epochs. We obtained an training accuracy and training loss of 91.54% and 0.3459 respectively. Validation Accuracy and Validation Loss were 85.27% and 0.4689. We used batch Size of 40 and Adam Optimizer was used here insted of RMSProp as opposed to previous models. This model architecture



(a) (b) (c)

Figure 11: Some of the outputs of Vanilla CNN V2 Model.

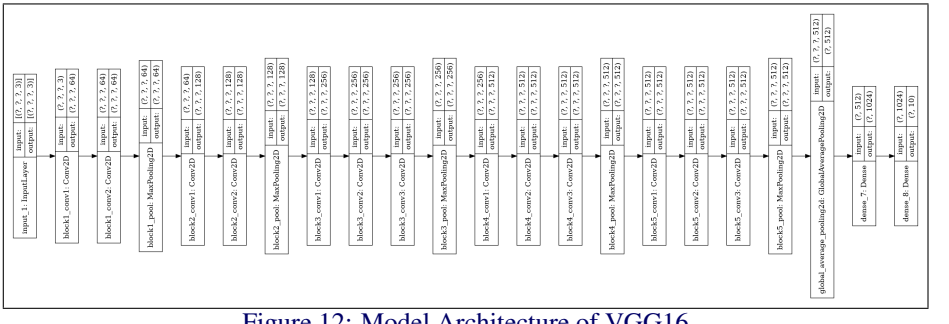


Figure 12: Model Architecture of VGG16

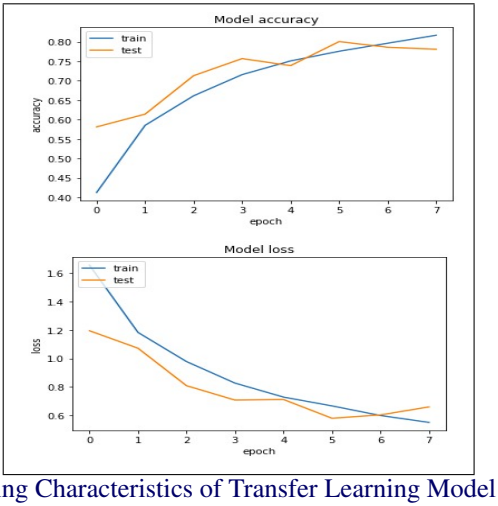
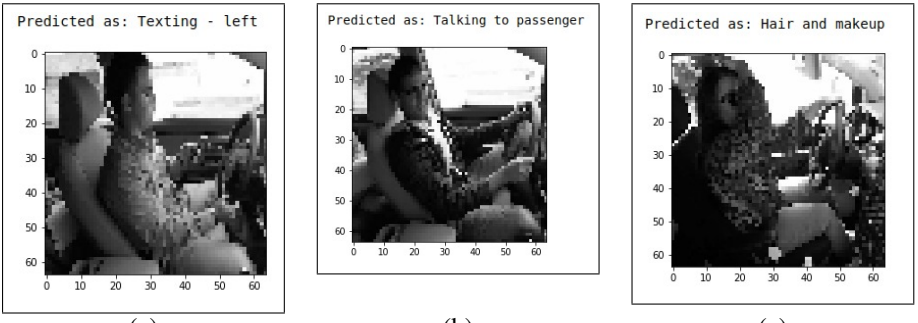


Figure 13: Training Characteristics of Transfer Learning Model using VGG16



(a) (b) (c)

Figure 14: Some of the outputs of Transfer Learning using VGG16.

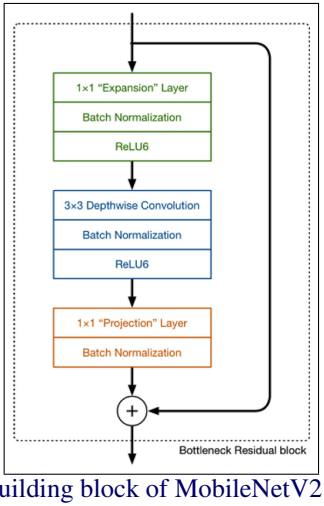


Figure 15: The Main building block of MobileNetV2 Source: ResearchGate

is based on repeating a block a number of times which makes this network deep. Figure 15 shows the main building block of this model architecture. [b] Some of the model outputs are shown in Figure 17.

1.7 Results

The results obtained by us using Machine Learning and Deep Learning approach are highlighted in table 2 and table 3 respectively.

Table 2: Results obtained using Machine Learning Based Methods

S.No.	Algorithm	Precison	Recall	F1-Score
1	SVM Linear	0.96	0.96	0.96
2	SVM RBF	0.97	0.97	0.97
3	GNB	0.70	0.70	0.70
4	Logistic Reg.	0.96	0.96	0.96
5	KNN	0.96	0.96	0.96

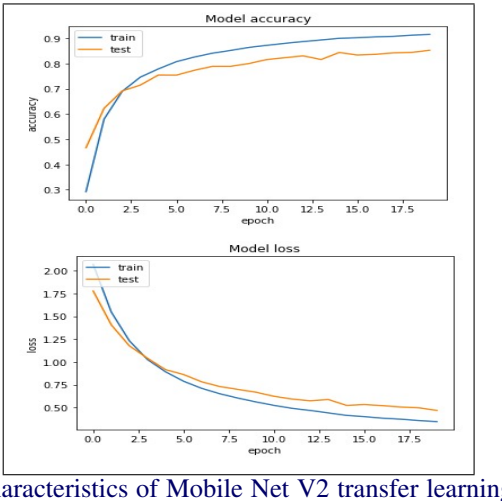
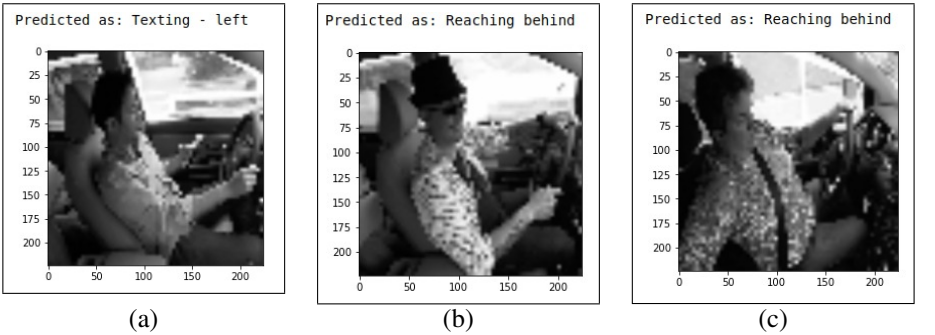


Figure 16: Training Characteristics of Mobile Net V2 transfer learning model. This shows some overfitting.



(a) (b) (c)

Figure 17: Some of the outputs of Transfer Learning using MobileNetV2.

Table 3: Results Obtained using Deep Learning based methods

No.	Model Type	Train Acc	Train Loss	Val.Acc	Val.Loss
1	Vanilla CNN V1	94.8%	0.211	97%	0.1212
2	Vanilla CNN V2	97.66%	0.0842	98.28%	0.0706
3	Vanilla CNN V2 with Augmentation	88.36%	0.3765	97.92%	0.0922
4	Transfer Learning VGG16	81.72%	0.5511	78.12%	0.6598
5	Transfer Learning MbNetV2	91.54%	0.3459	85.27%	0.4689

References

[1] Eraqi H.M. Moustafa M.N. Abouelnaga, Y. Real-time distracted driver posture classification. arXiv preprint arXiv:170609498, 2017.

[2] Hong S.K. Kim Y.G. Choi, I.H. Real-time categorization of driver's gazezone using the deep learning techniques. Int. Conf. on Big Data and SmartComputing (BigComp), Jeongseon, Republic of Korea, 2016, pp. 143–148.

[3] Cen K. Luo D. Colbran, S. Classification of driver distraction. (StanfordUniversity, Stanford, CA, 2016). Available at: http://cs229.stanford.edu/proj2016/report/SamCenLuo-ClassificationOfDriverDistraction-report.pdf.

[4] Karray F Craye, C. Driver distraction detection and recognition usingrgb-d sensor. CoRR, 2015, abs/1502.00250. Available at: http://arxiv.org/abs/1502.00250.

[5] Ji Q. Gu, H. Facial event classification with task oriented dynamic bayesian network. Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, CVPR, Washington, DC, USA, 2004, vol. 2, pp.II-870–II-875.

[6] Zhu Z. Lan P. Ji, Q. Real-time nonintrusive monitoring and prediction of driver fatigue. IEEE Trans. Veh. Technol., 2004, 53, (4), pp. 1052–1068.

[7] Andrew Zisserman Karen Simonyan. Very deep convolutional networks for large-scale image recognition. https://arxiv.org/abs/1409.1556v6.

[8] Jokela M. Markkula G. et al. Kutila, M. Driver distraction detection witha camera vision system. IEEE Int. Conf. on Image Processing, San Antonio,Texas, USA, 2007, vol. 6, pp. VI-201–VI-204.

[9] Stanford Vision Lab. Imagenet dataset. http://www.image-net.org/.

[10] Reyes M.L. Lee J.D. Liang, Y. Real-time detection of driver cognitive distraction using support vector machines. IEEE Trans. Intell. Transp. Syst.,2007, 8, (2), pp. 340–350.

[11] Csákvári M. Fóthi Á. et al. Lőrincz, A. Cognitive deep machine can train itself. arXiv preprint arXiv:161200745, 2016.

[12] Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen Mark Sandler, Andrew Howard. Mobilenetv2: Inverted residuals and linear bottlenecks. arXiv:1801.04381.

[13] Meng L Okon, O.D. Detecting distracted driving with deep learning. Interactive Collaborative Robotics, Hatfield, United Kingdom, 2017, pp. 170–179.

[14] Duy Do Ha Bai he Chowdhary Girish. Sheng, Weihua Tran. Real-time detection of distracted driving based on deep learning. IET Intelligent Transport Systems. 12. 10.1049/iet-its.2018.5172.

[15] Zoroofi R.A. Tabrizi, P.R. Drowsiness detection based on brightness andnumeral features of eye image. Fifth Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing, Kyoto, Japan, 2009, pp. 1310–1313.

[16] Tensorflow. Transfer learning and fine tuning. https://www.tensorflow.org/tutorials/images/transfer$_l$*earning*.