AI based Driver Activity Monitoring

Birender Panwar and **Debasis Satapathy** and **Saurabh Kirar** and **K Sudhakar Reddy and Dr. Narayana Darapaneni**

Abstract. Road accidents are very unfortunate events and most of these accidents take place due to driver distraction. The purpose of this paper is to build convolution neural network (CNN) based robust and predictive system that continuously monitors the driver activity and raise alarm and notification in case any unusual driver activity is observed while driving. Activities other than safe driving are considered as unusual driver activity or as a distraction. In this paper, three different network models namely VGG16, ResNet50, and custom CNN network have experimented. VGG16 and RestNet50 network are evaluated to achieve the highest possible classification accuracy for the cloud-based solution and the custom CNN model approach is considered to meet the low computational requirement with decent classification accuracy for edge devices. All three model performances are compared for the F1 score and the number of parameters needed to build the model. The Custom CNN model is optimized so that it can be deployed on edge device which in general have low computational power and memory. Such an edge device can be placed within the vehicles in front of a driver. The results show that our solution performs a baseline and state-of-the-art methods, yielding good classification accuracy. The applicability of the proposed approach is further demonstrated on edge devices which in this case is Raspberry PI4(RPI4) hardware with a camera integrated. We hope that this work will help to reduce road accidents due to driver distraction through the timely raising of alarm or notifications.

1 INTRODUCTION

The objective is to build a model that monitors driver activities under varieties of driving conditions and external environment including driving during day time and night time. A predictive system that can recognize any distraction while driving like expression arising due to attending phone calls, testing, drinking, talking to the passenger, or deviating from the standard driving alignment, and this is going to be a multi-class predictive system. This system is useful in the prediction of such anomalies and retrospection of an unlikely and any unfortunate event of accidents.

This work is implementing the ResNet50, VGG16, and custom CNN classification network on State Farm Distracted Detection Kaggle case study competition dataset. This is left-hand driving dataset contains 22,424 images belonging to 24 different drivers and categorized into 10 classes. We faced many challenges with the dataset such as duplications of images, misclassified images, it does not contain samples for night-time driving, and each of the class images is captured from different camera angles. We prepared the dataset by identifying duplication using the Structural Similarity Index Measure (SSIM) algorithm, manually removing misclassified images, adding noise through Python Imaging Library (PIL) to accommodate various lighting conditions and camera angle positions.

The clean dataset is then used as a training dataset to build all the three mentioned classification model. These three models are designed by trying with various network-layer changes and tuning hyper-parameters. Performance comparison analysis is carried out among all there models based on their classification accuracy, F1 score for each class, average weighted F1 score, volume in term of the number of parameters involved in the computation for predicting class and finally, we find out the optimal model best suited for cloud-based deployment and edge devices.

The applicability of the proposed solution is further tested by deploying a model on Raspberry PI4(RPI4) hardware. In our experiment, RPI4 is chosen because of its support for python packages, tensorflow, and Keras packages. This device is installed inside the vehicle in front of a driver to capture the live streaming images and the captured image is then fed to the model for prediction. We achieved satisfactory results in predicting the driver activity on-field by running the camera at 5FPS and 30FPS (Frame per second), hence show the true picture of the accuracy and robustness of the model. Some sample field test result is shared in this paper.

The main offerings of this paper are highlighted as follows:

1. The prime objective is to build a model that detects driver activity at any given time. At the core, it is a multi-class image classification problem.
2. Design a model that leverages existing ResNet50 and VGG16 classification architecture and using their convolutions layer with pre-trained weight for feature extractions and to build a model that can provide the highest accuracy score. Such a solution is targeted to be deployed on high-end systems having a large computing power and memory for executing the model.
3. To find an optimal model by designing custom CNN architecture that needs low computing power and memory for the prediction, and which can be deployed on edge hardware devices such as Raspberry PI hardware.
4. Evaluated extensively the model on filed where live images are captured through the camera and given to the model for prediction. The experimented results illustrate that the proposed model achieves a high level of accuracy over both test data and live camera data. This demonstrates the consistent applicability and robustness of the proposed model.
5. Also presented the overall approach to building industry solution that provides real-time alarm or notification to the driver, integration with a mobile application, web application

and databases for keeping historical records of driver activities.

2 LITERATURE SURVEY

Image Classification. Image classification, a fundamental problem in computer vision, can be described as categorizing images into one of the several pre-defined classes. It forms the basis for other computer vision tasks such as localization, detection, and segmentation. Traditionally, handcrafted features can be extracted from images using feature descriptors for classification, but the major disadvantage of this approach is that the accuracy of the classification result is profoundly dependent on the design of the feature extraction stage. In recent years, deep learning has developed to be a convenient, effective and robust tool to extract features from images, audio, etc., which does not require handcrafted features. Especially, DCNNs for the image classification tasks achieved state-of-the-art results in the ImageNet Large Scale Visual Recognition Challenge since the year 2012.

ResNet50. In theory, the performance of the neural network should be positively related to the depth of the network, because the deeper the network, the more parameters it has, the more complicated it is. But the early experimenters observe that as the number of network layers increases, the model accuracy will rise first and then reach saturation and continuing to increase layers will result in a decrease in accuracy sharply, which we called degradation. ResNet's proposal solves this problem very well. By introducing the residual network structure, it can make the network very deep, at the same time the final classification result is also very satisfactory. In this case, the depth of the network can be extended to tens, hundreds or even thousands of layers, providing the feasibility of extracting and classifying high-level semantic features. ResNet made a stunning appearance in the ILSVRC2015 competition, which raised the network depth to 152 layers, reducing the error rate to 3.57. In terms of image recognition error rate and network depth, it has greatly been improved compared with previous models. This also makes the network become the backbone network of the later convolutional neural network model, and many models with excellent performance subsequently are transformed based on ResNet. ResNet50 (as in 50 weight layers) implementation in the Keras core is based on the former. The ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The ResNet-50 has over 23 million trainable parameters.

VGG Net. VGG architecture is one of the most influential CNN architecture from literature. It reinforced the idea that networks should be deep and simple. It worked well on both image classification as well as localization tasks. VGG uses 3 × 3 filters in all thirteen convolution layers, ReLU activation function, 2 × 2 max-pooling with stride 2 and categorical cross-entropy loss function. In VGG16 and VGG19, the “16” and “19” stand for the number of weight layers in the network.

3 MODEL BUILDING WORKFLOW

The below figure illustrates the workflow for model building and shows how data preparation is carried out for handling duplication and misclassification. All images went through a horizontal flip to get a right-hand driving dataset. Noise is added to the data to accommodate various lighting conditions and adding effects of the external environment, and this dataset is then used for building a right-hand driving model. Dataset is preprocessed and then is divided into training and validation parts with a split ratio of 70:30, which means 70% of the dataset is reserved for training and 30% for validation. Model is finally deployed on Raspberry PI(RPI4) for real-time testing.

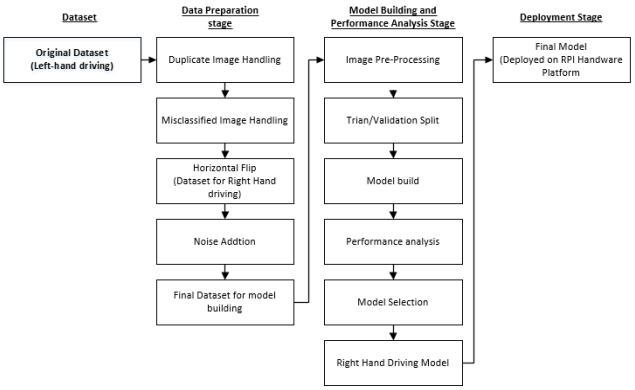

Figure 1. Workflow for building model

In this paper, the model is developed for Right Hand Driving as we targeted solutions for the India market. However, the same model can be used to predict the driver’s activity for left-hand driving. It can be achieved by performing a horizontal flip of the input images which create the mirror image and then it is fed to the model for prediction. The below diagram shows how the proposed model can be reused for left-hand driving.

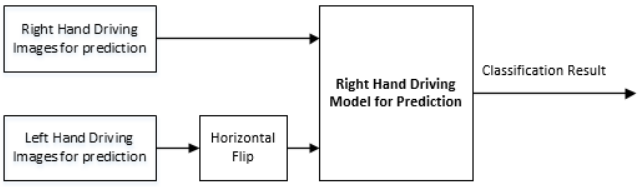

Figure 2. Using right-hand driving model for left-hand driving conditions

3.1 Architecture Selections

Various network architectures are considered before concluding the model that meets lower logarithmic loss, high accuracy, better F1 score and low computation time for a prediction. We used the various architecture parameters for selecting an optimal model such as depth of the network, filters at each CNN layers, max-pooling, fully connected layer, volume and number of epochs it took for convergence.

3.2 Model Hyperparameter Tuning

Various network parameters and model parameters were considered for choosing the best fit model and many permutation and combinations were evaluated for each model before finalizing the best model hyper-parameters. The various parameters that we experimented with to fine-tune the performance for each model are the number of filters, filter size in each convolutions layer, use of padding for convolution layers, number of model layers, the

activation functions, dropout of layers for regularization, the optimization algorithms, number of epochs for convergence, learning rate, batch size and pre-trained model weights.

3.3 Model Comparison and Error Analysis

As the model is for multi-class problems hence multi-class log loss between the predicted class of the image and the actual class will be the key driver to evaluate the model performance. Log loss considers the uncertainty of the model prediction based on how much it varies from the actual label.

After the model parameters were finalized for each model, the different individual models were trained and tested. The models were compared to each other and evaluated based on the below attributes:

- Models were compared on Accuracy, Loss and F1 Score.
- Confusion matrix to understand the model prediction per class.
- Volume/size, learning weights/parameters at each layer.
- How the model converges with Epochs.

4 Dataset(s) Description

The Dataset used in this paper is from State Farm Distracted Driver Detection Kaggle case study competition. The dataset contains 22,424 images categorized into 10 classes. Each image in the dataset is 640 X 480 pixels. This dataset contains images of 26 unique drivers and each class has data for each driver. It also contains a test dataset of 79,726 images which are not labeled. This test data contains images for the driver which does not exist in the training dataset. Test data is not used and seen while building the model. We have used test data only for verification to check the applicability and robustness of the model.

In training dataset, images are categorized into 10 classes as below:

- 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

4.1 Dataset Challenges

The existing dataset is for daytime driving and it does not contain data for night-time driving. Similarly, night-time driving data for each of the 10 classes are also needed. It is also observed that the dataset does not contain classes that lead to many common accidents and are very sensitive to save driving. An example such as a drowsing class is missing. On further exploration of data, we found that there exist duplicate images in each class. Also, there are many images which are wrongly classified.

5 Data Preparation

The availability of good quality dataset is an important factor in developing the right model. As existing dataset has many challenges of duplication, misclassification, lack of variability of driving conditions and hence we mitigated all these challenges in data preparation stages.

5.1 Duplicate Removal

As we have seen there are duplicate images for the drivers in each class and when the models were trained on this dataset as it is an unexpected accuracy of ~99% is observed for even basic model architecture, thereby making the models working well on validation set but not on real-time camera feed (production data).

The similarity between the images was measured to identify the duplicate images. We have leverage compare library from skimage for calculating Structural Similarity Index Measure (SSIM). SSIM value helps to identify whether two images are similar or not. This value can fall into the range [-1, 1] with a value of 1 being close to a perfect match. A similarity threshold of 0.8 was set to identify similar images.

The below figure shows the two different sample images of the same driver taken from a safe driving class. It demonstrates the similarity measure calculated using compare_ssim library of skimage package. For below two images, calculated SSIM value is 0.95 which indicates that these images are very similar to each another and hence image2 is ignored from the dataset.

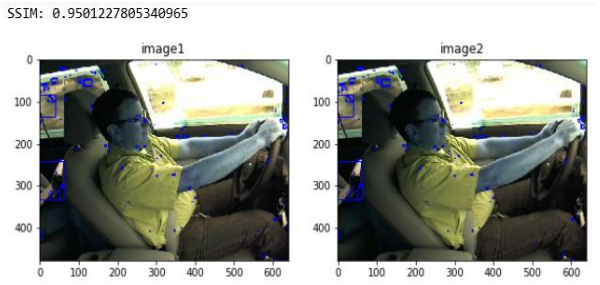

Figure 3. This figure illustrates the two duplicate images from same class and same driver. SSIM is calculated between these two images. The blue dots on the images represent the differences spotted on the two images.

Image per driver per class was compared with the rest within the class and wherever the SSIM value was greater than 0.8 the images were discarded else kept in training set. The below figure shows class wise duplicate data found on the dataset. Total of 6,569 duplicate images are identified.

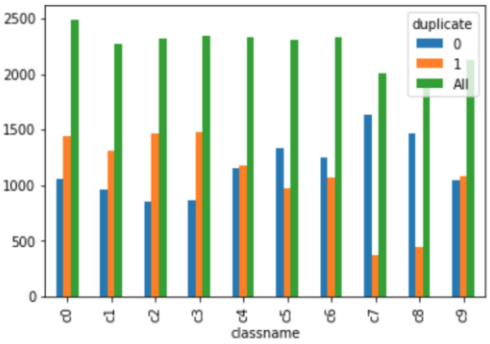

Figure 4. Duplicate data distribution for Structural Similarity Index Measure (SSIM) threshold value of 0.80

5.2 Treating Misclassified Data

The training dataset had incorrect labeling of the images and these misclassified images in each class were identified manually. The below figure shows the samples of misclassified images for 3 classes.

Figure 5. Sample image of misclassified data from class C0 and class C5

A total of 883 misclassified samples are identified and the below figure shows the distribution of misclassified images per class.

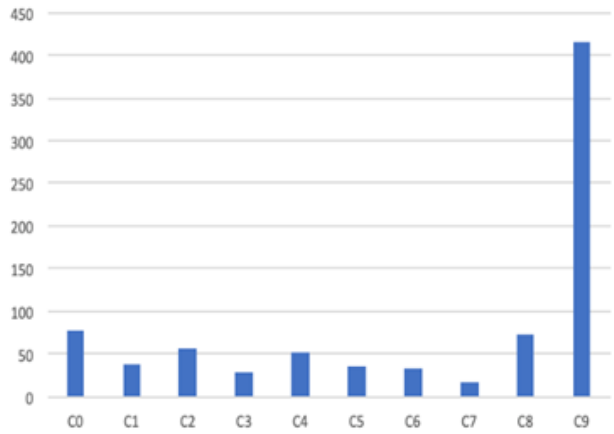

Figure 6. Misclassified data distribution

5.3 Creating Right Hand Driving Dataset

Right hand driving dataset is created by creating the mirror image of the existing Left-hand driving dataset. We leverage the horizontal flip feature of the Python Imaging Library (PIL) that helps us to create a mirror image for each sample. The below figure shows the samples of mirror images for classes.

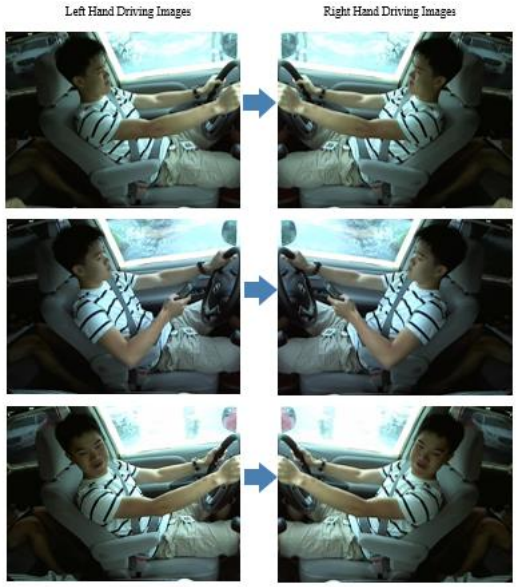

Figure 7. Sample image of right-hand driving condition

5.4 Adding Noise to the Dataset

To make the model more generic and avoiding prone to overfitting, it is necessary to include some more variation to the dataset to increase the robustness of the model. The given training set was not modular to cater all kinds of a real-time condition like change in brightness, change in driver position, etc., and to fulfill all the certain conditions some noises are incorporated to the existing training dataset.

Table 1. Image augmentation types added to the dataset

Augmentation Type	Parameter Value
Gaussian Blur	Radius=2
Sharpness	Value=2
Edge Enhancement	NA
Crop Border	0.20
Image shift (UP)	0.10
Brightness (Lighting)	5
Brightness (Darkness)	0.30

Figure 8. Sample images with result of noise additions

5.5 Data Preparation Summary

The below table shows how data distribution per class has changed after each data preparation step, i.e. post removal of duplicate data, handing of misclassified images and finally adding noise to add variability to the dataset.

Table 2. Table show summary of data distribution after duplicate, misclassified and noise addition

Class	Original	Duplicate	Mis-classified	Noise added	Final Image count
c0	2489	939	78	252	1724
c1	2267	700	39	266	1794
c2	2317	733	57	266	1793
c3	2346	761	29	266	1822
c4	2326	686	53	273	1860
c5	2312	669	36	280	1887
c6	2325	763	33	266	1795
c7	2002	367	18	280	1897
c8	1911	348	74	259	1748
c9	2129	603	416	616	1726
Total	**22,424**	**6,569**	**833**	**3,024**	**18,046**

Below is the final distribution of the data across all classes after removal of duplicates, misclassified treatment and adding noise

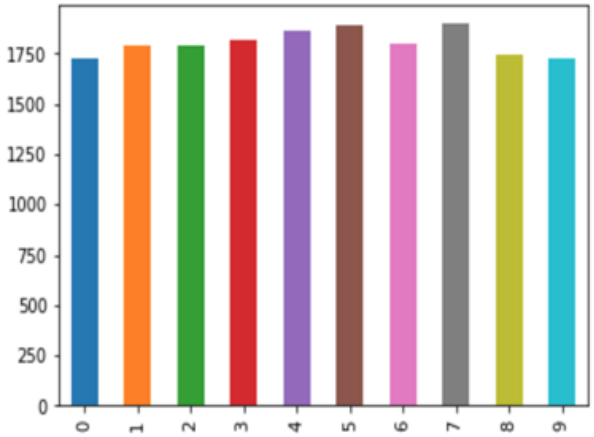

Figure 9. Final data distribution post data preparation

Finally, dataset of size 18,046 is used for building model.

6 Experiments and Implementations

Model(s) were trained using various input sizes. Initially, the models were trained with the standard size of 224*224 as this is the standard norm for these models (VGG16 and RESNET50) this image size resulted in a high number of model parameters and the model became computationally intensive for prediction. Models were trained with various image sizes and F1 scores were compared. It was observed that the image size 128*128 gave a better F1 score compared to other low size images and 128*128 image size was fixed for all the models.

6.1 Model using ResNet50

The model is build using ResNet50 architecture where the last layer is replaced with softmax layer of 10 classes. The two dense layers of size 1024 and 512 are added on top of the ResNet50 model. The model is build using Stochastic Gradient Descent (SGD) optimizer with a learning rate of 1e-3, decay of 1e-6, momentum of 0.9, batch size of 64 and epochs size of 50. Initial weights for training were taken from the pre-trained ResNet50 model. As per preprocessing step all images are resized to 128 X 128 and normalized. The total number of parameters of the model found to be 57,673,098 out of which 57,619,978 as trainable parameters. Below figure show model architecture using ResNet40.

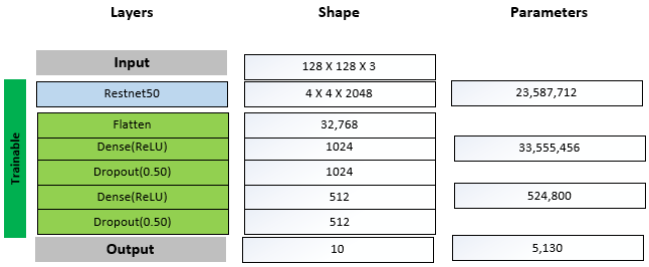

Figure 10. Model Architecture: Transfer Learning using ResNet50

6.2 Model using VGG16

The Below figure show model architecture using VGG16 architecture.

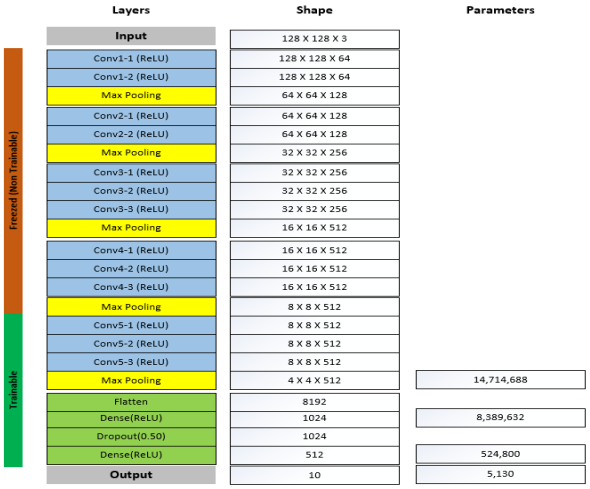

Figure 11. Model Architecture: Transfer Learning using VGG16

The proposed model is build using the VGG16 network where the last layer is replaced with a softmax layer of 10 classes. The three dense layers have also been added with outputs of 1024, 1024 and 512 after the existing convolutions block. ReLU activation function is used in the dense layer. As per preprocessing step all images are

resized to 128 X 128 and normalized. The bottom four convolution block of the VGG16 model are freeze and made non-trainable and initial weights for the model were taken from the pre-trained VGG16 model. After experimenting with various hyperparameters, the final model is build using RMSprop optimizer with a learning rate of 1e-4, batch size of 16, epochs of 200 and categorical cross-entropy is used as the loss function with metrics as accuracy. The total number of parameters of the model found to be 23,634,250 out of which 15,998,986 as trainable parameters.

6.3 Custom CNN Model

The custom CNN based is build using Convolution Network, max-pooling, and fully connected layer. Objective behind having custom CNN model to find model with volume less than 1million parameter and F1 score above 0.95. This model is build using 5 convolutions layer with each layer is followed by max-pooling of size 2X2 and dropout layer with rate of 0.25. In each convolution layer padding is set as "same". For first two layers kernel size is 3X3 and for layer 3,4 and 5, the kernel size is used as 5X5. As per preprocessing step all images are resized to 128 X 128 and normalized. After experimenting with various hyperparameters, the final model is build using RMSprop optimizer with a learning rate of 1e-3, batch size of 64, epochs of 50 and categorical cross-entropy is used as the loss function with metrics as accuracy. The total number of parameters of the model found to be 649,162 which is very less as compared to volume reported by ResNet50 and VGG16 model. Below figure show model architecture for custom model using CNN and dense network.

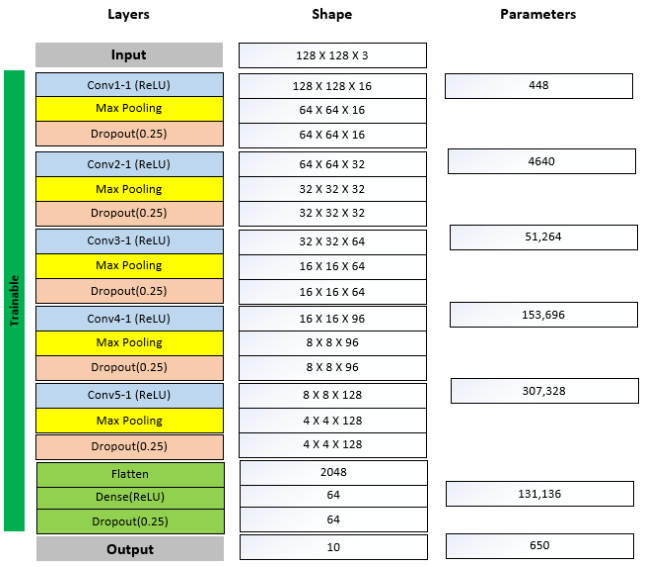

Figure 12. Model Architecture: Custom model using CNN

7 Model Performance Comparison

We performed a performance comparison analysis of the above three models (ResNet50, VGG, and custom-CNN) by comparing their validation accuracy, logarithmic loss, epoch convergence, model size and F1 score of each class and average weighted F1 score.

Table 3. Accuracy comparison for VGG16, ResNet50 & Custom CNN classification model

Models	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
VGG16 Model	0.9734	0.1153	0.9780	0.1022
ResNet50 Model	0.9998	0.0015	0.9889	0.0516
Custom CNN Model	0.9706	0.0989	0.9723	0.1000

Table 4. Model volume and epoch comparison for VGG16, ResNet50 & Custom CNN classification model

Models	Total parameters	Trainable Parameters	Epoch Convergence	Input size
VGG16 Model	23,634,250	15,998,986	8	128*128
ResNet50 Model	57,673,098	57,619,978	15	128*128
Custom CNN Model	649,162	649,162	29	128*128

Table 5. F1 Score comparison for VGG16, ResNet50 & Custom CNN classification model

Class	VGG16	ResNet50	Custom CNN
C0	0.97	0.99	0.97
C1	0.99	0.99	0.98
C2	0.99	0.99	0.97
C3	0.99	0.99	0.98
C4	0.98	0.99	0.98
C5	0.99	1.00	0.98
C6	0.98	0.99	0.97
C7	0.99	0.99	0.99
C8	0.97	0.98	0.96
C9	0.98	0.98	0.95
Average Weighted	0.98	0.99	0.97

Below are few recommendations after model comparison.

- The ResNet50 model has shown better results compared to the other two models (VGG16, Custom CNN) because of higher validation accuracy of 98.89, lower log loss of 0.0516, and better average weighted F1 score of 98. However, this model uses around 57M parameters for prediction and hence it is recommended to be deployed on the machine which has high computational power and memory such as cloud-based infrastructure or high-end machine.
- The custom CNN based model gave an validation accuracy close to 97% and with a weighted average F1 score of 97% and has less than 1M parameters for computation, so it is recommended to use the custom CNN model for edge devices.

8 Test Results

8.1 Performance Result on Test Data

We build the solution using the ResNet50 model and performed testing on test data (unseen data). As test data available with us is for left-hand driving, so we build a solution as per procedure is shown in Figure-2 where inputs test images are horizontally flipped before it is passed to the model for prediction. The solution first read the input test image file, it is horizontally flipped, resized to 128 x 128 X 3 and normalization and finally, the pre-processed input data is passed to the model for predicting top 3 class probability. These test images are selected at random from the test dataset of 79,726 images which are not labeled. These test images are unseen and never used for building the model. All the available test images are for the different drivers that does not exist in the training dataset. A few samples of test results are shown below:

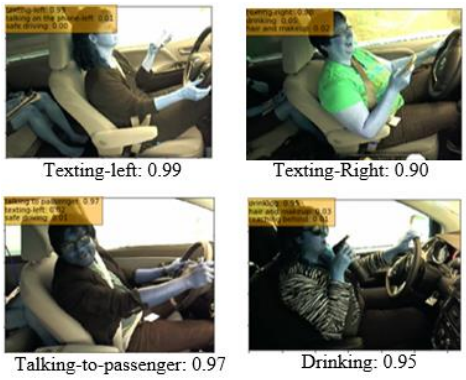

Figure 13. Result of ResNet50 model on unseen test data

8.2 Performance Result of Custom CNN Model deployed on Raspberry PI4 hardware

We also performed limited testing on the field with Raspberry PI4 (RPI4) hardware where the Custom CNN model is deployed for testing. The camera is configured for 5 FPS to capture the frame at 640 X 480 pixels and captured frame data is passed to the model for prediction.

Figure 14. The RPI4 board integrated with camera and LED screen.

The below samples are the actual field test results on Raspberry PI4 device facing real driver inside the vehicle.

Figure 15. Result of the Custom CNN model deployed on RPI4 board.

9 Future Work

While working on this project we faced few challenges while performing field testing on realtime camera feed and below is the action plan that can be considered for future work:

- Model training needs to happen on a more diversified data set to include noises like open market background and dataset need more diversified driver poses or camera angles.
- Use of advanced techniques like GANs to generate synthetic data for various driver's poses.
- Model can be made more generalized by introducing more classes like drowsing, taking selfies, another person's intervention to the driver while driving and many others.
- Explore other advanced Computer vision techniques like image segmentation, group convolution, pose detection algorithm, gesture recognition, object detection, etc.

ACKNOWLEDGEMENTS

We would like to thank Great Learning Faculties and Dr. Narayana Darapaneni who provided us an opportunity in doing the project, and who guided us to improve this paper considerably.

REFERENCES

[1] Distracted Driving in India, A study on Mobile Phone Usage, Pattern and Behaviour by Save Life Foundation. http://savelifefoundation.org/wp-content/uploads/2017/04/Distracted-Driving-in-India_A-Study-on-Mobile-Phone-Usage-Pattern-and-Behaviour.pdf

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Deep Residual Learning for Image Recognition*.

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

[4] https://www.kaggle.com/keras/vgg19/home

[5] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

[6] H. Eraqi, Y. Abouelnaga, M. Saad, M. Moustafa. ” Driver Distraction Identification with an Ensemble of Convolutional Neural Networks”, Journal of Advanced Transportation, Machine Learning in Transportation (MLT) Issue, 2019.

[7] International Conference on Learning Representations (ICLR)(2015)

[8] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016

[9] He, K., Zhang, X., Ren, S., & Sun, J. (2016, October). Identity mappings in deep residual networks. In European conference on computer vision (pp. 630-645). Springer, Cham.

[10] Szegedy, C., et al. "Going deeper with convolutions. Computer Vision." 2014 IEEE 12th International Conference on. 2014.

[11] Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z. Rethinking the inception architecture for computer vision Computer Vision and Pattern Recognition (CVPR) (2016).

[12] Sharif Razavian, Ali, et al. "CNN features off-the-shelf: an astounding baseline for recognition."Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2014.

[13] Chua, Leon O., and Tamas Roska. "The CNN paradigm." IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 40.3 (1993): 147-156.

[14] Shin, Hoo-Chang, et al. "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning." IEEE transactions on medical imaging 35.5 (2016): 1285-1298.

[15] Xu, Li, et al. "Deep convolutional neural network for image deconvolution." Advances in neural information processing systems. 2014.

[16] Jain, Anil K., Jianchang Mao, and K. Moidin Mohiuddin. "Artificial neural networks: A tutorial." Computer 29.3 (1996): 31-44.

[17] Haralick, Robert M., and Karthikeyan Shanmugam. "Textural features for image classification." IEEE Transactions on systems, man, and cybernetics 6 (1973): 610-621.

[18] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." CVPR (1) 1.511-518 (2001): 3.

[19] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[20] Wang, Heng, and Cordelia Schmid. "Action recognition with improved trajectories." Proceedings of the IEEE international conference on computer vision. 2013.

[21] Palaz, Dimitri, and Ronan Collobert. Analysis of cnn-based speech recognition system using raw speech as input. No. REP_WORK. Idiap, 2015.

[22] Alom, Md Zahangir, et al. "The history began from alexnet: A comprehensive survey on deep learning approaches." arXiv preprint arXiv:1803.01164 (2018).

[23] Trivedi, Mohan Manubhai, Tarak Gandhi, and Joel McCall. "Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety." IEEE Transactions on Intelligent Transportation Systems 8.1 (2007): 108-120.

[24] Zhang, Wei, Bo Cheng, and Yingzi Lin. "Driver drowsiness recognition based on computer vision technology." Tsinghua Science and Technology 17.3 (2012): 354-362.

[25] Wang, Qiong, et al. "Driver fatigue detection: a survey." 2006 6th world congress on intelligent control and automation. Vol. 2. IEEE, 2006.

[26] Horng, W. B., Chen, C. Y., Chang, Y., & Fan, C. H. (2004, March). Driver fatigue detection based on eye tracking and dynamic template matching. In IEEE International Conference on Networking, Sensing and Control, 2004 (Vol. 1, pp. 7-12). IEEE.

[27] Wang, L., Guo, S., Huang, W., & Qiao, Y. (2015). Places205-vggnet models for scene recognition. arXiv preprint arXiv:1508.01667.