Distracted Driver Detection using Deep Learning

Vaibhav Sundharam & Shagun Johari
Group 9
Department of ECE, Virginia Tech

May 13, 2020

Abstract

Deep Learning is one of the most popular tools used today for image processing and classification. One application of this tool is in distracted driver detection. With the increase in accidents nowadays due to distracted driving [1][2], it is crucial to address this issue and find a way to mitigate it. Hence, this project focuses on using deep learning techniques to find if a driver is distracted or not, using images of various drivers captured through a camera mounted in a car.

1 Introduction

The increase in the amount of accidents is quite alarming to see. Though factors like speeding drinking are some of the main causes of accidents, distracted driving contributes as well. It was reported that the escalation of distracted driving has been a contributing factor to the 15 percent increase in deadly crashes from 2014 to 2016 [1]. According to the National Highway Traffic Safety Administration (NHTSA), distracted driving has taken 2841 lives in 2018 alone [2]. Various activities come under distracted driving, the main component being wireless devices such as a cell phone. NHTSA reported that around 2.9 percent of drivers were using a cell phone which caused the accident in 2017 [2]. Hence, there comes a need to find a way to prevent accidents caused by distracted activities.

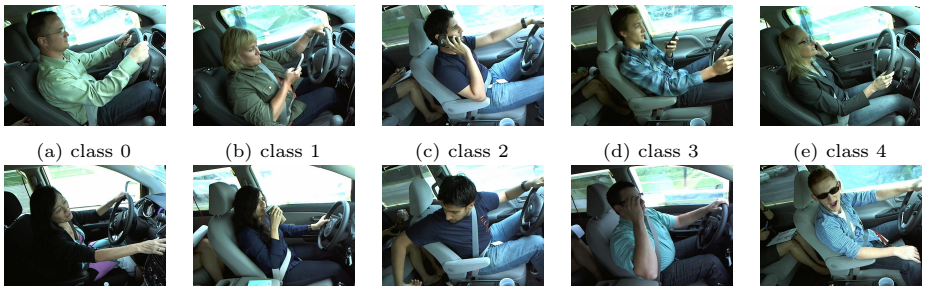
This project's objective is to detect activities that may cause a driver to be distracted using deep learning. It focuses on creating a model that will detect whether the driver is driving safely or is distracted. Section 2 elucidates various works related to our objective. Section 3 and 4 have explained the pre-processing of data and the evaluation metrics used in our model respectively. Section 5 defines the various architectures used in our model and lastly, section 6 analyzes the results of the model followed by the conclusion.

2 Related work

Image processing and machine learning-based classification models are among the most popular techniques used by researchers for analyzing whether a driver is distracted or not. For example, in [3] the authors have proposed a non-intrusive drowsiness detection system in which various facial features are used to classify if a driver is feeling sleepy. Convolution neural networks (CNN) is another such technique used for classification tasks. By using multi-facial feature fusion and two-stream CNN, Liu et al.[4] proposed a state of the art driver fatigue detection method. In [5] the authors have proposed an ensemble of CNN and have also incorporated skin detection technology to boost the performance of their ensemble model. In [6] the authors have proposed nine ensemble tree algorithms to identify any secondary tasks performed by the driver while driving. Apart from these many different approaches leveraging k nearest neighbours [7] and feature extraction [8] are also proposed.

3 Data pre-processing

The dataset that we've obtained is from State Farm which comprises of 22,424 labeled images and around 88,000 unlabeled images captured from a camera fastened in the car. The labeled dataset is almost equally distributed among 10 classes. The 10 classes show the different activities of a distracted driver with one class reserved for safe driving. The ten classes have been illustrated in the images shown in Fig 1. A total of 26 drivers have been used to get this dataset. To predict the classes of the unlabeled images, we will have to train our model on the labeled images first, but before feeding these images to the model, they have to be pre-processed. The original size of each image present in the dataset is 480 x 640 pixels. We resized each image to 244 x 244 pixels and then normalized them to a mean of 0 and variance 1.



(a) class 0 (b) class 1 (c) class 2 (d) class 3 (e) class 4

(f) class 5 (g) class 6 (h) class 7 (i) class 8 (j) class 9

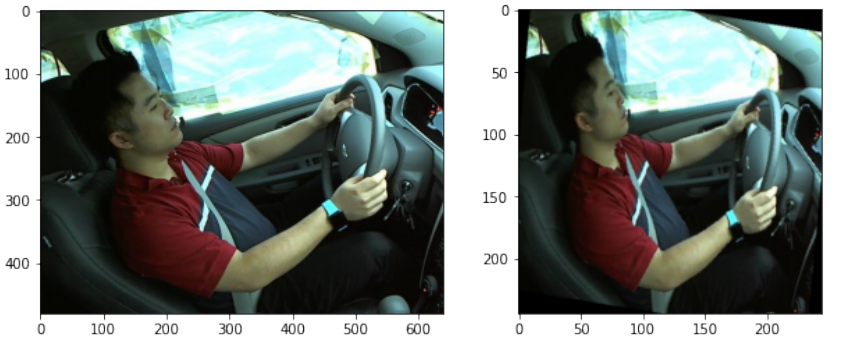
Figure (1) The 10 categories of distracted driving taken: a) safe driving b) texting-right c) talking on the phone-right d) texting-left e) talking on the phone-left f) operating radio g) drinking h) reaching behind i) hair and makeup j) talking to passenger

3.1 Data Leakage

Now to get our validation set, we initially split the labeled dataset randomly and took around 20 percent of it for validation. The problem with this course of action is that as the data is split randomly, there are numerous images of the same driver in different positions present in both the training and validation dataset. This is an example of data leakage and this problem had to be addressed to get valid results from our model. Samples of a validation dataset are kept separate from the training dataset and are not used to tune a model so that they can show how effective a model is without bias. Hence, to overcome data leakage, we have randomly split the images based on the driver IDs rather than simply splitting them in a ratio of 80/20. So now in our training data we have the images of 21 drivers and the rest of the drivers' images are used for the validation dataset.

3.2 Image Augmentation

While implementing convolution neural networks, it is important to have a large training dataset. If that is not the case, the best way to get more images would be to use a technique known as image augmentation. The objective of this technique is to create duplicate images having some variation such as rotation, shifting, flipping etc. A great advantage of having more images is that it can prevent overfitting, since CNNs usually have millions of parameters. The dataset that we are employing has around 22,000 images, so we have used image augmentation to synthetically increase the number of images in our training dataset. An example of this is shown in Fig. 2. Another additional advantage of using this method in our dataset is that there are some images which look very similar even though they belong to a different class, as shown in Fig. 3. Thus, having different looks of the same image can help to train the model better.



(a) Original image (b) Rotated and resized image

Figure (2) An example of image augmentation

(a) Class 8: Hair and make-up

(b) Class 2: Talking on the phone-right

Figure (3) Example showing class confusion

4 Model Parameters

4.1 Weights

On plotting the number of examples per class in our training data we realized that the number of examples in each class was different, with some of the classes having fewer number examples as compared to others. Fig. 4 shows the distribution of ten classes. We can observer from the figure that class 7 and class 8 have significantly fewer examples as compared to other classes. To account for this imbalance in our dataset we used the "weight" parameter in our loss function. This gives classes with fewer examples a higher weight. In other words, if we use weights, the cost function penalizes the model more when it misclassifies an example belonging to a class having a higher weight. For our dataset distribution as shown in Fig . 4 the weights were calculated as.

```
[1.0, 1.0816, 1.0508, 1.0633, 1.0616, 1.0788, 1.06, 1.2665, 1.3267, 1.1685]
```

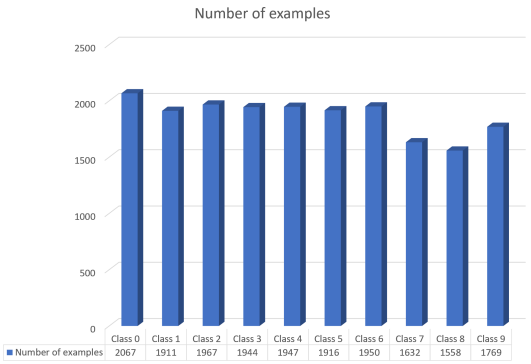


Figure (4) Number of examples in each class

4.2 Cost function and optimizer

In the project, we have employed cross-entropy loss. This criterion combines LogSoftmax and NLLLoss. Cross entropy loss is a parameter that is used in classification models whose output is a probability ranging from zero to one. As the predicted probability of a class diverges from the ground truth, cross-entropy loss increases. According to [13] the loss can be defined as,

$$L(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log(\sum_j \exp(x[j])$$

If we are using weights the loss function is transformed as [],

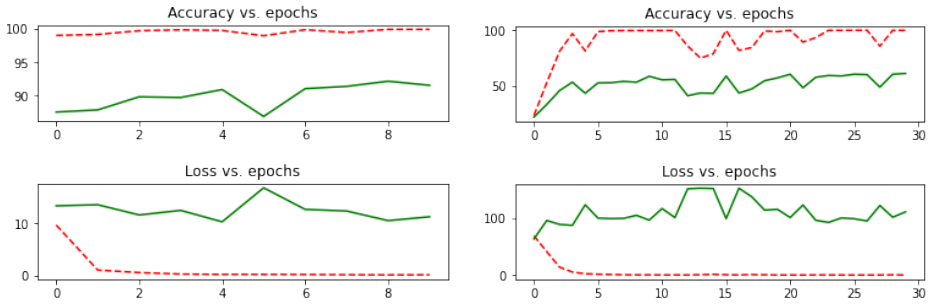
$$L(x, class) = weights[class](-x[class] + \log(\sum_j \exp(x[j]))$$

In this classification problem, it was important to classify the driver's actions with confidence and not just the accuracy of our predictions. That is why we chose cross-entropy loss as our evaluation matrix. The neural networks that are employed in this project try to minimize this loss while training. Stochastic gradient descent was used as the optimizer as it was much smoother in learning the parameters as compared to Adam.

4.3 Transfer learning

During training, we realized that training our models from scratch did not perform well even after using image augmentation and weights. Hence, to improve the performance of the neural networks we have used transfer learning. In transfer learning, a model is pre-trained on a large dataset (e.g. Imagenet) and the weights from the pre-trained model are reused as a starting point while training the model on a custom dataset, in this case, the driver's activity images. This is especially beneficial if we have a small data set to train on. We added custom layers to the neural networks to tailor the output to 10 classes. Since our training set had 18,661 images we decided to train the entire network on the training data not just the extra layers. Also, we observed that by using transfer learning we were able to get better performance. Fig[1]. 5(a) and Fig[1]. 5(b) show the difference in the accuracy/loss for RESNET34 when we used a pre-trained model as compared to when pre-training was not used. It can be observed that when pre-training is used. a higher accuracy is reached in lesser number of epochs as compared to when the model is trained from scratch.

[1] – – Training, —Validation



(a) With transfer learning (b) Without transfer learning

Figure (5) RESNET34 performance with and w/o transfer learning

4.4 Ensemble method

In the ensemble method, multiple classifiers are used to predict the output and the individual decisions from these classifiers are combined in some way to categorize new examples [14]. In the project, we have employed the majority voting technique, in which each model is fed with the training data to produce an output and the final prediction is the one that receives the most votes. For example, let the prediction from the 5 models we used be [class 6], [class 6], [class 6], [class 3], [class 3] respectively for a test image fed into them having true label as [class 6]. The majority voting algorithm will classify the image belonging to class 6. By using this method we were able to push the performance even further by a couple of notches. Fig. 6 shows the working of ensemble voting method.

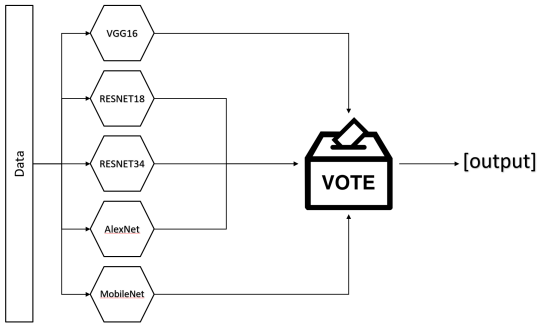


Figure (6) Ensemble voting method

5 Different models used

We have used 6 architectures and compared their results. The first architecture was made by us from scratch, explained in section 5.1. The rest of the architectures used are popular image classification models. We have also added extra

layers, consisting of two linear layers with ReLu as the activation function, on top of these pre-defined architectures to tailor the output to 10 classes.

5.1 CNN from scratch

We have created our own convolution neural network to test the data before using pre-defined architectures. The model consists of convolution layers, using rectified linear unit (ReLu) as the activation function. Pooling layers are also added. Pooling is done to reduce the number of parameters and hence reduce computation. We've added maxpooling, which calculates the maximum value of each strip of the feature map independently. In total we have 6 convolution layers. Each convolution layer is followed by a ReLu activation. A maxpooling layer is followed after two of these pairs.

5.2 Popular image classification models

AlexNet was one of the first deep networks which performed better than traditional techniques. This network accommodates 5 convolution layers, 3 fully connected layers, and 3 maxpooling layers [9]. The activation function used is ReLu. VGGNet is an improvement on AlexNet. It replaced large-sized kernel filters with consecutive smaller-sized kernel filters [10]. ResNet makes the neural network even deeper (with a network depth of 152) and uses residual modules. Basically, there is a link between the input and the output and the residual module is learning the features off the input already available [11]. The objective of MobileNet is to be a lighter deep neural network so that it can be used in mobile applications [12].

6 Results

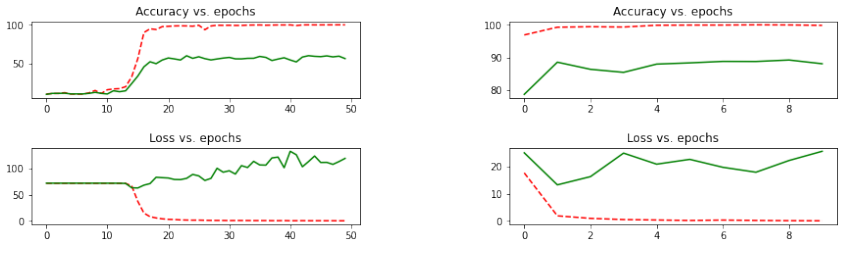
In this section we discuss the performance of the 6 models we used in this project. Table 1 presents the validation accuracy of different models.

Table (1) Models vs. validation accuracy

Model	Validation Accuracy	Validation Loss
Scratch	33.64%	237.50
VGG16	87.86%	47.24
ResNet18	88.98%	49.48
ResNet34	92.13%	39.88
AlexNet	82.41%	106.45
MobileNet	88.17%	56.70
Ensemble method	93.89%	—

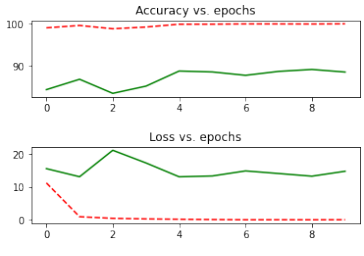
Fig.[1] 7 depicts the plot of accuracy and loss for the different models we used in the project. Among the 6 models, ResNet34 performed the best with the

accuracy of 92.13%. The performance of VGG16, MobileNet, and ResNet18 was almost equal with the accuracy lying in the range 87%-88%. By using the ensemble voting method we were able to boost the accuracy to 93.89%.

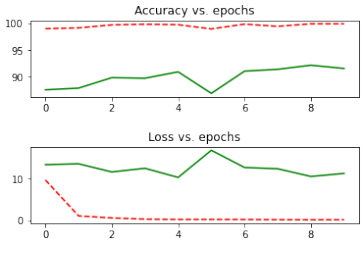


(a) CNN from scratch

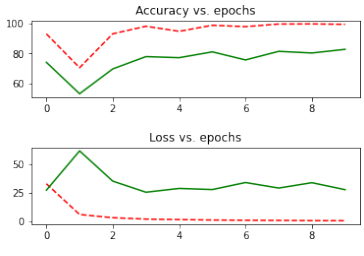
(b) VGG16



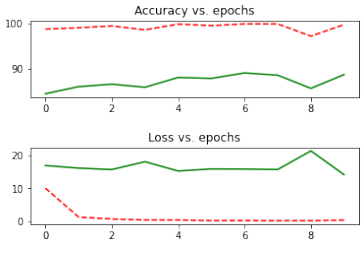
(c) ResNet18



(d) ResNet34



(e) AlexNet



(f) MobileNet

Figure (7) Plots of accuracy and loss for different models used vs. epochs

Now that our model has been trained, we took images of a particular driver from the unlabeled dataset and fed them to the ensemble model to predict the class of each image. To demonstrate this, we created a video which shows the images that were classified correctly and the ones that were misclassified, which can be found in the spotlight video.

7 Conclusion

To conclude, we can say that the majority voting ensemble method performed the best and gave the best accuracy as compared to the other architectures used. Using this model on the unlabeled images also gave good performance, giving only a few misclassifications.

References

Following are the references:

[1]Kidd, David & Chaudhary, Neil. (2018) Changes in the sources of distracted driving among Northern Virginia drivers in 2014 and 2018: A comparison of results from two roadside observation surveys *Journal of Safety Research* 68. 10.1016/j.jsr.2018.12.004.

[2]*Distracted Driving*.url:https://www.nhtsa.gov/risky-driving/distracted-driving

[3]Imranul Haq, Sheharyar, Jia-Cai Zhang, Driver Drowsiness Detection Based On Hog And Gabor Features *Social Science, Education and Human Science* url:http://www.dpi-proceedings.com/index.php/dtssehs/article/view/29597

[4]Liu, Weihuang & Qian, Jinhao Yao, Zengwei & Jiao, Xintao & Pan, Jiahui (2019) Convolutional Two-Stream Network Using Multi-Facial Feature Fusion for Driver Fatigue Detection *Future Internet* 11. 115. 10.3390/fi11050115.

[5]Eraqi, Hesham & Abouelnaga, Yehya & Saad, Mohamed & Moustafa, Mohamed (2019) Driver Distraction Identification with an Ensemble of Convolutional Neural Networks *Journal of Advanced Transportation. 2019*1-12. 10.1155/2019/4125865.

[6]Osman, Osama & Hajij, Mustafa & Karbalaieali, Sogand & Ishak, Sherif (2018) A hierarchical machine learning classification approach for secondary task identification from observed driving behavior data*Accident Analysis Prevention* 123. 274-281. 10.1016/j.aap.2018.12.005.

[7]Wali, Mousa & M, Murugappan & Ahmad, R.Badlishah (2013) Wavelet Packet Transform Based Driver Distraction Level Classification Using EEG*Mathematical Problems in Engineering* 2013. 10.1155/2013/297587.

[8]Botta, Marco & Cancelliere, Rossella & Ghignone, Leo & Tango, Fabio & Gallinari, Patrick & Luison, Clara (2019) Real-time detection of driver distraction: random projections for pseudo-inversion-based neural training *Knowledge and Information Systems* 60. 10.1007/s10115-019-01339-0.

[9]Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey (2012) ImageNet Classification with Deep Convolutional Neural Networks *Neural Information Processing Systems* 25. 10.1145/3065386.

[10]Simonyan, Karen & Zisserman, Andrew (2014) *Very Deep Convolutional Networks for Large-Scale Image Recognition* arXiv 1409.1556.

[11]He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian (2016) *Deep Residual Learning for Image Recognition* 770-778. 10.1109/CVPR.2016.90.

[12]Howard, Andrew & Zhu, Menglong & Chen, Bo & Kalenichenko, Dmitry & Wang, Weijun & Weyand, Tobias & Andreetto, Marco & Adam, Hartwig (2017) *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*

[13]*PyTorch Documentation* TORCH.NN url:https://pytorch.org/docs/stable/nn.html #torch.nn.CrossEntropyLoss

[14]Dietterich, Thomas G.(2000) Ensemble Methods in Machine Learning *Multiple Classifier Systems* Springer Berlin Heidelberg 10.1007/3-540-45014-9$_1$