Shreyas Satardekar Int. Journal of Engineering Research and Application www.ijera.com
ISSN : 2248-9622, Vol. 8, Issue4 (Part -III) April 2018, pp51-55

RESEARCH ARTICLE OPEN ACCESS

Distracted Driver Detection and Classification

Shreyas Satardekar*, Dharmin Shah*, Rohit Badugu*, Abhishek Pawar*, Prof. Pramila M. Chawan**

*B. Tech. Student (Department of Computer Engg and Info. Tech., V.J.T.I., Mumbai, Maharashtra, India
***Associate Professor (Department of Computer Engg. and Info. Tech., V.J.T.I., Mumbai, Maharashtra, India.*
Corresponding Auther : Shreyas Satardekar

ABSTRACT
The number of road accidents due to distracted driving has been on a rise in the recent years. As per the Union Road Transport and Highways Ministry Report 2016, 17 people were killed each hour in India due to road accidents. This makes it imperative to take measures to curb the number of road fatalities. The major cause of these accidents is driver error. This paper proposes solution to detect the distraction of driver, thus averting the possible accidents. The use of different Convolutional Neural Network (CNN) models namely: Small CNN, VGG16, VGG19, Inception for classification of distracted drivers according to State Farm Distracted Driver Detection challenge on Kaggle are depicted in this paper. The deep learning library used for the purpose is Keras running on top of TensorFlow. Our best result is a categorical cross entropy loss of 0.899 on the validation set.

Keywords- Classification, CNN, Keras, Transfer Learning, VGG.

Date of Submission: 10-04-2018 Date of acceptance: 24-04-2018

I. INTRODUCTION

According to the report compiled by the ministry Transport Research Wing [1], there has been a 3.2% rise in road fatalities which corresponds to the death of 1,50,785 people across the country in 2016. The number of road accidents in 2016 and 2015 was 4,80,652 and 5,01,000 respectively. Our project aims to mitigate the above problem by correctly identifying whether the driver is distracted or not. The software, if integrated with hardware can warn the driver if he gets distracted and thus prevent an accident from happening.

We input images of the driver to our model. Each image belongs to one of the 10 classes mentioned in the dataset section. The model then predicts the class of an image by giving as an output a probability for each class.

II. RELATED WORK

This problem was a public challenge hosted on Kaggle by State Farm insurance company two years ago [2]. Some of the solutions were based on SVM model that detect the use of mobile phone while driving [3]. Others were based on face and hand segmentation using RCNN [4]. Some approaches included the use of handcrafted features (HOG and BoWs) [5]. There are quite a few approaches based on Deep CNN models which are pre-trained on ImageNet such as AlexNet, ResNet-152, VGG-16. Some solutions consist of genetically-weighted ensemble of convolutional neural networks.

Techniques lacking in some the previous include data augmentation which augments more data to the dataset by zooming, rotating, shear, etc. help reduce overfitting. Some of them have not used ensemble and applied only a single model to the dataset. Others have not used dropout which is a regularization technique which helps reduce overfitting. Batch Normalization, which normalizes output of previous activation layer was also found missing in some implementations. Including batch normalization aids in faster learning and increased accuracy.

III. DATASET

State Farm is a large group of insurance and financial services companies throughout the United States. They released their dataset of 2D dashboard camera images for a Kaggle challenge. The dataset had 22400 training images and 79727 testing images. Resolution was 640 x 480 pixels.

The training images had corresponding labels attached. Labels belonged to one of the ten classes as mentioned below:
c0: normal 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
is shown in Fig 1.

Fig. 1- Example input image

The training set consists of 22400 images which are split into 2 parts i.e. train and validation sets. The images are split in such a manner that the same driver will not appear in both train and test set. This is due to the fact that the images are highly correlated to each other. In our initial approach, we randomly selected 150 images from each class to form the validation set consisting of 1500 images. However, this resulted in false high validation accuracy due to the high correlation between the images. Thus, we had to select images belonging to specific drivers to be a part of validation set such that the same drivers will not be part of the training set. The training set was thus split in this manner to ensure that validation set is not related to the training set.

The evaluation metric used for all the models is categorical cross-entropy or log loss. This is given as

$$\text{logloss}=-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{10} y_{ij}\log(p_{ij})$$

The logloss was used as the metric to judge the efficiency of the models. Here N stands for number of predictions and M is the number of classes which is 10 in our case. The value of y_{ij} is 1 if the image i belongs to class j with the probability value of p_{ij}.

IV. LITERATURE STUDY

4.1 Transfer Learning

Pre-trained models were used as a starting point instead of starting from scratch [6]. It had several benefits. The pretrained models we used, have been trained on a very large dataset (ImageNet), which contains 1.2 million images with 1000 categories. To adapt the model weights to our dataset we roughly fixed first 70 percent of the layers (made them untrainable) and trained last 30 percent. The reason being that the initial layers of the model include edge detection and shape detection modules, which are generalized for any image recognition application and these become

A sample input image

increasingly more abstract in the final layers, making it more specific to the application. In the distracted driver scenario, the last layer gives an output one of the 10 classes for a given image.

4.2 Convolutional Neural Network

Neural network is a layered architecture containing neurons. We input certain data to the network, the layers are interconnected to each other and have some initial weights. As we train the network the weights get updated and this means that the model has learnt the features of our dataset. Convolutional neural network is same as neural networks but for images. So, we provide images as input to the CNN model. It consists of input layer, output layer and number of hidden layers. Hidden layers include the Convolution layer, Pooling layer, Rectified Linear Units layer, Dropout layer and Fully Connected layer. [7]

4.2.1 Input Layer

The input layer holds raw pixel values of the images. In our case, images are colored with resolution of 640*480 pixels which are scaled down to 224*224 to reduce training time.

4.2.2 Conv Layer

The Conv layer contains a set of learnable filters of small dimensions. These filters are moved throughout entire region of input image and at each location a dot product is taken with the weights of filter and small region beneath the filer. For our project if 12 filters are used then output dimension would be 224*224*12.

4.2.3 Pooling Layer

The Pooling layers reduced the 2D dimensions of input volume to prevent from overfitting or to avoid computation inefficiencies. This is done by applying a small filter to input data on each depth slice. There various types of pooling filters like max pooling which select max value under filter, average pooling, etc.

4.2.4 ReLu Layer

It applied activation function to each element to increase non-linearity of the model. max(0,x) is an example of activation function.

4.2.5 Dropout Layer

The Dropout layer is added to prevent the model from overfitting. It is a regularization method which randomly sets some activation values to zero to remove some feature detectors. In our models we have added a dropout layer with value of 0.5.

4.2.6 FC Layer

In this layer each neuron is connected to outputs from previous layer. This layer gives the final prediction for each class. In our project there are 10 classes, so FC layer contains 10 neurons.

V. APPROACH

4.3 Small CNN

Initially we created a small CNN model from scratch. This model had 5 convolution blocks of 20 layers. Each block consists of layer of CNN with 32 filters and filter size of 3*3. After CNN layer, block contains ReLu layer, which is followed by MaxPooling layer with pool size of 2*2. After these 5 conv blocks the last block consists of 2 dense layers and dropout layer with dropout value of 0.5. For compiling SGD optimizer with learning rate of 0.01 and momentum of 0.9 value is used. The batch size is kept as 32. The images are augmented to prevent overfitting. The model is run for 20 epochs. The output of model is as follows:
After 20 epochs, the training accuracy reached 91.2% while validation accuracy is 54.45%. The training loss is 0.3084 and validation loss is 2.379.

Figure 2-Training and Validation loss of Small CNN

These results show that training accuracy is much higher than validation accuracy which implies overfitting. To reduce overfitting dropout layer was added, even after this model was overfitting. The reason for overfitting is small number of layers for large size of training data, due to which model is not able to fit accurately on training set.

This can be avoided by adding large number of layers but that would lead to very high computation cost, to solve this problem we used Pre-trained models which are trained on ImageNet dataset. These models have large number of layers and complex architecture which is suitable for classification of our data.

4.4 VGG-16 and VGG-19

VGG-16 and VGG-19 are part of VGG network architecture which was introduced in the paper Very Deep Convolutional Networks for Large Scale Image Recognition in 2014 by Simonyan and Zisserman [8]. The number of weight layers in VGG-16 and VGG-19 are 16 and 19 respectively. For training purposes, we had to make use of Google Cloud Platform. Due to unavailability of GPU, we selected a batch size of 32. The next hyperparameter which we had to decide on was the learning rate. The model did not perform well at learning rate of 0.001 and we decided to set the learning rate as 0.0001. VGG-16 consists of 5 blocks and we chose to train the top block of VGG-16 i.e. we froze the first 15 layers and unfroze the rest. We did not include the top i.e. we removed the last softmax layer of VGG-16 and added a Global Average Pooling Layer after the 5th block. These values are fed into a dense layer of 256 output dimension and 'relu' activation function. Then a dropout layer was added to combat overfitting and finally a Fully connected layer to output the final prediction values. We trained the model for 2-4 epochs and reached a validation accuracy of around 80-83%.

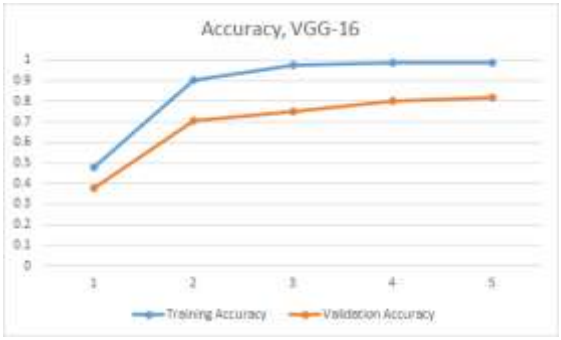

Figure 3-Training and validation accuracy of VGG-16

Since the data for training is very less, the model faced overfitting. To overcome overfitting, we decided to create an ensemble of VGG-16 models. We trained another VGG-16 model with batch size 64. This was the largest batch size which we could set while training on CPU. The rest of the hyperparameters were kept unchanged. This model gave a validation accuracy of around 75-77%

The same procedure was followed for VGG-19. In this model, we froze the top 17 layers and trained the rest. The learning rate was 0.0001 and the batch size was 32. The same set of layers which were used to replace the original softmax layer in the previous VGG-16 model were used in this case. This model gave a validation accuracy of 75-77%. Data Augmentation was used in both the models.

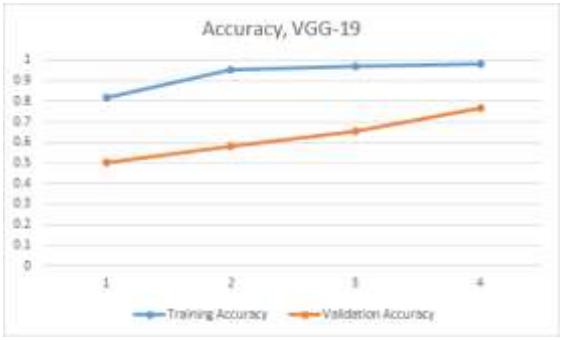

Figure 4-Training and validation accuracy of VGG-19

4.5 InceptionV3

The InceptionV3 model is pretrained on the ImageNet database introduced in the paper- Going deeper with convolutions [9]. These final layers need to be trained as per the requirement of the application. The inception base model has 313 layers. In the distracted driver scenario, initial 172 layers are set as non-trainable, while training the remaining final layers i.e. the top two blocks of the model. Adams Optimizer was used to compile the model. However, better results were obtained by replacing it with SGD optimizer with a learning rate of 0.0001.

The base model seemed to overfit the data to a great extent. due to availability of only 22400 images.

Therefore, the data fed to model for training is augmented. Data augmentation makes the model become more robust and prevent overfitting. The different augmentation techniques used were rescaling the image, rotating, random vertical and horizontal shift, shear intensity and random channel shift. Dropout layers were also added. Dropout helps in ignoring a few random neurons while training of the model on the dataset. Regularizers help reducing the weight of non-relevant features. In our model, regularizers were added to the layers over the base model.

Batch Normalization is the pre-processing step to solve the issue of "internal covariate shift". The batch size was set to 32. The first step was to train only the top layers since their weights were randomly initialized. In this step all convolution InceptionV3 layers are set as non-trainable.

The aforementioned steps helped achieve validation accuracy of 0.73.

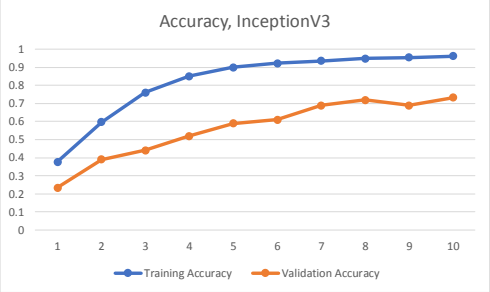

Figure 4- Training and validation accuracy of InceptionV3

4.6 Ensemble

It was found out that using an ensemble of different models will yield better results than using a single model as overfitting is a major concern in this problem. Thus, instead of relying on the predictions of one single model, we averaged the results of all the 3 of our models namely VGG-16, VGG-19 and InceptionV3 to get the final prediction values. The best VGG-16 model had a log loss value of 0.8157. Similarly, VGG-19 and InceptionV3 had the log loss values of 0.9631 and 1.0972 respectively. Creating an ensemble of these models gave us a log loss value of 0.795.

VI. CONCLUSION

Thus, after trying out several CNN models, our best ensemble was created after averaging the probabilities generated by VGG-16, VGG-19 and InceptionV3. The final log loss which we got was 0.795.

We only used CPU provided by Google Cloud Platform for this project. If we would have had access to more computing resources, we could have tried to improve our results with the help of the following:

1. Using KNN to find out the K nearest neighbors of an image and then generating the final probability by considering the average of the probabilities of these images. This approach works well because of high correlation between the images.
2. Trying ResNet-50 and ResNet-152. These two CNN models are widely used for image classification problems and could provide good results in this scenario.
3. Cropping out parts from the images which provide more information such as hands, eyes, etc. to improve accuracy.

REFERENCES

[1] Report on Road Accidents in India 2016- Ministry of Road Transport & Highways (MoRTH), Government of India pp. 1-2 http://morth.nic.in/showfile.asp?lid=2904

[2] Kaggle. A brief summary https://www.kaggle.com/c/state-farm-distracted-driver-detection

[3] Yehya Abouelnaga, Hesham M. Eraqi, and Mohamed N. Moustafa, “Real-time Distracted Driver Posture Classification”, arXiv preprint arXiv:1706.09498

[4] 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," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, 2016, pp. 46-53

[5] Hssayeni, Murtadha D; Saxena, Sagar; Ptucha, Raymond; Savakis, Andreas, “Distracted Driver Detection: Deep Learning vs Handcrafted Features”, Society for Imaging Science and Technology, Imaging and Multimedia Analytics in a Web and Mobile World 2017, pp. 20-26(7)

[6] dImageNet: VGGNet, ResNet, Inception, and Xception with Keras

https://www.pyimagesearch.com/2017/03/20/i

magenet-vggnet-resnet-inception-xception-keras/

[7] CS231n Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/convolutional-networks/

[8] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv preprint arXiv:1409.1556

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going Deeper with Convolutions", arXiv preprint arXiv:1409.4842

Shreyas Satardekar "Distracted Driver Detection and Classification "International Journal of Engineering Research and Applications (IJERA) , vol. 8, no. 4, 2018, pp.51-55