Capstone Project Machine Learning Engineer Nanodegree

By

Ashish Bansal

Definition

Project Overview

Every one in five car accidents is caused due to the driver being distracted says the research of CDC motor vehicle safety division.This has resulted in 425,000 people getting injured and more than 3,000 people were killed due to this problem of driver being distracted.

The government hopes to improve this disturbing statistics and insure their customers by making a machine learning model so that dashboard cameras can automatically the drivers engaging in distracted activities like talking on cell phone or taking selfie while driving etc.

I have created the machine learning model in this project which can detect what the driver is doing in the car by looking at their images.This model is a classification task which will predict the likelihood of what the driver is doing in each image.This can majorly reduce the risk of an accident and make world roads a safer place.

Problem Statement

We are given the 2D images of driver in the car. First,we have to make the algorithm learn the function behind the problem and then we should also have some test data so that we can test our function or model i.e how good is our model performing on the unseen examples.

After learning our algorithm will be able to tell whether the driver is driving attentively or whether they are engaged in distracting activities like taking selfie or talking on phone,etc.

For this following are the tasks that we have to perform,
1.Download and preprocess the Driver image data.
2.Divide the data into training,validation and testing data.
3.Build and train the model to classify images.
4.Test the model performance on validation during training and after the model is ready ,test it on the test data(unseen data).

This is classification problem in which there are 10 classes ,we need to find the function which if provided X(set of inputs) gives Y(set of outputs or class label in this case).It is a supervised learning problem since the data is labelled.In machine learning we don't know the true function we try to assume it to be some functions and try to learn parameters by some algorithm to make that function as close as possible to the true function.
We are going to use the convolutional deep neural networks (4-layer convnet) to generalise the given dataset by learning the weights and biases such that the the log-loss error is minimized.The log loss error is the objective function that we are going to minimize and find weights and biases such that the log-loss error is minimum.First,we will take random weights and this is going to give some error ,we will try to minimise the error by updating weights and biases which will be opposite to the direction of gradient.(this is known as gradient descent algorithm).Along with this we

are going to use back propagation algorithm so that we don't have to do recomputations.So the algorithm we are going to use is the gradient descent with back propagation which aims to minimize the log-loss error by learning weights and biases. Also ,we will be using activations function in the hidden layer (in this case we are using ReLu) which will bring non-linearity in th model and make model generalize better.
softmax function is used in the output layer so to make the sum of the probabilities equal to 1 and ReLu in other layers.

In high -level language,Driver distraction recognition problem can be treated as a multi-class classification process to map the input observations to a driver state. The developed system includes three main components.The first component is a variant of convolutional deep neural network for high-abstracted feature extraction. Followed by a max pooling layer which reduces the dimension of features. The last component includes 6 fully-connected layers and a softmax layer.

Metrics

The objective function that we are going to use for this problem is multi-class log-loss function.This function tell us how is our machine learning model is performing ,lower the value better the model, so,we will try to minimize the value of the function .The function is given by the following equation.

$$\text{Log-loss}= \frac{-1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} p_{ij} log(p_{ij})$$

Where N is the number of images in the test set ,M is the number of classes. P_{ij} is the actual probability of ith image belonging to jth which will be 1 for the true class and zero for others.log(p_{ij}) is the log of predicted probability.We will try to minimise the above function.

The submitted probabilities for a given image are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). In order to avoid the extremes of the log function, predicted probabilities are replaced with $max(min(p, 1 - 10^{-15}), 10^{-15})$

The above multi-class logarithmic function is selected other just taking accuracy as it gives probabilities rather than just giving yes or no which is kind of very harsh and the above function is differentiable and continuous so it makes the math easy.Also, its zero for the false class which makes the math simple and eliminates most of the terms.

Analysis

Data Exploration

The Dataset contains the driver images which was each taken in a car with driver involved in some activity like messaging ,talking on mobile ,taking selfie or paying attention to the driving etc.The dataset has been taken from the kaggle website(State Farm Driver Detection Competition).

When we will download the dataset we will have the following files

- driver_imgs_list.csv - a list of training images, their subject (driver) id, and class id.
- sample_submission.csv- a sample submission file in the correct format
- imgs.zip-zipped folder of all (train/test) images

Files or the dataset that we want can be downloaded from the following links https://www.kaggle.com/c/state-farm-distracted-driver-detection/download/driver_imgs_list.csv.zip

https://www.kaggle.com/c/state-farm-distracted-driver-detection/download/

Sample_submission.csv.zip

https://www.kaggle.com/c/state-farm-distracted-driver-detection/download/

imgs.zip

We have 10 classes in our dataset. The following are the classes with their description.

Classes	**Description**
c0	Safe Driving
c1	Texting (right hand)
c2	Talking on the phone (right hand)
c3	Texting (left hand)
c4	Talking on the phone (left hand)
c5	Operating the radio
c6	Drinking
c7	Reaching behind
c8	Hair and makeup
c9	Talking to passenger(s)

The images given are coloured and have dimensions 640 x 480 pixels .There are 102150 images in total of which 17939 are used for training and 4485 are kept for validation and remaining 79726 images are kept for testing.All the images in the dataset belong to the category of one of the above given classes.

First task in Data preprocessing would be to make all the images dimensions resize to 224 x 224.The output of the above images are one-hot encoded as
Safe driving:[1,0,0,0,0,0,0,0,0,0]
Talking on phone left:[0,0,0,0,1,0,0,0,0,0],etc.

Exploratory Visualization

First,we calculate the total number of images for each class.For example, we have almost 2000 images for class c7 and we have to find for others as well and then plot the bar graph for the data.This is given by the following bar graph by looking at this graph we can tell that the the classes are uniformly distributed and there is no as such called rare class.

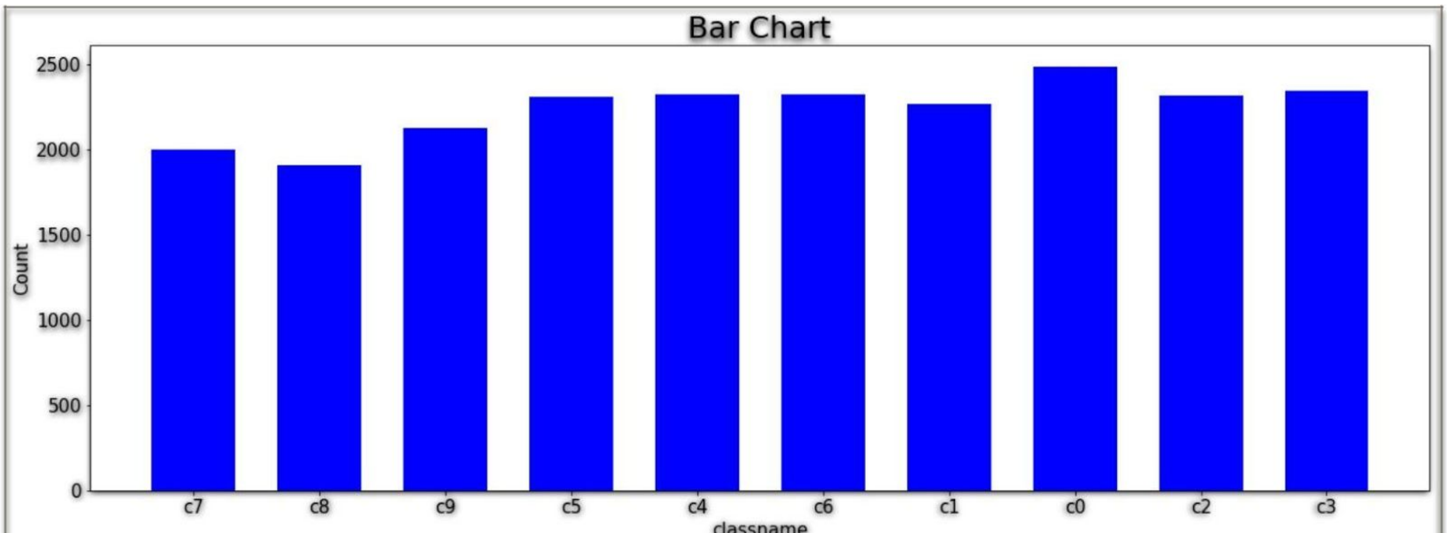


Fig.2 Image Count for each class - Training dataset

Benchmark

The benchmark that has been used for this model is multi-class logarithmic loss of 0.337 which is the score of loss among the top 20 percentile people in the kaggle leaderboard.This a relevant benchmark for this problem as this problem has been solved by many people and these are among the top 20 percentile of the people which makes this value a very good benchmark for this problem. Since many data scientists experts have solved this problem and recorded their results on the kaggle ,so ,this makes this benchmark loss of 0.337 very relevant.

Methodology

Data Preprocessing

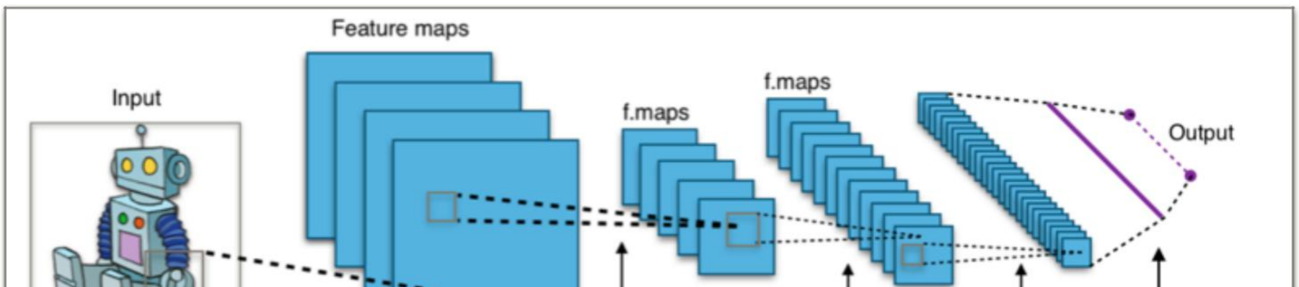
Data preprocessing is the first step and very crucial step in any machine learning model.The data needs to be standardised and normalized so that Feature values are in the same range and also have a zero mean .For example,suppose One feature may be in kg and other in inches.
Data preprocessing can be done in the following steps:
1.First divide the dataset into training,validation and testing data.
2.The images are resized to a square image.
3.The images pixels are normalised by dividing every pixel by 255 and mean is made zero by subtracting 0.5 in the numerator.
Now,our dataset is ready to learn it is a good data now(normalized and standardized).
 20% of the training data has been kept as validation set.Since the classes labels are uniformly distributed ,there is no need to change for them.

Implementation

Any Supervised machine learning model aim is to try to find the function which given a X(a set of features) correctly maps to Y(set of labels).There are many different algorithms or functions out there.The convolutional neural networks are being used for this problem due to its complexity which makes the prediction more accurate and also the amount of data given is very large so it can generalize more better.

4 convolutional network layers are used with 4 max pooling layer out of which two are fully connected convolutional layers.we have to learn optimal weights and biases so that the log-loss function error is minimum.softmax function is used in the output layer so to make the sum of the probabilities equal to 1 and ReLu in other layers.

Convolutional neural networks are set of deep,feed-forward neural networks that have find tremendous success especially in imagery data like videos and images.The first layers(zeroth layer) is called the input layer (here the set of inputs or rows of the dataset are feeded),then we are connected to the other layers by the weights and biases.We aim to find weights and biases such that the log-loss(objective function) is minimised.The gradient descent algorithm along with back propagation is used in low level to minimize the log loss error .Since the objective function is continuous and differentiable so,gradient descent can be easily applied .Backpropagation is used so that we don't have to make recomputations.weights and biases are updated in the direction opposite to the direction of gradient and are updated until the log loss error is minimum.



Typical CNN architecture(File:Typical cnn.png)

Aphex34 (https://commons.wikimedia.org/wiki/File:Typical_cnn.png), https://creativecommons.org/licenses/by-sa/4.0/legalcode

Convolutional layer(CNV) :*The convolutional layer is the core building block of a CNN. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map(refer Fig.4).*

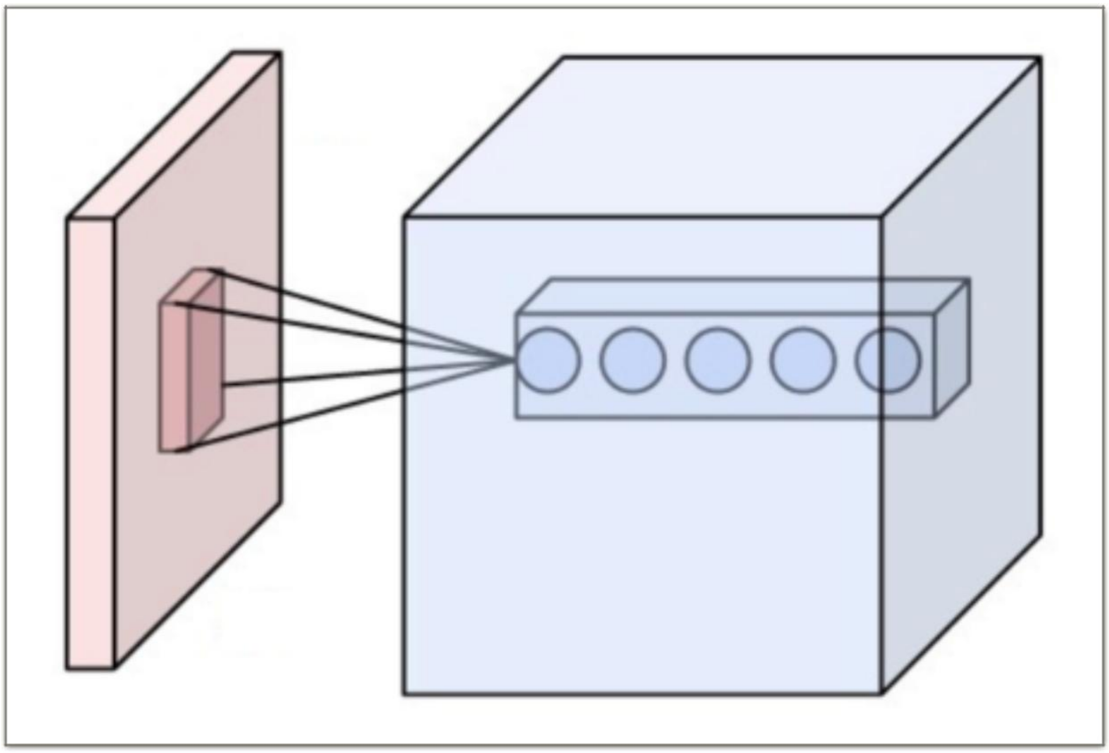


Fig.4 Neurons of a convolutional layer (blue), connected to their receptive field (red)

(File:Conv layer.png. (2016, August 18))

Pooling layer(PL) : Another important concept of CNNs is pooling, which is a form of non-linear down-sampling.
Other than convolutional layers conv. Nets also uses pooling layer to reduce the size of their representation ,to speed up computation as well as make the features more robust.There are several non-linear functions to implement pooling among which max pooling is the most common.For example,consider the below figure ,we have a 4 x 4
Input and we want to apply pooling to convert into 2 x 2 .This can be done by taking max of each 2 x 2 in 4 x 4 layer so it comes out to be the result as shown in the figure.This is known as max pooling.

In addition to max pooling, the pooling units can use other functions, such as average pooling or L2-norm pooling.

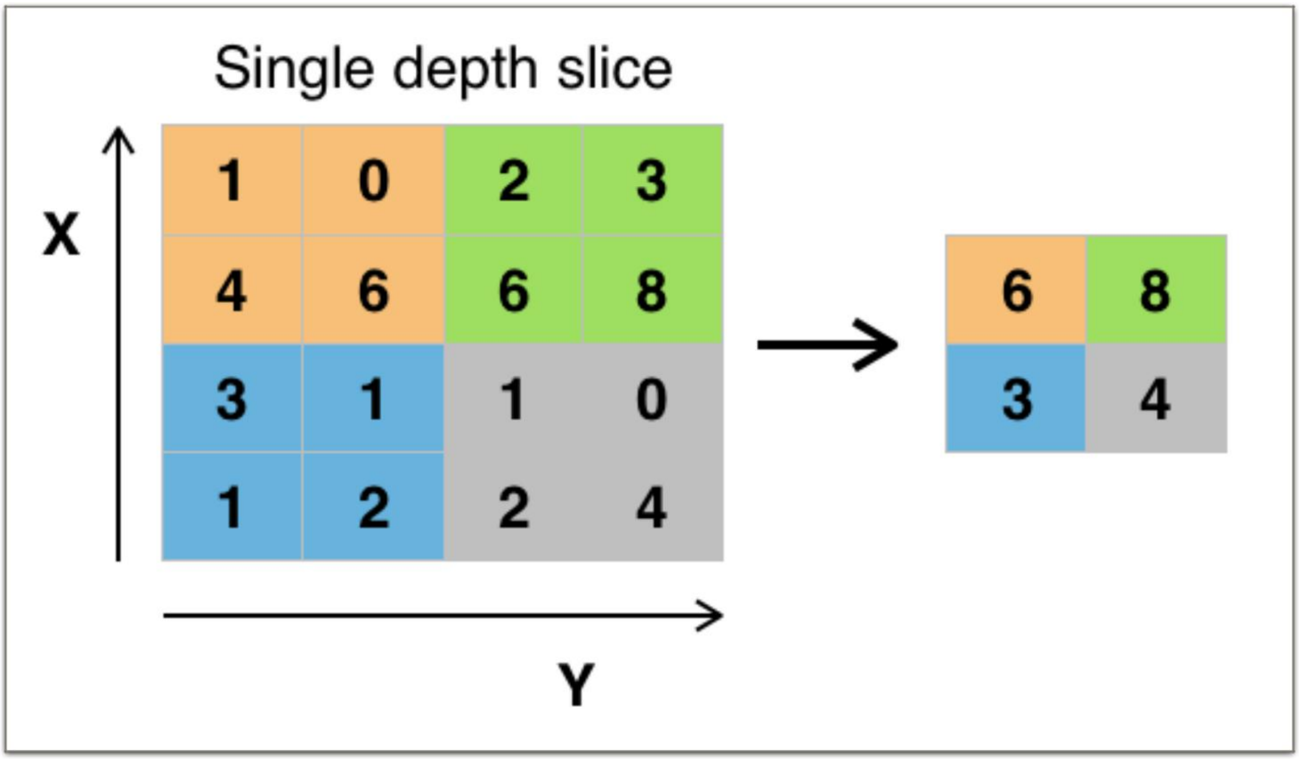


Fig.5 Max pooling with a 2x2 filter and stride = 2(File:Max pooling.png. (2016, August 18))

Aphex34 (https://commons.wikimedia.org/wiki/File:Max_pooling.png), https://creativecommons.org/licenses/by-sa/4.0/legalcode

A simple CNN (Convolutional Neural Net) shown in Fig.7 is created as a initial step.This is then trained with training data.

Following are the steps carried out for training a CNN:

1. The entire dataset i.e. train, validation and test dataset is pre-processed as discussed above(Refer “Data Preprocessing” section).

2. A simple CNN is created to classify images. The CNN consists of 4 convolutional layers with 4 max pooling layers in between.

3. Filters were increased from 64 to 512 in each of the convolutional layers and drop out is added

4. Flattening layer along with two fully connected layers were added at the end of the CNN

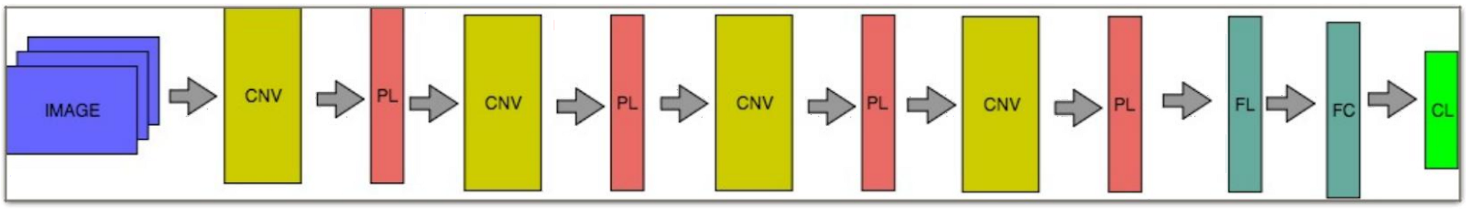
5. Number of nodes in the last fully connected layer were setup as 10 since the number of categories is equal to 10 ,using softmax activation function. This layer is being used as classification layer.

6. ”Relu” activation function was used for all other layers along with Xavier initialization.

7. The model is compiled with ‘rmprop’ as the optimizer and ‘categorical_crossentropy’ as the loss function.

8. The model is trained for 30 epochs with a batch size of 40.During training process the model parameters were saved when there is an improvement of loss for validation dataset.

9. Finally predictions were made for the test dataset and were submitted in suitable format.

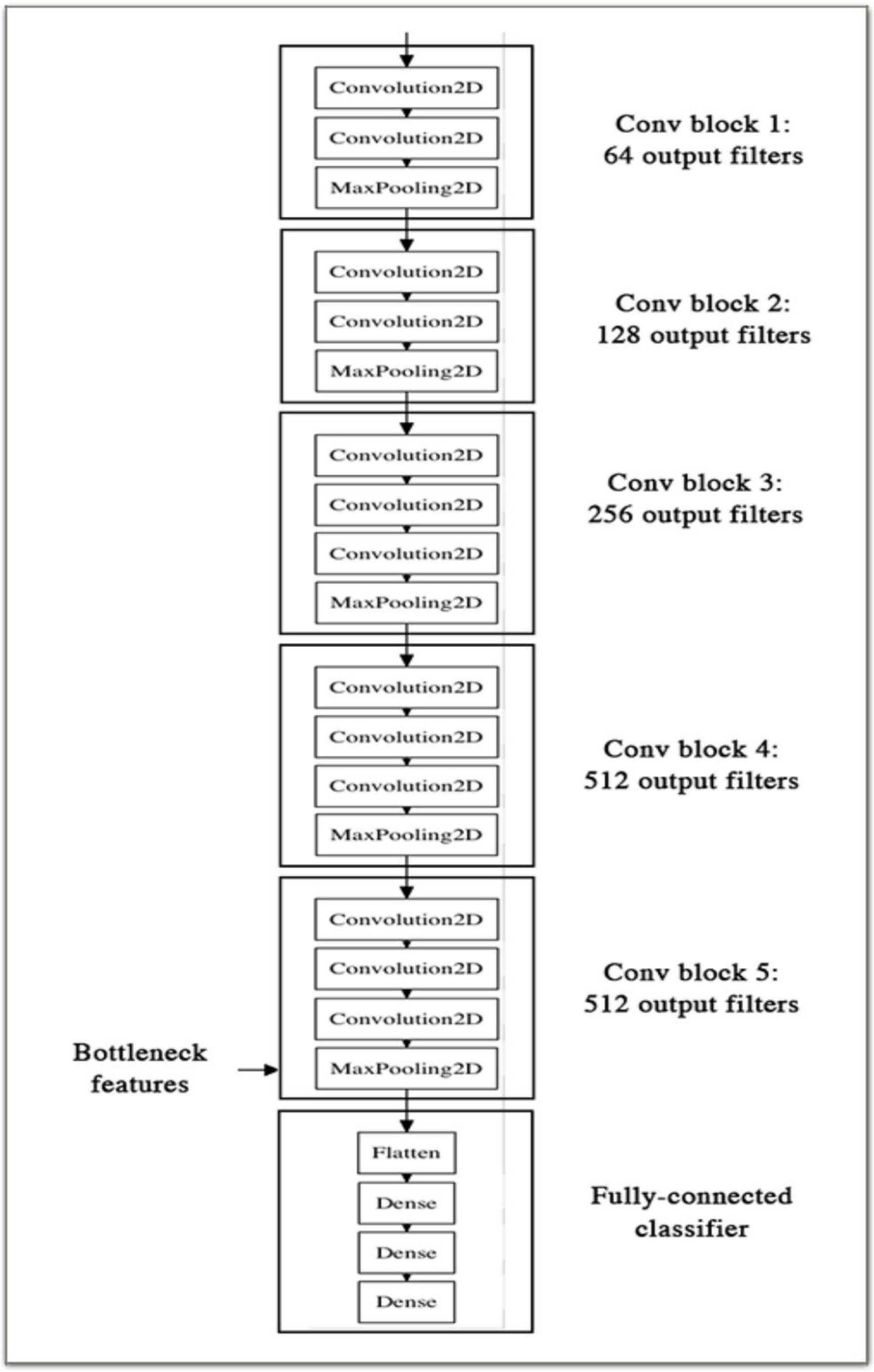


Transfer Learning : In this technique instead of training a CNN from scratch a pre-trained model is used as an initialization or scratch ,an already pre-trained model is used as an initialisation for weights .There are two types of transfer learning techniques as shown below

ConvNet as fixed feature extractor : In this technique a pre-trained network is initialised and the last layer which is fully connected is removed and after removing it this can be treated as a fixed feature extractor and once these fixed features were extracted the last layer is trained with these extracted fixed features.

As shown below in Fig.8 instantiate the convolutional part of the model. Execute the model on training and validation data once, recording the output (the "bottleneck features" from th VGG16 model: the last activation maps before the fully-connected layers) in two numpy arrays. Then a small fully-connected model is trained on top of the stored features.

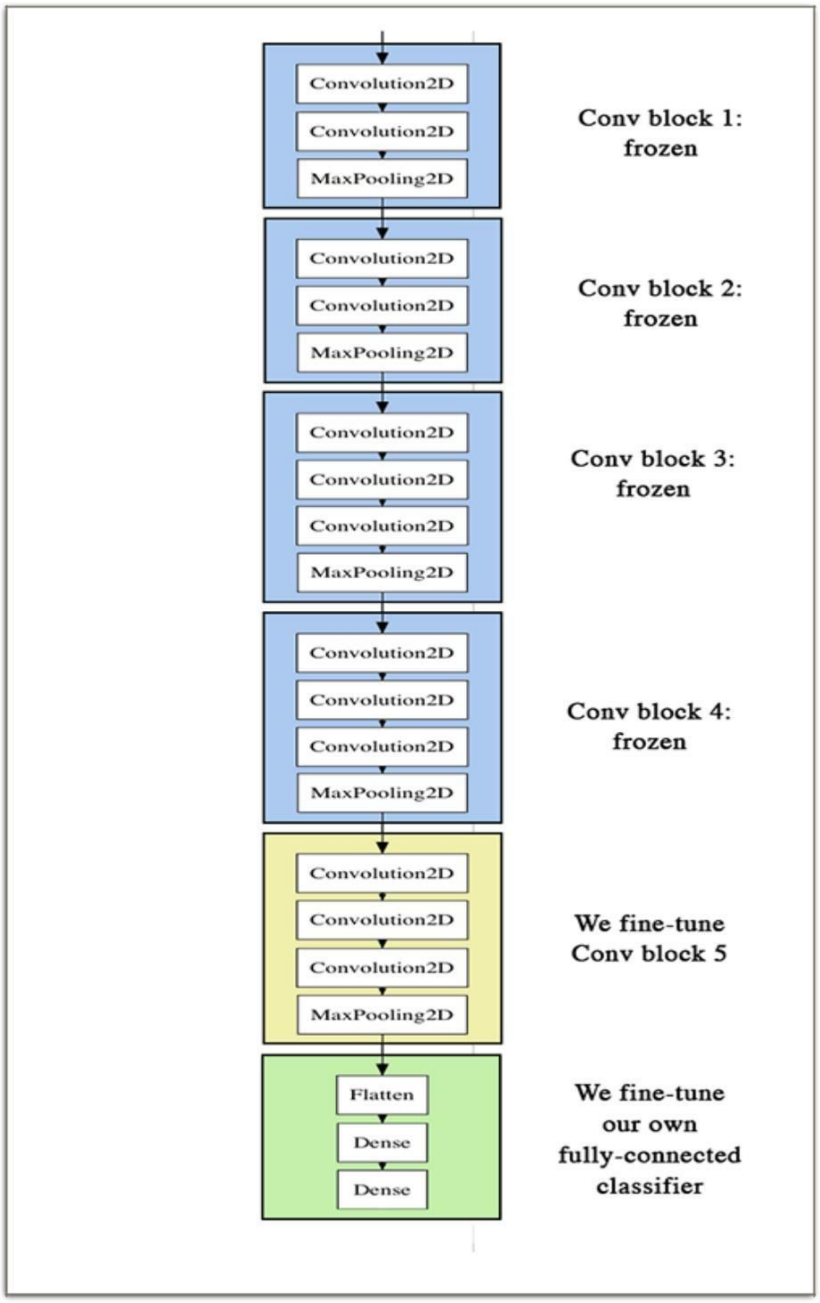
The below figure shows the instantiate convolutional part of the model. Executing the model on training data and validation data once ,then recording the output so called the “bottleneck features” from the VGG16 model:the last activation maps before those fully-connected layers in two numpy arrays.Then training of small fully-connected model takes place on the top of stored features.



Fine-Tuning the ConvNet:The second strategy is to not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine tune the weights of the pretrained network by continuously applying backpropagation.It is possible that we can fine tune all the layers of the convNet or we can also keep some of the earlier layers fixed so to manage overfitting concerns and so only fine -tune some higher-level portion of the network and not the whole network.

The above thing is motivated by the observation that the earlier features of the ConvNet contain more generic features that are useful to many tasks but later layers are driven more by the classes or details of the classes contained in the original dataset.

The below figure shows the instantiate convolutional base of VGG16.Adding previously defined layers which are fully-connected on top and then loading its weights.Freezing the layer of the VGG16 model up to the last convolutional block and than fine tune the model.This can be achieved or is done with a very slow learning rate and SGD optimizer is preferred over adaptive learning rate like RMSprop.This ensures that the magnitude of the updates is very small so that we are not wrecking the previously learned features.



Python and Keras with Tensorflow as the backend were used for carrying out the coding part.A large amount of memory was used when the images were uploaded due to the size of images and the quantity or large amount of dataset .Training was very slow it may be account due to the factor that learning rate was set to a very low value.

Refinement

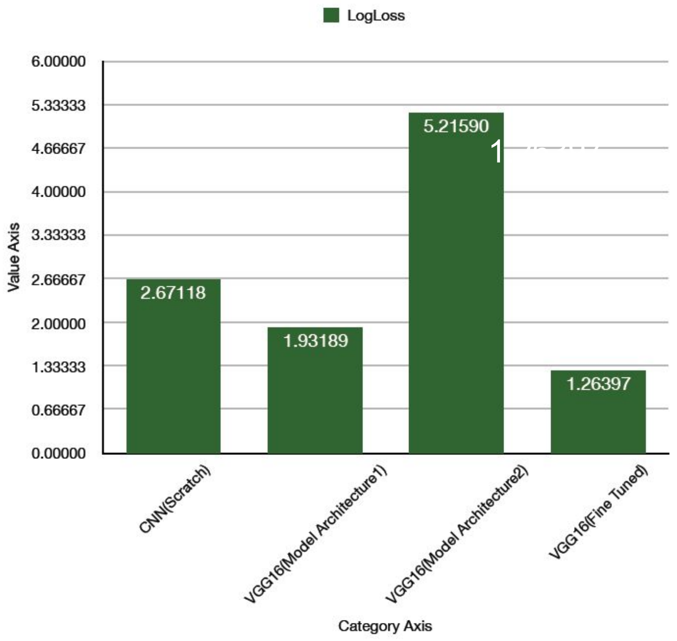
To get the initial result simple CNN architecture was built and evaluated. This resulted in a decent loss. The public score for the initial simple CNN architecture(initial unoptimized model) was 2.67118.
When simple CNN was tested without tuning parameters a loss of 2.67 was observed .
To improve this loss ,Transfer learning was applied to VGG16 along with investigating two types of architectures for fully connected layers.Model architecture one showed good results and was improved further by carrying out techniques like to account for good initialization of weights and biases xavier initialization was used and drop layers was added so that the model is not prone to overfitting to the training data and we increase the size of epochs and batches so that learning is optimum.Also we applied momentum gradient descent to account for the slow learning of the model (i.e model being stuck where gradient is low) we used (gamma=0.9) for this.The finely tuned parameters gave very good results and reduced the error from 2.67 to 1.26(approx).

Results

Model Evaluation and Validation

During the development of model i.e learning ,model was evaluated on
the validation dataset .

The comparison of the Public Scores for all the model architectures
considered for this data set is shown in below figure.



The final architecture chosen was the VGG16 fine-tuned model. This architecture along with hyper-parameters were chosen as they performed the best among all model combinations.

Following are the details of the final model parameters and training process :

1. VGG16 is instantiated and first 15 layers were frozen.

2. The last Conv layer i.e. Conv block 5 is fine tuned.

3. Our own layer(Global Average Pooling + Fully Connected layer) is added and fine tuned

4. Fine tuning is carried out with very slow learning rate and SGD optimizer

5. Training is carried out for 10 epochs with a batch size of 16

Finally the model is tried on the test data set. This resulted in Public Score of 1.26397. This can result in rank of 617 out of 1440 in Public Leaderboard i.e. in top 42.84%. It was observed the loss on validation dataset was

0.00751.Comparing with benchmark model the loss error is relatively high in comparison to the benchmark model value of 0.337 .This is not as significant solution as the other solutions on kaggle.This difference may be due to the image resizing to 224 x 224. This in comparison with public score on test dataset would suggest that there is overfitting. In Order to resolve overfitting we need to consider adding/increasing the drop out and L2 regularization.

Justification

The Finely tuned model has reduced the loss significantly to 1.260(approx) which is very less error and model is approximately generalized and due to this very low error this can be used by the insurance company to determine whether the driver is a risky driver or not.if clear images are provided the model is likely to perform much more better and give very good results,otherwise also the results are very good .

In order for further improve the model and reduce the loss, we need to take advantage of different algorithms and hardware(Refer “Improvement” section).

Conclusion

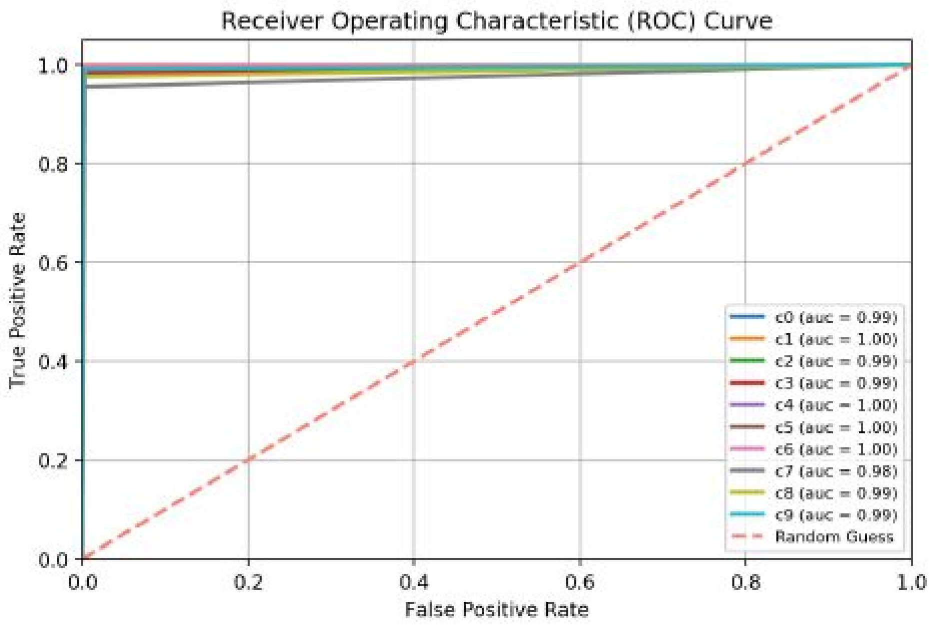
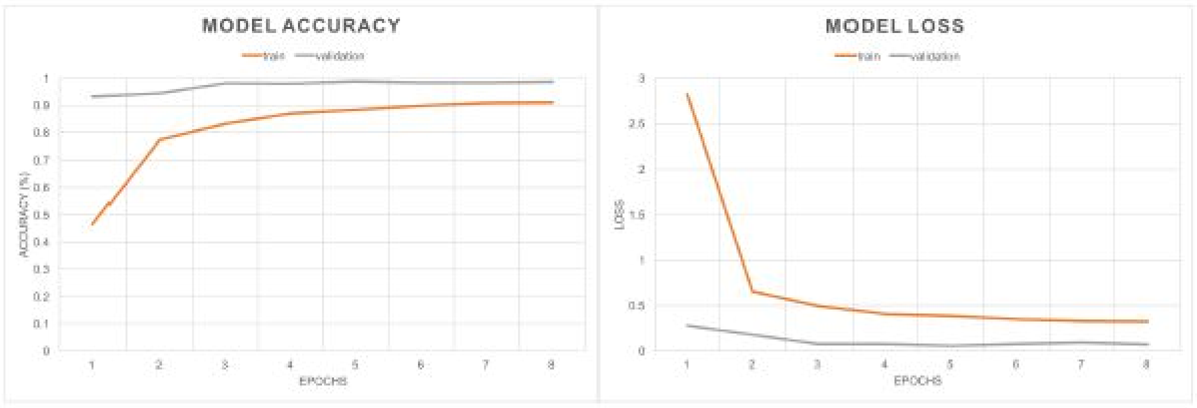


Figure 7: ROC Curve of distract driver predictions



(a) Model Accuracy (b) Model Loss

Figure 8: VGG16 Top Retrained - Model Accuracy and Model Loss

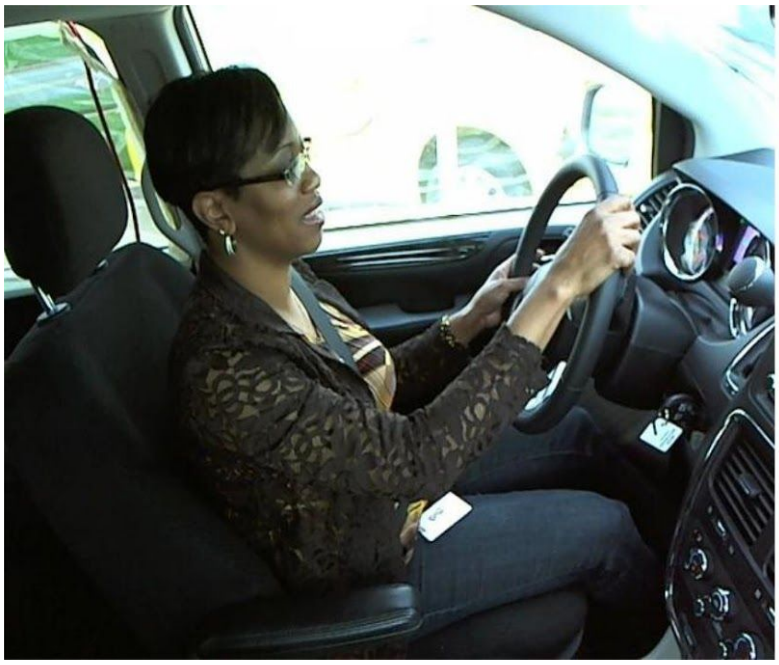
Let us analyze the model by looking at the Receiver Operating Characteristic (ROC) Curve shown in figure 7 which is plotted between True positive rate(TPR)
And False positive rate(FPR).An optimal solution will have FPR equal to zero and TPR equal to 1,which corresponds to the left corner of the graph.A quicker increase to the left corner indicates that we are close to optimal solutions.Also ,there sharp bend in the graph which indicates that there is a near perfect separation.

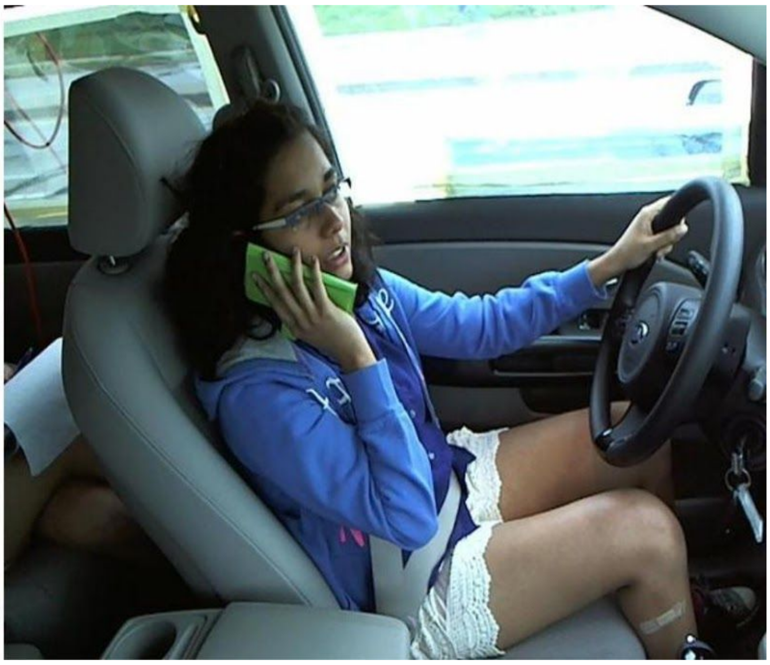
The change in accuracy and the categorical cross entropy 9 (referred to as simply Loss) can be observed in Figure 8a as the model is trained through the different epochs.

Both, the training and validation accuracies, progressively increase through the number of epochs in Figure 8 until they reach some threshold, indicating no further improvements are made and causing training to terminate. As the number of epochs increases ,both the training and validation accuracies increases upto a point and tham it becomes constant.

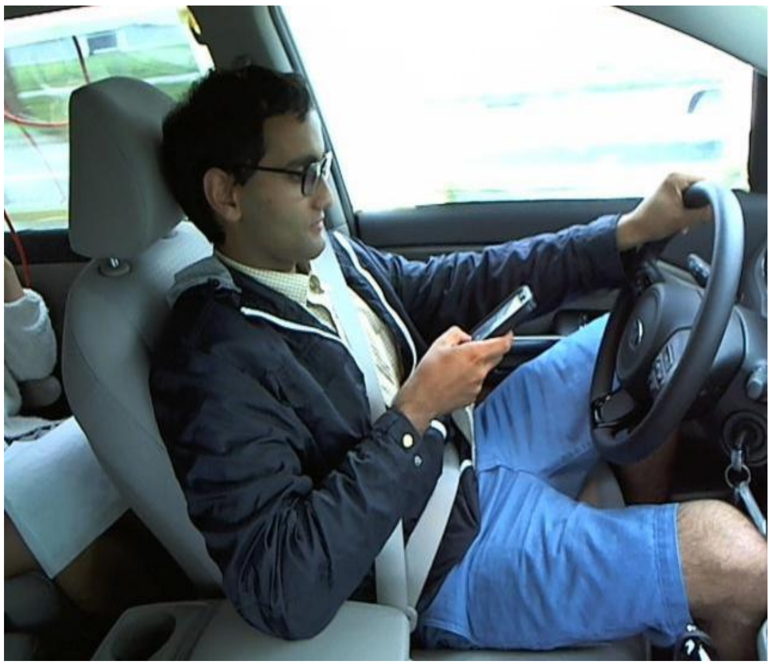
Similarly, in Figure 8b, both the validation and training loss decrease as the model learns, until eventually terminating when the model begins to overt.

Free-Form Visualization





Below are some of images classified by the VGG16 fine-tuned model.(figure 10 and figure 11)



The first image was misclassified as “c3: texting - left" although it is showing as attentive driving. Similarly the second image in Fig.11 was misclassified as “c9: talking to passenger” although it is showing as talking on the phone - right

The image of the man was an example of correctly classified image.
It is correctly classified as “c1: texting - right”

Misclassifications may be due to the following:

- Noise and not a clear image

- Too many objects(Clutter) in the image

Reflection

The process used to define and develop the model can be summarised as follows :

1. An initial problem is identified and relevant datasets were obtained
2. The datasets were downloaded and pre-processed
3. Relevant hardware is obtained from cloud to account for huge number of images in the dataset
4. The highest score in the Public Leadership board was taken as a benchmark and a target rank was defined.
5. An initial CNN was created from Scratch. After training this was tested and resulted in rank of 1362 out of 1440 in Public Leaderboard i.e. in top 94.58%
6. In-order for further improve the loss metric, pre-trained model such as VGG16 was used and transfer learning is applied
7. Technique such as “Using the bottleneck features of a pre-trained network” is applied to VGG16 for two types of model architectures as the top layer
8. “Using the bottleneck features of a pre-trained network” for VGG16 with model architecture1 was trained and tested resulted in rank of 1120 out of 1440 in Public Leaderboard i.e. in top 77.77%.
9. However when “Using the bottleneck features of a pre-trained network” for VGG16 with model architecture2 was trained and tested, did not result in any improvement of loss metric.

10. Finally “Fine-tuning the top layers of a a pre-trained network” was applied to VGG16 with model architecture1 as top layer. This resulted in further improvement of loss metric and when tested resulted in rank of 617 out of 1440 in Public Leaderboard i.e. in top 42.84%

I found steps 8,9,10 to be difficult. This is because of the fact that training and testing took large time despite running on GPU instances. This may be due to model complexity and huge size of data.

The interesting part of the project is the use of transfer learning to obtain good score even with decent sized datasets. Also I also enjoyed using Keras with Tensorflow as the backend due to its ease of use and simplicity.

Improvement

Following are the areas for improvement

1. To further improve the score , we need to consider using bigger size when re-sizing the image. Currently the algorithm was trained by using 224x224 re-sized images. An image size of 480x480 might be relevant as original image size is 640x480.

2. Currently VGG16 architecture was only investigated during Transfer learning. We also need to try using the below pre-trained models and verify the score.

    - VGG-19 bottleneck features
    - ResNet-50 bottleneck features
    - Inception bottleneck features
    - Xception bottleneck features

3. Model Ensembles and use of advanced Image Segmentation algorithms such as R-CNN,Fast R-CNN,Faster R-CNN and Mask R-CNN can also be investigated.

4.In Order to resolve overfitting we need to consider adding/increasing the drop out and L2 regularization.

References

Convolutional neural network. . In *Wikipedia, The Free Encyclopedia*. Retrieved 22:40, October 5, 2017, from https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=802888745

File:Conv layers.png. (2016, January 19). *Wikimedia Commons, the free media repository*. Retrieved 10:50, October 7, 2017 from https://commons.wikimedia.org/w/index.php?title=File:Conv_layers.png&oldid=185037832.

File:Conv layer.png. (2016, August 18). *Wikimedia Commons, the free media repository*. Retrieved

11:16, October 7, 2017

from https://commons.wikimedia.org/w/index.php?title=File:Conv_layer.png&oldid=204200325.

File:Max pooling.png. (2016, August 18). *Wikimedia Commons, the free media repository*. Retrieved

11:21, October 7, 2017

from https://commons.wikimedia.org/w/index.php?title=File:Max_pooling.png&oldid=204200335.

File:Typical cnn.png. (2016, August 18). *Wikimedia Commons, the free media*

repository. Retrieved

11:26, October 7, 2017

from https://commons.wikimedia.org/w/index.php?title=File:Typical_cnn.png&oldid=204200221.

CS231n Convolutional Neural Networks for Visual

Recognition(n.d.).Retrieved from http://cs231n.github.io/transfer-learning/#tf

Francois Chollet(Sun 05 June 2016). Building powerful image classification models using very little data.Retrieved from https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

File:vgg16 _original.png. (n.d.). Building powerful image classification models using very little data. Retrieved from https://blog.keras.io/img/imgclf/vgg16_original.png.

File:vgg16 _modified.png. (n.d.). Building powerful image classification models using very little data. Retrieved from https://blog.keras.io/img/imgclf/vgg16_modified.png.

https://keras.io/

https://www.tensorflow.org/

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

https://arxiv.org/pdf/1411.1792.pdf