Capstone Project

Yekabote Pavan Kumar,

Machine Learning Nanodegree

April 30, 2019

Definition

Project Overview

According to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year.

State Farm hopes to improve these alarming statistics, and better insure their customers, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors. Given a dataset of 2D dashboard camera images, State Farm is challenging Kagglers to classify each driver's behavior. Are they driving attentively, wearing their seatbelt, or taking a selfie with their friends in the backseat?

In this project, a machine learning has been created and refined to detect the activity of driver on given driver images. We do it, by predicting the likelyhood of the driver, doing task from a set of classes, on each given image.

Problem Statement

A dataset of 2D images containing a driver, performing an activity, are given. An algorithm has to be developed to classify the activity of driver within the given classes and determine if they are driving attentively, or getting distracted taking selfies and performing other interruptive activities.This can be used to automatically detect drivers engaging in distracted behaviour, while driving, from dashboard cameras.

Steps involved in achiving the expected result are as follows:

- Preprocess all driver images to make it as usable dataset
- Build and train a deep learning model to classify driver images
- Test the trained model and make changes for further improvements

Our model will be built and trained such that, the obtained categorical cross entropy loss will be among top 50% of Public Leader Board Submissions in Kaggle.

Metrics

The submissions are evaluated based on the categorical cross entropy loss, which is also called as multi class log loss.

Multiclass Log loss = $$-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\, ln\,(p_{ij})$$

where

- n is the number of images in the dataset
- m is the number of class labels in dataset
- ln is the natural logarithm
- y_{ij} is 1, if observation i belongs to class j, otherwise, 0
- p_{ij} is predicted probability that observation i belongs to class j

The deep learning model takes in an image and tries to predict the class of it. Then the predicted class is compared with the image's true class, for which we apply the above formula, and see how far is our predicted class to actual true class

The metric multi-class logarithmic loss is selected than accuracy as this gives the probability of the predictions than simply saying yes or no.This gives a more nuanced view of the model and performance. Also F1 score may not be appropriate metric as it is focussed on measuring the number of true positives against total number predicted as positive, number of actual positives. In addition to this F1 score also depends on the threshold used to identify each class. Hence multi-class logarithmic loss is the most relevant metric for this problem.

Then we will try to minimize the loss , using any of the optimizer functions such as Stocastic gradient descent, Adam etc.

Analysis

Data Exploration

Driver images have been captured, where each image contains picture of a driver doing some activity in car. For training, the images containing an activity from any of below classes, will be placed in that class named folder.

The 10 classes to predict are:

- 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

Following are the file descriptions and the URL's from which files have been obtained.

- Imgs.zip - A zip file containing all train and test images
- driver_imgs_list.csv - A list of training subject ids, class names and respective images.

- sample_submission.csv - A sample format for submission to calculate the evaluation metrics.

The following link will download a zip file of all above listed files:
https://www.kaggle.com/c/5048/download-all

There are 22424 total images. Of these 17943 are training images,4481 are validation images. All the training, validation images belong to the 10 categories shown above.The images are coloured and have 640 x 480 pixels each as shown in following figures.

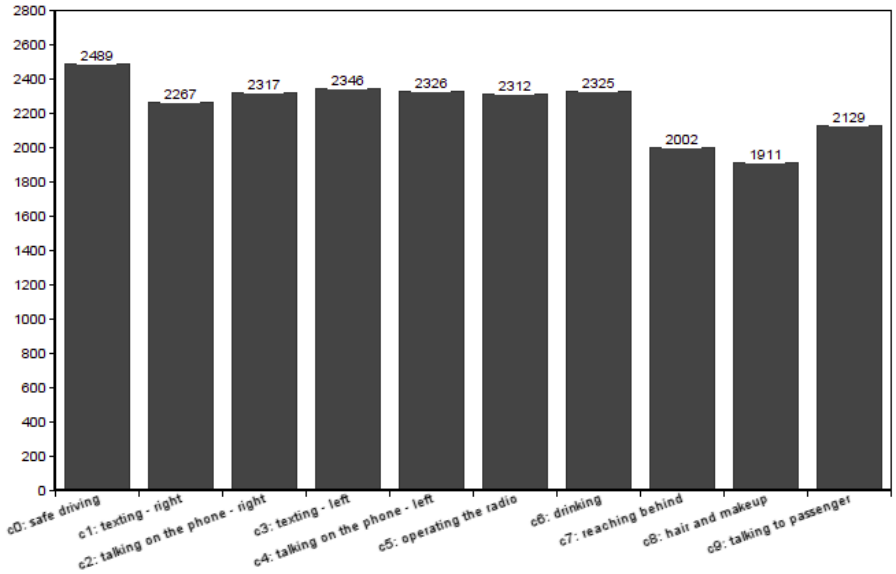
c0:Safe Driving

c3: texting – left

Mostly all the images in the dataset are good and no image is blurred. This dataset seems to be clean and effective one to start our CNN training process.

Exploratory Visualization

Following is the bar graph of count of images present in each class, which we are about to use for training and validation set.



1) Count of images in each class.

Benchmark

The best model in public leader board score, has categorical cross entropy loss of 0.08689. We are going to take this as our benchmark model. Our model will be built such that, the obtained categorical cross entropy loss will be among top 50% of all submissions made.

Following is the link to public leader board score:
https://www.kaggle.com/c/state-farm-distracted-driver-detection/leaderboard

Methodology

Data Preprocessing

Data preprocessing is carried out before model is built and training process is executed. Following are the steps carried out during preprocessing.

- Initially the images are divided into training and validation sets
- The images are resized to a square images i.e. 224 x 224 pixels
- All the three channels are used during training process as these are color images
- The images are normalised by dividing every pixel in every image by 255

Implementation

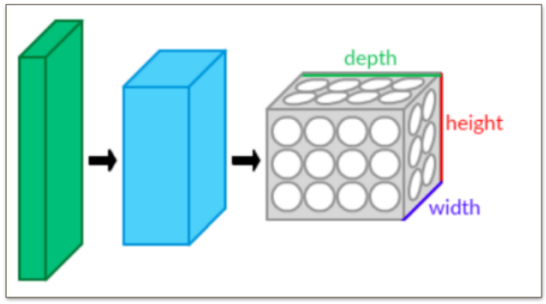
Below is the brief description of CNN's and different layers as specified in
https://en.wikipedia.org/wiki/Convolutional_neural_network

CNN Architecture :

In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.

Convolutional networks were inspired by biological processes in which the connectivity pattern between neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage. They have applications in image and video recognition, recommender systems and natural language processing. A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers are either convolutional, pooling or fully connected.



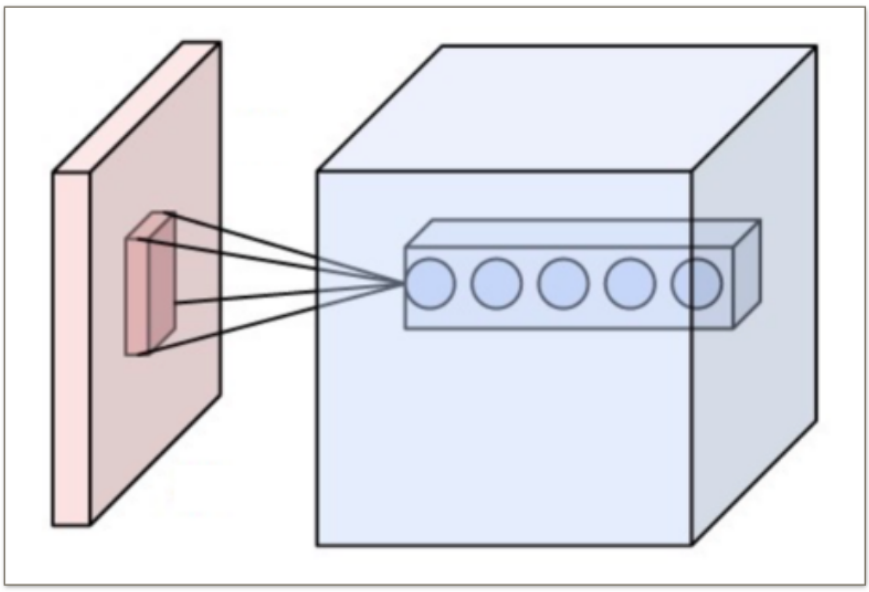
2) CNN layers arranged in 3 dimensions

A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function (as shown in above figure). A few distinct types of layers that are commonly used, are discussed below.

Convolutional layer(CNV) :

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. 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 in the below image.



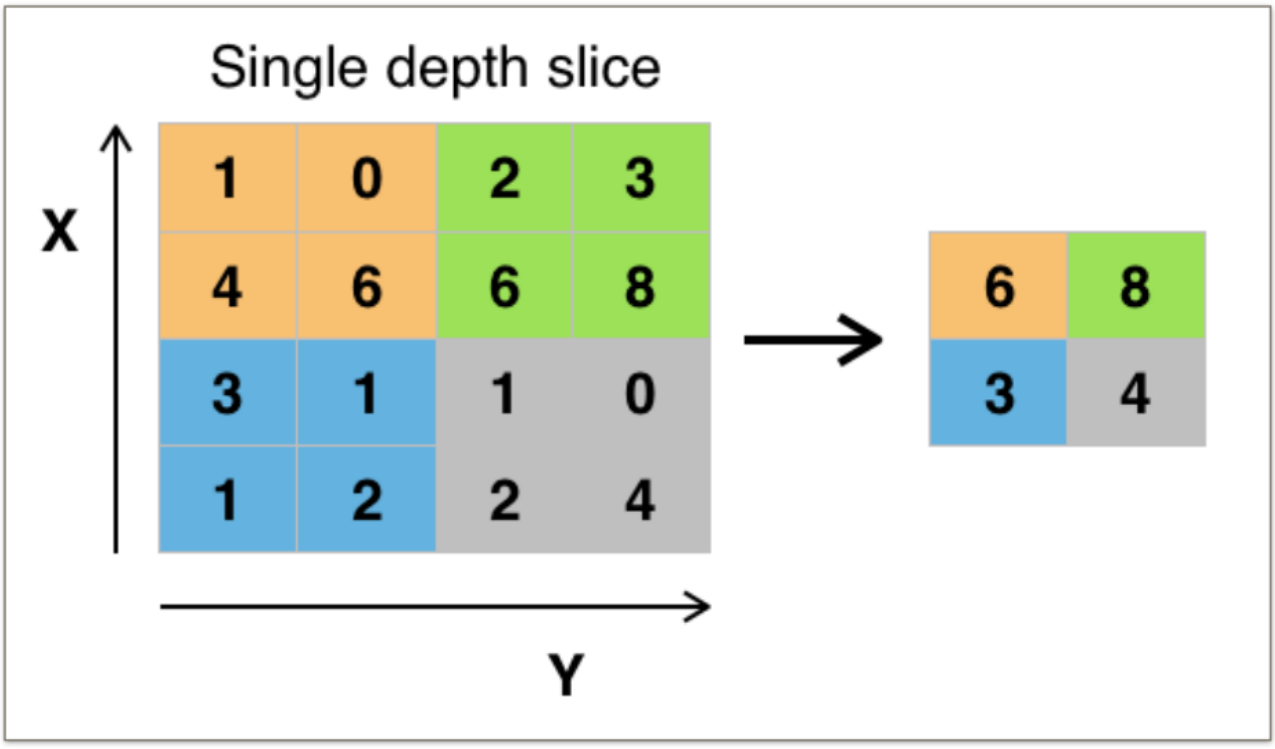
3) Neurons of a convolutional layer (blue), connected to their receptive field (red)

Pooling layer(PL)

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture.

The pooling operation provides another form of translation invariance. The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations. In this case, every max operation is over 4 numbers. The depth dimension remains unchanged.This is shown in the following figure.

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



4) Max pooling with a 2x2 filter and stride = 2

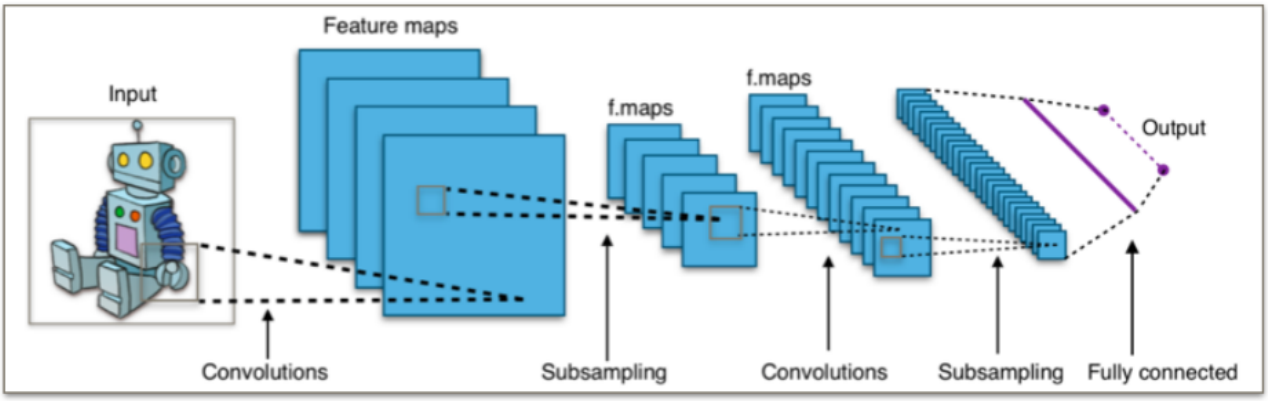
Fully connected layer(FC)

Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via full connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

Classification Layer(CL)

The classification layer specifies how training penalizes the deviation between the predicted and true labels and is normally the final layer. Various loss functions appropriate for different tasks may be used there. Softmax loss is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values in [0,1]

A typical CNN architecture is shown below.



5) Typical CNN architecture

A simple CNN has been created initially and trained on the given images dataset. Following is the explanation of the CNN.

1. The dataset is loaded using a generator which loads only a set of images from given dataset in system, because loading entire dataset into memory will cause to out of memory issue and may slow down system performance.
2. First an input layer with 224 x 224 x 3 dimensions have been added
3. Then a Convolution layer with 8 filters and kernel size of 2 is added.
4. The third layer is a convolution layer with 16 filters, kernel size of 2 and relu activation is added
5. The fourth layer is a convolution layer with 32 filters, kernel size of 2 and ReLu activation is added
6. The fifth layer is a MaxPooling layer, which chooses all max pixel values in each region
7. The sixth layer is a Convolution layer with 64 filters, kernel size 2, and ReLU activation is added
8. The seventh layer is a convolution layer with 128 fitlers, kernel size 2 and ReLU activation is added
9. The eighth layer is a Convolution layer with 512 filters, kernel size 2, and ReLU activation is added

10. The nineth layer is a Global Average Pooling layer which picks up the Average values of entire image and flattens the image to a vector
11. Then a fully connected layer with 500 nodes and ReLU activation has been added as tenth layer
12. The eleventh layer is a dropout layer with 10% dropout probability
13. The twelveth layer is a fully connected layer with 100 nodes and ReLU activation has been added
14. The thirteenth layer is a dropout layer with 10% dropout probability
15. The final layer is the Fully connected layer with 10 nodes correspoding to 10 classes with softmax as activation.
16. Then the model is compiled with Adam optimizer as optimization function and categorical cross entropy as loss function.
17. The model is then trained for 10 epochs with a batch size of 32. During the training process model parameters are saved when there is an improvement in loss of validation dataset.
18. Finally predictions are made for test dataset and were submitted in suitable format.

While building this model, I have tried many values for filters, because at the end filters are the ones which will see the object that we want our model to learn. After a lot of iterations, finally i found this architecture was better learning. So I chose this architecture for my scratch CNN model.

Transfer Learning

In this technique instead of training a CNN from scratch a pre-trained model is used as an initialization or fixed feature extractor. Below are two types of transfer learning techniques.

ConvNet as fixed feature extractor

In this technique a pre-trained network is initialized and the last fully-connected layer is removed.After removing this is treated as a fixed feature extractor.Once these fixed features were extracted, the last layer is trained with these extracted fixed features.

Fine-tuning the ConvNet

In this technique in-addition to replacing and retraining only the classifier, the weights of the pre-trained network are also fine-tuned. “It is possible to fine-tune all the layers of the ConvNet, or it’s possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a ConvNet contain more generic features (e.g. edge detectors or colour blob detectors) that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset.”

Refinement

To get the initial result, the simple CNN architecture was built, trained and evaluated. This resulted in decent loss. The public score for initial simple CNN's loss was 2.25145.

After this to improvise the predictions by decreasing loss, transfer learning technique was applied using VGG16 model.

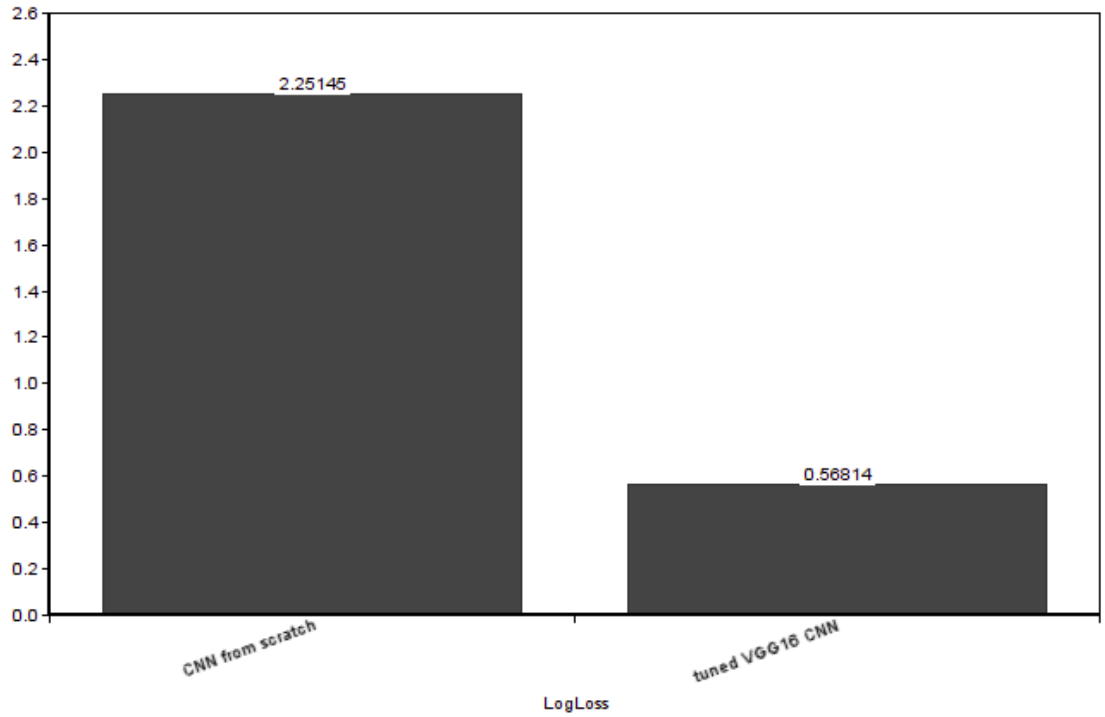
Following is the modifications made to the top FCN's of VGG16 model.

1. Initially load a VGG16 model with all convolution layers loaded with pretrained weights
2. Then a Dense layer with 5000 nodes, ReLU activation and L2(0.00001) regularisation was added
3. A dropout layer with 10% of dropout probability is added
4. A dense layer with 500 nodes, ReLU activation and L2(0.00001) regularisation was added
5. A dropout layer with 10% of dropout probability is added
6. Finally, a dense layer with 10 nodes, Softmax activation was added.
7. The model is then compiled with SGD as optimization function and categorical cross entropy as loss function.
8. Then the model was trained only for 6 epochs, which gave a significant improvement in prediction on validation dataset (accuracy around 96.54% on validation set)

Results

Model Evaluation and Validation

Finally this trained VGG16 with custom top layer model was used to predict on test dataset and the output is submitted in stated format. Surprisingly the loss was 0.56814 on public score, which makes the model's position around less than top 24.5% in the public leader board.



As above we can see that, the final model tuned VGG16 CNN is giving a least log loss value and 96.54%

accuracy on validation set, which actually means that it has learnt a lot better and faster. Faster is because, it has given these values only in 6 epochs which is too impressive and significant. Therefore this model can be said as best and robust model.

Justification

The selected model resulted in a decent public score. This can be used in a mobile application where the driver behaviour can be detected. This can aid in the safety of the driver and can help insurance companies identify risky driver.

In order for further improve the model and reduce the loss, we need to take advantage of different algorithms and hardware

Conclusion

Free Form Visualization

Below are some of the images classified by the VGG16 tuned model

c3: texting – left

c0: safe driving

c3: texting – left

As we can see above, our VGG16 tuned model has perfectly classified all the images to their respective classes.

Reflection

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 81.38% of Public Leaderboard
6. In-order for further improve the loss metric, pre-trained model such as VGG16 with "Fine-tuning the top layers" was used and transfer learning is applied
7. This pretrained fine tuned model has significantly improved the loss metric and when tested resulted in 354th place out of 1440 in public leader board i.e, top 24.5%

The most interesting part of this project was, using VGG16 with some fine tuned top layers, and using SGD instead of Adam optimizer. The results, I have seen after changing the optimizer to SGD was to amazing. The loss got reduced in a very short period i.e., only 6 epochs have given around 96.54% of validation accuracy. This was mind blowing

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.