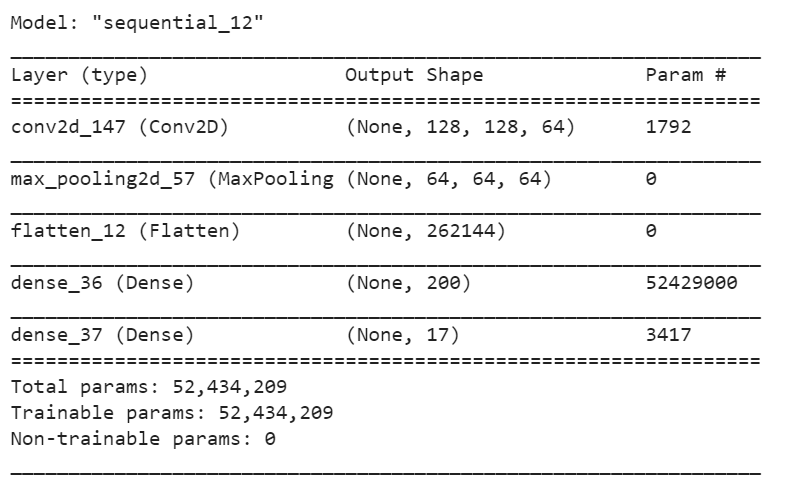
# Deep Learning

## Assignment 2

### Karthik Murugadoss – R00183157

#### Part A

1. Implementation of Baseline Model with one convolution layer, one max pooling, one fully connected layer and one softmax layer.



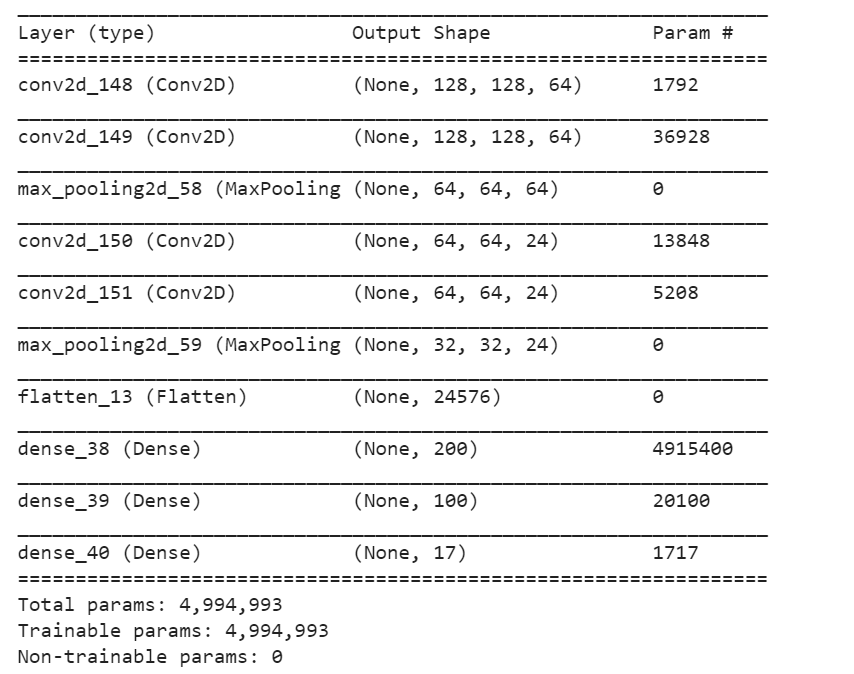
**Performance of the Baseline layer**



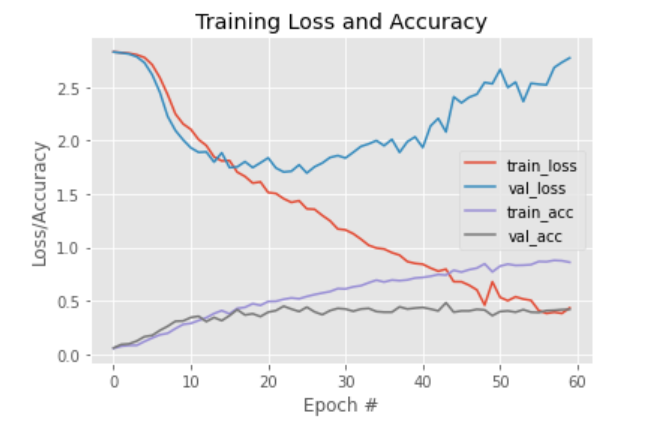
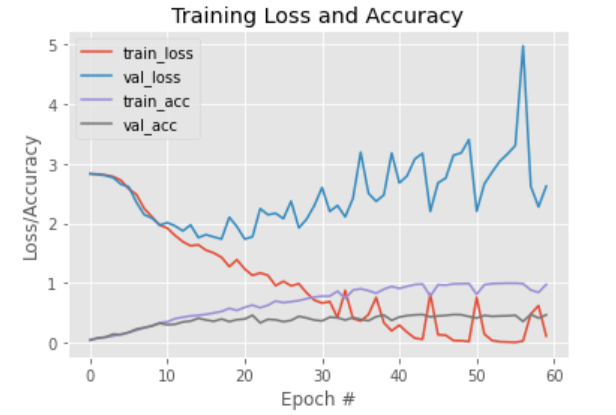
From the performance graph, we can see that the baseline CNN model seems to significantly overfitting, while the train loss came down to nearly zero but the validation loss was very high.

In the next step, we will build denser networks to see if the overall accuracy can be reduced.

1. First Denser CNN network. (CNN1)



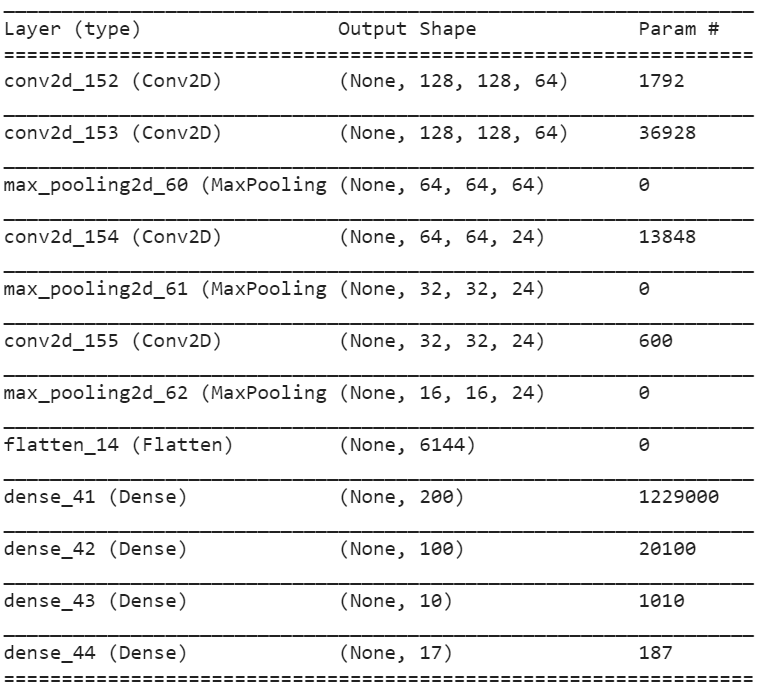
Performance of the network



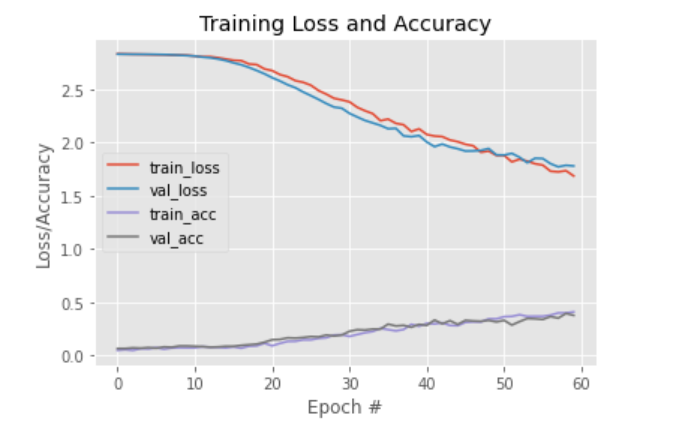
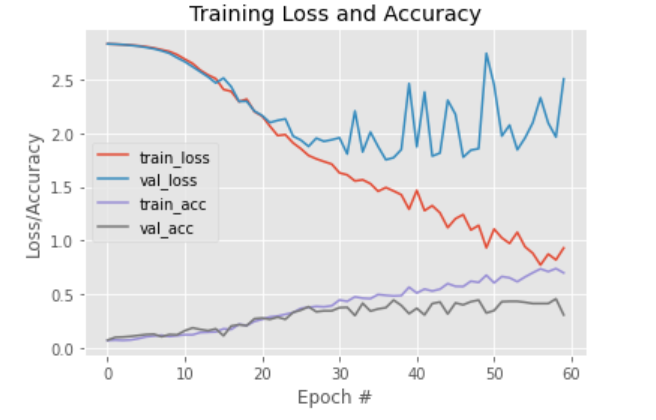
Without dropout With dropout

Even with a denser layer the overfitting hasn’t reduced. Hence dropout was introduced to see if the model performance improved. The network performance did improve a little bit that can be observed by the reduction in noise in validation loss curve in the second graph.

1. Second denser network. (CNN2)

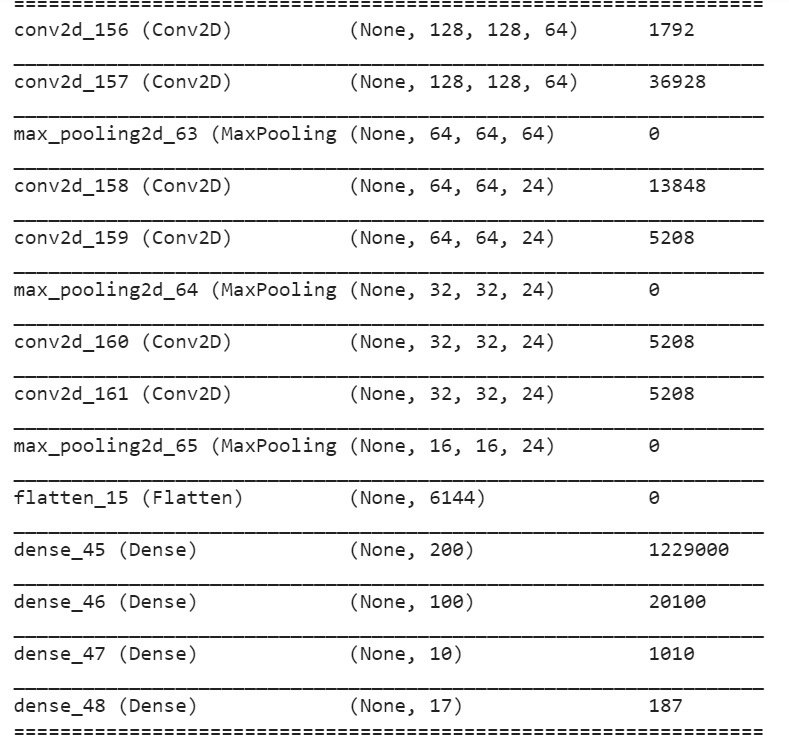


Performance of the second denser network

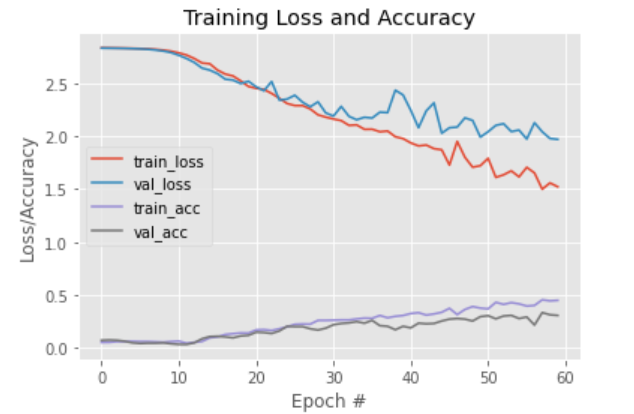


The denser the network became the overfitting also increased along with the increase in the performance of the network to tackle it drop out was introduced and the graph on the left shows how drop out layers has decreased the overfitting substantially.

1. Third Denser layer (CNN3)



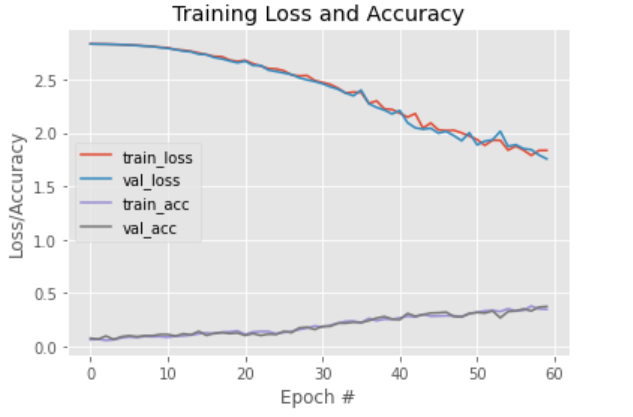
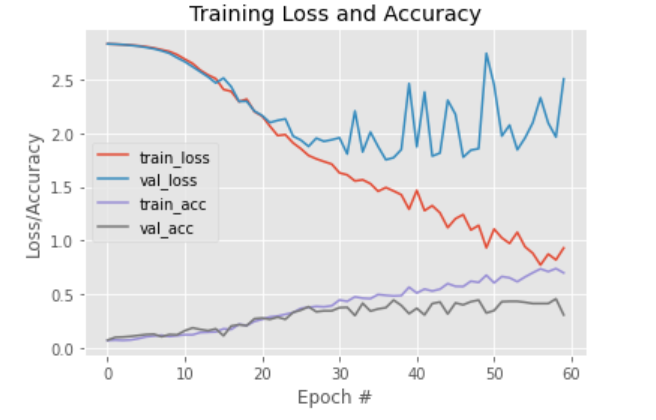
Performance of the model :



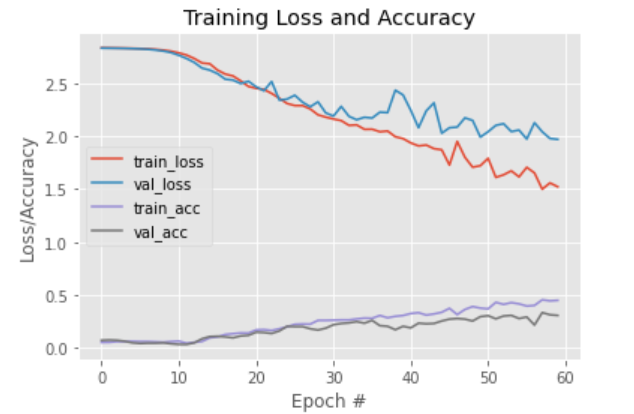
Comparison of all the three models along with the baseline model:

From the comparison we can observe that the baseline model had a much more validation accuracy than the rest of the networks but they seem to be overfitting tremendously. Drop out did reduce the over fitting but it did not improve the accuracy of the network. This is **due to the less amount of data in the training dataset.**  To counter this problem we would add data augmentation and see how the model react. For that we would be taking two of the denser layer which would be CNN2 and CNN3.

Comparison of CNN2 with Data Augmentation and without data augmentation



The left graph shows the CNN2 network without data augmentation and the right one shows the trend with data augmentation. We can see from the graph that even without dropout the network did not overfit which concludes that neural networks are data hungry and the more data it has the better it performs. The same goes with the CNN3 which can be observed in the below pair of graphs.



We can see that the validation accuracy is gradually increasing without much oscillation and inline with training accuracy which is a good sign. Due to time constraints and computational challenges, the models has been trained only for 60 epochs.

(II) Building an CNN Ensemble

To do this vgg16 architecture has been used and three models was initialized with the same architecture.

Methodology used

A picture containing text

Description automatically generated

From the above network architecture three base learners are initialized. Since it is a deeper architecture only three base learners are included and trained to their best.

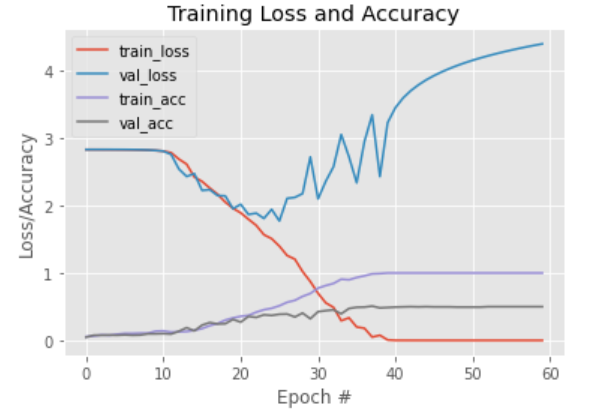
**The source of variability in your ensembleand why variability is an important factor when building an ensemble**?

The Source of variability comes from the fact that everytime the model is initiliazed, even though it is the same architecture, the weights are randomly initialized which introduces the variability in the ensemble models.

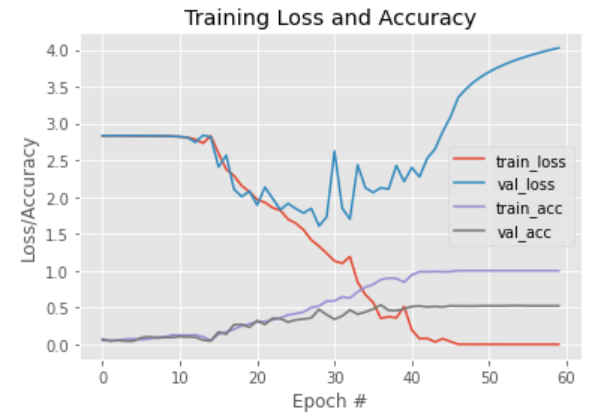
Variability is important factor in building an ensemble because of the way it works. We get a number of models and pick the results based on max voting, in this case we do it based on average. If there is no variability all the models would reproduce the same results and the purpose of using ensemble fails.

**Compare the performance of ensemble with each of the base learners**

**Base learner 1 performance**



**Base Learner 2 Performance**



**Base Learner 3 Performance**



**Part-B**

Part-I

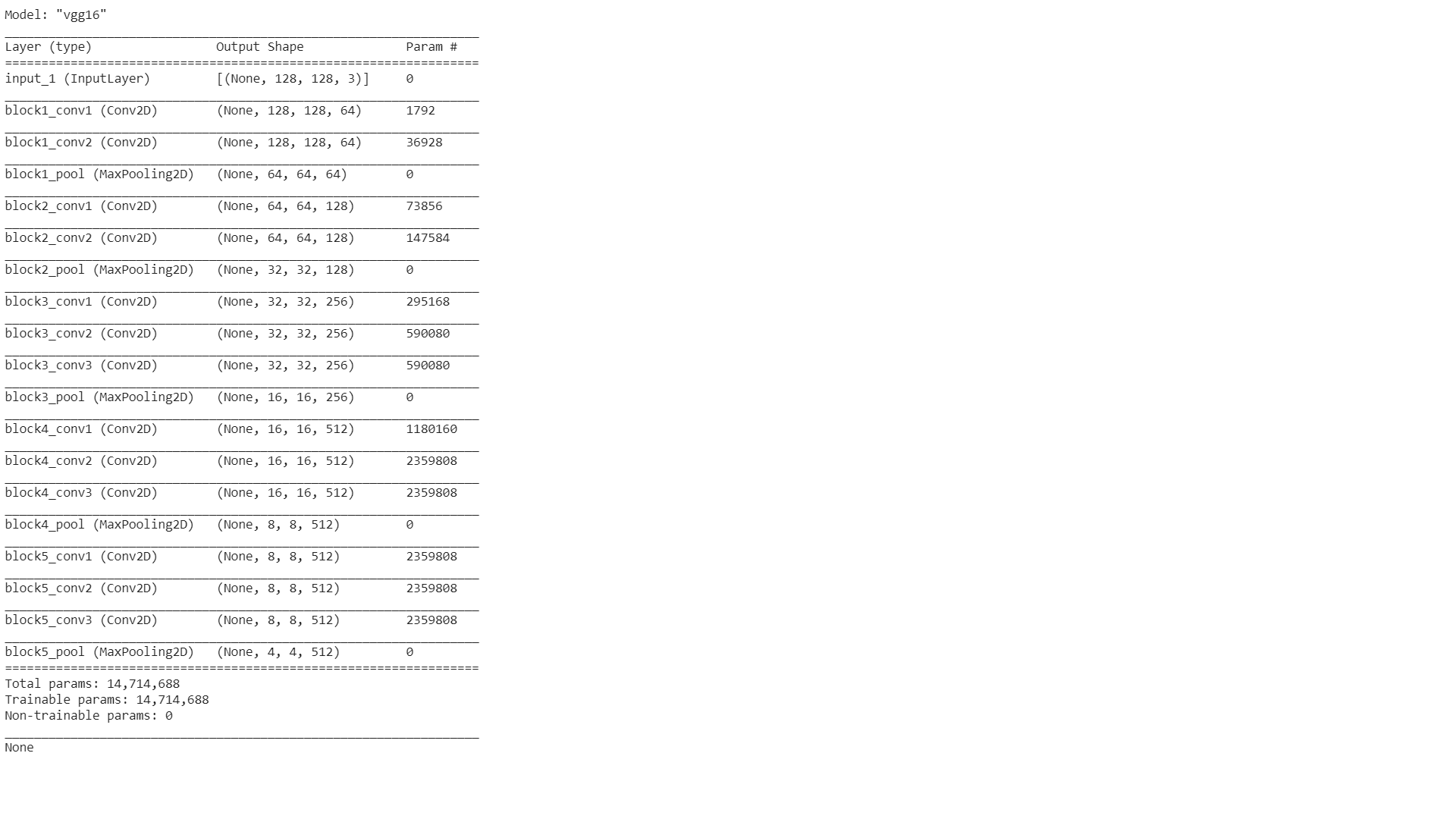
In this section, I have used two pre-trained models for feature extraction.

VGG16

InceptionV3

First, feature extraction has been done with selecting VGG16 model. This model is sliced i.e; fully connected layers are removed from the model.

Below is the summary of the truncated VGG16 model.



In the final layer of the truncated VGG network, the output shape is (None, 4, 4, 512). Training data is pushed to the network which is to be flattened and this data is sent to the secondary machine learning algorithm.

Selected secondary machine learning algorithms:

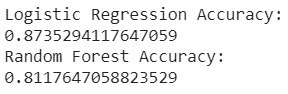
Logistic Regression

Random Forest Classifier

The output shape of the training data from the truncated VGG16 network is (1020, 4x4x512) = (1020, 8192). For feature extraction, testing data also has to be flattened by passing through VGG16 network. The flattened shape of the testing data is (340, 4x4x512) = (340, 8192).

This flattened training and testing data are passed through the selected machine learning algorithms for predictions and calculate the accuracy values of these machine learning algorithms.

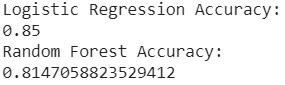
Below are the accuracy values of both the selected algorithms with VGG16.



Second, InceptionV3 model has been selected and fully connected layers are removed from the model. Training data is sent to the network for flattening the data and the output shape from the model is (1020, 2x2x2048) = (1020, 8192). Testing data also sent to the model to flatten the data and the shape of the flattened data is (340, 8192).

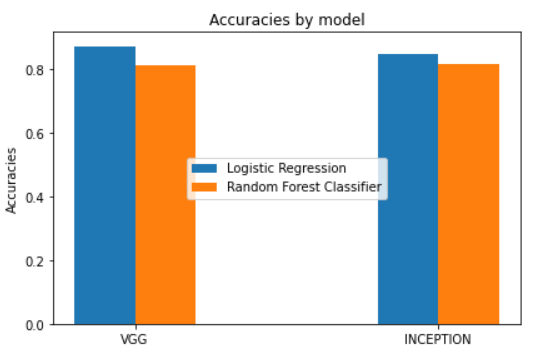
This flattened data (both training and testing data) is sent to the selected machine learning algorithms for predictions and calculated the accuracy values of these algorithms.

Below are the accuracy values of both the selected algorithms with InceptionV3.



In both of the selected CNN models, logistic Regression performed better than Random Forest Classifier.

Below graph shows the accuracy values for both CNN networks.



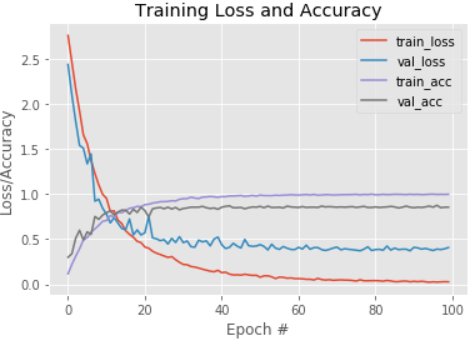
According to the above graph, VGG model performed better than InceptionV3. So, the best CNN model is VGG over InceptionV3.

With VGG16, logistic regression performed better than Random Forest Classifier. Therefore, the best pair is VGG16 with logistic Regression over VGG16 with Random Forest Classifier.

Part-II (Fine Tuning)

In this part, fine-tuning is done using truncated VGG pre-trained model used in the above section. All the weights in all layers are frozen and some fully connected layers are added to the network. Now, this model is compiled with SGD optimizer and sparse\_categorical\_crossentropy as the loss function. Next, the model is trained with 100 epochs. The final validation accuracy was 85.59%.

Below is the performance of the model throughout the training process.

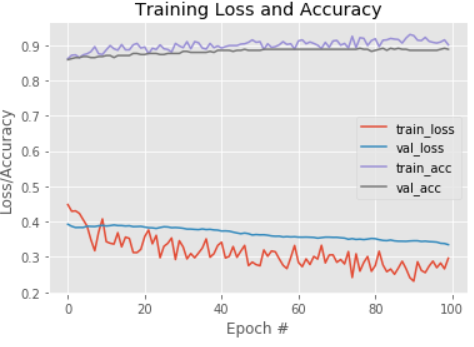


Observing the above graph, model performed well without any over fitting, but the final accuracy was 85.59%. To increase the accuracy, lets tune the model by unfreezing some weights.

Now, some weights in few convolutional layers are unfreeze and the model is trained with data augmentation technique to make sure there is no overfitting of the data.

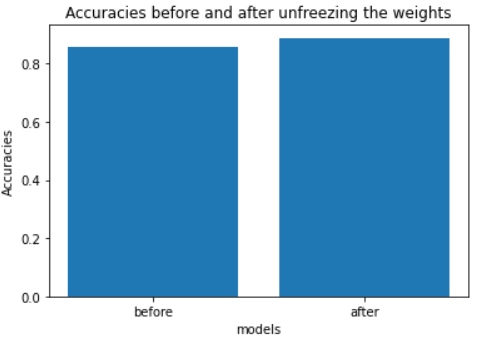
After doing data augmentation technique by unfreezing some weights, the model is trained and the final validation accuracy obtained was 88.82% where there is an improvement in the performance.

Below is the performance of the model with data augmentation technique.



In the above graph, the model is performed so well. There is no evidence of overfitting and the values of training loss and validation loss have improved a lot.

Let's compare the accuracy values of the model before and after unfreezing the weights in the convolutional layers.



**Part C**

**CapsNet – Capsule Neural Network**

**Convolutional Neural Network**

A ConvNet or CNN, Convolutional Neural Network is an Artificial Neural Network that helps the Deeplearning community at this point to address a lot of challenges in the world of Computer Vision, be it classifying an image or recognising objects in an image or you name it, ConvNet performs exceptionally well. It’s also made significant progress in the digitization of content of images and videos.

The ConvNet consists of Convolutional layers, Pooling layers and Fully connected Dense layers. Through these series of layers, an image gets broken down to pixels and various aspects or features of images are captured, compressed and processed. During the process of training, a number of parameters get adjusted to learning the ground truth or true label of an image.

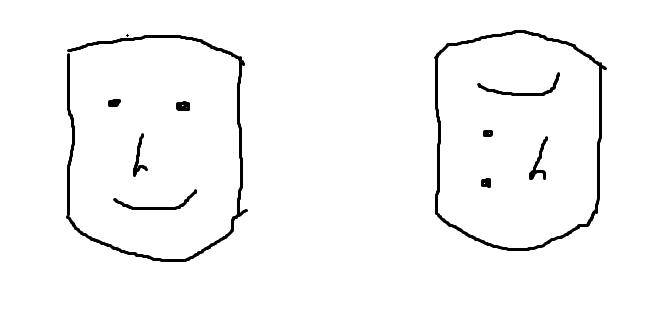
A picture containing text

Description automatically generated

**Challenges in ConvNet**

Ever since I learnt about ConvNet, there was a question that kept lingering in my mind. Doesn’t it really matter the way ConvNet process images? I’m happy that Geoffrey Hinton’s research on CapsNet answered that question. Yes, of course, ConvNet learns various aspects of an image. However, does ConvNet really know what part of an image has the particular aspect that it learnt?

**Spatial information**

One might wonder that why does it matter which part of an image has the learnt aspect. Well, a face is called as ‘face’ only when its aspects exist where they belong. Imagine a figure that has different parts of a face but not a face. What do you call it? Still a ‘face’? No, you call it ‘parts of a face’. If you input such an image to a face classifier model built on ConvNet, it’ll classify it as a ‘face’. It’s because of the fact that ConvNet disintegrates various features and learns them for prediction. Therefore, it’s important that which part of an image has the learnt aspect. 

**Data-hungry**

We know that neural networks are data-hungry. Especially, in order to build a foolproof image classifier model, one must supply many examples with a good number of variants in it. With the help of image augmentation, we have been able to achieve it, but still, the computational cost increases since we train with a large number of examples. This is yet another challenge we have with ConvNet.

**CapsNet overcomes challenges in ConvNet**

Pooling layers is remarkable in performing what they’re employed for in ConvNet. However, they treat the low-level details captured by Convolutional layers independently leading to a loss in spatial information of features.

Technically, as a result of this, equivariance nature of input is discarded, rather invariance is retained. In simple words, invariance implies that shuffling of features or objects in an image has no ill-effects. On the other hand, equivariance entails that shift or shuffling of features or objects in an image also leads to the relative shift in the image as well causing it to be a different one in terms of position.

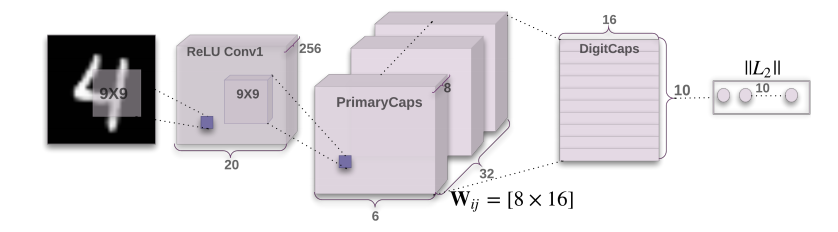
**Capsules**

CapsNet, Capsule Neural Network has taken a step to do away with Pooling layers and replace them with Capsules. A Capsule is a set of neurons that are responsible for capturing various properties of a feature in addition to spatial information, thus making it an activity vector or a pose vector **UI** as opposed to a traditional scalar output value of a neuron.

The presence of an entity is determined based on the length of the vector representing it, whilst the orientation signifies the properties that the capsule represents.

**Architecture of CapsNet**

A CapsNet consists of convolutional layers to capture low-level features that are followed by capsules at low-level also known as primary capsules and capsules at high-level. It employs dynamic routing by agreement technique for computation of outputs of capsules.



**Dynamic routing by agreement**

Capsules are at both lower layers and higher layers of a network. Capsules at lower levels are effectively be instantiated with a Gaussian and Uniform distribution. A set of capsules in the lower-level form a cluster to get routed to a capsule in the higher layer, thus outputting a higher probability for the presence of the entity and the pose.

Each capsule in the lower level computes the prediction vector **uj| I** that it belongs to a particular capsule in the higher layer. This prediction vector is computed by multiplying the input by a weight matrix **Wij**(Gaussian and Uniform distribution). The higher-level capsules compute output **sj** by a dot product of a lower-level predicted vector **uj|i** with a coefficient **cij** that represents a probability that the lower-level capsule belongs to that higher-level capsule.

The value of the coefficient gets increased as the lower-level capsule prediction gets more and more accurate in terms of predicting it belongs to a particular higher-level capsule. Throughout the learning process, the coefficients are adjusted using a softmax and a squashing function.

Prediction vector of a lower-level capsule



The output of a higher-level capsule



A Margin loss function is used for the existence determination of multiple entities as follows.



where,

Tk = 1 if an entity of class k is present, otherwise Tk=0

m+=0.9 and m-=0.1

lambda value can be 0.5 for down-weighting of the loss for the absence of a class

References:

<https://arxiv.org/pdf/1710.09829.pdf>

<https://techtv.mit.edu/collections/bcs/videos/30698-what-s-wrong-with-convolutional-nets>

<https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b>