Final Project (COMP-5411)

Aloukik Aditya (1115290)

Sarthak Rawat (1101124)

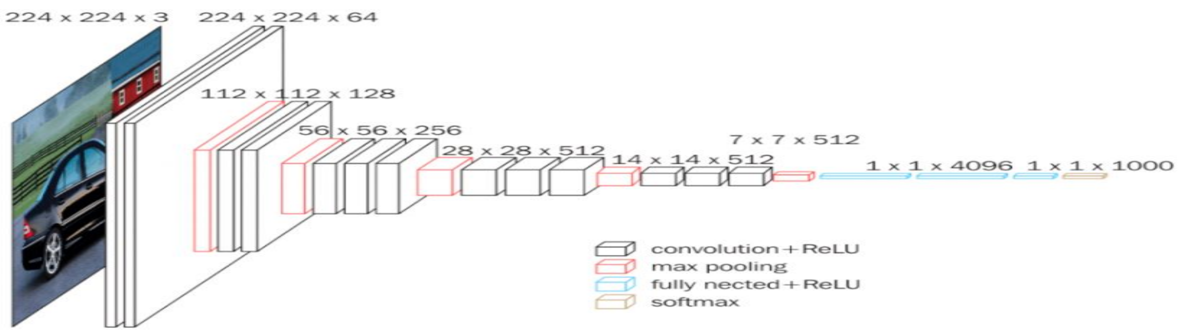
Task

Find top-1 accuracy on the following dataset CIFAR-100, CIFAR-10, Caltech256, Caltech101.

Downloading and Splitting and datasets:

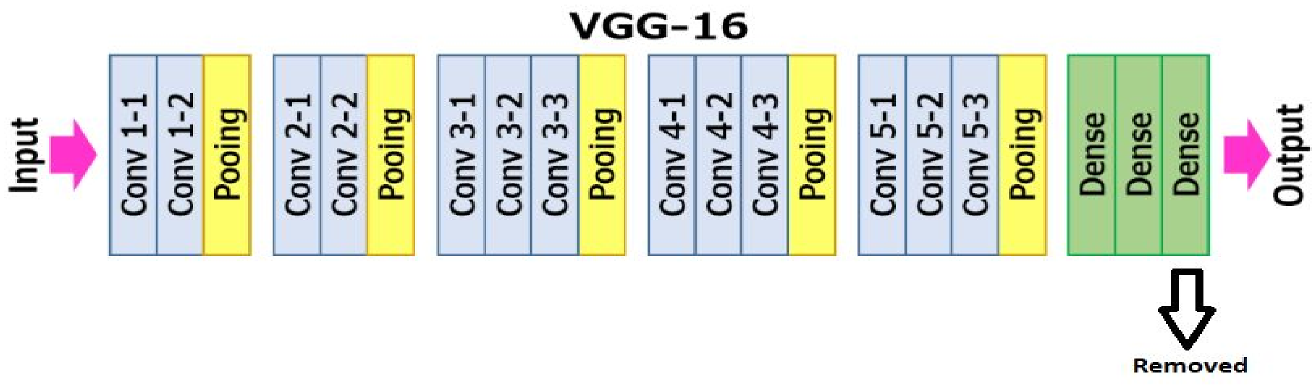
- Caltech256 : http://www.vision.caltech.edu/Image_Datasets/Caltech256/
- Caltech101 : http://www.vision.caltech.edu/Image_Datasets/Caltech101/
- CIFAR-100 :
  - Data set can be downloaded directly using python
  - *from tensorflow.keras.datasets import cifar100*
  - *(train_imgs, train_labels), (test_imgs, test_labels) = cifar100.load_data()*
- CIFAR-10 :
  - Data set can be downloaded directly using python
  - *from tensorflow.keras.datasets import cifar10*
  - *(train_imgs, train_labels), (test_imgs, test_labels) = cifar10.load_data()*

VGG16 Model

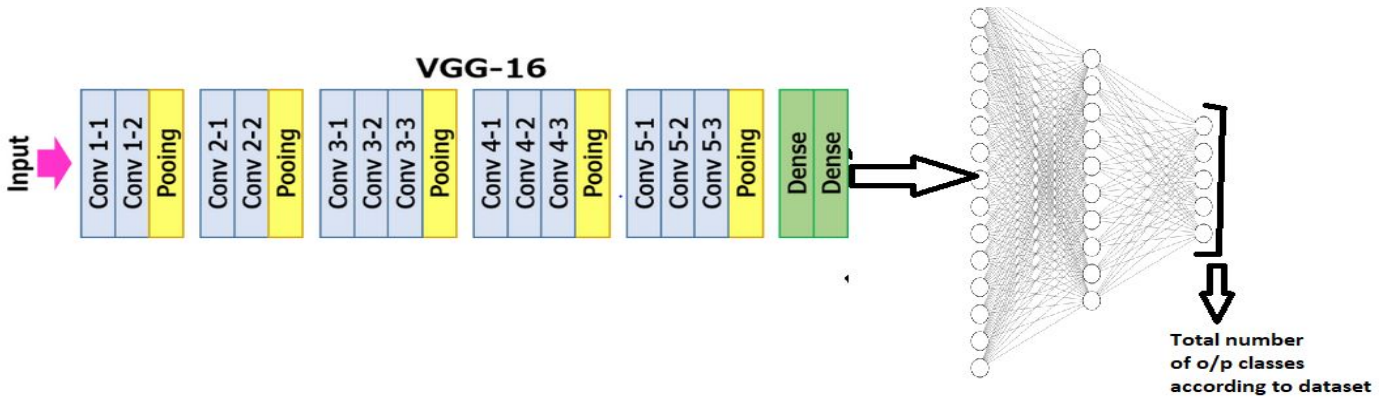


Our Model:

We have used the VGG16 model in our Project. The first top fully connected layer is removed and then new layers according to the number of categories of the dataset are added.



New Model:



- We keep two fully connected layers of VGG16
- And change the number of neurons in our FC layer according to Dataset.

Structure of our FullyConnected layer for different datasets:

- **Caltech-256:** We have used the following structure
  - 900 Dense neurons (with L1 regularizer)
  - Dropout with 0.5 rate
  - 257 Dense (number of classes)

- **Caltech-101**: We have used the following structure
  - 600 Dense neurons (with L1 regularizer)
  - Dropout with 0.5 rate
  - 102 Dense (number of classes)

- **CIFAR-100**: We have used the following structure
  - 1000 Dense neurons (with L1 regularizer)
  - Dropout with 0.3 rate
  - 100 Dense (number of classes)

- **CIFAR-10**: We have used the following structure
  - 800 Dense neurons (with L1 regularizer)
  - Dropout with 0.3 rate
  - 10 Dense (number of classes)

Preprocessing:

For preprocessing and normalization of images, we have used the inbuilt function of VGG16 from Keras.

Eg:

```
from tensorflow.keras.applications.vgg16 import preprocess_input

train_imgs = preprocess_input(train_imgs)
```

Both the test and train images were normalized using this function.

Resize images for cifar-10,100:

The images in cifar-10/100 are resized to (224,244,3) format so that they can be used with the VGG16 model. The code takes up a lot of RAM. (Google Cloud VM with 60gb ram was used so as to resize without any issues).

Data/Image augmentation:

We have used <u>one</u> Image augmentation technique to increase the number of training samples in Caltech256 and Caltech101

We have just used an image flip(mirror image) for augmentation:

Original	Fipped
0 20 40 60 80 100 120 140 160 0 50 100 150 200 250	0 20 40 60 80 100 120 140 160 0 50 100 150 200 250

How we have trained our model:

- Step1: We freeze our layers of VGG16 so that they are not trained initially.
- Step2: We then train only our fully connected layer for 2 epochs (so that it starts to extract meanings from VGG16 network)
- Step3: We then unfreeze VGG16 layers and then train our model with a low variable learning rate for a higher number of epochs.

Hyperparameters used:

- For the initial 2 epochs, we used Adam optimizer with LR=0.001.
- For the training full model, we used a variable learning rate. (Example : epoch 0-3 : LR=1e-4 , epoch 4-8 : LR =1e-5 , epoch 8-10 : LR = 1e-6)
- For Cifar datasets we used a dropout rate of 0.3
- For Caltech dataset, we used a dropout rate of 0.5

Experimental results

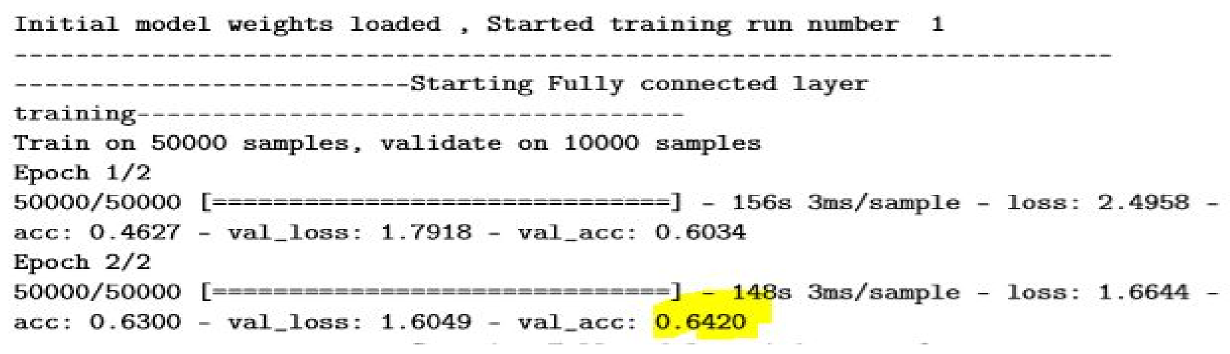
We ran every model 3 times and took the average accuracy of the three runs:

Results for the different dataset are as follows:

- **CIFAR-100**
- **CIFAR-10**
- **Caltech-256**
- **Caltech-101**

CIFAR-100

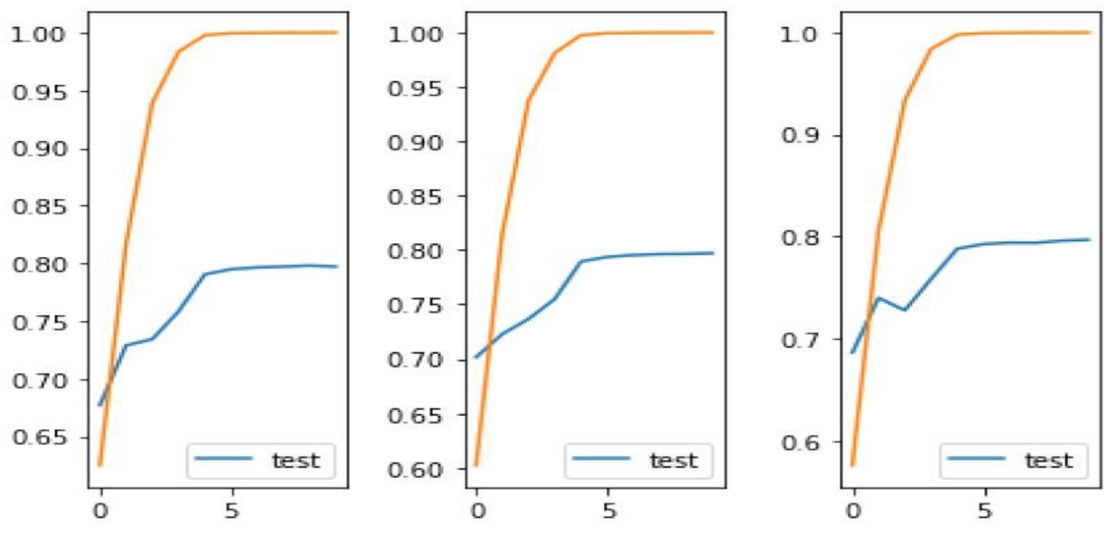
Accuracy after first two epochs (training of only FC layer):

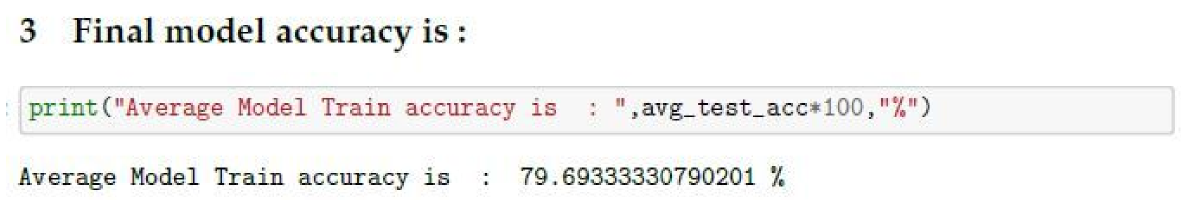


```
Initial model weights loaded , Started training run number  1
------------------------------------------------------------------------
--------------------------Starting Fully connected layer
training------------------------------------
Train on 50000 samples, validate on 10000 samples
Epoch 1/2
50000/50000 [==============================] - 156s 3ms/sample - loss: 2.4958 -
acc: 0.4627 - val_loss: 1.7918 - val_acc: 0.6034
Epoch 2/2
50000/50000 [==============================] - 148s 3ms/sample - loss: 1.6644 -
acc: 0.6300 - val_loss: 1.6049 - val_acc: 0.6420
```

After this, we unfreeze all our layers and Further train our model starting from these weights.

Training full model (All layers are trainable):





3 Final model accuracy is :

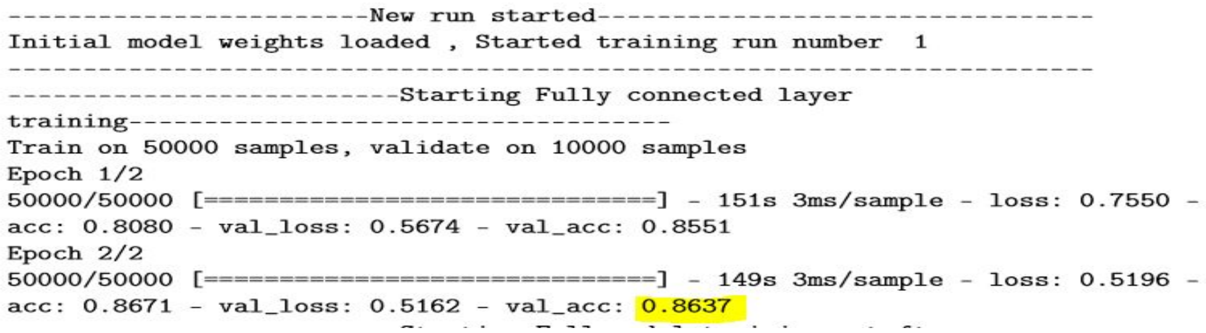
```
print("Average Model Train accuracy is  : ",avg_test_acc*100,"%")
```

```
Average Model Train accuracy is  :  79.69333330790201 %
```

Average accuracy for 3 runs: 79.69%

CIFAR-10

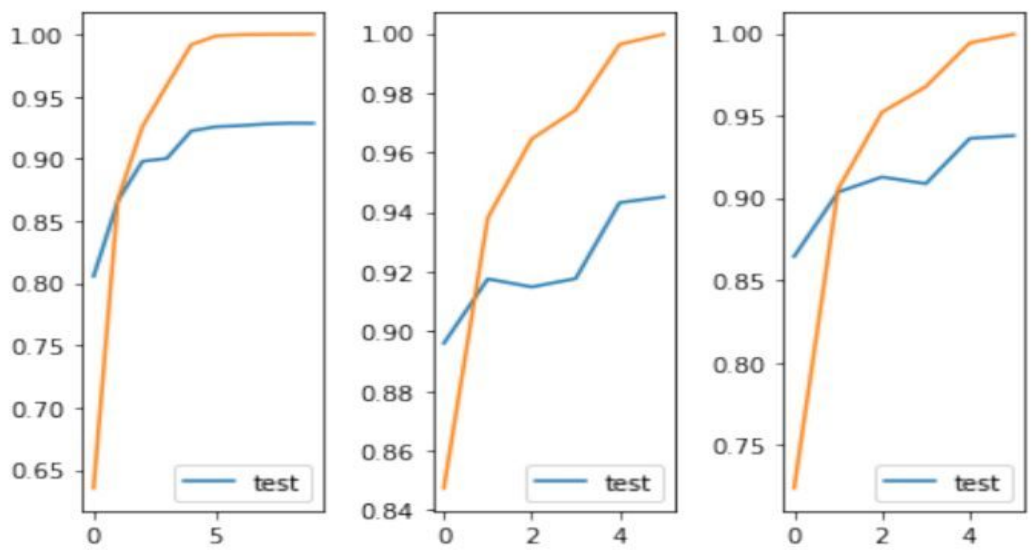
Accuracy after first two epochs (training of only FC layer):

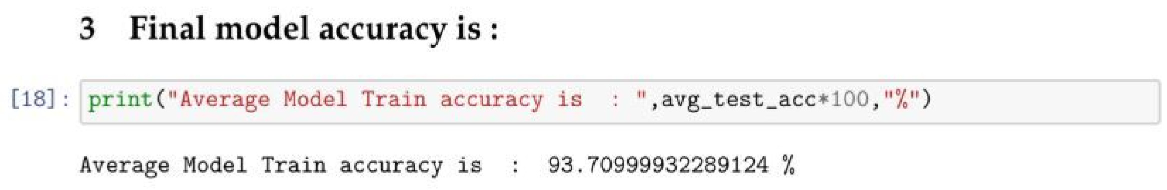


```
-------------------------New run started---------------------------------
Initial model weights loaded , Started training run number  1
-------------------------------------------------------------------------
---------------------------Starting Fully connected layer
training-------------------------------------------
Train on 50000 samples, validate on 10000 samples
Epoch 1/2
50000/50000 [==============================] - 151s 3ms/sample - loss: 0.7550 -
acc: 0.8080 - val_loss: 0.5674 - val_acc: 0.8551
Epoch 2/2
50000/50000 [==============================] - 149s 3ms/sample - loss: 0.5196 -
acc: 0.8671 - val_loss: 0.5162 - val_acc: 0.8637
```

After this, we unfreeze all our layers and Further train our model starting from these weights.

Training full model (All layers are trainable):





3 Final model accuracy is :

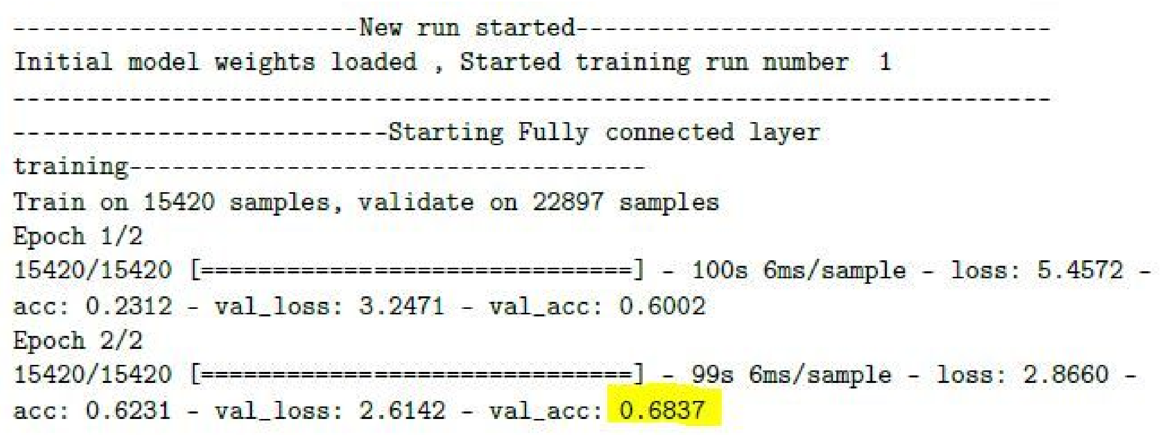
```
[18]: print("Average Model Train accuracy is  : ",avg_test_acc*100,"%")

Average Model Train accuracy is  :  93.70999932289124 %
```

Average accuracy for 3 runs: 93.70%

Caltech-256

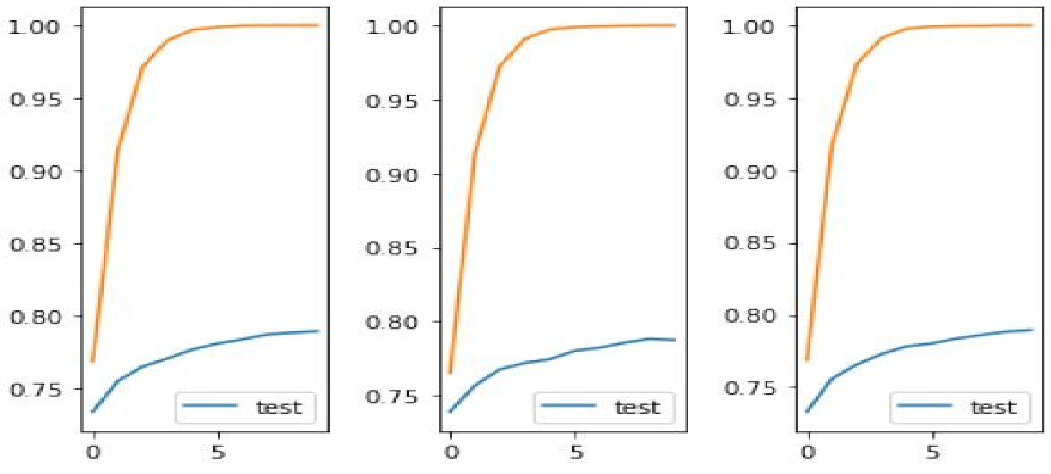
Accuracy after first two epochs (training of only FC layer):

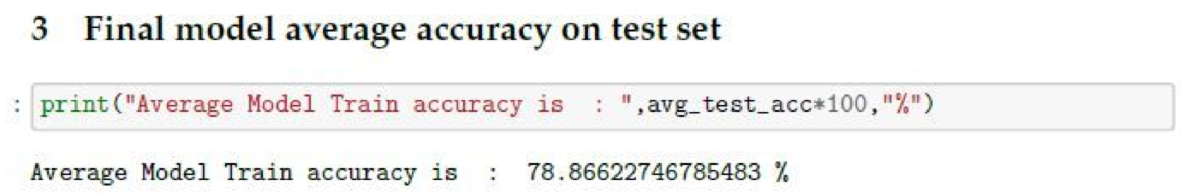


```
-------------------------New run started---------------------------------
Initial model weights loaded , Started training run number  1
--------------------------------------------------------------------------
---------------------------Starting Fully connected layer
training-------------------------------------
Train on 15420 samples, validate on 22897 samples
Epoch 1/2
15420/15420 [==============================] - 100s 6ms/sample - loss: 5.4572 -
acc: 0.2312 - val_loss: 3.2471 - val_acc: 0.6002
Epoch 2/2
15420/15420 [==============================] - 99s 6ms/sample - loss: 2.8660 -
acc: 0.6231 - val_loss: 2.6142 - val_acc: 0.6837
```

After this, we unfreeze all our layers and Further train our model starting from these weights.

Training full model (All layers are trainable):





3 Final model average accuracy on test set

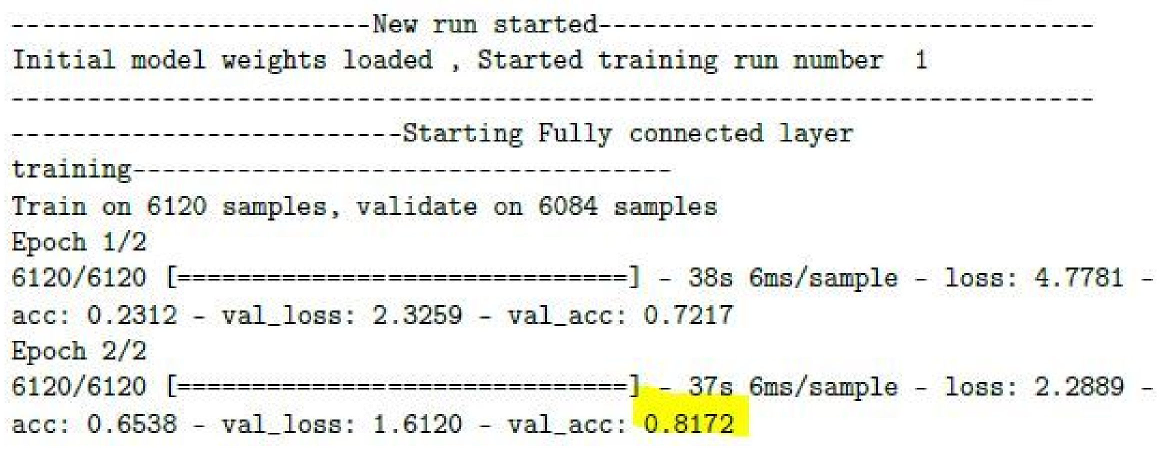
```
: print("Average Model Train accuracy is  : ",avg_test_acc*100,"%")

Average Model Train accuracy is  :  78.86622746785483 %
```

Average accuracy for 3 runs: 78.86%

Caltech-101

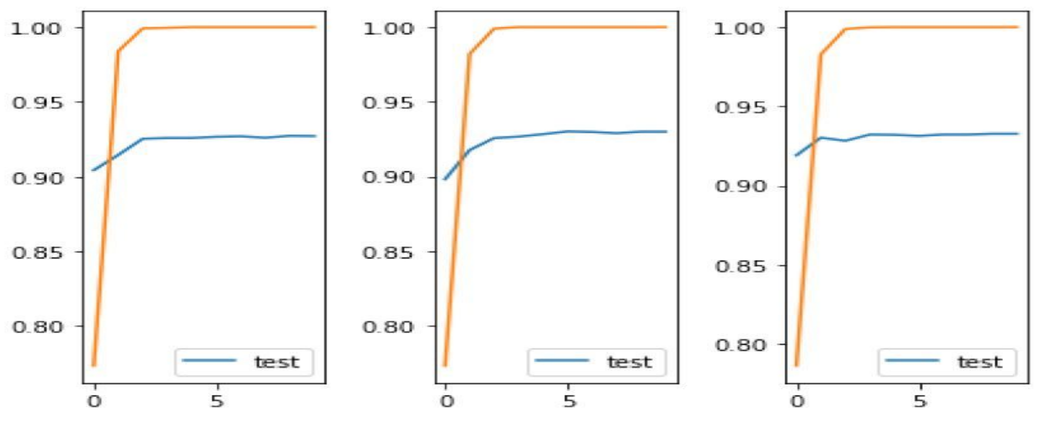
Accuracy after first two epochs (training of only FC layer):

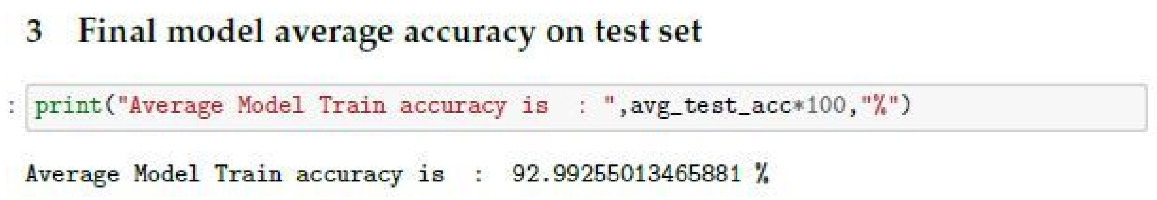


```
-------------------------New run started---------------------------------
Initial model weights loaded , Started training run number  1
-------------------------------------------------------------------------
---------------------------Starting Fully connected layer
training-----------------------------------
Train on 6120 samples, validate on 6084 samples
Epoch 1/2
6120/6120 [==============================] - 38s 6ms/sample - loss: 4.7781 -
acc: 0.2312 - val_loss: 2.3259 - val_acc: 0.7217
Epoch 2/2
6120/6120 [==============================] - 37s 6ms/sample - loss: 2.2889 -
acc: 0.6538 - val_loss: 1.6120 - val_acc: 0.8172
```

After this, we unfreeze all our layers and Further train our model starting from these weights.

Training full model (All layers are trainable):





3 Final model average accuracy on test set

```
: print("Average Model Train accuracy is  : ",avg_test_acc*100,"%")
```

```
Average Model Train accuracy is  :  92.99255013465881 %
```

Average accuracy for 3 runs: 92.99%

Final Result

OVERALL our final top-1 average accuracy for all the data sets is: ***86.31 percent***

- *CALTECH-256 : 78.86%*
- *CALTECH-101 : 92.99%*
- *CIFAR-100 : 79.69%*
- *CIFAR-10 : 93.70%*

Technical Specifications (Google cloud VM) :

- 16 core CPU
- 60 GB ram
- Nvidia P-100 GPU (16gb)

CONCLUSION

We have trained our models on the following dataset: cifar-10,100, caltech-256,101. Where we have used Transfer learning using the VGG16 model. Transfer learning has great potential. This is because they were already trained to extract important features of images from different categories from a much larger dataset(ImageNet). Freezing our model for two epochs (initially) helped us reach a better accuracy faster. This might be because our fully connected layer started learning to mean from the VGG16 model. Which further boosted our accuracy in further training of the model.