Pattern Recognition and Machine Learning (EE552) Project 3

Savinay Nagendra (sxn265)

March 4, 2018

Abstract

In this report, Convolutional Neural Networks have been explored on the problem of Image Classification. Different CNN architectures have been applied on the Wallpaper group data set which has 17 classes and have been compared with each other quantitatively, based on the performance, efficiency and accuracies. Preprocessing techniques such as Image Augmentation and feature dimensionality reduction using t-SNE algorithm have been applied to observe the change in performance of these algorithms. Layer wise visualization of convolution filters have been made to observe the features each layer is learning.

Contents

1 Introduction

Convolutional neural network (CNN) is a class of deep, feed-forward artificial neural networks which is explicitly applied to analyze visual imagery. But, they also have applications in video recognition, recommender systems and natural language processing. The CNN organizational architecture is similar to the visual cortex of animals. CNNs use a variation of multilayer preceptrons designed to require minimal pre-processing. Infact, CNNs are unique in terms that the amount of pre-processing required is very less when compared to other image classification algorithms. So, instead of hand-engineering filters, the networks learn the filters used for feature extraction.

A CNN consists of an input layer, an output layer as well as hidden layers. The hidden layers of a CNN typically consist of **convolutional layers, activation layers, pooling layers, fully connected layers** and **normalization layers**. Unlike regular neural networks, CNNs have neurons arranged in 3 dimensions to account for different channels of images. In a nut shell, a conv net is a list of layers that transform the image volume into an output volume.

Each hidden layer of a CNN will not have parameters or weights to be computed. The parameters to be learnt belong to Convolution layers and Fully connected layers. But, every layer will have hyper-parameters that have to be defined by the end-user. The assignment of hyper-parameters are problem specific. The weights are learnt by the network using the method of back propagation, such that the loss function will tend to minimize.

A CNN usually requires a large amount of data for training when compared to other methods of classification. The reason for this is that there is a lot of exploration involved during training. Since the available image data is usually lesser than the required amount, new image data is generated by the process of **Augmentation**. Data is sent in pre-defined batches, based on which weights of all the layers are updated. This is to ensure computational efficiency.

Further, a bigger network architecture has to be used to increase accuracy when there is a large amount of data. Pre-processing steps such as Feature dimensionality reduction can be helpful for achieving better accuracy on larger data sets. Also, in practical applications with smaller data sets, a pre-trained big network can be used to extract features, whcih can then be given to a classifier. This method is referred to as **Transfer Learning**.

The report is organized as follows: Section 2 deals with the explanation of architectures used, data augmentation, transfer learning using Alexnet and the projection algorithm t-SNE. Section 3 portrays the results for all the networks. Section 4 gives the conclusions.

2 Approach

2.1 Data

Wallpaper Group Data set:
This dataset consists of the features extracted from images containing 17 Wallpaper Groups. Each image in the data set is a gray scale image of size 256×256.

number of Classes: 17
number of Training Observations: 17000
number of Test Observations: 17000 The classes in the dataset are:

P1
P2
PM
PG
CM
PMM
PMG
PGG
CMM
P4
P4M
P4G
P3
P3M1
P31M
P6
P6M

2.2 Methods

In this report, the following methods have been covered:

1. Implementing a simple Convoutional Neural Network Architecture on the original data set to observe train, test and validation accuracies.
2. Augmenting data by the methods of Scaling, Rotation and Translation to generate a bigger data set.
3. Training the small CNN on the Augmented data set to observe the accuraciesa and performance.
4. Constructing deep neural network architectures to train the network on Augmented dataset - Skinny and Wide Networks.
5. Training the Deeper Neural Network configurations on the original data set to observe accuracies and performance.

6. Visualizing each layer of Convolution Filter Activations and Fully Connected layer activations.

7. Performing t-SNE on Fully connected layers of all Skinny and Wide networks to observe well clustered class distributions.

8. Transfer learning on Augmented data set using Alex Net.

2.2.1 Convolution Neural Network Architecture

The following image portrays the CNN architecture:
A typical CNN will have the following layers:

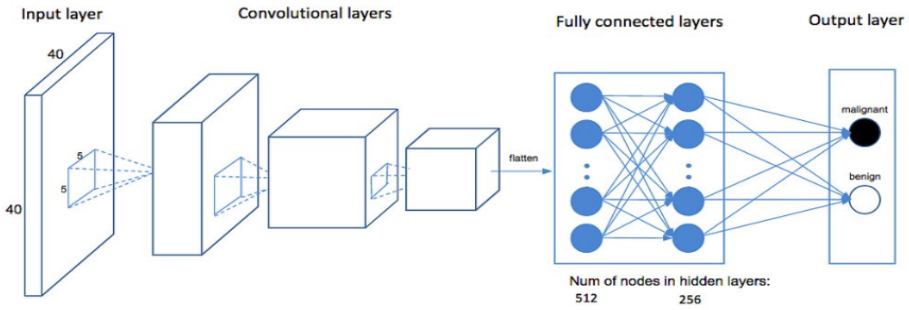


Figure 1: A general CNN architecture

1. **Input Layer**:

   This layer takes the raw pixel values of the image. The dimensions of input image is [**W** X **H** X **C**], where
   W: Width of image
   H: Height of image
   C: Number of channels (3 for color image - RGB)

2. **Convolutional Layer**:

   CONV layer will compute the output of neurons that are connected to local regions in inputs, each computing a dot product between their weights and a small region they are connected in the input volume. The filters are usually represented as a 3D matrix of dimensions [**w** X **h** X **K**], where
   w: width of filter
   h: height of filter
   K: Number of filters (depth of volume)
   Usually, the filter size will be square with w = h = F, being the size of the filter.
   In very deep network architectures, in order to reduce the image dimension reduction rate as it transverses from input to output, a zero padding is done on the image at each layer according to the filter size. Usually, the padding factor is given by

$P = (F-1)/2$, where F is the filter size. Convolution filters are applied using the sliding window technique. The output dimensions of the conv layer is given by: $(W-F+2P)/S+1$, where S is the stride, which is a hyper parameter of the sliding window.
Number of weights = $w \times h \times K$ Number of bias units = K

3. **ReLU Layer**:
   ReLU layer will apply an element wise activation function on the data, such as **max(0,x)**, thresholding at zero. This leaves the size of the volume unchanged.

4. **Pool Layer**:

   POOL layer will perform a downsampling operation along the spatial dimensions (width and height), resulting in a reduced volume. Stride is a hyper parameter of the POOL layer. The output of a MAX POOL layer will be $(W-F)/S+1$, where F is the factor of downsampling. Max Pool layers have been used in the project, with a downsampling factor of 2 and Stride 2.

5. **Fully Connected Layer**

   The final FC will compute the class scores, resulting in single dimensional vector, with each element corresponding to a class. So, the FC will have the neurons equal to the number of classes. However, there might be intermediate fully connected layers with any number of neurons. The hyperparameter for the FC layer is the number of output neurons.
   Input size: $(W \times H \times Depth) \times BatchSize$
   Output size: Number of neurons $\times BatchSize$
   Number of weights = $(W \times H \times Depth) \times$ Number of neurons Number of bias units = Number of neurons.

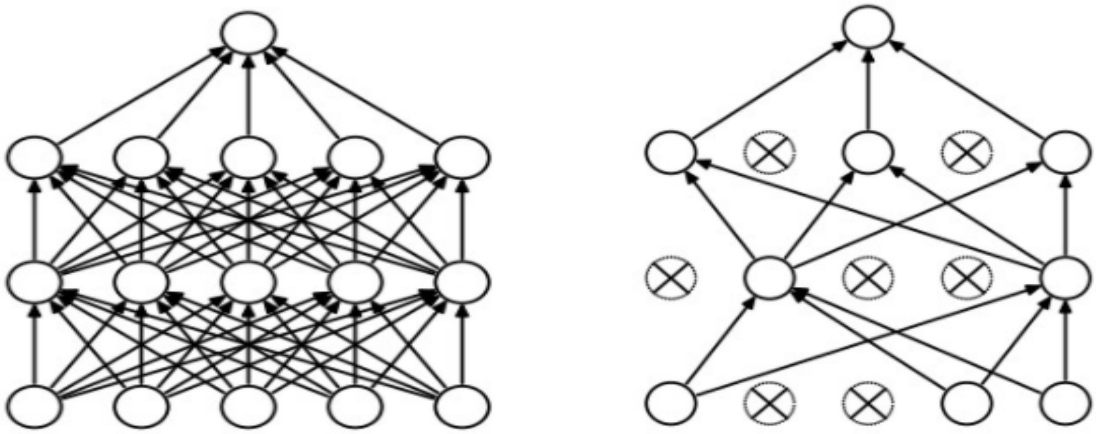
6. **Dropout Layer**
   Dropout is a technique used to improve over-fit on neural networks. It is often used along with other techniqes like L2 Regularization. During training ,half the neurons on a particular layer are deactivated. This will lead to generalization. During prediction, the dropout layer is deactivated. A probability value is the hyperparameter of the layer. This will specify the probability of a particular neuron being deactivated. Dropout layers are usuallyb used after the first fully connected layers. The random dropout of neurons happen during the processing of each batch. The figure below shows an example of a dropout layer.

7. **Softmax Layer**

   Activation function to convert the scores to probabilities.

   $Softmax(x_i) = \frac{exp(x_i)}{\sum_i exp(x_i)}$

In this report, different CNN architectures have been used on datasets of different sizes and their accuracies have been compared.



(a) Standard Neural Net (b) After applying dropout

Figure 2: Drop out Layer

CNN architectures can be classified based on the number of filters in Convolutional layers. If the numeber of layers in a network is more and number of filters in each layers is less, the network is called as **Skinny**. If the number of layers are less and each layer has comparatively more number of filters, the network is called as **Wide**. Both of these networks have been implemented in the project.

2.2.2 Augmentation

It has been established by research that deep CNNs, i.e., CNNs with more number of hidden layers, give better test and train acccuracies when compared to smaller networks. But, there are two issues with training deeper networks:

1. Very deep networks require a large amount of data for training. However, in applications where the data is visual imagery, the available data is less when compared to the demand.

2. Labelling of data is computationally expensive.

Hence, to avoid these two issues, labelled data is generated by the process of **Augmentation**. Data augmentation is a process where labelled data is generated by artificially inflating the training set with label preserving transformations.
The different data augmentation procedures are:

1. Scaling

2. Rotation

3. Translation

4. Contrast Stretching

5. Histogram Equalization

6. Adaptive histogram equalization

7. Contrast limited adaptive histogram equalization

Image augmentation is also used to avoid over-fitting of data. This is because the neural network is now exposed to transformed instances of the same image, which will aid in the network not memorizing the features. Hence, the classification can remain unbiased. In this project, **Scaling**, **Rotation** and **Translation** have been used to augment each sample of the training and test dataset. Subsection of results on Augmentation will portray the differences between Augmented and original data sets.

2.2.3 Transfer Learning

Transfer Learning is a method in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. Convolutional Neural Networks are generally used to extract features from images, which can later be used on a machine learning algorithm. Neurons in different convolution layers get exited by different features. The initial layers extract low level features such as blobs of colors, the middle layers extract features such as lines and edges, where as the final layers extract higher level features, which are a combination of the lower level features.

Recent advances in deep learning have resulted in the construction of very deep architectures of neural networks which are capable of handling very large data sets. These networks have been trained on classification and object detection problems with data sets with millions of training samples. Some examples of very deep convolutional neural network architectures are:

1. AlexNet

2. R-CNN

3. VGG Net

4. Google Net (Inception)

5. Fully Convolutional Networks

6. ResNet

These trained network models are available in most machine learning libraries. Now given an image, the feature extraction by convolution layers work in a similar way in any trained CNN architecture. The above mentioned deep CNNs, which have already been trained on millions of images would already have weights of the CONV layers oriented in such a way as to extract the required features. Hence, instead of training new networks built specific to a dataset, these deep networks can be used to train on our smaller data sets by training only a few layers instead of the entire network. This is the crux of transfer learning.

Transfer Learning can be achieved in two ways:

1. Extracting the features of the smaller data sets by running one or two epochs on our data set and then passing these features through a classification algorithm such as SVM or Fully Connected Neural Networks.

2. Chopping the last few layers from the big network and adding a few new layers according to the given problem and running the network through the data set by increasing the weight learning rates of the new layers.

Transfer learning is computationally effective and saves a lot of training time. It also aids in obtaining higher accuracy on smaller data sets. Choosing the differential learning rates and restoring the number of layers of trained network are some of the hyper parameters of transfer learning, which are problem specific.

2.3 Alex Net

Alex net has been used for transfer learning in this project. The architecture of Alex net is shown below:

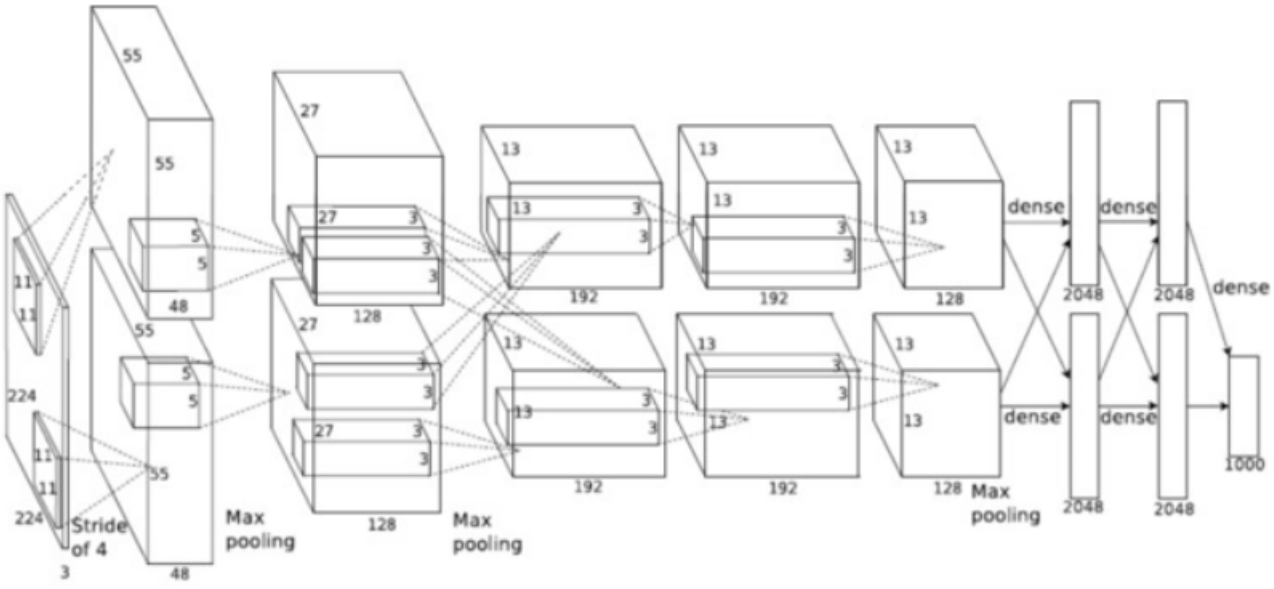


Figure 3: Alex Net architecture

The last fully connected layer, 'FC 8' has been chopped off and replaced by a new fully connected layer and the whole network os trained for 10 epochs. The learning rates of weights and bias for the new fully connected layer has been made higher, so that these are learnt faster than the CONV filter weights. This will accelerate the learning process and improve the efficiency. Results pertaining to alex net have been portrayed in the Transfer Learning section of results.

2.3.1 t-SNE

t-Distributed Stochastic Neighbor Embedding is a technique for dimensionality reduction that is particularly well suited for high-dimensional datasets. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points.

The t-SNE algorithm comprises of 2 stages: First, it constructs a probability distribution over pairs of high dimensional objects in such a way that similar objects have a high probability of being picked, while dissimilar objects have a very less probability of getting picked. Second, it defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the KullbackLeibler divergence between the two distributions with respect to the locations of the points in the map.
Originally, the algorithm was developed based on Euclidean distance between objects, but other distances can also be used.

3 Results

Results of different network architectures have been portrayed in this section

3.1 Image Augmentation

Original Dataset:
The original data set has 17000 images in the training set and 17000 images in the test set, each of size $\mathbf{256 \times 256}$. To improve the efficiency of training, to implement bigger CNNs and to avoid over-fitting of data, a much larger data set is necessary. This data set is generated by the process of Augmentation. Each image is Scaled, Rotated, Translated by a random number generator 5 times. Hence, the training set now has 85000 images. Finally, all the images are cropped to $\mathbf{128 \times 128}$, which are used for training the neural network.
Histograms of Rotation, Scaling and Translation have been portrayed below, for the 85000 images in the Augmented dataset. It can be observed that the Histogram of rotation is close to a **Uniform distribution**, histogram of scaling is a **skewed Gaussian** distribution and the Histogram of Translation is also a **skewed Bivariate Guassian** distribution. 5 sample images have been randomly chosen from the original training data set and ugmentations have bee applied separately to observe the variations. These sets of images have been portrayed below:

Samples of Augmented images are shown below:

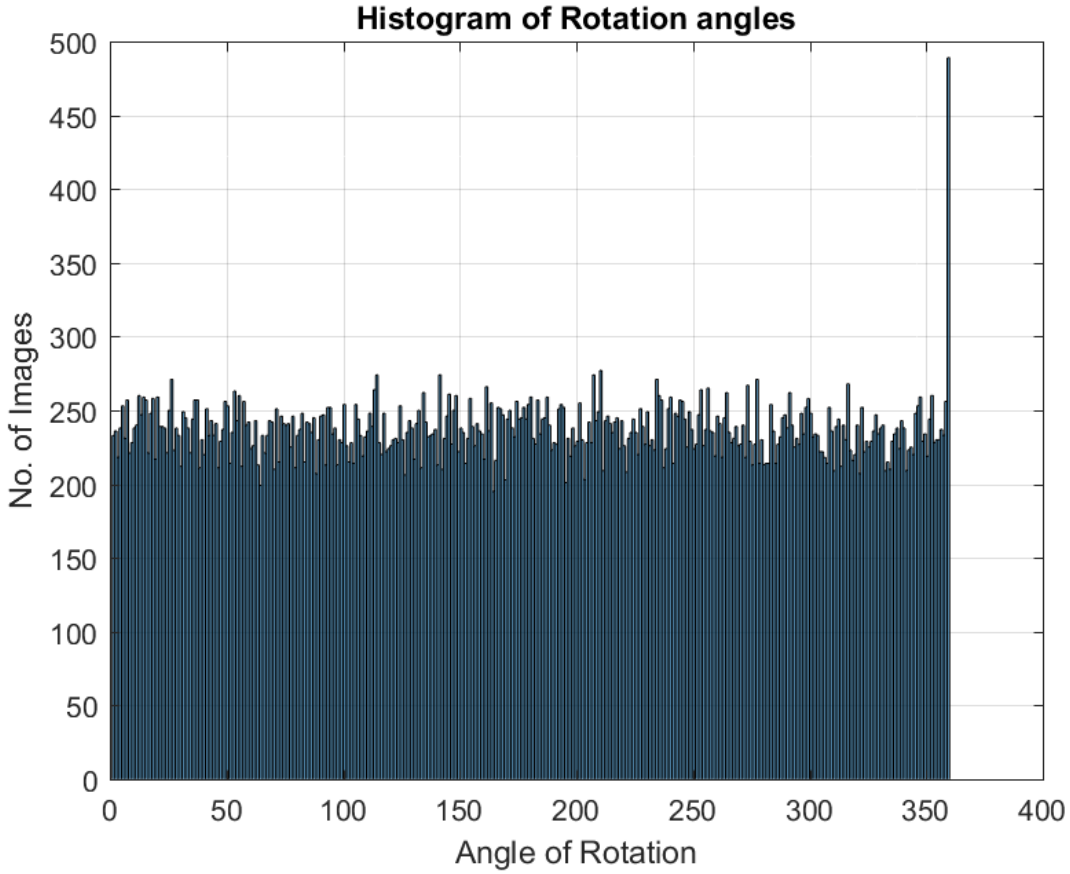


Figure 4: Histogram of Rotation

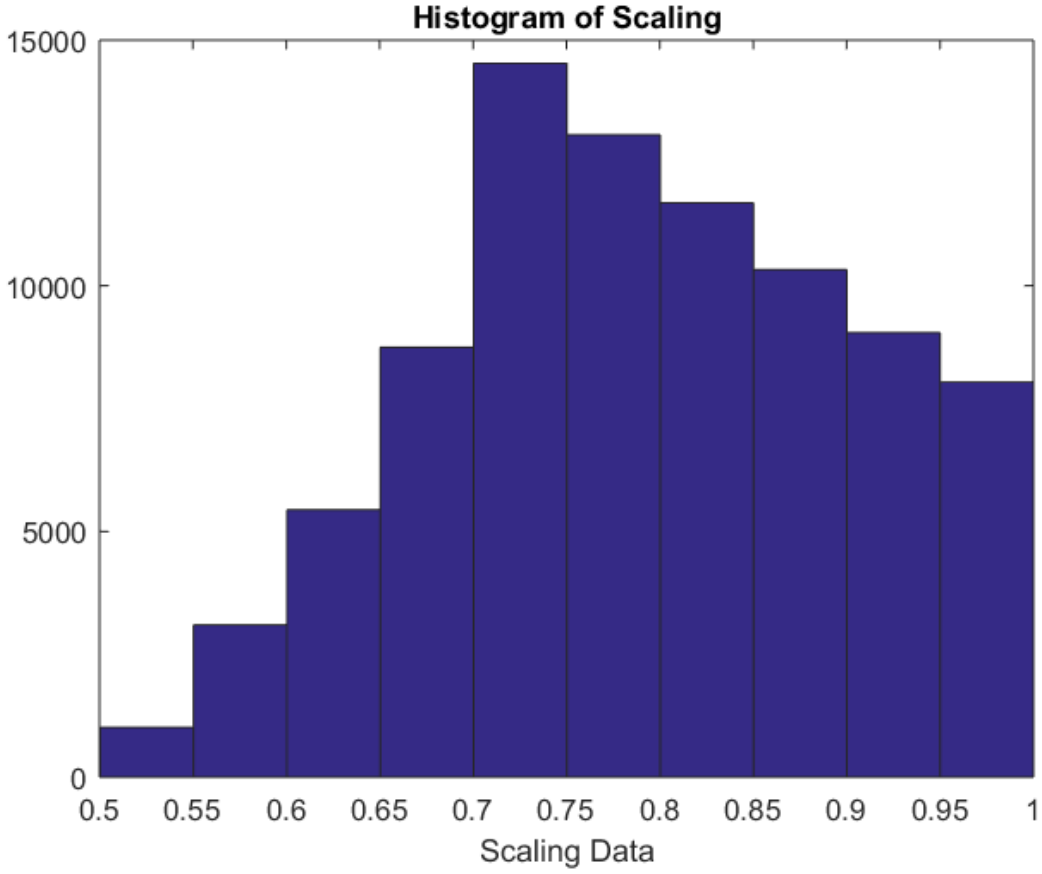


Figure 5: Histogram of Scaling

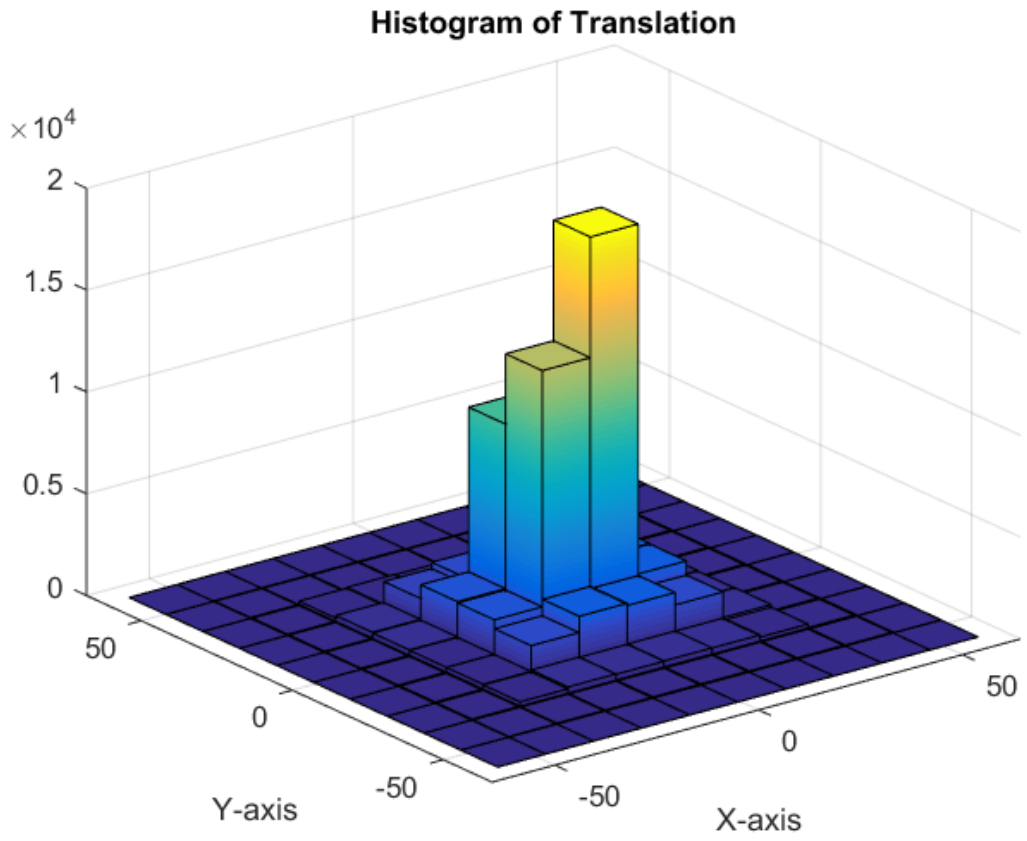
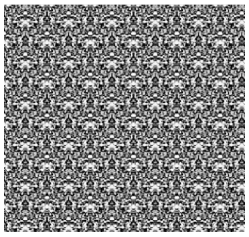
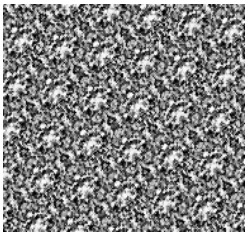


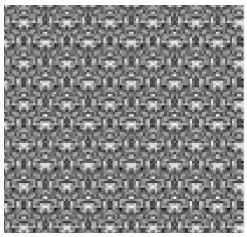
Figure 6: Histogram of Translation



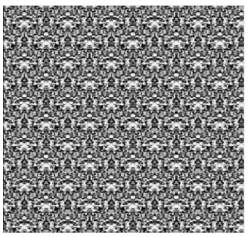
(a) Image1 - Original



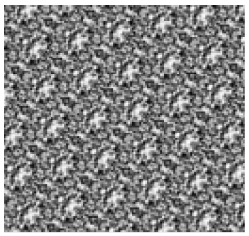
(b) Rotated by 36 degrees



(c) Scaled by a factor of 0.4

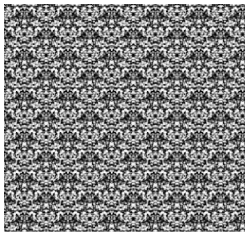


(d) Translated image

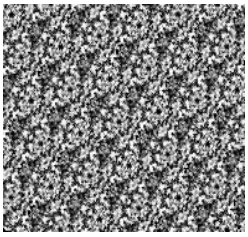


(e) Combined Augmentations on image 1

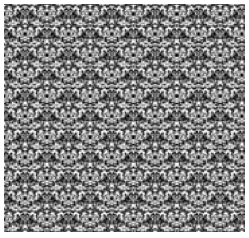
Figure 7: Augmentations for image 1



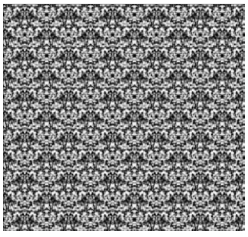
(a) Image14 - Original



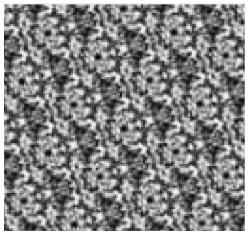
(b) Rotated by 59 degrees



(c) Scaled by a factor of 0.7

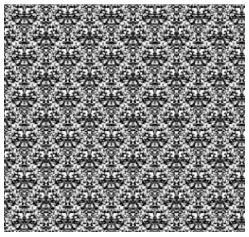


(d) Translated image

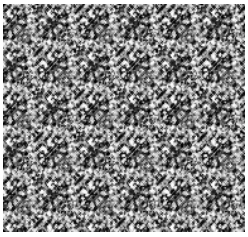


(e) Combined Augmentations on image 14

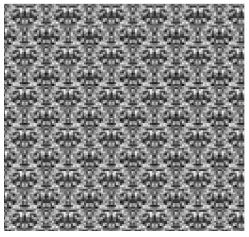
Figure 8: Augmentations for image 14



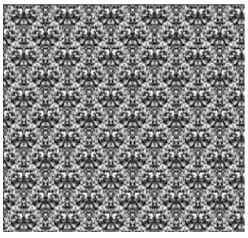
(a) Image134 - Original



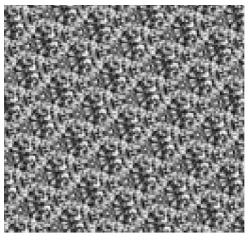
(b) Rotated by 45 degrees



(c) Scaled by a factor of 0.5

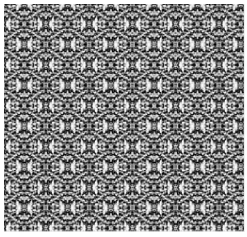


(d) Translated image

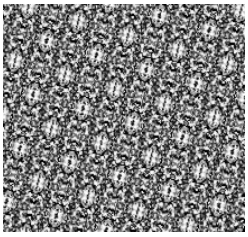


(e) Combined Augmentations on image 134

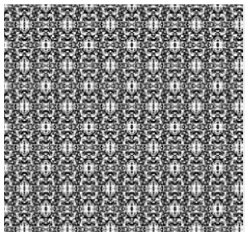
Figure 9: Augmentations for image 134



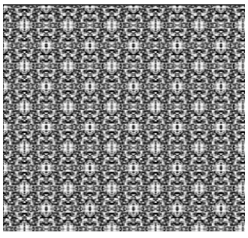
(a) Image1450 - Original



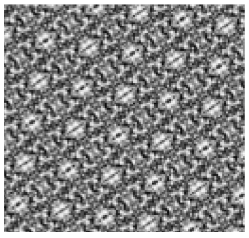
(b) Rotated by 345 degrees



(c) Scaled by a factor of 0.9

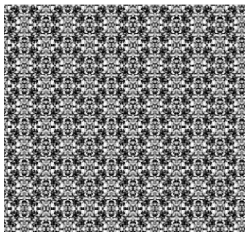


(d) Translated image

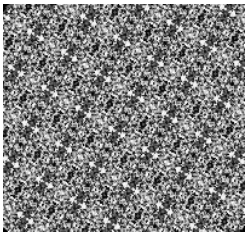


(e) Combined Augmentations on image 1450

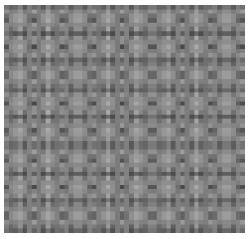
Figure 10: Augmentations for image 1450



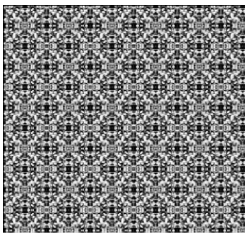
(a) Image17000 - Original



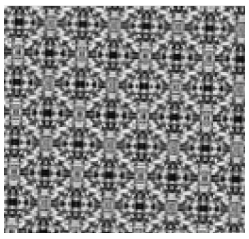
(b) Rotated by 60 degrees



(c) Scaled by a factor of 0.2



(d) Translated image



(e) Combined Augmentations on image 17000

Figure 11: Augmentations for image 17000

3.2 Starter Code

Network in the starter code has the following architecture:

INPUT - CONV 1 - ReLU - POOL 1 - FC 1 - DROPOUT - FC 2 - SOFTMAX
Batch Size: 250

INPUT image size: $256 \times 256 \times 1$

CONV 1 Layer:

1. Number of Filters: 20
2. Size of each filter: 5×5
3. Padding: [2,2]
4. Stride: [2,2]
5. Number of weights: $5 \times 5 \times 20 = \mathbf{500}$
6. Bias Units: 20
7. Output size:$\mathbf{128 \times 128 \times 20}$
8. Activation: ReLU
9. Total parameters to be learnt: **520**

POOL 1:

1. Output size: $\mathbf{64 \times \ 64 \times 20}$
2. Stride: [2,2]

FC 1:

1. Number of output neurons: 25
2. Input size: $(64 \times 64 \times 20) \times 250 = \mathbf{81920 \times 250}$
3. Output size: $\mathbf{250 \times 25}$
4. Number of weights = $81920 \times 25 = \mathbf{2048000}$
5. Number of bias units: 25×1
6. Total parameters to be learnt: **2048025**

DROPOUT Layer: Dropout Hyper-parameter: 0.25

FC 2:

1. Number of output neurons: 17

2. Input size: (250×25)
3. Output size: $\mathbf{250 \times 17}$
4. Number of weights = $\mathbf{25 \times 17 = 425}$
5. Number of bias units: 17×1
6. Activation Softmax
7. Total parameters to be learnt: **442**

Total number of parameters to be learnt in the entire network: **2048987**

The network is applied on a data set with 17000 training samples and 17000 test samples. Further, 10% of the training data is split into validation set.
The learning rates used are 5×10^{-4} and 1×10^{-4}
The code was run for 5 and 10 epochs with both the learning rates.

Following observations can be made from the results below:

1. From figures 18 and 25, it can be observed that, with a constant learning rate, as the number of epochs increases from 5 to 10, the train, test and validation accuracies also increase.
2. From figures 18 and 32, it can be observed that, when the learning rate decreases from 5×10^{-4} to 1×10^{-4},even though the train, test and validation accuracies increase, the training error is very high, when compared to the initial learning rate, i.e., the error never converges.

The results are as follows:

3.2.1 Starter Code on Original Data - 17000 samples - different learning rates and epochs

Starter Code - 5 epochs, learning rate $= 5 \times 10^{-4}$:
Training Accuracy = 67.54%
Validation Accuracy = 62.82%
Test Accuracy = 64.66%
Training Time = 358.47 seconds

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	226	137	167	370	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	900	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	459	441	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	456	444	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	491	409	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	900	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	348	552	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	900	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	855	45	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	858	42	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	301	599	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	242	658	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	893	7	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	900	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	4	896	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	895	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	890

Figure 12: Confusion Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.25111	0.15222	0.18556	0.41111	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	0.51	0.49	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	0.50667	0.49333	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	0.54556	0.45444	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.38667	0.61333	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.95	0.05	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0.95333	0.04667	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0.33444	0.66556	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0.26889	0.73111	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0.99222	0.00778	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00444	0.99556	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00556	0.99444	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01111	0.98889

Figure 13: Classification Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	312	130	206	352	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	701	299	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	670	330	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	705	295	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	556	444	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	997	3	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	23	394	583	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	345	650	5	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	970	30	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	995	5	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	983	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	981

Figure 14: Confusion Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.312	0.13	0.206	0.352	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	0.701	0.299	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	0.67	0.33	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	0.705	0.295	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.556	0.444	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.997	0.003	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0.023	0.394	0.583	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0.345	0.65	0.005	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0.97	0.03	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.995	0.005	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.017	0.983	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.019	0.981

Figure 15: Classification Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	39	14	15	32	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	58	42	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	57	43	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	50	50	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	42	58	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	92	8	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	95	5	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	36	64	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	37	63	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	2	98	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	2	98	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	3	97	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	98	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	97

Figure 16: Confusion Matrix for Validation Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.39	0.14	0.15	0.32	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	0.58	0.42	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	0.57	0.43	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	0.5	0.5	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.42	0.58	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.92	0.08	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0.95	0.05	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0.36	0.64	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0.37	0.63	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0.02	0.98	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0.98	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.03	0.97	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0.98	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.03	0.97

Figure 17: Classification Matrix for Validation Data set

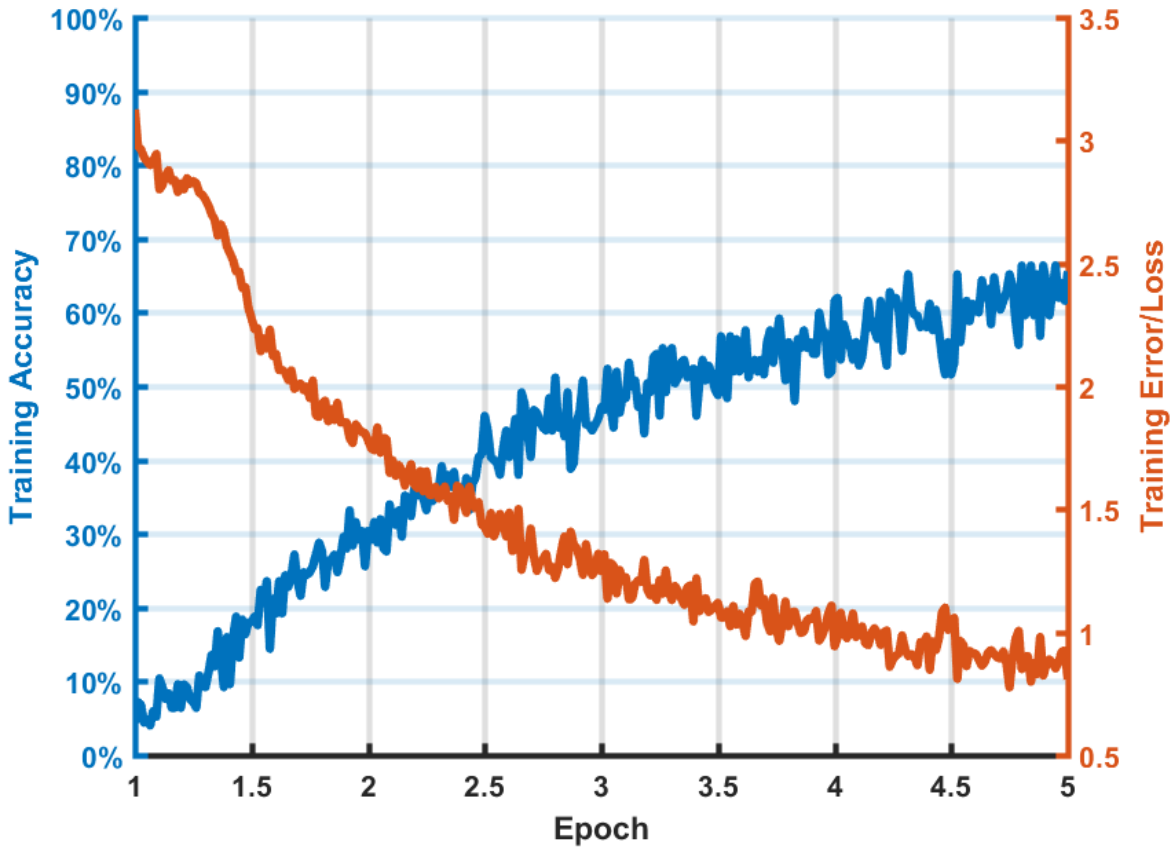


Figure 18: Error and Accuracy Plot

Starter Code - 10 epochs, learning rate $= 5 \times 10^{-4}$:
Training Accuracy = 71.97%
Validation Accuracy = 67.82%
Test Accuracy = 67.84%
Training Time = 767.14 seconds

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	798	102	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	614	202	84	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	94	806	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	93	790	17	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	0	900	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	683	217	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0	900	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	900	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	759	141	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	760	77	63	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	900	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	44	856	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	900	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	117	783	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	117	783	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	900	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	896

Figure 19: Confusion Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.88667	0.11333	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0.68222	0.22444	0.09333	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	0.10444	0.89556	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	0.10333	0.87778	0.01889	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	0.75889	0.24111	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.84333	0.15667	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0.84444	0.08556	0.07	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0.04889	0.95111	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.13	0.87	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.13	0.87	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00444	0.99556

Figure 20: Classification Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	953	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	701	233	66	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	152	848	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	145	774	81	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	679	321	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	831	169	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	838	57	105	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	115	884	1	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	220	780	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	226	764	10	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	988

Figure 21: Confusion Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.953	0.047	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0.701	0.233	0.066	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0	0.152	0.848	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	0.145	0.774	0.081	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	0.679	0.321	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.831	0.169	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0.838	0.057	0.105	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0.115	0.884	0.001	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.22	0.78	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.226	0.764	0.01	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.012	0.988

Figure 22: Classification Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	6	72	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	4	34	62	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	38	62	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	8	92	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	80	20	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	75	25	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	77	7	16	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	13	87	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	27	73	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	27	73	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	98

Figure 23: Confusion Matrix for Validation Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0.06	0.72	0.22	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0.04	0.34	0.62	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0	0.38	0.62	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0	0.08	0.92	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0	0.8	0.2	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.75	0.25	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0.77	0.07	0.16	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0.13	0.87	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.27	0.73	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.27	0.73	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0.98

Figure 24: Classification Matrix for Validation Data set

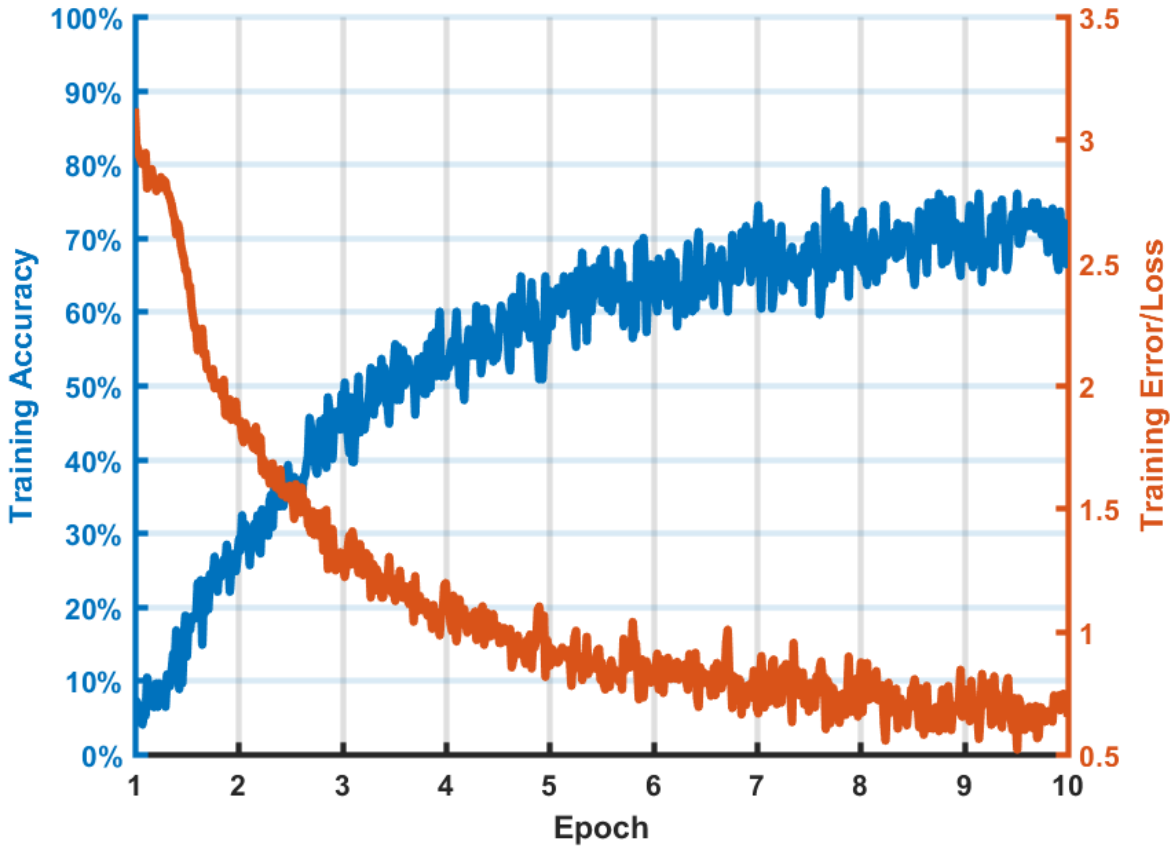


Figure 25: Error and Accuracy Plot

Starter Code - 5 epochs, learning rate $= 1 \times 10^{-4}$**:**
Training Accuracy = 72.97%
Validation Accuracy = 68.59%
Test Accuracy = 68.72%
Training Time = 296.76 seconds

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	806	94	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	517	383	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	423	477	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	200	700	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	200	700	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	240	660	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	332	126	442	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0	458	442	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0	900	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	818	82	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	785	115	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	900	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	900	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	8	892	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	1	899	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	899	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	895

Figure 26: Confusion Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.89556	0.10444	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0.57444	0.42556	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0.47	0.53	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0.22222	0.77778	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0.22222	0.77778	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0.26667	0.73333	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.36889	0.14	0.49111	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0	0.50889	0.49111	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0.90889	0.09111	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.87222	0.12778	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.00889	0.99111	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00111	0.99889	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00111	0.99889	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00556	0.99444

Figure 27: Classification Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	904	96	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	488	512	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	393	607	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	168	832	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	159	841	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	247	753	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	326	143	531	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0	473	527	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	945	55	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	894	106	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	994	6	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	1	989	10	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	986	14	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	993

Figure 28: Confusion Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.904	0.096	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0.488	0.512	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0.393	0.607	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0.168	0.832	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0.159	0.841	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0.247	0.753	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.326	0.143	0.531	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0	0.473	0.527	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0.945	0.055	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.894	0.106	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	0.994	0.006	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.001	0.989	0.01	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.986	0.014	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.007	0.993

Figure 29: Classification Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	13	56	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	48	52	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	24	76	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	22	78	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	11	89	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	30	12	58	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0	42	58	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	75	25	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	87	13	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	98	2	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	99	1	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	99	1	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	98	2	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	98

Figure 30: Confusion Matrix for Validation Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0.13	0.56	0.31	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0.48	0.52	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0.24	0.76	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0.22	0.78	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0.11	0.89	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.3	0.12	0.58	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0	0.42	0.58	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0.75	0.25	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.87	0.13	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	0.98	0.02	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0.99	0.01	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.99	0.01	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.98	0.02	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0.98

Figure 31: Classification Matrix for Validation Data set

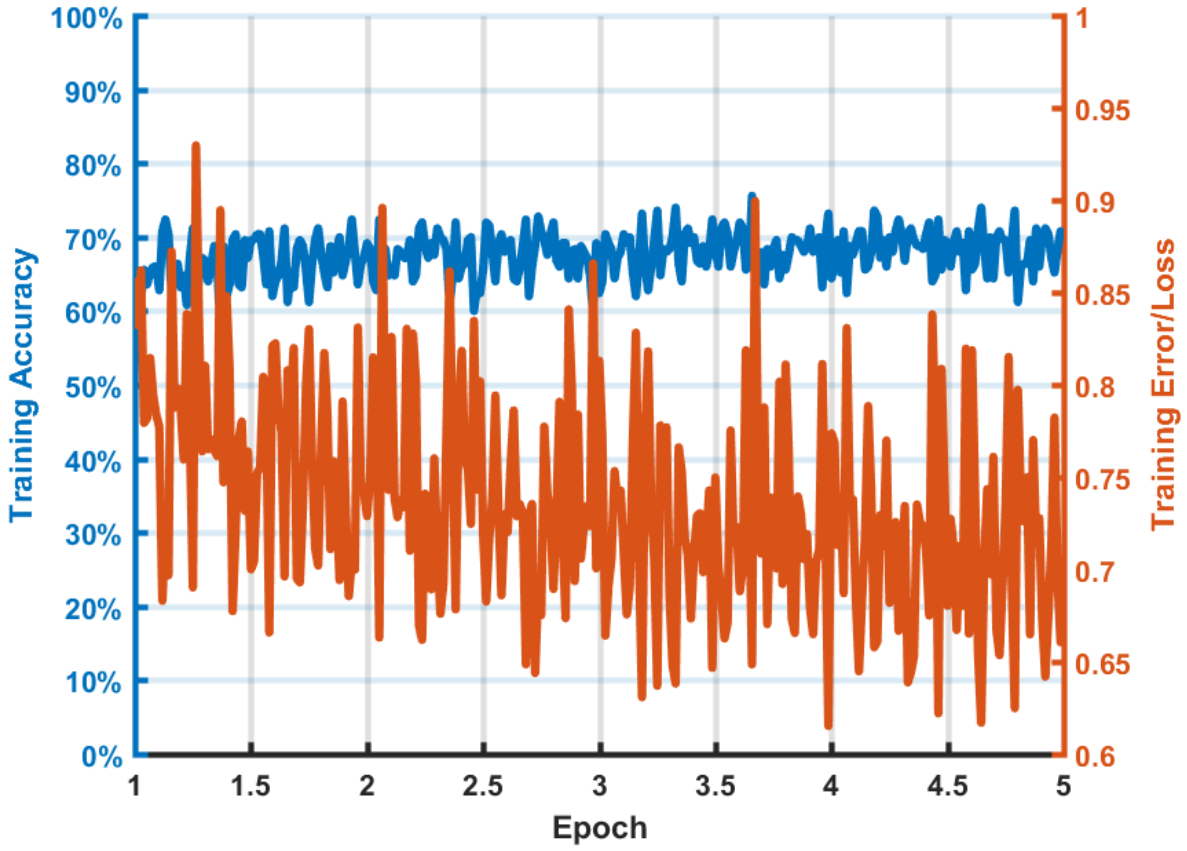


Figure 32: Error and Accuracy Plot

Starter Code - 10 epochs, learning rate $= 1 \times 10^{-4}$:
Training Accuracy = 80.39%
Validation Accuracy = 73.47%
Test Accuracy = 74.5%
Training Time = 765.4 seconds

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	684	216	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	552	348	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	482	418	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	652	248	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	652	248	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	654	246	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	724	176	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	60	840	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	60	840	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	811	89	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	809	91	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	900	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	900	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	900	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	900	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	900	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	900

Figure 33: Confusion Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.76	0.24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0.61333	0.38667	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0.53556	0.46444	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0.72444	0.27556	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0.72444	0.27556	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0.72667	0.27333	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.80444	0.19556	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0.06667	0.93333	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0.06667	0.93333	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0.90111	0.09889	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.89889	0.10111	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 34: Classification Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	763	237	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	541	459	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	427	573	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	598	402	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	595	405	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	621	379	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	709	239	52	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0	951	49	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	837	163	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	840	160	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	991	9	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	1	998	1	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	993	7
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000

Figure 35: Confusion Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.763	0.237	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0.541	0.459	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0.427	0.573	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0.598	0.402	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0.595	0.405	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0.621	0.379	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.709	0.239	0.052	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0	0.951	0.049	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0.837	0.163	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.84	0.16	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0.991	0.009	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.001	0.998	0.001	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.993	0.007
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 36: Classification Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	97	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	76	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	59	41	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	78	22	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	78	22	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	72	28	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	88	12	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	6	94	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	6	94	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	76	24	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	84	16	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	96	4	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	97	3	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

Figure 37: Confusion Matrix for Validation Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.97	0.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0.76	0.24	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0.59	0.41	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0.78	0.22	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0.78	0.22	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0.72	0.28	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.88	0.12	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0.06	0.94	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0.06	0.94	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0.76	0.24	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.84	0.16	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0.96	0.04	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.97	0.03	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 38: Classification Matrix for Validation Data set

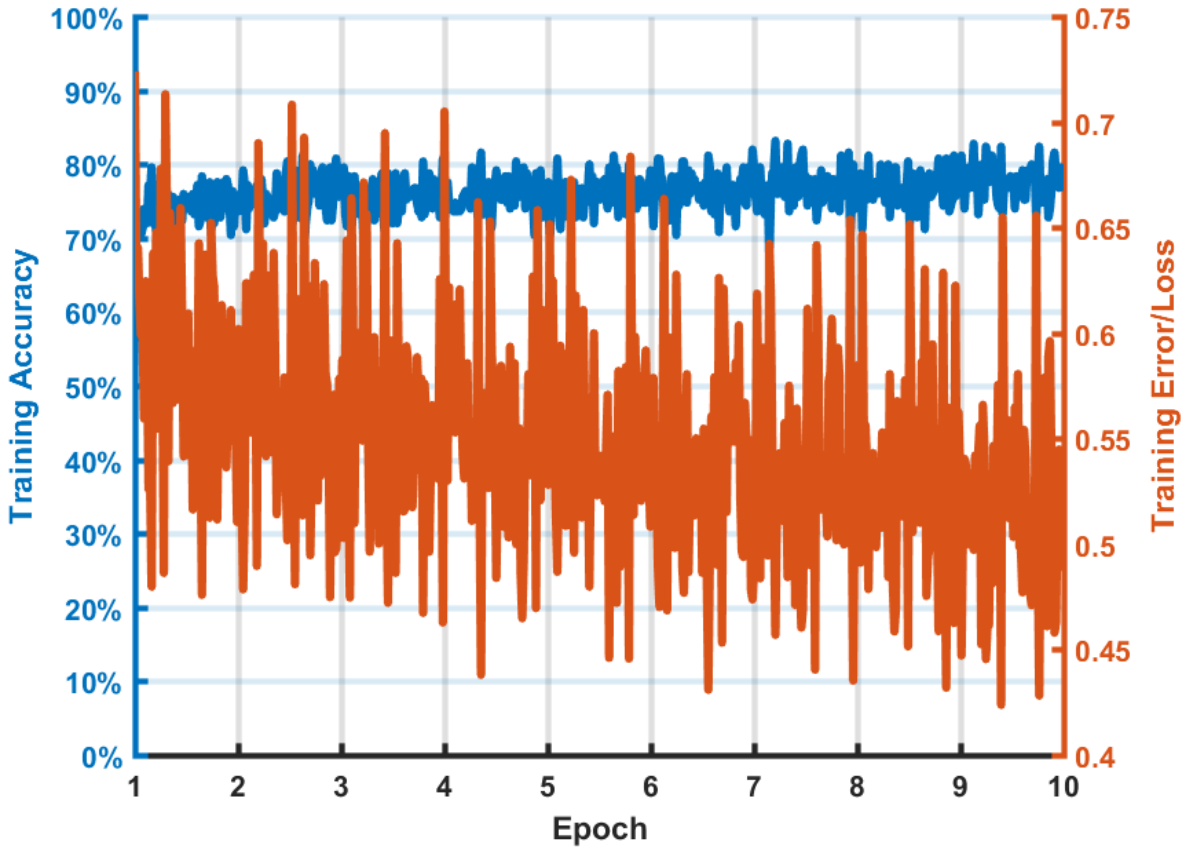


Figure 39: Error and Accuracy Plot

3.2.2 Starter code on Augmented data - 85000 samples, learning rate = 5×10^{-4}, epochs = 10

The starter code is run on the augmented dataset. The results are as follows:
Training Accuracy = 20.78%
Validation Accuracy = 18.96%
Test Accuracy = 17.88%
Training Time = 11716.77 seconds
From 46 it can be observed that the network in the starter code which gave very good accuracies on the original data fails to provide good results on augmented data. The reason is that the network is not deep enough to handle the variations in the augmented data set. Hence, increasing the number of hidden layers can improve the accuracies.

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	4500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	4500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	679	3821	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	4500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	1218	3282	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	152	696	3652	0	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	3290	1081	129	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	1059	3441	0	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	2268	2232	0	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	2417	202	1881	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	3972	528	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	465	328	3707	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	0	4055	445	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4500	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2192	2308	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4500	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1308	3192

Figure 40: Confusion Matrix for Train Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0.15089	0.84911	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0.27067	0.72933	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0.03378	0.15467	0.81156	0	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0.73111	0.24022	0.02867	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0.23533	0.76467	0	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0.504	0.496	0	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0.53711	0.04489	0.418	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	0.88267	0.11733	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	0.10333	0.07289	0.82378	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	0	0.90111	0.09889	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.48711	0.51289	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.29067	0.70933

Figure 41: Classification Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	695	305	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	805	195	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	716	123	161	0	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	98	291	428	183	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	766	234	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	822	42	136	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	602	207	87	104	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	439	561	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	768	232	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	230	770

Figure 42: Confusion Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0.695	0.305	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0.805	0.195	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0.716	0.123	0.161	0	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0.098	0.291	0.428	0.183	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.766	0.234	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0.822	0.042	0.136	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0.602	0.207	0.087	0.104	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.439	0.561	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.768	0.232	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.23	0.77

Figure 43: Classification Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	114	386	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	160	340	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	11	80	409	0	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	331	119	50	0	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	114	386	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	232	268	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	283	24	193	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	428	72	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	46	22	432	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0	415	85	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	244	256	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	172	328

Figure 44: Confusion Matrix for Validation Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0.228	0.772	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0.32	0.68	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0.022	0.16	0.818	0	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0.662	0.238	0.1	0	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0.228	0.772	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.464	0.536	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0.566	0.048	0.386	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.856	0.144	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	0.092	0.044	0.864	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0	0.83	0.17	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.488	0.512	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.344	0.656

Figure 45: Classification Matrix for Validation Data set

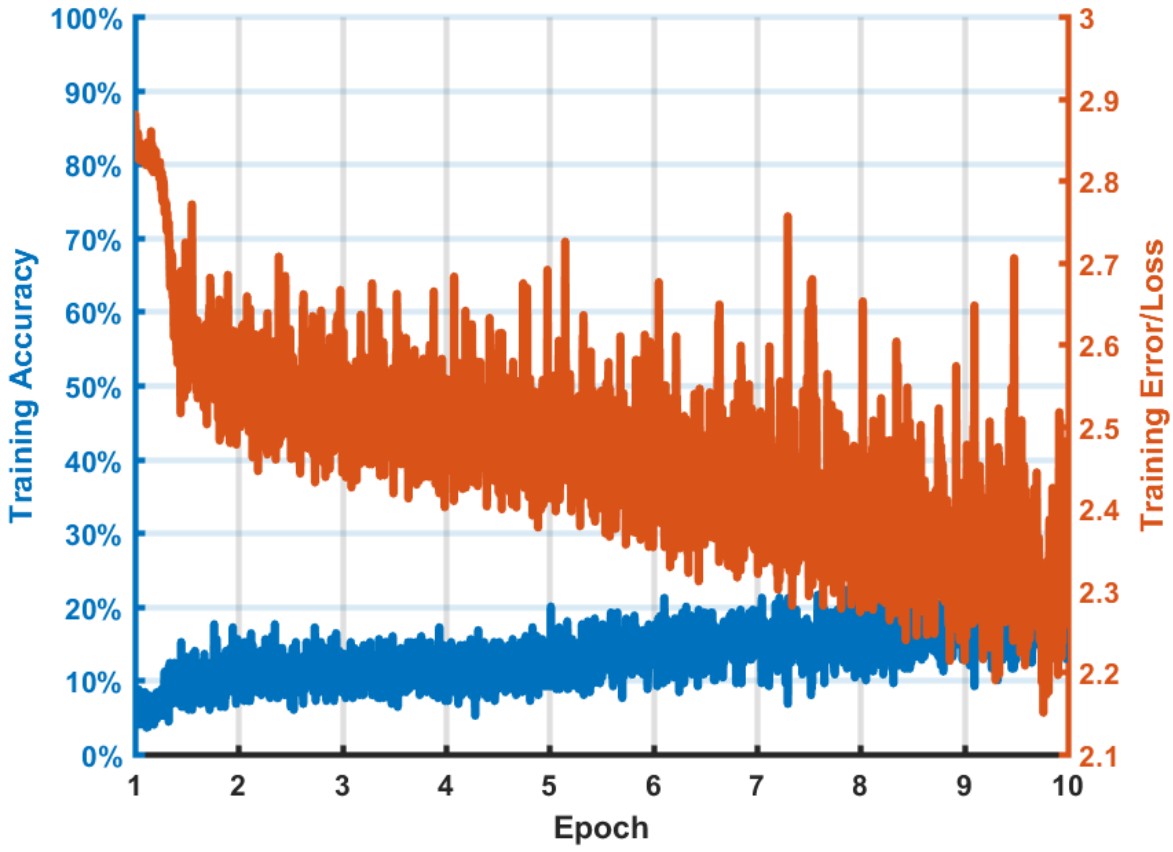


Figure 46: Error and Accuracy Plot

3.3 Skinny and Wide Network Architectures - Training on Augmented Data sets

The Augmented data set has 85000 training images and 17000 test images. 10% of training data is further divided into validation set. Since the training, test and validation accuracies of running starter code on the augmented dataset are approximately 15% to 20%, t can be inferred that the number of hidden layers in the starter code CNN architecture are not sufficient to handle the augmented data set. Hence, two different network architectures have been designed:

1. Skinny Network: It has more number of hidden layers, but less number of filters in each convolution layer.
2. It has lesser number of hidden layers when compared to skinny network, but each convolution layer has higher number of filters.

The following observations can be made from this section:

1. The two deep network architectures perform better than the network in the starter code on Augmented dataset, with increased accuracies.
2. From figures 53 and 60 it can be observed that the skinny netwrok is able to perform better than the wide netwrok architecture. The test accuracy of wide network is very less when compared to train and validation accuracies. This means that the wide network is over fitting the data, hence decreasing the overall efficiency.
3. Also, the wide network takes more time to train than the skinny network.

The filter architectures are as follows:

3.3.1 Skinny Network Architecture

: **INPUT - CONV 1 - ReLU - POOL 1 -CONV 2 - ReLU - POOL 2 - CONV 3 - ReLU - POOL 3 - FC 1 - DROPOUT - FC 2 - SOFTMAX**
Batch Size: 250

INPUT image size: $128 \times 128 \times 1$

CONV 1 Layer:

1. Number of Filters: 40
2. Size of each filter: 5×5
3. Padding: [2,2]
4. Stride: [1,1]
5. Number of weights: $5 \times 5 \times 40 = \mathbf{1000}$
6. Bias Units: 40

7. Output size:**128** $\times$ **128** $\times$ **40**
8. Activation: ReLU
9. Total parameters to be learnt: **1040**

POOL 1:

1. Output size: **64** $\times$ **64** $\times$ **40**
2. Stride: [2,2]

CONV 2 Layer:

1. Number of Filters: 40
2. Size of each filter: 5×5
3. Padding: [1,1]
4. Stride: [1,1]
5. Number of weights: $40 \times 25 \times 40 =$ **40000**
6. Bias Units: 40
7. Output size:**62** $\times$ **62** $\times$ **40**
8. Activation: ReLU
9. Total parameters to be learnt: **40040**

POOL 2:

1. Output size: **31** $\times$ **31** $\times$ **40**
2. Stride: [2,2]

CONV 3 Layer:

1. Number of Filters: 80
2. Size of each filter: 3×3
3. Padding: [1,1]
4. Stride: [1,1]
5. Number of weights: $9 \times 40 \times 80 =$ **28800**
6. Bias Units: 80
7. Output size:**31** $\times$ **31** $\times$ **80**
8. Activation: ReLU

9. Total parameters to be learnt: **28880**

POOL 3:

1. Output size: $\mathbf{15} \times \mathbf{15} \times \mathbf{80}$
2. Stride: [2,2]

FC 1:

1. Number of output neurons: 100
2. Input size: $(15 \times 15 \times 80) \times 250 = \mathbf{18000} \times \mathbf{250}$
3. Output size: $\mathbf{250} \times \mathbf{100}$
4. Number of weights $=18000 \times 100 = \mathbf{1800000}$
5. Number of bias units: 100×1
6. Total parameters to be learnt: **1800100**

DROPOUT Layer: Dropout Hyper-parameter: 0.25

FC 2:

1. Number of output neurons: 17
2. Input size: (250×100)
3. Output size: $\mathbf{250} \times \mathbf{17}$
4. Number of weights $= \mathbf{100} \times \mathbf{17} = \mathbf{1700}$
5. Number of bias units: 17×1
6. Activation Softmax
7. Total parameters to be learnt: **1717**

Total number of parameters to be learnt in the entire network: **1871777**

Skinny Network on Augmented Data: 15 epochs, learning Rate $= 1 \times 10^{-3}$:
The results are as follows:
Training Accuracy = 65.5%
Test Accuracy = 53.79%
Validation Accuracy = 60.11%
Training Time = 2463.63 seconds

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	3927	573	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	4500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	392	3321	787	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	4500	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	302	4198	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	465	4035	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	480	3798	222	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	3975	525	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	3786	714	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	663	3837	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	3073	1427	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	4500	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	643	3857	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	4040	460	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	2474	2026	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2097	2403	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	452	4048

Figure 47: Confusion Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.87267	0.12733	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0.08711	0.738	0.17489	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0.06711	0.93289	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0.10333	0.89667	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0.10667	0.844	0.04933	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0.88333	0.11667	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0.84133	0.15867	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0.14733	0.85267	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.68289	0.31711	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0.14289	0.85711	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.89778	0.10222	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.54978	0.45022	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.466	0.534	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.10044	0.89956

Figure 48: Classification Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	700	300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	886	114	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	629	371	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	740	260	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	810	190	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	941	59	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	807	193	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	535	465	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	504	163	333	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	848	152	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	624	376	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	602	398	0	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	76	699	225	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	223	777

Figure 49: Confusion Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.7	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0.886	0.114	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0	0.629	0.371	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	0.74	0.26	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0	0.81	0.19	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0	0.941	0.059	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.807	0.193	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0.535	0.465	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0.504	0.163	0.333	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.848	0.152	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0.624	0.376	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0.602	0.398	0	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0.076	0.699	0.225	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.223	0.777

Figure 50: Classification Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	413	87	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	24	350	126	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	41	459	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	67	433	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	72	410	18	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	382	118	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	349	75	76	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	294	206	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	128	372	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	185	208	107	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	340	160	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	127	373

Figure 51: Confusion Matrix for Validation Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.826	0.174	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0.048	0.7	0.252	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0.082	0.918	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0.134	0.866	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0.144	0.82	0.036	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	0.764	0.236	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0	0.698	0.15	0.152	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.588	0.412	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0.256	0.744	0	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0.37	0.416	0.214	0	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.68	0.32	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.254	0.746

Figure 52: Classification Matrix for Validation Data set

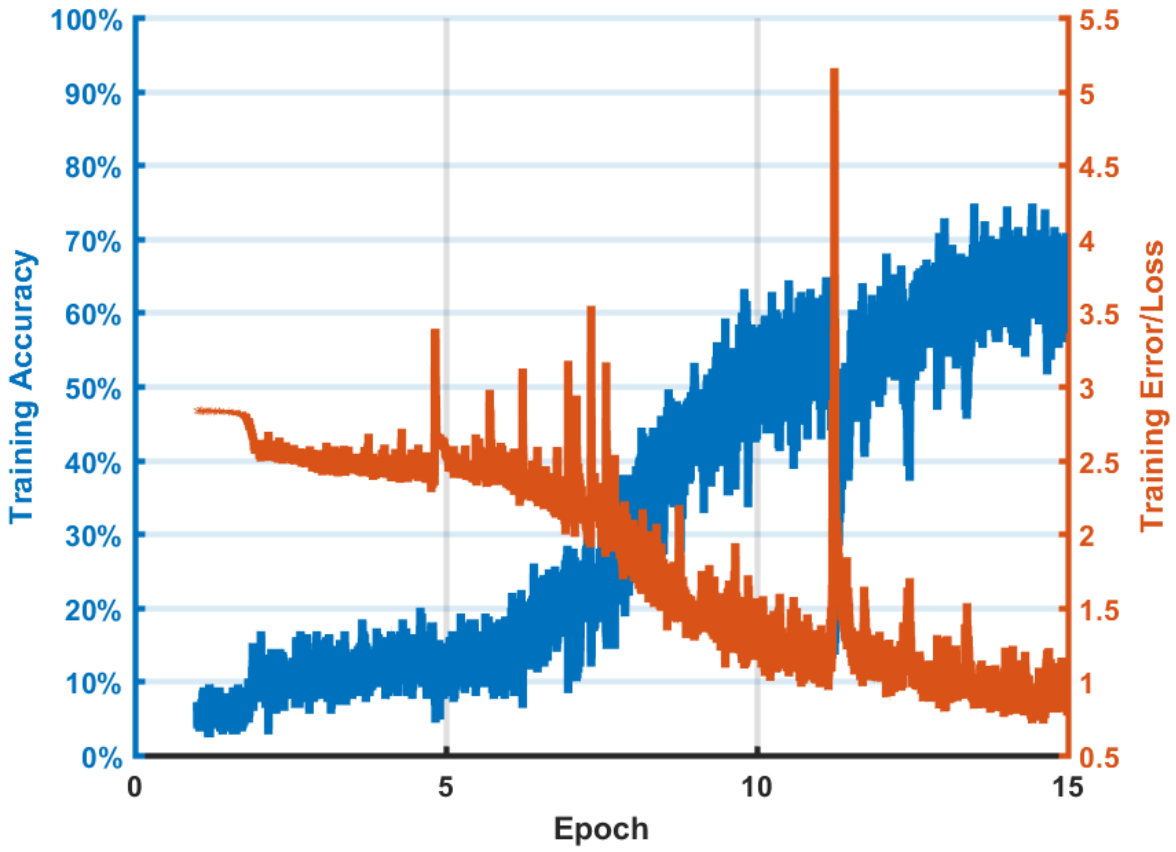


Figure 53: Error and Accuracy Plot

3.3.2 Wide Network Architecture

Wide Network has the following architecture:
INPUT - CONV 1 - ReLU - POOL 1 - CONV 2 - ReLU - POOL 2 - FC 1 - DROPOUT - FC 2 - SOFTMAX
Batch Size: 250

INPUT image size: $128 \times 128 \times 1$

CONV 1 Layer:

1. Number of Filters: 80
2. Size of each filter: 5×5
3. Padding: [2,2]
4. Stride: [1,1]
5. Number of weights: $5 \times 5 \times 80 = \mathbf{2000}$
6. Bias Units: 80
7. Output size:$\mathbf{128 \times 128 \times 80}$
8. Activation: ReLU
9. Total parameters to be learnt: **2080**

POOL 1:

1. Output size: $\mathbf{64 \times 64 \times 80}$
2. Stride: [2,2]

CONV 2 Layer:

1. Number of Filters: 120
2. Size of each filter: 5×5
3. Padding: [1,1]
4. Stride: [1,1]
5. Number of weights: $25 \times 80 \times 120 = \mathbf{240000}$
6. Bias Units: 120
7. Output size:$\mathbf{62 \times 62 \times 120}$
8. Activation: ReLU
9. Total parameters to be learnt: **240120**

POOL 2:

1. Output size: $\mathbf{31 \times 31 \times 120}$
2. Stride: [2,2]

FC 1:

1. Number of output neurons: 100
2. Input size: $(31 \times 31 \times 120) \times 250 = \mathbf{115320 \times 250}$
3. Output size: $\mathbf{250 \times 100}$
4. Number of weights = $115320 \times 100 = \mathbf{11532000}$
5. Number of bias units: 100×1
6. Total parameters to be learnt: **11532100**

DROPOUT Layer: Dropout Hyper-parameter: 0.25

FC 2:

1. Number of output neurons: 17
2. Input size: (250×100)
3. Output size: $\mathbf{250 \times 17}$
4. Number of weights = $\mathbf{100 \times 17 = 1700}$
5. Number of bias units: 17×1
6. Activation Softmax
7. Total parameters to be learnt: **1717**

Total number of parameters to be learnt in the entire network: **11776017**

Wide Network on Augmented Data: 15 epochs, learning Rate = 1×10^{-3}:
The results are as follows:
Training Accuracy = 54.85%
Test Accuracy = 40.71%
Validation Accuracy = 38.65%
Training Time = 5367.95 seconds

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	2892	1608	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	4500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	3449	1051	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	865	3635	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	2329	2171	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	2885	1615	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	793	1931	1776	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	4500	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	886	2377	1237	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	4500	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	25	2919	1556	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	685	3815	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1446	2471	583	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4500	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	165	4335	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4500	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1467	3033

Figure 54: Confusion Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.64267	0.35733	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0.76644	0.23356	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0.19222	0.80778	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0.51756	0.48244	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0.64111	0.35889	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0.17622	0.42911	0.39467	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.19689	0.52822	0.27489	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0.00556	0.64867	0.34578	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	0.15222	0.84778	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0.32133	0.54911	0.12956	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.03667	0.96333	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.326	0.674

Figure 55: Classification Matrix for Train Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	851	149	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	137	510	353	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	848	152	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	11	774	215	0	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	241	759	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	570	430	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	141	859	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	410	590	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	14	651	335	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	811	189	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	409	591	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	389	611	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	605	395

Figure 56: Confusion Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.851	0.149	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0.137	0.51	0.353	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0.848	0.152	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0.011	0.774	0.215	0	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0.241	0.759	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.57	0.43	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0.141	0.859	0	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0.41	0.59	0	0	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0.014	0.651	0.335	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0.811	0.189	0	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.409	0.591	0	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.389	0.611	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.605	0.395

Figure 57: Classification Matrix for Test Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	294	206	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	399	101	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	108	392	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	224	276	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	246	247	7	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	172	328	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	500	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	48	269	183	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	465	35	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	235	234	31	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	79	265	156	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	427	73	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	223	277

Figure 58: Confusion Matrix for Validation Data set

ACTUAL LABELS \ PREDICTED LABELS	P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
P1	0.588	0.412	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PM	0	0.798	0.202	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PG	0	0	0.216	0.784	0	0	0	0	0	0	0	0	0	0	0	0	0
CM	0	0	0	0.448	0.552	0	0	0	0	0	0	0	0	0	0	0	0
PMM	0	0	0	0	0.492	0.494	0.014	0	0	0	0	0	0	0	0	0	0
PMG	0	0	0	0	0	0	0.344	0.656	0	0	0	0	0	0	0	0	0
PGG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CMM	0	0	0	0	0	0	0	0.096	0.538	0.366	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0	0.93	0.07	0	0	0	0	0	0
P4M	0	0	0	0	0	0	0	0	0	0	0.47	0.468	0.062	0	0	0	0
P4G	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0.158	0.53	0.312	0	0
P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.854	0.146	0
P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.446	0.554

Figure 59: Classification Matrix for Validation Data set

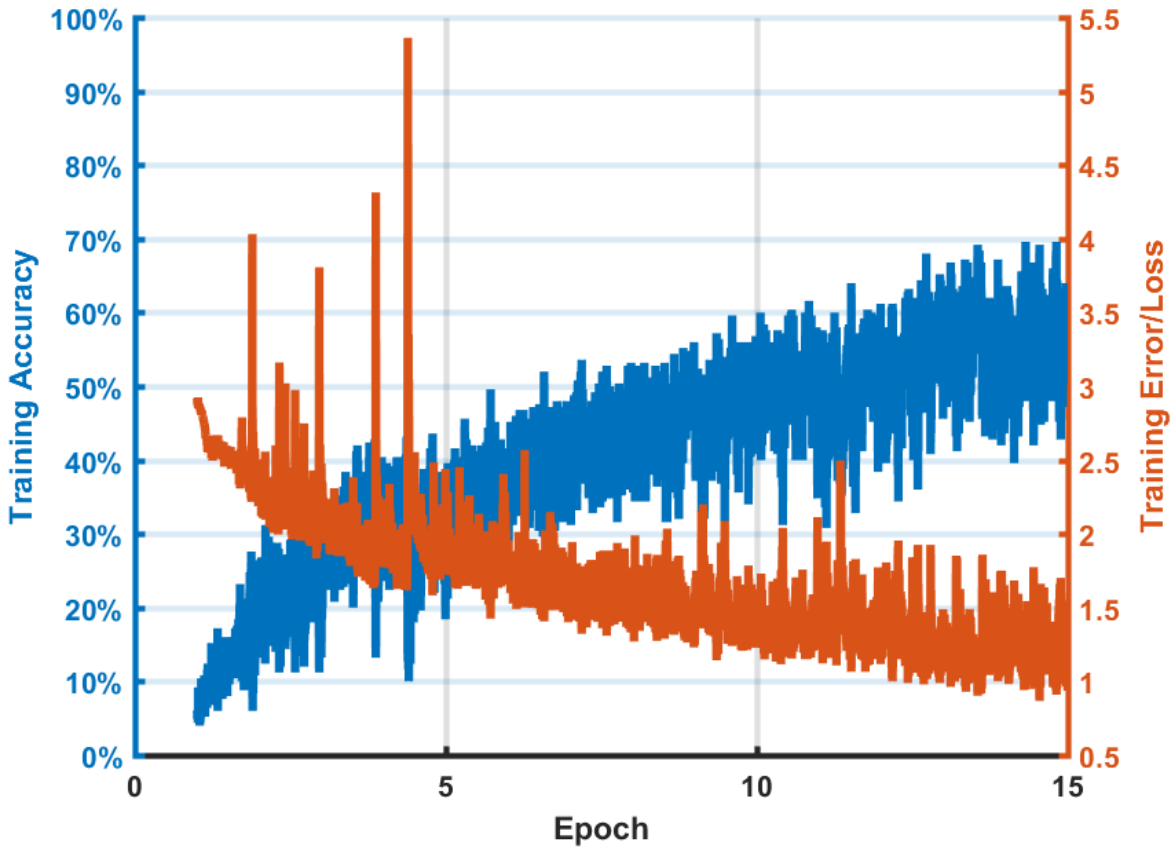


Figure 60: Error and Accuracy Plot

3.4 Application of Deep CNNs on the Original Dataset - Transfer Learning

It is evident from the previous section that the Deeper CNN architectures - Skinny and Wide Networks give good accuracies on Augmented data set. The perfomance of these networks on Augmented data is better than the original network in the starter code. Hence in this section, these two networks have been applied on the original data set to observe the changes in accuracy and performance.

The method of application of a trained Neural network model on a smaller data set in order to increase efficiency of training is referred to as Transfer learning. Here, the neural net models of both Skinny and Wide networks have been extracted and have been used to train on the original data set. Since these networks were designed to be applied on a image size of 128×128 and the images in original data are of size 256×256, the images have been re-sized to fit the newtorks. The results are as follows:
From this section, it can e observed that:

1. The deep CNN architectures when trained on original data set, surpassed the results of the baseline model in section 1 of results.

2. With a training for 5 and 3 epochs for skinny and wide respectively, the accuracies are as high as 90%.

3.4.1 Skinny Network Model on Original Dataset

The code was run for 5 epochs at a learning rate of 1×10^{-3}
The results are as follows:
Training Accuracy = 91.24%
Validation Accuracy = 90.12%
Test Accuracy = 89.59%
Training time = 164.15 seconds

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	865	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	700	200	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	702	198	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	528	372	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	490	410	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	469	431	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	297	603	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0	296	604	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	332	568	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	508	392	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	743	157	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	900	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	10	878	12	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	900	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	21	879	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	844	42
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	900

Figure 61: Confusion Matrix for Train Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	0.961111	0.038889	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	0.777778	0.222222	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	0.78	0.22	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	0.586667	0.413333	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	0.544444	0.455556	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	0.521111	0.478889	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	0.33	0.67	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0	0.328889	0.671111	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	0.368889	0.631111	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	0.564444	0.435556	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	0.825556	0.174444	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0.011111	0.975556	0.013333	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.023333	0.976667	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.015556	0.937778	0.046667
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 62: Classification Matrix for Train Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	23	739	238	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	762	238	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	415	585	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	372	628	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	346	654	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	112	888	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0	115	885	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	172	828	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	521	479	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	785	215	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	33	937	30	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	40	960	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	891	63
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000

Figure 63: Confusion Matrix for Test Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0.023	0.739	0.238	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	0.762	0.238	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	0.415	0.585	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	0.372	0.628	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	0.346	0.654	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	0.112	0.888	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0	0.115	0.885	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	0.172	0.828	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	0.521	0.479	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	0.785	0.215	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0.033	0.937	0.03	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.04	0.96	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.046	0.891	0.063
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 64: Classification Matrix for Test Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	82	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	80	20	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	45	55	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	38	62	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	35	65	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	12	88	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0	12	88	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	19	81	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	56	44	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	80	20	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	2	98	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	1	98	1	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	10
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

Figure 65: Confusion Matrix for Validation Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	0.82	0.18	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	0.8	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	0.45	0.55	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	0.38	0.62	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	0.35	0.65	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	0.12	0.88	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0	0.12	0.88	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	0.19	0.81	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	0.56	0.44	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	0.8	0.2	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0.98	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01	0.98	0.01	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9	0.1
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 66: Classification Matrix for Validation Data set

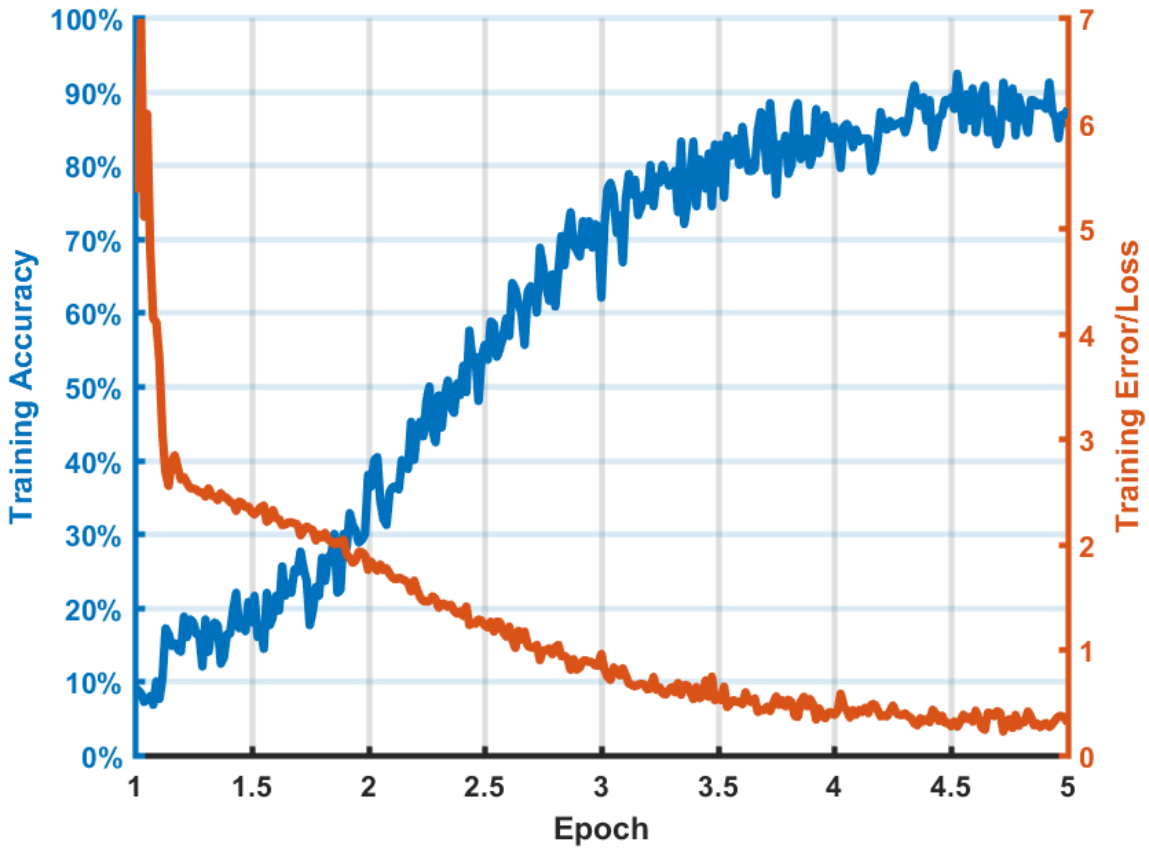


Figure 67: Error and Accuracy Plot

3.4.2 Wide Network Model on Original Dataset

The code was run for 3 epochs at a learning rate of 1×10^{-3}
The results are as follows:
Training Accuracy = 90.25%
Validation Accuracy = 88.65%
Test Accuracy = 87.18%
Training time = 267.25 seconds

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	142	758	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	847	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	833	67	0	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	900	0	0	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	164	736	0	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	171	729	0	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	185	715	0	0	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	421	479	0	0	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	452	448	0	0	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	432	468	0	0	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	203	199	498	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	159	741	0	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	150	750	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	60	840	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	845	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	858

Figure 68: Confusion Matrix for Train Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0.157778	0.842222	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0.941111	0.058889	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0.925556	0.074444	0	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0.182222	0.817778	0	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0.19	0.81	0	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0.205556	0.794444	0	0	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0.467778	0.532222	0	0	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0.502222	0.497778	0	0	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0.48	0.52	0	0	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0.225556	0.221111	0.553333	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0.176667	0.823333	0	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.166667	0.833333	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.066667	0.933333	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.061111	0.938889	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.046667	0.953333

Figure 69: Classification Matrix for Train Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	256	744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	127	873	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	71	929	0	0	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	358	642	0	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	371	629	0	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	432	568	0	0	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	767	233	0	0	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	815	185	0	0	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	773	227	0	0	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	493	141	366	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	281	719	0	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	240	760	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	101	899	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	89	911	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	61	939

Figure 70: Confusion Matrix for Test Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0.256	0.744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0.127	0.873	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0.071	0.929	0	0	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0.358	0.642	0	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0.371	0.629	0	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0.432	0.568	0	0	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0.767	0.233	0	0	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0.815	0.185	0	0	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0.773	0.227	0	0	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0.493	0.141	0.366	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0.281	0.719	0	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.24	0.76	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.101	0.899	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.089	0.911	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.061	0.939

Figure 71: Classification Matrix for Test Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	19	81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	97	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	94	6	0	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	17	83	0	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	18	82	0	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	21	79	0	0	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	56	44	0	0	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	62	38	0	0	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	61	39	0	0	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	38	24	38	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	26	74	0	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	25	75	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	12	88	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	90	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	92

Figure 72: Confusion Matrix for Validation Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0.19	0.81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0.97	0.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0.94	0.06	0	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0.17	0.83	0	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0.18	0.82	0	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0.21	0.79	0	0	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0.56	0.44	0	0	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0.62	0.38	0	0	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0.61	0.39	0	0	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0.38	0.24	0.38	0	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0.26	0.74	0	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.75	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.12	0.88	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.9	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.08	0.92

Figure 73: Classification Matrix for Validation Data set

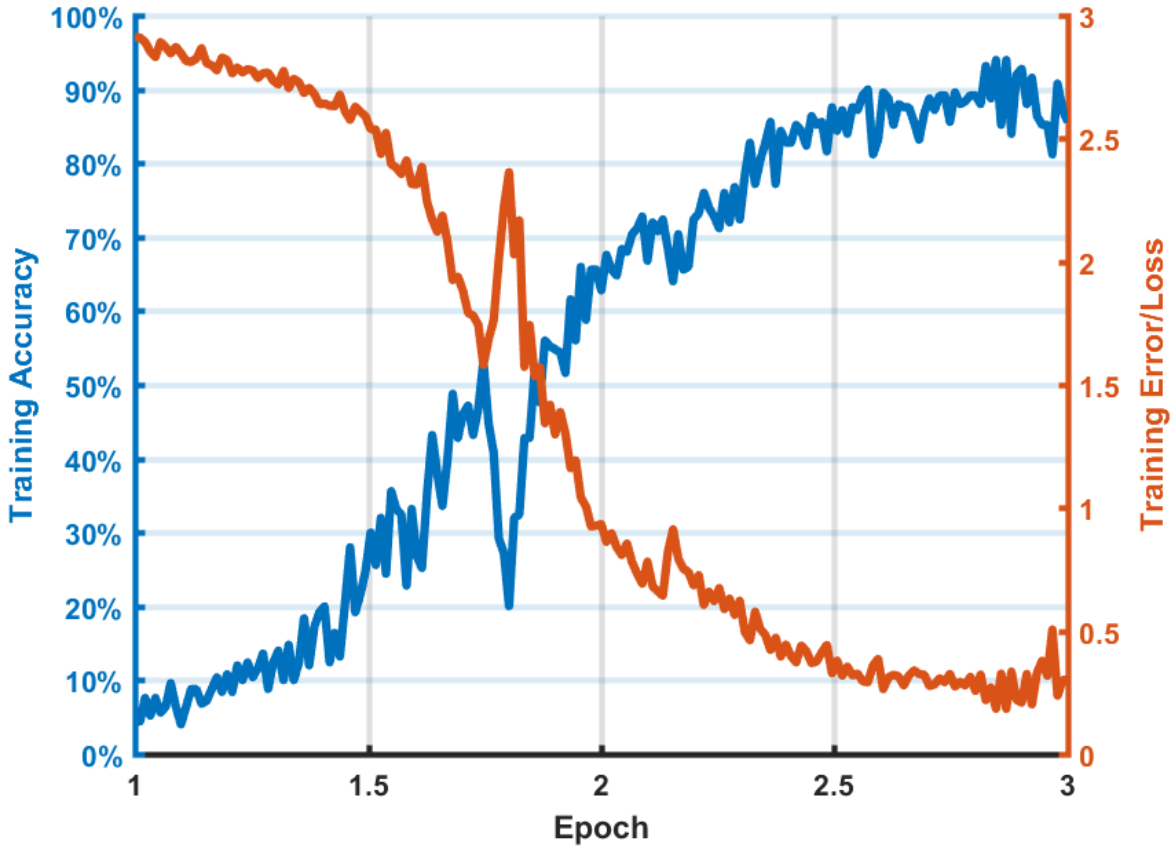


Figure 74: Error and Accuracy Plot

3.5 Visualization of Convolution Filters

In this section, activations/weights of each filter of convolution layers and activations of neurons of fully connected layers of each network architecture are visulaized using a function called **deepDreamImage** in MATLAB. The images from each filter/neuron have been consolidated into a single images using the matlab function **montage**.
The results are as follows:

3.5.1 Network in Starter Code - 10 epochs - learning Rate = 5×10^{-4}

It can be seen from figure 75 that there are 20 filter banks, each with 25 weights. Hence, this image has 500 5×5 images, each corresponding to one weight. This 5×5 image has the features that will excite that particular neuron.

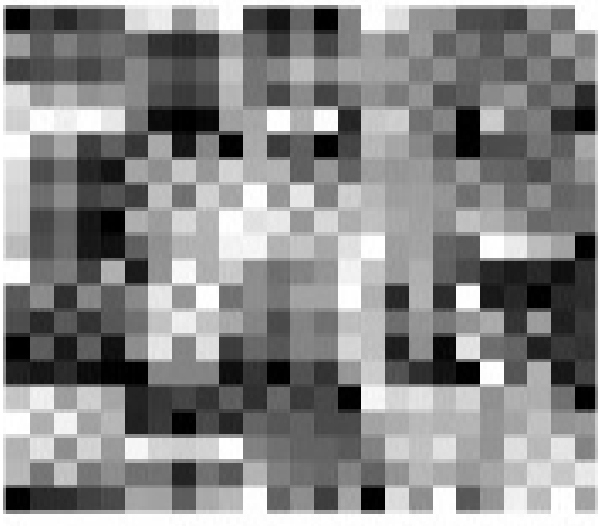


Figure 75: 20 filters of CONV 1

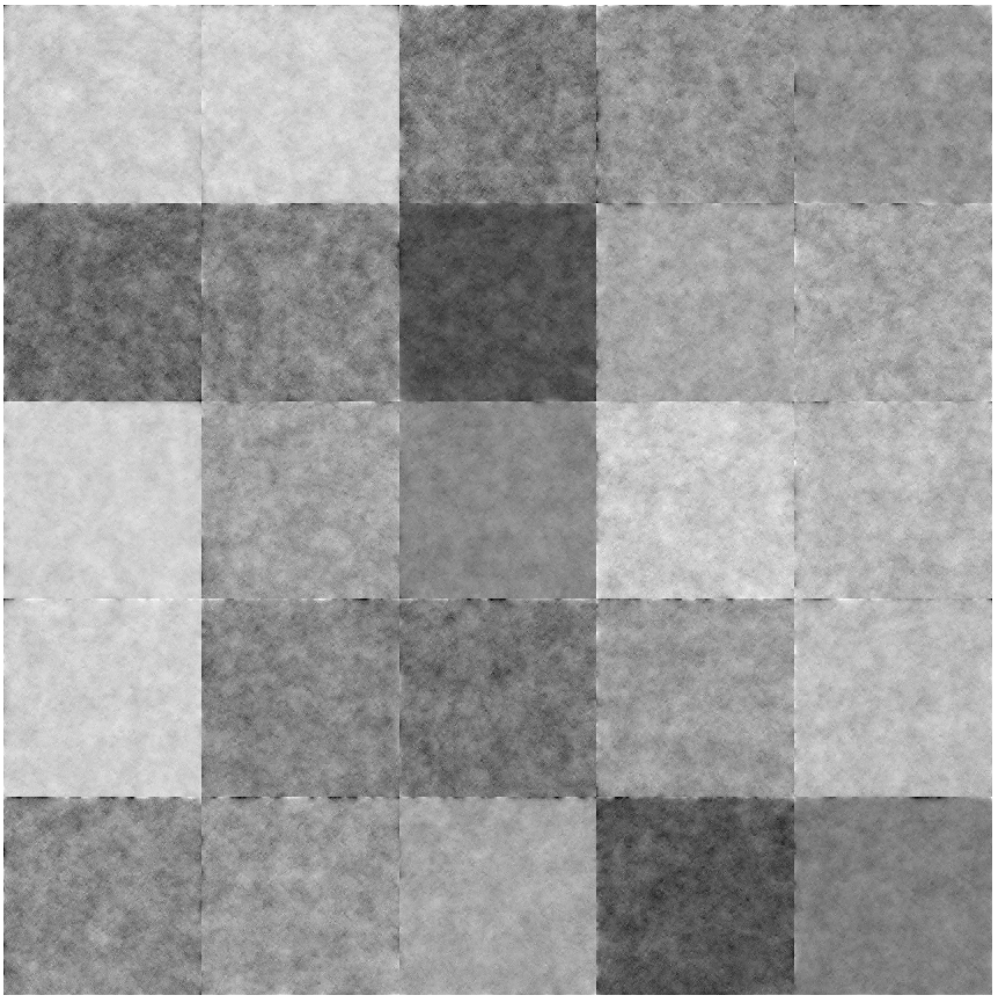


Figure 76: 25 neurons of FC 1

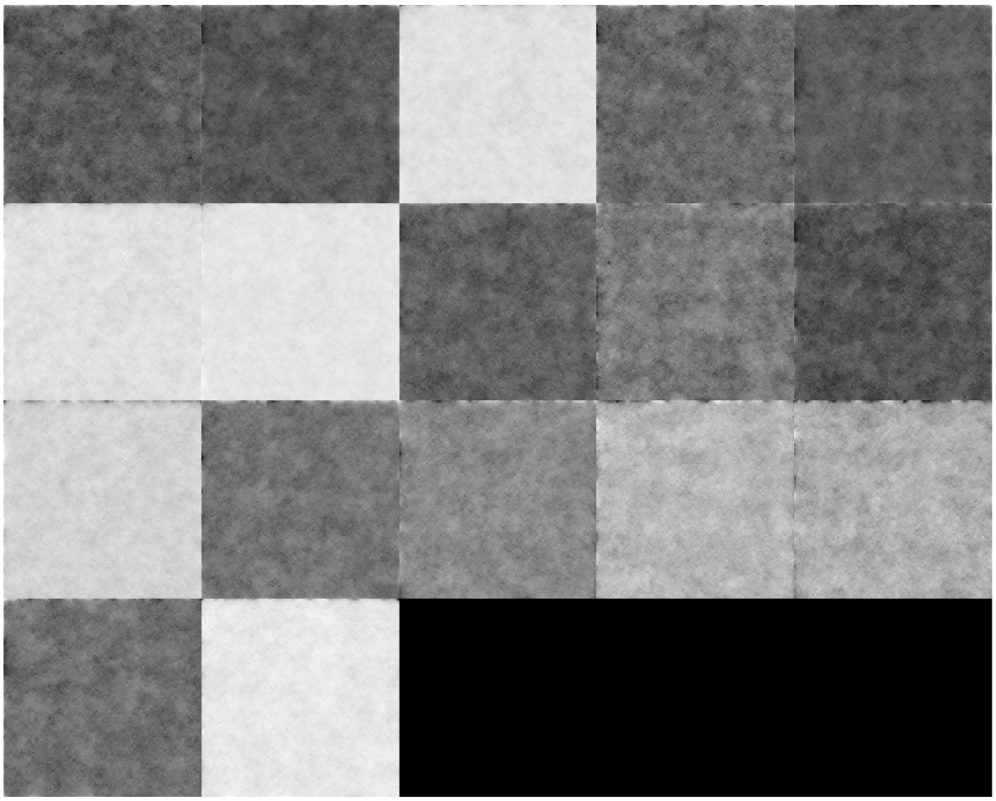


Figure 77: 17 neurons of FC 2

3.5.2 Skinny Network Architecture - 15 epochs, learning Rate = 1×10^{-3}

Figure 78: 40 filters of CONV 1

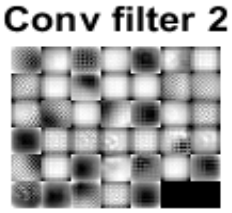


Figure 79: 40 filters of CONV 2

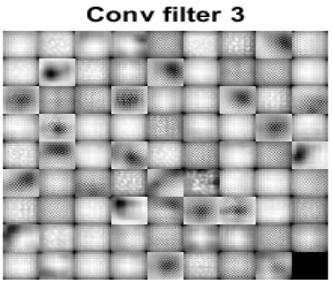


Figure 80: 80 filters of CONV 3

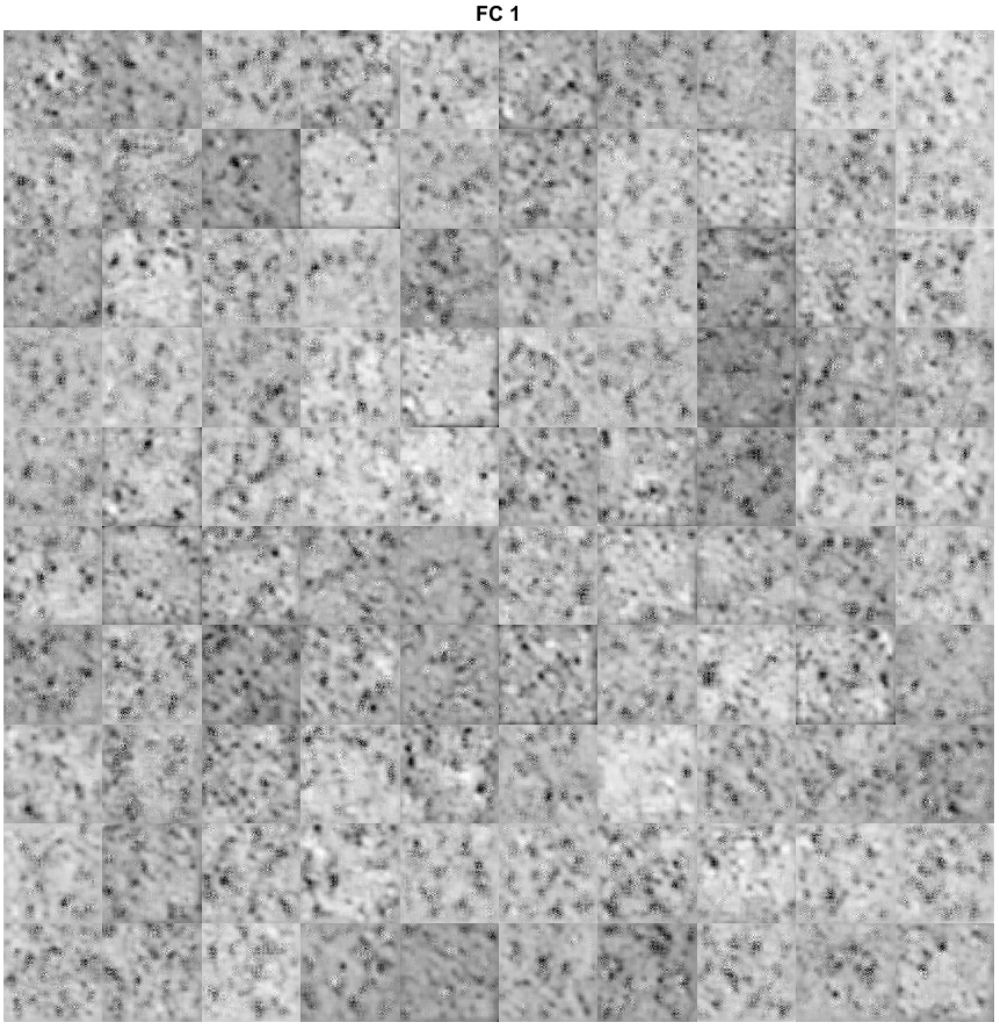


Figure 81: 100 neurons of FC 1

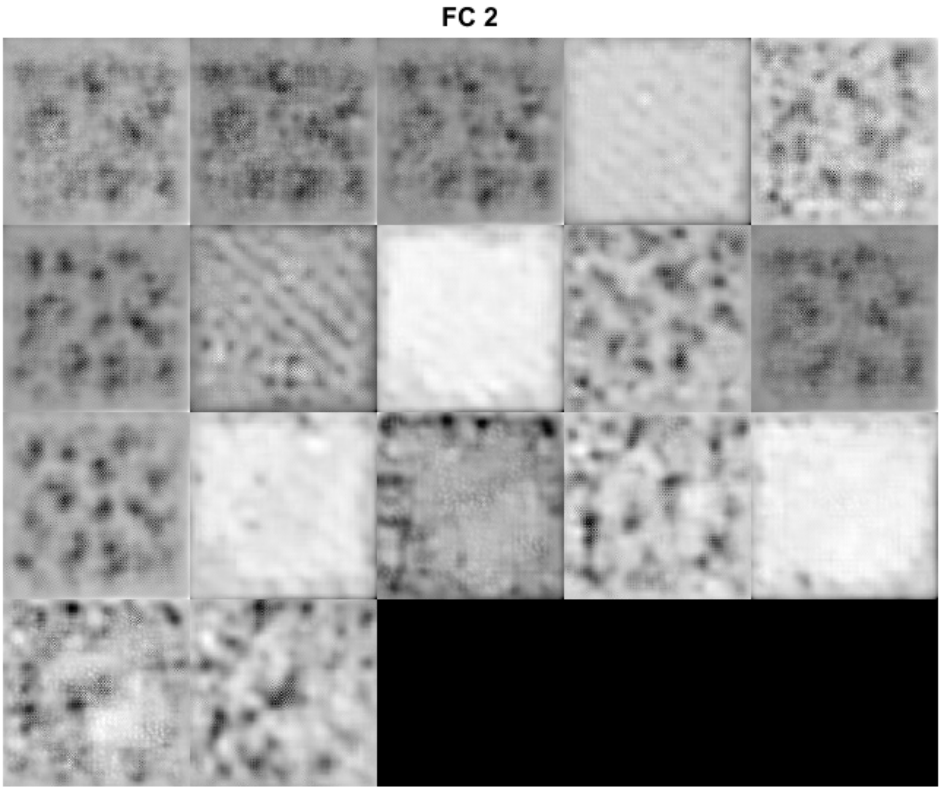


Figure 82: 17 neurons of FC 2

3.5.3 Wide Network Architecture - 15 epochs, learning Rate = 1×10^{-3}

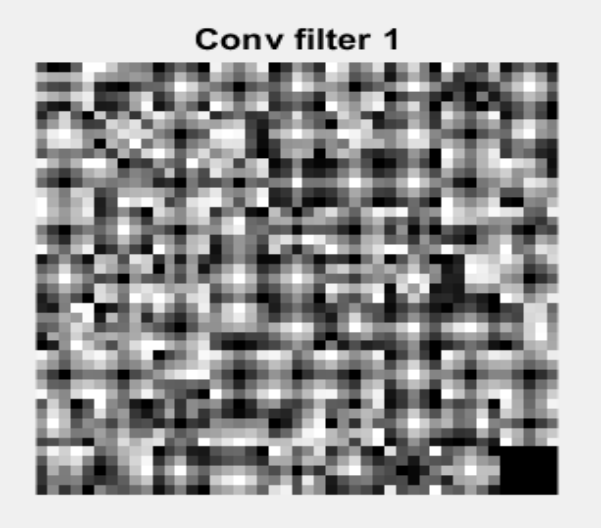


Figure 83: 80 filters of CONV 1

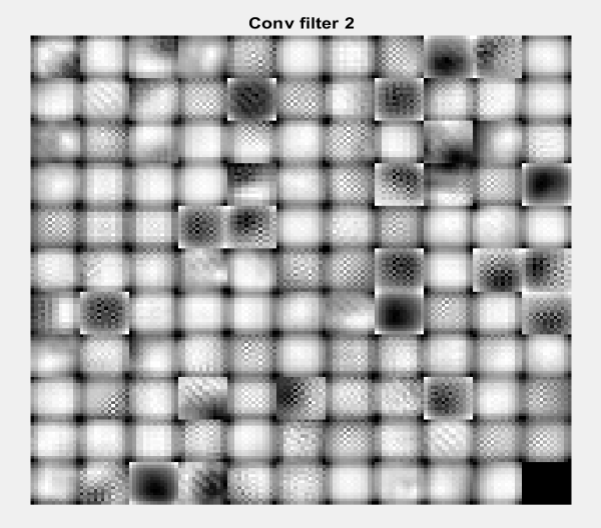


Figure 84: 120 filters of CONV 2

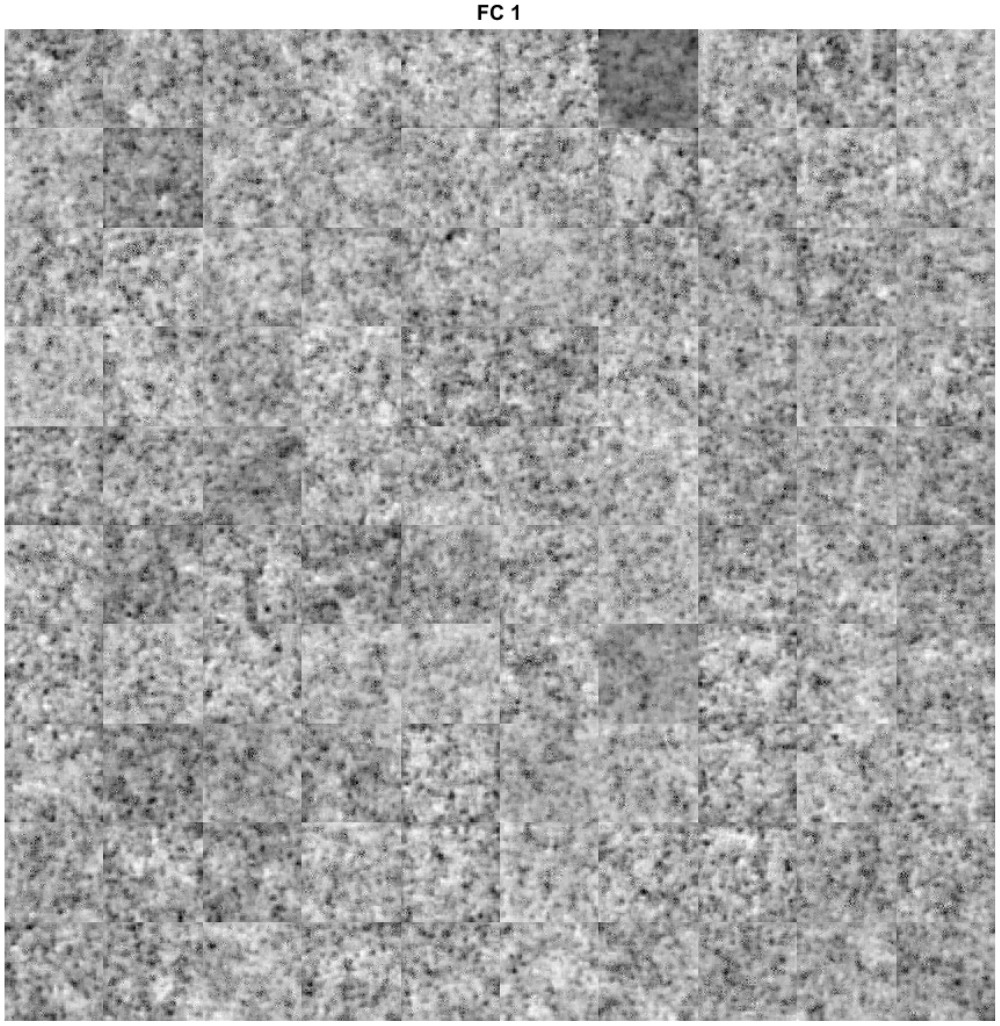


Figure 85: 100 neurons of FC 1

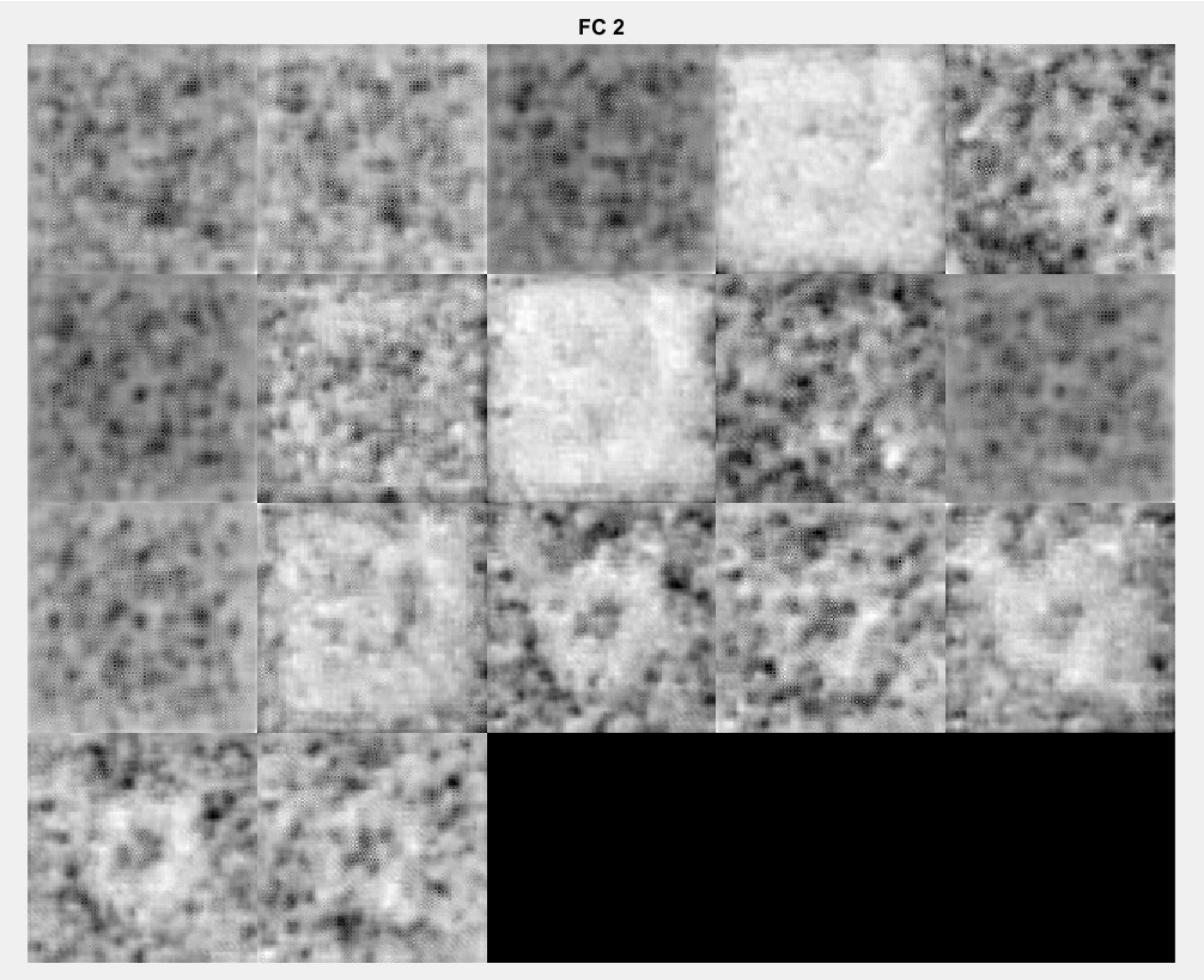


Figure 86: 17 neurons of FC 2

3.6 Pre-processing using t-SNE

The feature reduction algorithm t-SNE is applied on the fully connected layer ('FC 2') activations of Skinny and Wide Network Architectures. Figures 87 and 88 show the results of t-SNE for training activations of Skinny and Wide Networks. Due to lack of time, the plots for validation and test sets could not be generated.

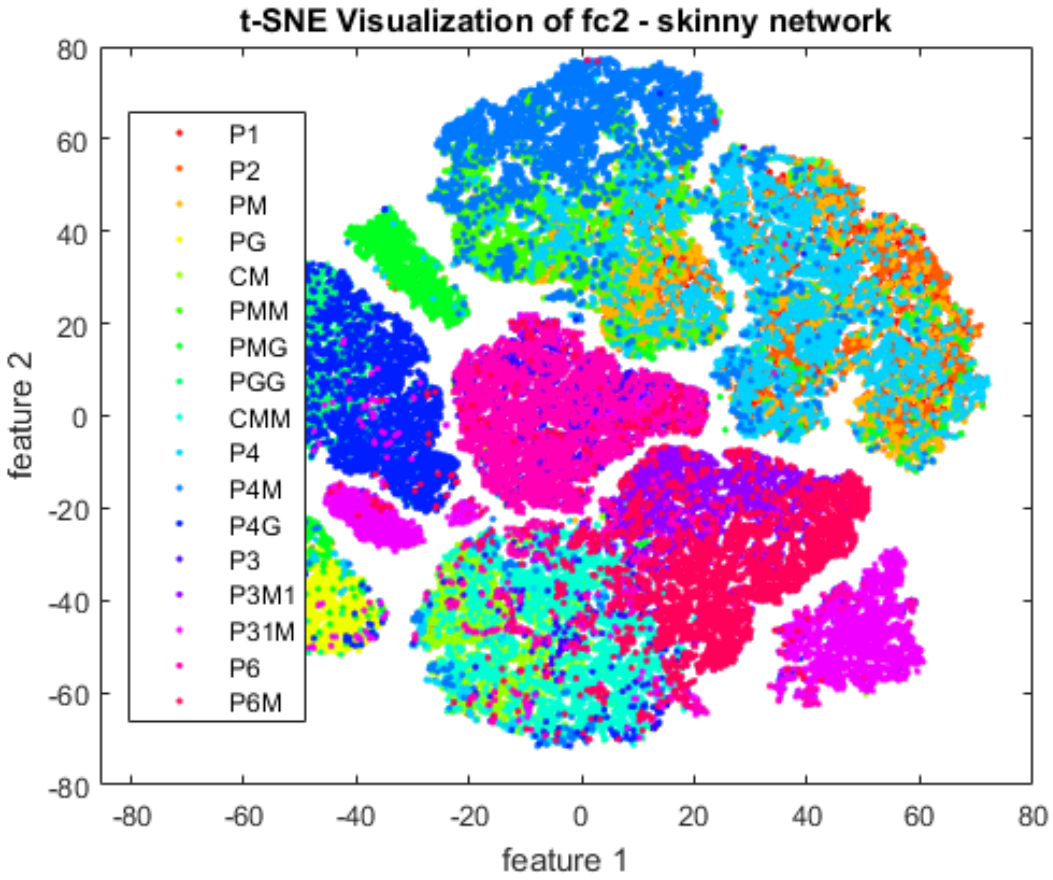


Figure 87: t-SNE applied on Skinny Network

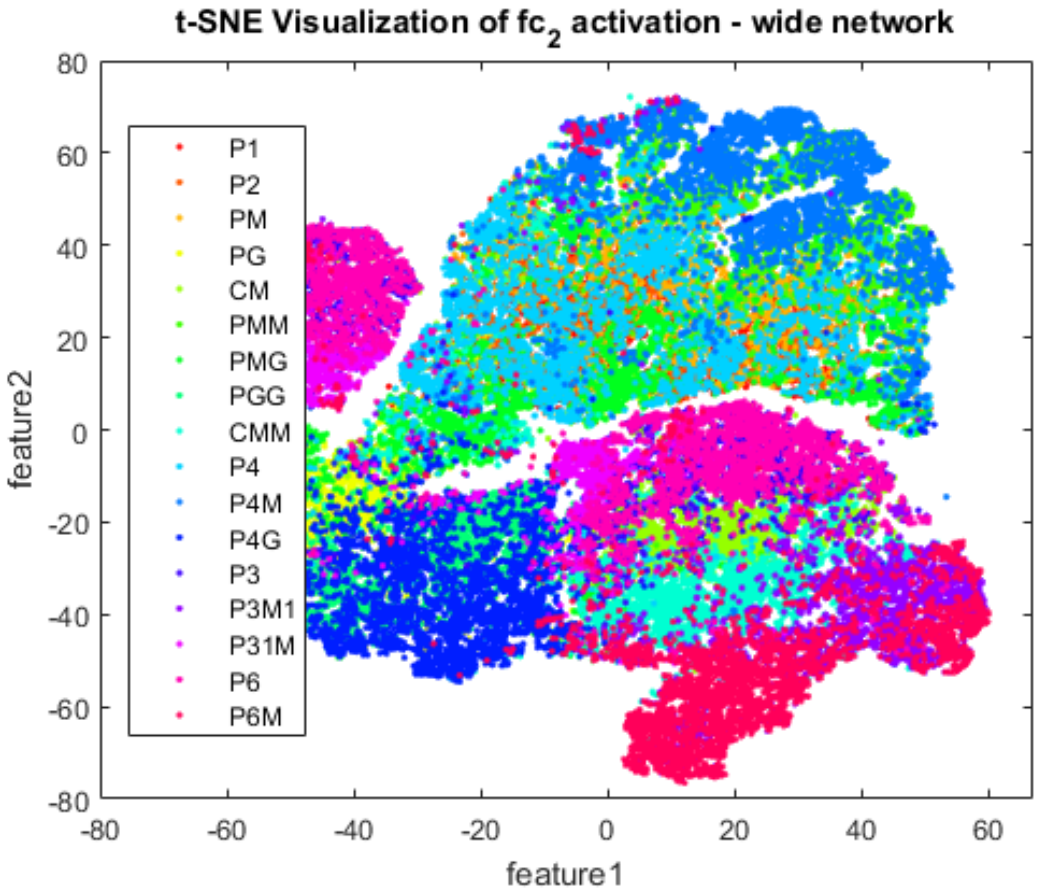


Figure 88: t-SNE applied on Wide Network

3.7 Transfer Learning using Alex Net

In this section, a pre-trained model of Alex Net is used to achieve classification on augmented data. This model is compared with the designed Skinny and Wide deep architectures.
Alex Net is a Convolutional Neural Network with 25 layer architecture. For training, the last fully connected layer of Alex Net, with 1000 output neurons has been chopped off, and a 17 output layer fully connected layer has been replaced in its position. Since Alex net has already been trained on ImageNet classification before, the weights of the Network are oriented such that the layers till the Fully Connected layer are capable of extracting features from a given image. This insight has been used to perform transfer learning on the augmented wallpaper dataset. The parameters used are:
Number of epochs: 100
Learning Rate:5×10^{-4}
Since the weights associated with the convolution layers are responsible for feature extraction, it is made sure that the orientation of these weights is not altered by a large amount by increasing the weight learning rate and bias learning rate factors of the new fully connected layer to 20. This will ensure a gradient/biased learning, where the weights of the fully connected layer are learnt 20 times faster than other layers.
The results of Alex net on augmented data set are as follows:
Training Accuracy = 90.73%
Test Accuracy = 83.64%
Validation Accuracy = 86.94%
Training Time = 3977.97 seconds

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	3499	551	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	3596	454	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	3707	343	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	3591	459	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	3625	425	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	3981	69	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	4050	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	15	3824	211	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	3272	778	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	3695	355	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	3263	787	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	3444	606	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	3853	197	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	4050	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	546	3504	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	301	3749	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	659	3391

Figure 89: Confusion Matrix for Train Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	0.863951	0.136049	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	0.887901	0.112099	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	0.915309	0.084691	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	0.886667	0.113333	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	0.895062	0.104938	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	0.982963	0.017037	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0.003704	0.944198	0.052099	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	0.807901	0.192099	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	0.912346	0.087654	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	0.805679	0.194321	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	0.85037	0.14963	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	0.951358	0.048642	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.134815	0.865185	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.074321	0.925679	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.162716	0.837284

Figure 90: Classification Matrix for Train Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	834	166	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	923	77	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	946	54	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	954	46	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	110	890	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	161	722	117	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	805	195	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	800	200	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	712	288	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	875	125	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	984	16	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	352	648	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	212	788	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	262	738

Figure 91: Confusion Matrix for Test Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	0.834	0.166	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	0.923	0.077	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	0.946	0.054	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	0.954	0.046	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0.11	0.89	0	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0.161	0.722	0.117	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	0.805	0.195	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	0.8	0.2	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	0.712	0.288	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	0.875	0.125	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	0.984	0.016	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.352	0.648	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.212	0.788	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.262	0.738

Figure 92: Classification Matrix for Test Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	357	93	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	390	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	397	53	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	385	65	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	389	61	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	427	23	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	431	19	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0	395	55	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	326	124	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	394	56	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	338	112	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	369	81	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	423	27	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	450	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	59	391	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	427	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88	362

Figure 93: Confusion Matrix for Validation Data set

		PREDICTED LABELS																
		P1	P2	PM	PG	CM	PMM	PMG	PGG	CMM	P4	P4M	P4G	P3	P3M1	P31M	P6	P6M
ACTUAL LABELS	P1	0.793333	0.206667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	P2	0	0.866667	0.133333	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PM	0	0	0.882222	0.117778	0	0	0	0	0	0	0	0	0	0	0	0	0
	PG	0	0	0	0.855556	0.144444	0	0	0	0	0	0	0	0	0	0	0	0
	CM	0	0	0	0	0.864444	0.135556	0	0	0	0	0	0	0	0	0	0	0
	PMM	0	0	0	0	0	0.948889	0.051111	0	0	0	0	0	0	0	0	0	0
	PMG	0	0	0	0	0	0	0.957778	0.042222	0	0	0	0	0	0	0	0	0
	PGG	0	0	0	0	0	0	0	0.877778	0.122222	0	0	0	0	0	0	0	0
	CMM	0	0	0	0	0	0	0	0	0.724444	0.275556	0	0	0	0	0	0	0
	P4	0	0	0	0	0	0	0	0	0	0.875556	0.124444	0	0	0	0	0	0
	P4M	0	0	0	0	0	0	0	0	0	0	0.751111	0.248889	0	0	0	0	0
	P4G	0	0	0	0	0	0	0	0	0	0	0	0.82	0.18	0	0	0	0
	P3	0	0	0	0	0	0	0	0	0	0	0	0	0.94	0.06	0	0	0
	P3M1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	P31M	0	0	0	0	0	0	0	0	0	0	0	0	0	0.131111	0.868889	0	0
	P6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.051111	0.948889	0
	P6M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.195556	0.804444

Figure 94: Classification Matrix for Validation Data set

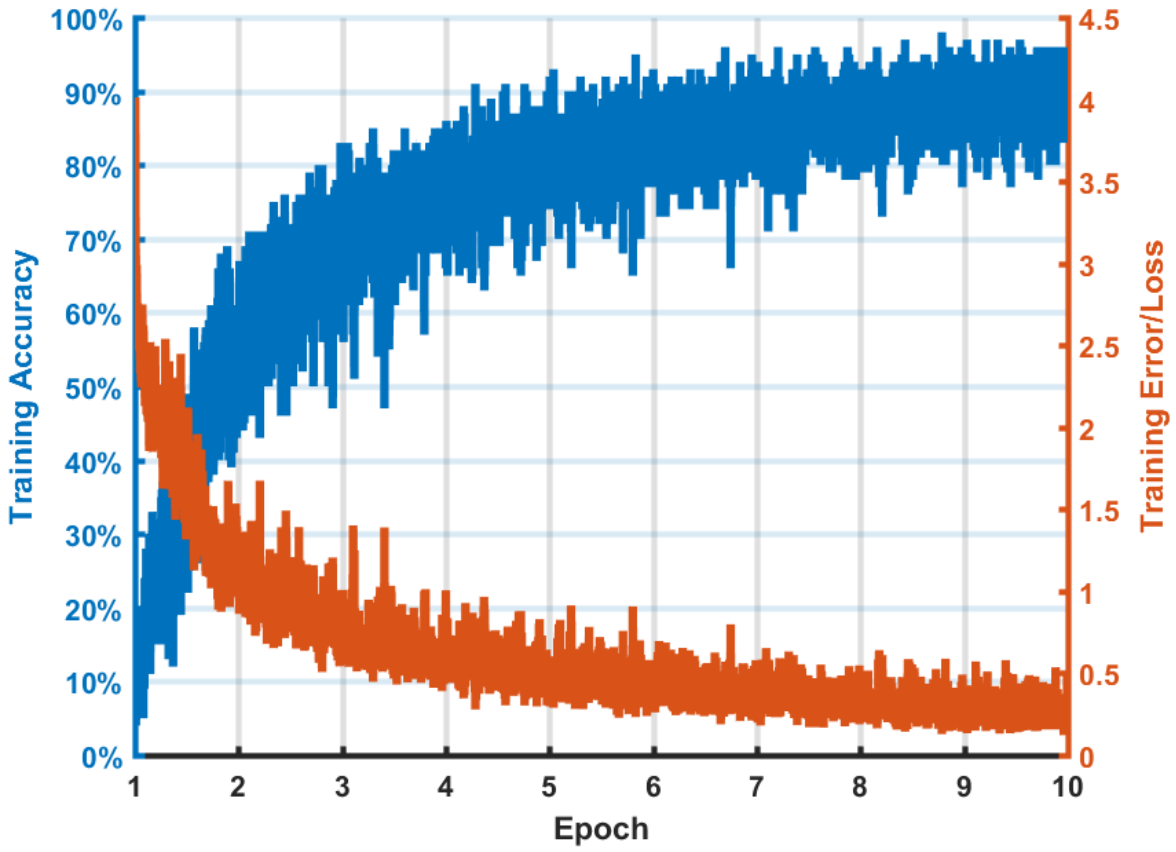


Figure 95: Error and Accuracy Plot

4 Conclusion

1. CNNs are generally used in applications of visula imagery due to their capability of extracting features automatically, witjout the need for any pre-processing of image data.

2. Performance efficiency of CNNs can be usually improved by making the netoworks deeper, i.e., increasing the number of hidden layers. But, thre is always a trade-off, since the number of parameters to be computed also increases.

3. A general problem associated with increasing the depth of the networks is **Vanishing Gradients**. The gradients that are back propagated through the network become very small as the dependency of farther layers from initial layers is very less. This will result in decrease in training efficiency.

4. CNNS require large amount of data for training. Labelling of data is computationally expensive. Also, image data available is usually sparse and lesser than the demand. Hence, data **augmentation** is a useful tool to generated labelled data, which will improve training efficiency and avoids the problem of **over-fitting.**

5. Initialization of weights is very important. Improper initialization can aid in the loos function from never converging.

6. Network in the starter code performed quite well on the original dataset, but performed very poorly on the augmented data set.

7. With the decrease in learning rate, the training accuracy increases, but the error also increases. Hence, **decease in error is not directly related to increase in accuracy, or vice-versa.**. This is because, the accuracy is calculated based on mini-batches of images. Hence, even if the error decreases, the accuracy of the mini-batch might not increase. Hence, these two are not necessarily directly related.

8. Deeper architectures - Skinny and Wide networks perform better than the baseline model on the augmented data set. However, Skinny network performs better than wide network, as wide network is over-fitting the data. Hence, it can be concluded that more number of layers with sufficiently more number of filters performs better than a network with very large number of filters in a layer.

9. t-SNE is a good pre-processing tool, which will increase classification efficiency.

10. Filter visualizations help in comprehending what each neuron in the network is getting excited about.

11. Transfer learning is a effective tool in increasing the efficiency and reducing the time taken for training.

References

[1] Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Springer, 2006

[2] http://cs231n.github.io/convolutional-networks/

[3] https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/

[4] EE552 Lecture Slides

[5] MATLAB Documentations