Project 3: Feature Selection

Saniya Naphade

April 6, 2020

Abstract

The objective of this project is to train convolutional neural networks and transfer learning and to analyze the testing accuracies of the implemented networks.

Contents

1 Introduction

Neural networks are a group of algorithms, modeled loosely after the human brain, designed to recognize patterns. They interpret sensory data through a form of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or statistic, must be translated.
Deep learning is the name we use for "stacked neural networks"; that is, networks composed of several layers. Deep learning maps inputs to outputs. It finds correlations. it's referred to as a "universal approximator", because it can learn to approximate an unknown function f(x) = y between any input x and any output y, assuming they're related in the slightest degree (by correlation or causation, for example). within the process of learning, a neural network finds the proper f, or the proper manner of remodeling x into y.

The layers are made of nodes. A node is just a place where computation happens, loosely patterned on a neuron in the human brain, which fires when it encounters sufficient stimuli. A node combines input from the data with a set of coefficients, or weights, that either amplify or dampen that input, thereby assigning significance to inputs with regard to the task the algorithm is trying to learn; e.g. which input is most helpful is classifying data without error? These input-weight products are summed and then the sum is passed through a node's so-called activation function, to determine whether and to what extent that signal should progress further through the network to affect the ultimate outcome, say, an act of classification. If the signals passes through, the neuron has been "activated."
Deep-learning networks are distinguished from the more commonplace single-hidden-layer neural networks by their depth; that is, the number of node layers through which data must pass in a multistep process of pattern recognition.
In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer's output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer.

2 Data

The feature selection module was implemented for two datsets, namely:
1. WALLPAPER

2.1 WALLPAPER

A wallpaper group (or plane symmetry group or plane crystallographic group) is a mathematical classification of a two-dimensional repetitive pattern, based on the symmetries in the pattern. Such patterns occur frequently in architecture and decorative art, especially in textiles and tiles as well as wallpaper. There are a total of 17,000 training observations with 100 observations per class, giving the total number of classes to be 17. The images are gray-scale images with dimension of 256x256.

2.2 Extra Credit

2.2.1 MOCAP

This dataset consisted of the Motion Capture data of 10 subjects performing taiji. 12 joint positions were taken into consideration when the data was made. Making 12 classes for the data to be classified.

3 Augmentation

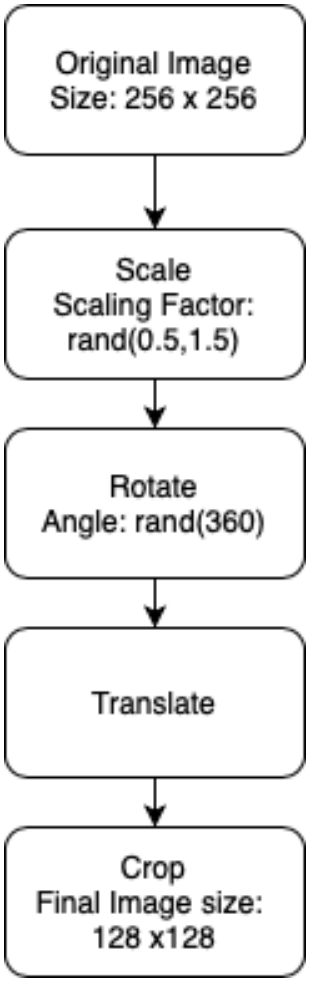


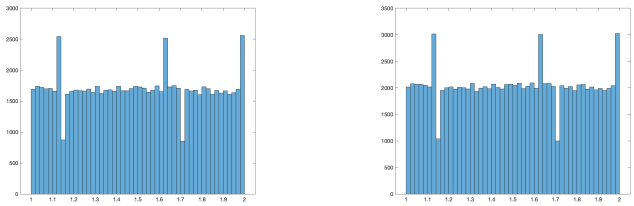
Figure 1: Histogram depicting the top 10% of the features selected via feature selection algorithm.

The given data was augmented by rotating, scaling and tranlating the original image. This section describes the various convolutional neural networks and their parameters that were used to train on the 85,000 augmented wallpaper images. The image dimensions that were used for training are 128 x 128. The following subsections give a detailed overview of the networks that were used to train the augmented dataset. As seen from

the algorithm, Variance ratio is implemented for Filter and forward selection method is used for feature selection.

3.1 Scaling

The original image is first scaled by a scaling factor between 50-150%. The command used for this was the **imresize**. Figure2 shows the histogram of various values considered in building the augmented dataset.

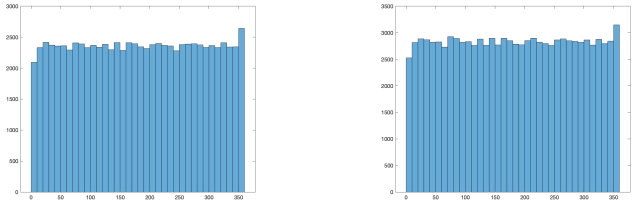


(a) Train Dataset (b) Test Data set

Figure 2: Histogram depicting the distribution of the various scaling factors values considered for building the augmented test dataset

3.2 Rotation

The scaled image taken was rotated freely i.e, in any angle between 0 to 360° using the **imrotate** command in matlab. The rotated image was cropped to the size of the original image in order to avoid the presence of the black corners of the rotated image. Figure3 shows the histogram of various values considered in building the augmented dataset.



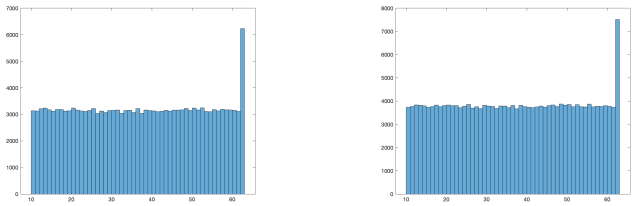
(a) Train Dataset (b) Test Data set

Figure 3: Histogram depicting the distribution of the various rotation angle values considered for building the augmented datasets

3.3 Translation

In order to have the proper translation, so that the pattern still remained in within a size of 128x128, we first found out the dimensions of the maximum sized rectangle that contains the pixels that could be placed in the rotated and resized image. Then we translated the image within the range of the size of the maximum size rectangle. This ensures that the translated image generated would be such that it would still contain the

pixels.
The image is translated using the **imtranslate** function.
The translation was given between 10 to 63 pixels from either side. The upper bound was 63 so that the final image would be of the size 128x128. Figure4 shows the histogram of various values considered in building the augmented dataset.

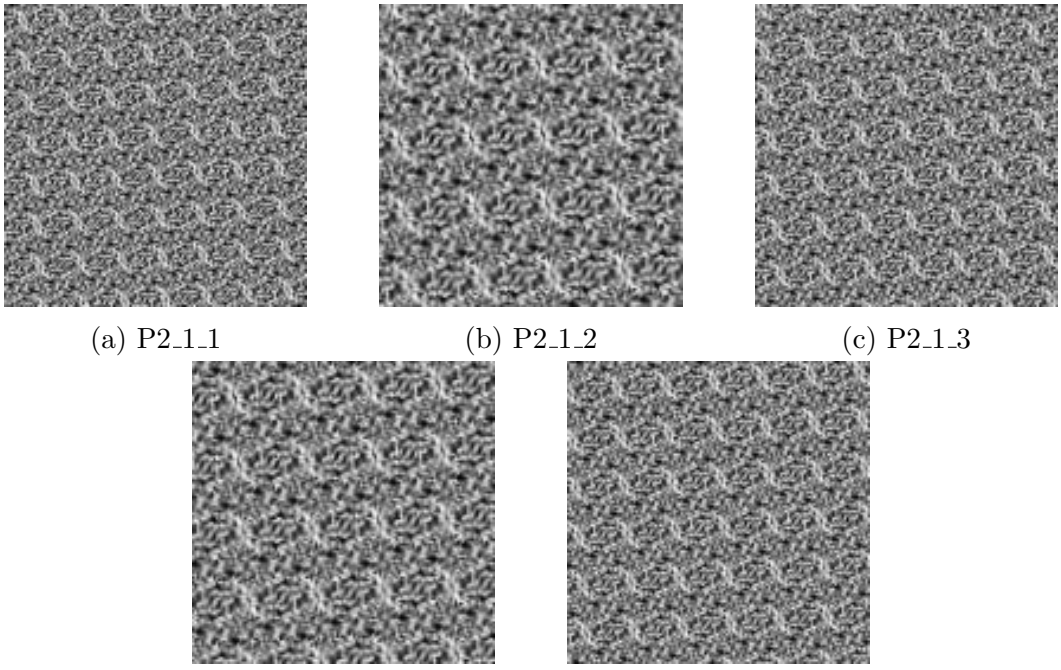


(a) Train Dataset (b) Test Data set

Figure 4: Histogram depicting the distribution of the various translation factors values considered for building the augmented test dataset

3.4 Cropping

The final scaled, rotated and translated image is then cropped to 128x128 size using the **imcrop**. The figure 5 shows 5 augmented images of the first image belonging to class P2



(a) P2_1_1 (b) P2_1_2 (c) P2_1_3

(d) P2_1_4 (e) P2_1_5

Figure 5: 2 Figures side by side

4 Networks

Layer command description:
imageInputLayer: It defines an image input layer for the network.

convolution2dLayer: Defines 2D convolution layer for Convolutional Neural Networks. filterSize specifies the height and width of the filters. It can be a scalar, in which case the filters will have the same height and width, or a vector [h w] where h specifies the height for the filters, and w specifies the width. numFilters specifies the number of filters.

BatchNormalizationLayer: Creates a batch normalization layer. This type of layer normalizes each channel across a mini-batch. This can be useful in reducing sensitivity to variations within the data.

ReLULayer: Creates a rectified linear unit layer. This type of layer performs a simple threshold operation, where any input value less than zero will be set to zero.

MaxPooling2DLayer: A max pooling layer divides the input into rectangular pooling regions, and outputs the maximum of each region. poolSize specifies the width and height of a pooling region. It can be a scalar, in which case the pooling regions will have the same width and height, or a vector [h w] where h specifies the height and w specifies the width. Note that if the 'Stride' dimensions are less than the respective pool dimensions, then the pooling regions will overlap.

fullyConnectedLayer: Creates a fully connected layer. outputSize specifies the size of the output for the layer. A fully connected layer will multiply the input by a matrix and then add a bias vector.

SoftmaxLayer: Creates a softmax layer. This layer is useful for classification problems.

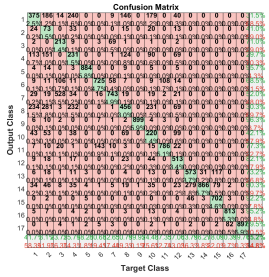
5 Training

5.1 Training Original Data on Default Network

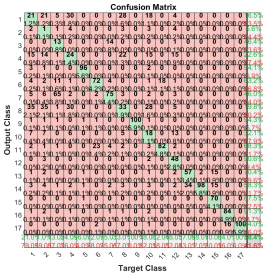
5.1.1 Learning Rate: 5.00E-5

Parameter	Value
Batch size	250
Number of Epochs	5
Learning Rate	5.00E-05
Training Accuracy(%)	65
Validation Accuracy(%)	58
Testing Accuracy(%)	61
Training Time(seconds)	82.45
Optimizer	sgdm

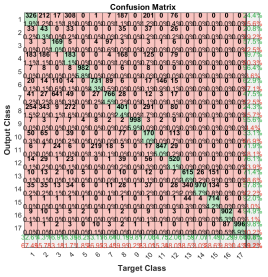
Table 1: Default Networks Parameter values



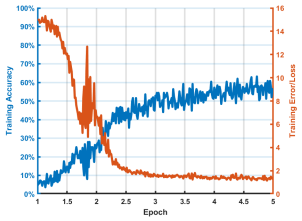
(a) Confusion matrix for Training



(b) Confusion matrix for Validation



(c) Confusion matrix for Testing



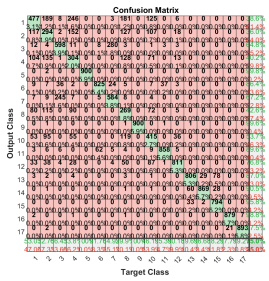
(d) Training accuracies and Training Error/Loss per Epoch

Figure 6: Results for Default Network with Learning rate of 5e-5

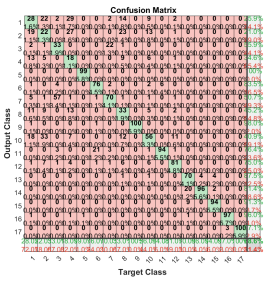
5.1.2 Learning Rate: 1.00E-5

Parameter	Value
Batch size	250
Number of Epochs	5
Learning Rate	1.00E-05
Training Accuracy(%)	74
Validation Accuracy(%)	69
Testing Accuracy(%)	68
Training Time(seconds)	81.64
Optimizer	sgdm

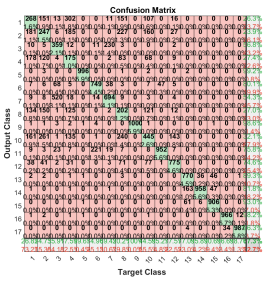
Table 2: Default Networks Parameter values



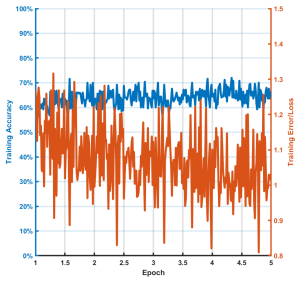
(a) Confusion matrix for Training



(b) Confusion matrix for Validation



(c) Confusion matrix for Testing



(d) Training accuracies and Training Error/Loss per Epoch

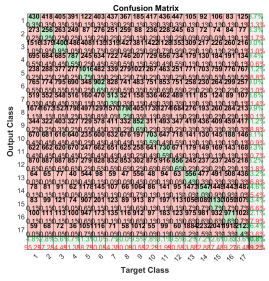
Figure 7: Results for Default Network with Learning rate of 1e-5

5.2 Training Augmented Data on Default Network

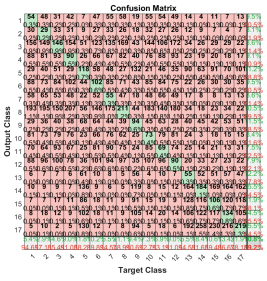
5.2.1 Learning Rate: 5.00E-5

Parameter	Value
Batch size	250
Number of Epochs	5
Learning Rate	5.00E-04
Training Accuracy(%)	11
Validation Accuracy(%)	11
Testing Accuracy(%)	11
Training Time(seconds)	381.21
Optimizer	sgdm

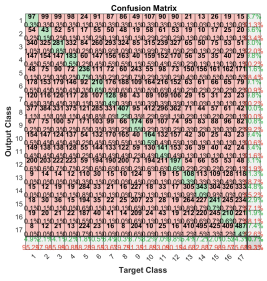
Table 3: Default Networks Parameter values



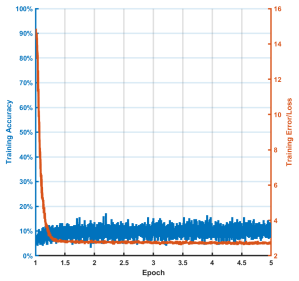
(a) Confusion matrix for Training



(b) Confusion matrix for Validation



(c) Confusion matrix for Testing



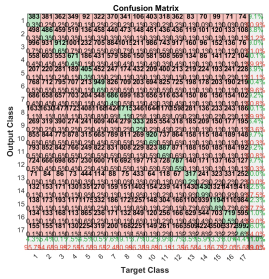
(d) Training accuracies and Training Error/Loss per Epoch

Figure 8: Results for Default Network with Learning rate of 5e-5

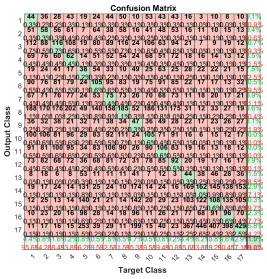
5.2.2 Learning Rate: 1.00E-5

Parameter	Value
Batch size	250
Number of Epochs	5
Learning Rate	1.00E-05
Training Accuracy(%)	11
Validation Accuracy(%)	11
Testing Accuracy(%)	10
Training Time(seconds)	263.83
Optimizer	sgdm

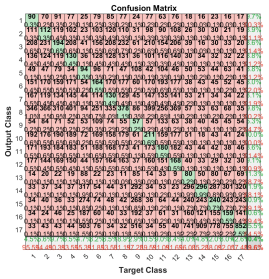
Table 4: Default Networks Parameter values



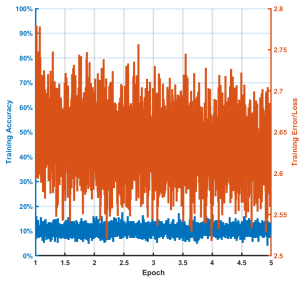
(a) Confusion matrix for Training



(b) Confusion matrix for Validation



(c) Confusion matrix for Testing



(d) Training accuracies and Training Error/Loss per Epoch

Figure 9: Results for Default Network with Learning rate of 1e-5

5.3 Training Augmented Data on Wide Network

$INPUT- > [CONV- > RELU- > MAXPOOL] * 2- > FC- > DROPOUT- > FC- > SOFTMAX$

The augmented dataset was trained on a wide cnn network with model layer parameters as shown in Figure 10.Wide networks comprise of higher number of filters per convolutional layer. The wide network was trained was trained with the learning rate of 1e-5. The results obtained for training the network for 20 epochs with batchsize of 100 are listed in Table 5.

Parameter	Value
Batch size	100
Number of Epochs	20
Learning Rate	1.00E-05
Training Accuracy(%)	63
Validation Accuracy(%)	56
Testing Accuracy(%)	54
Training Time(seconds)	16036.54
Optimizer	sgdm

Table 5: Wide Neural Networks Parameter values

Parameters for each layer are. in the table below:

Layer	Activations	Parameters	Values
imageInputLayer	128,128,1	inputSize	[128,128,1]
convolution2dLayer	124x124x100	filterSize	9
		numFilters	100
		Stride	[1,1]
		PaddingSize	[2,2,2,2]
BatchNormalizationLayer	124x124x100	Channels	100
ReLULayer	124x124x100		
MaxPooling2DLayer	124x124x100	PoolSize	[2,2]
		Stride	[2,2]
Convolution2dLayer	29x29x90	filterSize	9
		numFilters	90
		Stride	[2,2]
		PaddingSize	[2,2,2,2]
BatchNormalizationLayer	29x29x90		
ReLULayer	29x29x90		
MaxPooling2DLayer	14x14x90	PoolSize	[2,2]
		Stride	[2,2]
fullyConnectedLayer	1x1x300	OutputSize	300
DropoutLayer	1x1x300	Probability	0.25
fullyConnectedLayer	1x1x17	OutputSize	17
SoftmaxLayer	1x1x17		
ClassificationOutputLayer			

Figure 10: Wide Network

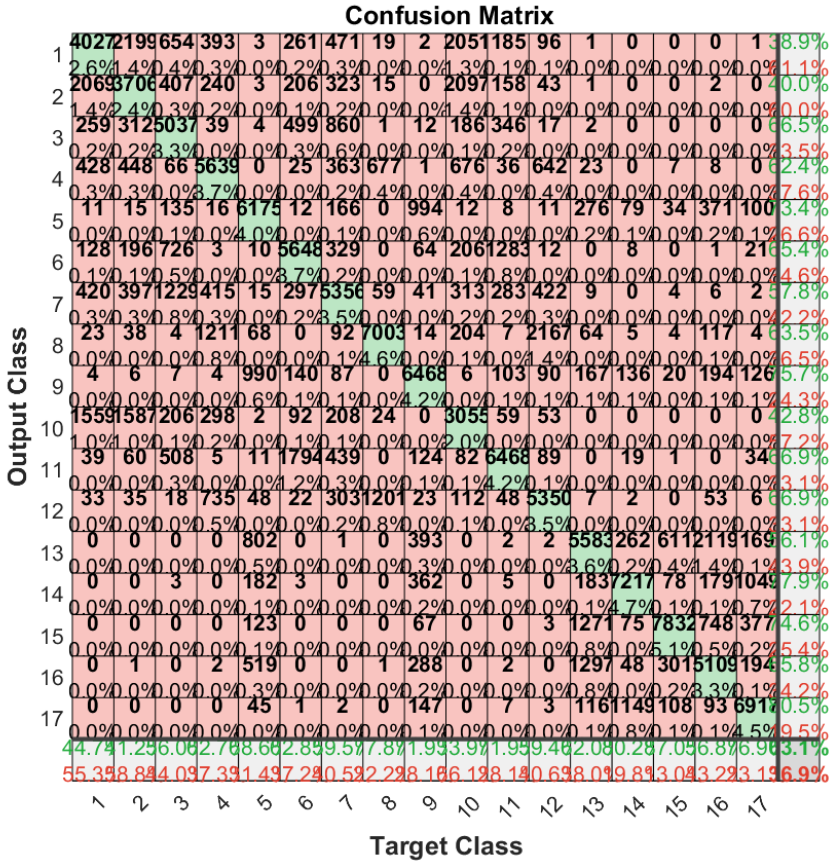


Figure 11: Confusion matrix for Training on Wide Neural network with 1e-5 learning rate

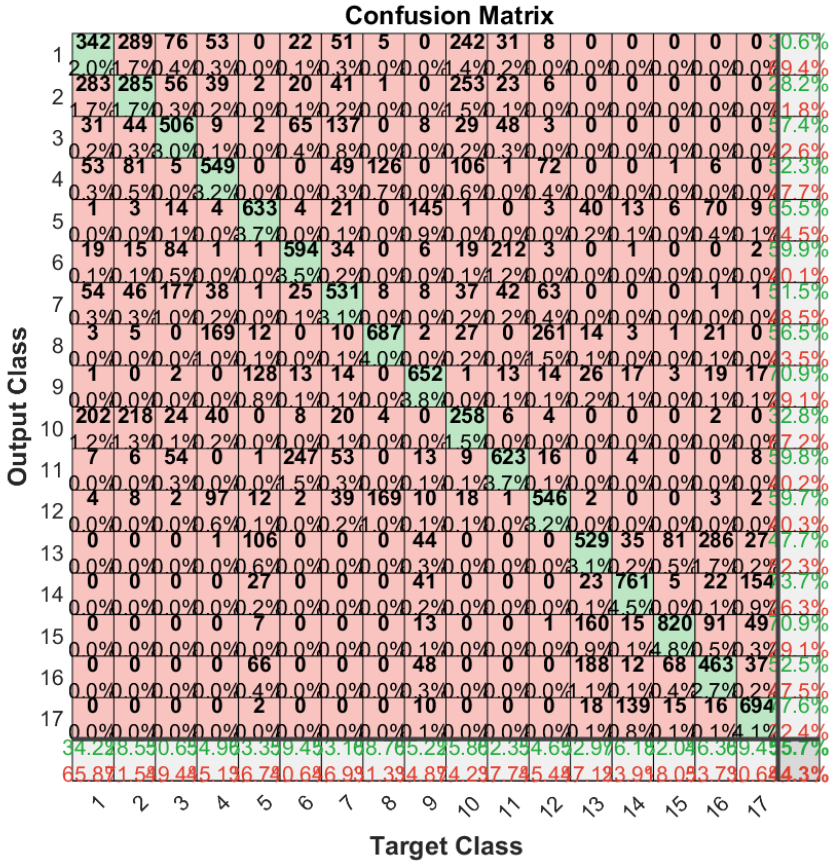


Figure 12: Confusion matrix for Validation on Wide Neural network with 1e-5 learning rate

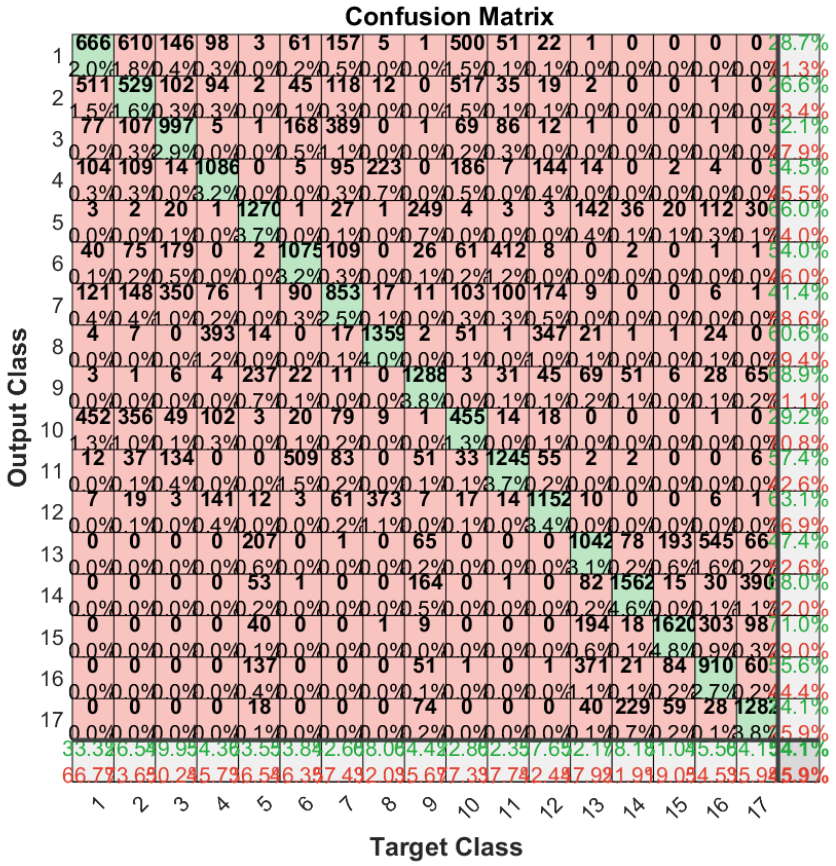


Figure 13: Confusion matrix for Testing on Wide Neural network with 1e-5 learning rate

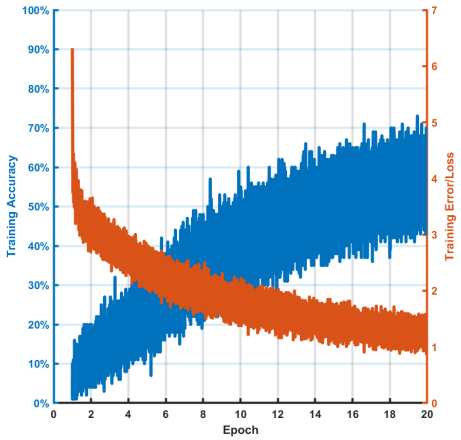


Figure 14: Training accuracies and Training Error/Loss obtained for each epoch

5.4 Skinny Network

A skinny network has fewer layers but many filters.

$INPUT- > [CONV- > RELU] * 2- > [CONV- > RELU- > MAXPOOL] * 2- > FC- > DROPOUT- > FC- > SOFTMAX$

The augmented dataset was trained on a wide cnn network with model layer parameters as shown in Figure 15.Wide networks comprise of higher number of filters per convolutional layer. The wide network was trained was trained with the learning rate of 1e-5. The results obtained for training the network for 20 epochs with batchsize of 100 are listed in Table 9. Parameters for each layer are. in the table below:

Parameter	Value
Batch size	100
Number of Epochs	20
Learning Rate	1.00E-05
Training Accuracy(%)	86
Validation Accuracy(%)	81
Testing Accuracy(%)	79
Training Time(seconds)	10867.54
Optimizer	sgdm

Table 6: Skinny Neural Networks Parameter values

Layer	Activations	Parameters	Values
imageInputLayer	128,128,1	inputSize	[128,128,1]
convolution2dLayer	126x126x80	filterSize	7
		numFilters	80
		Stride	[1,1]
		PaddingSize	[2,2,2,2]
BatchNormalizationLayer	126x126x80	Channels	100
ReLULayer	126x126x80		
Convolution2dLayer	62x62x60	filterSize	7
		numFilters	60
		Stride	[2,2]
		PaddingSize	[2,2,2,2]
BatchNormalizationLayer	62x62x60		
ReLULayer	62x62x60		
convolution2dLayer	30x30x40	filterSize	7
		numFilters	40
		Stride	[2,2]
		PaddingSize	[2,2,2,2]
BatchNormalizationLayer	30x30x40		
ReLULayer	30x30x40		
MaxPooling2DLayer	15x15x40	PoolSize	[2,2]
		Stride	[2,2]
convolution2dLayer	7x7x32	filterSize	7
		numFilters	32
		Stride	[2,2]
		PaddingSize	[2,2,2,2]
BatchNormalizationLayer	7x7x32		
ReLULayer	7x7x32		
MaxPooling2DLayer	3x3x32	PoolSize	[2,2]
		Stride	[2,2]
fullyConnectedLayer	1x1x300	OutputSize	300
DropoutLayer	1x1x300	Probability	0.25
fullyConnectedLayer	1x1x17	OutputSize	17
SoftmaxLayer	1x1x17		
ClassificationOutputLayer			

Figure 15: Skinny Network

5.4.1 Figures for Skinny NN

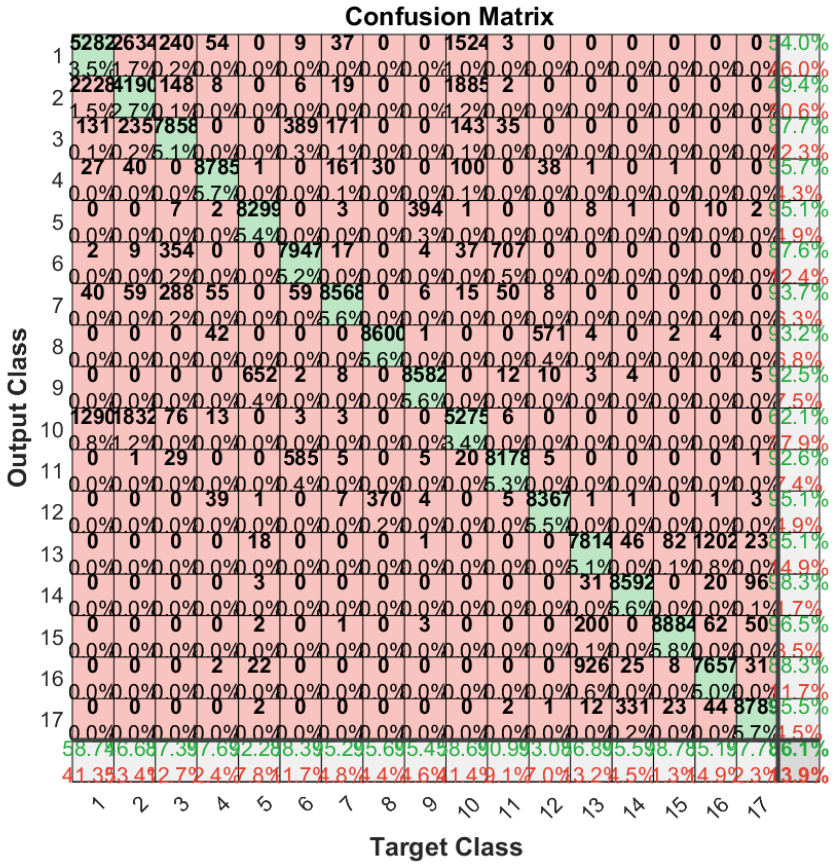


Figure 16: Confusion matrix for Training on Skinny Neural network with 1e-5 learning rate

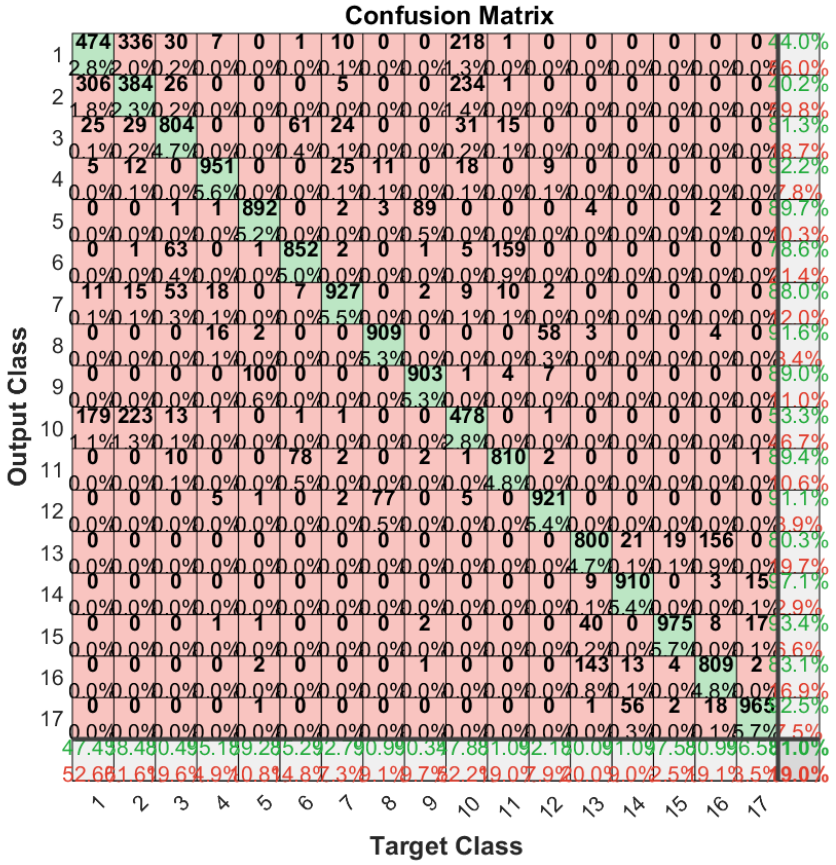


Figure 17: Confusion matrix for Validation on Skinny Neural network with 1e-5 learning rate

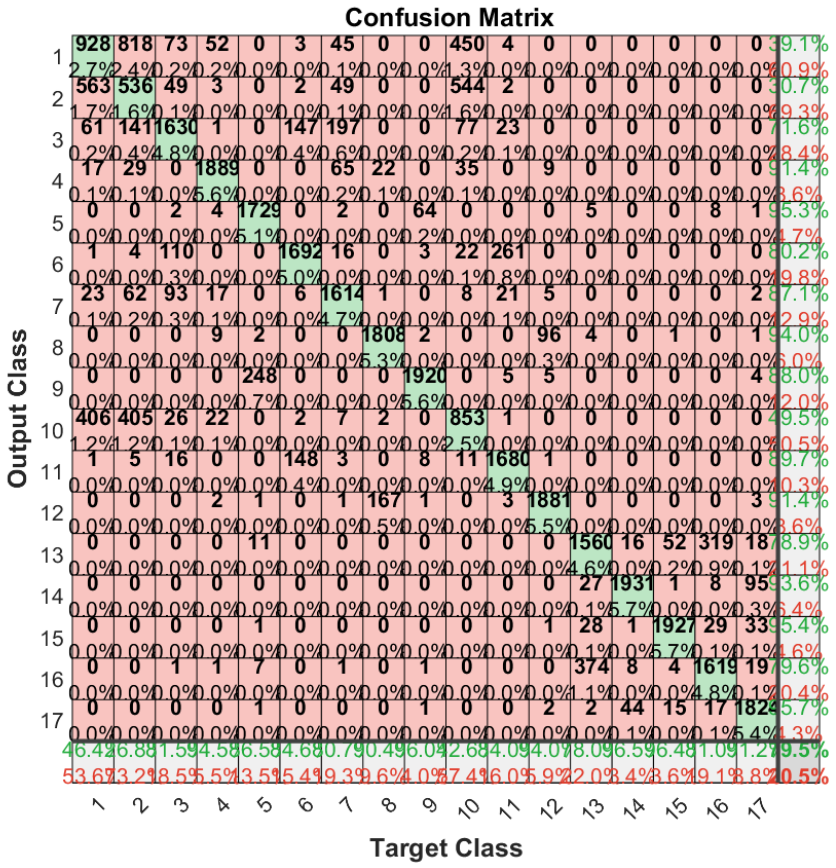


Figure 18: Confusion matrix for Testing on Skinny Neural network with 1e-5 learning rate

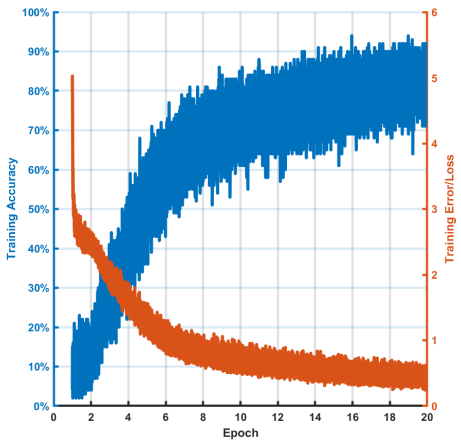


Figure 19: Training accuracies obtained for each epoch

5.5 Transfer Learning on Original Data set Using Skinny Net

The original dataset was also trained on the Skinny network using transfer learning. The parameters obtained and set for the new network are given in Table 7

Parameter	Value
Batch size	200
Number of Epochs	10
Learning Rate	1.00E-04
Training Accuracy(%)	100
Validation Accuracy(%)	98
Testing Accuracy(%)	97
Training Time(seconds)	822.93
Optimizer	sgdm

Table 7: Skinny Neural Networks Parameter with Original Dataset values

5.5.1 Figures for Original Data on Skinny NN

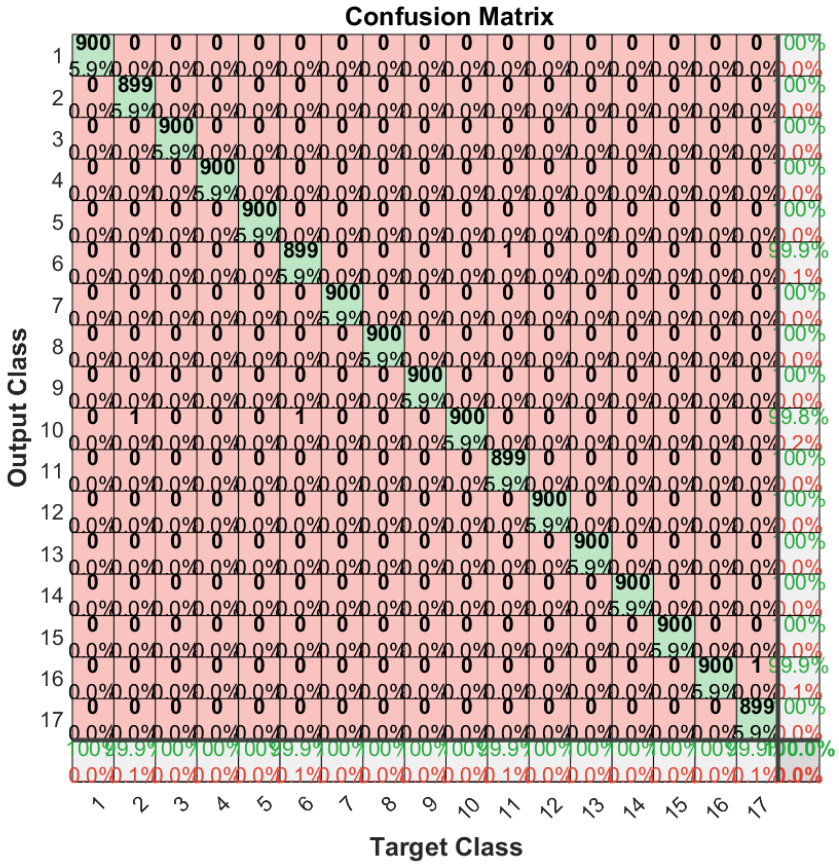


Figure 20: Confusion matrix for Training Original Data on Skinny Neural network with 1e-5 learning rate

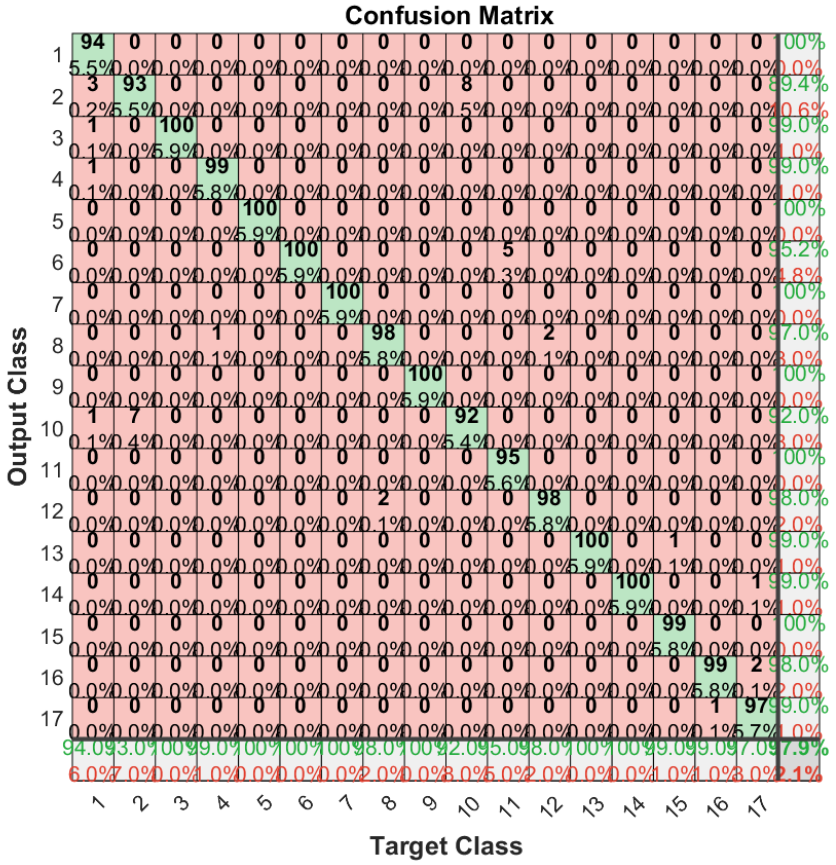


Figure 21: Confusion matrix for Validation on Skinny Neural network with 1e-5 learning rate

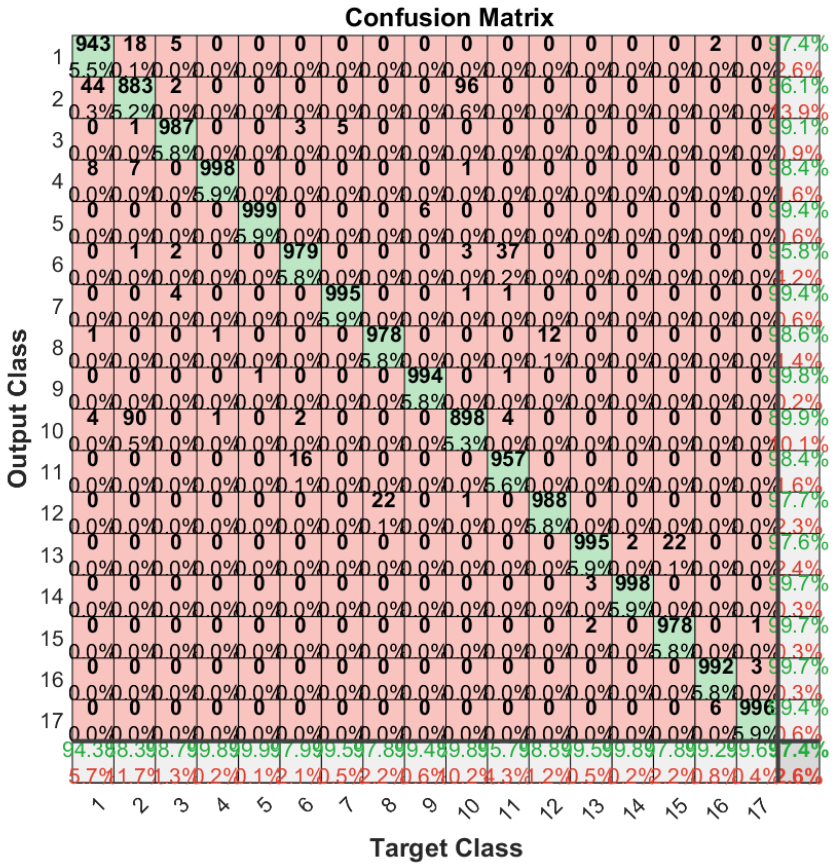


Figure 22: Confusion matrix for Testing on Skinny Neural network with 1e-5 learning rate

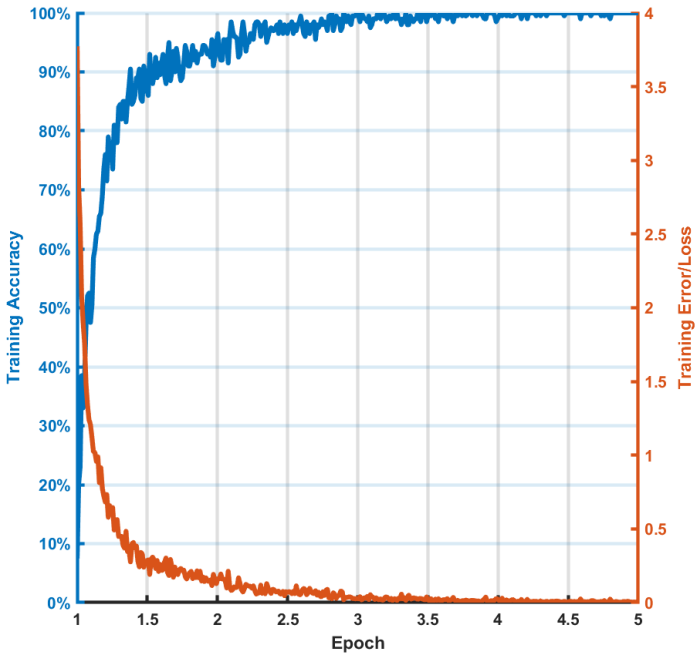


Figure 23: Training accuracies obtained for each epoch

5.6 Transfer Learning on Original Data set Using Wide Net

The original dataset was also trained on the Wide network using transfer learning. The parameters obtained and set for the new network are given in Table 8

Parameter	Value
Batch size	200
Number of Epochs	10
Learning Rate	1.00E-04
Training Accuracy(%)	100
Validation Accuracy(%)	98
Testing Accuracy(%)	98
Training Time(seconds)	1037.89
Optimizer	sgdm

Table 8: Wide Neural Networks Parameter with Original Dataset values

5.6.1 Figures for Original Data on Wide NN

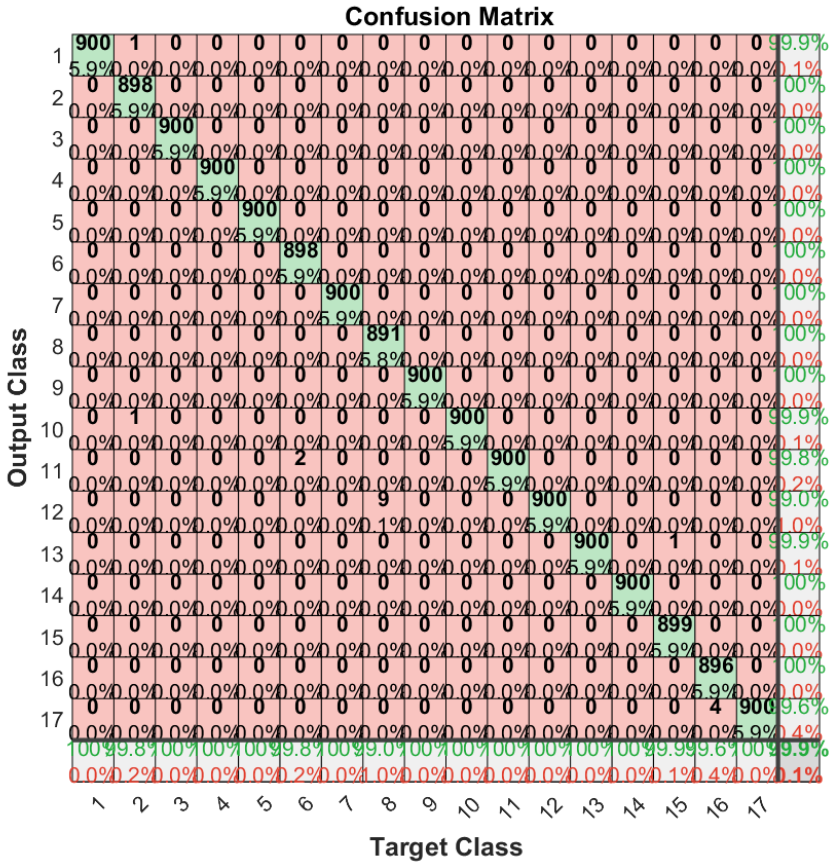


Figure 24: Confusion matrix for Training Original Data on Skinny Neural network with 1e-5 learning rate

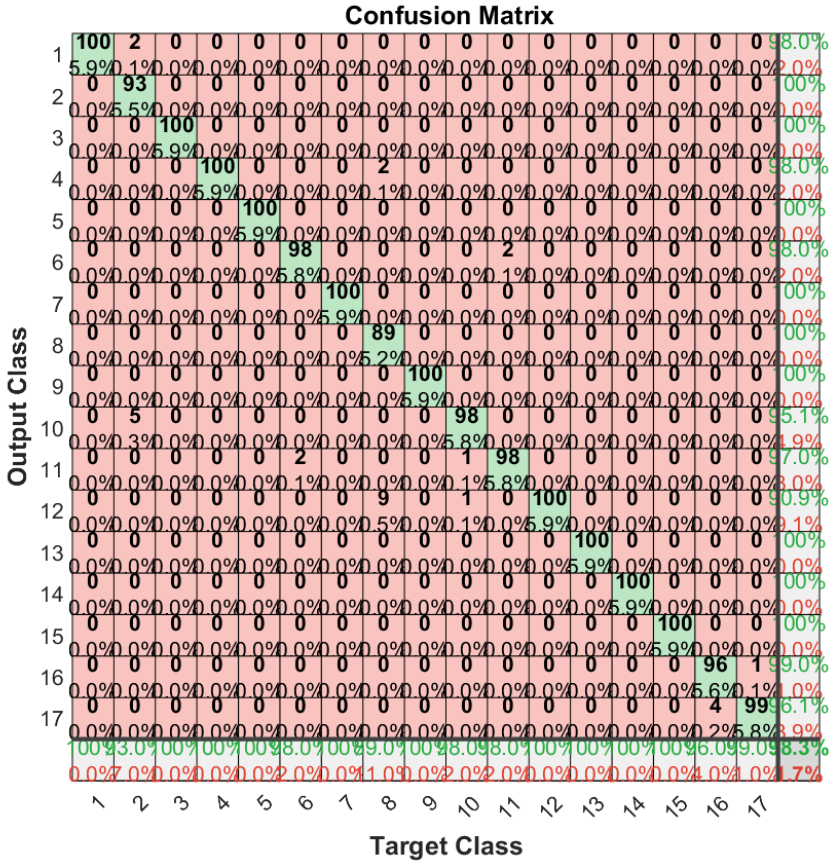


Figure 25: Confusion matrix for Validation on Skinny Neural network with 1e-5 learning rate

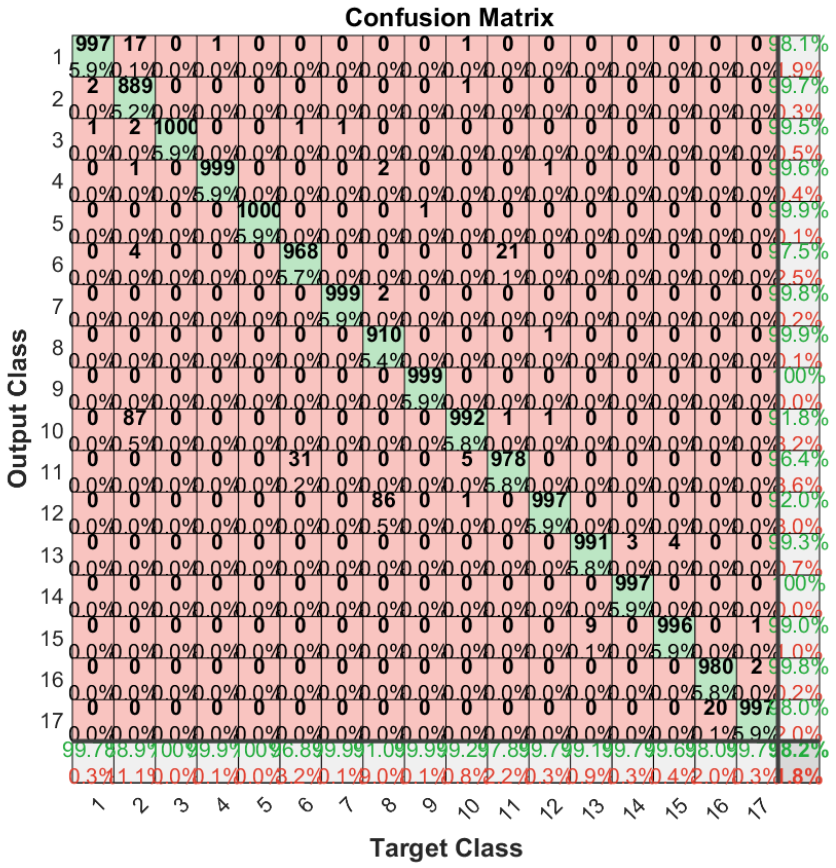


Figure 26: Confusion matrix for Testing on Skinny Neural network with 1e-5 learning rate

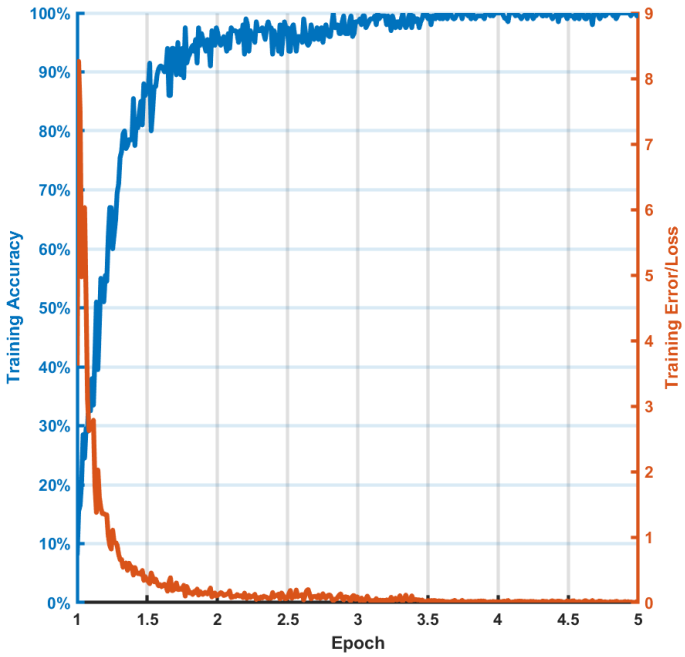


Figure 27: Training accuracies obtained for each epoch

5.7 Transfer Learning on Augmented Data Using AlexNet

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

The parameters for the network are listed in
For transfer learning, the last three layers of the AlexNet were removed and replaced

Figure 28: Alex Network Layers

with classification layers for our network according to the dataset.
The input layer was modified where the images were resized from $128x128$ to $227x227$ to be compatible with AlexNet.The model was trained for a variety of WeightLearnRateFactors and BiasLearnRateFactors in the finaly fully connected layer. The results is reported for WeightLearnRateFactor of 40 and a BiasLearnRateFactor of 40 whihc give the best convergence.

Parameter	Value
Batch size	150
Number of Epochs	10
Learning Rate	1.00E-04
Training Accuracy(%)	97
Validation Accuracy(%)	93
Testing Accuracy(%)	93
Training Time(seconds)	7824.26
Optimizer	sgdm

Table 9: Skinny Neural Networks Parameter values

Test accuracy of 93% was obtained for the augmented data.

5.7.1 Figures for Alex NN

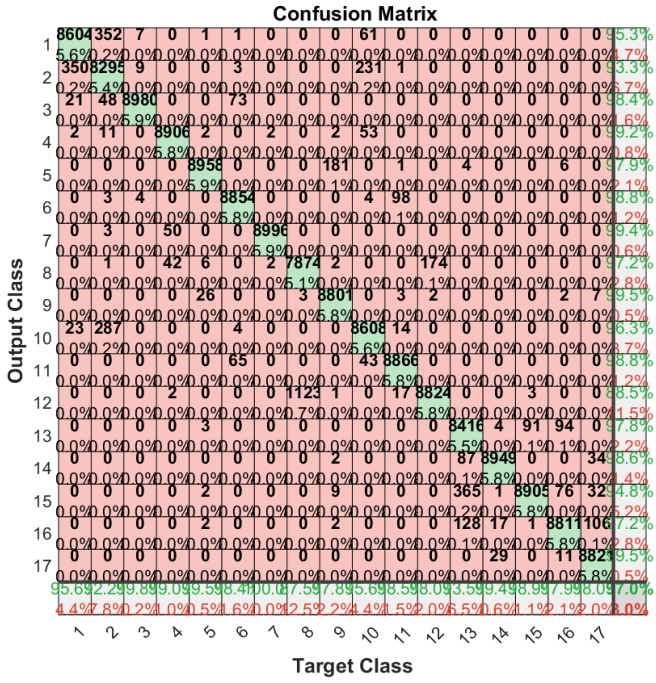


Figure 29: Confusion matrix for Training on Alex Neural network with 1e-5 learning rate

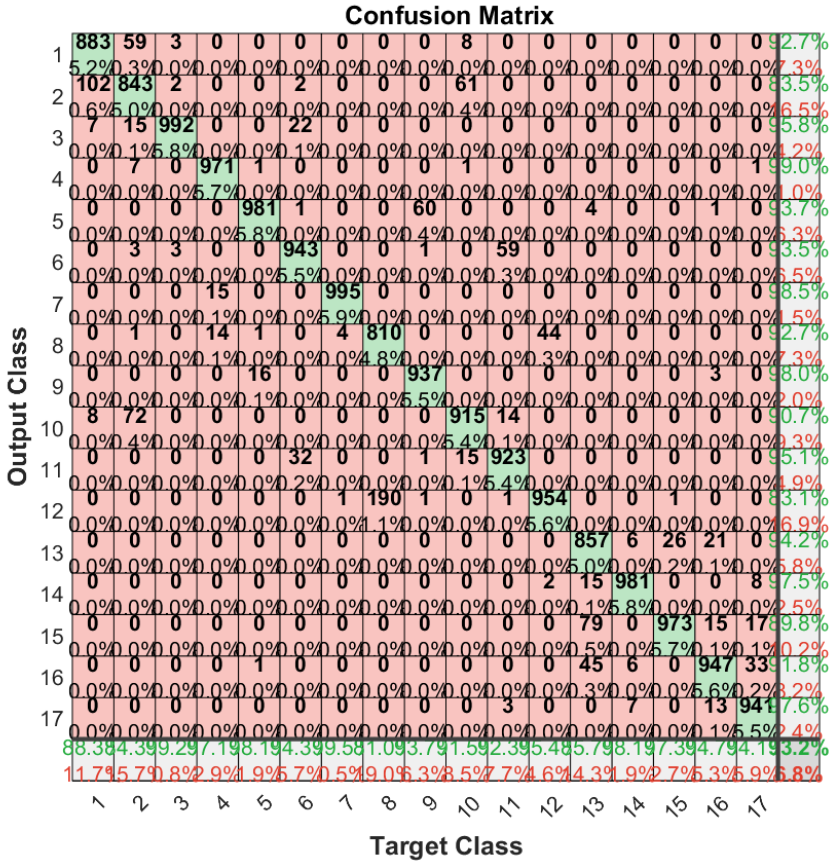


Figure 30: Confusion matrix for Validation on Alex Neural network with 1e-5 learning rate

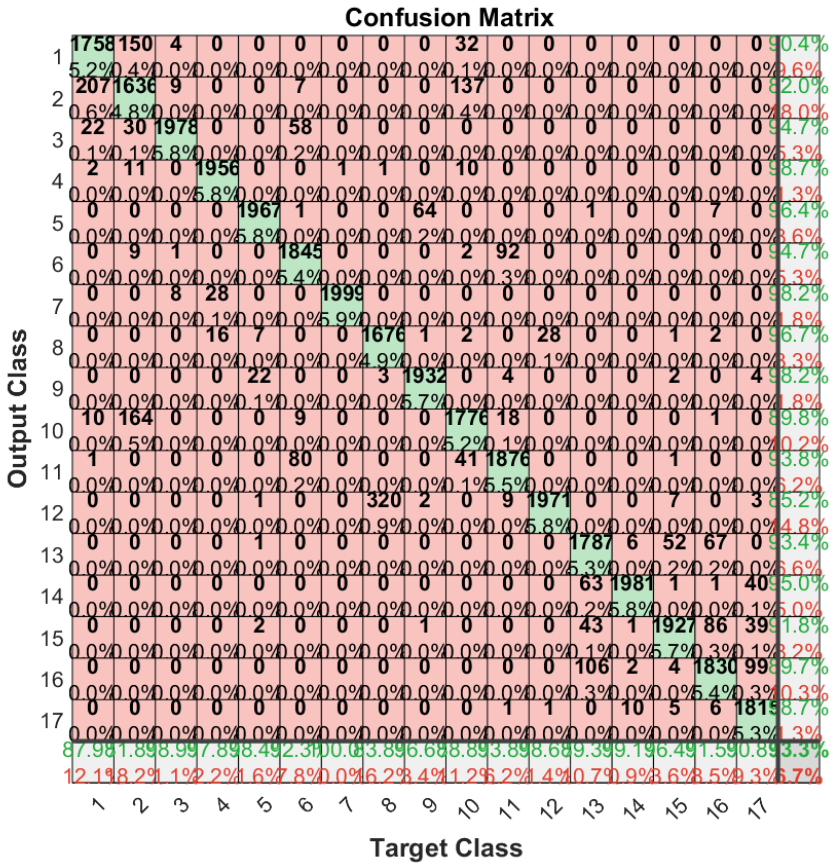


Figure 31: Confusion matrix for Testing on Alex Neural network with 1e-5 learning rate

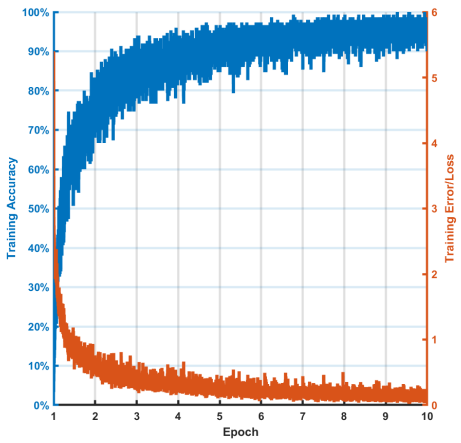
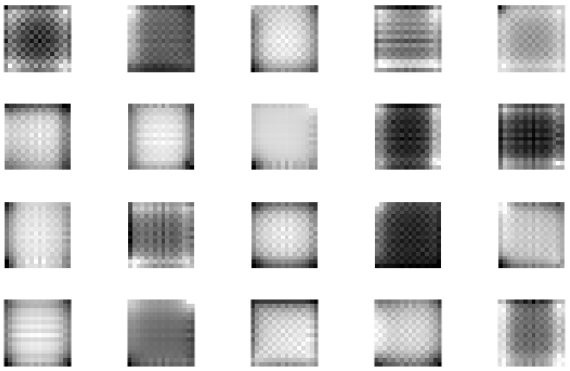


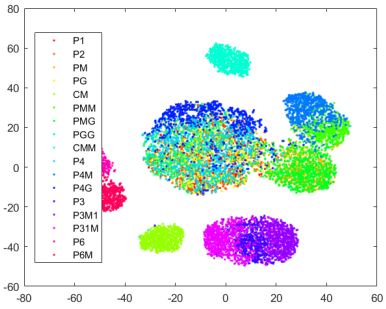
Figure 32: Training accuracies obtained for each epoch

6 Visulaization

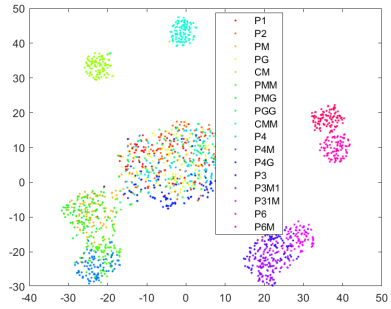
6.1 Original Dataset on Default Network



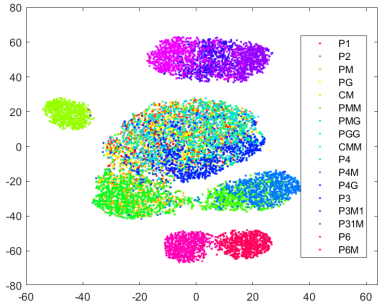
(a) First Layer



(b) t-SNE Visualization of Train



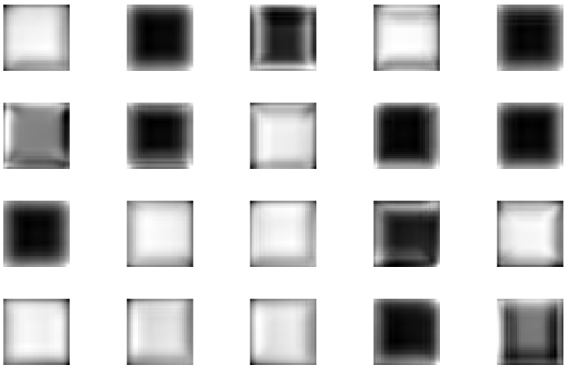
(c) t-SNE Visualization of Validation



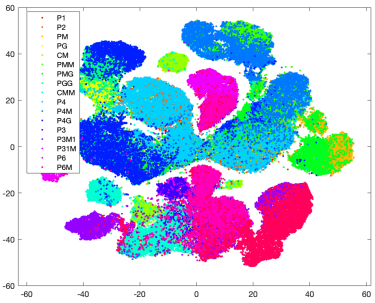
(d) t-SNE Visualization of Test

Figure 33: Visualization of Original Dataset on Default Network

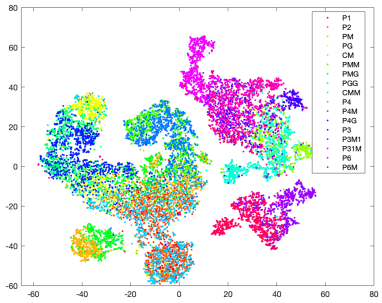
6.2 Augmented Dataset on Wide Network



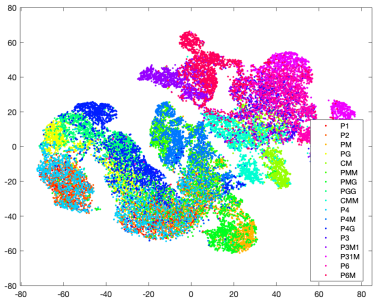
(a) First Layer



(b) t-SNE Visualization of Train



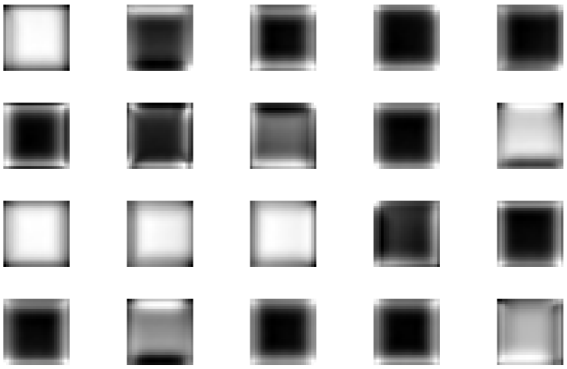
(c) t-SNE Visualization of Validation



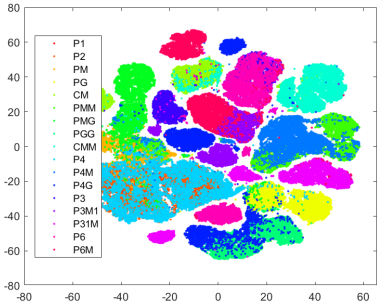
(d) t-SNE Visualization of Test

Figure 34: Visualization of Augmenented Dataset on Wide Network

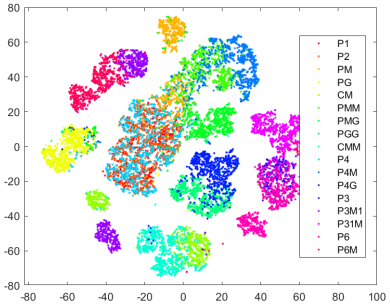
6.3 Augmented Dataset on Skinny Network



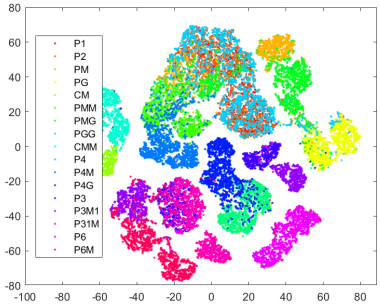
(a) First Layer



(b) t-SNE Visualization of Train



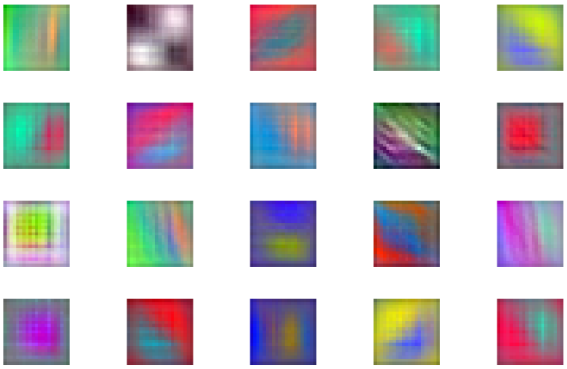
(c) t-SNE Visualization of Validation



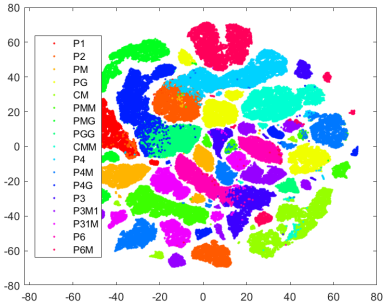
(d) t-SNE Visualization of Test

Figure 35: Visualization ofAugmented Dataset Skinny Network

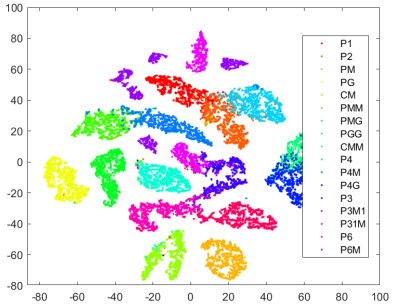
6.4 Augmented Dataset on AlexNet



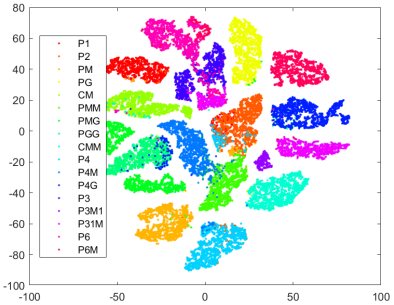
(a) First Layer



(b) t-SNE Visualization of Train



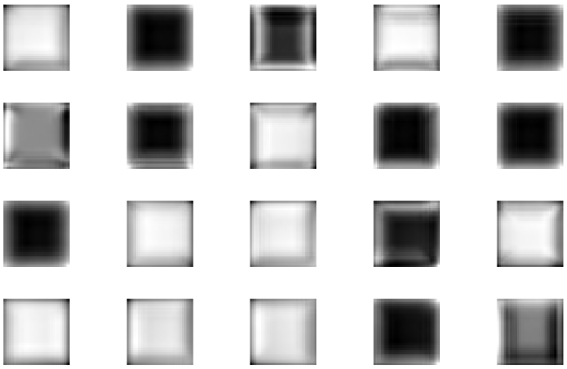
(c) t-SNE Visualization of Validation



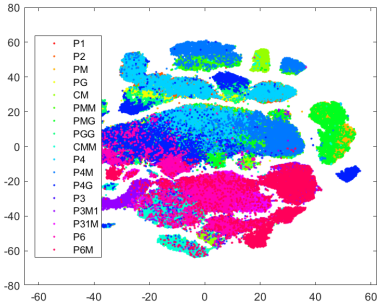
(d) t-SNE Visualization of Test

Figure 36: Visualization of Augmenented Dataset on Alex Network

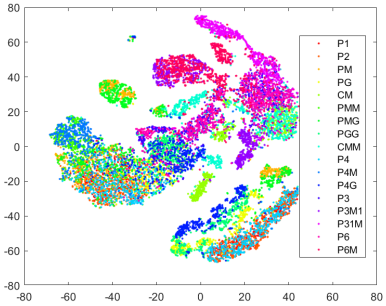
6.5 Original Dataset on Wide Network



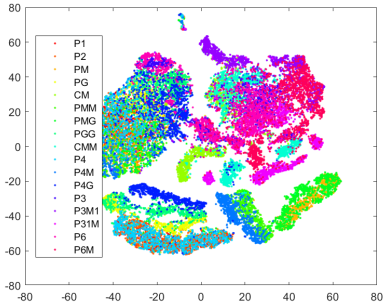
(a) First Layer



(b) t-SNE Visualization of Train



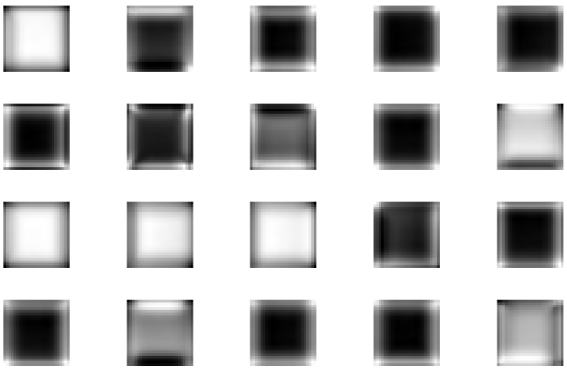
(c) t-SNE Visualization of Validation



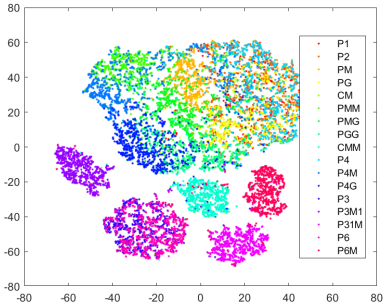
(d) t-SNE Visualization of Test

Figure 37: Visualization of Original Dataset on Wide Network

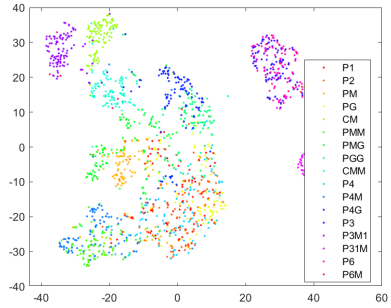
6.6 Original Dataset on Skinny Network



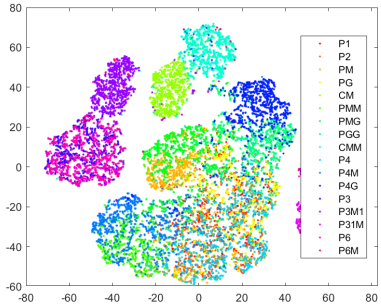
(a) First Layer



(b) t-SNE Visualization of Train



(c) t-SNE Visualization of Validation



(d) t-SNE Visualization of Test

Figure 38: Visualization of Original dataset on Skinny Network

7 Conclusion

Through this project we understood how to implement Neural Networks and Transfer Learning applications for pattern recognition problem. We understood how the learning rate and batch size played a role when it comes to designing the network. We implemented various neural networks such as Skinny Net and Wide Net. For the results obtained by training these networks on augmented and original dataset, it was seen that skinny/deeper networks perform better then wider network for a very large and convoluted dataset. Though skinny network takes longer to train when compared to wide network. When both the networks were trained on the original dataset, then the results obtained were almost similar.
We also learned how to implement image augmentation techniques which were useful in increasing the size of the dataset by scaling, rotating and translating the original image randomly.

8 Extra Credit

The MOCAP dataset was implemented in the extra credit portion.

8.1 Data Preprocessing

In this section the dataset was preprocessed according to the instructions give in the README file. Data related to subject 1, 2, 4, 8, 9, 10 was first extracted and the corresponding labels according to the years of practice were allocated.

8.2 Normalization

The dataset was normalized using the z-score method.
Z-score normalization is a strategy of normalizing data that avoids this outlier issue. The formula for Z-score normalization is below:

$$\frac{value - \mu}{\sigma}$$

Here, μ is the mean value of the feature and σ is the standard deviation of the feature. If a value is exactly equal to the mean of all the values of the feature, it will be normalized to 0. If it is below the mean, it will be a negative number, and if it is above the mean it will be a positive number. The size of those negative and positive numbers is determined by the standard deviation of the original feature. If the unnormalized data had a large standard deviation, the normalized values will be closer to 0.[1]
This method was used as the MOCAP captures the sensor data due to which there can be alot of outliers. This method helps to curb the effect of these outliers and this hence best suited for this type of data.

8.3 Network

Various networks were designed to get a good accuracy for the data. The following network was run for 10 epochs with the maximum batch size of 150 and learning rate of

1.00E-4. The parameters of the metwork are listed in Table 10

Parameter	Value
Batch size	150
Number of Epochs	10
Learning Rate	1.00E-04
Training Accuracy(%)	20
Validation Accuracy(%)	20
Testing Accuracy(%)	69
Training Time(seconds)	1008.39
Optimizer	sgdm

Table 10: MOCAP Convolutional Neural Networks Parameter values

As, the tarining dataset has been randomly split into 70:30 (i.e, 70% Training and 30% Validation), the accuracy of the network for the training and validation are low as all the labels haven't been equally divided.
I tried to visualize the t-SNE for the data as well but was unable to do so. The following error was obtained:
Warning: Maximum iteration reached: Not all binary searches converge to the optimal value.

$$> Intsne > binarySearchVariance(line475)$$
$$Intsne(line363)$$
$$Outofmemory.$$
$$Errorintsne > probMatXknn(line706)$$
$$colidx = colidx(sridx)';$$
$$Errorintsne(line369)$$
$$[colidx, rowcnt, cumrowcnt, probMatX] = probMatXknn(probMatX, knnidx);$$

I worked on it for 3 whole days but couldnot understand why it was going out of memory.

8.3.1 Figures for MOCAP Convolutional NN

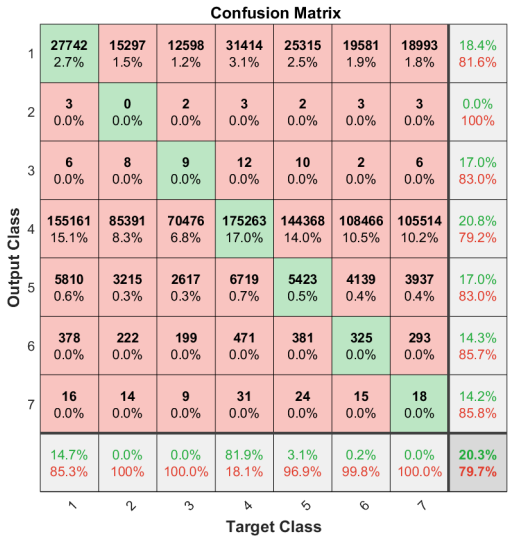


Figure 39: Confusion matrix for Training on MOCAP Convolutional Neural network with 1e-5 learning rate

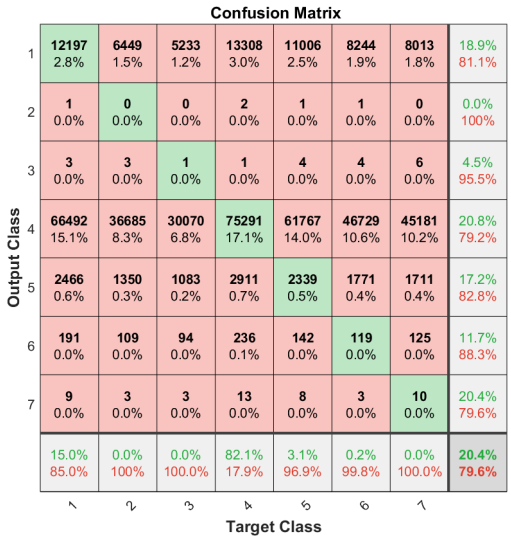


Figure 40: Confusion matrix for Validation on MOCAP Convolutional Neural network with 1e-5 learning rate

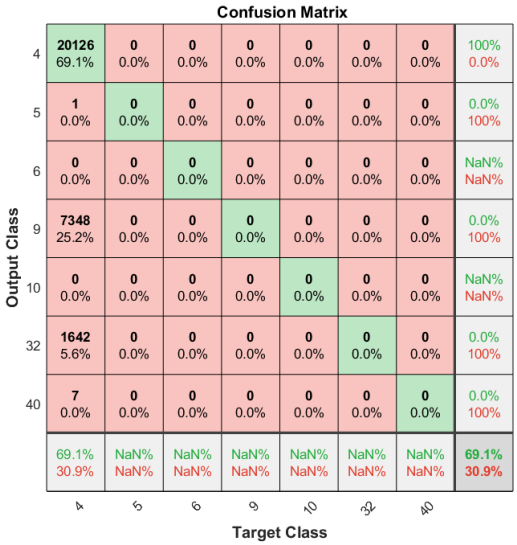


Figure 41: Confusion matrix for Testing on MOCAP Convolutional Neural network with 1e-5 learning rate

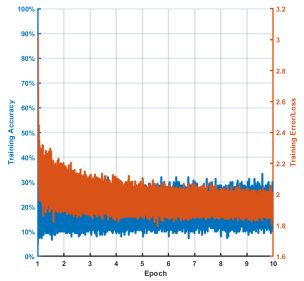


Figure 42: Training accuracies and Loss obtained for each epoch

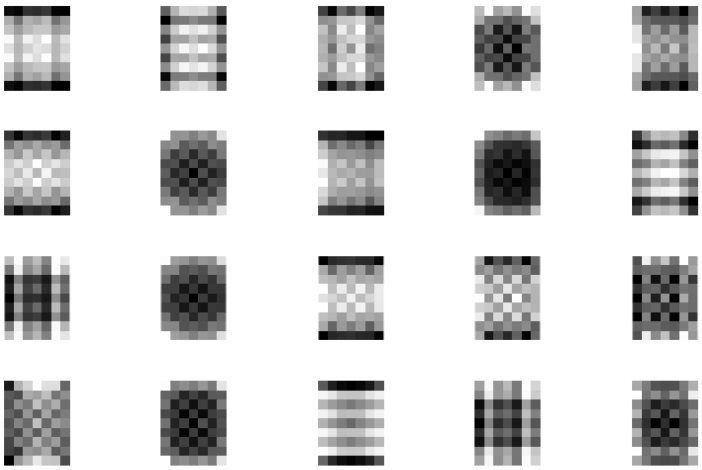


Figure 43: First Convolutional Layer Visualization

References

[1] Normalization Method,
`https://www.codecademy.com/articles/normalization` 41

[2] Bishop,Pattern Recognition and Machine Learning,
`http://users.isr.ist.utl.pt/ wurmd/Livros/school/Bishop`