Project 4: CNN

Due on Friday, Apl 6, 2020 at 11.59PM

Name: Ke LIANG ID: 926791183 email: kul660@psu.edu

contents

1 Introduction

The goal of this project is to train a CNN on the wallpaper images we previously classified in the last project. The dataset we use is Wallpaper Dataset.

2 Methods

2.1 Dataset

As for **Wallpaper Group** dataset, there are 17000 images, 1000 images per group, and each image containing a wallpaper pattern in the datasets. The baseline images are of size 256x256 and are grayscale. There are 17 classes and 500 dimensions (feature) in the data set including (1) P1 group, (2) P2 group, (3) PM group, (4) PG group, (5) CM group, (6) PMM group, (7) PMG group, (8) PGG group, (9) CMM group, (10) P4 group, (11) P4M group, (12) P4G group, (13) P3 group, (14) P3M1 group, (15) P31M group, (16) P6 group and (17) P6M group.[1]

We have to do augmentation on the dataset to improve the accuracy of the model we build. The augmentation method is shown below.

We will randomly choose the degree to rotate, the number to scale and the value to translate, then adding with a cropping making the size of the input image from 256x256 to 128x128. And the range of the rotation is from 0 to 360 degree, and the range of scale is from 1.5 to 2 times itself, meanwhile the translation for x and y are separated, both of them are randomly chosen from 0 to 20, and of course they are integral pixels, which make the transition inside the original pattern scale. And we do augmentation on training set 5 times and once on testing set. All the augmentation parameters are based on the description of the project 4. The comparison and the of the original data and the data with augmentation is shown in **result** part (chapter 3).

2.2 Original Network

The original network architecture is shown below.

INPUT -> CONV -> ReLU -> POOL -> FC -> DROPOUT -> FC -> SOFTMAX

The parameters of the network is explained below.

For the InputLayer, you can choose both 256x256x1 and 128x128x1 for this project.

For the CONV layer, the size of the filter is 5x5, and the number of filters is 20, and they do [2 2] padding on this layer and the stride is 2. The total parameters of this layer is 520 ((5x5+1)*20 = 520).

For the Pool layer, the pool size is 2x2.

The size of FullyConnectedLayer is 25.

The value of dropout is 0.4, and for the last FC layer the value of parameters is 17 which represents the number of classes.

2.3 Skinny Network

The Skinny network architecture is shown below by using analyzeNetwork() function.

INPUT -> (CONV -> batchNorm -> ReLU -> POOL)*3 -> FC -> DROPOUT -> FC -> SOFTMAX -> Output

	Name	Type	Activations	Learnables
1	imageinput 256x256x1 images with 'zerocenter' normalization	Image Input	256×256×1	-
2	conv_1 20 5x5x1 convolutions with stride [2 2] and padding [0 0 0 0]	Convolution	126×126×20	Weights 5×5×1×20 Bias 1×1×20
3	batchnorm_1 Batch normalization with 20 channels	Batch Normalization	126×126×20	Offset 1×1×20 Scale 1×1×20
4	relu_1 ReLU	ReLU	126×126×20	-
5	maxpool_1 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	63×63×20	-
6	conv_2 20 5x5x20 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	63×63×20	Weights 5×5×20×20 Bias 1×1×20
7	batchnorm_2 Batch normalization with 20 channels	Batch Normalization	63×63×20	Offset 1×1×20 Scale 1×1×20
8	relu_2 ReLU	ReLU	63×63×20	-
9	maxpool_2 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	31×31×20	-
10	conv_3 40 3x3x20 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	31×31×40	Weights 3×3×20×40 Bias 1×1×40
11	batchnorm_3 Batch normalization with 40 channels	Batch Normalization	31×31×40	Offset 1×1×40 Scale 1×1×40
12	relu_3 ReLU	ReLU	31×31×40	-
13	maxpool_3 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	15×15×40	-
14	fc_1 25 fully connected layer	Fully Connected	1×1×25	Weights 25×9000 Bias 25×1
15	dropout 10% dropout	Dropout	1×1×25	-
16	fc_2 17 fully connected layer	Fully Connected	1×1×17	Weights 17×25 Bias 17×1
17	softmax softmax	Softmax	1×1×17	-
18	classoutput crossentropyex with 'P1' and 16 other classes	Classification Output	-	-

2.4 Wide Network

The Skinny network architecture is shown below, by using analyzeNetwork() function.

INPUT -> (CONV -> batchNorm -> ReLU -> POOL)*2 -> FC -> DROPOUT -> FC -> SOFTMAX -> Output

	Name	Type	Activations	Learnables
1	imageinput 256x256x1 images with 'zerocenter' normalization	Image Input	256×256×1	-
2	conv_1 40 5x5x1 convolutions with stride [1 1] and padding ...	Convolution	256×256×40	Weights 5×5×1×40 Bias 1×1×40
3	batchnorm_1 Batch normalization with 40 channels	Batch Normalization	256×256×40	Offset 1×1×40 Scale 1×1×40
4	relu_1 ReLU	ReLU	256×256×40	-
5	maxpool_1 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	128×128×40	-
6	conv_2 80 3x3x40 convolutions with stride [1 1] and padding...	Convolution	128×128×80	Weights 3×3×40×80 Bias 1×1×80
7	batchnorm_2 Batch normalization with 80 channels	Batch Normalization	128×128×80	Offset 1×1×80 Scale 1×1×80
8	relu_2 ReLU	ReLU	128×128×80	-
9	maxpool_2 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	64×64×80	-
10	fc_1 25 fully connected layer	Fully Connected	1×1×25	Weights 25×327680 Bias 25×1
11	dropout 60% dropout	Dropout	1×1×25	-
12	fc_2 17 fully connected layer	Fully Connected	1×1×17	Weights 17×25 Bias 17×1
13	softmax softmax	Softmax	1×1×17	-
14	classoutput crossentropyex with 'P1' and 16 other classes	Classification Output	-	-

2.5 Alexnet

Since the input of the alexnet CNN is 227x227x3, I first write an code to modify all the augmentation data to the format for Alexnet CNN, then write the code for alexnet CNN. I set all the layers except the last 3 layers for the alexnet CNN, and then FC layer with WeightLearnRateFactor and BiasLearnRateFactor, and one softmaxLayer and one classificationLayer follow by the FC layer.

I finetune the FC layer with the parameters and values of batchsize learning rate to get good performance.

The architecture of the network is shown below.

ANALYSIS RESULT

	Name	Type	Activations	Learnables
1	data 227x227x3 images with 'zerocenter' normalization	Image Input	227×227×3	-
2	conv1 96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]	Convolution	55×55×96	Weights 11×11×3×96 Bias 1×1×96
3	relu1 ReLU	ReLU	55×55×96	-
4	norm1 cross channel normalization with 5 channels per element	Cross Channel Nor...	55×55×96	-
5	pool1 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	27×27×96	-
6	conv2 256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	27×27×256	Weights 5×5×48×256 Bias 1×1×256
7	relu2 ReLU	ReLU	27×27×256	-
8	norm2 cross channel normalization with 5 channels per element	Cross Channel Nor...	27×27×256	-
9	pool2 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	13×13×256	-
10	conv3 384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	13×13×384	Weight… 3×3×256×38.. Bias 1×1×384
11	relu3 ReLU	ReLU	13×13×384	-
12	conv4 384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	13×13×384	Weight… 3×3×192×38.. Bias 1×1×384
13	relu4 ReLU	ReLU	13×13×384	-
14	conv5 256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	13×13×256	Weight… 3×3×192×25.. Bias 1×1×256
15	relu5 ReLU	ReLU	13×13×256	-
16	pool5 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	6×6×256	-
17	fc6 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×9216 Bias 4096×1
18	relu6 ReLU	ReLU	1×1×4096	-
19	drop6 50% dropout	Dropout	1×1×4096	-
20	fc7 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×4096 Bias 4096×1
21	relu7 ReLU	ReLU	1×1×4096	-
22	drop7 50% dropout	Dropout	1×1×4096	-
23	fc8 1000 fully connected layer	Fully Connected	1×1×1000	Weights 1000×4096 Bias 1000×1
24	prob softmax	Softmax	1×1×1000	-
25	output crossentropyex with 'tench' and 999 other classes	Classification Output	-	-

2.6 Software

The codes are run in MatlabR2017b with gpu on the server. And all the procedure results are saved with the txt file in the folder named like "process_original_results_batchsize75.txt". While for the data part, both the data augmentation and data for alexnet are build by running in MatlabR 2019a. Just follow the readme file which includes all the information.

3 Results

The classification matrix/confusion matrix is shown below by using the heatmap function, and somehow since the server cannot show the figure. I save the matrix for each set of each situation as a .mat file which is shown in the folder which is shown below. Then follow the readme file and print for example “load(test_class_orig_batch

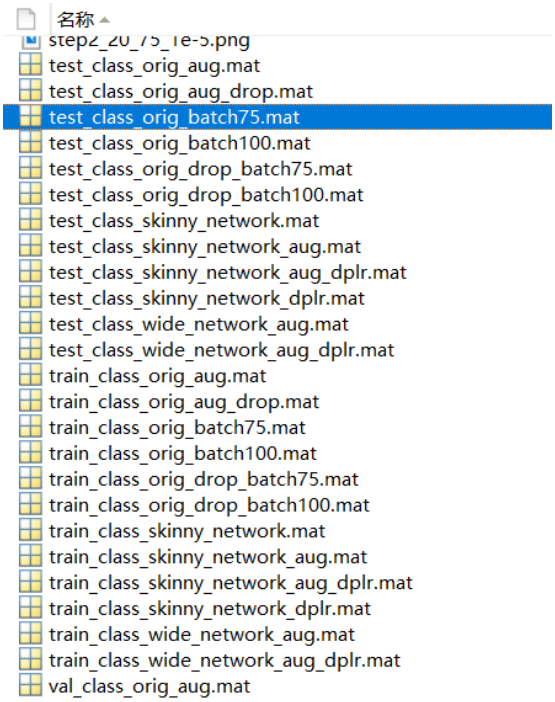

Figure 1 figure for the .mat files

75.mat) and then use heatmap(test_class) you will get the results”. We will only show the confusion matrix (actually the classification matrix has same meaning of the confusion matrix), and the TA said it is ok if we using heatmap to show one of them.

3.1 Results on Step 1

Here are the results of Step 1, we will explain that in the following 4 parts. The parameters for each part is different, especially for the batch size and learning rate. This part is aimed to get the conclusion for the influence of the value of the batch size and learning rate.

3.1.1 Results for batch size = 75, learning rate = 1e-4

Check the process_original_results_batchsize75.txt file. The accuracy of training set is 0.8431, and the accuracy of the validate set is 0.7635, and the accuracy of the testing set is 0.7605. This part takes 393.93s to finish the whole 20 epochs.

The training accuracy together with the loss figure is shown below.

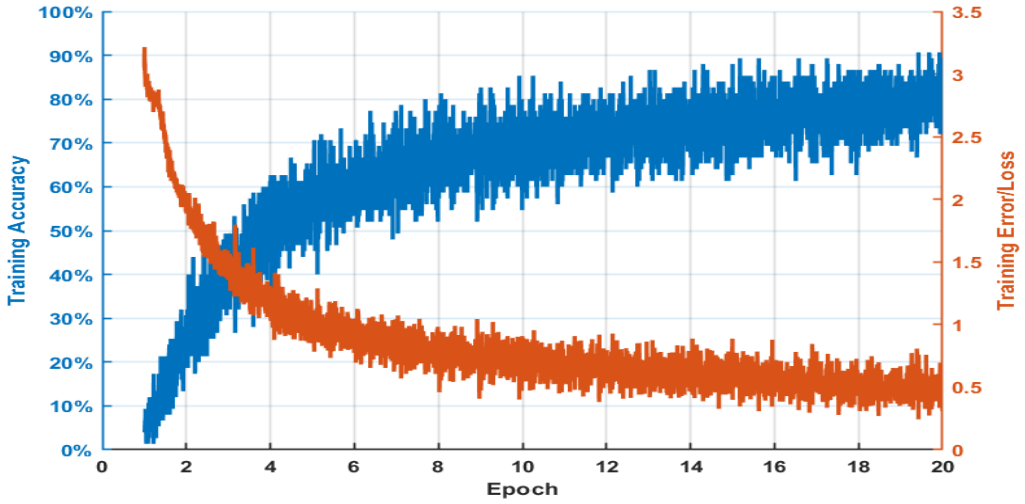

Figure 2 training accuracy together with the loss figure of original network batchsize75 lr = 1e-4

The confusion matrix for training set in this situation is shown below.

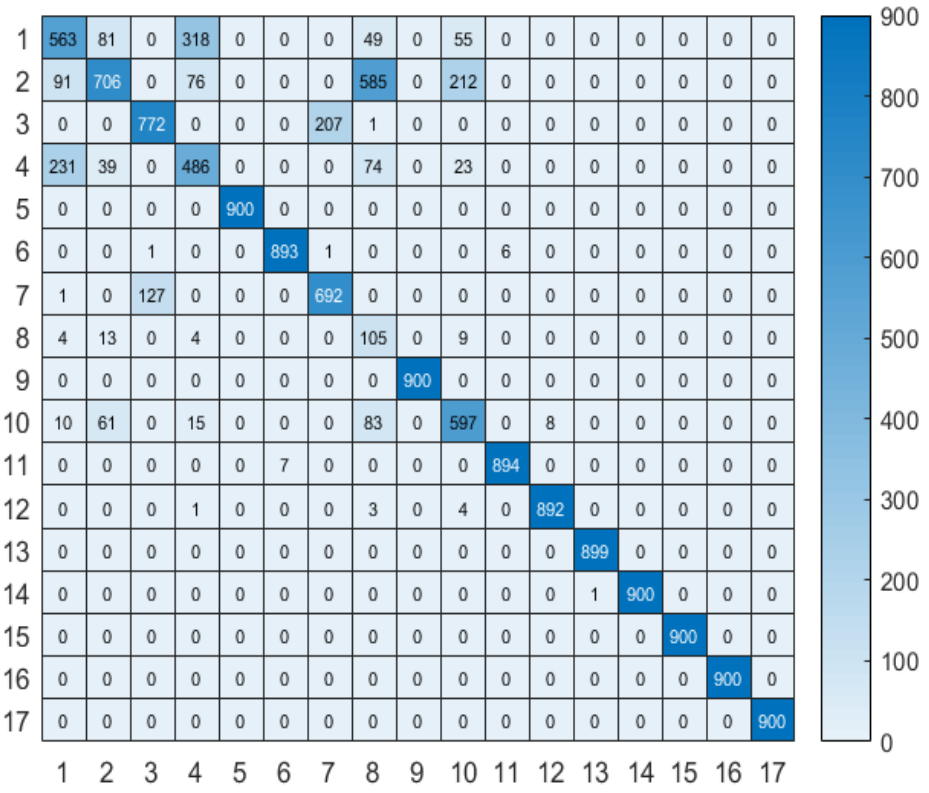

Figure 3 confusion matrix for training dataset of original network with batch = 75 lr = 1e-4

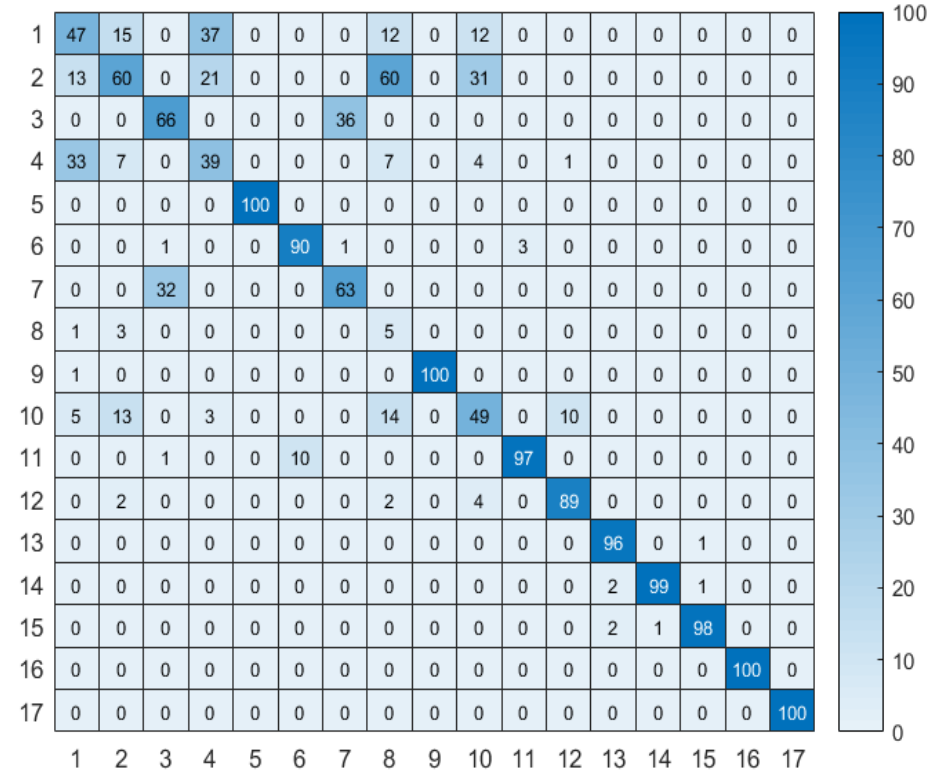

Figure 4 confusion matrix for val dataset of original network with batch = 75 lr = 1e-4

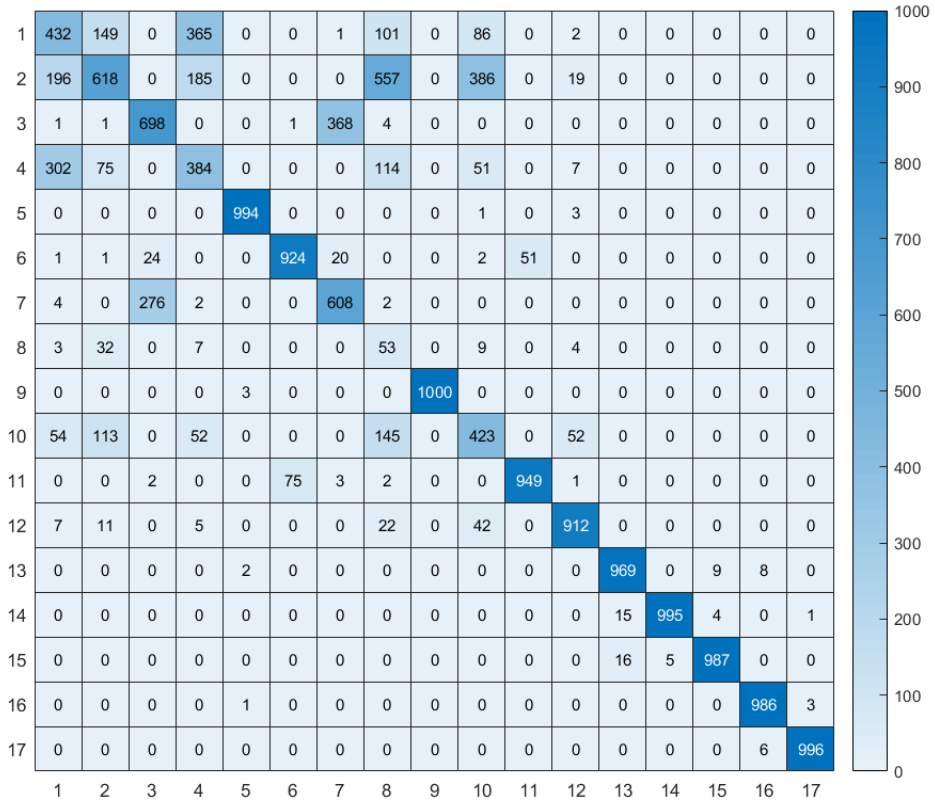

Figure 5 confusion matrix for testing dataset of original network with batch = 75 lr = 1e-4

3.1.2 Results for batch size = 75, learning rate = 1e-5

Check the process_original_results_batchsize75.txt file. The accuracy of training set is 0.8747, and the accuracy of the validate set is 0.7776, and the accuracy of the testing set is 0.7771. This part takes 402.37s to finish the whole 20 epochs.

This part is based on the procedure in the 3.1.1. The training accuracy together with the loss figure is shown below.

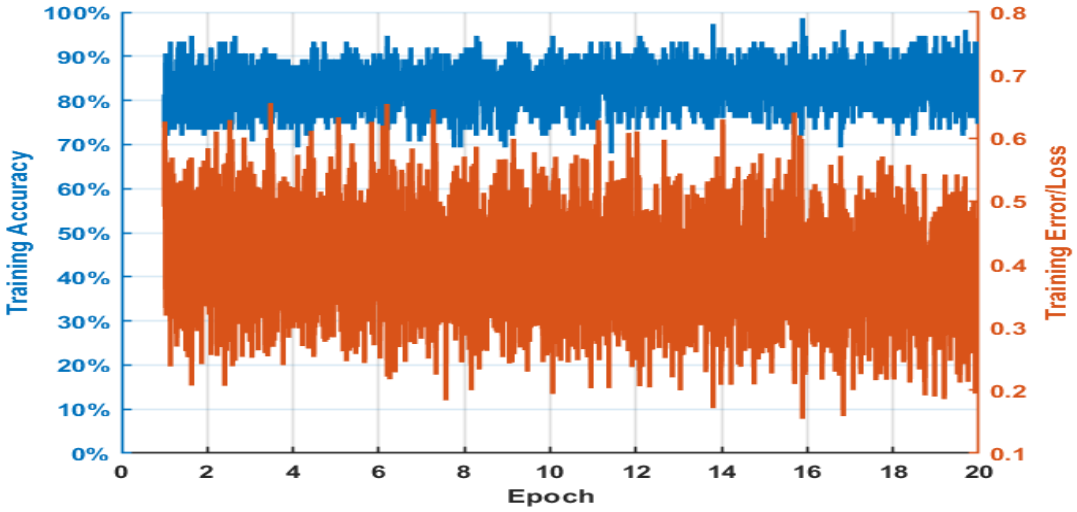

Figure 6 training accuracy together with the loss figure of original network batchsize75 lr = 1e-5

The confusion matrix for training set in this situation is shown below.

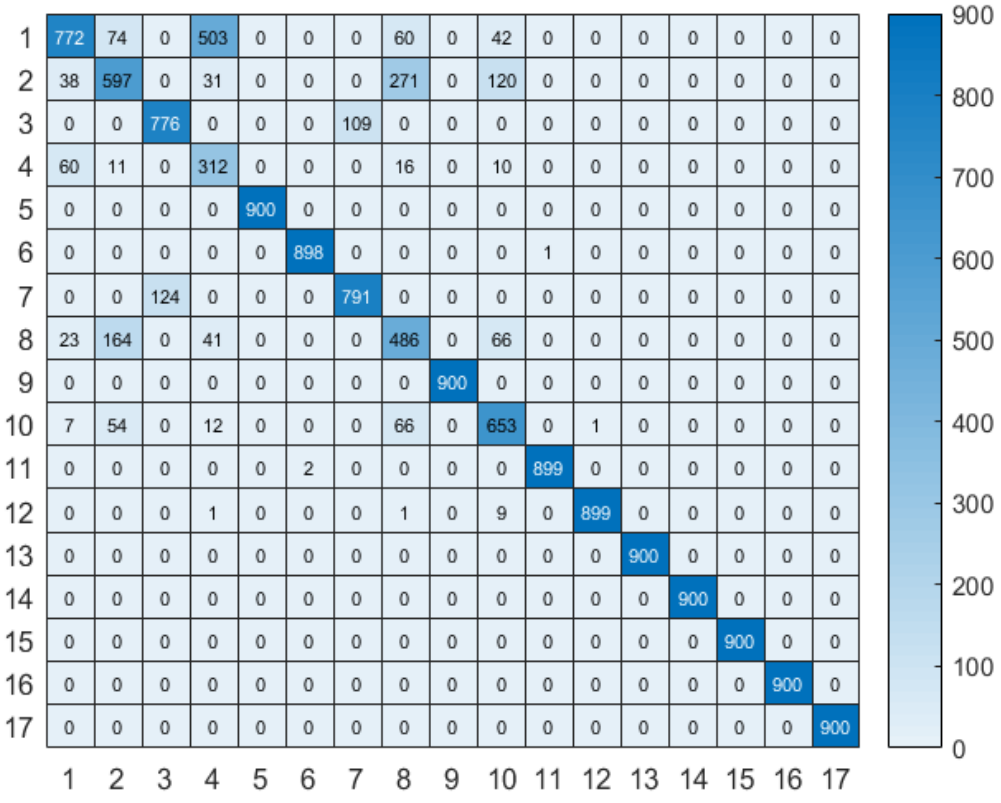

Figure 7 confusion matrix for training dataset of original network with batch = 75 lr = 1e-5

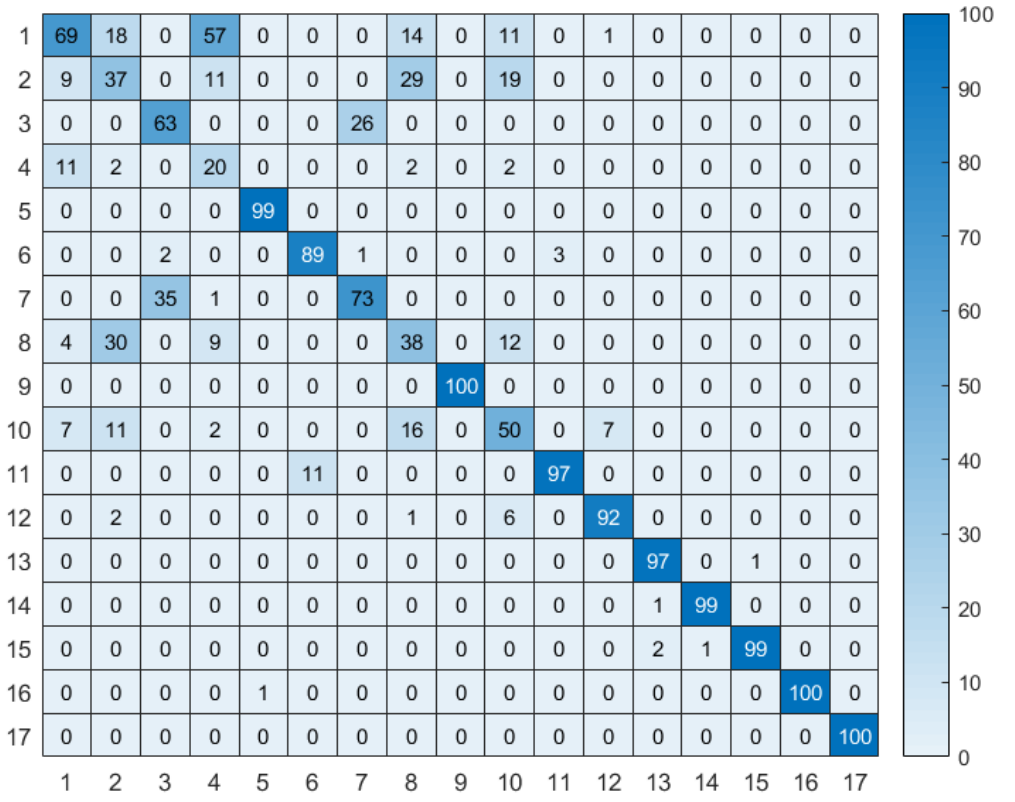

Figure 8 confusion matrix for val dataset of original network with batch = 75 lr = 1e-5

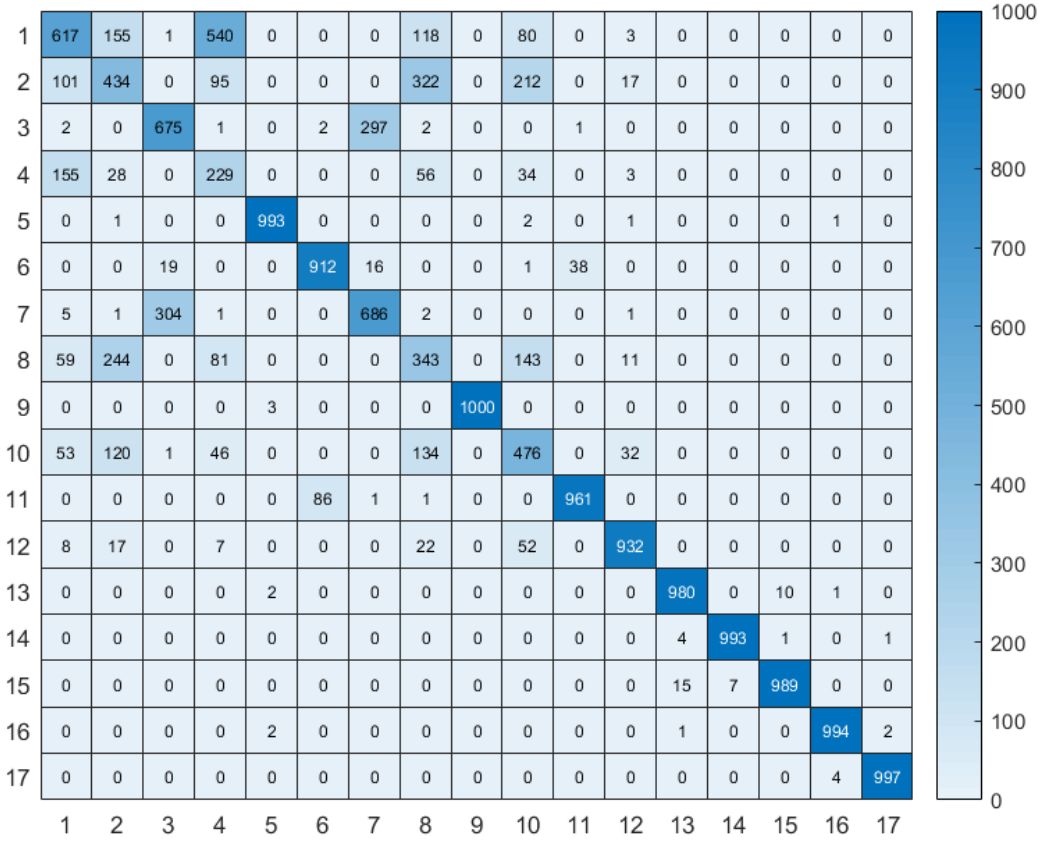

Figure 9 confusion matrix for testing dataset of original network with batch = 75 lr = 1e-5

3.1.3 Results for batch size = 100, learning rate = 1e-4

Check the process_original_results_batchsize100.txt file. The accuracy of training set is 0.7803, and the accuracy of the validate set is 0.7306, and the accuracy of the testing set is 0.7192. This part takes 354.28s to finish the whole 20 epochs.

The training accuracy together with the loss figure is shown below.

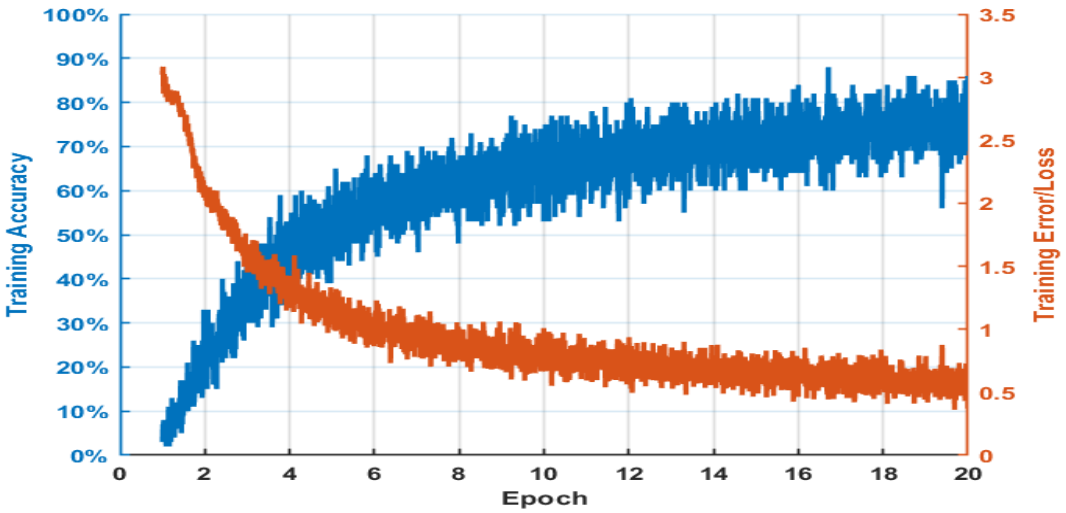

Figure10 training accuracy together with the loss figure of original network batchsize100 lr = 1e-4

The confusion matrix for training set in this situation is shown below.

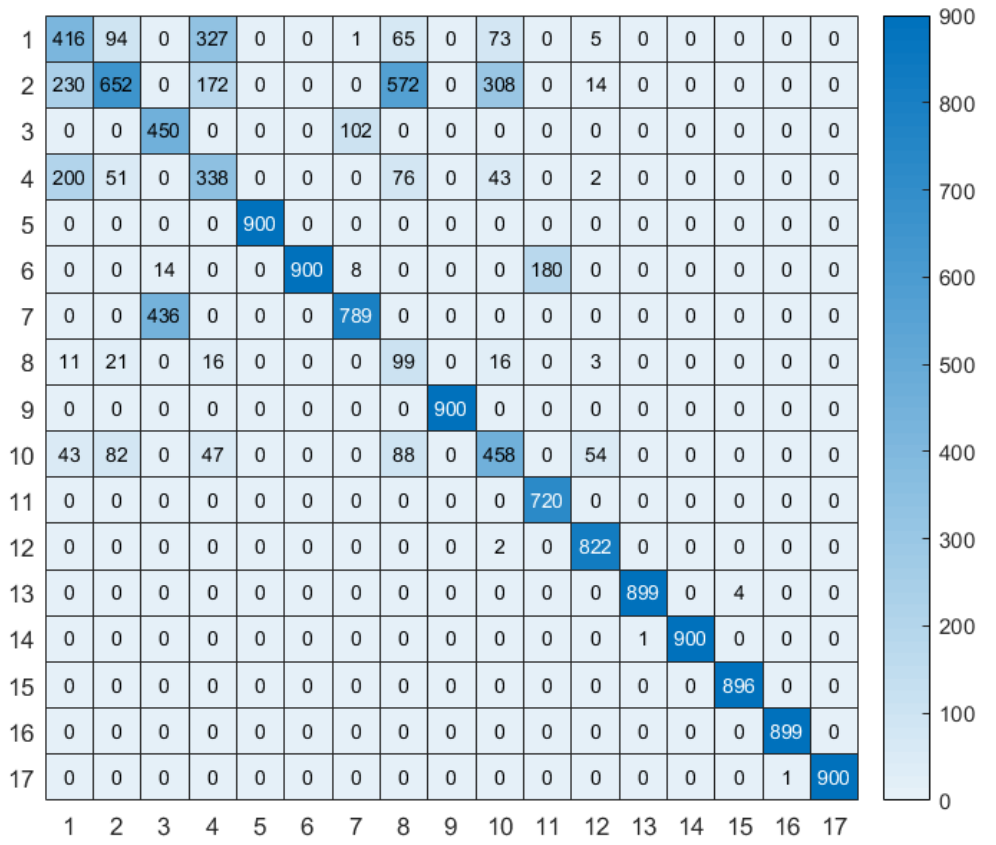

Figure11 confusion matrix for training dataset of original network with batch = 100 lr = 1e-4

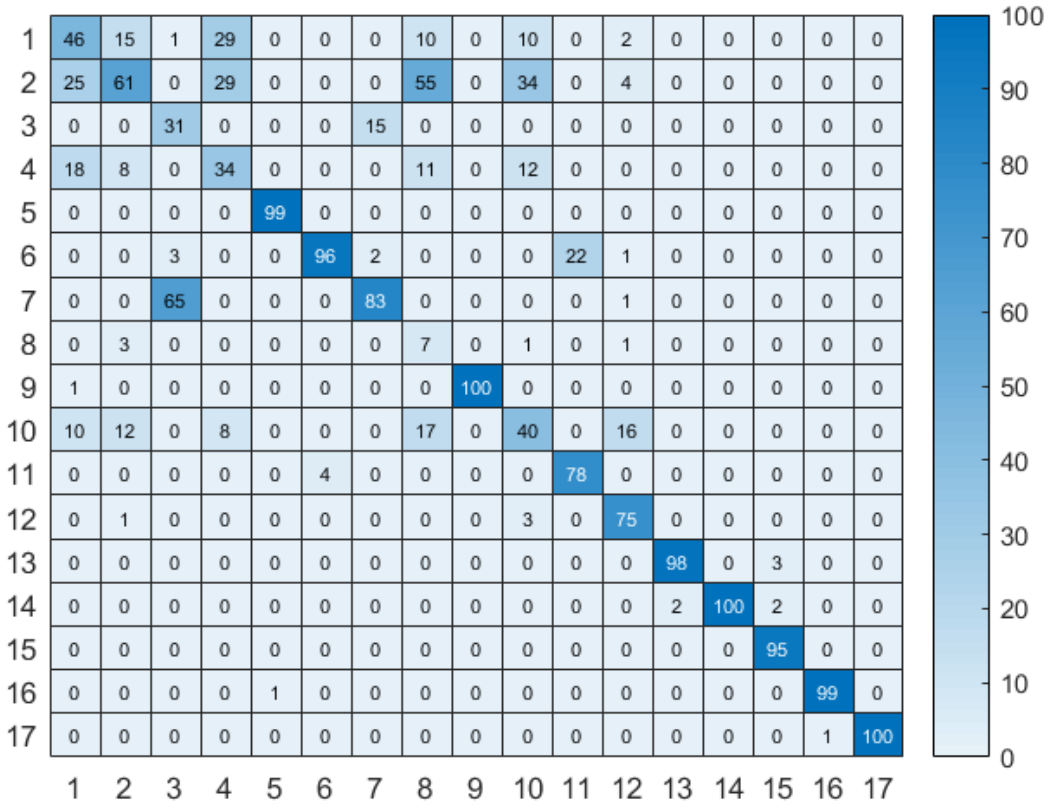

Figure12 confusion matrix for val dataset of original network with batch = 100 lr = 1e-4

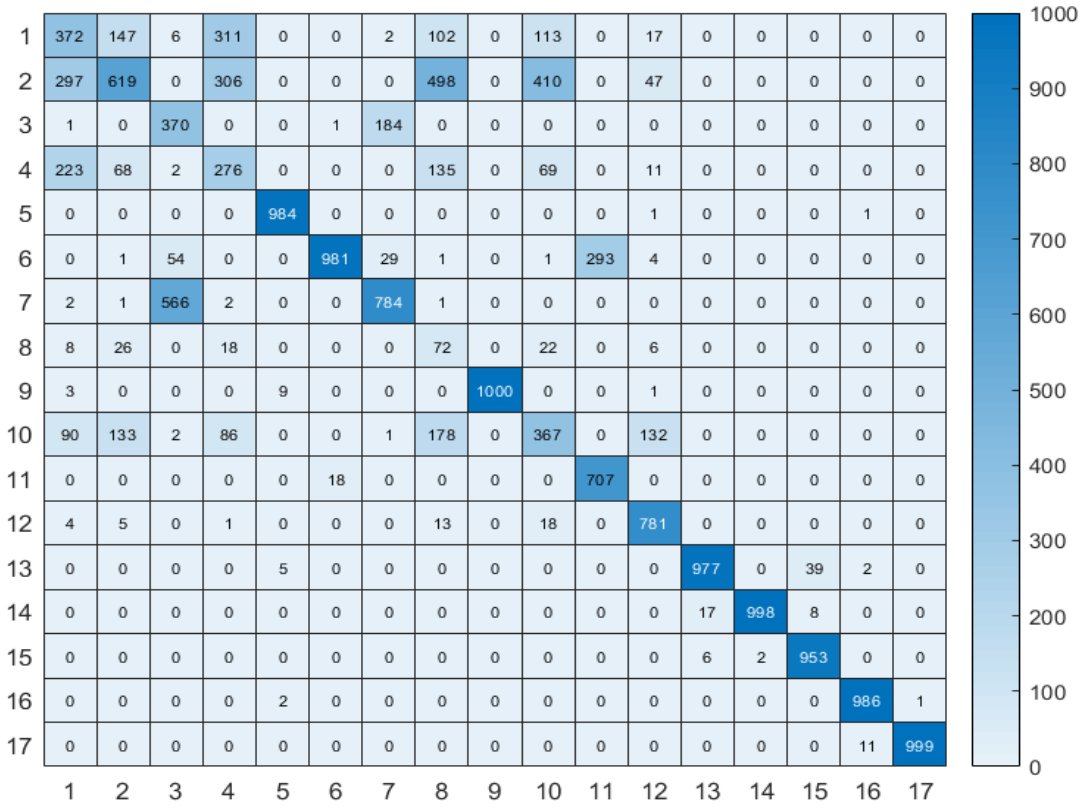

Figure13 confusion matrix for testing dataset of original network with batch = 100 lr = 1e-4

3.1.4 Results for batch size = 100, learning rate = 1e-5

Check the process_original_results_batchsize100.txt file. The accuracy of training set is 0.8316, and the accuracy of the validate set is 0.7636, and the accuracy of the testing set is 0.7488. This part takes 378.77s to finish the whole 20 epochs.

This part is based on the procedure in the 3.1.3. The training accuracy together with the loss figure is shown below.

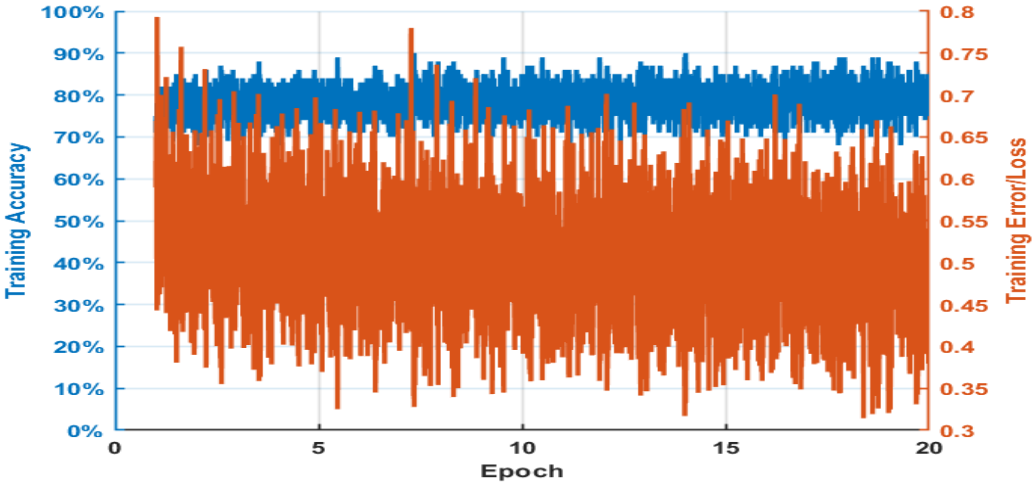

Figure14 training accuracy together with the loss figure of original network batchsize100 lr = 1e-5

The confusion matrix for training set in this situation is shown below.

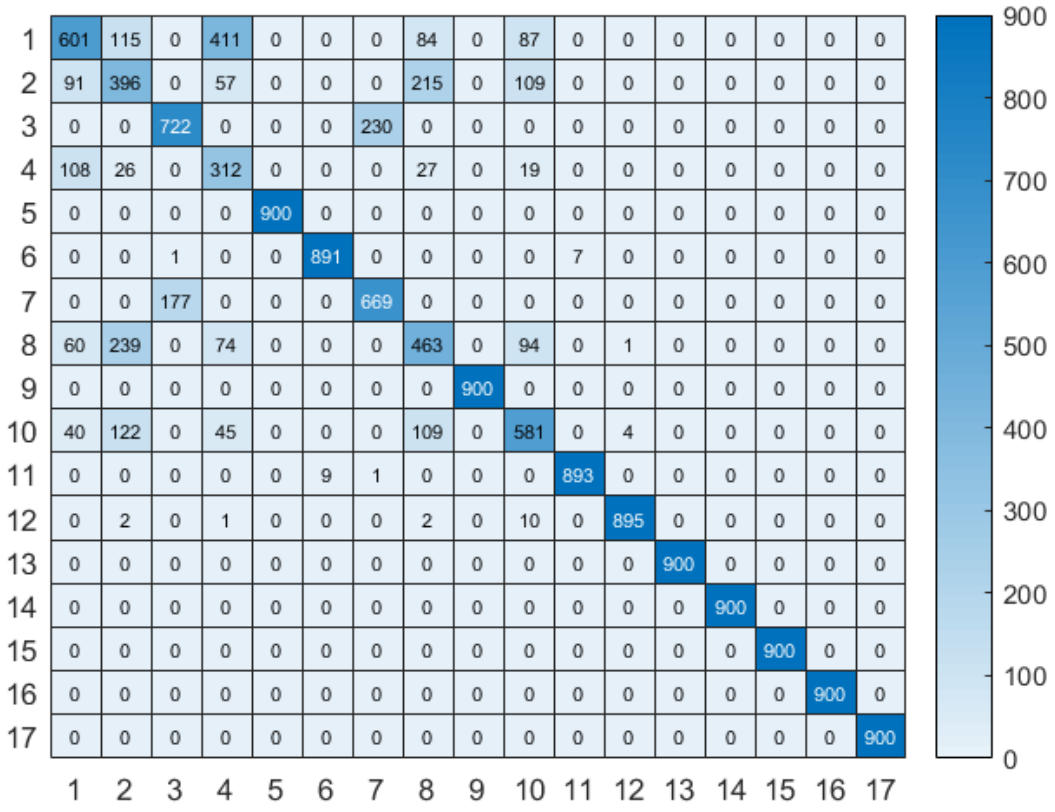

Figure 15 confusion matrix for training dataset of original network with batch = 100 lr = 1e-5

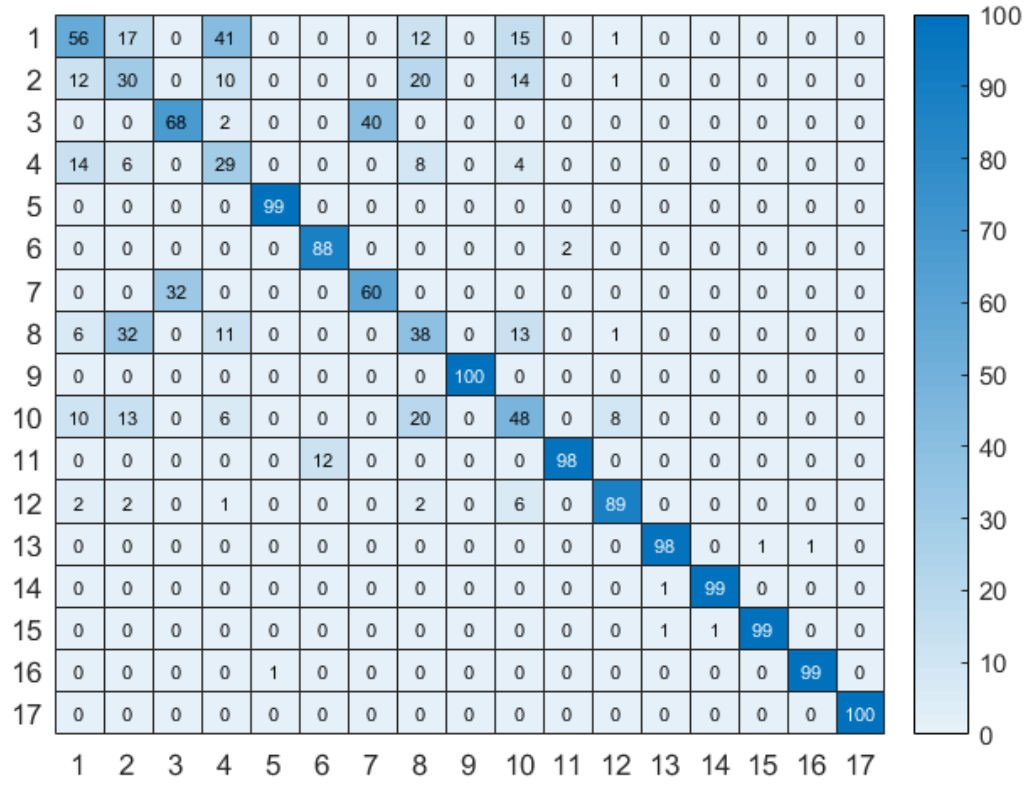

Figure 16 confusion matrix for val dataset of original network with batch =100 lr = 1e-5

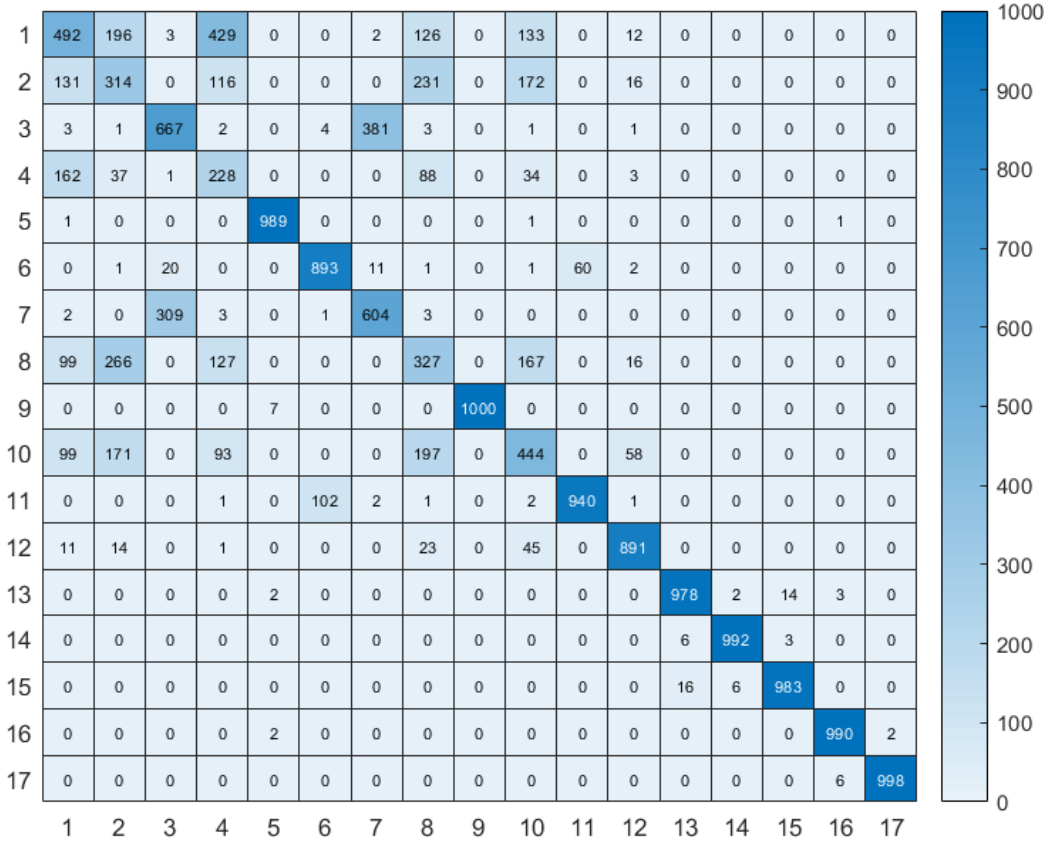

Figure 17 confusion matrix for testing dataset of original network with batch = 100 lr = 1e-5

3.1.5 Conclusion for step 1

Based on above images, it is easy to get if you we have good performance when batchsize is 75 and for the learning rate 1e-4, it is will make the converge more quickly and for the learning rate 1e-5, it will slow down the converge but it is good for us once the optimal results are inside an internal which is smaller than 1e-5.

3.2 Results on Step 2

3.2.1 Data with Augmentation

Here are some examples for the image with these 3 augmentations (scale, rotation and transition), and I save the degree, times for scale and x_transition and y_transition for each aug images. You can run

For example the original image is shown below.

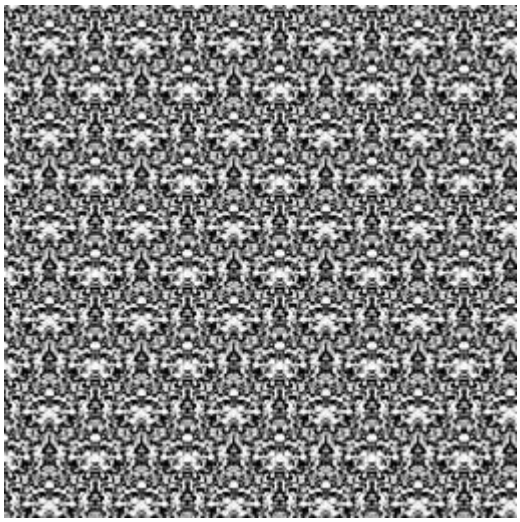

Figure 18 CM_1 of train set

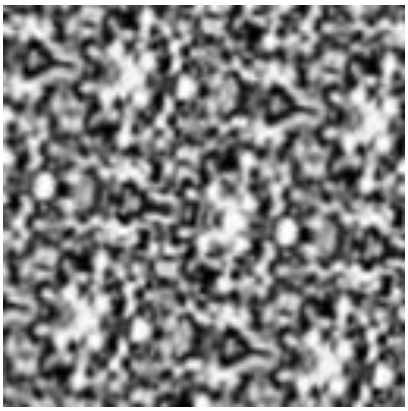

Figure 19 CM_1_aug_1 of train set

And the 5 new images with augmentation is shown below, and the augmentation parameters are shown in the table1.

Figure	Scale	Rotation	X_transition	Y_transition
CM_1_aug_1	up by 1.7085 times	259.32º	7 pixels	9 pixels
CM_1_aug_2	up by 1.5931 times	124.40 º	10 pixels	10 pixels
CM_1_aug_3	up by 1.6022 times	316.12 º	3 pixels	10 pixels
CM_1_aug_4	up by 1.5702 times	71.32 º	20 pixels	3 pixels
CM_1_aug_5	up by 1.9382 times	322.06 º	10 pixels	1 pixels

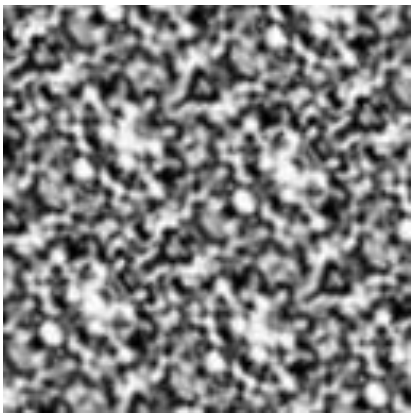

Figure 20 CM_1_aug_2 of train set

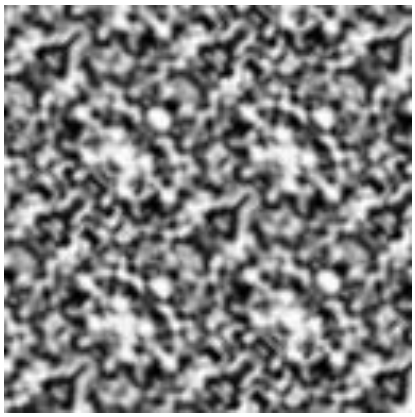

Figure 21 CM_1_aug_3 of train set

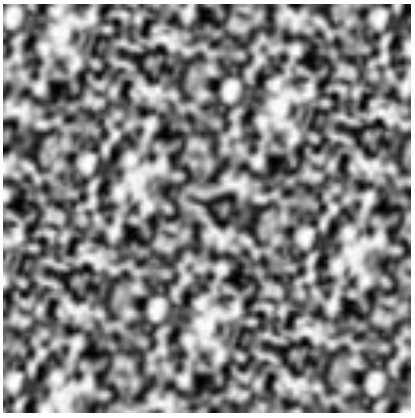

Figure 22 CM_1_aug_4 of train set

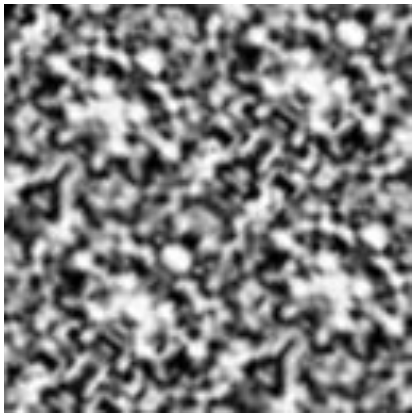

Figure 23 CM_1_aug_5 of train set

The analysis of all the new augmentation parameters including 5 augments on training set and 1 augments on testing sets is shown as following.

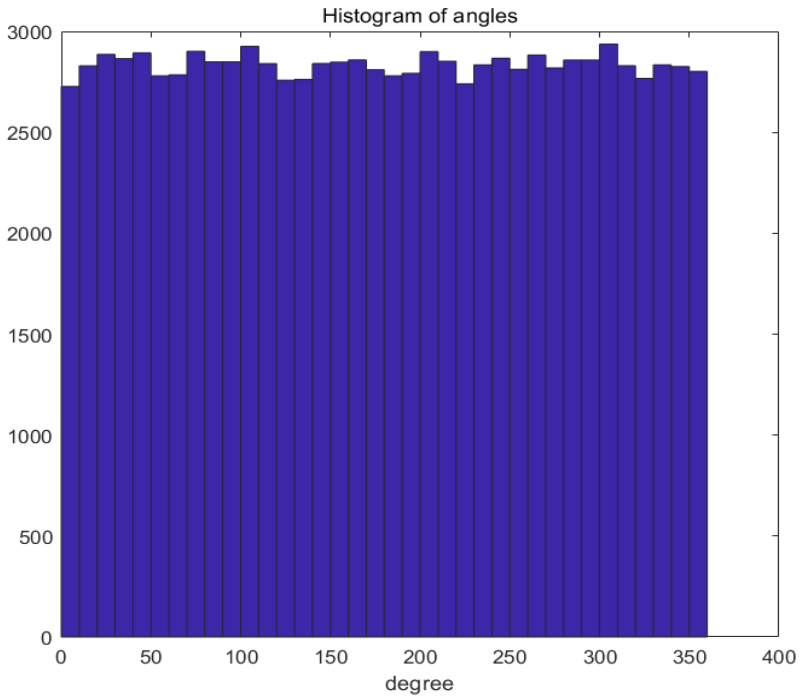

Figure24 Histogram of angles

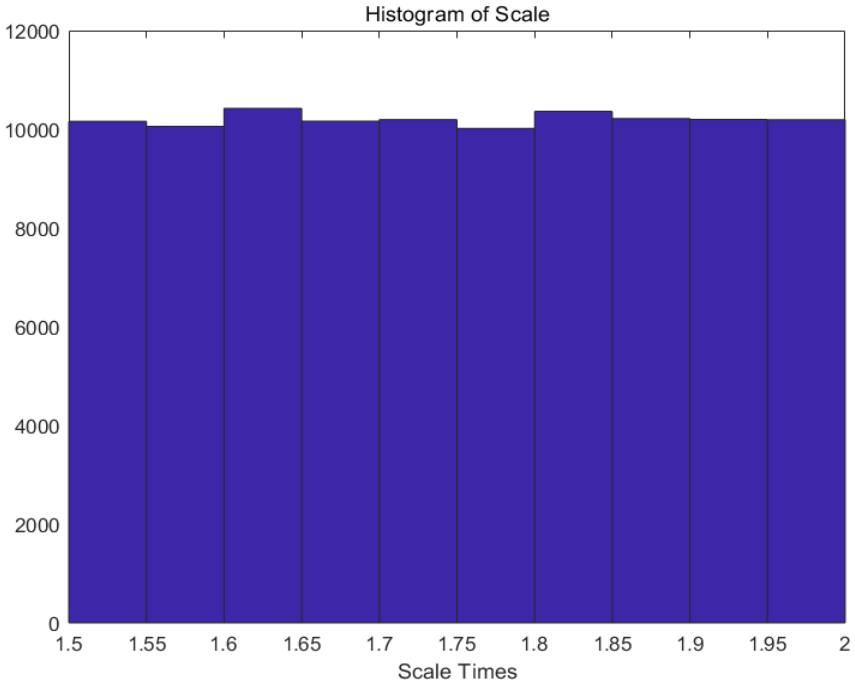

Figure24 Histogram of scale

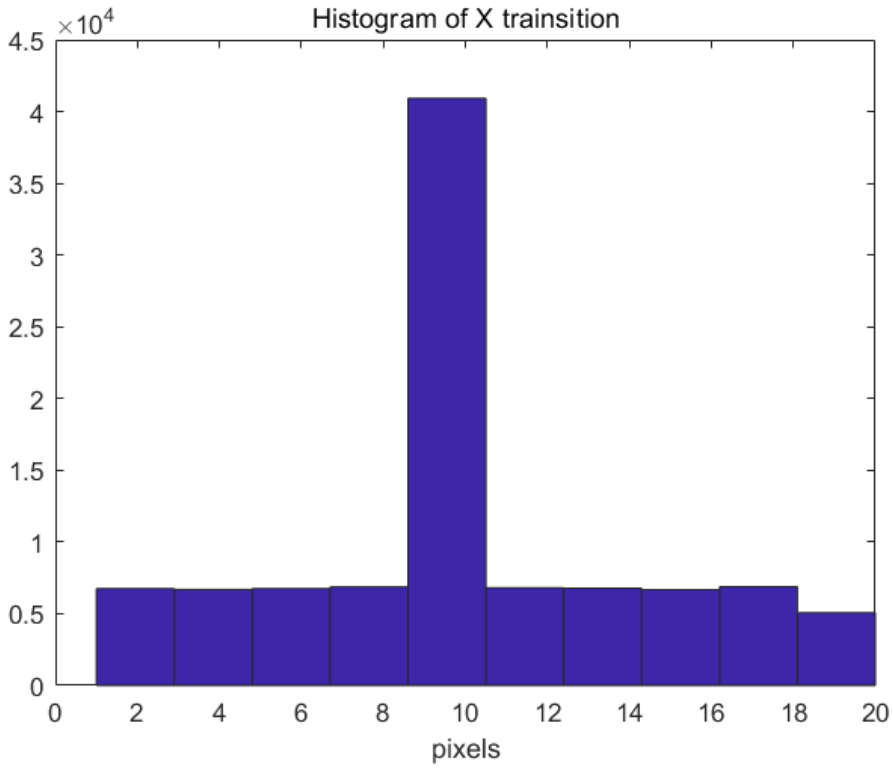

Figure25 Histogram of transition on x-direction

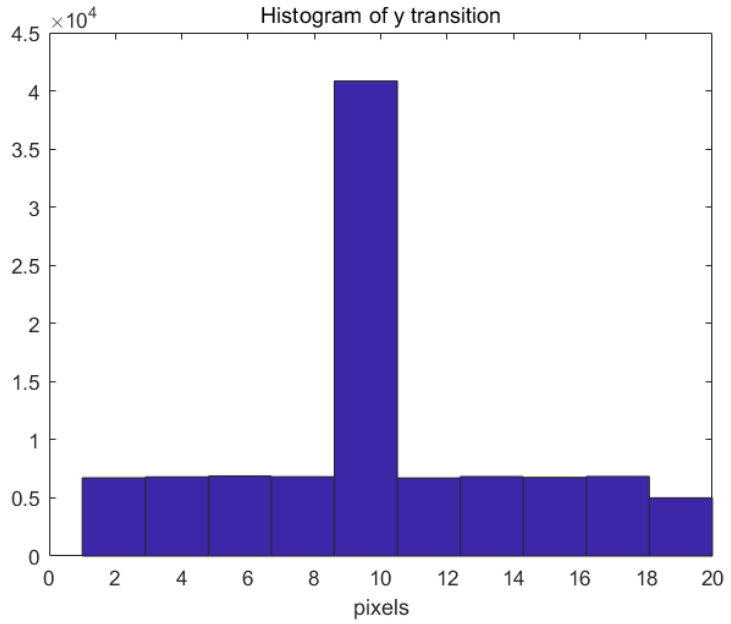

Figure26 Histogram of transition on y-direction

3.2.2 Original network on the Data with Augmentation

Then I try to test my original network on the data with augmentation.

Check the process_original_aug _results.txt file. And we separate the the whole process into 2 parts. The first part is with lr=1e-4 and the second part is with lr=1e-5, and both of these 2 parts have batchsize = 75.

For the first part with lr = 1e-4, the accuracy of training set is 0.2923, and the accuracy of the validate set is 0.2598, and the accuracy of the testing set is 0.2537. This part takes 1901.38 s to finish the whole 20 epochs.

The training accuracy together with the loss figure is shown below.

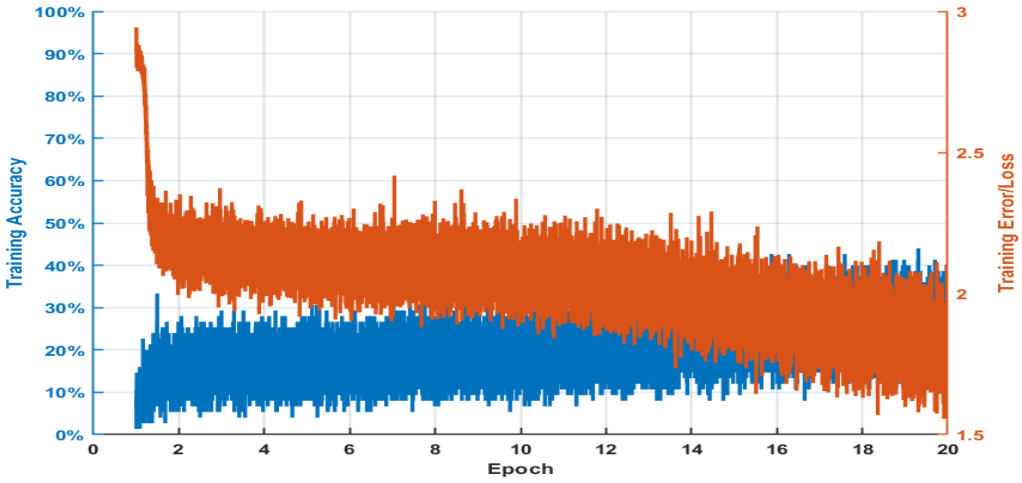

Figure27 training accuracy together with the loss figure of original network batchsize75 lr = 1e-4 (aug)

The confusion matrix for training set in this situation is shown below.

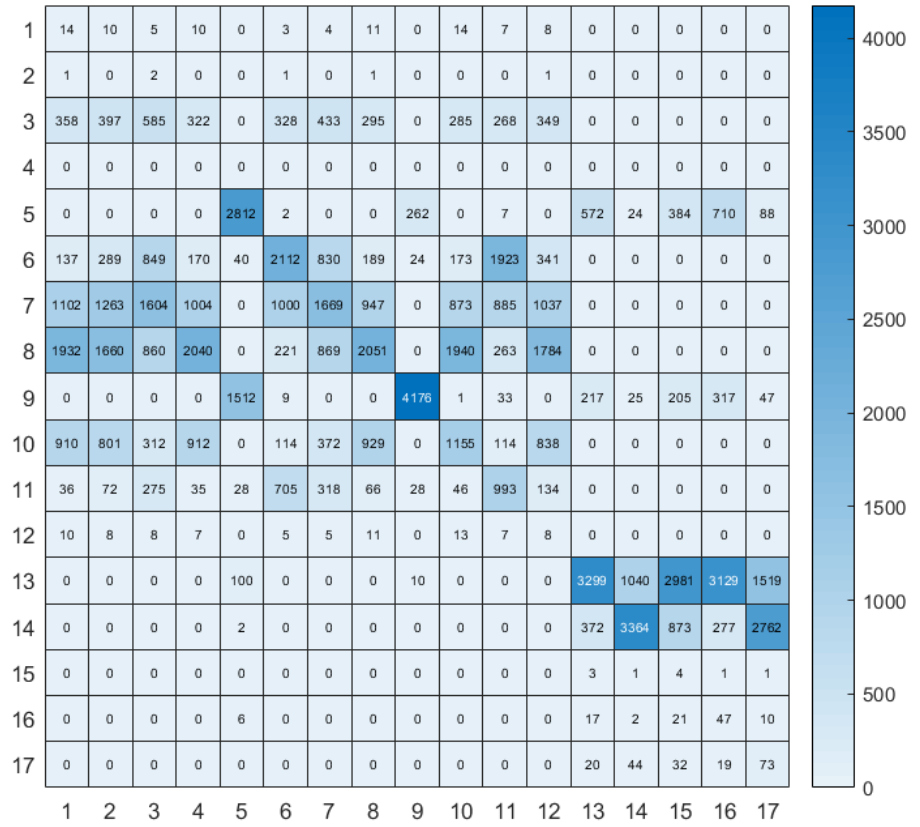

Figure28 confusion matrix for training dataset of original network with batch = 75 lr = 1e-4 (aug)

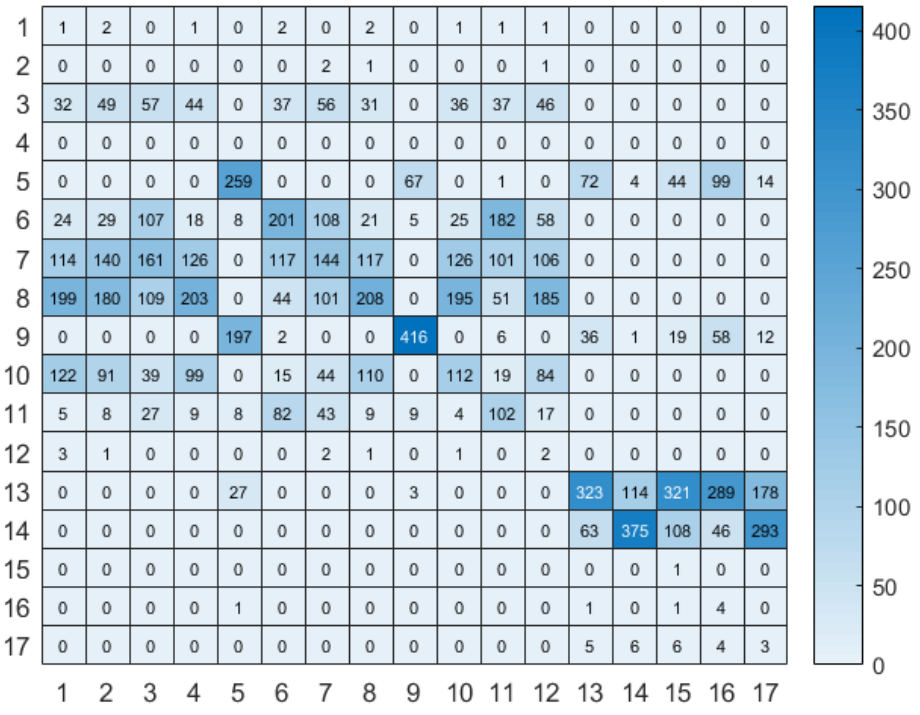

Figure29 confusion matrix for val dataset of original network with batch = 75 lr = 1e-4(aug)

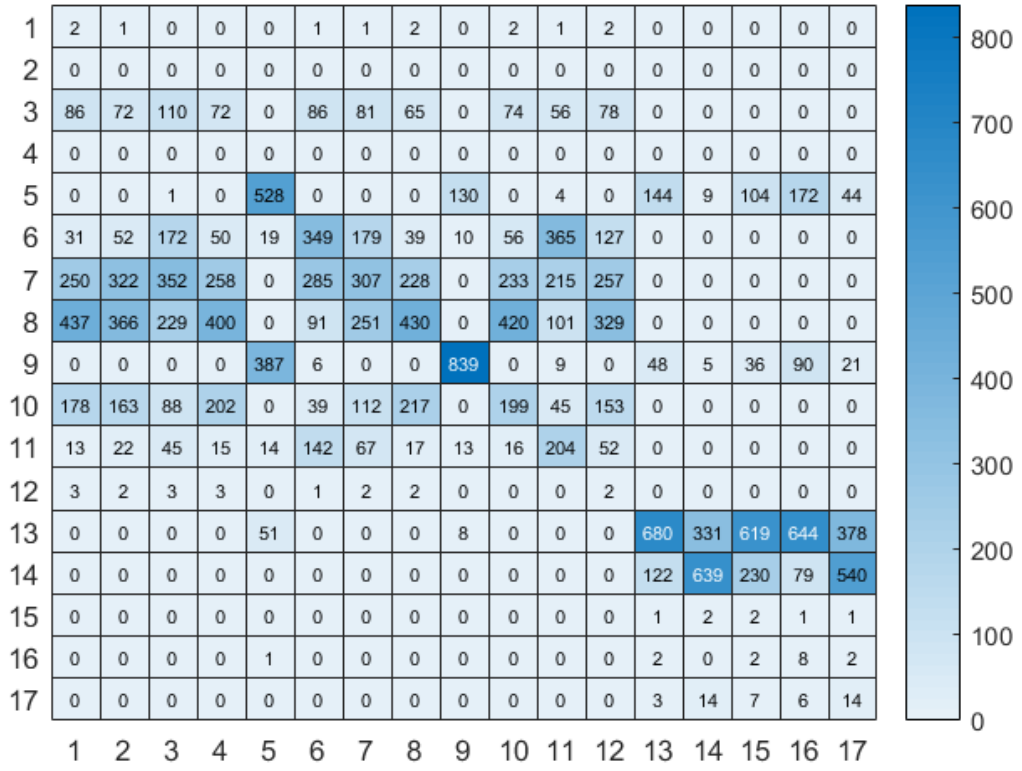

Figure30 confusion matrix for testing dataset of original network with batch = 75 lr = 1e-4(aug)

For the Second part with lr = 1e-5, the accuracy of training set is 0.3286, and the accuracy of the validate set is 0.2749, and the accuracy of the testing set is 0.2728. This part takes 717.65 s to finish the whole 20 epochs.

The training accuracy together with the loss figure is shown below.

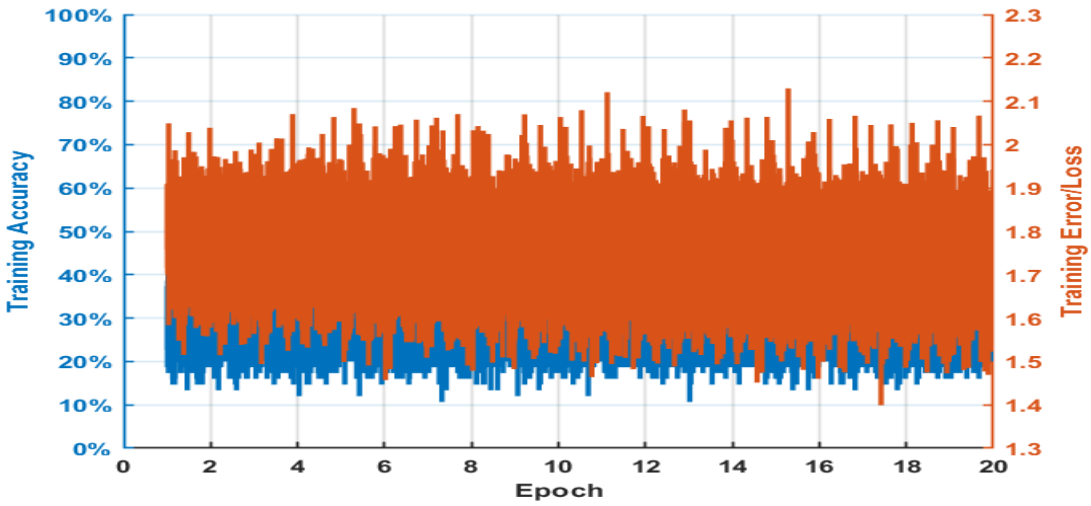

Figure31 training accuracy together with the loss figure of original network batchsize75 lr = 1e-5 (aug)

The confusion matrix for training set in this situation is shown below.

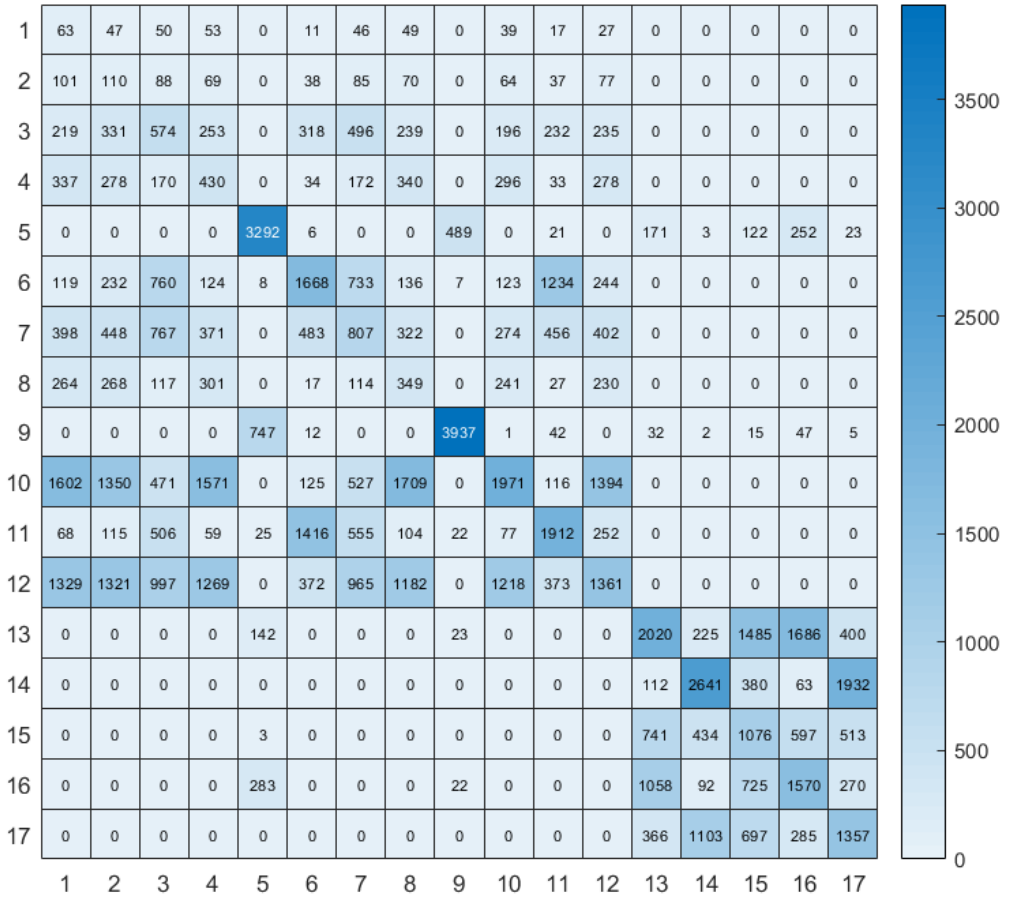

Figure32 confusion matrix for training dataset of original network with batch = 75 lr = 1e-5 (aug)

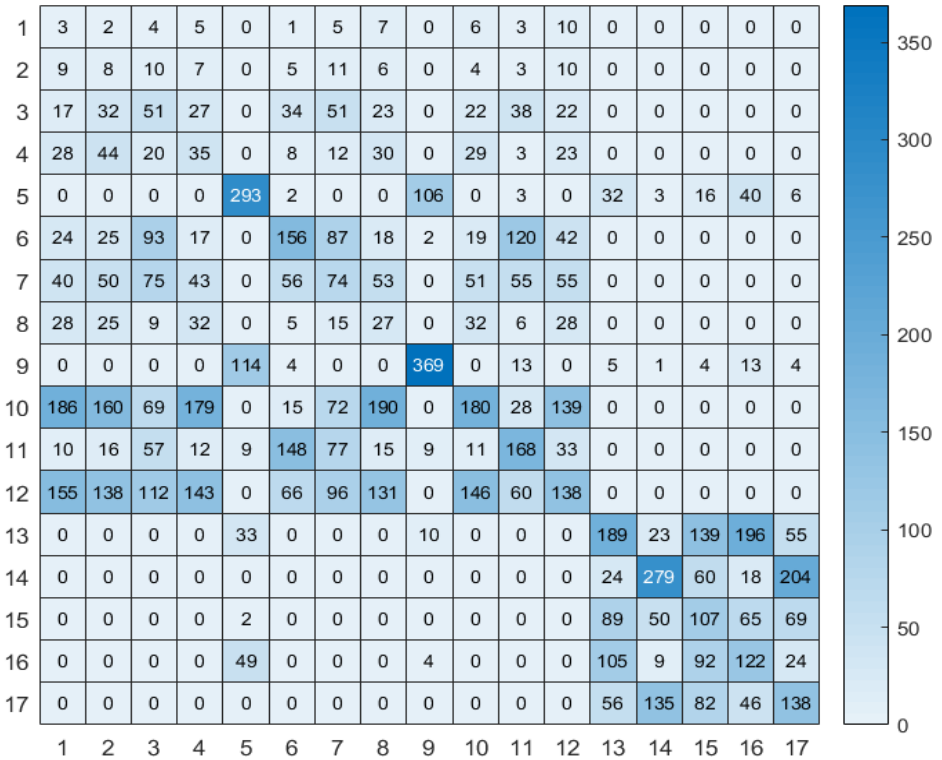

Figure33 confusion matrix for val dataset of original network with batch = 75 lr = 1e-5(aug)

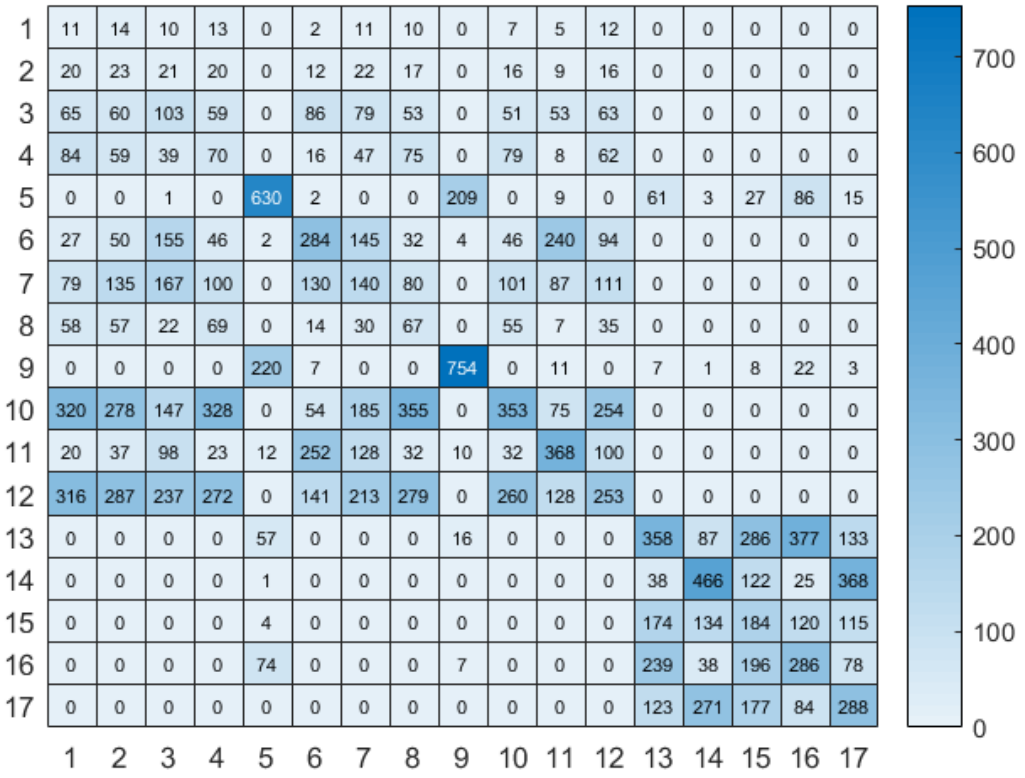

Figure34 confusion matrix for testing dataset of original network with batch = 75 lr = 1e-5(aug)

It seems the original network cannot work on the data with augmentation. So we design the skinny network and wide network to work on the data.

3.3 Results on Step 3

3.3.1 Results of Skinny Network

(1) For augmentation data

Check the process_skinny_network_aug.txt file. And we separate the the whole process into 2 parts. The first part is with lr=1e-4 and the second part is with lr=1e-5, and both of these 2 parts have batchsize = 75.

For the first part with lr = 1e-4, the accuracy of training set is 0.5309, and the accuracy of the validate set is 0.5096, and the accuracy of the testing set is 0.5091. This part takes 5735.38 s to finish the whole 30 epochs.

The training accuracy together with the loss figure is shown below.

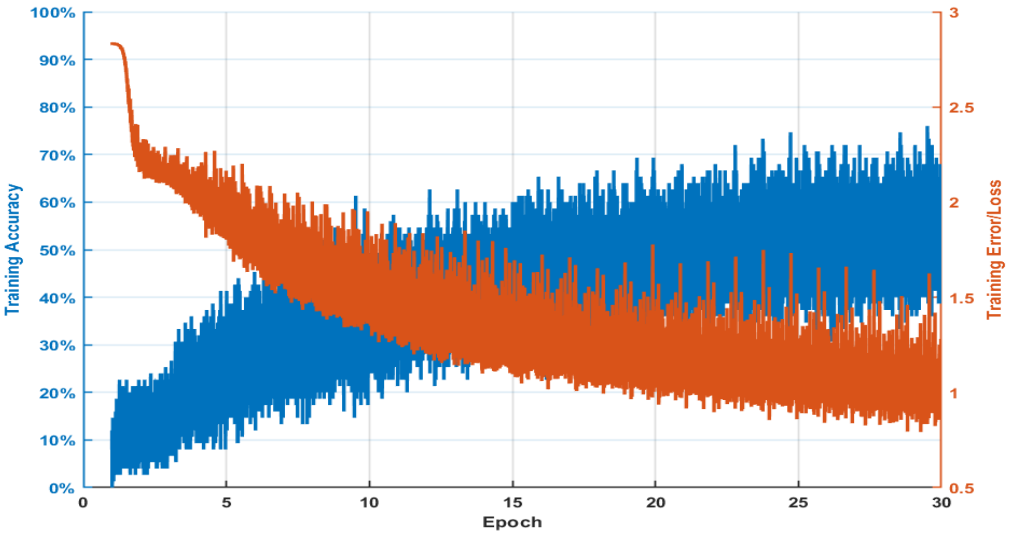

Figure35 training accuracy together with the loss figure of skinny network batchsize75 lr = 1e-4 (aug)

The confusion matrix for training set in this situation is shown below.

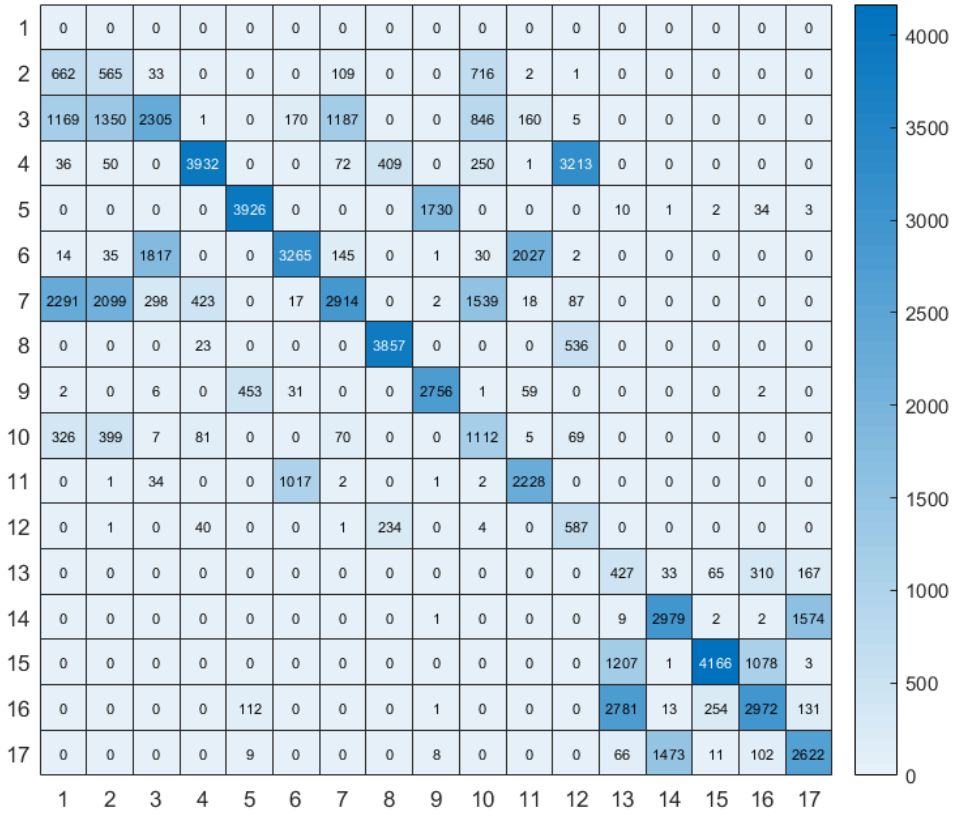

Figure36 confusion matrix for training dataset of skinny network with batch = 75 lr = 1e-4 (aug)

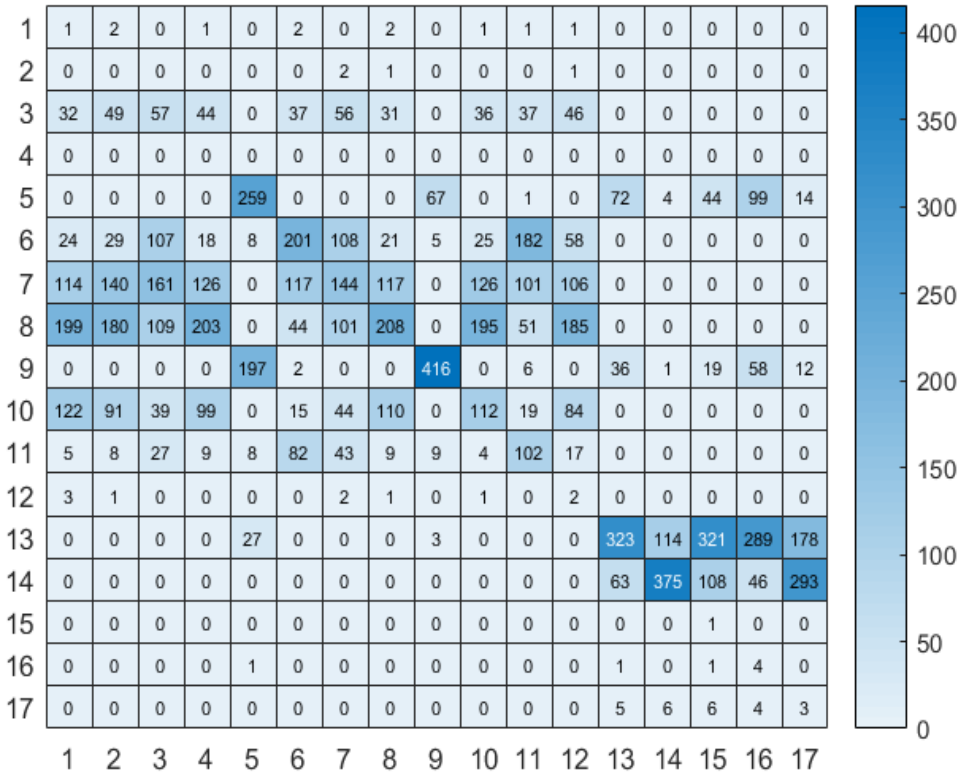

Figure37 confusion matrix for val dataset of skinny network with batch = 75 lr = 1e-4(aug)

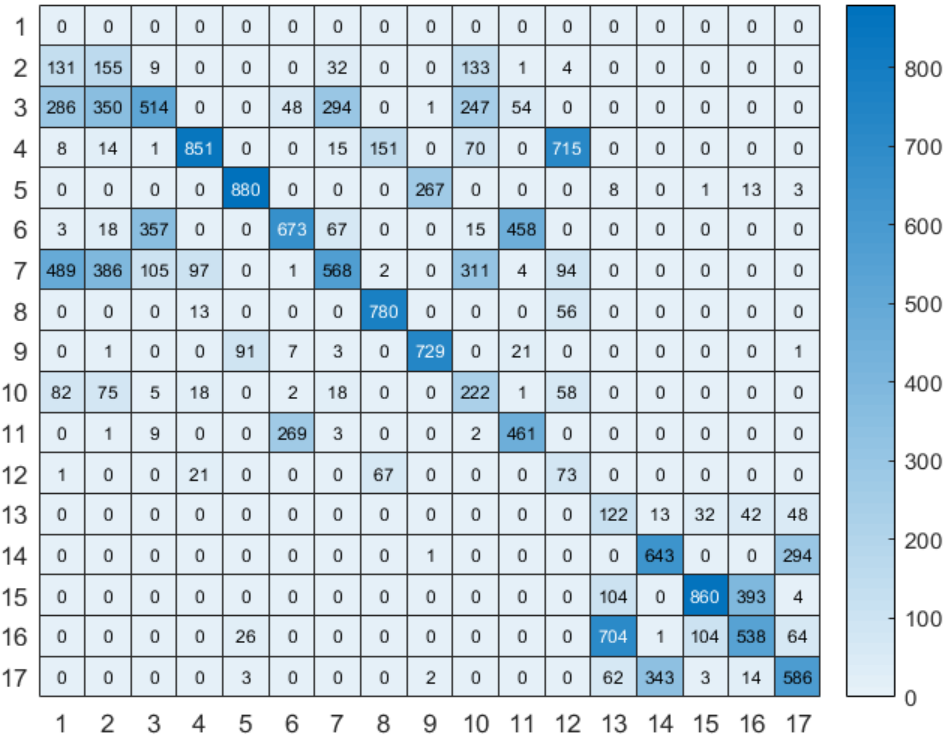

Figure38 confusion matrix for testing dataset of skinny network with batch = 75 lr = 1e-4(aug)

For the Second part with lr = 1e-5, the accuracy of training set is 0.6190, and the accuracy of the validate set is 0.5852, and the accuracy of the testing set is 0.5762. This

part takes 4806.36 s to finish the whole 30 epochs.

The training accuracy together with the loss figure is shown below.

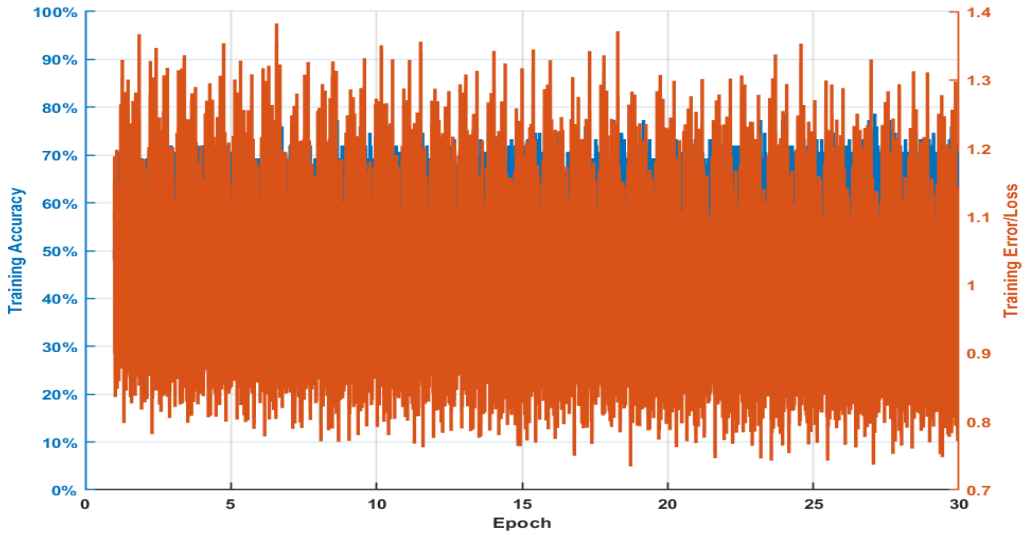

Figure39 training accuracy together with the loss figure of skinny network batchsize75 lr = 1e-5 (aug)

The confusion matrix for training set in this situation is shown below.

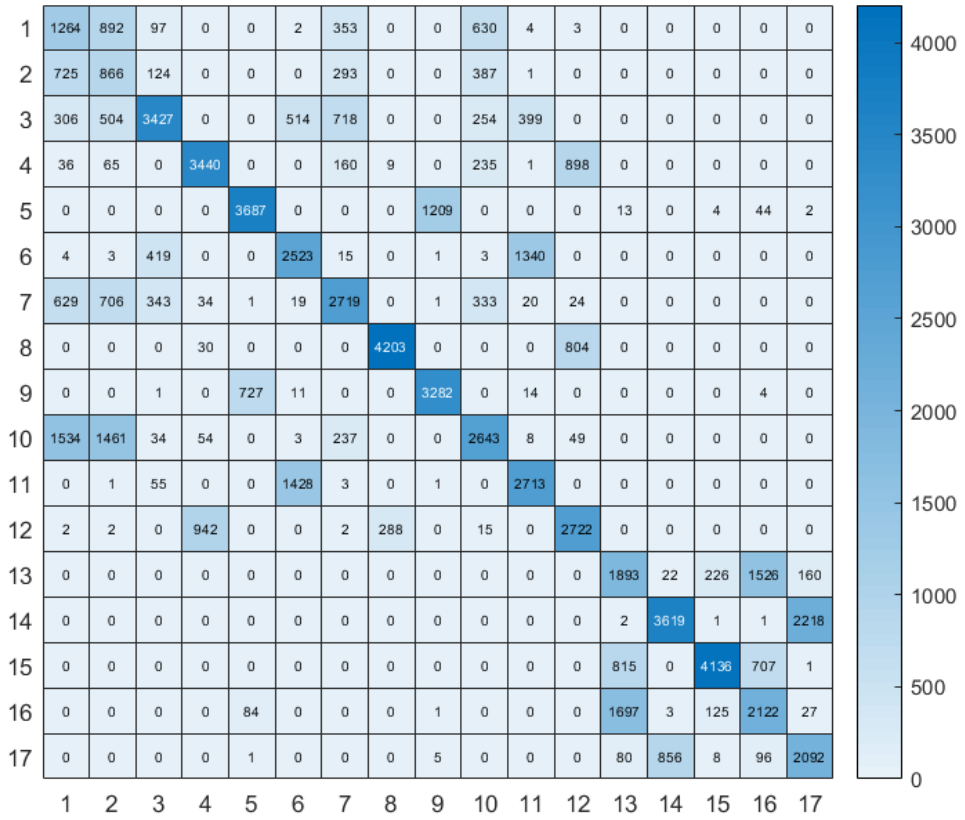

Figure40 confusion matrix for training dataset of skinny network with batch = 75 lr = 1e-5 (aug)

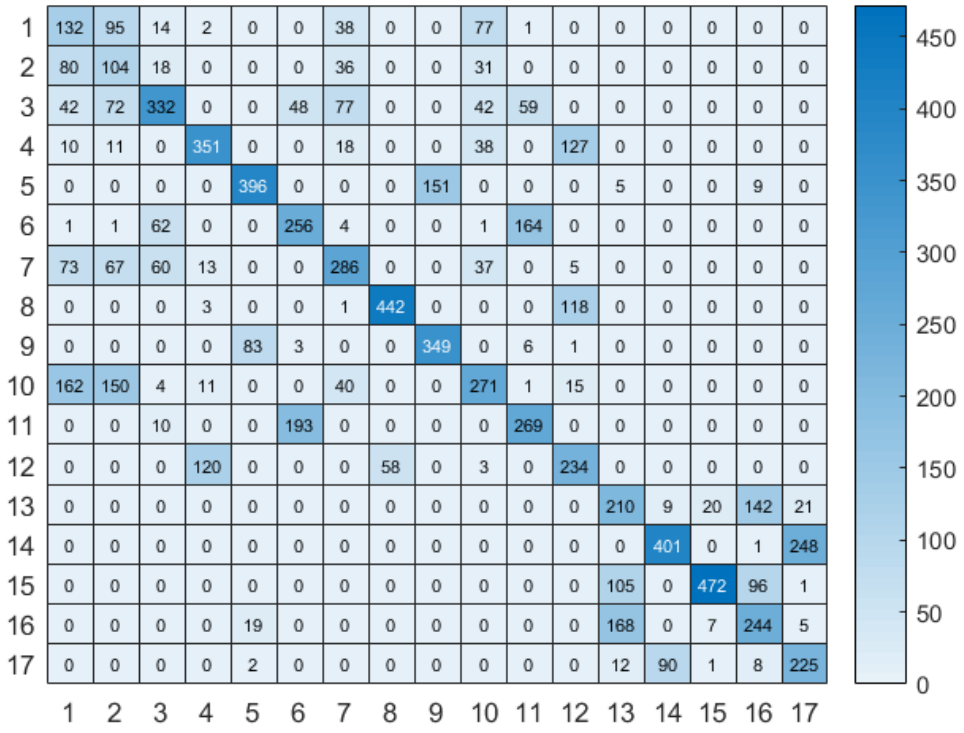

Figure41 confusion matrix for val dataset of skinny network with batch = 75 lr = 1e-5(aug)

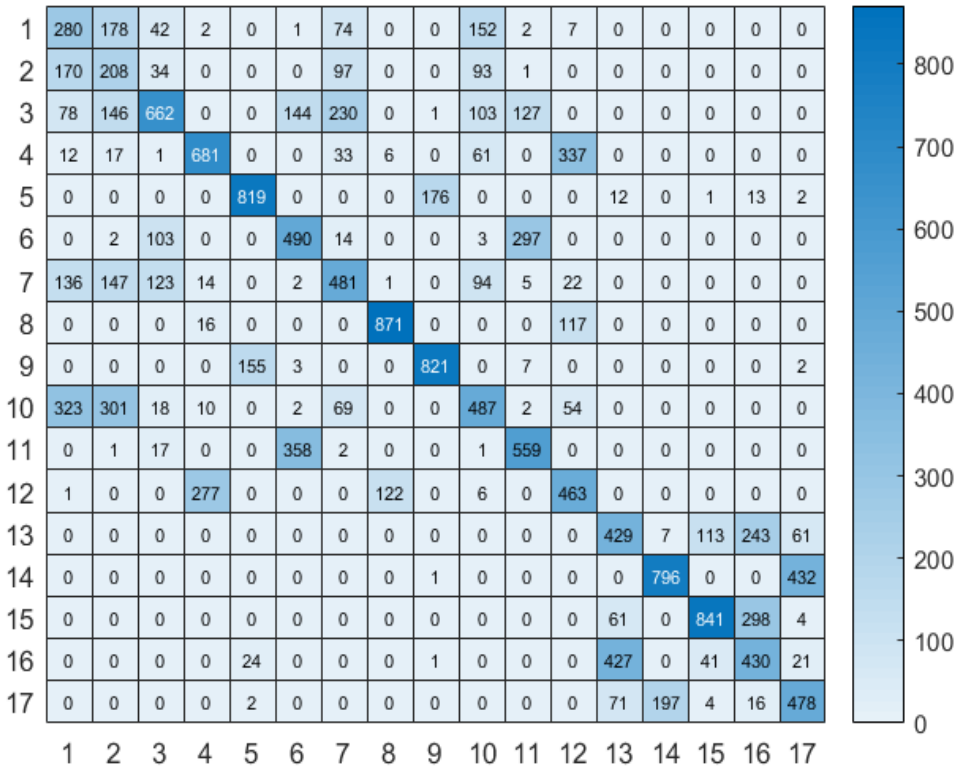

Figure42 confusion matrix for testing dataset of skinny network with batch = 75 lr = 1e-5(aug)

Here is the visualization of the first layer of the skinny network on the data with augmentation.

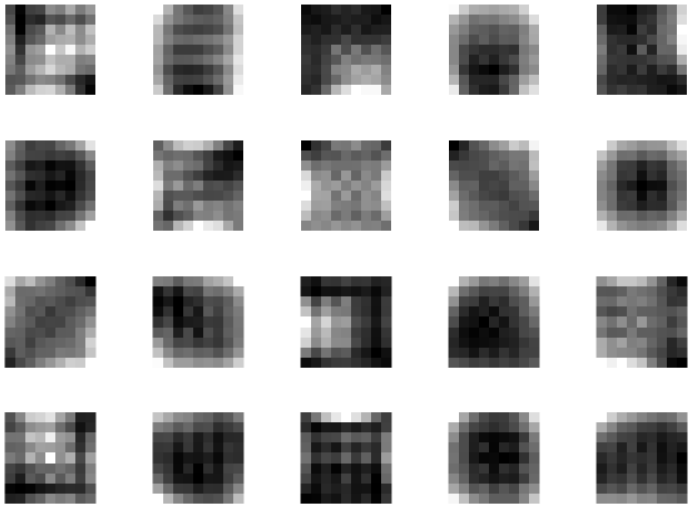

Figure43 visualization of the first layer of skinny network(aug)

Here is the visualization2 of the t-SNE multidimensional reduction on the fully connected layer activations of your network trained on the augmentations for both test val and test set for the data with augmentation.

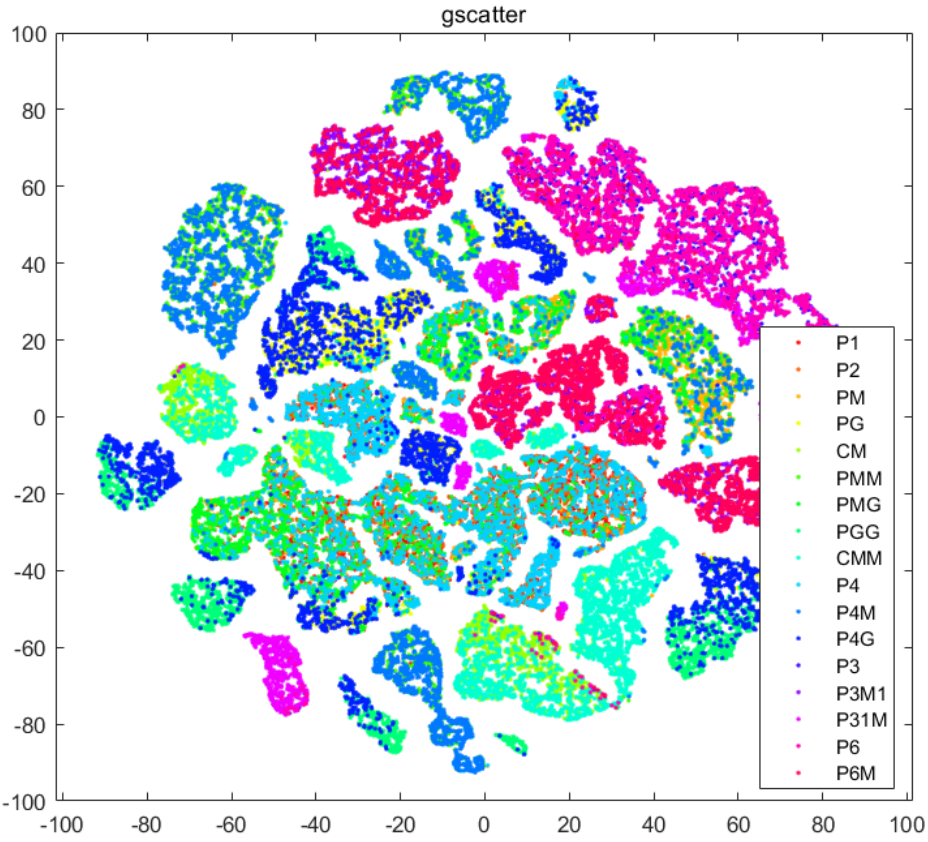

Figure44 visualization2 of the t-SNE of fully connected layer of skinny network on trainset(aug)

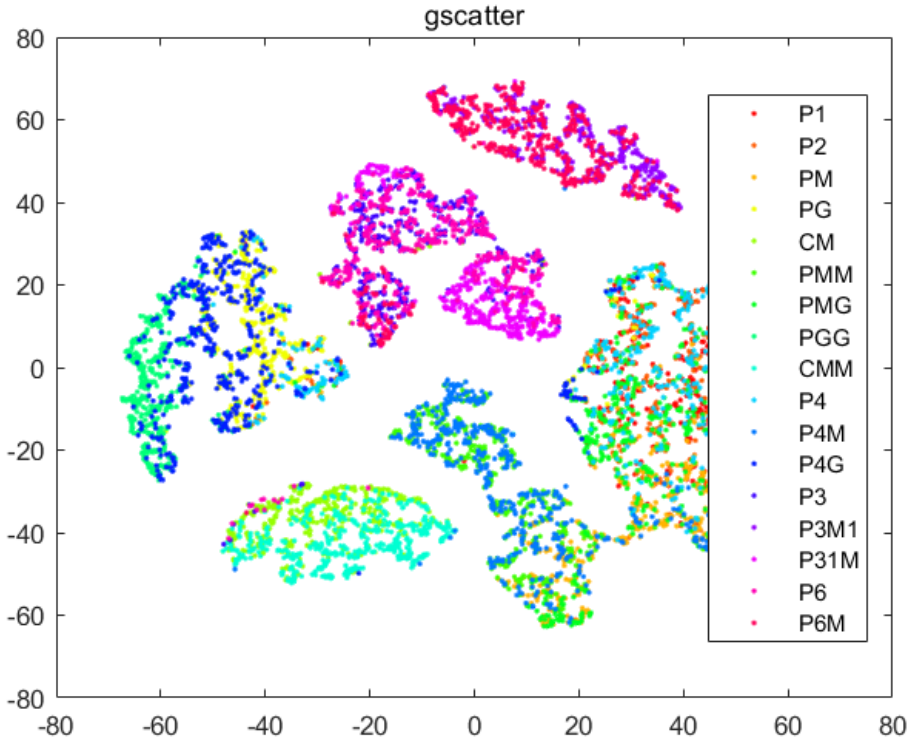

Figure45 visualization2 of the t-SNE of fully connected layer of skinny network on valset(aug)

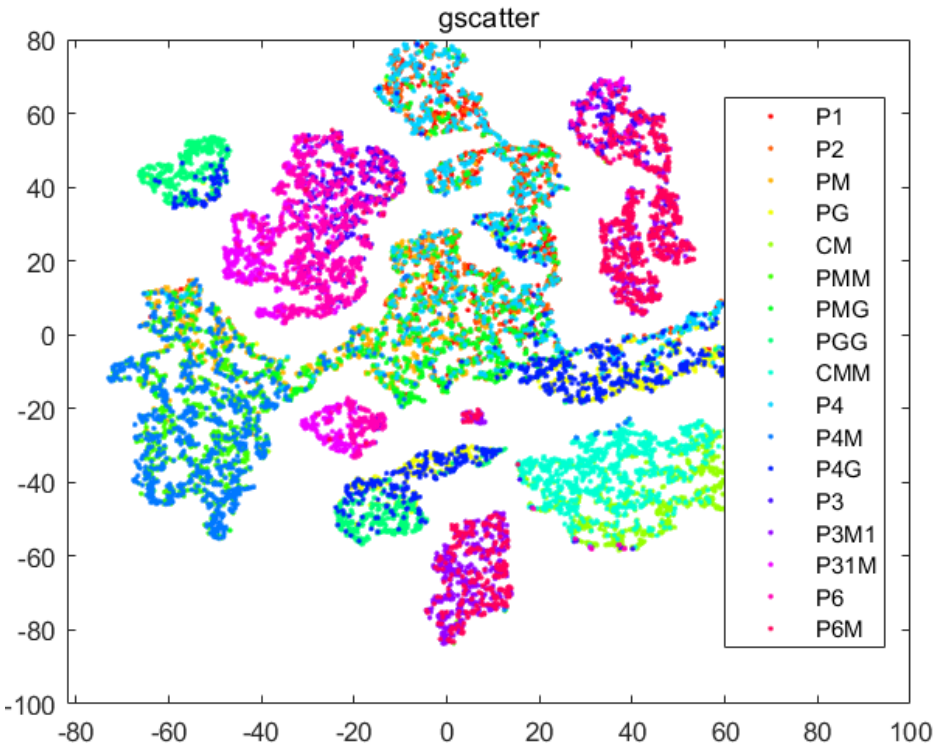

Figure46 visualization2 of the t-SNE of fully connected layer of skinny network on testset(aug)

(2) For orginal data

Check the process_skinny_network.txt file. And we separate the the whole process into 2 parts. The first part is with lr=1e-4 and the second part is with lr=1e-5, and both of these 2 parts have batchsize = 75.

For the first part with lr = 1e-4, the accuracy of training set is 0.9322, and the accuracy of the validate set is 0.8506, and the accuracy of the testing set is 0.8536. This part takes 1841.06 s to finish the whole 30 epochs.

The training accuracy together with the loss figure is shown below.

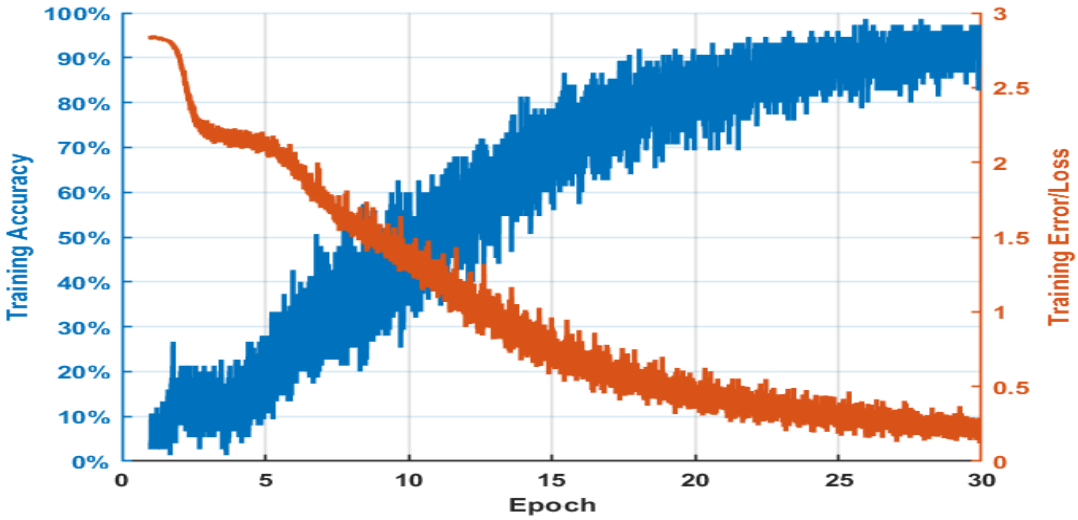

Figure47 training accuracy together with the loss figure of skinny network batchsize75 lr = 1e-4

The confusion matrix for training set in this situation is shown below.

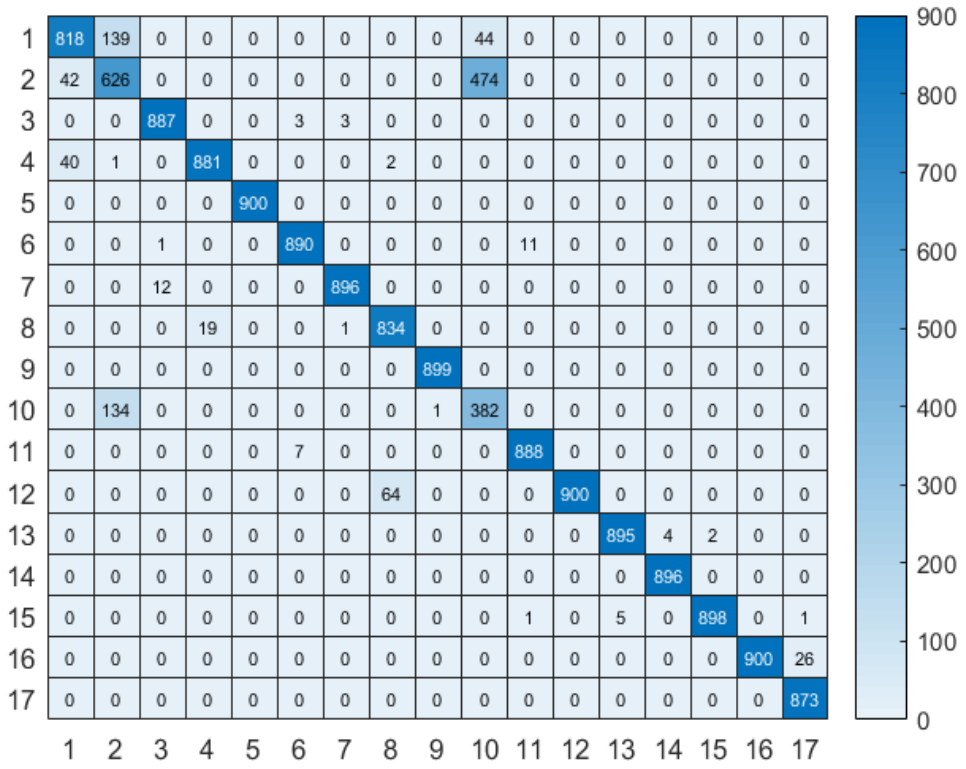

Figure48 confusion matrix for training dataset of skinny network with batch = 75 lr = 1e-4

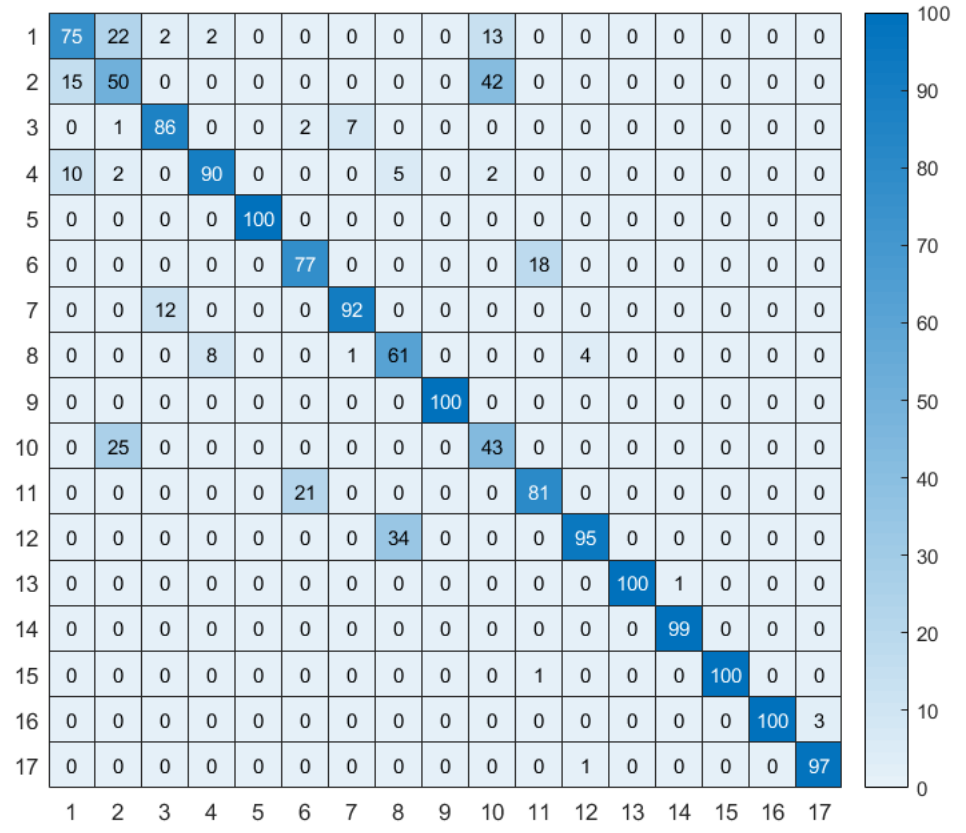

Figure49 confusion matrix for val dataset of skinny network with batch = 75 lr = 1e-4

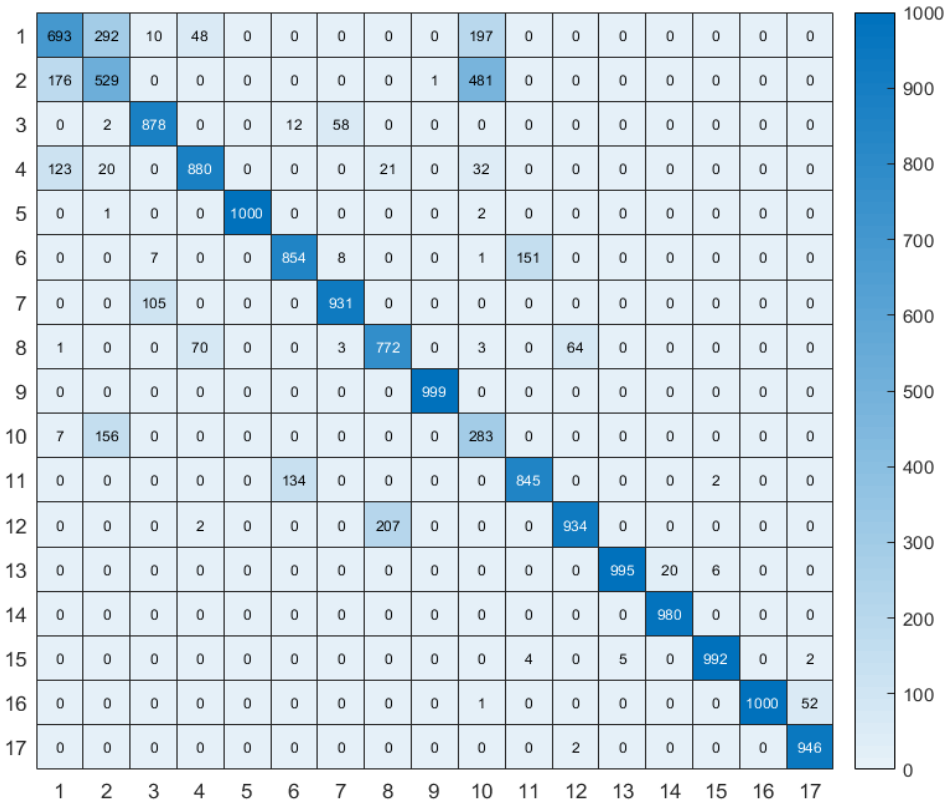

Figure50 confusion matrix for testing dataset of skinny network with batch = 75 lr = 1e-4

For the Second part with lr = 1e-5, the accuracy of training set is 0.9673, and the accuracy of the validate set is 0.8729, and the accuracy of the testing set is 0.5762. This part takes 1843.82 s to finish the whole 30 epochs.

The training accuracy together with the loss figure is shown below.

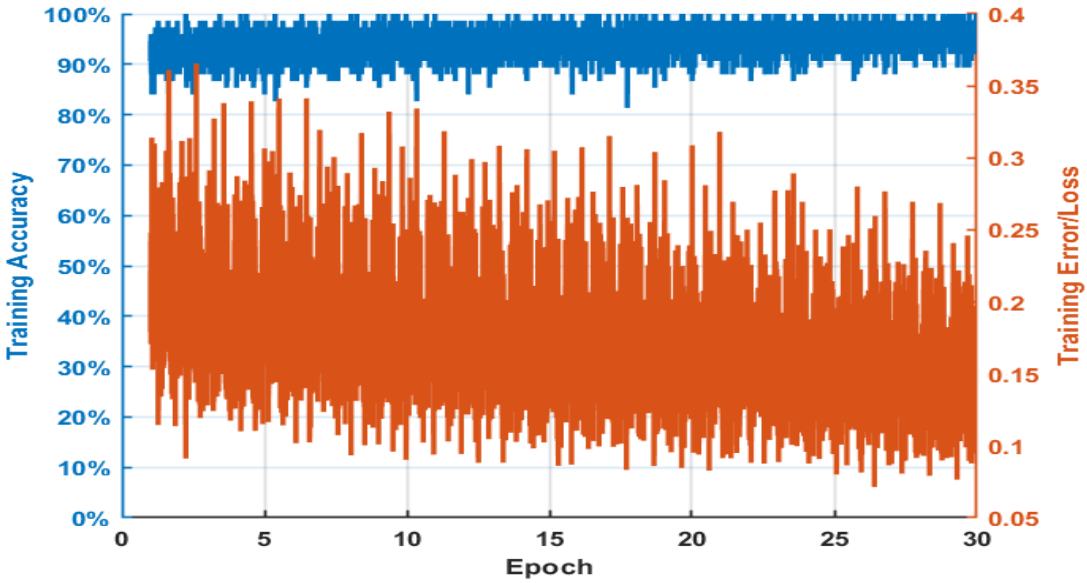

Figure51 training accuracy together with the loss figure of skinny network batchsize75 lr = 1e-5

The confusion matrix for training set in this situation is shown below.

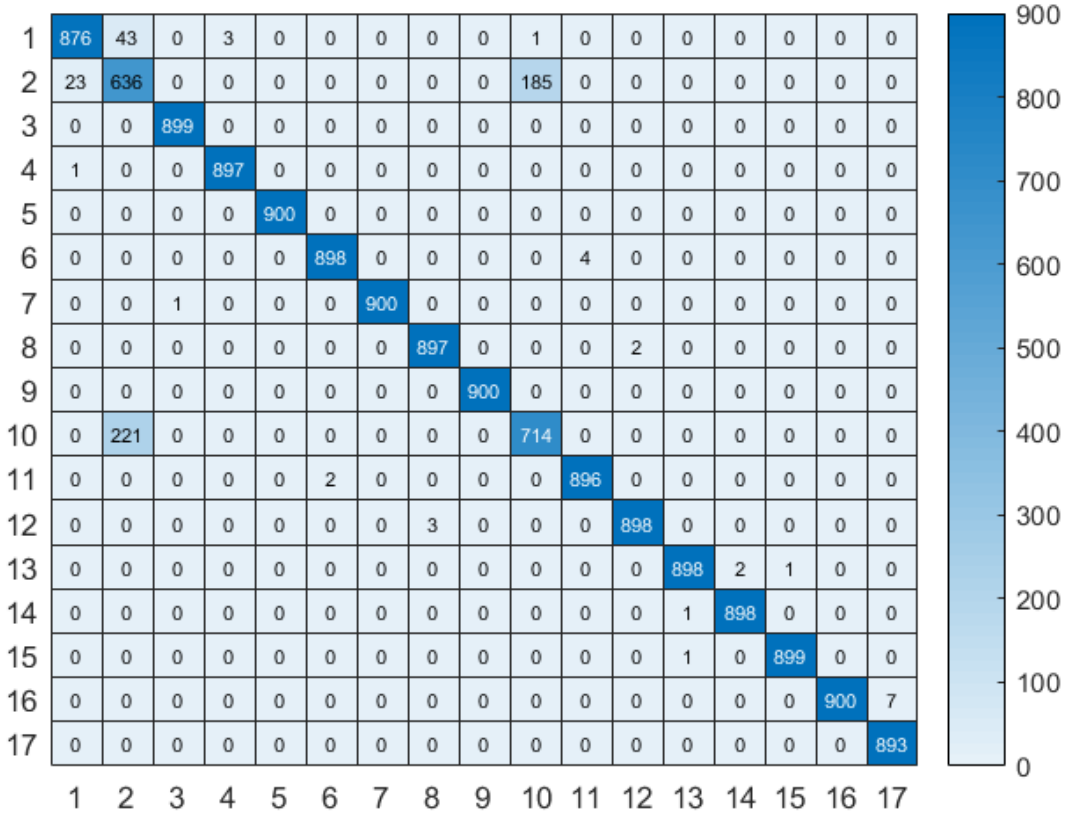

Figure52 confusion matrix for training dataset of skinny network with batch = 75 lr = 1e-5

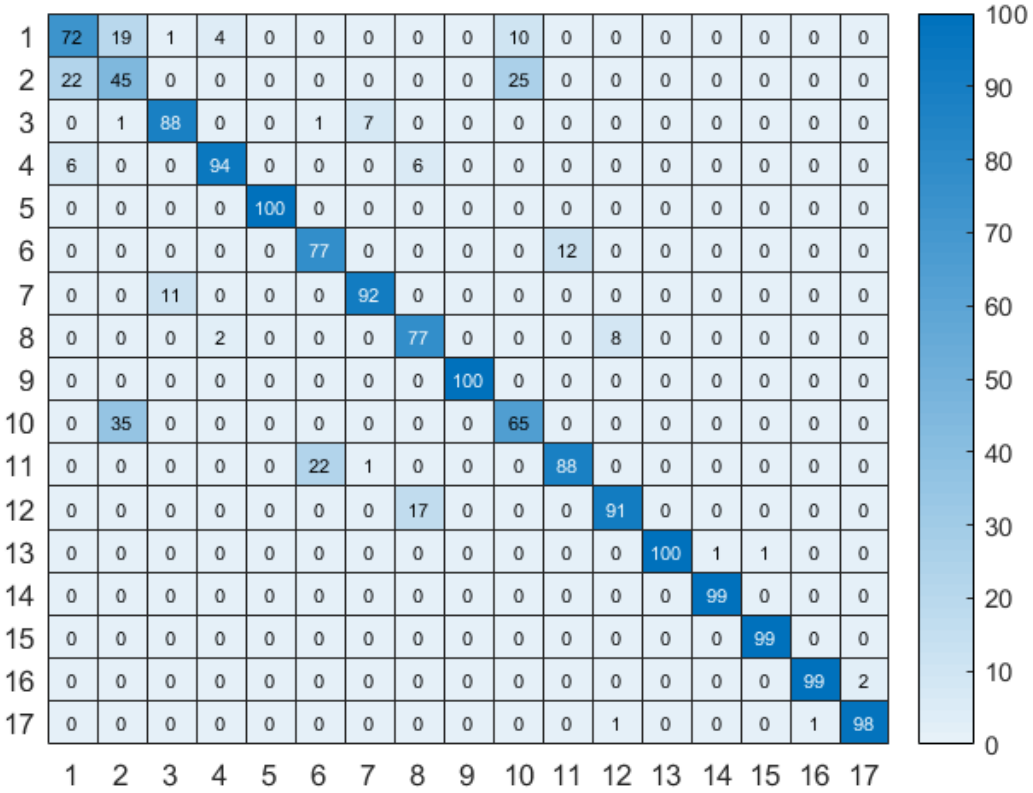

Figure53 confusion matrix for val dataset of skinny network with batch = 75 lr = 1e-5

Here is the visualization of the first layer of the skinny network on the original data.

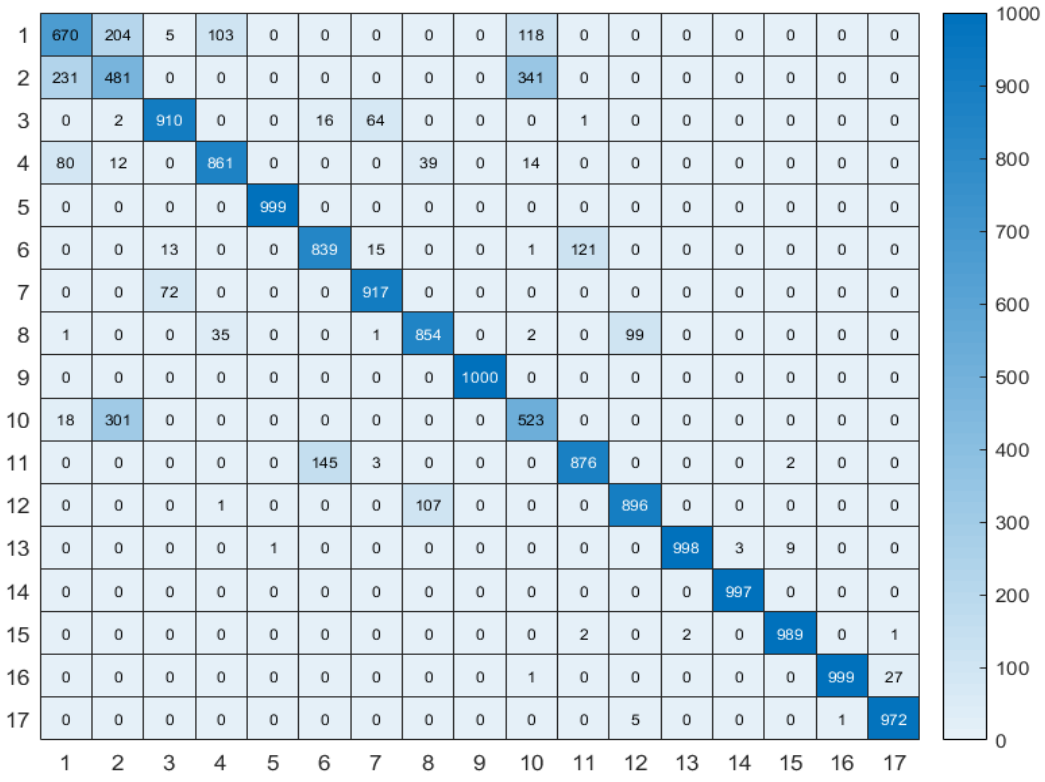

Figure54 confusion matrix for testing dataset of skinny network with batch = 75 lr = 1e-5

Here is the visualization of the first layer of the skinny network on the data with augmentation.

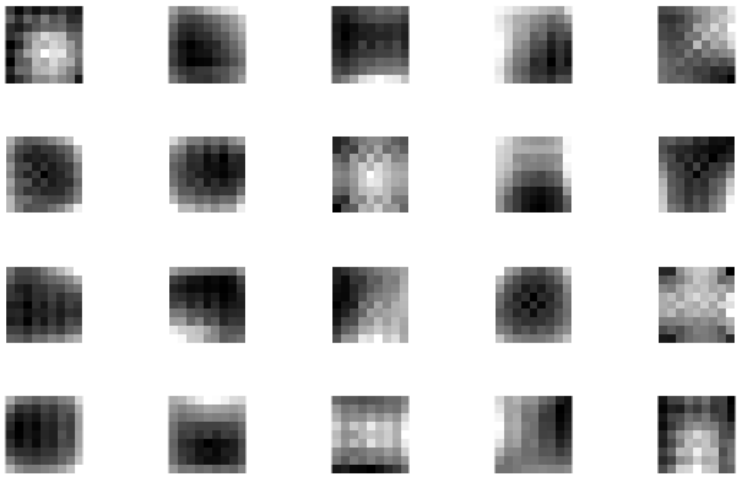

Figure55 visualization of the first layer of skinny network

Here is the visualization2 of the t-SNE multidimensional reduction on the fully connected layer activations of your network trained on the augmentations for both test val and test set for the data with augmentation.

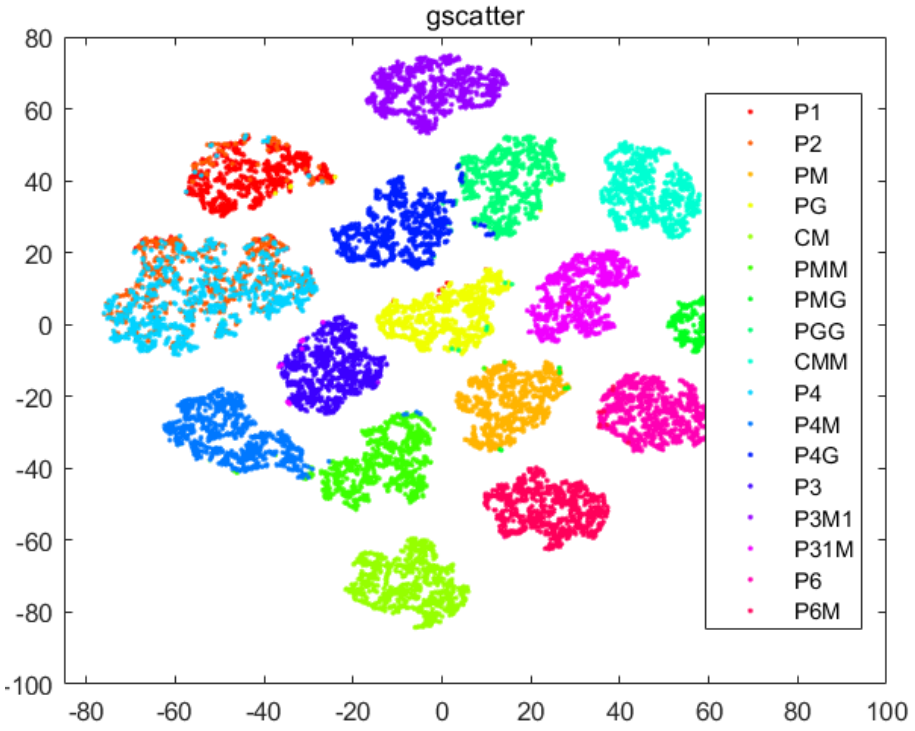

Figure56 visualization2 of the t-SNE of fully connected layer of skinny network on trainset

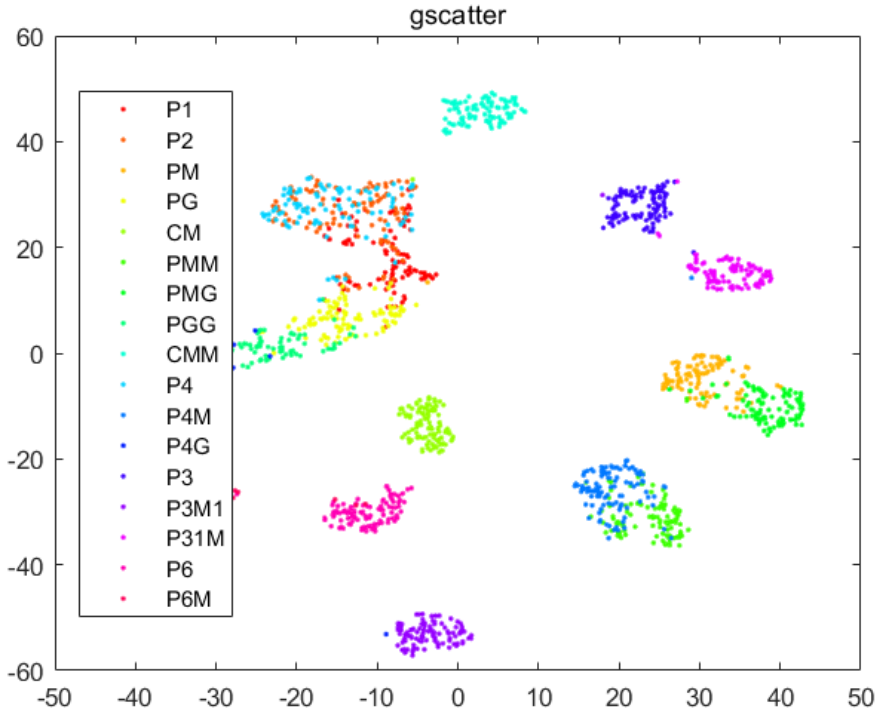

Figure57 visualization2 of the t-SNE of fully connected layer of skinny network on valset

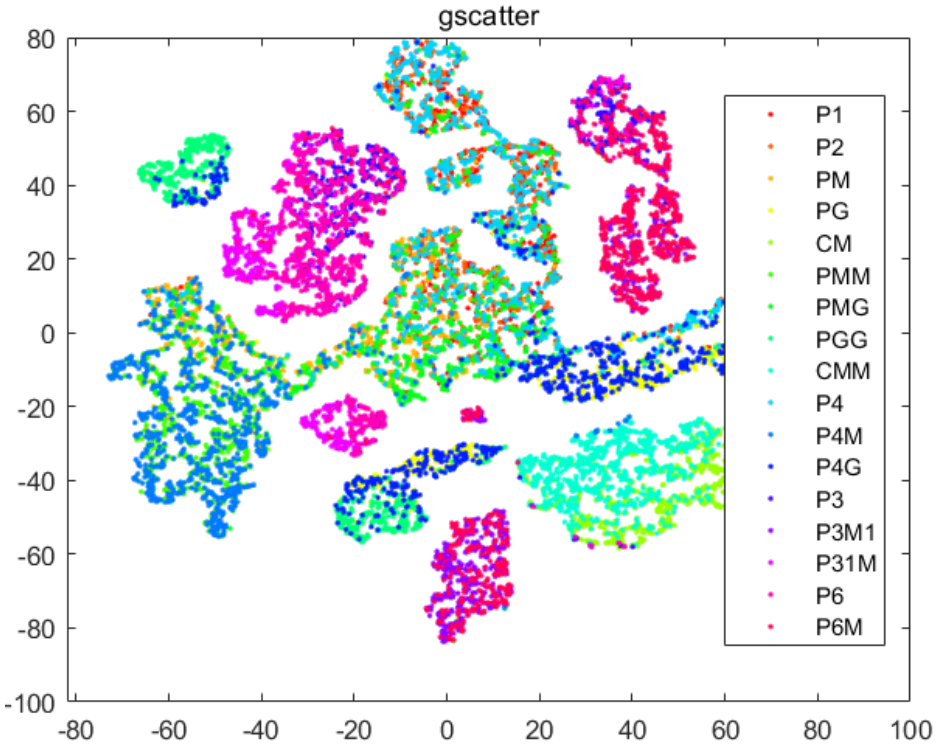

Figure58 visualization2 of the t-SNE of fully connected layer of skinny network on testset

3.3.3 Results of Wide Network

(1) For augmentation data

Check the process_wide_network_aug.txt file. And we separate the the whole process into 2 parts. The first part is with lr=1e-3 and the second part is with lr=5e-5, and both of these 2 parts have batchsize = 75.

For the first part with lr = 1e-3, the accuracy of training set is 0.5405, and the accuracy of the validate set is 0.4238, and the accuracy of the testing set is 0.4247. This part takes 17978.75 s to finish the whole 30 epochs.

The training accuracy together with the loss figure is shown below.

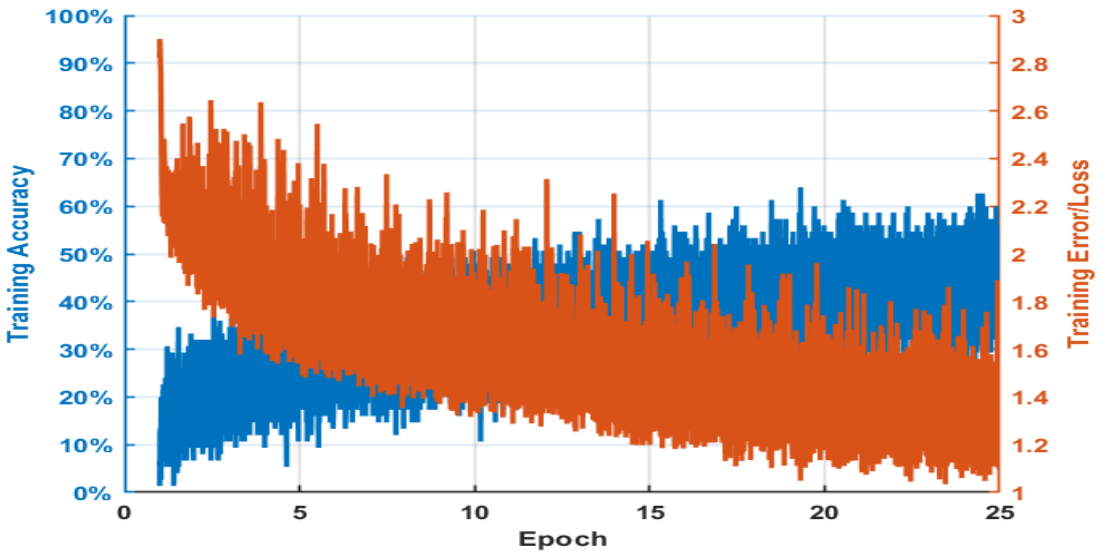

Figure59 training accuracy together with the loss figure of wide network batchsize75 lr = 1e-3 (aug)

The confusion matrix for training set in this situation is shown below.

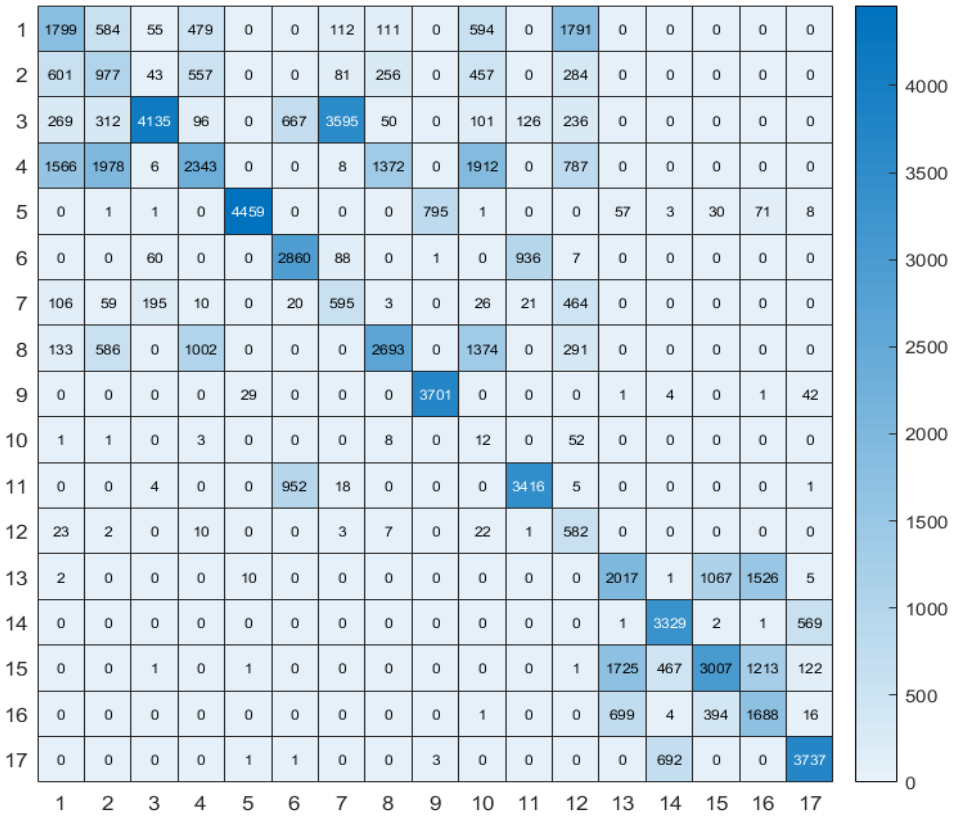

Figure60 confusion matrix for training dataset of wide network with batch = 75 lr = 1e-3 (aug)

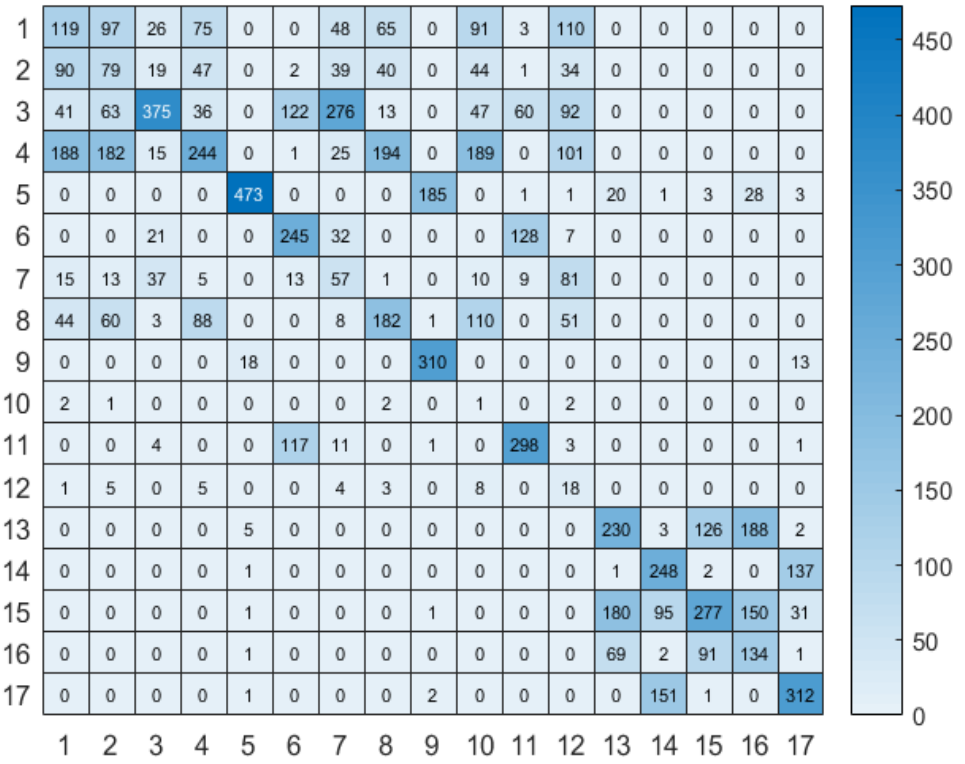

Figure61 confusion matrix for val dataset of wide network with batch = 75 lr = 1e-3(aug)

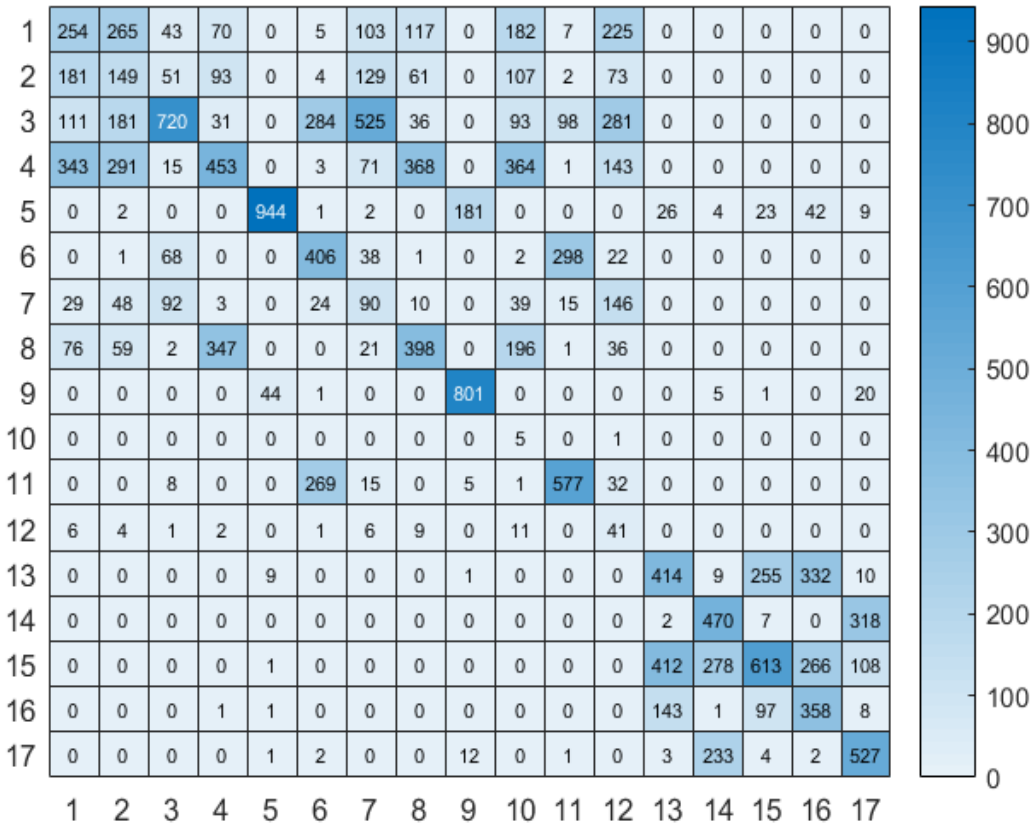

Figure62 confusion matrix for testing dataset of wide network with batch = 75 lr = 1e-3(aug)

For the Second part with lr = 5e-4, the accuracy of training set is 0.6052, and the

accuracy of the validate set is 0.4331, and the accuracy of the testing set is 0.4348. This part takes 3985.62 s to finish the whole 5 epochs.

The training accuracy together with the loss figure is shown below.

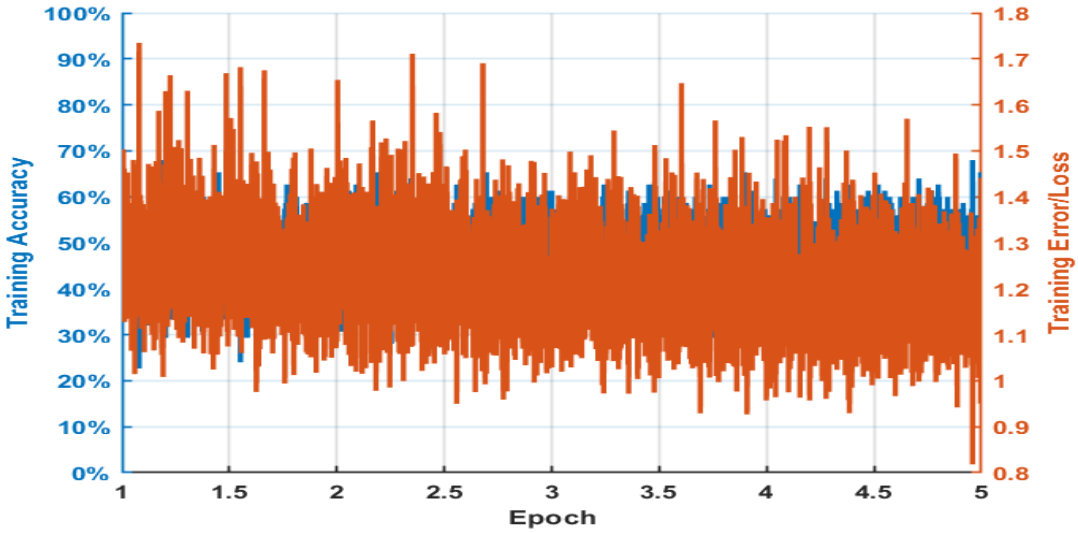

Figure63 training accuracy together with the loss figure of wide network batchsize75 lr = 5e-4 (aug)

The confusion matrix for training set in this situation is shown below.

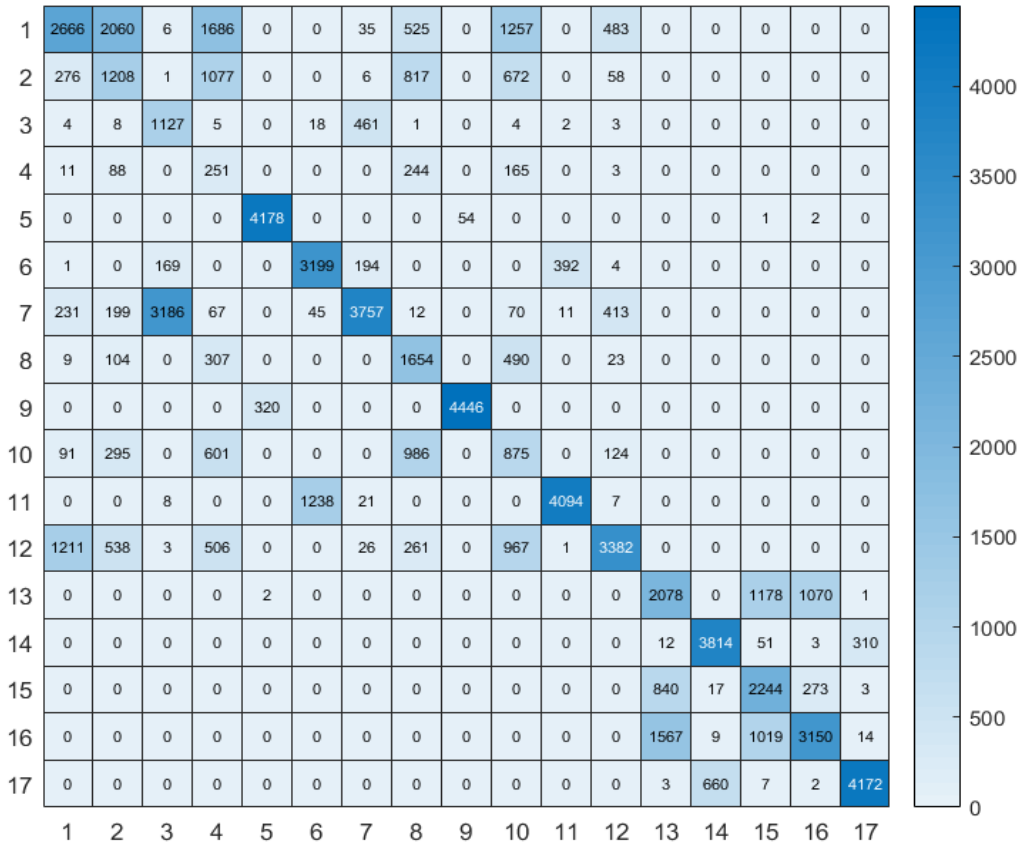

Figure64 confusion matrix for training dataset of wide network with batch = 75 lr = 5e-4 (aug)

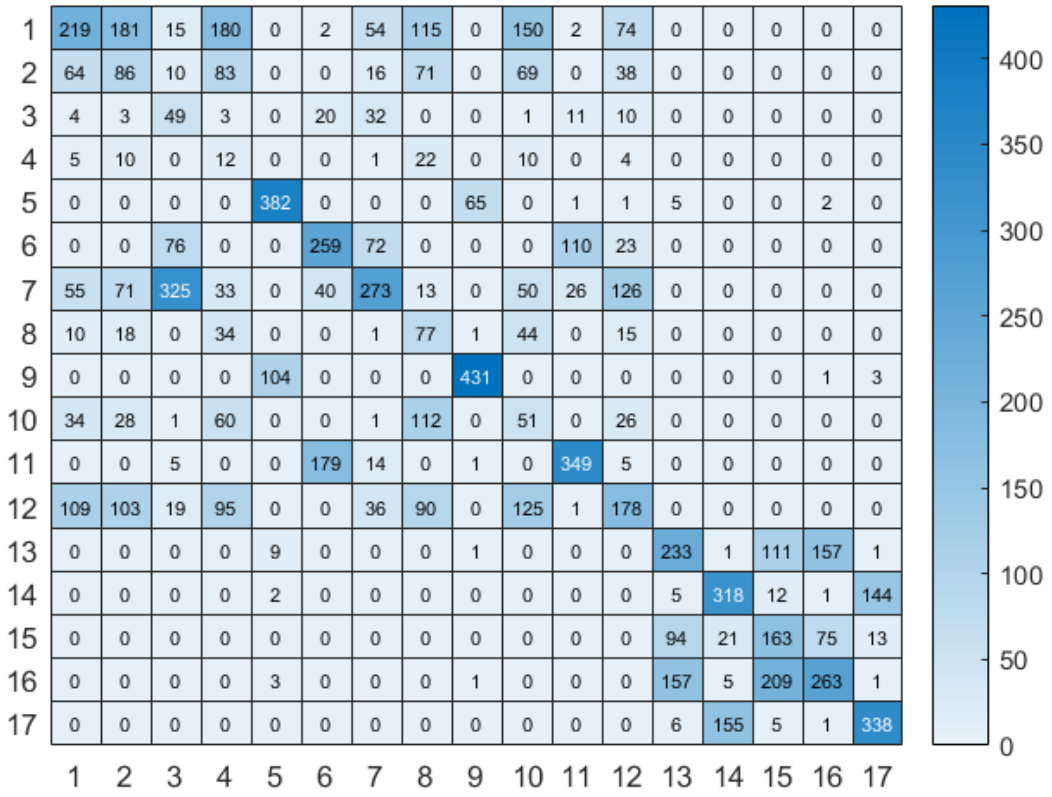

Figure65 confusion matrix for val dataset of wide network with batch = 75 lr = 5e-4 (aug)

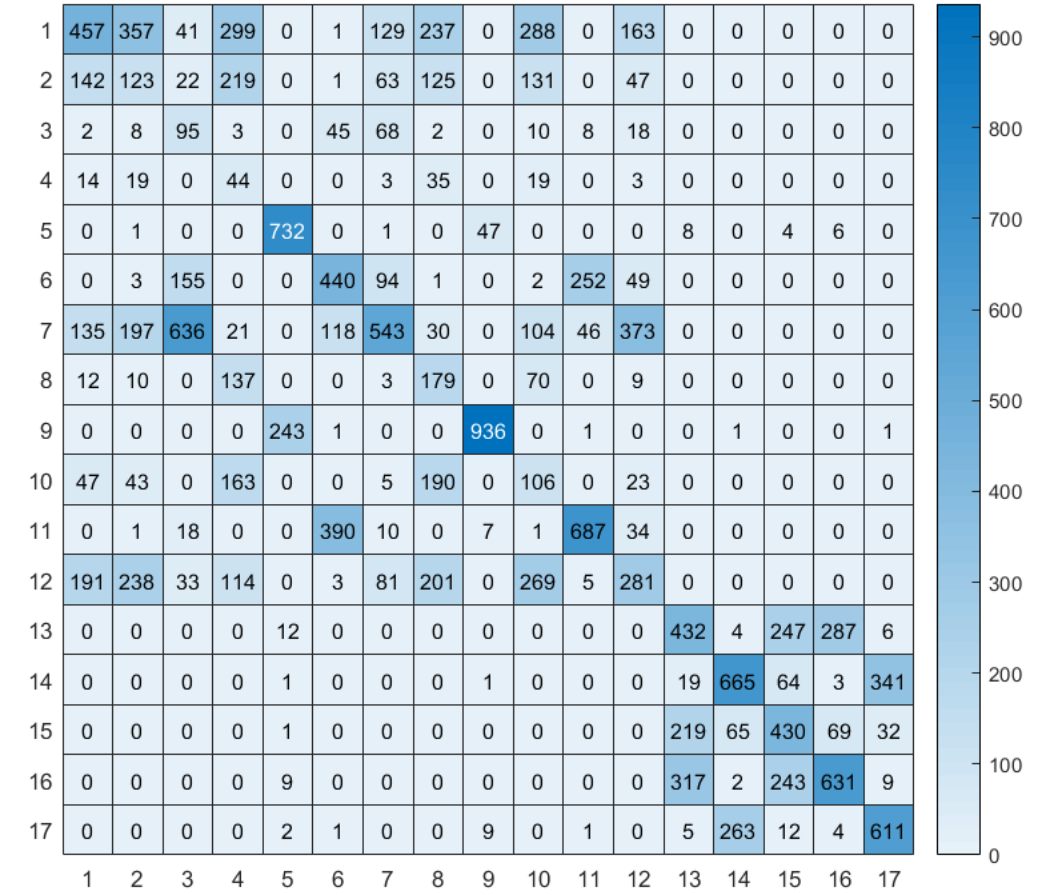

Figure66 confusion matrix for testing dataset of wide network with batch = 75 lr = 5e-4 (aug)

Here is the visualization of the first layer of the skinny network on the data with augmentation.

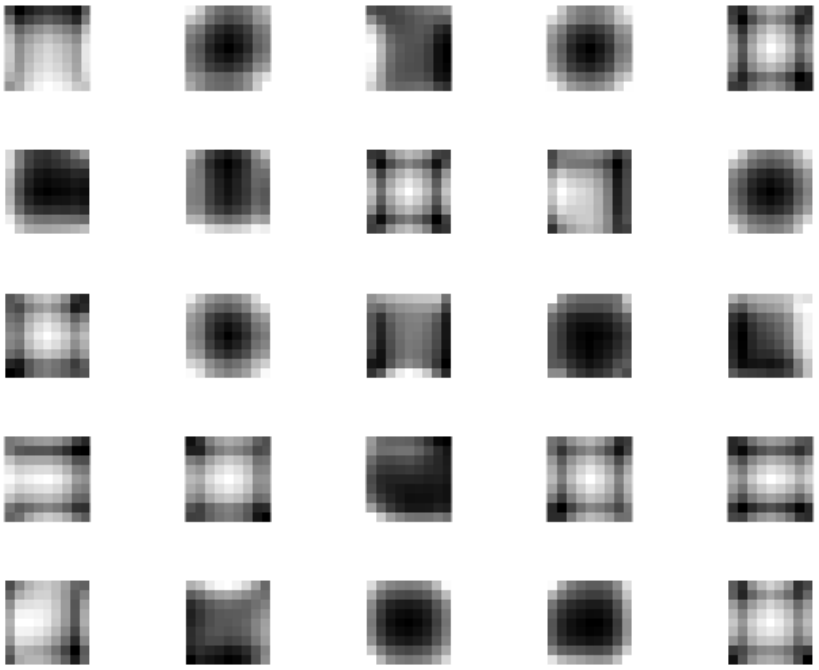

Figure67 visualization of the first layer of wide network(aug)

Here is the visualization2 of the t-SNE multidimensional reduction on the fully connected layer activations of your network trained on the augmentations for both test val and test set for the data with augmentation.

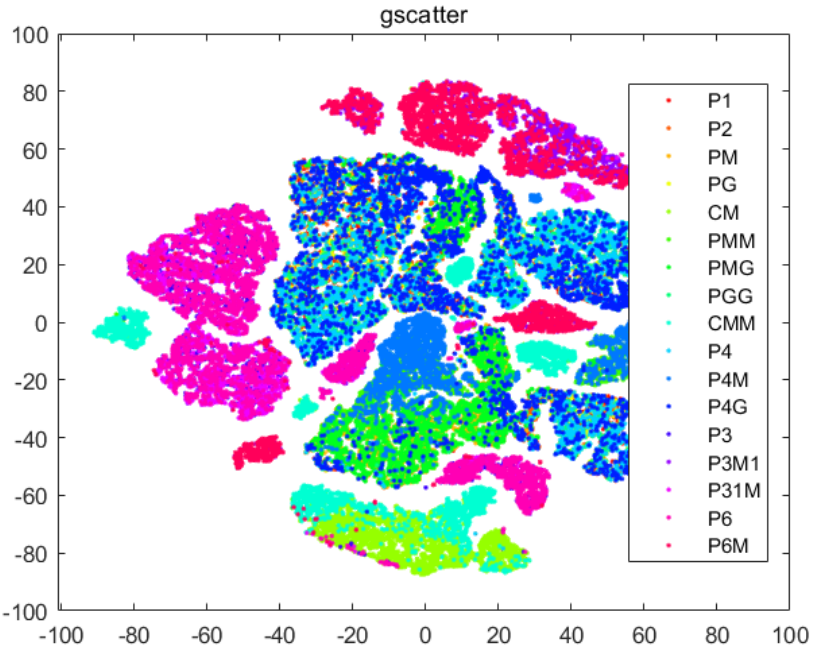

Figure68 visualization2 of the t-SNE of fully connected layer of wide network on trainset(aug)

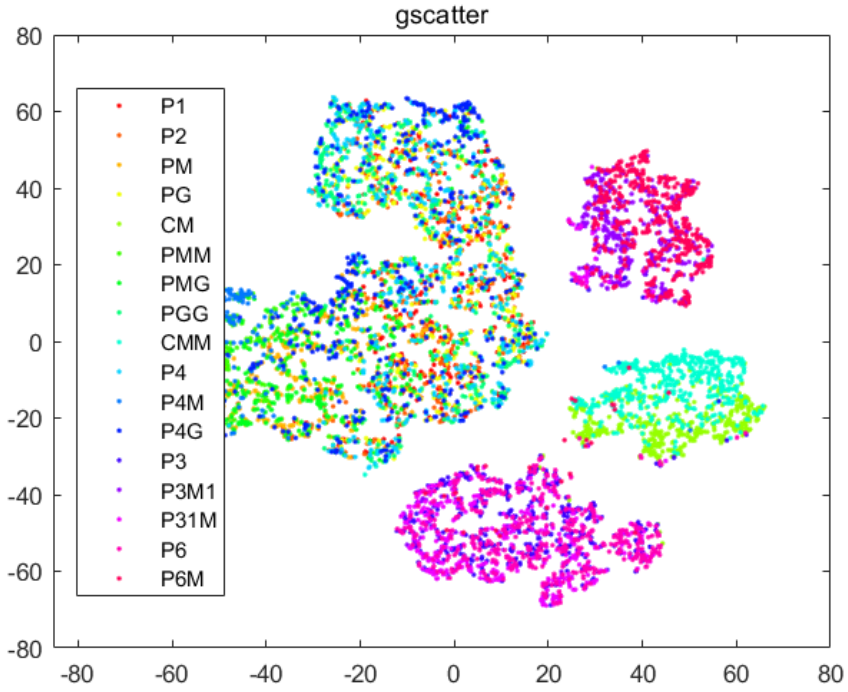

Figure69 visualization2 of the t-SNE of fully connected layer of wide network on valset(aug)

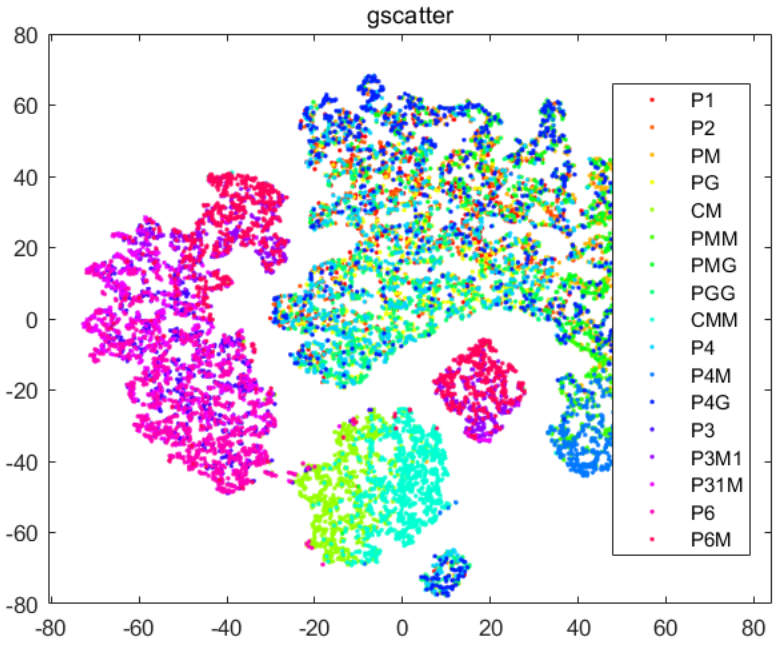

Figure70 visualization2 of the t-SNE of fully connected layer of wide network on testset(aug)

(2) For original data

Check the process_wide_network.txt file. And we separate the the whole process into 2 parts. The first part is with lr=1e-3 and the second part is with lr=5e-5, and both of

these 2 parts have batchsize = 75.

For the first part with lr = 1e-3, the accuracy of training set is 0.8917, and the accuracy of the validate set is 0.8535, and the accuracy of the testing set is 0.8438. This part takes 13709.69 s to finish the whole 15 epochs.

The training accuracy together with the loss figure is shown below.

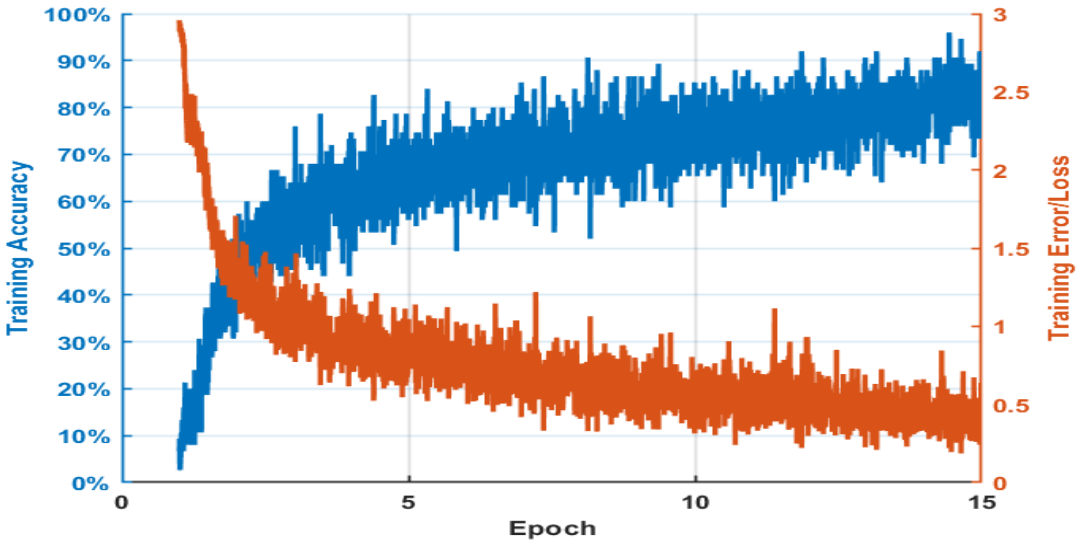

Figure71 training accuracy together with the loss figure of wide network batchsize75 lr = 1e-3

The confusion matrix for training set in this situation is shown below.

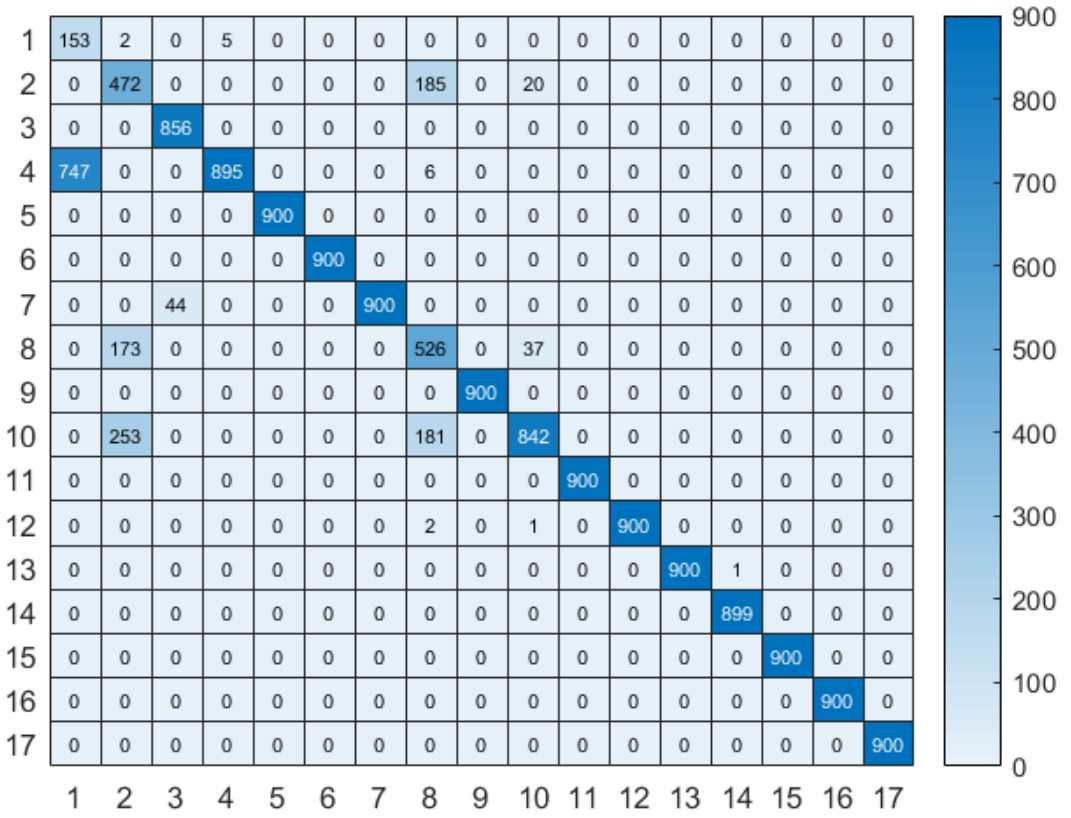

Figure72 confusion matrix for training dataset of wide network with batch = 75 lr = 1e-3

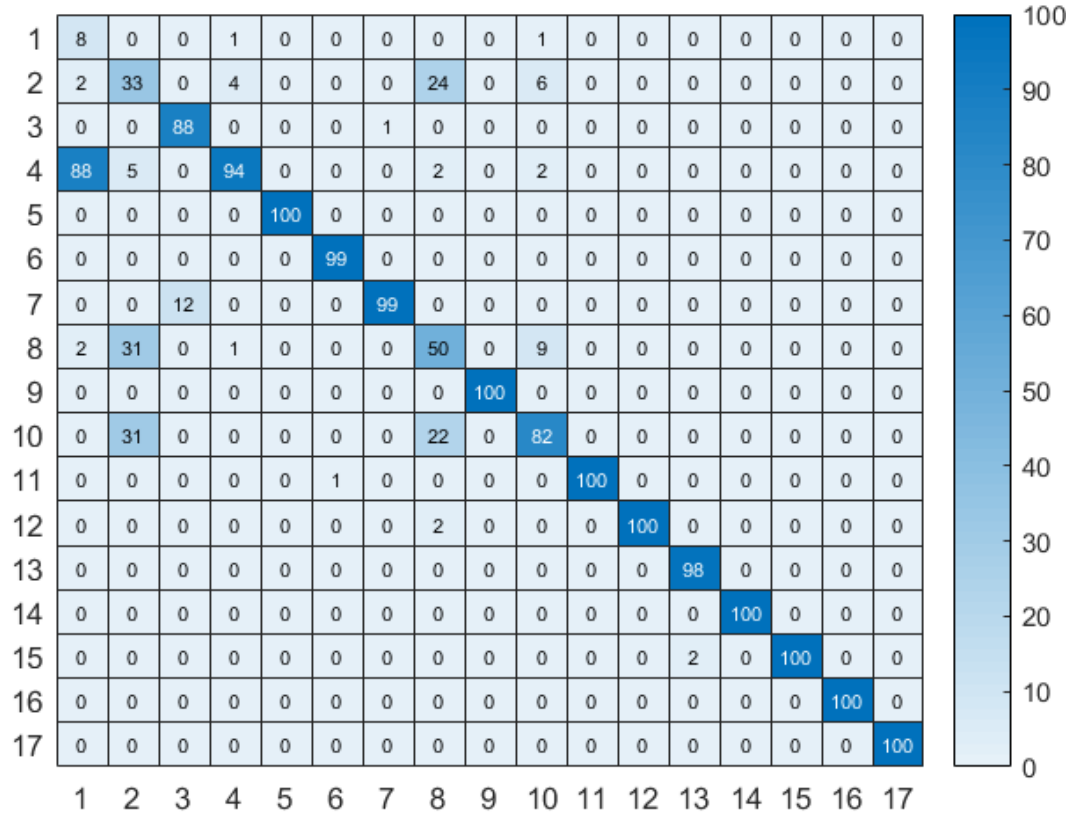

Figure73 confusion matrix for val dataset of wide network with batch = 75 lr = 1e-3

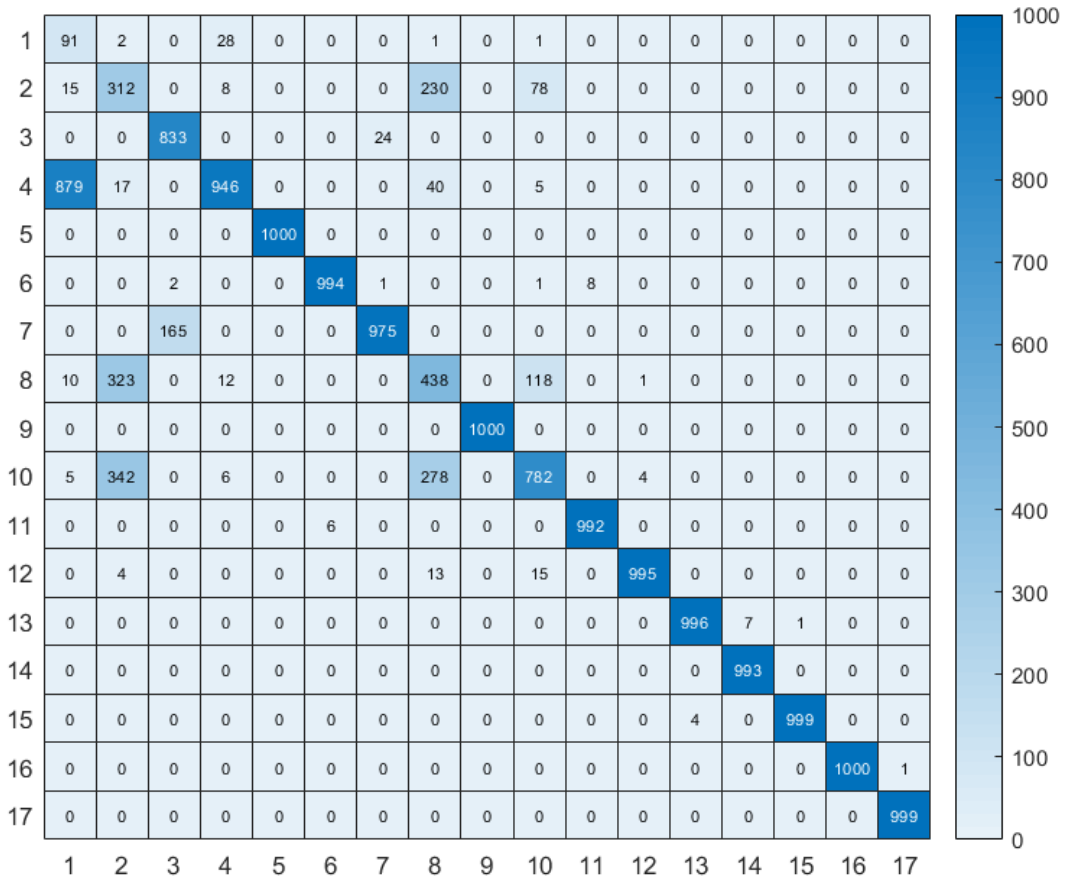

Figure74 confusion matrix for testing dataset of wide network with batch = 75 lr = 1e-3

For the Second part with lr = 5e-4, the accuracy of training set is 0.9220, and the accuracy of the validate set is 0.8824, and the accuracy of the testing set is 0.8670. This part takes 4630.21 s to finish the whole 5 epochs.

The training accuracy together with the loss figure is shown below.

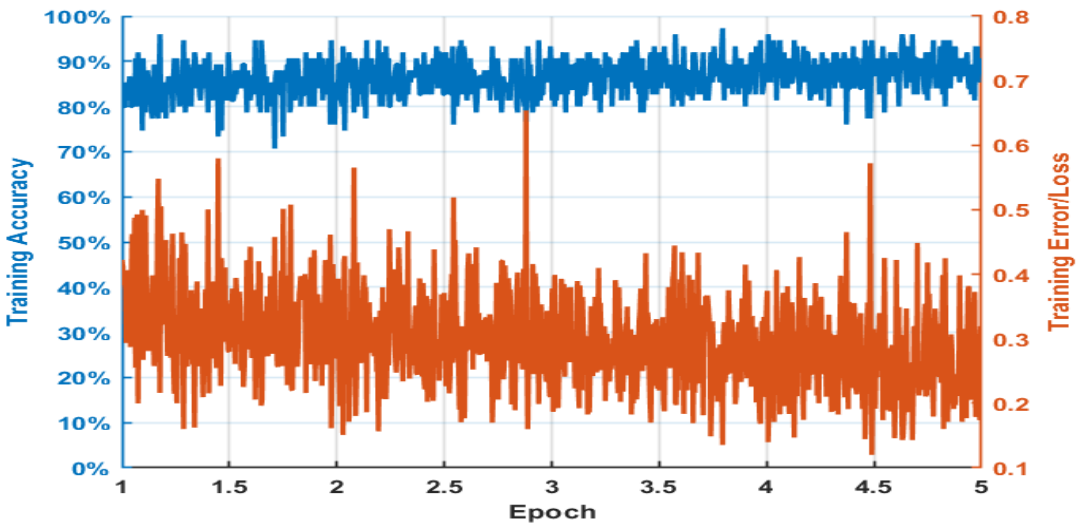

Figure75 training accuracy together with the loss figure of wide network batchsize75 lr = 5e-4

The confusion matrix for training set in this situation is shown below.

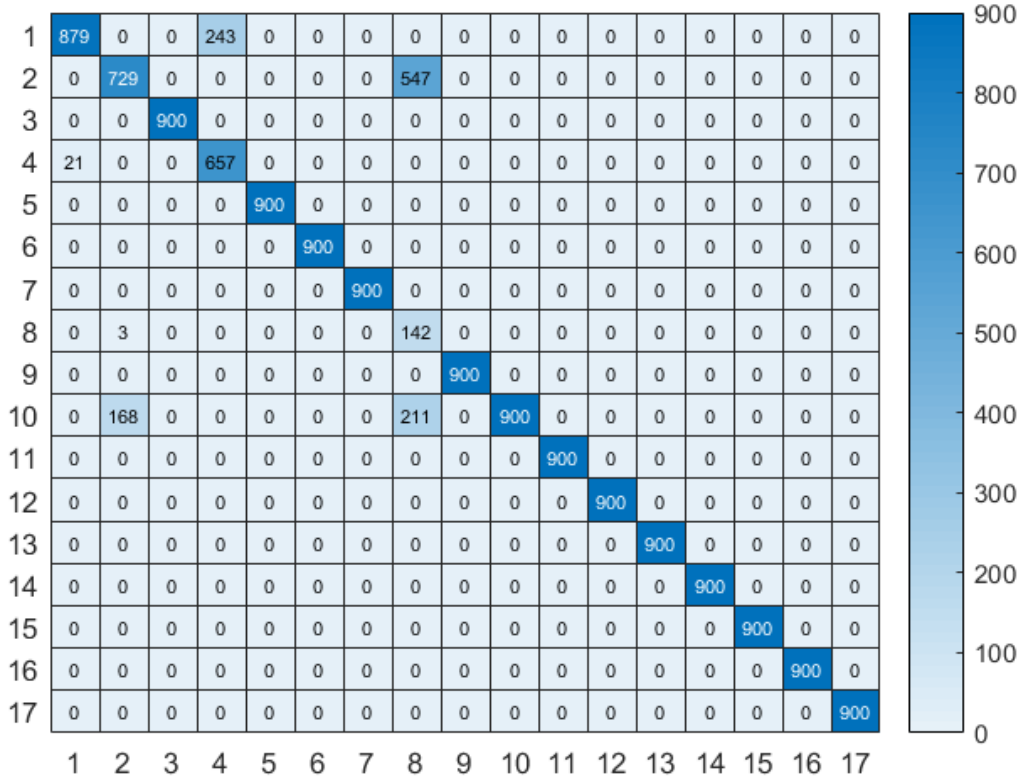

Figure76 confusion matrix for training dataset of wide network with batch = 75 lr = 5e-4

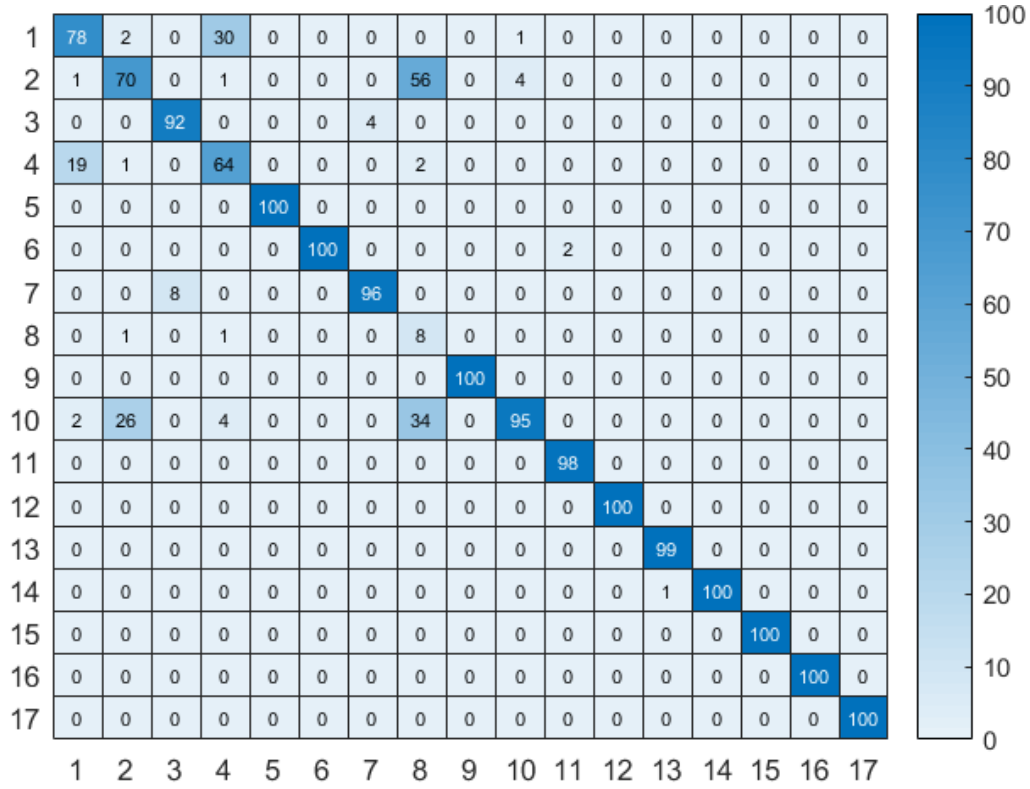

Figure77 confusion matrix for val dataset of wide network with batch = 75 lr = 5e-4

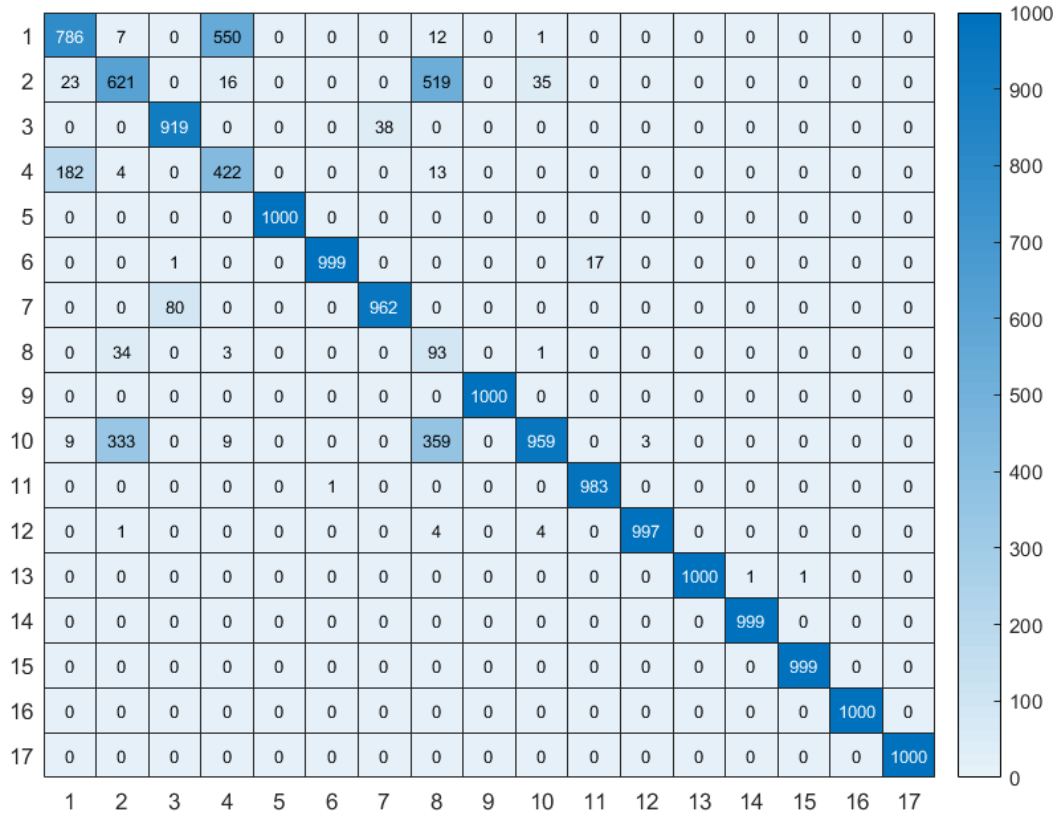

Figure78 confusion matrix for testing dataset of wide network with batch = 75 lr = 5e-4

Here is the visualization of the first layer of the skinny network on the data with augmentation.

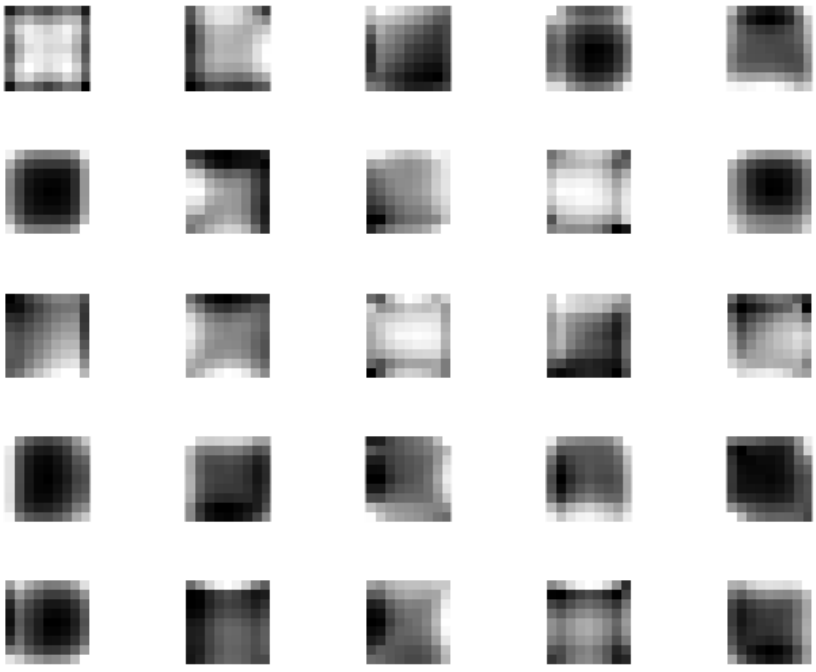

Figure79 visualization of the first layer of wide network

Here is the visualization2 of the t-SNE multidimensional reduction on the fully connected layer activations of your network trained on the augmentations for both test val and test set for the data with augmentation.

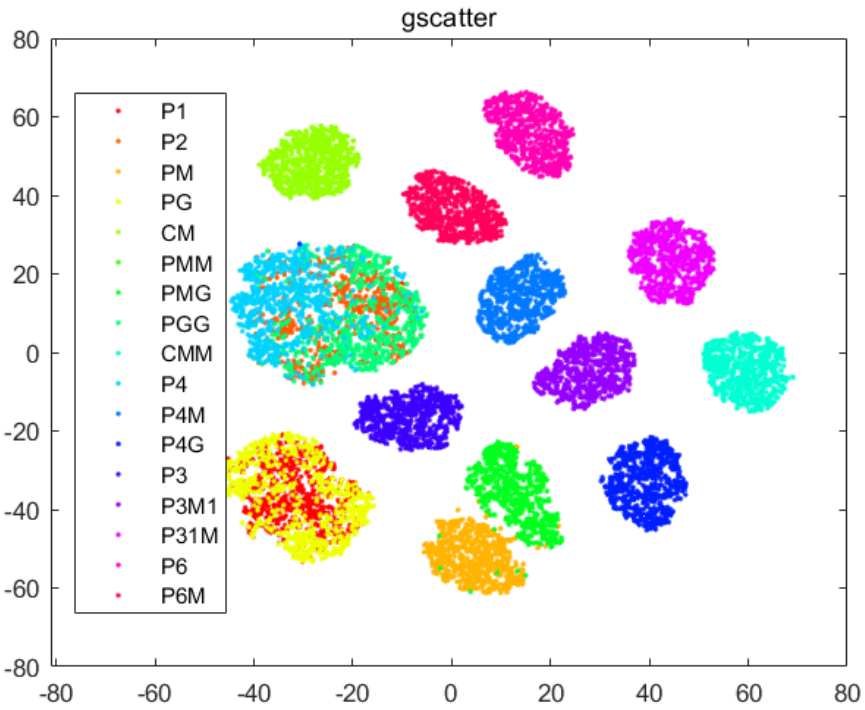

Figure80 visualization2 of the t-SNE of fully connected layer of wide network on trainset

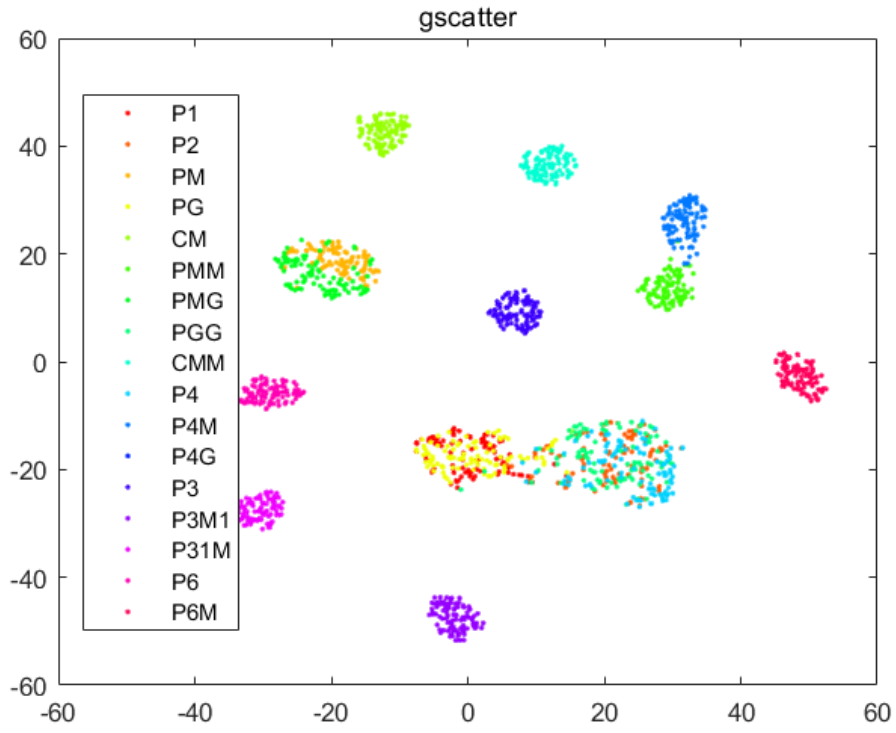

Figure81 visualization2 of the t-SNE of fully connected layer of wide network on valset

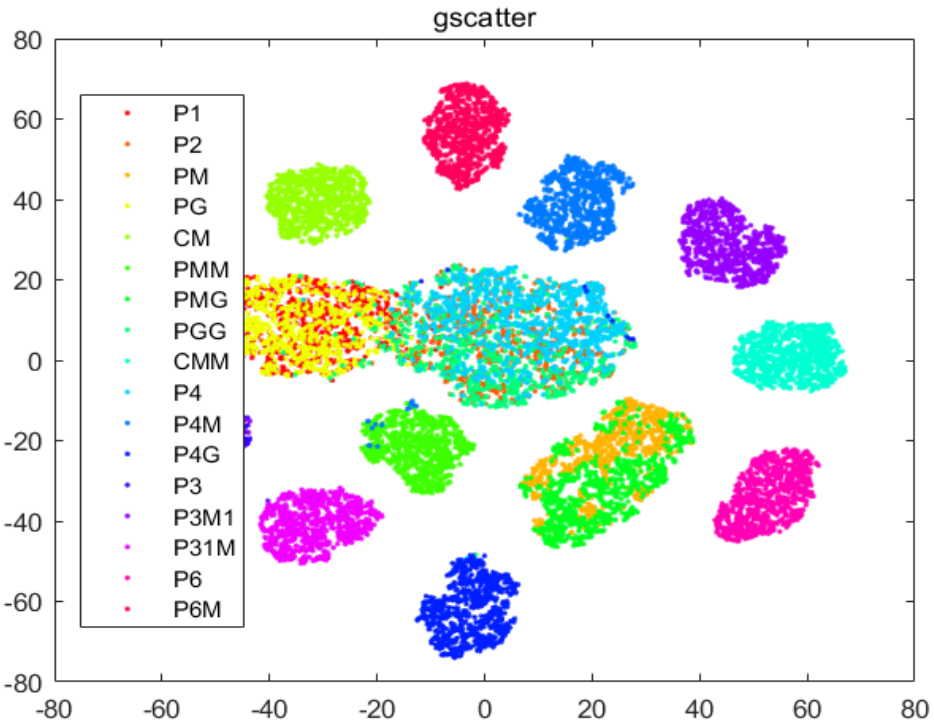

Figure82 visualization2 of the t-SNE of fully connected layer of wide network on testset

3.3.4 Results of Alexnet

As the TA said, I only train the alexnet on the augmentation data. Due to the format of the first layer input of the alexnet is 227x227x3 instead of 128x128x1, I write the code to resize the image and enlarge the 227x227x1 we get from the first step to 227x227x3.

Check the process_alexnet_aug.txt file. And we separate the the whole process is with lr=1e-4 and batchsize = 75, (actually I choose batchsize =125 first which works worse than 75). The accuracy of training set is 0.9663, and the accuracy of the validate set is 0.9055, and the accuracy of the testing set is 0.9085. This part takes 17159.28 s to finish the whole 20 epochs.

The training accuracy together with the loss figure is shown below.

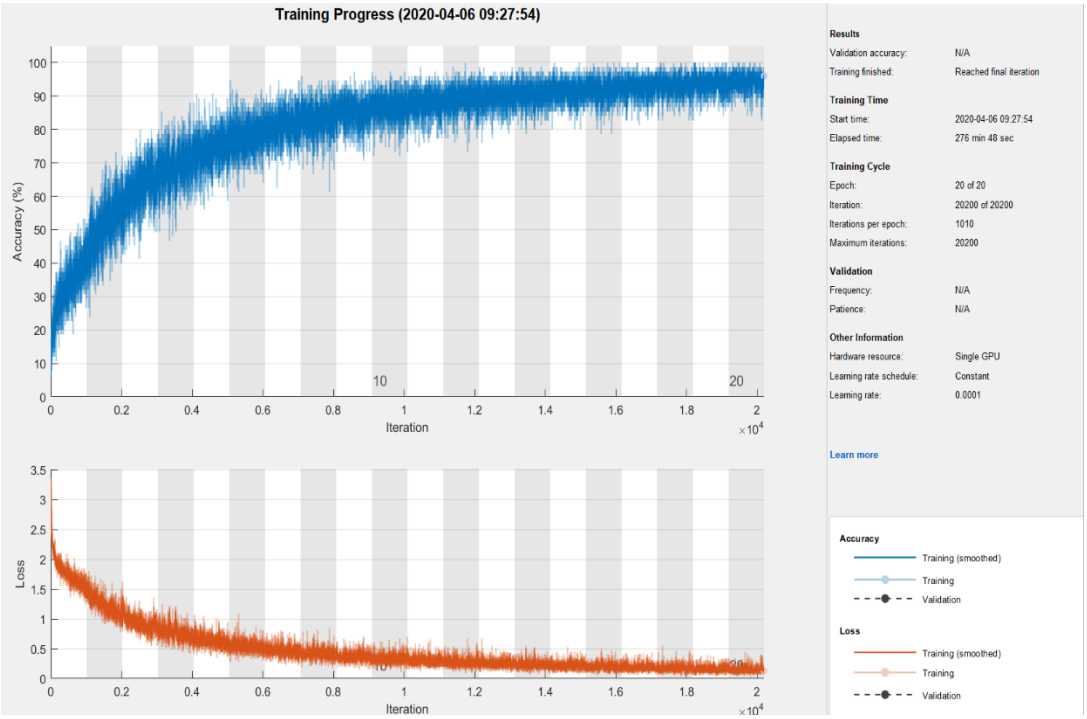

Figure83 training accuracy together with the loss figure of alexnet batchsize75 lr = 1e-4 (aug)1

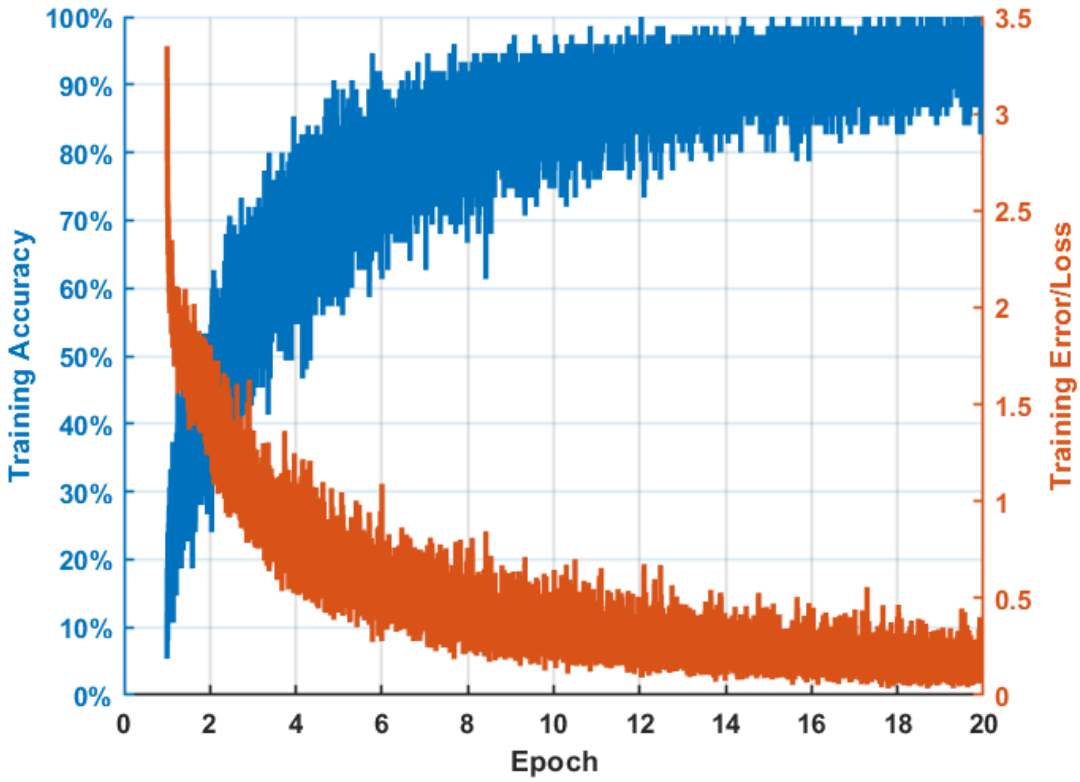

Figure84 training accuracy together with the loss figure of alexnet batchsize75 lr = 1e-4 (aug)2

The confusion matrix for training set in this situation is shown below.

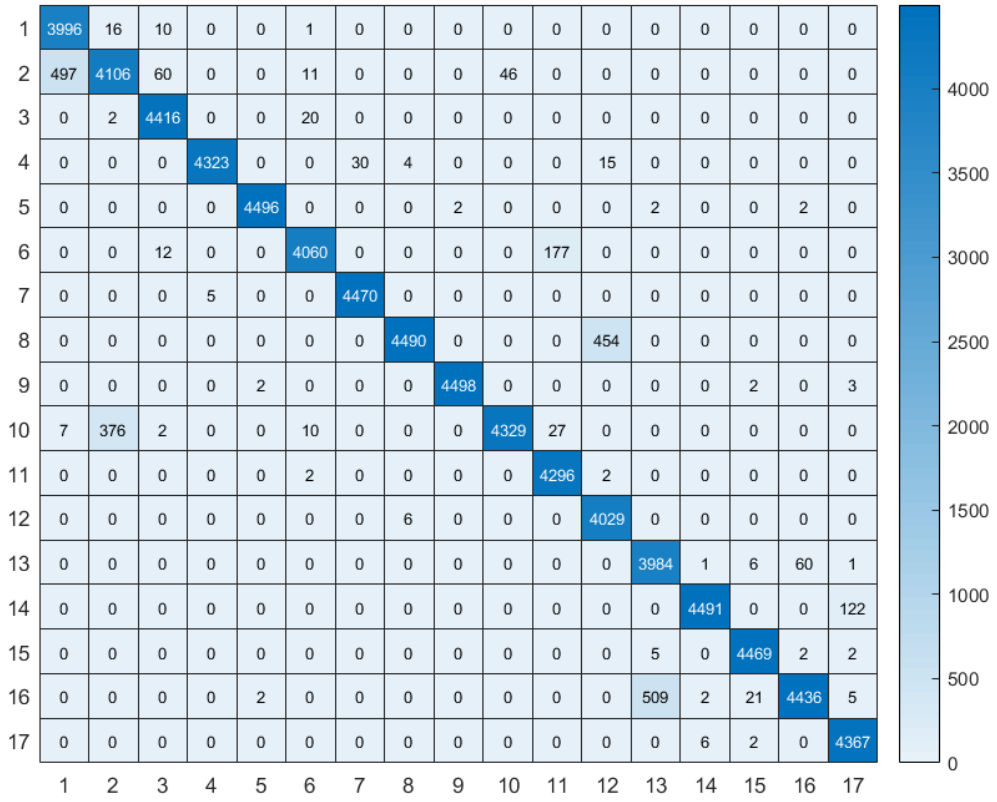

Figure85 confusion matrix for training dataset of alexnet batchsize75 lr = 1e-4 (aug)

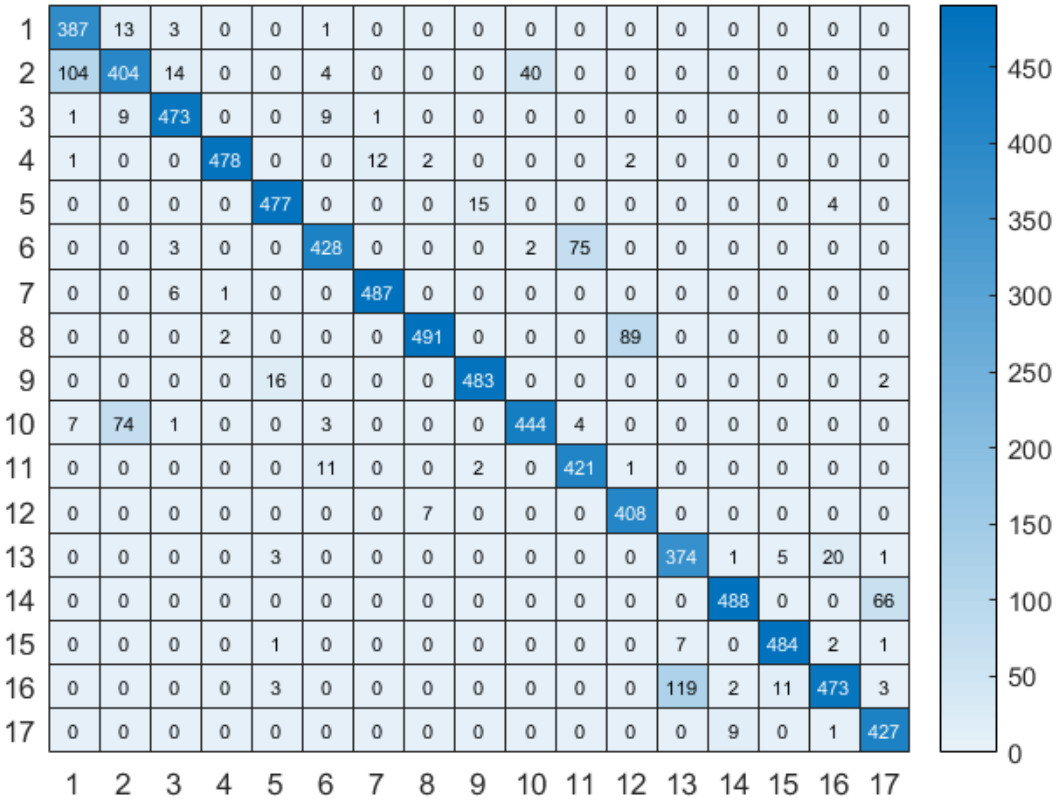

Figure86 confusion matrix for val dataset of alexnet batchsize75 lr = 1e-4 (aug)

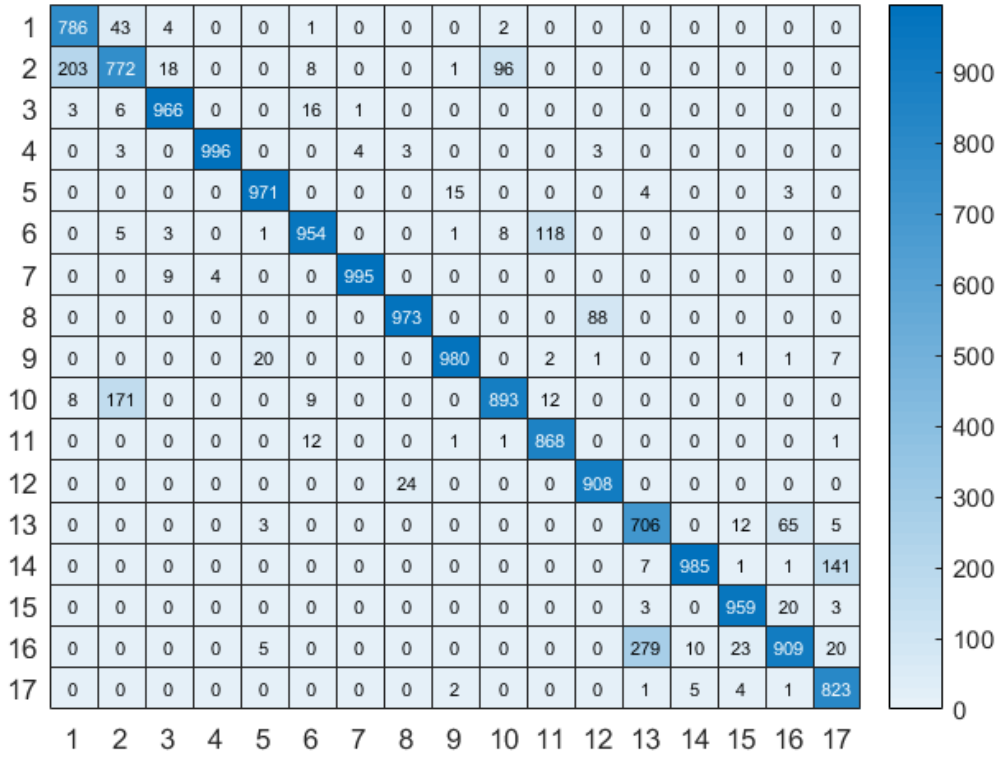

Figure87 confusion matrix for testing dataset of alexnet batchsize75 lr = 1e-4 (aug)

Here is the visualization of the first layer of the skinny network on the data with augmentation.

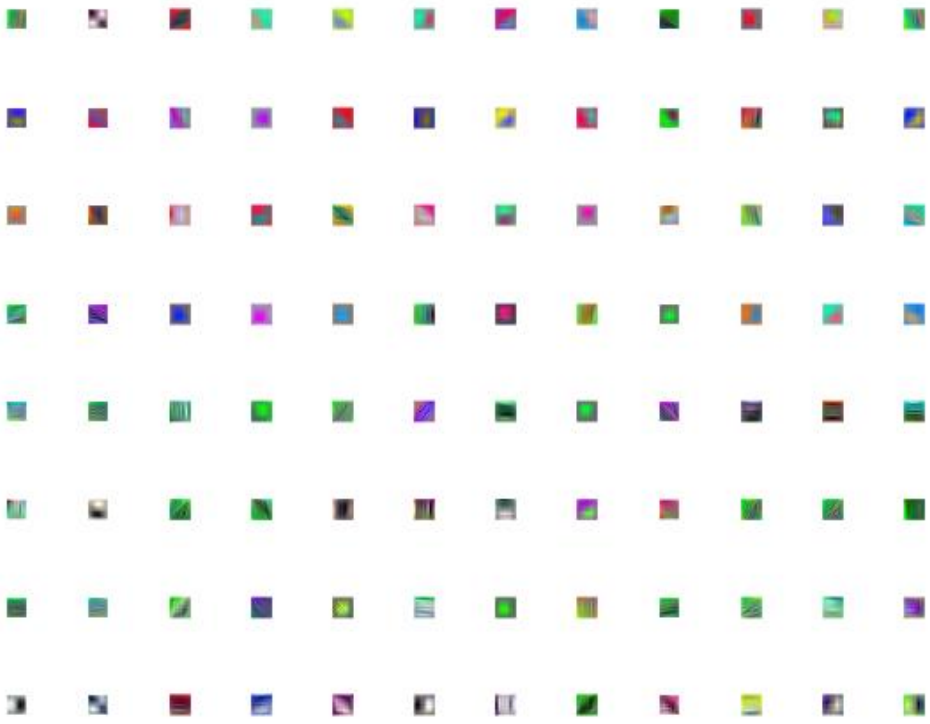

Figure88 visualization of the first layer of alexnet batchsize75 lr = 1e-4 (aug)

Here is the visualization2 of the t-SNE multidimensional reduction on the fully connected layer activations of your network trained on the augmentations for both test val and test set for the data with augmentation.

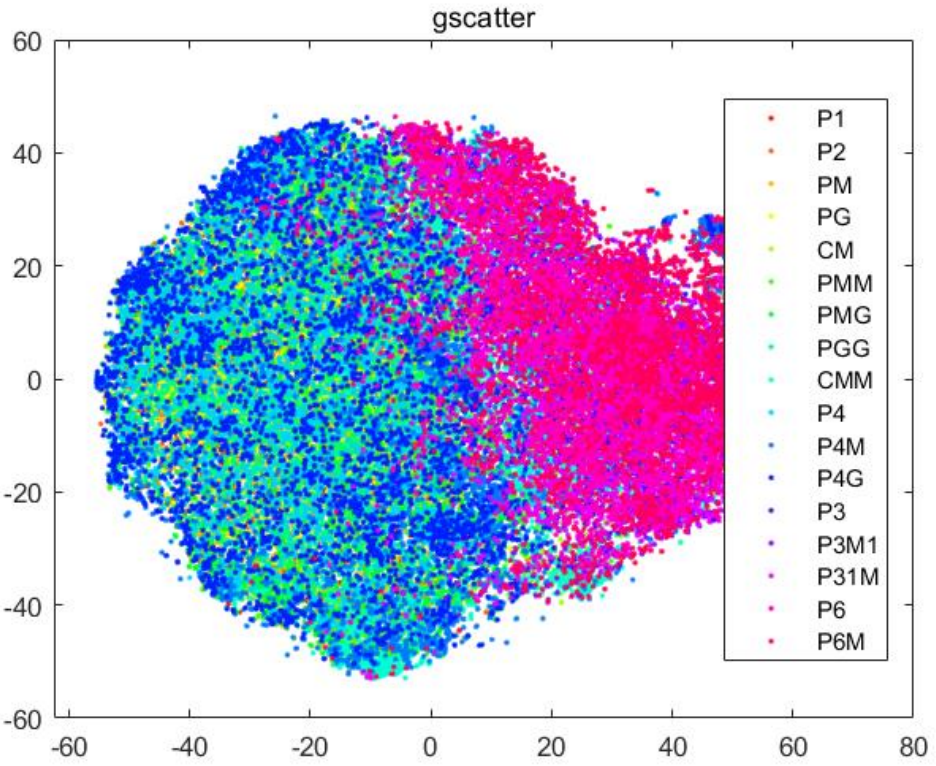

Figure68 visualization2 of the t-SNE of fully connected layer of alexnet on trainset(aug)

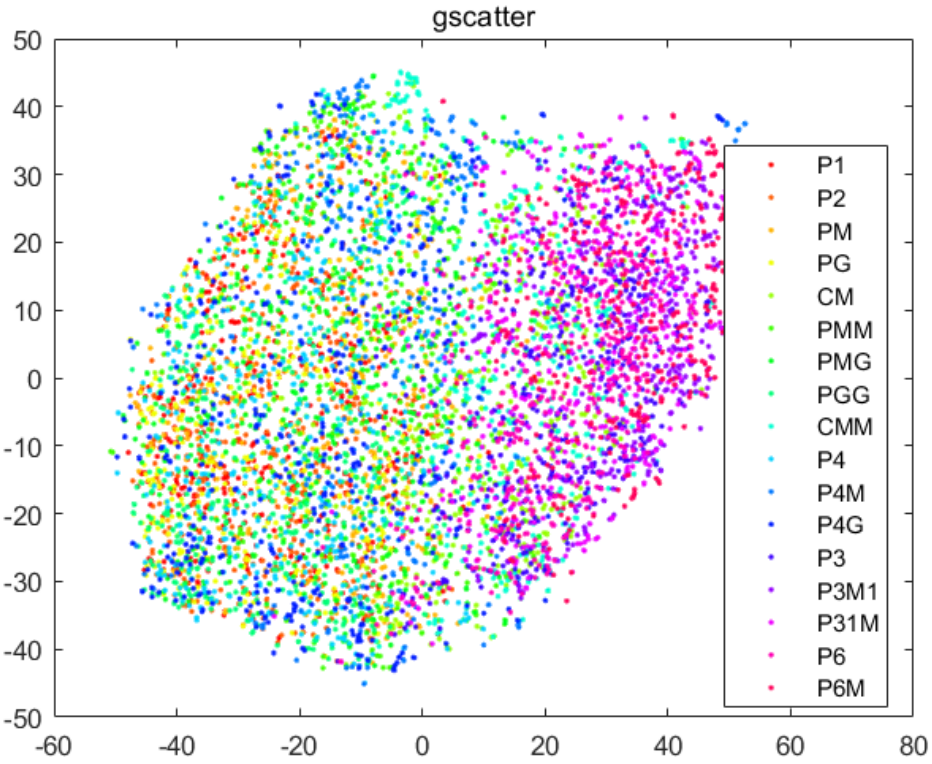

Figure69 visualization2 of the t-SNE of fully connected layer of alexnet on valset(aug)

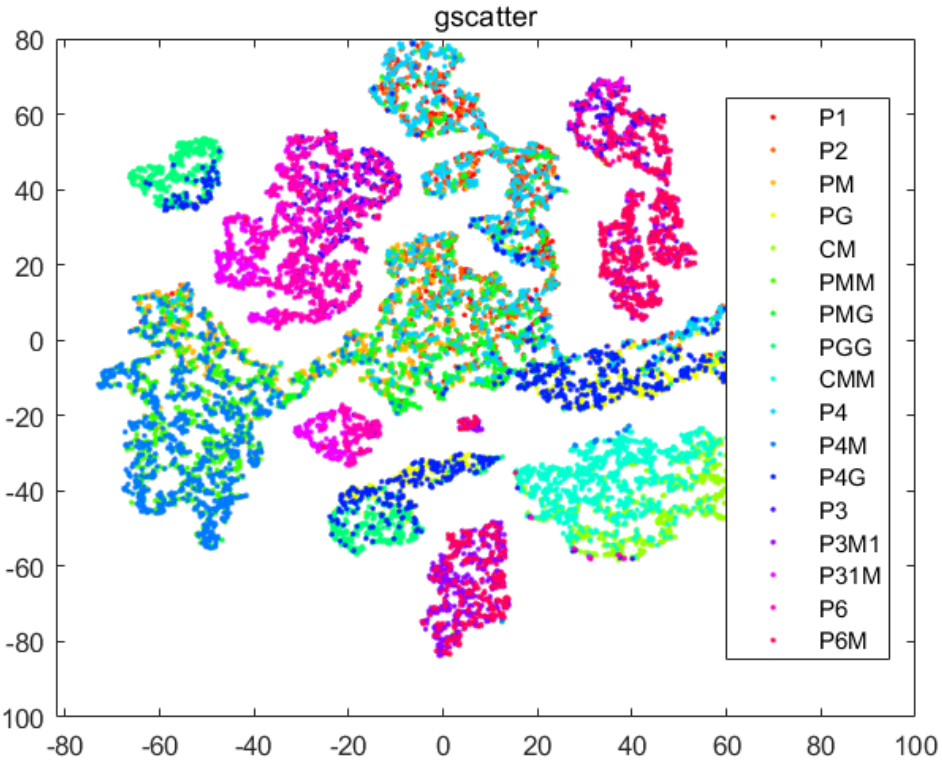

Figure70 visualization2 of the t-SNE of fully connected layer of alexnet on testset(aug)

4 Conclusions

The comparison of original and the augmented data for the for each network is hsown below.

	Original Data	Augmentation Data
Original network	0.8747	0.3286
Skinny network	0.9673	0.6190
Wide Network	0.9220	0.6052
Alexnet	——	0.9663

Table The accuracy of the training set

	Original Data	Augmentation Data
Original network	0.7776	0.2749
Skinny network	0.8729	0.5852
Wide Network	0.8824	0.4331
Alexnet	——	0.9055

Table The accuracy of the val set

	Original Data	Augmentation Data
Original network	0.7771	0.2728
Skinny network	0.8695	0.5762
Wide Network	0.8670	0.4348
Alexnet	——	0.9085

Table The accuracy of the testing set

Based on the tables and “Chapter 3 Results”, we can answer the questions for the project, first, we can find for my original network batch size =75 is better in results and if we reduce the training rate, the speed of converging will decrease and sometimes if it is

stuck in the big training rate training, reducing the training rate will make the network training more further again. And the network I build no matter the skinny network and wide network did increasing the accuracy.

From this project, I learned a lot about creating CNN, which is seriously related to the parameters and number of filters, and if you want to accelerate the converge, the batch normalization layer is a great choice. If you want to try avoid the overfitting, you can modify the rate of the dropout layer. And it is annoyed if you met the crush and save the checkpoint will help a lot. In usual, the accuracy will increase together with the decreasing of the loss, while if you are overfitting the accuracy will not increase.

Reference

1 Christopher, M. Bishop. PATTERN RECOGNITION AND MACHINE LEARNING. Springer-Verlag New York, 2016.

2. https://www.mathworks.com/help/deeplearning/ref/activations.html

3. https://www.mathworks.com/help/stats/tsne.html

4. https://www.mathworks.com/help/deeplearning/ref/activations.html