Pattern Recognition Project 3 – Convolutional Neural Network

Ming-Ju Li

March 4, 2018

Abstract

The convolutional neural network (CNN) is one of the blocks of machine learning. In this project, we will discuss the basic concept of CNN. We are mainly building the CNN and use it to identify the pattern of the wallpaper dataset.

Contents

1 Introduction **3**

2 Theory **4**

3 Back-propagation **5**

4 Data **6**

5 Process Flow **6**

6 Step 1 – Training and Testing the CNN **7**
- 6.0.1 conclusion . . . 7
- 6.1 Functionality of each Layer . . . 7
- 6.2 Confusion Matrix, Accuracy, and other relative result . . . 8
  - 6.2.1 learning rate: 5×10^{-4} . . . 8
  - 6.2.2 learning rate: 1×10^{-4} . . . 9
  - 6.2.3 conclusion . . . 10

7 Step 2 – Augmenting the Training and Testing the data **12**
- 7.1 Statistics . . . 12
- 7.2 Confusion Matrix, Accuracy, and other results . . . 14
  - 7.2.1 Learning Rate: 5×10^{-4} . . . 14
  - 7.2.2 Learning Rate: 1×10^{-4} . . . 16
  - 7.2.3 conclusion . . . 16

8 Step 3 – Building the Neural Network **17**
8.1 Skinny Network 18
8.1.1 Skinny Net on Original Data 18
8.1.2 conclusion 18
8.1.3 Skinny Net on Augmented Data 20
8.2 Wide Network 21
8.2.1 Wide Net on Original Data 22
8.2.2 Wide Net on Augmented Data 25

9 Conclusion **28**

1 Introduction

The convolutional neural network (Figure 1) is built based on neurons (Figure 2). Neurons are the function set calculated from the input images, with the activation functions after the computation of the input images and output the result after activation functions. In this project, we separated the dataset into three subsets, training data, validation data, and testing data. The purpose of having a validation data is to further confirm the accuracy of the trained functions in case of overfitting problems.

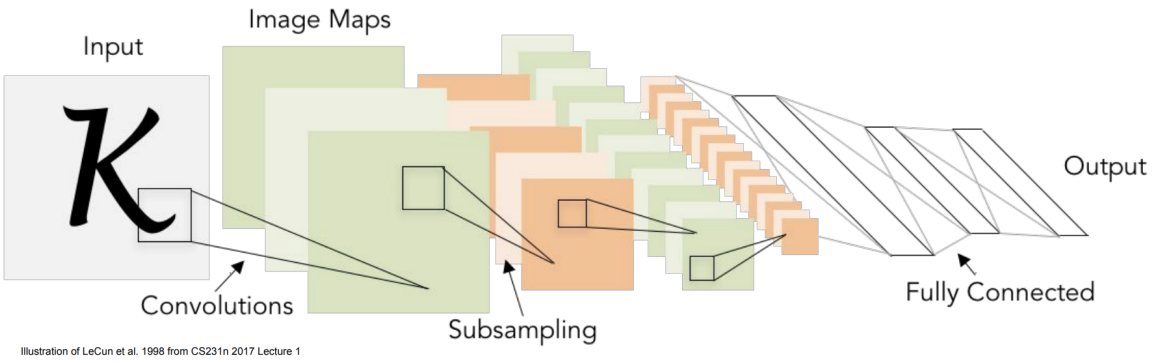


Figure 1: The concept of the Convolutional Neural Network

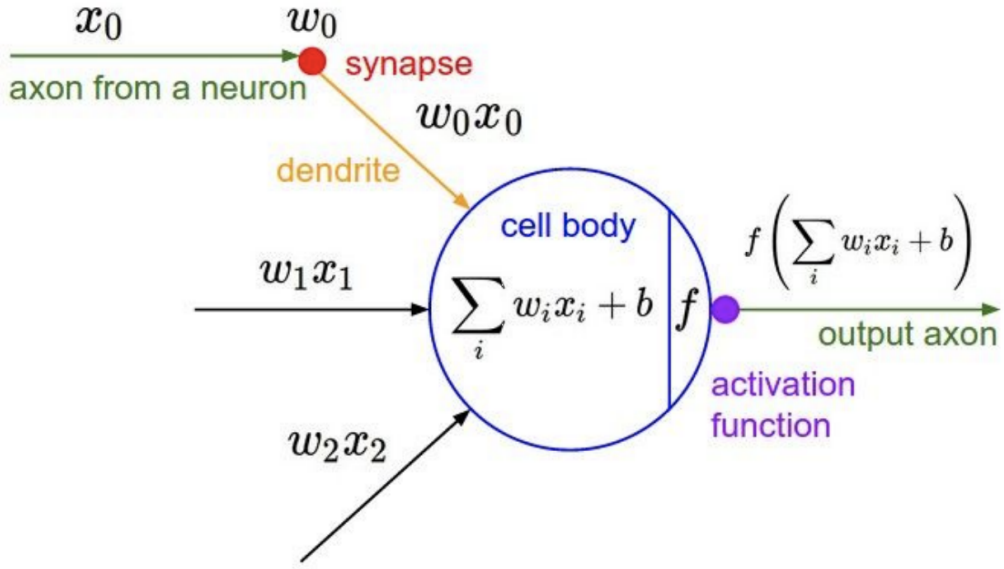


Figure 2: Neuron. The neurons take all the previous output as input, multiply with the weight function, and use activation function to determine if fire the output or not.

2 Theory

The intuition of neural network came from the logistic regression. The limitation of logistic regression is that it cannot always separate the data. Therefore we need to do feature transformation of the data and cascade with classification (Figure 3). If we see the feature transform as multiplying with some weight function (Figure 4) , repeat this structure for many times and connect them, the neural network is formed (Figure 5).

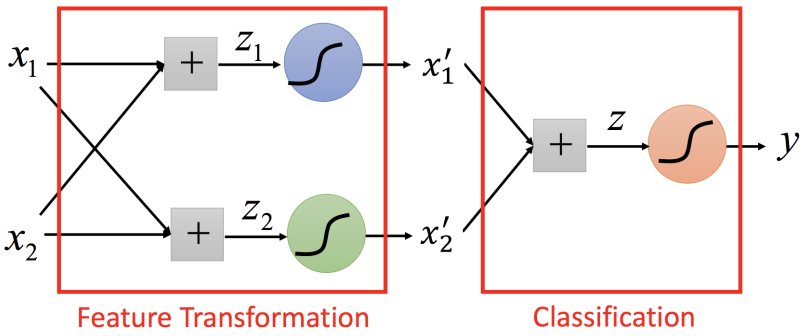


Figure 3

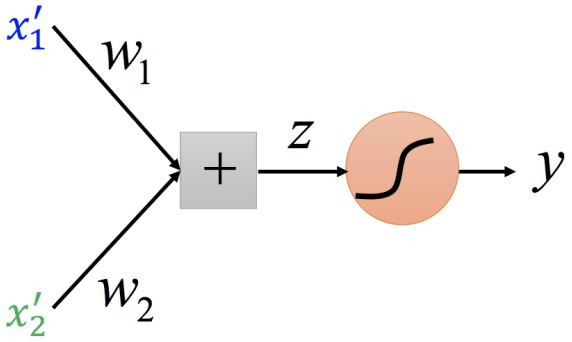


Figure 4

What one convolution layer does is to compute the convolution with the input, and send the result to the activation function to see if it fires or not. The convolution is simply the inner product of the filter to the input image added by a bias.

$$\mathbf{wx} + \mathbf{b} \tag{1}$$

and pass the result to the activation function,

$$y_k = f(\mathbf{w}^k\mathbf{x} + \mathbf{b}) \tag{2}$$

The final output of the CNN will be,

$$y_{total} = f(\mathbf{w}_L ... f(\mathbf{w}_2 f(\mathbf{w}_1 x + b^1) + b^2) + ... + b^L) \tag{3}$$

This produces one pixel of an activation map. The filters slide over the full input image with a given stride, i.e., the rate of the filter to slide, to create one activation map. One

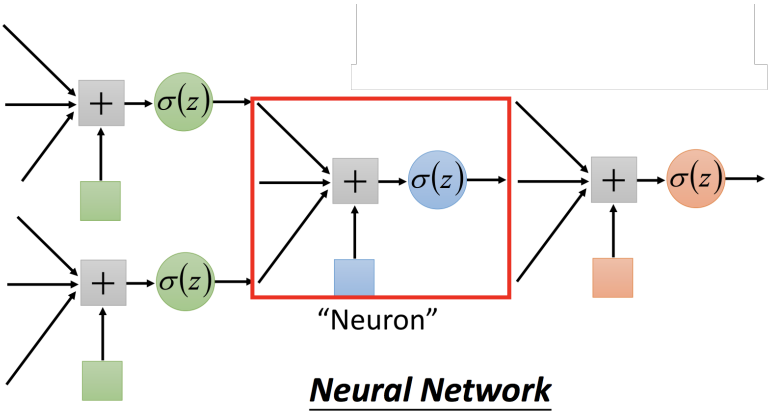


Figure 5

filter creates one activation map. Many filters create stack of activation maps, or the volume. The output dimension depends on the number of filters. The padding of the images

3 Back-propagation

The most important part of the convolution neural network is the back-propagation. It updates the weight of each layer. We use the stochastic gradient descend (SGD) in this project. The SGD basically calculates the gradient descend of each neuron. This is how the gradients is obtained,

$$\frac{df(x)}{dx} = \lim_{h \to \text{inf}} \frac{f(x+h) - f(x)}{h} \tag{4}$$

so basically the gradient is the partial derivatives with respect of that particular point.

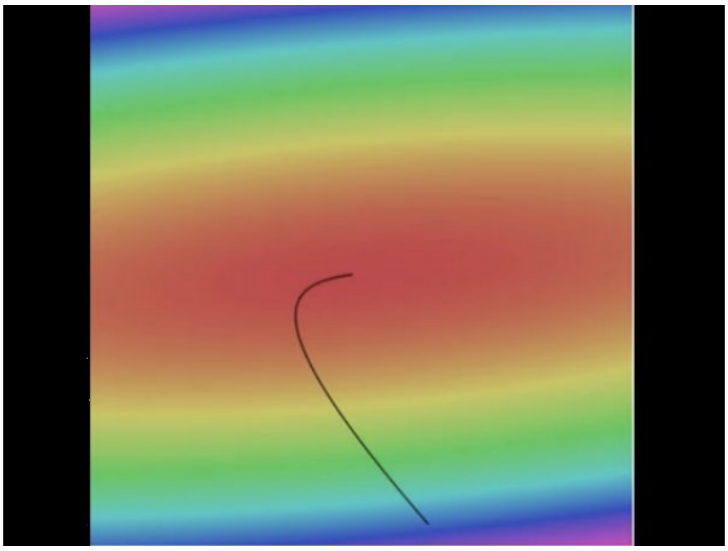


Figure 6

The above figure shows the intuition of SGD. First came up with one random point. Then compute the gradient of that point and update the weight matrix with the multiplication of the gradient and the learning rate.

$$\mathbf{w}_{new} = \mathbf{w}_{old} - lr \times \frac{\partial L}{\partial w} \tag{5}$$

where the "lr" is the abbreviation of "learning rate", which is a hyperparameter that controls the "step size" of gradient descend. By doing so, we are gradually "walking" to the lowest point of the loss function, which gives us the optimal value of the weight function correspondingly.

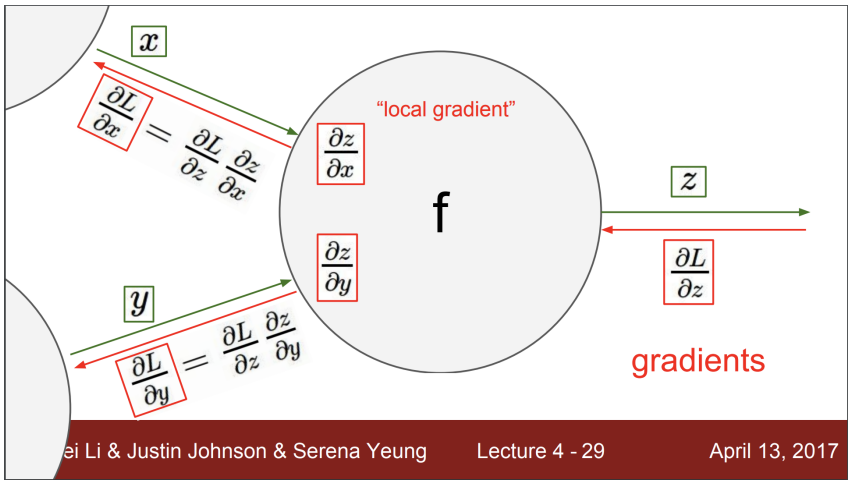


Figure 7: Backpropagation

4 Data

The dataset consists of 17,000 images, 1,000 images per group, and each image containing a wallpaper pattern. The images are 256x256 and 1 channel (grayscale). We separated the validation data from the training date with the ratio of 0.9. We will be discovering the patterns of the dataset by training the CNN to do so. In the later of this project, we also create a dataset based on the original dataset and augment it with different parameters to get a larger dataset.

5 Process Flow

The project is broken into three steps. The first step is to train the neural network based on the original data and apply the result on validation data. The neural network is provided by the starter codes. The second step is to augment the original data by rotating, scaling, and translation. Then we train the neural network with the new augmented dataset. The last step is to build our own neural network and apply it onto the original and the augmented dataset to see its performance. There are different types of network we built, which will be discuss later in this report.

6 Step 1 – Training and Testing the CNN

In this step, we run totally 10 epochs with batch size of 250. Also, we implement 7 hidden layers, which are:

$$CONV \rightarrow ReLU \rightarrow POOL \rightarrow FC \rightarrow DROPOUT \rightarrow FC \rightarrow SOFTMAX \quad (6)$$

The output of the current layer is the input of the next layer. For example, the output of the $CONV$ layer is the input of the $ReLU$ layer. The following is what each layer do. The $CONV$ means the convolution layer. It computes the inner product of the given filters and the input images. The filters slide through the whole images to get the output images. The following shows the number of parameters of this network:

6.0.1 conclusion

Layer	Conv	Pool	FC	Dropout	FC
Type	filter	pool size	output size	rate	output size
size	$5\times5\times20$	2×2	25	0.4	17
parameters	520	N/a	N/a	N/a	N/a

Table 1: Accuracy of each learning rate on each original data in the start network

6.1 Functionality of each Layer

$ReLU$ is the activation function, which has outputs called activation map.

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

It has the output plot shown as Figure 7.

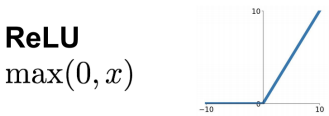


Figure 8: ReLU

The reason why we need activation function is that we need to add non-linearity into the networks. If there is only linear layers, having multiple layers would be equivalent to having only one layer and there would be no sense to build the deep network.

$POOL$ layer is doing the maximum pooling to the input. It selects the maximum value of a certain pooling area showing in Figure 2. This example selects the maximum value from each 4 pixels. The purpose of pooling is to alleviate the computing expense for learning a classifier. In the fully connected (FC) layer, the input volumes are been

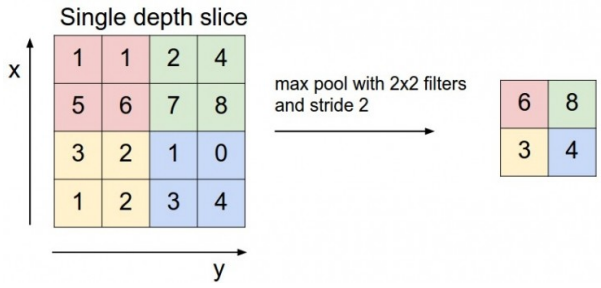


Figure 9: Maxpooling

stretched out and become N-dimensional vectors, where N is the number of classes. For the *DROPOUT* layer, it randomly drops some neurons to make the network easier to compute and this process helps prevent overfitting. Also, it makes the training network to be slightly different for every batch.

6.2 Confusion Matrix, Accuracy, and other relative result

In the plot of classification and confusion matrix, the darker of a block, the higher the percentage it is. Therefore, if the dark block is on the right position, the diagonal entries, instead of other positions of the row, the accuracy is high, vice versa.

6.2.1 learning rate: 5×10^{-4}

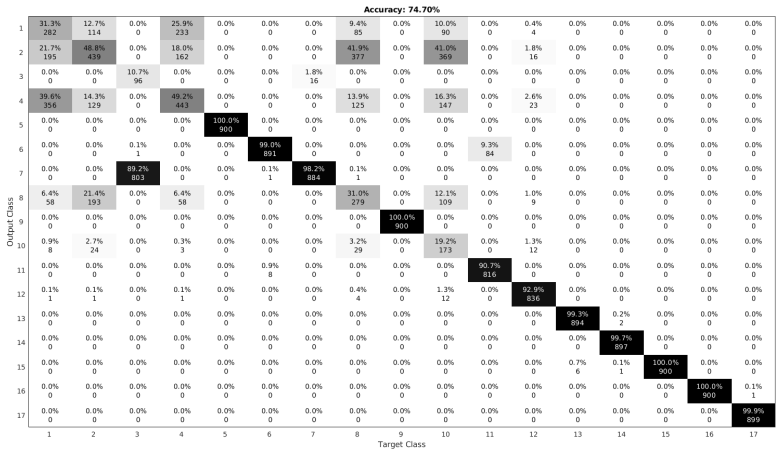


Figure 10: classification matrix and the confusion matrix of the training data with learning rate 5×10^{-4}

Figure 11 shows the plot of accuracy to training loss. The orange line is the loss of the training process and the blue line is the training accuracy. It is obvious that the loss decreases significantly while the accuracy increases as the epoch increases. The accuracy reaches approximately **70%** in the 7^{th} epoch and the loss is approximately 0 at the same epoch.

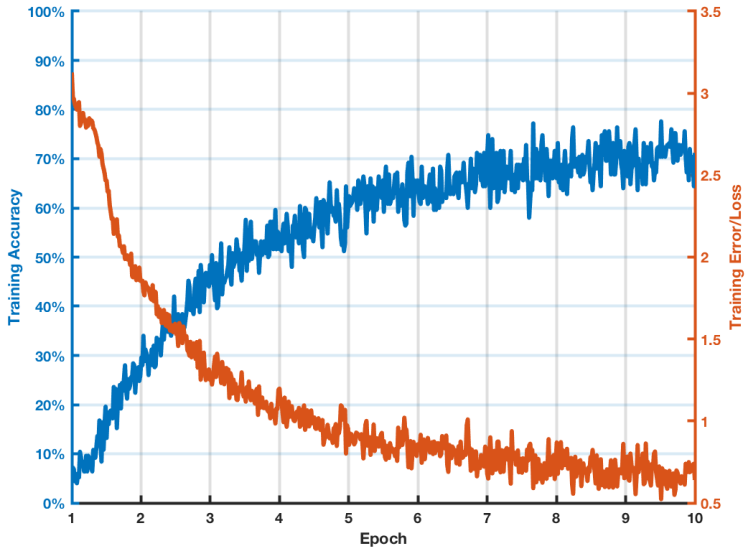


Figure 11: The training accuracy versus the loss of each batch

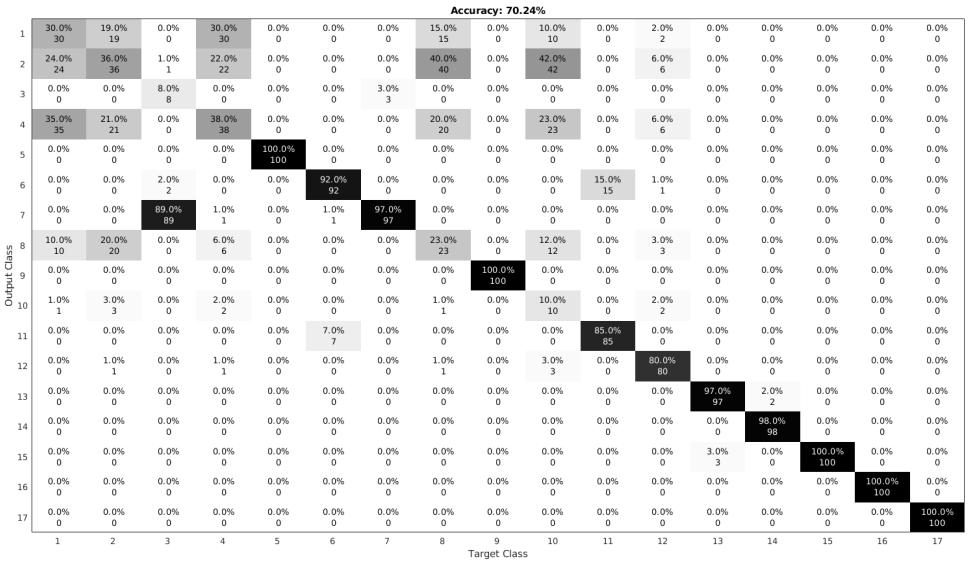


Figure 12: The confusion and classification matrix of validation data with learning rate 5×10^{-4}

6.2.2 learning rate: 1×10^{-4}

For this section, we trained the new network starting from the result of the previous network. Also, the difference between these two parts is the learning rates, which are 5×10^{-4} *v.s.* 1×10^{-4}, respectively.

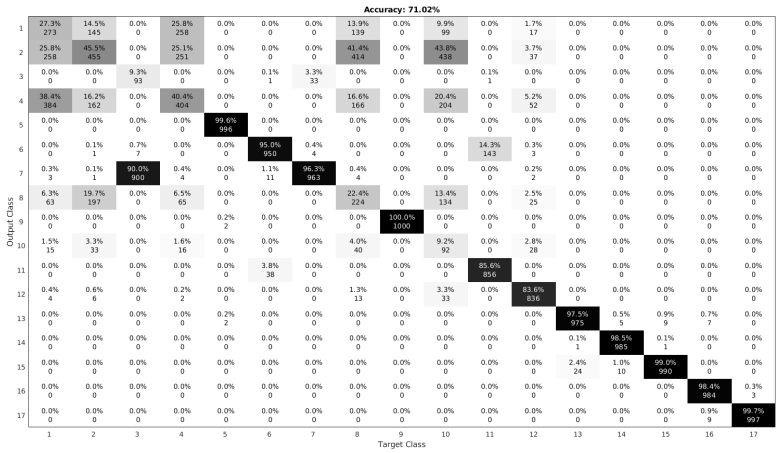


Figure 13: The confusion and classification matrix of testing data with learning rate 5×10^{-4}

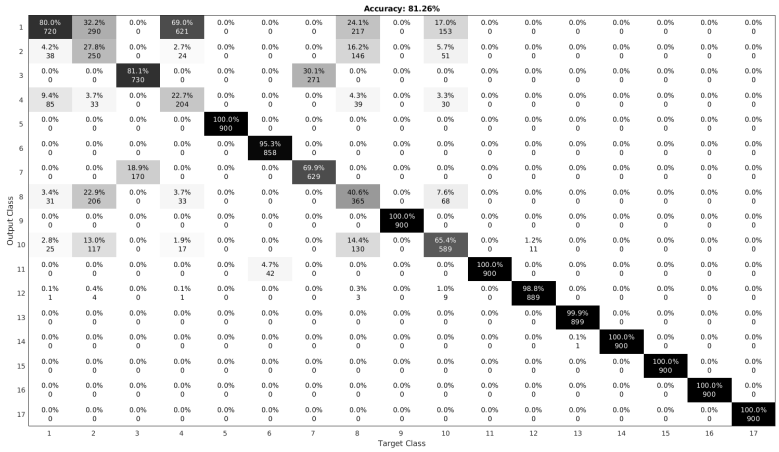


Figure 14: classification matrix and the confusion matrix of the training data with learning rate 1×10^{-4}

Learning Rate	5×10^{-4}	1×10^{-4}
Training	74.70%	81.26%
Validation	70.24%	74.41%
Testing	71.02%	74.55%

Table 2: Accuracy of each learning rate on each original data in the start network

6.2.3 conclusion

From the accuracy presented in Table 2, it is obvious that with the lower learning rate, it is not necessarily guaranteed the accuracy will significantly increase. The accuracy in

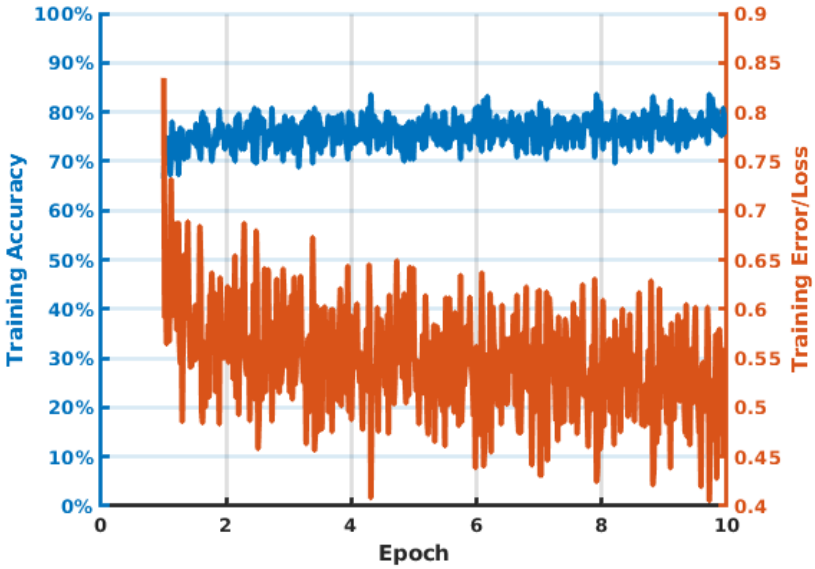


Figure 15: The training accuracy versus the loss of each batch. As shown in the figure, the accuracy and the loss are already converged. They sustain the same value no matter how the epoch increases. This is the because of applying the fine-tuned network.

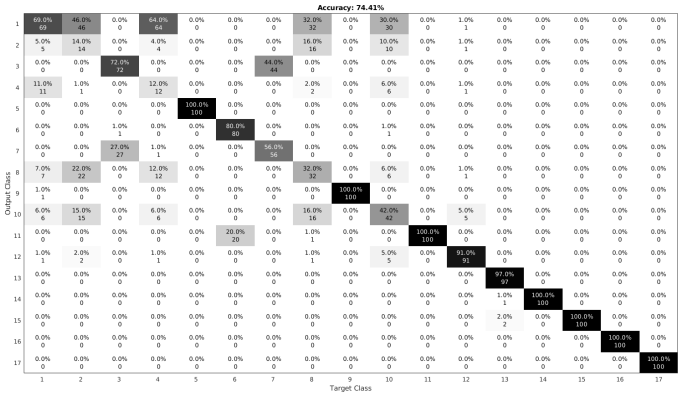


Figure 16: The confusion and classification matrix of validation data with learning rate 1×10^{-4}

the first part, where the learning rate is larger than the second part, is approximately **70%**. For the second part, the accuracy increases to approximately **75%**. This probably be the contribution of the fine-tuned network from the network with larger learning rate. However, if we change the structure of the network, we are able to significantly increase the accuracy. This can be shown in the last section of this report.

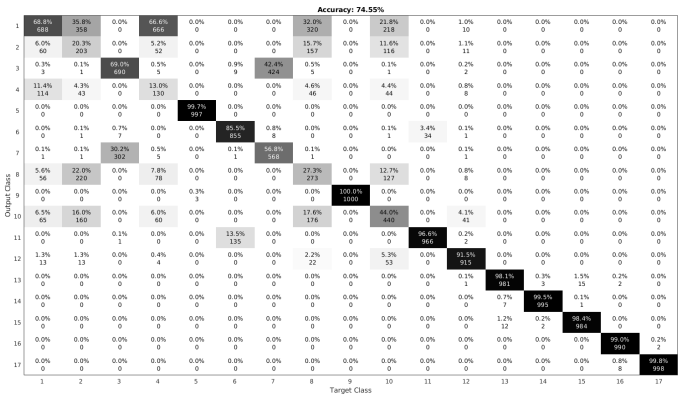


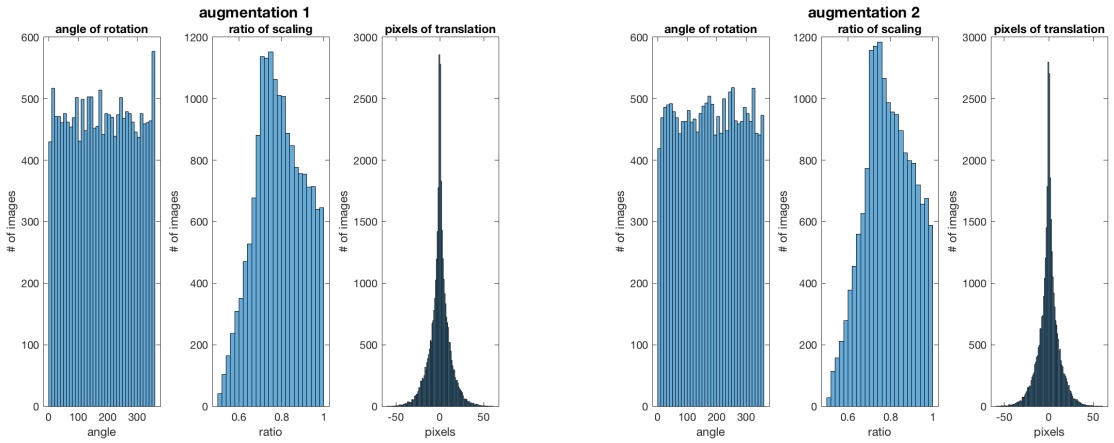
Figure 17: The confusion and classification matrix of testing data with learning rate 1×10^{-4}

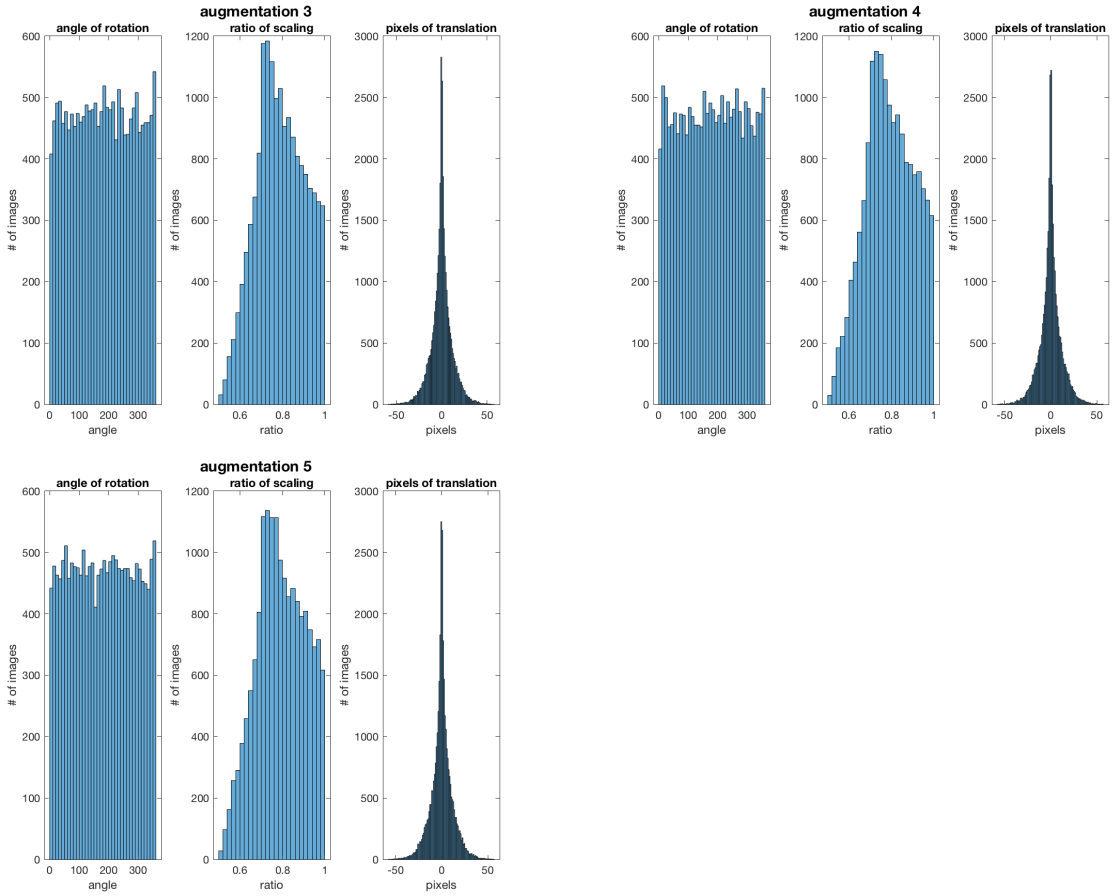
7 Step 2 – Augmenting the Training and Testing the data

We are doing a supervising learning in this project. However, the labeling processing is very expensive. So the augmentation of the original dataset becomes a solution to create more dataset without re-labeling them. In this project, we augment the original dataset by the combination of rotation, scaling, and translation. The value of each augmentation is randomly chosen. Therefore, every image in each class has different angle of rotation, ratio of scaling, and pixels of translation. We have created 5 different augmented training datasets and 1 augmented testing dataset. We will present the statistics of the different training dataset in the next sector.

7.1 Statistics

Here are the statistics of the augmentations. The histograms from the left to right are angle of rotation, ratio of scaling, and the pixels of translation, respectively.





mean	1	2	3	4	5
rotating angle	180.1°	179.9°	180.5°	180.4°	179.6°
scale ratio	0.7897	0.7911	0.7901	0.7904	0.7917
translated pixels	[-0.2062, 0.0612]	[-0.0205 ,-0.0423]	[-0.0662, 0.0581]	[0.0291, -0.0367]	[0.0517, 0.0879]

Table 3: Mean of each statistic of each augmentation

standard deviation	1	2	3	4	5
rotating angle	104.25°	103.53°	103.67°	103.84°	103.85°
scale ratio	0.1129	0.1116	0.1125	0.1128	0.1131
translated pixels	[11.44, 11.48]	[11.23, 11.42]	[11.42, 11.52]	[11.47, 11.46]	[11.39, 11.46]

Table 4: Standard deviation of each statistic of each augmentation

7.2 Confusion Matrix, Accuracy, and other results

The accuracy of the augmented dataset lies approximately in 20%. It is expected to have more accurate prediction and classification with larger dataset. However, the result does not meet our expectation. The reasons will be discussed in the end of this section.

7.2.1 Learning Rate: 5×10^{-4}

The confusion matrix and the classification matrix of the augmented data is shown as:

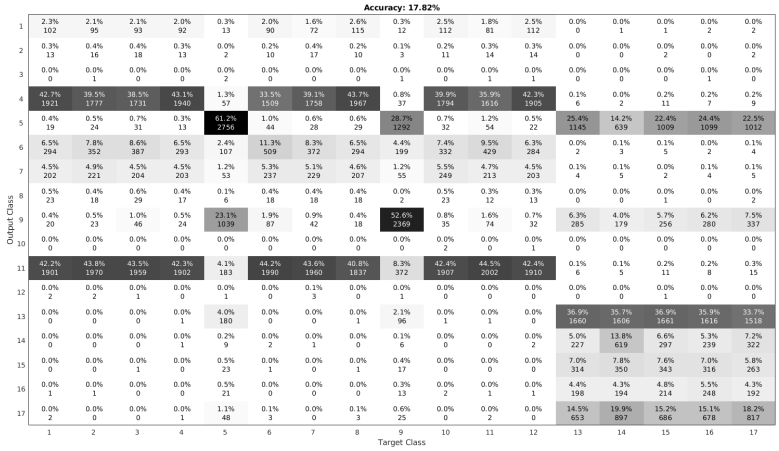


Figure 18: Confusion and classification matrix of training data with the learning rate of 5×10^{-4}. The accuracy is **17.82**%. It often mis-classify class 1 through 13 to the 4^{th} and the 11^{th} class, while the last 4 classes are often messed together.

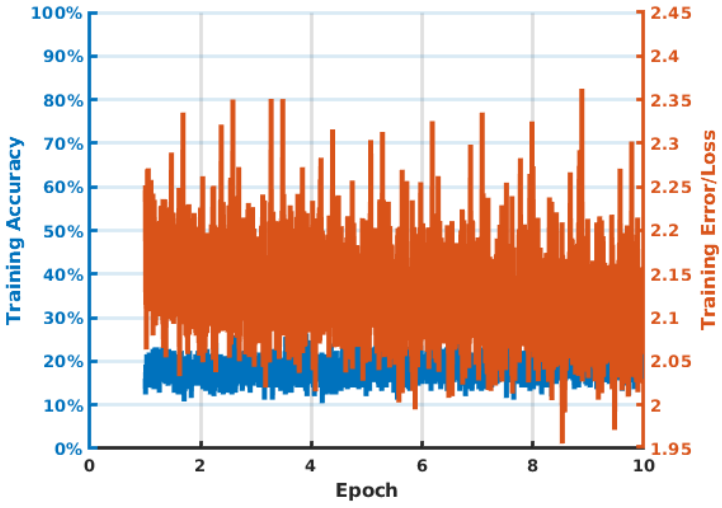


Figure 19: The accuracy versus loss. From this figure, it is obvious that the training loss is huge, which is result from underfitting.

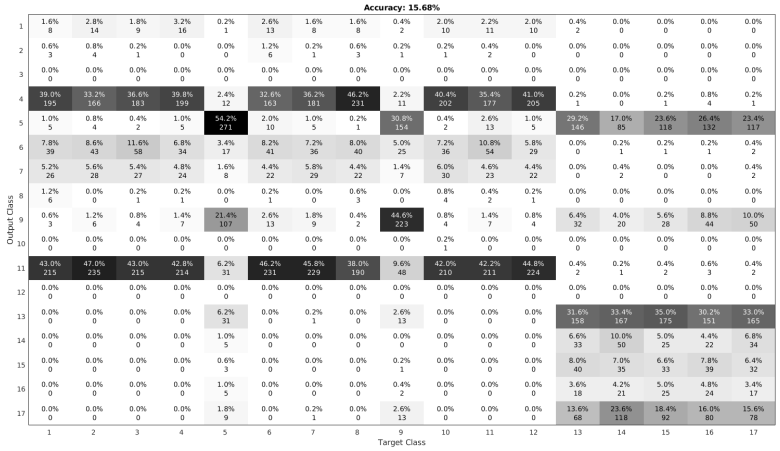


Figure 20: Confusion and classification matrix of validation data with the learning rate of 5×10^{-4}. The accuracy is 15.68%, which is significantly lower than the previous dataset. Same to the training data, the network has high likelihood to classify class 1 to 13 as class 4 and 11 and mess class 14 to 17 together.

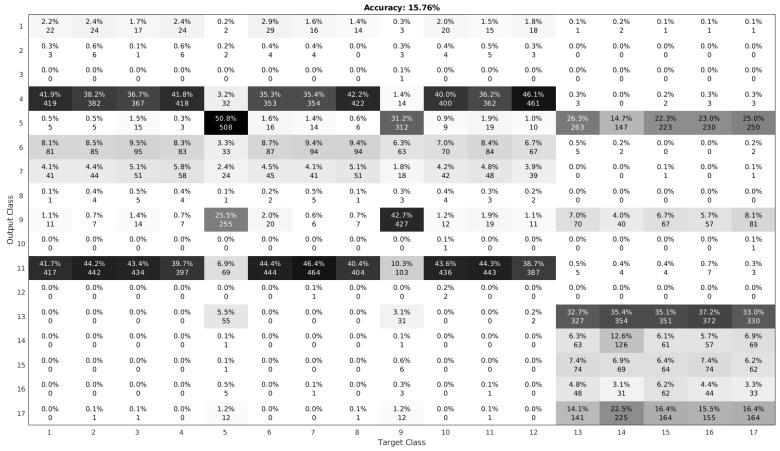


Figure 21: Confusion and classification matrix of testing data with the learning rate of 5×10^{-4}. The accuracy is 15.76%

7.2.2 Learning Rate: 1×10^{-4}

The network in this section has lower learning and is using the tuned network of higher learning rate in previous section. The accuracy is higher than the previous network but still mis-classifies many classes. Unlike the previous section, which mainly classifies into the 4^{th} and the 11^{th} class, this network has distributed the classes more evenly to other classes. Therefore, it has higher accuracy than the previous one. The classification matrix and the confusion matrix is shown here. (Figure 22 23 24).

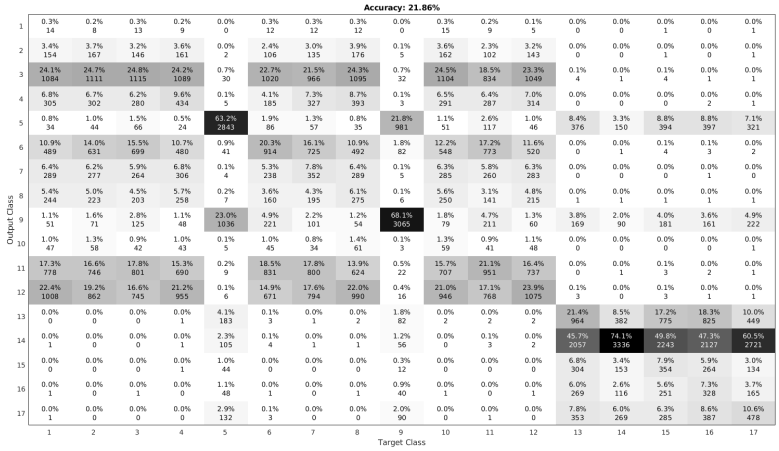


Figure 22: Confusion and classification matrix of the training data under lower learning rate with accuracy of 21.86%.

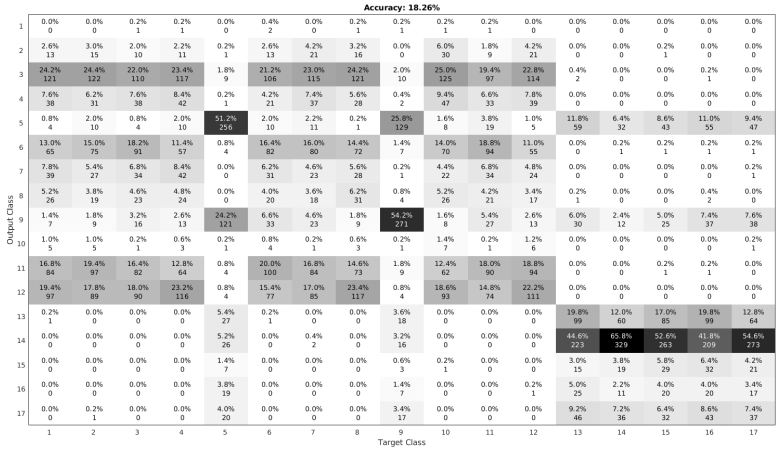


Figure 23: Confusion and classification matrix of the validation data under lower learning rate with accuracy of 18.26%.

7.2.3 conclusion

To cross validate the accuracy under different learning rate, refer to Table 5. In this step, there are some reasons that cause the low accuracy. First, this is probably caused by the

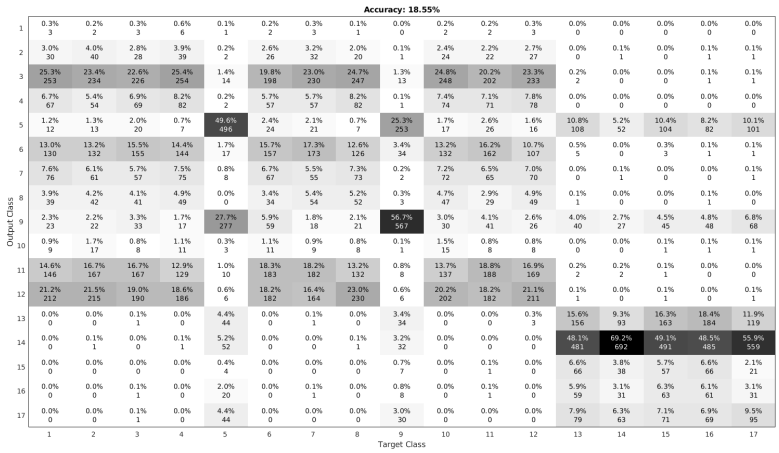


Figure 24: Confusion and classification matrix of the testing data under lower learning rate with accuracy of 18.55%.

Learning Rate	5×10^{-4}	1×10^{-4}
Training	17.82%	21.86%
Validation	15.68%	18.26%
Testing	15.76%	18.55%

Table 5: Accuracy of each learning rate on each augmented data in the start network

randomness of the augmentation. The convolutional neural network with such structure is not able to recognize such varying randomness. Second, the number of pixels in every image of the augmented data is $\frac{1}{4}$ of the original image (the original images are 256×256, while the augmented images are 128×128). The amount of the information has reduced. Therefore we supposed that this is one of the reasons that causes this unsatisfying result.

With lower learning rate, the accuracy performance can be better than the high learning rate. However, the improvement of using lower learning rate is not significant. The learning rate has some positive influence on accuracy but not too influential.

8 Step 3 – Building the Neural Network

In this step, we created 2 different kinds of network structure. One is with more filters in a layer but with less layer. The other is with more layers but less filters in each layers. We build these two different structures to see which influence more on the accuracy of neural network.

8.1 Skinny Network

The skinny network is the network with more layers but with less filter numbers. In this project, we use 24 layers of network structure.

$$[CONV \rightarrow ReLU \rightarrow CONV \rightarrow ReLU \rightarrow MAX] * 3 \rightarrow [FC \rightarrow ReLU] * 2 \quad (7)$$

$$\rightarrow DropOut \rightarrow FC \rightarrow SOFTMAX \quad (8)$$

In order to prevent overfitting, the dropout rate is set to 40%. The hyperparameters of each CONV layer are shown as:

Hyper- parameter	filter size	filter number	stride size	Padding number	number of parameters
$CONV_1$	8×8	10	[1 1]	[2 2 2 2]	650
$CONV_2$	7×7	20	[1 1]	[2 2 2 2]	1000
$CONV_3$	6×6	30	[1 1]	[1 1 1 1]	1110
$CONV_4$	5×5	30	[1 1]	[1 1 1 1]	780
$CONV_5$	3×3	20	[1 1]	[2 2 2 2]	200
$CONV_6$	2×2	10	[2 2]	[1 1 1 1]	40

Table 6: Filter size and number of each CONV layer

We trained 10 epochs with batch size of 100.

8.1.1 Skinny Net on Original Data

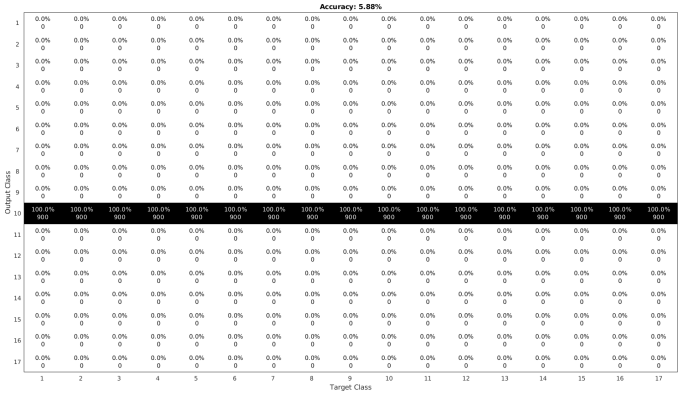


Figure 25: The confusion and classification matrix of original training data with learning rate of 5×10^{-4}. The accuracy is 5.88%. As the figure shows, all the classes are classified to class 10.

8.1.2 conclusion

The accuracy as well as the classification and confusion matrices of the learning rate of 1×10^{-4} is the same as those of 5×10^{-4}. Therefore it is not presented.

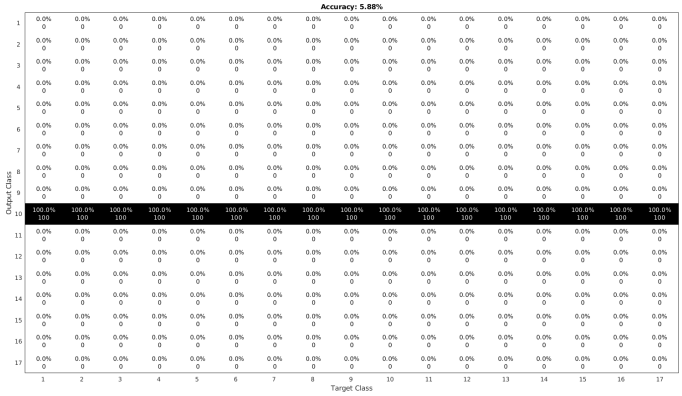


Figure 26: The confusion and classification matrix of original validation data with learning rate of 5×10^{-4}. The accuracy is 5.88%. As the figure shows, all the classes are classified to class 10.

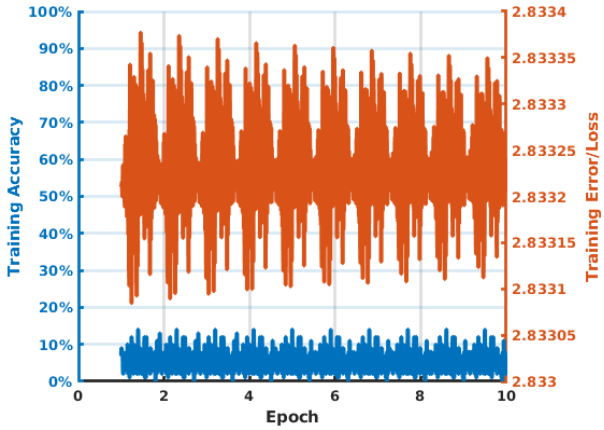


Figure 27

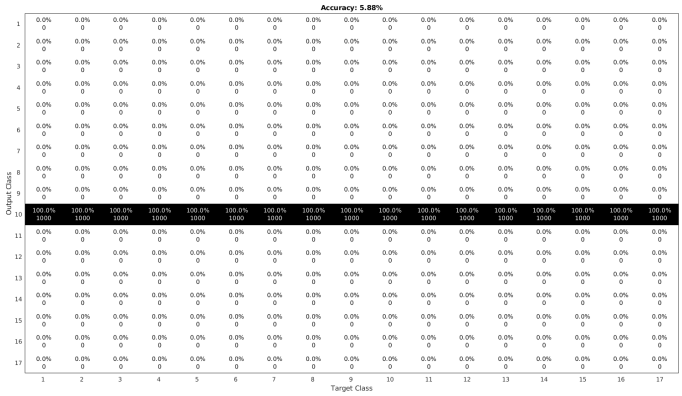


Figure 28: The confusion and classification matrix of original testing data with learning rate of 5×10^{-4}. The accuracy is 5.88%. As the figure shows, all the classes are classified to class 10.

8.1.3 Skinny Net on Augmented Data

Same as previous section, the matrices and the accuracy of 1×10^{-4} is the same as those of 5×10^{-4} so it is not presented.

The accuracy of the skinny network is 5.88%, which is equal to $\frac{1}{17}$. The network is randomly guessing a class of an image. In these cases, they are separating the images into the same class.

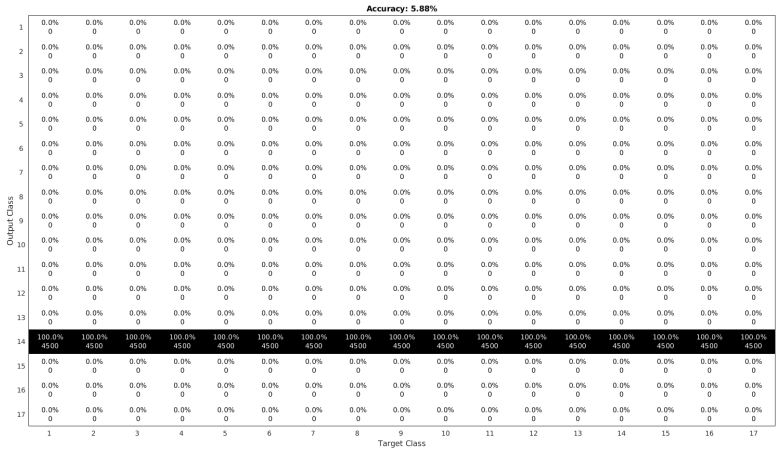


Figure 29: The confusion and classification matrix of original training data with learning rate of 1×10^{-4}. The accuracy is 5.88%. As the figure shows, all the classes are classified to class 14.

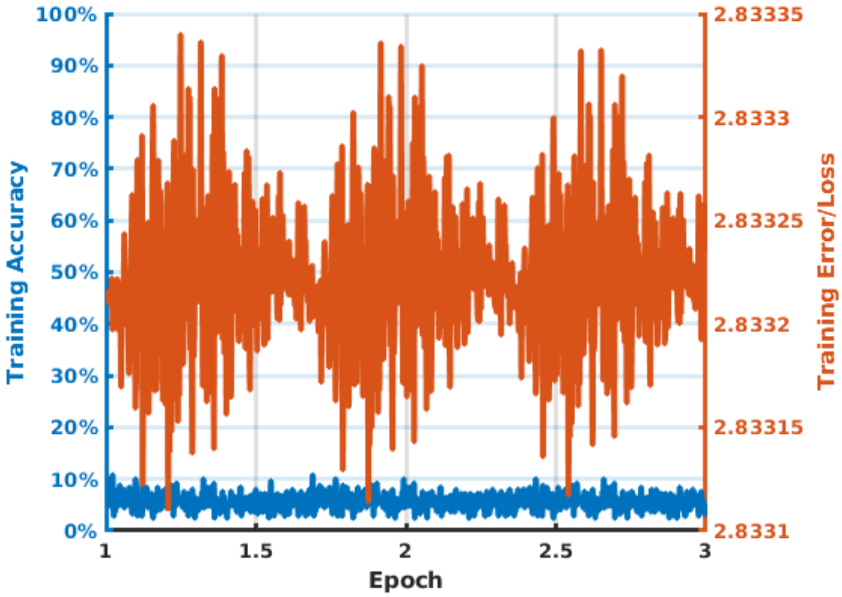


Figure 30

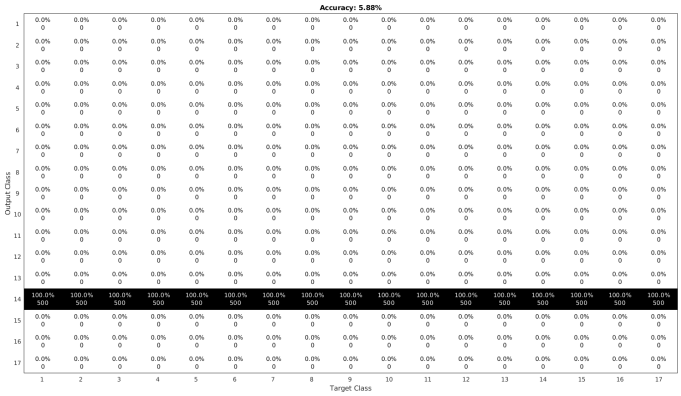


Figure 31: The confusion and classification matrix of original training data with learning rate of 1×10^{-4}. The accuracy is 5.88%. As the figure shows, all the classes are classified to class 14.

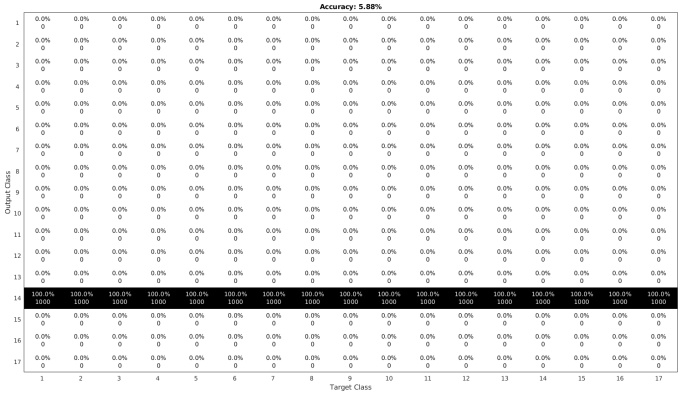


Figure 32: The confusion and classification matrix of original training data with learning rate of 1×10^{-4}. The accuracy is 5.88%. As the figure shows, all the classes are classified to class 14.

8.2 Wide Network

This one is with more filters in each layer but with less layers. The structure is $[CONV \rightarrow ReLU \rightarrow POOL] * 2 \rightarrow FC \rightarrow ReLU \rightarrow DROP \rightarrow FC \rightarrow SOFTMAX$ The dropout rate is 25%. The hyperparameters are shown as Table 7

Hyper-parameter	filter size	filter number	stride size	Padding number	number of parameters
$CONV_1$	5×5	40	[2 2]	[1 1 1 1]	1040
$CONV_2$	3×3	80	[1 1]	[1 1 1 1]	800

Table 7: Filter size and number of each CONV layer

We trained 10 epochs and the batch size of 100 on each the dataset.

8.2.1 Wide Net on Original Data

The confusion and the classification matrix of the training data is shown in the same figure:

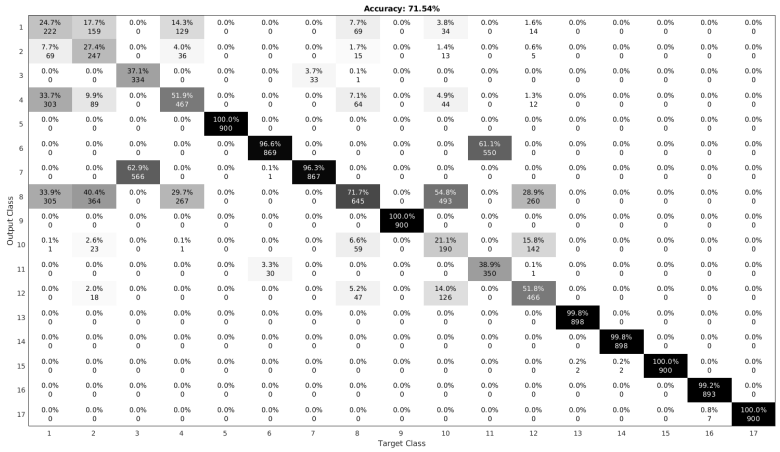


Figure 33: Confusion matrix and classification matrix of the training data on wide network with learning rate 5×10^{-4}. The accuracy is 71.54%

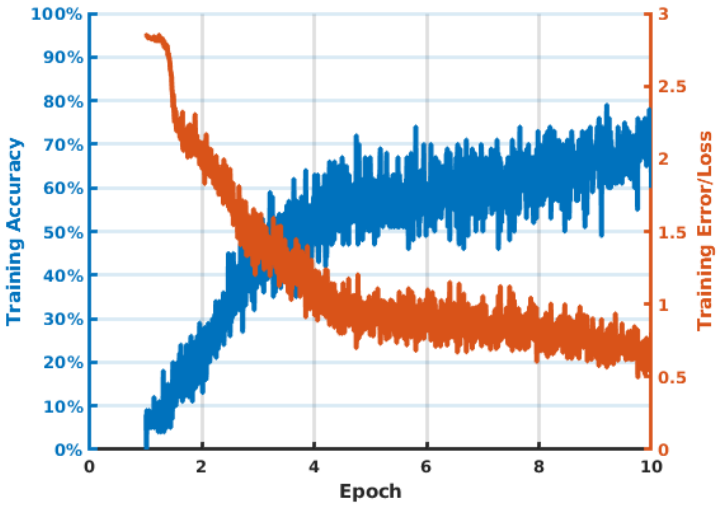


Figure 34: Training accuracy versus training loss of wide network on original data.

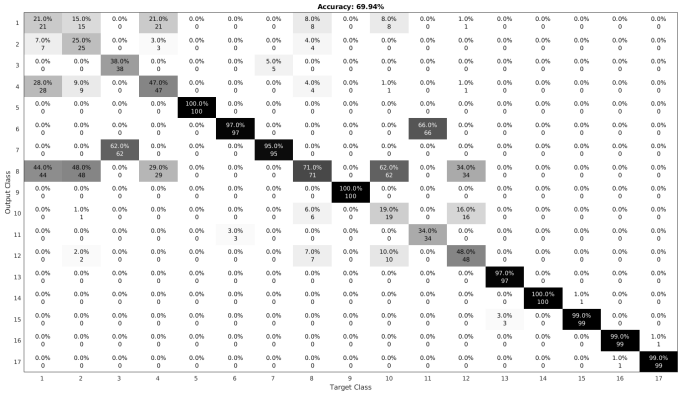


Figure 35: Confusion matrix and classification matrix of the validation data on wide network with learning rate 5×10^{-4}. The accuracy is 69.94%

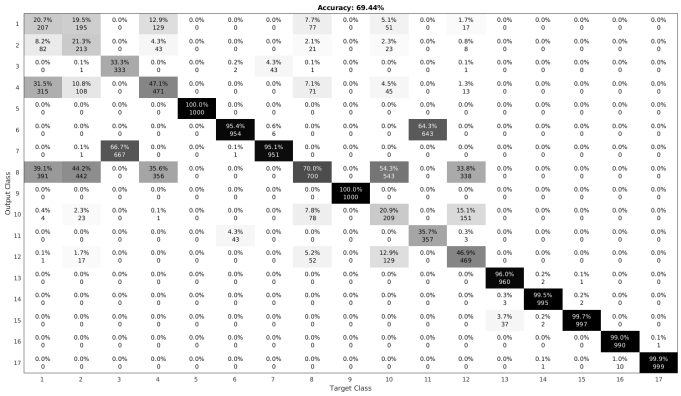


Figure 36: Confusion matrix and classification matrix of the testing data on wide network with learning rate 5×10^{-4}. The accuracy is 69.44%

Applying this already fine-tuned network with a smaller learning rate, we get a better result. The accuracy of the training dataset is 84.67% (Figure 37).

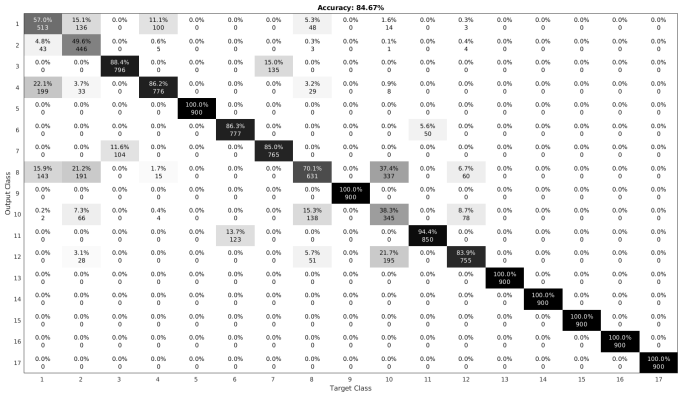


Figure 37: Confusion matrix and classification matrix of the training data on wide network with learning rate 1×10^{-4}. The accuracy is 84.67%

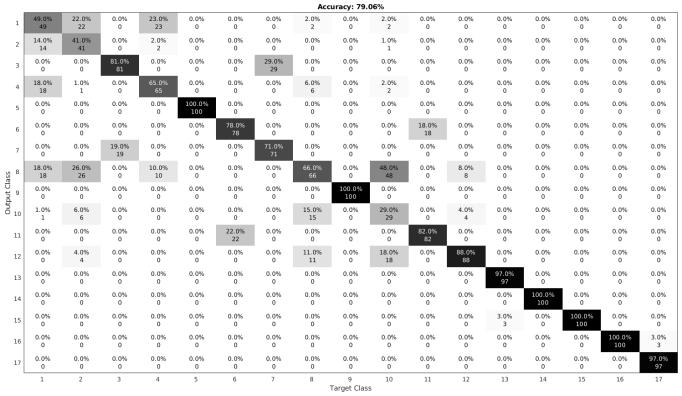


Figure 38: Confusion matrix and classification matrix of the training data on wide network with learning rate 1×10^{-4}. The accuracy is 79.06%

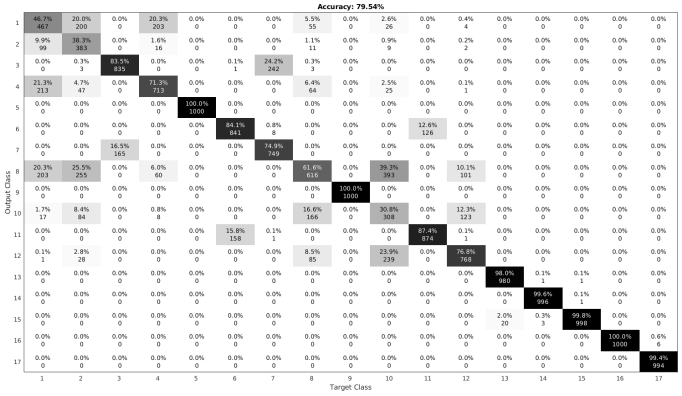


Figure 39: Confusion matrix and classification matrix of the training data on wide network with learning rate 1×10^{-4}. The accuracy is 79.54%

The result and the accuracy is shown as follow:

Learning Rate	5×10^{-4}	1×10^{-4}
Training	71.54%	84.67%
Validation	69.94%	79.06%
Testing	69.44%	79.54%

Table 8: Accuracy of each learning rate on each original data in wide network

8.2.2 Wide Net on Augmented Data

Applying the wide network on the augmented data will get us better results comparing to the first network.

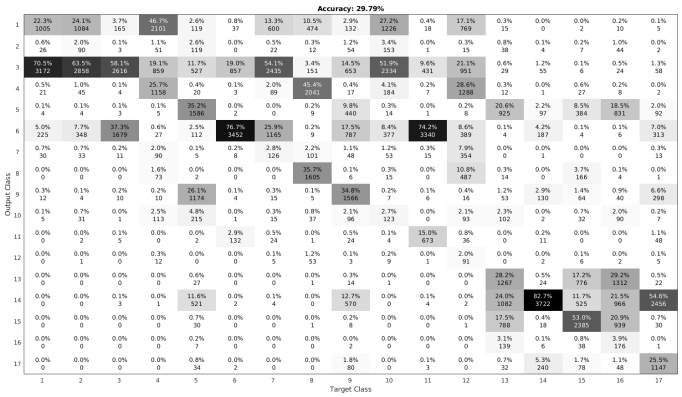


Figure 40: Confusion matrix and classification matrix of the training data with learning rate 5×10^{-4}. The accuracy is 29.79%

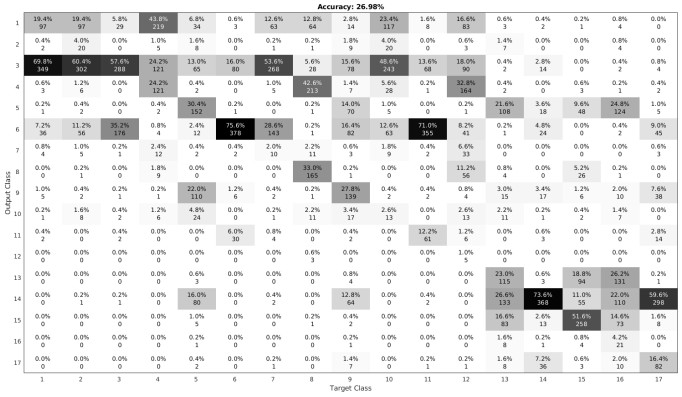


Figure 41: Confusion matrix and classification matrix of the training data with learning rate 5×10^{-4}. The accuracy is 26.98%

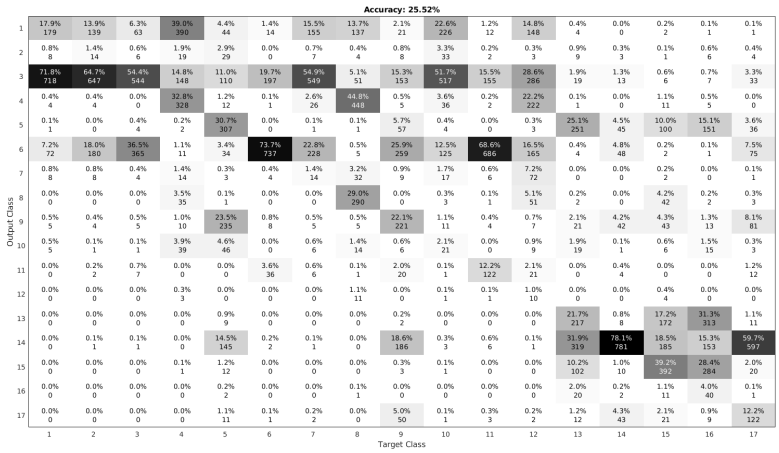


Figure 42: Confusion matrix and classification matrix of the training data with learning rate 5×10^{-4}. The accuracy is 25.52%

The performance of the lower learning rate with fine-tuned network is shown as:

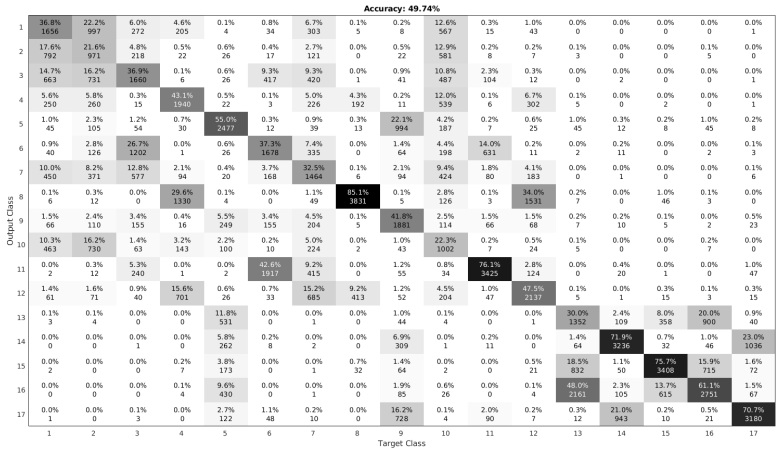


Figure 43: Confusion matrix and classification matrix of the training data with learning rate 1×10^{-4}. The accuracy is 49.74%

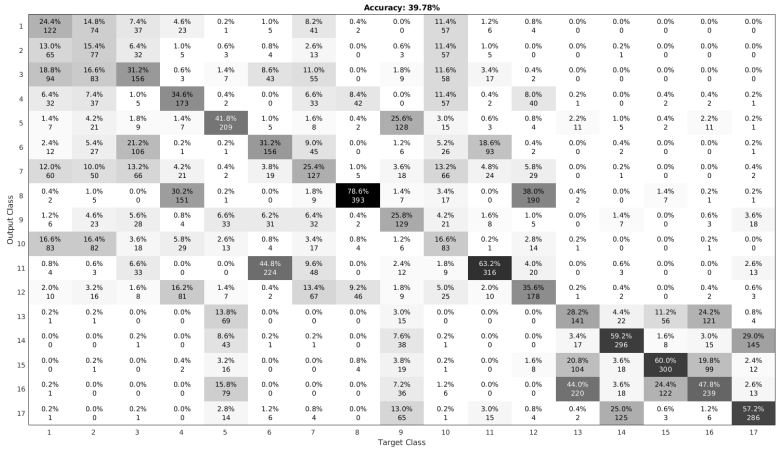


Figure 44: Confusion matrix and classification matrix of the training data with learning rate 1×10^{-4}. The accuracy is 39.78%

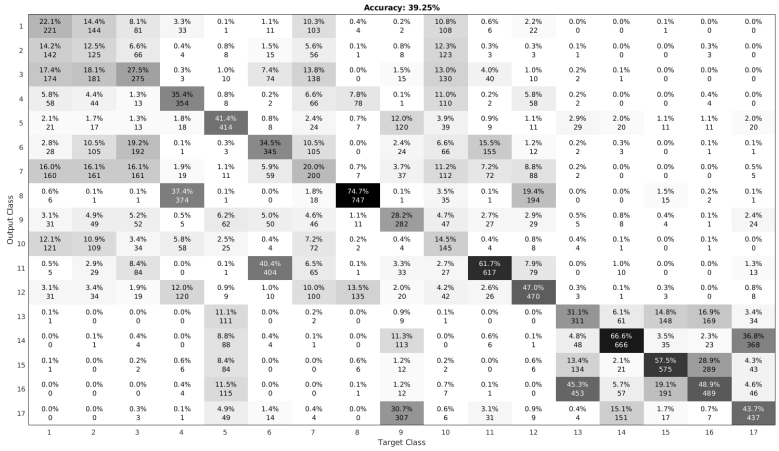


Figure 45: Confusion matrix and classification matrix of the training data with learning rate 1×10^{-4}. The accuracy is 39.25%

Learning Rate	5×10^{-4}	1×10^{-4}
Training	29.79%	49.74%
Validation	26.98%	39.78%
Testing	25.52%	39.25%

Table 9: Accuracy of different learning rate on each augmented data in the wide network

9 Conclusion

Compare the table 2 with table 8, it is clear that the difference of accuracy between two structure is not significant. However, when it comes to comparing table 5 and 9, the accuracy of the wide network significantly increases. This is because the wide network has enough parameters to fit into the pattern of the augmented data. In conclusion, the neural network is sensitive to their hyper-parameters. Also, the numbers of parameters of each layer are important for the network to fit into the datasets. It is not just the large amount of data, the numbers of the parameters also matter. However, naively increases the number of parameter will result in overfitting. Thus, to dropout some neurons at some moments is an efficient way to prevent overfitting.

References

[1] F.-F. Li, A. Karpathy, and J Johnson *CS231n: Convolutional Neural Network for Visual Recognition* Stanford University, 2016

[2] C. Olah *Understanding convolutions* GITHUB blog, posted on August, vol. 27, p. 2015, 2015

[3] H.-Y. Li *Machine Learning* National Taiwan University, 2017