PHUNG THANH HAI (馮清海)

Student ID : 0880803

REPORT

DEEP LEARNING HOMEWORK II

I. MNIST

1. Normal case

In this case, I have been used 7 layers CNN based high – level API to predict the label of MNIST. I used Tensorflow to read the image data set. The detail implementation layer can be seen below:

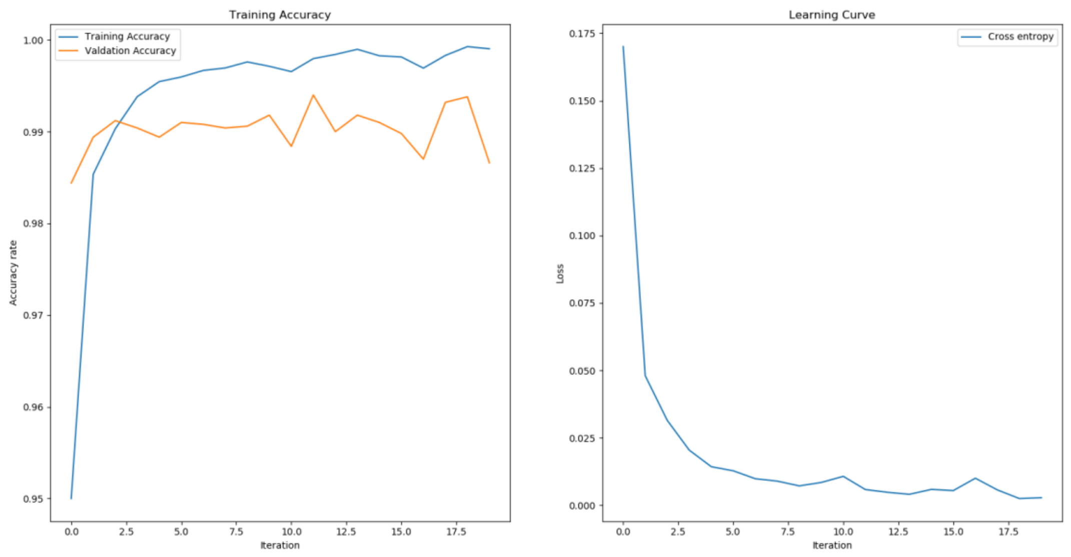
Network Architecture

Network Architecture	32-1-128-1-1-256-10
Error Rate Training/Validation Training	**99.78% / 98.76%**
Epochs	20
Learning Rate	0.01
Batch Size	128
Training Size / Validate Size/ Testing Size	55000/5000/10000

In this architecture, I used the first layer is Conv2D with 32 filters and *kernel_size* is 3x3, then is MaxPooling2D with *pool_size = 3x3* the Conv2D with 128 filters and *kernel_size* is 3x3 and then the MaxPooling2D again with *pool_size* is 2x2 then after that is the Flatten layer and Dense with different architecture for every single layer, *256 filters* with ReLU activation function and *10 filters* with Softmax activation function, respectively.

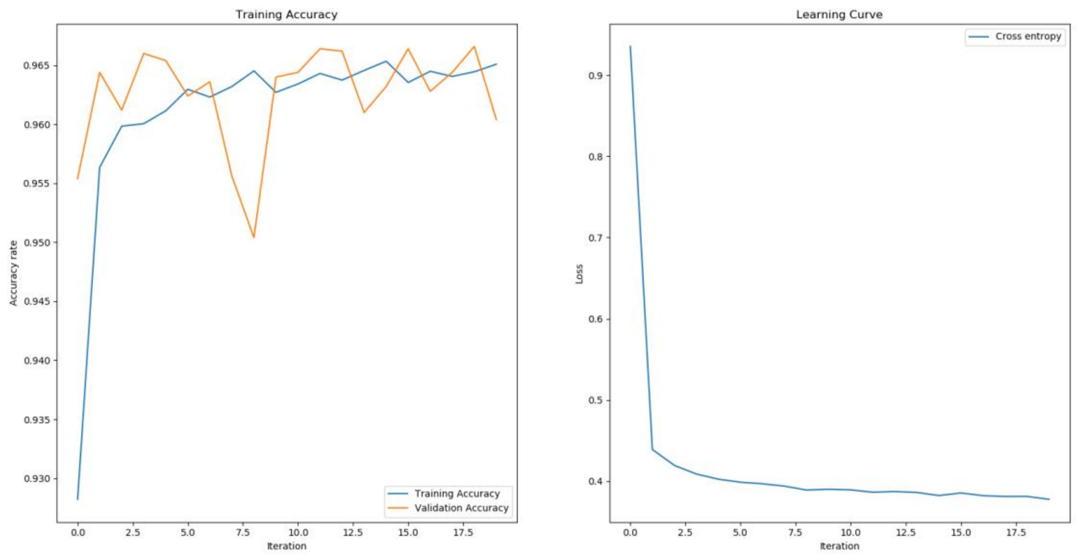
Design the filter size and strides:

In this case, I just adjust the size of *strides* and *kernel size* from 3x3 to 5x5. However, we can see the effect by the different of **Training Accuracy, Validation Accuracy** of the first case and second case. When we change the kernel size, the accuracy is lower but more balance compares to each other, respectively.



```
Training Accuracy: 0.9976181818181818, Validation Accuracy: 0.9876
```

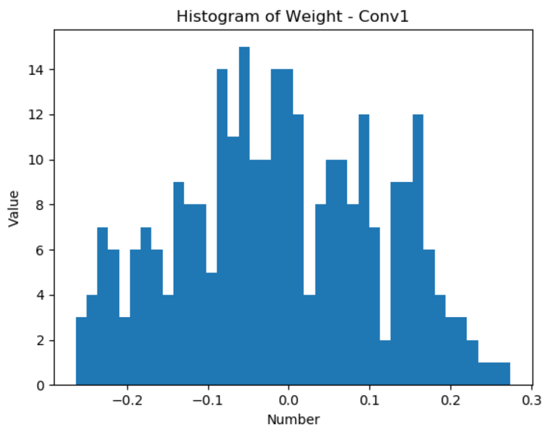
(a)



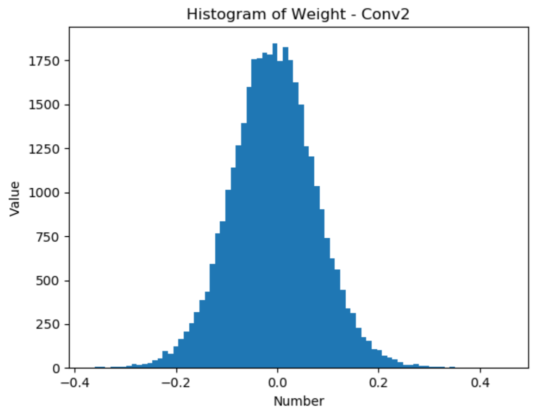
```
Training Accuracy: 0.9608000000086697, Validation Accuracy: 0.9604
```

(b)

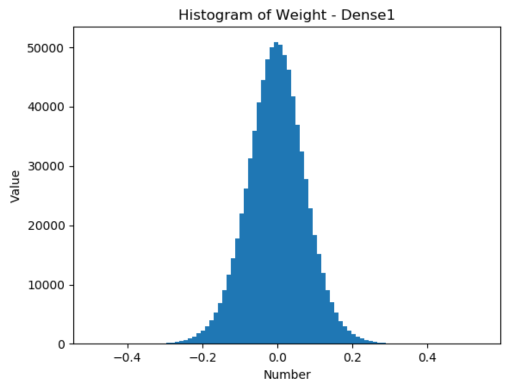
Figure 1. Accuracy with different CNN architecture. (a) kernel size = 3x3 and strides (b). kernel size = 5x5 and strides = (2,1)



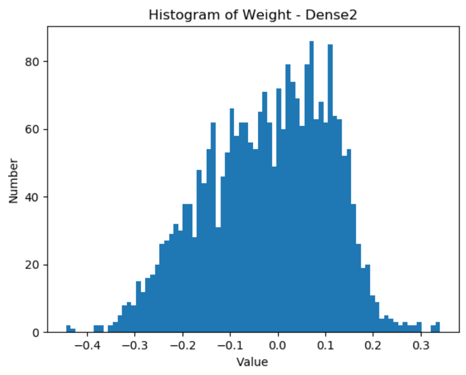
(a)



(b)



(c)



(d)

Figure 2. Result of CNN model; (a) Histogram of Weight - Conv1; (b) Histogram of Weight – Conv2; (c) Histogram of Weight – Dense1; (d) Histogram of Weight – Dense2

2. Show some examples of correctly classified and incorrect classified

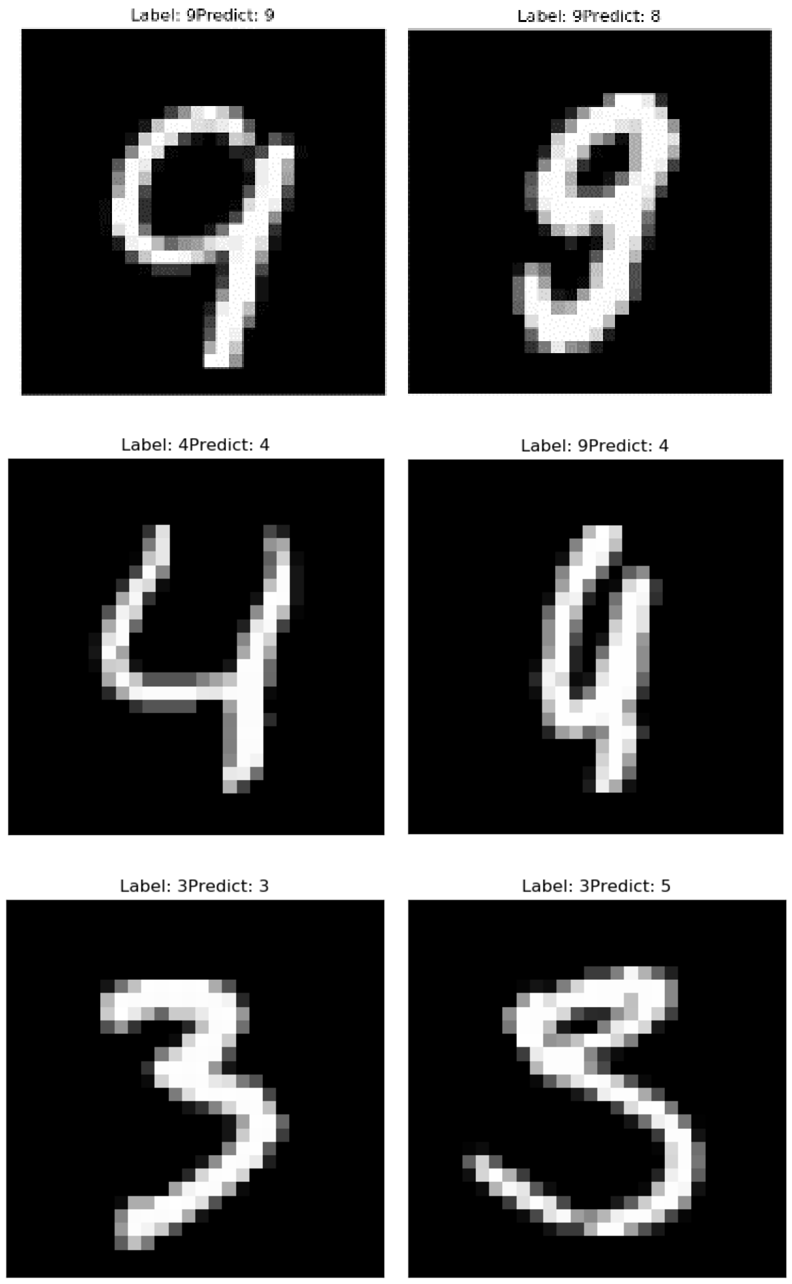


Figure 3: Examples of correct classified and incorrect classified images

3. Feature map

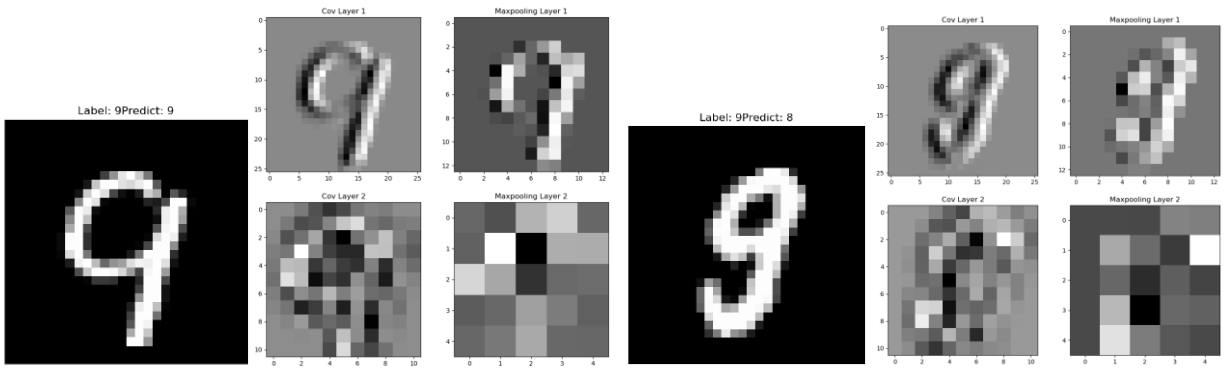
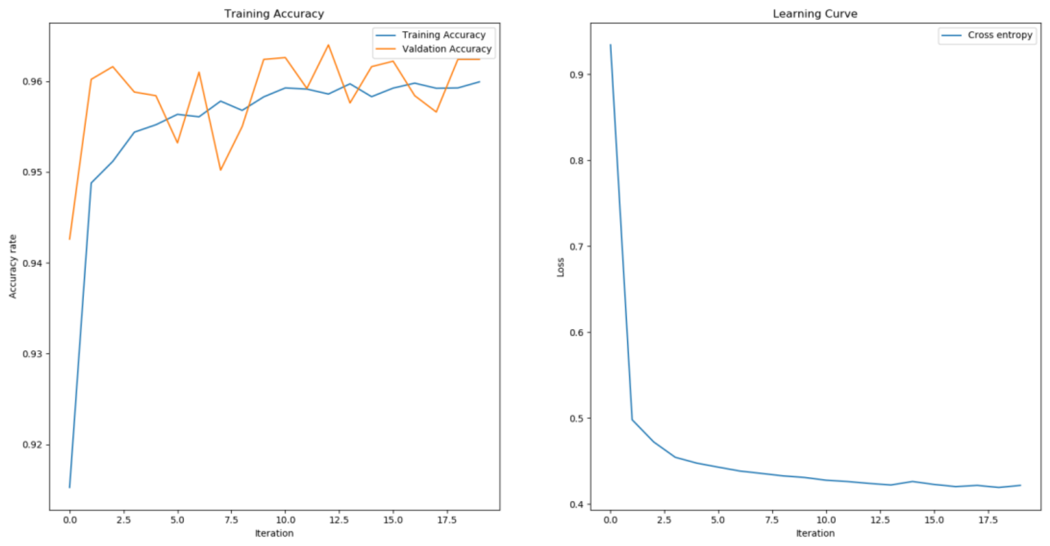


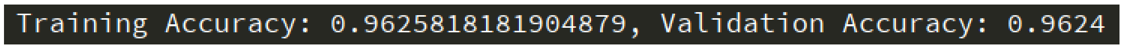
Figure 4: Examples of correct classified and incorrect classified images

4. L2 regularization

Following we are adding the L2 regularize and see the effect on the CNN model based this equation as shown:

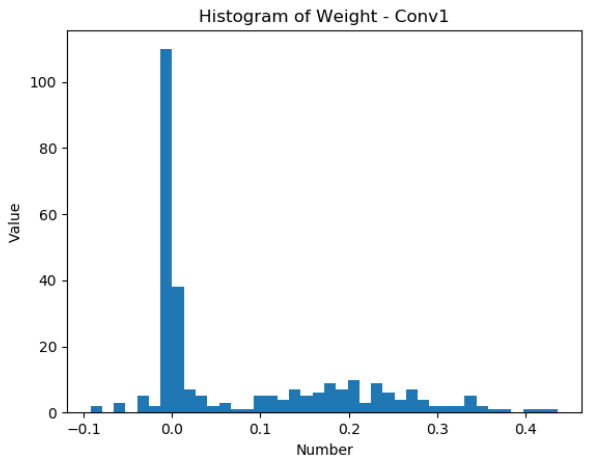
$$E = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk} \ln t_{nk} + \alpha \|w\|_2^2$$



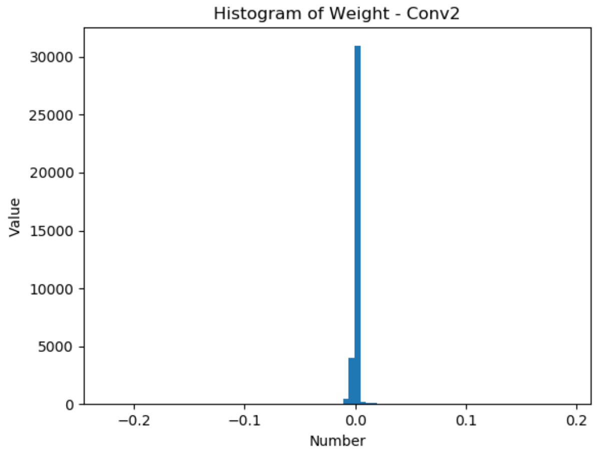


```
Training Accuracy: 0.9625818181904879, Validation Accuracy: 0.9624
```

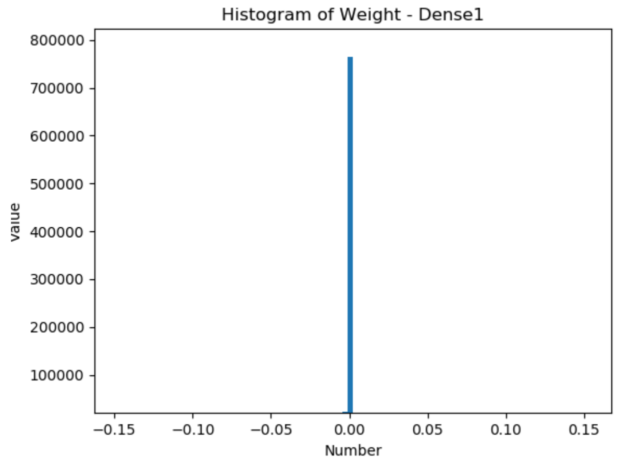
Figure 5: Accuracy of CNN architecture added l2 regularized.



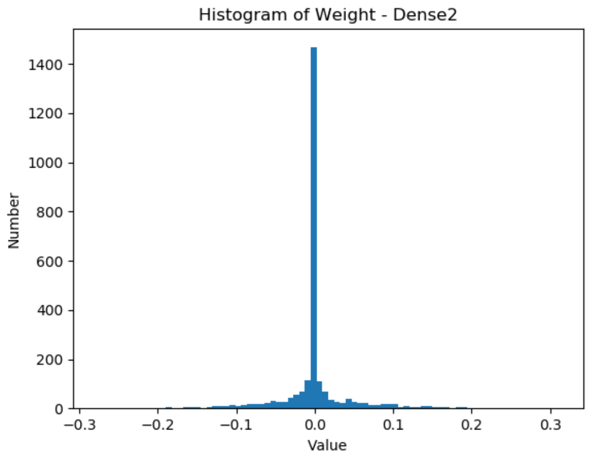
(a)



(b)



(c)



(d)

Figure 6. Result of CCN with L2; (a) Histogram of Weight - Conv1; (b) Histogram of Weight – Conv2; (c) Histogram of Weight – Dense1; (d) Histogram of Weight – Dense2

Comment: with the added L2, we don't see the much different between the normal case with the L2 norm case. Unfortunately, it may due to the dataset is not so complicate enough for the prediction, so this model is working without overfitting. But for the more complicate dataset, may be the regularize L2 will extend his effect to solve overfitting. We will see it soon in the second problem of this report.

II. CIFAR-10

1. Normal case

In this case, I have been used 7 layers CNN based high – level API to predict the label of MNIST. I used Tensorflow to read the image data set. The detail implementation layer can be seen below:

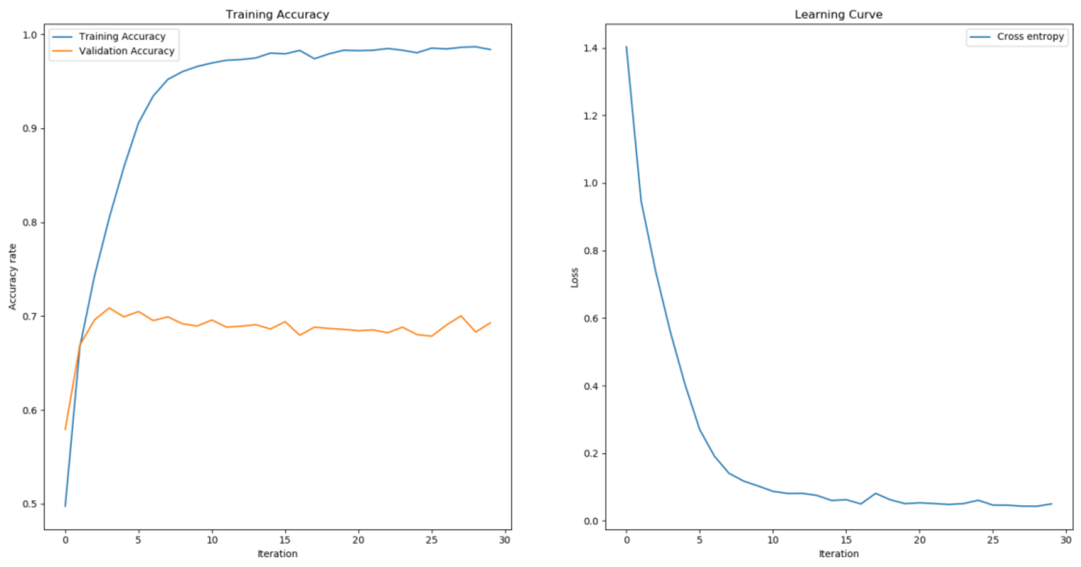
Network Architecture

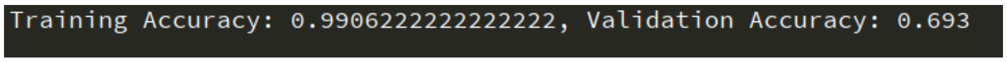
Network Architecture	32-64-1-128-1-1-256-128-10
Error Rate Training/Validation Training	**99.06% / 69.3%**
Epochs	30
Learning Rate	0.01
Batch Size	128
Training Size / Validate Size/ Testing Size	45000/5000/10000

In this architecture, I used the first layer is Conv2D with 32 filters and *kernel_size* is 3x3, and the second is is Conv2D with 64 filters and *kernel_size* is 3x3, then is MaxPooling2D with *pool_size = 3x3* the Conv2D with 128 filters and *kernel_size* is 3x3 and then the MaxPooling2D again with *pool_size* is 2x2 then after that is the Flatten layer and Dense with different architecture for every single layer, *256 filters* with ReLU activation function, *128 filters* with ReLU activation function and *10 filters* with Softmax activation function, respectively.

Design the filter size and strides:

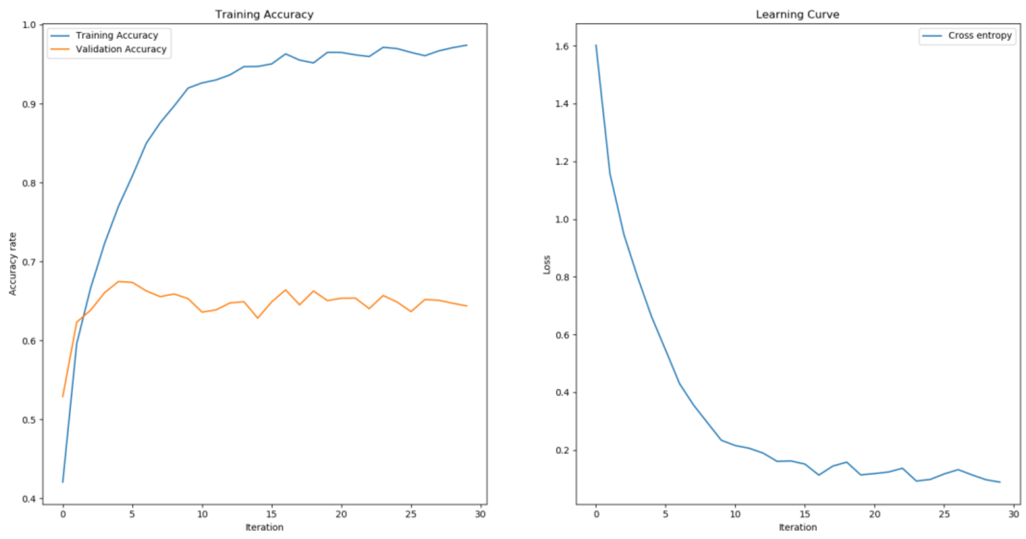
In this case, I just adjust the size of *strides* and *kernel size* from 3x3 to 5x5. However, we can see the effect by the different of **Training Accuracy, Validation Accuracy** of the first case and second case. When we change the kernel size, the accuracy is lower but more balance compares to each other, respectively.

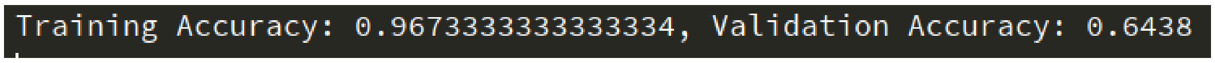




```
Training Accuracy: 0.9906222222222222, Validation Accuracy: 0.693
```

(a)

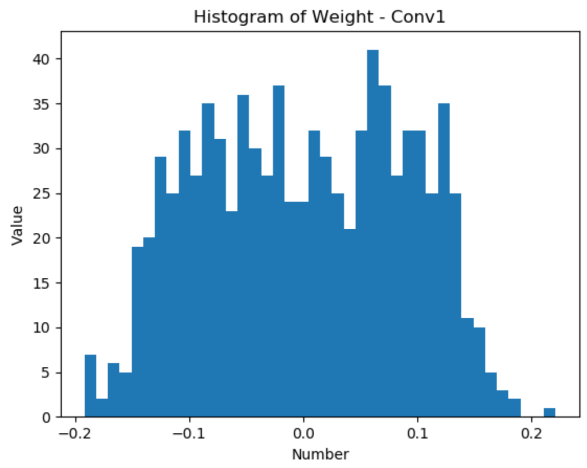




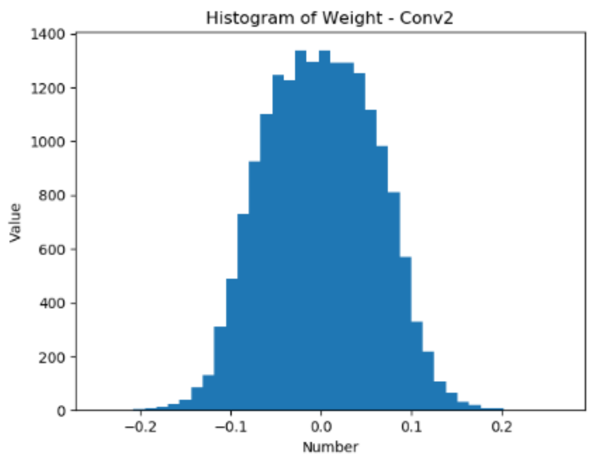
```
Training Accuracy: 0.9673333333333334, Validation Accuracy: 0.6438
```

(b)

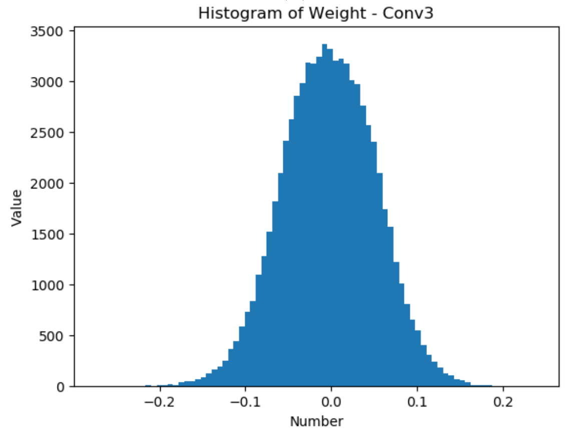
Figure 7. Accuracy with different CNN architecture. (a) kernel size = 3x3 and strides (b). kernel size = 5x5 and strides = (2,1)



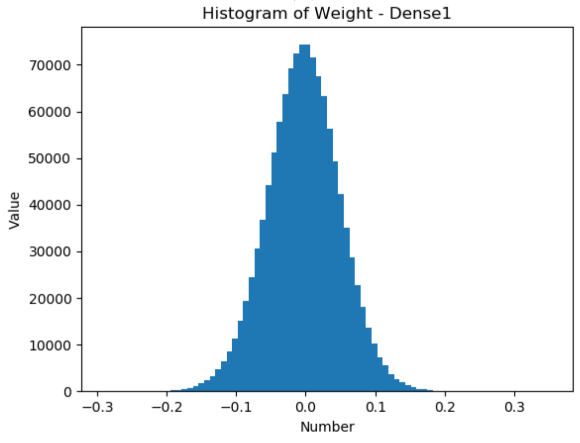
(a)



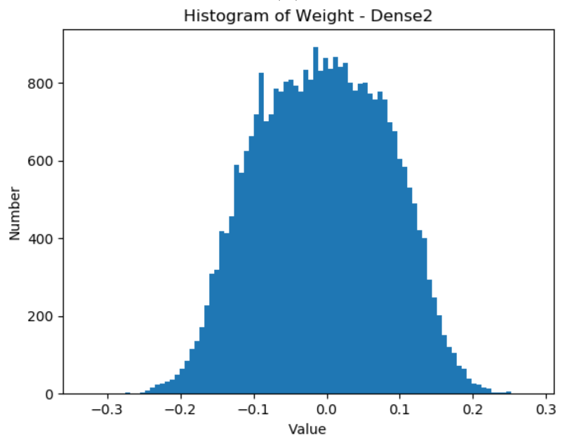
(b)



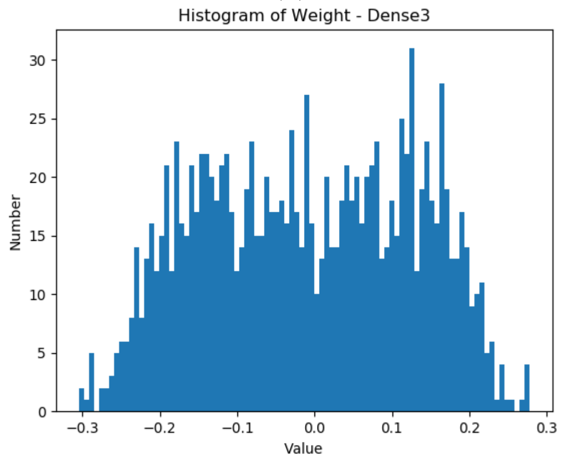
(c)



(d)



(e)



(f)

Figure 8. Result of CNN model; (a) Histogram of Weight - Conv1; (b) Histogram of Weight – Conv2; (c) Histogram of Weight – Conv2; (d) Histogram of Weight – Dense1; (e) Histogram of Weight – Dense2; (f) Histogram of Weight – Dense2

2. Show some examples of correctly classified and incorrect classified

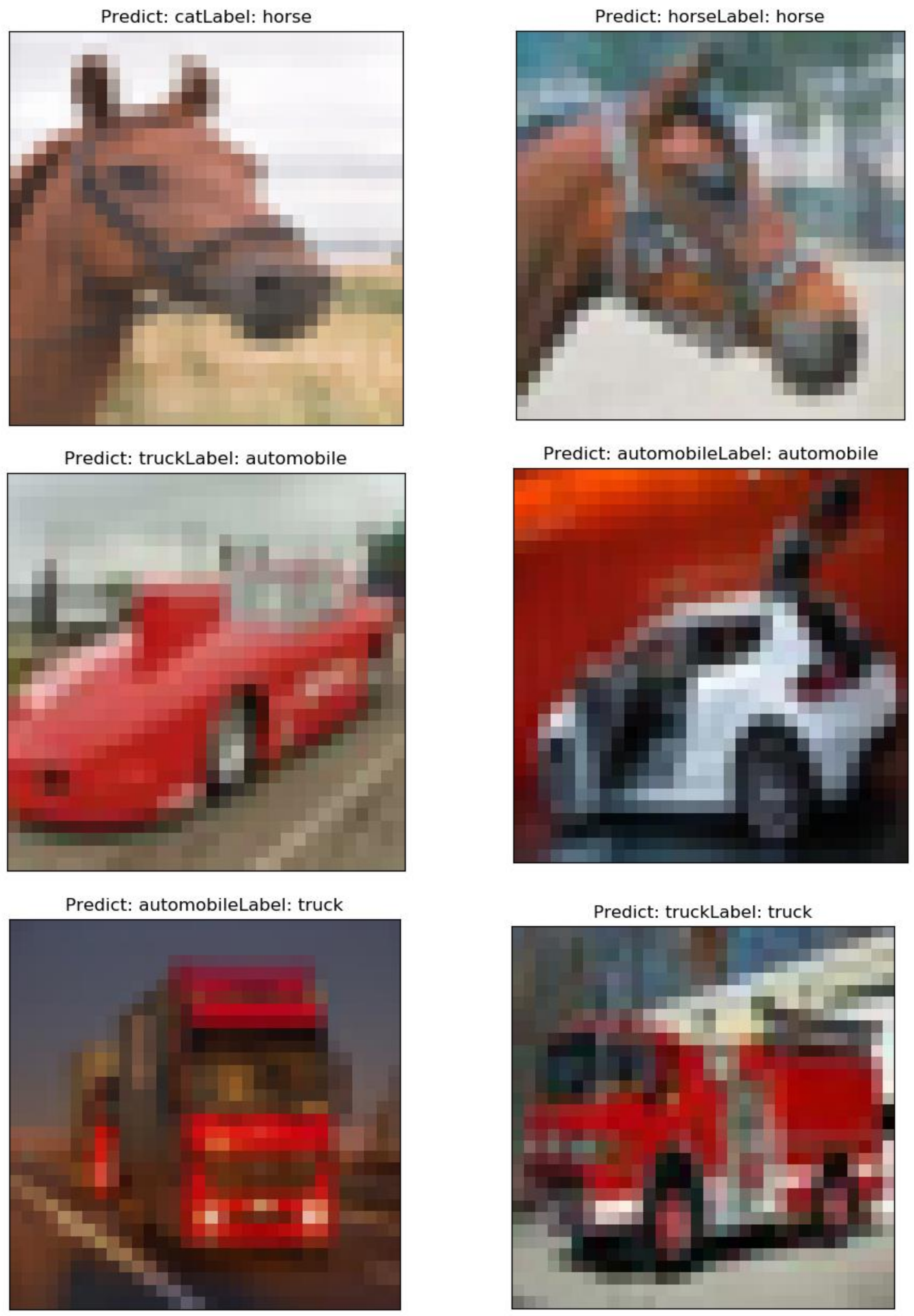

Predict: catLabel: horse
Predict: horseLabel: horse
Predict: truckLabel: automobile
Predict: automobileLabel: automobile
Predict: automobileLabel: truck
Predict: truckLabel: truck

Figure 9: Examples of correct classified and incorrect classified images

3. Feature map

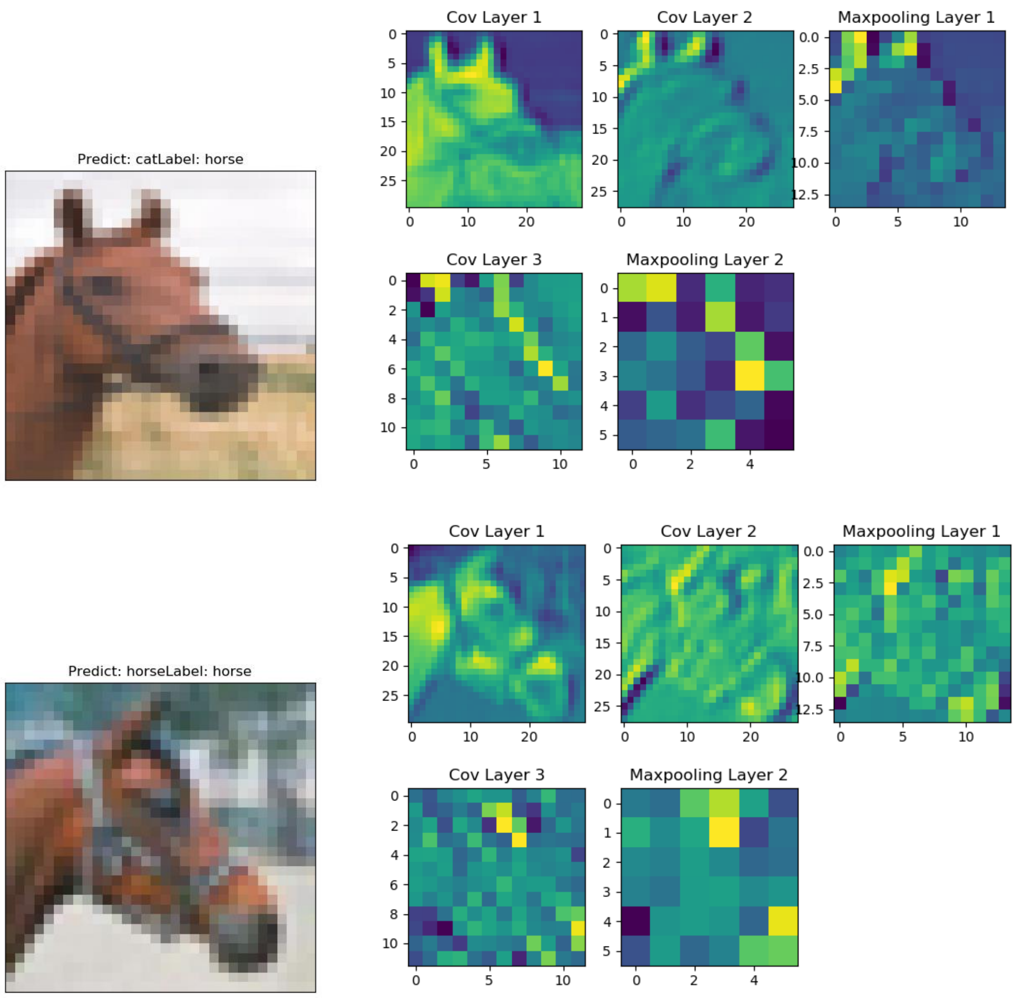
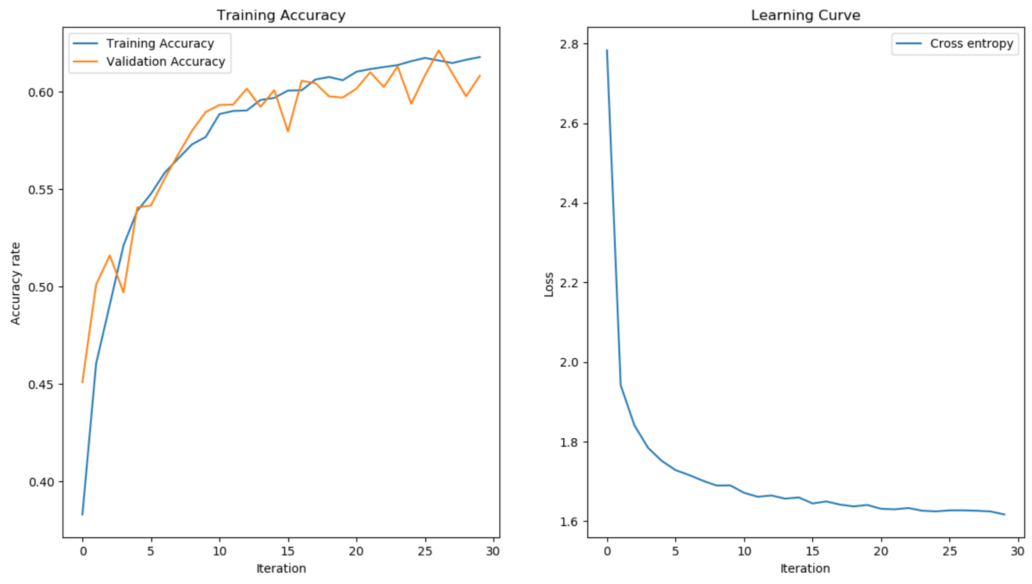


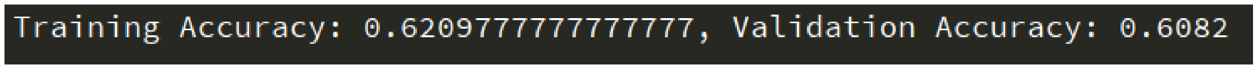
Figure 10: Examples of correct classified and incorrect classified images

4. L2 Regularization

Following we are adding the L2 regularize and see the effect on the CNN model based this equation as shown:

$$E = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk} \ln t_{nk} + \alpha \|w\|_2^2$$

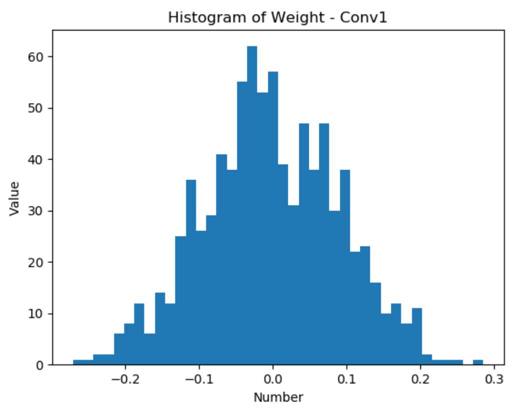




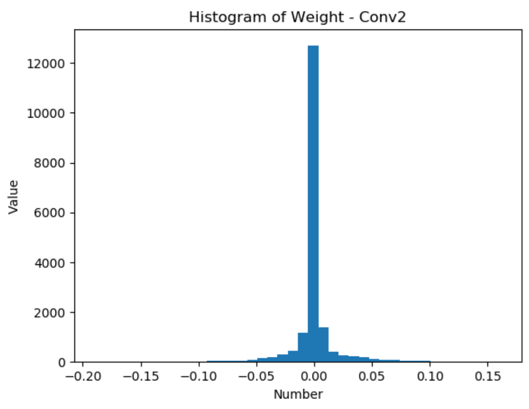
```
Training Accuracy: 0.6209777777777777, Validation Accuracy: 0.6082
```

Figure 11: Accuracy of CNN architecture added l2 regularized.

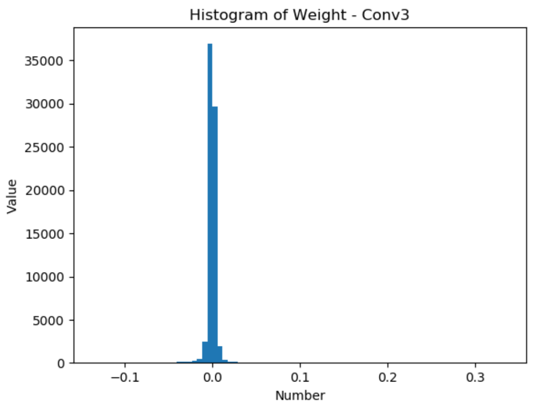
Comment: with the added L2, we see the much different between the normal case with the L2 norm case. Unfortunately, it due to the dataset is so complicate for the prediction, so this model may work with overfitting. So in this case study, we see the robust effect of L2 regularization on the case of overfitting as mentioned above.



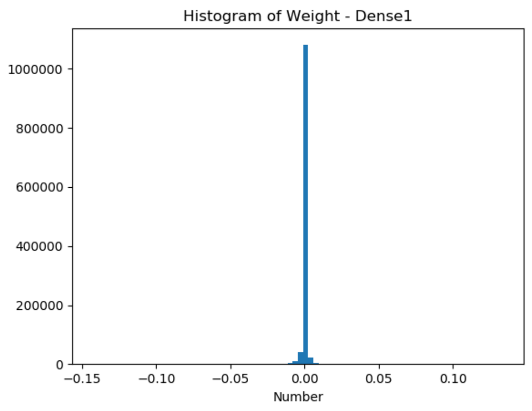
(a)



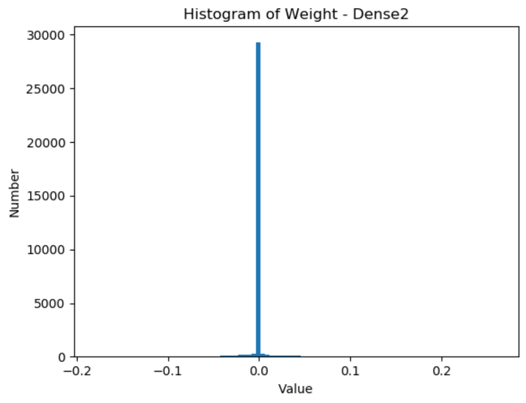
(b)



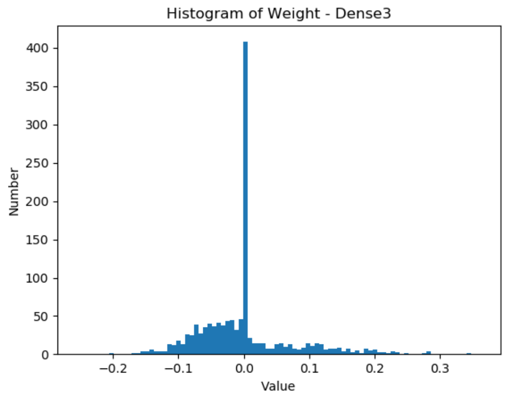
(c)



(d)



(e)

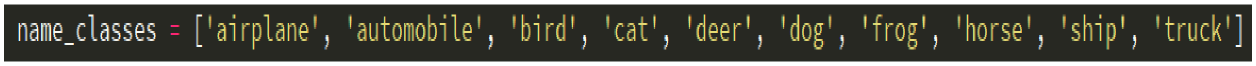


(f)

Figure 12. Result of CNN model; (a) Histogram of Weight - Conv1; (b) Histogram of Weight – Conv2; (c) Histogram of Weight – Conv2; (d) Histogram of Weight – Dense1; (e) Histogram of Weight – Dense2; (f) Histogram of Weight – Dense2

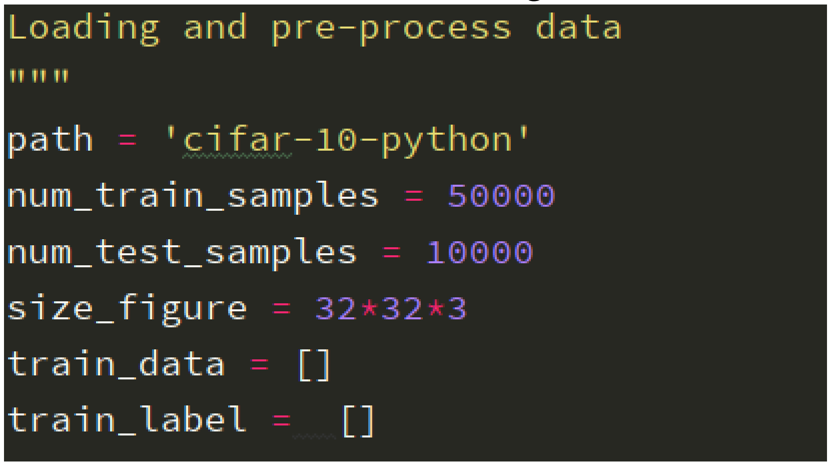
5. CIFAR-10 Dataset Pre-Processing Explanation

The original one **batch data** have **(10000x3072)** matrix expressed like array. With 10000 rows and 3072 columns. Especially 3072 columns reshape from 32x32x3 RGB photo. And then ten labels of dataset list from 0 to 9 which corresponded to ten different things as shown:



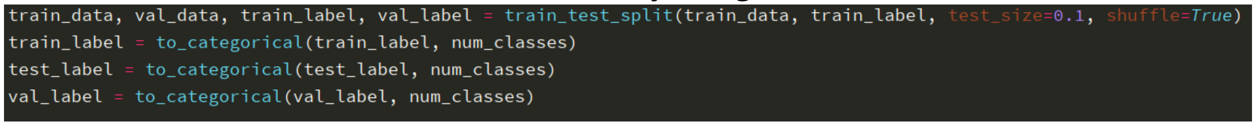
```
name_classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

We need to care about the size of images like this:



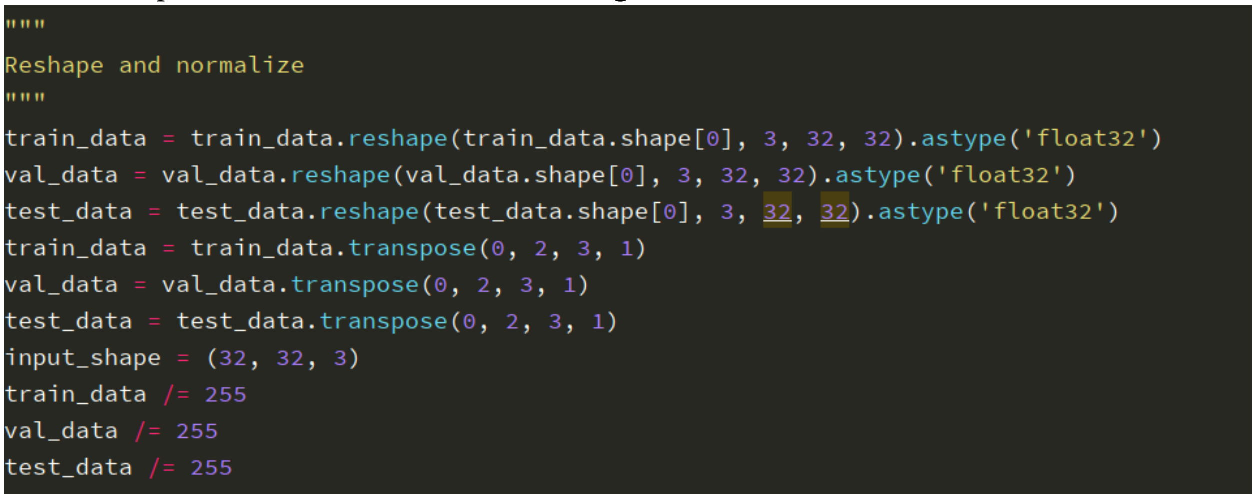
```
Loading and pre-process data
"""
path = 'cifar-10-python'
num_train_samples = 50000
num_test_samples = 10000
size_figure = 32*32*3
train_data = []
train_label =  []
```

For now, what you need to know is the output of the model. It is a set of probabilities of each class of image based on the model's prediction result. In order to express those probabilities in code, a vector having the same number of elements as the number of classes. We need to host those labels in to one vector by using One hot vector function.



```
train_data, val_data, train_label, val_label = train_test_split(train_data, train_label, test_size=0.1, shuffle=True)
train_label = to_categorical(train_label, num_classes)
test_label = to_categorical(test_label, num_classes)
val_label = to_categorical(val_label, num_classes)
```

Then reshape and normalize for further design CNN model and calculation.



```
"""
Reshape and normalize
"""
train_data = train_data.reshape(train_data.shape[0], 3, 32, 32).astype('float32')
val_data = val_data.reshape(val_data.shape[0], 3, 32, 32).astype('float32')
test_data = test_data.reshape(test_data.shape[0], 3, 32, 32).astype('float32')
train_data = train_data.transpose(0, 2, 3, 1)
val_data = val_data.transpose(0, 2, 3, 1)
test_data = test_data.transpose(0, 2, 3, 1)
input_shape = (32, 32, 3)
train_data /= 255
val_data /= 255
test_data /= 255
```