Deep Learning HW2

MNIST

1-1 The effect of stride size and filter size

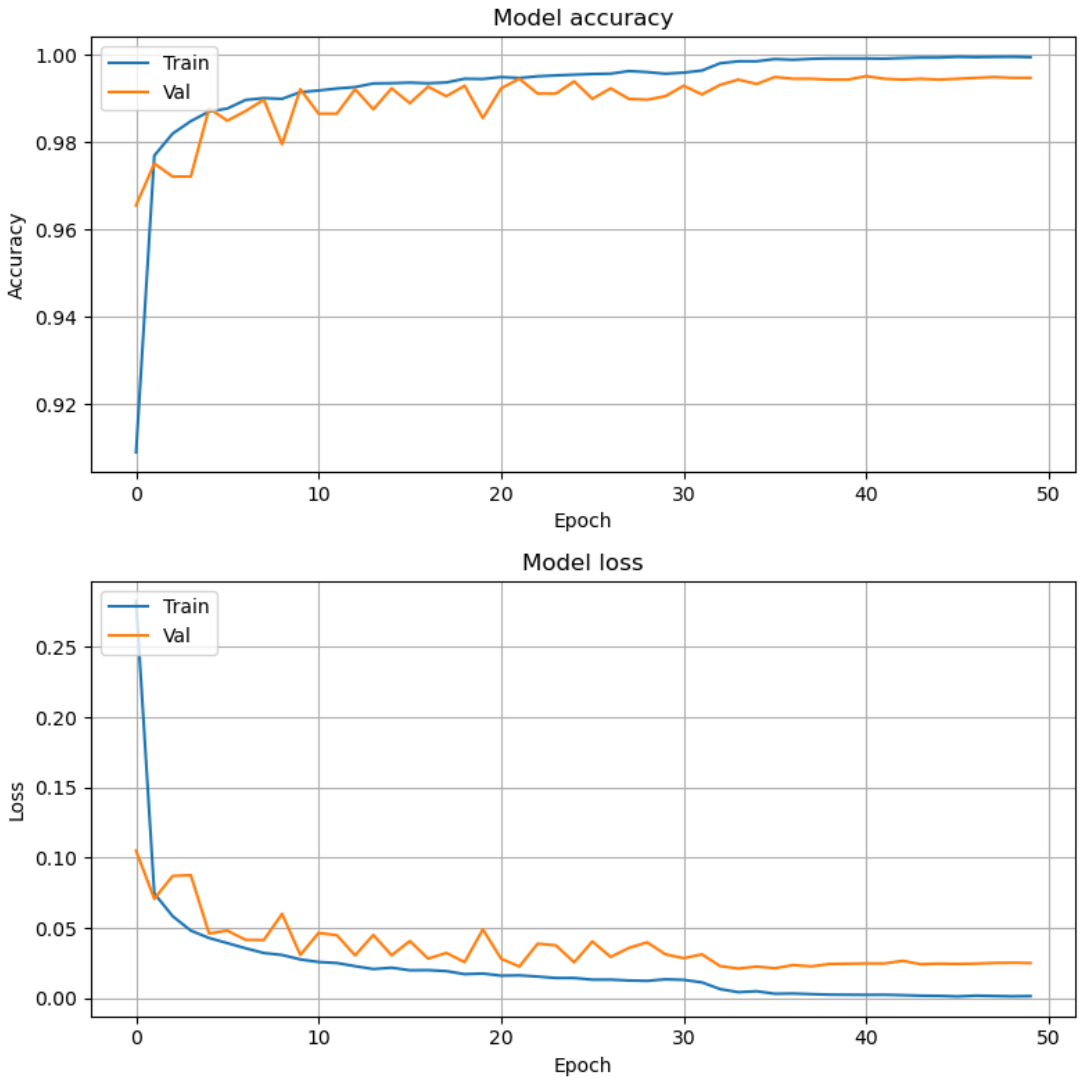


figure1. Learning curve

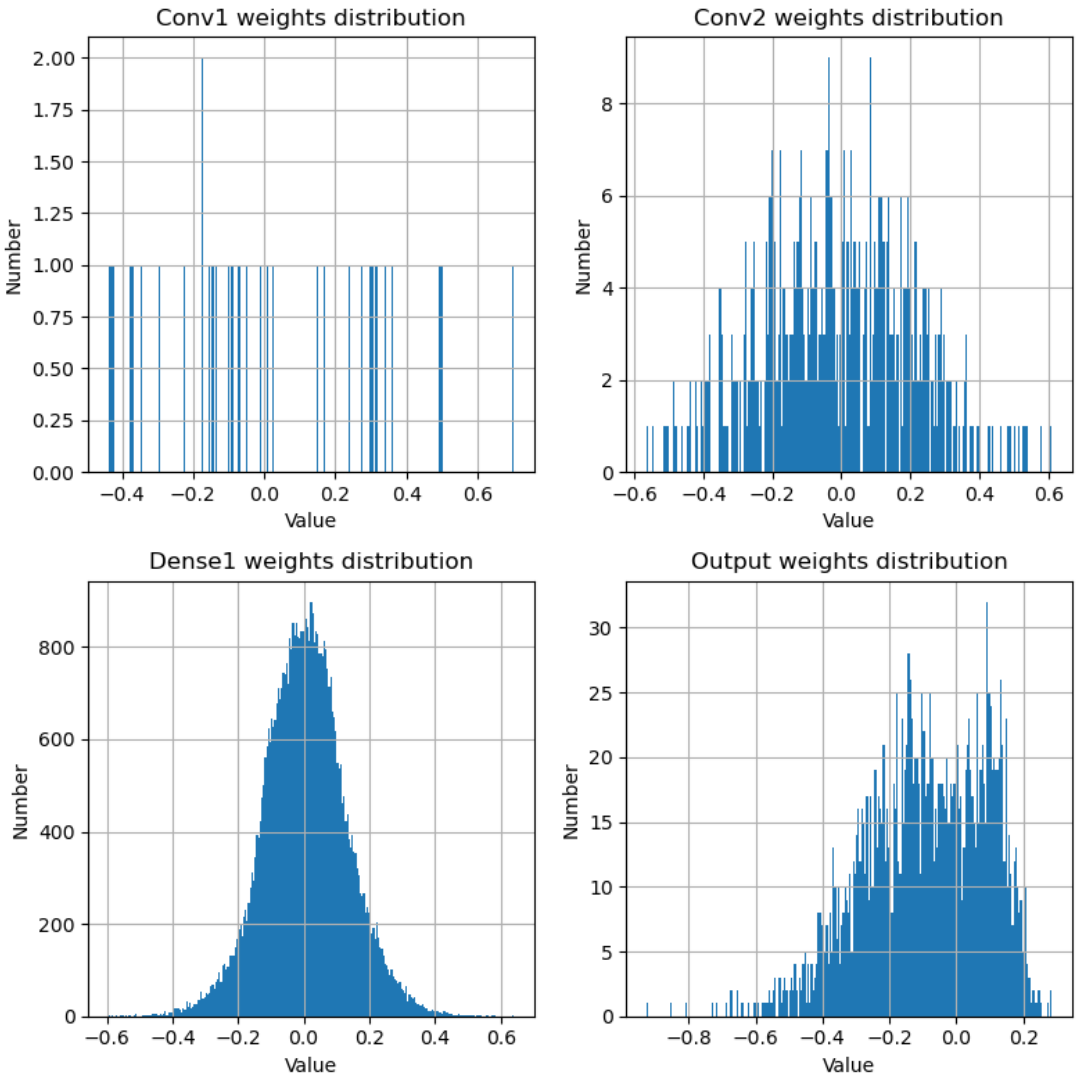


figure2. Histograms of layers

Stride size is how far the filter block move every time. It can control the output image size. With a larger stride, the output image size will be smaller. It can be used to control the size of data and accelerate the computation time.

However, when the original image size is not that large, large stride size may lose too many information from original image and reduce the accuracy.

Kernel size (filter size) is the size of filter that padding through the image. It is usually set to 5x5 for first layer and 3x3 for the rest. Very small filter sizes will capture very fine details of the image. On the other hand, having a bigger filter size will leave out minute details in the image. So it is popular to use 3x3 filter size.

My experiment: (Stride = 1) (Epochs = 50)

For first two layer kernel size = 5x5 and the rest = 3x3:

Train acc: 0.9999

Test acc : 0.9950

Wrong case: 50

Kernel size all = 3x3:

Train acc: 0.9998

Test acc : 0.9961

Wrong case: 39

In this case, we can see 3x3 filter size is slightly better than 5x5 filter size. So I use 3x3 filter to test different stride. I change stride size on only first CNN layer.

For stride = 1, filter size = 3x3:

Train acc: 0.9998

Test acc : 0.9961

Wrong case: 39

For stride = 2, filter size = 3x3:

Train acc: 0.9989

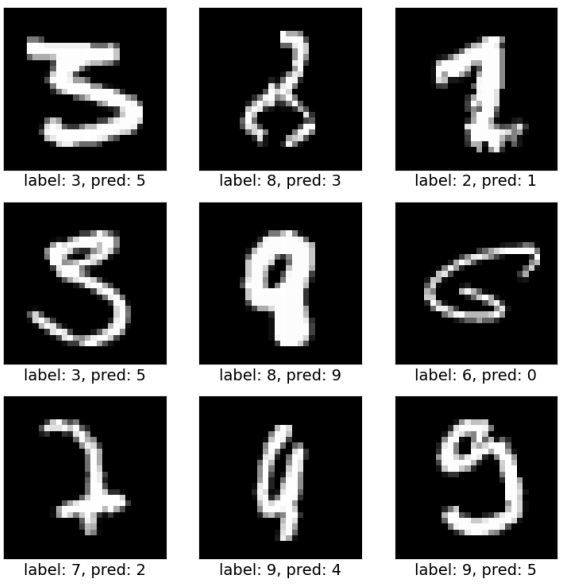
Test acc : 0.9932

Wrong case: 67

In this case, we can see stride size 1 is better than 2. When filter jump too far, I think it will lose some information and association between pixels.

1-2 Correct and incorrect results

In MNIST, correct prediction don't have problem. However, when I observe wrong case one by one, I can see some case that I think the label may be wrong. For some case the image can not even be determined my me what number it is. For these cases, I think it is reasonable to predict wrong. While I can see some cases are predicted wrong because of the corner of number. I think that is because the model is not good enough to learn the detail.



1-3 How feature map changes with increasing depth

CNN1 output

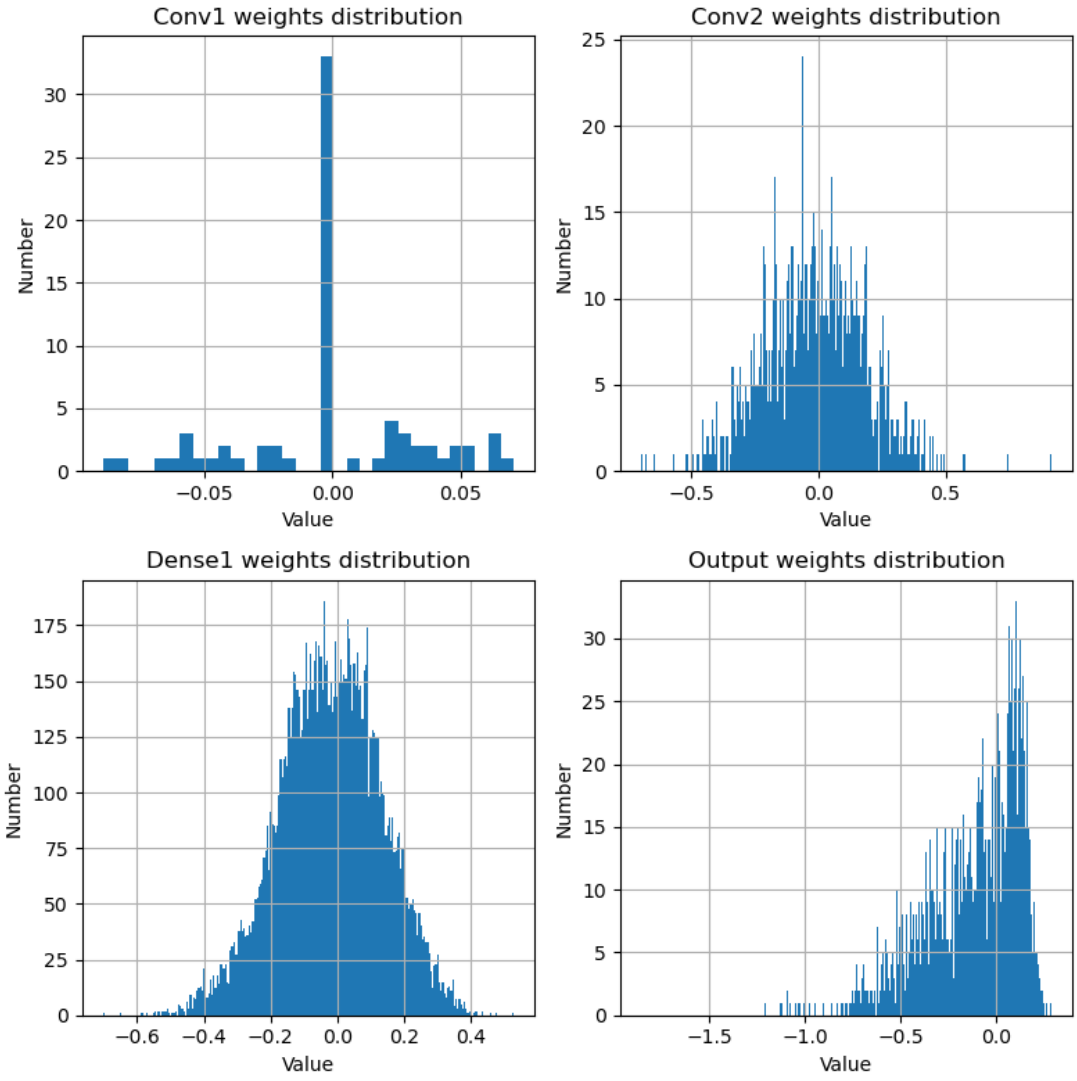
CNN2 output

To observe the feature maps changes, I visualizes it and I can see the main difference is that after filtered many times, the feature maps are more and more blurred. The image is being more abstract when it comes to deeper network. And this can be considered as the feature is being extracted and the redundant information is being abandoned.

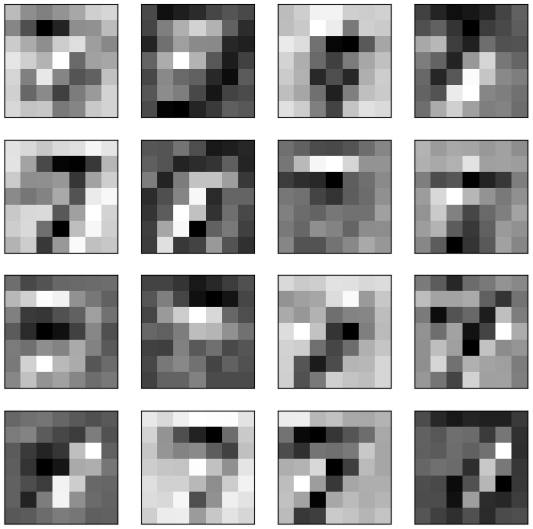
1-4 Add L2 regularization to CNN

I changed the first two CNN layers to use L2 regularization to test for 50 epochs, same as before. It seems that it is less able to learn well when adding L2 regularization to CNN layers. I wonder if it is because it can reduce the overfitting, so I train for 100 epochs next time. But I didn't get any better results. The train accuracy can still only reach 0.9989, and test accuracy

0.9929.



I think the reason may be this. When adding L2 regularization to weights, it will limit the weights from changing too much, which cause most of the weights to be small and close to 0. Also, we can see the filter is more blurred than not using L2 regularization.



CIFAR10

2-1 The effect of stride size and filter size

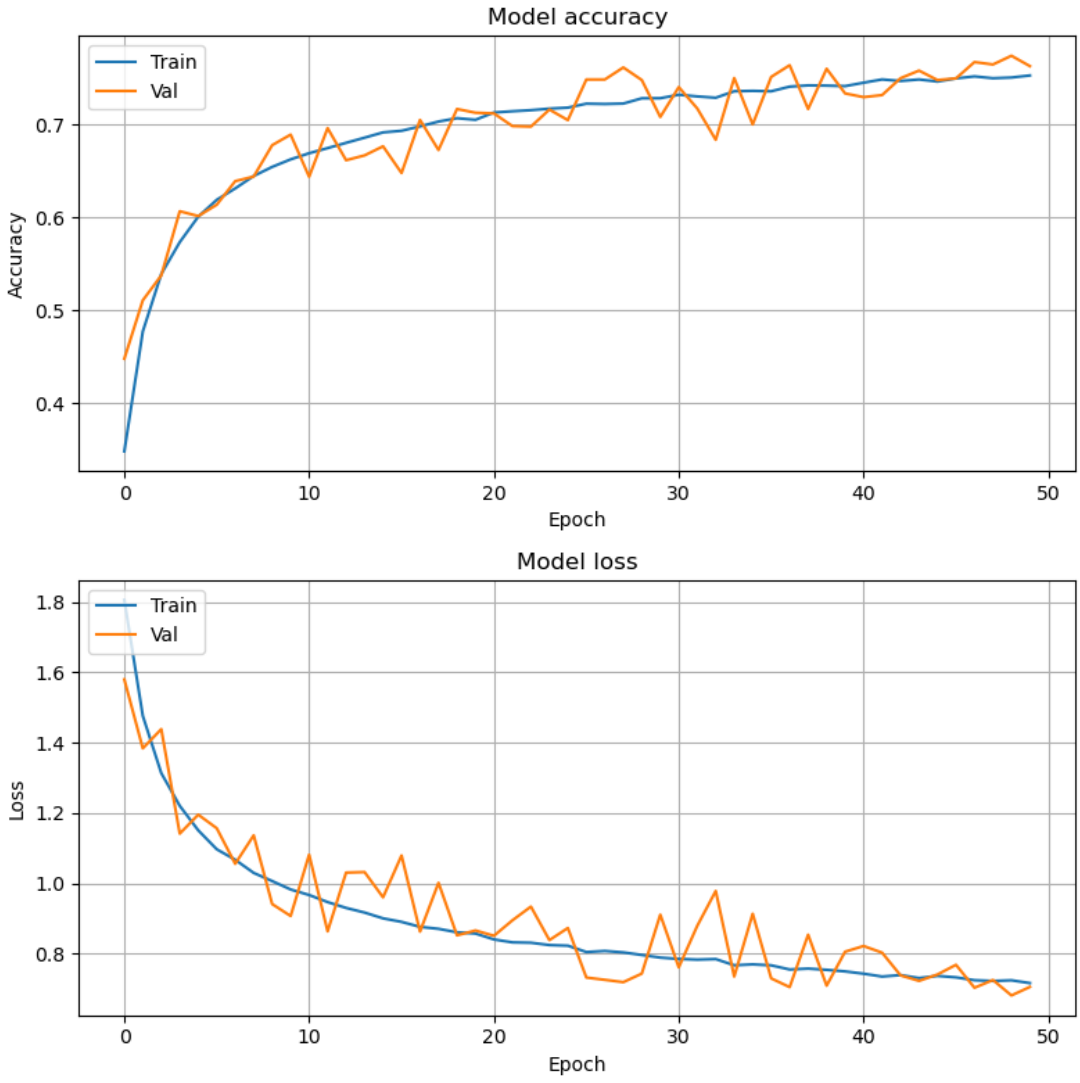


figure1. Learning curve

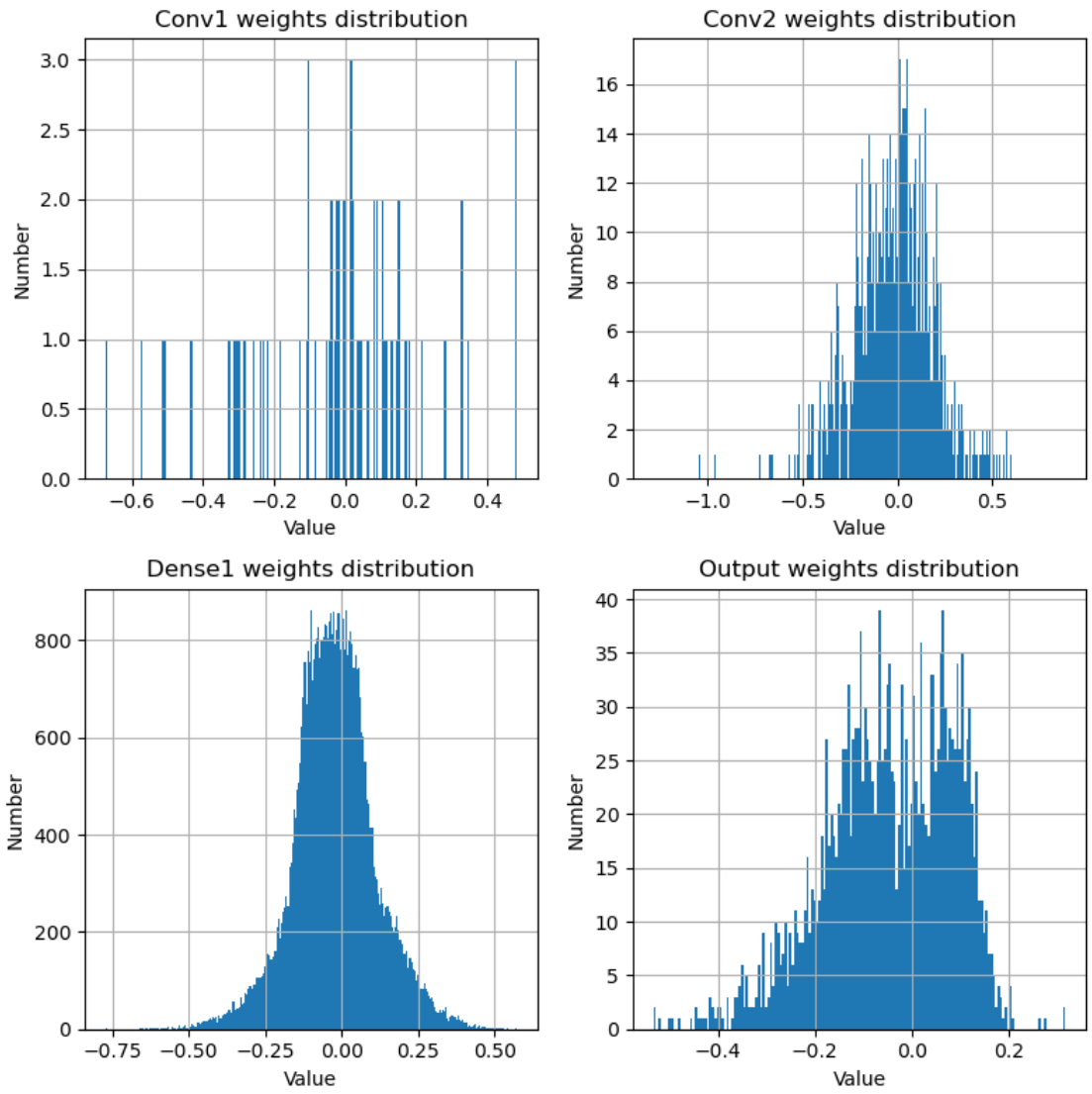


figure2. Histograms of layers

My experiment: (Stride = 1) (Epochs = 50)

For first two layer kernel size = 5x5 and the rest = 3x3:

Train acc: 0.7671

Test acc : 0.7611

Kernel size all = 3x3:

Train acc: 0.7532

Test acc : 0.7538

In this case, we can see 5x5 filter size is slightly better than 3x3 filter size. But the 5x5 one has reduce it's learning rate by my learning rate planner. It will reduce if the val_loss didn't reduce for 10 epochs. This means at the same learning rate 5x5 will stop reducing its val_loss earlier. If I give them more time to train, I think 3x3 would still be better.

For stride = 1, filter size = 3x3:

Train acc: 0.7532

Test acc : 0.7538

For stride = 2, filter size = 3x3:

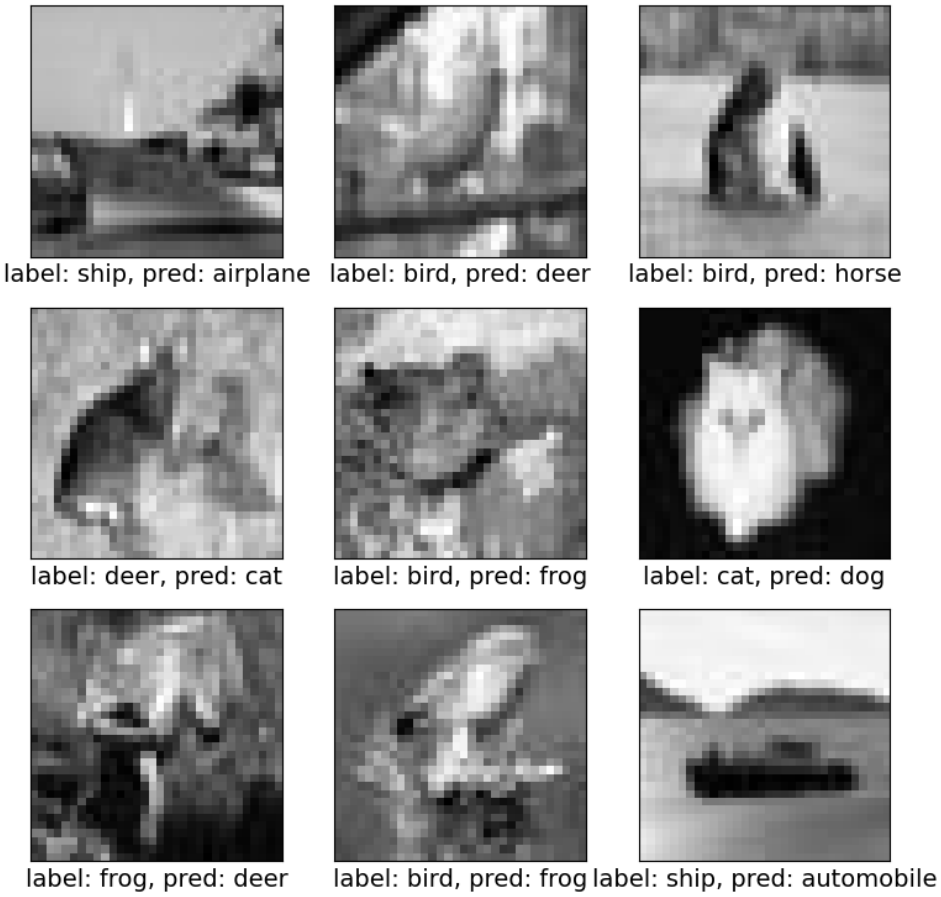
Train acc: 0.6837

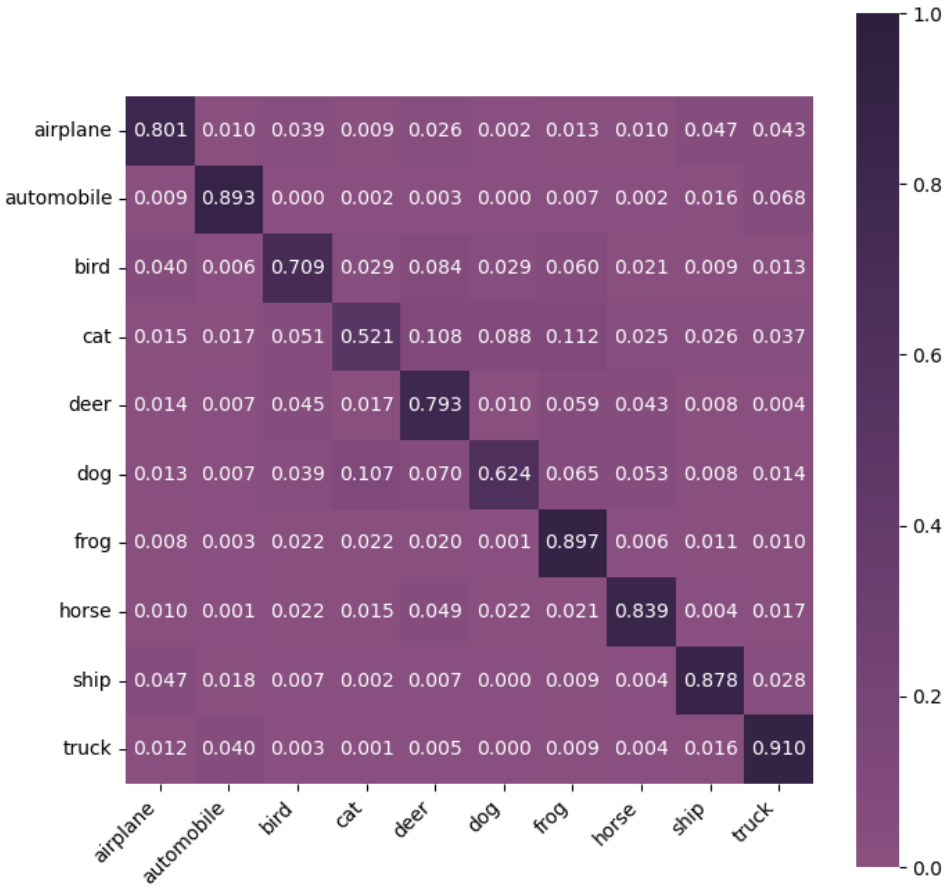
Test acc : 0.6954

For Stride = 2, it is obviously that the performance is worse than stride = 1. I think it is always better to use stride = 1 unless the data is too large to train.

2-2 Correct and incorrect results

Wrong cases

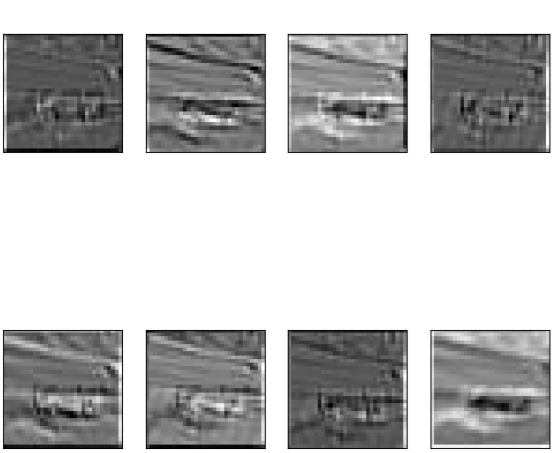


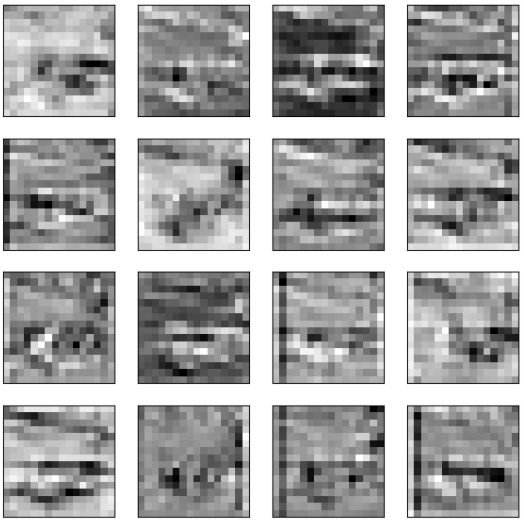


In CIFAR10, it is more difficult to recognize the label by eyes because of the resolution. From the result, I can see that cat and dog are the most difficult object to train. I think that is because when the resolution is small, it is hard to see the small detail of cat and dog. But cat and dog, in often, is recognized by its detail.

2-3 How feature map changes with increasing depth

CNN1 output CNN2 output





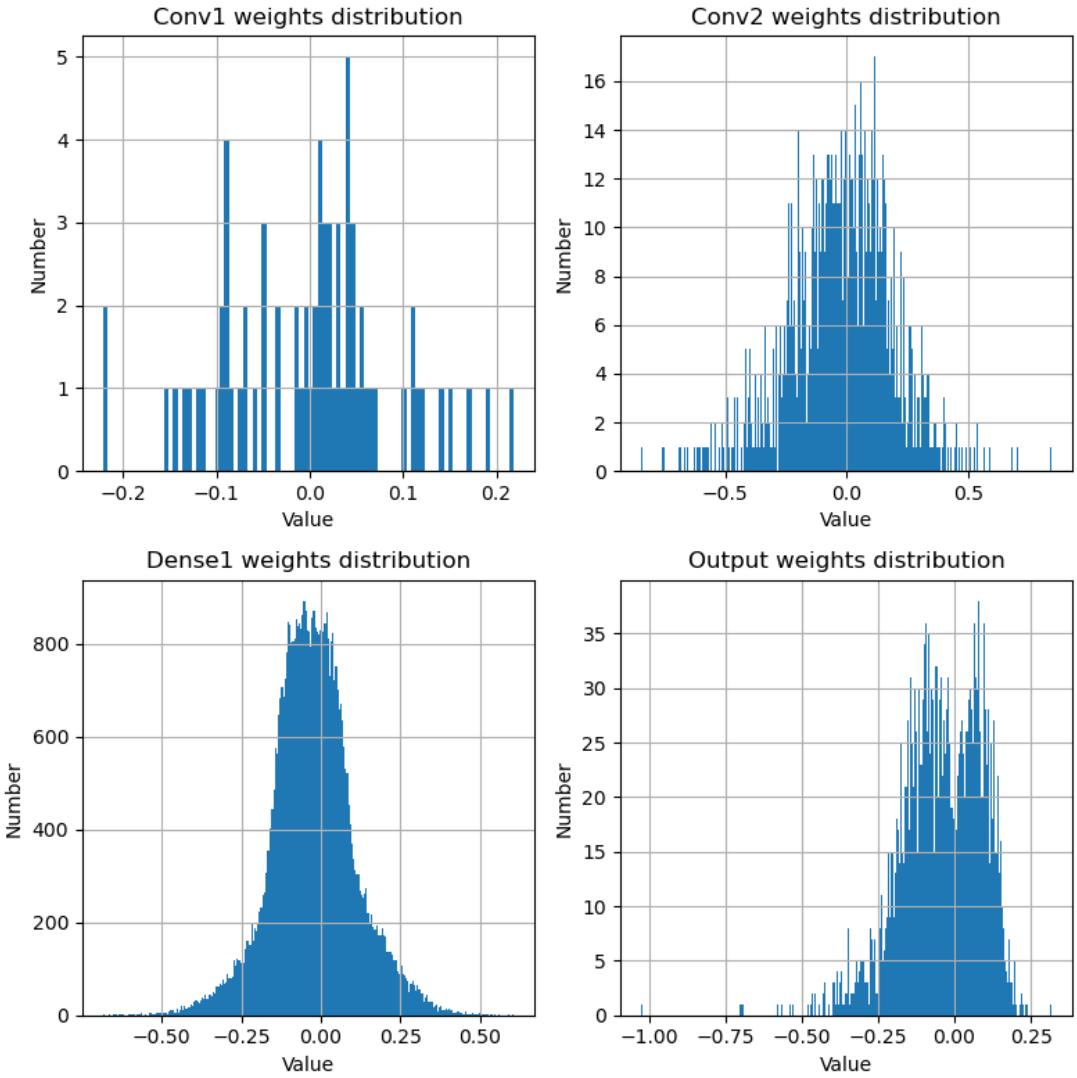
This part is same as the MNIST, after many filters and maxpooling, the size of the image will getting smaller and smaller. And the features will be extract.

2-4 Add L2 regularization to CNN

Same as MNIST. I changed the first two CNN layers to use L2 regularization to test for 50 epochs. This time there is something different. The train accuracy is less than no L2 regularization result, but the test accuracy is higher.

Without L2:	With L2:
train acc: 0.7532	train acc: 0.7500
test acc : 0.7538	test acc : 0.7620

We can see that with L2 regularization, even thought the train acc is lower, it seems to have better generalization ability. This is thanks to its regularization of weights, which make it not to be so easy to overfit.



In this picture we can see that the value of Conv1 is lower than not using L2 regularization.

2-5 Preprocessing

Step1. I download the dataset from CIFAR10 website and transform the packages to jpg images. And I put the same class object to each folder.

Step2. I use ImageDataGenerator in Keras to augmented the dataset with zoom, rotation, brightness,…etc.. The detail is in the following picture.

Step3. Train with Keras.

```
    # Data input and data augmentation
    train_datagen = ImageDataGenerator(rescale = 1./255, shear_range=0.1,
                                       zoom_range = 0.1, horizontal_flip = True,
                                       rotation_range=20, brightness_range=(0.8, 1.2),
                                       width_shift_range=0.1, height_shift_range=0.1,
                                       zca_whitening=False, fill_mode='nearest')
    val_datagen = ImageDataGenerator(rescale = 1./255)
    test_datagen = ImageDataGenerator(rescale = 1./255)

    training_set = train_datagen.flow_from_directory(
        'CIFAR10/train',
        target_size = (32, 32),
        batch_size = batch_size,
        color_mode="grayscale",
        class_mode = 'categorical')
    validation_set = val_datagen.flow_from_directory(
        'CIFAR10/val',
        target_size = (32, 32),
        batch_size = batch_size,
        color_mode="grayscale",
        class_mode = 'categorical')
    test_set = test_datagen.flow_from_directory(
        'CIFAR10/test',
        target_size = (32, 32),
        batch_size = batch_size,
        color_mode="grayscale",
        class_mode = 'categorical')
```