Introduction to Artificial Intelligence

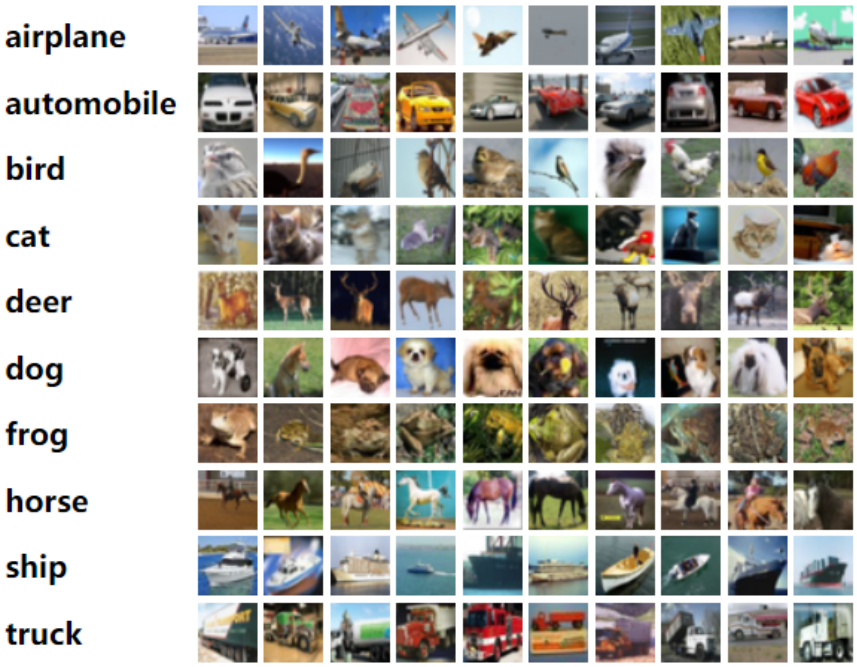
Homework Report #2

Junghun Lee
Economics / BigData Science / Japanese Culture
November 18, 2022

Convolutional Neural Network

Convolutional neural networks are a specialized type of artificial neural networks that use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. They are specifically designed to process pixel data and are used in image recognition and processing.

This homework was about the classification of **CIFAR10**. Unlike the **MNIST** question learned in class, RGB color information existed as three layers. At first, I did have no idea what to do. But after I slowly reviewed what I learned in class and tried it again, I was able to do my assignment. The following image is sample of **CIFAR10**.



CIFAR 10 Data Set

CIFAR10 consists of 10 classes, each containing about 10000 images. pytorch uses 50000 images as learning data and 10000 images as test data. This dataset is so popular that anyone who does deep learning have made model to it at least once using this. As such, there are endless learning methods. I'll introduce how I trained this data and tested model in next chapter.

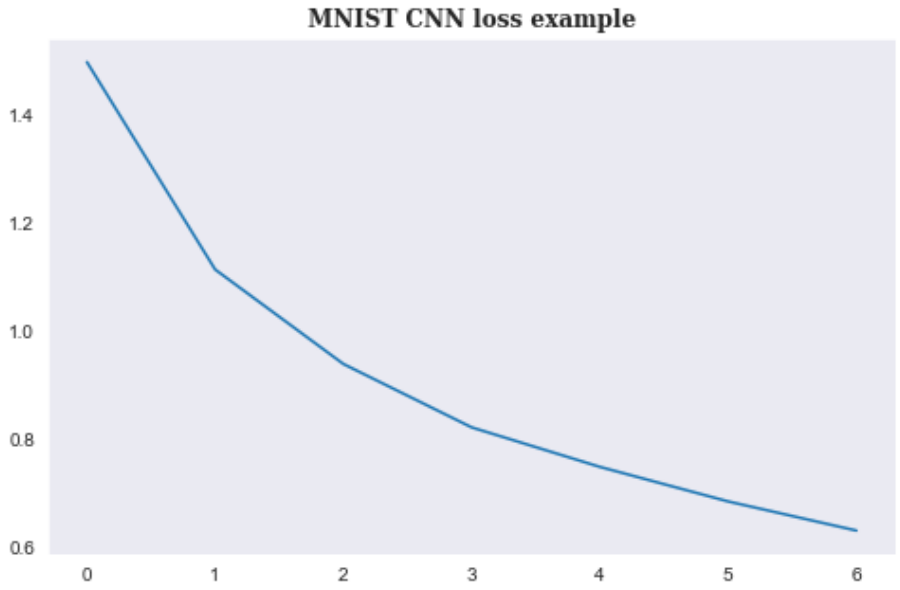
Model Train

Before designing a model, I checked the given conditions. The conditions given in the homework were that we had to use three *ConvNet* layers, two active layers, and one output layer. One more, remember to vectorize from the calculated values to 1D tensor after the *ConvNet* layer. I coded *ConvNet* Class based on what I learned in class. What changed was that the input channel had three initial inputs for RGB representation, and that the input and output of the *ConvNet* layer had to be adjusted accordingly. After finishing the work, I was able to move on to the next step when I saw that it was worked without an error.

I created the ConvNet Class as follows. In the first layer, the input channel value is 3 and the kernel size is 5 * 5. In order to maintain the same size of the input channel and the output channel, the padding parameter is designated as 2. After batch normalization, Max pooling reduces the size of the image in half. As a result, the output becomes 12 * 16 * 16 through the first ConvNet layer. With the same logic, the size of the feature map is 24 * 8 * 8 after passing the second ConvNet layer and finally 48 * 4 * 4 after passing through the last ConvNet layer. After vectorization, all information is sorted into a one-dimensional tensor, making it a tensor consisting of a total of 48*4*4 elements. It is designed to be output as a tester composed of 128 elements once the activation layer is passed, next is 32 elements after passing second activation layer, and finally as a parameter of 10 classes after the last node.

I instanced trainloader and testloader and tested the model. For the first attempt, I try to set the learning rate as 0.0001 and epoch as 5. Batch size is specified as 20. In this cases, the loss value did not decrease because the variable name of the learning model was not changed by mistake. In the **second** attempt, **the model's accuracy was 63%**, which was close to the homework threshold of **65%.** kaboom!

I realized that the learning rate was too small in this attempt, so I raised it to 0.005. In addition, I designated Batch size as a multiplier of 2 as 64, and raised epoch to 7 for more learning. As a result of modifying the hyper parameter value, the loss curve was finally obtained as shown below.



Loss Curve

```
        img = img.to(device)
        lab = lab.to(device)
        out = test_model(img)
        _, pred = torch.max(out.data, 1)
        correct += (pred == lab).sum().item()

    print("Accuracy of the network on the {} test images: {}%".format(len(test_loader) * 
                                                                      100 * correct /
                                                                      (len(test_loader) *
                                                                       batch_size)))
```

```
Accuracy of the network on the 10048 test images: 70.89968152866243%
```

The final accuracy of my model is about 70.9%