1. API (img2num)

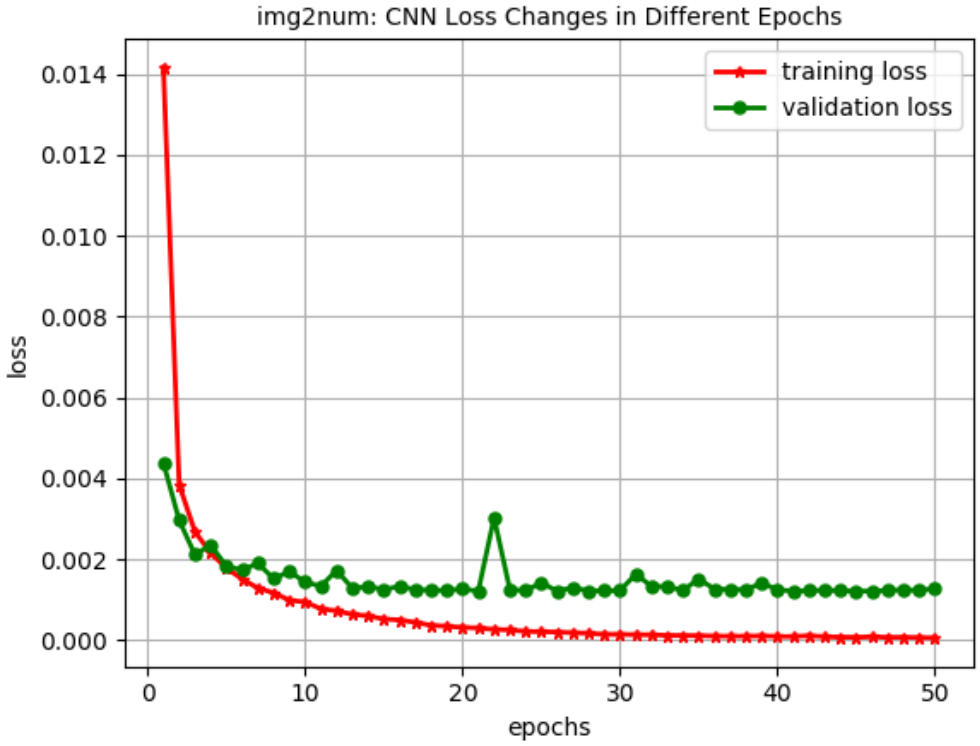
It's in file *img2num.py*. It has 2 public methods, *train()* and *forward([28x28 ByteTensor] img)*.

__init()__. It sets the basic structure and operations of LeNet-5. It also sets the learning rate, loss function, and optimization method. These are the same with those in *NnImg2Num* API in homework 4.

train(). It performs training and validation of the network. The process is, for each epoch, to perform *forward -> backward -> update* for each training batch of all training batches, and then perform validation for each validation batch. The whole process is repeated for *max_epoch* times. The whole process uses onehot labeling method to label the image and train the network.

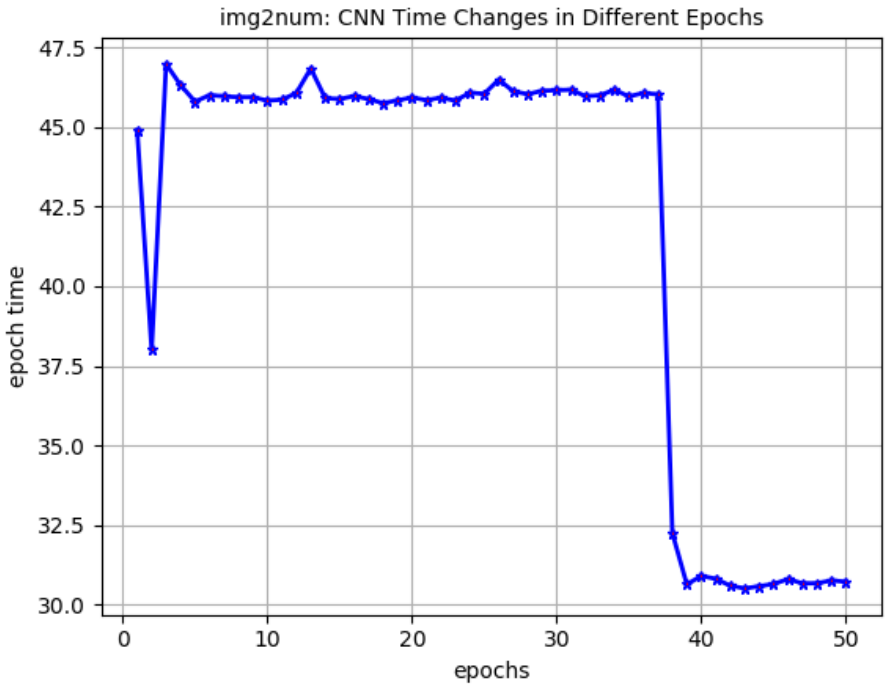
forward(). It reads in one image of 28x28, converts it to 1D and output a predicted label for it.

The training/validation loss vs epochs chart is



Because the way we calculate the training loss (average each batch loss after each forward-backward-update), the training loss is a little bit higher than validation loss, but eventually, the training loss is less than the validation loss. And at the end of 50^{th} epoch we have a validation accuracy of *99.1%.*

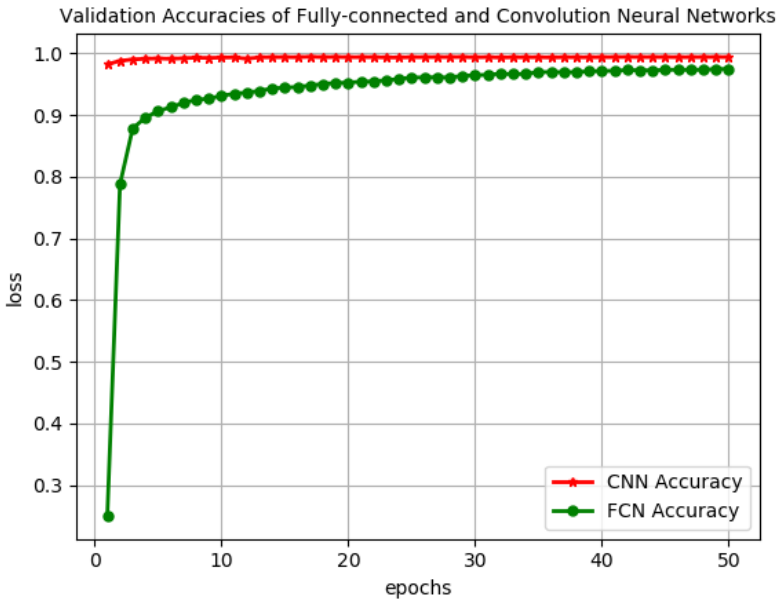
The time vs epochs chart is



From the picture, we can see that the time taken for each epoch rises after the first 2 epochs, then drops down. The total time taken is *1564.88s*.

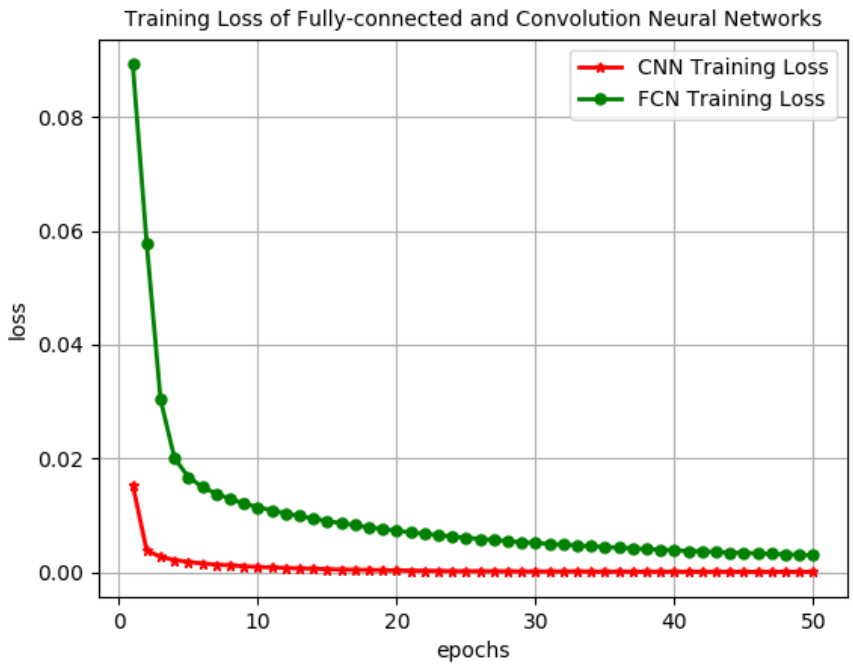
1.1 Comparison of Convolutional Neural Network and Fully-Connected Neural Network

The accuracy comparison of the 2 neural networks is shown in the figure



From the figure we can see that CNN can achieve higher validation accuracy than FCN, using less time.

The training loss comparison of the two neural network is shown in the figure below



From the figure we can see that CNN's training loss is lower than FCN, and converges much faster.

2. API(img2obj)

It's in *img2obj.py*. It has 3 public methods, *forward(), train(), view()* and *cam().*

__init__(): it sets the basic structure and operations to get LeNet-5. The cross-entropy loss function is applied. The optimizer method is Adam, and the learning rate is 0.0001. It also downloads the CIFAR-100 dataset to training and validation folders.

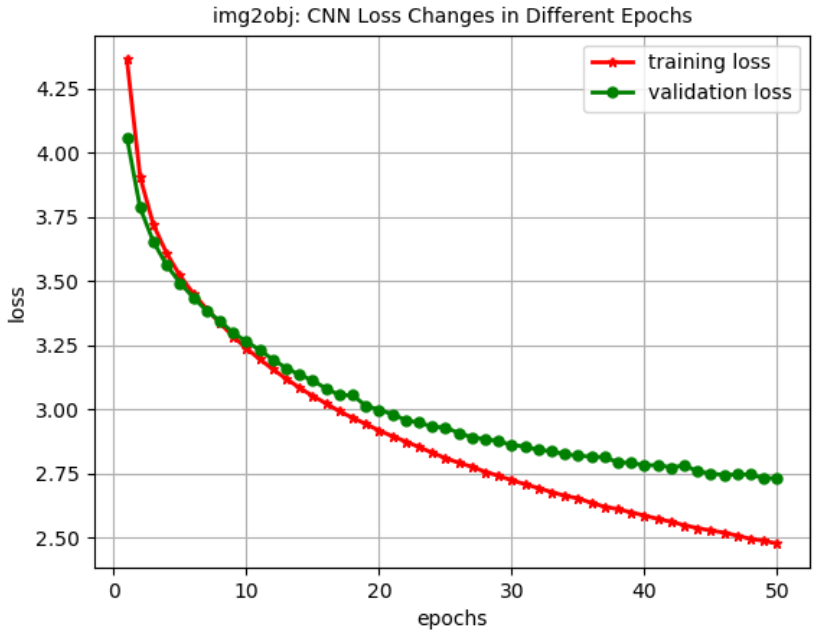
forward(): It uses an image as input and output the prediction result.

train(): It trains the neural network, and outputs the loss map.

view(): it views an image and the caption in the picture

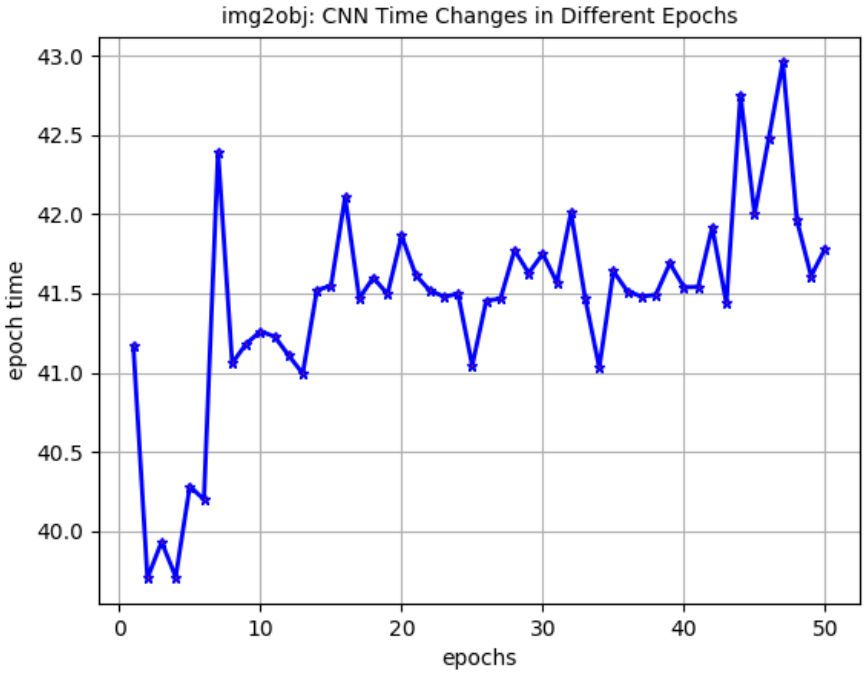
cam(): it fetchs images from the camera, and continuously outputs the result.

The training/validation loss vs epochs chart is



Because the way we calculate the training loss (average each batch loss after each forward-backward-update), the training loss is a little bit higher than validation loss, but eventually, the training loss is less than the validation loss. And at the end of *50th* epoch we have a validation accuracy of around *32%.*

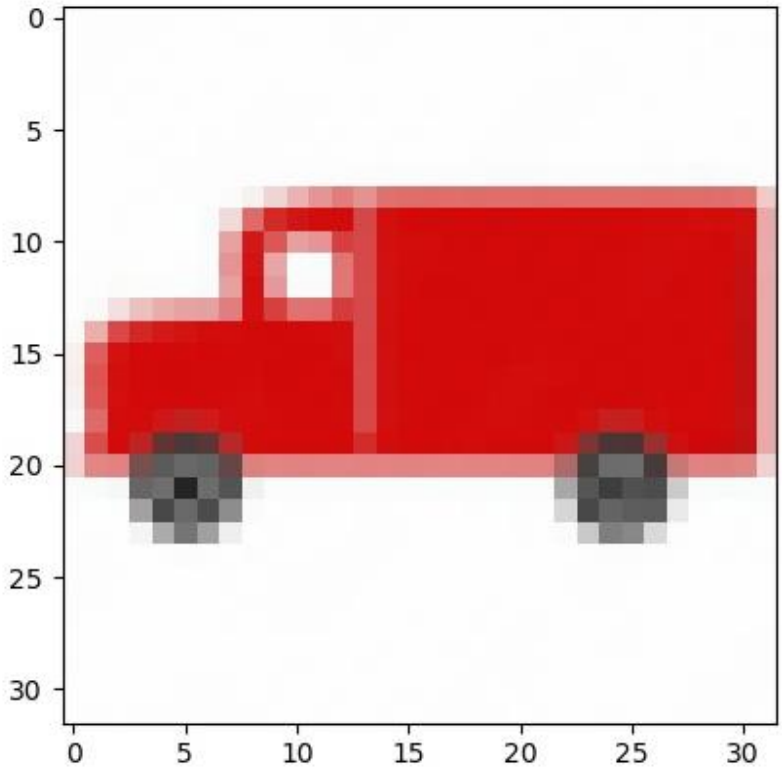
The time vs epochs chart is



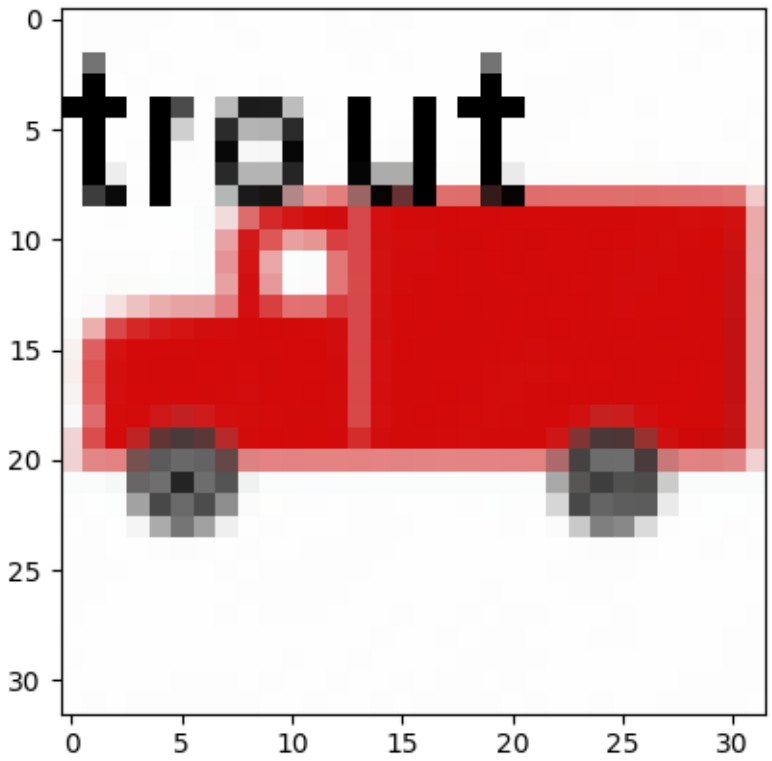
2.1 view result

The original input image is

Scale it to 32x32



The view result is



We can see in the picture that it was recognized as a trout by the neural network.

2.2 cam result

3 Other files

3.1 compare_img2num.py

It has the code to draw images to compare CNN and FCN on MNIST dataset.

3.2 test.py

It has the test code to run the API's mentioned above.