Homework 1: Convolutional Neural Network

***GPUs may be needed for speeding up the neural network training process in this homework.**

Description

In this homework you will practice how to write a Convolutional Neural Network (CNN) classifier in Python with the Pytorch framework. You need to understand how a CNN works, including backpropagation and gradient descent in order to implement this homework successfully. The goal of this homework is:

- To implement and understand the CNN architecture.

Instruction

- The dataset used in this homework is **MNIST** and **CIFAR-10**. You may need these packages: Pytorch, NumPy, and OpenCV (for reading images). The commonly used classifiers are Softmax and SVM.
- Requirements:
  1. Contain a **training function** that will be called to train a model, also contain an **argument** that could determine which dataset to be used. For example,
     "**python CNNclassify.py train --mnist**", and
     "**python CNNclassify.py train --cifar**"
  2. Save the models in a folder named "**model**" after finishing the training process.
  3. Show the **testing accuracy** in each iteration of the training function. For CIFAR-10 dataset, the test accuracy should be greater than or equal to **75%** in the end.
     1) ***You can add as many layers as you want for both CONV layers and FC layers.*** Optimization techniques such as mini-batch, batch normalization, dropout and regularization might be used.
     2) In the **first CONV** layer, the filter size should be **5*5**, the stride should be **1**, and the total number of filters should be **32**. All **other** filter sizes, strides and filter numbers are **not** acceptable and may result in **a final grade of 0** in this HW.
     3) For other CONV layers (if any), there is no limitation for the size of filters and strides.

Loop	Train Loss	Train Acc %	Test Loss	Test Acc %
1/500	2.3616	24.3023	1.7271	40.7832
50/500	0.9411	66.9585	0.9494	66.8908
100/500	0.6667	76.8950	0.8285	71.5091
150/500	0.4969	82.9228	0.7840	73.5166
200/500	0.3714	87.6287	0.7697	74.3572
250/500	0.2821	90.9035	0.7680	75.0890
300/500	0.2149	93.1630	0.7810	75.7911
350/500	0.1705	94.7071	0.7962	75.8604
400/500	0.1296	96.1417	0.8210	76.0186
450/500	0.1046	96.9289	0.8349	76.4438
500/500	0.0862	97.5184	0.8499	76.6317

Model saved in file: ./model/model.ckpt

Fig. 1 The screenshot of the training result for CIFAR-10.

4) You can choose as many CONV layers as you want, however, please be aware that the computational cost of CONV layer is very high, and the training process may take quite long.

5) You can also choose as many **FC** layers as you want in order to enhance the model accuracy. There is no limitation for the size of FC layers.

4. Implement a **testing function** that accepts the command "**python CNNclassify.py test xxx.png**" to test your model by loading it from the folder "model" created in the training step. The function should (1) read "xxx.png", load model checkpoint trained on the corresponding dataset and predict the output as shown in Fig. 2 (take CIFAR-10 for an example), and (2) visualize the **output of the first CONV** layer in the trained model for each filter (i.e., **32** visualization results), and save the visualization results as "**CONV_rslt_cifar.png**" and "**CONV_rslt_mnist.png**" as the example shown in Fig. 3. The testing result would match the true image type when the classifier achieves high accuracy.

```
prediction result: car
```

Fig. 2 The screenshot of the testing result for CIFAR-10.

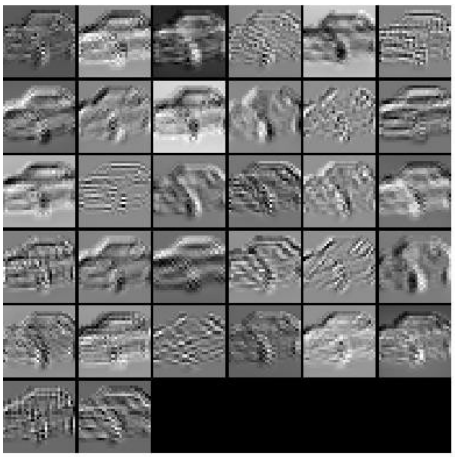

Fig. 3 The screenshot of the first CONV layer visualization for CIFAR-10 (car shape).

Submission

- You need to submit a **zip** file including:
  1. a python file named "**CNNclassify.py**";
  2. a generated model folder named "**model**";
  3. 4 screenshots of training and testing results (2 for CIFAR-10 and 2 for MNIST);
  4. 2 screenshot of the **visualization results** from the **first CONV** layer(1 for CIFAR-10 and 1 for MNIST).
- The "**CNNclassify.py**" file should be able to run with the following commands:
  1. **python CNNclassify.py train –mnist** and **python CNNclassify.py train --cifar** to train your neural network classifier and generate a model in the model folder on MNIST and CIFAR-10 dataset.
  2. **python CNNclassify.py test xxx.png** to (1) predict the class of 2 images from two datasets and display the prediction result; (2) save the visualization results from the first CONV layer of two datasets as "**CONV_rslt_mnist.png**" and "**CONV_rslt_cifar.png**".
- The **zip** file should be named using the following convention:
  <Last-Name>_<First-Name>_HW1.zip

Ex: Potter_Harry_HW1.zip

- Note:

  Do not put any print function other than showing the results.
  Comment your code.

Grading criteria

- Your model will be tested by running "**python CNNclassify.py test xxx.png**" with additional testing images to verify (1) the **test function** and (2) the **visualization function**. Please make sure your functions work correctly.
- The testing accuracy on CIFAR-10 should be greater than or equal to **75%** in the end. There will be 1- point deduction for every 1% of accuracy degradation based on 75%.
- Upload the zip file to Canvas before 11:59PM (EST Time) 10/20/2023.