CSCI 8000 HW 1

Fall 2024 Special Topics in Efficient Deep Learning Due 10/03/2024

Homework 1: Convolutional Neural Network

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

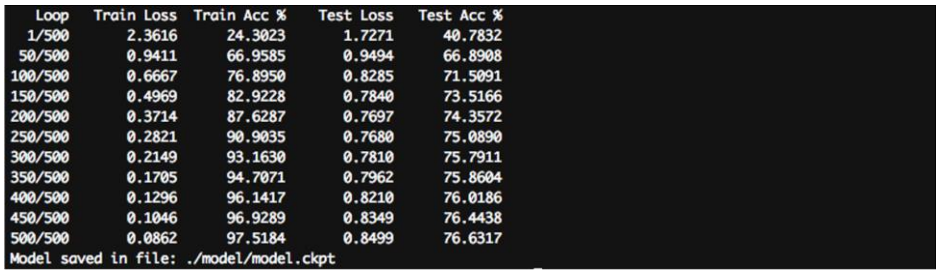
Description

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 **CIFAR-10**. You may need these packages: PyTorch, NumPy, THOP (for MACs and parameters counting), OpenCV (for reading images), Matplotlib. The commonly used classifier is Softmax.
- Requirements:
  1. Contain a **training function** that will be called to train a model with the command "**python CNNclassify.py train**".
  2. Save the model in a folder named "**model**" after finishing the training process.
  3. Show the **testing accuracy** in each epoch of the training function. The test accuracy should be greater than or equal to <u>**75%**</u> in the end using the CIFAR-10 dataset.
     1) <u>***You can add as many layers as you want for both CONV layers and FC layers***</u>**.** 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**, no padding, and the total number of filters should be **32**. All **other** filter sizes, strides and filter numbers are **not** acceptable and may result in <u>**a final grade of 0**</u> 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.

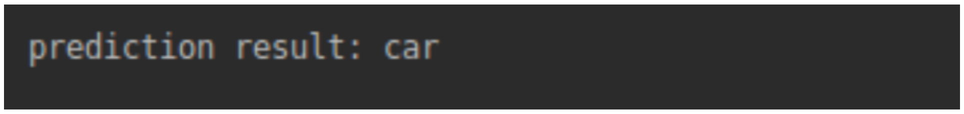
     4) You can choose as many CONV layers as you want, however, please be aware that the computational cost of CONV layer is very high, and the training process may take quite long.

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

6) You need to manually set the random seed for your model training process. You need to train your network 3 times using different random seeds. In your pdf report, you need to show the 3 random seeds used for the training and the corresponding accuracies of the trained model. You also need to report the mean and standard deviation of the 3 accuracies. You can use numpy.std() to calculate the standard deviation.

7) During the training process, you need to store the training accuracy and testing accuracy at the end of each training epoch (store in a list or something else). And at the end of the entire training, you need to draw a **single** plot that contains **two curves** (training accuracy and testing accuracy) that show the trend of accuracy during training. For the plot, the X-axis should be the number of training epochs, the Y-axis should be the accuracy. Two curves should have different colors and a legend should be shown in the plot. You can use the Matplotlib package to draw the plot. You only need to draw one plot to show one training process, no need to do this for different seeds.

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" and predict the output as shown in Fig. 2, 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.png**" as shown in Fig. 3. The testing result would match the true image class when the classifier achieves high accuracy. *You need to provide your own testing image and submit the image together with your code.



```
prediction result: car
```

Fig. 2 The screenshot of the testing result.

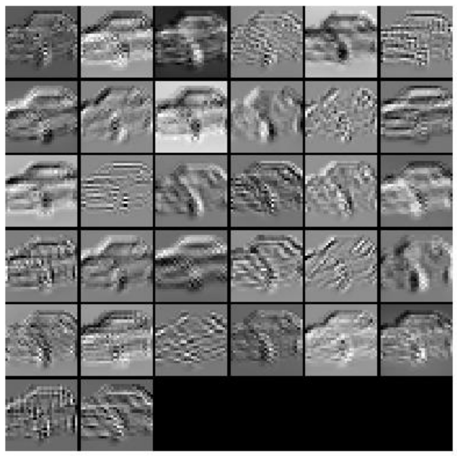


Fig. 3 The screenshot of the first CONV layer visualization (car shape).

5. Implement a testing function that accepts the command "**python CNNclassify.py resnet20**". The function should load a pretrained ResNet-20 model and test the accuracy of this model using the entire CIFAR-10 testing dataset.

6. Use THOP package/library to count the computation cost (MACs) and number of parameters of **your neural network** and **ResNet-20** model, respectively. The THOP package and its example can be found in https://pypi.org/project/thop/

7. You need to write a separate function, i.e., inference_speed_test(args), which can be used to test the model inference speed of your own model and ResNet-20 model. There are several points to keep in mind:
   - The inference time should be measured based on a single input, not a batch. You can use either dummy input or real image as the input.
   - You should run several warmup iterations before measuring the time.
   - You should run multiple inference iterations (e.g., 100 or 1000) and measure the overall running time and calculate the average time as your final inference time.

Submission

- You need to submit a **zip** file including:
  1. a python file named "**CNNclassify.py**";
  2. a generated model folder named "**model**";
  3. a report (.pdf) that includes:
     a. your code
     b. two screenshots of training and testing results of your neural network design.
     c. one screenshot of the **visualization results** from the **first CONV** layer
     d. a screenshot to show the testing accuracy of ResNet-20 model on entire CIFAR-10 testing dataset.
     e. a screenshot of the plot shows the curves of the training accuracy and testing accuracy.
     f. a screenshot of the computation costs (MACs) and the number of parameters of your Neural Network and ResNet-20 counted by using **THOP** package.
     g. Show the 3 random seeds you used for training and the corresponding final accuracies obtained. Show the mean and std deviation of the 3 accuracies.
     h. Show the inference time of your own model and ResNet-20 model, respectively.

- The "**CNNclassify.py**" file should be able to run with the following commands:
  1. **python CNNclassify.py train**
     to train your neural network classifier and generate a model in the model folder;
  2. **python CNNclassify.py test xxx.png**
     to (1) predict the class of an image and display the prediction result; (2) save the visualization results from the first CONV layer as "**CONV_rslt.png**".
  3. **python CNNclassify.py resnet20**
     to test the accuracy of a pretrained ResNet-20 model on CIFAR-10 testing dataset.

- The **zip** file should be named using the following convention:
  <Last-Name>_<First-Name>_HW2.zip
  Ex: Potter_Harry_HW2.zip

- Note:
  Do not put any print function other than showing the results.
  Comment your code.
  Do not include CIFAR dataset in your submission!

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 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 the eLC before 11:59PM (EST Time) 10/03/2024.
- Late fee: -10 points/day