Assignment: Homework 6

How to Hand It In

1. Put all your solutions in one folder. Compress this folder and name it <firstname>_<lastname>_HW6.zip. For example, “Zhiyao_Duan_HW6.zip”.
2. Make sure you have a report to show graphs and explain your answers, in addition to source code.
3. Submit to the corresponding entry on Blackboard.

When to Hand It In

It is due at 11:59 PM on the date specified on the course calendar. **Late assignments will receive a 20% deduction of the full grade each day.**

Problems (10 points in total)

1. (4 points) Comparing MLP with 2D CNN using CIFAR-10.
    a. (1 point) Follow the CIFAR-10 tutorial and run it: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html. Modify some of the code to run the training process on GPU. Compare the training time for 10 epochs with a large batch size between CPU and GPU.
    b. (0.5 points) Run the training and plot the loss and accuracy curves. (Note that for the training loss, we are plotting one point for each iteration, but for the validation loss, we have one point for every epoch. Make sure you align them.) Feel free to change the batch size, the learning rate, and the number of epochs. Based on the learning curves, which epoch do you think is the best to stop the training?
    c. (0.5 points) Implement an MLP that replaces the first two convolutional layers with a single dense layer with 400 neurons. The input of the MLP is flattened images. Run the training and plot the learning curves.
    d. (1 point) Visualize the weights of 20 neurons in the first hidden layer. To visualize them, you need to reshape the weights into the image size (3*32*32).
    e. (0.5 points) Visualize the 2D CNN filters of the first convolutional layer. The weights of each filter are also reshaped to a color image.
    f. (0.5 points) Discuss your findings by comparing the performance of MLP and CNN and your visualized filters.

2. (6 points + 2 bonus) Constrained Learning Challenge for Cancer Detection
    You have to design a CNN image classifier from scratch for this question. In “detect_cancer.zip”, we have provided you with the training set: “train_x.h5”, “train_y.h5”, and the validation set “val_x.h5”, “val_y.h5”.
    a. (1 point) Define your custom dataset that gets images from the h5 file. You can use “h5py” package in python to read the .h5 files. The dataset must return a tuple containing the image tensor and the corresponding label.

b. (4 point + 2 bonus) You are free to design your own custom CNN classifier from scratch and write the training code. Report the training and validation accuracies.

However, there are some important **constraints** to follow: (1) Your maximum training steps (not epochs!) must be **5,000**, (2) the seed value must be set to **0** (using `torch.manual_seed(0)`) (3) the total trainable parameters cannot exceed **250,000**, and (4) the batch size must be **less than or equal to 16**.

Respecting these constraints, you are free to explore! You can create your own unique architectural designs, construct any custom loss functions, optimization techniques, use any training strategies, and any forms of data augmentation to achieve the best results. Remember to keep track of all the results for every modification you have made (use *git* if you like), this will be required for your report.

The TAs possess the actual test set using which we shall test your model performance. You will be required to submit the model weights (mapped back to CPU) in a **.pth** file along with the network module code.

> Students with the **best 5 test accuracies** shall get **two bonus points** for creating a well-designed model and training strategy!

The 4 points are split as follows:

- (1.5 point) Coding Effort
- (2.5 points) Total score calculated using the formula

$$\mathbf{score} = \min\left[\max\{0, 4(\mathbf{acc} - 0.75)\} \times \left(\frac{\max(0, \mathbf{rem})}{250000}\right)^2 \times 25, 2.5\right]$$

  where –
  - $\mathbf{acc}$**:** Test Accuracy
  - $\mathbf{rem}$**:** Remaining trainable parameters = 250000 – trainable_params
  - $\mathbf{score}$**:** Total score for this sub-question

The key takeaway from this scoring technique is that (1) models **below 75%** accuracy receive **0**, (2) higher accuracy **increases** your score, (3) using **fewer** parameters **increases** score **quadratically** (remember, too few parameters might affect your accuracies), (4) models with **more than 250000 trainable parameters** will receive **0**, and (5) **large** models with **marginal** gains are **heavily penalized**.

Your task is to optimize your model given these constraints! We have created a function `test_score()` which you can use on the validation set to obtain your possible approximate score for this problem. We shall also post the top 5 test accuracies in the announcements once the grading is done.

c. (1 points) You must create a report describing your process of choosing the best model. This must include –
    i. A short, compressed history of the important modifications made to converge to the best model.
    ii. **Preprocessing** (including data augmentation if any): How was it done? Why choose that method?
    iii. **Model Architecture**: Describe the model architecture. Why choose this way of modelling the architecture? How do you think it allows for better performance?
    iv. **Loss Function**: How was it formulated? Why choose this/these loss/losses?
    v. **Optimization**: Describe the optimization strategy. Why choose this strategy?
    vi. **Specific training strategies**: Describe the training strategies used. Why choose this strategy?
    vii. Any other important information.

For answering any of the questions, feel free to refer to the history of all your modifications. You must also add your explanation of why that change performed better/worse as well.

Good luck!