CS 6886: Systems Engineering for Deep Learning

Assignment 3

Getting into the Flow of Deep Learning

Instructions

1. This assignment is released on 22nd Apr and is due on **10th May** (midnight)
2. The total marks (in overall weightage for course) is **10 marks** for this assignment. There are bonus marks on optional questions.
3. You are required to submit a zip file of your submissions. Include all plots and tables as part of a PDF report in the zip file. In addition, you are required to add models and Python notebooks in the zip file.
4. Copying of code or parts of the report is disallowed.
5. You are allowed to discuss with each other and refer to sources online. You are required to clearly state any such collaboration or references in the report.

Part A (10 points)

Preparation

1. This assignment is based on the Fashion MNIST classification problem. Go through the details of the dataset from the official repository. Download the train and test datasets with 60,000 and 10,000 images, respectively. We will use these datasets in all subsequent benchmarks and evaluations.
2. Also get familiar with MLFlow which we discussed in a recitation. You can read a paper and follow this tutorial to get familiar.
3. You will run the experiments on a Google Colaboratory instance. This will ensure that all of you run your experiments on similar setups which would enable comparison of inference times. You may follow a tutorial on the tool if you are not familiar with it.
4. Since this assignment involves comparison of different solutions, we have to standardise on the DL framework to use. **All problem statements must be done with PyTorch.**
5. When required to submit your model as part of the zip file, use this serialisation method of PyTorch.

Step 1 (2 points)

6. To start off you will train a basic model on the Fashion MNIST dataset which has the following layers:
    a. Conv2d with 16 (3,3) filters
    b. Conv2d with 16 (3,3) filters
    c. Fully connected layer with 100 outputs
    d. Fully connected layer with 10 outputs
7. Use a Xavier-uniform initialisation, cross-entropy loss function, and SGD optimiser with momentum. Note that you are **not allowed to use GPUs** for this training. Use the standard CPU core in the Google Colaboratory instance.
8. (2 points) Plot the loss function as a function of the training progress. Also include the final train and test accuracy of your model. In the zip file of the submission, include the pickle
9. (Bonus 1 point) It is often useful to graphically visualise the performance of your model on the data. One tool to do this is Google Facets. Think of a useful way to visualise the performance of your model on Google Facets. Include a screenshot and a description in the report.

Step 2 (3 points)

10. In your model, there are several hyper-parameters which affect the train and test accuracy. These include the following (with **minimum required number of configurations** to consider for each parameter given in brackets):
    a. Model Hyperparameters
        i. Filter sizes for convolutional layers (at least 2 for each layer)
        ii. Channel sizes for convolutional layers (at least 3 for each layer)
        iii. Output size of hidden dense layer (at least 2)
        iv. For the two convolution and Linear layers specified, this amounts to at least 72 different configurations to evaluate.
    b. Training hyperparameters (Pick the best configuration obtained from (a.))
        i. Initialisation method for weights and biases (at least 3)
        ii. Batch size (at least 4)
        iii. Learning parameters - learning rate and momentum (at least 4)
        iv. For the above training hyperparameters, this amounts to at least 48 configurations to evaluate.
11. Change each of these parameters within reasonable values and run individual training sessions with an MLFlow experiment, logging test accuracy and training time (to first reach 90% train accuracy). (Hint: One way to do this in Google Colab is to log the last completed training hyperparameters into a text file, and while resuming, read the text file for the point from which training will resume).
12. In the report,
    a. (0.5 point) Identify the top 3 configurations (in terms of test accuracy and less overfitting) and include screenshots of the MLFlow overall table showing different experiments,

b. (0.5 point) Write a reasoning as to why certain hyper-parameters performed well.
c. (0.5 point) Draw a Pareto plot of all models wrt test accuracy and training time
d. (1.5 points) Give an example loss-vs-epoch plot for each of the following situations from your experiments:
    i. Learning rate is too high
    ii. Learning rate is too small
    iii. Momentum is too high
    iv. Batch size is too small
    v. Overfitting due to a large model

Step 3 (3 points)

13. Take the trained model which has the highest test accuracy in Step 2. You will try to optimise this network to be more efficient, i.e., reduce inference time. Among several techniques you will use *pruning.* The simplest form of pruning is to prune entire channels in convolutional layers and entire neurons in dense layers.
14. (1 points) For the first convolutional layer, you will try to prune channels. Propose a heuristic by which you can decide *one* channel to prune from the first layer (based on the trained weights and biases). You can emulate pruning by simply turning the weights and biases of that layer to 0. Forcing these parameters to be zero, recompute the test accuracy. Report the drop in accuracy, if any wrt Step 2. Now retrain the network for 5 epochs, while keeping the chosen layer forced to 0. Report the improved test accuracy. Thus, this pruning step is captured by a triplet (test accuracy before pruning, test accuracy after pruning, test accuracy after pruning and retraining). The difference between the first and last number is called the pruning drop.
15. (1 point) Repeat this experiment for layer 1 multiple times, removing one channel at a time, until the incremental pruning drop does not exceed 1% for a single step.
16. (1 point) Repeat these steps for the second convolutional layer and the two fully connected layers (dropping one neuron at a time). Report all results with the help of MLFlow tables.

Step 4 (2 points)

17. What good is a pruned model if you cannot infer faster. In this step you would do just that.
18. (0.5 point) Write an analytical formula that computes the number of FLOPs in the forward pass of one of your pruned networks, i.e., given number of channels of convolutional layers and neurons in FC layers compute the number of FLOPs for a single inference. Use this method to draw a Pareto plot of inference FLOPs in x-axis and test accuracy as measured in 16) in y-axis.
19. (0.5 point) Choose the model that has the lowest inference FLOPs amongst all models which have an accuracy greater than 87%. Report this model.

20. (1 point) Rewrite this model using matrix multiplication in PyTorch, i.e., without using conv2d/FC layers. The matrices should be properly loaded from the trained models. Measure the time taken for this computation and compare with the time taken of the original network in 8).
21. (Bonus 2 points) Write the optimised matrix multiplication code to work on GPUs. Measure now the time taken for your optimised code and the original network in 8) on GPUs. Is it faster?

---

22. (Bonus 1 point) Coin an engaging catch-22 puzzle that deeply relates to deep learning.