COMS 6998-015
PRAC DEEP LEARNING SYS
Instructor: Parijat Dube and Chen Wang

Homework 2

Due: Oct. 14, 2024

Colab Link

Problem 1 - *Training and I/O Optimization* 30 points

This assignment will give you experience with training, deploying, and making inferences from a deep learning model. We will work with a CNN in PyTorch to classify images. We will use the CIFAR10 dataset, which contains 50K 32×32 color images. The refence code is at **pytorch-cifar**. We will work with the ResNet-18 model, as described in **Deep Residual Learning for Image Recognition**.

Model

Create a ResNet-18 model as defined in **Deep Residual Learning for Image Recognition**. You can rely on existing open-source implementations. However, your code should define the layers and not just import the model using torch.

Specifically, The first convolutional layer should have 3 input channels, 64 output channels, 3×3 kernel, with *stride*=1 and *padding*=1.
Followed by 8 basic blocks in 4 sub groups (i.e. 2 basic blocks in each subgroup):

- The first sub-group contains convolutional layer with 64 output channels, 3×3 kernel, *stride*=1, *padding*=1.
- The second sub-group contains convolutional layer with 128 output channels, 3×3 kernel, *stride*=2, *padding*=1.
- The third sub-group contains convolutional layer with 256 output channels, 3×3 kernel, *stride*=2, *padding*=1.
- The forth sub-group contains convolutional layer with 512 output channels, 3×3 kernel, *stride*=2, *padding*=1.
- The final linear layer is of 10 output classes.

For all convolutional layers, use ReLU activation functions, and use batch normal layers to avoid covariant shift. Since batch-norm layers regularize the training, set bias to 0 for all the convolutional layers. Use SGD optimizers with 0.1 as the learning rate, momentum 0.9, weight decay 5e-4. The loss function is cross entropy.

DataLoader

Create a PyTorch program with a DataLoader that loads the images and the related labels from the The torchvision CIFAR10 dataset. Import CIFAR10 dataset for the torchvision package, with the following sequence of transformations:

1. Random cropping, with size 32×32 and padding 4
2. Random horizontal flipping with a probability 0.5
3. Normalize each image's RGB channel with mean(0.4914, 0.4822, 0.4465) and variance (0.2023, 0.1994, 0.2010)

You will only need one data loader to complete this assignment. For your convenience, here are the default settings for the train loader: minibatch size of 128 and 3 IO processes (i.e., *num_workers*=2).

Homework 2

C1: Training in PyTorch — 10 points

Create a main function that creates the *DataLoaders* for the training set and the neural network, then run 5 epochs with a complete training phase on all the minibatches of the training set.
Write the code as device-agnostic, use the *ArgumentParser* to be able to read parameters from input, such as the use of cuda, the *data_path*, the number of dataloader workers and the optimizer (as string, eg: '*sgd*'). **Calculate the per-batch training loss, value and the top-1 training accuracy of the predictions, measured on training data.**
Note: (i) Typically, we would like to examine test accuracy as well, however, it is sufficient to just measure training loss and training accuracy for this assignment. (ii) You don't need to submit any outputs for C1. You'll only need to submit the relevant code for this question. C2-C3 will be based on the code you wrote for C1.

C2: Time Measurement of code in C1 — 10 points

Report the running time (by using *time.perf_counter()* or other timers you are comfortable with) for the following sections of the code:
(C2.1) Data-loading time for each epoch
(C2.2) Training (i.e., mini-batch calculation) time for each epoch
(C2.3) Total running time for each epoch.

Note: Data-loading time here is the time it takes to load batches from the generator (exclusive of the time it takes to move those batches to the device).

C3: I/O optimization for code in C2 — 10 points

(C3.1) Report the total time spent for the *Dataloader* varying the number of workers starting from zero and increment the number of workers by 4 (0,4,8,12,16...) until the I/O time does not decrease anymore. Draw the results in a graph to illustrate the performance you are getting as you increase the number of workers
(C3.2) Report how many workers are needed for the best runtime performance.

Problem 2 - *Training a simple chatbot using a seq-to-seq model* (25 points)

We will train a simple chatbot using movie scripts from the Cornell Movie Dialogs Corpus based on the **PyTorch Chatbot Tutorial**. This tutorial allows you to train recurrent sequence-to-sequence model. You will learn the following concepts:

- Handle loading and pre-processing of **the Cornell Movie-Dialogs Corpus dataset**
- Implement a sequence-to-sequence model with **Luong attention mechanism(s)**
- Jointly train encoder and decoder models using mini-batches
- Implement greedy-search decoding module
- Interact with the trained chatbot

We will use the code in the tutorial as the starting code for the assignment:

1. **[5 points]** Make a copy of the notebook of the tutorial, follow the instructions to train and evaluate the chatbot model in your local Google Colab environment

Homework 2

2. Learn how to use Weights and Biases (W&B) to run a hyperparameter sweep and instrument the notebook to use the Weights and Biases integration to help you run some hyperparameters sweeps in the next steps. Watch the video tutorial provided in the references section.
3. **[5 points]** Create a sweep configuration using the using the **W&B Random Search** strategy for the following hyperparameters:
   - Learning rate: [0.0001, 0.00025, 0.0005, 0.001]
   - Optimizer: [adam, sgd]
   - Clip: [0, 25, 50, 100]
   - teacher_forcing_ratio: [0, 0.5, 1.0]
   - decoder_learning_ratio: [1.0, 3.0, 5.0, 10.0]
4. **[5 points]** Run your hyperparameter sweeps using the GPU-enabled Colab and observe the results in the W&B console.
5. **[10 points]** Extract the values of the hyperparameters that give the best results (Minimum loss of the trained model). Explain which hyperparameters affect the model convergence. Use the feature importance of W&B to help guide your analysis.

References:

- The Cornell Movie Dialogs Corpus
- Hyperparameter sweeps with Weights and Biases Framework video tutorial
- Sample Google Colab project that accompanies the video above
- Weights and Biases Website

Problem 3 - *Serving and Scaling the Chatbot Model* (30 points)

Building upon the chatbot model developed in Problem 2, we will now focus on serving the model and scaling it for batch inference using Ray and TorchX. This problem will cover the following concepts:

- Implementing a server to expose the chatbot model via an API
- Using Ray for efficient batch prediction
- Leveraging TorchX and Ray for distributed inference

Complete the following tasks:

1. **[10 points]** Implement a Flask API to serve the trained chatbot model:
   - Create a Flask server that loads the trained model
   - Implement an endpoint that accepts user input and returns the model's response
   - Ensure proper error handling and input validation
2. **[10 points]** Implement batch prediction using Ray:
   - Follow the tutorial at Ray Batch Prediction Tutorial

- Modify your code to use Ray for batch prediction of chatbot responses
- Start a local Ray cluster on your laptop for testing
- Compare the performance of batch prediction with and without Ray

3. **[10 points]** Implement distributed inference using TorchX and Ray:
   - Follow the tutorial at Distributed Training with TorchX and Ray
   - Rewrite your batch inference job using TorchX
   - Configure the job to use the Ray scheduler
   - Launch the distributed inference job and analyze its performance

Submission Guidelines:

- Submit your code files for each part of the question
- Include a brief report (maximum 2 pages) discussing:
  - Your implementation approach for each part
  - Challenges faced and how you overcame them
  - Performance comparisons between different approaches (single-instance vs. Ray batch vs. TorchX distributed)
  - Potential improvements or optimizations for real-world deployment

References:

- Flask Documentation
- Ray Documentation
- TorchX Documentation

Problem 4 - *Paper Reading* (15 points)

Choose one research paper from the provided list that aligns with your interests or area of study. Read the paper thoroughly and critically, then address the following points:

a) (5 points) Summarize the main contributions of the paper. What are the key findings or innovations presented?

b) (5 points) Highlight one significant concept, technique, or insight that you learned from this paper which was previously unknown to you. Explain why you find this new knowledge valuable or interesting.

c) (5 points) Brainstorm and propose one potential improvement to the research or a novel application of the paper's findings. This could be an extension of the work, a different approach to the problem, or an application in a new domain.

Your response should be concise yet comprehensive, maximumly one page long. Ensure your work is original and any references to external sources are properly cited.

COMS 6998-015
PRAC DEEP LEARNING SYS
Instructor: Parijat Dube and Chen Wang

Homework 2

Due: Oct. 14, 2024

Important: *The paper selection and reading will contribute to your project proposal which is due in mid-October. Ideally you should have formed a project team and you and your team mate should identify papers from the list that are related to your project idea. Then you can divide the readings of the selected papers between the two of you. The selected papers will form related work for your project.*

Submission: Include your response in your homework document under "Problem 4". Clearly indicate which paper you chose at the beginning of your answer.