High Performance Machine Learning

Homework Assignment 3

Dr. Kaoutar El Maghraoui and Dr. Parijat Dube

Due Date: March 31, 2024 Spring 2024 Max Points: 100

Instructions:
This lab is intended to be performed **individually**, great care will be taken in verifying that students are authors of their own submission.

Problem 1 - *Training a simple chatbot using a seq-to-seq model* (50 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. 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 (**10 points**)
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. Create a sweep configuration using the using the **W&B Random Search** strategy for the following hyperparameters: (**15 points**)
    - 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. Run your hyperparemeter sweeps using the GPU-enabled Colab and observe the results in the W&B console. (**5 points**)

5. Extract the values of the hyperparameters that give the best results (Minimum loss of the trained model). Explain which hyperparemeters affect the model convergence. Use the feature importance of W&B to help guide your analysis. (**10 points**)

6. Use the Pytorch profiler to measure the time and memory consumption of the selected model's operators. You can see an example how to do this here: PyTorch Profiler Example (**5 points**)

7. Use the Pytotch profiler to examine the sequence of profiled operators and CUDA kernels in Chrome trace viewer (chrome://tracing) as shown in the reference example: PyTorch Profiler Example (**5 points**)

8. Save the trained model that had the lowest loss.

***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
- PyTorch Profiler Example

Problem 2 - *Training a simple chatbot using a seq-to-seq model* **(50 points + 10 points Bonus)**

In this part of the lab, you will experiment with the process of transitioning the sequence-to-sequence model that you trained to TorchScript using the TorchScript API. This module has two core modalities for converting an eager-mode model to a TorchScript graph representation: **tracing** and **scripting**. You can use this **PyTorch Torchscript tutorial** to help you perform this section of the lab.

1. **Q1:** Explain the differences between tracing and scripting and how they are used in TorchScript. (**2 points**)

2. **Q2:** Explain the changes needed in the chatbot model to allow for scripting. (**3 points**)

3. Convert the model that you trained in the previous exercise to Torchscript (**10 points**)

4. Print graph of the converted model (**10 points**)

5. Evaluate the Torchscript model (**10 points**)

6. **Q3:** Compare the evaluation latency of the TorchScripted model and the regular original PyTorch model on CPU and GPU. How much speedup you are getting if any? You need to create a latency comparison table similar to what is shown in the tables below for your trained model (**10 points**)

	Latency on CPU (ms)	Latency on GPU(ms)
PyTorch	25.96	4.02
TorchScript	23.01	2.41

PyTorch vs TorchScript for ResNet

	Latency on CPU (ms)	Latency on GPU(ms)
PyTorch	86.23	16.49
TorchScript	81.57	10.54

PyTorch vs TorchScript for BERT

7. Save and serialize it for use in a non-Python deployment environment. (**5 points**)
8. **(Bonus Question)**: Show how to use the model in a non-Python environment. For example in a C++ program. (**10 points**)

***References**:*

- PyTorch JIT and TorchScript

Appendix - Submission Instructions

Please submit a *.targz* archive containing the following files:

- A Jupyter notebook file named chatbot.ipynb containing all the code needed to train, evaluate and torchscript the chatbot model.
- the saved model files with no hyperparameter tuning
- the saved model file with hyperparameter tuning
- the Serialized torchscripted model file
- A report with your name and link to your wandb project, graphs from your hyperparameter sweeps, answers to the questions Q1, Q2, and Q3. Make sure that your W&B project is publicly accessible.
- The C++ code that you created if you decide to do the bonus question called nonPython_chatbot.cc and Readme file explaining how to run your code.