Homework 11

11-967

Due date: 04/13/2025 11:59 PM EDT

In this homework, you will explore strategies for making language models more efficient both in their training and inference phases. Large language models require significant computational resources due to their high memory consumption, large parameter counts, and the extensive compute needed for both training and serving. Efficiently training these models involves optimizing memory usage and using distributed training setups, which allow us to leverage multiple GPUs to process larger models or batches without running out of memory. Here you learn efficient **inference**.

For this homework, if you are using AWS, please use the following setup, which is the setting we tested on:

- Instance: g5.2xlarge
- Image: ami-0aada1758622f91bb
- Disk space: 100 GB

There is a bit of coding in this homework, but there is no code submission on Gradescope. You only need to upload your PDF.

Problem 1: Efficient Inference

[Question 1.1] (***Code, 5 points***) : In this problem, we will explore efficient inference. First, take a look at huggingface's `generate()` method. Next, take a look at vllm's `generate()` method.

DELIVERABLES FOR Q1.1

A. Implement both generate methods in `lm_inference.py` according to the following setting:

- The input to the model is "hello"
- The sampling strategy is greedy sampling for both generate methods
- Generate the number of output tokens according to `num_new_tokens` variable provided in the script

B. Now run `lm_inference.py`. Attach the resulting plot here. Is one `generate()` method faster than the other? In one sentence, describe why you think this is the case.

[Question 1.2] (***Written, 2 points***) There are many efficient training methods people are using nowadays, such as flash attention, gradient checkpointing, data parallelism, and deepspeed zero 1/2.

DELIVERABLES FOR Q1.2

A. For each of the above techniques, justify whether or not it makes sense to use them during inference.

[Question 1.3] (***Written, 5 points***) Answer the following questions regarding other efficient inference techniques.

DELIVERABLES FOR Q1.3

A. One method for efficient inference is using PagedAttention. In a regular setting, the KV cache, which stores the key and value weights, requires varying amounts of memory depending on the sizes of the inputs. In order to handle this, a contiguous chunk of memory with the maximum input length (e.g., 2048 tokens) is pre-allocated. List **two** reasons why this is not effective.

B. In order to fix this problem, how does PagedAttention store the KV cache?

C. Quantization reduces the memory used by model parameters by mapping them to lower precision, and is another method for efficient inference. Static quantization requires a calibration dataset. Suppose the model performs well on a set of tasks, and we want the quantized model to also perform well on these tasks. What is one way to construct the calibration dataset?

D. Consider a model that was trained with fp32. What problem can occur if we downcast to fp16 for inference?