Assignment 4 - Quantization and Model Compression

> CS7.501: Advanced NLP

> Instructor: Manish Shrivastava

> Due Date: 14th November

General Instructions

1. You should implement the assignment in Python.
2. Ensure that the submitted assignment is your original work. Please do not copy any part from any source, including your friends, seniors, or the internet. If any such attempt is caught, serious actions, including an F grade in the course, are possible.
3. A single .zip file needs to be uploaded to the Courses Portal.
4. Please start early to meet the deadline. Late submissions won't be evaluated.

Introduction

As LLMs grow increasingly complex, the resource demands for deploying these models also escalate, particularly in terms of memory, computational power, and inference latency. Many applications, especially on edge devices, cannot support full-precision large models. Quantization offers a powerful solution by reducing the bit-width of model weights and activations, effectively minimizing resource requirements while retaining accuracy within acceptable limits. This assignment introduces quantization techniques that can be applied to LLMs, showing how they can maintain robust performance with a lighter computational load. Through practical exploration and analysis, you'll learn how to apply and evaluate various quantization strategies.

Quantization is utilized across fields like mobile development, Internet of Things (IoT), and healthcare. In mobile development, quantization enables large scale LLMs to run efficiently on devices with limited computational power, providing responsive, on-device AI for apps. In IoT, where sensors and smart devices often operate on constrained power and memory, quantization allows for the deployment of intelligent systems directly on edge devices, reducing latency and reliance on cloud connectivity. Furthermore, in fields like autonomous driving and robotics, quantization makes real-time processing possible by reducing the memory and latency required for complex decision-making tasks. As LLMs expand into these domains, efficient techniques such as quantization become essential for deploying sophisticated AI applications at scale.

Part 1: Quantization from Scratch

> 50 marks

Background

Quantization involves reducing the precision of a model's parameters, often from floating-point formats (e.g., FP16) to lower-bit integers, such as 8-bit or even 4-bit representations. This process enables models to consume less memory and run faster, with minimal accuracy degradation. While entire models can be quantized, selectively quantizing specific components (e.g., attention layers, feed-forward networks) can achieve a balance between efficiency and performance.

Tasks

- **Whole-Model Quantization**: Convert a pre-trained model from the original precision of the model (FP16/FP32) to INT 8-bit precision.
- **Selective Component Quantization**: Quantize only specific model components, such as maybe only a few selective blocks of the decoder (recommended), or the Feed-Forward Network (FFN) weights or specific attention matrices (query, key, and value), while leaving other weights at full/original precision.

    - Feel free to try out various component configurations and analyze their impact (this part is not graded)

These tasks need to be done without using any quantization libraries. Keep activations unmodified (original precision) in the above tasks.

Analysis and Evaluation

- Analyze how whole-model and selective quantization affect model performance and metrics.
- Metrics and evaluation criteria that need to be analyzed for the above tasks are mentioned at the end of the document in red.

Part 2: Bitsandbytes Integration and NF4 Quantization

> 48 marks

Background

Bitsandbytes is a popular library that facilitates efficient low-bit quantization on the Hugging Face platform, making it easier to work with large language models on constrained devices. Nonlinear quantization methods, such as NF4 (Nonlinear 4-bit quantization), apply non-linear scaling to values, which can help maintain model accuracy while further reducing bit precision.

Tasks

- **Bitsandbytes Quantization**: Use Bitsandbytes to quantize a model from original precision (FP32/FP16) to both 8-bit and 4-bit formats.
- **NF4 Quantization**: Apply NF4 quantization and compare its effectiveness with linear quantization.

Analysis and Evaluation

- Explain the concept of NF4 quantization and how it differs from linear quantization scales.
- Discuss the impact of linear vs. nonlinear quantization on model accuracy and efficiency.
- Metrics and evaluation criteria that need to be analyzed for the above tasks are mentioned at the end of the document in red.

Bonus: GGML Models, GGUF Formats, and llama.cpp

> 2 marks

Background

GGML (General Graph Model Library) and GGUF (General Graph Unified Format) are lightweight model formats designed for efficient local deployment. GGML models are especially useful for running large language models (LLMs) on devices with limited resources, making them accessible for edge computing and mobile applications. Llama.cpp is a tool that enables running GGML models locally, facilitating on-device processing.

Tasks

- **Using GGML with llama.cpp**: Deploy a GGML model on your local device using llama.cpp.
- **Model Conversion and Deployment**: Convert a Hugging Face model to GGML format, run it locally, and push the converted model to the Hugging Face Hub.

Analysis and Evaluation

- Explore GGML and GGUF models, their format, and deployment benefits.
- Metrics and evaluation criteria that need to be analyzed for the above tasks are mentioned at the end of the document in red.

Analysis and Evaluation

Model to be used

- This up to your discretion. Make sure the same model is used across all experiments. You can download/use the models from huggingface (GPT2, Llama, Phi3, Gemma etc)

Metrics for Each Part

1. **Memory Footprint**: Record memory usage before and after quantization.
2. **Inference Latency**: Measure inference latency pre- and post-quantization.
3. **Perplexity**: Compute perplexity on either the Wikipedia or PTB (Penn Tree Bank) dataset pre- and post-quantization. Recommended minimum size: 3000 data points.

Analysis

- Discuss the changes in perplexity, speed, and memory usage after quantization.
- Evaluate the impact of each quantization approach on these metrics, using visualizations and graphs where applicable.
- Summarize findings in a report that includes both quantitative and qualitative insights.

Submission Format

Zip the following into one file and submit it on the Moodle course portal:

- Source code (Should include .py files only.)
- Quantized models (if file size permits, otherwise provide a link)
- Report answering all the questions in PDF format

README file (should contain instructions on how to execute the code and any other information necessary for clarity)
If the model size crosses the file size limits, upload them to your OneDrive folder and share the read-only accessible links in the README file.

FAQs

1. Can I reduce size of dataset or number of epochs due to computational constraints?
   Yes, you can. However, make sure to mention this in your report, with appropriate justification for your choices.
2. Can I submit my code in Jupyter Notebooks?
   Only the Python script will be considered for grading.
3. Purple sections are not graded

Resources

https://www.datacamp.com/tutorial/quantization-for-large-language-models

Exploration Directions

If you're interested in further exploring quantization, here are some advanced topics

- **GPTQ**
- **AWQ**
- **Brain Float 16 (BF16) Quantization**
- **1.5-bit Quantization**
- **Quantization-Aware Training (QAT)**
- Does a larger quantized model outperform a smaller non-quantized model in terms of both efficiency and accuracy? Which one is better?