Report on Quantization Using Different methods.

AWQ Quantization of LLaMA-3.2-1B Model

Objective: Evaluate the performance of the LLaMA-3.2-1B model after AWQ quantization, comparing memory usage, inference time, and response quality against the original model.

Executive Summary

The quantization of the LLaMA-3.2-1B model using AWQ (Activation-aware Weight Quantization) to optimize memory usage and inference speed. The quantized model achieved significant memory savings (approximately 60% reduction) while maintaining comparable response quality, though inference times were longer than the original model. The experiment was conducted on a GPU-enabled environment (CUDA) using the Hugging Face Transformers and AutoAWQ libraries. The results demonstrate that AWQ quantization is a viable approach for deploying large language models on resource-constrained environments, with some trade-offs in inference speed.

Model Details

- Model: LLaMA-3.2-1B (unsloth/Llama-3.2-1B)

- Source: Hugging Face Hub

- Quantization Method: AWQ (Activation-aware Weight Quantization)

- Quantization Configuration:

    - Zero-point quantization: Enabled
    - Quantization group size: 128
    - Weight bit-width: 4 bits
    - Version: GEMM

- Calibration Data: 256 examples from the C4 English dataset (first 512 characters of each example)

- Tokenizer: AutoTokenizer from Hugging Face, configured with padding and EOS token

- Hardware: GPU (CUDA-enabled, T4 GPU)

- Software:

    - Python 3
    - PyTorch
    - Transformers
    - AutoAWQ
    - Datasets (for calibration data)

Experimental Setup:

The experiment involved the following steps:

- Tokenizer Loading: Loaded the tokenizer for LLaMA-3.2-1B.
- Calibration Data: Retrieved 256 text samples from the C4 dataset for quantization calibration.
- Quantization: Applied AWQ to the model, reducing weight precision to 4 bits.
- Testing: Loaded both the quantized and original models to compare memory usage and response quality.
- Prompt Testing: Evaluated both models on five diverse prompts to assess response quality and inference time.

Generation Parameters:

- Maximum new tokens: 200

- Number of beams: 10 (beam search for higher quality)

- No-repeat n-gram size: 2

- Early stopping: Enabled

- Temperature: 0.5 (not used as `do_sample=False`)

Results:

Memory Usage

- Baseline GPU Memory: 559.26 MB (after clearing cache)

- Original Model Memory: 3900.52–3900.64 MB

- Quantized Model Memory: 1542.51–1543.51 MB

- Memory Reduction: Approximately 60% (from ~3900 MB to ~1543 MB)

The quantized model significantly reduced memory footprint, making it more suitable for deployment on resource-constrained devices.

Inference Time

Prompt #	Prompt Description	Original Model (s)	Quantized Model (s)	Difference (s)
1	Capital of France	18.1827	44.0275	+25.8448
2	Robot story	5.7728	40.8406	+35.0678
3	Car distance calculation	5.2746	41.9568	+36.6822
4	Sky appears blue	7.8843	40.1865	+32.3022
5	Favorite book	17.8862	34.5810	+16.6948
Average		**11.0001**	**40.3185**	**+29.3184**

Response Quality (Accuracy and Relevance)

Prompt 1: Capital of France

- Original: Provided a detailed response identifying Paris as the capital and including additional context about France. Accurate but verbose.

- Quantized: Correctly answered with a multiple-choice format (Paris, Marseille), but included extraneous information about other countries. Slightly less focused but accurate.

- Assessment: Both models answered correctly, with the original being more descriptive and the quantized more concise but less relevant.

Prompt 2: Robot Story

- Original: Did not generate a story, instead providing instructions for writing one. Failed to meet the creative task requirement.

- Quantized: Similarly provided instructions rather than a story, with additional constraints (e.g., 1,000 words). Also failed to meet the requirement.

- Assessment: Both models underperformed, likely due to the model's tendency to interpret creative prompts as instructional tasks.

Prompt 3: Car Distance Calculation

- Original: Correctly calculated the distance (150 miles) but presented it as a multiple-choice question with incorrect options (48 miles selected). Inconsistent.

- Quantized: Also calculated correctly but chose an incorrect multiple-choice answer (48 miles). Included additional irrelevant details about speed units.

- Assessment: Both models understood the problem but provided incorrect answers due to the multiple-choice format, indicating a potential issue with prompt interpretation.

Prompt 4: Sky Appears Blue

- Original: Incorrectly attributed the blue sky to "different wavelengths" without mentioning Rayleigh scattering. Partially correct but incomplete.

- Quantized: Mentioned Rayleigh scattering and water vapor, which is closer to the correct explanation, though water vapor is not a primary factor. More accurate but still imperfect.

- Assessment: The quantized model provided a slightly better explanation, though both responses lacked full scientific accuracy.

Prompt 5: Favorite Book

- Original: Chose “The Lord of the Rings” with a personal explanation, meeting the prompt’s requirements. Relevant and coherent.

- Quantized: Chose “The Catcher in the Rye” with a brief explanation, also meeting the requirements. Slightly less detailed but relevant.

- Assessment: Both models performed well, with the original providing more detail and the quantized being more concise.

Quantization Process

- Quantization Time: Approximately 9 minutes and 53 seconds (593 seconds) for 16 layers.

GGUF Quantization Report for Llama-3.2-1B Model

Executive Summary

The unsloth/Llama-3.2-1B model from Hugging Face to the GGUF format, followed by quantization to Q4_K_M and Q8_0 formats using llama.cpp. The quantized models were tested for inference performance, memory usage, and output quality using a set of predefined prompts. The results demonstrate significant reductions in model size and inference time with quantization, though with some trade-offs in output quality.

Model Overview

- **Model ID**: unsloth/Llama-3.2-1B
- **Source**: Hugging Face Hub
- **Original Format**: PyTorch (Hugging Face Transformers)
- **Converted Format**: GGUF (FP16)
- **Quantized Formats**:
    - Q4_K_M
    - Q8_0

Methodology

1. **Setup**:
    - Cloned llama.cpp repository
    - Installed dependencies: git, cmake, build-essential, psutil, torch, transformers, huggingface_hub.

2. **Model Conversion**:
    - Downloaded the model from Hugging Face using snapshot_download.
    - Saved the tokenizer using AutoTokenizer.
    - Converted the model to GGUF FP16 format using convert_hf_to_gguf.py.
3. **Quantization**:
    - Quantized the GGUF FP16 model to Q4_K_M and Q8_0 using llama-quantize binary.
4. **Inference Testing**:
    - Tested three models: Original FP16, Quantized Q4_K_M, and Quantized Q8_0.
    - Used three test prompts:
        - "What is the capital of France?"
        - "Explain the theory of relativity in simple terms."
        - "Write a short poem about the stars."
    - Measured inference time, memory usage, and output quality.

Results

Model Sizes

- **Original FP16 GGUF**: ~2.47 GB (estimated based on original safetensors file size)
- **Quantized Q4_K_M**: ~1.2 GB (approximate, based on typical Q4_K_M compression ratios)
- **Quantized Q8_0**: ~1.8 GB (approximate, based on typical Q8_0 compression ratios)

Inference Performance

the inference time and memory usage for each model and prompt:

Model	Prompt	Inference Time (s)	Baseline Memory (MB)	Peak Memory (MB)
Original FP16	Capital of France	109.96	1566.64	1566.64
Original FP16	Theory of Relativity	117.84	1566.64	1566.64
Original FP16	Poem about Stars	108.10	1566.64	1566.90
Quantized Q4_K_M	Capital of France	34.91	1566.90	1566.90

Quantized Q4_K_M	Theory of Relativity	35.27	1566.90	1566.90
Quantized Q4_K_M	Poem about Stars	15.01	1566.90	1566.90
Quantized Q8_0	Capital of France	81.94	1566.90	1566.90
Quantized Q8_0	Theory of Relativity	72.04	1566.90	1566.90
Quantized Q8_0	Poem about Stars	54.51	1566.90	1566.90

Output Quality

- **Original FP16**:
    - **Capital of France**: Correctly identified Paris, France, but repeated the question multiple times.
    - **Theory of Relativity**: Provided a partial explanation, mentioning the speed of light as a relative speed limit, but cut off before completion.
    - **Poem about Stars**: Misinterpreted the prompt, generating a response about writing a sonnet with specific requirements rather than a poem.
- **Quantized Q4_K_M**:
    - **Capital of France**: Correctly identified Paris and provided additional context about France's administrative divisions, though with some repetition.
    - **Theory of Relativity**: Explained the principle of relativity and the constant speed of light, but the response was incomplete.
    - **Poem about Stars**: Failed to generate a poem, instead repeating the prompt with an unrelated addition about feeling alone.
- **Quantized Q8_0**:
    - **Capital of France**: Generated a list of unrelated questions about France's geography, failing to directly answer the prompt.
    - **Theory of Relativity**: Provided a definition of the speed of light and its constancy, but the response was incomplete and repetitive.
    - **Poem about Stars**: Failed to generate a poem, repeating the prompt with an instruction to draw an illustration.

Accuracy Observations

- The **Original FP16** model provided the most accurate response for the "Capital of France" prompt but struggled with completeness and relevance for the other prompts.
- The **Q4_K_M** model showed reasonable accuracy for the "Capital of France" and "Theory of Relativity" prompts but failed to produce a poem.

- The **Q8_0** model exhibited the lowest accuracy, generating irrelevant or repetitive responses across all prompts.
- Quantization introduced noticeable degradation in output quality, particularly for complex tasks like poem generation, with Q8_0 performing worse than Q4_K_M in terms of relevance.

Key Findings

1. **Performance**:
    - **Q4_K_M** was the fastest, with inference times ~3-7x lower than FP16 (15.01–35.27s vs. 108.10–117.84s).
    - **Q8_0** was faster than FP16 but slower than Q4_K_M (54.51–81.94s).
2. **Model Size**:
    - Quantization significantly reduced model size (~50% for Q4_K_M, ~30% for Q8_0), making them more suitable for resource-constrained environments.
3. **Output Quality**:
    - Quantization compromised output quality, with Q4_K_M retaining more coherence than Q8_0.
    - The FP16 model produced more accurate responses for simple factual queries but struggled with creative tasks.

Model Quantization Analysis for GPT-2

Objective: Evaluate the performance, accuracy, and efficiency of quantization techniques (LLM.int8() and GPTQ) applied to the GPT-2 model for deployment optimization.

1. Executive Summary

This report presents the results of implementing and testing two quantization techniques—LLM.int8() and GPTQ on the GPT-2 model to optimize its memory footprint, inference speed, and performance. The experiments were conducted using PyTorch on a CUDA-enabled GPU (T4).

Key findings include:

LLM.int4()): Achieves mixed-precision quantization with minimal accuracy loss (MSE: 1.5778e-05) and supports efficient matrix operations.

GPTQ: Reduces model size significantly (from 510 MB to 195.92 MB) with a 4-bit quantization.

Memory Efficiency: GPTQ model uses ~316.89 MB compared to 805.60 MB for the original GPT-2 model.

Inference Time: Original model is faster (average 1.4172 seconds) than GPTQ (average 2.7737 seconds).

Accuracy/Perplexity: GPTQ model has a perplexity of 180.87, indicating reasonable performance but potential degradation in response quality for complex prompts.

Response Quality: Quantized model responses are less coherent for open-ended prompts but acceptable for factual queries.

2. Experimental Setup

Environment

- Hardware: Google Colab with NVIDIA T4 GPU (16 GB VRAM)
- Software:
- Python 3.11
- PyTorch 2.6.0+cu124
- Transformers, BitsAndBytes, AutoGPTQ libraries
- Dataset: C4 English dataset (128 samples) for GPTQ calibration
- Model: GPT-2 (base model, 124M parameters)

Quantization Techniques

GPTQ:

- 4-bit quantization with group size 128, damping factor 0.01.

- Uses Hessian-based optimization for weight quantization.

- Calibration dataset: 128 samples from C4 English dataset.

- Non-quantized layer: `lm_head`.

Evaluation Metrics

- Accuracy: Mean Squared Error (MSE) for LLM.int4(), perplexity for GPTQ.

- Memory Usage: GPU memory footprint (MB).

- Inference Time: Time per prompt (seconds).

- Response Quality: Subjective evaluation of generated text coherence and relevance.

Accuracy

- LLM.int4():

  - MSE: 1.5778e-05 (compared to FP32 matrix multiplication).

  - Indicates high fidelity for quantized matrix operations.

- GPTQ:

  - Perplexity: 180.87 on a test set.

  - Suggests moderate degradation in language modeling capability compared to FP32 models (typical GPT-2 perplexity ~20-50 on similar datasets).

  - Response quality varies: Factual prompts (e.g., capital of France) yield reasonable answers, but creative prompts (e.g., short story) produce less coherent outputs.

Inference Time

Inference times were measured for five prompts using beam search (num_beams=5, temperature=0.5, max_new_tokens=100).

Question	Original Model (seconds)	GPTQ Model (seconds)
Capital of France	1.82	2.66
Robot Story	0.83	2.82
Car Distance	1.80	2.74
Sky Blue	1.32	2.70
Favorite Book	1.32	2.95
Average	**1.42**	**2.77**

Memory Efficiency

- Original Model: 805.60 MB GPU memory.

- GPTQ Model: 316.89 MB GPU memory (60.7% reduction).

- Baseline Memory: 185.19 MB (before loading models).

- Disk Size:

  - Original: ~510 MB (download size).

  - GPTQ: 195.92 MB (61.6% reduction).

Response Quality

Sample responses for the prompt "What is the capital of France, and what is its largest city?":

-Original:

  > The capital, Paris, is located in the center of the country. It has a population of 1.5 million people. The city is divided into two parts: the north and the south. In the northern part, there is a city called Marseille, which was founded by the French in 1789...

  - Issues: Incorrect population (Paris: ~2.2M), Marseille is not in the north, and historical inaccuracies.

- GPTQ:

  > The French capital, Paris, has a population of 1.5 million people. It is located in the heart of the French Riviera, which is a UNESCO World Heritage Site. The city is home to a number of museums...

  - Issues: Incorrect population and location (French Riviera is not near Paris).

For creative prompts (e.g., robot story), GPTQ responses were less coherent, often repeating phrases or diverging from the task.

Implementation Details

GPTQ

- Code: Uses AutoGPTQ library with 4-bit quantization.

- Process:

  - Hessian approximation using activation outer products.

  - Iterative quantization with Hessian inverse for error minimization.

  - Calibration with 128 C4 dataset samples.

LLaMA Model Quantization using GPTQ

Executive Summary

The quantization process and performance evaluation of the LLaMA-3.2-1B model, quantized to 4-bit precision using GPTQ. Conducted on a CUDA-enabled GPU, the process achieved significant memory reduction while maintaining acceptable response quality. Key metrics include memory usage, inference time, and qualitative response accuracy compared to the original model. The quantized model shows substantial memory efficiency but has slower inference times and occasional response inaccuracies.

Model and Quantization Details

- **Model**: LLaMA-3.2-1B (unsloth/Llama-3.2-1B)
- **Quantization Method**: GPTQ, 4-bit precision
- **Quantization Configuration**:
    - Bits: 4
    - Group Size: 128
    - Damping Factor: 0.01
    - Description Activation: Disabled
- **Calibration Dataset**: 128 samples from C4 English dataset, each limited to 512 characters
- **Device**: CUDA-enabled GPU
- **Output Directory**: llama-GPTQ

Performance Metrics

Memory Usage

- **Baseline GPU Memory**: 4776.94 MB
- **Quantized Model Memory**: 5708.85 MB
- **Original Model Memory**: 10423.11 MB
- **Memory Reduction**: Approximately 45.24% compared to the original model

Inference Time

Inference times were measured for five prompts, comparing the original and quantized models. The quantized model consistently exhibited longer inference times.

Prompt	Original Model (seconds)	Quantized Model (seconds)
1: Capital of France	4.5696	10.9780
2: Robot Story	2.5769	11.8616
3: Car Distance	3.9241	11.9497
4: Sky Blue	4.0849	11.9820
5: Favorite Book	3.8953	12.0766
Average	**3.8102**	**11.7696**

Accuracy and Response Quality

Responses were evaluated qualitatively for correctness and relevance. The original model generally provided more accurate and contextually appropriate responses, while the quantized model occasionally produced less precise or incomplete answers.

- **Prompt 1: Capital of France**
  - **Original**: Correctly identified Paris as the capital and provided additional context.
  - **Quantized**: Correct but formatted as a multiple-choice question, less detailed.
- **Prompt 2: Robot Story**
  - **Original**: Repeated the prompt without generating a story.
  - **Quantized**: Attempted a story but was incomplete and vague.
- **Prompt 3: Car Distance**
  - **Original**: Correctly calculated 150 miles with clear explanation.
  - **Quantized**: Incorrect and irrelevant response about speed.
- **Prompt 4: Sky Blue**
  - **Original**: Partially correct but oversimplified explanation.
  - **Quantized**: Incorrect, described sky as glass with dust particles.
- **Prompt 5: Favorite Book**
  - **Original**: Provided a coherent response favoring *The Lord of the Rings*.
  - **Quantized**: Similar but less detailed and slightly incomplete.

Process Overview

1. **Model Loading**: Loaded LLaMA-3.2-1B model and tokenizer from Hugging Face.
2. **Calibration Data**: Extracted 128 samples from the C4 dataset for quantization calibration.

3. **Quantization**: Applied 4-bit GPTQ quantization, saving the model to llama-GPTQ.
4. **Testing**: Evaluated both models on five diverse prompts, measuring memory, inference time, and response quality.

Conclusion

The quantization of LLaMA-3.2-1B to 4-bit precision achieved a 45.24% reduction in memory usage, making it suitable for resource-constrained environments. However, increased inference times and reduced response quality indicate trade-offs. Implementing recommended optimizations could enhance the quantized model's viability for deployment.