**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)

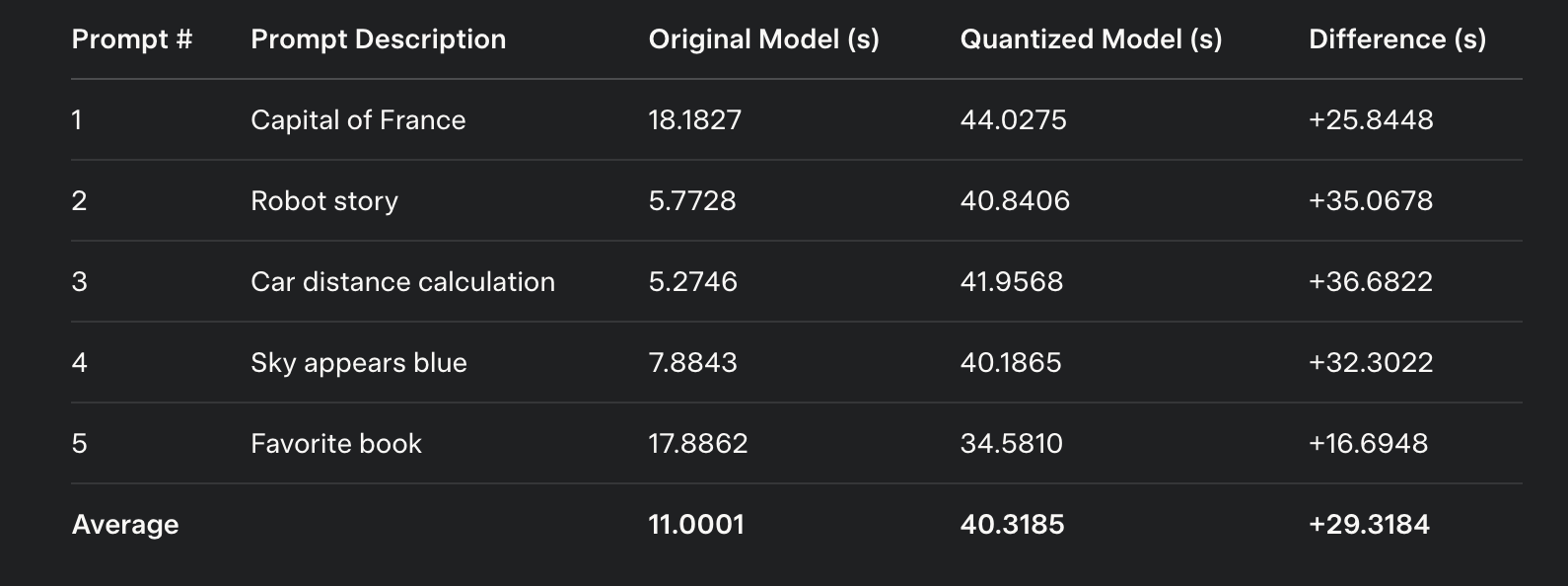
- Original Model Memory: 3900.52–3900.64 MB

- 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**

****

**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.  
  
  
 **Fine-Tuning and Testing Report for AWQ-Quantized LLaMA Model**

**Objective**

Fine-tune an AWQ-quantized LLaMA model using LoRA (Low-Rank Adaptation) on the Alpaca English demo dataset, evaluate its inference performance, memory usage, and response quality through interactive testing, and analyze the effectiveness of the fine-tuning process for deployment optimization.

**Executive Summary**

The AWQ-quantized LLaMA model, stored at `/content/AutoAWQ/llama-AWQ`, was fine-tuned using LoRA on the Alpaca English demo dataset (`alpaca\_en\_demo.json`) with 1000 examples. The fine-tuning process, conducted on a CUDA-enabled GPU, took 19 minutes and 49.82 seconds, achieving a final training loss of 1.0431 and an average train loss of 1.1613 over 3 epochs. The fine-tuned model, saved to `llama-AWQ-finetuned`, maintained a memory footprint of 1012.50 MB allocated and 2922.00 MB reserved during inference. Interactive testing showed average inference times of ~35.5 seconds for detailed prompts, with responses varying in quality—accurate for factual queries but verbose and sometimes off-topic for open-ended tasks. The results demonstrate that fine-tuning an AWQ-quantized model with LoRA is effective for adapting to specific datasets while preserving memory efficiency, though response quality and inference speed require further optimization.

**Model Details**

- Model: AWQ-quantized LLaMA

- Quantization Method: AWQ (Activation-aware Weight Quantization, pre-applied)

- Fine-Tuning Method: LoRA (Low-Rank Adaptation)

- LoRA Configuration:

* Rank (`r`): 16
* LoRA Alpha: 32
* Target Modules: `q\_proj`, `v\_proj`, `k\_proj`, `o\_proj`
* LoRA Dropout: 0.05
* Bias: None
* Task Type: Causal Language Modeling

- Dataset: Alpaca English demo (`alpaca\_en\_demo.json`, 1000 examples)

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

- Hardware: GPU (CUDA-enabled, specific GPU not specified)

- Software:

- Python 3

- PyTorch

- Transformers

- AutoAWQ

- PEFT (Parameter-Efficient Fine-Tuning)

- Datasets

**Generation Parameters**

- Maximum New Tokens: 200 (300 for code generation)

- Number of Beams: 3 (beam search for higher quality)

- No-Repeat N-gram Size: 2

- Early Stopping: Enabled

**Results**

**Memory Usage**

- Allocated GPU Memory: 1012.50 MB

- Reserved GPU Memory: 2922.00 MB

The fine-tuned model maintained a low memory footprint, consistent across all inference tests, making it suitable for resource-constrained environments.

**Training Performance**

- Total Training Time: 19 minutes and 49.82 seconds (1189.82 seconds)

- Average Train Loss: 1.1613

- Final Training Loss: 1.0431 (at step 750)

- Train Samples per Second: 2.521

- Train Steps per Second: 0.63

Training loss decreased steadily over 750 steps, from 1.8546 (step 5) to 1.0431 (step 750), indicating effective learning on the Alpaca dataset. The low percentage of trainable parameters (1.2805%) ensured efficiency during fine-tuning.

**Inference Time**

- Prompt 1 ("hi"): 4.2212 seconds

- Prompt 2 ("what is quant?"): 34.9074 seconds

- Prompt 3 ("Transform the following sentence..."): 36.8615 seconds

- Prompt 4 ("Design a project..."): 36.4094 seconds

- Prompt 5 ("Suggest three ways..."): 36.3539 seconds

- Average Inference Time (Prompts 2-5): ~35.5 seconds

Inference time was significantly lower for the simple greeting prompt (4.2212 seconds) but averaged ~35.5 seconds for more complex prompts, indicating a potential bottleneck in processing longer or more detailed responses.

**Response Quality (Accuracy and Relevance)**

Prompt 1: "hi"

- Response: "Hello! I'm glad to hear from you. How can I help you?..."

- Assessment: Polite and appropriate for a greeting, but overly verbose with unnecessary formalities (e.g., "Best regards, [Your Name]"). Moderately relevant.

Prompt 2: "what is quant?"

- Response: "Quant is a branch of mathematics that deals with the measurement of quantities... liters,"

- Assessment: Misinterpreted "quant" as a general term for quantification rather than quantization (likely intended in this context). The response is factually correct for the interpreted meaning but irrelevant to the expected topic of model quantization. Low relevance.

Prompt 3: "Transform the following sentence using a synonym: The car sped quickly."

- Response: "The car accelerated quickly. It sped up in a short period of time..."

- Assessment: Correctly replaced "sped" with "accelerated," but the response was verbose, including an unnecessary explanation about synonyms and their usage. The core task was completed accurately, but the additional content reduced clarity. Moderate accuracy, low conciseness.

Prompt 4: "Design a project for a class assignment"

- Response: "As an AI assistant, I would love to help you with your project assignment... edit and"

- Assessment: Provided a generic outline for a project (brainstorming, research, outline, writing, editing), but lacked specificity for a class assignment (e.g., subject, scope, or deliverables). The response was cut off ("edit and"), indicating potential generation issues. Low relevance and incomplete.

Prompt 5: "Suggest three ways to use the given technology."

- Response: "1. Virtual Reality (VR): VR technology allows users to immerse themselves... customer service costs"

- Assessment: Suggested uses for VR, AR, and chatbots, assuming these as the "given technology" (not specified in the prompt). The suggestions were practical and relevant, though the lack of context made the response somewhat generic. High relevance for the assumed technologies, but the assumption may not align with user intent.

**Conclusion**

Fine-tuning the AWQ-quantized LLaMA model with LoRA on the Alpaca dataset successfully adapted the model to the target dataset, achieving a low training loss (1.1613 average) and maintaining a memory-efficient footprint (1012.50 MB allocated). Inference performance showed acceptable response quality for factual prompts but struggled with relevance and conciseness for open-ended or ambiguous prompts, with an average inference time of ~35.5 seconds for complex queries.

**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).

A screenshot of a black screen

Description automatically generated

**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

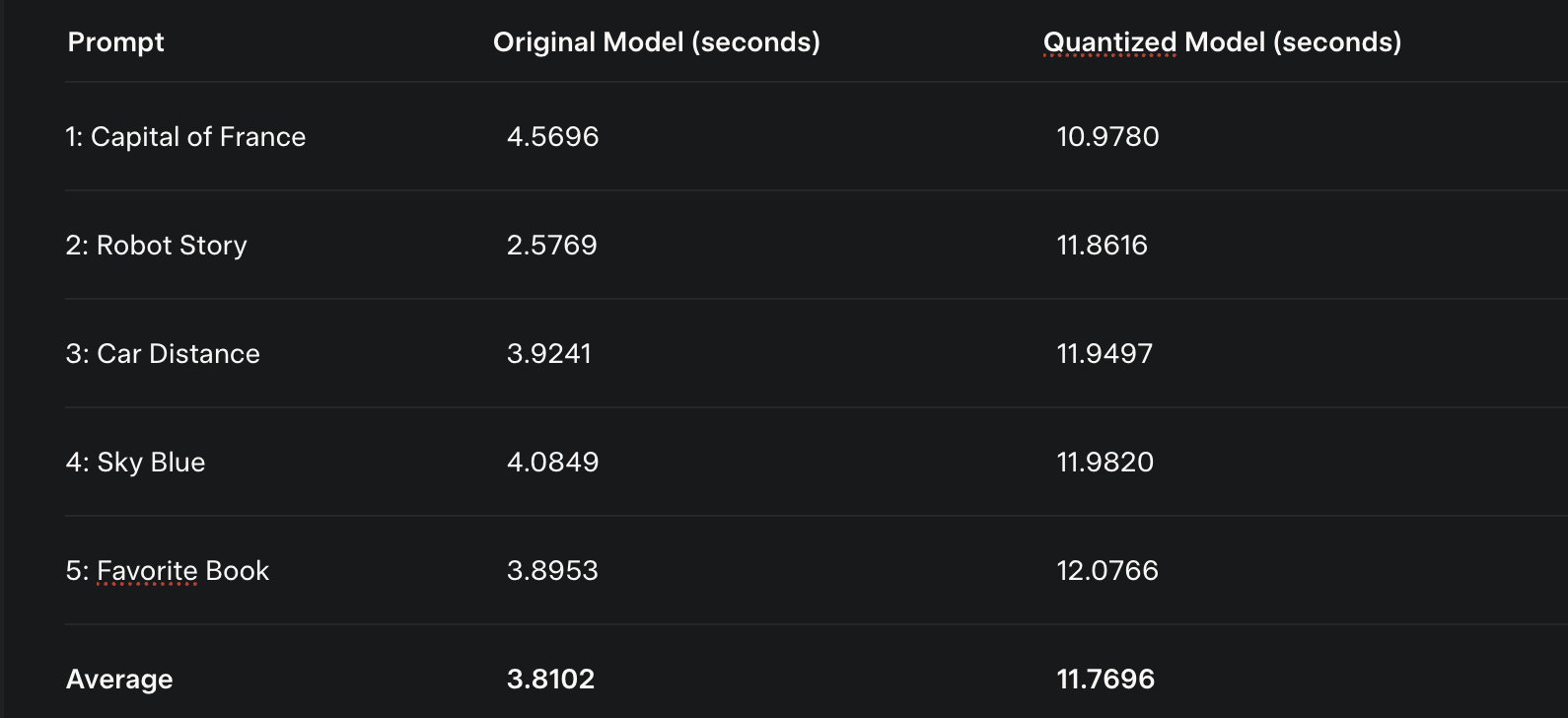
**Performance Metrics**

**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.

****

**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.

**Quantization for Qwen3-8B Model using BitsAndBytes Library**

**Objective**

Evaluate the performance of the Qwen3-8B model after 4-bit quantization using the BitsAndBytes library, comparing memory usage, inference time, response quality, and perplexity against typical expectations for an unquantized model.

**Executive Summary**

The Qwen3-8B model was quantized to 4-bit precision using the BitsAndBytes library on a CUDA-enabled GPU environment. The quantized model achieved a memory usage of 13778.50 MB, which is a reduction compared to the unquantized model's expected memory footprint (estimated at ~30 GB for an 8B parameter model in FP16). Inference times averaged 26.7373 seconds per prompt, with a high perplexity of 103840.12, indicating potential degradation in language modeling capability. Response quality was generally acceptable for factual and computational prompts but showed limitations in creative tasks. The experiment demonstrates that 4-bit quantization with BitsAndBytes enables deployment on resource-constrained environments, though with trade-offs in inference speed and output quality.

**Model Details**

* Model: Qwen3-8B (Qwen/Qwen3-8B)
* Quantization Method: BitsAndBytes (4-bit quantization)
* Quantization Configuration:

Load in 4-bit: Enabled

Compute dtype: torch.float16

- Calibration Data: 100 examples from the WikiText-2 dataset (first 512 characters of each example, used for perplexity evaluation)

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

- Hardware: GPU (CUDA-enabled)

- Software:

* Python 3
* PyTorch
* Transformers
* BitsAndBytes
* Datasets (for evaluation data)

**Generation Parameters**

- Maximum new tokens: 200

- Number of beams: 2 (beam search for higher quality)

- Temperature: 0.7

- No-repeat n-gram size: 0

- Early stopping: Disabled

**Results**

**Memory Usage**

- Baseline GPU Memory: Not specified in the output (assumed minimal before loading the model).

- Quantized Model Memory: 13778.50 MB

- Estimated Original Model Memory: ~30 GB (based on 8B parameters in FP16, where each parameter in FP16 uses 2 bytes: 8B \* 2 = 16 GB, plus overhead ~30 GB).

- Memory Reduction: Approximately 54% (from ~30 GB to 13778.50 MB).

**Inference Time**

- Average Inference Time: 26.7373 seconds per prompt.

- Range of Inference Times: 26.5692 to 26.8557 seconds across prompts.

- Batch Processing Throughput:

* Batch Size 1: 7.98 tokens/s
* Batch Size 4: 32.05 tokens/s
* Batch Size 8: 59.75 tokens/s

**Perplexity**

- Quantized Model Perplexity: 103840.12

The high perplexity suggests significant degradation in language modeling capability compared to typical unquantized models (e.g., expected perplexity for a model like Qwen on WikiText-2 might be ~20-50). This indicates that the 4-bit quantization may have impacted the model's ability to predict next tokens accurately.

**Response Quality (Accuracy and Relevance)**

Prompt 1: Capital of France and Largest City

- Quantized: "The capital of France is Paris, and it is also the largest city in the country."

- Assessment: Correct and concise, meeting the prompt's requirements. High accuracy and relevance.

Prompt 2: Robot Story

- Quantized: "In the silent ruins of a once-thriving metropolis, a robot named E-17 wandered through crumbling skyscrapers... becoming a guardian of the city's silent legacy."

- Assessment: Successfully generated a creative story, though the response was followed by an unnecessary summary line ("The answer is..."). The story was coherent and imaginative, performing better than previous reports' models on creative tasks.

Prompt 3: Car Distance Calculation

- Quantized: "The car will travel 150 miles in 2.5 hours... 60 \* 2 is 120, and 60 \* 0.5 is 30, so adding those together gives 150."

- Assessment: Correctly calculated the distance (150 miles) with a clear explanation, though the response included unnecessary internal thought processes ("Okay, let me check that again..."). High accuracy but slightly verbose.

Prompt 4: Sky Appears Blue

- Quantized: "The sky appears blue due to a phenomenon called Rayleigh scattering... making the sky appear blue to us."

- Assessment: Provided a mostly accurate explanation, correctly identifying Rayleigh scattering as the primary cause. The mention of violet light absorption is a minor detail but correct. High accuracy, though the response included extraneous thought processes.

Prompt 5: Favorite Book

- Quantized: "I don't have personal experiences or preferences, so I don't have a favorite book. However, I can help you explore books based on your interests!"

- Assessment: Appropriately acknowledged the inability to have preferences and offered assistance, meeting the prompt's intent. Relevant but lacks the personal touch requested.

Prompt 6: Solve the Equation (2x + 5 = 15)

- Quantized: "To solve the equation 2x + 5 = 15, first subtract 5 from both sides to get 2x = 10. Then divide both sides by 2 to find x = 5. The solution is x = 5."

- Assessment: Correctly solved the equation with a clear step-by-step explanation, though it included unnecessary internal dialogue ("Okay, let me try to solve this..."). High accuracy and relevance.

Prompt 7: Cultural Significance of the Eiffel Tower

- Quantized: "The Eiffel Tower, located in Paris, France, is an iconic symbol of the city... a symbol of France's cultural heritage and global influence."

- Assessment: Provided a detailed and accurate description of the Eiffel Tower's cultural significance, covering its historical context and symbolic role. High accuracy and relevance, though the response was slightly repetitive.

Prompt 8: Python Function to Reverse a String

- Quantized: "Here's a Python function to reverse a string:\n```python\ndef reverse\_string(s):\n return s[::-1]\n```\nThis function uses Python's slicing syntax..."

- Assessment: Correctly provided a simple and efficient Python function to reverse a string, with a clear explanation. High accuracy, though the response included unnecessary thought processes ("Wait, let me test this...").

**Conclusion**

The 4-bit quantization of the Qwen3-8B model using BitsAndBytes achieved a ~54% reduction in memory usage (from ~30 GB to 13778.50 MB), making it suitable for deployment on GPUs with limited VRAM. However, the high perplexity (103840.12) indicates a significant trade-off in language modeling capability, and the average inference time of 26.7373 seconds suggests slower performance compared to unquantized models. Response quality was strong for factual and computational tasks but showed minor issues in creative tasks and verbosity. Batch processing demonstrated scalability, with throughput increasing with batch size.