Optimizing LLM Inference with Spec Decoding and Quantization

Ryan Huang, Michael Khanzadeh,
Niranjan Sundararajan

Executive Summary

Goal:

- Achieve speed up on Llama3-8B model under computational constraints
- Explore trade off between speed / memory usage and accuracy across approaches

Solution Approach:

- Implement **speculative decoding** using Llama3-1B and Llama3-3B as draft models
- Apply **quantization** techniques
- Apply a combined strategy of **spec decoding + quantization**

Value of Our Solution:

- Demonstrates a meaningful optimization/exploration on Llama3-8B Model

Problem Motivation

LLMs such as Llama3 generate only one token at a time during inference

- Suggests potential area for optimization

Speculative Decoding uses a smaller draft model to propose multiple generated tokens to the main target model

- Allows for speed up through parallelization as main model can now do one forward pass to compute logits for multiple tokens at once
- Retains accuracy since target model verifies proposed tokens and in case of rejection, rolls back and restarts speculative decoding at that point

Problem Motivation

- Modern LLMs are massive (some exceeding 1 trillion parameters)
- They require large amounts of VRAM to load and run efficiently
- As model size grows, inference becomes slower and more expensive
- There's a growing need for techniques that reduce memory usage and improve throughput
- LLMs generate 1 token at a time -> Use spec decoding to have a smaller/faster draft model propose multiple at a time
- Weights and activations can be stored in reduced precision with minimal accuracy loss.

Background Work

- Huggingface repo for speculative decoding: https://huggingface.co/docs/transformers/main/en/generation_strategies
- Speculative decoding paper: https://arxiv.org/pdf/2211.17192
- Possibility of optimizing spec decoding by using draft model logits for optimized rejection - https://arxiv.org/abs/2405.04304
- Huggingface library for quantization (bitsandbytes): https://huggingface.co/docs/transformers/en/quantization/bitsandbytes
- HumanEval - https://arxiv.org/pdf/2107.03374

Technical Challenges

- Lack of Hardware (GPU resources)
- Hard to modify library code for spec decoding, nuances within the codebase that were not clearly documented or tested
- Developing suites for metrics
- Developing pipelines for reproducible experiments
- Dealing with OOM and pivoting the project direction as needed depending on hardware available and memory required
- Other models explored as well (Llama2 and Mistral), but dropped due to sizes of models not having a reasonable difference.

Approach (1/2)

Speculative Decoding revolves around using a lightweight draft model to generate multiple tokens and propose that to the target model so that it can process the logits for multiple tokens in one forward pass

Evaluation metrics:

- Throughput / Latency
- Total # of rollbacks (rollback occurs whenever target model rejects a proposed token from the draft model)
- Acceptance rate

Experimented with:

- Varying # assistant tokens (3, 5, 7)
- Sampling vs greedy
- Confidence threshold (0.0001 and 0.2)
- Combining quantization with speculative decoding (4 bit, 8 bit)

Assistant Confidence Threshold:

- Optimization reducing unnecessary draft/target forward passes by early stopping draft generation
- Varied to explore draft model always proposing k tokens and optimizing by early stopping’

Dataset:

- WikiText2: 100m+ tokens from featured Wikipedia articles

Approach (2/2)

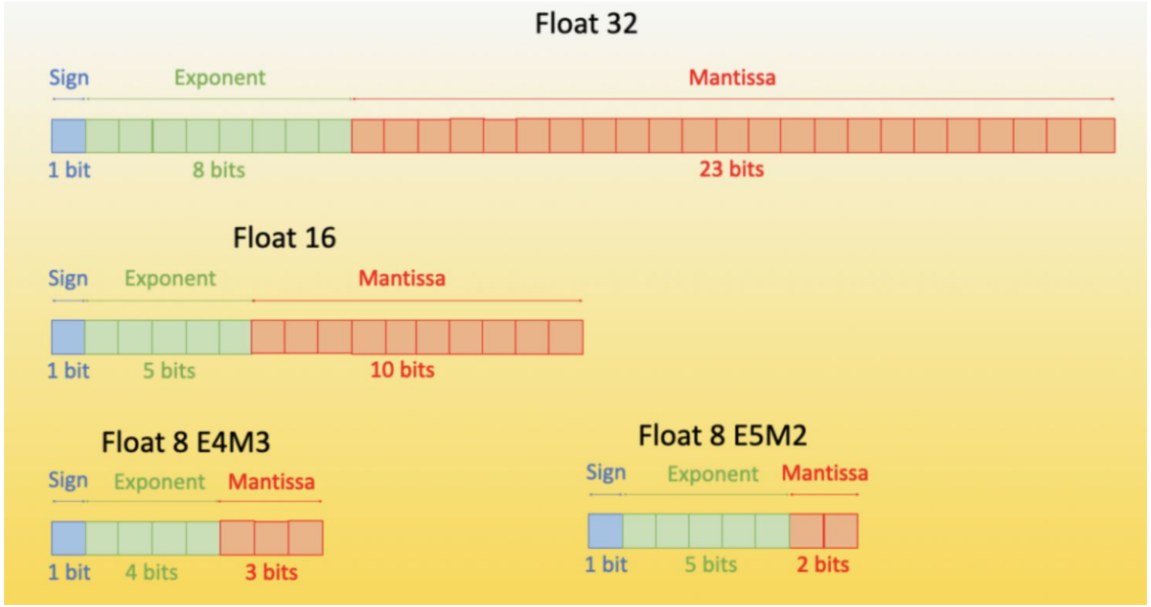
Evaluate the tradeoff between memory and performance using quantization with **bitsandbytes**

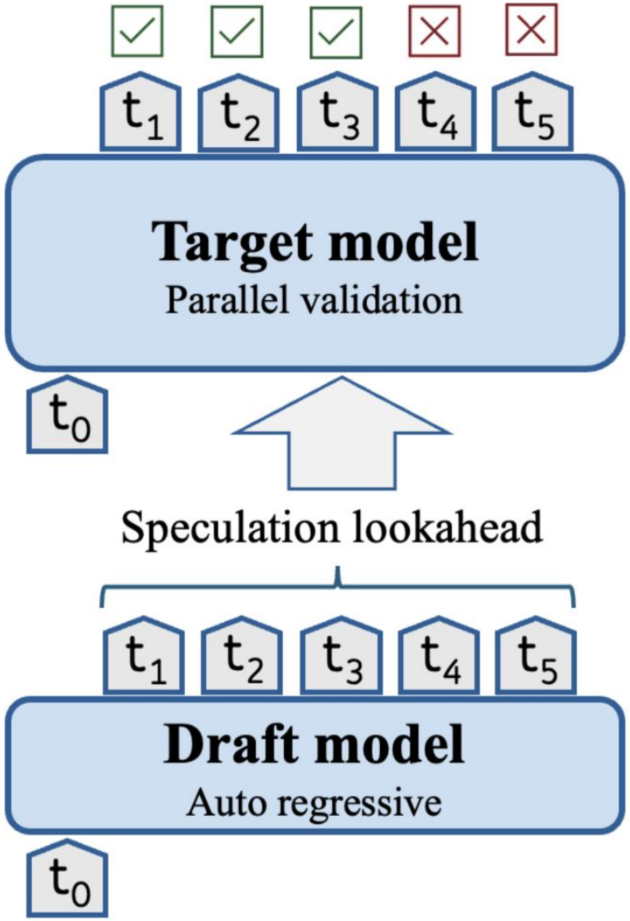
1. Fp16 (Baseline)
2. 8-bit
3. 4-bit (NF4)

Evaluation Metrics:

1. HumanEval Accuracy: Functional correctness on coding tasks
2. Perplexity: Model confidence (Wikitext)
3. Throughput/Latency: Inference speed (Wikitext)
4. GPU Utilization : Hardware/memory efficiency

Solution Diagram/Architecture





Implementation Details

- Github link: https://github.com/nin-ran-jan/HPML_Project
- Used HuggingFace's transformers and datasets libraries
- Implemented wrapper superclasses to track key metrics
  - Total rollbacks and acceptance rate
- Used wikitext-2-raw-v1 test set and took subset only including samples >= 128 tokens, and clipped all prompts to be exactly 128 tokens for consistency
- Quantized to 4 bit and 8 bit using bitsandbytes
- All workflows mapped to **3 bash scripts** for ease of reproducibility.

Experiment Design Flow

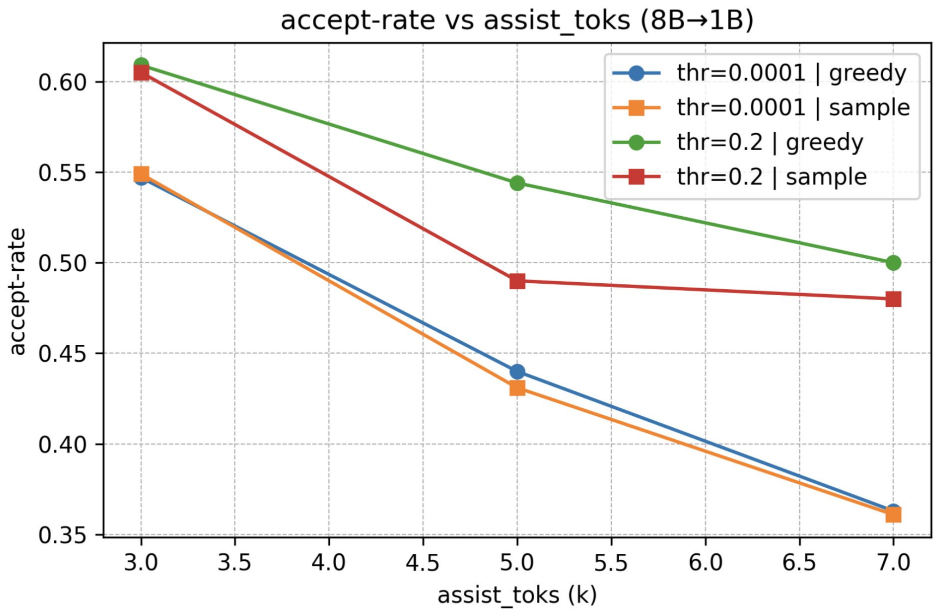
Speculative Decoding Focused Experimentation:

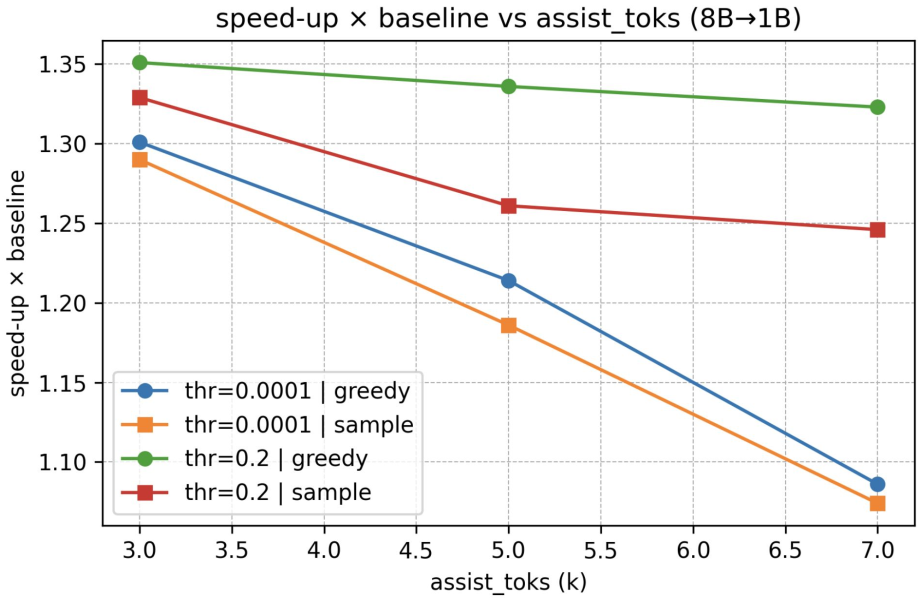
- Llama3 8B-1B and Llama3 8B-3B speculative decoding - varied on # assistant tokens, sampling and confidence threshold
- Quantize draft + target model runs at different precisions

Quantization Focused Experimentation:

- Baseline Llama3 8B
- Llama3 8B quantized - 4 bit, 8 bit

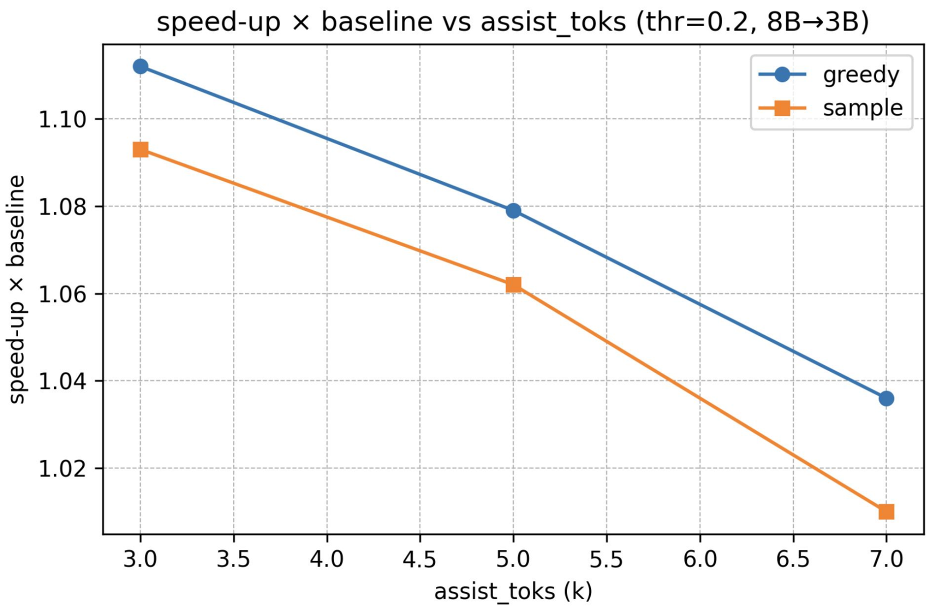
Experiment Evaluation: Speculative Decoding 8B-1B

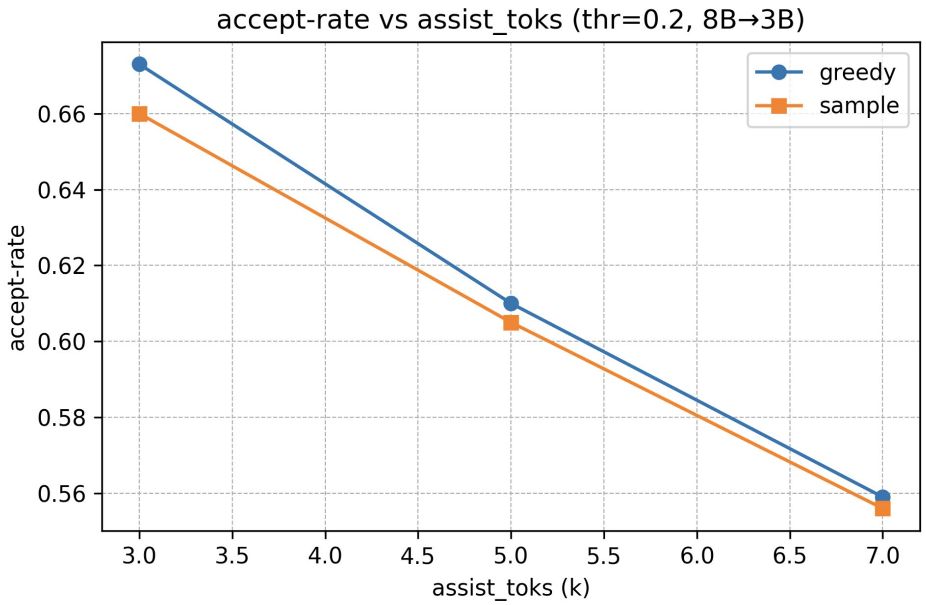




- Best performance was with thr=0.2, greedy, assist_tokens=3
- More rollbacks -> lower acceptance rate -> less speedup

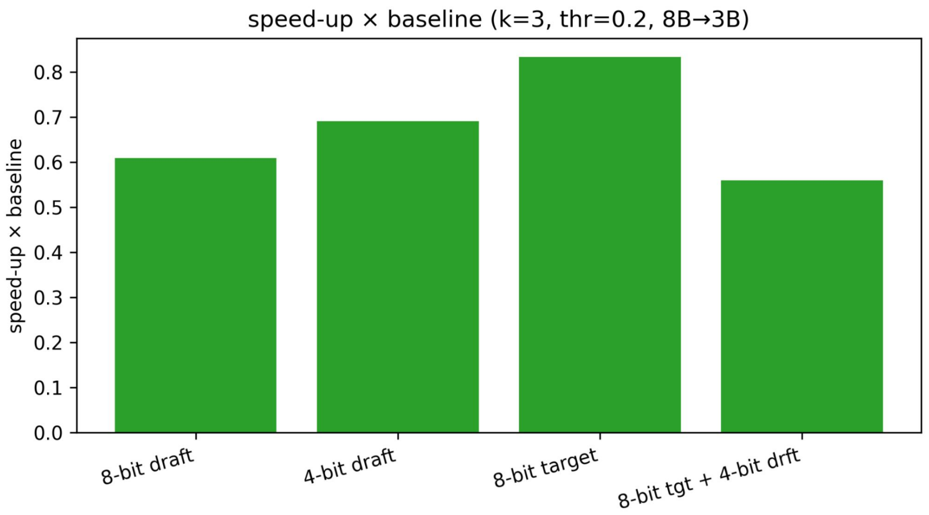
Experiment Evaluation: Speculative Decoding 8B-3B

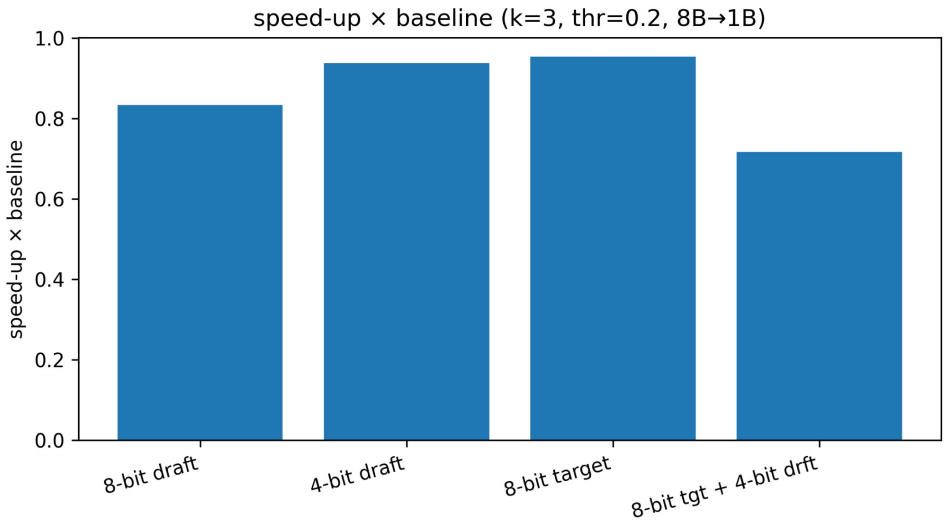




- Overall speed up was worse compared to using 1B as the draft model
- 3B model is computationally heavier than 1B taking longer per forward pass

Experiment Evaluation: Quantization for 8B-1B and 8B-3B

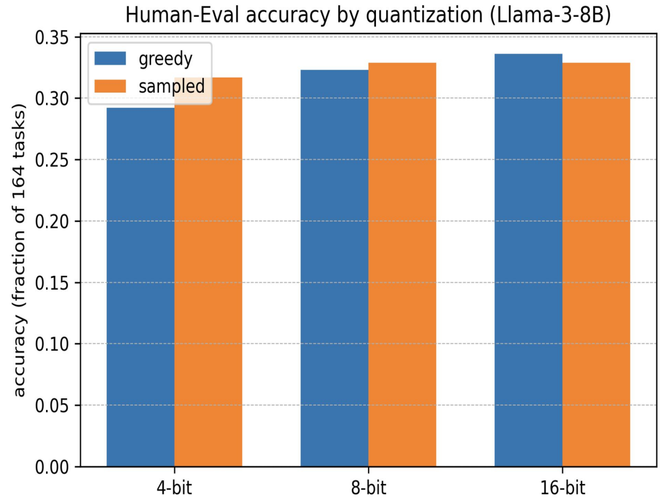




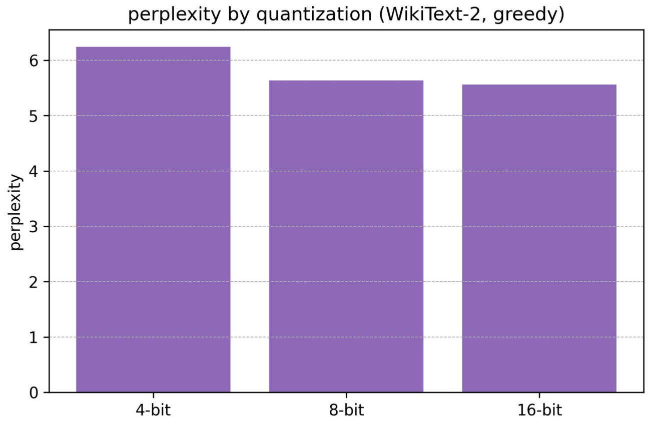
- Overall speedup when combined with quantization was worse
- Speedup gained from quantizing to lower precision likely not enough to account for lowered output quality leading to more rollbacks

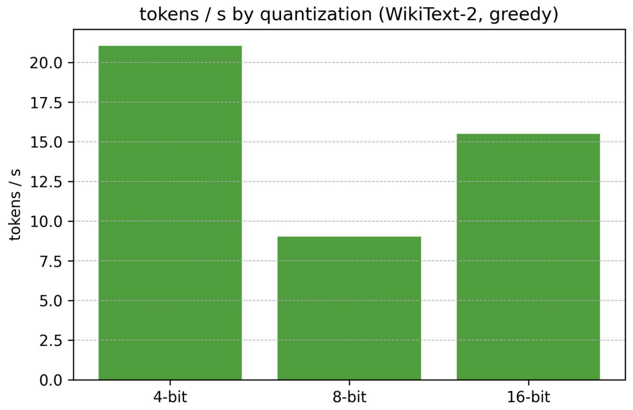
Experiment Evaluation: Quantization on HumanEval

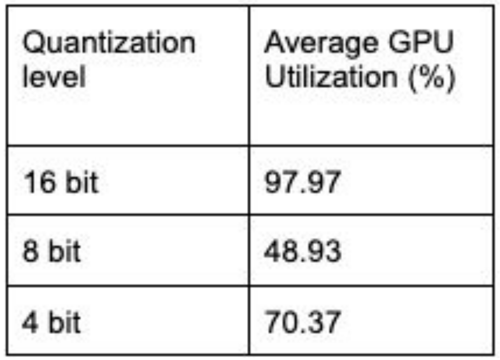
- 8-bit accuracy got same accuracy as 16 bit in sampled and -3.9% for greedy
- 4-bit compared to 16-bit dropped -3.65% for sampled and -13% for greedy
- Sampling for same bit yielded +8% 4 bit, +1.8% 8 bit, -2% 16 bit, and compared to 16-bit counter part had lower acc drop
- 8-bit gives strong balance of reduced memory and accuracy
- 4-bit is strong where memory is tight and slight accuracy loss is acceptable



Experiment Evaluation: Quantization Eval on Wikitext







Quantization level	Average GPU Utilization (%)
16 bit	97.97
8 bit	48.93
4 bit	70.37

- 4-bit achieves the highest throughput (21.0 tok/s) and lowest latency (47.5 ms/token) (a ~35% speedup over 16-bit)
- 8-bit performs worst in speed (9.0 tok/s) and GPU utilization (49%) (likely due to suboptimal kernel support on L4 GPUs)
- Perplexity increases slightly as bit precision decreases tradeoff for compression
  - 16-bit: 5.56
  - 8-bit: 5.64 (+1.4%)
  - 4-bit: 6.24 (+12.3%)
- Conclusion: 4-bit offers high performance benefits, but at a small cost in model confidence. 8-bit for inference speed seems to be suboptimal for L4 GPU but strong perplexity; 16-bit remains best for quality with resources permit.

Conclusions

- Speculative decoding was able to get maximum speedup of about 1.4x for Llama3 8B using 1B as the draft model on confidence threshold = 0.2
    - Strong for inference speedup and assistant threshold heuristic works well
- A combination of quantization and speculative decoding did not produce strong results
    - Reduced precision -> increased rollbacks -> overall slower inference time
    - With more VRAM, can explore a larger target model (~70B) with 8B draft, which may yield better results for spec decoding with quantization
- Quantization (4-bit) delivered up to 35% speedup over fp16, with only ~4 (accuracy) and ~12% (perplexity) degradation, making it ideal for memory- and speed-constrained scenarios
- 8-bit quantization preserved accuracy very well, but had very slow inference on L4 GPUs, likely due to lack of kernel/hardware optimization.

Contributions

- The team would have regular meetings 1-2x a week, oftentimes in the meetings would synchronously work together, other times if applicable we would asynchronously split up the tasks
- Overall for all major decisions all team member were involved, and even for parts that one didn't directly code they were often involved in discussions regarding direction or how to handle unexpected blockers/situations
- **Ryan:** Spec decoding optimization for key evaluation metrics like rollback and accepted token percentage, hyperparameter tuning
- **Michael:** HumanEval, quantization evaluation, and spec decoding optimization for confidence threshold
- **Niranjan:** Baselines for spec decoding and quantization, hardware and system configurations across hardwares, profiling