


# Dynamic Precision Scheduling (DPS) for Efficient LLM Inference



Leonid Alexeyev, Jonathan Huang,  
Sujin Lee, Darin Mao, Devam Shrivastava



# Table of Contents

---

1

Motivations

2

System Architecture

3

Dataset

4

Experimental Results

5

Project Summary

6

Q&A

# Motivation

---

- **LLMs are powerful but computationally expensive**
  - high inference latency and memory cost
- **Quantization** reduces the precision of the floating point tensors
  - theoretically, this reduces the computational cost
- Generally, quantization is performed on entire models
- Quantize some layers only? Yes, shown in LSAQ paper
  - can we automatically optimize before inference time?

We want to build a system to autonomously determine which layers to quantize, to achieve a balance of performance and output quality

- The search space is exponential in size (quantization levels  $^{\wedge}$  layers), so our system needs to be smarter

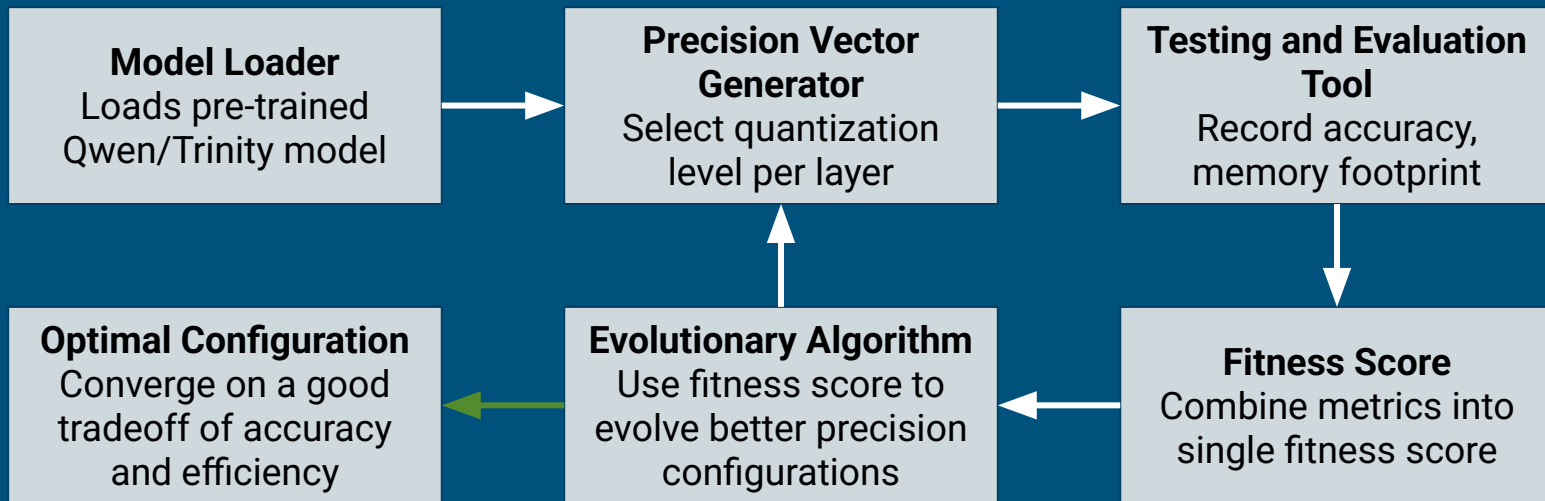


# System Architecture



# System Architecture

---



# System Architecture

---

## 1. Model Management Layer:

- Loads the base model, applies per-layer bit-width settings, runs inference, and collects evaluation metrics.

## 2. Precision Vector System

- Creates and manages per-layer precision vectors; for example, [8, 4, 4, 16, 8, 8, ...]

## 3. Evaluation Pipeline

- Run the quantized model on dataset prompts and computes accuracy, latency, and memory footprint.

# System Architecture

---

## 4. **Fitness Module**

- Convert evaluation metrics into a single numerical fitness score for optimization.

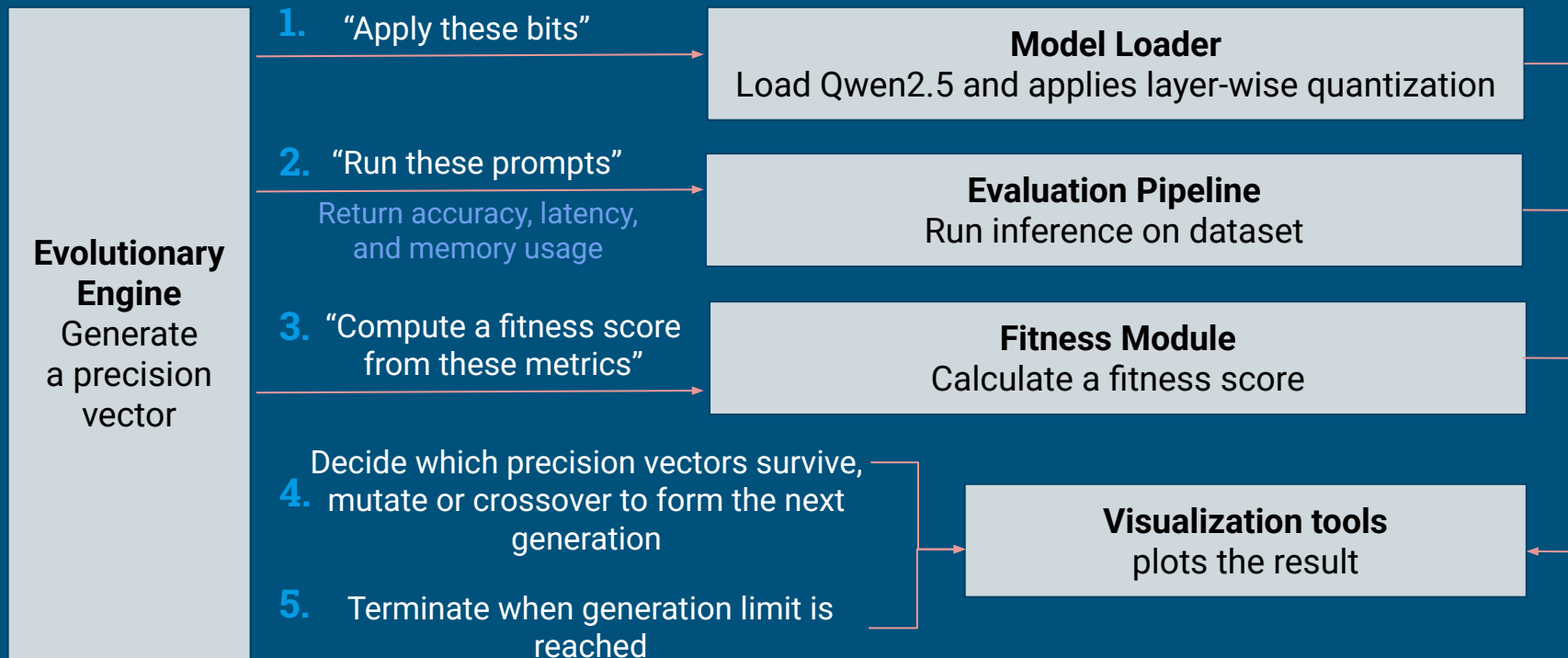
## 5. **Evolutionary Search Engine**

- Search for Pareto-optimal per-layer precision configurations through evolutionary optimization

## 6. **Visualization**

- Produce plots for Pareto frontier (accuracy vs memory), Fitness progression, Bit-width heatmaps and 3D trade-off

# System Architecture





# System Architecture

---

## Our components

- Evolutionary search algorithm
- Precision vector system
- Custom quantization wrapper around HF models
- Dataset evaluation and timing pipeline
- LLM-based autograder
- Fitness scoring module
- Pareto frontier visualization
- Precision heatmap visualization

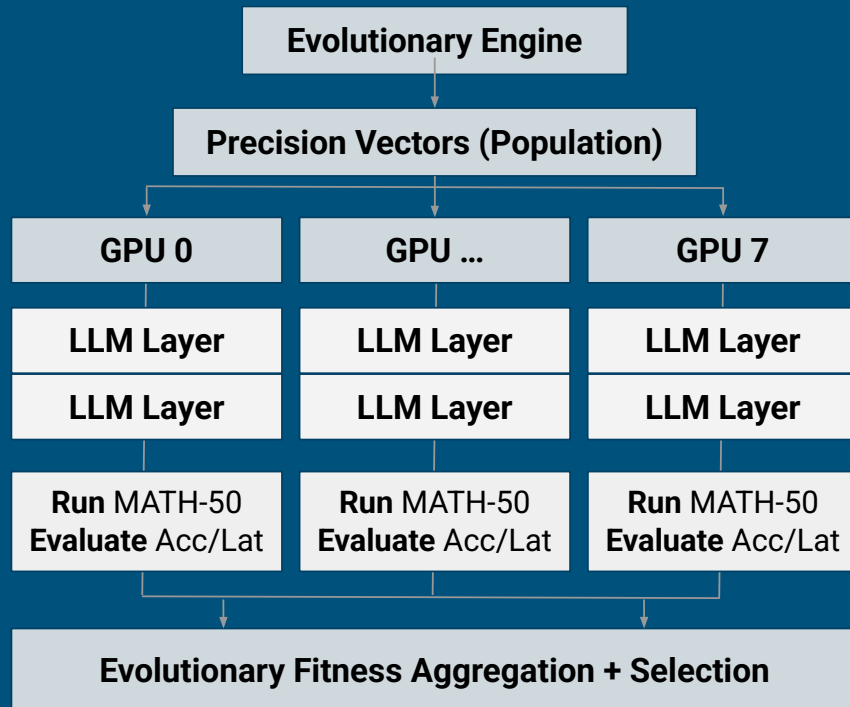
## Open-Source Components

- HuggingFace Transformers (Model loading and inference)
- BitsAndBytes (4-bit, 8-bit quantization backend)
- Qwen2.5-7B-Instruct (base LLM)
- PyTorch (GPU inference)
- Matplotlib/Seaborn

# System Architecture

Experiments were run on 8x H200 GPUs rented from Vast.ai

- 141GB VRAM per GPU
- 428.2 TFLOPS total performance
- Python multiprocessing library used for coordinating model runs and collecting results





# Dataset



# Dataset

- MATH-500: 500 problems across algebra, geometry, number theory, calculus

- All 500 samples test, no need for train/validation since we are using this for evaluation
- Columns are formatted in LaTeX

problem	solution	answer	subject	level
string · lengths	string · lengths	string · lengths	string · classes	int64
20	45	1	7 values	1
Convert the point $(0,3)$ in rectangular coordinates to polar coordinates. Enter...	We have that $r = \sqrt{0^2 + 3^2} = 3$ . Also, if we draw the line connecting the origin and...	$\left( 3, \frac{\pi}{2} \right)$	Precalculus	2
Define $\lfloor p = \sum_{k=1}^{\infty} \frac{\lfloor \frac{1}{k^2} \rfloor}{k^2} \rfloor$ and $q = \sum_{k=1}^{\infty} \frac{\lfloor \frac{1}{k^2} \rfloor}{k^2}$ . Express you...	We count the number of times $\frac{1}{n^3}$ appears in the sum $\sum_{j=1}^{\infty} \frac{1}{j^3}$ .	$p - q$	Intermediate Algebra	5
If $f(x) = \frac{3x-2}{x-2}$ , what is the value of $f(-2) + f(-1) + f(0)$ ? Express you...	$f(-2)+f(-1)+f(0)=\frac{3(-2)-2}{-2-2}+\frac{3(-1)-2}{-1-2}+\frac{3(0)-2}{0-2}=-\frac{8}{4}-\frac{5}{3}+\frac{2}{2}=-2-\frac{5}{3}+1=-\frac{5}{3}$	$-\frac{5}{3}$	Algebra	3
How many positive whole-number divisors does 196 have?	First prime factorize $196=2^2 \cdot 7^2$ . The prime factorization of any divisor of 196...	9	Number Theory	3
The results of a cross-country team's training run are graphed below. Which...	Evelyn covered more distance in less time than Briana, Debra and Angela, so her average spee...	$\text{Evelyn}$	Algebra	2
A regular hexagon can be divided into six equilateral triangles. If the perimeter o...	The side length of the hexagon is equal to the side length of one of the equilateral...	42	Prealgebra	2
What is the smallest positive perfect cube that can be written as the sum of three...	The sum of three consecutive integers takes the form $(k-1)+(k)+(k+1)=3k$ and hence is a...	27	Number Theory	3
The set of points $(x,y,z)$ that satisfy $\begin{cases} 2x - 3y = -z \\ x + y + z = 1 \end{cases}$ is a line. The set of poin...	For the first line, let $t = 2x = 3y = -z$ . Then $\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{5} \begin{pmatrix} t \\ t \\ -t \end{pmatrix}$ .	$90^\circ$	Precalculus	4
What is the distance, in units, between the points $(2, -6)$ and $(-4, 3)$ ? Express...	We use the distance formula: $\sqrt{(2 - (-4))^2 + ((-6) - 3)^2} = \sqrt{6^2 + (-9)^2} = \sqrt{36 + 81} = \sqrt{117}$ .	$3\sqrt{13}$	Algebra	3
The expression $2 \cdot 3 \cdot 4 \cdot 5 \cdot 1$ is equal to 121, since multiplication is...	By the associative property of multiplication, it doesn't help to insert parentheses that...	4	Prealgebra	5
What is the least positive integer multiple of 30 that can be written with only the...	Let $M$ be the least positive multiple of 30 that can be written with only the digits 0 an...	2220	Number Theory	3

- "MATH-50": A subset created by our team for faster evaluation, due to resource constraints

# Dataset Example 1: test/precalculus/807.json

## Problem

Convert the point  $(0, 3)$  in rectangular coordinates to polar coordinates. Enter your answer in the form  $(r, \theta)$ , where  $r > 0$  and  $0 \leq \theta < 2\pi$ .

## Solution

We have that  $r = \sqrt{0^2 + 3^2} = 3$ . Also, if we draw the line connecting the origin and  $(0, 3)$ , this line makes an angle of  $\frac{\pi}{2}$  with the positive  $x$ -axis.

Therefore, the polar coordinates are  $\boxed{\left(3, \frac{\pi}{2}\right)}$ .

Answer:

$(3, \pi/2)$

Subject:

Precalculus

Level:

2

# Dataset Example 2: test/number\_theory/515.json

---

## Problem

What is the smallest positive perfect cube that can be written as the sum of three consecutive integers?

## Solution

The sum of three consecutive integers takes the form  $(k-1) + (k) + (k+1) = 3k$  and hence is a multiple of 3. Conversely, if a number  $n$  is a multiple of 3, then  $n/3 - 1$ ,  $n/3$ , and  $n/3 + 1$  are three consecutive integers that sum to give  $n$ . Therefore, a number is a sum of three consecutive integers if and only if it is a multiple of 3. The smallest positive perfect cube that is a multiple of 3 is  $3^3 = \boxed{27}$ .

Answer:

27

Subject:

Number Theory

Level:

3

# Dataset Example 3: test/intermediate\_algebra/1197.json

## Problem

Let  $p(x)$  be a polynomial of degree 5 such that

$$p(n) = \frac{n}{n^2 - 1}$$

for  $n = 2, 3, 4, \dots, 7$ . Find  $p(8)$ .

## Solution

Let  $q(x) = (x^2 - 1)p(x) - x$ . Then  $q(x)$  has degree 7, and  $q(n) = 0$  for  $n = 2, 3, 4, \dots, 7$ , so

$$q(x) = (ax + b)(x - 2)(x - 3) \cdots (x - 7)$$

for some constants  $a$  and  $b$ .

We know that  $q(1) = (1^2 - 1)p(1) - 1 = -1$ . Setting  $x = 1$  in the equation above, we get

$$q(1) = 720(a + b),$$

so  $a + b = -\frac{1}{720}$ .

We also know that  $q(-1) = ((-1)^2 - 1)p(-1) + 1 = 1$ . Setting  $x = -1$  in the equation above, we get

$$q(-1) = 20160(-a + b),$$

so  $-a + b = \frac{1}{20160}$ . Solving for  $a$  and  $b$ , we find  $a = -\frac{29}{40320}$  and  $b = -\frac{3}{4480}$ . Hence,

$$\begin{aligned} q(x) &= \left(-\frac{29}{40320}x - \frac{3}{4480}\right)(x - 2)(x - 3) \cdots (x - 7) \\ &= -\frac{(29x + 27)(x - 2)(x - 3) \cdots (x - 7)}{40320}. \end{aligned}$$

In particular,

$$q(8) = -\frac{(29 \cdot 8 + 27)(6)(5) \cdots (1)}{40320} = -\frac{37}{8},$$

so

$$p(8) = \frac{q(8) + 8}{8^2 - 1} = \boxed{\frac{3}{56}}.$$

Answer:

3/56

Subject:

Intermediate Algebra

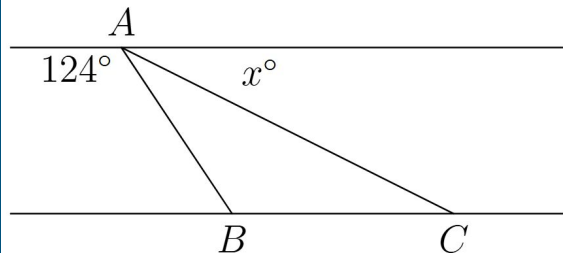
Level:

5

# Dataset Example 4: test/geometry/434.json

## Problem

$\overline{BC}$  is parallel to the segment through  $A$ , and  $AB = BC$ . What is the number of degrees represented by  $x$ ?



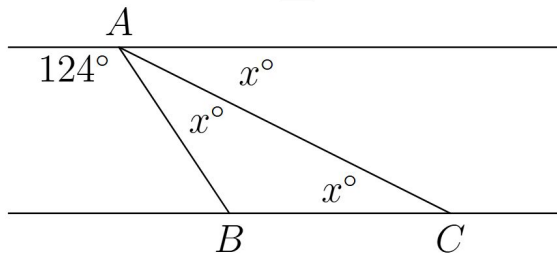
## Solution

Angle  $\angle BCA$  and the angle we're trying to measure are alternate interior angles, so they are congruent. Thus,  $\angle BCA = x^\circ$ . Since  $AB = BC$ , we know that  $\triangle ABC$  is isosceles with equal angles at  $C$  and  $A$ . Therefore,  $\angle BAC = x^\circ$ .

The sum of the three angles at  $A$  is  $180^\circ$ , since they form a straight angle. Therefore,

$$124 + x + x = 180,$$

which we can solve to obtain  $x = \boxed{28}$ .



Answer:

28

Subject:  
Geometry

Level:

1



## Dataset Example 5: test/counting\_and\_probability/134.json

---

### Problem

In how many ways can 8 people sit around a round table if 3 of the people – Pierre, Rosa, and Thomas – all want to sit together? (Two seatings are considered the same if one is a rotation of the other.)

### Solution

First choose three consecutive seats for Pierre, Rosa, and Thomas. It doesn't matter which three consecutive seats that we choose, since any three such seats can be rotated to any other such seats. Once the three seats are chosen, there are  $3!$  ways to seat the three friends there. The other five seats are for the other five people, so there are  $5!$  ways to seat them there. The answer is  $3! \times 5! = \boxed{720}$ .

Answer:

720

Subject:  
Counting and  
Probability

Level:

4



# Experimental Results



# Experimental Setup

---

- Environment: [Vast.ai](#), Pace/ICE
- GPU: 8x H200 ([Vast.ai](#)), 2x H200 (Pace/ICE)
- Model: Qwen2.5-7B-Instruct, Trinity-Mini, Mistral-7B
- Datasets: MATH-50
- Quantization Levels: 4, 8, 16 bits
- Metrics: Accuracy, inference latency, VRAM usage, loss

# Experiment Screenshots (All models)

generation progress

```
AutoLLMTuner [SSH: llm-big]
nanpath: can't set the locale; make sure $LC_* and $LANG are correct
Activate onnxruntime with environment set to /usr/bin/...
(main) root@20327961:~/workspace$ deactivate
* cd root@20327961:~/workspace$ cd AutoLLMTuner/src/
* root@20327961:~/workspace$ python3 main.py --run python --evolutionary main_search --config ./configs/qa2.5_run_config.yaml --run-id qa2.5 --seq-len 4096
'torch_dtype' is deprecated; use 'dtype' instead
Loading checkpoint shards: 100%
Generations: 0%
[INFO] [Epoch 0] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 0] Archival: 1 | best_loss=0.0709
Generations: 10%
[INFO] [Epoch 1] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 1] Archival: 2 | best_loss=0.0709
Generations: 20%
[INFO] [Epoch 2] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 2] Archival: 3 | best_loss=0.0709
Generations: 30%
[INFO] [Epoch 3] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 3] Archival: 4 | best_loss=0.0709
Generations: 40%
[INFO] [Epoch 4] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 4] Archival: 5 | best_loss=0.0709
Generations: 50%
[INFO] [Epoch 5] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 5] Archival: 6 | best_loss=0.0709
Generations: 60%
[INFO] [Epoch 6] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 6] Archival: 7 | best_loss=0.0709
Generations: 70%
[INFO] [Epoch 7] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 7] Archival: 8 | best_loss=0.0709
Generations: 80%
[INFO] [Epoch 8] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 8] Archival: 9 | best_loss=0.0709
Generations: 90%
[INFO] [Epoch 9] Starting evaluation phase
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
Loading checkpoint shards: 100%
[INFO] [Epoch 9] Archival: 10 | best_loss=0.0709
Generations: 100%
```

GPU	CI	PID	Type	Process name	GPU Memory Usage		
0	NVIDIA H100	179W / 70M	On	00000000-4C-00-0 Off	33%	Default	Disabled
1	N/A	35C	PO	42611PUB / 14277...	20%	Default	Disabled
2	NVIDIA H100	169W / 70M	On	00000000-5D-00-0 Off	28%	Default	Disabled
3	N/A	42C	PO	42139PUB / 14277...	26%	Default	Disabled
4	NVIDIA H100	169W / 70M	On	00000000-5E-00-0 Off	26%	Default	Disabled
5	N/A	43C	PO	42139PUB / 14277...	26%	Default	Disabled
6	NVIDIA H100	179W / 70M	On	00000000-5F-00-0 Off	33%	Default	Disabled
7	N/A	37C	PO	41863PUB / 14277...	32%	Default	Disabled
8	NVIDIA H100	169W / 70M	On	00000000-60-00-0 Off	27%	Default	Disabled
9	N/A	46C	PO	42139PUB / 14277...	26%	Default	Disabled
10	NVIDIA H100	169W / 70M	On	00000000-61-00-0 Off	28%	Default	Disabled
11	N/A	35C	PO	42139PUB / 14277...	20%	Default	Disabled
12	NVIDIA H100	169W / 70M	On	00000000-62-00-0 Off	28%	Default	Disabled
13	N/A	43C	PO	42139PUB / 14277...	26%	Default	Disabled
14	NVIDIA H100	179W / 70M	On	00000000-63-00-0 Off	33%	Default	Disabled
15	N/A	37C	PO	41863PUB / 14277...	32%	Default	Disabled
16	NVIDIA H100	169W / 70M	On	00000000-64-00-0 Off	27%	Default	Disabled
17	N/A	46C	PO	42139PUB / 14277...	26%	Default	Disabled
18	NVIDIA H100	169W / 70M	On	00000000-65-00-0 Off	28%	Default	Disabled
19	N/A	35C	PO	42139PUB / 14277...	20%	Default	Disabled

status of 8 GPUs

# Experimental Result #1

- test/precalculus/807.json
- answer is  $(3, \pi/2)$
- fully int8 quantized model produces the correct answer

To convert the point from rectangular (Cartesian) coordinates  $(x, y) = (0, 3)$  to polar coordinates  $(r, \theta)$ , we use the following formulas:

1. The distance from the origin (radius),  $r$ , is given by:

$$r = \sqrt{x^2 + y^2}$$

2. The angle  $\theta$  is given by:

$$\theta = \arctan\left(\frac{y}{x}\right)$$

However, we need to adjust this value based on the quadrant of the point.

Let's calculate these values for our point  $(0, 3)$ .

**Step 1: Calculate  $r$**

$$r = \sqrt{0^2 + 3^2} = \sqrt{9} = 3$$

**Step 2: Calculate  $\theta$**

For the point  $(0, 3)$ , since  $x = 0$  and  $y > 0$ , the point lies on the positive y-axis. Therefore, the angle  $\theta$  is  $\frac{\pi}{2}$  radians (or 90 degrees).

Thus, the polar coordinates for the point  $(0, 3)$  are:

$$(r, \theta) = (3, \frac{\pi}{2})$$

Final Answer:  $(3, \pi/2)$

# Experimental Result #2

- test/intermediate\_algebra/1994.json
- answer is  $p - q$
- fully int8 quantized model produces the correct answer

$$\sum_{n=2}^{\infty} \sum_{j=1}^{n-1} \frac{1}{n^3}$$

The inner sum is simply the sum of  $\frac{1}{n^3}$  taken  $n - 1$  times:

$$\sum_{j=1}^{n-1} \frac{1}{n^3} = (n-1) \cdot \frac{1}{n^3} = \frac{n-1}{n^3}.$$

Therefore, the sum  $S$  becomes:

$$S = \sum_{n=2}^{\infty} \frac{n-1}{n^3}.$$

We can split the fraction  $\frac{n-1}{n^3}$  into two simpler fractions:

$$\frac{n-1}{n^3} = \frac{n}{n^3} - \frac{1}{n^3} = \frac{1}{n^2} - \frac{1}{n^3}.$$

So, the sum  $S$  can be written as:

$$S = \sum_{n=2}^{\infty} \left( \frac{1}{n^2} - \frac{1}{n^3} \right).$$

We can separate this into two sums:

$$S = \sum_{n=2}^{\infty} \frac{1}{n^2} - \sum_{n=2}^{\infty} \frac{1}{n^3}.$$

Notice that  $\sum_{n=2}^{\infty} \frac{1}{n^2}$  is just the series for  $p$  starting from  $n = 2$ , which is  $p - 1$ . Similarly,  $\sum_{n=2}^{\infty} \frac{1}{n^3}$  is just the series for  $q$  starting from  $n = 2$ , which is  $q - 1$ .

Thus, we have:

$$S = (p - 1) - (q - 1) = p - q.$$

Therefore, the sum  $\sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \frac{1}{(j+k)^3}$  in terms of  $p$  and  $q$  is:

$$p - q.$$

# Experimental Result #3

- test/algebra/1072.json
- answer is 243/625
- fully int8 quantized model doesn't produce an answer
- half quantized model produces the wrong answer
- fully unquantized model produces the right answer

Now, calculate  $\left(\frac{3}{5}\right)^7$ :

$$\left(\frac{3}{5}\right)^7 = \frac{3^7}{5^7} = \frac{2187}{78125}$$

Substitute this back into the expression for  $a_8$ :

$$a_8 = \frac{125}{9} \cdot \frac{2187}{78125} = \frac{125 \cdot 2187}{9 \cdot 78125}$$

Simplify the numerator and the denominator:

$$125 \cdot 2187 = 273375$$

$$9 \cdot 78125 = 703125$$

Thus,

$$a_8 = \frac{273375}{703125}$$

Simplify the fraction by finding the greatest common divisor (GCD) of 273375 and 703125. The GCD is 375:

$$\frac{273375 \div 375}{703125 \div 375} = \frac{729}{1875}$$

Further simplify by dividing both the numerator and the denominator by their GCD, which is 27:

$$\frac{729 \div 27}{1875 \div 27} = \frac{27}{69.4444} \approx \frac{27}{69.4444} = \frac{27}{69.4444} = \frac{27}{69.4444} = \frac{27}{69.4444}$$

Finally, the simplified form is:

$$\frac{27}{69.4444} = \frac{27}{69.4444} = \frac{27}{69.4444} = \frac{27}{69.4444}$$

Therefore, the eighth term of the sequence is:

$$\frac{27}{69.4444}$$

# Model Selection

---

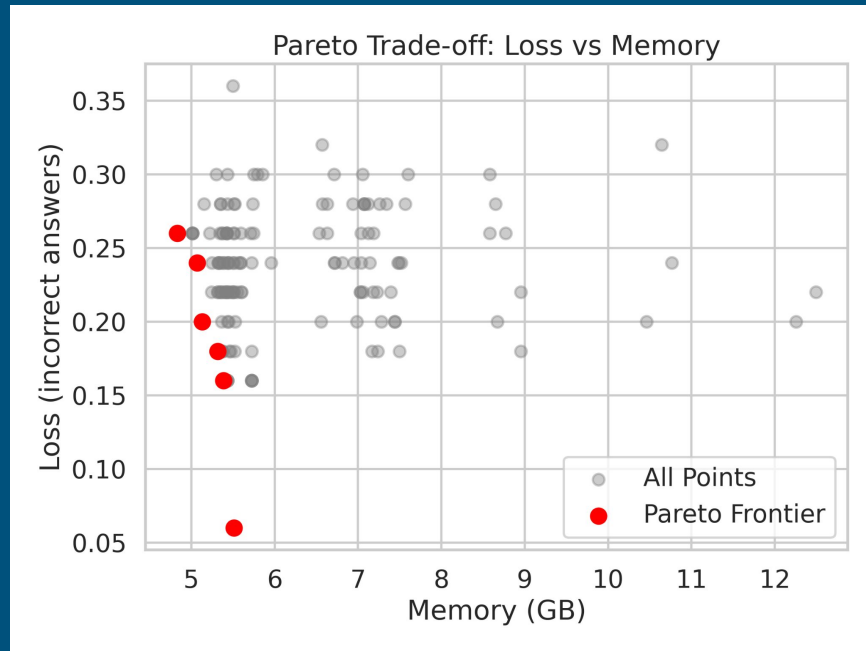
These preliminary experimental results show that the evaluated model is a good fit for our project

- The VRAM consumption is low enough to load multiple times on one GPU
  - This is necessary due to how our system produces quantized models
- Different quantization levels produce measurably different accuracy
  - Some models, though around the same size, produce extremely poor accuracy
  - These models would not show any meaningful results



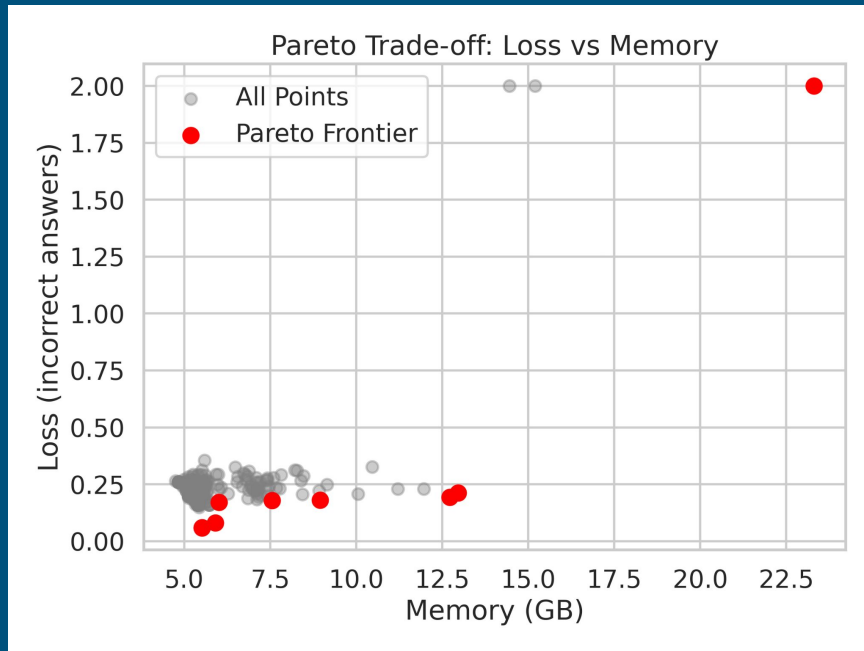
# Measurement Result #1 (QWEN): Loss vs. Memory

- Total of 160 models were tested
- Steep downward slope on the graph
  - Accuracy increases with very slight increases in memory
  - Pareto frontier lies entirely on low-memory side
- Higher memory budgets do not guarantee better accuracy



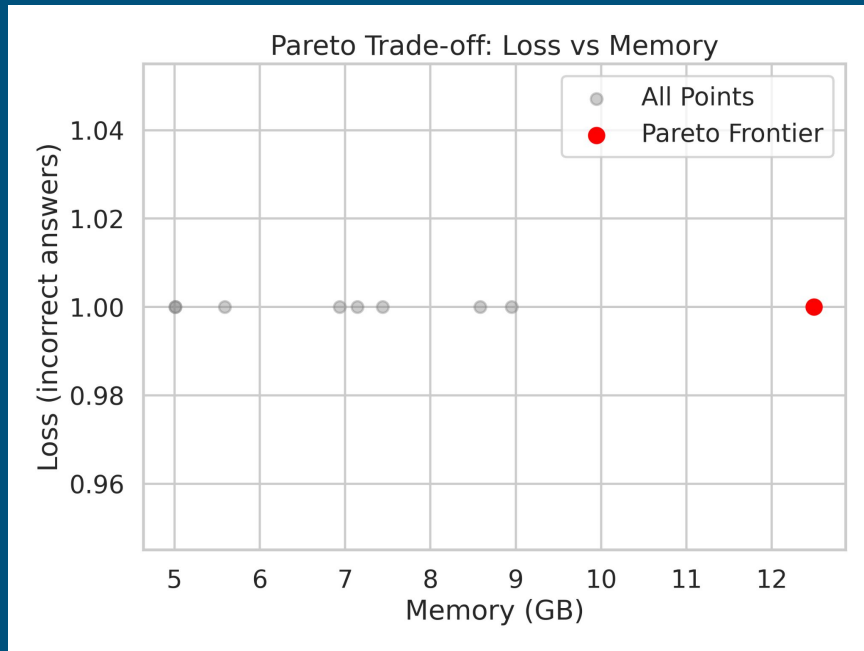
# Measurement Result #1 (Trinity): Loss vs. Memory

- Steep horizontal slope on the graph
  - Accuracy increases with very slight increases in memory
  - Pareto frontier lies entirely on low-loss side
- Trinity: Built for math purposes, causing conflict with dataset
  - But still evident decrease in loss, correlating to better/more accurate responses



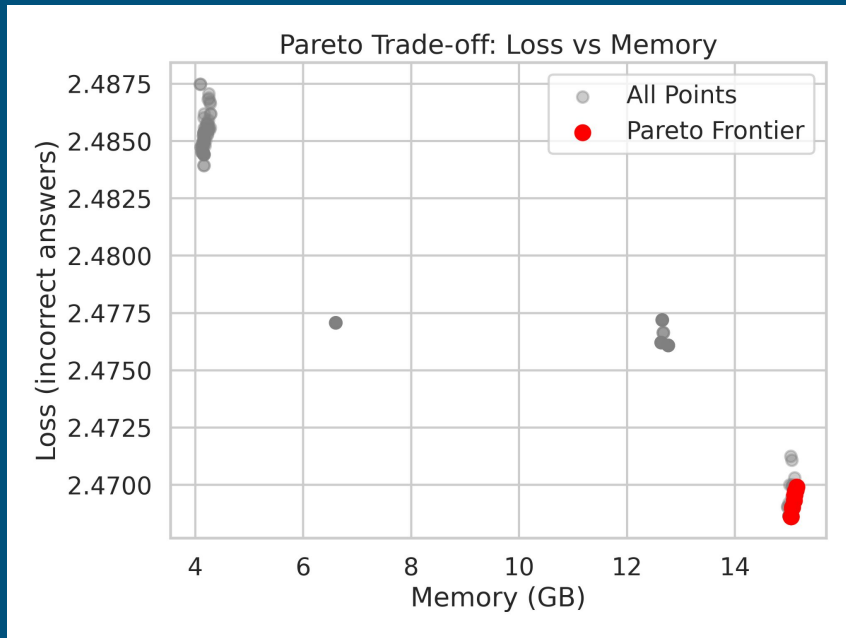
# Measurement Result #1 (Orchestrator) : Loss vs. Memory

- Due to some constraints, the evolutionary search was terminated early
- Loss is not good, regardless of quantization
- Within the explored range, loss appears largely insensitive to memory, suggesting that performance may be more constrained by model
- This makes sense, as math is not what this model is designed for



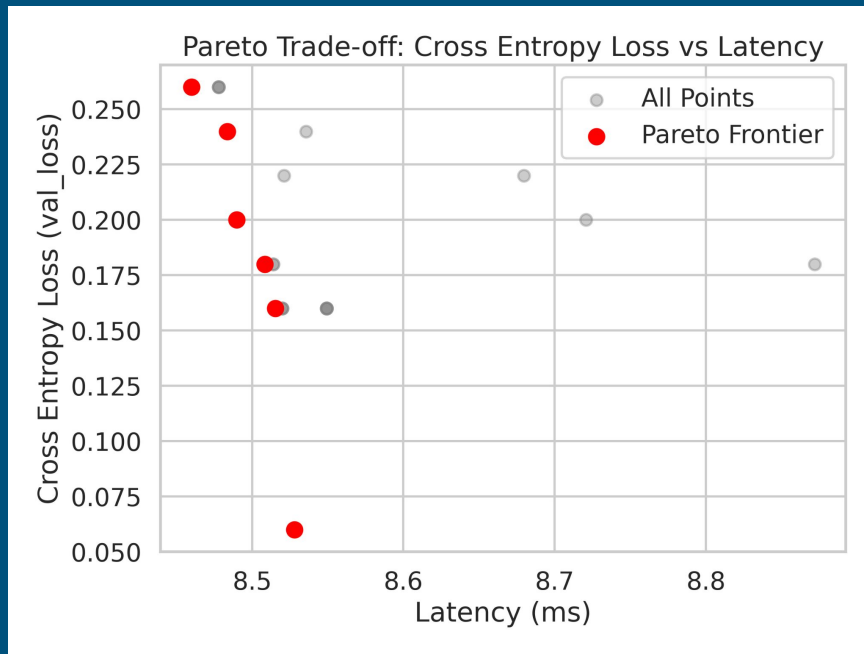
# Measurement Result #1 (Mistral) : Cross Entropy Loss vs. Memory

- In this result, quantization did not seem to help with loss (better large models)
- Distinct groups of memory with different sizes



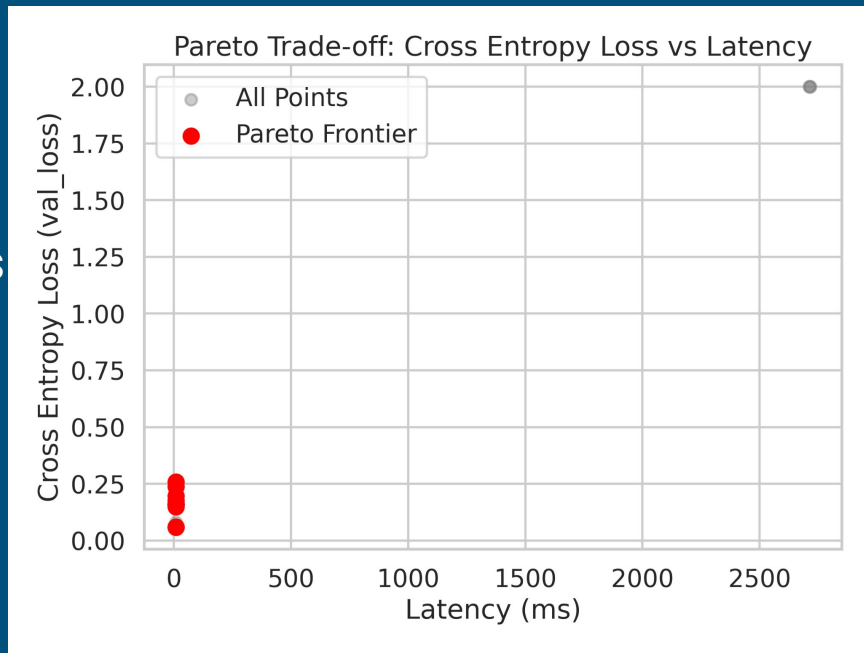
# Measurement Result #2 (QWEN): Loss vs. Latency

- Steep downward slope, but with slight curve outward
  - Loss decreases as latency increases (very slightly)
- In our experimental results, the speed increase from quantization did not have a significant effect on trade off



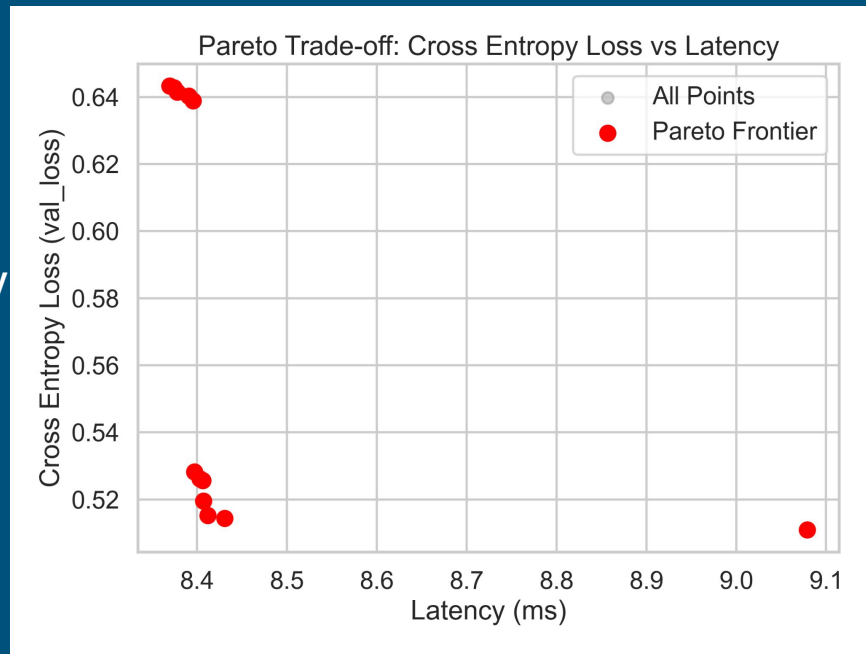
# Measurement Result #2 (Trinity): Loss vs. Latency

- Downward slope
  - Loss decreases as latency increases
- Clear view of low latency and cross entropy loss for Trinity
  - Once again reliant on pretraining of the model



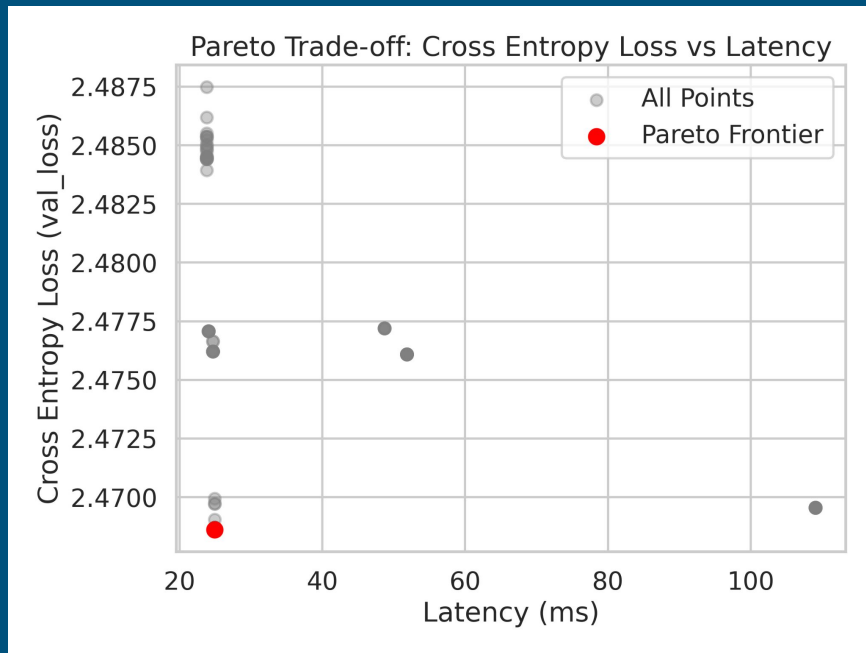
# Measurement Result #2 (Orchestrator): Loss vs. Latency

- Most configurations operate within a very narrow low-latency range
- Cross entropy loss shows relatively high variability
- The single high-latency point acts as an outlier



# Measurement Result #2 (Mistral): Loss vs. Latency

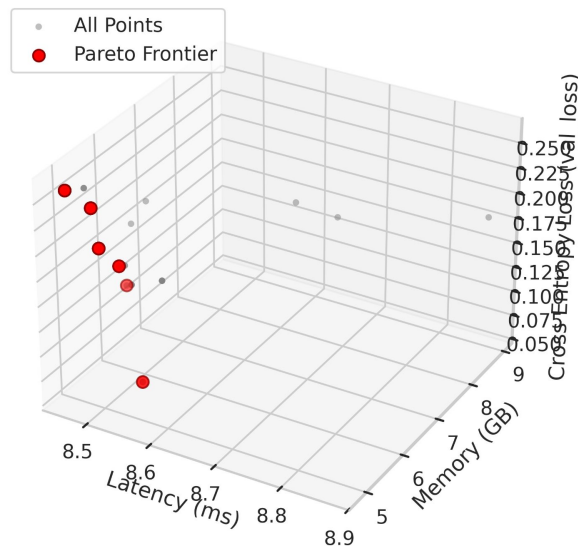
- Vertical line, all models centered around about 25 ms latency
- Some outliers of higher latency for lower cross entropy than main group





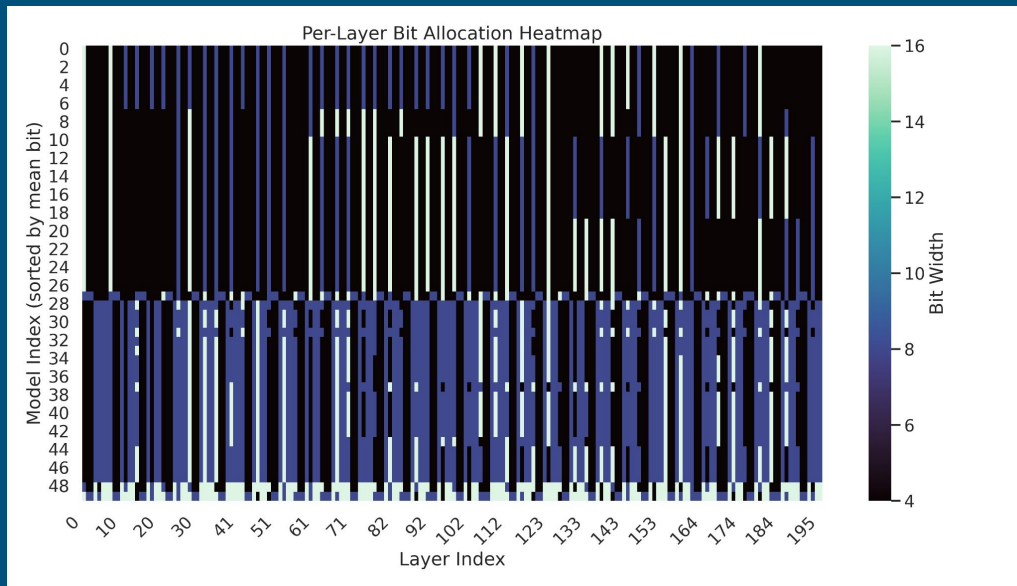
## Measurement Result #3 (All models): Loss vs. Memory vs. Latency

- Best pretrained models lie in low-latency, low-memory corner
  - Best configuration are faster and smaller, implying stronger quantization
- High-memory models consistently have higher latency without significant accuracy benefit



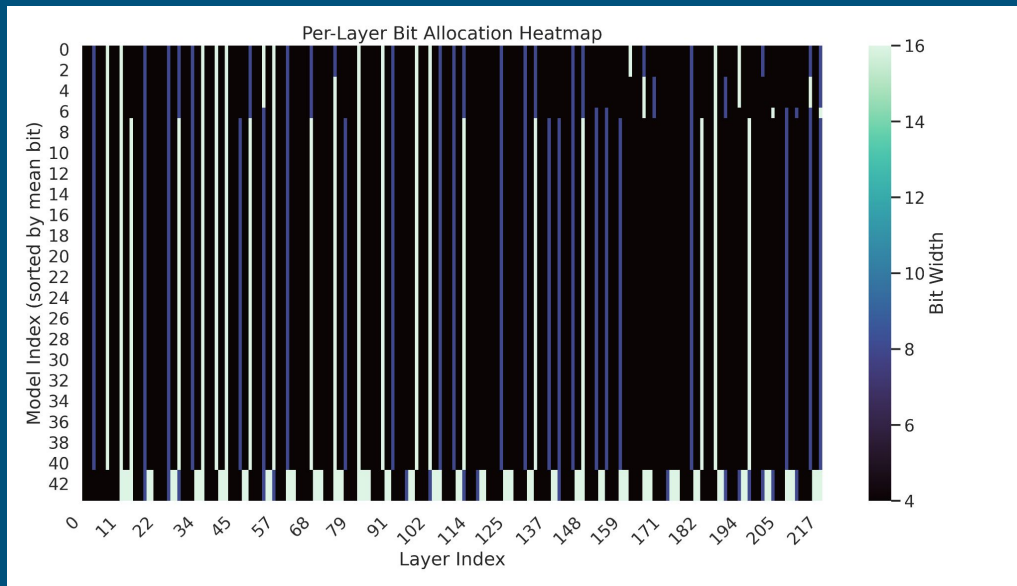
# Measurement Result #4 (Qwen): Bit Allocation Heatmap

- Most models do not have uniform bit widths
  - However, certain layers consistently get assigned similar bit sizes
- Best-performing models use non-uniform precision vectors



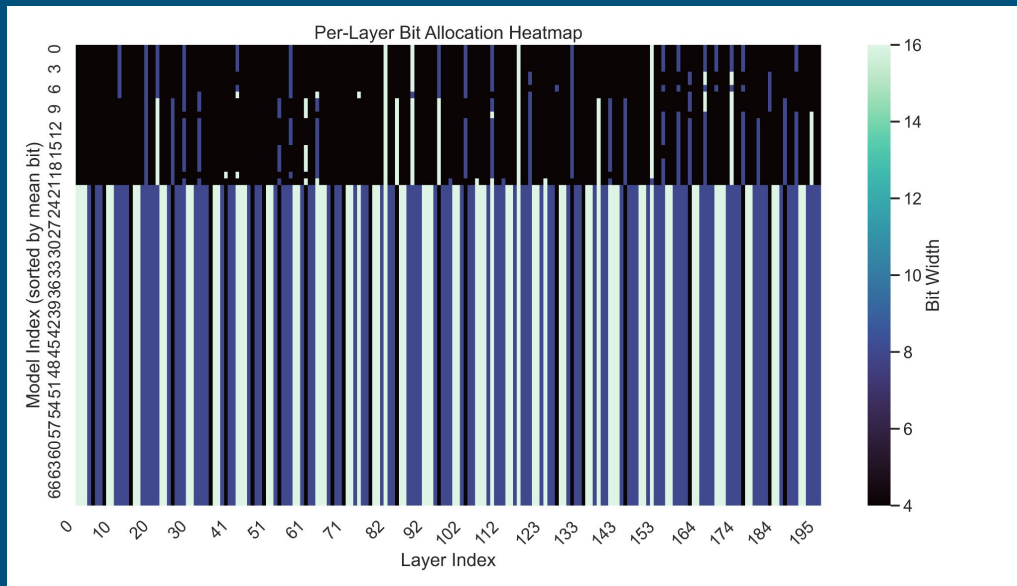
# Measurement Result #4 (Trinity): Bit Allocation Heatmap

- Most models do not have uniform bit widths
  - However, certain layers consistently get assigned similar bit sizes
- Best-performing models use non-uniform precision vectors



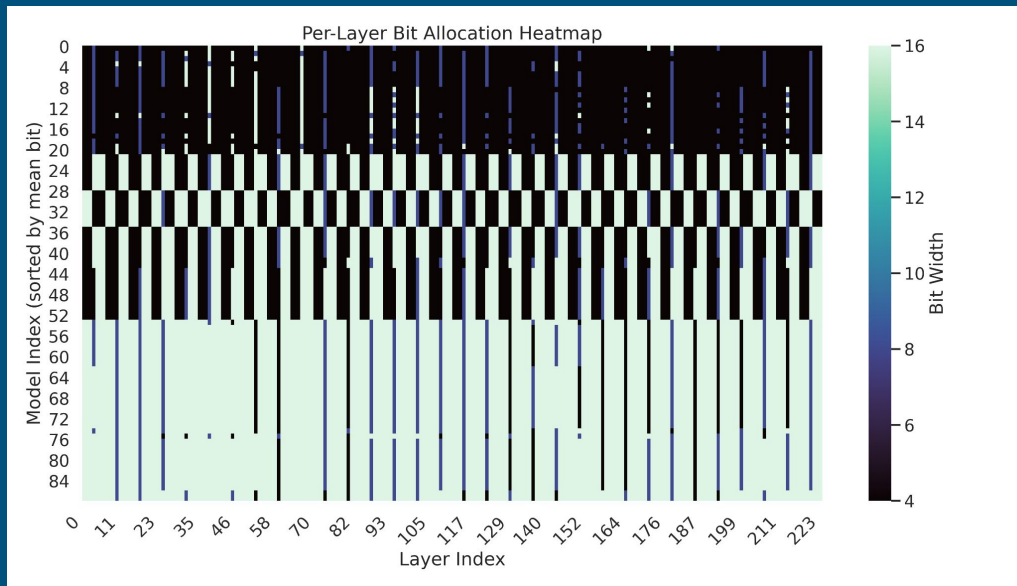
# Measurement Result #4 (Orchestrator): Bit Allocation Heatmap

- Most models do not have uniform bit widths
  - However, certain layers consistently get assigned similar bit sizes
- Best-performing models use non-uniform precision vectors



# Measurement Result #4 (Mistral): Bit Allocation Heatmap

- Most models do not have uniform bit widths
  - However, certain layers consistently get assigned similar bit sizes
- Best-performing models use non-uniform precision vectors





# Project Summary



# Key Contributions

Successfully created an automated tool that finds pareto-optimal layer quantization configurations for LLMs under arbitrary memory and latency constraints.

```
1 latency_lut:
2   - key:
3     - matmul
4     - 128
5     - 3584
6     - 3584
7     - 2
8   us: 22.3411201313138
9   - key:
10    - matmul
11    - 128
12    - 3584
13    - 3584
14    - 3
15   us: 22.3411201313138
16   - key:
17     - matmul
18     - 128
19     - 3584
20     - 3584
21     - 4
22   us: 22.3411201313138
```

## LUT Table For Latency Estimation

Created a LUT table to enable fast, hardware-aware latency estimation.

## Accurate VRAM Computation

Computed VRAM usage from parameter count x bit width

## Parallel GPU Evaluation

Parallel Evaluation across multiple GPUs for faster processing

## Pareto Frontier Visualization

Created and generated 2D/3D pareto frontier plots for easy referencing

# Lessons Learned

---

## Accuracy vs Cross Entropy Evaluation

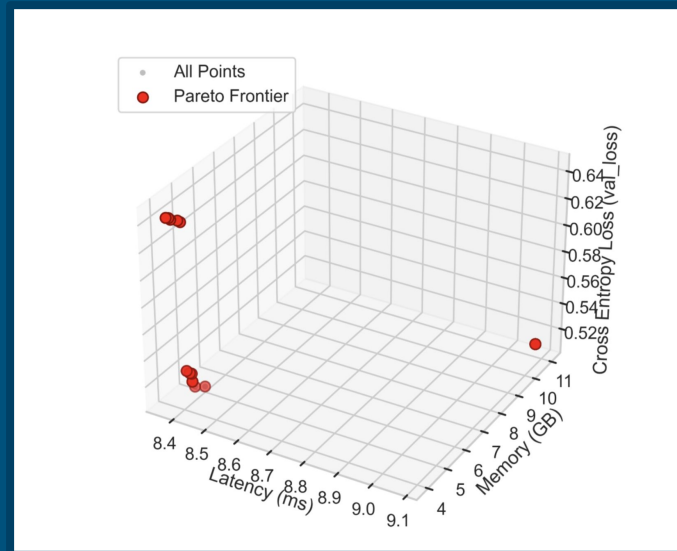
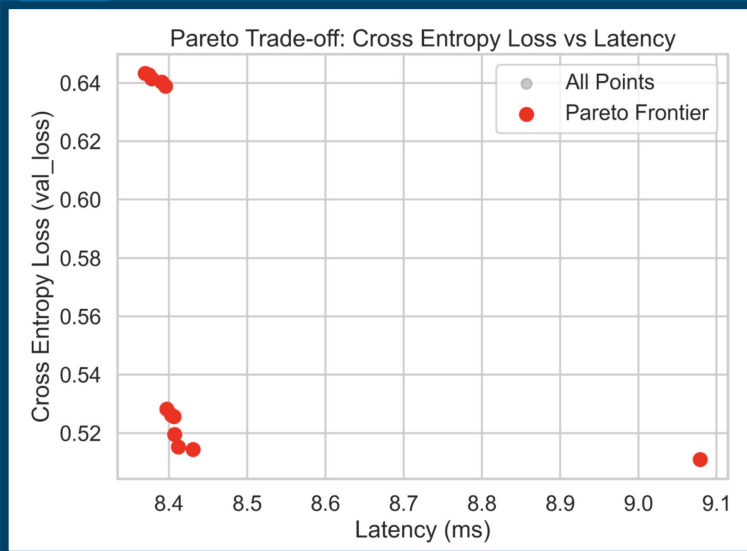
- Cross entropy loss - measures how much the model's next-token predictions diverge from Math-500 gold solutions.
- Accuracy Metric - Directly computes correctness to see if generated answer is correct or not

**Why Cross Entropy?** - Sensitive to small perturbations in quantized weights providing smooth, stable optimization

**Why Accuracy Performed Better** - The model wasn't trained on Math-500, token level cross entropy loss produced noisy comparisons. There was really bad convergence overall. Accuracy offered a more clear ranking of "better vs worse" models.

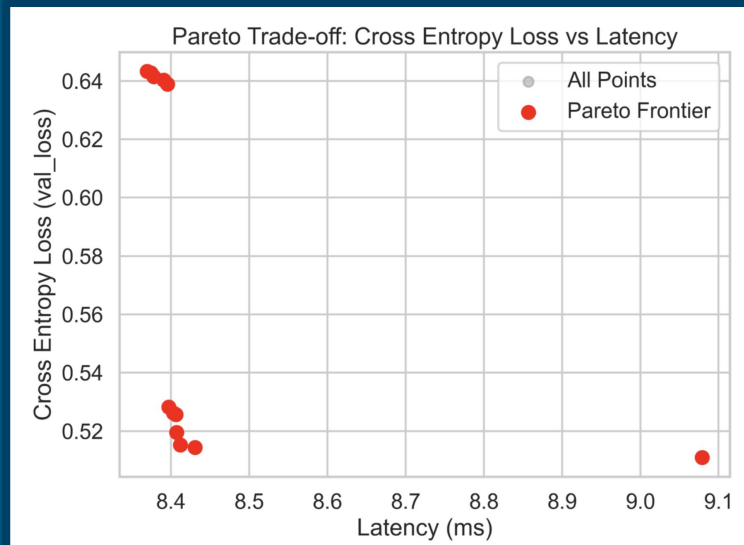


# Lessons Learned - Preliminary Results (QWEN)



Note: You can't see the non-pareto points because they cluster around the Pareto Frontier!

# Lessons Learned - Preliminary Results (QWEN)



```

{
  "n_all": 646.0,
  "n_frontier": 81.0,
  "frontier_ratio": 0.12538699690402477,
  "objectives_all": {
    "latency_ms": {
      "count": 69.0,
      "min": 8.36955105460832,
      "p25": 8.407168718608318,
      "median": 9.07914864820832,
      "p75": 9.07914864820832,
      "max": 9.07914864820832,
      "mean": 8.87077684122861,
      "std": 0.315163011811839
    },
    "memory_gb": {
      "count": 646.0,
      "min": 3.928752128,
      "p25": 4.421451776,
      "median": 4.882038784,
      "p75": 10.676994048,
      "max": 12.497321984,
      "mean": 7.07940405260681,
      "std": 3.006206082828667
    },
    "val_loss": {
      "count": 646.0,
      "min": 0.5109182275003857,
      "p25": 0.5309303532595988,
      "median": 0.5514976450690516,
      "p75": 0.6375011700287736,
      "max": 0.6512670234243483,
      "mean": 0.5792923453484726,
      "std": 0.05284727858420227
    }
  },
  "val_loss": {
    "count": 646.0,
    "min": 0.5109182275003857,
    "p25": 0.5309303532595988,
    "median": 0.5514976450690516,
    "p75": 0.6375011700287736,
    "max": 0.6512670234243483,
    "mean": 0.5792923453484726,
    "std": 0.05284727858420227
  },
  "bits_all": {
    "counts": {
      "4": 70436,
      "8": 31834,
      "16": 24346
    },
    "fractions": {
      "4": 0.556296202691603,
      "8": 0.251421621280091,
      "16": 0.19228217602830605
    }
  },
  "bits_frontier": {
    "counts": {
      "4": 6621,
      "8": 5538,
      "16": 3717
    },
    "fractions": {
      "4": 0.4170445956160242,
      "8": 0.3488284202569917,
      "16": 0.23412698412698413
    }
  },
  "per_layer_all": [
    {
      "layer": 0,
      "mean": 8.910216718266254,
      "std": 5.703835080711555,
      "hist": {
        "4": 351,
        "8": 46,
        "16": 249
      }
    }
  ],
  "objectives_frontier": {
    "latency_ms": {
      "count": 69.0,
      "min": 8.36955105460832,
      "p25": 8.407168718608318,
      "median": 9.07914864820832,
      "p75": 9.07914864820832,
      "max": 9.07914864820832,
      "mean": 8.87077684122861,
      "std": 0.315163011811839
    },
    "memory_gb": {
      "count": 81.0,
      "min": 3.928752128,
      "p25": 4.296671232,
      "median": 11.024728064,
      "p75": 11.024728064,
      "max": 11.024728064,
      "mean": 8.322950637037835,
      "std": 3.3449665092870706
    },
    "val_loss": {
      "count": 81.0,
      "min": 0.5109182275003857,
      "p25": 0.5109182275003857,
      "median": 0.5109182275003857,
      "p75": 0.5256021411882507,
      "max": 0.6432530486363405,
      "mean": 0.536011221675021,
      "std": 0.04862506209113528
    }
  }
}

```

```

{
  "bits_all": {
    "counts": {
      "4": 70436,
      "8": 31834,
      "16": 24346
    },
    "fractions": {
      "4": 0.556296202691603,
      "8": 0.251421621280091,
      "16": 0.19228217602830605
    }
  },
  "bits_frontier": {
    "counts": {
      "4": 6621,
      "8": 5538,
      "16": 3717
    },
    "fractions": {
      "4": 0.4170445956160242,
      "8": 0.3488284202569917,
      "16": 0.23412698412698413
    }
  },
  "per_layer_all": [
    {
      "layer": 0,
      "mean": 8.910216718266254,
      "std": 5.703835080711555,
      "hist": {
        "4": 351,
        "8": 46,
        "16": 249
      }
    }
  ],
  "objectives_frontier": {
    "latency_ms": {
      "count": 69.0,
      "min": 8.36955105460832,
      "p25": 8.407168718608318,
      "median": 9.07914864820832,
      "p75": 9.07914864820832,
      "max": 9.07914864820832,
      "mean": 8.87077684122861,
      "std": 0.315163011811839
    },
    "memory_gb": {
      "count": 81.0,
      "min": 3.928752128,
      "p25": 4.296671232,
      "median": 11.024728064,
      "p75": 11.024728064,
      "max": 11.024728064,
      "mean": 8.322950637037835,
      "std": 3.3449665092870706
    },
    "val_loss": {
      "count": 81.0,
      "min": 0.5109182275003857,
      "p25": 0.5109182275003857,
      "median": 0.5109182275003857,
      "p75": 0.5256021411882507,
      "max": 0.6432530486363405,
      "mean": 0.536011221675021,
      "std": 0.04862506209113528
    }
  }
}

```

```

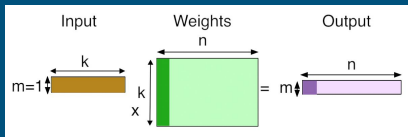
{
  "objectives_frontier": {
    "latency_ms": {
      "count": 69.0,
      "min": 8.36955105460832,
      "p25": 8.407168718608318,
      "median": 9.07914864820832,
      "p75": 9.07914864820832,
      "max": 9.07914864820832,
      "mean": 8.87077684122861,
      "std": 0.315163011811839
    },
    "memory_gb": {
      "count": 81.0,
      "min": 3.928752128,
      "p25": 4.296671232,
      "median": 11.024728064,
      "p75": 11.024728064,
      "max": 11.024728064,
      "mean": 8.322950637037835,
      "std": 3.3449665092870706
    },
    "val_loss": {
      "count": 81.0,
      "min": 0.5109182275003857,
      "p25": 0.5109182275003857,
      "median": 0.5109182275003857,
      "p75": 0.5256021411882507,
      "max": 0.6432530486363405,
      "mean": 0.536011221675021,
      "std": 0.04862506209113528
    }
  }
}

```

Summary stats demonstrate that quantization is indeed happening and frontier had higher precision overall.

# More Lessons Learned

## Hardware-Aware ML Model Engineering



- Predicting memory costs from GEMM kernels
- Working with memory bandwidth and quantization overhead
- GPU profiling with warm-up runs, CUDA synchronization, and caching

## Understanding Layer Quantization

```
{  
  "layer": 140,  
  "mean": 10.56,  
  "std": 5.8030139952072695,  
  "hist": {  
    "4": 72,  
    "8": 11,  
    "16": 92  
  }  
},
```

```
{  
  "layer": 15,  
  "mean": 5.142857142857143,  
  "std": 2.4511554959459674,  
  "hist": {  
    "4": 135,  
    "8": 35,  
    "16": 5  
  }  
},
```

- Understanding which groupings of layers are quantization sensitive through evolutionary testing

## Developing an LLM Evaluation Pipeline

\_\_init\_\_.py  
bnb\_quantize.py  
cache.py  
config.py  
core\_evolutionary.py  
data\_models.py  
evaluator.py

frontier.py  
group\_ties.py  
logging\_utils.py  
lut.py  
main\_search.py  
objectives.py  
utils.py

- GPU parallelism
- Cache aware
- Stateful - not needing to load a new model every time due to high latency costs

# Future Improvements

---

- **Pluggable APIs** - Currently our model requires a lot of extra code
  - bitsandbytes currently requires its own file for layer quantization
    - Potentially could create an API that allows you to pass in a quantization scheme (1 bit, 2 bit, etc).
  - Additional APIs for figures and results
    - Adding an API where you input memory or latency budgets and get returned a clear quantization scheme
- Limited generalizability across diverse model architectures
  - Applying the method to new models requires substantial model-specific implementation
- Clearer results - graphs can tell a story but there must be a better way to present the different layer schemes.

# README Explanation

---

- Project Overview & Technical Stack: Describes AutoLLMTuner as an evolutionary algorithm framework for LLM quantization,
  - Detailed information about datasets (MATH-500), performance measurement tools (BitsAndBytes, hardware benchmarking, caching), and external resources (Qwen2.5-7B model, frameworks)
- Setup & Usage: Provides installation requirements, configuration files explanation, and instructions for running experiments
  - Hardware benchmarking, evolutionary search, HPC execution, visualization generation
- Experimental Results & Analysis: Comprehensive analysis of test runs showing that optimal solutions achieve 94% accuracy at 5.51 GB memory (vs 12.49 GB baseline),
  - Detailed performance metrics, Pareto analysis, bit allocation strategies, and visualization explanations
- Code Architecture: Documents the project structure with core components (evolutionary algorithm, quantization backend, visualization tools)
  - Explains how evaluation, frontier computation, and results generation are integrated throughout the pipeline

# Open Source References

---

- HuggingFace Transformers  
(Model loading and inference) - <https://huggingface.co/docs/transformers/en/index>
- HuggingFace MATH500  
(Dataset for testing) - <https://huggingface.co/datasets/HuggingFaceH4/MATH-500>
- BitsAndBytes  
(4-bit, 8-bit quantization backend) - <https://huggingface.co/docs/bitsandbytes/en/index>
- Qwen2.5-7B-Instruct  
(base LLM) - <https://huggingface.co/Qwen/Qwen2.5-7B-Instruct>
- PyTorch (GPU inference) - <https://pytorch.org/>
- Matplotlib - <https://matplotlib.org/>
- Seaborn - <https://seaborn.pydata.org/>

Q&A

Thank You!!