Team 27_final_project

GitHub Repository Link

https://github.com/fakraze/Edge-AI-final

Hugging Face Space / Model Page

https://huggingface.co/BrianGodd/hqq-llama3-3b

https://huggingface.co/fakraze/qlora-wikitext2-llama3-3b

Methodology - Describe your approach

- **HQQ (Hessian-aware Quantization for Transformers)**
  - It uses **second-order Hessian information** to identify which weights are less sensitive to quantization, allowing more aggressive compression and minimal accuracy loss(like PPL)

- **Customized Layer-Wise Quantization**
  - **Important modules (e.g., `o_proj`)** get higher precision (e.g., 8-bit, smaller group size)
  - **Less critical modules (e.g., `down_proj`)** use aggressive 4-bit with larger group sizes
  - We have conducted over 15 experiments; for detailed information, please refer to the **Experimental Results** section.
  - Best Config Setting
    - `q_proj` → `4-bit, group_size=32`
      (**Very Aggressive Compression**)
      => Query projections are involved early in attention computation and can tolerate aggressive quantization.
    - `k_proj` / `v_proj` → `8-bit, group_size=256`
      (**Moderate Precision, Large Grouping**)
      => Keys and values are essential for attention map quality. Over-compression harms model interpretability.
    - `o_proj` → `8-bit, group_size=128`

(**High Precision Output Stability**)
=> The output of attention is directly used in residuals. Preserving this helps maintain semantic fidelity.

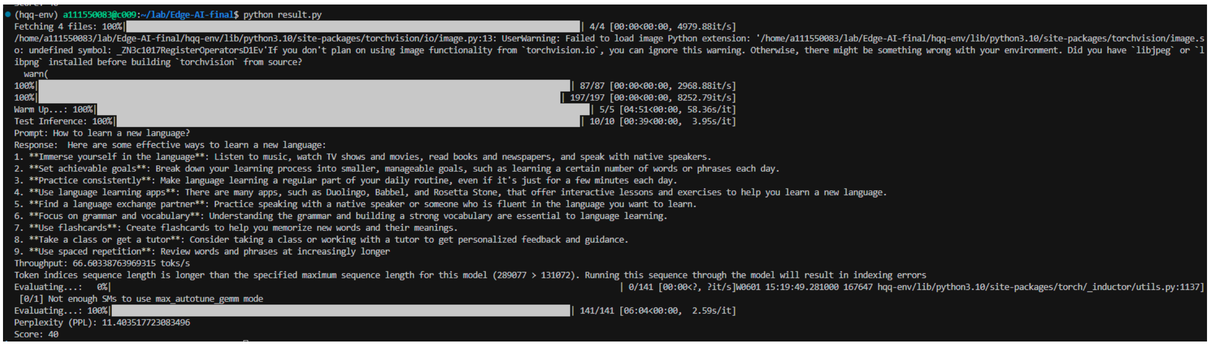
- `gate_proj` / `up_proj` → `4-bit, group_size=128`
  (**Efficient Feedforward Layers**)
  => MLPs take a large computing share. By using 4-bit quantization but with **a larger group size**, we balance speed and stability.
- `down_proj` → `4-bit, group_size=128`
  (**Safe for Compression**)
  => As a dimensionality reduction layer, it is less prone to amplifying noise. Fully compressible.

- **torch.compile**
  - **torch.compile()** converts model forward pass into optimized TorchDynamo graphs.
  - **fullgraph=True**: compile the full forward pass as a single graph.
  - **mode='max-autotune'**: enables kernel auto-tuning for best performance.

- **Backend Inference**
  - Switches the **HQQLinear backend** to **gemlite**, an efficient CUDA backend.
  - Enables optimized kernel dispatching for HQQ layers like `q_proj`, `gate_proj`, etc.

Experimental Results - Screenshot your results

- **Best Result: (throughput) 66.603 / (PPL) 11.403**



```
(hqq-env) a111550083@c009:~/lab/Edge-AI-final$ python result.py
Fetching 4 files: 100%|████████████████████| 4/4 [00:00<00:00, 4979.88it/s]
/home/a111550083/lab/Edge-AI-final/hqq-env/lib/python3.10/site-packages/torchvision/io/image.py:13: UserWarning: Failed to load image Python extension: '/home/a111550083/lab/Edge-AI-final/hqq-env/lib/python3.10/site-packages/torchvision/image.s
o: undefined symbol: _ZN3c1017RegisterOperatorsD1Ev'If you don't plan on using image functionality from `torchvision.io`, you can ignore this warning. Otherwise, there might be something wrong with your environment. Did you have `libjpeg` or `l
ibpng` installed before building `torchvision` from source?
  warn(
100%|████████████████████| 87/87 [00:00<00:00, 2968.88it/s]
100%|████████████████████| 197/197 [00:00<00:00, 8252.79it/s]
Warm Up...: 100%|████████████████████| 5/5 [04:51<00:00, 58.36s/it]
Test Inference: 100%|████████████████████| 10/10 [00:39<00:00,  3.95s/it]
Prompt: How to learn a new language?
Response:  Here are some effective ways to learn a new language:
1. **Immerse yourself in the language**: Listen to music, watch TV shows and movies, read books and newspapers, and speak with native speakers.
2. **Set achievable goals**: Break down your learning process into smaller, manageable goals, such as learning a certain number of words or phrases each day.
3. **Practice consistently**: Make language learning a regular part of your daily routine, even if it's just for a few minutes each day.
4. **Use language learning apps**: There are many apps, such as Duolingo, Babbel, and Rosetta Stone, that offer interactive lessons and exercises to help you learn a new language.
5. **Find a language exchange partner**: Practice speaking with a native speaker or someone who is fluent in the language you want to learn.
6. **Focus on grammar and vocabulary**: Understanding the grammar and building a strong vocabulary are essential to language learning.
7. **Use flashcards**: Create flashcards to help you memorize new words and their meanings.
8. **Take a class or get a tutor**: Consider taking a class or working with a tutor to get personalized feedback and guidance.
9. **Use spaced repetition**: Review words and phrases at increasingly longer
Throughput: 66.60338763969315 toks/s
Token indices sequence length is longer than the specified maximum sequence length for this model (289077 > 131072). Running this sequence through the model will result in indexing errors
Evaluating...:   0%|          | 0/141 [00:00<?, ?it/s]W0601 15:19:49.281000 167647 hqq-env/lib/python3.10/site-packages/torch/_inductor/utils.py:1137]
 [0/1] Not enough SMs to use max_autotune_gemm mode
Evaluating...: 100%|████████████████████| 141/141 [06:04<00:00,  2.59s/it]
Perplexity (PPL): 11.403517723083496
Score: 40
```

- **Comparison of HQQ Config (first time run on Kaggle)**

q_proj	k_proj	v_proj	o_proj	gate_proj	up_proj	down_proj	Throughput	PPL
4b/64	8b/64	8b/64	8b/64	8b/64	8b/64	4b/64	36.8	11.1
mix:2~8b	8b/64	8b/64	8b/64	8b/64	8b/64	4b/64	45.32	11.45
mix:2~8b	8b/64	8b/64	8b/64	8b/64	8b/64	4b/64	49.18	11.64
mix:4~8b	8b/64	8b/64	8b/64	8b/64	8b/64	4b/64	50.3	11.491
mix:2~8b	mix	mix	8b/64	mix	mix	4b/64	60.58	19.88
mix:2~8b	mix	mix	8b/64	mix	mix	4b/32	53.64	17.04
4b/32~64	8b/128	8b/128	8b/64	mix:4~8b	mix:4~8b	4b/64	52.87	11.309
4b/32~64	8b/128	8b/128	8b/64	mix:4~8b	mix:4~8b	4b/128	53.71	11.305
4b/32~64	8b/128	8b/128	8b/64	mix:4~8b	mix:4~8b	4b/128	53.92	11.315
4b/32~64	8b/128	8b/128	8b/64	mix:4~8b	mix:4~8b	4b/128	54.7	11.31
4b/32~64	8b/128	8b/128	8b/64	mix:4~8b	mix:4~8b	4b/128	54.05	11.295
4b/32~64	8b/128	8b/128	8b/64	mix:4~8b	mix:4~8b	4b/128	55.36	11.295
4b/32	8b/128	8b/128	8b/64	4b/64	4b/64	4b/128	56.75	11.417
4b/32	8b/256	8b/256	8b/64	4b/64	4b/64	4b/128	57.01	11.412
4b/32	8b/256	8b/256	8b/128	4b/128	4b/128	4b/128	60.09	11.403
(nbit/group size)								

("mix" means using different settings in different layer number, total 28 layers)

Team Member Contributions

黃永恩: HQQ Quantiztion, QLoRA

曾紹幃: QLoRA, GPTQ, SPDA

林庠安: vLLM, GPTQ

游子毅: Quantization, Flash Attention

Insights or Explorations

1. I think Edge AI is a rapidly evolving field in recent years, but the development environment is still not very user-friendly. It often takes a lot of time to resolve

package conflicts. Additionally, training models can be very time-consuming — for example, fine-tuning with QLoRA took around 8 hours, which is quite substantial.
2. In the additional experiments section, we explored numerous acceleration methods; however, only a few combinations were compatible. Since each method operated in a distinct domain, we frequently encountered version conflicts among various packages, which posed considerable challenges for us.

Optional Enhancement

Experiment: vLLM

Setting environment

```
!pip install vllm
from vllm import LLM, SamplingParams
```

vLLM is an efficient and fast inference library for large language models (LLMs), designed to maximize GPU utilization and support high-throughput, low-latency serving. It uses techniques like PagedAttention to handle dynamic batching and long sequences efficiently.
Here's what the parameters mean:

- model → the name or path of the LLM (here: Meta's Llama-3.2-3B-Instruct).
- dtype → the precision used (float16 reduces memory use & speeds up computation).
- tensor_parallel_size → how many GPUs to split the model across (for parallelism).
- max_model_len → the maximum length (in tokens) the model can process per sequence.
- gpu_memory_utilization → the target percentage of GPU memory to use.
- max_num_seqs → maximum number of sequences (requests) processed simultaneously.
- max_num_batched_tokens → maximum number of tokens combined across all batches.

Setting parameters

```
    llm = LLM(
        model=model_name,
        dtype="float16",
        tensor_parallel_size=tensor_parallel_size,
        max_model_len=4096,                    # <— added to avoid KV cache overflow
        gpu_memory_utilization=0.95,
        max_num_seqs=2*4,
        max_num_batched_tokens=8192*8,# <— added to better use GPU memory

    )
```

Result

```
Time Record: [7.4061533203125, 7.44703173828125, 7.38351708984375, 7.4169296875, 7.41438232421875, 7.39722119140625,
Throughput Record: [34.565852059513965, 34.37611238905287, 34.67182331739107, 34.51562988812546, 34.52748844145619,

Throughput: 34.58350353246011 toks/s
```

Perplexity: Not moving

Throughput: Growing from ~27 → 35 tokens/s

Experiment: Enable SDPA Attention Acceleration

SDPA (Scaled Dot-Product Attention) is an optimized attention backend in PyTorch that improves speed and memory efficiency. It uses fused kernels (like FlashAttention) when supported by hardware.

To evaluate the potential acceleration provided by native attention mechanisms in PyTorch, we enabled attn_implementation="sdpa" when loading the model:

```
### === TODO: Load your model (you may change this part) ===
model_name = "meta-llama/Llama-3.2-3B-Instruct"
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16,
    device_map=device,
    attn_implementation="sdpa"  # 啟用原生 Attention 加速, spda
)
print(f"Loaded model: {model_name}")

#####################################
```

Result

```
**Step 1: Set Your Goals and Motivation**

Before you start learning a new language, it's essential to define your goals and motivation. Why do you want to learn a new language? Is it for travel, work, or personal enrichment? Setting s
pecific, achievable goals will help you stay motivated and focused throughout the learning process.

* Identify your target language and level of proficiency (beginner, intermediate, advanced).
* Set realistic goals, such as passing a language proficiency test or being able to hold conversations with native speakers.
* Find a language learning buddy or join a language exchange group to stay motivated and accountable.

**Step 2: Choose the Right Learning Resources**

There are many effective ways to learn a new language, and the right resources can make a significant difference in your success. Here are some popular options:

* **Language learning apps:** Duolingo, Babbel, and Rosetta Stone are popular apps that offer interactive lessons and exercises.
* **Language courses:** Enroll in a language course at a local college or language school, or online courses like Coursera or edX.
* **Language exchange websites

Time Record: [9.18232421875, 9.114021484375, 9.176744140625, 9.128677734375, 9.1401103515625, 9.247796875, 9.2052333984375, 9.2142275390625, 9.178115234375, 9.27251953125]
Throughput Record: [27.879651589437074, 28.088588603711788, 27.896604294185387, 28.043491888864143, 28.008414576333518, 27.68226891877964, 27.810267151233074, 27.783121147673185, 27.8924368961
06677, 27.608461663222773] toks/s

Throughput: 27.878415942494815 toks/s
Token indices sequence length is longer than the specified maximum sequence length for this model (289077 > 131072). Running this sequence through the model will result in indexing errors
Evaluating...: 100% 141/141 [02:06<00:00,  1.11it/s]
Perplexity (PPL): 11.0475492477417
```

Perplexity: Not moving

Throughput: Almost not moving

Conclusion:

Enabling SDPA did not lead to noticeable improvements in either perplexity or throughput on our T4 GPU. This suggests that the hardware may not fully support the optimized fused kernels required to benefit from SDPA. Therefore, SDPA alone is not effective for acceleration in this setup.

Experiment: QLoRA Fine-Tuning

QLoRA (Quantized Low-Rank Adapter) is a memory-efficient fine-tuning method that combines 4-bit quantization with trainable LoRA adapters. It enables large models like LLaMA to be fine-tuned on a single GPU (e.g., T4) with low memory cost and good performance.

We fine-tuned the model using QLoRA on WikiText-2.

Result

```
Throughput: 23.714939419139757 toks/s
Token indices sequence length is longer than the specified maximum sequence length for this model (289077 > 131072). Running this sequence through the model will result in indexing errors
Evaluating...: 100%| | 141/141 [01:50<00:00,  1.28it/s]
Perplexity (PPL): 10.036947250366211
```

Perplexity: 11.04 → 10.03

Throughput: ~27 → 23.7 tokens/s

Conclusion

QLoRA improved model accuracy but reduced inference speed due to the added adapter layers. This tradeoff is acceptable for accuracy-focused tasks but may not suit real-time applications.

Experiment: GPTQ

GPTQ (Grouped-Precision Quantization) is a post-training quantization method that compresses large language models by converting their weights to low-bit precision (e.g., 4-bit), without requiring retraining. It uses group-wise calibration to minimize accuracy loss, aiming to reduce model size and improve inference speed. However, performance may vary depending on hardware and model structure.

We quantized the original model using GPTQ.

Result:

```
Test Inference: 100%                    10/10 [01:49<00:00, 10.89s/it]
Prompt: How to learn a new language?
Response: anTven Mess11}vevelve``-````devam0*devdev*0**``|#*]|||#valW|||igersteadelt* North}|w||j1-a|| above|6|0|9|| L|||j| Above|L|| above||||od| above||| # T|||| above-# L. L|W-; L- Nuevo*
L... L-G... 5... American...1 200...USA... (5... now-j...L-g... L...ize.L- L when... - L... dig... Mata- above... now- Calls...* L... USA... L... L:...j now...-1:L... L-...- USA above...jan
G...L-j-j-e...A...L...G... L... Near...L... USA... NOW- L-00... where now when...ok...A...We... L... nowCL...G...above... now K- USA*... beyond coding above... when...]j k...0... Nem. L... abo
ve- now L...5:

Time Record: [11.58907421875, 11.29222265625, 10.57205859375, 11.21425390625, 10.7472109375, 10.6272265625, 11.11995703125, 10.8300146484375, 10.753609375, 10.9135478515625]
Throughput Record: [22.08977138016916, 22.6704704461623, 24.214773095501222, 22.8280902269677, 23.820133566630297, 24.08906956997982, 23.021671691767583, 23.63801050231584, 23.8059604987279, 2
3.457083203547626] toks/s

Throughput: 23.428491614992822 toks/s
Token indices sequence length is longer than the specified maximum sequence length for this model (289077 > 131072). Running this sequence through the model will result in indexing errors
Evaluating...: 100%                    141/141 [05:45<00:00,  2.46s/it]
Perplexity (PPL): 15.345739364624023
```

Perplexity: 11.04 → 15.35

Throughput: ~27 → 23.4 tokens/s

Conclusion

GPTQ reduced model size, but caused a notable drop in output quality and increased perplexity, without significant speedup on T4 GPU.

This shows that GPTQ alone is insufficient to retain the original LLM's performance.

Experiment: QLoRA + GPTQ

We applied GPTQ post-quantization on the QLoRA-fine-tuned model.

Result

```
Throughput: 24.390556502922653 toks/s
Token indices sequence length is longer than the specified maximum sequence length for this model (289077 > 131072). Running this sequence through the model will result in indexing errors
Evaluating...: 100%                    141/141 [05:49<00:00,  2.48s/it]
Perplexity (PPL): 113.49898529052734
```

Perplexity: 11.04 → 113.50 (vs. 10.03 for QLoRA alone)

Throughput: ~27 → 24.4 tokens/s

Conclusion

Despite combining two optimization methods, the perplexity skyrocketed, and generation quality degraded severely.

The QLoRA + GPTQ combination led to catastrophic degradation in model accuracy.

This suggests that applying GPTQ after QLoRA may be incompatible or require careful calibration.

Experiment: Enable xformers Flash Attention 1

We implemented xformers Flash Attention 1.x through dynamic module patching to replace the standard attention mechanism in Llama-3.2-3B-Instruct. We use a recursive function to locate all attention modules within the model architecture and substitute their forward methods with memory-efficient xformers implementations. This should maintains\ the original model structure while enabling hardware-optimized attention computation that reduces memory complexity from $O(N^2)$ to $O(N)$ through block-wise processing.

Result

```
Time Record: [8.2422080078125, 8.349005859375, 8.2673203125, 8.2757958984375, 8.2835986328125, 8.2964189453125, 8.3120126953125, 8.3
32095703125, 8.3816259765625, 8.3748779296875]
Throughput Record: [31.05963835871972, 30.662333254029356, 30.9652935078533, 30.933580666040072, 30.904442784799826, 30.856686684637
683, 30.798798002843444, 30.724563077688334, 30.54299973726477, 30.56760971912487] toks/s

Throughput: 30.813400745006458 toks/s
Token indices sequence length is longer than the specified maximum sequence length for this model (289077 > 131072). Running this se
quence through the model will result in indexing errors
Evaluating...: 100%|████████████████████████████████████████| 141/141 [01:36<00:00,  1.46it/s]
Perplexity (PPL): 11.0475492477417
```

Perplexity: Not moving

Throughput: ~27 → 30.8 tokens/s

Conclusion

The 11% throughput improvement indicates that xformers Flash Attention was successfully enabled yet not too effective.