Edge AI: Final Project

Group 37: 江尚軒、蔡昀叡、陳建樺、李任本耀

GitHub Repository Link

https://github.com/AndyChiangSH/1132-edge-ai-final-project

Hugging Face Space / Model Page

LoRA model:
https://huggingface.co/x21530317x/llama3.2-3B-instruct_lora_bf16/tree/main

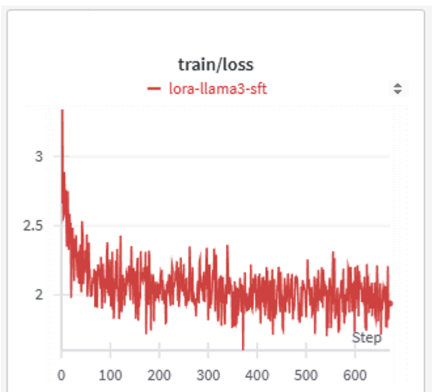
Methodology – Describe your approach

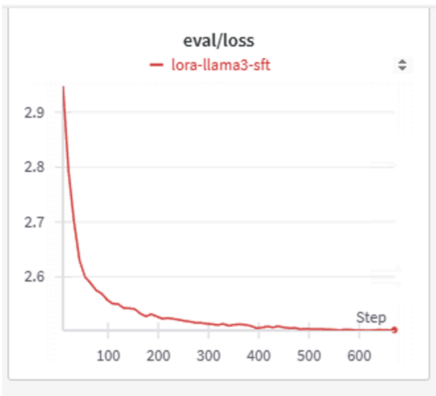
Model Training

我們使用 SFTTrainer 訓練 LoRA，使用 H100x6 訓練 Salesforce/wikitext 的training set，練了1.5hr，訓練參數設置如下：rank=8, alpha=16, dropout=0.1, max_seq_length=2048。訓練層數為："q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"。訓練過程如下：

LoRA

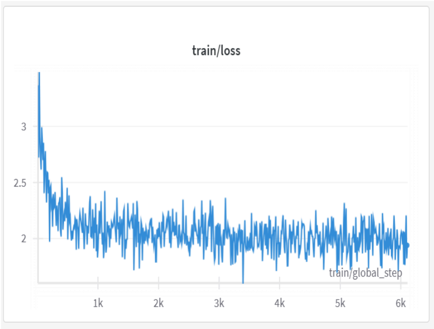
batch_size = 2, learning rate = 2e-5

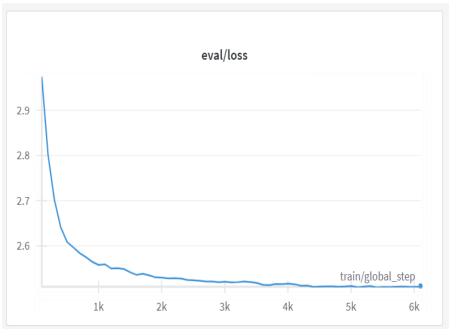




quant + LoRA

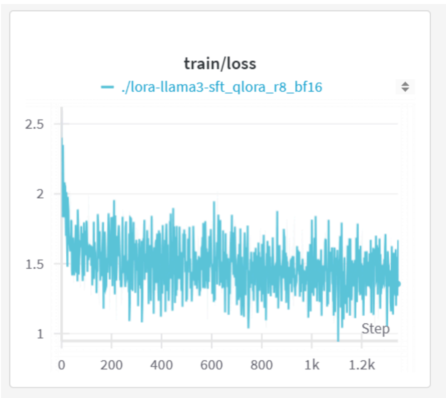
先在 load model 時 qunat 成 8-bit 再練 LoRA，參數同 LoRA

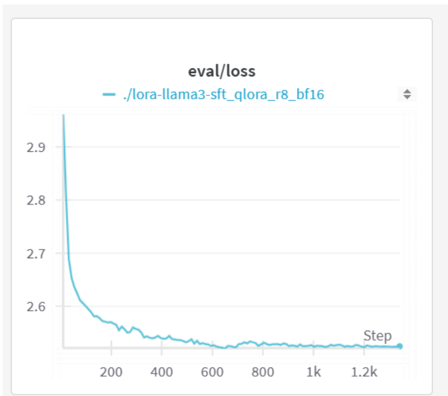




qLoRA

batch_size = 1, learning rate = 2e-5
bnb_config: load_in_4bit=True, bnb_4bit_quant_type="nf4",
bnb_4bit_compute_dtype="bfloat16", torch.float16, llm_int8_threshold=6.0,





KV cache

我們主要在 generate function 中實作 KV cache，將 cache_implementation 設為 static 且 max_cache_size 設為 input_ids.shape[1]+max_new_tokens+16。
max_cache_size 若設定的太小會早成 cache 效果不好，若設定的太大會導致overflow 或者浪費空間。
除此之外，past_key_values 用來設定 KV cache 的初始值並在 prefilled 階段做使用，如此一來就可以在每次 generate 時只計算新的 token 的 attention
拿到 output 後，手動將 logit 設為 output.logits[:,-1,:] 取最後一個 token 的 logits 可以加快速度
最後在 decoding phase 時，每次生成一個 token 並同時更新 past_key_values，除此之外，我們會使用告訴 KV cache 目前序列的最後一個 token 的位置以及這一輪要存入 KV Cache 的位置，能幫助 model 精準儲存與檢索。

HQQ & torch.compile

就和先前的作業一樣，我先對模型的 forward pass 進行 torch.compile。

```
model.generation_config.cache_implementation = "static"
model.forward = torch.compile(model.forward, mode="reduce-overhead", fullgraph=True)
```

接著再用 HQQ 來 quantize 模型，其中 q_config 的 nbits=4，group_size=128，並且 quantize 模型的 "q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj" 這幾層。

```python
# TODO: Quantize
quant_config = get_quant_config_slm(model)
AutoHQQHFModel.quantize_model(model, quant_config=quant_config, compute_dtype=torch.float16, device=device)

prepare_for_inference(model, backend=backend)
torch.cuda.empty_cache()
torch.cuda.ipc_collect()
```

Experimental Results - Screenshot your results

Throughput: 76.67, PPL: 10.39

```
Throughput: 76.67294299739254 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 [17:13<00:00,  7.33s/it]
Perplexity (PPL): 10.39219856262207
```

Kaggle rank: 32

32 37 76.700 4 3d

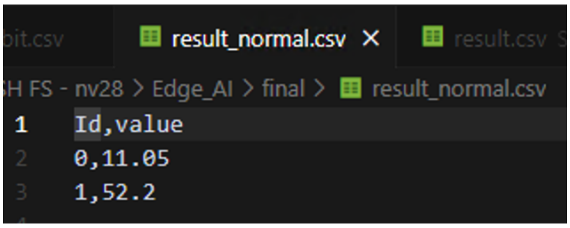
Team Member Contributions

Name	江尚軒	蔡昀叡	陳建樺	李任本耀
Contributions	HQQ & torch.compile	KV Cache	LoRA model training	EGALE & vLLM
Percentage	25	25	25	25

Insights or Explorations

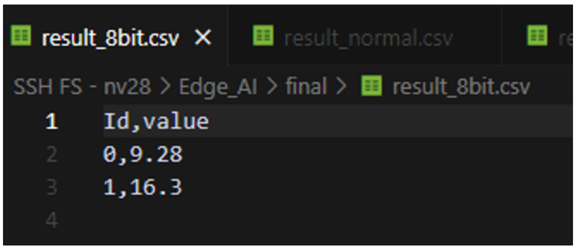
Different eval on A6000

Base model

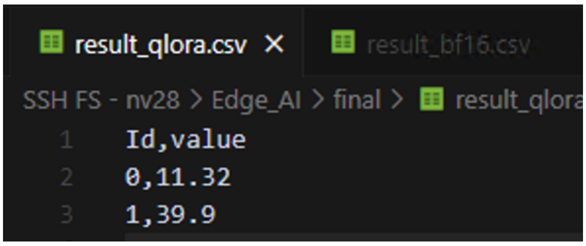


LoRA (throughput 維持，PPL下降了)

quant + LoRA (雖然 PPL 下降，但 throughput 下降很多)



qLoRA (PPL 差不多，但 throughput 下降了)



KV cache implementation (Eval on A6000)

如下表，KV cache並沒有使 PPL 上升，只對 Throughput 有增益，最終可以獲得 80.46% 的增益

	Base	Max cache size = Max new token * 1.5	Max cache size = Max new token * 2	Max cache size = input_ids.shape[1]+max_new_tokens+16, 只計算最後一個 token 的 logits
PPL	11.05	11.05	11.05	11.05
Throughput	52.2	93.45	93.45	94.2
Improvement (%)	0.00	79.02	79.02	80.46

HQQ

我也嘗試過把 HQQ 的 nbits=8，但 throughput 沒有太大的提升。
然而，當我把 nbits=2 時，PPL 則會直接炸開。
因此，我最後選擇使用 nbits=4 當成折衷方案。

關於 Group size，我們發現調大也有助於提升 throughput。我們從 group_size=64 一路往上測試，throughput 均有提升，調到 group_size=512 是最適合的大小，如果再往上調就沒有甚麼幫助了。

- Base model, group_size = 512

```
Throughput: 73.37621633689942 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 [13:02<00:00,  5.55s/it]
Perplexity (PPL): 12.684470176696777
```

- LoRA model, group_size = 512

```
Throughput: 76.67294299739254 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 [17:13<00:00,  7.33s/it]
Perplexity (PPL): 10.39219856262207
```

- LoRA model, group_size = 1024

```
Throughput: 76.56201426294241 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 [17:00<00:00,  7.24s/it]
Perplexity (PPL): 10.779273986816406
```

Speculative Decoding with EAGLE

Plan

1. Validate every experiment on an RTX A6000 first.

2. Once I confirm throughput gains, port the same setup to the T4 (final project target).
3. Ignore perplexity for now; if it slips below spec, we'll plug in my teammate's LoRA afterwards.

Baseline reproduction (A6000)

- Original code → throughput **50**
- OUR_BEST → **161.9**

Configuration	Quantization	Tokens / s
Baseline	FP16	50
OUR_BEST	HQQ 4b	161.9
Baseline + EAGLE	FP16	60
EAGLE + bitsandbytes	4-bit	13
EAGLE + bitsandbytes	8-bit	75
EAGLE + HQQ	4-bit	60

Takeaway

bitsandbytes' 4-bit path is much slower than HQQ with Gemlite; likely because Gemlite has Triton-level kernels while bitsandbytes does not.

vLLM

Best so far (A6000) — throughput **196** using "mobiuslabsgmbh/Llama-3.2-3B-Instruct_4bitgs64_hqq_hf" (already quantized with HQQ to 4-bit).
T4 limitation — same model crashes:
ValueError: The quantization method hqq is not supported for the current GPU. Minimum capability: 80. Current capability: 75.
Tried to quantize with HQQ, the job crashes during the **save_to_safetensors ()** call:
AutoHQQHFModel.save_to_safetensors(model, save_dir)
# hqq/models/base.py, line 580
# total_size += tensors[key].numel() * tensors[key].element_size()
#   ^^^^^
AttributeError: 'int' object has no attribute 'numel'