Edge LLM - Reducing LLM Memory Footprint to <2GB

STUDENTS: Chaitanya K Mullapudi, Delilah Yan, James Yao, Sai Jayanth Kalisi, Lucas Xie, Vincent Wang, Yao Zhang

Objective

Large Language Models (LLMs) simplify life by retrieving information, serving as personal assistants, improving learning and accessibility, and increasing productivity. However, LLMs' enormous size and RAM utility makes storage and inference possible using only the most powerful computers. This fosters privacy, security, and judicial concerns about accessing LLMs via the internet.

We aim to reduce the size of LLMs with 7 billions parameters from 30GB to <2GB using combinations of Quantization, Pruning and Knowledge Distillation. We benchmark how these methods interact with our model on various processing units of our OrangePi-5 to demonstrate the viability of size-reduction of LLMs, with minimal accuracy drops.

Approach

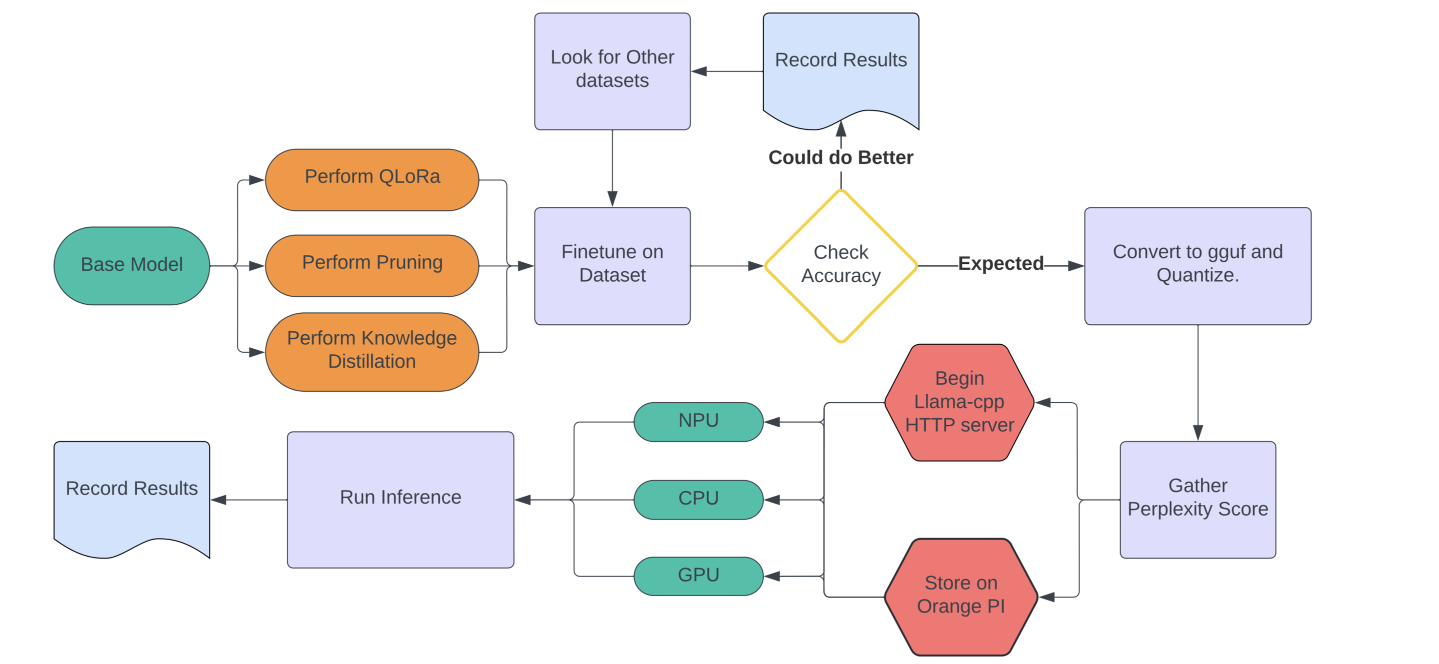

Image 1: A flowchart depicting the continual updating of the model and evaluating the changed version for both accuracy and speed

We hope to reduce LLM size without sacrificing much accuracy or speed of inference.

Hardware

- Orange pi 5 with Rockchip RK3588S new generation 8-core 64-bit processor, integrated ARM Mali-G610 GPU, built-in NPU with 6Tops computing power.
- Inference speed is determined by throughput in the form of seconds per token.
- Generally, the NPU is faster than the GPU which is faster than the CPU. This holds true because both NPU and GPU are designed for higher throughput

Model in gguf format	Size (GB)	Output Length	NPU avg s/token	GPU avg s/token	CPU avg s/token
LLama.cpp	7.5	256	0.484	1.1468	2.296
llama-7b-qa-openass-9k-max-len-1024-q8	6.67	256	6.877	1.797	1.3777
Short-GPT-30-percent-lora-q2_K	1.86	1288	0.1684	0.167	0.682
Short-GPT-25-percent-lora-2-q2_K	1.94	1024	0.1873	x	0.854
llama-2-oasst1-9k-max-len-1024-v2.0_Q2_K	2.4	1024	0.3851	0.273	1.25
llama-2-oasst1-9k-max-len-1024-v2.0_Q3_K_M	3.1	1024	0.2563	0.299	1.1527
llama-2-oasst1-9k-max-len-1024-v2.0_Q4_K_M	3.8	1024	0.2435	0.27	1.115
llama-2-oasst1-9k-max-len-1024-v2.0_Q5_K_M	4.5	1024	0.307	0.345	1.311
llama-2-oasst1-9k-max-len-1024-v2.0_Q8_0	6.7	1024	0.352	0.386	1.273
TinyLlama-ggml-model-Q4_K_M	0.69	1024	0.04036	x	0.1812

Table 1: The results of hardware inference per model on each processing unit.

Methodologies

Quantized Low Rank Adapters (QLoRA)

1. QLoRA is a Parameter Efficient Fine-Tuning technique to fine-tune LLMs without utilizing much computational resources.
2. Applies two-level quantization by applying 4-bit NormalFloat (NF4) quantization and then using LoRA to fine-tune the model.
3. After fine-tuning using LoRA, the adapter layers are then merged with the base model by adding the learned weights.
4. Beside is a overview of working of QLoRA module during training.

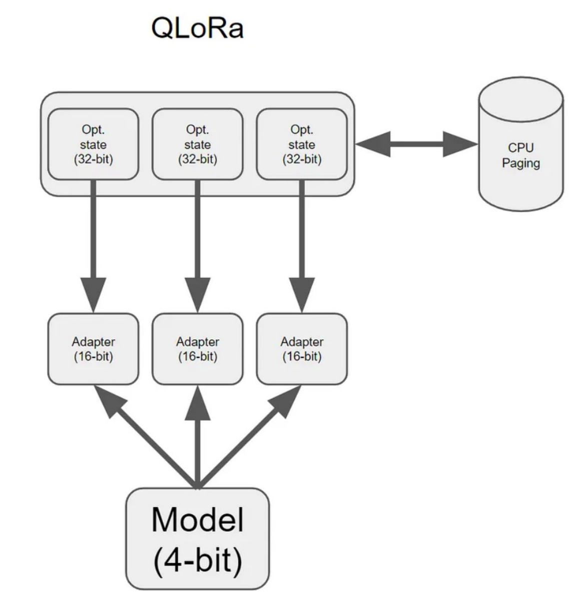

Image 2: QLoRA block diagram

Quantization & Pruning

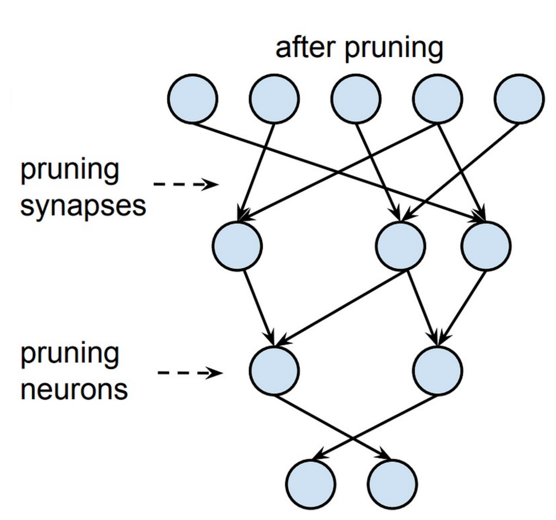

Image 3: The different ways a model can be pruned

Quantization

1. GPTQ & AWQ: GPTQ accelerates OBS by randomly selecting weights to quantize, avoiding the time-consuming greedy approach, while AWQ focuses on activation distribution to identify crucial weights for model performance.

Pruning

1. SparseGPT & Wanda (*Dataset:* OpenAssistant): Reduces model complexity through the selective removal of trivial weights, setting certain weights to zero. This helps with hardware acceleration.
2. ShortGPT (*Dataset:* Tulu-v2 (ShareGPT, Stanford_Alpaca_Data, Open-Assistant)): Directly removes layers based on dataset importance, reducing model size and restoring its capabilities with further LORA processing.

Knowledge Distillation

Two distinct approaches for knowledge distillation: task-specific and task-agnostic. The former relies on labeled data to transfer task-specific knowledge, while the latter employs unsupervised learning for distillation, followed by model fine-tuning.

1. Task-specific
Dataset: ag_news / For finetuning: open-assistant and dolly-15k.
Teacher: tulu-7b
Student: TinyLlama-1.1B-Chat-v0.4

2. Task-agnostic
Dataset: babyLlama-10M, babyLlama-10M-dev
Teacher: Llama-2 360M, GPT-2 1.1B
Student: BabyLlama-58m

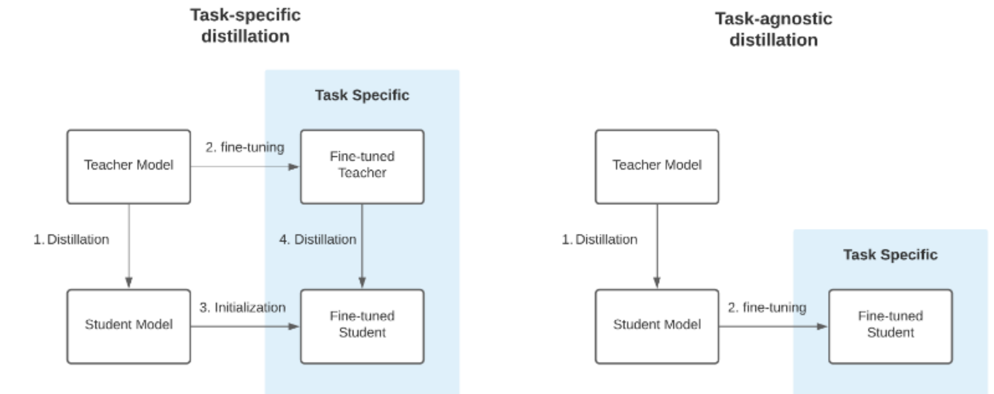

Image 4: The difference between two main Knowledge distillation methods

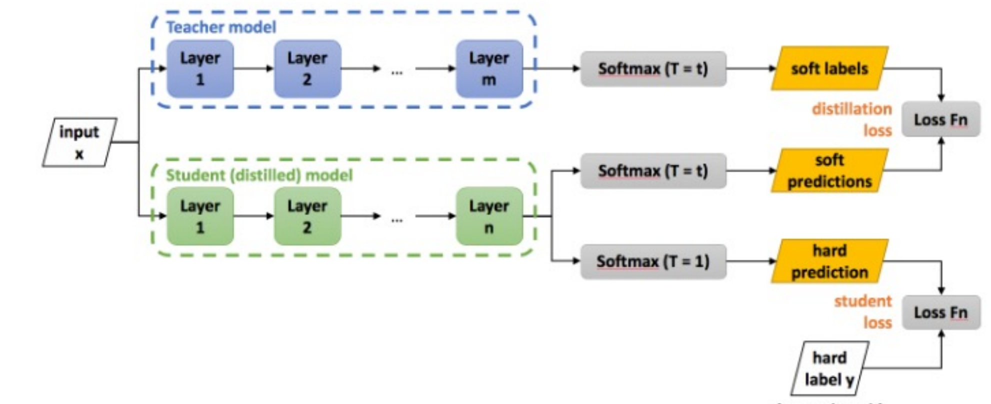

Image 5: The basic training path using Knowledge Distillation. This was used in Baby Llama/Task agnostic methods

Results

Method	MODEL	BERT (in %)			ROUGE (in %)	BLEURT (in %)	Size in GB (gguf 4-bit)
		Precision	Recall	F1	Rouge-1	Scores	
Base	Original LLaMA2-7B	86.06	85.14	85.59	38.35	32.84	13
QLoRA	LLaMA2-oasst-9k-1024-default	83.62	82.33	82.97	24.94	27.88	3.8
	Gemma-2B-it-oasst-12k-512-default	82	81.95	81.97	21.99	31.05	1.52
	LLaMA3-8B-it-oasst-12k-512-default	82.64	83.63	83.13	27.00	37.07	4.58
Pruning	LLaMA2-GPTQ-oasst	**85.57**	85.02	**85.30**	**37.48**	31.67	3.56
	LLaMA2-GPTQ+LoRA-wikitext2	85.22	84.73	84.97	36.48	30.69	3.56
	ShortGPT-25-percent-lora-5K	81.67	82.01	81.84	21.30	**45.92**	2.90
	ShortGPT-25-percent-lora-10K	81.99	82.27	82.13	21.99	45.83	2.90
Knowledge Distillation	Tinyllama-oasst-1k	83.49	**85.39**	84.43	29.01	41.09	**0.637**
	TinyLLaMA-distilled-oasst-15k	78.35	83.55	80.86	19.53	37.57	0.637
	Baby-LLaMA	73.21	75.67	74.23	25.94	28.54	0.298
	BabyLLaMA-distilled	74.10	79.12	76.52	9.06	24.60	0.222

Table 2: Results of our different size reduction methods based on accuracy metrics like Bert, Rouge, and Bleurt, with the size of each model

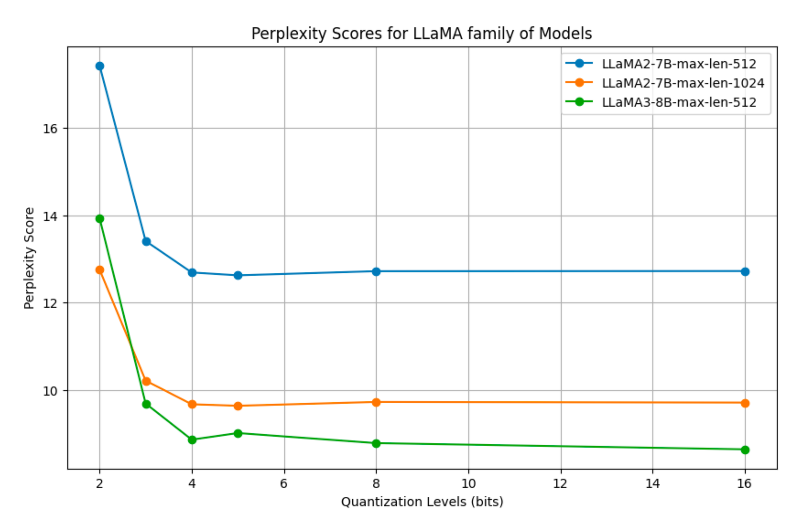

Image 6: Perplexity Scores for LLaMA family of Models

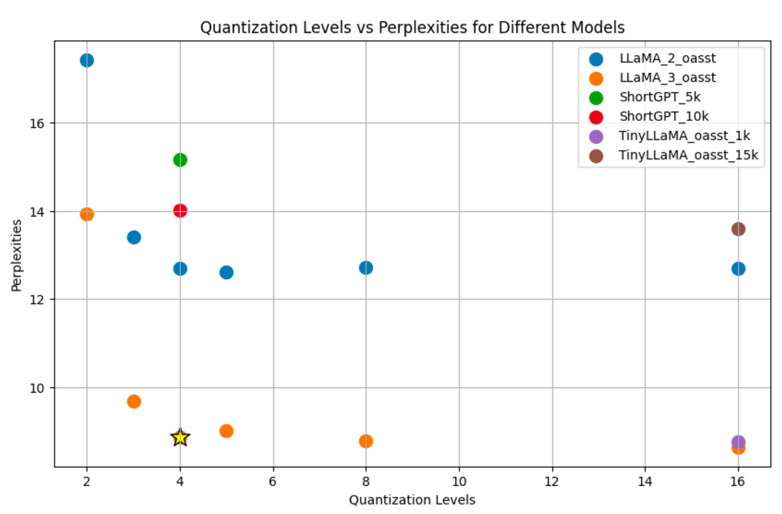

Image 7: Perplexity Scores for different quantized models

Conclusion

General Trends:

- **Hardware Throughput:** Models performed significantly better using NPU while CPU was the worst choice
- **Accuracy:** Most models have a higher BERT score followed by BLEURT and ROUGE

Best Performing Models:

- **Hardware Throughput:** Tiny Llama model (using Knowledge Distillation)
- **Accuracy:** GPTQ models (using Quantization)
- **Perplexity:** QLoRA models
- **Size:** Baby Llama (using Knowledge Distillation)

Each technique has its own merits and demerits and edge devices need to prioritize LLM's use case before employing these methods. Smaller models generally had faster throughputs, usually with less accuracy.

Future Work

- Different use-cases math & coding.
- LLM inference on general hardware
- Further reduce inference Latency
- General rescaling process for any other pre-trained model, such as Mistral, OpenELM, Llama3.

References:
[1] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer., QLoRA: Efficient Finetuning of Quantized LLMs, Advances in Neural Information Processing Systems, 2024.
[2] EJ Hu, Y Shen, P Wallis, Z Allen-Zhu, Y Li, S Wang, L Wang, W Chen., LoRA: Low-Rank Adaptation of Large Language Models, arXiv preprint arXiv:2106.09685, 2021.
[3] E Frantar, S Ashkboos, T Hoefler, D Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers, arXiv preprint arXiv:2210.17323, 2022.

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY of WASHINGTON

ADVISERS: Mansi Manohara, Mahdi Heydari, Sankalp Dayal, Larry Arnstein, Amisha Somaiya

SPONSOR: Amazon