Advancing Model Refinement: Muon-Optimized Distillation and Quantization for LLM Deployment

Anonymous authors
Paper under double-blind review

Abstract

Large Language Models (LLMs) enable advanced natural language processing but face deployment challenges on resource-constrained edge devices due to high computational, memory, and energy demands. Optimizing these models requires addressing three key challenges: acquiring task-specific data, fine-tuning for performance, and compressing models to accelerate inference while reducing resource demands. We propose an integrated framework combining GPTQ-based quantization, low-rank adaptation (LoRA), and a specialized data distillation process to significantly reduce model size and complexity while preserving or enhancing task-specific performance. By leveraging data distillation, knowledge distillation via Kullback-Leibler divergence, Bayesian hyperparameter optimization, and the Muon optimizer, our framework achieves up to 2× memory compression (e.g., reducing a 6GB model to 3GB) and enables efficient inference for specialized tasks. Empirical results demonstrate the framework's superior performance on standard LLM benchmarks compared to GPTQ quantization alone, with the Muon optimizer notably enhancing fine-tuned models' resistance to accuracy decay during quantization.

1 Introduction

Recent advancements in Large Language Models (LLMs) have seen their rise across industrial and government applications. Billions of dollars are now spent on LLM training and inference each year across data centers and edge devices, enabling diverse end-users to complete an ever-expanding array of tasks. However, the size and expense of these models strains the current infrastructure. A perennial question remains: how should researchers and engineers optimize models for efficient inference on specific user tasks?

The inference challenge is compounded on edge devices, where the local hardware imposed strict memory, energy, and network constraints. Authors have implemented a variety of solutions to shrink model memory requirements, reduce compute required during inference, and accelerate inference speed; many of these methods have deep roots in the field LeCun et al. (1989); Hinton et al. (2015); Gray and Neuhoff (1998). Model pruning, low-rank approximation, quantization, and distillation all form a family of core model compression methods that decrease memory requirements Sander et al. (2025). It remains a continual challenge to effectively integrate novel techniques in each family of methods with each other as the field matures Zheng et al. (2025); Dong et al. (2025).

To address these limitations, a sophisticated and holistic optimization strategy is required to substantially reduce the model size and computational complexity of LLMs while preserving their performance on specific tasks. This paper introduces a novel framework designed to enable the efficient deployment of LLMs on resource-constrained edge platforms. By integrating advanced compression techniques, such as quantization and low-rank adaptation, with specialized data distillation processes, the framework achieves significant reductions in model size and computational requirements without compromising performance. This approach specializes a model for a specific task, boosting the final accuracy of the compressed model. The following sections detail the framework, its components, and empirical evaluations demonstrating its effectiveness in optimizing LLMs for edge deployment in specific domains.

2 Background and Related Work

We review key techniques central to model compression pipelines for large language models (LLMs). These include pruning, quantization, and low-rank approximations, knowledge distillation for performance enhancement, and parameter-efficient fine-tuning via Low-Rank Adaptation (LoRA). We also discuss synthetic data generation, which supports distillation and training in resource-constrained settings. Finally, we discuss the recent Muon optimizer, which we integrate into the framework.

2.1 Model Compression

Model compression schemes aim to reduce the size, inference latency, and computational footprint of LLMs while preserving performance. Common techniques include pruning, low-rank approximations, and quantization. Sander et al. (2025) Pruning removes redundant weights or neurons, often guided by magnitude-based criteria or advanced methods like the Lottery Ticket Hypothesis, which identifies sparse subnetworks that match full model accuracy. Frankle and Carbin (2019) Recent advancements, such as SparseGPT Frantar and Alistarh (2023), enable one-shot pruning for generative models without extensive retraining.

Low-rank approximations decompose high-dimensional weight matrices into lower-rank factors, reducing parameters without full retraining. By subsequently deleting columns corresponding to small singular values from the approximation, authors decrease the number of parameters while preserving performance. Techniques that utilize SVD-based compression include Fisher-Weighted SVD Hsu et al. (2022) and SVD-LLM Wang et al. (2024) Wang et al. (2025), leveraging SVD for efficient model compression.

Quantization reduces the precision of model weights and activations (e.g., from FP16 to INT4), significantly lowering memory usage. Post-training quantization methods like GPTQ Frantar et al. (2023)–adopted as one component of our pipeline–use Hessian-based approximations to minimize reconstruction error introduced by quantization in LLMs. CALDERA Saha et al. (2024) combines quantization with low-rank adaptation, achieving efficient compression while preserving model performance through joint optimization. Other methods, like Activation-aware Weight Quantization (AWQ) Lin et al. (2024), prioritize salient weights to maintain output fidelity.

2.2 Distillation

Knowledge distillation (KD) transfers knowledge from a large "teacher" model to a smaller "student" model to improve performance on downstream tasks Hinton et al. (2015). By using soft labels (probability distributions) from the teacher, KD provides both regularization and more informative labels, aiding generalization.

We employ logit-based distillation, where the student (S1) output is aligned to the teacher (T2) output via KL divergence. Anthropic's recent work Cloud et al. (2025) explores information transfer in KL-divergence distilled models of similar architectures, showing that additional information, potentially even outside the training data distribution, can be transferred by KL Divergence. Synthetic data generation enhances KD by creating diverse teacher-annotated datasets. A notable framework in synthetic data generation is the self-instruct pipeline Wang et al. (2023), where LLMs bootstrap their own training data.

2.3 Low-Rank Adaptation

Low-Rank Adapters (LoRA) are a parameter-efficient fine-tuning (PEFT) method that adds trainable low-rank matrices into linear layers, while freezing original weights. Hu et al. (2021) LoRA leverages the intrinsic low-rank properties of weight updates during fine-tuning. Hu et al. (2021) Variants like QLoRA Dettmers et al. (2023) integrate quantization of the weights for lower memory footprints with full-precision adapters. In distillation pipelines, LoRA complements KD by enabling student models to adapt to teacher-generated data without overfitting or catastrophic forgetting, as supported by frameworks like Hugging Face's PEFT library.

2.4 Muon Optimizer

A recent advancement in the optimization of 2d tensors, Muon factors the gradients of 2d layers approximately using Newton-Schulz and then performs descent over the spectral norm of each layer Jordan et al. (2024). By eliminating the step in the noisy principal direction, Muon accelerates learning, achieving 30-40% reduction in training time and tokens required for small models. Further research has shown the solutions that Muon finds are quantitatively different than Adam-optimized solutions, with a lower population of outlier channel activations Park et al. (2025). Authors find that a smaller domain of channel activations decreases rounding errors during quantization, subsequently increasing compressed model accuracy and perplexity. Park et al. (2025)

Other authors show a more complicated picture when Adam pre-trained models are used with Muon fine-tuning, and vice versa. They find that depending on which benchmark is tested, Adam pre-training paired with Muon fine tuning gives suboptimal accuracy. Liu et al. (2025)

In addition, we find logit-based distillation losses used in combination with Muon an underexplored topic, with only a single paper on distillation of specific latent features in the vision context. Chen et al. (2025)

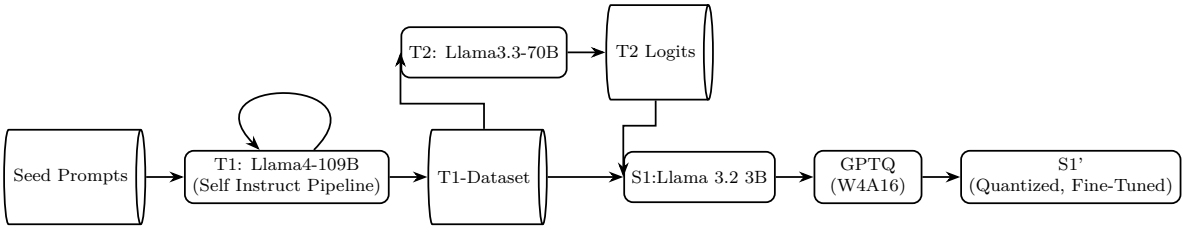

Figure 1: Pipeline Overview

3 Methodology

We present an end-to-end framework for compressing a Large Language Model (LLM) and specializing it for a specific task while maintaining performance on resource-constrained edge devices. The framework integrates knowledge distillation, data distillation, low-rank adaptation (LoRA), GPTQ-based compression, and Bayesian hyperparameter optimization, as illustrated in Figure 1. The framework provides a consistent way to fine-tune a compact student model (S_1) using knowledge distillation from a knowledge teacher model (T_2), on a task-specific dataset generated through data distillation using a high-performance teacher (T_1), in conjunction with quantization, and hyperparameter optimization to optimize the S_1 for edge deployment.

3.1 Framework

To create an efficient, task-specialized model, we first choose a compact student model, S_1, and a teacher model, T_2, both sharing the same tokenizer to minimize distribution shift during knowledge distillation, as highlighted by Boizard et al. (2025). The shared tokenizer ensures that the vocabulary spaces of T_2 and S_1 are aligned, reducing discrepancies in their output distributions that could inflate the Kullback-Leibler Divergence (KLD) loss, which measures the difference between their probability distributions. This alignment is critical for effective knowledge transfer, as unaligned tokens could lead to mismatched logit distributions, increasing $\mathcal{L}_{\text{KL}}$ and degrading performance. For our experiments, we select Meta's Llama 3.1 70B Instruct model as T_2 and Llama 3.2 3B Instruct as S_1 (AI, 2024a). The compact size of S_1 enhances resource efficiency for edge deployment, while T_2's high performance on the target task provides robust, learnable information encoded in the model's output logits.

Our knowledge distillation employs Low-Rank Adaptation (LoRA) (Hu et al., 2021) for parameter-efficient fine-tuning of S_1. LoRA introduces trainable low-rank matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$ (where rank $r \ll \min(d,k)$, typically $r = 8$–64) as adapters to the frozen pre-trained weights $W \in \mathbb{R}^{d\times k}$, updating them as $\Delta W = BA$. This optimizes only a small fraction of parameters, reducing memory and computational costs

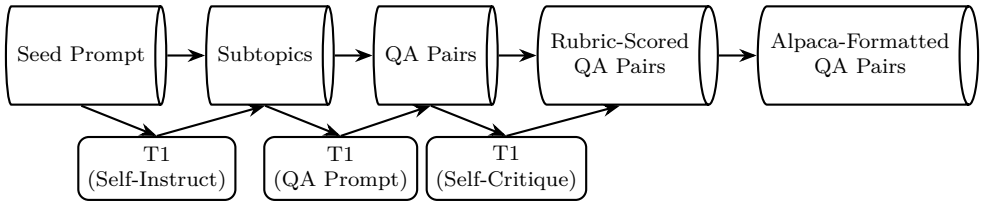

Figure 2: Self-Instruct Pipeline Detail

compared to full fine-tuning, while mitigating overfitting and achieving performance comparable to full-rank methods.

The distillation process minimizes a combined loss function balancing Cross Entropy (CE) loss and KLD between T_2 and S_1 logits. The CE loss is:

$$\mathcal{L}_{\text{CE}}(y_b, p_S) = -\sum_i y_b(i) \log p_S(i), \tag{1}$$

where y_b is the ground-truth label, and p_S is the student's softmax output. The softmax outputs for T_2 and S_1 are:

$$p_T(i) = \frac{\exp(t_i/T)}{\sum_j \exp(t_j/T)}, \tag{2}$$

$$p_S(i) = \frac{\exp(s_i/T)}{\sum_j \exp(s_j/T)}, \tag{3}$$

where t_i and s_i are the logits of T_2 and S_1, respectively, and T is the distillation temperature. The KLD loss, which benefits from the shared tokenizer to ensure p_T and p_S operate over the same vocabulary space, is:

$$\mathcal{L}_{\text{KL}}(p_T, p_S) = T^2 \sum_i p_T(i) \log \frac{p_T(i)}{p_S(i)}, \tag{4}$$

and the combined loss is:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{KL}}(p_T, p_S) + (1 - \alpha)\mathcal{L}_{\text{CE}}(y_b, p_S), \tag{5}$$

where $\alpha \in [0, 1]$ balances the two terms. We perform a single epoch of fine-tuning with a 16-sample Bayesian Hyperparameter Optimization (HPO) to optimize α, LoRA rank, LoRA scaling, distillation temperature, weight decay, and learning rate. As one of our experimental conditions, we use either Adam (Kingma and Ba, 2017) or Muon (Jordan et al., 2024) as the optimizer. Accuracy results are labeled by optimizer in Table 3. After fine-tuning, the LoRA is merged into S_1.

While T_2 is effective for task-specific knowledge transfer with its aligned vocabulary to S_1, a suitable task-aligned dataset is still required for fine-tuning via distillation. To address scenarios where available task-specific data is sparse, we introduce a first teacher model, T_1, which can be a more powerful open or closed-source model with a broader data distribution, including more tail cases specific to the task, to generate a high-quality, task-aligned dataset via a self-instruct pipeline. For our experiments, we utilize Meta's Llama 4 Scout 109B as T_1 (AI, 2024b). Using the Self-Instruct (Wang et al., 2023) and Distilabel (Álvaro Bartolomé Del Canto et al., 2024) frameworks, T_1 generates a task-specific dataset with vLLM, using fixed inference parameters (temperature 0.7, top-p sampling 0.95) for consistent and diverse outputs. The pipeline starts with seed prompts containing task-specific keywords (e.g., 'Astronomy' or 'Virology' for MMLU), which T_1 uses to generate k number of instruction prompts for sub-topics (e.g., 'black hole formation' or 'viral replication cycles'). T_1 then creates question-answer pairs for each instruction prompt to cover the task domain comprehensively, ensuring tail cases improve generalization during recovery training. T_1 also evaluates the generated pairs using a scoring rubric to discard poorly formed entries. We target 600 question-answer pairs for each dataset, losing less than 10% in each task due to attrition from self-critique. The dataset is formatted in Alpaca format (Taori et al., 2023) to preserve S_1's instruction tuning, as shown in Figure 2.

The dataset is tailored to specific tasks to ensure S_1''s specialization. We select 8 benchmarks: MMLU, assessing knowledge and reasoning across 57 academic disciplines; ARC-e, testing commonsense reasoning with easy science questions; CommonsenseQA, evaluating textual commonsense reasoning; HellaSwag, benchmarking grounded commonsense inference in physical or social scenarios; OpenBookQA, measuring advanced scientific reasoning; PIQA, testing physical commonsense reasoning; SIQA, evaluating social intelligence; and WinoGrande, testing pronoun resolution in challenging contexts. These benchmarks, their respective abbreviations, and their respective citations are given in the appendix in table 2. For each task, we selected 20 key phrases that span the space of topics in the benchmark. For example, Commonsense Question Answer requires a model to be an expert in "social interactions", "physical causality", "object properties and uses," and other general skills. These key phrases are inserted into a seed prompt template, which are used to guide T_1's Self-Instruct data distillation, in turn producing a robust dataset for S_1's fine-tuning.

To enable edge deployment, we compress S_1' using GPTQ 4-bit quantization (Frantar et al., 2023), producing the final model S_1''. GPTQ, a state-of-the-art method, minimizes quantization error compared to alternatives like round-to-nearest quantization (Xiao et al., 2024), even outperforming AWQ (Lin et al., 2024) in accuracy preservation on some benchmarks. We use w4a16 quantization (4-bit weights, 16-bit activations) on linear layers, achieving approximately 2× memory compression compared to full-precision models, suitable for resource-constrained hardware (Han et al., 2016).

3.2 Model Optimization

To address the challenges of deploying LLMs on edge devices, which are limited by memory (often 1-8 GB), compute power, and energy consumption—the optimization process must balance model performance with resource constraints. These devices impose strict limitations on model size, inference latency (ideally sub-100 ms for real-time applications), and power usage, making direct deployment of massive LLMs like Llama-7B (requiring approximately 14 GB in FP16) infeasible without significant adaptations. The optimization problem incorporates non-differentiable hyperparameters, and is also highly non-convex, ruling out traditional optimization solution strategies like gradient descent. One approach involves reformulating the problem into a subproblem or series of subproblems by relaxing constraints. A series of such subproblems decompose the intractable, high-dimensional optimization problem into a sequence of lower-dimensional subproblems.

To optimize the pipeline's performance, we relax the compute power and energy constraints while retaining the memory constraint. Our performance metric we minimize is the validation loss on a 10% split of the input T1 dataset, e.g., we take maximum performance as the minimum of the loss. We employ Bayesian hyperparameter optimization using Optuna (Akiba et al., 2019), modeling the objective as a Gaussian process to efficiently search the hyperparameter space. Key hyperparameters are described in Table 1 and results summarized in Table 6 in the appendix.

$$\begin{aligned} \text{Minimize} \quad & \text{Loss}(\mathbf{a}, \mathbf{r}, \mathbf{w_d}, \eta, \mathbf{T}, \alpha) \\ \text{subject to:} \quad & h(\hat{m}) \leq \text{Memory}_{\text{budget}} \end{aligned}$$

$$\begin{aligned} \mathbf{a} \in [0.5, 2.0], \quad \mathbf{r} \in [8, 64], \quad \eta \in [1e-5, 1e-3], \\ \mathbf{T} \in [0.5, 8], \quad \alpha \in [0, 1] \quad \mathbf{w_d} \in [0, 2] \end{aligned}$$

We solve for the minima of validation loss under the following assumptions of limited discrete choices. We let Optuna iteratively samples configurations to maximize expected improvement, converging on optima with fewer evaluations than grid or random search (Snoek et al., 2012). Additionally, we run two experimental conditions. We compare the use of the Adam optimizer (Kingma and Ba, 2017) against the Muon (Jordan et al., 2024) optimizer, running full HPO under both conditions. Recall that researchers have found that Muon-pre-trained models are more robust when quantized, losing less accuracy2. If this holds true not only with pre-training but also with parameter-efficient fine-tuning, we should see a lower accuracy decrease in the Muon-optimized models. We measure the accuracies of the fine-tuned models, and their subsequently quantized versions, in table 4 and subtract the accuracy of the quantized model from the fine-tuned model to measure the decrease in accuracy due to quantization.

Variable	Description
$\hat{m}$	Vector of memory hyperparameters (weight quantization, activation quantization, parameter count)
$\theta_{\mathbf{s}}$	Student model parameters
$\mathbf{p_t}$	Teacher output probability vector
$\mathbf{p_s}$	Student output probability vector
$\mathbf{y_b}$	Supervised labels
$\mathbf{a}$	LoRA scaling
$\mathbf{r}$	LoRA rank
$\mathbf{w_d}$	Weight decay
η	Learning rate
$\mathbf{T}$	Distillation temperature
α	Distillation loss weight
$\text{Memory}_{\text{budget}}$	Constraint on max allowable memory.

Table 1: Variable Definitions

We integrate Self-Instruct using T_1's, T_2's knowledge distillation, S_1's LoRA-based fine-tuning, GPTQ compression, and Bayesian optimization with Adam and Muon to produce a compact, task-specialized model S_1'' suitable for edge deployment. This framework allows flexible selection of models, tasks, and quantization methods, minimizing compression error while maximizing performance, as validated across the selected benchmarks. We describe this pipeline algorithmically in the appendix. 1

4 Results & Discussion

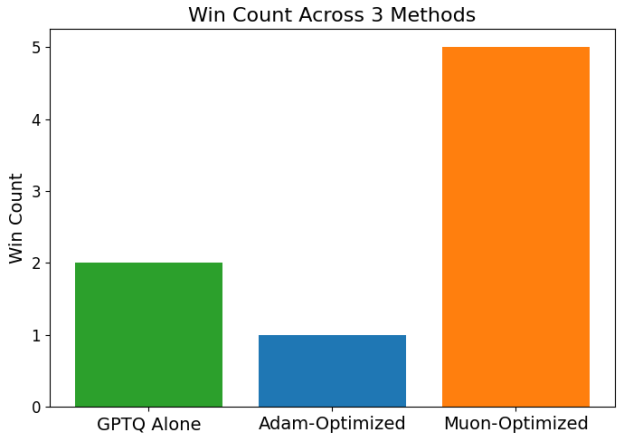

Figure 3: Win Count across 3 methods: GPTQ Alone vs Adam-Optimized vs Muon-Optimized

First, we observe that when the pipeline is used with the Muon optimizer, it surpasses the accuracy of merely GPTQ-quantized models across 5 of the benchmarks studied (see table 3 and the plot 3 above). Including the Adam variant, our pipeline has superior accuracy in 6 of 8 benchmarks compared to GPTQ alone. This represents an important, incremental improvement in the simultaneous compression and specialization of small models for specific tasks.

Second, we observe that across tasks, our hyperparameter optimization consistently chooses to drop the Cross Entropy term entirely from the loss function (see table 6). The consistent choice of distillation $\alpha = 1$ indicates that KL Divergence from T2 minimizes the training set loss, justifying our choice to add KL Divergence from a T2 teacher to the pipeline. This T2 teacher minimizes our loss on the validation set most consistently.

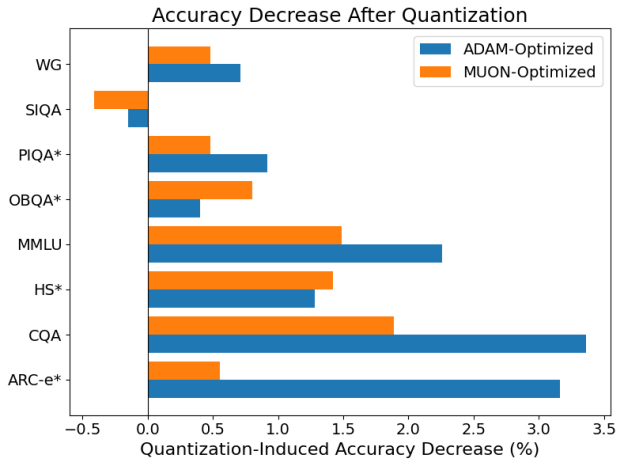

Figure 4: Comparison of Accuracy Decrease due to Quantization across 8 Benchmarks; Adam-Optimized vs Muon-optimized. Note: in SIQA benchmark, accuracy increased when quantized

Third, we compare pre- and post- quantization accuracy. We hypothesized that the use of Muon in fine-tuning may allow the model to better resist introduced quantization error, if previous authors' work Park et al. (2025) holds under our methodology that incorporates both LoRA and distillation loss. To evaluate this hypothesis, we compare the degradation of accuracy between the LoRA-tuned models and their merged, quantized versions. We find that Muon-optimized models degrade less when quantized, as hypothesized. Muon-optimized models lose less accuracy than Adam-optimized models during quantization across 6 of the benchmarks studied. Examine the ARC-e benchmark scores, for example. The Adam-optimized model loses more than 3% of its accuracy when quantized, whereas the Muon-optimized model loses only .5%. This represents an important experimental extension of the body of Muon work to fine-tuning with distillation loss and LoRA; even a single weight update after pre-training can increase the resistance of the model to quantization error. While the work so far on Muon has indicated mixed results on fine-tuning Adam-optimized checkpoints with Muon, we recommend researchers investigate the use of Muon when combined with quantization-based compression methods for fine-tuning.

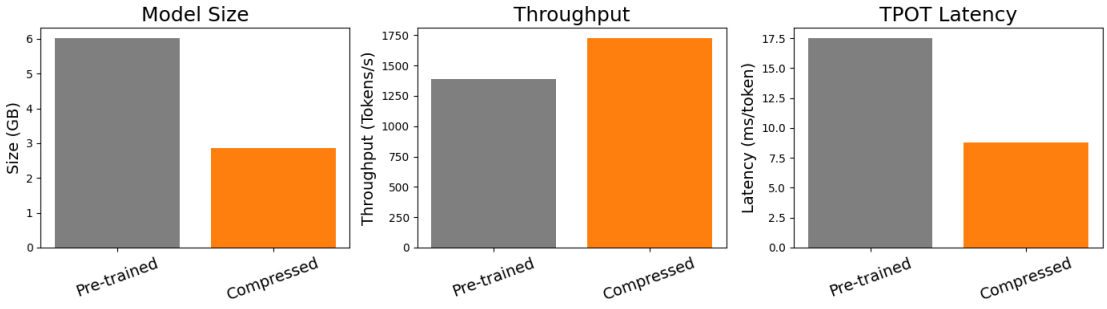

Figure 5: Throughput Comparison: Pre- and Post- Compression

Fourth, we provide vLLM-deployed throughput metrics (Time per Output Token (TPOT), Intertoken Latency (ITL), and output token Throughput) for both full-precision and W4A16 quantized models in table A of the appendix, and plotted in figure 5. We see a decrease of 50% time for TPOT and ITL, as expected. Throughput gains are more modest than TPOT and ITL suggest, despite the faster generation, due to per-sequence prefill overhead using MarlinLinearKernel in vLLM.

5 Conclusion

In this work, we introduced an end-to-end framework that simultaneously specializes and compresses small LLMs for resource-constrained edge deployment. By orchestrating (1) high-quality synthetic data generation via Self-Instruct with a powerful teacher T_1, (2) logit-based knowledge distillation from a tokenizer-aligned teacher T_2, (3) parameter-efficient fine-tuning with LoRA, (4) Bayesian hyperparameter optimization, (5) the Muon optimizer, and (6) GPTQ post-training quantization, we deliver higher accuracy than naïve GPTQ quantization alone while delivering $2\times$ memory reduction and up to 50% lower per-token latency.

Our most notable findings are twofold. First, hyperparameter optimization systematically eliminates the supervised cross-entropy term ($\alpha = 1$) across tasks, revealing that pure KL-divergence alignment with a strong teacher T_2 is the dominant signal for minimizing loss on synthetic distillation data. Second, and more importantly, we extend recent observations about Muon's quantization robustness from pre-training to the fine-tuning regime: even a single epoch of Muon-optimized LoRA fine-tuning (combined with distillation loss) yields models that degrade less under aggressive 4-bit quantization than their Adam-optimized counterparts. This effect is strong enough that Muon-optimized models achieve higher final accuracy than Adam-optimized ones on the majority of benchmarks despite identical architecture and compression level.

These results underscore a broader insight: optimizer choice is not merely a training detail but a critical design lever in the compression-aware fine-tuning pipeline. Small models destined for quantization benefit disproportionately from the Muon optimizer, which performs gradient descent under the spectral norm of weight updates.

Ultimately, our framework demonstrates that high-performing, task-specialized, edge-ready LLMs need not sacrifice accuracy for deployability when modern compression, distillation, and optimization techniques are unified under a single coherent pipeline.

References

Meta AI. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models, 2024a. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

Meta AI. Introducing llama 4: Advancing multimodal intelligence, April 2024b. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Llama 4 Maverick, Meta AI Blog.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. URL https://arxiv.org/abs/1907.10902.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. Towards cross-tokenizer distillation: the universal logit distillation loss for llms, 2025. URL https://arxiv.org/abs/2402.12030.

Weiye Chen, Qingen Zhu, and Qian Long. Muon-accelerated attention distillation for real-time edge synthesis via optimized latent diffusion, 2025. URL https://arxiv.org/abs/2504.08451.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Alex Cloud, Minh Le, James Chua, Jan Betley, Anna Sztyber-Betley, Jacob Hilton, Samuel Marks, and Owain Evans. Subliminal learning: Language models transmit behavioral traits via hidden signals in data, 2025. URL https://arxiv.org/abs/2507.14805.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. URL https://arxiv.org/abs/2305.14314.

Yanjie Dong, Haijun Zhang, Chengming Li, Song Guo, Victor C. M. Leung, and Xiping Hu. Fine-tuning and deploying large language models over edges: Issues and approaches, 2025. URL https://arxiv.org/abs/2408.10691.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1803.03635.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023. URL https://arxiv.org/abs/2301.00774.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023. URL https://arxiv.org/abs/2210.17323.

R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998. doi: 10.1109/18.720541.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016. URL https://arxiv.org/abs/1510.00149.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization, 2022. URL https://arxiv.org/abs/2207.00112.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605. Morgan Kaufmann, 1989.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024. URL https://arxiv.org/abs/2306.00978.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025. URL https://arxiv.org/abs/2502.16982.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Jungwoo Park, Taewhoo Lee, Chanwoong Yoon, Hyeon Hwang, and Jaewoo Kang. Outlier-safe pre-training for robust 4-bit quantization of large language models, 2025. URL https://arxiv.org/abs/2506.19697.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. *Advances in Neural Information Processing Systems*, 37:88981–89018, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WINOGRANDE: an adversarial winograd schema challenge at scale. *CoRR*, abs/1907.10641, 2019. URL http://arxiv.org/abs/1907.10641.

Jacob Sander, Achraf Cohen, Venkat R. Dasari, Brent Venable, and Brian Jalaian. On accelerating edge ai: Optimizing resource-constrained environments, 2025. URL https://arxiv.org/abs/2501.15014.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *CoRR*, abs/1904.09728, 2019. URL http://arxiv.org/abs/1904.09728.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms, 2012. URL https://arxiv.org/abs/1206.2944.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *CoRR*, abs/1811.00937, 2018. URL http://arxiv.org/abs/1811.00937.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*, 2024.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. Svd-llm v2: Optimizing singular value truncation for large language model compression, 2025. URL https://arxiv.org/abs/2503.12340.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023. URL https://arxiv.org/abs/2212.10560.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024. URL https://arxiv.org/abs/2211.10438.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. A review on edge large language models: Design, execution, and applications, 2025. URL https://arxiv.org/abs/2410.11845.

Álvaro Bartolomé Del Canto, Gabriel Martín Blázquez, Agustín Piqueres Lajarín, and Daniel Vila Suero. Distilabel: An ai feedback (aif) framework for building datasets with and for llms. https://github.com/argilla-io/distilabel, 2024.

A Appendix

Table 2: Eight language benchmarks considered

Task	Full Name	Reference
ARC-e	ARC (Easy)	Clark et al. (2018)
CQA	CommonsenseQA	Talmor et al. (2018)
HS	HellaSwag	Zellers et al. (2019)
MMLU	MMLU	Hendrycks et al. (2021)
OBQA	OpenBookQA	Mihaylov et al. (2018)
PIQA	Physical Interaction QA	Bisk et al. (2020)
SIQA	SocialIQA	Sap et al. (2019)
WG	WinoGrande	Sakaguchi et al. (2019)

Table 3: T1:Llama4 + T2:Llama3.3 Accuracy Comparison Table. Used to plot figure 3

Task	Llama3.2-3B	GPTQ Alone	Adam	Muon
ARC-e*	.7100	.6932	.6843	**.6999**
CQA	.7371	**.7265**	.7133	.7256
HS*	.7162	**.7064**	.7050	.7020
MMLU	.6069	.5787	.5836	**.5914**
OBQA*	.3940	.3800	**.3900**	.3720
PIQA*	.7682	.7622	.7612	**.7666**
SIQA	.4744	.4765	.4734	**.4770**
WG	.6875	.6748	.6748	**.6803**

Table 4: Difference in accuracy: Pre- and Post- Quantization for Adam and Muon optimized models. Used to plot figure 4

Task	LoRA-Adam	Quant-Adam	AdamAccDrop	LoRA-Muon	Quant-Muon	MuonAccDrop
ARC-e*	.7159	.6843	.0316	.7054	.6999	**.0055**
CQA	.7469	.7133	.0336	.7445	.7256	**.0189**
HS*	.7178	.7050	**.0128**	.7162	.7020	.0142
MMLU	.6062	.5836	.0226	.6063	.5914	**.0149**
OBQA*	.3940	.3900	**.0040**	.3800	.3720	.0080
PIQA*	.7704	.7612	.0092	.7704	.7666	**.0048**
SIQA	.4734	.4749	-.0015	.4729	.4770	**-.0041**
WG	.6819	.6748	.0071	.6851	.6803	**.0048**

Table 5: Throughput and latency comparison: Pre- and Post- Quantization. Used to plot figure 5 *Note: Metrics measured on 1x Ampere A40 GPU, deployed with vLLM, using 1000 prompts with input 1024 length, output length 1024, max concurrency 8*

Model	Size (GB)	Throughput (Tok/s)	TPOT (ms/tok)	ITL (ms/tok)
Pre-Quant	6.01	1387.64	17.49	17.54
Post-Quant	**2.86**	**1722.82**	**8.82**	**8.82**

Algorithm 1 Pipeline

Require: Teacher T, base model S, seed prompts $\mathcal{D}_{\text{seed}}$
Ensure: 4-bit quantized student S''
1: $\mathcal{D}_{\text{gen}} \leftarrow$ Self-Instruct-with-Rubric(T, $\mathcal{D}_{\text{seed}}$) ▷ seed → subtopics → questions → answers → rubric
2: Search space: $\mathbf{a}$, $\mathbf{r}$, $\mathbf{w_d}$, η, $\mathbf{T}$, α
3: $(\hat{a}, \hat{r}, \hat{w_d}, \hat{\eta}, \hat{T}, \hat{\alpha}) \leftarrow$ Optuna-HPO on validation split of $\mathcal{D}_{\text{gen}}$ ▷ using $\mathcal{L} = \alpha\, D_{\text{KL}}(T \parallel S) + (1-\alpha)\,\text{CE}$
4: $S' \leftarrow$ LoRA-finetune(S_0, $\mathcal{D}_{\text{gen}}$, $(\hat{a}, \hat{r}, \hat{w_d}, \hat{\eta}, \hat{T}, \hat{\alpha})$)
5: $\mathcal{D}_{\text{calib}} \leftarrow$ subsample 128 sequences from $\mathcal{D}_{\text{gen}}$
6: $S'' \leftarrow$ GPTQ(S, 4-bit, group-size=128, calibration=$\mathcal{D}_{\text{calib}}$)
7: **return** S''

Table 6: Quasi-optimal hyperparameters. *Note: Hyperparameters optimized via Optuna HPO using 16 samples*

Task	Optimizer	Rank	LoRA Scale	LearningRate	Distill α	Distill T	W Decay	EvalLoss
ARC-e	Adam	48	0.5	3.77e-4	1.0	0.51	0.01	0.079
ARC-e	Muon	16	1.5	4.17e-4	1.0	2.01	0.02	0.950
CQA	Adam	64	2	4.99e-4	1.0	0.51	0.04	0.115
CQA	Muon	24	1.75	4.91e-4	1.0	0.51	0.1	0.115
HS	Adam	24	1.75	2.34e-4	1.0	0.51	0.08	0.146
HS	Muon	48	2	4.94e-4	1.0	6.01	0.03	1.947
MMLU	Adam	24	1.0	7.56e-4	1.0	0.51	0.03	0.173
MMLU	Muon	32	1.0	5.92e-5	1.0	1.01	0.09	0.389
OBQA	Adam	48	1.75	1.35e-4	1.0	0.51	0	.110
OBQA	Muon	56	2	4.96e-4	0.7	0.51	0.07	1.716
PIQA	Adam	8	0.5	1.22e-4	1.0	6.01	0	2.666
PIQA	Muon	32	2	2.61e-4	1.0	6.51	0.07	2.106
SIQA	Adam	56	1.75	1.79e-4	1.0	6.51	0.08	2.258
SIQA	Muon	24	2	3.73e-4	0.5	0.51	0.07	1.716
WG	Adam	24	1.5	3.42e-4	1.0	8.01	0.03	1.844
WG	Muon	24	1.75	3.22e-4	1.0	6.01	0.01	2.049

Table 7: Accuracies of Pre-Trained Models T1 (Llama4-Scout), T2 (Llama3.3-70B-Instruct), and S1 (Llama 3.2 3B Instruct)

Task	Llama4Scout	Llama3.3 70B	Llama3.2 3B
ARC-e*	.8384	.8283	.7100
CQA	.8239	.8346	.7371
HS*	.8265	.8519	.7162
MMLU	.7987	.8221	.6069
OBQA*	.4580	.4640	.3940
PIQA*	.8090	.8471	.7682
SIQA	.4744	.5307	.4744
WG	.7545	.8311	.6875