A Lightweight LLM as a STEM Study Assistant

Sébastien Delsad | 326423 | sebastien.delsad@epfl.ch
Bryan Gotti | 325403 | bryan.gotti@epfl.ch
Lindsay Bordier | 325529 | lindsay.bordier@epfl.ch
Alexandre Huou | 342227 | alexandre.huou@epfl.ch
TPU Burners

Abstract

We present a domain-adapted LLM for answering STEM multiple-choice questions. We explore fine-tuning Qwen3-0.6B-Base using SFT, Direct Preference Optimization (DPO), retrieval-augmented generation (RAG), quantization, and dataset filtering. Combining alignment, retrieval, and efficiency techniques, our approach improves MCQA accuracy while enabling resource-efficient deployment.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing tasks. However, they are still hard to deploy in resource-limited, high-stakes areas like higher education, where accuracy and efficiency are key.

This project fine-tunes Qwen3-0.6B-Base to answer STEM multiple-choice questions for EPFL students. Challenges include limited model capacity, technical question complexity, and deployment constraints.

We use quantization, retrieval augmentation, and Direct Preference Optimization to boost performance, alignment, and responsiveness, building a practical student assistant and a case study in small-model adaptation.

2 Approach

2.1 DPO

When fine-tuning the base model Qwen3-0.6B-Base into a DPO model, we use the original DPO loss (Rafailov et al., 2024) given by the negative expectation of:

$$\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\mid x)}{\pi_{\text{ref}}(y_w\mid x)}-\beta\log\frac{\pi_\theta(y_l\mid x)}{\pi_{\text{ref}}(y_l\mid x)}\right)\right].$$

Here, π_θ denotes the target policy (the model being trained), and π_{ref} represents the reference policy (typically a baseline model). Each data sample consists of (x, y_w, y_l), where x is the input prompt, y_w is the preferred (winning) output, and y_l is the less preferred (losing) output. The hyperparameter β controls the sensitivity to deviation from the reference model.

2.2 MCQA

The MCQA model is trained using Qwen3-0.6B-Base as a base model. Two main methods are compared for the training of the MCQA model: LoRA (Hu et al., 2021) and full supervised fine-tuning (SFT). Distilling Chain-of-Thought reasoning is not be explored due to the limited compute resources available.

Training is done via teacher-forcing, with a maximum likelihood loss defined as:

$$\mathcal{L}_{\text{MCQA}} = -\sum_{t=1}^{T}\log P(y_t^*|y_{<t}^*),$$

where y_t^* are the gold tokens from the training sequences, and $P(x|y)$ is the model policy.

To investigate different dataset sizes and configurations, we employ a filtering method based on embedding similarity. The core idea is to select training samples that are more similar to a target dataset, with the goal of enhancing performance on that target. Details of the embedding similarity computation are provided in Appendix A.3.

2.3 Quantization

To improve inference efficiency, we evaluate three quantization strategies: weight-only Post-Training Quantization (PTQ), activation-aware PTQ, and Quantization-Aware Training (QAT). Our focus is on comparing MCQA accuracy and memory usage trade-offs across these methods on MCQA.

We compare two weight-only quantization methods: BitsAndBytes (BnB) (Dettmers et al., 2022), which applies block-wise 4-bit quantization without activation handling, and GPTQ (Frantar et al., 2023), which requires a calibration set to approximate the second-order of the loss to better minimize quantization error.

SmoothQuant (Xiao et al., 2023) is an activation-aware PTQ technique that rescales weights and activations, respectively $W' = W \cdot s$ and $X' = X/s$, where s is a learned per-channel scale. This balances dynamic ranges and reduces activation outliers, enabling more accurate quantization. We evaluate its effectiveness when combined with GPTQ quantization.

EfficientQAT (Chen et al., 2024) is a two-stage QAT framework (see **??**) for large language models. We chose this method for its ability to train low-bit LLMs with minimal accuracy loss, superior performance over recent QAT methods (Chen et al., 2024), and low memory requirements (Lang et al., 2024).

2.4 RAG

Retrieval Augmented Generation (RAG) (Lewis et al., 2021) is an architecture that conditions the model's responses on contextually relevant chunks of knowledge retrieved from a corpus. By leveraging dense embeddings in a shared latent space, the RAG architecture enables efficient and semantically meaningful retrieval of supporting information. This architecture relies on two key components: a retriever and a generator, each of which can be fine-tuned on dedicated datasets. Additionally, it will requires a corpus from which embeddings are computed to build a searchable index against which embedded queries are compared throughout inference. We study how this method can help improve the model's accuracy on MCQA.

Accordingly, our efforts to improve performance focus on the following components:

Embedder: `gte-large` model (Li et al., 2023) is used as a retriever for the final version but we also experimented with fine-tuning gte-small and various SentenceTransformers on triplet datasets we made.

Retriever: We used the MCQA model as it was already quite performant, thus choosing to focus on the other parameters.

Corpus: The corpus is constructed by combining several HuggingFace datasets, sampling examples across various STEM domains. To improve diversity, we used GPT-based prompting to generate variant questions while preserving topic relevance, similarly to what was done for the preference pairs.

Hyperparameters: Hyperparameters are tuned experimentally by running multiple evaluation loops. Key parameters include the number of retrieved chunks (top_k), similarity functions and batch size.

3 Experiments

3.1 DPO

3.1.1 Data

After initial experiments with the default TRL Ultrafeedback dataset (hug), we found it lacked sufficient STEM content. To better match our small model and target domain, we curated a new dataset (Bartolome et al., 2023) of 63K preference pairs, filtering for STEM relevance using either keyword-matching or a distilled BART classifier. Keyword filtering proved more effective and was retained. We further enriched the data with coding- and math-focused preference pairs (Bartolome et al., 2023), ensuring all prompt+completion sequences fit within 1024 tokens using the Qwen3-0.6B-Base tokenizer. The final dataset was split into 24K training, 3K validation, and 3K test pairs.

3.1.2 Evaluation Metrics

As part of training, we apply early stopping based on **reward accuracy**, denoted R_{acc} (Eq. 1), defined as the proportion of validation samples where the model assigns a higher *reward* to the chosen than to the rejected response:

$$R_{\text{acc}} = \frac{1}{N}\sum_{i=1}^{N} 1\left(r_{\text{chosen},i} > r_{\text{rejected},i}\right). \qquad (1)$$

Here, the reward for each response is computed as the difference between the policy and reference log-probabilities:

$$r_{\text{chosen}} = \log \pi_\theta(y_w \mid x) - \log \pi_{\text{ref}}(y_w \mid x),$$
$$r_{\text{rejected}} = \log \pi_\theta(y_l \mid x) - \log \pi_{\text{ref}}(y_l \mid x).$$

3.1.3 Baselines

For DPO training, we use the Qwen3-0.6B-Base model as the reference policy π_{ref}.

3.1.4 Experimental Details

SFT before DPO: We run experiments to test whether supervised fine-tuning on the base model before running DPO has a positive impact on model performance. For that purpose, we run DPO both on the Qwen3-0.6B-Base model and the best SFT MCQA model obtained in 2.2 to compare the results.

Hyper-parameters: We run many experiments to understand the impact of the DPO training parameters on final model accuracy.

When first starting to run DPO with default TRL parameters, we have to use a batch size of 1 to be able to keep a max size of at least 1024 tokens for each sample (prompt+completion) while avoiding Out of Memory (OOM) errors on the hardware at our disposal. To work around this issue, we tune the number of gradient accumulation steps as we cannot use a batch-size higher than one. This should allow to stabilize training but does not provide the speedup that increasing the batch-size provides.

For the learning-rate, we sweep through orders of magnitude of the learning rate, from 10^{-4} to 10^{-8}.

We also tune the β, which is the main parameter of the DPO loss function, controlling the deviation from the reference model.

Early stopping on validation set: DPO training is quite sensitive to the number of training epochs, and can easily overfit with more than a single epoch. Therefore, we run most of our experiments with 3 epochs and implement early stopping.

Loss functions: We test the use of 2 loss functions for our DPO training. First, the default sigmoid DPO loss function is used. Then, we also try to use the "robust" DPO loss: an unbiased estimate of the DPO loss that is robust to preference noise in the data.

3.1.5 Results

Hyper-parameters: For the learning-rate, the sweep through orders of magnitude of the learning rate shows that 10^{-5} as the best start learning-rate yields the best accuracy on final models.

Early stopping on validation set: Analysis of the evolution of the accuracy on the validation set shows that best performance is almost always obtained at the end of the first epoch of training.

SFT before DPO & β tuning: Table 1 shows the accuracy of the final models on the test-set preference pairs in experiments with or without SFT in the training pipeline before DPO, crossed with different values of beta. This is done because the model might need to diverge differently from the pre-DPO model for optimal performance.

'Robust loss experiment': We run DPO with both 'Robust' loss and the normal DPO loss using

β	Accuracy No-SFT (%)	Accuracy SFT (%)
0.07	83.1	81.9
0.10	83.4	81.5
0.13	83.2	81.0

Table 1: Test-set accuracy of the final models with or without SFT before DPO and different values of β

the best configuration obtained through previous experiments. The 'Robust' loss yields a Test accuracy of $\mathbf{83.9}\%$ which is the overall best final model accuracy. This accuracy is slightly better than that of the normal loss with 83.4%.

3.2 MCQA

3.2.1 Data

The first experiments use the 0.9M split of AM-DeepSeek-R1-Distilled (Zhao et al.), which contains mostly open-ended STEM questions with DeepSeek-R1-generated answers that were carefully validated by the authors using methods like math checking and coding test cases.

Next, a 0.4M MCQ-only dataset is created by combining several STEM-focused MCQ datasets (Pal et al., 2022; Lifferth et al., 2023; Johannes Welbl, 2017; Mihaylov et al., 2018; Khot et al., 2020; Ling et al., 2017; Vujasinovic, 2024; Badertdinov, 2024), along with filtered MCQs from the 0.5M split of *AM-DeepSeek-R1-Distilled* (chosen for its higher MCQ count).

Finally, an extended dataset is built by adding all open questions from the same 0.5M split to the MCQ-only set, resulting in 0.8M samples total.

All samples are converted to the Harness format used in the course evaluation pipeline to match evaluation-time inputs.

We refer to these datasets as: *AM-0.9M* (0.9M split of AM-DeepSeek-R1-Distilled), *MCQ-0.4M* (MCQ-only), and *Mixed-0.8M* (MCQ + open questions from 0.5M split of AM-DeepSeek-R1-Distilled).

For evaluation, we use two benchmarks: MMLU (Hendrycks et al.) ($\sim$14k samples), denoted E_{MMLU}, and a filtered version of NLP4Education (Borges et al.) ($\sim$2k samples), denoted E_{NLP}, with only English MCQs and one correct answer.

3.2.2 Evaluation Metrics

The main metric used for evaluation is the accuracy on MCQs from MMLU and NLP4Education. The evaluation pipeline is imposed by the course and compares the log-likelihoods of the MCQ options, selecting the most likely one as the output of the model. Single token continuation is used, meaning the log-likelihood of only the letters are compared, without including the full text of the options.

3.2.3 Baselines

The baseline used as a reference for metrics comparison is the base model Qwen3-0.6B-Base, which corresponds to the policy before fine-tuning.

3.2.4 Experimental Details

We first compare LoRA (*AM-LoRA*) and full fine-tuning (*AM-FFT*) on *AM-0.9M*. LoRA adapters are applied to query/value projections with rank 16, $\alpha = 32$, and dropout 0.05. Due to lower performance, all later experiments use full fine-tuning.

To boost performance, we change the training set to the MCQ-only dataset *MCQ-0.4M*. Three 300k-sample subsets are tested: one filtered for similarity to NLP4Education (*MCQ-NLP*), one for mean similarity to NLP4Education and MMLU (*MCQ-NLP-MMLU*), and one random (*MCQ-Random*). Filtering details are in Appendix A.3.

We also test multi-epoch training on a smaller filtered subset (120k samples, mean similarity > 0.5). The model is trained for 6 epochs (*MCQ-NLP-MMLU-sm*).

Since smaller datasets does not yield clear gains, we run two final experiments: one on the full *MCQ-0.4M* dataset (*MCQ-Full*), and one on *Mixed-0.8M*, filtered at mean similarity > 0.42 (*Mixed-NLP-MMLU*).

3.2.5 Results

Accuracies on both evaluation datasets can be found in Table 2, with a summary of the different configurations that are run.

Experiment name	**Dataset size**	**Epochs**	**Accuracy (%)**	
			E_{NLP}	E_{MMLU}
(Baseline)	–	–	44.44	49.88
AM-LoRA	900k	1	44.19	43.30
AM-FFT	900k	1	45.16	45.44
MCQ-NLP	300k	1	44.85	51.03
MCQ-NLP-MMLU	300k	1	44.50	**51.40**
MCQ-Random	300k	1	44.29	50.16
MCQ-NLP-MMLU-sm	120k	1	45.46	50.88
MCQ-NLP-MMLU-sm	120k	6	45.82	**51.33**
MCQ-Full	427k	1	46.28	50.86
Mixed-NLP-MMLU	430k	1	**47.30**	50.97

Table 2: Performance on NLP4Education (E_{NLP}) and MMLU (E_{MMLU}), for the different experiments of the MCQA model

3.3 Quantization

3.3.1 Data

GPTQ requires a calibration set to determine the quantization parameters, while EfficientQAT relies on a training dataset. For both methods, we use a subset of the dataset originally employed for supervised fine-tuning (SFT) (see 3.2.1). We denote this dataset by C and explore different dataset sizes $|C|$. For evaluation, we consider the datasets E_{NLP} and E_{MMLU} (defined in 3.2.1).

3.3.2 Evaluation Metrics

Our primary metric is MCQA accuracy, i.e., the fraction of questions whose predicted answer exactly matches the gold label. Regarding the quantized models, we compare as well the average peak VRAM usage during inference, to capture the runtime footprint, and the throughput in tokens per second[1].

3.3.3 Baselines

We use the best model obtained via SFT (see section 3.2) as the primary reference baseline. As this model is not quantized, it serves as an upper bound for accuracy when evaluating the quantized variants.

3.3.4 Experimental Details

We apply post-training quantization (PTQ) to the best SFT checkpoint using three methods: Bits-and-Bytes (BnB) (Dettmers et al., 2022) for 8/4-bit weight-only quantization, GPTQ with block size 64 (Frantar et al., 2023), and SmoothQuant (Xiao et al., 2023) with a smoothing strength of 0.8 to rescale activations, followed by GPTQ for W8A8/W4A8 quantization. The `lm_head` layer is excluded from quantization. We focus primarily on 4-bit compression, with 8-bit and 2-bit variants for comparison. Calibration sets of size 20, 200, and 2000 are sampled from the SFT corpus.

We additionally evaluate quantization-aware training (QAT) using an adapted version of EfficientQAT (Chen et al., 2024) for Qwen3. During the E2E phase, quantization parameters are jointly

[1]measured on a single A100-20GB with batch size 1.

Model	Accuracy (%)		Peak VRAM (MB)	Tok/s
	E_{NLP}	E_{MMLU}		
SFT	–	–	2400	2650
BnB-8bit	**46.7**	50.7	**1230**	**850**
W8A8	46.1	46.8	2640	590
BnB-4bit	43.5	47.2	620	1770
GPTQ-4bit	43.8	48.0	**590**	2720
W4A8	43.1	46.9	2650	160
QAT-4bit	**45.9**	49.8	790	**2840**
QAT-2bit	**37.3**	**37.7**	**490**	**1760**

Table 3: MCQA performance on NLP4Education (E_{NLP}) and MMLU (E_{MMLU}). Tok/s represents the throughput in tokens generated per second on a single A100-20GB. The peak VRAM and the throughput are computed on E_{NLP} and rounded to the nearest ten.

optimized across the model using 1k samples, with a block size of 64.

3.3.5 Results

Table 3 and fig. 2 summarise the performance of the different quantization methods.

Moreover, we experimented with different calibration set sizes ($|C| = 20, 200, 2000$) and observed no significant impact on performance across GPTQ and SmoothQuant methods (see Appendix A.2.3 for details). The QAT method reported in Table **??** was trained using 1k samples. We further evaluated it when trained on 20k samples but observed no notable improvement in performance.

3.4 RAG

3.4.1 Data

The construction of the RAG corpus follows a highly iterative and hands-on process. The final corpus is the result of sampling from six diverse HuggingFace datasets spanning encyclopedic, educational and scientific domains. To enrich the data and improve generalization we generate variants of sampled entries using the OpenAI API, prompting GPT-4 to rewrite questions while preserving semantic essence but altering surface details. Each row in the final corpus consists of a generated question (text), and the name of the source dataset (source). The total corpus is capped at 100k chunks and 51.2M tokens to remain within evaluating memory and performance constraints. A FAISS index is built once and reused across all evaluations to accelerate evaluation and hyperparameter testing.

3.4.2 Evaluation Metrics

The RAG component is evaluated using the MCQA part of the lighteval framework which is explained in 3.2.2.

3.4.3 Baselines

We rely on the default baseline models provided by the Lighteval EPFL evaluation suite. This yields an accuracy of 0.32 using Lighteval. However, while this baseline serves as a point of comparison, the performance of the RAG system is also assessed in light of the iterative improvements made to the MCQA models throughout the project. It should also be noted that MCQA's final accuracy on our preferred eval set (nlp4ed) is 47% which the RAG model outperformed at 57.95%

3.4.4 Experimental Details

Although we experimented with fine-tuning gte-small and SentenceTransformer-based models, gte-large consistently outperforms them and is therefore used. In order to finetune the embedders, we build a dataset made of "anchor", "positive" and "negative" fields by choosing positive fields to be any random question in the same subject as the anchor one and a negative as a random one from a different subject using MMLU.

The RAG corpus is built by sampling question-based content from: 'fmars/wiki-stem', 'deepmind/aqua-rat', 'HuggingFaceTB/openstax-paragraphs', 'openlifescienceai/medmcqa', 'allenai/qasc'.

The process involves substantial manual iteration-testing combinations of datasets, formats, and sampling strategies until a performant and compact corpus is obtained for indexing.

3.4.5 Results

Corpus	Embedder	Accuracy (Dataset)
Old	GTE-Large	0.44 (NLP4)
Old	GTE-Small	0.49 (NLP4)
Old	SentenceTransformer	0.34 (MMLU)
Old	GTE-Small-FineTuned	0.35 (NLP4)
Old	SentenceTransformer-FT	0.22 (MMLU)
New	GTE-Large	0.579 (NLP4)
New	GTE-Large	0.592 (MMLU)

Table 4: Performance of RAG configurations on NLP4Education (NLP4) and MMLU.

Our results are summarized in 4

4 Analysis

4.1 DPO

Gradient accumulation for stable training: Training with batch size 1 leads to noisy loss and unstable learning (Fig. 1). Since larger batch sizes cause OOM errors, we increase gradient accumulation to 4. This smooths the loss curve and improves final model accuracy.

No gain from SFT before DPO: Table 1 shows no performance gain from using an SFT model as the DPO base. This may be due to mismatch: SFT used mostly MCQ data, while DPO was evaluated on open-ended tasks.

Effect of 'Robust' loss: The 'Robust' loss slightly outperforms the default, suggesting some dataset noise but overall reliable annotations—consistent with the strong final performance.

DPO yields major gains: DPO fine-tuning significantly boosts model performance. The best model assigns higher log-probabilities to preferred completions in 83.9

4.2 MCQA

As expected, LoRA (*AM-LoRA*) underperforms compared to full fine-tuning (*AM-FFT*), likely due to fewer trainable parameters. Still, testing other LoRA setups (e.g., CorDA) is needed before drawing general conclusions. Dataset size may also play a role.

Filtering using embedding similarity on NLP4Education (*MCQ-NLP*) improves MMLU accuracy over the random baseline (*MCQ-Random*), suggesting some generalization. However, *MCQ-NLP* slightly overfits to NLP4Education compared to *MCQ-NLP-MMLU*, motivating the use of average similarity to reduce benchmark-specific bias.

Using more epochs on a smaller dataset (*MCQ-NLP-MMLU-bis*) has limited impact, as seen between epochs 1 and 6.

Training on a medium-size dataset ($\sim$400k), especially when mixing in open questions (*Mixed-NLP-MMLU*), yields the best NLP4Education performance. While smaller datasets do better on MMLU, we prioritize performance on NLP4Education, as it directly reflects our target domain (EPFL STEM courses). Thus, we select *Mixed-NLP-MMLU* as our best model.

4.3 Quantization

Activation-aware methods (W8A8, W4A8) significantly increase VRAM usage and reduce throughput compared to weight-only quantization. This overhead arises from frequent dequantization of activations for stable computation.

Weight-only methods (BnB, GPTQ) achieve substantial VRAM savings with minimal throughput compromise. GPTQ-4bit, in particular, offers the lowest memory footprint (595 MB) while maintaining good throughput (1697 tok/s), though slightly behind in accuracy.

Quantization-aware training (QAT-4bit) achieves optimal trade-offs, delivering near-baseline accuracy (45.9%) with significant VRAM reduction (792 MB) and the highest throughput (2839 tok/s). Conversely, aggressive QAT-2bit suffers accuracy loss but remains significantly better than random.

4.4 RAG

Embedders finetuning One of the most surprising findings was the poor performance of fine-tuned embedders compared to the off-the-shelf gte-large. While we expected domain-specific tuning to yield gains, the large capacity and pretraining of gte-large likely enabled better generalization, especially on diverse benchmarks like MMLU and NLP4ED

Corpus building the corpus-building process proved rewarding as we expanded and diversified the pool of sampled datasets, we observed clear accuracy improvements, confirming the value of a richer retrieval base.

5 Ethical considerations

Integrating MCQA, DPO, RAG, and quantization brings major advances to educational AI, but also raises ethical concerns around language, bias, and accessibility. While English support is strong, high-resource languages (e.g., French, German) need localized content, and low-resource ones (e.g., Kurdish, Quechua) face major hurdles due to limited data and degraded performance post-quantization. Language adapters and multilingual tuning are essential.

Students, educators, and developers benefit from aligned, efficient tools. Yet risks remain: DPO may amplify cultural bias, RAG can spread misinformation, and quantization can worsen disparities across user groups and devices. These effects often impact marginalized communities the most.

To reduce harm, systems should be co-developed with diverse users, tested across languages, and retain full-precision options. Transparency tools like content tracking, adversarial tests, and uncertainty estimates improve safety. High-risk domains (e.g., medical, legal) need stricter oversight. Inclusive feedback channels must empower underrepresented users to report issues and guide fixes.

Scaling these systems ethically requires combining safeguards, openness, and inclusive design.

6 Conclusion

We explored multiple strategies to adapt a small LLM for university-level STEM MCQA, combining supervised fine-tuning, DPO, RAG, quantization, and dataset filtering. Each component contributed to improving accuracy while addressing resource constraints. Our results highlight the importance of targeted data selection, preference alignment, and retrieval augmentation in low-capacity models. Future work may explore combining the RAG architecture on top of the SFT model and training with SFT from the DPO fine-tuned model.

References

trl-lib/ultrafeedback_binarized · Datasets at Hugging Face — huggingface.co. https://huggingface.co/datasets/trl-lib/ultrafeedback_binarized. [Accessed 11-06-2025].

Ibragim Badertdinov. 2024. ibragim-bad/arc_easy (revision 269afe).

Alvaro Bartolome, Gabriel Martin, and Daniel Vila. 2023. Notus. https://github.com/argilla-io/notus.

Beatriz Borges, Negar Foroutan, Deniz Bayazit, Anna Sotnikova, Syrielle Montariol, Tanya Nazaretsky, Mohammadreza Banaei, Alireza Sakhaeirad, Philippe Servant, Seyed Parsa Neshaei, Jibril Frej, Angelika Romanou, Gail Weiss, Sepideh Mamooler, Zeming Chen, Simin Fan, Silin Gao, Mete Ismayilzada, Debjit Paul, Philippe Schwaller, Sacha Friedli, Patrick Jermann, Tanja Käser, Antoine Bosselut, EPFL Grader Consortium, and EPFL Data Consortium. Could ChatGPT get an engineering degree? evaluating higher education vulnerability to AI assistants. 121(49):e2414955121. Publisher: Proceedings of the National Academy of Sciences.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Efficientqat: Efficient quantization-aware training for large language models. *arXiv preprint arXiv:2407.11062*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022. 8-bit optimizers via block-wise quantization. *9th International Conference on Learning Representations, ICLR*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models.

Matt Gardner Johannes Welbl, Nelson F. Liu. 2017. Crowdsourcing multiple choice science questions.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. *arXiv:1910.11473v2*.

Jiedong Lang, Zhehao Guo, and Shuyu Huang. 2024. A comprehensive study on quantization techniques for large language models. In *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, pages 224–231. IEEE.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Will Lifferth, Walter Reade, and Addison Howard. 2023. Kaggle - llm science exam. https://kaggle.com/competitions/kaggle-llm-science-exam. Kaggle.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *ACL*.

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

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model.

Milos Vujasinovic. 2024. stem_mcqa_questions (revision 8afa2ae).

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*.

Han Zhao, Haotian Wang, Yiping Peng, Sitong Zhao, Xiaoyu Tian, Shuaiting Chen, Yunjie Ji, and Xiangang Li. 1.4 million open-source distilled reasoning dataset to empower large language model training. Version: 1.

A Appendix

A.1 DPO Results

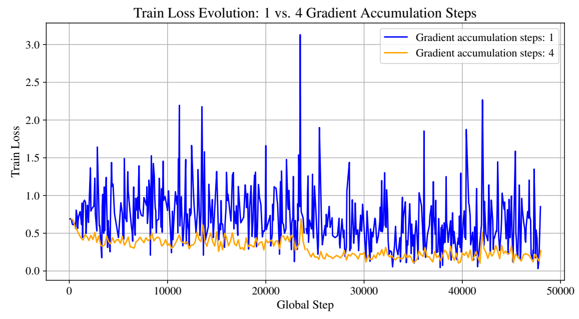

Figure 1: Train Loss Evolution: 1 vs. 4 Gradient Accumulation Steps (x-axis dilated for fair comparison)

A.2 Quantization

A.2.1 EfficientQAT

Block-wise Training (Block-AP): Each transformer block is quantized and trained independently, updating weights and quantization parameters (scaling factors s and zero points z) to minimize reconstruction loss. **End-to-End Quantization Parameter Tuning (E2E-QP):** With weights fixed, s and z are refined jointly across the model to capture inter-block dependencies.

A.2.2 Visualization of the Performance

Figure 2 summarizes Table 3 by showing the quantized model's accuracy on E_{NLP} over average peak VRAM during evaluation. We clearly see that the activation aware quantization methods consume a significant amount of VRAM compared to the other methods. We infer that this is due to the up-casting of the weights from INT4 and INT8 to FP16. Also, We see how QAT-4bit is

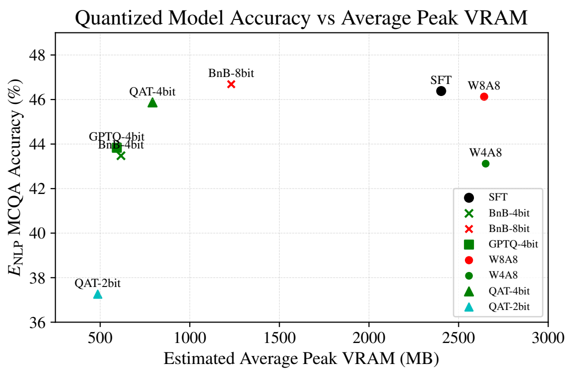

Figure 2: Quantized model accuracy vs peak VRAM usage. The marker size is proportional to the throughput.

A.2.3 Optimal Calibration Set for Quantization

To assess the influence of calibration size on post-training quantization, we evaluate GPTQ and SmoothQuant (W8A8) using three calibration set sizes ($|C| = 20, 200, 2000$) on the NLP4Education benchmark. As shown in Table 5, accuracy remains

Model	**Calibration Size** ($|C|$)	E_{NLP} **Accuracy (%)**
GPTQ-4bit	20	43.1
GPTQ-4bit	200	43.8
GPTQ-4bit	2000	43.9
W8A8	20	46.1
W8A8	200	46.1
W8A8	2000	46.4

Table 5: Accuracy across calibration set sizes ($|C|$) for GPTQ and SmoothQuant W8A8 on the NLP4Education dataset.

stable across calibration sizes for both methods. GPTQ shows minor fluctuations (< 0.9 points absolute difference), while SmoothQuant performs between 46.8%, and 47.0%. These results suggest that even small calibration sets suffice for effective PTQ in constrained settings.

A.3 Computing embeddings similarity for MCQA dataset filtering

To compute the similarity score of a training sample with respect to a given target dataset, we proceed as follows.

All samples in the target dataset, as well as the training sample are embedded using the sentence transformer all-MiniLM-L6-v2 which maps each sample to a 384-dimension vector. Embeddings are then normalized, and a cosine similarity score in the range of $[-1, 1]$ is computed between the training sample and all samples from the target dataset. The maximum cosine similarity score is then kept as the similarity score for the training sample.

This similarity metric is computed for all training samples in *MCQ-0.4M* and *Mixed-0.8M*, for the two target datasets NLP4Education and MMLU, creating two similarity scores for each training sample. Then, multiple filtering strategies are explored to discover the impact of such methods on performance. Generalization and overfitting issues are also discussed in Section 4.2.