STEMQwen: A Modular Fine-Tuning Framework for STEM Multiple-Choice QA

Davit Darbinyan | 396153 | davit.darbinyan@epfl.ch
Siyuan Cheng | 368505 | siyuan.cheng@epfl.ch
Harkeerat Singh Sawhney | 394887 | harkeerat.sawhney@epfl.ch
Zeineb Mellouli | 342064 | zeineb.mellouli@epfl.ch
TutorFlow

Abstract

Large language models (LLMs) show promise as educational tools but remain hampered by high inference costs, limited structured reasoning, and misalignment to task-specific preferences, preventing reliable STEM support in settings like EPFL. We introduce **STEMQwen**, a modular pipeline that transforms a small LLM into a teaching assistant for STEM questions by integrating Direct Preference Optimization, MCQA fine-tuning, retrieval-augmented grounding, and quantization, yielding a compact model that balances accuracy, reasoning quality, and deployment efficiency.

1 Introduction

Large language models (LLMs) are increasingly used as AI assistants in education, but often fall short when EPFL students seek fast, accurate STEM answers—struggling with structured reasoning, reliable explanations, or efficient inference under limited compute (Ouyang et al., 2022). Students tackling physics derivations, algorithm design, or conceptual MCQs need compact, adaptable models that run on consumer hardware. Yet naive fine-tuning yields shallow or incorrect answers, and resource constraints make large-scale reinforcement learning or multi-stage pipelines hard to replicate in academic settings.

Prior work on retrieval-augmented generation (RAG), Direct Preference Optimization (DPO), and quantization has demonstrated value but has generally been explored in isolation. We introduce STEMQwen, a reproducible pipeline built on Qwen3-0.6B-Base that systematically integrates DPO alignment with human preference data, MCQA fine-tuning, RAG retrieval, and quantization under tight compute and time budgets. Our experiments show that this combination noticeably improves both answer quality and efficiency, offering a practical blueprint for specialized reasoning assistants optimized for educational use rather than scale alone.

2 Approach

STEMQwen is a modular framework for transforming a small pretrained language model into an efficient, domain-aligned reasoning assistant. It targets four core capabilities—structured reasoning, alignment to human preferences, factual grounding, and low-cost inference—through separate but interoperable training branches. All components share a common backbone: Qwen3-0.6B-Base (Qwen Team, 2025), an open-source transformer with 0.6 billion parameters. Its strong performance on multilingual and symbolic inputs, along with support for long-context prompts, makes it well-suited for STEM question answering under academic compute budgets.

As show in Appendix G, our system architecture branches from the base model into three training pipelines and one compression path. We apply Direct Preference Optimization (DPO) either directly or following supervised fine-tuning to align the model with human preferences. A separate branch specializes the model for multiple-choice STEM questions using curated prompts and rationale-augmented supervision. From this MCQA branch, we derive a quantized variant via post-training low-rank compression. In parallel, a retrieval-augmented pipeline (RAG) is introduced, combining separate SFT and LoRA stages tailored for noisy external context. Given their differing objectives and configurations, each module is trained and evaluated independently to support fair comparison.

2.1 DPO for Alignment

In order to align our base LLM with human preferences over solution quality, we employ Direct Preference Optimization (DPO) as described by Rafailov et al. (2023). Rather than fitting an explicit

reward model or resorting to reinforcement learning, DPO directly optimizes a preference-based likelihood objective. Given a dataset of triplets (x, y_w, y_l), where y_w is preferred to y_l for prompt x, we minimize the DPO loss (see Appendix B.1). By relying solely on the model's own likelihoods, DPO sidesteps the instability and complexity of reward-model training and policy-gradient methods while still yielding substantial alignment gains.

Within the STEMQwen pipeline, we branch from the Qwen3-0.6B-Base backbone into a dedicated alignment stage. We explore two variants: applying DPO directly to the pretrained base model, and applying DPO on top of a supervised fine-tuned checkpoint. In both cases, we sample pairwise comparisons from our preference corpus, compute the DPO loss (Appendix B.1), and update all transformer layers via standard gradient-based optimization.

2.2 Supervised Fine-Tuning for Multiple-Choice QA (MCQA)

We followed the STEMQwen pipeline to develop the MCQA model, involving supervised fine-tuning (SFT) of the pretrained language model. To enable both answer selection and rationale generation, each data point—comprising a question, a set of choices, the correct answer, and the reasoning—is formatted into a single coherent text sequence. This consistent formatting stabilizes training and improves performance.

Custom Evaluation Metric: Log-Likelihood. To more accurately assess the model's understanding beyond the simple generation, a custom evaluation was designed. Instead of generating a free-form answer, this method evaluates the model's confidence for each possible answer choice. For a given question (prompt, P) and a set of possible answer choices $C = \{C_1, C_2, \ldots, C_k\}$, the evaluation metric computes the log-likelihood of each choice token given the prompt. For a choice C_j consisting of tokens $(c_{j1}, \ldots, c_{jm})$, the score would as seen in Equation 2. Along with accuracy loss on the validation set, this metric is used to provide a more stable and reliable measure of the model's discriminative power.

2.3 Quantization for Efficient Inference

To support deployment under memory and latency constraints, we apply several quantization techniques to our fine-tuned MCQA model based on Qwen3-0.6B-Base. These include post-training quantization (PTQ) via **LLMCompressor** (vLLM contributors, 2024), quantization-aware fine-tuning with QLoRA, and lightweight baselines such as AWQ and BitsAndBytes.

We evaluate four configurations—W8A8, W4A8, W4A16, and QLoRA—each offering distinct trade-offs in compression and performance.

SmoothQuant. SmoothQuant (Xiao et al., 2024) redistributes activation variance into weights via per-channel scaling, improving quantization robustness. The exact transformation is detailed in Appendix B.3.

GPTQ. GPTQ (Frantar et al., 2023) minimizes output distortion using a second-order approximation and block-wise updates. It is applied to both weights and activations in W8A8 and W4A8, and to weights only in W4A16 (see Appendix B.3 for the objective).

AWQ. AWQ (Lin et al., 2024) is a lightweight alternative used in our W4A16 setting. It performs per-channel clipping and static quantization without second-order computation. The quantization formula is provided in Appendix B.3.

QLoRA. QLoRA (Dettmers et al., 2023) fine-tunes a frozen 4-bit model using low-rank adapters, preserving the quantized weights while learning additive corrections. The formulation of this augmentation is given in Appendix B.3.

BitsAndBytes 4-bit. As a baseline, we also evaluate static 4-bit weight-only quantization using the `bitsandbytes` library. This method requires no calibration and keeps activations in full precision for fast deployment.

These strategies reflect a range of accuracy–efficiency trade-offs. Section 3 summarizes their empirical performance.

2.4 Retrieval-Augmented Generation (RAG) for Context-Aware QA

Our RAG pipeline comprises a fine-tuned dual-encoder retriever, a custom STEM corpus, and a two-stage generator training strategy tailored to handle noisy retrieval input and the MCQA task format.

Retriever. We adopt a symmetric dual-encoder architecture (Sentence-BERT backbone). Both `bge-v1.5` (Li et al., 2023b) and `gte-small` (Team, 2023) are fine-tuned on STEM QA pairs using the InfoNCE loss with in-batch negatives and cosine similarity. We then compare their retrieval performance and select `gte-small` for the final system.

Generator. We fine-tune the `Qwen-0.6B-Base` model in two stages: (1) supervised fine-tuning (SFT) on QA pairs from the CamelAI dataset (without retrieval), intended to inject advanced STEM knowledge into the base model; (2) LoRA (Hu et al., 2021) adaptation on an MCQA-style dataset with multi-choice format and reasoning, where each example is prepended with retrieved documents. All LoRA training examples include retrieved context. We construct variants with one or two retrieved documents and truncate inputs to 256 or 512 tokens, and compare models trained on each configuration.

Inference. At test time, the trained `gte-small` retriever fetches the top-k relevant passages, which are concatenated with the question and options before being passed to the generator. To evaluate retrieval robustness, we also conduct ablations by disabling context at inference time.

3 Experiments

3.1 DPO Experiments

Data. Our preference-alignment corpus is a concatenation of the following seven sources:

- **Stack Exchange** crowd-rated answer pairs (Lambert et al., 2023).
- **Stanford Human Preferences** filtered to science subreddits (askacademia, askphysics, askscience) (Ethayarajh et al., 2022).
- **WebGPT Comparisons** restricted to the ARC subset and only clear preference-score gaps (Nakano et al., 2021).
- **STEM DPO** human preference pairs for multi-turn, STEM-related question answering (The Wordsmiths, 2024).
- **PY DPO** assistant-style responses to Python-related queries (Jon Durbin, 2024).
- **HelpSteer3** instruction tuning preferences limited to code-related tasks (Wang et al., 2025).
- **EPFL student-collected** preference pairs.

After filtering and formatting, we held out 5% each for validation and test, resulting in ∼155k training examples and ∼8.9k each for validation and test. The dataset includes prompt, chosen, and rejected fields, and is available at ddarbinyan/MNLP_M3_dpo_dataset.

Evaluation Method. The evaluation of the DPO-aligned models was conducted using the `lighteval-epfl-mnlp` framework. The primary metric used was reward accuracy, which measures the model's ability to correctly identify the preferred response from a given pair, thereby reflecting its alignment with the underlying human preferences. The performance was assessed on the `allenai/reward-bench` dataset (Lambert et al., 2024), specifically filtered to include only STEM-related subsets. This filtering ensured that the evaluation was directly relevant to the STEMQwen project's stated focus on answering STEM questions and emulating a teaching assistant. For comparative analysis, several baseline models were established: Qwen3-0.6B-Base, Qwen3-0.6B, Falcon3-7B-Base, Mistral-7B-v0.1, and the MCQA model. These baselines provided a crucial reference point for assessing the performance gains attributable to the DPO alignment process.

Experiment Details. The experimental setup for DPO training was built upon the Qwen3-0.6B-Base backbone. All runs used a per-device batch size of 4 with 4× gradient-accumulation steps (effective batch size 16) and a fixed learning rate of 5×10^{-6}. To accelerate training and keep the model within our GPU memory constraints, we trained in `bfloat16` precision, enabled unsloth's Triton-fused kernels, and applied gradient checkpointing. Experiments were conducted by applying DPO either directly to the pretrained base model or following an initial SFT step. We swept the DPO β-constant over 0.1 and 0.2, and also evaluated the effect of excluding EPFL student-collected preference pairs. Finally, we assessed training duration by running each variant for 1, 2, and 3 epochs. Throughout, Weights & Biases (wandb) was used to log loss curves, reward-accuracy, and hardware metrics, giving us a detailed view of convergence and resource utilization (see Appendix C.1).

Results And Analysis. Table 1 reports reward accuracy on the STEM subset of reward-bench dataset. The application of DPO significantly improved reward accuracy compared to all baselines. The Qwen3-0.6B-Base model scored 0%, while Qwen3-0.6B, Falcon3-7B-Base, and Mistral-7B-v0.1 ranged from 49% to 53%. In contrast, DPO-trained variants consistently achieved over 80% accuracy, demonstrating DPO's effectiveness in aligning models with human preferences for STEM reasoning.

Notably, the Qwen3 DPO model trained with-

Model	Reward Accuracy (%)
Qwen3-0.6B-Base	0.00
Qwen3-0.6B	53.04
Falcon3-7B-Base	50.38
Mistral-7B-v0.1	49.34
Qwen3 MCQA	64.57
Qwen3 SFT + DPO	80.22
Qwen3 DPO (All Pairs)	80.85
Qwen3 DPO (No EPFL Pairs)	81.83
Qwen3 DPO (No EPFL Pairs, 2 epochs)	81.76
Qwen3 DPO (No EPFL Pairs, 3 epochs)	82.04

Table 1: Reward accuracy (%) on the STEM subset of the Reward-Bench evaluation.

out EPFL preference pairs achieved the highest accuracy. This suggests that the EPFL-specific preference pairs might have introduced noise or biases that hindered generalization, or that their quality and consistency were not as high as the general STEM data, highlighting the importance of preference data curation. The Qwen3 SFT + DPO model performed marginally lower than DPO applied directly to the base model, indicating that a prior SFT step might not always be optimal for subsequent DPO alignment on this specific metric. Furthermore, increasing training epochs from 1 to 3 for the best-performing configuration yielded only marginal gains, suggesting that the model largely converged within the first epoch.

3.2 MCQA Experiments

Dataset: The dataset utilized for training the MCQ Model is compiled from various diverse sources. It encompasses fields such as Math (LI et al., 2024), Physics/Chemistry/Biology (Johannes Welbl, 2017), Algebraic Word Problems (Ling et al., 2017), Medical Questions (Pal et al., 2022), and non-MCQ questions (Sucharush, 2024) from the STEM field.

The training utilizes the AdamW optimizer with a cosine learning rate scheduler. To manage memory and improve computational efficiency, training was performed using bf16 precision and gradient check pointing. (See Appendix 2)

Answer-Only Prediction: An initial experiment focussed on a simpler task to predict only the correct answer letter, without any reasoning. The focus was for the model to gain the ability to select the right answer through implicit reasoning. However this approach yielded low accuracy, and the outcome suggested that without context provided by the model the model struggled to generalize the underlying logic.

Unmasked Loss Calculation: We also tried to train the model on the full input sequence, including the questions without applying loss masking. This meant the model was penalized for errors in generating both the prompt and the response. While this approach gave decent accuracy, it was still inconsistent across different datasets. The model spend a significant portion of its capacity learning to reproduce the input questions rather than learning to reason and answer correctly.

LoRa Fine-Tuning: This tuning method was used to be a computationally cheaper alternative to full SFT, however due to the base model's parameters were too low to make any significant benefits.

Dataset Curation: Switching from a very large, potentially noise dataset to a smaller, more diverse one which not only contained MCQ but different format questions led to a drastically improved performance.

Reasoning Token: The idea of introducing a special [REASONING] token was considered. The purpose was to signal the model the start of the reasoning section, potentially helping it better distinguish between the task of answering and the task of explaining. While not fully implemented, this point towards the future scope of this pipeline.

MCQ Results.

Dataset	Base Qwen	SFT-Masked	SFT-Unmasked	SFT-Implicit
MMLU	0.4386	0.4597	0.4355	0.2363
ARC	0.6578	0.6459	0.6357	0.2234
evals	0.4247	0.4351	0.4390	0.2342

Table 2: MCQ Model Accuracy Results (Same Datasets as RAG)

3.3 Quantization Experiments

Calibration and Training Data. For post-training quantization (PTQ), we used a subset from our fine-tuned MCQA dataset. For W8A8 and W4A8 (SmoothQuant + GPTQ) and W4A16(GPTQ, AWQ), we selected 512 samples with a maximum sequence length of 1024. QLoRA was fine-tuned separately using `zay25/openmath-20k` dataset, which is derived from the original `unsloth/OpenMathReasoning` collection (Unsloth Contributors, 2024). This dataset emphasizes mathematical reasoning and symbolic problem-solving across domains and includes prompts with `<think>` tokens before the solution to encourage intermediate reasoning steps.

Evaluation Method. In addition to accuracy, we monitored:

- **VRAM usage** For models quantized using `llmcompressor`, we used the `vLLM` runtime, which properly loads compressed tensor formats and reflects the actual memory footprint. For QLoRA, AWQ, and BitsAndBytes, VRAM was measured during inference by directly loading each model with `AutoModelForCausalLM` and recording the peak allocated memory during generation using PyTorch's `torch.cuda.max_memory_allocated()`.
- **Model size**, reported based on the serialized model artifacts hosted on the Hugging Face Hub.
- **Inference latency**, computed by averaging the time required to generate 10 tokens across several representative MCQA prompts in batches of 4, following a warm-up phase. The results reflect latency per token, averaged across multiple batches.

Our **baseline** is the uncompressed MCQA model fine-tuned on STEM multiple-choice questions (as described in Section 3.2). All quantized models are directly compared against this reference to measure degradation in performance, memory efficiency, and inference latency.

Experimental Details. Among all quantization methods, QLoRA is the only one involving additional fine-tuning. We applied low-rank adaptation on top of the quantized 4-bit MCQA model using the `zay25/openmath-20k` dataset. Each input was preprocessed by merging the prompt and completion into a single `text` field.

The LoRA configuration used rank 16, alpha 32, and a dropout rate of 0.05, targeting the `q_proj` and `v_proj` modules. Training was conducted for 1 epoch with a batch size of 2 and gradient accumulation over 4 steps, using a learning rate of 2e-5 and FP16 precision.

We used Hugging Face's `SFTTrainer` from the `trl` library with `wandb` logging enabled to monitor the training process. The token-level accuracy, loss progression, and gradient norm evolution are reported in Appendix C.3. All other quantization methods used post-training quantization without gradient updates.

Results and Analysis

See Appendix D for accuracy and VRAM plots of quantized models. Among all quantization methods, **AWQ emerges as the most balanced choice**, combining high accuracy (41.95%) with efficient VRAM usage (603.87 MB) and a compact model size (523 MB). While W8A8 achieves the highest accuracy (44.54%), its larger size (717 MB) and compatibility issues with compressed tensor runtimes like `vLLM` make it less practical for deployment. In contrast, AWQ maintains competitive accuracy—only 1.3 points behind W8A8 and 0.01 below the original full-precision MCQA model—while offering greater evaluation stability.

Model	Acc (%)	Avg VRAM (MB)	Model Size (MB)	Latency (ms/token)
MCQA	43.25	1433.84	1011	11.10
BitsAndBytes	37.14	596.47	506.88	18.09
AWQ	41.95	603.87	523.28	16.73
QLoRA	37.01	600.16	506.88	18.00
W8A8	44.54	721.53	717.53	15.75
W4A8	40.90	618.20	723.44	14.52
W4A16-GPTQ	38.83	602.77	513.44	16.87

Table 3: Comparison of quantized model variants.

QLoRA exhibits modest accuracy (37.01%) and a relatively good memory footprint (600 MB), making it less attractive despite its fine-tuning pipeline. BitsAndBytes delivers the smallest model (506 MB) and lowest VRAM usage (596 MB), but this comes at the cost of significant performance degradation (37.14%).

Beyond quantitative metrics, qualitative outputs further highlight key trade-offs. As shown in Appendix E.1, the QLoRA model generates interpretable responses with step-by-step reasoning, an important trait for educational use. In contrast, BitsAndBytes, though efficient, produces terse outputs with minimal explanation—favoring brevity over pedagogical clarity.

3.4 RAG Experiments

Data. We use following datasets:

- **Embedding training set:** 100k STEM QA pairs (factoid or short reasoning), used to fine-tune the retriever with InfoNCE loss.
- **SFT dataset:** 55k open-ended QA pairs from CamelAI dataset (Li et al., 2023a), filtered to avoid overlap with the retrieval corpus. Used for supervised fine-tuning of the generator.
- **MCQA dataset:** 60k multiple-choice questions with reasoning for LoRA fine-tuning.
- **Retrieval corpus:** 1.1k Wikipedia STEM entries and 55k filtered CamelAI entries, chunked into 200-token segments.
- **Retrieval-augmented dataset:** The MCQA dataset augmented with retrieved context pas-

sages. For each example, we retrieve top-k documents using a fine-tuned dual-encoder retriever and prepend them to the question.

Baselines. We compare five variants: (1) the base model Qwen-0.6B-Base with no fine-tuning or retrieval; (2) SFT only, fine-tuned on CamelAI but tested without retrieval; (3) SFT + RAG, which adds retrieval only at inference; (4) SFT + LoRA (no retrieval), trained with retrieval-augmented inputs but evaluated without retrieval; and (5) SFT + LoRA + RAG, which uses top-2 retrieval during both training and inference (full system).

Experimental Details. We fine-tuned both `bge-base-v1.5` and `gte-small` on the same dataset and selected `gte-small` for downstream use based on validation retrieval accuracy (Accuracy@1 and @3; see Appendix F.1). We used the Hugging Face `transformers` library with the AdamW optimizer (learning rate 5e-5). Standard loss masking was applied to exclude prompt and context tokens from the causal loss computation. SFT was run for 1 epoch with batch size 4. LoRA was configured with rank 8, $\alpha = 32$, and dropout 0.1. The retriever was fine-tuned for 1 epoch with batch size 64. Evaluation was conducted on a filtered subset of MMLU_Pro (math, CS, engineering, physics, chemistry, biology) (Zhang et al., 2023) and ARC-Challenge (Clark et al., 2018).

Results and Analysis. Table 4 shows model's MCQ accuracy on different test sets; Appendix F.2 reports a LoRA ablation.

Model variant	MMLU	ARC	evals
Base	43.19	65.61	42.47
SFT	42.30	65.36	43.90
+ RAG	43.25	63.82	44.55
+ LoRA	44.56	65.02	42.60
+ LoRA + RAG	45.52	65.70	44.81

Table 4: Accuracy (%) on test sets. RAG uses $k = 2$ retrieved documents.

We can observe that retrieval improves performance consistently on MMLU and MNLP-evals when combined with either SFT or LoRA. However, its effect on ARC-Challenge is less stable, possibly due to domain mismatch or noise sensitivity. Notably, the full system (LoRA + RAG) achieves the highest scores overall.

To better understand these trends, we inspected model outputs on factual questions (see App. E.2 for an example). Without retrieval, models often select distractors with high confidence. Retrieval improves both SFT and LoRA variants, with LoRA+RAG achieving the best accuracy. However, LoRA+RAG occasionally continues generating follow-up questions after answering, suggesting overfitting to the multiple-choice format. This may be mitigated with decoding constraints such as stop sequences or length limits.

4 Ethical Considerations

While STEMQwen is designed to support educational access, it raises ethical concerns that merit attention. First, biases in alignment and retrieval corpora may propagate into the model's responses, disproportionately affecting marginalized or underrepresented groups. Second, quantized models, though efficient, may degrade in accuracy on out-of-distribution or linguistically diverse inputs, which risks unfair treatment of users from low-resource language backgrounds. Although the system targets English MCQA, future adaptations to high- and low-resource languages (e.g., French, Urdu) would require careful re-evaluation of corpus quality and alignment strategies. Finally, misuse of such models—e.g., for automated grading or content generation without oversight—could reinforce inequities or propagate misinformation. Ongoing validation and transparent deployment are critical to mitigate these risks.

5 Conclusion

We presented **STEMQwen**, a lightweight, modular framework that integrates DPO alignment, MCQA fine-tuning, RAG grounding, and advanced quantization to adapt Qwen3-0.6B into an effective STEM teaching assistant. Our results show that, with proper alignment, small models can deliver accurate and interpretable answers: DPO improves preference consistency, task-specific fine-tuning boosts multiple-choice accuracy, AWQ and SmoothQuant+GPTQ offer strong accuracy–efficiency trade-offs, and RAG enhances factual grounding. STEMQwen demonstrates that compact, resource-efficient reasoning assistants are viable, though challenges remain in annotation bias, retrieval coverage, and out-of-distribution robustness for quantized models. Future work will explore multilingual extensions, joint retrieval–generation training, and real-world classroom deployment.

References

Peter Clark, Oren Etzioni, et al. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2022. Understanding dataset difficulty with $\mathcal{V}$-usable information. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean 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.

Jon Durbin. 2024. PY-DPO v0.1: Preference-aligned Python QA Pairs. https://huggingface.co/datasets/jondurbin/py-dpo-v0.1. Accessed: 2025-06-07.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2024. Rewardbench: Evaluating reward models for language modeling. https://huggingface.co/spaces/allenai/reward-bench.

Nathan Lambert, Lewis Tunstall, Nazneen Rajani, and Tristan Thrush. 2023. Huggingface h4 stack exchange preference dataset.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).

Jiayao Li, Shou Yao, Yitao Chen, Zexue Hu, Bowen Zhang, Renrui Lu, Hongming Tang, Wayne Xin Zhao, Chang Xu, Zhiyuan Li, et al. 2023a. Camel: Communicative agents for "mind" exploration of large scale language model society. *arXiv preprint arXiv:2303.17760*.

Yuhao Li, Zhiyuan Wen, Yixin Liu, Shuofei Qiao, Hao Liu, Liang Wang, Kehai Chen, and Qun Liu. 2023b. Bge: Baai general embedding. https://huggingface.co/BAAI/bge-base-en-v1.5. BAAI, Beijing Academy of Artificial Intelligence.

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

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

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. In *arXiv*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

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.

Qwen Team. 2025. Qwen3 technical report.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Sucharush. 2024. Camel full. Hugging-Face. Optional note about the dataset.

Alibaba DAMO NLP Team. 2023. Gte: General text embeddings. https://huggingface.co/thenlper/gte-small. Alibaba DAMO Academy.

The Wordsmiths. 2024. STEM-DPO: Pairwise preference comparisons for stem instruction. https://huggingface.co/datasets/thewordsmiths/stem_dpo. Accessed: 2025-06-07.

Unsloth Contributors. 2024. Openmathreasoning: A dataset for multi-step mathematical reasoning. https://huggingface.co/datasets/unsloth/OpenMathReasoning.

vLLM contributors. 2024. Llmcompressor: Efficient compression for large language models. https://github.com/vllm-project/llm-compressor.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. 2025. Helpsteer3-preference: Open human-annotated preference data across diverse tasks and languages.

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

Zixuan Zhang, Yan Huang, Tao Li, Yifei Liu, Yankai Lin, and et al. 2023. Mmlu-pro: A comprehensive and challenging benchmark for large language models. *arXiv preprint arXiv:2310.16944*.

A Team Contribution

- Conceptualization of the training pipeline and report writing: All members
- Preference dataset collection: All members
- DPO dataset preparation and training: Davit Darbinyan
- MCQA dataset preparation and training: Harkeerat Singh Sawhney
- Quantization (all methods): Zeineb Mellouli
- RAG document preparation and training: Siyuan Cheng

B Equations

B.1 DPO Loss

The DPO loss minimized in our alignment stage is given by

$$\mathcal{L}_{\mathrm{DPO}}(\pi_\theta;\pi_{\mathrm{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\Big(\beta\log\frac{\pi_\theta(y_w\mid x)}{\pi_{\mathrm{ref}}(y_w\mid x)} - \beta\log\frac{\pi_\theta(y_l\mid x)}{\pi_{\mathrm{ref}}(y_l\mid x)}\Big)\right], \tag{1}$$

where σ is the logistic sigmoid and β is a temperature-like scaling constant.

B.2 MCQ Log-Likelihood Loss

$$S(C_j) = \sum_{k=1}^{m} \log P(c_{j,k}|P, c_{j,1}, ..., c_{j,k-1}; \theta) \tag{2}$$

$$\hat{C} = \underset{C_j \in C}{\mathrm{argmax}} S(C_j) \tag{3}$$

B.3 Quantization Equations

SmoothQuant. For input activations $\mathbf{A} \in \mathbb{R}^{n\times d}$ and weights $\mathbf{W} \in \mathbb{R}^{d\times m}$, SmoothQuant applies the following per-channel scaling:

$$\mathbf{A}_{:,i} \leftarrow \frac{\mathbf{A}_{:,i}}{\alpha_i}, \quad \mathbf{W}_{i,:} \leftarrow \alpha_i \cdot \mathbf{W}_{i,:} \tag{4}$$

where α_i is computed from activation statistics (e.g., max values).

GPTQ. GPTQ minimizes the difference between original and quantized model outputs using a second-order approximation:

$$\min_{\hat{\mathbf{W}}\in\mathcal{Q}} \left\| f(x;\hat{\mathbf{W}}) - f(x;\mathbf{W}) \right\|_2^2 \tag{5}$$

Here, f is the forward pass function and $\mathcal{Q}$ is the quantized weight space.

AWQ. AWQ performs static per-channel quantization without second-order approximation:

$$\hat{\mathbf{W}} = \mathrm{round}\Big(\frac{\mathbf{W}}{s}\Big) \cdot s \tag{6}$$

where s is a scale factor minimizing the quantization error.

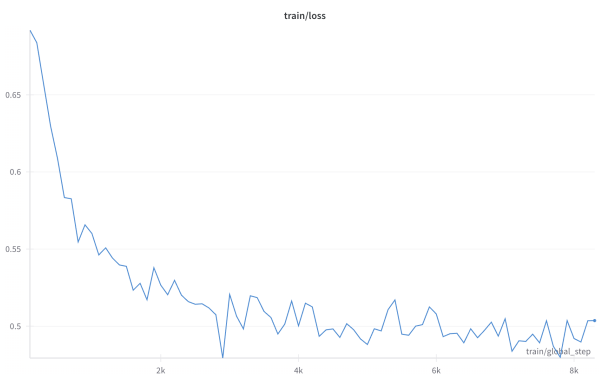
QLoRA. QLoRA fine-tunes a frozen quantized model using trainable low-rank adapters:

$$\hat{\mathbf{W}} = \mathbf{W}_{\text{quant}} + \mathbf{A}\,\mathbf{B} \tag{7}$$

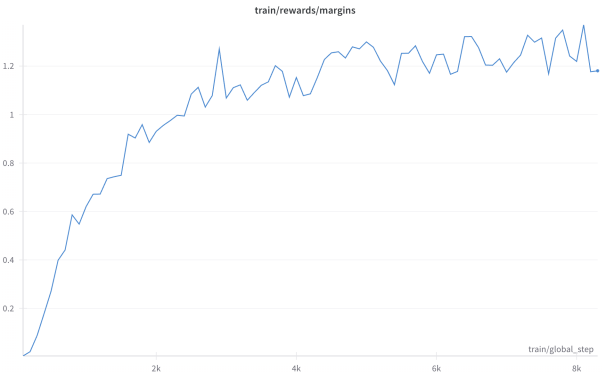
with $\mathbf{A} \in \mathbb{R}^{d\times r}$, $\mathbf{B} \in \mathbb{R}^{r\times m}$, and rank $r \ll \min(d, m)$. This allows adaptation while keeping base weights fixed.

C Training Curves

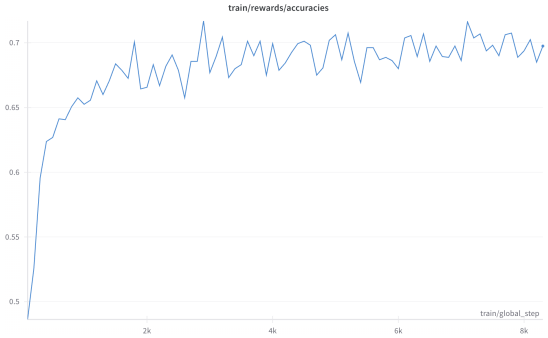
C.1 DPO Training Curves



(a) Training loss over time.



(b) Reward margin between preferred and rejected samples.

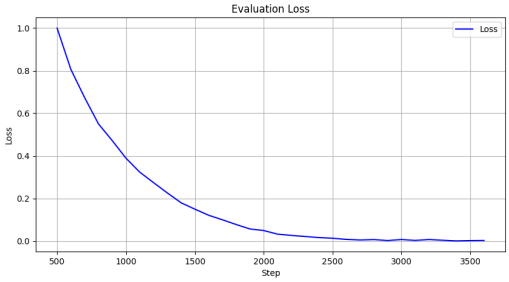


(c) Reward accuracy over training steps.

Figure 1: DPO training diagnostics: (a) training loss, (b) reward margin, and (c) reward accuracy.

C.2 MCQ Training Curves

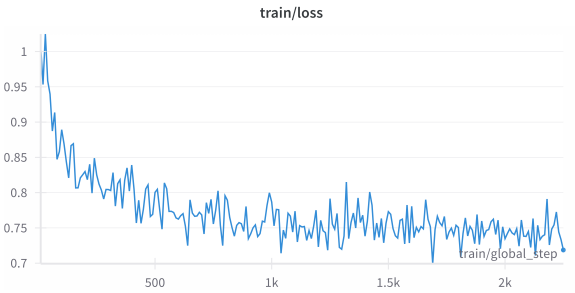
(a) Training Loss over Time.



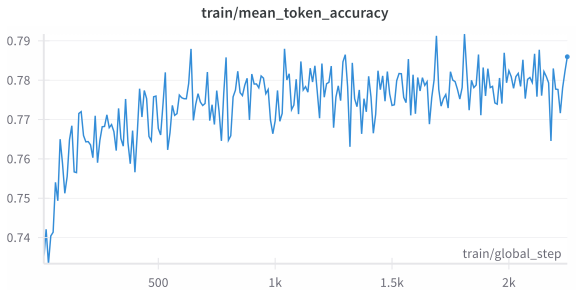
(b) Evaluation Loss over Time.

Figure 2: MCQ training diagnostics: (a) training loss, and (b) evaluation.

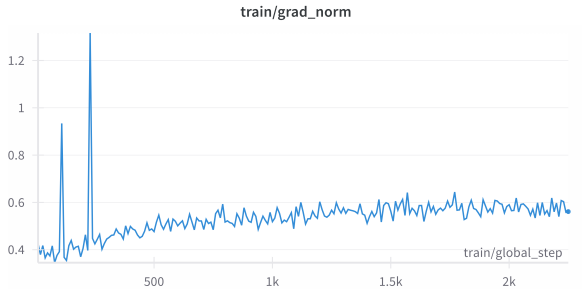
C.3 Quantized Training Curves



(a) Training loss decreasing over time.



(b) Token-level accuracy increasing during training.

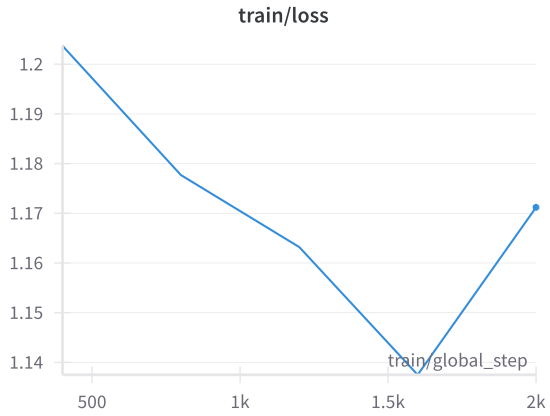


(c) Gradient norm indicating optimization stability.

Figure 3: QLoRA training diagnostics: (a) loss, (b) mean token-level accuracy, and (c) gradient norm.

C.4 RAG Generator

(a) SFT training loss



(b) LoRA training loss

Figure 4: Training loss curves for SFT and LoRA fine-tuning. Loss values are logged periodically during training.

D Quantization Plots

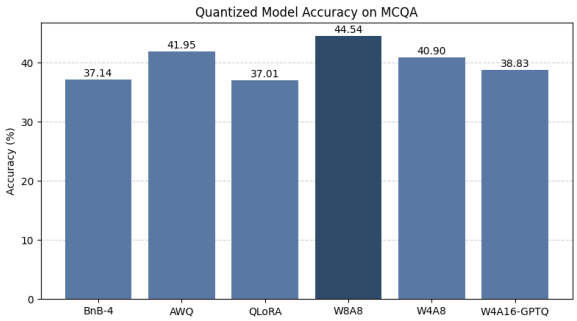


Figure 5: Accuracy comparison of quantized models.

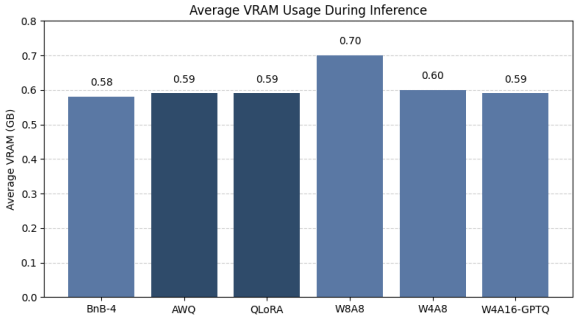


Figure 6: Average VRAM usage across quantized models.

E Sample Generations

E.1 Quantized MCQA

Prompt

Which of the following is a prime number?
A: 4 B: 6 C: 7 D: 9
Answer:

Qlora Output

To determine which of the given numbers is a prime number, we need to check if each number has any divisors other than 1 and itself...

BitsAndBytes Output

The number 7 is a prime number.
Answer: C

E.2 RAG MCQA

We present a representative question to illustrate the effect of retrieval augmentation and task-format adaptation.

Question:
Excited states of the helium atom can be characterized as para- (antiparallel electron spins) and ortho- (parallel electron spins). The observation that an ortho- state has lower energy than the corresponding para- state can be understood in terms of which of the following?

Options:

- A. The Heisenberg uncertainty principle
- **B. The Pauli exclusion principle**
- C. The Bohr model of the atom
- D. Nuclear hyperfine coupling

Top-2 retrieved documents:

- **Doc 1:** Discusses singlet and triplet excited states based on spin multiplicity; relevant to the question as it covers spin alignment and energy levels.
- **Doc 2:** Describes quantum chemical calculations of van der Waals binding energy between hydrogen and helium; unrelated to spin states.

Model Variant	Selected Answer and Excerpt
Base	**D**. Nuclear hyperfine coupling. "Let's think step by step: The Pauli exclusion principle... but it does not directly explain the energy difference..."
Base + RAG	**A**. The Heisenberg uncertainty principle.
SFT	**D**. Nuclear hyperfine coupling. "This phenomenon occurs due to the interaction between the magnetic moments of the electrons and the nucleus..."
SFT + RAG	**B**. The Pauli exclusion principle. "No two electrons in an atom can have the same set of quantum numbers... including the formation of chemical bonds..."
SFT + LoRA	**D**. Nuclear hyperfine coupling. "...interaction between the nuclear spins and the electron spins..."
SFT + LoRA + RAG	**B**. The Pauli exclusion principle. *(Then hallucinates a new MCQ on protein structure)*

Table 5: Example generation on a physics question involving spin configuration. Only retrieval-augmented SFT and LoRA variants produce the correct answer (B). LoRA+RAG also shows overalignment to MCQA format.

F Tables

F.1 Retriever Comparison

Base encoder	Accuracy@1	Accuracy@3
bge-base-v1.5	92.4%	97.9%
gte-small	**92.9%**	**98.2%**

Table 6: Validation retrieval accuracy of fine-tuned retrievers.

F.2 RAG Ablation Studies

Training setting	MMLU	ARC	evals
LoRA (512 tokens, 1 retrieved doc)	44.16	65.70	46.10
LoRA (256 tokens, 2 retrieved docs)	44.61	65.36	44.81
LoRA (512 tokens, 2 retrieved docs)	45.52	65.70	44.81

Table 7: Accuracy (%) of LoRA variants trained under different context configurations.

G STEMQwen Pipeline

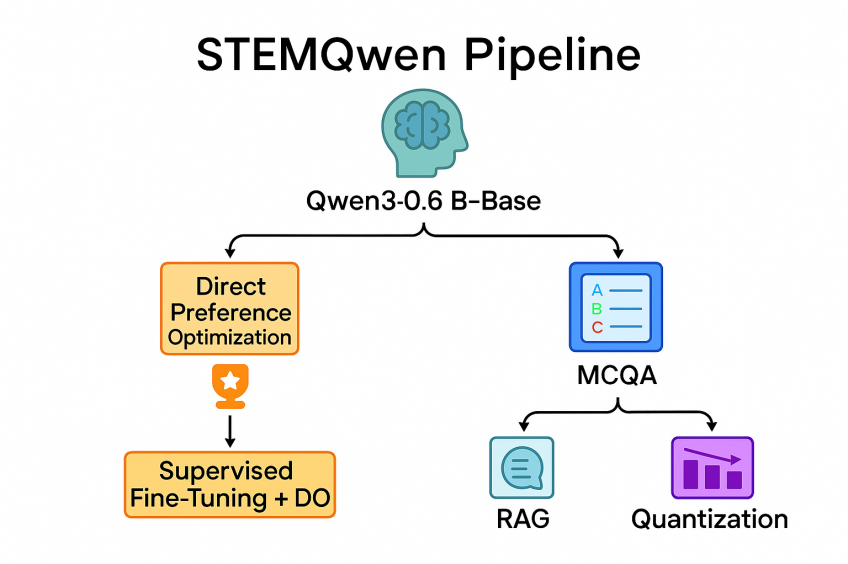


Figure 7: STEMQwen pipeline