Quantization-Aware Distillation for NVFP4 Inference Accuracy Recovery

Meng Xin, Sweta Priyadarshi, Jingyu Xin, Bilal Kartal, Aditya Vavre, Asma Kuriparambil Thekkumpate, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Ido Shahaf, Akhiad Bercovich, Kinjal Patel, Suguna Varshini Velury, Chenjie Luo, Zhiyu Cheng, Jenny Chen, Chen-Han Yu, Wei Ping, Oleg Rybakov, Nima Tajbakhsh, Oluwatobi Olabiyi, Dusan Stosic, Di Wu, Song Han, Eric Chung, Sharath Turuvekere Sreenivas, Bryan Catanzaro, Yoshi Suhara, Tijmen Blankevoort, Huizi Mao[1]

Abstract. This technical report presents quantization-aware distillation (QAD) and our best practices for recovering accuracy of NVFP4-quantized large language models (LLMs) and vision-language models (VLMs). QAD distills a full-precision teacher model into a quantized student model using a KL divergence loss. While applying distillation to quantized models is not a new idea, we observe key advantages of QAD for today's LLMs: 1. It shows remarkable effectiveness and stability for models trained through multi-stage post-training pipelines, including supervised fine-tuning (SFT), reinforcement learning (RL), and model merging, where traditional quantization-aware training (QAT) suffers from engineering complexity and training instability; 2. It is robust to data quality and coverage, enabling accuracy recovery without full training data. We evaluate QAD across multiple post-trained models including AceReason Nemotron, Nemotron 3 Nano, Nemotron Nano V2, Nemotron Nano V2 VL (VLM), and Llama Nemotron Super v1, showing consistent recovery to near-BF16 accuracy.

NVFP4 checkpoints Nemotron 3 Nano 30B-A3B 🤗, Nemotron Nano 9B V2 🤗, Nemotron Nano 12B V2 VL 🤗, Llama 3.1 Nemotron Nano VL 8B 🤗

QAD codes: Megatron-LM version, NeMo version, HuggingFace Transformers version

1. Introduction

The rapid expansion of large language models (LLMs) has increased the demand for more efficient numerical formats to lower computational cost, memory demand, and energy consumption during training and inference. 8-bit floating point formats (FP8 and MXFP8) have emerged as popular data types for accelerated training of LLMs (Micikevicius et al., 2022; DeepSeek-AI, 2024). Recent advances in narrow-precision hardware (NVIDIA, 2024) have positioned 4-bit floating point (FP4) as the next logical step (Chmiel et al., 2025; Chen et al., 2025b; Liu et al., 2023a; Rouhani et al., 2023), delivering a two- to three-fold boost in arithmetic performance and reducing memory usage by half compared to FP8.

NVFP4, which features a smaller block size (16 vs. 32 for MXFP4), FP8 scale factors (E4M3) for fine-grained scaling, and second-level FP32 scaling for a larger dynamic range, has demonstrated superior accuracy to INT4 and MXFP4 on many models (Egiazarian et al., 2025). For very large LLMs, NVFP4 with post-training quantization (PTQ) shows decent accuracy on different benchmarks. However, for small LLMs, the accuracy drop from PTQ is often non-negligible.

There are many efforts to incorporate quantization into the training process. NVFP4 quantized training (NVIDIA, 2025) has demonstrated promising convergence on pretraining tasks. With the primary goal of training speedup, models trained with NVFP4 are still evaluated in BF16.

Quantization-aware training (QAT) (Jacob et al., 2018) is an effective training-based method for inference accuracy recovery, which has shown great results since the CNN era. Many QAT methods reuse the same pipelines and training objectives as the original high-precision models. However, this approach faces significant challenges for modern LLMs in real practices, such as:

1. **Complex training pipelines**: Modern LLMs undergo multi-stage post-training (Bakouch et al., 2025; Bercovich and Itay Levy, 2025), such as SFT, RL, model merging, making it difficult to

[1]Contact: huizim@nvidia.com

replicate the original training procedure.
2. **Data availability and quality**: The original training data of open models may not be available, and public datasets are often of worse quality.

This technical report evaluates quantization-aware distillation (QAD) for NVFP4 inference accuracy recovery on post-trained LLMs. QAD uses the original full-precision model as a teacher and trains the quantized model as a student using KL divergence loss rather than task-specific objectives. We demonstrate the following key findings:

1. QAD aligns the quantized model to the high-precision model better than QAT.
2. QAD works effectively for models trained through multi-stage post-training pipelines including SFT & RL, demonstrating remarkable stability.
3. QAD is robust to incomplete data coverage and can recover accuracy even with partial domain data, enabling cross-domain knowledge transfer.

The report is organized as follows: Section 2 provides a background on NVFP4 and quantization methods; Section 3 describes the QAD method and training setup and demonstrates key properties through comprehensive results across LLMs and VLMs; Section 4 provides detailed analysis of design choices and the impact of training data.

2. Background and Related Work

2.1. NVFP4 Format and Post-Training Quantization

NVFP4 Format. NVFP4 is a 4-bit floating-point format designed for efficient training and inference on modern GPU architectures (NVIDIA, 2025; Alvarez et al., 2025). Compared to FP8, NVFP4 offers 2-3$\times$ higher arithmetic throughput and approximately 1.8$\times$ memory reduction. NVFP4 extends the MXFP4 format (Rouhani et al., 2023) with a smaller block size (from 32 to 16) and two-level scaling (per-block E4M3 scales plus per-tensor FP32 scale). The smaller block size enables more localized adaptation to data distributions, while E4M3 scales provide non-power-of-two scaling factors for lower quantization error, and the second-level FP32 scale extends the overall dynamic range (Alvarez et al., 2025).

Post-Training Quantization (PTQ). PTQ is a simple and cost-effective method that requires no training (Nagel et al., 2021; Wu et al., 2020; Vanhoucke et al., 2011). It involves a process called calibration that determines quantization parameters (e.g., scale factors and zero points) using a small representative dataset (calibration set). With PTQ, training and inference are completely decoupled, making it attractive for practitioners without access to original training pipelines or computational resources for fine-tuning.

The simplest PTQ method is max calibration, which sets the scale factor such that the maximum absolute value in the calibration data maps to the maximum representable value in the quantized format. Despite its simplicity, max calibration works surprisingly well in many cases. More sophisticated calibration methods have been proposed, including MSE-based calibration (Jacob et al., 2018), KL divergence minimization (Migacz, 2017), and learnable clipping thresholds (Choi et al., 2018). More sophisticated PTQ methods have been proposed to minimize quantization error through advanced techniques. For example, AdaRound (Nagel et al., 2020) and BRECQ (Li et al., 2021) optimize weight rounding and block-wise reconstruction, respectively. For LLMs specifically, ZeroQuant (Yao et al., 2022) and GPTQ (Frantar et al., 2023) apply layer-wise calibration with efficient approximations. Other approaches focus on preserving sensitive information (AWQ (Lin et al., 2023)) or optimizing quantization parameters via gradient descent (OmniQuant (Shao et al., 2024a)). A prominent direction is transformation-based PTQ, which applies reversible transformations (e.g., rotations, affine transformations) to suppress outliers and flatten distributions; examples include SmoothQuant (Xiao et al., 2023), QuaRot (Ashkboos et al., 2024), QuIP# (Tseng et al., 2024), SpinQuant (Liu et al., 2024b), and FlatQuant (Sun et al., 2024). Additionally, low-rank approximation methods like SVDQuant (Sui et al., 2024) and EoRA (Liu et al., 2025b) have been proposed to mitigate quantization error by absorbing outliers or compensating for information loss. These advanced methods typically achieve better accuracy than simple max calibration, especially for lower bit-widths.

PTQ works quite well for FP8 on most LLMs. For NVFP4, PTQ also works well for very large models (See Appendix C). However, PTQ often struggles with small models and sensitive tasks. Recent work has highlighted that common PTQ algorithms often fail to improve over baseline NVFP4 performance because the small block size neutralizes traditional outlier mitigation techniques (Egiazarian et al., 2025).

2.2. Quantization-aware training

Quantization-aware training (QAT) (Jacob et al., 2018) is used to recover inference accuracy after a model has been trained in full precision. Unlike native quantized training (DeepSeek-AI, 2024; NVIDIA, 2025), which quantizes weights, activations and gradients to speedup both forward and backward passes, QAT only quantizes the weights and activations, thus only helping forward pass. Gradients remain in high precision to ensure stable convergence.

QAT fine-tunes the quantized model using the same loss function (e.g., cross-entropy for language modeling) and ideally the same training data as the original full-precision model. Modern LLMs typically undergo multi-stage post-training pipelines that include supervised fine-tuning (SFT) and reinforcement learning (RL) stages. For supervised training stages, QAT is straightforward to apply. For RL stages, the equivalent approach would be quantization-aware RL (QARL), which quantizes both the actor's forward pass and rollout generation. However, QARL remains less explored—existing work on quantized RL has focused on accelerating training throughput (Huang et al., 2025; Liu et al., 2025a) rather than post-hoc inference accuracy recovery. This motivates our investigation of QAD as an alternative for RL-trained models.

2.3. Knowledge Distillation for Quantization

Knowledge distillation (KD) (Hinton et al., 2015) transfers knowledge from a teacher model to a student model by learning soft labels (e.g. probability distributions) from the teacher, typically using KL divergence loss for classification tasks. Theoretical work shows that soft labels provide a better estimate of true class probabilities with lower variance (Menon et al., 2020), and that matching teacher distributions provides implicit regularization that accelerates convergence (Phuong and Lampert, 2021).

The combination of knowledge distillation and quantization has been studied extensively for CNNs. Mishra and Marr (2017) demonstrated that low-precision networks can be significantly improved through distillation from full-precision teachers. Polino et al. (2018) proposed quantized distillation, incorporating the distillation loss directly into the training of weight-quantized networks. Kim et al. (2019) introduced a three-phase training procedure to mitigate the strong regularization effect of KD on quantized models.

For large language models, Liu et al. (2023b) demonstrated that data-free distillation using model-generated data is highly effective, enabling quantization-aware training without access to the original training data. Kim et al. (2023) proposed token-scaled logit distillation, which reweights the per-token KL divergence based on teacher confidence to prevent overfitting. Du et al. (2024) introduced BitDistiller for sub-4-bit LLMs, combining asymmetric quantization with a self-distillation objective that blends forward and reverse KL divergence.

Prior QAD work has primarily targeted integer quantization. This report demonstrates that simple KL divergence-based distillation is highly effective for NVFP4 inference accuracy recovery, particularly for LLMs trained through complex multi-stage post-training pipelines where replicating the original training is impractical and standard QAT risks degrading model capabilities.

3. Quantization-Aware Distillation

3.1. Method Overview

Quantization-aware distillation (QAD) uses the original high-precision model as a teacher to train the quantized model as a student using distillation loss. It differs from standard quantization-aware training (QAT), which trains the quantized model using the same task loss as the original model, as illustrated in Figure 1.

The key difference between QAD and QAT lies in the loss function:

- **QAT**: Uses the same task-specific loss, e.g., next-token cross entropy for language modeling, as the original model training pipeline.
- **QAD**: Uses KL divergence between the high-precision teacher and the quantized student.

Formally, for a given input x and vocabulary V, let $p_{\text{teacher}}(y|x)$ denote the output distribution from the full-precision teacher model and $p_{\text{student}}(y|x)$ denote the output distribution from the quantized student model. The QAD loss is the KL divergence between the teacher and student distributions:

$$\mathcal{L}_{\text{QAD}} = D_{\text{KL}}(p_{\text{teacher}} \| p_{\text{student}}) = \sum_{y \in V} p_{\text{teacher}}(y|x) \log \frac{p_{\text{teacher}}(y|x)}{p_{\text{student}}(y|x)} \tag{1}$$

Table 1 implies the key difference between QAD and QAT. Remarkably, QAT achieves nearly the same cross-entropy loss as the BF16 baseline, which might suggest successful capability recovery. However, the KL divergence reveals a different story: QAD achieves nearly zero KL divergence, while QAT exhibits significant divergence from the teacher. This demonstrates that although QAT can match validation loss, it significantly changes the model's output distribution, effectively acting as an additional post-training stage. In con-

Table 1 | QAD better aligns the model with BF16 baseline. Llama Nemotron Super V1 trained with ∼0.3B tokens sampled from its SFT dataset and evaluated on 5,000 held-out samples (∼8M tokens).

Methods	KL Divergence (vs BF16)	Cross Entropy (vs labels)
BF16	0	0.408
QAT	0.311	0.408
QAD	0.004	0.416

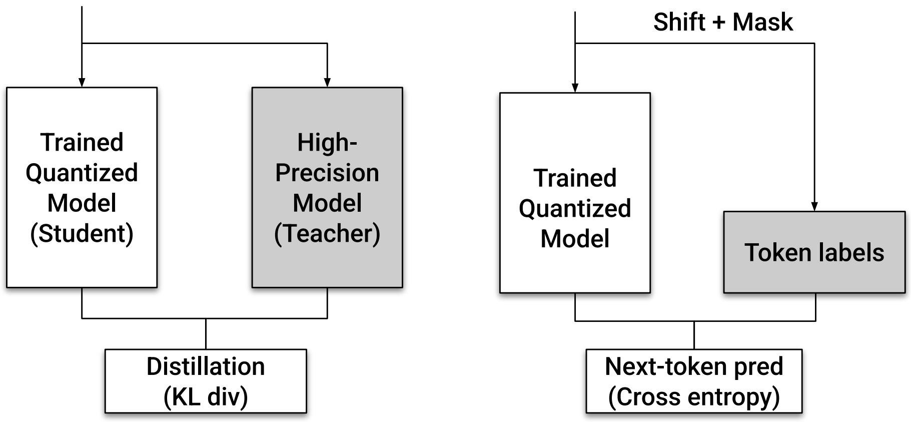
Trained Quantized Model (Student)
High-Precision Model (Teacher)
Distillation (KL div)
Shift + Mask
Trained Quantized Model
Token labels
Next-token pred (Cross entropy)

Figure 1 | Comparison of quantization-aware training (QAT) and quantization-aware distillation (QAD). QAT trains with next-token prediction (cross-entropy) on target datasets, while QAD uses distillation loss (KL divergence) with the full-precision teacher model providing soft targets.

trast, QAD faithfully preserves the original BF16 model's output distribution.

3.2. QAD for Post-Trained Models

For single-stage post-training, we tested on Nemotron Nano 12B v2 VL (NVIDIA, 2025b) and found that QAT can match QAD performance (see Appendix A). However, modern state-of-the-art LLMs typically undergo complex multi-stage pipelines involving SFT, RL, and model merging (Bercovich and Itay Levy, 2025; NVIDIA, 2025a). Applying QAT to such pipelines is challenging: it requires replicating each stage with quantized forward passes and reproducing model merging procedures between stages. A more practical approach is to run QAD or QAT as a single stage using a mixture of SFT data or model-generated data from RL prompts. In this simplified setting, we demonstrate that QAD consistently outperforms QAT, achieving near-BF16 accuracy regardless of the complexity of the original pipeline.

Table 2 | Results on SFT-heavy models. Both QAD and QAT are trained with a mixture of the model's SFT data (math, coding, science, etc). QAD achieves near-BF16 accuracy and significantly outperforms QAT on reasoning tasks, especially on AIME25 and GPQA-D.

Model	Method	MATH500	AIME25	GPQA-D	IFEval-Instruction
Llama Nemotron Super V1	BF16	95.8	46.0	66.5	87.5
	NVFP4 PTQ	91.4	32.3	62.1	86.9
	NVFP4 QAT	94.3	41.5	63.3	87.2
	NVFP4 QAD	**94.6**	**45.6**	**64.5**	**87.8**
Nemotron Nano V2	BF16	97.8	71.1	64.0	90.3
	NVFP4 PTQ	97.2	69.8	59.0	89.8
	NVFP4 QAT	97.2	67.1	56.9	86.2
	NVFP4 QAD	**97.2**	**71.5**	**62.7**	**89.3**

Table 3 | QAD results on RL-heavy models. For both models, NVFP4 QAT breaks the RL model's capabilities, while QAD successfully recovers near-BF16 performance, demonstrating the necessity of distillation for RL models.

(a) Nemotron 3 Nano[2]

Method	AA-LCR	AIME25	GPQA-D	LiveCodeBench-v5	SciCode
BF16	35.9	89.1	73.0	72.1	33.0
NVFP4 PTQ	31.3	85.0	71.6	68.9	30.5
NVFP4 QAT	24.8	83.3	66.0	62.0	25.8
NVFP4 QAD	**34.3**	**87.9**	**72.7**	**68.9**	**32.3**

(b) AceReason Nemotron 1.1 7B

Method	AIME24	AIME25	LiveCodeBench-v6
BF16 Baseline	73.0	63.5	54.3
NVFP4 PTQ	69.4	58.7	52.0
NVFP4 QAT	62.1	46.1	45.9
NVFP4 QAD	**71.7**	**62.0**	**53.3**

SFT-Heavy Models. We evaluate QAD on two SFT-heavy models that both undergo multi-stage post-training: Llama Nemotron Super V1 49B (Bercovich and Itay Levy, 2025) and Nemotron Nano 9B

[2]Benchmark numbers differ from the model card, as different evaluation tools and settings are used.

V2 (NVIDIA, 2025a). Table 2 shows that QAD consistently outperforms QAT on challenging reasoning benchmarks across both models. For Llama Nemotron Super V1, QAD outperforms QAT particularly on AIME25 (+4.1%) and GPQA Diamond (GPQA-D) (+1.2%), recovering to near-BF16 performance. For Nemotron Nano 9B V2, QAD achieves near-BF16 performance and significantly outperforms QAT on AIME25 (+4.4%) and GPQA-D (+5.8%). These results demonstrate QAD's effectiveness on multi-stage trained models.

In addition, Nemotron Nano 12B v2 VL is a VLM model that undergoes a single SFT stage after pre-training. We reported the results in Appendix A.

RL-Heavy Models We evaluate QAD on two RL-heavy models: Nemotron 3 Nano 30B-A3B (NVIDIA, 2025), a hybrid Mamba-Transformer model post-trained with multi-stage RL, and AceReason Nemotron 1.1 7B (Chen et al., 2025a; Liu et al., 2025c), a Qwen2.5-based model (team, 2025a) specialized for math and code through RL. For RL-trained models, the RL training data typically contains only prompts, as the model generates responses during training. However, RL models are typically initialized from a cold-start SFT phase that teaches the base model to solve problems with chain-of-thought reasoning (DeepSeek-AI, 2025; Chen et al., 2025a; team, 2025b; Moonshot, 2025; Ling Team, 2025). This cold-start SFT data provides a practical option for QAD and QAT training. Another option is to use generated data from RL prompts (see Section 4.1) if cold-start SFT data is unavailable. In our experiments, we train Nemotron 3 Nano with a mixture of cold-start SFT data and RL-generated data, and AceReason with only cold-start SFT data.

However, using cold-start SFT data (or mixtures containing it) poses a fundamental challenge for QAT, as it can break the capabilities learned during RL training. Table 3 demonstrates this issue for both models: QAT significantly degrades performance across all benchmarks compared to PTQ, even when the training data includes RL-generated samples. In contrast, QAD successfully recovers near-BF16 performance, as it matches the teacher's output distribution rather than relearning from the data distribution. This demonstrates that distillation is essential for recovering accuracy in RL-trained models. An alternative approach would be to incorporate QAT into the RL training process itself, which remains an active research topic.

These results underscore QAD's key advantage for RL models: it avoids both the complexity of RL training and the risk of breaking learned capabilities, requiring only the full-precision teacher model.

3.3. Robustness to Incomplete Domain Coverage

A key advantage of QAD is robustness to incomplete data coverage—the ability to recover accuracy even when the training data does not cover all domains or capabilities of the model.

Cross-Domain Transfer on Multi-Domain Model. AceReason Nemotron is trained on both math and code domains. Table 4 shows that QAD with partial data (math-only or code-only) nearly matches full data (math+code) results across all benchmarks. Remarkably, QAD trained with only code data recovers strong math performance, demonstrating effective cross-domain knowledge transfer through distillation.

These results demonstrate that the teacher's output distributions encode implicit knowledge about all domains and capabilities, even when the input data comes from limited domains. By training the quantized student to match these distributions, QAD enables cross-domain knowledge transfer, allowing the student to approximate the teacher's behavior across domains not explicitly represented in the training data.

3.4. Training and Evaluation Setup

Quantization Configuration. For Llama Nemotron Super V1 and AceReason Nemotron, we quantize all GEMM layers to NVFP4. For Nemotron Nano 9B V2, a hybrid Mamba-Transformer architecture with 4 Transformer layers and 52 Mamba layers, we employ selective quantization: we keep attention layers in

Table 4 | QAD on AceReason Nemotron 1.1 7B with partial domain coverage. QAD trained with math-only or coding-only data achieves near-full data performance, demonstrating cross-domain knowledge transfer.

	AIME24	AIME25	LiveCodeBench-v6
BF16 Baseline	73.0	63.5	54.3
NVFP4 PTQ	69.4	58.7	52.0
NVFP4 QAD (math only)	71.0	61.7	53.1
NVFP4 QAD (code only)	71.0	62.0	53.3
NVFP4 QAD (math+code)	71.7	62.0	53.3

Transformer blocks and the first and last two layers at BF16 to maintain a better PTQ baseline. For Nemotron 3 Nano, a Mixture-of-Experts hybrid Mamba-Transformer with only 6 self-attention layers, we keep the 6 self-attention layers and their preceding Mamba-2 layers at BF16, quantize the remaining network to NVFP4, and quantize KV-Cache to FP8.

Hyperparameters. We use conservative learning rates: 1e-6 for Llama Nemotron Super V1 and Nemotron Nano 9B V2, and 1e-5 for AceReason Nemotron and Nemotron 3 Nano. The softmax temperature is set to $T = 1$ for both teacher and student to precisely match the teacher's output distribution. Batch sizes and sequence lengths are kept similar to those used in the original post-training.

Data. QAD requires significantly less data than the original post-training. The exact amount depends on the model size and task complexity. We report the amount of data required for convergence.

- Llama Nemotron Super V1 49B: ~0.3 billion tokens
- Nemotron Nano 9B V2: ~6 billion tokens
- Nemotron Nano 12B V2 VL: ~0.5 billion tokens
- Nemotron 3 Nano 30B-A3B: ~2.5 billion tokens
- AceReason Nemotron 7B: ~0.8 billion tokens

Evaluation. For each experiment, we evaluate the top 10 checkpoints with the lowest validation loss and select the one that performs best on average across evaluation benchmarks. For Llama Nemotron Super V1, Nemotron Nano 9B V2, and AceReason Nemotron, we report results using multiple sampling runs per problem: 48 runs for AIME 2024 (Zhang and Math-AI, 2024) and AIME 2025 (Zhang and Math-AI, 2025), 12 runs for LiveCodeBench (Jain et al., 2024), 20 runs for GPQA Diamond (GPQA-D) (Rein et al., 2023), and 5 runs for IFEval (Zhou et al., 2023). All evaluations use temperature $T = 0.6$ and top-$p = 0.95$ for sampling. For Nemotron 3 Nano, we report average results across: 16 runs for AIME 2025, 8 runs for LiveCodeBench-v5, 8 runs for GPQA-D, and 5 runs for AA-LCR (Team, 2025) and SciCode (Tian et al., 2024). For SciCode, we report subtask accuracy. All evaluations use temperature $T = 1.0$ and top-$p = 1.0$ for sampling.

4. Ablation study

This section provides detailed ablation studies of QAD, specifically focusing on the impact of training data quality and learning rate sensitivity.

4.1. Training Data Quality

An important practical question of QAD is the sensitivity to data quality and source. We conduct ablation studies on AceReason Nemotron to understand the impact of different data sources. Similar robustness is observed for Nemotron 3 Nano (see Appendix B).

Table 5 shows QAD's performance with different data sources for AceReason Nemotron. Cold-start SFT

Table 5 | Impact of training data on QAD for AceReason Nemotron 1.1 7B. We test different data sources: (1) **SFT data**: original cold-start SFT data used during training; (2) **Generated from RL prompts**: BF16-generated samples from RL prompts; (3) **Generated from RL prompts (correct only)**: filtered to include only correct solutions; (4) **Generated from BOS token**: data generated by providing a single initial token following Liu et al. (2023b); (5) **Random tokens**: completely random token sequences. QAD shows remarkable robustness across all data sources. All experiments use the same amount of data.

Training data	AIME24	AIME25	LiveCodeBench-v6
BF16 Baseline	73.0	63.5	54.3
NVFP4 PTQ	69.4	58.7	52.0
SFT data	**71.7**	**62.0**	**53.3**
Generated from RL prompts	**71.9**	**61.3**	**52.6**
Generated from RL prompts (correct only)	70.5	61.6	52.3
Generated from BOS token	70.1	60.9	52.4
Random tokens	68.6	60.0	51.7

data and BF16-generated data from RL prompts both achieve near-BF16 performance, demonstrating that synthetic data is highly effective for QAD. Interestingly, using all generated samples (including incorrect ones) performs better than using only correct samples, suggesting that incorrect generations also provide useful information for distillation. Even with completely random tokens, QAD maintains comparable performance to the PTQ baseline without breaking the model, demonstrating remarkable stability.

For Nemotron 3 Nano, we observe similar robustness across different data sources (SFT data, RL-generated data, and their mixtures), with all achieving comparable performance (see Appendix B for details). These results demonstrate that QAD is remarkably robust to data source and quality.

4.2. Learning Rate

QAD requires careful learning rate selection, with different optimal ranges depending on the original training. Overall, we recommend a learning rate between 1e-5 to 1e-6.

For SFT-trained models, we find that using learning rates at or below the original post-training learning rate works best. As shown in Tables 6 and 7, Nemotron Nano 9B V2 achieves optimal performance at

Table 6 | Learning rate sensitivity of QAD for AceReason Nemotron 1.1 7B (RL-heavy) and Nemotron Nano 9B V2 (SFT-heavy). For AceReason, the optimal LR (1e-5) is notably higher than standard RL rates, while for Nano V2 9B, increasing the learning rate above the original SFT LR (1e-6) degrades performance or causes unstable training.

Model	Learning Rate	AIME24	AIME 25	LiveCodeBench
AceReason Nemotron 1.1 7B	1e-6	70.8	61.0	52.6
	5e-6	71.0	60.9	53.2
	1e-5	**71.7**	**62.0**	**53.3**
	1e-4	72.4	61.8	53.0
Nemotron Nano 9B V2	**1e-6**	**80.4**	**71.5**	**67.8**
	5e-6	80.0	71.0	66.8
	1e-5	80.8	69.4	67.4
	1e-4	78.8	65.2	64.0

Table 7 | Learning rate sensitivity for Nemotron Nano 12B v2 VL (SFT-trained). The model achieves optimal performance at LR of 2e-6, which is 10× lower than its original SFT LR (2e-5).

Learning Rate	AI2D	ChartQA	DocVQA	InfoVQA	OCRBench	TextVQA
1e-4	67.0	76.0	75.0	47.6	685	70.6
2e-5	85.3	87.6	91.6	72.2	820	82.8
2e-6	**87.1**	**89.7**	**94.0**	**78.9**	**857**	**84.7**

1e-6 (matching its original SFT learning rate), while Nemotron Nano 12B v2 VL performs best at 2e-6 (10× lower than its original SFT learning rate of 2e-5). Higher learning rates degrade performance or even diverges, likely because these models are already well-converged on the SFT data distribution after post-training.

For RL-trained models, the situation is notably different. Since the final RL stage shifts the model away from the cold-start SFT data distribution, QAD benefits from larger learning rates. Table 6 shows that for AceReason Nemotron, the optimal QAD learning rate is 1e-5, which is substantially larger than typical RL learning rates (∼1e-6) (Shao et al., 2024b; Luo et al., 2025; Chen et al., 2025a).

Table 8 | KL divergence vs. MSE on different models. KL divergence consistently outperforms MSE.

Model	Loss	GPQA-D	AIME24	AIME25	LiveCodeBench
AceReason Nemotron 1.1 7B	KL-Div	/	71.7	62.0	53.3
	MSE	/	71.7	60.1	52.4
Nemotron Nano 9B V2	KL-Div	62.7	80.4	71.5	67.8
	MSE	60.3	80.0	71.5	66.7

4.3. Additional Choices

KL Divergence vs. MSE. We use KL divergence as the distillation loss for QAD, which is the standard choice for matching probability distributions. While other distance metrics such as MSE on logits could theoretically be used, Table 8 shows that KL divergence consistently outperforms MSE across benchmarks. This is likely because KL divergence is better suited to measure distributional differences and provides better gradients for probability matching.

Using a Larger Teacher. Unlike traditional knowledge distillation where a larger teacher transfers knowledge to a smaller student, QAD uses the original BF16 model as the teacher to recover its exact distribution. While using a larger teacher from the same model family is feasible, as they typically trained with similar data and techniques. We tested this on Nemotron Nano 9B V2 using both the original 9B BF16 teacher and a larger 12B BF16 teacher. Table 9 shows that the 9B teacher outperforms the 12B teacher, One potential reason is that adapting to a different distribution requires more triaining data compared to typical QAD. For efficient accuracy recovery, we still recommend using the original model as the teacher.

Table 9 | Results of Nemotron Nano 9B V2 NVFP4 distilled by different teachers. Using the original model as teacher outperforms using a larger teacher.

Teacher	AIME24	AIME25	LiveCodeBench
9B BF16	80.4	71.5	67.8
12B BF16	80.2	69.8	66.7

5. Conclusion

This technical report presents quantization-aware distillation (QAD) as a practical and effective method for recovering inference accuracy of LLMs and VLMs quantized to NVFP4 format. Through experiments on Nemotron Nano, Nemotron Nano VL, Llama Nemotron Super, and AceReason Nemotron, we show that QAD reliably brings NVFP4 models back to near-BF16 accuracy across a wide range of tasks, including models trained with complex SFT, RL, and model merging pipelines where replicating the original training procedure is impractical for standard QAT.

Our ablations further demonstrate that QAD is robust to data coverage and quality: it can leverage partial-domain or synthetic data, and remains stable even when trained on random tokens. Combined with modest data and compute requirements compared to original post-training, these properties make QAD a practical default for NVFP4 accuracy recovery when PTQ alone is insufficient. The NVFP4 QAD checkpoints and code are available at the links provided in the abstract, enabling practitioners to adopt these techniques in real deployments.

References

Eduardo Alvarez, Omri Almog, Eric Chung, Simon Layton, Dusan Stosic, Ronny Krashinsky, and Kyle Aubrey. Introducing NVFP4 for efficient and accurate low-precision inference. NVIDIA Developer Blog, 2025. URL https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/.

Saleh Ashkboos et al. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Akhiad Bercovich and etc Itay Levy. Llama-nemotron: Efficient reasoning models, 2025. URL https://arxiv.org/abs/2505.00949.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning, 2025a. URL https://arxiv.org/abs/2505.16400.

Yuxiang Chen, Xiaoming Xu, Pengle Zhang, Michael Beyer, Martin Rapp, Jun Zhu, and Jianfei Chen. Tetrajet-v2: Accurate nvfp4 training for large language models with oscillation suppression and outlier control. *arXiv preprint arXiv:2510.27527*, 2025b.

Brian Chmiel, Maxim Fishman, Ron Banner, and Daniel Soudry. Fp4 all the way: Fully quantized training of llms. *arXiv preprint arXiv:2505.19115*, 2025.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. In *arXiv preprint arXiv:1805.06085*, 2018.

DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, and Xiaowen Chu. Bitdistiller: Unleashing the potential of sub-4-bit llms via self-distillation. *arXiv preprint arXiv:2402.10631*, 2024.

Vage Egiazarian, Roberto L Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the gap between promise and performance for microscaling FP4 quantization. *arXiv preprint arXiv:2509.23202*, 2025.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, Yao Lu, Xiaojuan Qi, Song Han, and Yukang Chen. Qerl: Beyond efficiency – quantization-enhanced reinforcement learning for llms, 2025. URL https://arxiv.org/abs/2510.11696.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *CoRR*, abs/2403.07974, 2024.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images, 2016. URL https://arxiv.org/abs/1603.07396.

Jangho Kim, Yash Bhalgat, Jinwon Lee, Chirag Patel, and Nojun Kwak. Qkd: Quantization-aware knowledge distillation. *arXiv preprint arXiv:1911.12491*, 2019.

Minsoo Kim, Sihwa Lee, Janghwan Lee, Sukjin Hong, Du-Seong Chang, and Wonyong Sung. Token-scaled logit distillation for ternary weight generative language models. *arXiv preprint arXiv:2308.06744*, 2023.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Zhuang, and Yanwei Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *International Conference on Learning Representations (ICLR)*, 2021.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Inclusion AI Ling Team. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model, 2025. URL https://arxiv.org/abs/2510.18855.

Liyuan Liu, Feng Yao, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Flashrl: 8bit rollouts, full power rl, August 2025a. URL https://fengyao.notion.site/flash-rl.

Shih-yang Liu, Zechun Liu, Xijie Huang, Pingcheng Dong, and Kwang-Ting Cheng. Llm-fp4: 4-bit floating-point quantized transformers. *arXiv preprint arXiv:2310.16836*, 2023a.

Shih-Yang Liu, Maksim Khadkevich, Nai Chit Fung, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, Jan Kautz, Yu-Chiang Frank Wang, Pavlo Molchanov, and Min-Hung Chen. Eora: Fine-tuning-free compensation for compressed llm with eigenspace low-rank approximation, 2025b. URL https://arxiv.org/abs/2410.21271.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12), December 2024a. ISSN 1869-1919. doi: 10.1007/s11432-024-4235-6. URL http://dx.doi.org/10.1007/s11432-024-4235-6.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023b.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024b.

Zihan Liu, Zhuolin Yang, Yang Chen, Chankyu Lee, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron 1.1: Advancing math and code reasoning through sft and rl synergy, 2025c. URL https://arxiv.org/abs/2506.13284.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013 2025. Notion Blog.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning, 2022. URL https://arxiv.org/abs/2203.10244.

Minesh Mathew, Viraj Bagal, Rubèn Pérez Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V Jawahar. Infographicvqa, 2021a. URL https://arxiv.org/abs/2104.12756.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. Docvqa: A dataset for vqa on document images, 2021b. URL https://arxiv.org/abs/2007.00398.

Aditya Krishna Menon, Ankit Singh Rawat, Sashank J. Reddi, Seungyeon Kim, and Sanjiv Kumar. Why distillation helps: a statistical perspective. *arXiv preprint arXiv:2005.10419*, 2020.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

Szymon Migacz. 8-bit inference with tensorrt. NVIDIA GPU Technology Conference (GTC), 2017.

Asit Mishra and Debbie Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *arXiv preprint arXiv:1711.05852*, 2017.

Moonshot. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. *International Conference on Machine Learning (ICML)*, 2020.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

NVIDIA. Nvidia blackwell architecture. Technical report, NVIDIA, 2024. Available at: https://www.nvidia.com/en-us/data-center/technologies/blackwell-architecture/.

NVIDIA. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2025. URL https://arxiv.org/abs/2512.20848.

NVIDIA. Pretraining large language models with nvfp4. *arXiv preprint arXiv:2509.25149*, 2025.

NVIDIA. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025a. URL https://arxiv.org/abs/2508.14444.

NVIDIA. Nvidia nemotron nano v2 vl, 2025b. URL https://arxiv.org/abs/2511.03929.

Mary Phuong and Christoph H. Lampert. Towards understanding knowledge distillation. *arXiv preprint arXiv:2105.13093*, 2021.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Marius Hall, Vijay Korthikanti, Vinay Choudhary, Yifei Liu, Ankur Mathews, Jinchen Zhu, Zhaodong Yao, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqiu Li, Yu Qiao, Ping Luo, Kaipeng Wang, and Xiaogang Zhang. Omniquant: Omnidirectionally calibrated quantization for large language models. *International Conference on Learning Representations (ICLR)*, 2024a.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024b. URL https://arxiv.org/abs/2402.03300.

Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326, 2019.

Yang Sui et al. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024.

Artificial Analysis Team. Artificial analysis long context reasoning benchmark(lcr), 2025.

Qwen team. Qwen2.5 technical report, 2025a. URL https://arxiv.org/abs/2412.15115.

Qwen team. Qwen3 technical report, 2025b. URL https://arxiv.org/abs/2505.09388.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu Huerta, and Hao Peng. Scicode: A research coding benchmark curated by scientists, 2024. URL https://arxiv.org/abs/2407.13168.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks, 2024. URL https://arxiv.org/abs/2402.04396.

Vincent Vanhoucke, Andrew Senior, Mark Z Mao, et al. Improving the speed of neural networks on cpus. In *Proc. deep learning and unsupervised feature learning NIPS workshop*, volume 1, page 4, 2011.

Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *CoRR*, abs/2004.09602, 2020. URL https://arxiv.org/abs/2004.09602.

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

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A. Llama Nemotron VL results

Nemotron Nano 12B v2 VL is a VLM model that undergoes a single SFT stage after pre-training. unlike other post-trained LLMs, QAT for this model achieves comparable accuracy to QAD. We believe this is attributed to the simple training pipeline and also the small accuracy drop from PTQ.

Table 10 | Accuracy comparison of different methods on Llama Nemotron Nano 12B v2 VL with AI2D (Kembhavi et al., 2016), ChartQA (Masry et al., 2022), DocVQA (Mathew et al., 2021b), InfoVQA (Mathew et al., 2021a), OCRBench (Liu et al., 2024a), and TextVQA (Singh et al., 2019)

Method	AI2D	ChartQA	DocVQA	InfoVQA	OCRBench	TextVQA
Baseline	87.3	89.7	94.3	79.3	855	85.2
PTQ	86.8	89.6	93.8	78.2	850	84.8
QAT	86.5	89.8	93.7	78.3	848	84.8
QAD	**86.7**	**89.4**	**93.9**	**78.4**	**858**	**85.2**

B. Nemotron 3 Nano Data Quality Ablation

For Nemotron 3 Nano, we test QAD with three different data sources: (1) cold-start SFT data only, (2) BF16-generated data from RL prompts only, and (3) a mixture of both. Table 11 shows that all three data sources achieve similar performance, with the SFT+RL mixture performing slightly better.

Table 11 | Impact of training data on QAD for Nemotron 3 Nano 30B-A3B. All three data sources achieve comparable performance, demonstrating QAD's robustness to data composition.

Training data	AA-LCR	AIME25	GPQA-D	LiveCodeBench-v5	SciCode
BF16 Baseline	35.9	89.1	73.0	72.1	33.0
NVFP4 PTQ	31.3	85.0	71.6	68.9	30.5
SFT data	32.6	86.0	72.7	70.0	31.7
Generated from RL prompts	34.0	82.7	73.9	70.4	33.1
SFT+RL generations mixture	34.3	87.9	72.7	68.9	32.3

C. PTQ for Large Models

An empirical finding is that larger LLMs are more robust to quantization. In Table 12, we show one example each for NVIDIA trained models and community models. More results and quantized checkpoints can be found under HuggingFace - NVIDIA collections.

Table 12 | PTQ results on large models (hundreds of billions of parameters). Large models are robust to NVFP4 PTQ, achieving near-original accuracy without any fine-tuning.

Model	Method	MATH500	AIME24	GPQA-D	GSM8K
Llama Nemotron Ultra V1 (253B)	BF16	96.6	75.0	75.7	/
	NVFP4 PTQ	96.2	76.0	74.8	/
DeepSeek R1 (671B)	Official FP8	95.4	80.0	69.7	96.3
	NVFP4 PTQ	94.2	80.0	69.2	96.1

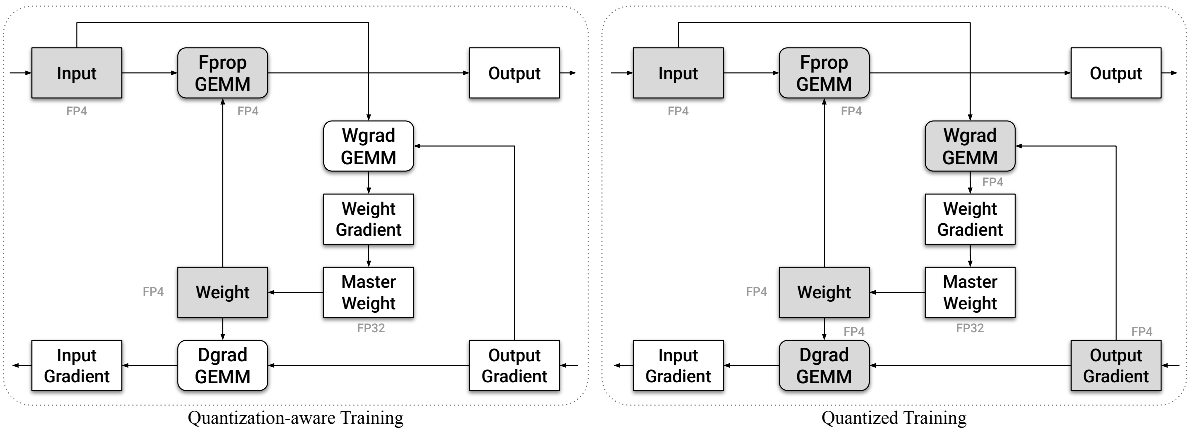

Figure 2 | Comparison of quantization-aware training (QAT) and native quantized training. QAT only quantizes the forward pass for inference recovery, while native quantized training quantizes all three GEMMs (Fprop, Wgrad, Dgrad) to reduce training cost. QAD has a similar compute graph as QAT.

D. Comparison of QAT/QAD with native quantized training

Native quantized training traces its origins to mixed precision training (FP16/BF16), with both approaches sharing the primary goal of reducing computational cost during training. Native quantized training is primarily used during pretraining, where training is compute-bound since batch sizes can be made arbitrarily large. The core idea is to quantize three GEMMs: forward propagation (Fprop), weight gradient (Wgrad), and data gradient (Dgrad). To perform these GEMMs in low precision, all three of their inputs (activations, weights, and output gradients) must be quantized. Examples include DeepSeek V3's FP8 training (DeepSeek-AI, 2024) and recent NVFP4 and MXFP4 pretraining (NVIDIA, 2025; Chmiel et al., 2025; Chen et al., 2025b; Rouhani et al., 2023). These methods focus on reducing the cost of pretraining frontier models from scratch.

QAT/QAD select the quantization targets similarly, as both of them quantize weights and/or activations while leaving gradients in high precision. As a result, the two GEMMs (Wgrad and Dgrad) in backward pass cannot be calculated in lower precision. Figure 2 illustrates the key difference between QAT and quantized training.