SCOPE-Q: ONE-CHECKPOINT QUANTIZATION-AWARE TRAINING FOR CROSS-FRAMEWORK LLM DEPLOYMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are often deployed through several frameworks, such as OpenVINO, TensorRT, Core ML, and AIMET, each of which quantizes the model with its own quantization configuration. Most LLM quantization methods optimize a model for a single, fixed quantizer. We find that this assumption is costly in practice: at the same 4-bit precision, the MMLU accuracy of one checkpoint varies by 12–15 points across these four frameworks, mainly because of differences in their quantization configurations. In this paper, we propose SCOPE-Q, a unified quantization-aware training method that produces a single checkpoint for multiple deployment frameworks. SCOPE-Q learns a low-rank update to the frozen pretrained weights while sampling one target quantization configuration at each training step, and distills knowledge from the full-precision model. After training, the update is merged into the weights, yielding a standard floating-point checkpoint that each framework quantizes without modification. Experiments on four LLMs ranging from 0.5B to 4.6B parameters show that a single SCOPE-Q checkpoint improves worst-case accuracy across frameworks by 7.5–8.4 points over round-to-nearest quantization and reduces the accuracy gap between frameworks by 44–69%, while training only 1.1–3.6% of the parameters.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed on hardware far more constrained than the accelerators they are trained on, including personal computers, mobile devices, and embedded neural processors. Weight quantization makes such deployment practical by storing the weights at low precision (Jacob et al., 2018; Nagel et al., 2021), and post-training quantization (PTQ) (Frantar et al., 2023; Lin et al., 2024; Shao et al., 2024) and quantization-aware training (QAT) (Esser et al., 2020; Liu et al., 2024; Chen et al., 2024) now preserve much of the full-precision accuracy at 4 bits. Most of these methods share a common assumption: the quantizer is known when the model is optimized, and the model is deployed with that same quantizer.

This assumption does not hold for a model released to many users. A single floating-point checkpoint reaches its users through several deployment frameworks, such as OpenVINO (Intel Corporation, 2024), TensorRT (NVIDIA Corporation, 2024), Core ML (Apple Inc., 2024), and AIMET (Qualcomm Innovation Center, 2024), and each framework quantizes the weights with its own configuration. Even at the same bit width, these configurations group weights for scaling at different granularities, from one scale per output channel to one per group of 64 or 128 weights, and use either symmetric or affine scales. In the setting we study, the developer releases one checkpoint, and each downstream user quantizes it independently with the configuration of their target framework.

Figure 1 shows that this heterogeneity has a large effect on accuracy. Quantized with the native configurations of the four frameworks, the same checkpoint differs by 12.1–15.1 MMLU points between the best and worst framework on each of the four models we study. We call this difference the *cross-framework quantization gap*. When all four frameworks use one common configuration, the gap falls to 0.3–5.7 points, which suggests that the configuration, rather than the framework implementation, is its dominant contributor. A common configuration may nevertheless differ from

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

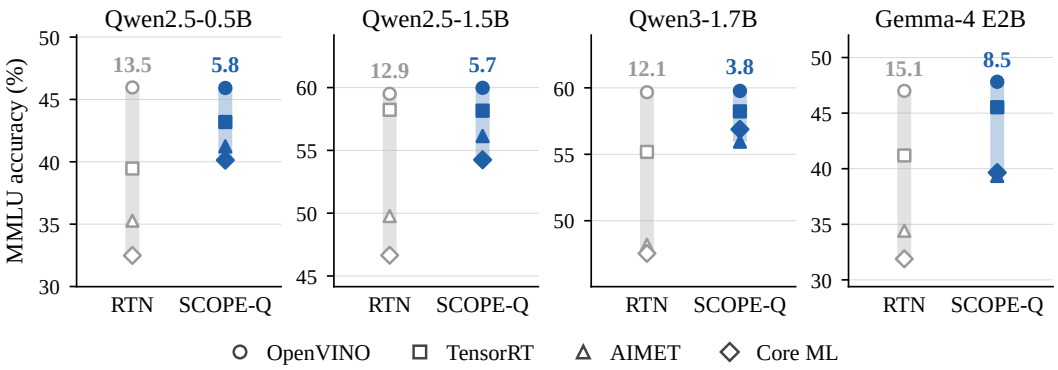

Figure 1: **The cross-framework quantization gap.** Each marker is the MMLU accuracy of one checkpoint after quantization with one framework's native configuration; bars span the best and worst framework, and the numbers give the gap. Round-to-nearest (RTN) quantization of the pretrained checkpoint leaves a gap of 12.1–15.1 points; a single SCOPE-Q checkpoint narrows it to 3.8–8.5 points and raises worst-case accuracy on all four models.

the configuration preferred for a particular hardware target: TensorRT provides no per-channel 4-bit mode, whereas Apple recommends per-channel scales on the Neural Engine (Apple Inc., 2026b).

Existing LLM QAT methods are not designed for this setting. They typically specialize a model to one target configuration, so supporting several deployment targets can require several specialized checkpoints, and a model trained for one configuration can lose accuracy on the others. We propose SCOPE-Q, a quantization-aware training method that optimizes a single checkpoint for a predefined set of deployment configurations. SCOPE-Q freezes the pretrained weights and learns a low-rank update (Hu et al., 2022) through a simulated quantizer whose configuration is resampled from the set at every step, together with a distillation loss (Hinton et al., 2015) toward the full-precision model. The quantization parameters are recomputed from the weights at every step, as each framework does at deployment, so the improvement is carried entirely by the floating-point weights. After training, the update is folded into the checkpoint, which each framework then quantizes with its own unmodified procedure. Our contributions are:

- **Measurement.** We characterize the cross-framework quantization gap of LLMs across four production frameworks and show, with a common-configuration control, that the quantization configuration is its dominant contributor.
- **Method.** We propose SCOPE-Q, a configuration-sampled low-rank QAT method that trains 1.1–3.6% of the parameters and exports one portable floating-point checkpoint for a set of deployment quantizers.
- **Results.** On four models from 0.5B to 4.6B parameters, a single SCOPE-Q checkpoint improves worst-case accuracy by 7.5–8.4 points over round-to-nearest (RTN) quantization and narrows the gap by 44–69%. A matched single-configuration control attributes one third to one half of this gain to multi-configuration training, and we identify when a common configuration or per-framework QAT remains competitive.

Section 2 defines the deployment setting, Section 3 reviews related work, Section 4 presents SCOPE-Q, Section 5 reports experiments, and Section 6 concludes.

2 Deployment Setting and Problem Formulation

2.1 Quantization configurations

We consider weight-only uniform quantization. A configuration $q = (b, \mathcal{G}, \sigma)$ specifies the bit width b, a partition $\mathcal{G}$ of each weight matrix into groups that share quantization parameters, and a scaling rule $\sigma \in \{\text{symmetric}, \text{affine}\}$. Per-channel quantization uses one group per output row, and group-g

Table 1: Frameworks and the native 4-bit weight configurations used in this study.

Framework	Quantization toolchain	Scaling	Granularity
OpenVINO	NNCF (Intel Corporation, 2024)	affine	group-64
TensorRT	Model Optimizer (NVIDIA Corporation, 2024)	symmetric	group-128
Core ML	Core ML Tools (Apple Inc., 2024)	symmetric	per-channel
AIMET	AIMET (Qualcomm Innovation Center, 2024)	symmetric	per-channel

quantization splits each row into groups of g consecutive weights. For a group $g \in \mathcal{G}$, symmetric quantization computes

$$s_g = \frac{\max_{i \in g} |w_i|}{2^{b-1} - 1}, \qquad Q_q(w_i) = s_g \cdot \mathrm{clip}\big(\lfloor w_i / s_g \rceil, -(2^{b-1} - 1), 2^{b-1} - 1\big), \tag{1}$$

and affine quantization uses $s_g = (\max_{i \in g} w_i - \min_{i \in g} w_i)/(2^b - 1)$ with zero point $z_g = -\lfloor \min_{i \in g} w_i / s_g \rceil$, reconstructing $s_g[\mathrm{clip}(\lfloor w_i/s_g \rceil + z_g, 0, 2^b - 1) - z_g]$. The step size grows with the dynamic range of a group, and merging groups cannot reduce that range: if a group g' is the union of groups $g_1, \ldots, g_K$, then

$$s_{g'} \geq \max_k s_{g_k}, \tag{2}$$

so coarser grouping cannot reduce the step size or the corresponding rounding-error bound $s_g/2$.

2.2 Frameworks and native configurations

We distinguish a *framework*, the deployment software, from a *configuration*, the quantizer q it applies. A deployment target $t_m = (F_m, q_m)$ pairs framework F_m with its *native configuration* q_m, the configuration we associate with it in this study (Table 1), and $\mathcal{T} = \{t_1, \ldots, t_M\}$ is the set of targets. Different frameworks may share a nominal configuration, as Core ML and AIMET do here, and may still produce different results because their implementations differ. A *common configuration* is a single configuration applied to all four frameworks. Throughout, we assume that downstream users independently quantize a released floating-point checkpoint with the native configuration of their target framework, deriving the quantization parameters from the weights with the framework's data-free procedure.

2.3 Evaluation and objective

Let $A_m(W)$ be the accuracy of checkpoint W after framework F_m quantizes it with q_m. We evaluate a checkpoint by its mean accuracy $\bar{A}(W) = \frac{1}{M} \sum_{m=1}^{M} A_m(W)$, its worst-case accuracy $A_{\min}(W) = \min_m A_m(W)$, and its cross-framework quantization gap

$$\mathrm{Gap}(W; \mathcal{T}) = \max_m A_m(W) - \min_m A_m(W). \tag{3}$$

Because a smaller gap can also result from degrading the best framework, we report it together with worst-case accuracy. Standard single-target QAT optimizes trainable parameters θ for one configuration; covering several distinct configurations generally requires multiple specialized runs, and framework-specific QAT paths may remain separate even when two frameworks share a nominal configuration. We instead seek one set of parameters for all targets,

$$\theta^\star = \arg\min_\theta \; \mathbb{E}_{m \sim p(\mathcal{T})} \, \mathbb{E}_{x \sim \mathcal{D}} \left[\ell \left(f_{Q_{q_m}(W_\theta)}(x) \right) \right], \tag{4}$$

where Q_q is the configuration-level quantizer of Section 2.1, which our training simulator implements, and $p(\mathcal{T})$ is a distribution over targets. We write $\mathcal{Q} = (q_1, \ldots, q_M)$ for the list of target configurations. We sample targets uniformly rather than unique configurations, which weights every framework equally, as the mean accuracy does; a configuration shared by several frameworks therefore receives proportionally more probability, and in our setup the per-channel configuration is sampled with probability $1/2$. A worst-case objective, $\min_\theta \max_m$, is a natural alternative, which we compare with the expectation in Section 5.4.

3 Related Work

Quantizing LLMs for one configuration. Post-training quantization (PTQ) methods handle outlier features (Dettmers et al., 2022; Yao et al., 2022), reconstruct layer outputs with approximate second-order updates (Frantar et al., 2023; Nagel et al., 2020; Li et al., 2021a), rescale salient channels (Lin et al., 2024; Xiao et al., 2023), learn clipping and equivalent transformations (Shao et al., 2024), or rotate weights and activations (Ashkboos et al., 2024; Liu et al., 2025); others use sparse outliers, incoherence processing, or codebooks that require custom kernels (Dettmers et al., 2024; Chee et al., 2023; Tseng et al., 2024; Egiazarian et al., 2024). QAT trains through simulated quantization with the straight-through estimator (Bengio et al., 2013; Jacob et al., 2018; Choi et al., 2018; Esser et al., 2020), and LLM variants add distillation (Liu et al., 2024; Du et al., 2024), block-wise training (Chen et al., 2024), or low-rank and scale parameters on a frozen or quantized base (Dettmers et al., 2023; Kim et al., 2023; Xu et al., 2024; Li et al., 2024; Jeon et al., 2024; Bondarenko et al., 2024). These methods optimize for one configuration chosen in advance, and several export quantized weights or learned quantization parameters that a different framework would recompute.

Training across multiple quantizers. Robust-quantization methods regularize a model to tolerate post-training quantization at several bit widths (Alizadeh et al., 2020; Shkolnik et al., 2020), and AdaBits, any-precision networks, and MultiQuant (Jin et al., 2020; Yu et al., 2021; Xu et al., 2022) train shared weights across bit widths, in the spirit of once-for-all training (Cai et al., 2020; Shen et al., 2021). The closest predecessor is EQ-Net (Xu et al., 2023), which already trains shared weights across multiple bit widths, granularities, and symmetry choices. SCOPE-Q differs in its target setting: large language models, parameter-efficient adaptation of a pretrained checkpoint, independently implemented deployment quantizers, and the export of a single floating-point checkpoint that carries no quantizer state.

Deployment quantization. Production frameworks implement their own quantizers (Intel Corporation, 2024; NVIDIA Corporation, 2024; Apple Inc., 2024; Qualcomm Innovation Center, 2024), and MQBench (Li et al., 2021b) benchmarked quantization under the hardware-specific settings of several backends for vision models. We measure the corresponding gap for LLMs and optimize one checkpoint for a known set of framework configurations.

4 SCOPE-Q

SCOPE-Q has three components (Figure 2): a low-rank update to each linear layer of a frozen pretrained model, training of that update under a quantizer configuration resampled at every step, and folding of the update into the weights before export.

4.1 Low-rank parameterization

Let $\mathcal{S}$ be the linear layers to be quantized: the four attention projections and three feed-forward projections of each decoder layer. For each $\ell \in \mathcal{S}$ with frozen weight $W_\ell \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we introduce trainable matrices $A_\ell \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B_\ell \in \mathbb{R}^{d_{\text{out}} \times r}$ with $r \ll \min(d_{\text{in}}, d_{\text{out}})$ and define

$$\widetilde{W}_\ell = W_\ell + \Delta W_\ell, \qquad \Delta W_\ell = \tfrac{\alpha}{r} B_\ell A_\ell, \tag{5}$$

with B_ℓ initialized to zero so that training starts from the pretrained model. Only A_ℓ and B_ℓ are trained, which amounts to 1.1–3.6% of the parameters at rank 32 for the models we study.

4.2 Configuration-sampled quantization-aware training

At every step, a target $m \sim p(\mathcal{T})$ is drawn, uniformly by default, and its configuration $q = q_m$ is applied to the updated weight,

$$\widehat{W}_\ell^{(q)} = Q_q\big(W_\ell + \tfrac{\alpha}{r} B_\ell A_\ell\big)\,, \tag{6}$$

so that layer ℓ computes $h_{\ell+1} = \widehat{W}_\ell^{(q)} h_\ell$. Equation (6) quantizes the *sum* of the frozen weight and the update, rather than adding a floating-point side path that would remain active at inference, and

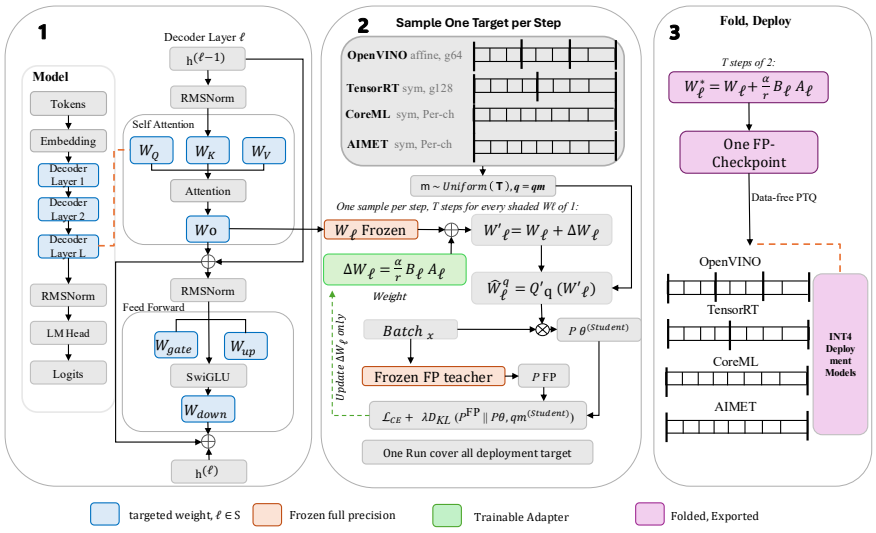

Figure 2: **Overview of SCOPE-Q. (1)** The seven linear projections of every decoder layer receive a low-rank update $\Delta W_\ell = \frac{\alpha}{r} B_\ell A_\ell$. **(2)** At every step, a deployment target m is sampled uniformly, and its configuration $q = q_m$ is applied to the merged weight $W_\ell + \Delta W_\ell$; the quantized student is trained with language-modeling and distillation losses toward the frozen full-precision teacher, and only ΔW_ℓ receives gradients. **(3)** The update is folded into the weights, yielding one floating-point checkpoint that each framework quantizes with its own procedure.

the scales and zero points of Q_q are recomputed from $\widetilde{W}_\ell$ at every forward pass instead of being learned, which matches how the frameworks derive them at deployment. Gradients pass through rounding with the straight-through estimator (Bengio et al., 2013). Because the target is drawn independently of the minibatch, the per-step gradient is an unbiased estimate of the gradient of the expected objective in Equation (4) under the straight-through surrogate, and sampling one target per step introduces no factor-of-M quantizer cost relative to the same low-rank distillation recipe trained against a single configuration. Because the same update is optimized under repeatedly sampled configurations, improvements that depend narrowly on one quantization grid are disfavored.

4.3 Distillation Objective

Let $p_{\mathrm{FP}}(\cdot \mid x_{\le j})$ and $p_{\theta,q_m}(\cdot \mid x_{\le j})$ be the next-token distributions of the frozen full-precision model and of the quantized model under the configuration q_m of target m. We combine the language-modeling loss $\mathcal{L}_{\mathrm{CE}} = -\sum_j \log p_{\theta,q_m}(x_{j+1} \mid x_{\le j})$ with a distillation loss $\mathcal{L}_{\mathrm{KD}} = \sum_j D_{\mathrm{KL}}(p_{\mathrm{FP}} \,\|\, p_{\theta,q_m})$, which gives a dense target at every position:

$$\mathcal{L}_{\mathrm{SCOPE}} = \mathbb{E}_{m\sim p(\mathcal{T})}\, \mathbb{E}_{x\sim\mathcal{D}} \left[\mathcal{L}_{\mathrm{CE}} + \lambda\, \mathcal{L}_{\mathrm{KD}}\right]. \tag{7}$$

4.4 Folding and Export

After training, the update is merged, $W^\star_\ell = W_\ell + \frac{\alpha}{r} B_\ell A_\ell$, and A_ℓ and B_ℓ are discarded (Algorithm 1). Because training quantized the same sum, folding is exact. The exported artifact is an ordinary floating-point checkpoint with no adapter or quantizer state, and each framework obtains its quantized weights as $Q_q(W^\star_\ell)$ with its own data-free procedure, so M frameworks cost one training run and M data-free quantizations.

Algorithm 1 SCOPE-Q: configuration-sampled QAT with folding

Require: pretrained weights $\{W_\ell\}_{\ell \in \mathcal{S}}$, data $\mathcal{D}$, targets $\mathcal{T}$, rank r, scale α, weight λ, steps N

Initialize A_ℓ randomly and $B_\ell \leftarrow 0$ for every $\ell \in \mathcal{S}$
for $i = 1, \ldots, N$ **do**
  Sample a batch $x \sim \mathcal{D}$ and a target $m \sim p(\mathcal{T})$, and set $q \leftarrow q_m$
  $\widehat{W}_\ell^{(q)} \leftarrow Q_q\big(W_\ell + \frac{\alpha}{r} B_\ell A_\ell\big)$ for every ℓ, recomputing the scales of q
  $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{CE}}(x) + \lambda\, \mathcal{L}_{\mathrm{KD}}\big(p_{\mathrm{FP}}(\cdot \mid x) \,\|\, p_{\theta,q_m}(\cdot \mid x)\big)$
  Update $\{A_\ell, B_\ell\}$ with $\nabla\mathcal{L}$, using the straight-through estimator for rounding
end for
return $W_\ell^\star \leftarrow W_\ell + \frac{\alpha}{r} B_\ell A_\ell$ for every ℓ ▷ floating-point checkpoint

5 Experiments

5.1 Protocol

Models and frameworks. We evaluate Qwen2.5-0.5B and Qwen2.5-1.5B (Qwen Team, 2024), Qwen3-1.7B (Qwen Team, 2025), and Gemma-4 E2B (Gemma Team, 2026), using the text model of the `gemma-4-E2B-it` checkpoint (Google, 2026); together they span 0.5B to 4.6B parameters and include base and post-trained releases, and we add Qwen2.5-7B to study scale. Each checkpoint is quantized with 4-bit weights at the native configurations of Table 1, with activations, embeddings, and the language-model head in floating point, and every reported MMLU accuracy is produced by the framework's own quantizer.

Evaluation. We measure 5-shot accuracy on a fixed, balanced MMLU subset (Hendrycks et al., 2021) consisting of the first 100 test questions of each subject (5,700 questions across 57 subjects), identical across all experiments, scored by the argmax over the next-token logits of the four answer letters. We call a difference significant when its 95% interval under a paired bootstrap ($B = 10{,}000$) excludes zero; the bootstrap resamples questions, or MMLU subjects where per-question predictions are unavailable, and so measures evaluation-set variability for the reported checkpoints, while training-run variability is estimated from three seeds on Qwen2.5-0.5B.

Baselines. *RTN* applies each framework's data-free round-to-nearest quantizer to the pretrained checkpoint. *Per-framework QAT* runs each framework's own QAT path at its native configuration, namely TensorRT Model Optimizer, AIMET, NNCF with rank-128 LoRA, and ANEMLL for Core ML, all on WikiText-2, and produces one checkpoint per framework. *Calibration-based PTQ* applies AWQ in OpenVINO, GPTQ in TensorRT and Core ML, and Sequential MSE in AIMET, with 128 WikiText-2 sequences of 128 tokens. *Single-configuration training* applies the SCOPE-Q recipe to one configuration at a time.

Implementation. SCOPE-Q uses rank $r = 32$, $\alpha = 64$, and $\lambda = 1$, and trains for 2,000 steps with AdamW at learning rate 10^{-4}, sequence length 512, and batch size 8 on WikiText-2 (Merity et al., 2017), about 8.2M tokens. Training does not invoke the deployment toolchains: targets are sampled uniformly and their configurations applied by configuration-equivalent fake quantizers implemented in PyTorch, with Core ML and AIMET sharing one per-channel simulator, and the folded checkpoint is then evaluated with each framework's own quantizer. We report one run per model and the mean of three seeds for Qwen2.5-0.5B, with standard deviations of 0.23 points for mean and 0.75 points for worst-case accuracy. Every run uses a single NVIDIA RTX A6000 GPU (48 GB); one SCOPE-Q run takes 59, 149, 173, and 209 minutes on Qwen2.5-0.5B, Qwen2.5-1.5B, Qwen3-1.7B, and Gemma-4 E2B, and covers all four frameworks in place of four per-framework runs.

5.2 Magnitude and source of the gap

The RTN rows of Table 2 quantify the gap. Although every framework uses 4-bit weights, accuracy spans 12.13–15.11 points across the four frameworks, and the per-channel configurations are always the weakest, group-64 the strongest, and group-128 in between, an ordering qualitatively consistent with the larger step-size bounds of coarser grouping (Equation (2)), although the native configura-

Table 2: **One-checkpoint robustness under native configurations.** MMLU accuracy (%) over the four frameworks of Table 1. FP checkpoints is the number of distinct floating-point checkpoints trained or released for the four frameworks, and ΔWorst is the change in worst-case accuracy relative to RTN. Best values per model in bold. *Mean of three seeds.

Model	Method	FP checkpoints	Mean ↑	Worst ↑	Gap ↓	ΔWorst
Qwen2.5-0.5B	RTN	1	38.30	32.49	13.47	–
	Per-framework QAT	4	41.76	33.00	14.02	+0.51
	Calibration-based PTQ	1	39.21	27.19	18.63	−5.30
	SCOPE-Q*	1	**42.62**	**40.13**	**5.79**	+7.64
Qwen2.5-1.5B	RTN	1	53.54	46.65	12.86	–
	Per-framework QAT	4	56.94	51.54	8.90	+4.89
	Calibration-based PTQ	1	54.54	44.33	14.93	−2.32
	SCOPE-Q	1	**57.13**	**54.25**	**5.73**	+7.60
Qwen3-1.7B	RTN	1	52.65	47.54	12.13	–
	Per-framework QAT	4	57.22	50.68	9.78	+3.14
	Calibration-based PTQ	1	55.07	48.33	11.93	+0.79
	SCOPE-Q	1	**57.71**	**55.95**	**3.82**	+8.41
Gemma-4 E2B	RTN	1	38.63	31.89	15.11	–
	Per-framework QAT	4	**46.29**	34.95	20.05	+3.06
	Calibration-based PTQ	1	44.83	**41.63**	**6.14**	+9.74
	SCOPE-Q	1	43.09	39.35	8.47	+7.46

tions also differ in their scaling rules. Constraining all four frameworks to symmetric group-128 reduces the RTN gap to 0.28–5.65 points (Table 4), which suggests that the configuration is the dominant contributor in our setting and that implementation differences account for the remainder. As a toolchain-to-device sanity check, an earlier SCOPE-Q checkpoint, trained with the AIMET configuration simulated as affine per-channel, matches its toolchain accuracy on the OpenVINO CPU runtime and on the Apple Neural Engine to within 0.23 points.

5.3 One-checkpoint robustness

Table 2 compares methods by the number of checkpoints they require and by their accuracy across the four frameworks. Relative to RTN, SCOPE-Q improves worst-case accuracy by 7.46–8.41 points and reduces the gap by 44–69%, to 3.82–8.47 points, on all four models; for the seed-0 Qwen2.5-0.5B checkpoint, all three improvements are significant under evaluation-set resampling. The improvement concentrates on the coarsest configurations: on Qwen2.5-0.5B, Core ML accuracy rises from 32.49 to 40.13 while OpenVINO accuracy is unchanged. The improvement extends beyond MMLU: under simulated versions of the four configurations, SCOPE-Q reduces the worst-case perplexity of Qwen2.5-0.5B by 48% on WikiText-2 and 39% on C4 (Raffel et al., 2020) and recovers 40–67% of the worst-case accuracy lost to RTN on ARC-Challenge, HellaSwag, and WinoGrande (Clark et al., 2018; Zellers et al., 2019; Sakaguchi et al., 2020).

Per-framework QAT uses each framework's own recipe, so it provides practical deployment context rather than the controlled comparison, which Section 5.4 supplies. It maintains four checkpoints and improves every framework over RTN, but its worst-case accuracy rises by only 0.51–4.89 points and its gap remains 8.90–20.05 points. Under the evaluated framework-specific QAT recipes, the single SCOPE-Q checkpoint exceeds the four per-framework checkpoints on all three statistics for the three models up to 1.7B; on Gemma-4 E2B, per-framework QAT attains the highest mean accuracy, while SCOPE-Q attains higher worst-case accuracy and a smaller gap. SCOPE-Q is likewise ahead of calibration-based PTQ on these three models, significantly so in worst-case accuracy and gap on Qwen3-1.7B; the margin is largest on Core ML, where GPTQ at per-channel granularity improves on RTN by at most 0.8 points. On Gemma-4 E2B, calibration-based PTQ attains higher point estimates than SCOPE-Q, and the differences are not significant.

Table 3: **Training on individual configurations versus all deployment targets.** Each row applies the SCOPE-Q recipe, with identical rank, data, budget, and objective, to one configuration or to all four targets; statistics are over the four frameworks. Best values per model in bold. *Mean of three seeds.

	Qwen2.5-0.5B			Qwen3-1.7B		
Trained against	Mean ↑	Worst ↑	Gap ↓	Mean ↑	Worst ↑	Gap ↓
None (RTN)	38.30	32.49	13.47	52.65	47.54	12.13
OpenVINO configuration only	39.04	32.25	14.98	54.55	49.79	10.61
TensorRT configuration only	41.20	36.54	8.46	54.97	51.93	6.88
Per-channel configuration only	39.89	37.60	**3.72**	53.40	51.89	3.95
All four targets (SCOPE-Q)	**42.62***	**40.13***	5.79*	**57.71**	**55.95**	**3.82**

Table 4: **Native versus common configuration.** Worst-case accuracy and gap over the four frameworks with their native configurations (Table 1) or with symmetric group-128 on all four. Best values per model in bold. †Seed 0 for both SCOPE-Q columns, so that native and common results come from the same checkpoint.

	RTN				SCOPE-Q			
	Native		Common		Native		Common	
Model	Worst	Gap	Worst	Gap	Worst	Gap	Worst	Gap
Qwen2.5-0.5B	32.49	13.47	39.33	5.65	40.96†	4.20†	**43.30**†	**2.24**†
Qwen2.5-1.5B	46.65	12.86	57.95	**0.28**	54.25	5.73	**58.16**	0.93
Qwen3-1.7B	47.54	12.13	55.18	2.82	55.95	3.82	**58.23**	**0.35**
Gemma-4 E2B	31.89	15.11	40.49	**0.95**	39.35	8.47	**41.21**	4.40

5.4 Effect of multi-configuration training

In Table 3, we train the same recipe, with identical rank, data, budget, and objective, against one configuration at a time and evaluate each checkpoint on all four frameworks. The best single-configuration run improves worst-case accuracy over RTN by 5.11 and 4.39 points on the two models, against 7.64 and 8.41 for SCOPE-Q; multi-configuration training therefore contributes 33% and 48% of the worst-case gain. The difference is significant under evaluation-set resampling for the seed-0 Qwen2.5-0.5B checkpoint and for Qwen3-1.7B, and on Qwen2.5-0.5B each of the three training seeds of SCOPE-Q exceeds the best single-configuration run, by 1.91–3.36 points. Single-configuration training transfers poorly to finer configurations: the per-channel-only run loses 4.64 and 6.44 points on OpenVINO relative to RTN, and its small gap on Qwen2.5-0.5B comes from this loss on the strongest framework. SCOPE-Q loses at most 0.04 points relative to RTN on any framework of either model, exceeds the per-channel-only run on both per-channel frameworks while sampling per-channel quantization on half of the steps, and on Qwen3-1.7B stays within 0.63 points of the best single-configuration run on every framework. Because the best single configuration differs between the two models, identifying it in advance would require training every candidate. On Qwen2.5-0.5B, replacing the expectation with a hard or smooth worst-case objective lowers the gap by about 1.2 points at a cost of about 1.0 point in mean accuracy, differences much smaller than the gain over RTN, so we keep the expectation, which has an unbiased stochastic gradient and no additional hyperparameters.

Multi-configuration training does not appear to act by shrinking the quantization step. On Qwen2.5-0.5B at symmetric per-channel granularity, SCOPE-Q and the per-channel-only run change the median step size (1.04×10^{-2}) and the relative rounding error (0.179) by less than 0.2%. This suggests that the improvement is not driven primarily by reducing these aggregate rounding-error statistics; it is consistent with a functionally better placement of the residual error, although we do not measure sensitive directions directly. Nor does the gain come from fine-tuning on the training corpus: without quantization, the folded checkpoint's WikiText-2 perplexity rises from 12.51 to 12.77, whereas after quantization it is 14.03 against 22.07 for the pretrained model.

5.5 Comparison with a common configuration

A common configuration is a simpler alternative, and it is effective (Table 4): besides reducing the RTN gap (Section 5.2), symmetric group-128 on all four frameworks raises RTN's worst-case accuracy by 6.8–11.3 points. On Qwen2.5-1.5B and Gemma-4 E2B, common-configuration RTN reaches a higher worst-case accuracy than native SCOPE-Q, and on the other two models native SCOPE-Q is higher by 0.8–1.6 points. The two approaches address different settings. A common configuration requires control over how every framework quantizes the model, whereas our setting assumes that downstream users quantize a released checkpoint independently with each framework's native configuration. A universally supported common configuration may also differ from the configuration preferred for a particular hardware target, as the TensorRT and Neural Engine examples of Section 1 illustrate. The approaches are complementary: quantized with the common configuration, the SCOPE-Q checkpoint exceeds common-configuration RTN in worst-case accuracy on all four models, by 0.2–4.0 points.

5.6 Scale and scope

At 7B the native RTN gap is smaller, 5.74 points. SCOPE-Q reduces it to 3.21 points and raises worst-case accuracy from 69.33 to 71.53 with 1.06% of the parameters trained. Per-framework QAT fits in the memory of a single GPU for only three frameworks and reaches 73.63 over those three. This comparison is not matched, but it suggests that framework-specific QAT can be stronger where it is feasible, while SCOPE-Q preserves the one-checkpoint objective and improves over RTN on all four frameworks. At 7B, SCOPE-Q's parameter states occupy about 32 GB before activations: 15.2 GB each for the frozen fp16 weights and the full-precision teacher, which can instead reside on the CPU, and 1.3 GB for the fp32 adapter parameters, gradients, and AdamW moments. Under an optimistic lower-bound accounting that stores weights, gradients, and both moment buffers in fp16, full-model QAT needs about 61 GB before activations, and fp32 moments would need more. The gains of SCOPE-Q are specific to the configurations represented in training. On two configurations not represented by any target, symmetric group-32 and per-tensor quantization, the SCOPE-Q and pretrained checkpoints perform alike: both reach 46.04 at group-32, and per-tensor quantization reduces both to chance.

6 Conclusion

We studied the accuracy variation that arises when one LLM checkpoint is quantized independently with the native configurations of different deployment frameworks, and found a gap of 12–15 MMLU points that is dominated by the quantization configuration. SCOPE-Q trains a low-rank update under configurations sampled from the deployment set and exports one portable floating-point checkpoint. Across four models it improves worst-case accuracy by 7.5–8.4 points and reduces the gap by 44–69%, and training against multiple configurations rather than one accounts for one third to one half of the worst-case gain.

Limitations and future work. SCOPE-Q provides robustness across a predefined set of deployment configurations rather than across arbitrary quantizers: configurations absent from training receive little or no benefit, so a new deployment target must be added to $\mathcal{T}$, together with its associated configuration. When one party controls every deployment, a common configuration is a strong and simpler alternative, and at 7B framework-specific QAT can be more accurate where it is feasible. Our evaluation covers 4-bit weight-only quantization with MMLU as the primary benchmark. Extending configuration sampling to activation quantization and mixed precision, adapting the sampling distribution toward the most sensitive configurations, and combining it with calibration-based methods are natural directions.

Reproducibility Statement

Section 5.1 and Table 1 specify the models, frameworks, configurations, baselines, and hyperparameters, and the appendix provides the baseline settings, toolchain versions, and per-framework results for every experiment. All models are public Hugging Face checkpoints, and training and evaluation use public datasets (WikiText-2, C4, MMLU, ARC, HellaSwag, and WinoGrande). The MMLU subset consists of the first 100 test questions of each subject, taken in dataset order from the `cais/mmlu` test split without shuffling, so it is deterministic. Code, configurations, and per-question predictions will be released [Omitted for Review]. Because SCOPE-Q exports a standard floating-point checkpoint, all deployment results can be reproduced with the unmodified public toolchains.

Ethics Statement

This work improves the accuracy of deploying existing language models on diverse hardware and introduces no new model capabilities or training data. SCOPE-Q is trained to match the predictions of the full-precision model, and the resulting checkpoints inherit its behavior, biases, and limitations. More accurate on-device inference may broaden access to language models and allow private data to be processed locally, but it does not by itself address model safety or fairness. The study uses only publicly available models and datasets and involves no human subjects.

AI Use Statement

In this work, we used generative AI tools to provide feedback on the experimental methodology, including suggestions for control experiments and statistical analyses, to propose and refine hypotheses about observed results, and to help interpret results. We have not used generative AI tools to generate synthetic datasets, formulate mathematical claims, write proofs, or clean datasets, and translation and qualitative data analysis are not applicable to this work.

References

Milad Alizadeh, Arash Behboodi, Mart van Baalen, Christos Louizos, Tijmen Blankevoort, and Max Welling. Gradient ℓ_1 regularization for quantization robustness. In *International Conference on Learning Representations (ICLR)*, 2020.

Apple Inc. Core ML tools. https://github.com/apple/coremltools, 2024.

Apple Inc. Optimizing OPT model — guide to Core ML tools. https://apple.github.io/coremltools/docs-guides/source/opt-opt1_3.html, 2026a. Accessed September 2026.

Apple Inc. Performance — guide to Core ML tools. https://apple.github.io/coremltools/docs-guides/source/opt-quantization-perf.html, 2026b. Accessed September 2026.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2404.00456.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Yelysei Bondarenko, Riccardo Del Chiaro, and Markus Nagel. Low-rank quantization-aware training for LLMs. *arXiv preprint arXiv:2406.06385*, 2024.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2020.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.14314.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations (ICLR)*, 2024.

Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. BitDistiller: Unleashing the potential of sub-4-bit LLMs via self-distillation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *International Conference on Machine Learning (ICML)*, 2024.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1902.08153.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.17323.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Google. gemma-4-E2B-it model card. https://huggingface.co/google/gemma-4-E2B-it, 2026. Accessed September 2026.

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

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

Intel Corporation. Neural network compression framework (NNCF). https://github.com/openvinotoolkit/nncf, 2024.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Hyesung Jeon, Yulhwa Kim, and Jae-Joon Kim. L4Q: Parameter efficient quantization-aware fine-tuning on large language models. *arXiv preprint arXiv:2402.04902*, 2024.

Qing Jin, Linjie Yang, and Zhenyu Liao. AdaBits: Neural network quantization with adaptive bit-widths. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. LoftQ: LoRA-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Yuhang Li, Mingzhu Shen, Jian Ma, Yan Ren, Mingxin Zhao, Qi Zhang, Ruihao Gong, Fengwei Yu, and Junjie Yan. MQBench: Towards reproducible and deployable model quantization benchmark. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021b.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024. arXiv:2306.00978.

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. In *Findings of the Association for Computational Linguistics (ACL)*, 2024.

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. In *International Conference on Learning Representations (ICLR)*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *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 Corporation. TensorRT model optimizer. https://github.com/NVIDIA/TensorRT-Model-Optimizer, 2024.

Qualcomm Innovation Center. AIMET: AI model efficiency toolkit. https://github.com/quic/aimet, 2024.

Qualcomm Innovation Center. Low-power blockwise quantization (LPBQ) — AIMET documentation. https://quic.github.io/aimet-pages/releases/latest/techniques/lpbq.html, 2026. Accessed September 2026.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *AAAI Conference on Artificial Intelligence*, 2020.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2308.13137.

Mingzhu Shen, Feng Liang, Ruihao Gong, Yuhang Li, Chuming Li, Chen Lin, Fengwei Yu, Junjie Yan, and Wanli Ouyang. Once quantization-aware training: High performance extremely low-bit architecture search. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. arXiv:2010.04354.

Moran Shkolnik, Brian Chmiel, Ron Banner, Gil Shomron, Yury Nahshan, Alex Bronstein, and Uri Weiser. Robust quantization: One model to rule them all. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with Hadamard incoherence and lattice codebooks. In *International Conference on Machine Learning (ICML)*, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2211.10438.

Ke Xu, Qiantai Feng, Xingyi Zhang, and Dong Wang. MultiQuant: Training once for multi-bit quantization of neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

Ke Xu, Lei Han, Ye Tian, Shangshang Yang, and Xingyi Zhang. EQ-Net: Elastic quantization neural networks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. arXiv:2308.07650.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. QA-LoRA: Quantization-aware low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Haichao Yu, Haoxiang Li, Humphrey Shi, Thomas S. Huang, and Gang Hua. Any-precision deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. arXiv:1911.07346.

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

A Implementation details

One target is drawn uniformly per optimization step (Algorithm 1), so that in expectation every target is sampled equally often. The frozen full-precision teacher is a separate copy of the pretrained model, held on the GPU or, to save memory, on the CPU, and is evaluated without the low-rank update or quantization. At rank 32, the adapters amount to 17.6 M parameters on Qwen2.5-0.5B (3.56%), 36.9 M on Qwen2.5-1.5B (2.39%), 34.9 M on Qwen3-1.7B (2.03%), 52.7 M on Gemma-4 E2B (1.14%), and 80.7 M on Qwen2.5-7B (1.06%), counting tied embeddings once.

B Baseline settings and per-framework results

Table 5 lists the training settings of the per-framework QAT baselines, all of which use WikiText-2. Combined, the four per-framework runs process 12.2M tokens on the Qwen models and 11.1M tokens on Gemma-4 E2B, against 8.2M tokens for the single SCOPE-Q run. Table 6 gives the per-framework accuracies behind Table 2 and Figure 1.

Table 5: Training settings of the per-framework QAT baselines and of SCOPE-Q. All runs use WikiText-2. [a]1,000 steps on Gemma-4 E2B. [b]Gemma-4 E2B: $r = 256$, 2 epochs, learning rate 10^{-4}. [c]Gradient accumulation of 8.

Framework (QAT path)	Training data	Budget	Learning rate
TensorRT (Model Optimizer)	seq. 512, batch 8	500 steps[a]	10^{-5}
AIMET	seq. 512, batch 8	1,000 steps	10^{-5}
OpenVINO (NNCF, LoRA $r = 128$)	$1{,}024 \times 1{,}024$ tokens	5 epochs[b]	10^{-5}
Core ML (ANEMLL)	512×256 tokens	6 epochs[c]	10^{-5}
SCOPE-Q (all four)	seq. 512, batch 8	2,000 steps	10^{-4}

Table 6: **Per-framework results under native configurations.** MMLU accuracy (%) with OpenVINO (OV, affine group-64), TensorRT (TRT, symmetric group-128), AIMET (AIM), and Core ML (CML) (both symmetric per-channel). Per-framework QAT trains one checkpoint per framework; calibration-based PTQ uses AWQ (OV), GPTQ (TRT, CML), and Sequential MSE (AIM); SCOPE-Q is one checkpoint quantized with RTN by every framework. *Mean of three seeds.

Model	Method	OV	TRT	AIM	CML	Mean ↑	Worst ↑	Gap ↓
Qwen2.5-0.5B	RTN	45.96	39.46	35.28	32.49	38.30	32.49	13.47
	Per-framework QAT	47.02	44.65	42.37	33.00	41.76	33.00	14.02
	Calibration-based	45.82	42.25	41.58	27.19	39.21	27.19	18.63
	SCOPE-Q*	45.92	43.19	41.24	40.13	42.62	40.13	5.79
Qwen2.5-1.5B	RTN	59.51	58.23	49.77	46.65	53.54	46.65	12.86
	Per-framework QAT	60.44	58.58	57.18	51.54	56.94	51.54	8.90
	Calibration-based	59.26	57.63	56.91	44.33	54.54	44.33	14.93
	SCOPE-Q	59.98	58.16	56.14	54.25	57.13	54.25	5.73
Qwen3-1.7B	RTN	59.67	55.18	48.19	47.54	52.65	47.54	12.13
	Per-framework QAT	60.46	59.39	58.35	50.68	57.22	50.68	9.78
	Calibration-based	60.26	55.88	55.82	48.33	55.07	48.33	11.93
	SCOPE-Q	59.77	58.23	55.95	56.88	57.71	55.95	3.82
Gemma-4 E2B	RTN	47.00	41.19	34.42	31.89	38.63	31.89	15.11
	Per-framework QAT	48.72	55.00	46.47	34.95	46.29	34.95	20.05
	Calibration-based	47.44	47.77	41.63	42.47	44.83	41.63	6.14
	SCOPE-Q	47.82	45.53	39.35	39.65	43.09	39.35	8.47

C Frameworks and configurations

Table 7 lists the toolchain versions and the 4-bit weight granularities each framework accepts, and Figure 3 illustrates the native partitions. Symmetric group-128 is accepted by all four frameworks,

which enables the common-configuration experiments of Appendix I. The native configurations are consistent with each vendor's recommended path: Apple positions per-block 4-bit quantization for the Mac GPU and recommends per-channel scales for the Neural Engine (Apple Inc., 2026b;a), and Qualcomm's low-power blockwise format uses block scales that are integer multiples of a per-channel floating-point scale (Qualcomm Innovation Center, 2026). Models are loaded from the Hugging Face Hub (`Qwen/Qwen2.5-0.5B`, `Qwen/Qwen2.5-1.5B`, `Qwen/Qwen3-1.7B`, Qwen2.5-7B, and the text model of `google/gemma-4-E2B-it`); training uses PyTorch 2.5.1 and Transformers 4.57.1, and evaluation uses Transformers 5.12.1.

Table 7: Framework versions, native configurations, and other supported 4-bit weight granularities.

Framework	Toolchain (version)	Native configuration	Other granularities
OpenVINO	NNCF 3.3.0 (OpenVINO 2026.2.1)	affine group-64	per-channel, g32, g128
TensorRT	ModelOpt 0.45.0	symmetric group-128	configurable block size; no per-channel mode
Core ML	coremltools 9.0	symmetric per-channel	g32, g64, g128
AIMET	AIMET 2.11.0 (torch 2.4.0)	symmetric per-channel	g32, g64, g128

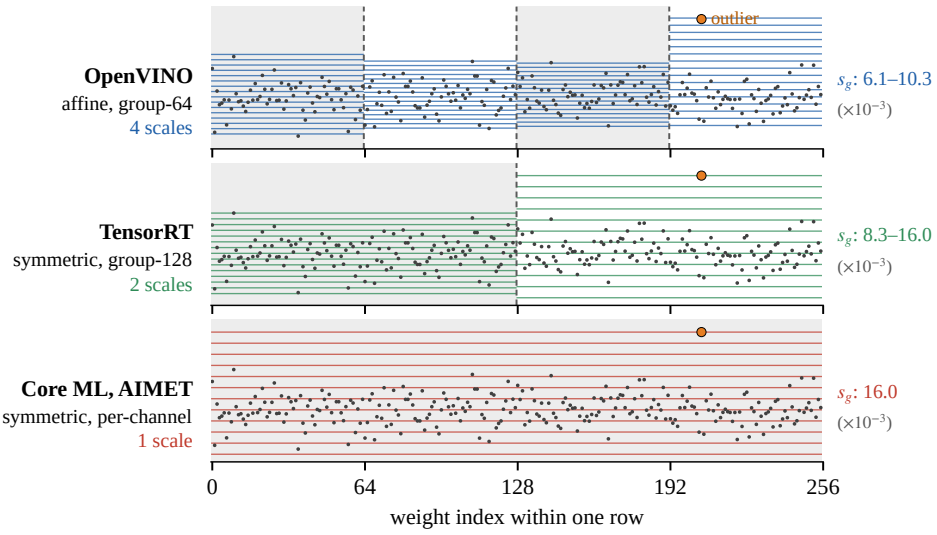

Figure 3: **Quantization granularity of the native configurations.** An illustrative row of 256 weights (dots) with one outlier (orange), and the 4-bit quantization levels (horizontal lines) of each configuration within each group (alternately shaded; dashed lines mark group boundaries). Symmetric configurations place 15 levels symmetrically around zero, and the affine configuration places 16 levels across the range of each group. Because all weights in a group share one scale, an outlier can dominate the group's dynamic range and enlarge its step size s_g: per-channel quantization coarsens the grid for the whole row, whereas group-128 and group-64 confine the coarser grid to the outlier's group. A coarser partition can therefore enlarge the step size and the corresponding worst-case rounding-error bound.

D Statistical significance and seed variance

For two checkpoints evaluated on the same questions, we resample question indices with replacement $B = 10{,}000$ times, recompute all four framework accuracies of both checkpoints on each resample, and take the 2.5th and 97.5th percentiles of the differences in mean accuracy, worst-case accuracy, and gap (Table 8). Resampling the same indices for every framework and both checkpoints preserves the correlation between them. Where per-question predictions are unavailable, we resample the 57 MMLU subjects instead, which yields wider intervals. Table 9 reports three training seeds on

Qwen2.5-0.5B; the seed varies the adapter initialization and the order in which configurations are sampled.

Table 8: Paired bootstrap 95% intervals ($B = 10{,}000$) for differences in mean accuracy, worst-case accuracy, and gap. Point estimates are differences of the reported values (seed 0 on Qwen2.5-0.5B). †Resampled by MMLU subject.

Model	SCOPE-Q minus	Δ Mean	Δ Worst	Δ Gap
Qwen2.5-0.5B	RTN	$+4.39$ $[+3.09, +5.64]$	$+8.47$ $[+6.65, +10.02]$	-9.28 $[-11.16, -7.14]$
Qwen2.5-0.5B	Per-channel configuration only	$+2.79$ $[+1.45, +4.13]$	$+3.36$ $[+1.47, +5.00]$	$+0.47$ $[-1.18, +2.23]$
Qwen3-1.7B†	TensorRT configuration only	$+2.74$ $[-0.55, +6.00]$	$+4.02$ $[+0.56, +7.44]$	-3.06 $[-4.86, -1.23]$
Qwen3-1.7B†	Calibration-based PTQ	$+2.64$ $[-1.60, +7.09]$	$+7.62$ $[+3.47, +11.96]$	-8.11 $[-10.18, -5.96]$
Gemma-4 E2B†	Calibration-based PTQ	-1.74 $[-5.08, +1.71]$	-2.28 $[-5.70, +1.18]$	$+2.33$ $[-0.16, +4.63]$

Table 9: Seed variance of SCOPE-Q on Qwen2.5-0.5B.

Run	OV	TRT	AIM	CML	Mean	Worst	Gap
Seed 0	45.16	43.30	41.32	40.96	42.69	40.96	4.20
Seed 1	46.14	44.28	41.30	39.51	42.81	39.51	6.63
Seed 2	46.46	41.98	41.11	39.91	42.36	39.91	6.54
Mean	45.92	43.19	41.24	40.13	42.62	40.13	5.79
Std.	0.68	1.15	0.12	0.75	0.23	0.75	1.38

E Quantization cost and fine-tuning

Evaluating trained checkpoints without quantization separates what training changes in the full-precision model from what quantization then costs (Table 10). We perform this decomposition at the TensorRT configuration, whose QAT path allows the quantizer to be disabled. On the Qwen models, the unquantized accuracy of SCOPE-Q is slightly below that of the pretrained model, so its quantized gains reflect quantization tolerance rather than fine-tuning, and its quantization cost is lower than that of TensorRT QAT on Qwen2.5-1.5B and Qwen3-1.7B. On Gemma-4 E2B, TensorRT's full-model QAT raises unquantized accuracy by 10.16 points, which accounts for its lead on that framework.

Table 10: Unquantized and quantized MMLU accuracy of trained checkpoints at the TensorRT configuration (symmetric group-128). "vs. pretrained" is the unquantized change relative to the pretrained checkpoint (49.05, 61.77, 62.58, and 49.00 for the four models). Quantized values come from separate evaluations and differ from Table 2 by at most 0.19 points.

Model	Checkpoint	Unquantized	vs. pretrained	Quantized	Quantization cost
Qwen2.5-0.5B	SCOPE-Q	47.77	-1.28	43.30	-4.47
	TensorRT QAT	48.47	-0.58	44.46	-4.02
Qwen2.5-1.5B	SCOPE-Q	61.09	-0.68	58.16	-2.93
	TensorRT QAT	61.63	-0.14	58.40	-3.23
Qwen3-1.7B	SCOPE-Q	59.82	-2.76	58.23	-1.59
	TensorRT QAT	62.16	-0.42	59.39	-2.77
Gemma-4 E2B	SCOPE-Q	49.82	$+0.82$	45.53	-4.29
	TensorRT QAT	59.16	$+10.16$	55.00	-4.16

F Single-configuration ablation

Tables 11 and 12 give the per-framework accuracies behind Table 3. The per-channel-only run attains a small gap partly by lowering its best framework, which is why we report the gap together with worst-case accuracy.

Table 11: **Single-configuration ablation on Qwen2.5-0.5B.** Each row trains the SCOPE-Q recipe against one configuration, with rank, data, optimizer, budget, and objective identical to the full method. Underlined entries are each run's own configuration; one per-channel run serves both per-channel frameworks.

Trained against	OV	TRT	AIM	CML	Mean ↑	Worst ↑	Gap ↓
OpenVINO configuration only	47.23	39.21	37.49	32.25	39.04	32.25	14.98
TensorRT configuration only	45.00	44.89	36.54	38.35	41.20	36.54	8.46
Per-channel configuration only	41.32	39.82	40.82	37.60	39.89	37.60	3.72
All four targets (SCOPE-Q)*	45.92	43.19	41.24	40.13	42.62	40.13	5.79
None (RTN)	45.96	39.46	35.28	32.49	38.30	32.49	13.47

Table 12: Single-configuration ablation on Qwen3-1.7B. Columns as in Table 11.

Trained against	OV	TRT	AIM	CML	Mean ↑	Worst ↑	Gap ↓
OpenVINO configuration only	60.40	56.60	51.40	49.79	54.55	49.79	10.61
TensorRT configuration only	55.96	58.81	53.19	51.93	54.97	51.93	6.88
Per-channel configuration only	53.23	52.63	51.89	55.84	53.40	51.89	3.95
All four targets (SCOPE-Q)	59.77	58.23	55.95	56.88	57.71	55.95	3.82
None (RTN)	59.67	55.18	48.19	47.54	52.65	47.54	12.13

G WEIGHT-RANGE ANALYSIS

If training improved accuracy by shrinking the dynamic range of each group, the step sizes of the trained checkpoints would be smaller than those of the pretrained model. Table 13 shows that both SCOPE-Q and the per-channel-only run change the median step size and the relative rounding error by less than 0.2%, while improving Core ML accuracy by 8.47 and 5.11 points. This suggests that the gain is not explained by these aggregate statistics; it is consistent with a functionally better placement of the residual error, although the analysis does not measure sensitive directions directly.

Table 13: Median step size and relative rounding error $\left(\sum_\ell \|Q_q(W_\ell) - W_\ell\|_F^2 / \sum_\ell \|W_\ell\|_F^2\right)^{1/2}$ over the 168 quantized layers of Qwen2.5-0.5B at symmetric per-channel granularity.

Checkpoint	Median s_g	Relative rounding error
Pretrained	1.039×10^{-2}	0.1791
SCOPE-Q (all four)	1.040×10^{-2}	0.1792
Per-channel only	1.040×10^{-2}	0.1794

H RESULTS BEYOND MMLU

Table 14 evaluates Qwen2.5-0.5B on WikiText-2 and C4 perplexity and on zero-shot ARC-Challenge, HellaSwag, and WinoGrande. SCOPE-Q reduces the worst-case perplexity by 48% on WikiText-2 and 39% on C4, a corpus it is not trained on. On the multiple-choice suites, where RTN loses 1.4–4.1 points on the worst configuration, SCOPE-Q recovers 40–67% of that loss. Without quantization, SCOPE-Q does not improve on the pretrained model on any metric, including WikiText-2; its gain lies in quantization tolerance.

Table 14: Language modeling and zero-shot accuracy on Qwen2.5-0.5B under the four framework configurations (simulated). "Mean" and "worst" are taken over the four configurations; perplexity (ppl) is lower-better. HellaSwag uses the first 2,000 validation examples; ARC-Challenge and WinoGrande use the full evaluation sets.

Checkpoint	Weights	WikiText-2 ppl ↓	C4 ppl ↓	ARC-C ↑	HellaSwag ↑	WinoGrande ↑
Pretrained	Unquantized	12.51	19.28	32.34	48.90	56.43
	Quantized, mean	22.07	32.76	30.18	47.10	55.86
	Quantized, worst	28.29	40.91	28.24	45.70	55.01
SCOPE-Q	Unquantized	12.77	20.40	31.31	47.60	56.27
	Quantized, mean	14.03	23.30	30.67	47.60	56.18
	Quantized, worst	14.64	24.83	29.86	47.05	55.96

I Common configuration

Table 15 constrains all four frameworks to symmetric group-128. The RTN gap falls from 12.13–15.11 to 0.28–5.65 points, which suggests that the configuration is the dominant contributor to the native gap and that the remainder reflects implementation differences between frameworks. Table 16 reports accuracy under this common configuration. SCOPE-Q quantized at group-128 exceeds RTN in mean and worst-case accuracy on all four models, so the benefit of training carries over when frameworks share a configuration; on Qwen2.5-0.5B the worst-case improvement is significant ($+3.97$, interval $[+0.14, +7.93]$). Compared with native SCOPE-Q, common-configuration RTN has lower mean accuracy on three of four models and worst-case accuracy that is higher on two models and lower on the other two.

Table 15: **Gap under a common configuration.** All four frameworks use symmetric group-128; the native gap is taken from Table 2. ‡Dominated by the difference between OpenVINO and TensorRT (4.32 points).

	Native gap	Common-configuration gap	
Model	RTN	RTN	SCOPE-Q
Qwen2.5-0.5B	13.47	5.65	2.24
Qwen2.5-1.5B	12.86	0.28	0.93
Qwen3-1.7B	12.13	2.82	0.35
Gemma-4 E2B	15.11	0.95	4.40‡

Table 16: **Common versus native configurations.** The first two groups use symmetric group-128 on all four frameworks; the last uses each framework's native configuration (Table 1). †Seed 0 for both SCOPE-Q groups.

	Harmonized RTN		SCOPE-Q at group-128		Native SCOPE-Q	
Model	Mean	Worst	Mean	Worst	Mean	Worst
Qwen2.5-0.5B	41.43	39.33	44.04†	43.30†	42.69†	40.96†
Qwen2.5-1.5B	58.07	57.95	58.60	58.16	57.13	54.25
Qwen3-1.7B	56.31	55.18	58.39	58.23	57.71	55.95
Gemma-4 E2B	41.19	40.49	44.17	41.21	43.09	39.35

J Training objective

We compare the expectation of Equation (7) with the hard minimax objective $\mathcal{L}_{\max} = \max_m \mathcal{L}_{q_m}$, its smooth relaxation $\mathcal{L}_{\text{soft}} = \tau \log \sum_m \exp(\mathcal{L}_{q_m}/\tau)$ with $\tau = 0.5$, and a minimax over losses normalized by their running means (Table 17). The worst-case objectives reduce the gap by about 1.2 points and mean accuracy by about 1.0 point, and normalization restores both to the level of the

expectation. The four objectives differ by at most 1.3 points on either statistic, much less than the improvement over RTN; we use the expectation, which has an unbiased stochastic gradient and no additional hyperparameters.

Table 17: **Training objectives on Qwen2.5-0.5B.** In this experiment the AIMET configuration is simulated as affine per-channel, so values differ from Table 2.

Objective	Mean ↑	Gap ↓
Uniform expectation	42.75	5.96
Hard minimax	41.66	4.80
Soft worst-case	41.75	4.76
Loss-normalized minimax	42.78	6.04

K Configurations outside the training set

SCOPE-Q optimizes the configurations of the targets in $\mathcal{T}$. On two configurations not represented by any target, symmetric group-32 (finer than every target configuration) and symmetric per-tensor (coarser), the SCOPE-Q and pretrained checkpoints perform alike (Table 18): both reach 46.04 at group-32, and per-tensor quantization reduces both to chance. A new deployment target is incorporated by adding it, with its configuration, to $\mathcal{T}$.

Table 18: Accuracy on configurations outside the training set of SCOPE-Q, Qwen2.5-0.5B.

Configuration	Pretrained	SCOPE-Q	Δ
Symmetric group-32	46.04	46.04	0.00
Symmetric per-tensor	25.11	24.74	−0.37

L Model scale and training memory

Table 19 reports Qwen2.5-7B. The native RTN gap is smaller at this scale, 5.74 points; SCOPE-Q reduces it to 3.21 and raises mean and worst-case accuracy by 1.22 and 2.20 points while training 1.06% of the parameters. Per-framework QAT reaches higher accuracy over the three frameworks whose QAT fits in memory, an indicative rather than matched comparison. The four QAT paths differ in what they train: OpenVINO trains absorbable rank-128 adapters on frozen quantized weights, TensorRT and AIMET optimize the targeted linear projections, and Core ML optimizes all weights together with learnable scales. Under an optimistic lower-bound accounting with fp16 weights, gradients, and both moment buffers, full-model QAT at 7B needs about 61 GB before activations, and Core ML's full-model QAT does not fit in the memory of a single GPU. SCOPE-Q's parameter states occupy 15.2 GB of frozen weights, 15.2 GB for the teacher, and 1.3 GB of fp32 adapter state for 80.7 M trainable parameters, about 32 GB before activations, or 17 GB with the teacher on the CPU.

Table 19: Qwen2.5-7B under each framework's native configuration. Per-framework QAT fits in the memory of a single GPU for three of the four frameworks, and its statistics cover those three.

Method	Frameworks	Mean ↑	Worst ↑	Gap ↓
RTN	4	71.83	69.33	5.74
Per-framework QAT	3	74.26	73.63	1.33
SCOPE-Q	4	73.05	71.53	3.21

M On-device validation

On the OpenVINO CPU runtime and the Apple Neural Engine, a folded Qwen2.5-0.5B checkpoint, trained with the AIMET configuration simulated as affine per-channel, reaches 46.26 and 41.12, against 46.35 and 40.89 in the toolchain evaluation, within the evaluation interval.