DISCRETE WEIGHT UPDATES FOR EXTREME LOW-BIT TRAINING WITHOUT MASTER WEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory consumption has become a major bottleneck for training and deploying increasingly large models. While extreme low-bit models with binary or ternary weights remarkably reduce the memory required for inference, their training still commonly relies on full-precision master weights that are optimized in a continuous space and quantized for the forward pass. Removing such master weights is challenging, as discrete weights can only change through transitions between a finite set of states and thus cannot directly represent the small continuous updates made by conventional optimizers. In this paper, we propose Discrete Updates for Extreme Low-bit Training (DUET), a scalable framework that directly trains extreme low-bit weights without maintaining master weights. We first derive a transition budget from the magnitude of a continuous optimization step, specifying the expected number of discrete weight transitions. Given this transition budget, we select the weight transitions that maximize the alignment of the resulting discrete update with the continuous step. We further show that optimizer states used for these transition decisions can be significantly compressed while retaining training quality. Our experiments demonstrate that DUET maintains competitive training quality over long training horizons and on large models, with training memory reductions that increase with model size and reach $12.0\times$ at 235B parameters compared to conventional training with master weights.

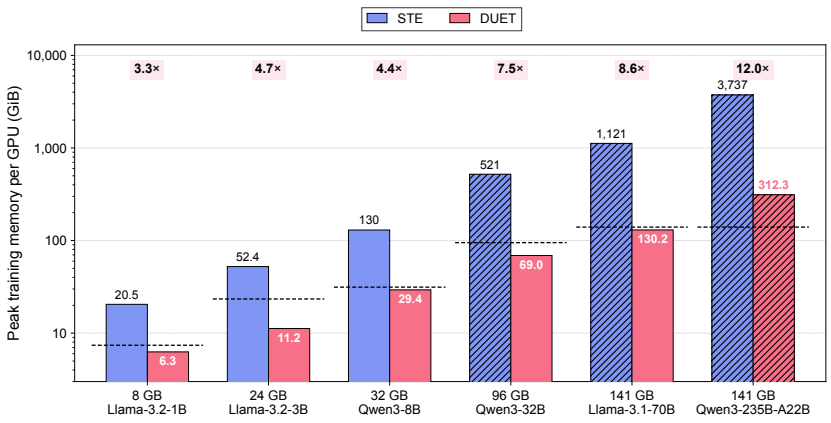


Figure 1: **Practical GPU training memory.** We report peak allocated memory for six models at batch size 1 and sequence length 256. DUET reduces memory for all models, fitting five within single-GPU capacity, while STE exceeds GPU capacity in every case.

1 INTRODUCTION

The memory requirement is still a key obstacle to training and deploying large models, as it often determines the number of GPUs required. Recent extreme low-bit models with binary or ternary weights reduce weight storage and memory bandwidth at inference, enabling cost-effective deployment (Ma et al., 2024; ML, 2026; Team, 2025). However, such memory efficiency does not extend to training. In conventional training schemes for quantized models, *e.g.*, quantization-aware training

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.

(QAT; Jacob et al., 2018), a low-bit model is trained with full-precision master weights (*i.e.*, latent weights), which are continuously updated through the straight-through estimator (STE; Bengio et al., 2013) and quantized for the forward pass. As a result, unlike their cost-effective inference, training extreme low-bit models costs almost as much as training full-precision models.

The gradients and optimizer states also require a large amount of memory, but they can be reduced by existing memory-efficient optimizers (Dettmers et al., 2022; Shazeer & Stern, 2018; Zhang et al., 2023). However, the master weights are challenging to replace, as they accumulate fine-grained updates during training; such updates cannot be individually captured by binary or ternary weights, which take only a few discrete values. To address this, we focus on that the weights affect the model through the activations, each of which sums up the contributions of many weights. Hence, a small change in the activations can be made by moving only a few weights by a full quantization interval, instead of slightly moving all of them. Our key idea is thus to control *how many* weights change, rather than *how much* each weight changes. Specifically, we use a continuous optimizer to set this number and select the weights to update, without maintaining master weights.

In this paper, we introduce DUET (**D**iscrete **U**pdates for **E**xtreme Low-Bit **T**raining), a scalable framework that directly trains binary and ternary weights. For each output channel, we set the transition budget (*i.e.*, the expected number of weight changes) to match the total magnitude of feasible continuous updates measured in quantization intervals. Stochastic rounding then gives an integer transition count without changing the expected total movement. Given this budget, minimizing the expected squared error from the continuous update in the same units yields a TopK allocation that selects weights with the largest update magnitudes. The learning signal of each weight accumulates in its momentum until the weight changes, after which we reset the momentum to prevent the same accumulated signal from driving repeated transitions. Since allocation depends on rankings and each transition has a fixed size, small quantization errors in the optimizer states can leave the discrete update unchanged. This motivates us to use compressed optimizer states, and direct gradient accumulation further reduces training memory without storing a full weight-gradient matrix.

Extensive experiments demonstrate that DUET enables effective and scalable full-parameter training of extreme low-bit LLMs. Across five Qwen3 (Yang et al., 2025) dense and MoE models from 8B to 235B, DUET consistently outperforms LoRA (Hu et al., 2022) adaptation of CAT-Q ternary checkpoints (Wang et al., 2026) in both average commonsense accuracy and validation perplexity. On Qwen3-8B, DUET improves average accuracy from 77.1% to 79.1% compared with STE-based full-parameter fine-tuning with 8-bit Adam (Dettmers et al., 2022), while reducing peak training memory by 61.6%. DUET also outperforms the STE-based AbsMean baseline (Ma et al., 2024) when converting pretrained Llama-3.2-1B (Grattafiori et al., 2024) to ternary weights through QAT on 10B tokens. To assess stability over long training runs, we pretrain models with 30M to 200M parameters from scratch using 100 tokens per parameter, achieving lower final validation loss than STE. At the 1B scale, DUET closely matches STE after 20B tokens of pretraining from scratch. We further demonstrate that DUET achieves greater training memory reductions as model size increases, reaching $12.0\times$ at 235B parameters compared to conventional training with master weights, and scales full-parameter training on a single H200 GPU from 8B with STE to 70B with ours (Figure 1).

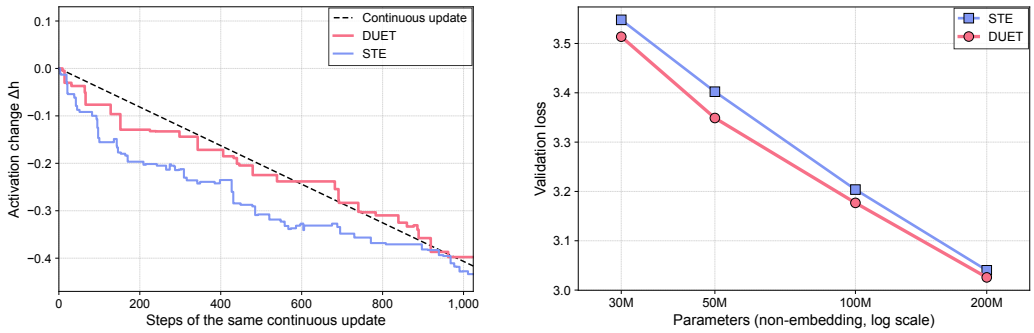
Overall, our work shows that extreme low-bit models can be trained without master weights, extending their memory efficiency from inference to training. This could make updating deployed models light enough to run on edge devices, such as robots adapting to new environments. We believe that DUET can inspire more researchers, even with limited GPU resources, to explore large models.

2 Method

Here, we introduce DUET (Discrete Updates for Extreme Low-Bit Training), a framework that directly trains binary and ternary weights without full-precision master weights. We convert each continuous optimizer update into a transition budget and allocate it to individual weights. We further reduce training memory through optimizer-state compression and direct gradient accumulation.

2.1 Quantization and Optimization Signal

We consider a linear layer with O output and I input channels and discrete weights $Q \in \mathcal{C}^{O\times I}$, where $\mathcal{C} = \{-1, +1\}$ for binary weights and $\mathcal{C} = \{-1, 0, +1\}$ for ternary weights. We learn positive



(a) Activation displacement **(b) Ternary pretraining at scale**

Figure 2: **Discrete update alignment and ternary training at scale.** We compare discrete updates for a fixed continuous update and ternary pretraining across model sizes. In (a), DUET better tracks the target activation displacement than STE over 1,000 steps. In (b), DUET achieves lower validation loss than STE across 30M–200M models after $100N$ tokens, where N is the parameter count.

group-wise scales α via LSQ (Esser et al., 2020). The operators $\odot$ and $\oslash$ denote element-wise multiplication and division; scales are broadcast to the weight shape. We write $x^{\odot 2}$ for element-wise squaring and treat matrices as vectors in norms and inner products. Conventional quantization-aware training (QAT) maintains full-precision master weights $W \in \mathbb{R}^{O\times I}$ and quantizes them as

$$Q = \mathcal{Q}_{\mathcal{C}}(W \oslash \alpha), \qquad \widehat{W} = \alpha \odot Q, \tag{1}$$

where $\mathcal{Q}_{\mathcal{C}}$ maps each normalized weight to $\mathcal{C}$. QAT accumulates continuous updates in W using the straight-through estimator (STE; Bengio et al., 2013) until a quantization boundary is crossed, whereas DUET updates Q directly.

Continuous optimizer update. For the current minibatch loss ℓ_t, let $G_t = \nabla_{\widehat{W}_t}\ell_t$. With scales fixed, the gradient with respect to Q is $g_t = \alpha_t \odot G_t$. Let m_t, v_t denote Adam's first and second moments (Kingma & Ba, 2014), with decay rates $\beta_1, \beta_2 \in [0, 1)$. Both moments start at zero, and hats denote their standard bias corrections. Before the reset described below, we compute

$$u_t = -\widehat{m}_t \oslash (\sqrt{\widehat{v}_t} + \epsilon), \qquad z_t = \eta_t u_t, \tag{2}$$

with $\epsilon > 0$ and learning rate $\eta_t > 0$. We use z_t as the reference update in effective-weight units.

2.2 Discrete Weight Transition Budget

We convert the continuous optimizer update into a transition budget specifying the expected number of weight changes. Each weight is allowed at most one adjacent-level transition per step.

Feasible transitions. Let $\Delta_{\mathcal{C}}$ be the spacing between adjacent code values (2 for binary and 1 for ternary weights). We define the candidate code change in the direction of u_t as

$$\delta_t = \Delta_{\mathcal{C}} \operatorname{sign}(u_t) \odot \mathbf{1}[Q_t \odot \operatorname{sign}(u_t) < 1]. \tag{3}$$

The candidate change is zero when $u_t = 0$ or when the transition would leave $\mathcal{C}$. The corresponding transition sizes in effective-weight units are $a_t = \alpha_t \Delta_{\mathcal{C}}$. Let I_t be a binary mask that selects only feasible transitions. The resulting changes in the discrete codes and effective weights are $d_t = \delta_t \odot I_t$ and $\Delta\widehat{W}_t = \alpha_t \odot d_t$, respectively. The total transition count is $K_t = \|I_t\|_1$, and its expectation $\bar{K}_t = \mathbb{E}[K_t]$ defines the budget. Each entry of $\pi_t = \mathbb{E}[I_t]$ gives the selection probability of the corresponding transition. We take all one-step expectations over transition selection, holding the current weights, scales, minibatch, and optimizer signal fixed.

Loss change under discrete transitions. With scales and other parameters fixed, define the minibatch loss $F_t(Q) = \ell_t(\alpha_t \odot Q)$. Assume its gradient is Lipschitz-continuous with constant L_t on each line segment from Q_t to a feasible $Q_t + d_t$. Taylor's theorem and $\|d_t\|_2^2 = \Delta_{\mathcal{C}}^2 K_t$ then give

$$\mathbb{E}[F_t(Q_t + d_t) - F_t(Q_t)] \le \langle g_t, \delta_t \odot \pi_t \rangle + \frac{L_t \Delta_{\mathcal{C}}^2}{2} \bar{K}_t. \tag{4}$$

At a fixed total budget, the curvature term in this bound is the same for every allocation.

From update magnitude to transition count. Each transition contributes one to $\|\Delta\widehat{W}_t \oslash a_t\|_1 = K_t$. We divide the magnitudes $|z_t|$ of the continuous reference update by the transition sizes a_t and retain only feasible coordinates. The resulting scores s_t sum to the transition budget:

$$s_t = \mathbf{1}\{\delta_t \neq 0\} \odot (|z_t| \oslash a_t), \qquad \bar{K}_t = \|s_t\|_1. \tag{5}$$

For fixed u_t and α_t, a smaller learning rate yields fewer expected transitions without changing their size.

Channel-wise budgets. We split the total budget by output channel to preserve each channel's share of the reference movement. For output channel o (matrix row), we set $\bar{K}_{t,o} = \sum_j s_{t,oj}$, where j indexes input channels. We empirically observed more stable training with this row-wise allocation than with global allocation. The channel's transition count is $K_{t,o} = \sum_j I_{t,oj}$, and its feasible coordinates form the set $J_{t,o} = \{j : \delta_{t,oj} \neq 0\}$. We assume $\bar{K}_{t,o} \leq |J_{t,o}|$, while individual scores may exceed one.

Stochastic transition count. We round each channel's budget down or up, choosing the upper value with probability $\rho_{t,o} = \bar{K}_{t,o} - \lfloor \bar{K}_{t,o} \rfloor$:

$$K_{t,o} = \lfloor \bar{K}_{t,o} \rfloor + \mathrm{Bernoulli}(\rho_{t,o}). \tag{6}$$

This rounding preserves $\mathbb{E}[K_{t,o}] = \bar{K}_{t,o}$. Its count variance $\rho_{t,o}(1-\rho_{t,o}) \leq 1/4$ is minimal among integer counts with this mean. Minimality follows from $(K-k)(K-k-1) \geq 0$ for integer K and $k = \lfloor \bar{K}_{t,o} \rfloor$, with equality for the two adjacent counts.

2.3 Weight Allocation Under Transition Budget

We allocate each channel's budget to its weights to minimize the expected squared error between the discrete and continuous updates, measured in quantization intervals.

Allocation under a fixed budget. For output channel o, the expected squared error expands as

$$\mathbb{E}\|(\Delta\widehat{W}_{t,o} - z_{t,o}) \oslash a_{t,o}\|_2^2 = \|z_{t,o} \oslash a_{t,o}\|_2^2 + \bar{K}_{t,o} - 2\sum\nolimits_{j \in J_{t,o}} s_{t,oj}\pi_{t,oj}. \tag{7}$$

With the budget $\bar{K}_{t,o}$ fixed, the first two terms are constant, so an optimal allocation assigns inclusion probability to the largest feasible scores first. Selecting the top $K_{t,o}$ feasible scores using the stochastic count in Equation (6) achieves this minimum; ties are broken in a fixed order. The expected update need not equal the reference at each coordinate. Figure 2(a) illustrates the allocation.

Applying transitions and resetting momentum. We select feasible transitions with TopK, apply them to the weights, and reset the corresponding first moments to zero:

$$I_{t,o} = \mathrm{TopK}(s_{t,o}, K_{t,o}), \qquad Q_{t+1} = Q_t + d_t, \qquad m_t \leftarrow (1 - I_t) \odot m_t. \tag{8}$$

Zero reset makes each changed weight rebuild momentum from gradients at its new value. Without reset, retained momentum can drive ternary weights through zero toward ± 1 (see Figure 4(b)). The second moment and global bias-correction counter are retained.

Expected loss reduction. Each transition moves opposite to the first moment, while its first-order effect on the loss depends on the current gradient. Let $e_t = g_t - \widehat{m}_t$ measure their difference before reset. To relate the scores to the first moment, let $h_t = \alpha_t \odot (\sqrt{\widehat{v}_t} + \epsilon)$ and $h_{\min,t} = \min_{o,j} h_{t,oj} > 0$. On the feasible coordinates $J_t = \{(o,j) : j \in J_{t,o}\}$, the relation $|\widehat{m}_{t,oj}| = \Delta_{\mathcal{C}} h_{t,oj} s_{t,oj}/\eta_t$ gives a predicted first-order gain of at least $(\Delta_{\mathcal{C}}^2 h_{\min,t}/\eta_t)\langle s_t, \pi_t \rangle$. In Equation (4), the contribution of e_t is at most $\Delta_{\mathcal{C}} \bar{K}_t \|e_{t,J_t}\|_\infty$, while the curvature term is $L_t \Delta_{\mathcal{C}}^2 \bar{K}_t/2$, where e_{t,J_t} denotes the entries of e_t on J_t. These two bounds are constant for fixed channel budgets, so TopK minimizes the resulting loss bound by maximizing $\langle s_t, \pi_t \rangle$. With $C_t = \max\{1, \|s_t\|_\infty\}$, the capacity assumption allows us to increase the comparison probabilities s_t/C_t on feasible coordinates to meet each channel's budget without exceeding one. This comparison gives $\langle s_t, \pi_t \rangle \geq \|s_t\|_2^2/C_t$, and hence

$$\mathbb{E}[F_t(Q_t + d_t) - F_t(Q_t)] \leq -\frac{\Delta_{\mathcal{C}}^2 h_{\min,t}}{\eta_t C_t}\|s_t\|_2^2 + \Delta_{\mathcal{C}}\|s_t\|_1 \left(\|e_{t,J_t}\|_\infty + \frac{L_t \Delta_{\mathcal{C}}}{2}\right). \tag{9}$$

The current minibatch loss decreases in expectation when the guaranteed gain, given by the magnitude of the negative term, exceeds the sum of the bounds on gradient error and curvature cost. A smaller learning rate reduces the expected transition count, while each selected weight still moves by one quantization interval. As long as $C_t = 1$, changing only η_t scales all terms in the bound by the same factor, leaving its sign unchanged.

2.4 Memory-Efficient Optimizer States

Optimizer-state compression. We store the first moment in block-32 MXINT8 with stochastic rounding, motivated by 8-bit Adam (Dettmers et al., 2022). We factorize the second moment into row and column statistics as in Adafactor (Shazeer & Stern, 2018). With an 8-bit shared scale per block and FP32 factors, these require approximately 8.25 and $32/O + 32/I$ bits per weight, respectively. We use the reconstructed moments to compute u_t. We evaluate the effect of state compression on training quality in Table 6.

Direct gradient accumulation. Following AdamA (Zhang et al., 2023), we accumulate gradient contributions directly into the optimizer states without storing a separate full weight-gradient matrix. For P data-parallel workers, let $g_t^{(p)}$ be the gradient from worker p. Define the average gradient $\bar{g}_t = P^{-1}\sum_p g_t^{(p)}$ and the average of the squared gradients $\overline{g_t^{\odot 2}} = P^{-1}\sum_p (g_t^{(p)})^{\odot 2}$. We use $g_t = \bar{g}_t$ and update the moments as

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\bar{g}_t, \qquad v_t \approx \beta_2 v_{t-1} + (1-\beta_2)\frac{1}{P}\overline{g_t^{\odot 2}}. \tag{10}$$

The first-moment aggregation is exact before state quantization. For the second moment, we approximate the squared average gradient $\bar{g}_t^{\odot 2}$ by omitting cross-worker products.

3 Experiments

We evaluate the effectiveness of DUET for full-parameter adaptation of low-bit language models without full-precision master weights. Specifically, we directly update the ternary weights of Qwen3 (Yang et al., 2025) checkpoints released by CAT-Q (Wang et al., 2026). We additionally fine-tune the QAT-trained binary Bonsai-27B (1-bit) and Ternary-Bonsai-27B (1.58-bit) checkpoints (ML, 2026) to assess whether DUET can adapt models trained with discrete weights.

3.1 Experimental Setup

Implementation details. We optimize the continuous parameters, including quantization scales, using AdamW (Loshchilov & Hutter, 2019) with BF16 computation, while directly updating the binary or ternary weights with DUET. All main experiments are conducted on 4× H200 GPUs. Our implementation stores first-moment states in 8-bit precision, uses factorized second-moment states, and accumulates gradients directly into the optimizer states. For adaptation, we train for one epoch with a sequence length of 256, a global batch size of 128, and a learning rate of 1×10^{-5} with 3% warm-up followed by linear decay. Complete training settings are provided in Appendix B.2.

Dataset and evaluation. We use Commonsense170K (Hu et al., 2023) for adaptation and evaluate on eight Commonsense reasoning benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Challenge and ARC-Easy (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). We report per-task and average accuracy, validation loss and perplexity, and peak GPU memory.

3.2 Scalable Full-Parameter Adaptation of Low-Bit Models

In this section, we evaluate DUET on Qwen3 models from 8B to 235B and released binary and ternary Bonsai-27B checkpoints. Table 1 reports Qwen3 adaptation results on Commonsense170K (Hu et al., 2023). DUET and Ternary + LoRA (Hu et al., 2022) are initialized from CAT-Q (Wang et al., 2026) ternary checkpoints, whereas FP + LoRA uses the corresponding full-precision checkpoints. Table 2 compares DUET with STE (LSQ; Esser et al. 2020) full fine-tuning using 8-bit Adam (Dettmers et al., 2022) on Qwen3-8B, reporting adaptation quality and peak GPU

Table 1: **Commonsense adaptation of Qwen3 dense and MoE models.** We compare DUET full-parameter fine-tuning with Ternary + LoRA across five Qwen3 models (8B–235B). DUET and Ternary + LoRA use CAT-Q ternary checkpoints (Wang et al., 2026), while FP + LoRA uses the corresponding full-precision checkpoints. We report average accuracy over eight commonsense tasks and validation perplexity. DUET consistently outperforms Ternary + LoRA on both metrics.

	Commonsense accuracy (%) ↑								Summary	
Method / Model	BoolQ	PIQA	SIQA	HelS	WinG	ARC-c	ARC-e	OBQA	**Avg.↑**	**PPL↓**
Qwen3-8B	*86.54*	*77.42*	*51.79*	*74.92*	*68.03*	*56.57*	*80.93*	*41.80*	*67.25*	–
FP + LoRA	*71.31*	*88.41*	*79.94*	*92.05*	*77.66*	*91.64*	*96.80*	*87.80*	*85.70*	*3.093*
Ternary + LoRA	65.08	79.43	75.90	78.97	68.27	75.00	88.68	77.80	76.14	3.300
DUET (full FT)	**66.97**	**82.05**	**76.71**	**84.65**	**76.24**	**75.26**	**87.63**	**83.40**	**79.11**	**3.018**
Qwen3-14B	*89.42*	*79.65*	*52.25*	*78.83*	*73.16*	*60.58*	*82.83*	*46.20*	*70.36*	–
FP + LoRA	*73.58*	*89.61*	*80.76*	*94.89*	*82.56*	*93.17*	*97.73*	*94.20*	*88.31*	*3.012*
Ternary + LoRA	67.61	85.31	77.18	90.05	77.43	83.45	92.59	85.60	82.40	3.156
DUET (full FT)	**68.56**	**86.24**	**78.30**	**91.54**	**82.00**	**81.57**	**91.04**	**86.80**	**83.26**	**2.830**
Qwen3-32B	*86.30*	*81.88*	*52.46*	*82.66*	*73.56*	*60.92*	*83.12*	*46.20*	*70.89*	–
FP + LoRA	*74.56*	*92.17*	*81.32*	*96.38*	*84.45*	*94.71*	*98.53*	*93.80*	*89.49*	*2.978*
Ternary + LoRA	69.79	88.68	80.71	94.05	82.87	89.16	95.75	89.80	86.35	3.135
DUET (full FT)	**71.68**	**89.72**	**80.09**	**95.03**	**85.56**	**88.05**	**95.58**	**93.00**	**87.34**	**2.702**
Qwen3-30B-A3B	*88.56*	*80.14*	*50.00*	*77.66*	*69.93*	*56.23*	*79.25*	*45.00*	*68.35*	–
FP + LoRA	*72.60*	*91.29*	*80.04*	*94.12*	*76.64*	*93.52*	*98.32*	*91.00*	*87.19*	*3.077*
Ternary + LoRA	64.01	82.43	70.93	71.11	63.61	77.47	90.57	76.40	74.57	3.380
DUET (full FT)	**68.90**	**85.36**	**77.69**	**90.25**	**81.53**	**81.48**	**91.71**	**86.80**	**82.97**	**2.881**
Qwen3-235B-A22B	*89.54*	*82.92*	*54.91*	*85.87*	*79.01*	*65.27*	*84.89*	*45.00*	*73.43*	–
FP + LoRA	OOM (4×H200)									
Ternary + LoRA	71.83	90.97	82.19	94.87	87.69	89.93	96.68	92.20	88.29	2.810
DUET (full FT)	**73.24**	**91.13**	**82.09**	**95.26**	**88.95**	**90.53**	**96.59**	**91.60**	**88.67**	**2.714**

Table 2: **Ternary Qwen3-8B fine-tuning and GPU memory.** We compare DUET with LoRA adaptation and STE-based full-parameter fine-tuning using 8-bit Adam (Dettmers et al., 2022). We report peak memory, eight-task average accuracy, validation perplexity, and loss.

Method	**Peak**	**Avg.↑**	**PPL↓**	**Loss↓**
Qwen3-8B(FP)	–	*67.25*	–	–
Ternary + LoRA	36.8	76.1	3.300	1.194
Ternary + STE	115.5	77.1	3.285	1.189
DUET	**44.4**	**79.1**	**3.018**	**1.1045**

Table 3: **Adaptation of QAT-trained low-bit checkpoints.** We evaluate DUET on Bonsai-27B (binary) and Ternary-Bonsai-27B (ternary), reporting eight-task average accuracy and validation perplexity and loss where available.

Method	**Avg.↑**	**PPL↓**	**Loss↓**
Bonsai-27B (1-bit, 1.125 bit/weight)			
No FT	63.54	–	–
DUET	**84.25**	**2.66**	**0.977**
Ternary-Bonsai-27B (1.58-bit)			
No FT	59.65	–	–
DUET	**87.28**	**2.60**	**0.955**

memory. Table 3 reports the performance of the released QAT-trained binary and ternary Bonsai-27B checkpoints before and after full-parameter fine-tuning (ML, 2026).

Scalability of full-parameter adaptation. As summarized in Table 1, DUET outperforms Ternary + LoRA across all five Qwen3 models in both average Commonsense accuracy and validation perplexity. Across the 8B–32B dense models, DUET improves accuracy by 2.97, 0.86, and 0.99 points, respectively. The largest gain occurs on Qwen3-30B-A3B (+8.40 points), where validation perplexity decreases from 3.380 to 2.881. FP + LoRA achieves higher accuracy on the four models it can train, although the gap narrows to 2.15 points at 32B. DUET yields lower validation perplexity in all four comparisons. At the largest scale, full-parameter fine-tuning with DUET on Qwen3-235B-A22B achieves 88.67% average accuracy and 2.714 validation perplexity, compared with 88.29% and 2.810 for Ternary + LoRA. These results demonstrate that DUET supports full-parameter adaptation of ternary models up to 235B without full-precision master weights.

Table 4: **Llama-3.2-1B ternary QAT over 10B tokens.** We train Llama-3.2-1B with DUET on English Ultra-FineWeb under the Tequila (Huang et al., 2026) setup and compare with ternary QAT baselines. We report zero-shot accuracy (%) on five benchmarks and their average. AbsMean (Ma et al., 2024), used in BitNet b1.58, serves as our STE baseline. Baseline details are in Appendix B.4.

Method	ARC-e	ARC-c	HelS	PIQA	WinG	**Avg.**
BF16	*65.4*	*31.3*	*47.7*	*74.2*	*60.3*	*55.8*
LSQ (Esser et al., 2020)	37.6	17.7	25.8	57.4	50.6	37.8
SEQ (Liu et al., 2025)	42.1	18.0	27.3	60.4	51.0	39.8
DLT (Chen et al., 2024)	42.4	17.4	25.6	56.3	51.3	38.6
TWN (Li et al., 2016)	40.7	22.0	28.4	60.1	49.2	40.1
AbsMean (Ma et al., 2024)	60.3	25.9	36.0	68.3	54.1	48.9
Tequila (Huang et al., 2026)	64.5	30.5	39.1	71.0	54.2	51.9
DUET	**62.5**	**29.6**	**37.9**	**70.7**	**55.3**	**51.2**

Table 5: **Practical GPU training capacity.** We report peak allocated memory (GiB) at batch size 1 and sequence length 256, including activations and temporary buffers.

GPU	**GiB**	**Model**	**Peak**$_{B=1}$
RTX 5060	8	Llama-3.2-**1B**	6.3
RTX 4090	24	Llama-3.2-**3B**	11.2
RTX 5090	32	Qwen3-**8B**	29.4
RTX PRO 6000	96	Qwen3-**32B**	69.0
H200	141	Llama-3.1-**70B**	130.2

Table 6: **Optimizer-state compression.** We compare validation loss and perplexity for STE (BitNet; Ma et al. 2024) and DUET with dense or compressed states. Δ Loss denotes the increase from compression.

Method	**Compression**	**Loss**↓	**PPL**↓	**Δ Loss**↓
STE	×	3.5479	34.74	+0.0316
STE	✓	3.5795	35.86	
DUET	×	**3.5137**	**33.57**	+0.0039
DUET	✓	**3.5176**	**33.70**	

Comparison with STE full fine-tuning. Table 2 shows that DUET improves adaptation quality while substantially reducing peak GPU memory relative to STE-based full fine-tuning on Qwen3-8B. DUET achieves 79.1% average accuracy, outperforming STE with 8-bit Adam (Dettmers et al., 2022) (77.1%) by 2.0 percentage points, while reducing peak memory from 115.5 to 44.4 GiB, a reduction of 71.1 GiB (61.6%). Validation loss decreases from 1.189 to 1.1045, and validation perplexity decreases from 3.285 to 3.018. Compared with Ternary + LoRA (76.1%), DUET improves average accuracy by 3.0 percentage points at the cost of 7.6 GiB additional peak memory. Because the reported peak measurements include activations and temporary buffers, these values reflect end-to-end training memory rather than optimizer-state memory alone. Thus, direct discrete full-parameter updates achieve higher adaptation accuracy than both baselines while requiring substantially less peak memory than STE-based full fine-tuning.

Adaptation of QAT-trained low-bit checkpoints. As shown in Table 3, full-parameter adaptation with DUET increases average accuracy from 63.54% to 84.25% for binary Bonsai-27B and from 59.65% to 87.28% for Ternary-Bonsai-27B, corresponding to gains of 20.71 and 27.63 percentage points, respectively. After fine-tuning, the models achieve validation perplexities of 2.66 and 2.60. For both checkpoints, DUET directly updates the released discrete weights while preserving their original binary or ternary formats, without full-precision master weights. These results show that DUET can further adapt low-bit models previously trained with QAT.

Overall, these results show that DUET supports full-parameter adaptation across model scales and low-bit formats without full-precision master weights. It consistently outperforms Ternary + LoRA on the evaluated CAT-Q checkpoints, further adapts QAT-trained binary and ternary models, and substantially reduces peak training memory compared with STE-based full fine-tuning.

3.3 Ablation Study and Analysis

We further examine the stability and scalability of DUET over longer training horizons and larger model scales. We evaluate 10B-token ternary QAT on Llama-3.2-1B (Table 4) and from-scratch

Table 7: **Scaling ternary pretraining from scratch.** We train 30M–200M models for $100N$ tokens each (3B–20B tokens). STE (BitNet; Ma et al. 2024) denotes AbsMean ternarization with full-precision master weights, while DUET directly updates ternary weights with uncompressed optimizer states. DUET achieves lower final validation loss and perplexity across all model sizes.

Method	30M		50M		100M		200M	
	Val. loss	PPL	Val. loss	PPL	Val. loss	PPL	Val. loss	PPL
STE	3.5479	34.74	3.4022	30.03	3.2038	24.63	3.0401	20.90
DUET	**3.5137**	**33.57**	**3.3489**	**28.47**	**3.1768**	**23.97**	**3.0257**	**20.61**

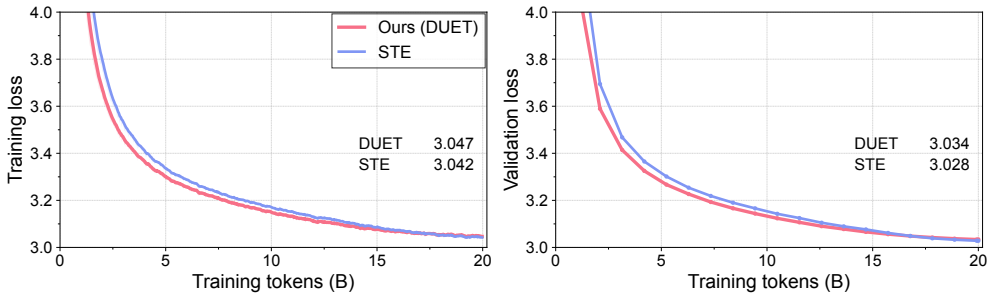


Figure 3: **Ternary Gemma-1B pretraining over 20B tokens.** Starting from random initialization, we compare DUET with STE (BitNet AbsMean; Ma et al. 2024). DUET directly updates ternary weights without full-precision master weights, maintains lower loss for most of training, and converges comparably to STE at 20B tokens.

pretraining from 30M to 1B parameters (Table 7; Figure 3). We also assess practical GPU capacity (Table 5), optimizer compression (Table 6), and individual design components (Table 8).

QAT fine-tuning over 10B tokens. We conduct 10B-token ternary QAT fine-tuning on the English portion of Ultra-FineWeb (Wang et al., 2025), starting from the same pretrained Llama-3.2-1B checkpoint used by Tequila (Huang et al., 2026). Training details are in Appendix B.2. As shown in Table 4, DUET achieves 51.2% average zero-shot accuracy, 2.3 points above STE-based AbsMean (Ma et al., 2024) (48.9%) and 0.7 points below Tequila (51.9%). While AbsMean updates full-precision master weights through STE, DUET directly updates ternary weights and learns their quantization scales using LSQ (Esser et al., 2020). DUET exceeds the LSQ baseline (37.8%) by 13.4 points. Its gain over AbsMean holds across all five benchmarks, reaching 3.7 points on ARC-Challenge; it also exceeds Tequila by 1.1 points on WinoGrande. These results show that direct ternary updates remain competitive after 10B tokens without full-precision master weights.

Practical GPU training capacity. We investigate the practical training capacity of DUET across consumer and data center GPUs at batch size 1 and sequence length 256. Table 5 shows that DUET scales from a 1B model on RTX 5060 to a 70B model on a single H200, with intermediate 3B, 8B, and 32B models fitting on RTX 4090, RTX 5090, and RTX PRO 6000, respectively. Measured peak memory ranges from 6.3 to 130.2 GiB, including activations and temporary buffers. For the same model sizes, Figure 1 shows that STE-based QAT requires 3.3–8.6× more peak memory than DUET and exceeds the corresponding GPU capacity in all five cases. These results show that DUET enables full-parameter low-bit training on memory-limited GPUs.

Effect of optimizer-state compression. We examine the effect of optimizer-state compression on DUET and STE (Table 6). For DUET, compression increases validation loss by only +0.0039, compared with +0.0316 for STE, an 8.1× smaller degradation. Perplexity shows the same trend (+0.13 vs. +1.12). In DUET, the aggregate update magnitude determines the expected number of flips per row, while relative magnitudes rank feasible weights for selection. State compression may perturb individual values, but often preserves both the transition budget and the highest-ranked weights. This robustness helps explain the small degradation, with compressed DUET still achieving lower validation loss and perplexity than both STE variants.

Table 8: **Components of ternary training on a 30M-parameter model.** We vary first-moment precision (FP32, MXINT8, or MXINT4), factorization of the second moment v, TopK transition selection, and momentum resetting after a weight change. Check marks indicate enabled components. We report final validation loss and perplexity (PPL), where lower is better.

Components				Results	
Moment	**Factored v**	**TopK**	**Resetting**	**Loss↓**	**PPL↓**
FP32	×	✓	✓	**3.5137**	**33.57**
MXINT8	×	✓	✓	3.5366	34.35
MXINT4	×	✓	✓	4.0893	59.70
MXINT8	✓	✓	✓	3.5176	33.70
MXINT8	✓	×	✓	3.7148	41.05
MXINT8	✓	✓	×	3.6191	37.31

Scaling ternary models to 200M parameters. We evaluate the scalability of DUET across ternary model sizes and pretraining budgets. Following the scaling protocol of ECO (Nikdan et al., 2026), we train models at four sizes (30M–200M parameters) from scratch for $100N$ tokens, where N is the parameter count. Table 7 shows lower validation loss for DUET at every size (by 0.0144–0.0533), along with lower perplexity. At 200M parameters after 20B tokens, DUET achieves lower validation loss (3.0257 vs. 3.0401) and perplexity (20.61 vs. 20.90) than STE. Figure 2(b) shows that final validation loss falls as model size grows, with DUET consistently below STE at every scale. These results show that DUET maintains its advantage over STE as model size and training budget increase, scaling to 200M parameters and 20B tokens without full-precision master weights.

Training ternary Gemma-1B from scratch. We investigate whether DUET can train a ternary 1B model from scratch for 20B tokens without full-precision master weights. We pretrain a ternary Gemma-1B (Team et al., 2025) with DUET under the settings described in Appendix B. As shown in Figure 3, DUET reduces both training and validation loss faster early in training and remains below STE for most of the run. At 20B tokens, DUET achieves a training loss of 3.047 (STE: 3.042; +0.005) and a validation loss of 3.034 (STE: 3.028; +0.006). Despite these small final gaps, direct discrete updates closely match STE convergence. These results demonstrate that direct ternary updates can support 20B-token pretraining at the 1B-parameter scale without full-precision master weights.

Component analysis. We analyze first-moment precision, second-moment factorization, TopK selection, and Resetting on a 30M-parameter model (Table 8). With dense second moments, MXINT4 increases validation loss relative to MXINT8 (4.0893 vs. 3.5366), whereas factoring the MXINT8 second moment lowers it to 3.5176. The MXINT4 drop is consistent with its coarser quantization grid blurring differences between first-moment magnitudes, which can perturb transition budgets and the TopK ranking of candidate weights. Starting from the factored MXINT8 configuration with both TopK and Resetting enabled, replacing TopK with Bernoulli sampling increases loss to 3.7148 (+0.1972), while disabling Resetting increases it to 3.6191 (+0.1015). These ablations suggest that moment precision, TopK, and Resetting matter for optimization with compressed states.

4 Conclusion

In this paper, we propose DUET, a scalable discrete weight update framework that directly trains extreme low-bit weights without master weights. DUET controls the expected number of weight changes and allocates them to minimize the expected squared error from the continuous optimizer update, measured in quantization intervals. Across fine-tuning and pretraining, DUET matches or outperforms STE-based training while substantially reducing training memory. It achieves greater training memory reductions as model size increases, reaching $12.0\times$ at 235B parameters, and scales full-parameter training on a single H200 GPU from 8B with STE to 70B with DUET. These results show that the memory efficiency of extreme low-bit models can extend from inference to training. This could allow deployed models to be updated even on edge devices, such as robots adapting to new environments. More broadly, we hope our work encourages researchers to explore larger models with fewer bits, even with limited GPU resources.

AI USE STATEMENT

We used generative AI tools to assist with figure and table editing, and routine coding. AI-assisted code was reviewed and tested by the authors, and all reported results were independently produced and verified by the authors. All AI-assisted text and figures were reviewed against the underlying experiments and data. The authors take full responsibility for the content of this paper.

REFERENCES

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

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Tianqi Chen, Zhe Li, Weixiang Xu, Zeyu Zhu, Dong Li, Lu Tian, Emad Barsoum, Peisong Wang, and Jian Cheng. Ternaryllm: Ternarized large language model. *arXiv preprint arXiv:2406.07177*, 2024.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

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, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=shpkpVXzo3h.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. *Advances in neural information processing systems*, 32, 2019.

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*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore, 2023. Association for Computational Linguistics.

Hong Huang, Decheng Wu, Rui Cen, Guanghua Yu, Zonghang Li, Kai Liu, Jianchen Zhu, Peng Chen, Xue Liu, and Dapeng Wu. Tequila: Trapping-free ternary quantization for large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9CZzD5LWdy.

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 *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 2704–2713. IEEE, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016. URL https://arxiv.org/abs/1605.04711.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, Lin Xiao, Yuandong Tian, Bilge Soran, Raghuraman Krishnamoorthi, Tijmen Blankevoort, and Vikas Chandra. Paretoq: Improving scaling laws in extremely low-bit LLM quantization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=PMSNd8xTHp.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Prism ML. Bonsai 27b: Full 27b-class reasoning in binary and ternary transformer weights — on laptops and phones. Technical report, July 2026. URL https://prismml.com.

Van Minh Nguyen, Cristian Ocampo, Aymen Askri, Louis Leconte, and Ba-Hien Tran. B$\oplus$LD: Boolean logic deep learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DO9wPZOPjk.

Mahdi Nikdan, Amir Zandieh, Dan Alistarh, and Vahab Mirrokni. ECO: Quantized training without full-precision master weights. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=dfVm5yZZ6G.

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. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pp. 4596–4604. PMLR, 2018.

Cuauhtemoc Daniel Suarez-Ramirez, Miguel Gonzalez-Mendoza, Leonardo Chang, Gilberto Ochoa-Ruiz, and Mario Alberto Duran-Vega. A bop and beyond: a second order optimizer for binarized neural networks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1273–1281. IEEE, 2021.

Falcon-LLM Team. Falcon-e, a series of powerful, universal and fine-tunable 1.58bit language models, May 2025. URL https://falcon-lm.github.io/blog/falcon-edge.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Shigeng Wang, Chao Li, Yangyuxuan Kang, Jiawei Fan, and Anbang Yao. CAT-q: Cost-efficient and accurate ternary quantization for LLMs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=9uZJLXt7fq.

Yudong Wang, Zixuan Fu, Jie Cai, Peijun Tang, Hongya Lyu, Yewei Fang, Zhi Zheng, Jie Zhou, Guoyang Zeng, Chaojun Xiao, et al. Ultra-fineweb: Efficient data filtering and verification for high-quality llm training data. *arXiv preprint arXiv:2505.05427*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Yijia Zhang, Yibo Han, Shijie Cao, Guohao Dai, Youshan Miao, Ting Cao, Fan Yang, and Ningyi Xu. Adam accumulation to reduce memory footprints of both activations and gradients for large-scale dnn training. In *ECAI 2023: 26th European Conference on Artificial Intelligence, September 30–October 4, 2023, Kraków, Poland–Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, pp. 3058–3065, 2023.

Kaiyan Zhao, Tsuguchika Tabaru, Kenichi Kobayashi, Takumi Honda, Masafumi Yamazaki, and Yoshimasa Tsuruoka. Direct quantized training of language models with stochastic rounding. In Hung-yi Lee and Tongliang Liu (eds.), *Proceedings of the 17th Asian Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 1150–1165. PMLR, 09–12 Dec 2025.

A Related Work

Extreme Low-Bit Large Language Models. Extreme low-bit LLMs have emerged as a promising direction for reducing the memory and computational costs of large language models. Representative low-bit pretrained LLMs (Ma et al., 2024; Team, 2025; ML, 2026) have demonstrated the scalability of binary and ternary pretraining. Another line of work starts from pretrained full-precision LLMs (Yang et al., 2025; Grattafiori et al., 2024) and applies additional QAT to obtain ternary models (Chen et al., 2024; Liu et al., 2025; Huang et al., 2026). Prior work improves ternary QAT through learnable quantization and knowledge distillation (Chen et al., 2024), refined quantization and training schemes (Liu et al., 2025), and improved optimization of ternary weights (Huang et al., 2026). However, these approaches still require substantial training compute and memory because QAT typically maintains full-precision weights and optimizer states during optimization. In contrast, DUET reduces the training-memory cost by updating discrete weights without full-precision master weights and compressing its optimizer states.

Direct Optimization of Discrete Weights. A line of work explores optimizing quantized weights directly without full-precision latent weights. Early studies considered threshold-based flipping (Helwegen et al., 2019; Suarez-Ramirez et al., 2021) and Boolean optimization (Nguyen et al., 2024). Bop and Bop2ndOrder use first- and second-order gradient statistics for binary weight flipping, while BOLD directly optimizes Boolean weights. For language models, direct quantized training uses stochastic rounding to preserve small updates (Zhao et al., 2025), while ECO (Nikdan et al., 2026) feeds quantization error into optimizer momentum for FP8 and INT4 training. At binary and ternary precision, however, individual transitions can be much larger than continuous optimizer steps. In contrast, DUET converts the continuous update signal into a per-output-channel transition budget and assigns transitions to feasible weights with the largest update magnitudes, controlling both how many weights change and which weights change without full-precision master weights.

B Implementation & Dataset Details

We provide model initialization, training configurations, datasets, evaluation protocols, and baselines for the experiments in Section 3.

B.1 Fine-Tuning Checkpoints

For adaptation after PTQ, we use CAT-Q (Wang et al., 2026) ternary Qwen3 (Yang et al., 2025) dense (8B, 14B, 32B) and MoE (30B-A3B, 235B-A22B) checkpoints. We use Bonsai-27B and Ternary-Bonsai-27B (ML, 2026). For ternary QAT fine-tuning, we initialize from pretrained Llama-3.2-1B (Grattafiori et al., 2024) following the Tequila (Huang et al., 2026) training setup.

B.2 Training Settings

For DUET, we set the first- and second-moment coefficients for discrete weights to $(\beta_1, \beta_2) = (0.999, 0.999)$ and clip the gradient norm at 1.0. For pretraining and QAT, continuous parameters and discrete weights share a learning rate η, while scales use $\eta/10$ with weight decay 0.1.

Fine-tuning settings. For Commonsense170K adaptation, each run uses one epoch, a sequence length of 256, and a global batch size of 128. For DUET, we use a learning rate of 1×10^{-5} with a 3% warm-up followed by linear decay. For FP + LoRA (Hu et al., 2022) and Ternary + LoRA, we set both the rank and scaling factor to 128, disable dropout, and use a learning rate of 2×10^{-5} with cosine decay.

Ternary QAT fine-tuning. We train on the English portion of Ultra-FineWeb (Wang et al., 2025) for 10B tokens. Following Tequila (Huang et al., 2026), we set the learning rate to $\eta = 1 \times 10^{-3}$.

From-scratch optimizer settings. For the following from-scratch experiments, DUET uses uncompressed optimizer states to study the convergence of its discrete update rule without the effects of state compression. Following ECO (Nikdan et al., 2026), continuous parameters use AdamW (Loshchilov & Hutter, 2019) with $(\beta_1, \beta_2) = (0.9, 0.98)$, $\epsilon = 10^{-9}$, and weight decay 0.1.

Scaling ternary training from scratch. We follow ECO's C4 pretraining setup (Nikdan et al., 2026), training 30M, 50M, 100M, and 200M-parameter models for $100N$ tokens each (Table 7). We use its T5 tokenizer (Raffel et al., 2020), batch size and sequence length of 512, and AdamW (Loshchilov & Hutter, 2019) warm-up and cosine schedule.

Gemma-1B training from scratch. Following ECO's Gemma training setup (Nikdan et al., 2026), we train the Gemma-1B (Team et al., 2025) architecture from random initialization on C4 for 20B tokens. We use the Gemma tokenizer, a global batch size of 256, and a sequence length of 512. We linearly warm η from 10^{-6} to 10^{-4} over the first 10% of training and then decay it to 10^{-5} with a cosine schedule (Figure 3).

B.3 Datasets and Evaluation

Commonsense adaptation. For adaptation after PTQ and adaptation of QAT-trained low-bit checkpoints (Tables 1, 2, and 3), we use Commonsense170K from LLM-Adapters (Hu et al., 2023) and reserve 120 examples for validation. We evaluate on eight commonsense benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQA (SIQA) (Sap et al., 2019), HellaSwag (HelS) (Zellers et al., 2019), WinoGrande (WinG) (Sakaguchi et al., 2021), ARC-Challenge (ARC-c) and ARC-Easy (ARC-e) (Clark et al., 2018), and OpenBookQA (OBQA) (Mihaylov et al., 2018). We compute accuracy from generated responses using task-specific answer extraction and report per-task accuracy, their unweighted average, validation loss, and perplexity (PPL).

Ternary QAT fine-tuning. We train on the English portion of Ultra-FineWeb (Wang et al., 2025) and evaluate zero-shot performance on ARC-e and ARC-c (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021). We report accuracy on each benchmark and their unweighted average (Table 4).

Training from scratch. For the 30M–200M scaling experiments, we train on C4 (Raffel et al., 2020) and report final validation loss and perplexity after $100N$ training tokens, where N denotes the number of model parameters (Table 7). For Gemma-1B (Team et al., 2025), we likewise train on C4 for 20B tokens and report training and validation loss throughout pretraining (Figure 3).

Memory measurements. We report peak allocated GPU memory in GiB. The measurements in Table 2 and the device-capacity experiments in Table 5 include activations and temporary buffers, reflecting the memory footprint of the full training workload.

B.4 Baselines

Adaptation after PTQ. We compare DUET with Ternary + LoRA (Hu et al., 2022) using the same CAT-Q (Wang et al., 2026) ternary Qwen3 (Yang et al., 2025) checkpoints. For Qwen3-8B, we additionally compare against STE-based full-parameter fine-tuning with LSQ (Esser et al., 2020) and 8-bit Adam (Dettmers et al., 2022). The original full-precision Qwen3 models and their FP + LoRA adaptations serve as full-precision references.

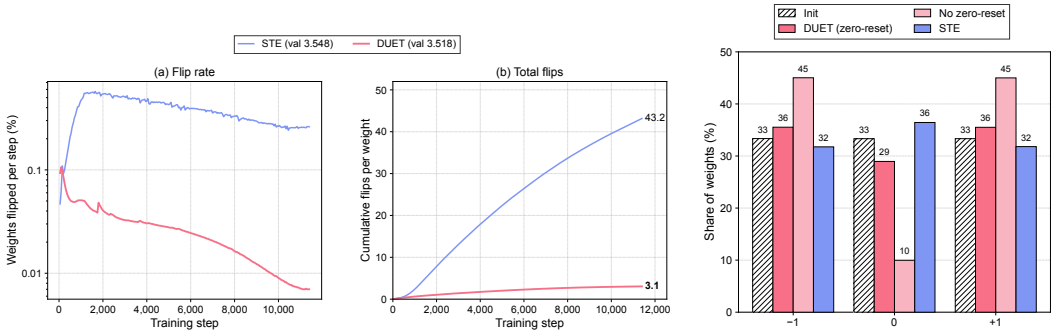
Adaptation of QAT-trained checkpoints. For the 1-bit Bonsai-27B and 1.58-bit Ternary-Bonsai-27B checkpoints (ML, 2026), we compare full-parameter adaptation with DUET against the corresponding released checkpoints without fine-tuning (No FT).

Ternary QAT baselines. We compare DUET with statistics-based ternarization methods TWN (Li et al., 2016) and AbsMean (Ma et al., 2024), learnable quantizers LSQ (Esser et al., 2020), DLT (Chen et al., 2024), and SEQ (Liu et al., 2025), and the ternary-weight optimization method Tequila (Huang et al., 2026). AbsMean, as used in BitNet b1.58, serves as our STE baseline, while BF16 provides the full-precision reference. We use the baseline results reported by Tequila and report their unweighted average over the five evaluation benchmarks.

Pretraining from scratch. Our STE baseline uses BitNet b1.58 AbsMean ternarization (Ma et al., 2024) with full-precision master weights for the 30M–200M scaling experiments and Gemma-1B. In contrast, DUET directly optimizes ternary weights without full-precision master weights.

B.5 Discrete Update Dynamics

We further analyze the discrete weight-update dynamics during training (Figure 4). As shown in Figure 4(a), DUET performs substantially fewer weight transitions than STE, accumulating only 3.1 flips per weight compared with 43.2 for STE, while achieving a lower validation loss (3.518 vs. 3.548). Figure 4(b) further examines the effect of resetting the first-moment state after each transition. With resetting, 29% of weights remain at zero after training, compared with 10% when resetting is disabled. This result suggests that moment resetting suppresses repeated transitions caused by accumulated momentum and stabilizes optimization in the discrete weight space.



(a) Flip rate and total flips **(b) Weight distribution**

Figure 4: **Ternary weight-update dynamics on a 30M-parameter model.** (a) The share of weights changed per step and cumulative flips per weight during training; DUET makes 3.1 flips per weight versus 43.2 for STE while reaching lower validation loss (3.518 vs. 3.548). (b) Final shares of weights in the -1, 0, and $+1$ states compared with initialization and STE. Zero resetting retains 29% of DUET weights at zero, versus 10% when resetting is disabled.