Training Strategies for Binary/Ternary Neural Networks

Robin Kiemes[1]

Supervisor(s): Qing Wang[1]**, Braden Refalo**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Robin Kiemes
Final project course: CSE3000 Research Project
Thesis committee: Qing Wang, Braden Refalo, Julia Olkhovskaia

Abstract

Binary and ternary neural networks offer substantial reductions in memory and computational cost, making them attractive for deployment on resource-constrained devices. Training these networks remains challenging because quantization functions are non-differentiable, requiring gradient approximations such as the Straight-Through Estimator (STE).

This work presents a systematic ablation study of the effects of different training configurations on ResNet-20 on CIFAR-10. We evaluated eleven STE variants and independently examined the effects of weight clipping and batch normalization. All ternary variants perform within 0.73 percentage points of the 91.61% full-precision baseline, with the *polynomial_ste* achieving the best result of 91.23%. For binary, all variants reach 1.66 percentage points below the baseline, with *tanh_ste* being the highest performer (90.35%). We find that the choice of STE has only a minor impact on final accuracy; however, STEs differ in training stability, with smoother estimators providing more consistent convergence.

Batch normalization had the greatest effect on performance; removing it reduced accuracy by up to 8.66 percentage points. Weight clipping yielded a smaller but consistent benefit, with an optimal clipping factor of $f = 4.0$, improving accuracy by 0.26 and 0.5 percentage points, respectively. Combining these findings, we identified effective training configurations for both ternary and binary networks: the optimal ternary setup (Using Trained Ternary Quantization) achieved 91.52% accuracy on ResNet-20/CIFAR-10, while the optimal binary configuration (Using XNOR-Net quantization) reached 90.78% accuracy, an improvement over prior baselines in both cases.

1 Introduction

The introduction of deep neural networks (DNNs) has significantly advanced the state of the art in numerous applications, including computer vision, natural language processing, and reinforcement learning [1; 2]. Since then, there has been strong motivation to make DNNs more efficient, as they typically require gigabytes of memory to store millions to billions of floating-point parameters and perform billions of floating-point operations (FLOPs).

Binary and ternary neural networks (BNNs/TNNs) provide a promising approach to reducing the computational and memory demands of DNNs through extreme quantization of weights, activations, or both, such as by reducing full-precision parameters to binary (1-bit) or ternary ($\log_2 3 \approx 1.58$ bits, corresponding to three discrete values) representations. The significant efficiency gains, including reduced storage requirements and the replacement of floating-point operations with fast bitwise computations, make quantized networks highly suitable for deployment on edge devices and other resource-constrained environments. Despite efficiency gains, training these models remains challenging due to the non-differentiable nature of their parameters [3].

Various strategies have been proposed to address the training of quantized networks. The use of Straight-Through Estimators (STEs) enables gradient-based backpropagation by approximating the non-differentiable gradients of quantized weights [4]. However, different STE formulations can significantly influence optimization behavior, affecting convergence, stability, and bias. Recent studies show that STE choice is a tradeoff between estimating error and gradient stability [5]. Furthermore, weight clipping is used to constrain parameter magnitudes and reduce divergence. Batch normalization is implemented to stabilize quantized activation distributions.

1.1 Related Works

There has been extensive research on improving the accuracy of binarized/ternarized networks. Papers such as XNOR-Net [6], DoReFa [7], and Trained Ternary Quantization [8] look to reduce quantization error and increase accuracy by optimizing the quantization scheme. ReActNet [9] has shown increased accuracy by optimizing activations and architectural structure. Numerous works have sought to increase stability and accuracy by proposing optimized STE surrogates [5]; however, they warn of obstacles such as gradient instability due to estimation errors and gradient vanishing. Commonly used straight-through estimators (STEs), such as the Identity STE, have theoretical convergence guarantees in certain theoretical conditions. However, different variants can lead to different optimization behavior, motivating further study of their effects [10].

Other studies have foregone STEs and opted for estimating gradients through Hypernetworks, where an additional network is used for estimating the non-differentiable networks, trained alongside the original network, in order to better estimate the gradient while maintaining training stability [11; 12; 13].

1.2 Research Gap and Motivation

The training dynamics of binary neural networks are highly sensitive to the choice of training strategy. Prior research has primarily concentrated on developing novel quantization methods and improving surrogate gradient formulations. However, comparatively fewer studies have systematically investigated the individual contributions and interactions of key training mechanisms, such as the straight-through estimator (STE) variant, weight clipping schemes, and batch normalization, under controlled experimental settings.

This motivates a systematic analysis of how key training components, specifically straight-through estimator (STE) variants, weight clipping strategies, and batch normalization, individually and jointly influence optimization in weight-quantized neural networks. In particular, there is limited controlled evidence on how these mechanisms interact to affect convergence stability and final task performance in binary and ternary settings.

This leads to the following research question: How do different straight-through estimator (STE) variants, weight clipping, and batch normalization affect the training stability, convergence behavior, and final accuracy of weight-quantized neural networks?

2 Background

Quantized neural networks reduce the precision of weights and activations to discrete sets, enabling significant improvements in memory efficiency and computational throughput. However, this discretization introduces optimization challenges due to non-differentiability and information loss during training. This section reviews the core concepts underlying binary and ternary quantization, the challenges of training such models, and the key techniques for stabilizing optimization under low-precision constraints.

2.1 Binary and Ternary Quantization

A **quantization function** maps continuous floating-point parameters to a discrete set of values. Let $w \in \mathbb{R}$ denote a full-precision weight and $\hat{w}$ its quantized version. In binary neural networks (BNNs), weights and/or activations are constrained to $\{-1, +1\}$, for example via the sign function:

$$\hat{w} = \text{sign}(w) = \begin{cases} +1 & \text{if } w \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Ternary neural networks (TNNs) extend this idea by quantizing weights to $\{-1, 0, +1\}$. A common formulation introduces a threshold $\Delta > 0$, such that:

$$\hat{w} = \begin{cases} +1 & \text{if } w > \Delta \\ 0 & \text{if } |w| \leq \Delta \\ -1 & \text{if } w < -\Delta \end{cases}$$

where Δ controls the sparsity level of the resulting network [8]. Binary and ternary weights transform full-precision floating-point multiplications into efficient bitwise operations (e.g., XNOR-popcount in the binary case), yielding substantial reductions in memory footprint and computational cost [6].

2.2 Training Quantized Networks

The quantization function is non-differentiable and has zero gradient almost everywhere. To enable learning, a loss function $\mathcal{L}$ is defined over the network outputs and serves as the scalar objective for gradient-based optimization. Let $\hat{w} = Q(w)$ denote the quantized weight. The backpropagation chain rule yields:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial \hat{w}} \cdot \frac{\partial \hat{w}}{\partial w} = \frac{\partial \mathcal{L}}{\partial \hat{w}} \cdot \underbrace{\frac{d}{dw} Q(w)}_{=0 \text{ a.e.}}$$

To enable optimization, quantized network training introduces a surrogate derivative:

$$\frac{\partial \hat{w}}{\partial w} \approx h(w)$$

This yields the Straight-Through Estimator (STE) formulation:

$$\frac{\partial \mathcal{L}}{\partial w} \approx \frac{\partial \mathcal{L}}{\partial \hat{w}} \cdot h(w)$$

In practice, a full-precision copy of the weights, referred to as the **latent weights**, is maintained during training, while the quantized weights are used only in the forward pass [4].

2.3 Straight-Through Estimators

The Straight-Through Estimator [14], replaces the zero gradient of the quantization function with a surrogate $h(w)$. Several variants exist, each affecting gradient descent and stability [5].

Identity STE. The simplest variant passes the gradient unmodified:

$$h(w) = 1$$
$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial \hat{w}}$$

This simple estimator ensures an uninterrupted gradient flow through the non-differentiable quantization operation, avoiding the zero-gradient issue induced by quantization functions such as $\text{sign}(x)$. However, because the backward pass ignores the quantizer's structure, it introduces an approximation error into the surrogate. In particular, the identity surrogate does not approximate the geometry of $\text{sign}(x)$, leading to biased gradient estimates that may encourage updates inconsistent with the true quantized objective. This can manifest as instability, especially when parameters drift far from quantization boundaries. Nevertheless, [10] shows that the identity STE can still converge under suitable conditions, despite its lack of structural fidelity.

More generally, this approximation error motivates the design of alternative surrogate functions whose shapes more closely match the quantizer's behavior. These include clipped, piecewise-linear, and smooth approximations, which trade off gradient flow against fidelity to the underlying discrete mapping. Common STE shapes are summarized in Appendix 6.

2.4 Weight Clipping and Batch Normalization

Two additional training components interact with the STE to shape optimization dynamics.

Weight clipping projects full-precision weights back into a bounded interval $[-c, c]$ after each parameter update [3]. Formally, given an updated weight w, clipping is applied as

$$w \leftarrow \text{clip}(w, -c, c) = \min(\max(w, -c), c).$$

By keeping weights near the quantization decision boundary, clipping ensures that the straight-through estimator (STE) gradient remains active and reduces the risk of parameter drift during training.

Batch normalization (BN) standardizes activations prior to quantization, stabilizing the input distribution to the quantization function. For a mini-batch $\mathcal{B} = \{x_1, \ldots, x_m\}$, BN is defined as

$$\mu_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^{m} x_i, \quad \sigma^2_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2,$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \quad y_i = \gamma \hat{x}_i + \beta,$$

where γ and β are learnable scale and shift parameters, and ϵ is a small constant for numerical stability.

3 Experimental Setup

We designed a modular ablation framework that isolates the effect of each training strategy variable. The STE variant, weight clipping, and batch normalization were independently configurable and evaluated within a unified experimental setup.

3.1 Dataset

We compared all selected configurations of training under the same dataset: **CIFAR-10** [15], a standard benchmark of 60,000 color images across 10 classes (50,000 training, 10,000 test). Each image is 32x32 pixels with RGB color channels. CIFAR-10 is widely used for controlled ablation studies due to its moderate scale and computational efficiency. In particular, given the extremely large number of training configurations considered, restricting experiments to a single dataset ensures a consistent and computationally feasible evaluation setting.

3.2 Quantization Schemes

Quantization was applied to *weights only*. Prior work has shown that aggressively quantized weights can achieve substantial reductions in model size and memory bandwidth while preserving accuracy, whereas activations are often more challenging to quantize due to their input-dependent and highly dynamic distributions, and are therefore typically kept at higher precision [16]. As a result, many low-bit inference methods prioritize weight quantization to achieve favorable accuracy–efficiency trade-offs. Following this design choice, we quantized only network weights.

We considered two quantization schemes: **TTQ** [8] for ternary weight quantization and **XNOR** [6] for binary weight quantization. For TTQ, each weight was mapped to a ternary value using learned per-layer positive and negative scale factors $W_p, W_n > 0$ and a fixed threshold $\Delta = 0.7\alpha$, where $\alpha = \|W\|_1/n$:

$$Q(w_i) = W_p \cdot \mathbf{1}_{w_i > \Delta} - W_n \cdot \mathbf{1}_{w_i < -\Delta}. \tag{1}$$

Weights satisfying $|w_i| \leq \Delta$ were mapped to zero. An extra FP32 configuration (full-precision 32 bits weights without quantization) served as the accuracy ceiling baseline.

3.3 Network Architectures

We used **ResNet-20** [17] (approximately 270K parameters) following the standard CIFAR-10 variant across all experiments (Architecture details in Appendix B). ResNet-20 is a suitable architecture for this setting, as it has been widely used in prior work on TTQ and XNOR-Net quantization schemes, providing a reproducible baseline. Additionally, its moderate depth and compact size make it sensitive to changes in training configuration, thereby amplifying observable effects. Full layer-by-layer details are provided in Table 6 in the Appendix.

3.4 STE Strategies

Eleven representative STE variants were implemented, as listed in Table 1. All methods followed the formulation $\hat{g}w_i = g\hat{w}i \cdot h(w_i)$, where $g\hat{w}_i = \frac{\partial \mathcal{L}}{\partial \hat{w}_i}$ denotes the gradient of the loss with respect to the quantized weight, and $h(w_i)$ denotes the surrogate gradient associated with each STE variant.

3.5 Training Protocol

All runs used **Stochastic Gradient Descent (SGD)** with momentum $\mu = 0.9$ and ℓ_2 weight decay $\lambda = 10^{-4}$. We opted for SGD over adaptive optimizers such as Adam because adaptive per-parameter learning rates may dampen instabilities introduced by different STE formulations, an effect we wish to isolate. The learning rate followed cosine annealing [18]:

$$\eta_t = \tfrac{1}{2}\eta_0\big(1 + \cos\big(\tfrac{\pi t}{T}\big)\big), \tag{2}$$

with initial learning rate $\eta_0 = 10^{-1}$, T equal to the total number of training epochs, and $\eta_{\min} = 0$. The main experiments were trained for **160 epochs**, following standard practice for ResNet-20 on CIFAR-10, which was generally sufficient for convergence under typical optimization and learning-rate schedules.

Where weight clipping was applied (Experiment 2), full-precision weights were clamped immediately after each optimizer step.

3.6 Experiment Design

Four controlled ablation experiments were conducted, each isolating one factor. (Table 2). Both binary and ternary networks were evaluated in parallel, employing XNOR quantization for binary networks and TTQ for ternary networks.

1. **STE ablation**

   Each of the eleven STE variants was trained from random initialization with XNOR/TTQ weight-only quantization on ResNet-20 for 160 epochs; each run was repeated with five fixed seeds (0, 1, 2, 7, and 42). Results were reported as mean $\pm$ std across seeds. A full-precision (FP32) baseline was included as the accuracy ceiling. This experiment aimed to evaluate the effects of the STE choice on accuracy and convergence behavior.

2. **Weight Clipping**

   For both quantization schemes, the clipping bounds were based on the value $\alpha = \text{mean}|W|$, measured before the first training step, and was varied on a predefined scale $f \in \{0.75, 1.0, 2.0, 4.0\}$. An unclipped run was also included as our baseline ($f = \infty$). Clipping was performed immediately after the gradient update step, keeping latent weights (full-precision hidden weights) within the clipping bounds. The baseline STE `identity_ste` ($h(w) = 1$) was used throughout. Although α was the same quantity in both cases, it played a structurally different role in each quantization scheme (binary and ternary).

   In **TTQ** (ternary), α sets the sparsity threshold $\Delta = 0.7\,\alpha$: weights satisfying $|w| \leq \Delta$ are mapped to zero, while the remainder are mapped to the learned scales

Table 1: STE surrogate functions $h(w)$ evaluated in this study, grouped by functional family. Hyperparameters follow their default implementations unless otherwise stated.

Family	Strategy	$h(w)$
Clipped	`identity_ste` [14]	1
	`clip_ste` [3]	$\mathbf{1}_{\lvert w\rvert\le 1}$
	`leaky_ste`	$\mathbf{1}_{\lvert w\rvert\le 1} + 0.01 \cdot \mathbf{1}_{\lvert w\rvert>1}$
Smooth	`tanh_ste`	$1 - \tanh^2(w)$
	`sigmoid_ste`	$\sigma(w)\big(1-\sigma(w)\big)$
	`softsign_ste`	$\big(1+\lvert w\rvert\big)^{-2}$
Kernel	`triangle_ste`	$\max(0,\, 1-\lvert w\rvert)$
	`polynomial_ste`	$\max\big(0,\, 1-(2\lvert w\rvert-1)^2\big)$
	`cosine_ste`	$\frac{1}{2}\big(\cos(\pi w)+1\big)\cdot\mathbf{1}_{\lvert w\rvert\le 1}$
	`cauchy_kernel_ste`	$\gamma^2/(w^2+\gamma^2),\quad \gamma{=}0.5$
Advanced	`binary_relax_ste`	$(2/T)\,\sigma(w/T)\big(1-\sigma(w/T)\big),\quad T{=}1$

Table 2: Overview of the four ablation experiments.

#	Name	Variable	Architecture
1	STE ablation	STE variants	ResNet-20
2	Weight clipping	Clip factor $f \times \Delta_{\text{init}}$	ResNet-20
3	BN ablation	BN order (pre/post/none)	ResNet-20
4	STE instability	STE variants	ResNet-20

$\pm w_{\text{p/n}}$. The clip bound was therefore expressed relative to this threshold:

$$c_{\text{TTQ}} = f \times \Delta_{\text{init}} = f \times 0.7\,\alpha_{\text{init}}.$$

In **XNOR-Net** (binary), α was the quantized magnitude: $\hat{w} = \alpha \cdot \text{sign}(w)$, making it the scale factor used in the forward pass. The clipping boundary was fully expressed by α:

$$c_{\text{XNOR}} = f \times \alpha_{\text{init}}.$$

Both weight clipping constraints limited the range of the latent weights, reducing how far they could deviate from the quantization thresholds. This helped prevent weights from becoming inactive and kept them within the region where the quantized representation remains a good approximation.

3. **Batch Normalization**

   BN placement was varied across three conditions on ResNet-20 with XNOR/TTQ and `identity_ste`. In all cases, the change was applied uniformly to every convolutional block in the network, including the initial stem layer.

   `pre` The standard ordering: Conv→BN→ReLU. Within each residual block, the skip connection was added *before* the final ReLU, so the non-linearity acted on the summed signal.

   `post` Conv→ReLU→BN. The skip connection was added *after* the second BN, meaning no further activation was applied at the residual junction. Downsampling shortcuts (stride >1 or channel mismatch) retained a BN in both `pre` and `post` conditions.

   `none` BN layers are replaced with `nn.Identity`; the block reduces to Conv→ReLU. Shortcut projections also omit BN, and the skip connection was again summed before the final ReLU, matching the `pre` topology.

   BN parameters were never frozen; all modes were trained from scratch.

4. **STE Instability**

   Starting from a pretrained checkpoint, each STE continued training for 100 epochs using the same learning rate schedule. A stable STE maintains or improves upon the baseline accuracy; an unstable one degrades or oscillates despite the warm-start, directly revealing sensitivity to the gradient estimation error.

3.7 Evaluation Metrics

Primary evaluation used **top-1 validation accuracy**, recorded after every epoch alongside training loss and validation loss. The best validation accuracy across all epochs was reported per run. In addition, five per-layer weight statistics were logged after each epoch:

SQNR Signal-to-quantization-noise ratio (dB): $10\log_{10}\big(\sum w_i^2 / \sum(w_i - Q(w_i))^2\big)$.

MSE / MAE / L_∞ Mean squared error, mean absolute error, and worst-case absolute error of the quantization residual $w - Q(w)$.

Sparsity Fraction of weights mapped to zero (non-zero only under ternary schemes).

Mean / Std First two moments of the latent (full-precision) weight distribution.

Sign-flip rate Fraction of weights whose quantized class changed relative to the previous epoch. Persistent high values indicate that latent weights are oscillating across the quantization threshold rather than converging.

3.8 Software and Hardware

Experiments were implemented in Python 3.11 using PyTorch 2.x. Quantization-aware layers (`QuantConv2d`,

`QuantLinear`) overrode the standard forward pass; each STE was injected via a custom `torch.autograd.Function` that substituted the upstream gradient with $h(w) \cdot g_{\hat{w}}$. All runs were executed on CUDA when available, with an automatic CPU fallback.

4 Results

We report the maximum validation accuracy across 160 training epochs unless otherwise noted. The FP32 ResNet-20 achieved **91.61%**, serving as our baseline.

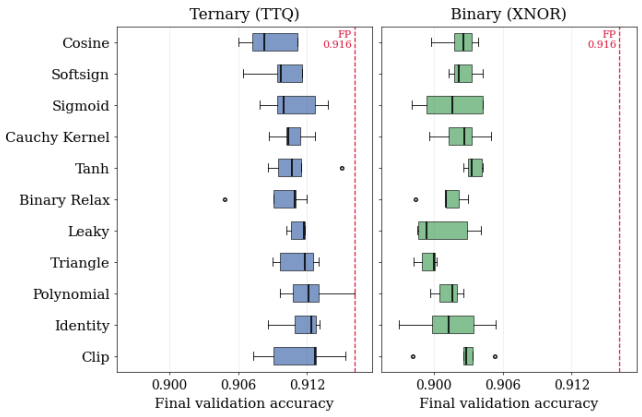

Figure 1: Box-and-whisker plots of final-epoch validation accuracy for each of the 11 STE variants, evaluated under TTQ (ternary, left) and XNOR (binary, right) weight-only quantization on ResNet-20/CIFAR-10 (160 epochs, 5 independent seeds per strategy). Each box spans the interquartile range; whiskers extend to the full range; circles are outliers. The red dashed line marks the full-precision baseline (91.6%).

4.1 Effect of STE Variant

Figure 1 and Table 3 summarize the STE ablation results. All eleven surrogate functions produced competitive accuracy under weight-only quantization, with mean scores ranging from 90.88% (`cosine_ste`) to 91.23% (`polynomial_ste`). The gap between the best-performing quantized model and the FP32 baseline was **0.38 percentage points**, confirming the performance degradation caused by extreme quantization in networks of the same width.

The narrow spread of final validation accuracies across STE variants (0.35 pp for TTQ and 0.4 pp for XNOR) indicates that the network is robust enough that the specific choice of surrogate has little effect on final accuracy. Welch two-sample t-tests against the `identity_ste` baseline confirmed this: no variant reaches statistical significance at $\alpha = 0.05$ (Table 7 in Appendix C), with p-values ranging from 0.087 (`cosine_ste`, TTQ) to 0.927 (`leaky_ste`, TTQ). The closest candidate, `cosine_ste` on TTQ ($p = 0.087$, $\Delta = -0.30$ pp), fell short of statistical significance given the small sample ($n = 5$ seeds), as a difference of this magnitude would require a larger number of seeds to achieve significance.

Regarding optimization dynamics (learning stability and convergence rate), individual learning rates are reported in Appendix E. These results showed broadly similar learning rates across all STE variants, while revealing differences in training stability along the learning curve. Using sign-flip rate as an indicator for convergence, we observed little difference in convergence behavior for the ternary network (Table E in Appendix E). In contrast, the binary network showed a slight increase in early-epoch learning rate for the softsign STE.

Since all STEs converged to comparable validation accuracies and exhibit similar convergence speeds as reflected in the sign-flip rates, the average per-epoch variation per STE (Table 11) served as a useful indicator of training stability. Under this metric, polynomial and softsign STEs appear more stable in the ternary network, while sigmoid and triangle perform better in the binary network. Conversely, clip and identity were the least stable in the binary setting, whereas cosine and binary relax exhibit the lowest stability in the ternary case.

Table 3: STE ablation: final validation accuracy over 5 seeds, ResNet-20, weight only quantization, binary (XNOR) and ternary (TTQ), 160 epochs. Sorted by mean accuracy. FP32 ceiling (mean across seeds): 91.61%.

Ternary (TTQ)

Strategy	Mean (%)	Std (%)	SE (%)
`polynomial_ste`	**91.23**	0.25	0.111
`identity_ste`	91.16	0.19	0.083
`clip_ste`	91.14	0.32	0.143
`leaky_ste`	91.12	0.08	0.034
`triangle_ste`	91.12	0.18	0.080
`tanh_ste`	91.11	0.25	0.110
`sigmoid_ste`	91.08	0.24	0.109
`cauchy_kernel_ste`	91.07	0.15	0.066
`softsign_ste`	90.97	0.21	0.095
`binary_relax_ste`	90.96	0.29	0.128
`cosine_ste`	90.88	0.24	0.105

Binary (XNOR)

Strategy	Mean (%)	Std (%)	SE (%)
`tanh_ste`	**90.35**	0.07	0.033
`softsign_ste`	90.26	0.12	0.054
`clip_ste`	90.25	0.26	0.117
`cauchy_kernel_ste`	90.24	0.20	0.091
`cosine_ste`	90.23	0.16	0.071
`sigmoid_ste`	90.15	0.28	0.126
`identity_ste`	90.14	0.32	0.145
`polynomial_ste`	90.13	0.12	0.052
`binary_relax_ste`	90.12	0.17	0.078
`leaky_ste`	90.07	0.26	0.115
`triangle_ste`	89.95	0.09	0.038

4.2 Effect of Weight Clipping

Applying weight clipping yields a monotonic improvement in accuracy as the clipping factor f increases over the range $[0.75, 4.0]$ (Figure 12). For binary quantization, the best performance was achieved at $f = 2.0$, yielding a 0.1 percentage-point (pp) improvement over the no-clipping baseline. At $f = 4.0$, both binary and ternary quantization outperformed their respective no-clipping baselines. In particular, TTQ attained an accuracy of 91.54%, corresponding to a gain of 0.5 pp, while XNOR reached 90.83%, representing an improvement of 0.26 pp.

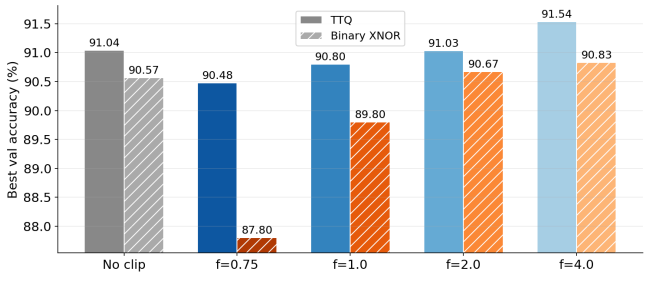

Figure 2: Peak Validation Accuracy on each weight clipping value (no clipping, f=0.75, f=1.0, f=2.0, f=4.0) for both binary (XNOR) and ternary (TTQ) networks (ResNet-20 · CIFAR-10)

Interestingly, at tighter clipping boundaries, the performance of both the binary and ternary networks degraded below the baseline. Binary quantization (XNOR) achieved an accuracy of 87.80% at $f = 0.75$, corresponding to a 2.77 pp decrease in performance. Similarly, ternary quantization (TTQ) achieved an accuracy of 90.48%, representing a 0.44 pp drop relative to the baseline.

Weight clipping also appeared to influence the learning rate and convergence speed, as reflected by the sign-flip rate shown in Figure 3. We observed a clear proportionality between tighter clipping bounds and higher sign-flip rates, suggesting increased effective update sensitivity under stronger clipping. In both networks, the no-clip baseline consistently converged to a more stable regime with minimal sign flipping, indicating more stable quantized weight configurations over training, whereas stricter clipping bounds consistently have a higher sign-flip rate per epoch than other clipping bounds.

Without clipping, both schemes produced heavy-tailed latent weight distributions, with the quantized grid confined to a narrow band at initialization and therefore appearing as small, isolated spikes against the broad latent envelope (Figure 12). Latent weights drift past the quantization grid, increasing the quantization error, leading to smaller SQNR and higher MAE (Table 13 in Appendix G). At low clip factors ($f = 0.75$ and $f = 1.0$), quantization error was significantly reduced. The quantized levels fell outside the clipped latent weight boundaries, so the grid and the distribution it must represent are misaligned. As f increased toward 4.0, the quantized levels spread to match the latent distribution, producing a closer visual overlap between the orange and blue histograms.

4.3 Effect of Batch Normalization

Batch normalization has a significant effect on quantized training (Figure 4). Both binary and ternary networks exhibited a severe drop in performance when batch normalization is removed, with TTQ achieving 82.38% accuracy and XNOR achieving 85.94%. The `pre` ordering (Conv→BN→ReLU) achieved 90.90% on ternary and 90.15% on binary, outperforming the standard `post` ordering (Conv→ReLU→BN) by **0.14** and **0.19** percentage points, respectively.

4.4 STE Instability

Figure 5 shows the fine-tuning trajectories when each STE is applied to the warm-started checkpoint (TTQ identity, 90.96% val accuracy). All strategies exhibited an immediate accuracy drop in the first one to two epochs before recovering: the epoch-1 accuracy ranges from **69.3%** (`cosine_ste`) to **81.4%** (`clip_ste`), with trough epochs occurring as late as epoch 14 for `cauchy_kernel_ste`. By the end of training, all strategies surpassed the warm-start baseline, converging to peak accuracies between **91.30%** (`clip_ste`, `cosine_ste`) and **91.61%** (`leaky_ste`). Stability varied substantially across estimators: `polynomial_ste` shows the smoothest trajectory (epoch std=3.17, total variation=105). In contrast, `identity_ste` is the most volatile (std=3.98, total variation=165), suggesting that estimators with sharper gradient profiles introduce more oscillation when fine-tuning weights that are already close to convergence.

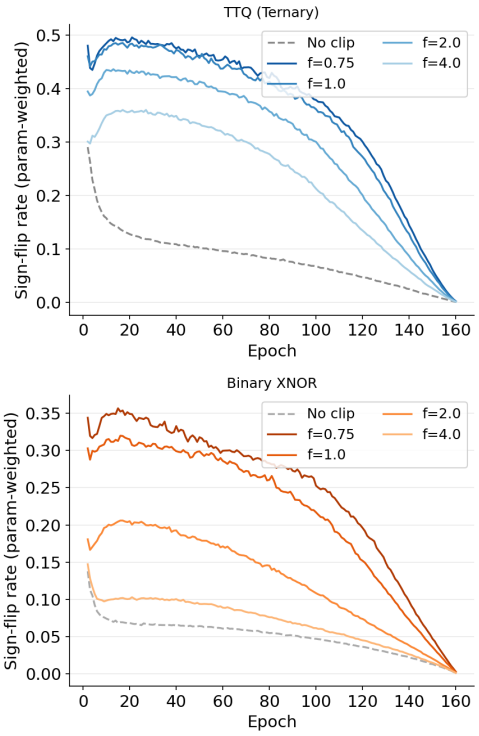

Figure 3: Sign-flip rate per epoch on each weight clipping value (no clipping, f=0.75, f=1.0, f=2.0, f=4.0) for both ternary (above) and binary (below) networks (ResNet-20 · CIFAR-10)

4.5 Optimized Training Configuration

Table 4 reports the peak results across all four experiments, highlighting the optimal parameter settings for each variable. For the subsequent study, we retrained a ResNet-20 model on CIFAR-10 using TTQ (ternary) and XNOR (binary). The final configurations were selected using the previously identified optimal parameters: *polynomial_ste* ($f = 4.0$, pre-BN) for ternary quantization and *tanh_ste* ($f = 4.0$, pre-BN) for binary quantization. Both were trained under the same experimental framework as the STE variant studies (5 seeds, 160 epochs).

Table 5 shows that the optimal training configurations achieved higher final performance and greater stability for

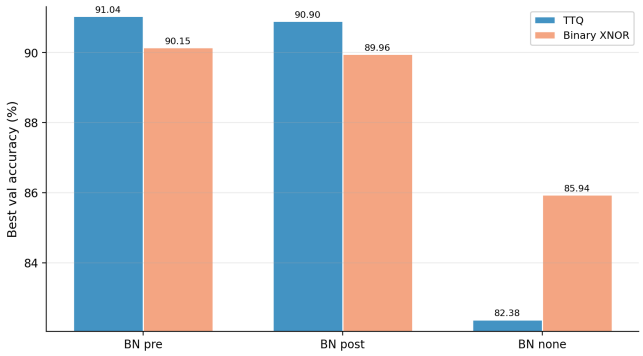

Figure 4: Peak Validation Accuracy on each Batch Normalization configuration (BN none, BN Pre, BN Post) for both binary (XNOR) and ternary (TTQ) networks (ResNet-20 · CIFAR-10)

Table 4: Peak results across all four experiments.

Ternary (TTQ)

Experiment	Best config	Best Val Acc.
1 — STE ablation	`polynomial_ste`	91.23%
2 — Weight clipping	$f = 4.0$	91.54%
3 — BN placement	pre-BN	91.04%
4 — STE instability	`polynomial_ste`	91.17%

Binary (XNOR)

Experiment	Best config	Best Val Acc.
1 — STE ablation	`tanh_ste`	90.35%
2 — Weight clipping	$f = 4.0$	90.83%
3 — BN placement	pre-BN	90.15%
4 — STE instability	`sigmoid_ste`	90.54%
FP32 ceiling	Resnet-20, no quant.	91.61%

both ternary and binary quantization. For TTQ, the proposed configuration achieved a final validation accuracy of 91.52%, representing a 0.40 percentage-point improvement over our reproduction of the TTQ baseline (91.12%, *identity_ste*, no weight clipping, post-BN [8], trained under the identical ResNet-20/CIFAR-10 protocol). Similarly, the optimal XNOR configuration reached a validation accuracy of 90.78%, representing a 0.64 percentage-point gain over the reproduced baseline (90.14% [6]). In addition to improving accuracy, both optimized configurations exhibit reduced variability across runs, as indicated by their lower standard deviations, demonstrating enhanced training stability.

5 Discussion

This study investigated the impact of three training configuration choices for low-bit neural networks: surrogate gradient design, weight clipping, and batch normalization ordering. We analyzed how each component affected training dynamics, focusing on the gradient fidelity of the straight-through estimator (STE), the regularization effects of weight clipping, and the stabilizing role of batch normalization (BN).

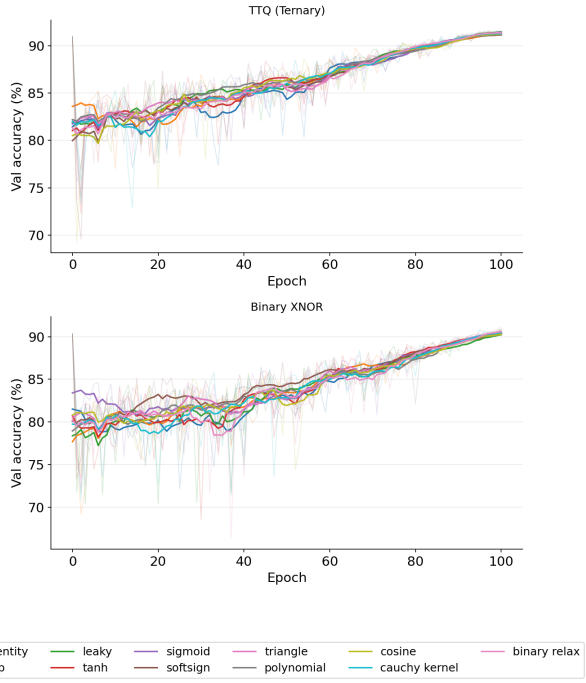

Figure 5: Fine-tuning Validation accuracy per epoch across all eleven STEs after pre-training (XNOR ResNet-20, CIFAR-10)

Table 5: Validation accuracy on ResNet-20 / CIFAR-10. Mean and standard deviation across seeds. The optimized configuration uses TTQ with `polynomial_ste`, Pre-BN, and clip factor f=4.0.

Model	Mean (%)	Std (%)
Full-precision baseline	91.61	—
`TTQ baseline` [8]	91.12	0.19
`Optimal TTQ Configuration`	**91.52**	**0.10**
`XNOR baseline` [6]	90.14	0.32
`Optimal XNOR Configuration`	**90.78**	**0.28**

5.1 STE Design as a Secondary Effect of the Quantization Scheme

Across the evaluated STE variants, performance differences were generally small and, in most cases, statistically insignificant. However, the limited differences that emerged revealed an important insight into the relationship between STE design and the underlying quantization scheme.

Recent work, such as RESTE [5], characterizes STE selection as a trade-off between surrogate gradient approximation accuracy and optimization stability. Under this view, STEs with lower approximation error should provide more accurate gradients, while simpler estimators may offer improved stability at the cost of greater bias. Our results did not fully support this interpretation. During standard training, all evaluated STE variants exhibited similar convergence behavior and comparable levels of stability. More importantly, the instability experiment (Table 8) demonstrated an opposite trend to the expected trade-off.

When training was resumed from a pretrained checkpoint,

`identity_ste`, despite being regarded as one of the most stable estimators, exhibited the greatest degradation in stability. In contrast, `polynomial_ste` achieved both the highest validation accuracy and the strongest stability around the pretrained solution. This observation is consistent with the theoretical analysis of [10], which argues that STEs must provide gradient behavior that remains aligned with the quantization objective near convergence. Since `identity_ste` propagates gradients unchanged ($h(w) = 1$), it continues to update weights even when they have already converged to desirable quantized states, preventing the optimizer from settling at a stable minimum.

The performance of `polynomial_ste` in ternary TTQ is attributable to its alignment with the quantization structure rather than any intrinsic advantage as a surrogate gradient. Its gradient profile suppresses updates near the discrete states and better matches the three-level structure at $\{-1, 0, 1\}$, promoting stable convergence. A similar pattern holds for binary XNOR quantization, where `sigmoid_ste`, `softsign_ste`, and `tanh_ste` perform best due to their smooth approximations of the sign function.

Taken together, these results suggest that STE choice is largely a secondary effect of the quantization scheme itself. Rather than a universally optimal STE existing, the most effective surrogate gradient appears to be the one whose shape best reflects the structure of the corresponding quantization function. Consequently, future STE design may benefit more from tailoring surrogate gradients to specific quantizers than from seeking a single general-purpose estimator.

5.2 Effect of Weight Clipping on Latent Dynamics

Weight clipping directly constrains the evolution of the latent full-precision weights, thereby limiting the mismatch between the latent weight distribution and the quantization grid. It also acts as a regularizer, preventing *quantization drift*, a phenomenon in which latent weights migrate into regions that are poorly represented by the discrete quantization mapping. Overextended training periods can lead to dead weights, reducing the effectiveness of gradient updates and ultimately impairing the network's ability to continue learning.

Our results show that excessively restrictive clipping boundaries lead to poor performance in both binary and ternary networks. However, increasing the clipping range revealed an intermediate regime that outperforms both extreme cases ($f = 0$ and $f \to \infty$). This suggests a trade-off among maintaining latent weights within the effective quantization range, improving quantization fidelity, and preserving weight expressivity. An optimal clipping threshold is therefore expected to balance these competing objectives, acting as a regularizer that mitigates quantization drift and dead weights while retaining sufficient representational capacity to avoid underfitting.

5.3 Batch Normalization and Optimization Stability

Our results show that batch normalization is essential for successfully training binary and ternary networks, and that removing BN results in substantial performance degradation compared to configurations that include normalization. Previous work has suggested that BN plays a critical role in preventing gradient explosion and stabilizing optimization in low-precision networks [19].

In quantized networks, BN normalizes activations to approximately zero mean and unit variance before quantization. This is particularly important when binarization is performed using a zero threshold, as centering the distribution around zero reduces quantization error and improves the representativeness of the resulting binary values [6].

This effect is illustrated in Figure 14. Without BN, both the latent weight distribution and the corresponding quantized weights become noticeably skewed toward one side of the quantization threshold. Such an imbalance reduces the effective utilization of the available quantization levels and can impair gradient flow during training.

The placement of batch normalization significantly affects optimization stability. In the pre-activation configuration, BN is applied before nonlinearities and quantization, preserving stable activation statistics prior to thresholding and yielding smoother optimization dynamics and more balanced quantized representations. In contrast, post-activation BN normalizes already transformed signals, weakening the alignment between activation statistics and quantization thresholds. In low-bit settings, this mismatch increases sensitivity to distributional shifts and can degrade training stability.

6 Future Work

This study is limited to ResNet-20 on CIFAR-10, which, while standard for controlled ablations, may not fully capture behavior in larger-scale architectures or more complex datasets. Additionally, only weight-only quantization is considered; extending these findings to activation quantization may introduce nonlinear effects due to instability in the activation distribution.

Finally, STE formulations are treated as fixed functions, whereas adaptive or learned surrogate gradients may offer improved performance but are outside the scope of this work.

7 Conclusions

In this research paper, we studied the effects of different training configurations, namely surrogate gradient (STE) design, weight clipping, and batch normalization (BN), on optimization dynamics and final performance in binary and ternary neural networks. Across experiments, we observed that overall performance is governed less by any single component in isolation and more by the interaction between gradient approximation, latent weight constraints, and normalization-induced scaling effects.

STE has limited direct influence on training dynamics, but it is necessary because it preserves gradient flow under the chosen quantization scheme, making its effectiveness depend on compatibility with that scheme and on reducing gradients at quantization boundaries.

In contrast, weight clipping plays a more direct role in shaping latent weight dynamics. It introduces a fundamental tradeoff: tighter clipping improves regularization and reduces quantization drift, but overly restrictive bounds reduce model

capacity and can lead to underfitting. This implies the existence of an optimal clipping range that balances these competing effects.

Lastly, batch normalization is found to be essential for stable low-bit training. BN reduces distributional skew, improving the quantized weights.

A central takeaway is that there are many possible training configurations, and an optimal one may vary across low-bit training tasks. The results suggest that optimal performance in low-bit networks is less about selecting a single "best" STE or regularizer, and more about ensuring coherence between (i) gradient approximation, (ii) weight distribution control, and (iii) normalization-induced scale management. When training configuration choices are well selected, there exists an optimal training approach that can increase performance and training stability.

8 Responsible Research

This section outlines the primary ethical considerations relevant to the research. Throughout the study, we focused on ensuring reproducibility, the responsible and ethical use of large language models (LLMs), and environmental sustainability. The subsections below describe the measures taken to address and uphold these principles.

8.1 Reproducibility

The experiments are fully documented and made publicly available to ensure transparency and reproducibility. The model architecture used can be found in Appendix B, and the database that we use is publicly available. The complete codebase has been released publicly to allow full replication of all results and to facilitate further research building on this work[1]. To ensure experimental consistency, all runs used fixed random seeds. In addition, we employed incremental seed values across experiments to mitigate the "good seed" bias and reduce the risk of reporting results dependent on favorable initialization.

8.2 Usage of LLMs

Large language models (LLMs) were used to assist with programming the testbench, rewriting, and improving the paper's content. The extent of this use includes confirming implementations, generating plots, fixing grammar, and suggesting ways to make content clearer and more concise. All other content, such as experimental design, research, and analysis, was developed independently.

8.3 Environmental Responsibility

Our research promotes the use of extreme quantized models. The implications of using a quantized model mean more efficient inference by reducing the great computational demands of floating-point operations and reducing the memory requirements of large models. As a result, both memory bandwidth usage and energy consumption are reduced, enabling more efficient deployment on resource-constrained hardware and lowering the overall environmental footprint associated with model inference.

[1]The full source code is available online here: https://github.com/robin-kronos/research-project-efficient-learning

References

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015.

[2] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 160–167, 2008.

[3] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4107–4115, 2016.

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *arXiv preprint arXiv:1511.00363*, 2015.

[5] Xiao-Ming Wu, Dian Zheng, Zuhao Liu, and Wei-Shi Zheng. Estimator meets equilibrium perspective: A rectified straight through estimator for binary neural networks training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17009–17018, 2023.

[6] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[7] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2018.

[8] Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

[9] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. ReActNet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision (ECCV)*, 2020.

[10] Penghang Yin, Raphael Gontijo Lopes, et al. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Machine Learning (ICML)*, 2019.

[11] Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox,

and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[12] Xinquan Chen, Junqi Gao, Biqing Qi, Dong Li, Yiang Luo, Fangyuan Li, and Pengfei Li. Fast and slow gradient approximation for binary neural network optimization, 2024.

[13] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

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

[15] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[16] Yingsong Luo and Ling Chen. Daq: Density-aware post-training weight-only quantization for llms, 2024.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[18] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[19] Eyyüb Sari and Vahid Partovi Nia. Batch normalization in quantized networks, 2020.

A Common Surrogate Shapes

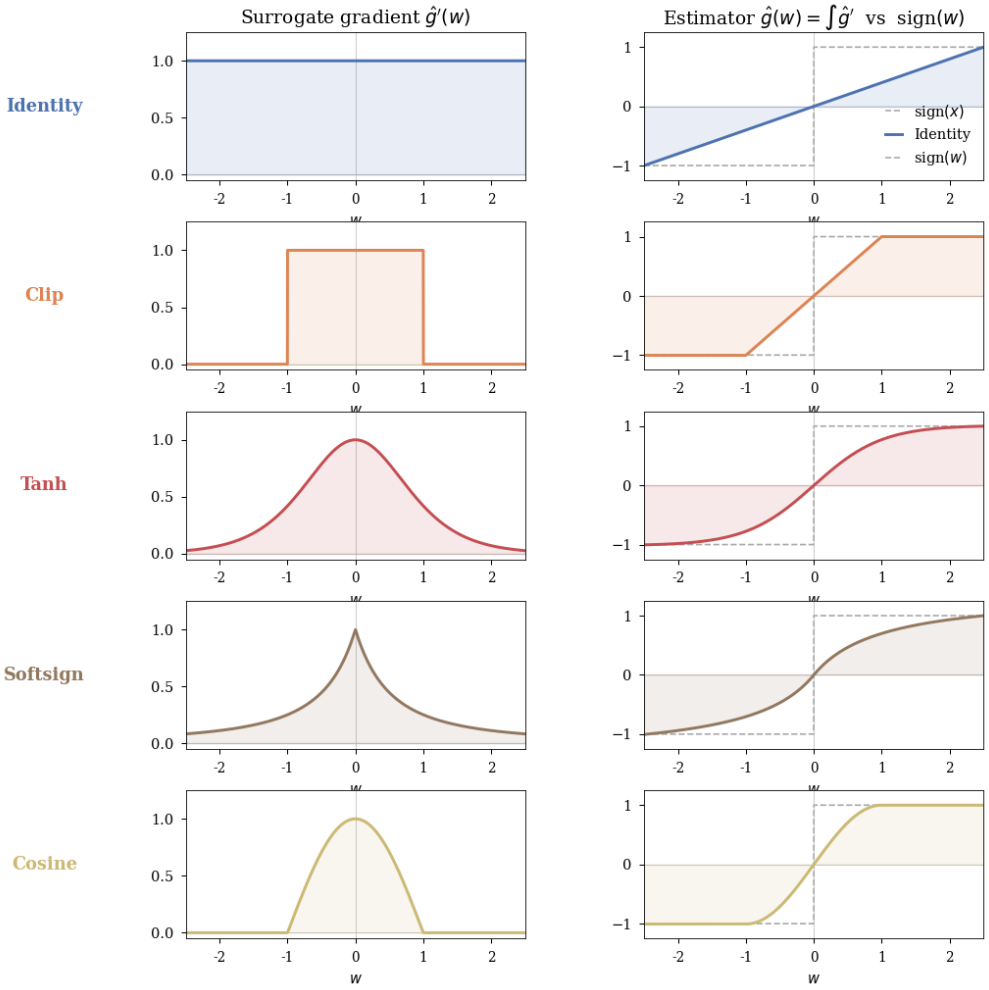

Figure 6: Surrogate gradient $\hat{g}'(w)$ (left) and its integral $\hat{g}(w)$ as an implicit estimator of $\mathrm{sign}(w)$ (right, dashed) for five common STE variants. Differences in support and decay rate determine how aggressively each STE attenuates gradients for weights far from the quantization boundary.

B Network Architecture Details

Table 6: Layer-by-layer summary of both architectures used in this study. All convolutional layers use 3×3 kernels with padding 1. Convolutions inside BasicBlock use `QuantConv2d`; The initial convolution and classifiers are full-precision.

Architecture	Layer / Stage	Configuration
ResNet-20	Initial conv	3×3, 16 ch (FP32)
	Stage 1 (×3)	BasicBlock (QuantConv2d), 16 ch
	Stage 2 (×3)	BasicBlock (QuantConv2d), 32 ch
	Stage 3 (×3)	BasicBlock (QuantConv2d), 64 ch
	Global avg pool	$8\times8 \rightarrow 1\times1$
	Classifier	Linear $64 \rightarrow 10$ (FP32)

C Validation Accuracy Significance Test

Table 7: STE ablation: final validation accuracy and Welch two-sample t-test versus `identity_ste` (n=5 seeds, two-tailed). No variant reaches $p < 0.05$ in either scheme.

Scheme	Strategy	Mean (%)	Std (%)	SE (%)	Δ (%)	t	p
TTQ	`polynomial_ste`	**91.23**	0.25	0.111	+0.08	0.55	0.599
	`identity_ste`	91.16	0.19	0.083	—	—	—
	`clip_ste`	91.14	0.32	0.143	−0.01	−0.07	0.944
	`leaky_ste`	91.12	0.08	0.034	−0.03	−0.38	0.720
	`triangle_ste`	91.12	0.18	0.080	−0.04	−0.33	0.750
	`tanh_ste`	91.11	0.25	0.110	−0.05	−0.36	0.727
	`sigmoid_ste`	91.08	0.24	0.109	−0.08	−0.59	0.576
	`cauchy_kernel_ste`	91.07	0.15	0.066	−0.09	−0.83	0.433
	`softsign_ste`	90.97	0.21	0.095	−0.18	−1.44	0.189
	`binary_relax_ste`	90.96	0.29	0.128	−0.20	−1.31	0.232
	`cosine_ste`	90.88	0.24	0.105	−0.28	−2.07	0.074
XNOR	`tanh_ste`	**90.35**	0.07	0.033	+0.21	1.39	0.232
	`softsign_ste`	90.26	0.12	0.054	+0.12	0.75	0.487
	`clip_ste`	90.25	0.26	0.117	+0.10	0.56	0.592
	`cauchy_kernel_ste`	90.24	0.20	0.091	+0.10	0.56	0.594
	`cosine_ste`	90.23	0.16	0.071	+0.09	0.53	0.614
	`sigmoid_ste`	90.15	0.28	0.126	+0.01	0.06	0.952
	`identity_ste`	90.14	0.32	0.145	—	—	—
	`polynomial_ste`	90.13	0.12	0.052	−0.01	−0.09	0.931
	`binary_relax_ste`	90.12	0.17	0.078	−0.03	−0.16	0.880
	`leaky_ste`	90.07	0.26	0.115	−0.07	−0.37	0.723
	`triangle_ste`	89.95	0.09	0.038	−0.19	−1.25	0.270

D Learning Curves of STE ablation test

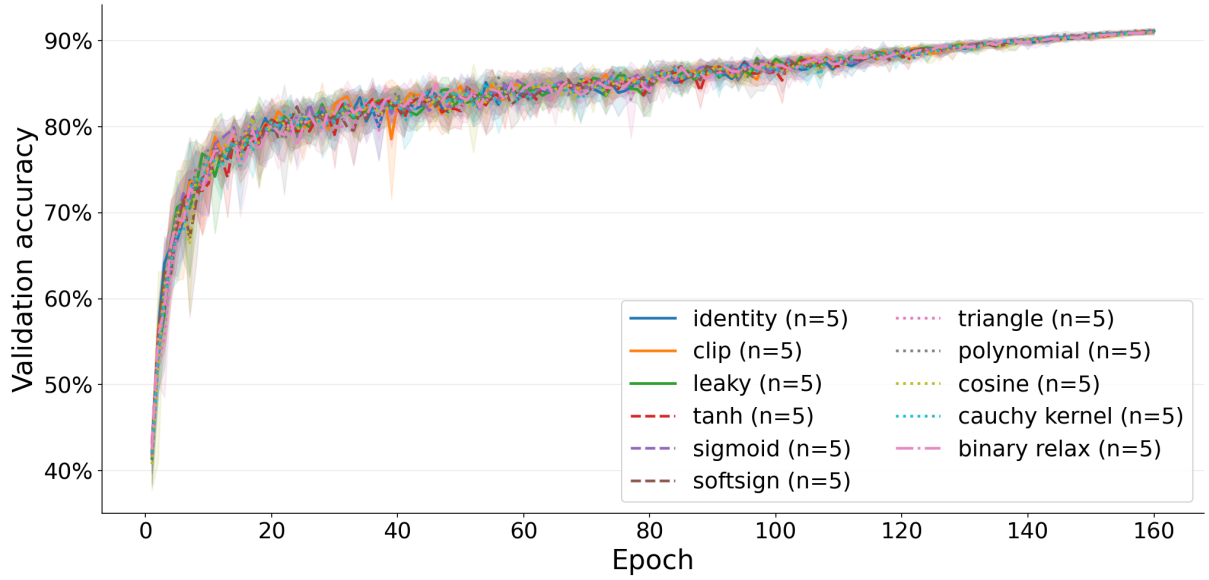

Figure 7: Validation accuracy over 160 training epochs for all 11 STE variants under TTQ (ternary) quantization on ResNet-20/CIFAR-10, averaged across 5 seeds. Shaded bands show ±1 standard deviation.

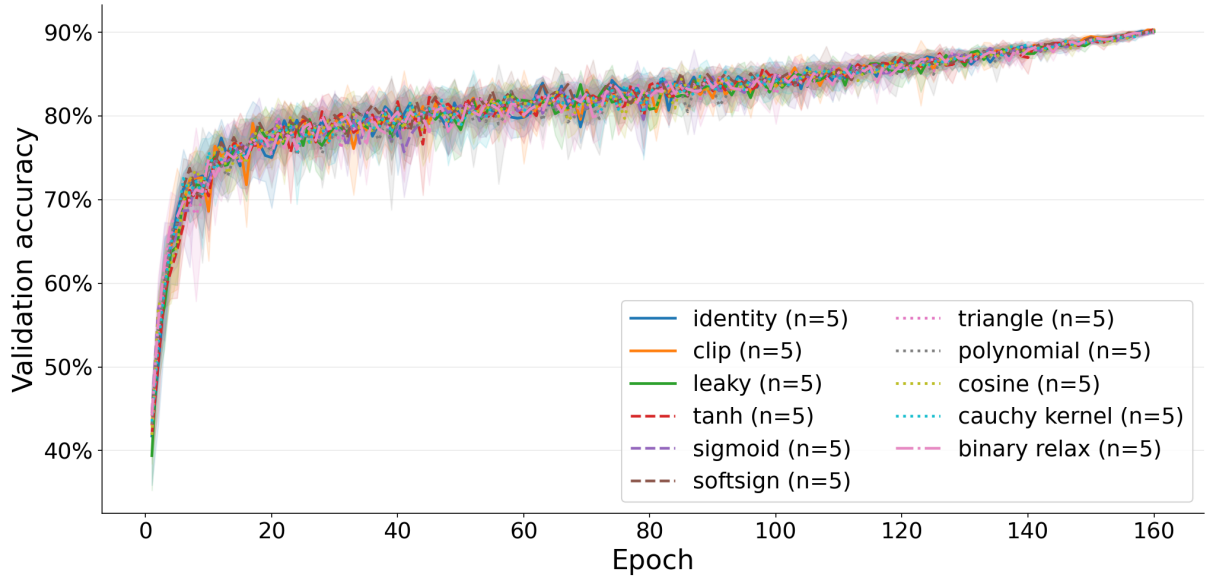

Figure 8: Validation accuracy over 160 training epochs for all 11 STE variants under XNOR (binary) quantization on ResNet-20/CIFAR-10, averaged across 5 seeds. Shaded bands show ±1 standard deviation.

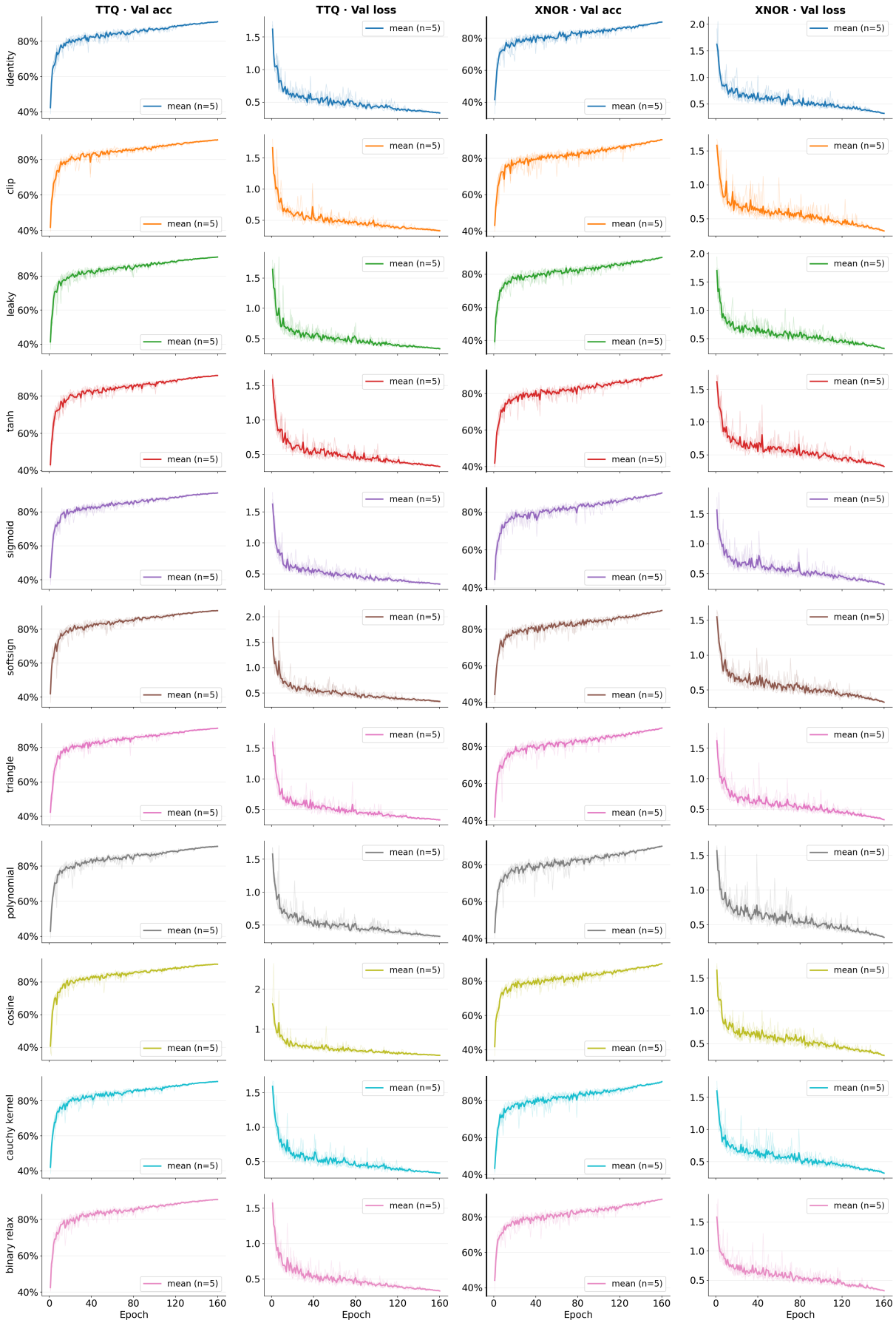

Figure 9: Individual learning curves with val accuracy & loss for each strategy (ResNet-20, CIFAR-10)

E Evaluation Metrics of STE ablation experiment

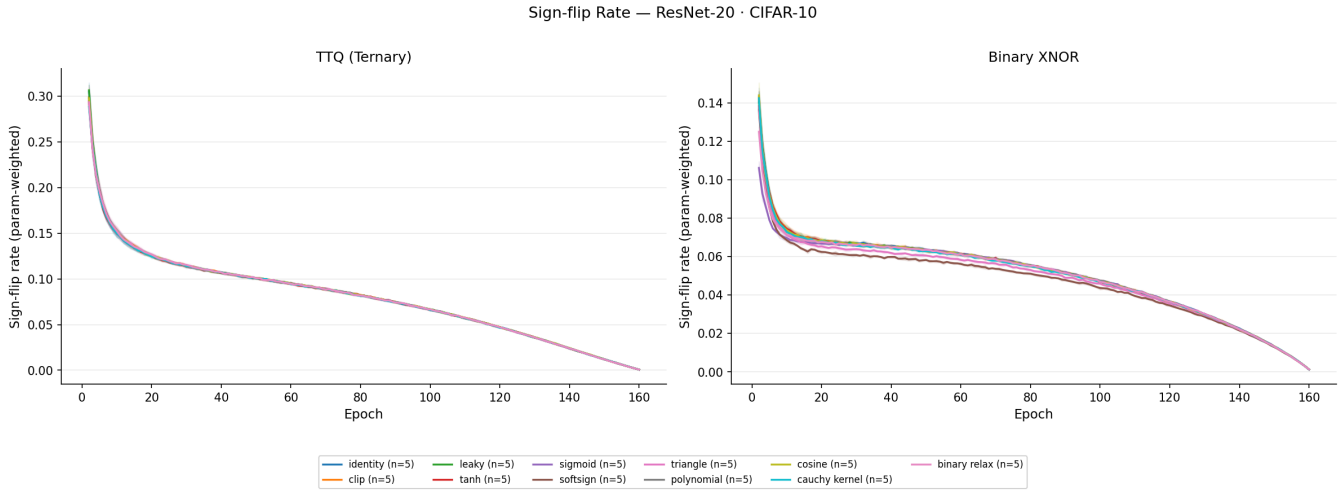

Figure 10: Aggregate sign flip rates of the STE ablation test (ResNet-20, CIFAR-10)

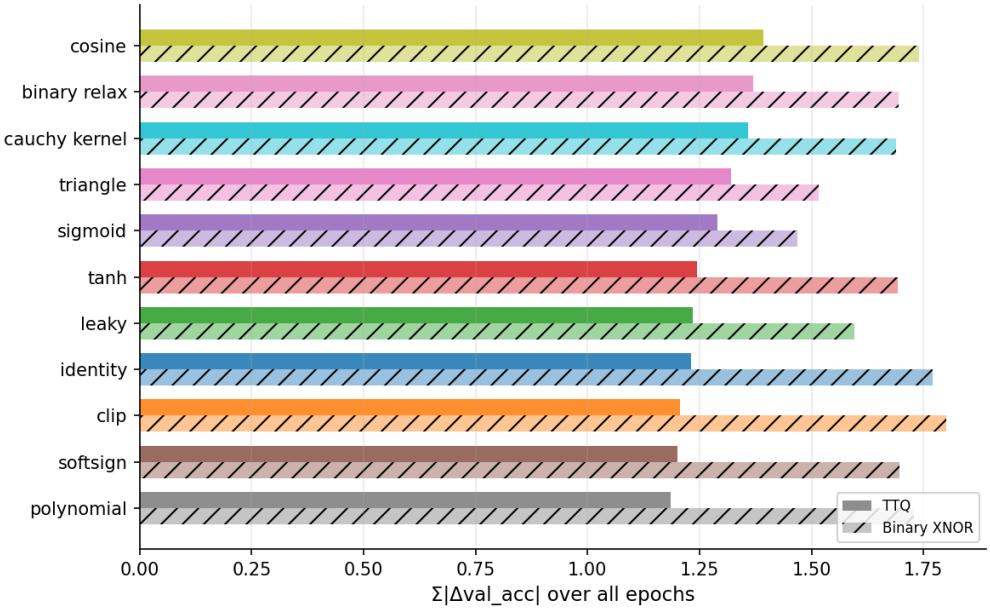

Figure 11: Average variation per epoch over each STE (ResNet-20, CIFAR-10)

F Weight Clipping Experiment Histograms

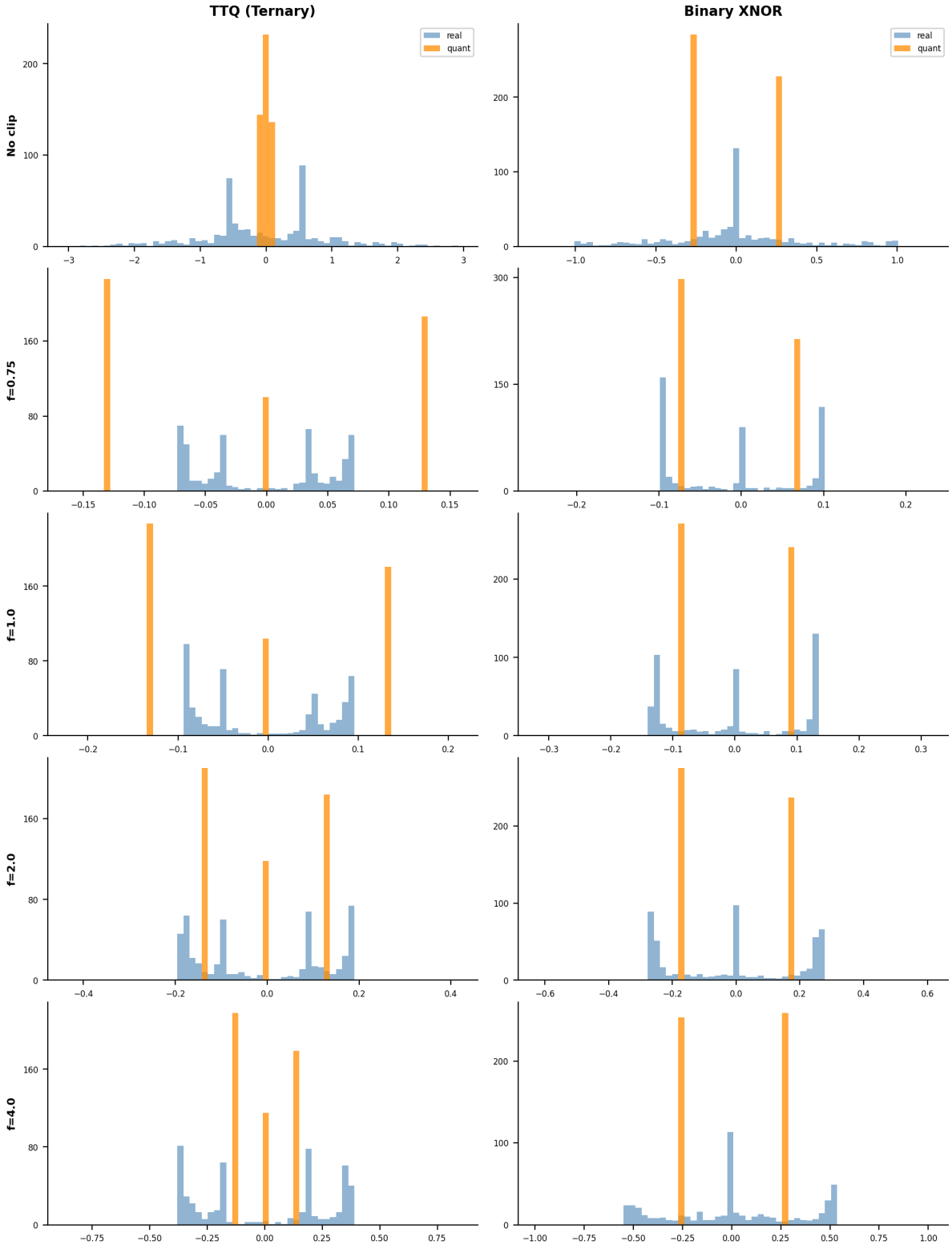

Figure 12: Histogram of both latent and quantized weights of a single layer between ternary (TTQ) and binary (XNOR) networks for different weight clip values

G Weight Clipping Metrics

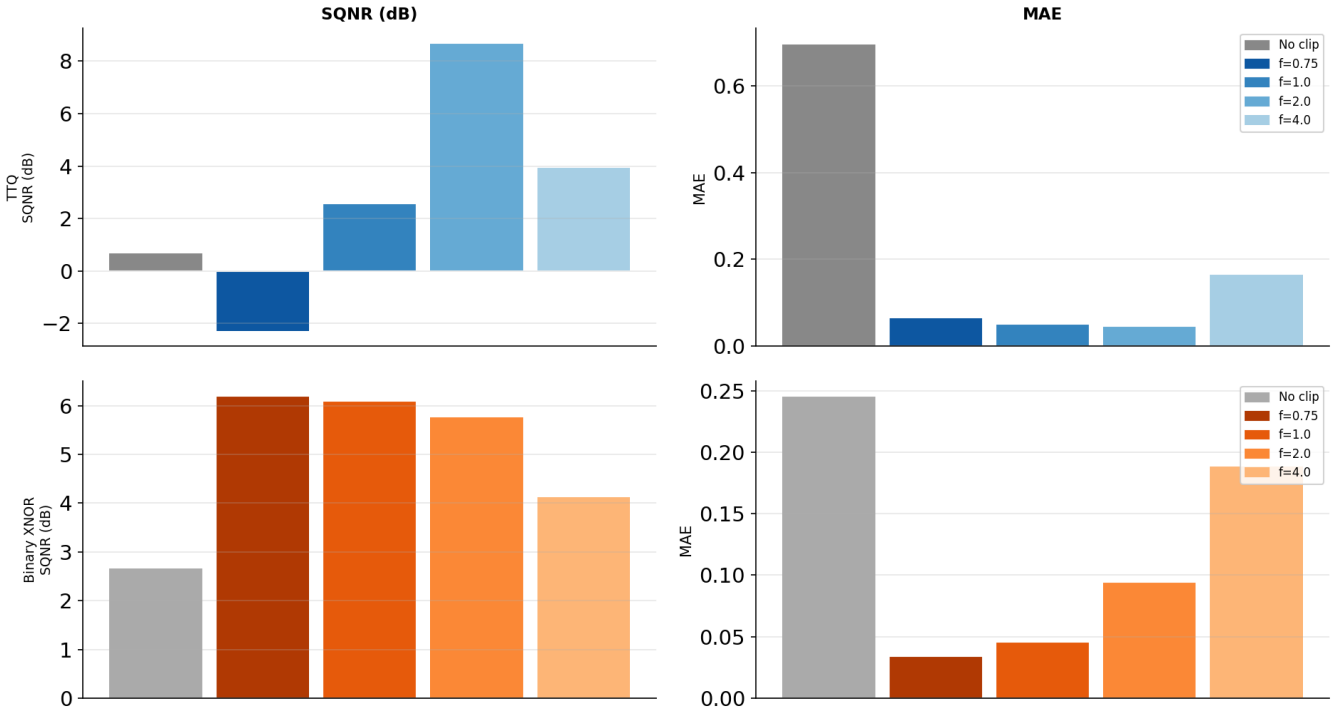

Figure 13: Quantization error metrics (SQNR and MAE) at the final training epoch, across TTQ (top) and binary XNOR (bottom) weight clipping factors. Each bar represents a clip factor; higher SQNR and lower MAE indicate better weight fidelity under quantization.

H Batch Normalization Experiment Histograms

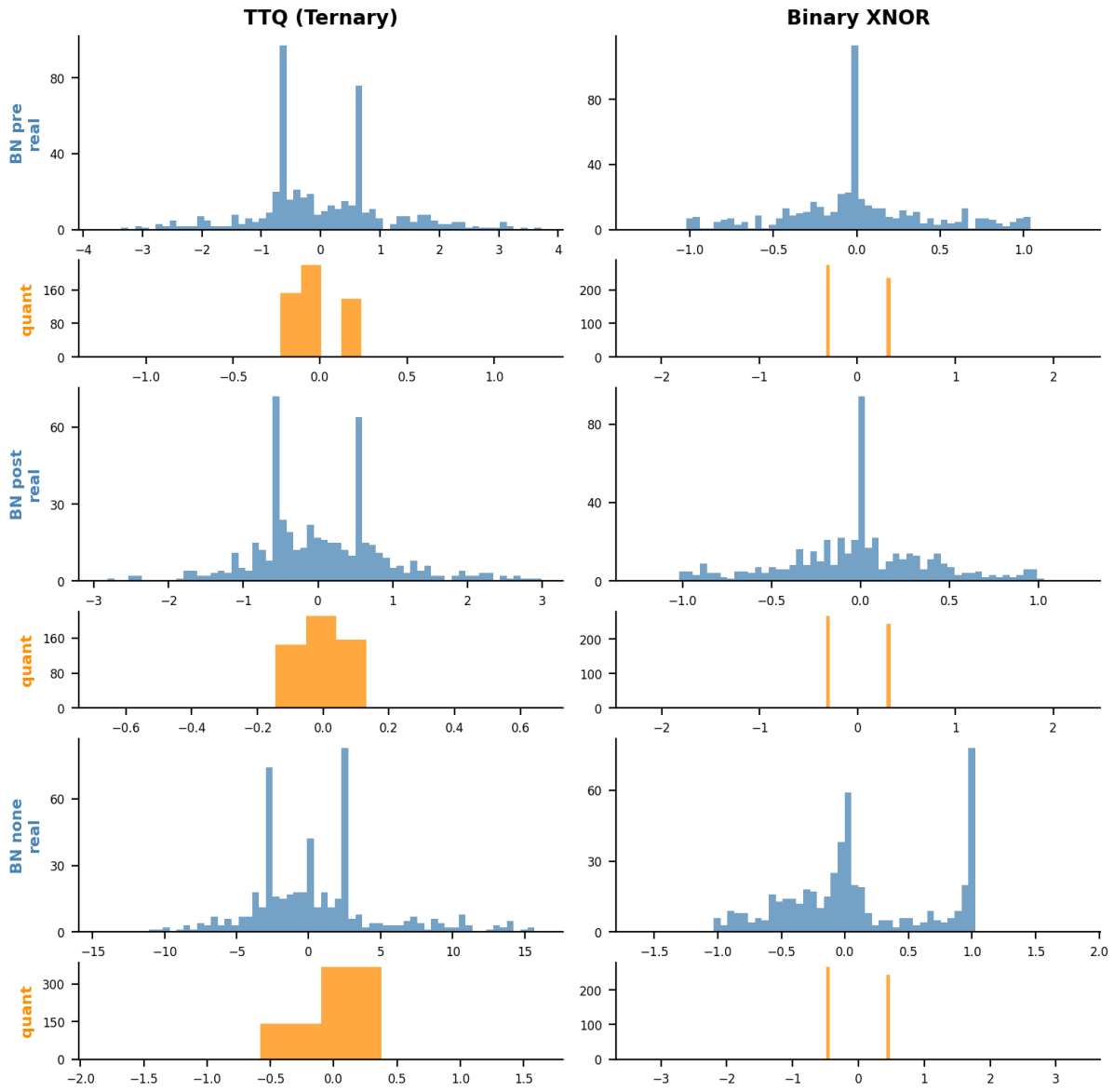

Figure 14: Histogram of both latent and quantized weights of a single layer between ternary (TTQ) and binary (XNOR) networks between different batch normalization placements.

I Accuracy and Loss curves of instability test

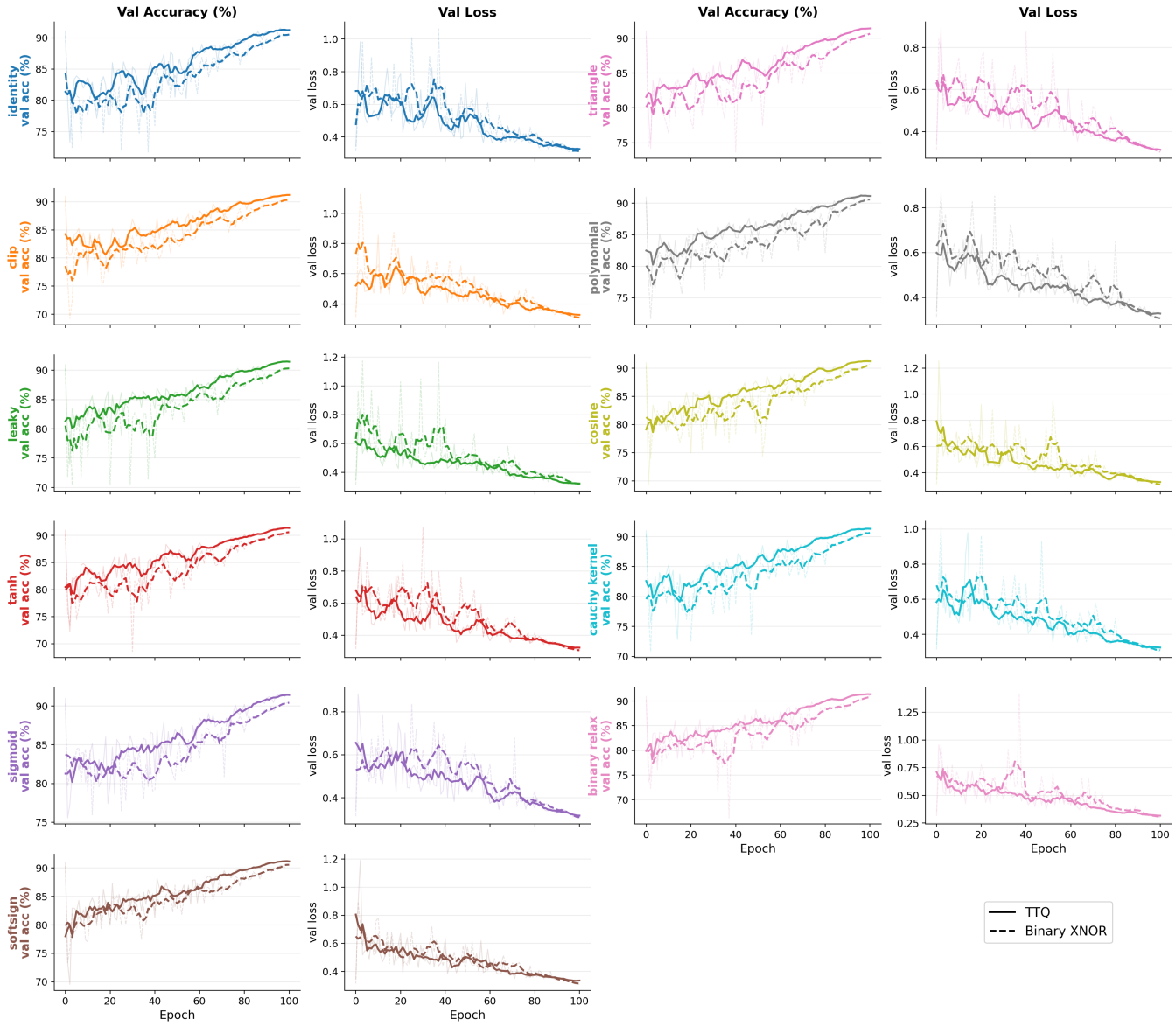

Figure 15: Per-strategy validation accuracy and loss over 100 fine-tuning epochs, starting from a quantized warm-start checkpoint, for all 11 STE variants under TTQ (solid) and binary XNOR (dashed) quantization on ResNet-20/CIFAR-10. Raw epoch values are shown faintly behind a 5-epoch centered moving average.

J Instability test results and metrics

Table 8: Instability metrics for each STE after fine-tuning from the ternary (TTQ) and binary (XNOR) quantized warm-start checkpoints (ResNet-20, CIFAR-10, 100 epochs). Strategies are sorted by epoch accuracy and standard deviation (in descending order). *Final* = last-epoch validation accuracy; *Std* = standard deviation of the per-epoch validation accuracy curve; *TV* = total variation $\sum |\Delta \text{acc}|$. Bold indicates the best value in each column.

	TTQ (Ternary)			Binary XNOR		
STE	Final	Std	TV	Final	Std	TV
`identity`	91.15%	3.98	164.7	90.60%	4.54	236.8
`cosine`	91.28%	3.89	144.1	90.73%	4.00	215.8
`softsign`	91.13%	3.83	162.1	90.90%	3.64	180.2
`cauchy kernel`	**91.53%**	3.79	135.2	90.77%	4.31	227.2
`binary relax`	91.43%	3.74	158.2	**91.12%**	4.48	236.4
`tanh`	91.45%	3.66	154.5	90.84%	4.49	208.4
`sigmoid`	91.52%	3.64	162.9	90.54%	**3.58**	186.9
`leaky`	91.47%	3.44	140.9	90.49%	4.66	254.5
`clip`	91.30%	3.44	142.0	90.56%	4.13	**149.5**
`triangle`	91.49%	3.32	120.4	90.70%	3.96	178.3
`polynomial`	91.17%	**3.17**	**105.4**	90.80%	3.89	176.8