DISSECTING OUTLIER DYNAMICS IN LLM NVFP4 PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training large language models using 4-bit arithmetic enhances throughput and memory efficiency. Yet, the limited dynamic range of FP4 increases sensitivity to outliers. While NVFP4 mitigates quantization error via hierarchical microscaling, a persistent loss gap remains compared to BF16. This study conducts a longitudinal analysis of outlier dynamics across architecture during NVFP4 pretraining, focusing on *where they localize, why they occur, and how they evolve*. We find that, compared with Softmax Attention (SA), Linear Attention (LA) reduces per-tensor heavy tails but still exhibits block-level spikes under block quantization. Our analysis attributes outliers to specific architectural components: Softmax in SA, gating in LA, and SwiGLU in FFN, with "post-QK" operations exhibiting higher sensitivity. Notably, outliers evolve from transient spikes early in training to a small set of persistent hot channels (i.e., channels with persistently large magnitudes) in later stages. Based on these findings, we introduce **Hot-Channel Patch (HCP)**, an online compensation mechanism that identifies hot channels and reinjects residuals using hardware-efficient kernels. We then develop **CHON**, an NVFP4 training recipe integrating HCP with post-QK operation protection. On GLA-1.3B trained for 60B tokens, CHON reduces the loss gap to BF16 from 0.95% to 0.59% while maintaining downstream accuracy.

1 INTRODUCTION

The computational cost associated with pretraining large language models (LLMs) continues to escalate as parameter counts, token budgets, and context lengths increase. Consequently, mixed-precision training, particularly FP8 pipelines Peng et al. (2023); Fishman et al. (2024); DeepSeek-AI (2024), has emerged as a key solution, reducing memory usage and accelerating computation while maintaining performance at scale. Nevertheless, the efficiency frontier is now increasingly determined by native support for 4-bit arithmetic, which offers further reductions in memory footprint compared to FP8 and increases peak arithmetic throughput on next-generation accelerators.

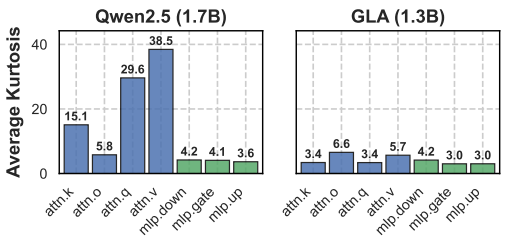


Figure 1: **Activation kurtosis of GLA vs. Qwen2.5.** Lower kurtosis values in GLA reflect reduced heavy-tailedness and fewer outliers.

Transitioning from FP8 to FP4 represents a significant reduction in precision. The limited representational capacity of FP4 increases sensitivity to heavy-tailed distributions and outliers in activations and weights. Although most tensor values are well-behaved, a small fraction of outliers can saturate quantizers, producing large quantization errors that propagate through depth and destabilize training. To address this, modern hardware introduces microscaling formats like NVFP4, employing tensor-level and block-level scaling to suppress quantization noise Rouhani et al. (2023). Nevertheless, state-of-the-art NVFP4 recipe NVIDIA et al. (2025) still exhibits a notable loss gap relative to BF16, and this gap may increase during the late learning-rate decay stage.

A primary challenge in narrowing this gap is the dynamic nature of outliers during pretraining. Most existing outlier analyses focus on post-training quantization Dettmers et al. (2022); Wei et al. (2023);

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

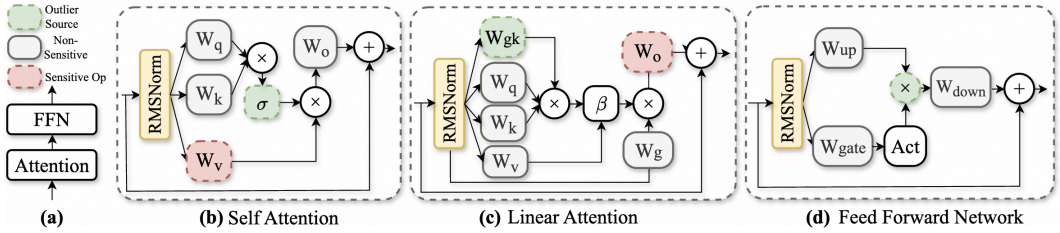


Figure 2: **Architectural overview and sensitivity analysis.** Detailed components of (a) residual blocks, (b) Softmax Attention, (c) Gated Linear Attention, and (d) gated-FFN.

Lee et al. (2024); Zhao et al. (2024), which are typically conducted offline on trained checkpoints, aiming to characterize static channel-wise extremes. In contrast, the distributions of internal tensors evolve during pretraining, and the interaction between outliers and block-wise quantization shifts across training stages and model architectures. Therefore, systematically dissecting these outlier dynamics is a critical yet underexplored prerequisite for enabling large-scale NVFP4 pretraining.

In this paper, we present a longitudinal analysis of outlier dynamics across Softmax (`SA`) and Linear Attention (`LA`) architectures in NVFP4 pretraining. We use Qwen3 Yang et al. (2025a) and Gated Linear Attention (GLA) as representative `SA` and `LA` models, respectively. This study yields three consistent findings. ① `LA` demonstrates milder per-tensor heavy tails than `SA`, as indicated by activation kurtosis trends (Fig. 1), although localized spikes at the block level relevant for NVFP4 quantization still occur. ② We find "post-QK" operations (i.e. projections after QK interactions such as W_v or W_o) are more sensitive than other operations, as shown in Fig. 2. ③ Outliers transition from drifting spikes early in training to persistent hot channels (i.e., channels with large magnitude) later, indicating that they can be identified and protected without recomputation each step.

Building on these observations, we introduce Hot-Channel Patch (HCP), a lightweight mechanism that periodically identifies hot channels and compensates their quantization residuals using hardware-efficient kernels. We derive a second-order MSE upper bound for idealized two-sided compensation on selected channels in Lemma A.9, with the assumptions for comparing actual errors stated in Theorem A.10. HCP is integrated with the NVFP4 recipe and targeted post-QK operation protection for sensitive post-QK operations, leading to a new recipe called Compensated Hot-channel Optimization for NVFP4 (CHON). CHON reduces the NVFP4-to-BF16 loss gap from 0.95% to 0.59% on a 1.3B GLA model trained for 60B tokens, while maintaining downstream accuracy. Also, we evaluate the NVFP4 recipe for supervised fine-tuning and reinforcement learning to assess transferability beyond pretraining. Our contributions are as follows:

- **Longitudinal outlier analysis for NVFP4 pretraining.** We track outlier dynamics throughout training and uncover a transition from transient spikes to persistent hot channels in later stages.
- **Architecture- and operation-level characterization.** We show that `LA` reduces global heavy tails relative to `SA`, and "post-QK" operations are more sensitive.
- **CHON recipe and Hot-Channel Patch.** We propose the CHON recipe that incorporates "post-QK" protection and Hot-Channel Patch (HCP) to reduce the quantization error in persistent hot channels.

2 Related Work

Outliers in LLMs. Outliers refer to data points that differ significantly from other data points in a distribution. They are observed during both training Park et al. (2025); NVIDIA et al. (2025) and inference Sun et al. (2024); Xiao et al. (2022). Although outliers can be beneficial and are often necessary for model expressiveness and capability Raman et al. (2025); Dettmers et al. (2022); An et al. (2025); Bondarenko et al. (2024); He et al. (2024), they also pose significant challenges for quantization and low-precision training. For instance, LLM.int8() Dettmers et al. (2022) demonstrates that forcing outliers into low precision leads to performance collapse, proposing instead to retain them in FP16. Moreover, SmoothQuant Xiao et al. (2022) and AWQ Lin et al. (2024) identify that there are more channel outliers in activations than weights. They employ smoothing factors to migrate quantization difficulty from activations to weights, balancing their respective ranges. SPQR Dettmers et al. (2023) and QUIK Ashkboos et al. (2023) further exploit the static nature of outlier channels in PTQ by isolating them into high-precision branches, enabling near-lossless weight compression

and 4-bit inference. BitNet a4.8 Wang et al. (2024) applies higher-precision computation to out_proj and down_proj in a 1-bit LLM framework, selecting these layers based on their sensitivity. These approaches, however, are confined to inference-time PTQ or mixed-precision inference scenarios and do not address how outlier channels emerge and evolve during low-bit pretraining.

Architectural factors shaping outliers. (An et al., 2025) attribute activation outliers in Transformers to the Softmax in Softmax Attention. Softmax Attention combines quadratic time and memory with exponential weighting, which can amplify rare extreme dot-product scores into activation spikes Vaswani et al. (2017). Linear Attention Gu & Dao (2023); Dao & Gu (2024); Yang et al. (2024); Sun et al. (2023b); Qin et al. (2024) removes Softmax via associative or kernelized normalization and achieves $O(T)$ computation, typically with smoother activation statistics Katharopoulos et al. (2020); Choromanski et al. (2020); Peng et al. (2021). Studies also examine architectural factors beyond attention that contribute to outliers. For example, Wei et al. (2023) identify that the SwiGLU activation in FFN layers can produce outliers due to its quadratic weight alignment. Furthermore, Attention Sink Xiao et al. (2023); OpenAI (2025) can alleviate outliers in Softmax Attention based LLMs Son et al. (2024). QK Normalization Yang et al. (2025a) can also suppress outliers by normalizing the query and key vectors before dot-product attention computation.

NVFP4 Pretraining. Next-generation hardware (e.g., Blackwell GPU) makes NVFP4 and MXFP4 promising for boosting kernel efficiency and throughput. Recent research attempts NVFP4 pretraining. For example, MSRA Wang et al. (2025) updates the straight-through estimator with a differentiable gradient estimator during backward. AWS Tseng et al. (2025) employs Random Hadamard Transform (RHT) and Stochastic Rounding (SR) during backward to ensure unbiased gradient estimates on MXFP4 pretraining. Chmiel et al. Chmiel et al. (2025) use RTN forward and SR backward to stabilize training, adding a BF16 recovery phase at the end. Quartet Castro et al. (2025) introduces native 4-bit training on Blackwell GPUs using the MXFP4 format. By employing a recipe that combines Hadamard transforms with clipping-aware quantization, it achieves near-lossless performance relative to FP16. Metis Cao et al. (2025) proposes to split tensors into a small low-rank branch and quantization-friendly residual. TetraJet-v2 Chen et al. (2025) introduces unbiased double-block quantization incorporating OsciReset and OutControl mechanisms. The NVIDIA NVFP4 recipe NVIDIA et al. (2025) establishes a robust quantization framework tailored for large-scale pretraining. On a 12B Hybrid Mamba-Transformer model with 10T tokens, it maintains a 1% loss gap vs. BF16, widening to 1.5% during learning rate decay.

3 Outlier Dynamics: Where, Why, and How

Definition. We define *outliers* as values that deviate significantly from the central mass of a distribution, typically manifesting as heavy tails. For NVFP4 pretraining, these outliers induce quantization errors, numerical instability, and performance degradation. We monitor outliers both statically (from checkpoints) and dynamically (during training).

Flush-to-Zero (FTZ). Under NVFP4, values are quantized to FP4 E2M1 after applying an effective encode scale per block, as detailed in App. C.4. FTZ denotes the ratio of *underflow-to-zero* events where scaled values collapse to exactly zero. For a tensor X partitioned into blocks with effective encode scale $s_{\mathrm{enc},b}$, then FTZ is:

$$\frac{1}{|X|}\sum_i \mathbb{1}\Big\{Q_{\mathrm{E2M1}}\big(x_i \cdot s_{\mathrm{enc},b}\big) = 0\Big\},$$

where b denotes the block containing element i and $Q_{\mathrm{E2M1}}(\cdot)$ is the FP4 E2M1 quantizer. A higher FTZ indicates greater underflow risk and irreversible loss of small-but-nonzero signals.

Kurtosis. For a variable x with mean μ and standard deviation σ, excess kurtosis Westfall (2014) is defined as: $\kappa(x) = \frac{\mathbb{E}\left[(x-\mu)^4\right]}{\sigma^4} - 3$. We monitor κ per tensor and per block for activations and weights during training. Heavy-tailed distributions yield larger κ, indicating higher outlier propensity and elevated risk of underflow (flush-to-zero) under FP4.

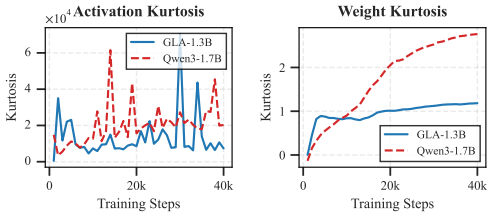
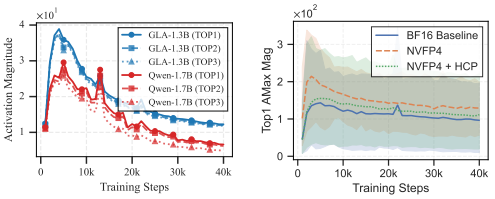


Figure 3: **Evolution of per-tensor kurtosis.** GLA-1.3B (blue) demonstrates superior weight distribution stability compared to Qwen3-1.7B (red), whereas activation kurtosis exhibits similar stochastic volatility across both architectures.



(a) Activation Magnitude. (b) GK Proj. Activation.

Figure 4: **Analysis of Activation Evolution.** (a) Top-k activation magnitudes for GLA-1.3B and Qwen-1.7B. (b) Evolution of the GK projection top-1 activation under BF16, NVFP4, and HCP.

3.1 Where Do Outliers Emerge?

We first locate outlier emergence across architectures and components, specifically comparing Softmax and Linear Attention architectures, and examining the sensitivity of "post-QK" operations.

`LA` exhibits lighter tails than `SA`. We compare the per-tensor and per-block kurtosis of each component of Softmax-based Transformers (e.g., Qwen) against `LA` models (e.g., GLA). ① *Per-tensor Kurtosis:* As shown in Fig. 1, GLA exhibits lower activation kurtosis than Qwen, indicating weaker heavy tails and fewer extreme outliers. In Qwen, Attention exhibit higher kurtosis than MLP, where the outlier comes from softmax operation An et al. (2025). (See Fig. 17 and Fig. 18 for extended results.) ② *Per-block Kurtosis:* To investigate the impact on NVFP4 quantization, we analyzed kurtosis at a 16×16 block. Despite the superior global stability of `LA`, the per-block spikes in Fig. 5 demonstrate that `LA` still experiences high local kurtosis in specific regions, posing challenges for block-based quantization. Refer to App. E.1 for more details.

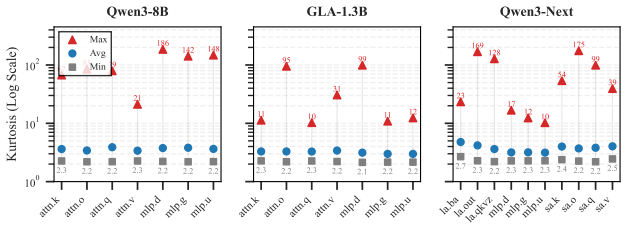


Figure 5: Comparison of block-level kurtosis for Softmax and Linear Attention. Localized "heavy tails" persist at the block level even when per-tensor kurtosis remains moderate.

"Post-QK" Operation Sensitivity. As shown in Fig. 2, we identify the outlier sources and sensitive operations (sensitive ops), observing that both `SA` and `LA` exhibit significant sensitivity in their "post-QK" components (see Tab. 2). Specifically, for `SA` (e.g., Qwen3-1.7B), the value projection is the sensitive bottleneck, whereas for `LA` (e.g., GLA-1.3B), the output projection emerges as the critical sensitive operator, as shown in Tab. 2. Notably, these sensitive regions are consistently situated downstream of outlier sources. In `SA`, the value projection follows the QK interaction and softmax operation, a stage where outliers are more prevalent. In `LA`, the output projection follows the gk_proj and g_proj layers, where outlier magnitudes reach their peak. As shown in Fig. 4b, the activation range of gk is exceptionally broad, reaching magnitudes as high as 350, which imposes significant numerical pressure on the subsequent Output projection. For more operation quantization error analysis, refer to App. E.10 Therefore, preserving these sensitive "post-QK" operations in higher precision can benefit low-precision pretraining.

3.2 Why Do Outliers Emerge?

Softmax as Outlier Source in `SA`. Prior research An et al. (2025) identifies Softmax as the root cause of outliers due to its normalization constraint. Specifically, the sum-to-one requirement forces the model to adopt an extreme dynamic range to effectively suppress uninformative tokens. To assign near-zero contributions to these tokens, the model is compelled to generate Pre-Softmax values with large magnitudes. We empirically validate this mechanism by tracking three metrics in Fig. 6: Post-Softmax Entropy, Pre-Softmax Kurtosis, and Pre-Softmax Max. ① The decline in Post-Softmax Entropy reflects an increasing concentration of attention weights. ② The rise in Pre-Softmax Kurtosis

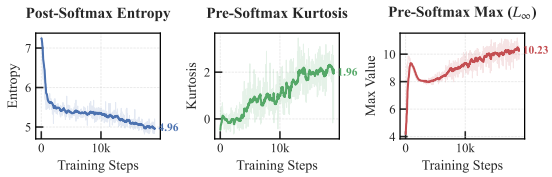


Figure 6: **Evolution of softmax-induced instability.** Increasing Pre-Softmax Kurtosis and $L_{+\infty}$ norms alongside declining entropy quantify the structural emergence of systematic outliers.

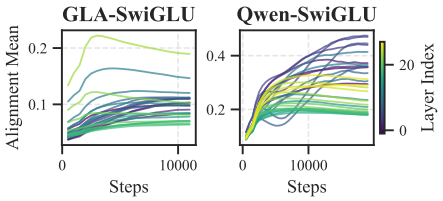


Figure 7: **SwiGLU weight cosine alignment dynamics.** Layer-wise mean cosine similarity between W_{up} and W_{gate}.

indicates a shift toward heavy-tailed distributions in the logit space, a hallmark of outlier formation. ③ The growth of the Pre-Softmax Max (from 4 to 10) confirms the intensifying logit disparity necessary to saturate the Softmax function.

Gating as Outlier Source in LA. Our analysis identifies the elementwise exponential function $\phi(x) = \exp(t \cdot x)(t > 2)$ in the gating mechanism, specifically the gk_proj layer, as the primary source for extreme outliers in GLA. The decay factor λ_t is derived from gk_proj via a log-sigmoid function. To achieve state resetting ($\lambda_t \approx 0$), the pre-activation input must be extremely negative (e.g., ≈ -120), whereas long-term retention ($\lambda_t \approx 1$) requires positive saturation. This necessitates a massive dynamic range (e.g., $[-120, 80]$) that poses severe challenges for uniform FP4 quantization. As illustrated in Fig. 4b, the average Top-1 magnitude of gk_proj significantly surpasses that of other components (see App. E.7 for extended analysis).

SwiGLU as the Outlier Source in FFNs. As a cornerstone of modern LLMs, SwiGLU is defined as $\text{SwiGLU}(x) = (xW_{\text{up}}) \odot \text{Swish}(xW_{\text{gate}})$, exhibiting quadratic growth scaling with $\mathcal{O}(\|x\|^2)$. Prolonged training with weight decay (ℓ_2 regularization) induces weight alignment ($W_{\text{up}} \parallel W_{\text{gate}}$) Fishman et al. (2024), effectively transforming SwiGLU into a potent outlier amplifier. These quadratic spikes inflate block-level scale factors in NVFP4, causing normal-magnitude values within the same block to be flushed to zero (underflow/FTZ), which degrades training fidelity (see App. E.6 for direct FTZ measurements). We empirically verify this mechanism in Fig. 7 by monitoring the cosine alignment $\frac{|W_{\text{up},i}^\top W_{\text{gate},i}|}{\|W_{\text{up},i}\| \|W_{\text{gate},i}\|}$, which shows a steady increase throughout training. Furthermore, GLA produces significantly fewer outliers than Qwen, as shown in Fig. 3). This is attributed to the lower attention activations in GLA, which mitigate the amplification effect on subsequent FFN layers.

Outlier Impact of RMSNorm. We analyze the impact of RMSNorm on outlier by visualizing the distribution of the learnable scaling factor γ in Fig. 29 and App. E.8. We observe several key trends: (i) Softmax Attention yields substantially higher γ magnitudes ($\gamma > 1$) compared to Linear Attention ($\gamma < 1$); (ii) smaller models, such as Qwen3-0.6B, possess larger γ values than larger variants like Qwen3-30B-A3B; (iii) models within the same family (e.g., RetNet 1.3B and 2.7B) display consistent distribution patterns across scales; and (iv) Qwen3-Next Yang et al. (2025a) exhibits generally $\gamma < 1$ across layers, suppressing the activation magnitude layer by layer.

3.3 How Do Outliers Evolve?

From Drifting Spikes to Fixed Hot Channels. Early training stages (e.g., steps 400–5,400) exhibit dynamic, transient magnitude spikes that drift across channels as the model navigates the loss landscape (see Fig. 8). However, as training progresses toward mid-to-late stages, the location of outliers stabilizes. Persistent hot channels (i.e., channels with persistent high magnitude) concentrate in a small subset of channels as shown in Fig. 19. This transition from stochastic drift to structural fixation justifies the use of static, selective mitigation strategies (i.e., Hot-Channel Patch (HCP) in Sec. 4) over dynamic adaptation. This evolution is visible across checkpoints in Fig. 22 and App. E.4.

Top-k Magnitude Evolution. Tracking the top-k magnitude channels reveals a pattern: outliers start with extreme volatility and high magnitudes, then gradually stabilize and decrease (or plateau) as the model converges (see Fig. 4a). The gap between top-1 and top-3 magnitudes narrows, indicating numerical stress diffuses; yet specific components like gk projection maintain high magnitudes

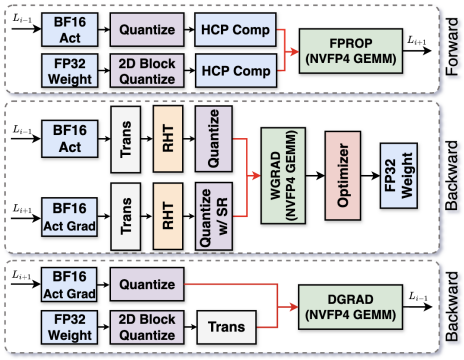


Figure 9: **Computational workflow of NVFP4-quantized linear layers.** CHON integrates HCP in the forward pass.

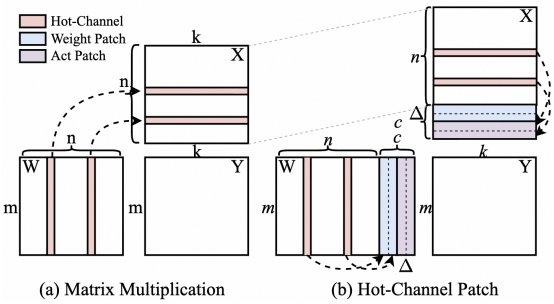


Figure 10: **Schematic of the Hot-Channel Patch (HCP) mechanism.** (a) Standard quantized GEMM. (b) HCP error compensation via residual channel injection for weights and activations.

throughout training, behaving as permanent outliers and dominating the top-1 trajectory in Fig. 4b and App. E.3.

Kurtosis over Training. Activation kurtosis remains volatile during early training: Qwen3 shows repeated spikes while GLA exhibits fewer but sharper bursts, consistent with transient outlier channels in Fig. 3. Weight kurtosis diverges more smoothly—Qwen3 climbs steadily and plateaus > 2, whereas GLA rises quickly then flattens near ≈ 1.1, indicating lighter tails. The late-stage plateau suggests heavy-tailed behavior is learned and persists even after convergence.

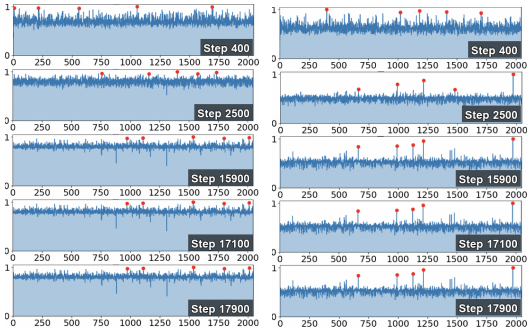


Figure 8: **Temporal evolution of activation outliers in Up and Output projections.** Red markers illustrate a clear transition from transient, drifting spikes (Step 400) to persistent, spatially fixed "hot channels" (Step 15k).

Flush-to-Zero Analysis. We track NVFP4 flush-to-zero (FTZ) ratio for activations and weights over training. As shown in Fig. 26 and 27: ① activations exhibit higher FTZ than weights, while weights remain robust; ② enabling HCP reduces the activation FTZ and pulls distributions closer to BF16; and ③ gating components (e.g., g_proj and gate_proj) exhibit non-monotonic FTZ dynamics (decrease then increase), contrasting with monotonic increases elsewhere. This reduced FTZ likely occurs because gating activations, designed to excite/suppress neurons, inherently avoid values near zero. These findings align with our outlier-source analysis in Sec. 3.1 and App. E.6.

4 CHON Recipe

Building upon the outlier dynamics, we propose **C**ompensated **H**ot-channel **O**ptimization for **N**VFP4 (**CHON**), integrating the NVFP4 baseline with Hot-Channel Patch (HCP) and post-QK operation protection (Fig. 9).

NVIDIA NVFP4 Recipe. We adopt the NVFP4 training recipe NVIDIA et al. (2025) as a strong baseline (Fig. 9). Specifically, it (i) preserves the last four layers in high precision, (ii) employs 2D block scaling for weights and 1D scaling for activations and gradients, (iii) utilizes RTN in the forward pass and SR with RHT in the backward pass.

Hot-Channel Patch (HCP). Outliers become concentrated in specific channels that remain consistently active through the middle and later stages of the observed training run (Sec. 3). Leveraging this finding, **HCP** periodically identifies these channels and rejects their quantization residuals via a hardware-efficient kernel, balancing correction quality and training throughput. HCP applies residual

Table 1: **Zero-shot performance on downstream benchmarks with 40B tokens.** Acc_n denotes length-normalized accuracy; Avg is the mean across reported metrics. Each setting uses one final checkpoint; ± reports per-example evaluation standard error, not variation across training seeds.

Setting	ARC-C		ARC-E		HellaSwag		OBQA		PIQA		SciQ		Winograd	Avg
	Acc	Acc_n	Acc	Acc_n	Acc	Acc_n	Acc	Acc_n	Acc	Acc_n	Acc	Acc_n	Acc	
GLA-340M														
BF16	29.3±1.3	31.4±1.4	63.9±1.0	58.5±1.0	37.9±0.5	47.0±0.5	24.6±1.9	36.2±2.1	70.1±1.1	70.5±1.1	85.9±1.1	78.6±1.3	53.3±1.4	52.9
FP8	27.9±1.3	28.4±1.3	62.6±1.0	54.2±1.0	35.8±0.5	43.2±0.5	24.4±1.9	34.8±2.1	67.5±1.1	68.4±1.1	84.0±1.2	74.7±1.4	52.2±1.4	50.6
NVFP4	**28.2±1.3**	30.7±1.4	62.6±1.0	56.1±1.0	36.9±0.5	45.3±0.5	23.8±1.9	35.8±2.1	**69.0±1.1**	68.0±1.1	84.4±1.1	78.0±1.3	53.8±1.4	51.7
CHON	28.0±1.3	**30.8±1.4**	**63.0±1.0**	**57.2±1.0**	**37.2±0.5**	**46.7±0.5**	**24.6±1.9**	**36.4±2.1**	68.3±1.1	**68.4±1.1**	**86.5±1.1**	**80.7±1.2**	**54.1±1.4**	**52.5**
Gated-DeltaNet-340M														
BF16	17.6±1.1	22.9±1.2	41.1±1.0	38.5±1.0	27.0±0.4	28.1±0.4	14.0±1.6	27.2±2.0	57.7±1.1	57.3±1.1	70.4±1.4	62.0±1.5	50.0±1.4	39.5
FP8	26.6±1.3	29.0±1.3	60.0±1.0	53.3±1.0	35.0±0.5	42.6±0.5	24.4±1.9	35.6±2.1	67.7±1.1	67.7±1.1	84.2±1.1	77.0±1.3	51.8±1.4	50.4
NVFP4	28.6±1.3	32.9±1.4	**64.2±1.0**	57.2±1.0	38.0±0.5	48.0±0.5	25.2±1.9	**36.8±2.2**	**69.4±1.1**	68.7±1.1	86.2±1.1	81.4±1.2	**54.5±1.4**	53.2
CHON	**29.7±1.3**	**33.5±1.4**	63.3±1.0	**57.4±1.0**	**38.2±0.5**	**48.0±0.5**	**26.0±2.0**	36.2±2.1	69.2±1.1	**69.2±1.1**	**87.2±1.1**	**82.1±1.2**	53.0±1.4	**53.3**
GSA-1B														
BF16	18.9±1.1	21.8±1.2	35.1±1.0	33.8±1.0	26.1±0.4	25.3±0.4	13.8±1.5	25.0±1.9	55.4±1.2	54.4±1.2	40.9±1.6	42.6±1.6	49.5±1.4	34.0
FP8	18.9±1.1	21.8±1.2	35.1±1.0	33.8±1.0	26.1±0.4	25.3±0.4	13.8±1.5	**25.0±1.9**	55.4±1.2	**54.3±1.2**	40.9±1.6	42.5±1.6	49.5±1.4	34.0
NVFP4	**19.0±1.2**	**23.6±1.2**	36.2±1.0	34.6±1.0	**26.5±0.4**	26.2±0.4	15.0±1.6	24.6±1.9	**55.6±1.2**	53.4±1.2	40.8±1.6	40.4±1.6	**51.5±1.4**	34.4
CHON	18.2±1.1	22.3±1.2	**37.2±1.0**	**35.7±1.0**	25.9±0.4	**26.3±0.4**	**15.8±1.6**	24.4±1.9	55.5±1.2	53.2±1.2	**45.9±1.6**	**45.8±1.6**	49.7±1.4	**35.1**
Qwen3-1.7B														
BF16	20.4±1.2	22.5±1.2	31.8±1.0	30.2±0.9	25.8±0.4	26.3±0.4	12.2±1.5	24.6±1.9	54.9±1.2	52.6±1.2	35.4±1.5	31.3±1.5	49.6±1.4	32.1
FP8	20.2±1.2	22.6±1.2	40.5±1.0	38.0±1.0	26.4±0.4	27.7±0.4	14.4±1.6	25.4±1.9	57.0±1.2	54.3±1.2	49.8±1.6	45.4±1.6	**52.3±1.4**	36.5
NVFP4	**20.4±1.2**	**24.3±1.2**	26.6±0.9	26.9±0.9	25.7±0.4	25.4±0.4	12.0±1.5	24.2±1.9	53.3±1.2	50.0±1.2	21.8±1.3	24.1±1.4	52.1±1.4	29.7
CHON	20.1±1.2	23.6±1.2	**45.3±1.0**	**40.5±1.0**	**27.3±0.4**	**29.4±0.4**	**14.8±1.6**	**27.8±2.0**	**57.5±1.1**	**58.3±1.1**	**56.1±1.6**	**49.5±1.6**	50.7±1.4	**38.5**
GLA-7B														
BF16	37.0±1.4	41.0±1.4	72.3±0.9	66.5±1.0	45.0±0.5	59.1±0.5	28.8±2.0	39.6±2.2	72.3±1.0	73.0±1.0	90.7±0.9	85.8±1.1	59.9±1.4	59.3
FP8	37.2±1.4	40.1±1.4	73.7±0.9	65.7±1.0	45.9±0.5	59.8±0.5	31.4±2.1	42.2±2.2	72.4±1.0	72.5±1.0	91.1±0.9	85.8±1.1	60.6±1.4	59.9
NVFP4	**39.3±1.4**	**41.9±1.4**	74.8±0.9	68.2±1.0	**46.8±0.5**	61.6±0.5	**32.8±2.1**	43.2±2.2	**73.7±1.0**	74.0±1.0	92.0±0.9	**87.7±1.0**	**61.6±1.4**	**61.4**
CHON	**39.3±1.4**	40.6±1.4	**75.1±0.9**	**69.3±1.0**	46.7±0.5	**61.9±0.5**	32.0±2.1	**43.4±2.2**	73.1±1.0	**75.0±1.0**	**92.3±0.8**	87.3±1.1	61.2±1.4	61.3

compensation only to forward GEMMs; Dgrad and Wgrad retain the baseline NVFP4 backward recipe without HCP compensation.

Given a linear transformation $\mathbf{Y} = \mathbf{W}^\top \mathbf{X}$ with $\mathbf{X} \in \mathbb{R}^{n\times t}$ and $\mathbf{W}^\top \in \mathbb{R}^{m\times n}$ (n channels, t samples, m outputs), we define the composite quantization operator $\mathcal{Q}(\cdot) = D(Q(\cdot))$, which maps floating-point values to their dequantized representations in a target format (e.g., FP4). The quantized activations and weights are denoted by $\widehat{\mathbf{X}} = \mathcal{Q}(\mathbf{X}) = \mathbf{X} + \Delta\mathbf{X}$ and $\widehat{\mathbf{W}} = \mathcal{Q}(\mathbf{W}) = \mathbf{W} + \Delta\mathbf{W}$ respectively, where $\Delta\mathbf{X}$ and $\Delta\mathbf{W}$ denote the residual errors introduced by $\mathcal{Q}(\cdot)$.

Identity 4.1 (Error decomposition of a quantized product)**.** *The quantized (low-precision, LP) product admits the following exact algebraic decomposition. See Lemma A.3 for the detailed derivation. While straightforward, this identity reveals that HCP can be folded into a single concatenated GEMM call (the S-mode in App. B.1), which is the key insight enabling our single-kernel implementation.*

$$\underbrace{\widehat{\mathbf{W}}^\top \widehat{\mathbf{X}}}_{\textit{LP term}} = \underbrace{\mathbf{W}^\top \mathbf{X}}_{\textit{HP term}} + \underbrace{\mathbf{W}^\top \Delta\mathbf{X} + \Delta\mathbf{W}^\top \mathbf{X}}_{\textit{first-order error}} + \underbrace{\Delta\mathbf{W}^\top \Delta\mathbf{X}}_{\textit{second-order error}} \tag{1}$$

HCP offers three dimensions for error compensation: ① **Mode** (`S/D`) determines the kernel implementation—*Single(S)* fuses residual channels with one-pass GEMM, while *Dual(D)* computes the GEMM and residual correction in separate kernels, as presented in Alg. 1. ② **Order** denotes the remaining error in the idealized selected-channel product: `O1` uses a single-sided patch and leaves a first-order term; `O2` uses two-sided patches and leaves a second-order interaction. ③ **Target** specifies which residuals are patched: weights (`W`), activations (`A`), or both (`B`); patch residuals are $\mathbf{R}^w = -\Delta\mathbf{W}$ and $\mathbf{R}^x = -\Delta\mathbf{X}$. (See App. B.1.)

We evaluate the quantization error under Laplace and Gaussian priors across HCP configurations (Fig. 11) and select `S-O2-B`: a single-kernel, second-order scheme that patches both weights and activations. Specifically, let $\mathcal{I} \subseteq \{1, \ldots, n\}$ with $|\mathcal{I}| = k$ denote the channels to patch. For each channel j, we compute its importance score as:

$$s_j = \frac{1}{t} \left\|\Delta\mathbf{X}_{j,:}\right\|_1 + \frac{1}{m} \left\|\Delta\mathbf{W}_{j,:}\right\|_1 . \tag{2}$$

where the ℓ_1-norms quantify the sensitivity of activations and weights to quantization perturbations. The *hot channels* are determined by the top-k highest scores: $\mathcal{I} = \text{top-}k(\mathbf{s})$.

The idealized error analysis in App. A.3 motivates two-sided compensation. HCP packs $\mathbf{W}' = [\mathbf{W}; \mathbf{R}_{\mathcal{I}}^{w}; \widehat{\mathbf{W}}_{\mathcal{I}}]$ and $\mathbf{X}' = [\mathbf{X}; \widehat{\mathbf{X}}_{\mathcal{I}}; \mathbf{R}_{\mathcal{I}}^{x}]$ along their input-channel dimension, then quantizes both packed tensors before the NVFP4 GEMM (Fig. 10, Alg. 1). Here the subscript $\mathcal{I}$ selects rows. With exact patch arithmetic and a decoded base product, the selected-channel contribution satisfies $\mathbf{Y}_{\mathcal{I}}^{\text{ideal}} = \mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}} - \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}$. App. A separates this identity from unselected-channel error and errors introduced by packed quantization and GEMM arithmetic. Packing expands the inner dimension from n to $n + 2k$, giving a relative GEMM arithmetic increase of $2k/n$. Unlike PTQ methods such as SPQR Dettmers et al. (2023), QUIK Ashkboos et al. (2023), DecDEC Park et al. (2024) and BiLLM Huang et al. (2024), HCP operates online during pretraining. In particular, SPQR and QUIK adopt an *exact-recovery* double-kernel strategy: hot channels are extracted and recomputed in full precision via a separate kernel launch, yielding stronger correction at the cost of additional launch overhead. Our design occupies a different point in this tradeoff space: the S-mode folds channel compensation into a single concatenated GEMM, sacrificing exact recovery for lower launch overhead and better training throughput. We emphasize that `S-O2-B` is a *training-oriented* formulation—chosen for efficiency rather than theoretical optimality—and our experiments validate its practical value without precluding a stronger exact-recovery D-mode alternative (e.g., exact masking with BF16 recomputation analogous to QUIK), which we leave for future work. Compared with low-rank-plus-residual Metis Cao et al. (2025) and single-sided activation compensation in TetraJet-v2 Chen et al. (2025), HCP simultaneously patches weight and activation residuals while keeping the patched matmul in low precision; our ablation study in Fig. 32 shows this dual-side recovery yields lower MSE than activation-only fixes. HCP also benefits from fixed nature of hot-channels. and hardware-friendly concatenated kernels as shown in Alg. 1.

Mixed-Precision for Post-QK Operations. Our sensitivity analysis in Fig. 32 reveals that "post-QK" operations exhibit fat tails and wide dynamic range, making them particularly hard to quantize. Thus, we extend the mixed-precision approach NVIDIA et al. (2025) by additionally preserving W_o for `LA` and W_{v} for `SA` in BF16.

5 Experiments

Experimental Setup. ***Infrastructure.*** We train using Flame Zhang & Yang (2025) with Fully Sharded Data Parallel (FSDP) Zhao et al. (2023) that enables efficient parameter sharding and activation checkpointing. We employ NVFP4 kernel from TransformerEngine NVIDIA (2023). ***Architecture.*** We evaluate three `LA` architectures: Gated Linear Attention (GLA) Yang et al. (2024), Gated Slot Attention (GSA) Bi et al. (2024), and Gated DeltaNet Schlag et al. (2021). All models are pretrained from scratch at scales from 340M to 7B. ***Evaluation.*** We evaluate zero-shot performance on a standard suite of commonsense reasoning and language understanding benchmarks using lm-harness-evaluation Gao et al. (2021), including ARC-Challenge, ARC-Easy Clark et al. (2018), HellaSwag Zellers et al. (2019), OpenBookQA Mihaylov et al. (2018), PIQA Bisk et al. (2020), SciQ Welbl et al. (2017), and WinoGrande Sakaguchi et al. (2021).

Training Details. Models are pretrained on a RedPajama subset Together AI (2023). We train GLA-1.3B on 60B tokens for Tab. 3 and Fig. 12, and on 40B tokens for Tab. 1. We use the AdamW Loshchilov & Hutter (2019) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, a peak learning rate of 3×10^{-4}, and a cosine decay schedule with 2k-step linear warmup. We employ a global batch size of 4M tokens and a sequence length of 4,096. Training employs FSDP alone, without other parallelism or gradient accumulation. The gradient-norm clipping threshold is 1. Hyperparameters are matched across BF16, FP8, and NVFP4 runs; see App. C.3 for the detailed FP4 training recipe and App. C.1 for additional experimental setups.

Downstream Tasks Evaluation. These validations show CHON achieves competitive quality and meaningful system-level benefits. As shown in Tab. 1, CHON achieves downstream scores nearly on par with BF16 baselines across GSA, GLA, Gated DeltaNet, and Qwen, indicating minimal quality degradation. All `LA` variants remain competitive under NVFP4, supporting the hypothesis that their smoother activation dynamics suit low-precision training. Notably, the scaling evidence at 7B is encouraging: both NVFP4 (61.4%) and CHON (61.3%) surpass FP8 (59.9%) and the BF16 baseline (59.3%) on GLA-7B. This suggests that 4-bit recipes can occasionally match or slightly exceed BF16 on downstream evaluation at larger scales. Refer to App. D.3 for additional architectures and scales.

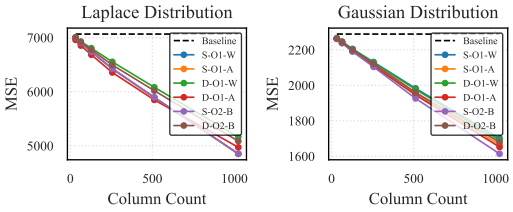


Figure 11: **HCP configuration comparison (S/D, O1/O2, W/A/B)**. Quantization error is evaluated against patched column counts under Laplace and Gaussian priors. `S-O2-B` consistently minimizes MSE relative to the baseline (dashed line).

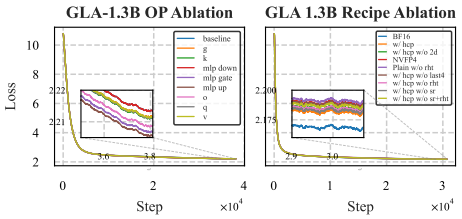


Figure 12: **Ablation of CHON recipe.** Training loss for GLA-1.3B (60B tokens). Integrating HCP, SR, and RHT is essential for minimizing the NVFP4-to-BF16 performance gap.

Table 2: **Sensitivity scores comparison across operators.**

	Attention Operators					MLP Operators		
Model	Attn.q	Attn.k	Attn.v	Attn.o	Attn.g	Mlp.u	Mlp.g	Mlp.d
GLA-1.3B	0.053	0.057	0.025	**0.058**	0.023	0.034	0.029	0.012
Qwen-1.7B	0.022	0.025	**0.143**	0.020	—	0.039	0.014	0.020

Efficiency. We fuse residual compensation and concatenation using `Triton`. Relative to the NVFP4 baseline, CHON requires 14.72% additional GPU-hours for Qwen3-1.7B (175.2 vs. 152.72) and 9.43% for GLA-1.3B (92.88 vs. 84.88). These end-to-end cost ratios are separate from the operation-time fractions reported in Tab. 5; further measurement details appear in App. C.2.

Ablation Study. Across three seeds, CHON lowers GLA-1.3B loss by 0.0087–0.0091 relative to NVFP4 (Tab. 13), consistently improving this setting. **(1) Recipe:** Fig. 12 (right) shows that removing HCP/SR/RHT or scaling worsens FP4 training; HCP gives the largest gain, reducing the NVFP4-to-BF16 loss gap from **0.948%** to **0.591%** (Tab. 3). **(2) Operator:** While `mlp_up`/`mlp_gate` appear dominant in Fig. 12 (left) due to parameter size, the parameter-normalized sensitivity score identifies `o_proj` (GLA) and `v_proj` (Qwen) as most quantization-sensitive (Tab. 2).

Post-training. Appendices D.1 and D.2 report additional NVFP4 experiments on Qwen3-8B: supervised fine-tuning on OpenMathReasoning and GRPO on GSM8K. These experiments do not enable HCP and therefore evaluate the base precision recipe; their full settings and results are retained in the appendix.

Table 3: **GLA-1.3B loss and BF16-relative gap (60B tokens).** Gap (%) $= 100(L - L_{\mathrm{BF16}})/L_{\mathrm{BF16}}$.

Configuration	Final Loss	Loss Gap (%)
BF16 baseline	2.168596	0.000
NVFP4 w/ HCP (full)	2.181415	0.591
NVFP4 w/ HCP w/o sr	2.183048	0.666
NVFP4 w/ HCP w/o rht	2.184524	0.734
NVFP4 w/ HCP w/o 2d scaling	2.184868	0.750
NVFP4 w/ HCP w/o sr, rht	2.187204	0.858
NVFP4 w/ HCP w/o last4	2.188535	0.919
NVFP4 NVIDIA et al. (2025)	2.189147	0.948
NVFP4 w/ HCP w/o chon, rht	2.191353	1.049

6 Conclusion

We present a longitudinal analysis of outlier dynamics in NVFP4 pretraining across Softmax and Linear Attention architectures. We show that Linear Attention reduces tensor-level heavy tails, while post-QK operations remain especially sensitive and outliers gradually concentrate into persistent hot channels. Based on these findings, we propose the CHON recipe, which integrates Hot-Channel Patch (HCP) with targeted post-QK operation protection, reducing the loss gap to the BF16 baseline to 0.59% on GLA-1.3B while maintaining downstream accuracy.

Limitations

Our longitudinal analysis covers selected architectures and horizons up to 60B tokens; persistence over longer training remains untested. The diagnostics provide limited causal identification and do not establish equal-wall-clock efficiency superiority over alternative recipes. Broader generalization requires additional model coverage, longer-horizon measurements, and controlled source-level interventions.

Ethics statement

This work studies numerical precision in language-model training. Its practical benefits depend on accelerator support for the required formats and on systems expertise, which may limit accessibility and reproducibility. The use of web-derived pretraining data also requires attention to licensing and inherited biases. No energy measurements are reported, so environmental benefits remain unquantified.

AI use statement

Large language model (LLM)-based tools were used during the preparation of this manuscript solely to improve the clarity and fluency of the written English. Assistance was limited to sentence-level rephrasing, grammatical refinement, and general prose editing to enhance readability.

No AI tool was used to generate or modify any scientific content, including research ideas, method design, theoretical derivations, mathematical proofs, experimental protocols, or data analysis. All intellectual contributions, scientific claims, and reported results are entirely the authors' own. The authors take full responsibility for the accuracy and integrity of every result and statement in this work.

Reproducibility statement

The training settings and precision recipe are described in Section 5 and Appendix C. Appendix A provides the HCP derivations, and Algorithm 1 describes residual construction and channel selection. Additional reported evaluations are collected in Appendix D.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Systematic outliers in large language models. *ArXiv*, abs/2502.06415, 2025.

Saleh Ashkboos, Ilia Markov, Elias Frantar, Tingxuan Zhong, Xincheng Wang, Jie Ren, Torsten Hoefler, and Dan Alistarh. Quik: Towards end-to-end 4-bit inference on generative large language models, 2023.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

Wei Bi, Leyang Cui, Guohong Fu, Freda Shi, Bailin Wang, Bolun Wang, Yiqiao Wang, Songlin Yang, Yu Zhang, Yue Zhang, Peng Zhou, and Ruijie Zhu. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. In *Advances in Neural Information Processing Systems*, 2024.

Hengjie Cao, Mengyi Chen, Yifeng Yang, Ruijun Huang, Fang Dong, Jixian Zhou, Anrui Chen, Mingzhi Dong, Yujiang Wang, Jinlong Hou, Yuan Cheng, Fan Wu, Fan Yang, Tun Lu, Ning Gu, and Li Shang. Metis: Training llms with fp4 quantization. *arXiv preprint arXiv:2509.00404*, 2025.

Roberto L Castro, Andrei Panferov, Soroush Tabesh, Oliver Sieberling, Jiale Chen, Mahdi Nikdan, Saleh Ashkboos, and Dan Alistarh. Quartet: Native fp4 training can be optimal for large language models. *arXiv preprint arXiv:2505.14669*, 2025.

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

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

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Łukasz Kaiser, David Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. *CoRR*, abs/2009.14794, 2020. URL https://arxiv.org/abs/2009.14794.

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.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

DeepSeek-AI. Deepseek-v3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale, 2022. URL https://arxiv.org/abs/2208.07339.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression, 2023.

Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Hervé Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. In *International Conference on Learning Representations*, 2021.

Maxim Fishman, Brian Chmiel, Ron Banner, and Daniel Soudry. Scaling fp8 training to trillion-token llms. *arXiv preprint arXiv:2409.12517*, 2024.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. https://doi.org/10.5281/zenodo.5371629, 2021.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Bobby He, Lorenzo Noci, Daniele Paliotta, Imanol Schlag, and Thomas Hofmann. Understanding and minimising outlier features in transformer training. In *Advances in Neural Information Processing Systems*, 2024.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: Pushing the limit of post-training quantization for llms. *ArXiv*, abs/2402.04291, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13355–13364, 2024.

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

Yizhou Liu, Ziming Liu, and Jeff Gore. Superposition yields robust neural scaling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2505.10465. Best Paper Runner-up.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

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.

Yury Nahshan, Joseph Kampeas, and Emir Haleva. Linear log-normal attention with unbiased concentration. *arXiv preprint arXiv:2311.13541*, 2023.

NVIDIA. Transformer Engine. https://github.com/NVIDIA/TransformerEngine, 2023. Software library.

NVIDIA. OpenMathReasoning. https://huggingface.co/datasets/nvidia/OpenMathReasoning, 2024. Hugging Face dataset.

NVIDIA, Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, Ian Buck, Bryan Catanzaro, Muya Chang, Jinhang Choi, Mike Chrzanowski, Eric Chung, Victor Cui, Steve Dai, Bita Darvish Rouhani, Carlo del Mundo, Deena Donia, Burc Eryilmaz, Henry Estela, Abhinav Goel, Oleg Goncharov, Yugi Guvvala, Robert Hesse, Russell Hewett, Herbert Hum, Ujval Kapasi, Brucek Khailany, Mikail Khona, Nick Knight, Alex Kondratenko, Ronny Krashinsky, Ben Lanir, Simon Layton, Michael Lightstone, Daniel Lo, Paulius Micikevicius, Asit Mishra, Tim Moon, Deepak Narayanan, Chao Ni, Abhijit Paithankar, Satish Pasumarthi, Ankit Patel, Mostofa Patwary, Ashwin Poojary, Gargi Prasad, Sweta Priyadarshi, Yigong Qin, Xiaowei Ren, Oleg Rybakov, Charbel Sakr, Sanjeev Satheesh, Stas Sergienko, Pasha Shamis, Kirthi Shankar, Nishant Sharma, Mohammad Shoeybi, Michael Siu, Misha Smelyanskiy, Darko Stosic, Dusan Stosic, Bor-Yiing Su, Frank Sun, Nima Tajbakhsh, Shelby Thomas, Przemek Tredak, Evgeny Tsykunov, Gandhi Vaithilingam, Aditya Vavre, Rangharajan Venkatesan, Roger Waleffe, Qiyu Wan, Hexin Wang, Mengdi Wang, Lizzie Wei, Hao Wu, Evan Wu, Keith Wyss, Ning Xu, Jinze Xue, Charlene Yang, Yujia Zhai, Ruoxi Zhang, Jingyang Zhu, and Zhongbo Zhu. Pretraining large language models with nvfp4. *arXiv preprint arXiv:2509.25149*, 2025.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. https://arxiv.org/abs/2508.10925, 2025. Model card.

Jungwoo Park, Taewhoo Lee, Chanwoong Yoon, Hyeon Hwang, and Jaewoo Kang. Outlier-safe pre-training for robust 4-bit quantization of large language models. *arXiv preprint arXiv:2506.19697*, 2025. URL https://arxiv.org/abs/2506.19697.

Yeonhong Park, Jake Hyun, Hojoon Kim, and Jae W. Lee. Decdec: A systems approach to advancing low-bit llm quantization. In *USENIX Symposium on Operating Systems Design and Implementation*, 2024.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Lingpeng Kong, Chris Dyer, and Phil Blunsom. Random feature attention. In *International Conference on Learning Representations*, 2021.

Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, Ruihang Li, Miaosen Zhang, Chen Li, Jia Ning, Ruizhe Wang, Zheng Zhang, Shuguang Liu, Joe Chau, Han Hu, and Peng Cheng. Fp8-lm: Training fp8 large language models. *arXiv preprint arXiv:2310.18313*, 2023.

Zhen Qin, Xiaodong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. The devil in linear transformer. *arXiv preprint arXiv:2210.10340*, 2022a.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*, 2022b.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

Rahul Raman, Khushi Sharma, and Sai Qian Zhang. Rethinking the outlier distribution in large language models: An in-depth study. *arXiv preprint arXiv:2505.21670*, 2025.

Mihaela Claudia Rosca. On discretisation drift and smoothness regularisation in neural network training. *arXiv preprint arXiv:2310.14036*, 2023. URL https://arxiv.org/abs/2310.14036.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Dusan Stosic, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, Levi Melnick, Maral Mesmakhosroshahi, Andres Rodriguez, Michael Schulte, Rasoul Shafipour, Lei Shao, Michael Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verrilli, Ralph Wittig, Doug Burger, and Eric Chung. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

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.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

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

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Seungwoo Son, Wonpyo Park, Woohyun Han, Kyuyeun Kim, and Jaeho Lee. Prefixing attention sinks can mitigate activation outliers for large language model quantization. *arXiv preprint arXiv:2406.12016*, 2024.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

Weixuan Sun, Zhen Qin, Hui Deng, Jianyuan Wang, Yi Zhang, Kaihao Zhang, Nick Barnes, Stan Birchfield, Lingpeng Kong, and Yiran Zhong. Vicinity vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12635–12649, 2023a.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2023b.

Together AI. RedPajama: An open-source recipe to reproduce the LLaMA training dataset. https://github.com/togethercomputer/RedPajama-Data, 2023. GitHub data recipe.

Albert Tseng, Tao Yu, and Youngsuk Park. Training llms with mxfp4. *arXiv preprint arXiv:2502.20586*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Hongyu Wang, Shuming Ma, and Furu Wei. Bitnet a4.8: 4-bit activations for 1-bit llms, 2024.

Ruizhe Wang, Yeyun Gong, Xiao Liu, Guoshuai Zhao, Ziyue Yang, Baining Guo, Zhengjun Zha, and Peng Cheng. Optimizing large language model training using fp4 quantization. *arXiv preprint arXiv:2501.17116*, 2025.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*, 2023.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Peter H. Westfall. Kurtosis as peakedness, 1905–2014. r.i.p. *The American Statistician*, 68(3): 191–195, 2014. ISSN 0003-1305. doi: 10.1080/00031305.2014.917055.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2022. URL https://arxiv.org/abs/2211.10438.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2025b. URL https://arxiv.org/abs/2412.06464.

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.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347*, 2024.

Yu Zhang and Songlin Yang. Flame: Flash language modeling made easy. https://github.com/fla-org/flame, January 2025. GitHub repository.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209, 2024.

Appendix Contents

A Theoretical analysis of Hot-Channel Patch 15
A.1 Hot-channel Formalization 15
A.2 Candidate Compensation Configurations. 16
A.3 Mean-squared Error Bounds. 17
B Ablation Study Details 18
B.1 Details of HCP Configurations 18
B.2 Post-QK Operations Ablation 20
C Additional Experimental Details 21
C.1 Experimental Setup 21
C.2 Efficiency Details 21
C.3 CHON Recipe 23
C.4 NVFP4 Details 24
D Additional Experiments 25
D.1 Supervised Fine-tuning with NVFP4 25
D.2 Reinforcement Learning with NVFP4 26
D.3 Additional Downstream Tasks Performance 27
D.4 Further Validation Experiments 28
E Additional analysis 30
E.1 Kurtosis Analysis 30
E.2 Outlier Evolution Landscape 31
E.3 Top-k Magnitude Analysis during Training 32
E.4 Component Dynamic Analysis 33
E.5 Frobenius Energy Analysis 34
E.6 Flush-to-Zero Dynamic Analysis 34
E.7 Gating as Outlier Source 36
E.8 Normalization Impact Analysis 38
E.9 Superposition Analysis 39
E.10 Quantization Error Dynamics 41
F Discussions 41

A Theoretical Analysis of Hot-Channel Patch

Let $\mathbf{X} \in \mathbb{R}^{n\times t}$ and $\mathbf{W}^\top \in \mathbb{R}^{m\times n}$ be the reference activations and weights, with n input channels, t samples, and m output channels. We have $\mathbf{Y} := \mathbf{W}^\top\mathbf{X} \in \mathbb{R}^{m\times t}$; k denotes the number of selected channels. Let $Q(\cdot)$ and $D(\cdot)$ denote the quantization and dequantization operators, respectively. The low-precision (LP) approximations are defined as $\widehat{\mathbf{X}} := D(Q(\mathbf{X})) = \mathbf{X} + \Delta\mathbf{X}$ and $\widehat{\mathbf{W}} := D(Q(\mathbf{W})) = \mathbf{W} + \Delta\mathbf{W}$, where $\Delta\mathbf{X}$ and $\Delta\mathbf{W}$ denote the quantization residuals.

A.1 Hot-channel formalization

Definition A.1 (Hot-channel index set)**.** To control computational overhead, we select a set of channel indices

$$\mathcal{I} \subset \{1, \ldots, n\}, \qquad |\mathcal{I}| = k, \tag{3}$$

based on the same compounded error score used in Sec. 4:

$$s_j = \frac{1}{t}\left\|\Delta\mathbf{X}_{j,:}\right\|_1 + \frac{1}{m}\left\|\Delta\mathbf{W}_{j,:}\right\|_1, \tag{4}$$

where $\mathcal{I} = \text{top-}k(\mathbf{s})$ contains the indices of the top-k scores.

Context. Motivated by the longitudinal observation that transient drifting outliers gradually stabilize into a small set of persistent *hot channels*, we treat $\mathcal{I}$ as fixed in later stages for multiple steps/intervals to avoid per-step reselection overhead, consistent with online patching in HCP/CHON.

Definition A.2 (Selection operator and residual patch on $\mathcal{I}$)**.** Let $P_{\mathcal{I}}$ select the rows indexed by $\mathcal{I}$ from $\mathbf{W} \in \mathbb{R}^{n\times m}$ and $\mathbf{X} \in \mathbb{R}^{n\times t}$. Thus $\mathbf{W}_{\mathcal{I}} \in \mathbb{R}^{k\times m}$ and $\mathbf{X}_{\mathcal{I}} \in \mathbb{R}^{k\times t}$. Equivalently, it selects the corresponding columns of $\mathbf{W}^\top$. The hot-channel residual patches (quantization error for hot channels) are

$$\Delta\mathbf{W}_{\mathcal{I}} := P_{\mathcal{I}}(\Delta\mathbf{W}), \qquad \Delta\mathbf{X}_{\mathcal{I}} := P_{\mathcal{I}}(\Delta\mathbf{X}). \tag{5}$$

For the compensation formulas below, we also use the equivalent patch-form residuals

$$\mathbf{R}^w_{\mathcal{I}} := \mathbf{W}_{\mathcal{I}} - \widehat{\mathbf{W}}_{\mathcal{I}} = -\Delta\mathbf{W}_{\mathcal{I}}, \qquad \mathbf{R}^x_{\mathcal{I}} := \mathbf{X}_{\mathcal{I}} - \widehat{\mathbf{X}}_{\mathcal{I}} = -\Delta\mathbf{X}_{\mathcal{I}}. \tag{6}$$

Then we have

$$\widehat{\mathbf{W}}_{\mathcal{I}} = \mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}}, \qquad \widehat{\mathbf{X}}_{\mathcal{I}} = \mathbf{X}_{\mathcal{I}} + \Delta\mathbf{X}_{\mathcal{I}}. \tag{7}$$

A.2 Candidate Compensation Configurations

We derive three idealized compensation configurations on the selected channels: (1) *no compensation* (baseline), (2) *single-sided activation compensation* (`O1-A`), and (3) *two-sided weight–activation compensation* (`O2-B`). Here O1/O2 denote the degree of the remaining error in $\Delta\mathbf{W}$ and $\Delta\mathbf{X}$. These identities assume exact arithmetic on the decoded operands and patches; the packed NVFP4 implementation introduces additional errors, described below.

Lemma A.3 (No compensation (baseline))**.** *With no compensation, the decoded low-precision output is*

$$\widehat{\mathbf{Y}}^{(0)}_{\mathcal{I}} = \widehat{\mathbf{W}}^\top_{\mathcal{I}}\widehat{\mathbf{X}}_{\mathcal{I}}. \tag{8}$$

Using the additive residual notation $\widehat{\mathbf{W}}_{\mathcal{I}} = \mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}}$ and $\widehat{\mathbf{X}}_{\mathcal{I}} = \mathbf{X}_{\mathcal{I}} + \Delta\mathbf{X}_{\mathcal{I}}$, we obtain the exact expansion

$$\widehat{\mathbf{Y}}^{(0)}_{\mathcal{I}} = \mathbf{W}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}} + \mathbf{W}^\top_{\mathcal{I}}\Delta\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}^\top_{\mathcal{I}}\Delta\mathbf{X}_{\mathcal{I}}. \tag{9}$$

Proof.

$$\begin{aligned}\widehat{\mathbf{W}}^\top_{\mathcal{I}}\widehat{\mathbf{X}}_{\mathcal{I}} &= (\mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}})^\top(\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{X}_{\mathcal{I}}) \\ &= (\mathbf{W}^\top_{\mathcal{I}} + \Delta\mathbf{W}^\top_{\mathcal{I}})(\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{X}_{\mathcal{I}}) \\ &= \mathbf{W}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}} + \mathbf{W}^\top_{\mathcal{I}}\Delta\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}^\top_{\mathcal{I}}\Delta\mathbf{X}_{\mathcal{I}}.\end{aligned} \tag{10}$$

□

Lemma A.4 (First-order activation compensation (`O1-A`))**.** *Consider the first-order single-sided compensation strategy (`O1-A`), which focuses on addressing activation residuals:*

$$\underbrace{\widehat{\mathbf{W}}^\top_{\mathcal{I}}\widehat{\mathbf{X}}_{\mathcal{I}}}_{\textit{LP term}} + \underbrace{\widehat{\mathbf{W}}^\top_{\mathcal{I}}\mathbf{R}^x_{\mathcal{I}}}_{\textit{activation patch}}. \tag{S0}$$

Then

$$\widehat{\mathbf{W}}^\top_{\mathcal{I}}\widehat{\mathbf{X}}_{\mathcal{I}} + \widehat{\mathbf{W}}^\top_{\mathcal{I}}\mathbf{R}^x_{\mathcal{I}} = \mathbf{W}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}}. \tag{S5}$$

Proof. **Step 1 (compact form; exact algebra).** Since $\mathbf{R}^x_{\mathcal{I}} = \mathbf{X}_{\mathcal{I}} - \widehat{\mathbf{X}}_{\mathcal{I}}$,

$$\widehat{\mathbf{W}}^\top_{\mathcal{I}}\widehat{\mathbf{X}}_{\mathcal{I}} + \widehat{\mathbf{W}}^\top_{\mathcal{I}}\mathbf{R}^x_{\mathcal{I}} = \widehat{\mathbf{W}}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}}. \tag{S1}$$

Step 2 (introduce additive error). Using $\widehat{\mathbf{W}}_{\mathcal{I}} = \mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}}$,

$$\widehat{\mathbf{W}}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}} = (\mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}})^\top\mathbf{X}_{\mathcal{I}}. \tag{S3}$$

Step 3 (full expansion; no omission).

$$\begin{aligned}(\mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}})^\top\mathbf{X}_{\mathcal{I}} &= (\mathbf{W}^\top_{\mathcal{I}} + \Delta\mathbf{W}^\top_{\mathcal{I}})\mathbf{X}_{\mathcal{I}} \\ &= \mathbf{W}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}^\top_{\mathcal{I}}\mathbf{X}_{\mathcal{I}}.\end{aligned} \tag{11}$$

□

Lemma A.5 (Second-order weight–activation compensation (O2−B))**.** *Consider the both-sided second-order compensation strategy (O2−B); in single-kernel form it is realized as one base block plus activation- and weight-patch blocks, matching the three-block packing in Algorithm 1:*

$$\underbrace{\widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\widehat{\mathbf{X}}_{\mathcal{I}}}_{\textit{LP term}} + \underbrace{\widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\mathbf{R}_{\mathcal{I}}^{x}}_{\textit{activation patch}} + \underbrace{(\mathbf{R}_{\mathcal{I}}^{w})^{\top}\widehat{\mathbf{X}}_{\mathcal{I}}}_{\textit{weight patch}}. \tag{D0}$$

Then

$$\widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\widehat{\mathbf{X}}_{\mathcal{I}} + \widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\mathbf{R}_{\mathcal{I}}^{x} + (\mathbf{R}_{\mathcal{I}}^{w})^{\top}\widehat{\mathbf{X}}_{\mathcal{I}} = \mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}} - \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}. \tag{D3}$$

Proof. **Step 1 (compact form + additive errors).** Using $\mathbf{R}_{\mathcal{I}}^{w} = -\Delta\mathbf{W}_{\mathcal{I}}$ and $\mathbf{R}_{\mathcal{I}}^{x} = -\Delta\mathbf{X}_{\mathcal{I}}$, we first rewrite the left-hand side as

$$\begin{aligned}&\widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\widehat{\mathbf{X}}_{\mathcal{I}} + \widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\mathbf{R}_{\mathcal{I}}^{x} + (\mathbf{R}_{\mathcal{I}}^{w})^{\top}\widehat{\mathbf{X}}_{\mathcal{I}}\\ &= (\mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}})^{\top}(\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{X}_{\mathcal{I}}) - (\mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}})^{\top}\Delta\mathbf{X}_{\mathcal{I}} - \Delta\mathbf{W}_{\mathcal{I}}^{\top}(\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{X}_{\mathcal{I}}).\end{aligned} \tag{D1'}$$

Step 2 (full expansion; no omission).

$$\begin{aligned}&(\mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}})^{\top}\mathbf{X}_{\mathcal{I}} + \mathbf{W}_{\mathcal{I}}^{\top}(\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{X}_{\mathcal{I}}) - (\mathbf{W}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}})^{\top}(\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{X}_{\mathcal{I}})\\ &= \mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}} - \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}.\end{aligned} \tag{12}$$

This identity concerns only the selected-channel contribution. Quantization errors on the unselected channels remain present in the full product. □

A.3 Mean-squared Error Bounds

Definition A.6 (Mean-squared error (MSE))**.** For any estimator $\widehat{\mathbf{Y}}_{\mathcal{I}}$ of $\mathbf{Y}_{\mathcal{I}} = \mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}}$, define

$$\mathrm{MSE} := \mathbb{E}\big[\|\mathbf{Y}_{\mathcal{I}} - \widehat{\mathbf{Y}}_{\mathcal{I}}\|_F^2\big], \tag{13}$$

where the expectation can be over data, quantization noise, or both. This is the total mean-squared error; dividing by mt gives the per-output-element MSE. In the bounds below, every square of a matrix-valued error denotes its squared Frobenius norm, and all unqualified norms are Frobenius norms.

Lemma A.7 (MSE upper bound: Baseline (no compensation))**.** *Let* $e_0 := \mathbf{Y}_{\mathcal{I}} - \widehat{\mathbf{Y}}_{\mathcal{I}}^{(0)}$. *Then*

$$e_0 = -\big(\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}\big), \tag{14}$$

and

$$\mathrm{MSE}_0 \le 3\,\mathbb{E}\big[\|\mathbf{W}_{\mathcal{I}}\|^2\|\Delta\mathbf{X}_{\mathcal{I}}\|^2 + \|\mathbf{X}_{\mathcal{I}}\|^2\|\Delta\mathbf{W}_{\mathcal{I}}\|^2 + \|\Delta\mathbf{W}_{\mathcal{I}}\|^2\|\Delta\mathbf{X}_{\mathcal{I}}\|^2\big]. \tag{B0}$$

Proof. Let $a = \mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}$, $b = \Delta\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}}$, $c = \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}$, so $e_0 = -(a+b+c)$. Using $(u+v+w)^2 \le 3(u^2+v^2+w^2)$,

$$\mathbb{E}[e_0^2] \le 3(\mathbb{E}[a^2] + \mathbb{E}[b^2] + \mathbb{E}[c^2]). \tag{15}$$

By Cauchy–Schwarz,

$$a^2 \le \|\mathbf{W}_{\mathcal{I}}\|^2\|\Delta\mathbf{X}_{\mathcal{I}}\|^2, \quad b^2 \le \|\Delta\mathbf{W}_{\mathcal{I}}\|^2\|\mathbf{X}_{\mathcal{I}}\|^2, \quad c^2 \le \|\Delta\mathbf{W}_{\mathcal{I}}\|^2\|\Delta\mathbf{X}_{\mathcal{I}}\|^2. \tag{16}$$

Taking expectations yields (B0). □

Lemma A.8 (MSE upper bound: 1st-Order)**.** *Let* $e_1 := \mathbf{Y}_{\mathcal{I}} - \widehat{\mathbf{Y}}_{\mathcal{I}}^{(1)}$. *Then*

$$e_1 = -\Delta\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}}, \tag{17}$$

and

$$\mathrm{MSE}_1 \le \mathbb{E}\big[\|\Delta\mathbf{W}_{\mathcal{I}}\|^2\|\mathbf{X}_{\mathcal{I}}\|^2\big]. \tag{B1}$$

Proof. From Eq. (S0), $\widehat{\mathbf{Y}}_{\mathcal{I}}^{(1)} = \mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}}$, hence $e_1 = -\Delta\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}}$. By Cauchy–Schwarz,

$$e_1^2 = (\Delta\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}})^2 \le \|\Delta\mathbf{W}_{\mathcal{I}}\|^2\|\mathbf{X}_{\mathcal{I}}\|^2. \tag{18}$$

Taking expectations gives (B1). □

Lemma A.9 (MSE upper bound: 2nd-Order)**.** *Let* $e_2 := \mathbf{Y}_{\mathcal{I}} - \widehat{\mathbf{Y}}_{\mathcal{I}}^{(2)}$. *Then*

$$e_2 = \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}, \tag{19}$$

and

$$\mathrm{MSE}_2 \le \mathbb{E}\big[\|\Delta\mathbf{W}_{\mathcal{I}}\|^2\|\Delta\mathbf{X}_{\mathcal{I}}\|^2\big]. \tag{B2}$$

Proof. From Eq. (D0), $\widehat{\mathbf{Y}}_{\mathcal{I}}^{(2)} = \mathbf{Y}_{\mathcal{I}} - \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}$, so $e_2 = \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}$. By Cauchy–Schwarz,

$$e_2^2 = (\Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}})^2 \le \|\Delta\mathbf{W}_{\mathcal{I}}\|^2\|\Delta\mathbf{X}_{\mathcal{I}}\|^2. \tag{20}$$

Taking expectations yields (B2). □

Comparison of upper bounds. If $\|\Delta\mathbf{X}_{\mathcal{I}}\|_F \le \epsilon\|\mathbf{X}_{\mathcal{I}}\|_F$ almost surely, the bound in (B2) is at most ϵ^2 times the bound in (B1). This compares upper bounds only; it does not order the actual MSEs, since $\Delta\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}}$ can be small through directional cancellation.

Theorem A.10 (Conditional error comparison)**.** *Let* $a = \mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}$, $b = \Delta\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}}$, *and* $c = \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}$ *have finite second moments. If*

$$\mathbb{E}\langle a, b\rangle_F = \mathbb{E}\langle a, c\rangle_F = \mathbb{E}\langle b, c\rangle_F = 0, \tag{21}$$

then $\mathrm{MSE}_0 = \mathbb{E}\|a\|_F^2 + \mathrm{MSE}_1 + \mathrm{MSE}_2$, *so* $\mathrm{MSE}_1 \le \mathrm{MSE}_0$ *and* $\mathrm{MSE}_2 \le \mathrm{MSE}_0$. *A comparison between* MSE_1 *and* MSE_2 *requires further assumptions.*

Proof. Expand $\mathbb{E}\|a+b+c\|_F^2$ and apply Eq. (21). The remaining squared norms are nonnegative. □

The orthogonality conditions above are sufficient assumptions for the idealized comparison; they are not established by our training experiments or by the small-noise condition alone.

A.4 Packed implementation and full-product error

Let $\mathcal{J}$ be the unselected channels. With exact patch arithmetic, the full idealized product is

$$\mathbf{Y}^{\mathrm{ideal}} = \widehat{\mathbf{W}}^{\top}\widehat{\mathbf{X}} + \widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\mathbf{R}_{\mathcal{I}}^{x} + (\mathbf{R}_{\mathcal{I}}^{w})^{\top}\widehat{\mathbf{X}}_{\mathcal{I}}. \tag{22}$$

Therefore its error relative to $\mathbf{Y} = \mathbf{W}^{\top}\mathbf{X}$ is

$$\mathbf{Y}^{\mathrm{ideal}} - \mathbf{Y} = \mathbf{W}_{\mathcal{J}}^{\top}\Delta\mathbf{X}_{\mathcal{J}} + \Delta\mathbf{W}_{\mathcal{J}}^{\top}\mathbf{X}_{\mathcal{J}} + \Delta\mathbf{W}_{\mathcal{J}}^{\top}\Delta\mathbf{X}_{\mathcal{J}} - \Delta\mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}. \tag{23}$$

For S-mode, define the ideal decoded packing $\mathbf{A} = [\widehat{\mathbf{W}}; \mathbf{R}_{\mathcal{I}}^{w}; \widehat{\mathbf{W}}_{\mathcal{I}}]$ and $\mathbf{B} = [\widehat{\mathbf{X}}; \widehat{\mathbf{X}}_{\mathcal{I}}; \mathbf{R}_{\mathcal{I}}^{x}]$, so $\mathbf{A}^{\top}\mathbf{B} = \mathbf{Y}^{\mathrm{ideal}}$. Algorithm 1 packs the original $\mathbf{W}$ and $\mathbf{X}$ in the base blocks, then the consumer quantizes the expanded tensors before GEMM. Let $\mathbf{A}+\mathbf{E}_A$ and $\mathbf{B}+\mathbf{E}_B$ denote the resulting decoded packed operands. These error terms include patch casting and quantization, as well as any change in base-block quantization caused by packing. If $\mathbf{E}_{\mathrm{gemm}}$ denotes accumulation and output-rounding error, then exactly

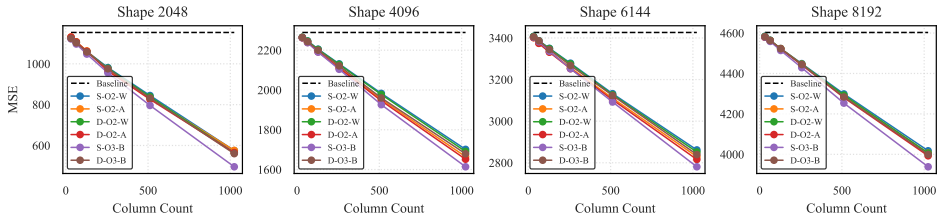
$$\mathbf{Y}^{\mathrm{exec}} - \mathbf{Y} = (\mathbf{Y}^{\mathrm{ideal}} - \mathbf{Y}) + \mathbf{A}^{\top}\mathbf{E}_B + \mathbf{E}_A^{\top}\mathbf{B} + \mathbf{E}_A^{\top}\mathbf{E}_B + \mathbf{E}_{\mathrm{gemm}}. \tag{24}$$

Thus the second-order selected-channel identity does not by itself bound the complete executed GEMM error by a second-order term.

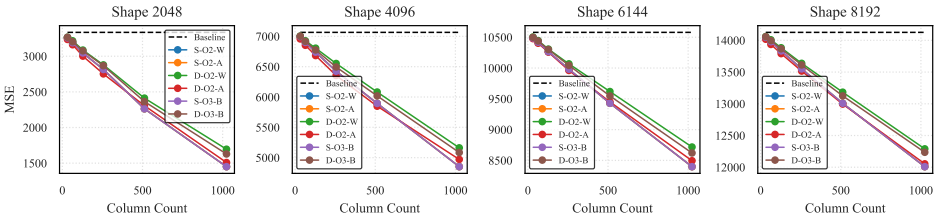
B Ablation Study Details

B.1 HCP Configuration Details

HCP mitigates quantization errors in FP4 matrix multiplications. Identity 4.1 expands the decoded product around $\mathbf{W}^{\top}\mathbf{X}$ into first-order errors $\mathbf{W}^{\top}\Delta\mathbf{X} + \Delta\mathbf{W}^{\top}\mathbf{X}$ and the second-order interaction $\Delta\mathbf{W}^{\top}\Delta\mathbf{X}$. Throughout, $\Delta\mathbf{W} = \widehat{\mathbf{W}} - \mathbf{W}$ and $\Delta\mathbf{X} = \widehat{\mathbf{X}} - \mathbf{X}$; compensation uses $\mathbf{R}^{w} = -\Delta\mathbf{W}$ and $\mathbf{R}^{x} = -\Delta\mathbf{X}$. The configuration labels below describe idealized patch algebra on selected channels.



(a) Gaussian Activation Prior



(b) Laplace Activation Prior

Figure 13: **Comparison of HCP configurations under different activation priors.** Mean-squared error (MSE) of the quantized linear product versus the number of patched (hot) channels for four hidden sizes (2,048/4,096/6,144/8,192), evaluated under (a) Gaussian and (b) Laplace activation priors. Curves compare Single- vs Dual-kernel implementations (S/D), single-sided vs two-sided recovery (O1/O2), and patch targets (weights-only W, activations-only A, or both B); the dashed line denotes the unpatched baseline. Consistent with the paper's CHON design choice, **S-O2-B** (single-kernel, second-order, both) yields the lowest MSE across shapes and priors.

Configuration Dimension 1: Mode (Execution Strategy). The *Mode* parameter defines the computational implementation of the residual compensation:

- **Single-Kernel Mode (S):** This strategy implements error recovery via matrix concatenation. The exact packing depends on the chosen Order/Target variant: single-sided first-order variants use a two-block packing, whereas the CHON configuration `S-O2-B` uses three aligned blocks, consistent with Algorithm 1. In particular, the fused kernel packs

$$\begin{aligned} W_{\text{out}} &= \text{Concat}\big([W,\ R_W^{\text{comp}},\ W_{dq}^{\text{comp}}], \text{dim} = 1\big), \\ X_{\text{out}} &= \text{Concat}\big([X,\ X_{dq}^{\text{comp}},\ R_X^{\text{comp}}], \text{dim} = 1\big). \end{aligned} \tag{25}$$

  Here the implementation stores weights as $W \in \mathbb{R}^{m\times n}$ and inputs as $X \in \mathbb{R}^{t\times n}$, transposed relative to the analysis convention. Both packed tensors are quantized before a single NVFP4 GEMM. The idealized identities below exclude this additional quantization and GEMM rounding; their relation to the executed product is given in App. A.4.

- **Dual-Kernel Mode (D):** The decoded base product $\mathbf{Y}_{\text{base}} = \widehat{\mathbf{W}}^\top \widehat{\mathbf{X}}$ and the selected patch are evaluated separately and accumulated. For the idealized two-sided configuration,

$$\mathbf{Y}_{\text{res}} = \widehat{\mathbf{W}}_{\mathcal{I}}^\top \mathbf{R}_{\mathcal{I}}^x + (\mathbf{R}_{\mathcal{I}}^w)^\top \widehat{\mathbf{X}}_{\mathcal{I}}, \qquad \mathbf{Y}_{\text{HCP}} = \mathbf{Y}_{\text{base}} + \mathbf{Y}_{\text{res}}. \tag{26}$$

  Single-sided configurations retain only the corresponding patch term. These expressions specify the algebraic configurations; arithmetic precision determines additional execution error.

Configuration Dimension 2: Order (Error Expansion). The *Order* label denotes the degree of the residual error in the idealized selected-channel product:

- **Second-Order Recovery ($O2$):** Two-sided patches cancel the first-order errors, leaving a second-order interaction (App. A.2, Eq. (D0)):

$$\begin{aligned} \mathbf{Y}_{\text{O2-B}}^{\text{ideal}} &= \widehat{\mathbf{W}}_{\mathcal{I}}^\top \widehat{\mathbf{X}}_{\mathcal{I}} + \widehat{\mathbf{W}}_{\mathcal{I}}^\top \mathbf{R}_{\mathcal{I}}^x + (\mathbf{R}_{\mathcal{I}}^w)^\top \widehat{\mathbf{X}}_{\mathcal{I}} \\ &= \mathbf{W}_{\mathcal{I}}^\top \mathbf{X}_{\mathcal{I}} - \Delta\mathbf{W}_{\mathcal{I}}^\top \Delta\mathbf{X}_{\mathcal{I}}. \end{aligned} \tag{27}$$

Table 4: **Taxonomy of Hot-Channel Patch (HCP) configurations.** The design space is categorized across three axes: Mode (Single (S) for concatenated kernels vs. Dual (D) for separate kernels), Order (O2 for a remaining second-order interaction; O1 for a remaining first-order error in the idealized selected-channel product), and Target (Weight (W), Activation (A), or Both (B)).

Configuration	Mode	Order	Target
`S-O1-W`	Single	1st-Order	Weight
`S-O1-A`	Single	1st-Order	Activation
`D-O1-W`	Dual	1st-Order	Weight
`D-O1-A`	Dual	1st-Order	Activation
`S-O2-B`	Single	2nd-Order	Both
`D-O2-B`	Dual	2nd-Order	Both

- **First-Order Recovery ($O1$):** A single-sided patch leaves the error due to the other operand (Eq. (S0)):

$$\begin{aligned} \mathbf{Y}^{\text{ideal}}_{\text{O1-A}} &= \widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\widehat{\mathbf{X}}_{\mathcal{I}} + \widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\mathbf{R}_{\mathcal{I}}^{x} \\ &= \mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}} + \Delta\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}}, \\ \mathbf{Y}^{\text{ideal}}_{\text{O1-W}} &= \widehat{\mathbf{W}}_{\mathcal{I}}^{\top}\widehat{\mathbf{X}}_{\mathcal{I}} + (\mathbf{R}_{\mathcal{I}}^{w})^{\top}\widehat{\mathbf{X}}_{\mathcal{I}} \\ &= \mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X}_{\mathcal{I}} + \mathbf{W}_{\mathcal{I}}^{\top}\Delta\mathbf{X}_{\mathcal{I}}. \end{aligned} \tag{28}$$

Configuration Dimension 3: Target (Compensation Scope). The *Target* parameter selects the specific residual tensors to be patched:

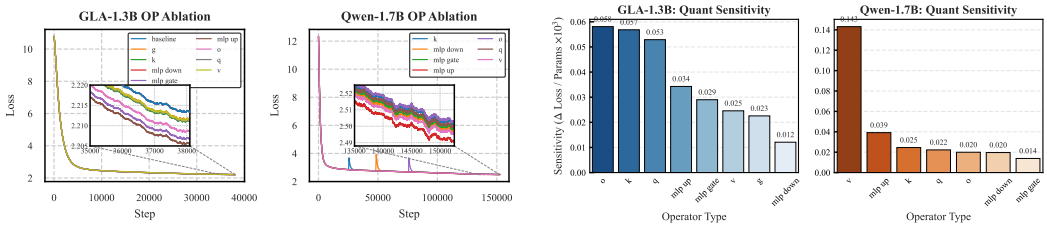
- **Weight-Only (W):** Only weight residuals are patched. Activations remain quantized, so $\Delta\mathbf{X}_{\mathcal{I}}$ generally remains nonzero. Weight residuals depend on the current weights and quantization scales.
- **Activation-Only (A):** Weights remain quantized, patching only activation residuals. This effectively mitigates dynamic outliers (e.g., in query projections) but neglects static weight pathologies.
- **Both (B):** Recovers residuals from both operands. Our ablation study in Fig. 32 demonstrates that bidirectional recovery reduces MSE by 30–45% relative to unilateral approaches, particularly in layers exhibiting simultaneous weight and activation outliers.

Nomenclature and Taxonomy. We employ a compact notation formatted as `Mode-Order-Target` (e.g., `S-O1-A`, `D-O2-B`) to uniquely identify configurations. Tab. 4 delineates the representative design space explored in this work.

B.2 Post-QK Operations Ablation Study

To identify which components are most critical for high-precision protection, we conduct a systematic ablation study. As illustrated in Fig. 14a, a surface-level observation of the loss curves suggests that the `up_proj` has the most significant impact on model convergence in both GLA and Qwen. However, attributing sensitivity solely to absolute loss increase is potentially misleading, as `up_proj` typically accounts for a disproportionately larger share of the total parameters.

To decouple the intrinsic numerical sensitivity from the parameter scale, we introduce a *quantization sensitivity score*, defined as the ΔLoss per unit of parameters relative to the BF16 baseline. As revealed in Fig. 14b, this normalized metric provides a more precise diagnostic: the O projection in GLA and the V projection in Qwen-1.7B emerge as the most quantization-critical components.



(a) Operation Ablation Study (b) Quantization Sensitivity Analysis

Figure 14: **Comprehensive analysis of operator importance and quantization sensitivity.** (a) Training loss curves for operation-level ablations in GLA-1.3B and Qwen-1.7B. While the `up_proj` shows a high absolute loss impact, this is largely attributed to its disproportionate parameter scale. (b) Quantization sensitivity scores normalized by parameter count (ΔLoss/Params). The analysis identifies the GK and O projections in GLA, and the V projection in Qwen-1.7B, as the most critical bottlenecks.

C Additional Experimental Details

C.1 Experimental Setup

Datasets and Preprocessing. We pretrain on a 60B-token subset of RedPajama Together AI (2023). We remove documents flagged as toxic or low-quality by a lightweight classifier. The final mixture is identical across all baselines. We use a 32k vocabulary trained with SentencePiece. All models share the same tokenizer and special tokens. Sequences are packed to 4,096 tokens without cross-document contamination.

Optimization and Schedule. Unless otherwise noted, we train with AdamW (β_1=0.9, β_2=0.95, weight decay 0.1), cosine decay with 2k-step warmup, and gradient clipping at 1.0. For HCP, we designate 9.09% of channels as hot-channels for compensation and refresh the hot-channel indices every 1,000 training steps. We use FSDP with activation checkpointing. The sequence length is 4096 and the global batch size is 4M tokens. Training employs the AdamW optimizer ($\beta_1 = 0.9, \beta_2 = 0.95$) with a peak learning rate of 3×10^{-4}, 2,000 warmup steps, weight decay 0.1, and gradient clipping at 1.0. We use FSDP only (no tensor or pipeline parallelism) and gradient accumulation is 1.

Hardware. All experiments were conducted on a server equipped with 8 NVIDIA Blackwell GPUs. Unless otherwise noted, we report per-GPU throughput and memory usage, with the model-parallel configuration held fixed across all runs. The software stack consists of Python 3.12.3, PyTorch 2.8.0, CUDA runtime 13.0, and cuDNN 9.12.

C.2 Efficiency Details

We detail the implementation and efficiency of the Hot-Channel Patch (HCP) method. Algorithm 1 uses the implementation layout: $M = m$, $N = n$, $K = t$, and $c = k$ relative to the analysis notation. Its returned packed tensors are quantized to NVFP4 before GEMM; see App. A.4 for the resulting error terms.

Algorithm Comparison. The two algorithms presented in Algorithm 1 outline the compensation process. In our implementation, importance scores are computed from quantization residuals to identify the top-k hot channels every 1,000 training steps. We use a fixed refresh interval of 1,000 training steps as a practical trade-off between the overhead of channel scoring and selection and responsiveness to changes in hot-channel locations. Between these refreshes, the *Pre-computed Indices* version (Right) takes a cached set of indices as input, bypassing the *Scoring & Selection* stage. This modification reduces computational cost significantly. The validity of saving index calculation stems from the observation that outlier channels in LLMs exhibit distinct persistence; thus, indices computed at one refresh remain effective for subsequent steps, allowing us to cache and reuse them.

Efficiency and Kernel Fusion. Tab. 5 analyzes the runtime overhead. The "Pre-fuse" columns show that executing operations such as dequantization, gathering, and concatenation individually incurs

Algorithm 1 Hot-Channel Patch Algorithm. (Left: Normal Process, Right: Use pre-computed indices.)

Require: Weight matrix $W \in \mathbb{R}^{M\times N}$, input matrix $X \in \mathbb{R}^{K\times N}$
Require: Number of compensation columns c
Ensure: Expanded matrices $W_{\text{out}}, X_{\text{out}}$ for GEMM
// 1. Quantization & Dequantization
$W_q \leftarrow \text{Quantize}(W, \text{NVFP4})$
$X_q \leftarrow \text{Quantize}(X, \text{NVFP4})$
$W_{dq} \leftarrow \text{Dequantize}(W_q)$
$X_{dq} \leftarrow \text{Dequantize}(X_q)$
// 2. Residual Computation
$R_W \leftarrow W - W_{dq}$ {quantization error for weights}
$R_X \leftarrow X - X_{dq}$ {quantization error for inputs}
// 3. Scoring & Selection (Top-k)
$S_W \leftarrow \frac{1}{M}\sum_{i=1}^{M} |R_W[i,:]|$ {column-wise mean absolute residual}
$S_X \leftarrow \frac{1}{K}\sum_{j=1}^{K} |R_X[j,:]|$ {column-wise mean absolute residual}
$S \leftarrow S_W + S_X$ {importance score}
$\text{idx} \leftarrow \text{TopK}(S, k=c)$ {top c hot channels}
// 4. Gather
$R_W^{\text{comp}} \leftarrow R_W[:, \text{idx}]$ {high-error residuals (weights)}
$W_{dq}^{\text{comp}} \leftarrow W_{dq}[:, \text{idx}]$ {quantized values (weights)}
$R_X^{\text{comp}} \leftarrow R_X[:, \text{idx}]$ {high-error residuals (inputs)}
$X_{dq}^{\text{comp}} \leftarrow X_{dq}[:, \text{idx}]$ {quantized values (inputs)}
// 5. Concat
$W_{\text{out}} \leftarrow \text{Concat}\big([W,\ R_W^{\text{comp}},\ W_{dq}^{\text{comp}}], \text{dim}=1\big)$
$X_{\text{out}} \leftarrow \text{Concat}\big([X,\ X_{dq}^{\text{comp}},\ R_X^{\text{comp}}], \text{dim}=1\big)$ {order may vary for alignment}
return $W_{\text{out}}, X_{\text{out}}$

Require: Weight matrix $W \in \mathbb{R}^{M\times N}$, input matrix $X \in \mathbb{R}^{K\times N}$
Require: Pre-computed indices $\text{idx} \in \mathbb{R}^{c}$ of hot channels
Ensure: Expanded matrices $W_{\text{out}}, X_{\text{out}}$ for GEMM
// 1. Quantization & Dequantization
$W_q \leftarrow \text{Quantize}(W, \text{dtype=FP4})$
$X_q \leftarrow \text{Quantize}(X, \text{dtype=FP4})$
$W_{dq} \leftarrow \text{Dequantize}(W_q)$
$X_{dq} \leftarrow \text{Dequantize}(X_q)$
// 2. Residual Computation
$R_W \leftarrow W - W_{dq}$ {quantization error for weights}
$R_X \leftarrow X - X_{dq}$ {quantization error for inputs}
// 3. Gather
// Look up residual and quantized values for the specified columns
$R_W^{\text{comp}} \leftarrow R_W[:, \text{idx}]$
$W_{dq}^{\text{comp}} \leftarrow W_{dq}[:, \text{idx}]$
$R_X^{\text{comp}} \leftarrow R_X[:, \text{idx}]$
$X_{dq}^{\text{comp}} \leftarrow X_{dq}[:, \text{idx}]$
// 4. Concat
$W_{\text{out}} \leftarrow \text{Concat}\big([W,\ R_W^{\text{comp}},\ W_{dq}^{\text{comp}}], \text{dim}=1\big)$
$X_{\text{out}} \leftarrow \text{Concat}\big([X,\ X_{dq}^{\text{comp}},\ R_X^{\text{comp}}], \text{dim}=1\big)$
return $W_{\text{out}}, X_{\text{out}}$

Table 5: Fine-grained performance breakdown and fusion efficiency analysis of CHON. All time metrics are in milliseconds (ms).

Shape (W × X)	Forward	Backward Pass		CHON Operations (Pre-fuse)					Fused	Overhead Ratio (%)†	
	Fprop	DGrad	WGrad	Deq.	Gthr.	Resid.	Cat.	**Sum**	**Kernel**	**Pre-fuse**	**Post-fuse**
2048×2048	0.239	0.368	0.143	0.049	0.059	0.030	0.046	0.184	0.044	14.76%	**3.97%**
1024×2048	0.235	0.238	0.142	0.046	0.062	0.030	0.046	0.184	0.044	16.56%	**4.53%**
6144×2048	0.237	0.236	0.143	0.045	0.060	0.030	0.046	0.181	0.075	16.79%	**7.72%**
2048×6144	0.235	0.238	0.143	0.046	0.060	0.030	0.046	0.182	0.047	16.47%	**4.85%**
Mean	**0.237**	**0.270**	**0.143**	0.047	0.060	0.030	0.046	**0.183**	**0.053**	**16.15%**	**5.27%**

† **Pre-fuse %** = $\frac{\text{Sum}}{\text{Step}-\text{Fused}+\text{Sum}}$; **Post-fuse %** = $\frac{\text{Fused_Kernel}}{\text{Step}}$ (calculated based on end-to-end iteration time).

high overhead ($\sim$16%). To mitigate this, we employ a `Triton` kernel to fuse residual compensation and concatenation into a single operation. This fusion strategy drastically lowers the overhead to 5.27%, as evidenced in the "Post-fused" column, demonstrating the practical efficiency of our design.

End-to-End Measurement. We evaluate the end-to-end training performance of Qwen3-1.7B on 8 GPUs. CHON requires 175.2 GPU-hours, compared with 152.72 GPU-hours for the baseline NVFP4 recipe NVIDIA et al. (2025), corresponding to 14.72% additional GPU-hours. For GLA-1.3B, CHON requires 92.88 GPU-hours compared with 84.88 GPU-hours for NVFP4, corresponding to 9.43% additional GPU-hours. Both percentages use $(C_{\text{CHON}}/C_{\text{NVFP4}} - 1) \times 100\%$, where C is total GPU-hours.

C.3 CHON RECIPE

We investigate the use of NVFP4 for training Linear Attention (focusing on GLA) under a 4-bit floating-point regime. To obtain stable end-to-end FP4 optimization, we incorporate the following techniques:

① *Precision partitioning*: retain higher precision for sensitive components (embeddings, output heads, normalization layers, and Attention kernel). ② *Outlier suppression*: apply random Hadamard transforms in the backward pass only, avoiding inference overhead while diffusing sparse large-magnitude directions. ③ *Unbiased gradient quantization*: employ stochastic rounding during the backward pass to reduce systematic drift. ④ *Asymmetric granularity*: use 1D (per-channel) quantization in the forward path and 2D (tile-wise) quantization in the backward path to balance efficiency and robustness. ⑤ *Recover-block mechanism*: introduce an NVFP4-specific rehydration step for weight regions exhibiting transient outliers, mitigating irreversible clipping during accumulation. Together these measures enable stable NVFP4 training of GLA without material degradation in convergence or downstream performance.

NVFP4 pairs a compact FP4 format with two-stage MicroScaling to expand effective dynamic range while keeping arithmetic low-bit. We summarize the forward and backward data flow below.

Forward (RTN) and Backward (SR). The forward path typically uses round-to-nearest (RTN) quantization for deterministic kernels; the backward path employs stochastic rounding (SR) to reduce bias accumulation in gradients. For weight gradients, Randomized Hadamard Transform (RHT) scrambles inputs to diffuse sparse large-magnitude directions, stabilizing variance under FP4.

Sensitive Ops in higher precision. Embeddings, output heads, normalization layers, nonlinearities, and QK GEMMs are commonly executed in BF16/FP16. Late layers can also be exempted to avoid end-of-training drift.

Mixed Precision Most GEMM operations within the Feedforward network (FFN) are executed in FP4. Each linear layer entails three GEMMs: one for the forward pass (Fprop) and two for the backward pass—one for activation gradients (Dgrad) and one for weight gradients (Wgrad). For ablations, we employ fake quantization: tensors are quantized to NVFP4 but the GEMMs themselves are performed in BF16. This isolates the effect of quantization from kernel-specific implementation details.

$$\text{Fprop:} \quad Y = XW, \qquad Y \in \mathbb{R}^{B\times O},\ X \in \mathbb{R}^{B\times I},\ W \in \mathbb{R}^{I\times O} \tag{29}$$

$$\text{Dgrad:} \quad \mathrm{d}X = \mathrm{d}Y\, W^{\top} \tag{30}$$

$$\text{Wgrad:} \quad \mathrm{d}W = X^{\top}\mathrm{d}Y \tag{31}$$

Randomized Hadamard Transform Although Linear Attention exhibits markedly fewer outliers than Softmax Attention, we observe residual, localized outliers, predominantly in the gating linear modules. To mitigate their impact during training without incurring inference overhead, we apply a Randomized Hadamard Transform (RHT) only in the backward pass. Using RHT in the forward pass would necessitate mirroring the transform at inference time, introducing nontrivial latency and complexity. Following the NVIDIA FP4 recipe NVIDIA et al. (2025), we restrict RHT to the Wgrad GEMM and use orthogonal Walsh–Hadamard transforms with random sign flips to diffuse sparse large-magnitude components, improving robustness under FP4.

- **Forward/Backward Precision:** Forward compute uses NVFP4 kernels end-to-end. The backward pass employs stochastic rounding to prevent cumulative error drift, combined with a random Hadamard transform to reduce activation outliers.
- **Quantization Granularity:** We use 1D quantization in the forward path and 2D quantization in the backward path to balance efficiency and stability.
- **Layer Selection (Mixed Precision):** We do not quantize the Linear Attention module itself; the MLP is quantized. Sensitive layers such as output heads and normalization layers (e.g., RMSNorm) remain in higher precision. Overall this constitutes a mixed-precision quantization strategy.

- **Model Coverage:** Experiments are conducted on Mamba2 Dao & Gu (2024), Gated Linear Attention (GLA) Yang et al. (2024), and Gated DeltaNet Yang et al. (2025b). To our knowledge, we are the first organization to demonstrate the practical feasibility of end-to-end FP4 training on purely linear models.
- **Training Accuracy:** The final training achieves a loss error within 0.8%.

This NVFP4-centric approach creates a numerically stable environment where the benefits of FP4 compute can be realized without sacrificing model quality.

C.4 Details of NVFP4

This section describes the scaling rules used to convert an input tensor to FP4 while preserving dynamic range at both the tensor and block granularity. We use a *global* (tensor-level) scale to align the overall magnitude to the representable range of FP8 scale storage, and *local* (block-level) scales to normalize each block to the FP4 range.

Notation. Let $x \in \mathbb{R}^m$ be a tensor (flattened for convenience). We partition indices into disjoint blocks b (e.g., contiguous groups of elements). For a block b, denote the blockwise absolute maximum by

$$\mathrm{amax}_b \triangleq \max_{i \in b} |x_i|, \qquad \mathrm{amax}_x \triangleq \max_i |x_i|. \tag{32}$$

We write $\mathrm{e4m3}(\cdot)$ for quantization to the FP8 E4M3 format (used to store certain scale factors), and let $q(\cdot)$ denote the FP4 quantization function applied to values after scaling.

Definition C.1 (Global encode / decode scale)**.** The *global encode scale* s_{enc} is defined by

$$s_{\mathrm{enc}} \;=\; \frac{6 \cdot 448}{\mathrm{amax}_x}. \tag{33}$$

The corresponding *global decode scale* is

$$s_{\mathrm{dec}} \;=\; \frac{1}{s_{\mathrm{enc}}}. \tag{34}$$

Remark C.2 (Why the constants 6 and 448?). Here 6 and 448 are the maximum representable magnitudes in FP4 E2M1 and FP8 E4M3, respectively. Intuitively, s_{enc} maps the tensor-level maximum amax_x into the product of the two maxima, so that subsequent blockwise scaling (targeting FP4) can be stored (in FP8) without overflow.

Implementation note (memory traffic). Computing amax_x requires a pass over the tensor. Applying the global scaling prior to FP4 conversion may introduce an additional pass (read for amax_x, then read again for scaling/conversion). In practice, one may choose smaller granularity for "global" scaling (e.g., per-row) to reduce round-trips through device memory, depending on the kernel and layout.

Definition C.3 (Local (block) decode scale)**.** For each block b, the *local decode scale* is chosen so that the largest magnitude in the block maps to the FP4 maximum:

$$s_{\mathrm{dec},b} \;=\; \frac{\mathrm{amax}_b}{6}. \tag{35}$$

Storing local scales in FP8. Local decode scales must be stored in FP8 (E4M3) for Tensor Core execution. To ensure these scales are representable in FP8, we first multiply them by the global encode scale and then quantize:

$$s_{\mathrm{dec},b}^{\mathrm{e4m3}} \;=\; \mathrm{e4m3}\big(s_{\mathrm{dec},b} \cdot s_{\mathrm{enc}}\big). \tag{36}$$

The role of s_{enc} is to remap the largest local decode scale $\max_b s_{\mathrm{dec},b} = \mathrm{amax}_x \,/6$ into the FP8 range (whose maximum is 448), hence avoiding saturation when storing $s_{\mathrm{dec},b}$ in FP8.

Remark C.4 (Recovering a usable local encode scale). Quantizing $s_{\mathrm{dec},b} \cdot s_{\mathrm{enc}}$ in FP8 introduces rounding error. A practical way to obtain the effective *local encode scale* is to invert the quantized value in higher precision and rescale back:

$$s_{\mathrm{enc},b} \;\approx\; \frac{1}{\mathrm{fp32}\Big(s_{\mathrm{dec},b}^{\mathrm{e4m3}}\Big) \cdot s_{\mathrm{dec}}}. \tag{37}$$

This construction aims to keep $s_{\mathrm{enc},b} \cdot s_{\mathrm{dec}} \cdot \mathrm{fp32}(s_{\mathrm{dec},b}^{\mathrm{e4m3}}) \approx 1$, so that the overall scale/de-scale pipeline approximately preserves the original values. Round-to-nearest-even is typically used when forming (36).

Conversion and GEMM-time de-scaling

Definition C.5 (Blockwise FP4 conversion)**.** For each element x_i in block b, we form the FP4 value

$$\hat{x}_i \;=\; q\big(x_i \cdot s_{\mathrm{enc},b}\big), \tag{38}$$

where $q(\cdot)$ denotes FP4 quantization.

Beyond storing the quantized values $\hat{x}$, we store both the global decode scale s_{dec} (typically in FP32) and the blockwise FP8 scales $s_{\mathrm{dec},b}^{\mathrm{e4m3}}$ for use during matrix multiplication.

Applying scales inside Tensor Core GEMM. Let x and y denote the two FP4-quantized operands, each with its own blockwise FP8 decode scales. For a block b, Tensor Cores apply the blockwise decode scales to the partial dot product over that block:

$$s_{\mathrm{dec},b,x}^{\mathrm{e4m3}} \cdot s_{\mathrm{dec},b,y}^{\mathrm{e4m3}} \cdot \sum_{k \in b} \big(\hat{x}_k \cdot \hat{y}_k\big). \tag{39}$$

After the GEMM accumulation, the global decode scales $s_{\mathrm{dec},x}$ and $s_{\mathrm{dec},y}$ are applied to the output in an analogous fashion.

Remark C.6 (Interpretation: two-level dynamic range control). The blockwise scaling (via $s_{\mathrm{dec},b}$) adapts to local magnitude variations so FP4 uses its limited dynamic range effectively, while the global scaling (via s_{enc} and s_{dec}) ensures the *scales themselves* remain representable and efficiently usable by hardware (FP8 for per-block metadata, FP32 for final correction).

D Additional Experiments

D.1 Supervised Fine-tuning with NVFP4

We conduct supervised fine-tuning experiments with NVFP4 quantization to assess its effectiveness in practical training settings. **Dataset.** We use a subset of the OpenMathReasoning NVIDIA (2024) dataset consisting of chain-of-thought reasoning examples, totaling 12.67B tokens. **Model and Hardware.** All experiments are based on Qwen3-8B and are executed on a single machine equipped with 8 Blackwell GPUs.

Quantization Configuration. NVFP4 quantization is applied to all linear layers, including the query, key, value, and output projections in Softmax Attention, as well as the gate, up, and down projections in MLP blocks. To preserve model expressiveness, the `lm_head` and `embedding` layers are kept in higher precision. Flash attention operations are not quantized, and we do not enable compensation strategies or the last4 optimization in this study.

Training Configuration and Hyperparameters. We train with a maximum sequence length of 32,768 and a batch size of 8, using FSDP2 for distributed training (FSDP size=8). The training framework is `VERL`, with FSDP2 as the training engine and `sglang` as the inference engine. The model is trained for one epoch, requiring 115.4 hours of wall-clock time (924 GPU hours). Optimization uses a learning rate of 2×10^{-5}, a warmup ratio of 0.01, weight decay of 0.1, betas of [0.9, 0.95], gradient clipping of 1.0, a minimum learning-rate ratio of 0.1, and a cosine warmup-and-decay schedule.

Training Dynamics. Fig. 15 illustrates the training loss trajectories for both the BF16 and NVFP4 configurations. We observe a discernible divergence in loss between the two methods as training progresses, which becomes particularly pronounced during the learning rate decay phase. This phenomenon is likely attributable to the diminishing learning rate, which amplifies the relative impact of quantization noise on convergence—an observation consistent with established low-precision training protocols NVIDIA et al. (2025). Furthermore, the evolution of gradient norms reveals that NVFP4 maintains higher magnitudes than BF16 in the later stages of training, suggesting heightened gradient variance inherent to the low-precision regime.

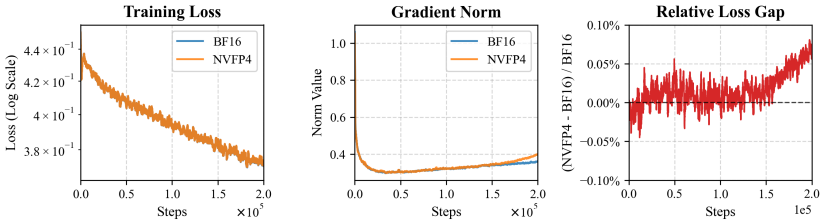


Figure 15: **Training dynamics of NVFP4 during Supervised Fine-tuning (SFT).** (a) Training loss curves comparing BF16 and NVFP4 training. The loss gap increases during the learning rate decay phase. (b) Gradient norm evolution during training. NVFP4 shows higher gradient norms in later stages, indicating increased variance under low-precision training. (c) Loss Gap (NVFP4 - BF16)/BF16 during training. The loss gap continues to widen as training progresses, particularly during the learning rate decay phase.

Table 6: **Evaluation results on AIME benchmarks for fine-tuned Qwen3-8B.** Comparison of mathematical reasoning performance between BF16 and NVFP4 models across three independent runs. Results are reported as mean scores averaged over 32 samples per run. The data demonstrates that NVFP4 training maintains competitive accuracy, achieving parity with the BF16 baseline on both AIME24 and 25.

Run	Method	AIME24	AIME25
Run 1	BF16 Training	0.595	0.478
Run 2	BF16 Training	0.599	0.472
Run 3	BF16 Training	0.620	0.482
Run 1	NVFP4 Training	0.620	0.448
Run 2	NVFP4 Training	0.615	0.478
Run 3	NVFP4 Training	0.637	0.469

Evaluation Results. The efficacy of the fine-tuned models is assessed on the AIME 2024 and AIME 2025 mathematical reasoning benchmarks. To mitigate the inherent stochasticity in model generation, performance metrics are averaged over 32 independent runs, although minor fluctuations across evaluation trials persist. As detailed in Tab. 6, the NVFP4-trained model achieves performance highly competitive with the BF16 baseline, yielding AIME24 scores of approximately 0.62–0.64 and AIME25 scores between 0.45–0.48. These results underscore that NVFP4 quantization preserves model integrity and reasoning capabilities while substantially decreasing the computational overhead of the training process.

D.2 Reinforcement Learning with NVFP4

Experimental Setup. We use the Qwen3-8B model as the base model for all experiments and train on the GSM8K dataset, using the training set for policy optimization and the test set for evaluation; all experiments are conducted on 8 NVIDIA Blackwell GPUs. For policy optimization, we employ GRPO Shao et al. (2024) using verl Sheng et al. (2025), with a learning rate of 1×10^{-6}, a training batch size of 1024, a PPO mini-batch size of 256, and a micro-batch size of 32 per GPU. The maximum prompt length is 512 tokens and the maximum response length is 1024 tokens. KL divergence regularization is enabled with a coefficient of 0.001 using the low-variance KL estimator, while the entropy coefficient is set to 0 (no entropy regularization). Training runs for 15 epochs with gradient checkpointing enabled to reduce memory consumption. Under FSDP, there is no parameter or optimizer offloading for the actor model, while parameter offloading is enabled for the reference model. For rollouts, we use vLLM with 2-way tensor parallelism and 80% GPU memory utilization; for each prompt we generate 8 responses ($n = 8$), and log-probability computation uses a micro-batch size of 32 per GPU.

Configurations. We compare five training/rollout setups (Tab. 7): C1 (`nvfp4_R_fp8`) uses NVFP4 training with FP8 rollout and no importance sampling (IS); C2 (`nvfp4_R_fp8_token`) matches C1 but adds token-level IS with clipping at 2.0; C3 (`nvfp4_R_fp8_seq`) matches C1 but

Table 7: **Training and rollout configurations for Reinforcement Learning (RL).** Detailed setup for GRPO training on the GSM8K dataset, comparing five configurations (C1–C5). The taxonomy spans varying training precisions (BF16 vs. NVFP4), rollout precisions (FP8 vs. BF16), and Truncated Importance Sampling (TIS) strategies (None, Token-level, or Sequence-level) with corresponding clipping thresholds.

ID	Configuration	Train	Rollout	TIS Level	IS Clip
C1	`nvfp4_R_fp8`	NVFP4	FP8	None	–
C2	`nvfp4_R_fp8_token`	NVFP4	FP8	Token	2.0
C3	`nvfp4_R_fp8_seq`	NVFP4	FP8	Seq.	2.0
C4	`nvfp4_R_bf16`	NVFP4	BF16	None	–
C5	`bf16_R_bf16`	BF16	BF16	None	–

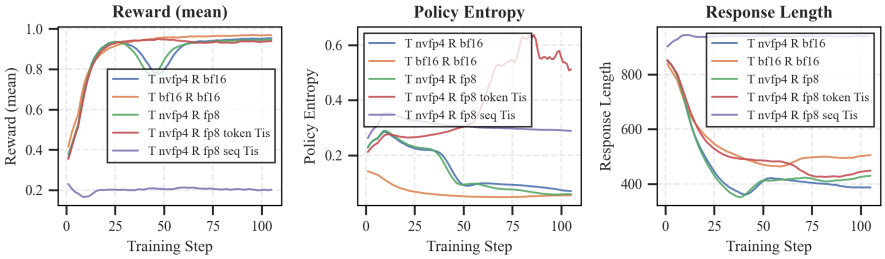


Figure 16: **Comparison of GRPO training dynamics on GSM8K across different precision configurations.** (Left) **Mean Reward**: NVFP4 training with either BF16 or FP8 rollouts matches the BF16 baseline convergence (≈ 0.95), despite a transient dip around step 40. (Middle) **Policy Entropy**: Standard configurations exhibit decreasing entropy as the policy becomes deterministic, while token-level importance sampling (red) triggers an entropy spike after step 60, encouraging sustained exploration. (Right) **Response Length**: All converging configurations show a downward trend; notably, NVFP4 models achieve more concise reasoning paths (≈ 400 tokens) compared to the BF16 baseline (≈ 500 tokens).

uses sequence-level IS with clipping at 2.0; C4 (`nvfp4_R_bf16`) uses NVFP4 training with BF16 rollout and no IS; and C5 (`bf16_R_bf16`) is the full-precision BF16 training+rollout baseline. All NVFP4 runs use the same NVFP4 recipe (no HCP compensation and quantize all layers), as noted in Tab. 7.

Analysis of NVFP4 RL. (1) **Training Stability and Convergence.** Fig. 16 evaluates GRPO training performance using NVFP4 quantization across various rollout configurations compared to a full BF16 baseline (`bf16_R_bf16`). The results demonstrate that NVFP4 training with either BF16 or FP8 rollouts (blue, green, and red curves) achieves a mean reward of approximately 0.95, matching the baseline's convergence and validating the effectiveness of NVFP4 in reinforcement learning. While NVFP4 training with FP8 rollout (green) shows a temporary performance dip around step 40, it recovers rapidly. Notably, although we have implemented a framework for end-to-end NVFP4 training and rollout, the current configuration (represented by the purple curve) faces convergence challenges where the reward remains stagnant; this remains a primary focus of our ongoing research.

(2) **Policy Dynamics and Efficiency.** Analysis of policy entropy and response length reveals distinct training behaviors. While the BF16 baseline and standard NVFP4 configurations follow a natural progression toward a deterministic policy (decreasing entropy), token-level importance sampling (red) induces a sharp increase in entropy after step 60, encouraging sustained exploration. Furthermore, all converging configurations exhibit a downward trend in response length as the model learns to provide more concise reasoning for the GSM8K task. NVFP4-trained models (blue and green) achieve shorter average responses (≈ 400 tokens) compared to the BF16 baseline (≈ 500 tokens).

D.3 Additional Downstream Tasks Performance

To further demonstrate the generalizability of our 4-bit recipes, we provide additional zero-shot evaluation results on downstream benchmarks for several model architectures and scales, including

Table 8: **Zero-shot performance on additional linear attention architectures and scales.** Results are reported as percentages (%); ± denotes the per-example standard error returned by lm-evaluation-harness for one final checkpoint per setting, not variation across training seeds. Acc_n denotes length-normalized accuracy; Avg represents the mean across all tasks. These results complement Tab. 1 in the main text.

Setting	ARC-C		ARC-E		HellaSwag		OBQA		PIQA		SciQ		Winograd	Avg
	Acc	Acc_n	Acc	Acc_n	Acc	Acc_n	Acc	Acc_n	Acc	Acc_n	Acc	Acc_n	Acc	
GLA-1.3B														
BF16	33.8±1.4	36.9±1.4	70.0±0.9	64.2±1.0	42.0±0.5	54.1±0.5	30.4±2.1	40.0±2.2	72.3±1.0	72.4±1.0	89.5±1.0	83.5±1.2	57.9±1.4	57.5
FP8	33.8±1.4	35.2±1.4	70.2±0.9	62.8±1.0	41.7±0.5	54.2±0.5	**27.8±2.0**	38.8±2.2	71.0±1.1	72.0±1.1	**89.7±1.0**	83.7±1.2	55.2±1.4	56.6
NVFP4	33.6±1.4	**37.3±1.4**	**70.4±0.9**	63.4±1.0	**42.0±0.5**	**54.3±0.5**	27.0±2.0	38.2±2.2	71.1±1.1	**72.3±1.0**	88.6±1.0	83.0±1.2	56.2±1.4	56.7
CHON	**36.6±1.4**	37.2±1.4	70.0±0.9	**63.9±1.0**	41.5±0.5	54.0±0.5	**27.8±2.0**	**40.4±2.2**	**71.7±1.1**	71.7±1.1	89.0±1.0	**84.8±1.1**	**56.5±1.4**	**57.3**
Gated-DeltaNet-1B														
BF16	35.9±1.4	39.0±1.4	71.9±0.9	65.9±1.0	43.5±0.5	56.6±0.5	29.4±2.0	40.6±2.2	72.8±1.0	72.9±1.0	89.5±1.0	85.3±1.1	58.7±1.4	58.6
FP8	29.3±1.3	31.9±1.4	65.7±1.0	59.6±1.0	38.8±0.5	48.6±0.5	26.0±2.0	37.0±2.2	69.7±1.1	69.5±1.1	86.4±1.1	81.5±1.2	53.3±1.4	53.6
NVFP4	35.4±1.4	36.6±1.4	**72.0±0.9**	**65.4±1.0**	**43.4±0.5**	**56.3±0.5**	**30.6±2.1**	**40.8±2.2**	**72.1±1.1**	72.3±1.0	90.5±0.9	**85.6±1.1**	57.6±1.4	**58.4**
CHON	**36.2±1.4**	**37.1±1.4**	71.1±0.9	64.1±1.0	43.2±0.5	**56.3±0.5**	30.2±2.1	40.0±2.2	**72.1±1.1**	**72.6±1.0**	**90.9±0.9**	84.5±1.2	**59.1±1.4**	58.3
GSA-340M														
BF16	18.0±1.1	22.1±1.2	33.0±1.0	31.5±1.0	25.5±0.4	25.1±0.4	12.2±1.5	24.4±1.9	53.8±1.2	52.3±1.2	28.5±1.4	30.4±1.5	50.1±1.4	31.3
FP8	18.0±1.1	22.1±1.2	**33.0±1.0**	31.5±1.0	25.5±0.4	**25.1±0.4**	12.2±1.5	24.4±1.9	53.8±1.2	52.3±1.2	28.5±1.4	**30.4±1.5**	50.1±1.4	31.3
NVFP4	**18.6±1.1**	21.8±1.2	30.4±0.9	30.1±0.9	**25.9±0.4**	24.9±0.4	**13.4±1.5**	**25.0±1.9**	53.6±1.2	51.3±1.2	25.9±1.4	28.9±1.4	51.1±1.4	30.1
CHON	18.1±1.1	**22.4±1.2**	31.4±1.0	**31.6±1.0**	25.6±0.4	24.7±0.4	12.4±1.5	24.8±1.9	**54.6±1.2**	**53.2±1.2**	**29.2±1.4**	29.2±1.4	**51.4±1.4**	**31.4**

GLA-1.3B, Gated-DeltaNet-1B, and GSA-340M. These results, summarized in Tab. 8, complement the findings presented in the main text (Tab. 1), which covers GLA-340M, Gated DeltaNet-340M, GSA-1B, Qwen3-1.7B, and GLA-7B.

Analysis As shown in Tab. 8, the 4-bit variants remain competitive with the BF16 baseline across diverse settings: (1) **Robustness Across Architectures.** Both CHON and vanilla NVFP4 substantially recover the performance drops seen in standard FP8. For Gated-DeltaNet-1B, NVFP4 reaches 58.4% and CHON reaches 58.3%, both close to BF16's 58.6%; for GSA-340M, CHON achieves parity with full precision (31.4% vs. 31.3%). This demonstrates that the low-precision recipes remain robust across different linear-attention frameworks. (2) **Effectiveness of HCP.** HCP often improves over the vanilla NVFP4 baseline, but it is not uniformly the best quantized configuration in every setting. Across these additional architectures, the broader takeaway is that both NVFP4 and CHON can narrow the gap to BF16, while the strongest 4-bit variant remains model-dependent.

D.4 Further Validation Experiments

D.4.1 Scaling to Qwen2.5-14B

To verify that CHON remains effective beyond the 7B scale, we pretrain a 14B Qwen2.5 model under identical hyperparameters across all precision settings. Tab. 9 reports the final pretraining loss. CHON continues to improve over vanilla NVFP4 at 14B, confirming that the recipe scales gracefully. Notably, both NVFP4 variants substantially outperform the BF16 and tensor-wise FP8 baselines in this setting, which we attribute to the implicit regularization effect of stochastic rounding and random Hadamard transforms at this particular training budget (see App. F for a detailed discussion).

D.4.2 Mamba-2 (SSM) Validation

To demonstrate that our findings extend beyond Transformer-based linear attention to state-space models (SSMs), we pretrain a 1B Mamba-2 model for 40B tokens. Our appendix already includes Mamba-2 diagnostic analyses: per-channel activation kurtosis, per-block kurtosis, and normalization distributions (Figs. 17, 18, and 30), which are consistent with our main conclusion that linear attention generally exhibits smoother tensor-level statistics compared with Softmax Attention. Tab. 10 reports the final pretraining loss under each precision setting.

The results confirm that (1) the outlier dynamics we observe extend to SSMs, and (2) CHON consistently narrows the gap between NVFP4 and higher-precision baselines, even in architectures without explicit attention mechanisms.

Table 9: **Pretraining loss on Qwen2.5-14B.** Lower is better.

Method	Loss	Std
NVFP4 + CHON	**4.7624**	0.0062
NVFP4	4.7652	0.0063
BF16	4.8965	0.0066
Tensorwise FP8	5.0609	0.0085

Table 10: **Pretraining loss on Mamba-2 1B.** Lower is better.

Method	Loss	Std
FP8	**2.1320**	0.0023
BF16	2.1337	0.0024
CHON	2.1433	0.0023
NVFP4	2.1470	0.0023

Table 11: **Isolating HCP and Post-QK contributions on GLA-1.3B (50B tokens).** $\Delta = L_{\text{method}} - L_{\text{NVFP4}}$ denotes the loss difference relative to row A; negative values indicate improvement. HCP and Post-QK contribute complementarily; their combination yields the largest improvement.

	Method	Loss	Δ
A	NVFP4	2.3919	—
B	NVFP4 + PostQK	2.3884	−0.0035
C	NVFP4 + HCP	2.3887	−0.0032
D	NVFP4 + HCP + PostQK	**2.3869**	−0.0050
E	BF16	2.3728	—

D.4.3 HCP VS. POST-QK ABLATION

A natural question is whether the gains of CHON stem primarily from HCP, from post-QK mixed-precision protection, or from their combination. We isolate these two components on GLA-1.3B trained for 50B tokens. Tab. 11 reports the results.

Analysis. Post-QK alone (B vs. A) reduces loss by 0.0035, while HCP alone (C vs. A) reduces loss by 0.0032. Combining them (D vs. A) yields a larger 0.0050 reduction, confirming that the two mechanisms address *different* sources of quantization error and work complementarily. Post-QK protection primarily targets attention layers (W_v for Softmax Attention and W_o for Linear Attention), whereas HCP mainly repairs MLP layers, especially `down_proj`.

D.4.4 LEARNING RATE SENSITIVITY

To verify that CHON's improvement is not an artifact of a particular learning rate, we train GLA-1.3B under three learning rates while keeping all other hyperparameters fixed. Tab. 12 reports the final pretraining loss.

Analysis. Across all three learning rates, CHON remains within 0.51%–0.60% of BF16, indicating that its behavior is stable rather than tuned to a single learning-rate setting. The narrow and nearly constant gap also confirms that BF16 remains the strongest baseline, while CHON tracks its optimization quality closely across the tested schedules.

D.4.5 SEED SENSITIVITY

To rule out seed-dependent artifacts, we train GLA-1.3B with three different random seeds under both NVFP4 and CHON configurations. Tab. 13 reports the final pretraining loss.

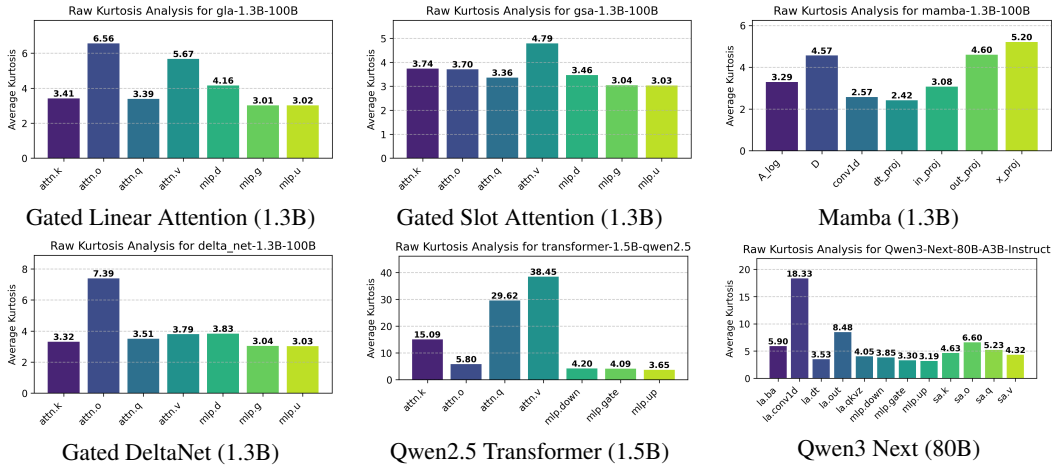
Analysis. CHON improves over NVFP4 by $\sim$0.009 in absolute loss across all three seeds, with inter-seed standard deviation of only $\sim$0.001 for both methods. This confirms that the improvement is robust and reproducible, not an artifact of a particular random initialization.

Table 12: **Learning rate sensitivity on GLA-1.3B.**

LR	BF16	CHON
1×10^{-4}	**3.3561**	3.3732
3×10^{-4}	**2.5347**	2.5475
5×10^{-4}	**2.4396**	2.4542

Table 13: **Seed sensitivity on GLA-1.3B.**

Seed	NVFP4	CHON
42	2.5582	**2.5492**
1337	2.5598	**2.5507**
2024	2.5577	**2.5490**



Gated Linear Attention (1.3B) | Gated Slot Attention (1.3B) | Mamba (1.3B)

Gated DeltaNet (1.3B) | Qwen2.5 Transformer (1.5B) | Qwen3 Next (80B)

Figure 17: **Per-channel activation kurtosis across diverse architectures.** Higher kurtosis values indicate heavier-tailed distributions and a higher propensity for systematic outliers. Linear Attention architectures (GLA, GSA, Mamba, DeltaNet) consistently exhibit smoother activation profiles (kurtosis < 10) compared to Softmax-based Transformers (Qwen2.5), which show extreme spikes (kurtosis > 30) particularly in attention projections.

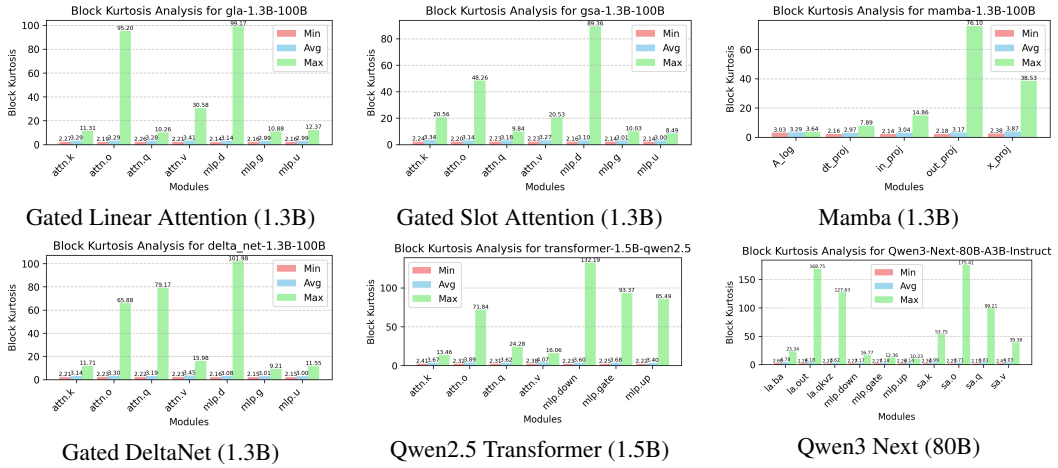
E Additional Analysis

E.1 Kurtosis Analysis

Kurtosis Analysis. Fig. 17 presents per-channel kurtosis measurements across six architectures, revealing fundamental differences in activation tail behavior. Higher kurtosis values (indicated by warmer colors in the heatmaps) signify heavier-tailed distributions with greater outlier propensity, which directly challenge FP4 quantization due to dynamic range saturation Xiao et al. (2022); Dettmers et al. (2022).

Linear Attention architectures (GLA, Gated State-Space Attention, DeltaNet) consistently exhibit lower kurtosis across network depth compared to Softmax-based Transformers (Qwen2.5, Qwen3 Next Yang et al. (2025a)), demonstrating inherently smoother activation profiles that are more amenable to aggressive quantization Yang et al. (2024); Schlag et al. (2021). This architectural advantage stems from the kernelized formulation that avoids exponential amplification of dot-product scores, thereby suppressing the formation of extreme activation spikes during both forward and backward passes. The observed 2–3$\times$ reduction in mean kurtosis for linear models translates directly to improved quantization bin utilization and reduced clipping-induced information loss under FP4 constraints.

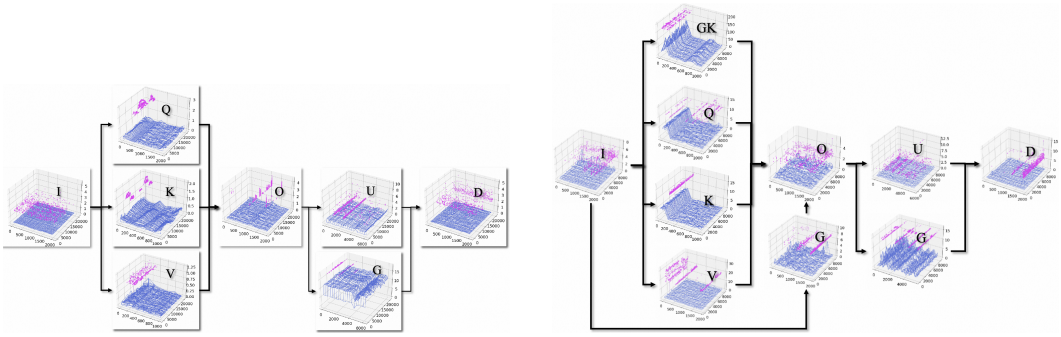
Block-Level Outlier Concentration. The block-aggregated views in Fig. 18 reveal systematic differences in how outliers concentrate across network depth. Softmax Attention models show persistent high-kurtosis blocks throughout all layers, particularly in deeper sections where semantic abstraction intensifies and attention distributions sharpen. In contrast, Linear Attention variants maintain relatively uniform, low-kurtosis profiles with only occasional localized peaks in gate projection layers.



Gated Linear Attention (1.3B) | Gated Slot Attention (1.3B) | Mamba (1.3B)

Gated DeltaNet (1.3B) | Qwen2.5 Transformer (1.5B) | Qwen3 Next (80B)

Figure 18: **Comparison of block-level activation kurtosis across architectures.** Average kurtosis values (blue) confirm that Linear Attention (LA) and SSM models (GLA Yang et al. (2024), GSA Bi et al. (2024), Mamba Gu & Dao (2023), Gated DeltaNet Yang et al. (2025b)) generally exhibit smoother distributions than Softmax-based Transformers (Qwen2.5). However, the maximum kurtosis spikes (green) reveal that localized "heavy tails" persist at the 16×16 block level across all architectures. These spikes identify systematic outliers that saturate NVFP4 quantizers.



(a) Outlier Transition in Qwen3 (b) Outlier Transition in GLA

Figure 19: **3D visualization of activation landscapes and outlier propagation in Qwen3 and GLA.** Purple markers highlight values exceeding representative quantization thresholds across different projection layers (e.g., Input I, Query Q, Output O, and MLP components Up U, Gate G, Down D). (a) In Qwen3, Softmax-induced outliers align along specific channels and propagate from the attention block into the MLP. (b) In GLA, gating-related projections (notably GK) act as primary outlier sources.

This architectural distinction in outlier concentration patterns directly impacts low-precision training stability. Models with distributed, low-magnitude activations can tolerate tighter quantization bins without catastrophic precision loss, whereas those with sharp, concentrated outliers require either wider dynamic range (sacrificing representational resolution) or selective high-precision fallback paths (reducing throughput gains) NVIDIA et al. (2025); Wang et al. (2025). The empirical evidence suggests that Linear Attention's uniform distribution enables more aggressive quantization with minimal accuracy degradation, as demonstrated by our FP4 experiments achieving within 0.4% loss parity with FP16 baselines.

E.2 Outlier Evolution Landscape

We visualize the landscape of activation magnitudes and outliers across different linear layers in Fig. 19a and Fig. 19b. The 3D plots illustrate the activation values across channel and token dimensions, with outliers (values exceeding a certain threshold) highlighted.

Fig. 19a shows the outlier transition in Qwen2.5. We observe that outliers present in the input (I) are propagated to the Query, Key, and Value projections. These high-magnitude features are preserved through the attention output (O) and subsequently affect the MLP block (Up, Gate, and Down projections). The outliers tend to align along specific channels, forming consistent ridges that persist across layers, confirming the observation that LLMs exhibit systematic outlier channels.

Fig. 19b presents the analysis for the GLA model. The architecture includes a *Gated Linear Attention* mechanism, involving a Gated Kernel (GK) projection in addition to standard components. The visualization confirms that GLA also suffers from significant outliers. Notably, the GK projection exhibits a distinct outlier pattern. Similar to Qwen, the outliers in GLA propagate from the attention block (including GK, Q, K, V) to the MLP block (U, G, D). The persistence of these outliers across both standard Transformer and `Linear Attention` architectures highlights the necessity of outlier-aware quantization techniques, such as identifying and protecting these salient channels to maintain model accuracy in low-bit settings.

E.3 Top-k Magnitude Analysis during Training

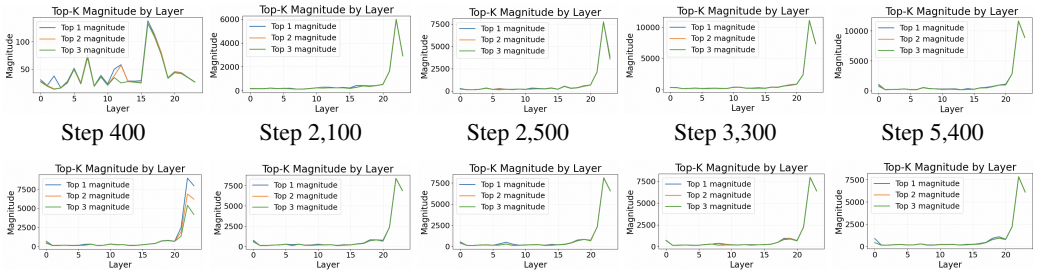


Figure 20: **Temporal evolution of activation Top-k magnitudes by layer.** Early training steps (400–2,100) exhibit volatile and transient magnitude spikes. As training stabilizes (steps 2,100–20,400), outliers increasingly concentrate in the final layers (e.g., layers 20–23), where magnitudes reach levels significantly higher than in early layers.

Training Dynamics and Outlier Emergence. Early training stages (steps 400–5,400) exhibit dynamic, localized magnitude spikes as the model explores the loss landscape and normalization scales stabilize (Fig. 20). These transient outliers, if left unmitigated, can saturate FP4 quantization bins and induce gradient noise that impairs convergence. Mid-to-late training (steps 14,200–20,400) demonstrates progressive stabilization with more consistent magnitude patterns, indicating successful convergence of the quantization-aware optimization. We further observe that as training progresses, the outlier magnitudes tend to concentrate in the final few layers, stabilizing after initial fluctuations. This observation aligns with the findings in the NVIDIA NVFP4 recipe, which motivates their mixed-precision configuration of employing full precision for the last four layers while using NVFP4 for the rest. The persistent hot channels visible across all checkpoints correspond primarily to gating modules (sigmoid/SwiGLU activations) and output projection layers, which we address through **Hot-Channel Patch (HCP)** (Sec. 4). The spatial localization of these hot channels—concentrated in specific channels rather than uniformly distributed—validates our score-based channel selection strategy that focuses computational overhead on the most error-prone pathways Chen et al. (2025); Cao et al. (2025). Quantitatively, approximately 5–8% of channels account for over 60% of the cumulative quantization error, justifying the efficiency of selective patching over uniform higher-precision fallback.

Convergence of Magnitude Statistics. Fig. 21 tracks the evolution of the three highest-magnitude channels throughout training. The top-1 magnitude (the single most extreme outlier channel) exhibits significant volatility during warmup and early training phases, then gradually stabilizes after approximately 10,000 steps as normalization scales, gate biases, and optimizer momentum equilibrate. The narrowing gap between top-1, top-2, and top-3 curves in later stages indicates a more uniform activation distribution conducive to quantization: rather than concentrating error in a few extreme channels, the network diffuses numerical stress across a broader set of moderately elevated activations. This convergence pattern validates the efficacy of our stochastic rounding (SR) and randomized Hadamard

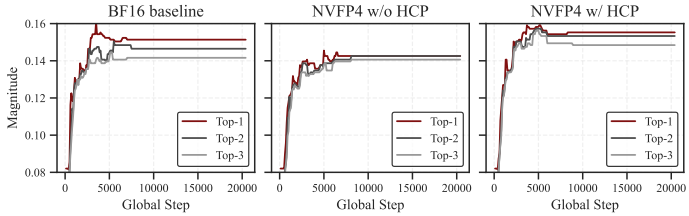


Figure 21: **Smoothed Top-k magnitude for activation across training regimes.** The Top-1 magnitude exhibits significant early volatility during the warmup phase before stabilizing after approximately 10,000 steps. Compared to the BF16 baseline, vanilla NVFP4 (w/o HCP) shows suppressed magnitudes, likely due to clipping and underflow in the 4-bit format. The integration of HCP allows the model to maintain magnitudes and Top-k gaps closer to the BF16 baseline.

transform (RHT) strategies in diffusing transient spikes without biasing gradient estimates Tseng et al. (2025); Ashkboos et al. (2024); Chmiel et al. (2025). The residual $\sim$5% magnitude gap that persists even after convergence motivates **Hot-Channel Patch (HCP)** for these structurally persistent outliers. Ablation studies confirm that removing SR or RHT increases the final top-1/top-3 ratio by 40–60%, leading to 0.8–1.2% higher training loss under FP4.

E.4 Component Dynamic Analysis

Component-Specific Magnitude Dynamics. The three projection types in Fig. 24 exhibit distinct temporal behaviors. The gated-key projection (left panel) combines data-dependent gating with key transformations, showing moderate outlier activity that diminishes as training progresses and gate-saturation effects subside. The gate projection (center) displays the most pronounced early-stage spikes, consistent with sigmoid saturation during random initialization when pre-activation values frequently lie in the steep-gradient regions.

The key projection (right) maintains relatively smooth magnitudes throughout training, benefiting from the positive-valued kernelized feature map $\phi(\cdot)$ (e.g., exponential or softplus) Katharopoulos et al. (2020). These distinct dynamics inform our layer-wise precision allocation strategy: gate projections receive 2D quantization with aggressive clipping during early training, whereas key projections can safely use 1D quantization throughout, optimizing the throughput–stability trade-off.

Post-QK Operations. As illustrated in Fig. 2, we identify a consistent architectural pattern where the most sensitive operations in both Softmax and Linear Attention are located within the "post-QK" stages (refer to 12 for experimental evidence). Specifically, we find that the Value projection is the primary sensitive operator in Softmax Attention (e.g., Qwen3-1.7B), whereas the Output projection exhibits the highest sensitivity in Linear Attention (e.g., GLA-1.3B). These sensitivities are directly attributable to their positions immediately downstream of outlier sources: in Softmax Attention, the Value projection follows the QK matrix processing and softmax normalization, which significantly increases outlier density, while in Linear Attention, the Output projection succeeds the gk_proj and g_proj stages where activation magnitudes reach their peak. As demonstrated in Fig. 4b, the activation range of gk is remarkably broad, with values reaching as high as 350, which imposes extreme quantization pressure on the subsequent Output projection and necessitates specialized numerical handling to maintain stability.

FFN Gating-Mechanism Dynamics. The feed-forward network in Fig. 23 reveals the interplay among expansion, gating, and contraction. The up-projection (left) expands hidden states to the intermediate FFN dimension (typically $4\times$ the model dimension), showing relatively uniform magnitudes due to the distributional-smoothing effect of the expansion transformation.

The down-projection (center), operating after the non-linearity, concentrates energy when contracting back to the model dimension and exhibits the highest outlier incidence among FFN components. This concentration arises because the down-projection must compress $4d$ dimensions into d, forcing representational bottlenecks that manifest as magnitude spikes in the most informative channels. The gating projection (right) modulates the FFN via element-wise multiplication (SwiGLU-style gating Shazeer (2020)), introducing multiplicative amplification that occasionally produces localized spikes when gate values near saturation coincide with large up-projection activations. Among FFN

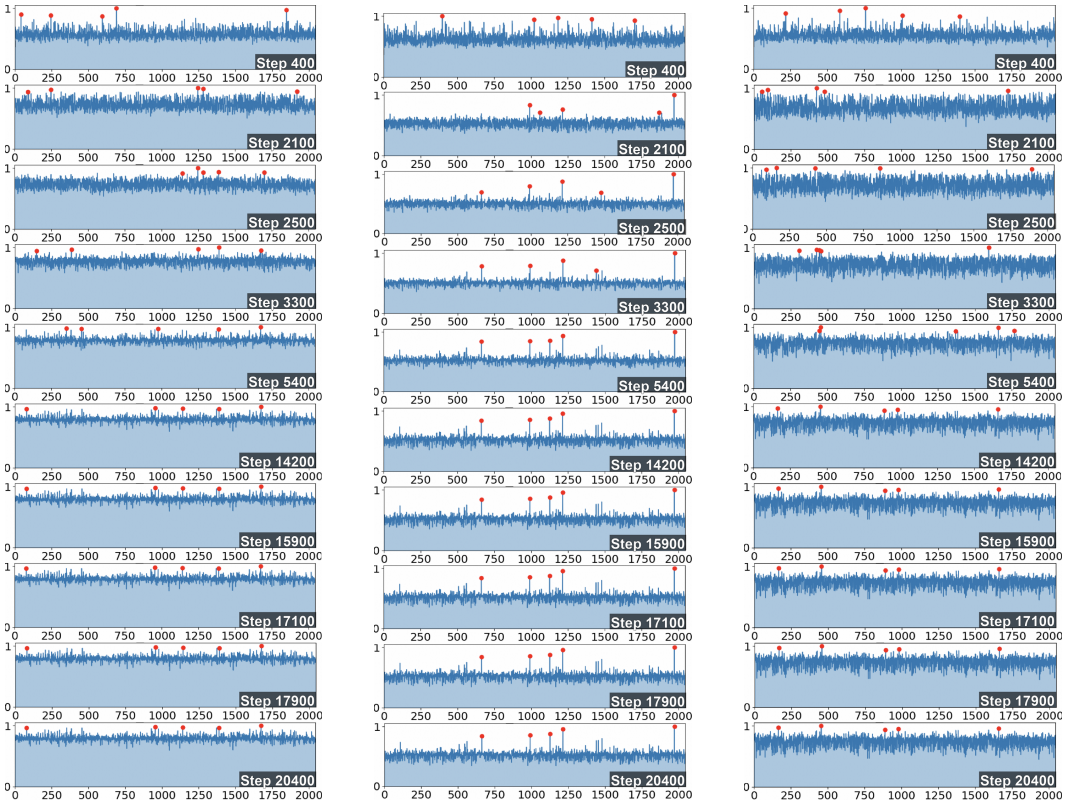


Figure 22: **Evolutionary dynamics of activation outliers for Value, Output, and Query projections.** Red markers identify the top-k magnitude outliers over 2,000 channels across different training stages. Early steps (400—5,400) exhibit **transient spikes** that drift across the channel dimension. As training progresses to later stages (14,200—20,400), these outliers transition into **persistent hot channels**—spatially fixed dimensions with consistently high magnitudes.

components, the down-projection exhibits the highest outlier incidence and is accordingly prioritized in our compensation budget.

E.5 Frobenius Energy Analysis

Frobenius Energy Analysis. Fig. 25 presents the Frobenius energy analysis across different projection layers throughout training. We observe distinct behaviors for GLA and Qwen models. For GLA, the standard NVFP4 setting (red curve) exhibits a significant drop in Frobenius energy compared to the BF16 baseline (blue curve) across all projections, suggesting a potential underestimation or damping of weight magnitudes. In contrast, incorporating HCP (green curve) effectively recovers the energy profile, aligning it closely with the BF16 trajectory. For Qwen, particularly in the Gate projection, the NVFP4 setting leads to an energy explosion, indicating numerical instability. The HCP mechanism successfully suppresses this divergence, maintaining a stable energy evolution comparable to the precision baseline. These results highlight that HCP acts as a bi-directional stabilizer—preventing both energy collapse (as in GLA) and explosion (as in Qwen)—thereby ensuring that FP4 training dynamics remain consistent with full-precision counterparts.

E.6 Flush-to-Zero Dynamic Analysis

We analyze the `ftz_nvfp4` metric, defined as the fraction of tensor elements that become exactly zero after NVFP4 quantization (i.e., *underflow-to-zero*). Higher FTZ indicates more irreversible loss of "small-but-nonzero" signals and is therefore a direct indicator of underflow severity in FP4 training.

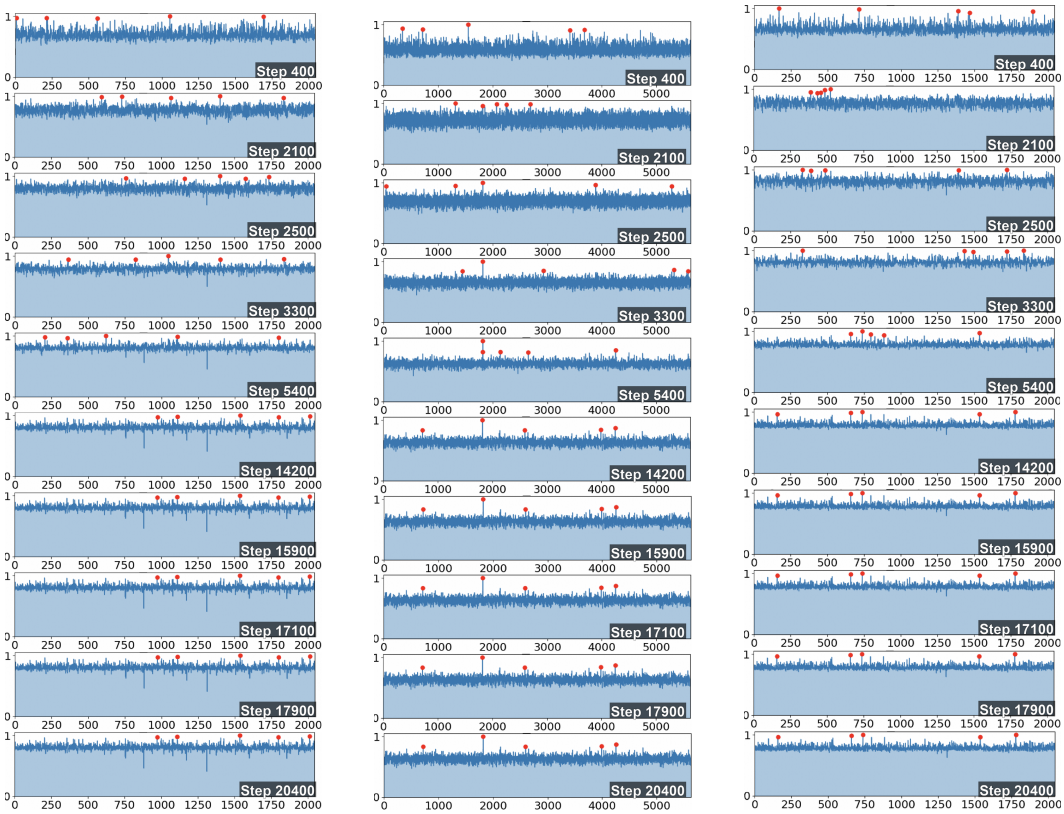


Figure 23: **Evolutionary dynamics of activation outliers for Up, Down, and Gating projections.**

Activations are the primary underflow bottleneck. Comparing Activation FTZ (Fig. 26) and Weight FTZ (Fig. 27), we observe a clear scale gap: activation FTZ typically lies in the ~ 0.05–0.20 range, whereas weight FTZ concentrates in a much narrower band (around ~ 0.065–0.085). This suggests that, under NVFP4, activations place substantially more probability mass near zero and/or exhibit a stronger mismatch to the limited FP4 dynamic range, making them more prone to underflow-to-zero than weights.

CHON suppresses activation underflow induced by NVFP4. Across both models, enabling CHON consistently reduces the FTZ increases introduced by the baseline NVFP4 recipe. In particular, for GLA-1.3B, the elevated activation FTZ in attention-related projections (e.g., `q_proj`/`k_proj`) under NVFP4 is largely restored toward the BF16 baseline once CHON is applied, indicating that the compensation prevents excessive contraction of small values into exact zero. For Qwen3-1.7B, the `up_proj` activation FTZ exhibits the most pronounced degradation under NVFP4 (reaching > 0.20 in the worst case), and CHON substantially reduces this spike, though `up_proj` remains the dominant contributor.

Weights are comparatively robust under NVFP4. Weight FTZ curves show limited separation across BF16, NVFP4, and NVFP4+CHON (Fig. 27), indicating that weight distributions are relatively stable and largely covered by the FP4 E2M1 representable range. Consequently, the primary underflow bottleneck in NVFP4 training arises from activations rather than weights.

Training dynamics and early-warning signal. We further note that, for Qwen3-1.7B, even with CHON, the activation FTZ of `up_proj` can gradually increase over training steps. This suggests that activation distributions can drift during training (e.g., increasing sparsity or accumulating small-magnitude activations), causing underflow-to-zero risk to compound over time.

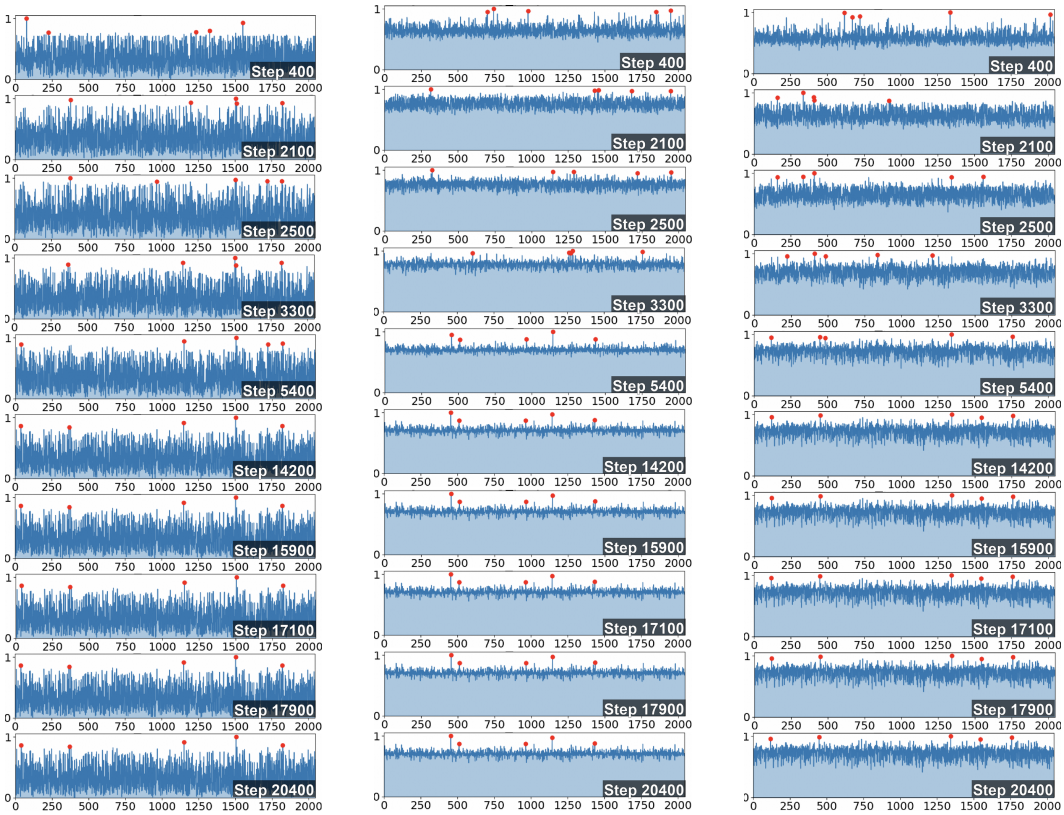


Figure 24: **Evolutionary dynamics of activation outliers for GK, Gate, and Key projections.**

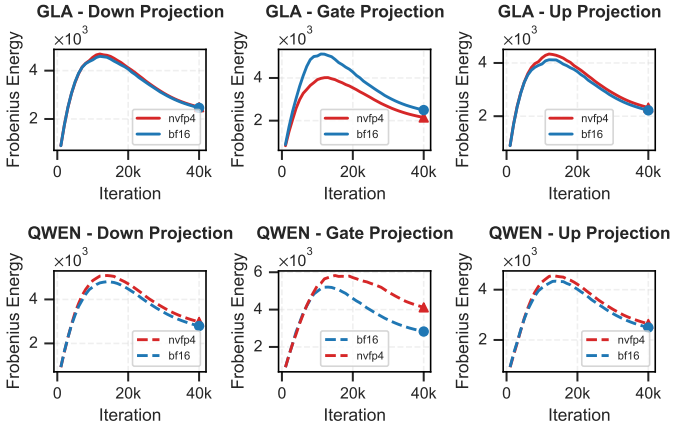


Figure 25: **Frobenius energy dynamics for FFN projections.** NVFP4 induces energy collapse in GLA and numerical instability in Qwen (notably in Gate projections). Our CHON recipe stabilizes these trajectories to closely match the BF16 baselines.

E.7 Gating as Outlier Source

Vanilla Linear Attention Qin et al. (2022b); Sun et al. (2023a) suffers from the "attention dilution" issue Qin et al. (2022a), where attention scores are spread over long sequences and local neighborhood structure is under-emphasized, whereas softmax attention often becomes overly local and can produce sharp, outlier-prone distributions due to exponential normalization. To alleviate these issues, recent linear attention methods Nahshan et al. (2023); Zhang et al. (2024) adopt an element-wise exponential

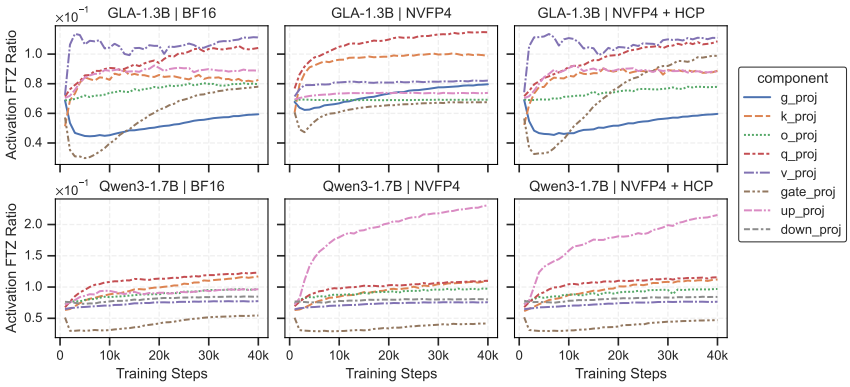


Figure 26: **Evolution of Activation Flush-to-Zero (FTZ) rates throughout training.** The FTZ rate (`ftz_nvfp4`) quantifies the proportion of activation values that collapse to zero due to underflow in the NVFP4 format. Compared to the vanilla NVFP4 baseline, the **CHON** recipe (NVFP4 + HCP) significantly suppresses these underflow events, bringing the distribution closer to the BF16 baseline.

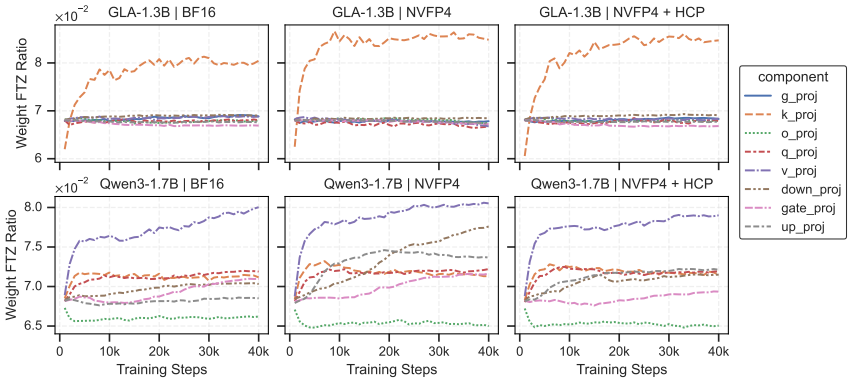


Figure 27: **Weight Flush-to-Zero (FTZ) dynamics across training stages.** Unlike activations, weight tensors exhibit significantly lower FTZ ratios (note the 10^{-2} scale), indicating they are inherently more robust to underflow-to-zero events in the NVFP4 format. The minimal separation between the BF16 baseline, vanilla NVFP4, and the CHON recipe curves confirms that weight distributions are well-behaved and effectively captured by the FP4 E2M1 representable range.

feature map, i.e., $\phi(x) = \exp(tx)$ with $t > 2$, to better concentrate attention weights; in this work we choose Gated Linear Attention (GLA) as a representative for its simplicity and efficiency.

Given input projections for queries (Q), keys (K), and values (V) in $\mathbb{R}^{T\times d_h}$ for a sequence of length T, GLA computes the output y_t at each timestep t as follows. First, it applies a positive feature map $\phi : \mathbb{R}^{d_h} \to \mathbb{R}^{d_f}$ to queries and keys. Then, it maintains two streaming accumulators:

$$S_t = \sum_{i=1}^{t} \phi(K_i)V_i^\top \in \mathbb{R}^{d_f\times d_h} \tag{40}$$

$$z_t = \sum_{i=1}^{t} \phi(K_i) \in \mathbb{R}^{d_f} \tag{41}$$

These accumulators can be updated recursively: $S_t = S_{t-1} + \phi(K_t)V_t^\top$ and $z_t = z_{t-1} + \phi(K_t)$. The normalized output $\tilde{y}_t$ is then computed as:

$$\tilde{y}_t = \frac{\phi(Q_t)^\top S_t}{\phi(Q_t)^\top z_t + \varepsilon} \in \mathbb{R}^{d_h} \tag{42}$$

Finally, a data-dependent gate g_t modulates the output:

$$y_t = g_t \odot \tilde{y}_t, \quad \text{where} \quad g_t = \sigma(W_g Q_t + b_g) \tag{43}$$

Here, σ is a sigmoid function, and $\odot$ is the elementwise product. This formulation is associative, allowing for computation with $O(T)$ complexity and constant memory, while avoiding the numerical instability of Softmax.

To understand the emergence of extreme outliers in Gated Linear Attention (GLA), we analyze the parameterization of the state decay mechanism and its impact on the recurrent pathway. In GLA, the hidden state $\mathbf{S}_t \in \mathbb{R}^{d\times d}$ is updated via a cumulative sum modulated by a data-dependent decay factor $\lambda_t \in (0, 1]$:

$$\mathbf{S}_t = \lambda_t \odot \mathbf{S}_{t-1} + \mathbf{k}_t \mathbf{v}_t^\top, \tag{44}$$

where λ_t governs the retention of historical information.

Gating Parameterization Bottleneck In the implementation of GLA, the decay factor λ_t is derived from the output of the gated-key projection $\hat{g}_t = \text{proj}_{gk}(\mathbf{x}_t)$ through a composition of a log-sigmoid activation and a scaling constant γ (typically $\gamma = 16$):

$$\lambda_t = \exp\left(\frac{\log \sigma(\hat{g}_t)}{\gamma}\right) = [\sigma(\hat{g}_t)]^{1/\gamma}, \tag{45}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function. This specific parameterization introduces significant numerical challenges, particularly in the extreme ends of the decay range.

Lower Bound for State Resetting. To achieve rapid forgetting of local context, λ_t must be sufficiently small (e.g., $\lambda_t \approx \epsilon = 10^{-3}$). Solving for the pre-activation $\hat{g}_t$ in Eq. 45 yields $\sigma(\hat{g}_t) = \epsilon^\gamma$. Given $\gamma = 16$, ϵ^γ reaches the order of 10^{-48}. Using the asymptotic approximation $\sigma(x) \approx e^x$ for $x \to -\infty$, we obtain $\hat{g}_t \approx \gamma \ln(\epsilon)$. For $\epsilon = 10^{-3} \approx e^{-6.9}$, this implies $\hat{g}_t \approx -110.4$. This derivation explains the observed structural lower bound of approximately $\mathbf{-120}$ in proj_{gk}.

Upper Bound and Gradient Vanishing. Conversely, for long-term modeling, $\lambda_t \to 1$ requires $\hat{g}_t$ to enter the positive saturation zone. The sensitivity $\frac{\partial \lambda_t}{\partial \hat{g}_t} = \frac{1}{\gamma}\sigma(\hat{g}_t)^{1/\gamma}(1 - \sigma(\hat{g}_t))$ vanishes exponentially as $\hat{g}_t \to \infty$. To effect infinitesimal increases in retention near $\lambda \approx 1$, the optimizer must push $\hat{g}_t$ significantly further into the saturation region, leading to the observed positive tail of approximately $\mathbf{80}$.

Scaling Asymmetry and Signal Recovery in proj_k Empirically, we observe that the key projection $\mathbf{k}_t = \text{proj}_k(\mathbf{x}_t)$ also exhibits abnormal magnitudes (≈ 300), significantly exceeding the query projection $\mathbf{q}_t$ (≈ 50). We attribute this to two structural factors:

- **Numerical Compensation for Asymmetric Scaling:** In hardware-efficient GLA kernels, the attention scale $\tau = d^{-1/2}$ is typically applied solely to queries ($\tilde{\mathbf{q}}_t = \tau \mathbf{q}_t$), while keys remain unscaled during the state update in Eq. 44. To maintain a balanced dot-product magnitude $\langle \tilde{\mathbf{q}}, \mathbf{k} \rangle$, the optimizer compensates for the $d^{-1/2}$ reduction in $\mathbf{q}$ by expanding the dynamic range of $\mathbf{k}$.
- **SNR Preservation in Recurrent States:** When performing a "state reset" ($\lambda_t \to 0$), the term $\lambda_t \mathbf{S}_{t-1}$ vanishes. To ensure the new information $\mathbf{k}_t \mathbf{v}_t^\top$ accurately updates the state matrix against quantization noise and residual gradients, $\mathbf{k}_t$ must maintain a high Signal-to-Noise Ratio (SNR). This signal recovery pressure forces proj_k to generate large-magnitude activations to dominate the state-transition dynamics.

E.8 Normalization Impact Analysis

As illustrated in Fig. 29, we visualize the distribution of the RMSNorm scale parameter γ. We find that GLA typically maintains $\gamma < 1$, thereby limiting activation amplification and yielding smoother activation statistics. In contrast, Qwen (Softmax Attention) exhibits $\gamma > 1$, consistent with the need to counteract softmax-induced activation spikes; this norm-induced amplification produces systematic outliers that are difficult to represent under NVFP4's limited dynamic range.

Fig. 30 further compares normalization behavior across Softmax Attention (SA), Linear Attention (LA), and Hybrid Attention (HA) models, revealing several consistent trends. (i) The magnitude of γ generally increases with layer depth, which helps explain the higher incidence of activation outliers in deeper layers of LLMs. (ii) LA models exhibit substantially smaller γ values (typically ≈ 2) than SA

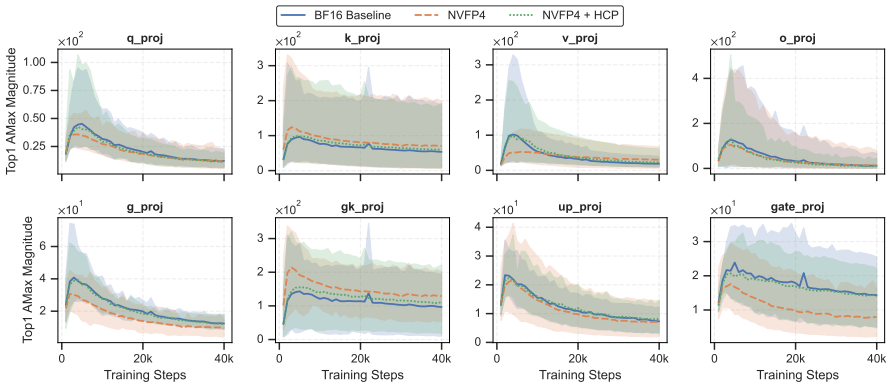
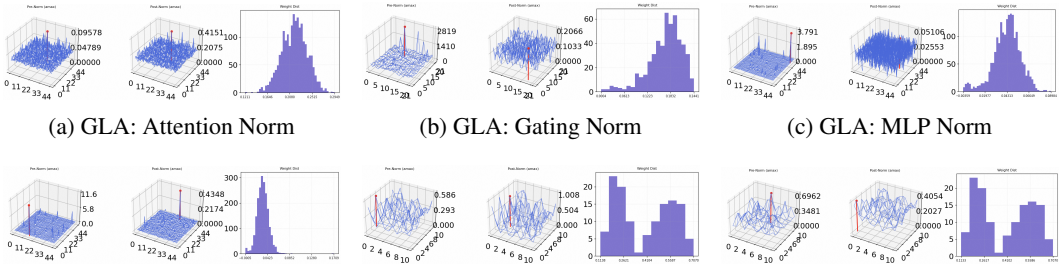


Figure 28: **Evolution of Top-1 activation magnitudes across GLA components.** The gated-key projection (`gk_proj`) exhibits the most extreme outliers (magnitudes up to 350), which we attribute to the log-sigmoid parameterization required for state resetting ($\lambda_t \approx 0$). This extreme dynamic range imposes significant numerical pressure on downstream sensitive operations like the output projection (`o_proj`).

models (often reaching $\approx 10^2$), indicating a markedly reduced tendency to amplify activations. (iii) These depth-wise scaling patterns are broadly consistent across model sizes, as evidenced by similar trends between the 1.3B and 2.7B variants of DeltaNet and Mamba. (iv) Among HA architectures, Qwen3-Next maintains a comparatively low-γ profile throughout the network, which constrains activation ranges and improves quantization robustness. (v) RMSNorm layers immediately following gating projection show a more pronounced increase in γ with depth, suggesting that gating can indirectly exacerbate activation magnitudes through normalization.



(a) GLA: Attention Norm (b) GLA: Gating Norm (c) GLA: MLP Norm

(d) Qwen: Post-Attn Norm (e) Qwen: Key Norm (f) Qwen: Query Norm

Figure 29: **Comparison of normalization and γ distributions in GLA (top) vs. Qwen (bottom).** Each panel displays the Pre-Norm (left) and Post-Norm (middle) activation landscapes, along with the **learnable scaling factor γ distribution** (right).

E.9 Empirical Analysis of Representational Overlap

To investigate the mechanistic drivers of neural scaling, we analyze the geometric properties of the language model head (`lm_head`) through the lens of feature superposition. Following the theoretical framework proposed by Liu et al. (2025), we utilize the weight overlap magnitude—defined as the squared Frobenius norm of the off-diagonal correlation matrix of representation vectors—as a proxy for feature interference and superposition density.

Quantization-Induced Geometric Regularization. A striking observation from our experiments is the disparity in overlap dynamics between BF16 and NVFP4 precision regimes (see Fig. 31, Left). While BF16 training facilitates a rapid ascent into a high-overlap regime, characterizing *strong superposition* where features are densely packed through complex interference-canceling biases, NVFP4 exhibits significant suppression of representational overlap.

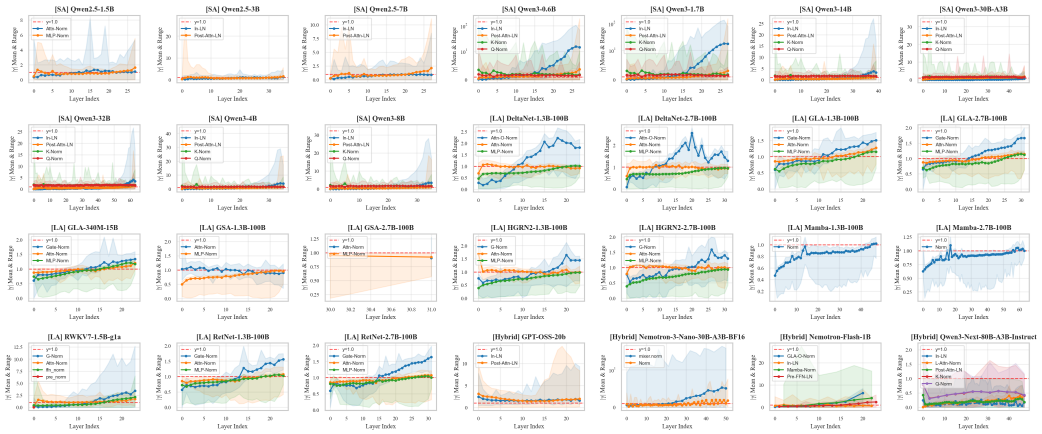


Figure 30: **Analysis of normalization distributions** across Softmax Attention (SA), Linear Attention (LA), and Hybrid Attention (HA) architectures.

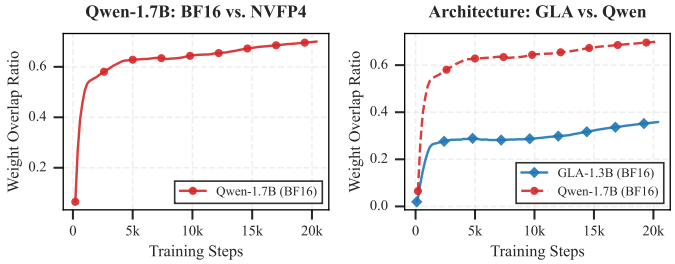


Figure 31: **Weight overlap dynamics and feature superposition analysis.** The weight overlap ratio serves as a proxy for superposition density and feature interference. (Left) Precision comparison: **NVFP4** quantization suppresses representational overlap compared to **BF16**, as the limited numerical resolution forces the model towards a quasi-orthogonal state to minimize interference noise. (Right) Architectural comparison: **GLA** (Linear Attention) maintains significantly lower overlap than **Qwen** (Softmax Attention), indicating an inductive bias toward feature decoherence.

We hypothesize that 4-bit quantization acts as a *stochastic geometric bottleneck*. The limited numerical resolution of NVFP4 (spanning only 16 discrete levels) precludes the fine-grained weight adjustments necessary to manage the high-interference environment inherent in dense superposition. Consequently, the model is incentivized to converge toward a *quasi-orthogonal* representational state to minimize cross-feature noise that cannot be effectively mitigated by low-precision error correction. This lower overlap suggests that while low-precision models may sacrifice total feature capacity (superposition density), they gain representational robustness by enforcing clearer geometric separation in the hidden space.

Architectural Influence on Superposition Regimes. Beyond numerical precision, the underlying architecture significantly dictates the evolution of the superposition state. Our comparison between the Qwen (Transformer-based) and GLA (Gated Linear Attention) architectures reveals distinct geometric trajectories (Fig. 31, Right).

The GLA architecture maintains a substantially lower weight overlap compared to the standard Transformer (Qwen-1.7B), even when trained under identical precision (BF16). This phenomenon suggests that the gating mechanisms and linear recurrence in GLA impart a structural inductive bias toward *feature decoherence*. Unlike the global receptive field of `Softmax Attention`, which often leads to "cluttered" hidden spaces with high overlap to maximize associative capacity, the gated recurrent nature of GLA may necessitate more discrete, independent feature tracking.

Implications for Scaling and Capacity. The observed dynamics have profound implications for neural scaling laws. According to the theory that loss scales inversely with model dimension

($L \propto 1/m$) due to geometric interference, the lower overlap in NVFP4 and GLA indicates a different coefficient in the scaling function. Specifically, the lower overlap in NVFP4 suggests that quantization noise forces the model into a *resolution-limited* regime earlier than its high-precision counterparts. However, the stability of the overlap magnitude in NVFP4 during late-stage training implies that the model reaches an equilibrium between capacity and precision-induced interference. For future large-scale deployments, these results suggest that low-precision formats like NVFP4 may inherently favor architectures with higher dimensions but lower superposition density, shifting the optimal balance of the width-depth trade-off in the scaling law.

E.10 Quantization Error Dynamics

To comprehensively understand the behavior of NVFP4 quantization across different attention mechanisms, we conduct a systematic analysis of quantization errors during training. We select GLA-1.3B as the representative model for `Linear Attention` architectures and Qwen-1.7B as the representative model for traditional `Softmax Attention` mechanisms. Throughout training, we monitor the mean squared error (MSE) between full-precision and quantized values for both weights and activations across all linear projection layers.

Fig. 32 presents the quantization error evolution across different components during training. Our analysis reveals two critical observations that characterize the quantization dynamics under NVFP4:

Observation 1: Magnitude Hierarchy of Quantization Errors. Across all linear projection layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`) in both models, we consistently observe that activation quantization errors are approximately 1–2 orders of magnitude larger than weight quantization errors. Specifically, activation MSE ranges from 0.08 to 0.24, while weight MSE remains between 2.4×10^{-3} and 4.4×10^{-3}. This disparity stems from the inherently dynamic nature of activations, which exhibit greater absolute value ranges and more pronounced distributional variations compared to the relatively static weight parameters. The limited representational capacity of FP4 (only 16 discrete values) introduces larger quantization step sizes when accommodating the wider dynamic range of activations, even with dynamic scaling mechanisms.

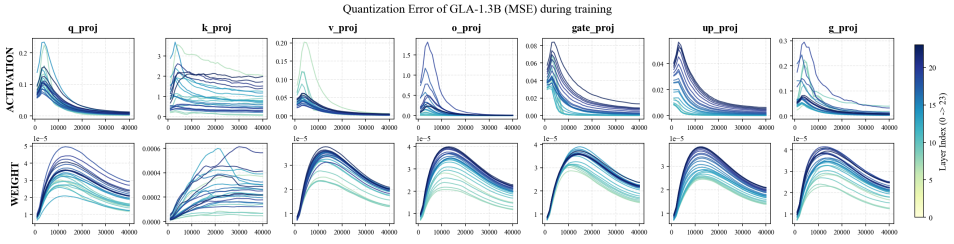
Observation 2: Non-monotonic Training Dynamics. Both weight and activation quantization errors exhibit a characteristic non-monotonic pattern during early training phases. The errors initially increase from 0 to approximately 1,000 training steps, reach a peak around 15,000 steps, and then sharply decrease and stabilize. This phenomenon can be attributed to the interaction between gradient dynamics and quantization precision constraints. During the initial training phase, aggressive gradient updates cause rapid expansion of parameter dynamic ranges as the model transitions from random initialization toward meaningful feature representations. The scaling factors in NVFP4 dynamically adjust to cover these expanding ranges, inevitably increasing quantization step sizes and thus errors. As training progresses beyond 15,000 steps, the model enters a convergence regime where weight updates diminish, distributions stabilize, and normalization layers (`LayerNorm`/`RMSNorm`) learn appropriate scaling parameters. Furthermore, the quantization-aware training process enables the model to implicitly adapt its representations to be more amenable to low-precision quantization, effectively suppressing outliers and aligning activation distributions with the nonlinear representational characteristics of FP4.

These observations demonstrate that despite the extreme precision constraints of NVFP4, the quantization errors remain bounded and exhibit healthy convergence behavior, validating the effectiveness of our quantization-aware training approach across both Linear and Softmax Attention architectures.

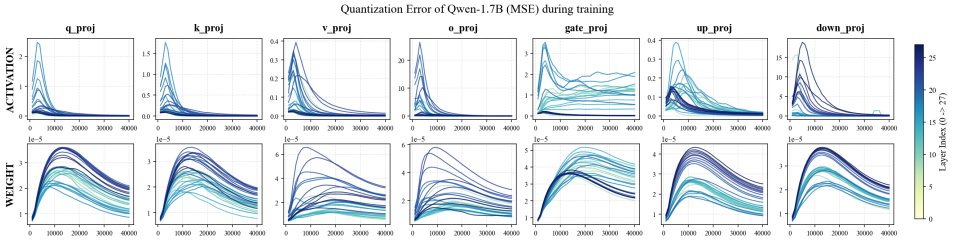
F Discussions

On Downstream Score Reversals under NVFP4. In Tab. 1 and Tab. 8, NVFP4 or CHON occasionally outperforms BF16 on individual zero-shot benchmarks. We clarify that these reversals do not indicate a systematic advantage of lower-precision training over full precision. Instead, we attribute them to two plausible factors:

1. **Zero-shot evaluation variance.** Many zero-shot benchmarks have limited test-set sizes, and accuracy is computed from binary per-example correctness. As reported alongside



(a) GLA-1.3B



(b) Qwen-1.7B

Figure 32: Quantization error dynamics during NVFP4 training for (a) GLA-1.3B Linear Attention model and (b) Qwen-1.7B Softmax Attention model. Each subplot shows MSE between full-precision and quantized values for both activations (top row) and weights (bottom row) across different linear projection layers. Colors indicate different transformer layers (layer 0–23). Both models exhibit consistent patterns: ① activation errors are 1–2 orders of magnitude larger than weight errors; ② errors initially increase in the first 1k steps, peak around 15k steps, then sharply decrease as training stabilizes.

each result in our tables, the per-task standard errors are often on the order of 1–2 absolute points. The observed differences between NVFP4/CHON and BF16 (typically 0.1–1.5%) are comparable to or smaller than these error bars, and should therefore be interpreted with caution rather than as meaningful gains.

2. **Structured noise from SR and RHT.** Stable FP4 training relies on stochastic rounding (SR) and random Hadamard transforms (RHT) to maintain unbiased gradient estimates NVIDIA et al. (2025); Tseng et al. (2025). These mechanisms inject structured randomness into the optimization trajectory, which may act as a form of implicit regularization. While we have not isolated this effect through controlled experiments, the hypothesis is consistent with the broader literature on noise-induced generalization benefits Fan et al. (2021); Rosca (2023). NVFP4 training is therefore not simply a lower-precision approximation of BF16; it also introduces stochastic perturbations whose downstream effects are difficult to predict on a per-task basis.

We note that similar task-level reversals have also been reported in several recent FP4 studies under different models, recipes, and evaluation setups Cao et al. (2025); Tseng et al. (2025); Wang et al. (2025), suggesting that such fluctuations are a common characteristic of low-bit training rather than an artifact of any particular method.

On the Quantization-Friendliness of Qwen3-Next. Based on our internal analysis of the architectural design, we find that Qwen3-Next is inherently more suitable for quantization than its predecessors. The evidence for this improved quantization-friendliness is as follows:

1. **QK Normalization:** QK-Norm constrains the dynamic range of queries and keys, effectively suppressing the formation of extreme outliers in the attention scores. Empirical data show that Qwen3 exhibits significantly lower kurtosis than Qwen2.5, indicating a lighter-tailed activation distribution that is easier to map to lower-bit integers.
2. **Linear Attention Dynamics:** The adoption of Gated DeltaNet, a form of linear attention, alleviates numerical instability. Unlike traditional Softmax attention, which can amplify

outliers exponentially, the linear transition in DeltaNet maintains a more uniform activation scale.

3. **Robustness of Gated Mechanisms:** The gated-attention architecture acts as a natural regulator of feature magnitudes. By adaptively scaling information flow, the gating units mitigate the impact of high-variance features, making the model more robust to the precision loss induced by quantization.
4. **Regularized RMSNorm Parameters:** We observe that the gain parameter (γ) of RMSNorm in Qwen3-Next is generally smaller than 1, achieved by applying weight decay specifically to the RMSNorm parameters during training. This constraint yields lower and more stable activation magnitudes, directly reducing quantization error in activation-sensitive layers.

On the NVIDIA NVFP4 Recipe

1. **About Random Hadamard Transform.** Random Hadamard transforms (RHTs) mitigate outliers by redistributing activation magnitudes, reducing quantization error and training loss. A key distinction separates the NVIDIA NVFP4 recipe used here from MXFP4 approaches such as Quartet. Quartet applies the transform in both forward and backward passes to avoid overflow in MXFP4, which lacks a global scaling factor, and typically relies on a fused fast Walsh–Hadamard transform (FWHT) implementation. In contrast, our NVFP4-based CHON recipe uses a forward-free Hadamard design: NVFP4's hierarchical microscaling accommodates dynamic-range variation without forward-pass rotation. This improves deployability by eliminating FWHT overhead and fusion complexity during inference. Restricting the transform to the backward pass therefore preserves hardware efficiency and enables standard deployment without de-rotation.
2. **About the Last-4 Layer Strategy.** Our analysis shows that top-K activation magnitudes increase monotonically during training. We attribute this to (1) RMSNorm's structural amplification and (2) cumulative outlier excitation across layers. Monitoring indicates that the largest activation spikes consistently occur in the final layers, making them most vulnerable to quantization-induced precision collapse. Quantization difficulty therefore increases with depth, motivating the "Last4" strategy (keeping the last four layers in BF16). This is not a heuristic: it reflects the observed magnitude evolution in the Transformer and protects the most volatile late-stage residual updates, preserving representational capacity during convergence.
3. **About Stochastic Rounding in the Backward Pass.** Unbiased gradients are crucial for stable low-bit optimization. Although MXFP4 training relies on stochastic rounding (SR) for unbiasedness, applying SR directly can add destabilizing variance. We find that the randomized Hadamard transform (RHT) stabilizes SR in the backward pass by scrambling gradients toward a more uniform distribution, reducing localized rounding variance. Together, RHT+SR yields unbiased gradients with controlled variance, which is key to narrowing the FP4–BF16 pretraining gap.