ADAHOP: FAST AND ACCURATE LOW-PRECISION TRAINING VIA OUTLIER-PATTERN-AWARE ROTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Hadamard transforms are increasingly used in MXFP4 LLM training to mitigate outlier-induced quantization error by redistributing magnitude within scaling blocks. We identify a key limitation of this approach: its effectiveness depends on the orientation of the outlier structure relative to the MXFP4 scaling axis. Outliers perpendicular to this axis create within-block concentration that Hadamard mixing can effectively reduce, whereas outliers aligned with it inflate the shared block scales, leaving a scale-driven error component that rotation does not directly address. We call this the *scaling-axis blind spot*. Measurements of LLM training tensors reveal stable Row-wise, Column-wise, and None outlier patterns whose combinations vary across GEMMs; notably, 97.8–100% of weight-gradient GEMMs contain a blind-spot operand. We introduce **AdaHOP**, an outlier-pattern-aware MXFP4 training framework that identifies these patterns in a short calibration phase, uses Inner Hadamard Transform (IHT) as the low-precision backbone, and selectively extracts only the blind-spot outlier rows or columns into a small BF16 path. With fused Triton kernels, AdaHOP-Lv1 executes less than 1% of GEMM FLOPs in BF16 while trailing BF16 by only 0.23 points in average zero-shot accuracy across models from 1B to 8B. On Llama3.1-8B, it reduces memory by $3.6\times$ and accelerates end-to-end training by $1.45\times$ over BF16.

1 INTRODUCTION

Low-Precision Training (LPT) replaces the BF16 matrix multiplications that dominate LLM training with hardware-efficient formats such as FP8 or MXFP4 (Chen et al., 2025; Rouhani et al., 2023), reducing memory and increasing throughput. Its main challenge is quantization error from outliers, which can dominate the quantization range. MXFP4 limits this effect by sharing one scale among 32 consecutive elements, while recent methods further apply Hadamard transforms to redistribute outlier energy before quantization (Ashkboos et al., 2024; Liu et al., 2024; Ashkboos et al., 2025; Tseng et al., 2025). Because these transforms must cancel inside the matrix multiplication, they are typically applied along the same reduction axis on which MXFP4 forms its scaling blocks. This raises a basic question: *are all outliers equally amenable to rotation along this axis?*

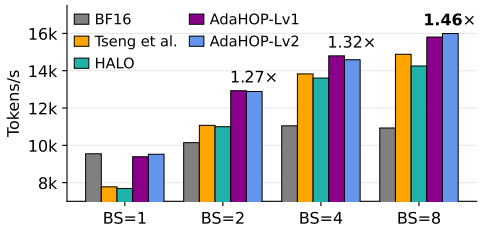
We show that the answer depends on the orientation of the outlier structure relative to the MXFP4 scaling axis. Outliers perpendicular to the axis create strong imbalance within scaling blocks, which Hadamard mixing can effectively smooth. Outliers parallel to the axis instead enlarge the shared block scales while leaving the normalized blocks relatively balanced, producing a scale-driven error that rotation does not directly address. We call this regime the *scaling-axis blind spot* (Figure 2). In LLM training, weights, activations, and gradients exhibit recurring **Row-wise**, **Column-wise**, and **None** patterns, whose combinations vary substantially across GEMMs. Notably, 97.8–100% of the measured weight-gradient GEMMs contain at least one blind-spot operand, leaving a scale-driven error component that uniform scaling-axis rotation cannot directly address.

This observation motivates **AdaHOP**, an outlier-pattern-aware framework for MXFP4 training. A short calibration phase identifies the outlier pattern of each GEMM operand, exploiting its observed stability over training. AdaHOP applies the Inner Hadamard Transform (IHT) as the default low-precision treatment and invokes *Outlier Extraction* (OE) only for blind-spot operands. The key idea is to let IHT smooth within-block imbalance, while OE removes the scale-dominated component that rotation cannot reduce. Specifically, OE isolates the few dominant outlier rows or columns from

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

Method	Mem.	Speed	Accel.	BF16	Quality
BF16	76.00	10930	1.00	100	✓
MXFP4	20.19	17640	1.61	0	✗
MXFP4+Had.	20.60	16305	1.49	0	✗
Tseng et al.	20.60	14881	1.36	0	▲
HALO	20.60	14251	1.30	0	▲
AdaHOP-Lv1	**20.94**	15804	**1.45**	0.45	✓
AdaHOP-Lv2	**28.04**	15992	**1.46**	5.99	✓

(a) End-to-end efficiency at `BS=8`.



(b) Throughput per batch size.

Figure 1: AdaHOP keeps BF16-level quality while retaining most of the speed and memory benefit of MXFP4 on Llama3.1-8B training with `seq_len=4096`. (a) Memory (GB), speed (tokens/s), and acceleration (×) cover the full training pipeline, and BF16 (%) is the share of GEMM FLOPs in BF16. Quality: ✓ within 0.3pt of BF16, ▲ within 1pt, ✗ otherwise, based on the average downstream score across the five models from 1B to 8B. (b) Numbers give the speedup of AdaHOP-Lv2 over BF16, which grows with the batch size.

the MXFP4 path and computes only their contribution in BF16, leaving the rest of the GEMM in MXFP4. Because the extracted component forms a thin dense GEMM, our fused Triton kernels on AMD CDNA4[1] execute the BF16 and MXFP4 paths concurrently with little additional overhead.

Across five models from 1B to 8B parameters, AdaHOP-Lv1 trails BF16 by only 0.23 points in average zero-shot accuracy, compared with 0.74 points for HALO, while executing less than 1% of GEMM FLOPs in BF16. On Llama3.1-8B, it reduces memory by $3.6\times$ and accelerates end-to-end training by $1.45\times$ over BF16 (Figure 1). Ablations further show that the improvement comes from pattern-aware assignment rather than simply adding high-precision computation.

Our contributions are threefold: **(1)** we identify and characterize the *scaling-axis blind spot* of Hadamard-based MXFP4 training; **(2)** we introduce a GEMM-level, pattern-aware strategy combining IHT with selective OE for blind-spot operands; and **(3)** we develop fused mixed-precision kernels that overlap the BF16 outlier path, enabling this selective treatment with low runtime overhead.

2 Related Work

LPT reduces precision to FP8 (Peng et al., 2023; Xi et al., 2024a;b; Micikevicius et al., 2022), INT4 (Wang et al., 2022; Chmiel et al., 2022; 2023; Schiemer et al., 2023), or MXFP4 (Tseng et al., 2025; Egiazarian et al., 2025; Chmiel et al., 2025; Castro et al., 2025). Lower bit-widths make training more sensitive to outliers, since a few extreme values dominate the quantization range and amplify rounding errors for the remaining values (Dettmers et al., 2022; Xiao et al., 2023a). Outlier-aware scaling (Xiao et al., 2023a; Lin et al., 2024) and rotation-based methods act on the outliers themselves, and the Hadamard-based methods below treat the microscaling block (Rouhani et al., 2023) as a given, asking how to spread outliers before quantization.

For inference, QuaRot, SpinQuant, QuIP, QuIP#, and FlatQuant (Ashkboos et al., 2024; Liu et al., 2024; Chee et al., 2023; Tseng et al., 2024; Sun et al., 2025) apply Hadamard or learned rotations to spread outlier energy across channels. Recent LPT methods carry the idea to training, where gradients add a third tensor type: HOT (Kim et al., 2025) applies Hadamard quantization with a low-rank approximation to the backward GEMMs only; Tseng et al. (2025) combine random Hadamard transforms with stochastic rounding for unbiased MXFP4 gradients; Quartet (Castro et al., 2025) uses fixed transforms in the forward pass and randomized ones in the backward pass; FP4 All the Way (Chmiel et al., 2025) reverts the backward pass to BF16 during fine-tuning; and HALO (Ashkboos et al., 2025) places rotations according to the row-wise outliers of gradients and the column-wise outliers of activations. All of these methods fix the transform assignment at the tensor or path level, so they cannot follow outlier structures that vary across the operand pairs of different matrix multiplications. AdaHOP decides per operand pair and treats the transform choice as a measurable property of the training tensors.

[1]The CDNA4 architecture is utilized on AMD Instinct™ MI350 Series GPUs.

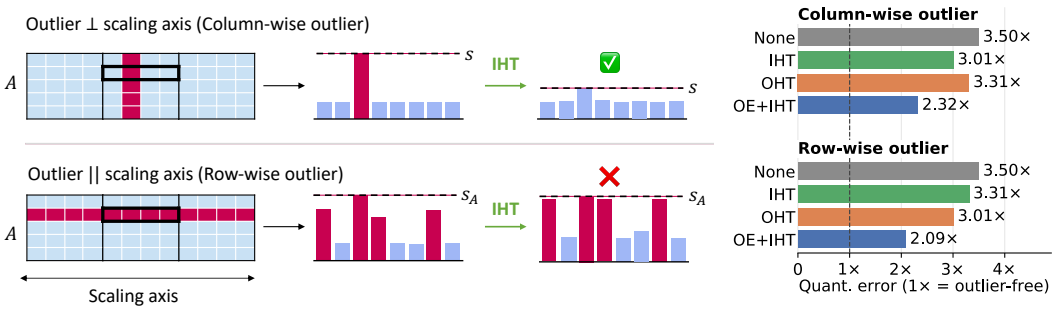


Figure 2: Outlier orientation relative to the MXFP4 scaling axis decides whether IHT is effective. s and s_A denote the shared MXFP4 scale of a block of A, set by its largest magnitude. (Top) For a Column-wise outlier, a single large value sets s, and IHT spreads it evenly over the block, lowering s, so IHT is effective. (Bottom) For a Row-wise outlier, several values in each block are large, so IHT dilutes them only partially and leaves s_A high. (Right) Quantization error of Llama3.2-3B's `layers.15.attention.wv`.

3 Preliminaries: MXFP4 Training with Hadamard Transforms

For a linear layer with input activation $X \in \mathbb{R}^{b\times d_{\text{in}}}$, weight $W \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$, and output gradient $G_Y \in \mathbb{R}^{b\times d_{\text{out}}}$, training requires three matrix multiplications: the forward $Y = XW^\top$, the weight gradient $G_W = G_Y^\top X$, and the input gradient $G_X = G_Y W$. We write each as $C = AB$, where $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$. The shared dimension k is the *reduction axis*, and it is central to both MXFP4 quantization and the Hadamard transforms used in low-precision training.

MXFP4 partitions this reduction axis into blocks of $b = 32$ consecutive elements. Each block shares a single E8M0 scale determined by its largest magnitude, and the elements are represented in FP4 relative to that scale. This makes quantization particularly sensitive to *outliers*, i.e., values whose magnitudes are much larger than others in the same block: a single outlier can determine the shared scale and thereby coarsen the quantization grid for all other values in that block.

A natural way to mitigate this outlier-induced imbalance is to redistribute magnitudes within each scaling block before quantization. Hadamard-based training methods implement exactly this idea through orthogonal mixing. Let $\mathcal{H}_b \in \mathbb{R}^{k\times k}$ denote a block-diagonal operator that applies a normalized b-point Walsh–Hadamard transform independently within each MXFP4 scaling block. The *Inner Hadamard Transform* (IHT) computes

$$C_{\text{IHT}} = Q(A\mathcal{H}_b)\, Q(\mathcal{H}_b^\top B), \tag{1}$$

which recovers AB in infinite precision because $\mathcal{H}_b\mathcal{H}_b^\top = I$. IHT therefore mixes the columns of A and the rows of B exactly along the reduction axis where MXFP4 forms its scaling blocks. Because the two transforms cancel inside the product, IHT can be inserted without changing the output representation, making it a natural choice for low-precision training. Prior Hadamard-based LPT methods exploit this property by applying reduction-axis mixing uniformly or to predefined computation paths, thereby spreading outlier energy within MXFP4 blocks before quantization (Kim et al., 2025; Yang et al., 2023).

A complementary option is the *Outer Hadamard Transform* (OHT), which mixes the rows of A and the columns of B. Unlike IHT, these transforms act on the outer dimensions of the product and therefore do not cancel inside the GEMM; the corresponding rotation must be undone outside the multiplication. IHT and OHT thus redistribute outliers along two orthogonal directions, whereas MXFP4 always forms its scaling blocks along the reduction axis.

This asymmetry raises a natural question: *does the effectiveness of rotation depend on how outliers are oriented relative to the MXFP4 scaling blocks?* In the next section, we show that it does, and identify the outlier structures for which rotation alone is insufficient.

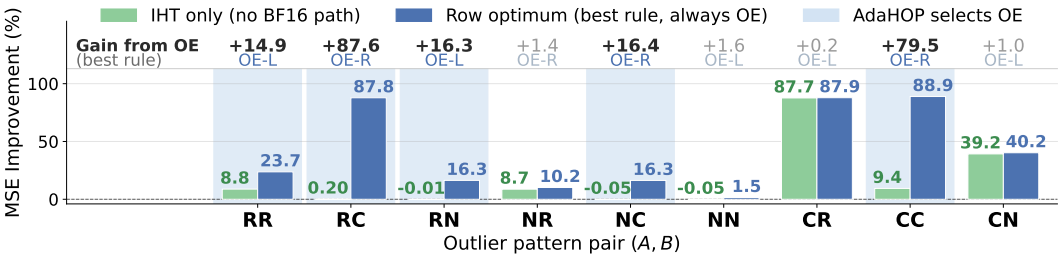


Figure 3: The BF16 outlier path pays off on the five pattern pairs in the blind spot. Bars give the MSE improvement (%) over untransformed MXFP4 on Llama3.2-3B's `layers.15.attention.wv` for IHT alone and for the row optimum, which is always an OE rule, and the top row gives their difference with the rule that attains it. Shading marks the pairs on which AdaHOP selects OE. **RR**, **NR**, and **CR** are not observed in training (Table 1) but are shown for completeness.

4 Analysis: The Scaling-Axis Blind Spot

Section 3 showed that MXFP4 forms its scaling blocks along the reduction axis, and that IHT mixes values along the same direction. We now show that the effectiveness of this scaling-axis rotation depends on how outliers are oriented relative to the axis. Outliers perpendicular to the scaling axis create within-block imbalance that IHT can reduce, whereas outliers parallel to the axis enlarge the shared block scales, leaving a scale-driven component that IHT does not directly address. This distinction determines when scaling-axis rotation is sufficient and when an additional treatment is needed.

4.1 From Outlier Orientation to the Scaling-Axis Blind Spot

Consider the left operand A, whose MXFP4 scaling blocks lie within its rows. Let $\gamma_b(A) \in [1, b]$ denote its within-block magnitude concentration, where $\gamma_b = 1$ corresponds to balanced blocks and larger values indicate stronger concentration (Definition 1). Under block-scaled quantization, Lemma 1 gives

$$\|Q(A) - A\|_F \leq \epsilon_q \sqrt{\gamma_b(A)}\, \|A\|_F, \tag{2}$$

where $\epsilon_q = 2^{-2}$ for MXFP4. The bound separates two sources of sensitivity: the magnitude imbalance within scaling blocks, captured by γ_b, and the overall operand energy, captured by $\|A\|_F$.

A Column-wise outlier is perpendicular to the scaling axis of A. Each affected block therefore contains a few unusually large values among otherwise ordinary ones, increasing its within-block concentration $\gamma_b(A)$. Since IHT mixes exactly along this axis, it can redistribute the concentrated energy within each block and substantially reduce the resulting quantization error. A Row-wise outlier is instead parallel to the scaling axis. Each affected block holds many large values rather than a few, so IHT can only partly spread them and the shared MXFP4 scale stays large.

To see how block scales enter the GEMM error, partition the reduction dimension into MXFP4 blocks indexed by β. Write $A_{i,\beta} = s_A(i,\beta)\bar{A}_{i,\beta}$ and $B_{\beta,j} = s_B(\beta,j)\bar{B}_{\beta,j}$, where s_A, s_B are the shared block scales and $\bar{A}, \bar{B}$ are the normalized blocks. Let $\delta_{ij,\beta} = Q(\bar{A}_{i,\beta})Q(\bar{B}_{\beta,j}) - \bar{A}_{i,\beta}\bar{B}_{\beta,j}$ denote the normalized block-dot-product error. Then

$$\widehat{C}_{ij} - C_{ij} = \sum_{\beta} s_A(i,\beta) s_B(\beta,j)\, \delta_{ij,\beta}, \tag{3}$$

Each block's normalized error is therefore weighted by the product of its two shared scales. For a Column-wise outlier, the dominant effect is a large $\delta_{ij,\beta}$ caused by within-block imbalance, which IHT can reduce. For a Row-wise outlier, $\delta_{ij,\beta}$ may already be moderate, but $s_A(i,\beta)$ remains large across many blocks, magnifying the residual block errors in the output. This explains the blind spot of IHT. Scaling a row leaves γ_b unchanged (Proposition 2), while IHT only redistributes values within that row. It can therefore reduce within-block concentration but does not directly alter the large row-wise scale that dominates this case. We refer to this regime as the *scaling-axis blind spot*. The argument is symmetric for a Column-wise outlier of the right operand B. OHT can spread

Table 1: Distribution of observed outlier-pattern pairs across computation paths for three LLM architectures after 30 training steps on C4. Shaded rows contain a blind-spot operand, and **bold** counts show that the weight-gradient path (∇W) falls almost entirely into them.

	Llama3.2-1B				Instella-3B				Llama3.1-8B			
Pattern Pair	Total	Fwd	∇W	∇X	Total	Fwd	∇W	∇X	Total	Fwd	∇W	∇X
RN	23	0	**15**	8	79	0	**33**	46	65	0	**27**	38
RC	69	0	**69**	0	111	0	**111**	0	92	0	**92**	0
NC	20	0	**20**	0	60	0	**60**	0	63	0	**63**	0
CC	8	0	**8**	0	45	0	**45**	0	37	0	**37**	0
CN	181	97	0	84	361	216	1	144	312	192	1	119
NN	35	15	0	20	100	36	2	62	103	32	4	67

these outliers across the outer dimension, but this redistributes rather than removes their energy and requires the resulting rotation to be undone outside the GEMM.

We summarize this geometry using three spatial patterns: **Row-wise (R)**, when a small number of rows carry disproportionately large energy; **Column-wise (C)**, when the same occurs along columns; and **None (N)**, when neither orientation dominates. For $C = AB$, the scaling axis lies along the rows of A and the columns of B. Hence a GEMM contains a blind-spot operand precisely when

$$p_A = \mathrm{R} \qquad \text{or} \qquad p_B = \mathrm{C}, \tag{4}$$

where (p_A, p_B) denotes the operand-pattern pair. The nine possible pairs therefore divide into five with a blind-spot operand, **RR**, **RC**, **RN**, **NC**, and **CC**, and four without one, **CR**, **CN**, **NR**, and **NN**. A pair in the former group may still benefit from IHT on the other operand, but one scale-driven contribution remains beyond what scaling-axis rotation alone can address.

Figure 3 validates this geometric split on real training tensors. Outside the blind spot, IHT nearly matches the best attainable error reduction, with less than 1.6 points of additional improvement from removing dominant rows or columns. On the five blind-spot pairs, however, IHT alone provides little improvement, indicating that a substantial scale-driven component remains beyond scaling-axis rotation. This empirical gap motivates the targeted treatment introduced in Section 5.

4.2 Heterogeneous but Stable Pattern Pairs in LLM Training

We next ask how often these geometries occur in actual LLM training. We identify the Row-wise, Column-wise, or None pattern of each tensor using the normalized coefficient of variation (CV) of its row and column variances (Section B). These patterns are properties of individual tensor locations rather than fixed tensor types. Weights are typically None and activations are often Column-wise, while gradients vary across projections and network depth.

What matters for quantization, however, is not a tensor pattern in isolation but the pair of patterns that meet in a GEMM. In particular, transposition in backpropagation changes a pattern's orientation relative to the scaling axis. Consider the common case in which G_Y and X are Column-wise while W is None. The input-gradient product

$$G_X = G_Y W$$

has pattern pair **CN** and lies outside the blind spot. In contrast, the weight-gradient product

$$G_W = G_Y^\top X$$

has pair **RC**, because transposing G_Y turns its Column-wise structure into Row-wise structure. The same gradient tensor can therefore be aligned with the scaling axis in one GEMM and perpendicular to it in another. This makes the operand pair, rather than the tensor type or computation path, the appropriate unit for reasoning about rotation.

Table 1 shows a pronounced path-dependent distribution. The forward path consists almost entirely of **CN** and **NN**, placing its operands outside the blind spot. The weight-gradient path is qualitatively different: 97.8–100% of its GEMMs contain a blind-spot operand across the three models. The input-gradient path is more heterogeneous, mixing primarily **RN**, **CN**, and **NN**. Across all GEMMs, blind-spot pairs account for 35.7–39.0% of the measured GEMM instances.

This asymmetry explains why a uniform rotation can appear effective on the forward path yet remain insufficient for a substantial fraction of backpropagation. It also rules out a simple path-level policy, since different geometries coexist even within the same computation path.

Despite this spatial heterogeneity, the pattern at a fixed tensor location is highly stable over training. In Llama3.2-3B, patterns calibrated from the first 30 steps agree with those re-detected at steps 100 and 1,000 at 99.8% and 100.0%, respectively, and still agree at 95.8% after 30,000 steps. The structure is therefore *heterogeneous across GEMMs but persistent over time*: different GEMMs may require different treatments, yet their treatment can be determined from a short calibration phase.

Together, these observations identify the GEMM operand pair as the natural unit of adaptation. Its geometry determines whether rotation can address the dominant outlier structure, while its temporal persistence makes this decision inexpensive to obtain. Section 5 turns this principle into a pattern-aware low-precision training algorithm.

5 METHODOLOGY

The analysis above suggests a simple division of labor: IHT handles within-block concentration, while only the blind-spot component requires additional precision. AdaHOP realizes this principle by applying IHT to every training GEMM and selectively adding Outlier Extraction (OE).

5.1 PATTERN-AWARE OUTLIER EXTRACTION

Consider $C = AB$ with calibrated pattern pair (p_A, p_B). AdaHOP applies IHT as the default low-precision treatment, as in Equation (1). For pairs outside the blind spot (**CR**, **CN**, **NR**, and **NN**), IHT alone is sufficient. When A is Row-wise or B is Column-wise, however, the corresponding operand contains a scale-dominated component that rotation cannot directly reduce. AdaHOP therefore extracts only this component from the MXFP4 path and evaluates it in BF16.

For a Row-wise left operand, **OE-Left** decomposes $A = A_{\text{res}} + A_{\text{out}}$, where A_{out} contains the selected outlier rows. Similarly, for a Column-wise right operand, **OE-Right** writes $B = B_{\text{res}} + B_{\text{out}}$, with B_{out} containing the selected outlier columns. The resulting products are

$$\widehat{C}_{\text{OE-L}} = Q(A_{\text{res}}\mathcal{H}_b)Q(\mathcal{H}_b^\top B) + A_{\text{out}}B, \tag{5}$$

$$\widehat{C}_{\text{OE-R}} = Q(A\mathcal{H}_b)Q(\mathcal{H}_b^\top B_{\text{res}}) + AB_{\text{out}}. \tag{6}$$

This decomposition gives IHT and OE complementary roles: IHT reduces within-block imbalance in the residual, while OE removes the few rows or columns responsible for the large block scales from the low-precision path. Only their contribution is computed in BF16; the rest of the GEMM remains in IHT-transformed MXFP4.

The decomposition is also hardware-friendly. Although A_{out} and B_{out} are written above as full-size matrices, only the selected r rows or columns are materialized in practice. The BF16 correction therefore becomes a thin dense GEMM rather than an irregular sparse operation, making it amenable to efficient execution alongside the much larger MXFP4 GEMM.

The remaining question is which operand to extract from. When only one operand lies in the blind spot, AdaHOP extracts from that operand. For **RC**, both operands lie in the blind spot, so AdaHOP chooses the side that removes more energy from the MXFP4 path. We measure this using the residual norm ratio $\eta_r(A) = \frac{\|A_{\text{res}}\|_F}{\|A\|_F}$, and analogously for B, and select the operand with the smaller ratio. For weight-gradient GEMMs, where $A = G_Y^\top$ and $B = X$, activation-column extraction typically gives $\eta_r(B) \approx 0.52$, compared with $\eta_r(A) \approx 0.95$ for gradient-row extraction, and therefore favors OE-Right. The corresponding error bound is given in Section K.

We use $r = 64$ throughout, which is large enough to cover the observed outlier support while remaining small enough for efficient overlap; the error reduction also largely saturates at this

Table 2: AdaHOP treatment of each observed pattern pair. Shaded pairs contain a blind-spot operand, and OE keeps IHT on the residual.

Pair	Lv1	Lv2
RN	OE-L	OE-L
RC	OE-R	OE-R
NC	OE-R	OE-R
CC	OE-R	BF16[†]
CN	IHT	IHT
NN	IHT	IHT

[†] If the post-IHT residual CV exceeds κ.

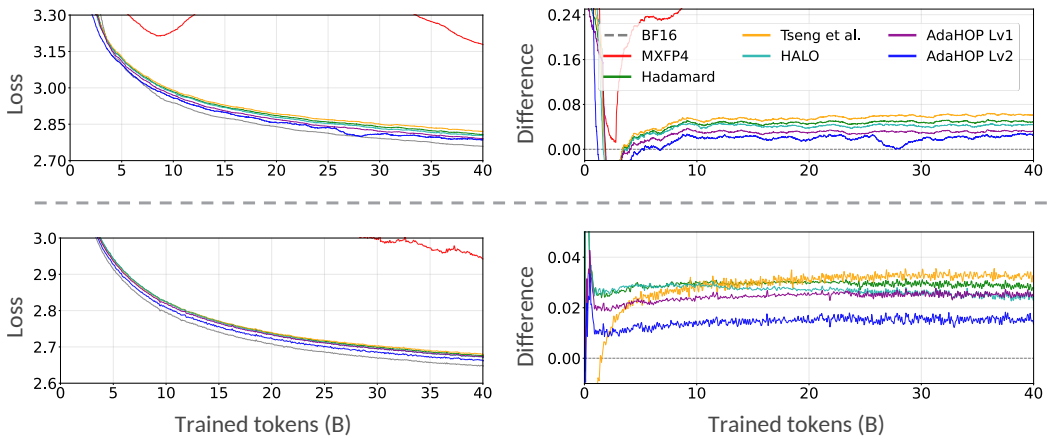


Figure 4: Training loss curves and loss difference relative to BF16 for (Top) Llama3.2-1B and (Bottom) Instella-3B. AdaHOP achieves the lowest loss gap relative to BF16 among all MXFP4-based methods.

point (Section M). Based on this policy, AdaHOP-Lv1 keeps all residual GEMMs in MXFP4, whereas AdaHOP-Lv2 additionally promotes **CC** GEMMs to BF16 when their post-IHT residual CV exceeds a threshold κ (Section N).

5.2 Calibration and Online Adaptation

The temporal persistence observed in Section 4.2 allows AdaHOP to determine its treatment policy from a short calibration phase at the start of training. During the first 30 BF16 steps, it applies the normalized-CV detector of Section 4.2 with $\tau = 2.0$ and assigns each tensor a Row-wise, Column-wise, or None pattern by majority vote. The resulting operand pair then determines whether OE is required and, if so, from which side to extract.

Importantly, calibration fixes only this structural choice; it does not fix the outlier indices themselves. During low-precision training, AdaHOP keeps the selected extraction axis but re-identifies the current top-r rows or columns at every step. This allows it to exploit the temporal stability of outlier orientation while adapting to changes in which individual features become outliers. The 30-step calibration accounts for only 0.026% of the training FLOPs for Llama3.2-3B, with additional sensitivity and transfer results reported in Sections B, F and G.

5.3 Hardware-Aware Implementation

Selective OE introduces an additional BF16 path, so its practical benefit depends on keeping this correction off the critical path. AdaHOP therefore co-designs OE with fused Triton kernels on AMD CDNA4. Fast Outlier Index Detection (FOID) identifies the current top-r rows or columns, after which the residual is transformed with the 32-point IHT and quantized to MXFP4. The resulting MXFP4 residual GEMM and thin BF16 correction GEMM are then executed concurrently within a single mixed-precision kernel, followed by a fused scatter-add.

To hide the correction latency, the dominant MXFP4 path uses large output tiles, while smaller BF16 tiles are interleaved across compute units. With $r = 64$, the BF16 branch remains small enough to overlap with the MXFP4 computation rather than extend its critical path. This hardware-aware execution allows AdaHOP to add selective high-precision correction without giving up the throughput advantage of MXFP4 training. Detailed analyses are provided in Section L.

6 Experimental Results

The preceding analysis argues that low-precision treatment should be chosen from the operand geometry of each GEMM. We now evaluate whether this principle (i) recovers BF16-level training

Table 3: Zero-shot accuracy (%) after training on C4. Best MXFP4-based result is in **bold**; second best is underlined. The Qwen3-0.6B block adds HOT and Quartet.

Model	Tokens	Method	PIQA	HellaSwag	ARC-E	Lambada	Avg.
Qwen3-0.6B	10B	BF16	69.09	38.79	43.85	31.35	45.77
		MXFP4	63.33	32.44	38.38	22.39	39.14
		HOT	64.46	32.78	38.38	24.57	40.05
		Quartet	65.12	34.28	40.23	28.44	42.02
		HALO	65.08	33.90	40.84	26.35	41.54
		AdaHOP-Lv1	**68.45**	**38.33**	**42.13**	**30.01**	**44.73**
Llama3.2-1B	40B	BF16	72.69	52.33	47.85	38.17	52.76
		MXFP4	64.69	37.94	37.29	11.93	37.96
		MXFP4+Had.	71.85	51.37	45.91	37.68	51.70
		Tseng et al.	72.14	50.95	46.17	36.81	51.52
		HALO	72.09	51.89	46.72	**39.03**	52.43
		AdaHOP-Lv1	**73.04**	52.24	46.91	38.21	52.60
		AdaHOP-Lv2	73.01	**52.32**	**47.05**	38.55	**52.73**
Instella-3B	40B	BF16	73.52	54.31	48.91	46.26	55.75
		MXFP4	68.17	42.07	40.91	27.71	44.72
		MXFP4+Had.	72.69	53.54	48.61	43.39	54.56
		Tseng et al.	73.45	53.19	48.86	45.12	55.16
		HALO	72.20	53.54	47.81	**45.90**	54.86
		AdaHOP-Lv1	73.23	53.32	49.41	45.02	55.25
		AdaHOP-Lv2	**73.83**	**54.26**	**50.67**	45.42	**56.05**
Llama3.2-3B	60B	BF16	73.72	57.75	48.53	47.53	56.88
		MXFP4	55.88	26.30	29.25	6.09	29.38
		HALO	73.45	**57.34**	47.98	**47.95**	56.68
		AdaHOP-Lv1	**73.67**	57.18	48.44	47.41	56.68
		AdaHOP-Lv2	**73.67**	57.14	**49.12**	47.28	**56.80**
Qwen3-4B	80B	BF16	75.30	62.77	54.63	52.05	61.19
		Tseng et al.	75.30	61.90	**54.67**	50.26	60.53
		HALO	74.46	62.22	52.57	51.68	60.23
		AdaHOP-Lv1	**75.68**	62.36	53.75	**52.00**	**60.95**
		AdaHOP-Lv2	74.68	**63.38**	53.70	50.94	60.68
Llama3.1-8B	160B	BF16	75.79	64.19	57.11	49.64	61.68
		HALO	74.98	63.17	55.31	48.04	60.37
		AdaHOP-Lv1	75.43	64.94	**56.31**	**49.84**	**61.63**
		AdaHOP-Lv2	**76.28**	**64.98**	55.89	48.57	61.43

quality across model scales, (ii) preserves the efficiency advantage of MXFP4 despite the selective high-precision path, and (iii) provides gains because of pattern-aware assignment rather than simply additional BF16 computation.

We implement AdaHOP on Torchtitan (Liang et al., 2025) and train six models on C4 (Raffel et al., 2020): Qwen3-0.6B and Qwen3-4B (Yang et al., 2025), Llama3.2-1B, Llama3.2-3B, and Llama3.1-8B (Grattafiori et al., 2024), and Instella-3B (Liu et al., 2025). We compare against BF16, naive MXFP4, MXFP4+Hadamard, Tseng et al. (2025), and HALO (Ashkboos et al., 2025), and additionally against HOT (Kim et al., 2025) and Quartet (Castro et al., 2025) on Qwen3-0.6B. All baselines are reimplemented in the same Torchtitan framework with the same MXFP4 format, scaling policy, and fused-kernel infrastructure. We report training loss, zero-shot accuracy on PIQA, HellaSwag, ARC-Easy, and LAMBADA (Bisk et al., 2020; Zellers et al., 2019; Clark et al., 2018; Paperno et al., 2016), memory, and throughput; full settings and implementation details are given in Sections H and I.

6.1 Q1. Does AdaHOP recover BF16-level training quality?

We first examine optimization behavior. Figure 4 reports both training loss and its difference from BF16 for models trained from scratch. AdaHOP consistently stays closest to the BF16 trajectory among MXFP4-based methods, with a final loss gap below 0.01 on both Llama3.2-1B and Instella-3B. In contrast, uniform or path-level Hadamard methods reduce the gap only partially, consistent with the blind-spot GEMMs left untreated by a fixed assignment.

The improvement persists in downstream accuracy. Across all six models in Table 3, AdaHOP achieves the strongest average zero-shot accuracy among the evaluated MXFP4-based methods. For

a consistent comparison across the five models from 1B to 8B, AdaHOP-Lv1 trails BF16 by only 0.23 points on average, compared with 0.74 points for HALO. The difference is especially clear on Llama3.1-8B: AdaHOP-Lv1 is only 0.05 points below BF16, reducing the 1.31-point gap of HALO by 96%. On Qwen3-0.6B, where HOT and Quartet are also evaluated, AdaHOP-Lv1 improves all four downstream tasks and reduces the average BF16 gap from 3.75 points for Quartet to 1.04 points. Together, these results show that the lower training loss translates into consistent recovery of downstream quality rather than merely improved optimization statistics.

6.2 Q2. Does selective OE preserve the efficiency of MXFP4 training?

Recovering BF16 quality is useful only if the selective high-precision path does not give up the efficiency benefit of MXFP4. Figure 1b shows end-to-end throughput on Llama3.1-8B, where AdaHOP's speedup increases with batch size, while Figure 1a reports the corresponding memory, throughput, and precision composition at BS=8.

AdaHOP-Lv1 reduces memory by 3.6× and accelerates end-to-end training by 1.45× over BF16 while remaining within 0.3 points of BF16 quality. It also runs 11% faster than HALO with only 1.7% more memory. Crucially, this quality recovery requires only 0.45% of GEMM FLOPs in BF16 on Llama3.1-8B; all of them belong to the thin OE correction path, while the remaining 99.55% stay in MXFP4. AdaHOP-Lv2 reaches a similar 1.46× end-to-end speedup, but its selective full-BF16 fallback for CC pairs increases the BF16 share to 5.99%.

The hardware-aware implementation keeps the small correction off the critical path. On representative Llama3.1-8B MLP and attention projections, the fused AdaHOP kernels are 2.13× and 1.26× faster than BF16, respectively (Table 12). The smaller end-to-end speedup therefore comes primarily from non-GEMM work in the training pipeline, rather than from the OE correction itself.

6.3 Q3. Is the gain due to pattern-aware assignment?

The previous results establish the quality–efficiency trade-off; we next isolate what produces the quality gain. Table 10 first compares AdaHOP with a count-matched random control on Qwen3-0.6B. Random OE applies extraction to the same number of GEMMs as AdaHOP-Lv1 (250 of 588), but places them independently of the operand patterns. Its BF16 loss gap is 0.0447, close to HALO and Quartet, whereas AdaHOP reaches 0.0041. Thus, simply adding an outlier path to a similar number of GEMMs does not reproduce the gain; where it is applied matters.

The component ablation on Llama3.2-1B gives the same conclusion. Replacing uniform IHT with pattern-aware IHT, without adding BF16 computation, reduces the loss gap from 0.0429 to 0.0233. OE without the pattern-aware rotation policy gives a larger 0.0549 gap, whereas combining the two reduces it further to 0.0167. The benefit is also not specific to AdaHOP: applying the same pattern-aware assignment to the components of Tseng et al. (2025) reduces its gap by 5.3×, from 0.0435 to 0.0082. Pattern information therefore improves the placement of low-precision treatments themselves, rather than compensating for a particular kernel or a particular choice of OE.

Finally, the improvement is larger than the observed seed variation. Over three Qwen3-0.6B runs, AdaHOP-Lv1 trails BF16 by 0.0175 in loss, compared with 0.0542 for HALO; their 0.0367 difference is more than four times the largest across-seed standard deviation of 0.0088. These short runs are used only to separate the methods from seed noise, not to estimate their converged absolute margin. Loss scaling (Micikevicius et al., 2018) and stochastic rounding (Gupta et al., 2015; Chmiel et al., 2023) provide much smaller improvements under the same MXFP4 setup, further indicating that the dominant gain comes from matching the treatment to the operand geometry.

7 Conclusion

We showed that Hadamard rotation in MXFP4 training has a scaling-axis blind spot: rotation cannot remove the error of outliers parallel to the scaling axis. AdaHOP therefore applies IHT where rotation suffices and allocates BF16 computation only to the scale-dominated outlier component. From 1B to 8B, AdaHOP-Lv1 stays within 0.23 points of BF16 average zero-shot accuracy with less than 1% of GEMM FLOPs in BF16 and a 1.45× end-to-end speedup, showing that low-precision training benefits from matching numerical precision to the geometry of the error.

AI USAGE STATEMENT

We used generative AI tools to refine material we had already produced: assisting with translation, refining hypotheses, and providing feedback on our research methodology and experiments. We wrote the methodology, the manuscript draft in this paper ourselves. We reviewed every suggestion before adopting it and take responsibility for all claims, numbers, and artifacts in this paper.

REFERENCES

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. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Saleh Ashkboos, Mahdi Nikdan, Soroush Tabesh, Roberto L Castro, Torsten Hoefler, and Dan Alistarh. Halo: Hadamard-assisted lower-precision optimization for llms. *arXiv preprint arXiv:2501.02625*, 2025.

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

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.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in neural information processing systems*, 36: 4396–4429, 2023.

Mengzhao Chen, Meng Wu, Hui Jin, Zhihang Yuan, Jing Liu, Chaoyi Zhang, Yunshui Li, Jie Huang, Jin Ma, Zeyue Xue, et al. Int vs fp: A comprehensive study of fine-grained low-bit quantization formats. *arXiv preprint arXiv:2510.25602*, 2025.

Brian Chmiel, Ron Banner, Elad Hoffer, Hilla Ben-Yaacov, and Daniel Soudry. Accurate neural training with 4-bit matrix multiplications at standard formats. In *The Eleventh International Conference on Learning Representations*, 2022.

Brian Chmiel, Ron Banner, Elad Hoffer, Hilla Ben-Yaacov, and Daniel Soudry. Accurate neural training with 4-bit matrix multiplications at standard formats. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=yTbNYYcopd.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Vage Egiazarian, Roberto L Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, et al. Bridging the gap between promise and performance for microscaling fp4 quantization. *arXiv preprint arXiv:2509.23202*, 2025.

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

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pp. 1737–1746. PMLR, 2015.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Seonggon Kim, Juncheol Shin, Seung-taek Woo, and Eunhyeok Park. Hot: Hadamard-based optimized training. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 4787–4796, 2025.

Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, et al. Torchtitan: One-stop pytorch native solution for production ready llm pretraining. In *International Conference on Learning Representations*, volume 2025, pp. 92161–92181, 2025.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024.

Jiang Liu, Jialian Wu, Xiaodong Yu, Yusheng Su, Prakamya Mishra, Gowtham Ramesh, Sudhanshu Ranjan, Chaitanya Manem, Ximeng Sun, Ze Wang, Pratik Prabhanjan Brahma, Zicheng Liu, and Emad Barsoum. Instella: Fully open language models with stellar performance. *arXiv preprint arXiv:2511.10628*, 2025.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *International Conference on Learning Representations*, 2018.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, et al. Fp8-lm: Training fp8 large language models. *arXiv preprint arXiv:2310.18313*, 2023.

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

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

Martin Schiemer, Clemens JS Schaefer, Jayden Parker Vap, Mark James Horeni, Yu Emma Wang, Juan Ye, and Siddharth Joshi. Hadamard domain training with integers for class incremental quantized learning. *arXiv preprint arXiv:2310.03675*, 2023.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. In *International Conference on Machine Learning*, pp. 57587–57613. PMLR, 2025.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235:48630, 2024.

Albert Tseng, Tao Yu, and Youngsuk Park. Training llms with mxfp4. In *International Conference on Artificial Intelligence and Statistics*, pp. 1630–1638. PMLR, 2025.

Maolin Wang, Seyedramin Rasoulinezhad, Philip HW Leong, and Hayden K-H So. Niti: Training integer neural networks using integer-only arithmetic. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):3249–3261, 2022.

Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. Coat: Compressing optimizer states and activation for memory-efficient fp8 training. *arXiv preprint arXiv:2410.19313*, 2024a.

Haocheng Xi, Yuxiang Chen, Kang Zhao, Kai Jun Teh, Jianfei Chen, and Jun Zhu. Jetfire: Efficient and accurate transformer pretraining with int8 data flow and per-block quantization. In *International Conference on Machine Learning*, pp. 54049–54063. PMLR, 2024b.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023a.

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

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

Yuedong Yang, Hung-Yueh Chiang, Guihong Li, Diana Marculescu, and Radu Marculescu. Efficient low-rank backpropagation for vision transformer adaptation. *Advances in Neural Information Processing Systems*, 36:14725–14736, 2023.

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.

A Limitations and Future Work

The evaluation uses a single dataset (C4) and AMD hardware only. The main results are single runs, because one Llama3.1-8B run on 160B tokens takes about 4,400 GPU-hours on 8 MI350X GPUs; the multi-seed evidence of Q3 is limited to 0.6B parameters and the component ablation to 1B. The comparison with Quartet is limited to quality, since its released kernel targets another vendor, and scales beyond 8B, MoE architectures (Jiang et al., 2024), and different normalization schemes (Team et al., 2024) are unverified. The calibration assumption has stated failure modes, namely activation patterns that depend on the training state, tensors with simultaneous row- and column-wise concentration, and outliers confined to a segment of a row, in each of which we expect a reduced gain rather than a collapse.

B Calibration Algorithm Details

Outlier Pattern Detection via CV. For a given tensor $T \in \mathbb{R}^{m \times n}$, we first compute the per-row and per-column variance vectors:

$$v_i^{\text{row}} = \text{Var}(T_{i,:}), \quad i = 1, \ldots, m, \tag{7}$$

$$v_j^{\text{col}} = \text{Var}(T_{:,j}), \quad j = 1, \ldots, n. \tag{8}$$

We then measure how non-uniformly these variances are distributed by computing the coefficient of variation (CV) of each variance vector:

$$\text{CV}_{\text{row}} = \frac{\text{std}(v^{\text{row}})}{\text{mean}(v^{\text{row}}) + \epsilon}, \tag{9}$$

$$\text{CV}_{\text{col}} = \frac{\text{std}(v^{\text{col}})}{\text{mean}(v^{\text{col}}) + \epsilon}, \tag{10}$$

where ϵ is a small constant for numerical stability. A high CV_{row} indicates that a few rows have much larger variance than others (row-wise outliers), while a high CV_{col} indicates the same for columns.

To ensure size-independent comparison, we normalize each CV by its expected value under an IID assumption. For IID data, sample variance follows a scaled chi-squared distribution with $\text{CV}(\text{Var}) \approx \sqrt{2/(n_s - 1)}$, where n_s is the number of samples per variance estimate. Since row variances are computed over n columns and column variances over m rows, we normalize:

$$\hat{\text{CV}}_{\text{row}} = \text{CV}_{\text{row}} \,/\, \sqrt{2/(n-1)}, \tag{11}$$

$$\hat{\text{CV}}_{\text{col}} = \text{CV}_{\text{col}} \,/\, \sqrt{2/(m-1)}. \tag{12}$$

For IID data, $\hat{\text{CV}}_{\text{row}} \approx \hat{\text{CV}}_{\text{col}} \approx 1$, so any deviation from unity signals structured outliers. Without this normalization a non-square matrix would show a spurious pattern, since its row and column variances are estimated from different numbers of samples. The outlier pattern is classified by comparing the *ratio* of the two normalized CVs against a threshold τ:

- **Row-wise**: if $\hat{\text{CV}}_{\text{row}}/\hat{\text{CV}}_{\text{col}} > \tau$ (row variances are disproportionately non-uniform, indicating outlier rows).
- **Column-wise**: if $\hat{\text{CV}}_{\text{col}}/\hat{\text{CV}}_{\text{row}} > \tau$ (column variances are disproportionately non-uniform, indicating outlier columns).
- **None**: if neither ratio exceeds τ.

In our experiments, we use $\tau = 2.0$.

C Why Variance Is Used for Outlier Detection

In AdaHOP, the calibration and Fast Outlier Index Detection stages identify outlier rows or columns by ranking candidates according to their within-row or within-column *variance* rather than by the more intuitive mean magnitude. The choice follows from what an outlier row or column of an LLM

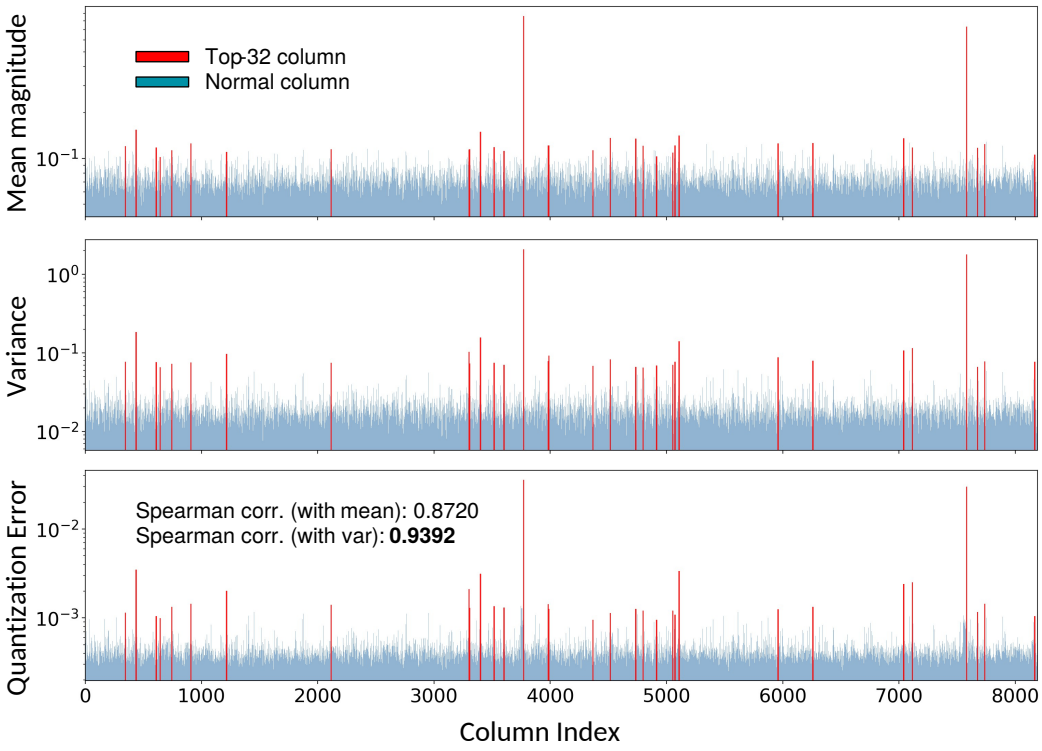


Figure 5: Per-column mean magnitude, variance, and quantization error of the activation tensor from Llama3.2-3B's `block.11.feed_forward.w2`. Red bars indicate the top-32 columns for each metric; light blue bars indicate the remaining columns.

training tensor is. It is large as a whole rather than at a few isolated positions, so its relative dispersion, measured by γ_b, stays at the level of an ordinary row or column, as the separation assumption of the theoretical analysis states, while its overall scale is far larger than that of its neighbors. Variance measures this scale, since it grows with the squared magnitude of the feature, so ranking rows or columns by variance ranks them by the share of $\|A\|_F$ they carry. That share is the quantity OE removes and therefore the one that decides difficulty under the blind spot of the scaling axis.

Mean magnitude misses the same structure, because it is diluted by sign cancellation and by the many ordinary values that surround the large ones. The quantization error of a feature grows with its energy, since the block scale and hence the grid spacing follow the squared magnitude of its values, so a column whose values alternate in sign around a large scale ranks high by variance and high by quantization error but lower by mean magnitude.

We measure the difference on the activation tensor of Llama3.2-3B's `block.11.feed_forward.w2`, computing the per-column mean magnitude, the per-column variance, and the per-column quantization error under MXFP4 (Figure 5). The top-32 columns by variance overlap those by quantization error more closely than the top-32 columns by mean magnitude do.

The Spearman rank correlation with the per-column quantization error puts a number on that overlap: $\rho = 0.9392$ for variance against $\rho = 0.8720$ for mean magnitude, so variance ranks the columns that will incur the largest error more faithfully and selects a better set to extract into the BF16 outlier path.

Variance is therefore the criterion FOID uses, since it tracks the quantity the extraction is meant to reduce, the per-column quantization error, and it stays cheap enough to recompute at every step of training.

Table 4: No fixed rule is optimal across the nine pattern pairs. Relative MSE improvement (%) of each fixed rule applied to every pair, measured over five seeds on tensors from Llama3.2-3B's `layers.15.attention.wv`; negative values mean increased error. Row optimum is the best rule for that pair, and the last two columns give the strategy AdaHOP assigns and its improvement. OE rules pay the BF16 outlier path on every pair, whereas AdaHOP pays it on five.

Pair (A, B)	all-IHT	all-OHT	all-OE-L	all-OE-R	**Row optimum**	**AdaHOP**	**Gain**
RR	8.79	0.12	**23.72**	10.23	OE-L (23.72)	IHT + OE-L	23.72
RC	0.20	3.07	16.58	**87.80**	OE-R (87.80)	IHT + OE-R	87.80
RN	−0.01	0.13	**16.33**	1.57	OE-L (16.33)	IHT + OE-L	16.33
NR	8.73	−0.04	10.17	**10.18**	OE-R (10.18)	IHT	8.73
NC	−0.05	0.14	1.53	**16.30**	OE-R (16.30)	IHT + OE-R	16.30
NN	−0.05	−0.08	**1.53**	**1.53**	OE-L (1.53)	IHT	−0.05
CR	87.65	−0.05	**87.86**	87.85	OE-L (87.86)	IHT	87.65
CC	9.39	2.68	10.84	**88.89**	OE-R (88.89)	IHT + OE-R	88.89
CN	39.24	−0.19	**40.24**	40.21	OE-L (40.24)	IHT	39.24
Average	17.10	0.64	23.20	38.28	41.43		40.96

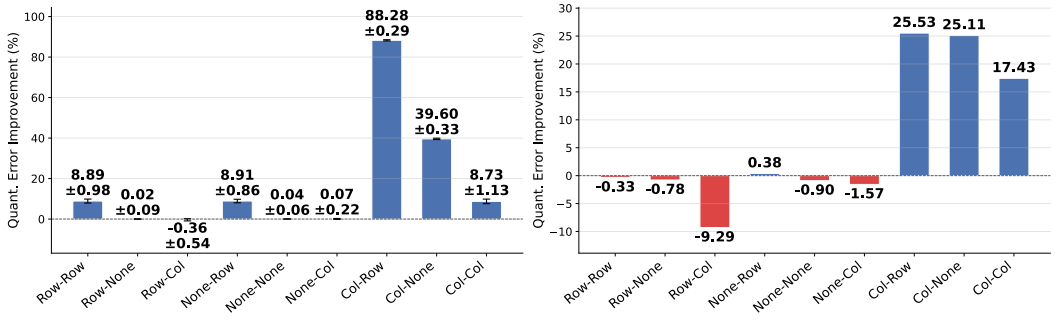


Figure 6: The fixed-rule ordering holds on tensors it was not measured on, so it is a property of the pattern pair rather than of one tensor. We report the relative MSE reduction from IHT over all nine outlier-pattern pairs for synthetic tensors with five random seeds (Left) and for real tensors from Llama3.2-3B's `block.11.feed_forward.w2` (Right), a different layer from the `layers.15.attention.wv` of Figure 3. Improvement is defined as $(E_{\text{base}} - E_{\text{IHT}})/E_{\text{base}} \times 100\%$, where $E_{\text{base}} = \text{MSE}(Q(A)Q(B), AB)$ and $E_{\text{IHT}} = \text{MSE}(Q(AH_k)Q(H_k^\top B), AB)$.

D Fixed-Rule Comparison Across Outlier Pattern Pairs

Figure 3 compares IHT alone with the best fixed rule on each pattern pair of real training tensors, and the gap between the two bars is what the BF16 outlier path buys. On the five pairs with an operand in the blind spot, **RR**, **RC**, **RN**, **NC**, and **CC**, the best rule leads IHT by more than 14 points, whereas on the remaining four the lead is below 1.6 points, so AdaHOP invokes OE exactly where the blind spot predicts. Table 4 applies each of the four fixed rules to every pattern pair, so the row optimum of the figure can be traced back to the rule that attains it.

Two properties of the table are not visible in the figure. The two OE rules are far from interchangeable, since OE-Left and OE-Right differ by more than 70 points on **RC** and **CC** while agreeing to within 0.04 points on **CR** and **NN**, which is why the decision rule of AdaHOP selects a side rather than simply enabling extraction. The OE rules also pay the BF16 outlier path on every pair, whereas AdaHOP pays it on five, so the averages of the last two columns are not comparable at equal cost.

The alignment effect is not specific to these tensors. Figure 6 repeats the IHT measurement on synthetic tensors with controlled outlier patterns and on the activation and gradient tensors of Llama3.2-3B's `block.11.feed_forward.w2`, and both reproduce the ordering of Figure 3: IHT helps on **CR**, **CN**, and **CC**, and is ineffective or harmful elsewhere. The magnitudes differ across tensors because the improvement is measured relative to each tensor's own untransformed error, so a tensor whose baseline error is already moderate leaves less for the transform to remove, but the set of pairs that benefit is the same.

E Outlier Patterns of Llama3.2-3B

This section shows the three patterns on the tensors of one block of Llama3.2-1B (Figure 7), then tracks them over 300 training steps in every representative block of Llama3.2-3B and discusses where the patterns may come from.

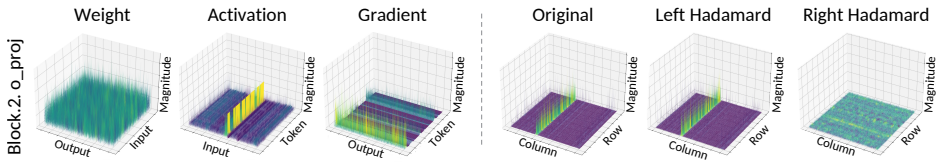


Figure 7: (Left) 3D visualization of weight, activation, and gradient tensors from Llama3.2-1B's `block.2.self_attn_o` after 100 training steps. (Right) A column-wise outlier tensor after different Hadamard directions. Right Hadamard suppresses column-wise outliers; Left Hadamard leaves the structure intact.

Figure 8 visualizes the outlier patterns of all 12 representative blocks of Llama3.2-3B across 300 training steps, and the gradient patterns move through three phases along the depth of the network while the weight and activation patterns stay fixed.

In the early blocks (`block.0`, `block.1`), the gradients of the K, V, and O projections are Row-wise. We attribute this to the proximity of the embedding layer: each weight row there maps directly to a token-level representation, so the loss signal concentrates on a few high-impact tokens and produces a token-wise, and therefore row-wise, structure in the gradients.

In the middle and later blocks, which make up most of the network, the gradients are predominantly Column-wise or None. The repeated attention, feed-forward, and layer normalization operations of `block.5` to `block.21` likely diffuse token-specific gradient energy across the feature dimensions, the same effect that would give the ∇X path its dominance of CN pairs.

In the late blocks (`block.22` to `block.27`), Row-wise outliers re-emerge in the K-projection gradient alone, and we attribute this to the asymmetry of the attention gradient. Near the output, attention concentrates on a small set of anchor tokens (Xiao et al., 2023b), and the key gradient $\nabla_K S = \nabla_S^\top Q/\sqrt{d}$ aggregates contributions from all query positions onto each key token. When many queries attend to the same anchor the gradient signal accumulates on that token's row and forms a row-wise outlier. The query gradient $\nabla_Q S = \nabla_S K/\sqrt{d}$ instead distributes gradients independently across query positions, so no single row dominates and the pattern recurs in K but not in Q. Why the transition happens around block 22 rather than elsewhere is left for future investigation.

Across all three phases the weight and activation patterns remain stable throughout training, so the gradient patterns vary with depth while staying fixed in time. *This is the combination a short calibration phase needs, since it detects each pattern once at the position where it occurs and reuses it for the rest of the run.*

F Calibration Robustness and Sensitivity

Persistence of the calibrated patterns. The 300-step pattern tracking is extended by re-detecting the patterns of Llama3.2-3B at later checkpoints of the same run and comparing them with the 30-step calibration. Table 5 reports the agreement together with the cost of running the calibration up to that point, which is the fraction of training executed in BF16 if the calibration were extended that far. Total training is about 1.41×10^6 PFLOP over 114,441 steps and calibration FLOPs are linear in the step count (12.32 PFLOP per step).

The patterns persist 100× beyond the tracked window: at 30,000 steps, 26% of the way into the run, agreement is still 95.8%. Extending the calibration does not help, since 1,000 steps returns the identical patterns at 33× the cost and 30,000 steps runs 26% of training in BF16 while agreement drops slightly, so 30 steps is kept as the default. If the assumption ever breaks in a longer run, the remedy is periodic re-calibration at the same 0.026% cost per invocation.

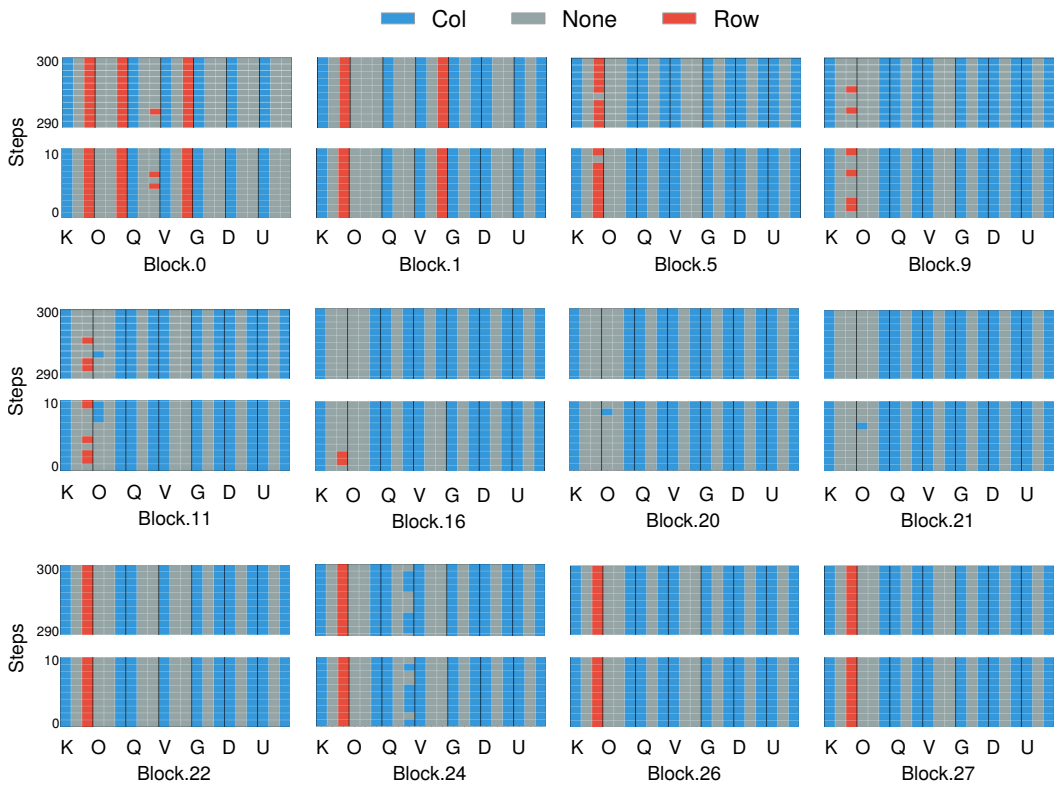


Figure 8: Outlier patterns of weight (W), activation (X), and gradient (G_Y) tensors across 300 training steps for all 12 representative blocks of Llama3.2-3B. Each row represents a tensor from a specific layer, and the color indicates the detected outlier pattern at each step. This extended view reveals depth-dependent transitions in gradient outlier patterns: Row-wise patterns appear in early blocks, fade to None in the middle layers, and re-emerge selectively in the K-projection gradient toward the final blocks.

Table 5: Agreement of the patterns re-detected at later steps of Llama3.2-3B with the 30-step calibration, and the cost of a calibration of that length.

Steps	Agreement (%)	FLOPs (PFLOP)	Share of training (%)
30 (default)	reference	369.7	0.0262
100	99.80	1,232	0.0874
1,000	100.00	12,323	0.8738
30,000	95.80	369,692	26.2144

Threshold sensitivity. Table 6 varies the classification threshold τ over a 3.3× range on Qwen3-0.6B, reporting the share of tensors classified as Row-wise or Column-wise, the share of GEMM pairs routed entirely to BF16, and the C4 training loss at 0.9B tokens.

Table 6: Sensitivity to the threshold τ on Qwen3-0.6B. The MXFP4 and Full-BF16 shares count GEMM pairs, not FLOPs; the OE pairs, which keep only their outlier path in BF16, are counted as MXFP4 here.

τ	R/C rate (%)	MXFP4 (%)	Full-BF16 (%)	C4 loss @ 0.9B
1.5	100.00	90.48	9.52	3.5092
2.0 (default)	89.29	94.90	5.10	3.5131
3.0	85.71	96.26	3.74	3.5134
5.0	83.84	98.64	1.36	3.5108

The four settings are indistinguishable within seed noise: the loss moves by 0.0042 across the range, below the across-seed standard deviation of 0.0088 measured on the same model, so the 3.5108 at $\tau = 5.0$ is not read as better than the default. Since quality does not separate the settings, τ is set by balancing detection coverage against cost. $\tau = 1.5$ classifies every tensor as Row-wise or Column-wise but nearly doubles the BF16 path, while $\tau = 5.0$ minimizes the BF16 path but leaves 16% of tensors as None. On a model whose pattern distribution differs from that of the three measured families, a missed pattern costs more, so $\tau = 2.0$ is kept as a default with margin. The value also has a reference point: the normalized CV is close to 1 for IID data, so $\tau = 2.0$ sits at twice that expectation, and borderline tensors fall into None and receive the default IHT, which makes the effect of a misclassification gradual.

Representativeness of FOID's partial sampling. FOID ranks rows or columns by the variance of their first 64 elements. This is valid because the targets are structural outliers that span an entire row or column, so any contiguous segment reflects that row's statistics, and Table 7 verifies the premise on the output gradient of Llama3.2-3B's `layers.15.attention.wv` (8,192 rows), the operand that OE-Left extracts on the ∇X path.

Table 7: Partial sampling in FOID against the full variance statistic, on the output gradient of Llama3.2-3B's `layers.15.attention.wv`. Every entry uses the leading 64 elements, which is the statistic the kernel computes.

Metric	**Value**
Top-64 index agreement, first 64 elements vs. full variance (%)	81.25
Spearman ρ, first 64 elements vs. full variance	0.9902
Outlier rows above five standard deviations of the row variance	40 / 8,192 (0.49%)
Energy share of the selected top-64 rows (%)	29.67

A rank correlation of 0.9902 means that partial sampling preserves the ranking of the full statistic almost exactly, so the selected set is largely unchanged, and with 40 outlier rows the budget $r = 64$ covers the actual outlier count, consistent with the MSE plateau beyond $r = 64$ in the rank ablation. One failure condition remains: outliers confined to a segment within a row rather than spanning it would be missed by the partial sample, and this is listed among the limitations. Dependence on sequence packing, token order, and batch composition is limited by construction, since FOID selects along whichever axis the calibrated pattern names and re-detects the indices at every step. No single batch fixes the indices, and the calibration itself averages over 30 steps so that no single batch determines the pattern either.

G Pattern Transfer to a Public Checkpoint

Per-run calibration assumes nothing about whether patterns transfer between training runs, but whether they do is informative about what determines them. We ran the 30-step calibration on the public Llama3.2-1B checkpoint, which differs from our from-scratch run in training data, optimizer, learning-rate schedule, and total token count all at once. We then compared the detected patterns across the 112 linear layers (16 blocks $\times$ 7 modules) and the 336 GEMM configurations they generate (Tables 8 and 9).

Table 8: Agreement between the patterns of the from-scratch Llama3.2-1B and the public checkpoint, per tensor (left) and per GEMM (right).

Tensor	**Agreement (%)**	**GEMM (operands)**	**Agreement (%)**
Activation	78.6	Forward (X, W)	77.7
Weight	99.1	∇W (G_Y, X)	78.6
Gradient	100.0	∇X (G_Y, W)	99.1
All three tensors	77.7	Overall (336 configurations)	85.1

The result splits by tensor type. Weight and gradient patterns agree across the two checkpoints while activation patterns do not: a Shapley attribution over the 50 mismatched GEMM configurations

Table 9: Per-block agreement of all three tensors between the two checkpoints (%).

Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Agreement	57	71	57	100	86	86	71	86	100	100	100	100	57	43	71	57

assigns 96% of the disagreement to the activation, and the ∇X GEMM, which uses no activation, stays at 99.1%. The disagreements are systematic rather than noisy, since in 23 of the 25 mismatched tensors both runs classified the pattern with at least 90% majority over the 30 steps. They also concentrate in depth: the middle blocks 8 to 11 match completely, while most of the disagreement sits in the first three and the last four blocks. The modal pattern triple per module type is identical in 6 of the 7 types, the exception being the attention output projection whose activation shifts from None to Column-wise in 9 of 16 blocks, so the pattern-pair distribution carries over to the public checkpoint.

This supports AdaHOP's design in two ways. Weight and gradient patterns appear to be set largely by the architecture and are insensitive to how the run is configured, consistent with the pattern structure reproducing across the three model families. The activation pattern depends on the training state, so the exposure of a setup change is confined to the two GEMMs that carry the activation, and it is exactly this component that the 30-step calibration re-measures in every run at 0.03% of its cost.

H Detailed Experimental Settings

Hardware Environment. All experiments are conducted on AMD Instinct™ MI350X GPUs with the ROCm 7.0.2 software stack. Custom fused kernels are implemented using Triton 3.6, targeting the AMD CDNA4 architecture.

Pattern-Pair Analysis. For the real tensor analysis, we extract the gradient, activation, and weight tensors from Llama3.2-3B's `block.11.feed_forward.w2` and trim them to the first 1024×1024 tile. The resulting kurtosis values are 16.00 for the gradient (originally Column-wise tensor, transposed to simulate Row-wise pattern), 222.00 for the activation (Column-wise), and 2.88 for the weight (None).

In the synthetic tensor analysis, we use 4096×4096 synthetic tensors A and B with 5 random seeds (42, 256, 1925, 2048, 7777) and amplify random columns' magnitudes by 25.00 for Column-wise tensors and 5.00 for Row-wise tensors. This is because Column-wise tensors tend to correspond to activations and exhibit higher kurtosis across LLMs. The resulting mean kurtosis values are Row: 9.08, Column: 208.30 when outliers are present, and None: 3.0.

For MSE improvement calculation, the error definition is $E_{\text{base}} = \text{MSE}(Q(A)Q(B), AB)$ and $E_{\text{IHT}} = \text{MSE}(Q(AH_k)Q(H_k^\top B), AB)$.

Training Experiments. We evaluate AdaHOP on six LLM architectures from 0.6B to 8B parameters: Qwen3-0.6B, Llama3.2-1B, Instella-3B, Llama3.2-3B, Qwen3-4B, and Llama3.1-8B. All models are trained on the C4 (Colossal Clean Crawled Corpus) dataset (Raffel et al., 2020) with a sequence length of 4,096 tokens, local batch size of 8 and global batch size of 128. We compare AdaHOP against seven baselines: BF16 full-precision training, which is the upper bound on quality; naive MXFP4 quantization without any outlier suppression; MXFP4+Hadamard, which applies uniform IHT to all layers; Tseng et al. (2025), which combines random Hadamard transforms with stochastic rounding; HALO (Ashkboos et al., 2025), which applies OHT to the gradient computation paths; and, on Qwen3-0.6B, HOT (Kim et al., 2025) and Quartet (Castro et al., 2025). We report training loss, zero-shot accuracy on downstream benchmarks (PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), ARC-Easy (Clark et al., 2018), LAMBADA (Paperno et al., 2016)), memory consumption, and training throughput.

All models are trained using the AdamW optimizer (Loshchilov & Hutter, 2017) with a learning rate of 4×10^{-4} and an epsilon of 10^{-8}, following a linear warmup schedule over 200 steps. Following the Chinchilla optimal scaling law (Hoffmann et al., 2022), the number of training tokens is scaled with model size: Llama3.2-1B is trained on 40B tokens, Llama3.2-3B on 60B tokens, Qwen3-4B on 80B tokens, and Llama3.1-8B on 160B tokens. For Instella-3B, training tokens are 40B due to a time limitation. Gradient norms are clipped to a maximum of 1.0 throughout training.

I Baseline Implementation and Reproducibility

Every baseline was reimplemented in the single Torchtitan-based framework of AdaHOP, starting from the code released with the original papers, so that the comparison is controlled at every level. Method-specific components follow the original specifications exactly, including the stochastic rounding and random Hadamard transform of Tseng et al. (2025) and the outer-dimension rotations of HALO on the gradient paths. All methods share the same MXFP4 format (block size 32 with an E8M0 scale), the same scaling policy, and the same fused CDNA4 kernel infrastructure, with method-specific operations implemented at a comparable level of optimization. They share the optimizer, learning-rate schedule, batch size, and token count as AdaHOP, and no baseline hyperparameter was retuned for MXFP4.

Two baselines required adaptation to be compared under the same precision budget. HOT (Kim et al., 2025) targets the backward pass only in its original form: it quantizes the input gradient to INT4 and the weight gradient to INT8 with a Hadamard low-rank approximation, keeps the forward pass in full precision, and was evaluated on vision and BERT models. We extended it to MXFP4 on both the forward and the backward path so that it meets the same conditions as every other method. Quartet (Castro et al., 2025) was implemented with its original algorithm, but its kernel code was not public during our development period and the version released since targets NVIDIA Blackwell. The comparison is therefore run under precision simulation and limited to quality, since reimplementing the kernel ourselves would confound design with implementation. FP4 All the Way (Chmiel et al., 2025) splits precision across training stages and reverts the backward pass to BF16 during fine-tuning, which is a different precision budget from the fully MXFP4 setting here, and is therefore not a direct comparison point.

The algorithmic and the kernel contributions are separated by construction. The quality results depend only on the numeric format and the algorithm, and every method was trained under identical precision conditions, so they do not depend on the kernel and should carry over to other hardware. The speed results depend on the kernel, and the kernel latency breakdown reports the IHT and quantization, the fused GEMM, and the outlier path separately; since all baselines share the same kernel infrastructure, the speed comparison is controlled as well. The assignment rule of AdaHOP should transfer unchanged to any architecture that supports concurrent mixed-precision execution, since the CDNA4-specific part is confined to the kernel.

J Loss Comparison and Component Ablation

Section 6.3 rests on the measurements of Table 10. The left table compares every method on Qwen3-0.6B under one MXFP4 format, scaling policy, and schedule, listing the assignment unit of each method next to its loss and gap so that the random-matched control can be read against AdaHOP-Lv1 at equal BF16 budget. The right table isolates the two components of AdaHOP on Llama3.2-1B. Loss scaling and stochastic rounding are orthogonal to AdaHOP because a global scalar preserves both γ_b and η_r and stochastic rounding satisfies the same per-element error model, so every bound of the theoretical analysis holds under them.

K Theoretical Analysis

Section 5.1 selects AdaHOP's strategy from error bounds, and this section states the error model behind them and proves the bounds. Throughout, $b = 32$ is the MXFP4 scaling block width, H_b is the normalized Walsh–Hadamard matrix of size b with $H_b^\top H_b = I$, and IHT denotes the block-diagonal transform $A \mapsto AH_b$, $B \mapsto H_b^\top B$ applied to every block of the reduction axis, which is the transform the kernel executes. The bounds are stated for a single product $C = AB$, following the convention of Tseng et al. (2025) and QuIP (Chee et al., 2023); the selection rule they justify is measured independently in the fixed-rule comparison. A pattern pair is written as (pattern of A, pattern of B).

Table 10: Where the gain comes from. (Left) C4 training loss of Qwen3-0.6B after 1B tokens (seed 42; mean ± std over three seeds where available) under a shared MXFP4 format and schedule. Random OE applies OE to 250 GEMMs at random positions, the same number of the 588 that AdaHOP-Lv1 selects, and therefore carries the same BF16 budget; SR, LS, and PA denote stochastic rounding, loss scaling, and pattern-aware assignment. (Right) Component ablation on Llama3.2-1B after 20B tokens.

Method	Assign.	Loss	Gap	3 seeds
BF16	–	3.4966	0	3.4868 ± 0.0088
MXFP4	–	3.9190	0.4224	–
MXFP4 + SR	–	3.8981	0.4015	–
MXFP4 + LS	–	3.8038	0.3072	–
Uniform IHT	Uniform	3.5584	0.0618	–
Tseng et al.	Uniform	3.5401	0.0435	–
HALO	Path-level	3.5411	0.0445	3.5410 ± 0.0021
Quartet	Path-level	3.5382	0.0416	–
Random OE	Random	3.5413	0.0447	–
Tseng et al. + PA	Pattern	3.5048	0.0082	–
AdaHOP-Lv1	Pattern	**3.5007**	**0.0041**	3.5043 ± 0.0077

Setting	Loss	Gap
BF16	2.8482	0
MXFP4	3.5328	0.6846
Uniform IHT	2.8911	0.0429
Pattern-aware IHT	2.8715	0.0233
OE only	2.9031	0.0549
AdaHOP-Lv1	2.8649	0.0167
AdaHOP-Lv2	**2.8598**	**0.0116**

K.1 Error Model

Assumption 1 (Block-scaled quantization error)**.** *For $\Delta_A := Q(A) - A$, at any scaling block β and any element $l \in \beta$,*

$$|\Delta_{A,l}| \le \epsilon_q \max_{l' \in \beta} |A_{l'}|.$$

This holds for block-scaled formats such as MXFP4, where one shared scale per block sets the grid spacing from the block maximum, with $\epsilon_q = 2^{-2}$ for the E2M1 element type. Stochastic rounding satisfies the same bound, so every statement below holds under it unchanged.

Definition 1 (Block outlier coefficient)**.** *For a scaling block β of A,*

$$\gamma_\beta(A) := \frac{b \max_{l \in \beta} A_l^2}{\|A_\beta\|_F^2}, \qquad \gamma_b(A) := \max_\beta \gamma_\beta(A), \qquad 1 \le \gamma_b(A) \le b.$$

γ_b is the square of the incoherence parameter of QuIP restricted to a scaling block. It equals 1 when every element of the block has the same magnitude and b when a single element carries all of the block's energy. Blocks are taken along the reduction axis, so each block of A lies within one row of A and each block of B within one column of B.

Lemma 1. $\|\Delta_A\|_F \le \epsilon_q \sqrt{\gamma_b(A)}\, \|A\|_F$.

Proof. $\|\Delta_A\|_F^2 \le \sum_\beta b\, \epsilon_q^2 \max_{l \in \beta} A_l^2 = \epsilon_q^2 \sum_\beta \gamma_\beta(A)\, \|A_\beta\|_F^2 \le \epsilon_q^2\, \gamma_b(A)\, \|A\|_F^2$. □

K.2 Product Error

Theorem 1 (Sum-form bound)**.**

$$\big\|Q(A)Q(B) - AB\big\|_F \le \epsilon_q \|A\|_F \|B\|_F \Big(\underbrace{\sqrt{\gamma_b(A)} + \sqrt{\gamma_b(B)}}_{\textit{first order}} + \underbrace{\epsilon_q \sqrt{\gamma_b(A)\gamma_b(B)}}_{\textit{second order}} \Big).$$

Proof. $Q(A)Q(B) - AB = \Delta_A B + A\Delta_B + \Delta_A \Delta_B$. Apply the triangle inequality and $\|XY\|_F \le \|X\|_F \|Y\|_F$ to each term, then substitute Lemma 1. □

Corollary 1 (Dominant operand)**.** $\max\big(\sqrt{\gamma_b(A)}, \sqrt{\gamma_b(B)}\big) \le \sqrt{\gamma_b(A)} + \sqrt{\gamma_b(B)} \le 2 \max\big(\sqrt{\gamma_b(A)}, \sqrt{\gamma_b(B)}\big)$.

The first-order error is therefore set by the operand with the larger γ_b, and transforming the other operand alone cannot reduce it by more than a factor of two. This is what makes the assignment asymmetric: a pair is only as good as its worse operand, so AdaHOP decides per pair which side to treat.

K.3 Effect of the Transforms on γ_b

Proposition 1 (Block Hadamard). *$\gamma_b(AH_b) = O(1)$ under two conditions:*

- *C1 (energy concentration) within each block, the outlier energy is concentrated on at most ρb elements with $\rho \ll 1$;*
- *C2 (no sign alignment) there is no worst-case sign alignment between the outlier support and the rows of H_b.*

A random sign diagonal H_bD satisfies both with probability $1 - \delta$.

Proof. Write $y_i = b^{-1/2} \sum_{l \in \beta} h_{il} a_l$ for the i-th mixed element of block β, with $h_{il} \in \{\pm 1\}$. With a random sign diagonal the terms $h_{il} a_l$ are independent and symmetric with $\sum_l a_l^2 = \|A_\beta\|_F^2$, so Hoeffding's inequality gives $|y_i| \le \sqrt{2\log(2b/\delta)}\; b^{-1/2} \|A_\beta\|_F$ for all i with probability at least $1 - \delta$, and orthogonality preserves $\|A_\beta\|_F$, hence $\gamma_\beta(AH_bD) \le 2\log(2b/\delta) = O(1)$ for the fixed $b = 32$. For the fixed H_b, the same concentration holds whenever the deterministic signs h_{il} do not align with the signs of the outlier entries: under C1 the sum is dominated by its ρb outlier terms, and C2 states that no row i aligns with all of them, which is the typical-case condition a fixed Walsh–Hadamard transform requires and the same condition QuIP# (Tseng et al., 2024) removes with a random sign diagonal. □

Proposition 2 (Scaling invariance). *$\gamma_b(A)$ is invariant to row scaling of A, and $\gamma_b(B)$ is invariant to column scaling of B.*

Proof. Each block of A lies within one row, so multiplying row i by c scales both $\max_{l \in \beta} A_l^2$ and $\|A_\beta\|_F^2$ by c^2 and preserves γ_β. The same holds for B along columns. □

Assumption 2 (Approximate separation). *A Row-wise A can be written as $A = D_r \tilde{A}$ and a Column-wise B as $B = \tilde{B} D_c$ with diagonal D_r, D_c and residuals satisfying $\gamma_b(\tilde{A}) = O(1)$ and $\gamma_b(\tilde{B}) = O(1)$.*

Together with Proposition 2, this means that a Row-wise A or a Column-wise B already has $\gamma_b = O(1)$: the outlier occupies a whole row of A (a whole column of B), so every block inside it is uniformly scaled and no element dominates its block. Conversely, a Column-wise A or a Row-wise B places a few large elements inside each block, so γ_b is large, and it is exactly this case that IHT lowers by Proposition 1. The outlier that a whole row of A carries therefore does not appear in $\gamma_b(A)$; it appears in $\|A\|_F$, which is the quantity OE addresses below.

Proposition 3 (Ineffectiveness of outer-axis mixing). *Let $e_j(A) := \sum_i A_{ij}^2$ be the energy of column j and, for a group g of b consecutive columns, $\Gamma_g(A) := b \max_{j \in g} e_j / \sum_{j \in g} e_j$. Then*

$$\textit{(a)} \quad \gamma_b(A) \ge \max_g \Gamma_g(A), \qquad \textit{(b)} \quad \Gamma_g(H_m A) = \Gamma_g(A) \ \textit{for any orthogonal } H_m.$$

Proof. (b) holds because an orthogonal transform along the rows preserves the energy of every column. For (a), fix g, let $j^\star := \arg\max_{j \in g} e_j$ and $f(i) := \sum_{j \in g} A_{ij}^2$ be the energy of the block of row i over g. Then $\sum_i A_{ij^\star}^2 / \sum_i f(i) = e_{j^\star} / \sum_{j \in g} e_j = \Gamma_g / b$, and since a ratio of sums cannot exceed the largest term-wise ratio, some row i satisfies $A_{ij^\star}^2 / f(i) \ge \Gamma_g / b$. The block of that row over g gives $\gamma_{i,g}(A) \ge b A_{ij^\star}^2 / f(i) \ge \Gamma_g$. The argument for BH_n is identical with row energies. □

OHT mixes A along its rows and B along its columns. By (b) it leaves every Γ_g unchanged, and by (a) γ_b can never fall below $\max_g \Gamma_g$, so OHT cannot lower the first-order term of Theorem 1 for any operand whose outliers concentrate in a few columns of A or rows of B. This is the formal counterpart of the measured all-OHT rule, which never exceeds 3.1% on any pair.

Proposition 4 (Blind spot of the scaling axis). *Let $A = D_r\tilde{A}$ with diagonal D_r, and let H_m and H_k be any orthogonal matrices applied along the outer and the reduction axis. Then*

(a) $\gamma_b(A) = \gamma_b(\tilde{A})$, *(b)* $\|H_m A H_k\|_F = \|A\|_F$, *(c)* $\epsilon_q\|A\|_F \le \epsilon_q\sqrt{\gamma_b(H_m A H_k)}\,\|A\|_F$,

so when $\gamma_b(\tilde{A}) = O(1)$ the bound of Lemma 1 for A lies within the factor $\sqrt{\gamma_b(\tilde{A})} = O(1)$ of the floor $\epsilon_q\|A\|_F$ that (c) assigns to every transform of A, whereas OE-Left with $r \ge |S|$ lowers the same bound to $\epsilon_q\eta_r(A)\sqrt{\gamma_b(A_{\mathrm{res}}H_k)}\,\|A\|_F$ with $\eta_r(A) < 1$ (Definition 2, Theorem 2).

Proof. (a) is Proposition 2, since D_r scales rows. (b) holds because orthogonal transforms on either side preserve the Frobenius norm. (c) follows from $\gamma_b \ge 1$ in Definition 1. The comparison with OE-Left substitutes $\|A_{\mathrm{res}}\|_F = \eta_r(A)\|A\|_F$ into Lemma 1 for the MXFP4 term of Theorem 2. □

A Row-wise A satisfies the premise by Assumption 2, so its outlier rows are parallel to the scaling axis of A and enter the bound only through $\|A\|_F$, which (b) protects from every orthogonal transform; a Column-wise B is the same statement transposed. Rotating the scaling block to the outer axis would move these rows out of the blind spot, but the tensor core factors the block scales out of the inner sum only when each scale is constant along k, so no such product is executable. The measurements are consistent with both halves: IHT and OHT barely change the error of an operand with outlier rows at $b = 32$, and quantizing with blocks along the outer axis swaps which of the row-wise and the column-wise operand is affected.

Definition 2 (Residual norm ratio). *With extraction budget r and outlier support S ($|S| \le r$), decompose $A = A_{\mathrm{out}} + A_{\mathrm{res}}$, where A_{out} holds the rows in S, and define*

$$\eta_r(A) := \frac{\|A_{\mathrm{res}}\|_F}{\|A\|_F} \in (0, 1].$$

$\eta_r(B)$ is defined in the same way over the columns of B.

Corollary 2 (Coverage of the nine pairs). *IHT maps $A \mapsto AH_b$ and $B \mapsto H_b^\top B$, so under C2 it achieves $\gamma_b(AH_b) = \gamma_b(H_b^\top B) = O(1)$ for all nine pattern pairs. Two groups of pairs remain distinguished:*

- *(i) pairs whose γ_b is already $O(1)$ and for which IHT has nothing to lower: RC, RN, NC, NN;*
- *(ii) the blind-spot pairs with $\eta_r < 1$, for which OE has something to lower: RR, RC, RN, NC, CC.*

Group (i) follows from Assumption 2 and Proposition 2, and group (ii) collects the pairs in which A is Row-wise (OE-Left extracts its rows) or B is Column-wise (OE-Right extracts its columns), that is, the pairs with an operand in the blind spot of Proposition 4. An operand whose γ_b is already $O(1)$ is covered by setting $H_b = I$.

K.4 Outlier Extraction

Theorem 2 (OE-Left bound). *For*

$$\hat{C} = \underbrace{Q(A_{\mathrm{res}}H_b)\,Q(H_b^\top B)}_{\mathrm{MXFP4}} + \underbrace{\mathrm{fl}_{\mathrm{bf16}}\big(A_{\mathrm{out}}B\big)}_{\mathrm{BF16}}$$

and $r \ge |S|$,

$$\big\|\hat{C} - AB\big\|_F \le \epsilon_q\|A_{\mathrm{res}}\|_F\|B\|_F\Big(\sqrt{\gamma_b(A_{\mathrm{res}}H_b)} + \sqrt{\gamma_b(H_b^\top B)} + \epsilon_q\sqrt{\gamma_b(A_{\mathrm{res}}H_b)\,\gamma_b(H_b^\top B)}\Big) + \epsilon_{\mathrm{bf16}}\|A_{\mathrm{out}}\|_F\|B\|_F.$$

Proof. Orthogonality of H_b gives $A_{\mathrm{res}}H_bH_b^\top B = A_{\mathrm{res}}B$, so Theorem 1 applies to the MXFP4 term with $\|A_{\mathrm{res}}H_b\|_F = \|A_{\mathrm{res}}\|_F$ and $\|H_b^\top B\|_F = \|B\|_F$. The BF16 term is not exact; its rounding error is bounded by $\epsilon_{\mathrm{bf16}}\|A_{\mathrm{out}}\|_F\|B\|_F$ and is added through the triangle inequality. □

Table 11: Normalized error $\Phi := \|\hat{C} - AB\|_F / (\epsilon_q \|A\|_F \|B\|_F)$ bounded by Theorem 1 and corollary 3 for the nine pattern pairs, substituting $\gamma_b = b = 32$ for an operand whose outliers sit inside its blocks, $\gamma_b = 1$ otherwise, $\epsilon_q = 2^{-2}$, and the calibrated η_r values (0.95 for row extraction, 0.52 for column extraction); the BF16 term of at most 2^{-6} is omitted. The OHT column equals the no-transform column because OHT cannot lower γ_b.

Pair (A, B)	No transform	IHT	OHT	**AdaHOP strategy**	**AdaHOP** Φ
RR	8.07	2.25	8.07	IHT + OE-Left	2.14
RC	2.25	2.25	2.25	IHT + OE-Right	1.17
RN	2.25	2.25	2.25	IHT + OE-Left	2.14
NR	8.07	2.25	8.07	IHT	2.25
NC	2.25	2.25	2.25	IHT + OE-Right	1.17
NN	2.25	2.25	2.25	IHT	2.25
CR	19.31	2.25	19.31	IHT	2.25
CC	8.07	2.25	8.07	IHT + OE-Right	1.17
CN	8.07	2.25	8.07	IHT	2.25

Corollary 3 (Normalized form and OE-Right)**.** *Dividing by* $\epsilon_q \|A\|_F \|B\|_F$ *and using* $\|A_{\text{out}}\|_F / \|A\|_F = \sqrt{1 - \eta_r(A)^2}$*,*

$$\Phi_{\text{OE-L}} \le \eta_r(A)\Big(\sqrt{\gamma_b(A_{\text{res}} H_b)} + \sqrt{\gamma_b(H_b^\top B)} + \epsilon_q \sqrt{\gamma_b(A_{\text{res}} H_b)\,\gamma_b(H_b^\top B)}\Big) + \frac{\epsilon_{\text{bf16}}}{\epsilon_q}\sqrt{1 - \eta_r(A)^2}, \qquad \frac{\epsilon_{\text{bf16}}}{\epsilon_q} \approx 2^{-6}.$$

OE-Right for $C = AB$ is OE-Left for $C^\top = B^\top A^\top$, so the same bound holds with $\eta_r(B)$, $\gamma_b(AH_b)$, and $\gamma_b(H_b^\top B_{\text{res}})$ in place of $\eta_r(A)$, $\gamma_b(A_{\text{res}} H_b)$, and $\gamma_b(H_b^\top B)$.

The condition $r \ge |S|$ is what makes the extraction exact, and calibration confirms it: in the output gradient of Llama3.2-3B's `layers.15.attention.wv`, 40 of 8,192 rows (0.49%) exceed five standard deviations of the row variance, so $r = 64$ covers the support with margin. In the ∇W GEMMs of the same layer, column extraction from the activation leaves $\eta_r \approx 0.52$ while row extraction from the gradient leaves $\eta_r \approx 0.95$.

Corollary 4 (Decision rule)**.** *Under Corollary 2, both residual coefficients are $O(1)$ after IHT, so the bound of Corollary 3 is minimized by the side with the smaller residual norm ratio:*

$$\textit{OE-Left} \iff \eta_r(A) < \eta_r(B), \qquad \textit{OE-Right} \iff \eta_r(B) < \eta_r(A).$$

When both equal 1, IHT alone is applied; a tie below 1 is resolved in favor of OE-Left.

Applied to the pairs observed in training, the rule reproduces the assignment of AdaHOP: RN selects OE-Left because only A has extractable rows, RC, NC, and CC select OE-Right because the activation-side columns leave the smaller η_r, and CN, NN, and CR fall to IHT alone.

K.5 Recomputed Error Ratios

Table 11 recomputes the bound for every pair. The conclusion that no fixed rule is optimal across the nine pairs is preserved. OHT is up to 8.6× behind IHT because it cannot lower γ_b at all, and IHT itself is up to 1.9× behind AdaHOP on the pairs of group (ii), where the remaining error sits in the norm factor that only extraction can reduce. The ordering matches the fixed-rule measurement, where all-OHT never exceeds 3.1% and the OE rules lead IHT by more than 14 points exactly on the pairs of group (ii). The substituted size is the mixing width of the transform actually applied, the MXFP4 scaling block of 32; substituting full tensor dimensions instead would overstate the reduction, since mixing across scaling blocks does not change the dynamic range inside one.

K.6 Connection to the CV-Based Detector

The theory is stated in terms of γ_b, which depends on block maxima, while calibration classifies tensors by the normalized CV of row and column variances. The two criteria are separable in principle,

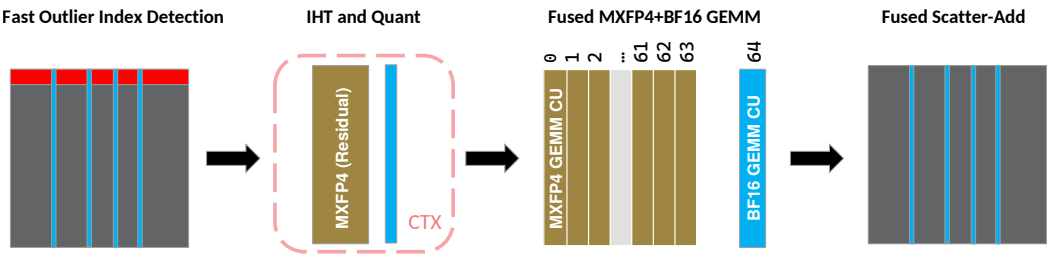


Figure 9: Hardware-aware implementation pipeline of AdaHOP. The pipeline consists of four stages: FOID, IHT and Quantization, Fused MXFP4+BF16 GEMM, and Fused Scatter-Add, all implemented as optimized Triton kernels targeting the AMD CDNA4 architecture. Note that the red block in Fast Outlier Index Detection represents the first 64 elements along each column.

but real tensors concentrate their outliers in a few rows or columns, so the rows with the highest variance and the rows with the highest maximum nearly coincide. On the output gradient of Llama3.2-3B's `layers.15.attention.wv`, the Spearman correlation between the variance-based and the max-based ranking is $\rho = 0.9941$. The γ_b reductions above therefore apply to the CV-based labels as an approximation, and the partial sampling of FOID preserves the same ranking at $\rho = 0.9902$.

L Hardware-Aware Implementation Details

The AdaHOP pipeline is implemented as fused Triton kernels for the AMD CDNA4 architecture (Figure 9). It runs in four stages, and each of them exists to keep the cost of OE off the critical path of the MXFP4 residual computation.

Extraction Rank Selection. We set $r = 64$ for the number of extracted outlier rows or columns. Although the outlier path is computed in BF16 rather than MXFP4, its size is constrained by the fused kernel's latency-hiding design. In stage 3, the MXFP4 residual GEMM and BF16 outlier GEMM execute concurrently on different Compute Units within a single kernel launch: BF16 tiles are evenly interleaved among the more numerous MXFP4 tiles via Bresenham-style scheduling, so that both tile types run in parallel from the first wave. The BF16 branch uses smaller output tiles than the MXFP4 branch so that each BF16 tile completes within the execution time of one MXFP4 tile. Setting $r = 64$ keeps the total number of BF16 tiles small enough to hide behind the dominant MXFP4 computation, so the outlier path adds almost nothing to wall-clock time. The rank ablation measures where this balance sits, since the MSE plateaus at $r = 64$ while the latency rises beyond it.

1. Fast Outlier Index Detection (FOID) exploits the fixed structure of outlier rows or columns by computing the variance of the first 64 elements along each row (or column) to quickly identify outlier indices. This metric is effective because variance grows with the squared magnitude of a row or column, so outlier rows or columns, which are large as a whole, rank far above ordinary ones. The top-r rows (or columns) by variance are selected as outlier features, and the corresponding entries in the residual tensor are zeroed out.
2. IHT and Quantization apply IHT to the residual tensor (with outliers removed) using 1D Hadamard Transform with block size 32, followed by MXFP4 quantization with 1D per-column scaling. For activation tensors in the Forward path, both the quantized residual and the BF16 outlier tensor are saved to the context for backpropagation.
3. Fused MXFP4+BF16 GEMM exploits the Compute Unit (CU) architecture of AMD CDNA4 to execute the MXFP4 residual multiplication and BF16 outlier multiplication concurrently within a single kernel launch. The MXFP4 branch uses large output tiles for maximum throughput, while the BF16 branch uses smaller output tiles so that per-tile latency is balanced; BF16 tiles are evenly interleaved among MXFP4 tiles via Bresenham-style scheduling to ensure concurrent execution from the first wave.
4. Fused Scatter-Add scatters and adds the result of the outlier matmul in-place to the residual matmul result using BF16 accumulation, avoiding the overhead of materializing intermediate results.

The one case the variance criterion does not rank is a feature whose magnitude is large but *constant* across all positions, since its variance is zero. Such a feature needs no extraction, because all of its values share one magnitude and therefore occupy a narrow sub-range of their block, where block-scaled quantization already rounds them accurately. The outlier features of LLM training tensors are not of this kind: their values vary in sign and magnitude around a mean near zero, so their variance tracks their energy, which is the *variation* the criterion ranks.

Table 12: Kernel latency (ms) of two Llama3.1-8B GEMMs, (1) an MLP and (2) an attention projection.

Module	(1)	(2)
BF16 GEMM	4.76	1.05
FOID	0.03	0.04
IHT and Quant	0.45	0.13
Fused GEMM	1.65	0.64
Fused Scatter-Add	0.07	0.02
AdaHOP Total	**2.22**	**0.84**
Speedup	2.14×	1.26×

Kernel latency breakdown. The two GEMMs of Table 12 are (1) the MLP projection with $(m, n, k) = 14336 \times 4096 \times 32768$ and (2) the attention projection with $4096 \times 4096 \times 32768$ of Llama3.1-8B. The four rows correspond to the four stages above, so the cost of the outlier path can be read separately from the cost of the transform and the fused GEMM: FOID costs 0.03 and 0.04 ms and the scatter-add 0.07 and 0.02 ms against totals of 2.22 and 0.84 ms, and the rest of the outlier path is absorbed into the fused GEMM row.

L.1 Numerical Stability and Cost of the Block Hadamard Transform

The transform is numerically benign for three reasons. First, a normalized Walsh–Hadamard matrix satisfies $H_b^\top H_b = I$, so it has condition number 1 and cancels inside the GEMM through $H_b H_b^\top = I$, leaving no structure in which rounding error can accumulate across layers or steps. Second, a 32-point fast Walsh–Hadamard transform consists of five butterfly stages, so its forward error under a unit roundoff u obeys $\|\mathrm{fl}(H_b x) - H_b x\|_2 \le 5u\|x\|_2$. Under the FP32 accumulator this is 3.0×10^{-7}, or 1.2×10^{-6} times the MXFP4 quantization step $\epsilon_q = 2^{-2}$. Third, intermediate values grow by at most a factor of 32, while BF16 carries the same 8-bit exponent as FP32, so neither overflow nor underflow is reachable. Sign alignment between the outlier support and the Hadamard rows, which the block Hadamard bound requires, is a combinatorial matter unrelated to rounding and is handled by a random sign diagonal.

The cost of the transform is independent of model scale. The block width is fixed at $b = 32$ by the MX specification, so the depth stays at $\log_2 32 = 5$ additions per element regardless of the tensor dimension k, and no $\log k$ term arises because mixing across scaling blocks would not change the dynamic range inside one. Since the GEMM performs $O(k)$ work per output element, the relative cost of the transform decreases as models grow. This cost is included in the IHT and Quant row of the kernel latency breakdown and therefore in every reported speedup.

M Rank Ablation Study for Outlier Extraction

We conduct an ablation study on the extraction rank r used in OE, measuring both the quantization error (MSE) and kernel latency as the rank varies over $\{0, 16, 32, 64, 128, 256\}$. Figure 10 summarizes the results for both real and synthetic tensors under the **RC** pattern pair, where IHT alone removes 0.20% of the error, so that the whole reduction has to come from OE.

Real Tensor. We extract the activation and gradient tensors from Llama3.1-8B's `block.25.feed_forward.w2` and crop both to 4096×4096. For forming an **RC** matrix multiplication scenario, the gradient tensor is transposed to produce a Row-wise outlier tensor,

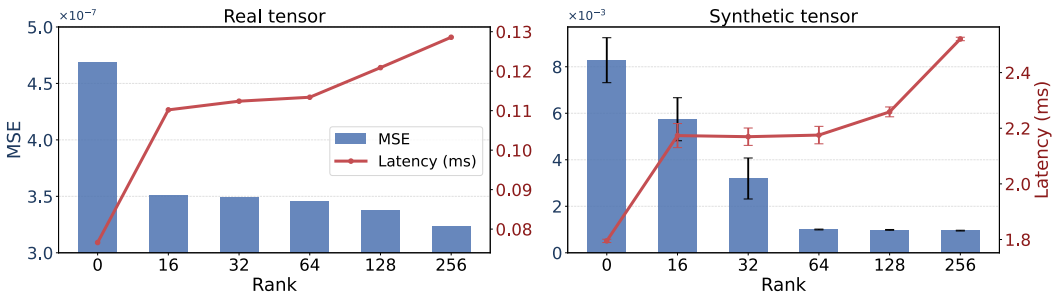


Figure 10: MSE and latency as a function of the OE extraction rank for (Left) real tensors from Llama3.1-8B and (Right) synthetic tensors. In both settings the MSE falls until rank 64 and plateaus after it, whereas the latency stays nearly flat between ranks 16 and 64 and rises beyond it.

while the activation tensor is used as-is as the Column-wise outlier tensor. In the left panel, the MSE falls from 4.69×10^{-7} at rank 0 to 3.46×10^{-7} at rank 64 while the latency rises only from 0.077 ms to 0.113 ms, whereas rank 128 costs 0.121 ms and rank 256 costs 0.129 ms for little further reduction in MSE. Rank 64 is therefore where the two curves separate on real tensors.

Synthetic Tensor. To simulate the MLP projection setting of Llama3.1-8B, we construct 4096×14336 synthetic tensors. The Column-wise outlier tensor is generated by amplifying random column magnitudes by $25\times$, and the Row-wise outlier tensor by amplifying random column magnitudes by $5\times$ and then transposing, since the two factors follow the kurtosis asymmetry between activations and gradients. We evaluate under the **RC** pattern pair using five random seeds (42, 256, 1925, 2048, 7777). In the right panel, the MSE falls from 8.28×10^{-3} at rank 0 to 1.00×10^{-3} at rank 64 and then plateaus, while the latency stays nearly flat between ranks 16 and 64 and rises at ranks 128 and 256, and the error bars over the five seeds are small enough that this ordering holds on each of them. The synthetic tensors therefore place the knee at the same rank as the real ones, which is why $r = 64$ is the default throughout AdaHOP.

N Precision Composition of Training FLOPs

Table 13 breaks the GEMM FLOPs of a full training run down by precision for the three Llama models. The full-precision GEMM column counts the CC pairs that Lv2 routes to BF16, and the outlier extraction column counts the BF16 outlier path of OE.

Table 13: Share of training GEMM FLOPs by precision (%).

Model	Level	MXFP4	Full-precision GEMM	Outlier extraction	BF16 total
Llama3.2-1B	Lv1	99.27	0.00	0.73	0.73
Llama3.2-1B	Lv2	98.95	0.33	0.72	1.05
Llama3.2-3B	Lv1	99.45	0.00	0.55	0.55
Llama3.2-3B	Lv2	96.59	2.92	0.49	3.41
Llama3.1-8B	Lv1	99.55	0.00	0.45	0.45
Llama3.1-8B	Lv2	94.01	5.63	0.36	5.99

All of Lv1's BF16 computation is the outlier path of OE. It stays below 1% at every scale and shrinks as the model grows, from 0.73% at 1B to 0.45% at 8B. The part of Lv2 that grows with scale is the full-precision GEMM for the CC pairs, from 0.33% to 5.63%, while its OE share also decreases. This is the cost side of the Lv1/Lv2 trade-off. The cost of Lv2 rises with the CC share, while its benefit appears to fall with scale as the post-IHT error on CC pairs shrinks: the downstream results show Lv2 ahead on Llama3.2-1B, Instella-3B, and Llama3.2-3B and Lv1 ahead on Qwen3-4B and Llama3.1-8B. Five points cannot establish a trend, so we report this ordering as an observation and settle the level by the residual CV rule at calibration time. Fixing either level still matches or beats each scale's strongest available baseline on all five models, so the choice sets the size of the margin rather than the outcome.

O Comparison with Prior Hadamard-Based LPT

Table 14 places AdaHOP among prior techniques by the unit at which each decides its treatment, the set of treatments it can choose from, whether the choice is derived from the training tensors, and whether the two operands of a GEMM can be treated differently.

Table 14: Decision unit and mode of adaptation of low-precision training methods.

Method	**Decision unit**	**Transform / handling set**	**Data-driven adaptation**	**Asymmetric**
Loss scaling, stochastic rounding	Global	Scaling / rounding rule	None	N/A
LLM.int8()	Feature dimension (inference)	Fixed decomposition (16-bit path)	Magnitude threshold	N/A
Tseng et al. (2025)	Same for all paths	IHT + SR	None	No
Quartet	Path level (fwd / bwd)	Fixed / random Hadamard	None	No
HALO	Path level (∇X)	OHT	None	No
HOT	Path level (backward only)	Hadamard + low-rank	Heuristic	No
AdaHOP	Operand pair (per GEMM)	IHT / OE-L / OE-R / BF16	30-step measurement (γ_b, η_r)	Yes

Prior methods fix the transform as a hyperparameter at the path level or globally, whereas AdaHOP decides it per operand pair from a measurement, and it is the only method in the table that handles the two operands asymmetrically. Its BF16 outlier path follows the mixed-precision decomposition of LLM.int8() (Dettmers et al., 2022), which isolates outlier feature dimensions into a 16-bit matmul at inference; AdaHOP carries this idea to training, to the MXFP4 format, and to structured row or column extraction combined with a Hadamard-smoothed residual. Table 15 measures how often the fixed choice is misaligned, counting the GEMMs of the three measured models whose pair falls in RN, RC, NC, or CC.

Table 15: GEMMs on which a uniform IHT assignment is misaligned with the operand pair.

Model	**Misaligned (all GEMMs)**	**Within ∇W**	**Within forward**
Llama3.2-1B	120 / 336 (35.7%)	112 / 112 (100.0%)	0 / 112 (0%)
Instella-3B	295 / 756 (39.0%)	249 / 252 (98.8%)	0 / 252 (0%)
Llama3.1-8B	257 / 672 (38.2%)	219 / 224 (97.8%)	0 / 224 (0%)

The uniform Hadamard assumption holds throughout the forward path and fails on 98 to 100% of the weight-gradient path, which may explain why prior work designed around the forward path did not encounter the failure. The same aggregation predicts baseline behavior: the ∇X path on which HALO applies uniform OHT is 75% CN pairs in Llama3.2-1B, and on CN the recomputed bound places OHT 3.6$\times$ behind IHT.