FLASHLIGHT: PRESERVING ACCEPTED PREFIXES IN LOW-BIT SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Target quantization accelerates speculative verification, making draft execution cost increasingly important. W4A4 drafting reduces this cost, but lower verifier agreement can offset the gain. On DFlash and DSpark, the conditional-acceptance gap to a 16-bit draft widens toward later positions even under basis-consistent rotation-based PTQ (Aligned PTQ). We call this phenomenon *quantization-induced suffix decay*. We propose **FLASHLight**, which reframes W4A4 draft quantization as selecting a low-bit representation for the deployed verifier rather than relying solely on reconstruction of the high-precision draft. With target and draft weights frozen, FLASHLight calibrates the quantization-sensitive interface, then selects an independent draft basis using verifier guidance and a prefix-coupled overlap objective for accepted-prefix preservation. On DFlash and DSpark, FLASHLight retains **97.21–97.68%** of the greedy-decoding AL of the 16-bit draft paired with the same W4A4 target, improving average AL over Aligned PTQ by approximately **11.8–22.8%**. Repeated measurements on a single RTX 4090 show **28.5–29.8%** higher mean end-to-end throughput at batch size 1 than that reference.

1 INTRODUCTION

Post-training quantization (PTQ) and speculative decoding reduce complementary costs in autoregressive (AR) decoding of large language models (LLMs). PTQ lowers the *cost per target call* through low-bit execution (Frantar et al., 2023), while speculative decoding reduces the *number of target calls* per output token by verifying multiple candidates proposed by a lightweight draft in a single pass (Leviathan et al., 2023; Chen et al., 2023). When these techniques are combined but only the target uses low-bit precision, the draft accounts for a larger share of execution cost. With a W4A4 target, the 16-bit drafts of EAGLE-3 (Li et al., 2025) and DFlash (Chen et al., 2026) account for 43.9% and 37.3% of the draft–verification core latency, respectively (Figure 2a).

However, the time per output token is approximately the cycle time divided by the effective acceptance length (AL). Draft quantization yields acceleration only when the relative reduction in cycle time exceeds the relative reduction in AL (Section 2.1). A quantized draft should therefore be evaluated by verifier-facing decoding efficiency, which depends critically on the prefixes accepted by the deployed verifier—the target that decides candidate acceptance—rather than by tensor reconstruction error alone.

Prior work combines quantization with speculative decoding in several ways, but separate feature-sharing drafts are often kept at higher precision or quantized only in weights (Section 5). Here we focus on W4A4 draft execution, where activation quantization makes preserving verifier-accepted prefixes substantially harder.

Our measurements show that this acceptance loss is position dependent. For DFlash (Chen et al., 2026) and DSpark (Cheng et al., 2026), our measured conditional-acceptance gap to a 16-bit draft under the same W4A4 target grows from 1.5 to 18.0 and from 6.1 to 13.3 percentage points, respectively, even under basis-consistent rotation-based quantization (Aligned PTQ) (Figure 1). We call this widening quantization penalty—rather than a generic decline in later-position acceptance—*quantization-induced suffix decay*. A DFlash same-prefix probe indicates that rollout prefix selection alone does not explain the position dependence. Because long accepted prefixes require consecutive acceptance, this degradation limits their survival (Section 2.2). We therefore formulate low-bit draft

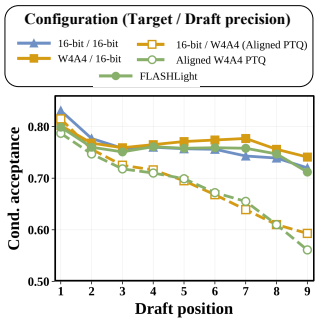
Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping 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.

quantization as accepted-prefix preservation under a fixed deployed verifier. Our goal is not to rely on reconstruction alone, but to optimize the low-bit draft representation for that verifier while retaining the shared W4A4 target–draft interface.

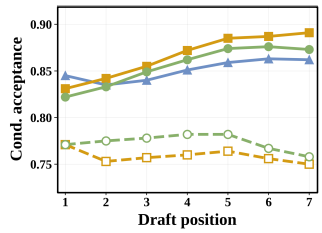
For feature-sharing drafts, accepted-prefix preservation exposes two constraints: a shared per-token A4 grid can under-resolve smaller-range sources, while conventional draft rotations are selected without verifier agreement. Because basis consistency permits an independent draft rotation through function-preserving fusion, the interface range and draft basis can be optimized separately (Section 2.3).

We address these constraints with **FLASHLight**, a range-first, verifier-guided W4A4 quantization method for pretrained feature-sharing drafts. FLASHLight first calibrates source scaling and activation clipping using a tensor-level reconstruction objective. With the calibrated interface parameters $S^\star$ and $c^\star$ fixed, it then optimizes an independent draft rotation using W4A4 forward passes. We use the QSD loss, which combines position-wise draft–verifier distribution overlaps through prefix products, as a soft surrogate for accepted-prefix preservation. The original target and draft weights remain frozen, and the optimized transformations are fused into existing weights and quantization kernels for deployment without a separate basis-conversion operator.

Across four datasets, FLASHLight retains 97.21–97.68% of the AL of the 16-bit draft paired with the same W4A4 target on DFlash and DSpark, improving average AL over Aligned PTQ by approximately 11.8–22.8%. Under a runtime protocol distinct from the AL evaluation, FLASHLight achieves 28.5–29.8% higher end-to-end throughput than the 16-bit draft at batch size 1 on a single RTX 4090. On tree-based EAGLE-3, FLASHLight retains 96.41% of the 16-bit-draft AL, although neither QSD over LK nor range-first ordering shows an established benefit (Appendix G).



(a) DFlash



(b) DSpark

Figure 1: Conditional acceptance across draft positions under greedy decoding for (a) DFlash (Llama-3.1-8B-Instruct, $K = 9$) and (b) DSpark (Qwen3-4B, $K = 7$).

Our contributions are as follows.

- **Quantization-induced suffix decay.** We identify a widening conditional-acceptance gap in feature-sharing W4A4 drafts and use same-prefix and interface analyses to diagnose the loss.
- **Verifier-aware low-bit representation optimization.** After range-first interface calibration, we select among full-precision-equivalent draft bases using verifier guidance to better preserve verifier-accepted predictions, with pretrained weights frozen and transformations fused for deployment.
- **Acceptance recovery and acceleration.** We evaluate DFlash and DSpark across four datasets and multiple batch sizes, with controlled comparisons examining the roles of range calibration and verifier supervision.

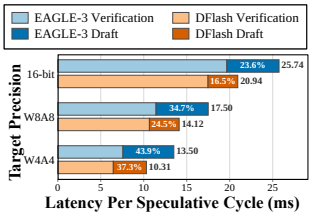
2 Background and Motivation

2.1 Draft Quantization Must Preserve Acceptance

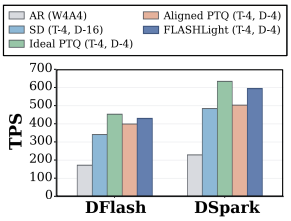
Speculative decoding alternates between draft proposal and target verification. We refer to the target in its role of deciding candidate acceptance as the verifier. We use effective acceptance length (AL) to denote the average number of output tokens committed per speculative cycle, including accepted draft tokens and any correction or bonus token actually emitted. Including drafting, verification, and other overheads, the time per output token is approximated by

$$T_{\text{token}} \approx \frac{T_{\text{cycle}}}{\text{AL}} = \frac{T_{\text{draft}} + T_{\text{verification}} + T_{\text{other}}}{\text{AL}}. \tag{1}$$

Draft quantization therefore yields acceleration only when the relative reduction in cycle time exceeds the relative reduction in AL (Appendix H.1).



(a) Core latency and draft share.



(b) Throughput potential.

Draft	Top-1 agreement ↑ d=1	Top-1 agreement ↑ d=9	Sign flip ↓ d=1	Sign flip ↓ d=9
16-bit	.837	.254	–	–
Aligned PTQ	.815	.170	.048	.467
Range-only	.825	.222	.035	.291
FLASHLight	.831	.235	.031	.254

(c) Same-prefix probe.

Figure 2: **Draft cost, throughput potential, and controlled acceptance loss.** (a) Core latency and 16-bit draft share across target precisions. (b) Batch-1 throughput; Ideal PTQ is an estimate that combines W4A4 draft execution cost with the 16-bit draft's AL. T/D denote target/draft precision (4 means W4A4); AR/SD denote autoregressive/speculative decoding, and TPS denotes output tokens/s. Panel (b) uses a different workload from Table 2 (Appendix H.1). (c) DFlash predictions on identical W4A4 target features and prefixes (24,799 anchors), without rollout survivor selection. Agreement is with the target's greedy token; sign flip is $\Pr(m_X < 0 \mid m_{\mathrm{REF16}} > 0)$, with m_X its draft-logit margin over the strongest competitor. Values are dataset-macro means; FLASHLight also averages three seeds. The same-prefix protocol is described in Section 2.2.

With a W4A4 target, the 16-bit draft accounts for 43.9% and 37.3% of the draft–verification core latency in EAGLE-3 and DFlash, respectively (Figure 2a). We use *Aligned PTQ* to denote a rotation-based W4A4 draft configuration that consistently accounts for the target-feature basis transformations in its computation (Section 2.3). Aligned PTQ reduces draft execution cost but falls short of Ideal PTQ, which combines that same execution cost with the 16-bit draft's AL (Figure 2b). Under this fixed-cost comparison, the throughput gap reflects the AL lost to draft quantization.

Weight-only quantization primarily reduces weight traffic, whereas W4A4 can also exploit the compute throughput of low-bit GEMMs. Reusing weights across multiple candidates increases arithmetic intensity in drafting and verification, potentially making 16-bit compute throughput a limiting factor (Figure 3). This hardware opportunity comes with a larger acceptance loss: in a separate DFlash precision-decomposition run, the AL reductions relative to that run's 16-bit draft are 1.4% for W4A16 and 15.1% for W4A4 (Appendix E.2). Further acceleration therefore requires preserving accepted prefixes while retaining low draft execution cost.

2.2 Aligned PTQ Still Exhibits Quantization-Induced Suffix Decay

Conditional acceptance A_k is the acceptance rate at position k among cycles that accept the preceding $k-1$ draft tokens and verify position k. With the W4A4 target fixed, we define

$$\Delta A_k = A_k^{\text{16-bit draft}} - A_k^{\text{W4A4 draft}}. \tag{2}$$

Here, QSD refers to the growth of the quantization-induced gap ΔA_k toward later positions; it does not require the raw conditional acceptance $A_k^{\text{W4A4 draft}}$ itself to decrease monotonically. The widening gap in Figure 1 persists under Aligned PTQ. Because A_k also depends on the prefixes reaching each position, we use additional controls to interpret this trend.

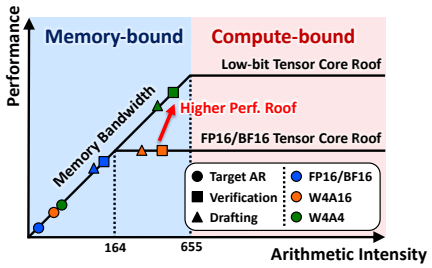


Figure 3: Theoretical roofline for DFlash (Llama-3.1-8B-Instruct) on an RTX 4090 (batch size 16, prompt length 512, verification budget 10; Appendix H.3).

Target-only quantization does not show the same additional suffix-biased loss, whereas pairing the Aligned PTQ draft with a 16-bit target still yields a widening gap (Appendix E).

Controlling prefix selection. We give all DFlash configurations identical W4A4 target features and prefixes from the target's fixed greedy continuations (up to 512 tokens). Using the first turns of prompts 0–31 in each dataset and excluding three continuations shorter than 10 tokens leaves 125 prompts. Every generated position except the final nine serves as an anchor, yielding 24,799 shared anchors; all nine draft slots are evaluated without conditioning on earlier proposals or acceptance.

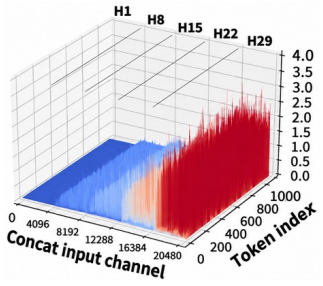
Figure 2c reports top-1 agreement over all shared anchors. Sign flips are measured among strictly positive-margin 16-bit-reference predictions, with exact ties excluded from flips. Aligned PTQ's sign-flip rate rises from 0.048 at $d = 1$ to 0.467 at $d = 9$, where its top-1 agreement is 0.170 versus the reference's 0.254. Thus, greater late-position degradation persists without rollout survivor selection in this controlled DFlash probe. A controlled precision-site intervention further shows that Range-only nearly matches the counterfactual with a 16-bit interface in final-position sign flips (Appendix E.5).

Long accepted prefixes require consecutive acceptance. Among DFlash cycles with an untruncated verification horizon, the dataset-average fraction accepting all nine draft tokens falls from 10.30% with the 16-bit draft to 3.49% with Aligned PTQ (Table 20). Recovering first-position accuracy or average tensor fidelity alone therefore does not ensure prefix survival. We next examine the interface range and draft-basis choices available to address this loss.

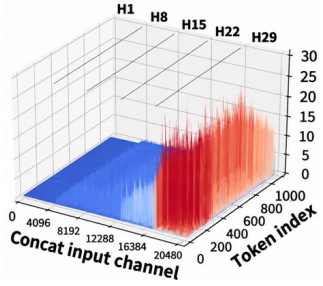
2.3 Two Interface Constraints: Range and Basis

EAGLE-3, DFlash, and DSpark concatenate hidden features from multiple target layers, referred to as sources, to form the draft input. Let $\mathbf{H} = [\mathbf{H}_1 \,|\, \cdots \,|\, \mathbf{H}_m]$ denote the concatenated features from m sources and W_c the original feature projection. Let $\overline{R}_T$ be the block-diagonal matrix containing the W4A4 target's per-source orthogonal rotations. The draft receives $\widetilde{\mathbf{H}} = \mathbf{H}\overline{R}_T$. We refer to the path that quantizes these features to A4 and passes them through the feature projection as the *target–draft interface* (Appendix B).

Shared range. Heterogeneous sources (Figure 4) share one per-token A4 grid (Appendix C.2); a large-range source can reduce smaller-range sources' effective resolution. Draft rotation follows input quantization and cannot change this grid or directly recover discarded information. Weight retraining can absorb basis changes but leaves the shared grid unchanged (Appendix I). Range calibration alone reduces the final-position sign-flip rate from 0.467 to 0.291 in the DFlash same-prefix probe (Figure 2c). Group-wise quantization and fine-grained mixed precision require scale- or precision-aware execution (Lin et al., 2025; Liu et al., 2025a). FLASHLight instead retains per-token A4 and per-output-channel W4 (Appendix H.7).



(a) Llama-3.1-8B-Instruct



(b) Qwen3-4B

Figure 4: Per-source range differences in the concatenated target features of (a) DFlash/Llama-3.1-8B-Instruct and (b) DSpark/Qwen3-4B. The number following H identifies the target layer from which the feature is extracted.

Independent draft basis. For an orthogonal R, rotation-based PTQ exploits $xW = (xR)(R^\top W)$ to redistribute activations concentrated in individual channels while preserving the pre-quantization computation (Ashkboos et al., 2024; Liu et al., 2025b). Draft activations also contain outliers (Appendix C.1), motivating the use of rotations for W4A4 draft quantization. When the target uses rotation, its inverse must be incorporated into the projection weight so that $\widetilde{\mathbf{H}}\,\overline{R}_T^\top W_c = \mathbf{H}W_c$. This is the basis alignment required by Aligned PTQ. However, incorporating an independent draft rotation R_D gives

$$W_c' = \overline{R}_T^\top W_c R_D, \tag{3}$$

which satisfies $\widetilde{\mathbf{H}}W_c' = \mathbf{H}W_cR_D$. With corresponding transformations applied to the remaining draft paths, this permits a draft basis independent of the target without changing the projection shape or GEMM count (Appendix B).

Design requirements. Bases representing the same full-precision computation can yield different predictions after quantization (Appendix B.4). Aligned PTQ's initial draft basis is not selected according to verifier acceptance (Table 3); related rotation-based PTQ methods are discussed in Section 5. Two design requirements follow. First, draft-basis selection cannot substitute for interface range calibration; we calibrate the range-control parameters separately and fix them before basis optimization to match the input conditions used during training and deployment. Second, among

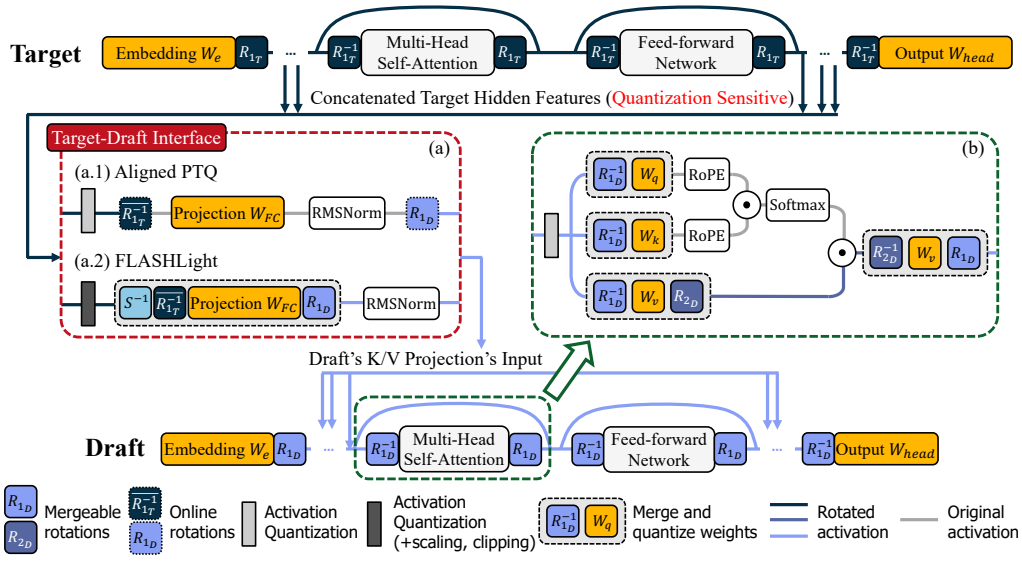


Figure 5: **Computational structure of FLASHLight, illustrated with DFlash.** (a) Target–draft interface: Aligned PTQ (a.1) aligns only the basis before quantization, whereas FLASHLight (a.2) first calibrates source scaling and clipping, and then fuses the QSD-optimized draft rotation into the projection weight. (b) The corresponding transformations are also fused into the internal draft weights.

full-precision-equivalent draft bases, we select a basis using verifier guidance toward the accepted-prefix preservation objective.

3 FLASHLIGHT

The two constraints require different remedies: range calibration reduces shared-grid resolution loss, then verifier guidance selects among full-precision-equivalent draft bases (Figure 5). The deployed verifier and original target and draft weights remain fixed, and the resulting transformations are fused for deployment.

3.1 LOW-BIT INTERFACE REPARAMETERIZATION

Let $X = \widetilde{\mathbf{H}}$. The source scale $S = \mathrm{blkdiag}(\alpha_1 I_{d_1}, \ldots, \alpha_m I_{d_m})$, where $\alpha_i > 0$ and d_i is the dimension of source i, and the clipping ratio c are determined during calibration and then fixed. Only the set of orthogonal draft rotations $\mathcal{R}_D = \{R_r\}_{r\in\mathcal{I}}$, including the residual draft rotation R_D and architecture-specific additional rotations, is optimized. Let $Q_A^{(c)}$ and Q_W denote the activation and weight quantize–dequantize operators, respectively. The FLASHLight feature projection is

$$\widehat{\mathbf{Y}}(R_D, S, c) = Q_A^{(c)}(XS)Q_W(S^{-1}\overline{R}_T^{\top}W_c R_D). \tag{4}$$

Here, S controls how the shared A4 range is allocated across sources, $\overline{R}_T^{\top}$ accounts for the target-side basis rotation in the feature projection, and R_D selects the draft basis. Before quantization, $(XS)(S^{-1}\overline{R}_T^{\top}W_c R_D) = \mathbf{H}W_c R_D$. Applying the corresponding transformations to the remaining draft paths (Appendix B) therefore preserves the full-precision function for any choice of S and R_D, while different choices generally produce different low-bit representations. FLASHLight exploits this representation space by determining the range-control parameters (S, c) and the draft basis R_D in two separate stages.

3.2 RANGE-FIRST INTERFACE CALIBRATION

The range parameters affect both the features received by the draft and the operations that consume them, such as the context K/V path in DFlash. FLASHLight therefore calibrates these parameters at

the initial Aligned-PTQ draft rotation $R_{D,0}$ and keeps them fixed throughout subsequent rotation optimization and deployment. Source scaling applies equivalent activation/weight scaling (Xiao et al., 2023; Sun et al., 2025) at the source-block level, while the range parameters are calibrated without verifier supervision by minimizing a tensor-level reconstruction error. With $W_{b,0} = \overline{R}_T^\top W_c R_{D,0}$, the objective for DFlash and DSpark is

$$\mathcal{J}_{\mathrm{proj}}(S, c \mid R_{D,0}) = \frac{\left\| Q_A^{(c)}(XS) Q_W(S^{-1} W_{b,0}) - X W_{b,0} \right\|_F^2}{\|X W_{b,0}\|_F^2 + \epsilon_{\mathrm{cal}}}. \tag{5}$$

We first select $S^\star = \arg\min_S \mathcal{J}_{\mathrm{proj}}(S, 1 \mid R_{D,0})$ with $c = 1$, and then select $c^\star = \arg\min_c \mathcal{J}_{\mathrm{proj}}(S^\star, c \mid R_{D,0})$. The search spaces and architecture-specific settings—EAGLE-3 uses a different tensor-level objective—are detailed in Appendices C.3, C.7, and G.

On DFlash's sink-excluded calibration inputs, source scaling achieves reconstruction error comparable to SmoothQuant-style per-channel scaling while using 5 rather than 20,480 scalars; this comparison does not measure acceptance (Appendix C.4). At fixed source scales, initial draft basis, and W4 weights, clipping fewer than 1% of interface activation values produces more uniform A4 code usage and lowers activation-induced projection NMSE on sink-excluded diagnostic inputs (Appendices C.8–C.10).

3.3 Verifier-Guided Draft-Basis Optimization

With the calibrated interface parameters $S^\star, c^\star$ fixed, the remaining task is to select a draft basis that preserves prefixes accepted by the deployed verifier. We search among full-precision-equivalent bases using the quantized verifier's distributions as supervision. Let i index prediction blocks anchored at context-token positions, and d the position within each block. Let $p_{i,d}$ be the frozen quantized-verifier distribution and $q_{i,d}(\cdot; \mathcal{R}_D)$ the W4A4 draft distribution. Their overlap over the shared vocabulary $\mathcal{V}$ is

$$a_{i,d} = \sum_{v \in \mathcal{V}} \min(p_{i,d}(v), q_{i,d}(v)) = 1 - \mathrm{TV}(p_{i,d}, q_{i,d}), \tag{6}$$

where TV denotes total variation distance. We use this verifier–draft overlap as a position-wise signal for selecting among full-precision-equivalent draft bases. It is not an estimator of measured acceptance, but a soft training surrogate for preserving greedy accepted prefixes (Appendix D.3).

To couple these position-wise signals across prefixes, we combine the overlaps through prefix products. The valid positions for each anchor form a contiguous prefix of length K_i. Let $\mathcal{I}_+$ denote the set of anchors with $K_i > 0$ and $N = |\mathcal{I}_+|$. The QSD loss is

$$\mathcal{L}_{\mathrm{QSD}} = -\frac{1}{N} \sum_{i \in \mathcal{I}_+} \frac{1}{K_i} \sum_{k=1}^{K_i} \prod_{d=1}^{k} a_{i,d}. \tag{7}$$

The QSD loss neither estimates the position-wise quantization gap nor assigns uniformly larger weights to later positions. Instead, an improvement at a later position is rewarded together with the overlap of the prefix required to reach that position.

The prefix-product form follows prior acceptance-aware objectives and is not itself our claimed contribution; our distinction is its use for verifier-guided basis optimization with frozen pretrained weights (Section 5; Appendix D.3). We maintain orthogonality through a Cayley parameterization (Wen & Yin, 2013) and backpropagate through quantization using a straight-through estimator (Bengio et al., 2013). Implementation details are provided in Appendix D and Algorithm 1.

3.4 Fused Deployment

We form $W_c^{\mathrm{FL}} = (S^\star)^{-1} \overline{R}_T^\top W_c R_D^\star$ by offline fusion and apply activation-side scaling and clipping within the existing A4 quantization kernel, without a separate basis-conversion operator. DFlash and EAGLE-3 retain the same kernel-launch counts as Aligned PTQ (Appendix B.8).

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate acceptance and prefix recovery, examine the roles of range calibration and verifier guidance, and measure end-to-end acceleration. The target models are Llama-3.1-8B-Instruct (Grattafiori et al., 2024) for DFlash and Qwen3-4B (Yang et al., 2025) for DSpark. DFlash proposes $K = 9$ tokens and DSpark proposes $K = 7$, with a threshold of 0 and no pruning for DSpark. We measure AL under greedy decoding ($T = 0$) on MT-Bench (Zheng et al., 2023), GSM8K (Cobbe et al., 2021), ShareGPT, and HumanEval (Chen et al., 2021), and report the equally weighted mean of the four dataset-level AL values as Avg. AL. Both targets use W4A4 quantization with SpinQuant rotations (Liu et al., 2025b). We compare four configurations: a 16-bit draft paired with the W4A4 target (the *16-bit draft reference*), Aligned PTQ, Range-only, and FLASHLight. Range-only applies the same $S^\star, c^\star$ as FLASHLight without further rotation optimization. Within each architecture, all W4A4 draft configurations use the same verifier, proposal settings, and evaluation prompts. Recovery is the AL ratio relative to the 16-bit draft reference, expressed as a percentage; FLASHLight results are averaged over three training seeds. For DFlash and DSpark, the calibration, rotation-training, validation, and evaluation prompt pools are text-disjoint (Appendix A.2). Additional settings and reporting conventions are provided in Appendix A.

4.2 ACCEPTANCE AND PREFIX RECOVERY

Table 1: Greedy ($T = 0$) acceptance. FLASHLight results are averaged over three training seeds. Table 10 provides the complete results, including those with a 16-bit target.

Method	MT-Bench	GSM8K	ShareGPT	HumanEval	Avg. AL	Recovery (%)
Llama-3.1-8B-Instruct + DFlash						
16-bit draft reference	3.97	4.01	3.79	5.26	4.26	100.0
Aligned PTQ	3.53	3.52	3.40	4.43	3.72	87.4
Range-only	3.73	3.79	3.57	4.93	4.00	94.1
FLASHLight	3.85	3.96	3.70	5.12	**4.16**	**97.7**
Qwen3-4B + DSpark						
16-bit draft reference	4.90	5.09	4.75	4.77	4.88	100.0
Aligned PTQ	3.88	4.10	3.69	3.76	3.86	79.1
Range-only	4.65	4.83	4.48	4.48	4.61	94.5
FLASHLight	4.76	5.02	4.60	4.59	**4.74**	**97.2**

FLASHLight reaches Avg. AL 4.16 and 4.74 on DFlash and DSpark versus 3.72 and 3.86 for Aligned PTQ, corresponding to 11.8% and 22.8% improvements and **97.68%** and **97.21%** recovery (Table 1). It exceeds Aligned PTQ on every dataset; draft variants are compared under the same W4A4 verifier because target quantization shifts AL differently by dataset (Appendix E.1).

Range calibration and verifier-guided basis optimization. The stages address distinct portions of the W4A4 loss: Range-only closes 53% and 74% of the gap to the 16-bit draft on DFlash and DSpark, while, with that interface fixed, verifier guidance adds 0.154 and 0.133 AL, respectively, with positive gains for all three seeds. Thus, range calibration recovers a large portion of the W4A4 acceptance loss, while verifier guidance further reduces residual disagreement (Appendix F.1).

Recovery at late draft positions. FLASHLight also recovers conditional acceptance at the later positions most affected by quantization-induced suffix decay (Figure 1). At the final position, conditional acceptance increases from 0.561 to 0.712 on DFlash and from 0.758 to 0.873 on DSpark, closing 84% and 87% of the respective gaps to the 16-bit draft reference (0.741 and 0.891). Across positions, the remaining ΔA_k ranges from 0.2–2.8 percentage points on DFlash and 0.6–1.7 percentage points on DSpark, with the largest gap still at the final position in both architectures (based on unrounded pooled values). In a separate dataset-macro aggregation over DFlash cycles with an untruncated verification horizon, the fraction accepting all nine draft tokens increases from 3.49% for Aligned PTQ to 8.88%, approaching the 16-bit draft reference of 10.30% (Table 20). Most of this late-position recovery comes from range calibration: Range-only closes 74% and 83% of the final-position gap on DFlash and DSpark, respectively. On DSpark, the additional improvement from rota-

Table 2: E2E throughput on a single RTX 4090 (tokens/s, mean over repeated runs). FLASHLight uses the seed-0 checkpoint. The 16-bit draft uses vendor BLAS; W4A16 GPTQ uses Marlin (Frantar et al., 2025) with per-channel W4 GPTQ on a basis-aligned draft; W4A4 uses a native INT4 custom backend. Parentheses give throughput ratios relative to the 16-bit draft; bold indicates the highest mean in each row.

Architecture	Batch	16-bit draft	W4A16 GPTQ	W4A4 Aligned PTQ	W4A4 FLASHLight
DFlash	1	340.6	432.8 ($\times$1.271)	388.6 ($\times$1.141)	**442.1** ($\times$1.298)
	2	631.9	769.1 ($\times$1.217)	713.5 ($\times$1.129)	**802.6** ($\times$1.270)
	4	1100.8	1255.8 ($\times$1.141)	1190.9 ($\times$1.082)	**1308.6** ($\times$1.189)
	8	1626.3	1739.3 ($\times$1.069)	1664.8 ($\times$1.024)	**1848.5** ($\times$1.137)
	16	OOM	1991.2	1923.4	**2120.9**
DSpark	1	516.3	**664.1** ($\times$1.286)	530.8 ($\times$1.028)	663.6 ($\times$1.285)
	2	964.6	**1192.4** ($\times$1.236)	1006.1 ($\times$1.043)	1191.9 ($\times$1.236)
	4	1605.5	1872.7 ($\times$1.166)	1660.0 ($\times$1.034)	**1941.6** ($\times$1.209)
	8	2437.4	2655.4 ($\times$1.089)	2439.1 ($\times$1.001)	**2764.3** ($\times$1.134)
	16	3196.0	3243.6 ($\times$1.015)	3094.5 ($\times$0.968)	**3490.7** ($\times$1.092)

tion optimization is not concentrated at later positions (Appendix E.4). A seed-0 K-sweep preserves FLASHLight $>$ Range-only $>$ Aligned PTQ across all tested proposal lengths (Appendix E.6).

4.3 Understanding FLASHLight

Why verifier guidance matters. Under matched training conditions with a TV objective, replacing the 16-bit-draft teacher with the deployed W4A4 verifier increases DFlash Avg. AL from 4.085 to 4.135 (paired difference $+0.050$; 95% t interval across three training seeds $[+0.018, +0.082]$; Table 19). Only the teacher changes; the initial rotation, range, training prompts, anchor normalization, and update budget are held fixed. The corresponding teacher advantage is not observed with KL (Appendix F.2).

What the learned representation preserves. In the controlled DFlash probe (Section 2.2), FLASHLight improves verifier agreement despite moving farther from the 16-bit draft outputs. For mean-centered logits $\mathbf{u}$ of the 16-bit draft and $\mathbf{w}$ of each W4A4 configuration, logit distance is $\|\mathbf{w}-\mathbf{u}\|_2/\|\mathbf{u}\|_2$. Relative to Range-only, this distance, averaged over nine positions and then equally across datasets, increases from 0.302 to 0.314. At $d = 9$, however, soft overlap over the full vocabulary (Equation 6) increases from 0.174 to 0.198, top-1 agreement from 0.222 to 0.235, and the sign-flip rate falls from 0.291 to 0.254 (Figure 2c). These controls show that reconstructing the 16-bit draft is not the right optimization target: verifier agreement can improve even as reconstruction worsens.

Other controls. With the same QSD objective, range-first yields a DFlash AL gain of $+0.045$ over rotation-first, which optimizes rotations at $S = I, c = 1$ before calibrating the range (95% t interval $[+0.005, +0.086]$). DSpark shows the same direction, but the difference is not established. In matched DFlash comparisons, QSD also achieves higher AL than the matched anchor-normalized LK, KL, and CE objectives, as well as TV/KL objectives that use the 16-bit draft rather than the verifier as the teacher (Samarin et al., 2026). The mean gain over the matched TV objective, which omits only the prefix product, is $+0.023$, but its t interval includes zero. We therefore do not claim an independently established advantage of the prefix-product structure; detailed controls are provided in Appendix F.

4.4 End-to-End Performance

Within each architecture, we measure end-to-end throughput using the same W4A4 target under a protocol distinct from the AL evaluation. Throughput is the total number of output tokens across 604 requests divided by elapsed time, including prefill (Appendix A.4). At batch size $B = 1$, FLASHLight achieves **29.8%** and **28.5%** higher throughput than the 16-bit draft on DFlash and DSpark, respectively (Table 2). On DFlash, it also achieves 13.8% higher throughput than Aligned PTQ on the same W4A4 execution path, consistent with acceptance recovery contributing to end-to-end acceleration.

FLASHLight's relative performance varies with batch size. Its gain over the 16-bit draft decreases as the batch grows, reaching 13.7% and 13.4% at $B = 8$ on DFlash and DSpark, respectively. In contrast, its gain over W4A16 GPTQ on DFlash grows from 2.1% at $B = 1$ to 6.5% at $B = 16$. On DSpark, its throughput is less than 0.1% lower at $B = 1$ and $B = 2$, but 3.7%, 4.1%, and 7.6% higher at $B = 4, 8,$ and 16, respectively. This trend is consistent with the roofline interpretation in Figure 3: low-bit activation execution can become more beneficial as batch-level parallelism increases. Because W4A16 GPTQ and FLASHLight differ in both AL and backend, these differences reflect complete end-to-end configurations, not the isolated effect of activation precision. Run-to-run variability is also large for DSpark at $B = 16$ (Table 27).

5 Related Work

Draft quantization. QSpec and SpecMQuant quantize targets but retain separate EAGLE-style drafts in 16-bit in the relevant comparisons (Zhao et al., 2025; Zhang et al., 2025); weight-only draft quantization often causes smaller acceptance changes (Tang et al., 2025; Kim et al., 2026). FLASHLight instead targets pretrained feature-sharing W4A4 drafts and optimizes them for the deployed verifier with frozen weights.

Low-bit representations. SmoothQuant and FlatQuant reshape ranges by equivalent scaling (Xiao et al., 2023; Sun et al., 2025), while QuaRot, SpinQuant, and OSTQuant use fixed or learned rotations (Ashkboos et al., 2024; Liu et al., 2025b; Hu et al., 2025). Fine-grained grouping or mixed precision is another design point; QServe, MixLLM, and COMET provide scale- or precision-aware execution (Lin et al., 2025; Zheng et al., 2026; Liu et al., 2025a). FLASHLight instead reparameterizes the interface while retaining per-token A4 and per-output-channel W4.

Verifier-aware objectives. LK losses and end-to-end TV optimize draft or prediction-head weights toward acceptance (Samarin et al., 2026; Li et al., 2026), so QSD's prefix-product form is not our claimed novelty. FLASHLight instead freezes pretrained weights and uses the deployed quantized verifier to select a function-preserving low-bit basis; verifier agreement can improve even when 16-bit-draft reconstruction worsens (Section 4.3).

6 Limitations and Future Work

Our evaluation is limited to two target models and three drafting architectures under greedy decoding and fixed proposal configurations on a single RTX 4090. Further evaluation is needed with sampling-based decoding, broader proposal lengths and pruning strategies, and other target–draft combinations and accelerators.

The soft overlap is only a surrogate for greedy accepted-prefix preservation. On tree-based EAGLE-3, the overall FLASHLight procedure improves acceptance, but neither QSD over LK nor range-first ordering shows an established advantage. This suggests that the most effective verifier-aware surrogate and optimization ordering may depend on proposal topology. Our current EAGLE-3 optimization follows a greedy chain whereas evaluation uses tree verification, and we have not isolated the effect of this structural difference. Extending verifier-aware representation optimization to objectives that directly model tree-structured acceptance is a natural direction for future work.

FLASHLight's representation space is restricted to source scaling, activation clipping, and draft rotations. Its end-to-end performance relative to W4A16 GPTQ depends on batch size and backend. Richer fusible transformations and hardware-aware precision selection remain directions for future work.

7 Conclusion

Quantization-induced suffix decay in DFlash and DSpark motivates treating low-bit draft quantization as accepted-prefix preservation under the deployed verifier. FLASHLight combines range-first interface calibration with verifier-guided basis optimization while keeping the original target and draft weights frozen. It retains 97.21–97.68% of the 16-bit-draft AL under the same W4A4 target and improves batch-1 end-to-end throughput by 28.5–29.8%. These results support verifier-aware low-bit representation selection over full-precision reconstruction alone.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript drafting and revision, literature retrieval and discovery, mathematical reasoning and exposition, code development, experiment planning and methodology feedback, and result interpretation. All AI-assisted outputs, including technical claims, mathematical derivations, implementations, experimental configurations, and reported results, were independently reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Algorithm 1 specifies range calibration followed by QSD-based rotation learning; Equation 7 defines the objective and its valid-length normalization. Table 10 reports acceptance across four datasets; FLASHLight entries are means over training seeds 0, 1, and 2, each evaluated at its final step-400 checkpoint without validation-based selection, and the Range-only rows apply the same $S^\star, c^\star$ without rotation training. Appendix F reports the component ablation (Table 17), the objective comparison under matched normalization, initialization, range, training rows, and update budget (Table 18), and the calibration-order ablation (Table 21); Appendix E provides the rollout position-wise diagnostic data and their aggregation rules; Section 2.2 and Figure 2c provide the controlled same-prefix probe. Appendix E.6 reports the seed-0 proposal-length sweeps and distinguishes $\mathbb{E}[L]$ from emitted-token AL. Architecture-specific checkpoints, quantizers, data, and precision exceptions are listed in Table 3. For DFlash, the draft-model code matches DFlash @94e4abc, while the speculative-decoding loop and W4A4 runtime use local FLASHLight modifications. The ablation, objective, and order comparisons follow the reporting conventions in Appendix A unless their sections state otherwise. The same-prefix protocol and agreement/sign-flip definitions appear in Section 2.2 and Figure 2c; Appendix E.5 reports the controlled precision-site intervention, and Section 4.3 defines the logit-distance diagnostic. Results for EAGLE-3 are reported in Appendix G.

All FLASHLight results, including acceptance, end-to-end throughput, and supporting analyses, use the range-first QSD configuration; the kernel-launch counts in Appendix B.8 were measured on an earlier checkpoint that shares the same execution graph. Alternative objectives are explicitly labeled as comparison arms. AL is a macro average over four datasets; recovery uses unrounded aggregates. Runtime throughput is computed from the total output tokens and the elapsed time of each run, including prefill, on a 604-request workload (524 first turns and 80 MT-Bench second turns, at most 256 new tokens, continuous batching, 8 warm-up requests excluded). The mean throughput across repeated runs is reported for each method and batch, as described in Appendix A.4; the FLASHLight column uses the seed-0 checkpoint and three repeats. DFlash, EAGLE-3, and DSpark use the native INT4 custom backend for W4A4 execution.

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. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b5b939436789f76f08b9d0da5e81af7c-Abstract-Conference.html.

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

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. URL https://arxiv.org/abs/2302.01318.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.06036.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri,

Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, Qinyu Chen, Chengqi Deng, Jiping Yu, Damai Dai, Zhengyan Zhang, Yixuan Wei, Yixuan Tan, Wenkai Yang, Runxin Xu, Yu Wu, Zhean Xu, Xuanyu Wang, Muyang Chen, Rui Tian, Xiao Bi, Zhewen Hao, Shaoyuan Chen, Huanqi Cao, Wentao Zhang, Anyi Xu, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. DSpark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026. URL https://arxiv.org/abs/2607.05147.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.17323.

Elias Frantar, Roberto L. Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. MARLIN: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pp. 239–251. Association for Computing Machinery, 2025. doi: 10.1145/3710848.3710871. URL https://doi.org/10.1145/3710848.3710871.

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. URL https://arxiv.org/abs/2407.21783.

Xing Hu, Yuan Cheng, Zhixuan Chen, Dawei Yang, Jiangyong Yu, Chen Xu, Zhihang Yuan, Zhe Jiang, and Sifan Zhou. OSTQuant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5cebc89b113920dbff7c79854ba765a3-Abstract-Conference.html.

Huawei. msModelSlim: Adapt rotation. msModelSlim Documentation, 2026. URL https://www.hiascend.com/document/detail/en/mindstudio/2610/TITools/msModelSlim/docs/en/user_guide/quantization_algorithms/outlier_suppression_algorithms/adapt_rotation.md. Accessed 2026-09-26.

Jaeyeon Kim, Jewon Lee, and Bo-Kyeong Kim. Quantize the target, quantize the drafter: Efficient inference with Qwen3.5-4B, 2026. URL https://arxiv.org/abs/2607.04244.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The Third International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Yucheng Li, Huiqiang Jiang, Yang Xu, Jianxin Yang, Yi Zhang, Yizhong Cao, Yuhao Shen, Fan Zhou, Rui Men, Jianwei Zhang, An Yang, Bowen Yu, Bo Zheng, Fei Huang, Junyang Lin, Dayiheng Liu, and Jingren Zhou. Breaking entropy bounds: Accelerating RL training via MTP with rejection sampling, 2026. URL https://arxiv.org/abs/2606.12370.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c7b5a35ea98b62512a869c19ea7b03cb-Abstract-Conference.html.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/fbe2b2f74a2ece8070d8fb073717bda6-Abstract-Conference.html.

Lian Liu, Long Cheng, Haimeng Ren, Zhaohui Xu, Yudong Pan, Mengdi Wang, Xiaowei Li, Yinhe Han, and Ying Wang. COMET: Towards practical W4A4KV4 LLMs serving. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. Association for Computing Machinery, 2025a. doi: 10.1145/3676641.3716252. URL https://doi.org/10.1145/3676641.3716252.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=ogO6DGE6FZ.

Alexander Samarin, Sergei Krutikov, Anton Shevtsov, Sergei Skvortsov, Filipp Fisin, and Alexander Golubev. LK losses: Direct acceptance rate optimization for speculative decoding. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.23881.

Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x. URL https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. FlatQuant: Flatness matters for LLM quantization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57587–57613. PMLR, 2025. URL https://proceedings.mlr.press/v267/sun25l.html.

Bangsheng Tang, Carl Chengyan Fu, Fei Kou, Grigory Sizov, Haoci Zhang, Jason Park, Jiawen Liu, Jie You, Qirui Yang, Sachin Mehta, Shengyong Cai, Xiaodong Wang, Xingyu Liu, Yunlu Li, Yanjun Zhou, Wei Wei, Zhiwei Zhao, Zixi Qi, Adolfo Victoria, Aya Ibrahim, Bram Wasti, Changkyu Kim, Daniel Haziza, Fei Sun, Giancarlo Delfin, Emily Guo, Jialin Ouyang, Jaewon Lee, Jianyu Huang, Jeremy Reizenstein, Lu Fang, Quinn Zhu, Ria Verma, Vlad Mihailescu, Xingwen Guo, Yan Cui, Ye Hu, and Yejin Lee. Efficient speculative decoding for Llama at scale: Challenges and solutions, 2025. URL https://arxiv.org/abs/2508.08192.

Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1–2):397–434, 2013. doi: 10.1007/s10107-012-0584-1. URL https://doi.org/10.1007/s10107-012-0584-1.

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 *Proceedings of*

the 40th International Conference on Machine Learning, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

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

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1910.07467.

Yudi Zhang, Weilin Zhao, Xu Han, Tiejun Zhao, Wang Xu, Hailong Cao, and Conghui Zhu. Speculative decoding meets quantization: Compatibility evaluation and hierarchical framework design. *arXiv preprint arXiv:2505.22179*, 2025. URL https://arxiv.org/abs/2505.22179.

Juntao Zhao, Wenhao Lu, Sheng Wang, Lingpeng Kong, and Chuan Wu. QSpec: Speculative decoding with complementary quantization schemes. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4779–4795. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.240. URL https://aclanthology.org/2025.emnlp-main.240/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2306.05685.

Zhen Zheng, Xiaonan Song, and Chuanjie Liu. MixLLM: LLM quantization with global mixed-precision between output-features and highly-efficient system design. In *Proceedings of Machine Learning and Systems*, volume 8, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/a66caa1703fe34705a4368c3014c1966-Abstract-Conference.html.

A Experimental Setup and Reporting Conventions

Reporting conventions. Unless otherwise stated, FLASHLight refers to the range-first QSD configuration, with $S^\star, c^\star$ selected at the initial draft basis $R_{D,0}$ and then fixed. FLASHLight AL is averaged over evaluations of the final step-400 checkpoints from training seeds 0, 1, and 2, whereas throughput, latency, and memory are measured using the seed-0 checkpoint. All rotation-optimization runs, including those for comparison objectives and calibration-order ablations, are evaluated at the final checkpoint after 400 updates. Held-out validation loss is recorded only to monitor training progress, not to select checkpoints or seeds. Neither range calibration nor rotation optimization uses test data.

For each architecture, AL is the macro average of the four dataset-level AL values, evaluated with the same W4A4 target on the same 604-turn test split. Single-seed results are marked as screening results. AL values in Tables 1, 10, and 23 are displayed to two decimal places. Recovery is computed from unrounded aggregates, whereas relative AL improvements are computed from the displayed means. ShareGPT evaluation uses revision `8b0048a` of `https://huggingface.co/datasets/Aeala/ShareGPT_Vicuna_unfiltered`.

A.1 Models, Checkpoints, and Execution Settings

Table 3 summarizes the model, quantization, data, and execution settings for the three architectures.

For DFlash, the draft-model implementation matches DFlash @94e4abc; the speculative-decoding loop and W4A4 runtime use local FLASHLight modifications. The trainable rotations and calibration settings are summarized in Table 4 of Appendix A.3.

A.2 Disjoint Data

For DFlash and DSpark, the prompt pools for range calibration, rotation optimization, validation, and evaluation are disjoint at the text level. The two architectures share the same rotation-training and validation prompt pools and the same evaluation file. DSpark calibrates the range on 96 prompts. DFlash uses the same 96-prompt pool for the lattice search over S, then re-checks S and selects c on a separate, disjoint set of 178 prompts. Rotation optimization uses 192 prompts; validation for monitoring learning curves uses 88 prompts; evaluation uses 604 turns. No overlap is detected in any of the 2,928 checks across predefined pool pairs and matching levels, including these pools. The checks cover exact text, normalized text, source IDs, 64-token prefixes, and containment; token-level checks use the Llama tokenizer.

EAGLE-3 uses 97 rows of a 112-row corpus for both range calibration and rotation optimization, with the remaining 15 rows held out for validation. This corpus does not overlap with the evaluation prompts at the prompt or normalized-text-hash level.

A.3 FLASHLight Optimization Settings

Table 4 summarizes QSD rotation optimization and range calibration for each architecture. Settings for matched comparisons that vary only the objective and for calibration-order ablations are given in the corresponding tables and subsections of Appendix F.

A.4 Runtime Evaluation Protocol

Tables 2 and 24 report the mean of run-level throughput values for each method and batch size. Each run processes 604 requests: 524 first turns from the four datasets (MT-Bench 80, GSM8K 200, ShareGPT 80, and HumanEval 164) and 80 MT-Bench second turns constructed using the first-turn responses generated in that run. Each request generates at most 256 new tokens, with a maximum length of 2,048 tokens. Requests are served with continuous batching: completed slots are immediately filled with waiting requests, and each new request's prefill is executed separately. Each process builds the model once and measures all batch sizes in sequence.

For each run, throughput is the total number of output tokens divided by the sum of the wall-clock times for first-turn and second-turn processing. This measurement includes all prefill and refill time

Table 3: Model, quantization, data, and execution settings for the three architectures. Layer indices follow each implementation.

Item	DFlash	EAGLE-3	DSpark
Target model	Llama-3.1-8B-Instruct @0e9e39f	Llama-3.1-8B-Instruct @0e9e39f	Qwen3-4B
Draft checkpoint / decoding code	z-lab/LLaMA3.1-8B-Instruct-DFlash-UltraChat @d3af30d; DFlash @94e4abc + local FLASH-Light modifications	yuhuili/EAGLE3-LLaMA3.1-Instruct-8B @ada412b; official EAGLE code @cb7e084	deepseek-ai/dspark_qwen3_4b_block7 @3457dff; DeepSpec @005e03b
Proposal configuration	Block size 10 ($K = 9$)	Tree depth 5, top-k 10, verification budget 60	$K = 7$; Vanilla Markov head (rank 256) + confidence head; threshold 0, no pruning
Target feature layer indices	$\{1, 8, 15, 22, 29\}$ (5×4096 into W_c)	$\{2, 16, 29\}$ (3×4096 into W_c)	$\{1, 9, 17, 25, 33\}$ of 36 (5×2560 into W_c)
Target quantization	W4A4 SpinQuant, 224 linear operators; asymmetric per-token A4; online Hadamard at the down-projection input	W4A4 SpinQuant, 224 linear operators; asymmetric per-token A4; online Hadamard at the down-projection input	W4A4 SpinQuant, 252 linear operators; symmetric per-token A4
Initial draft basis $R_{D,0}$ (= Aligned PTQ)	SpinQuant-style draft rotation learned with a draft-local objective	Random Hadamard R_D; fixed Hadamard R_2/R_4	Adapt Rotation (msModelSlim (Huawei, 2026)), 820 calibration texts (BF16 target)
Draft quantizer (W4 / A4)	Symmetric per-output-channel W4 (MSE scale) / asymmetric per-token A4 (codes $0, \ldots, 15$); 46 sites	Symmetric per-output-channel W4 (MSE scale) / asymmetric per-token A4 (codes $0, \ldots, 15$); 8 sites	Symmetric per-output-channel W4 (MSE scale) / symmetric per-token A4 (codes $-8, \ldots, 7$); 46 sites
16-bit components in the AL evaluation	Embedding and LM head (shared target weights, bf16); KV cache	Embedding and LM head (fp16); RMSNorm gains; KV cache	bf16: embeddings, LM head, Markov/confidence heads, RMSNorm (gains folded), q/k-norm, RoPE, SDPA, SiLU, residuals, KV cache
Range calibration data	S: 96 prompts ($2^{\nu/4}$ lattice search); S re-check and c: 178 prompts (GSM8K-train, ShareGPT, MBPP)	112-row corpus (GSM8K-train 48, ShareGPT 64): α on 97 rows, c on 40 of them	96 prompts
Rotation training / validation data	192 / 88 prompts (GSM8K-train, ShareGPT, MBPP) with W4A4-target greedy continuations (max. 512 new tokens)	97 / 15 rows of the 112-row corpus	192 / 88 prompts (GSM8K-train, ShareGPT, MBPP) with W4A4-target greedy continuations (max. 512 new tokens)
Test (all architectures)	MT-Bench 80×2, GSM8K 200, ShareGPT 80, HumanEval 164 (604 turns); greedy ($T = 0$), max. 1024 new tokens		
AL evaluation backend	Triton W4A4 kernels (target and draft), static-KV/CUDA-graph runtime; bf16	PyTorch fake-quant W4A4 (target and draft); fp16	HF Transformers 4.57.3 + official DeepSpec loop, SDPA, exact INT4 GEMM (int32 acc., fp32 epilogue); bf16
Runtime backend (Tables 2, 24)	native INT4 custom backend	native INT4 custom backend	static-KV + CUDA-graph runtime, native INT4 custom backend (W4A4), FlashAttention 2.8.3 (Dao, 2024)

Table 4: QSD rotation-optimization and range-calibration settings for FLASHLight. Source scaling and clipping are selected at the initial draft basis and then fixed while draft rotations are optimized using the quantized verifier's predictions. R_2 is a per-layer rotation over the attention-head dimension, R_4 is the fixed online Hadamard transform at the down-projection input, and R_C is folded into the output side of W_c.

Item	DFlash	EAGLE-3	DSpark
Range during rotation learning	$S^\star, c^\star$ selected at $R_{D,0}$ (fixed)	$\alpha^\star, c^\star$ selected at $R_{D,0}$ (fixed)	$S^\star, c^\star$ selected at $R_{D,0}$ (fixed)
Final objective / trainable rotations	QSD vs. W4A4 target; R_D (4096^2) + $5\times R_2$ (128^2); $R_C{:=}R_D$	QSD on draft-chain/W4A4 verifier distributions (target softmax probabilities restricted to the draft vocabulary, without renormalization); R_D (4096^2), embedding $\mathrm{blkdiag}(R_D, R_D)$; fixed Hadamard R_2/R_4	QSD vs. W4A4 SpinQuant target; R_D (2560^2) + $5\times R_2$ (128^2); $R_C{:=}R_D$
Optimizer / learning rate / update steps	Adam, 2×10^{-3} (constant), 400 steps, grad-norm clip 1.0	Adam, 2×10^{-3} (constant), 400 steps, grad-norm clip 1.0	Adam, 2×10^{-3} (constant), 400 steps, grad-norm clip 1.0
Depth horizon / rotation parameterization	d=1..9 (block 10, slot 0 dropped), Cayley $R{=}R_{D,0}D$, $D{=}\mathrm{Cay}(A)$; $A{=}0$ ($D{=}I$) init	d=1..6 (greedy top-1 chain), Cayley $R{=}R_{D,0}D$ ($R_{D,0}$: random Hadamard), $D{=}\mathrm{Cay}(A)$; $A{=}0$ ($D{=}I$) init	d=1..7 (block 7), Cayley $R{=}R_{D,0}D$, $D{=}\mathrm{Cay}(A)$; $A{=}0$ ($D{=}I$) init; fp64 polar at export
Rollout / anchors / data split	Teacher-forced; 1 row×32 anchors/step; 192 train / 88 val prompts (×12 anchors); S on 96 prompts, S re-check and c on 178 prompts	Self-rollout of the current quantized draft (greedy chain, target tree logits); 1 row×6 anchors/step; 97 train / 15 val rows (×3 anchors); α on 97 rows, c on 40 rows	Teacher-forced (incl. Markov input); 1 row×32 anchors/step; 192 train / 88 val prompts (×12 anchors); S, c on 96 prompts
Checkpoint / seeds	Final step 400 (val. loss logged every 100 steps, not used for selection); seeds 0, 1, 2	Final step 400 (val. loss logged every 100 steps, not used for selection); seeds 0, 1, 2	Final step 400 (val. loss logged every 25 steps, not used for selection); seeds 0, 1, 2
Range objective / search budget	$\mathcal{J}_{\mathrm{proj}}$ of W_c at $R_{D,0}$ (pos$\geq$2); S: $2^{\nu/4}$ coordinate descent from $S{=}I$, 2 passes (81 evals, 36,521 rows), re-checked over 160 fixed S candidates (fp64 ref., 68,990 rows); 10-pt c grid (1.00–0.60)	At $R_{D,0}$: α by RMS-normalized W_c-output NMSE (Adam 5×10^{-3}, 300 steps, $\prod\alpha{=}1$); c by final hidden-state NMSE vs. 16-bit draft, 11-pt grid (0.50–1.00)	$\mathcal{J}_{\mathrm{proj}}$ of W_c at $R_{D,0}$ (39,754 rows, pos-0 excluded); $2^{\nu/4}$ coordinate descent, 2 passes (81 evals) + 10-pt c grid (1.00–0.60)
Final scales / clipping / site list	$S^\star{=}(4, 4, 2.83, 1.41, 0.84)$; $c{=}0.70$ on W_c input only (other 45 sites $c{=}1$)	$\alpha^\star{=}(1.52, 1.12, 0.59)$; $c{=}0.70$ on all 8 A4 sites	$S^\star{=}(9.51, 6.73, 3.36, 2.83, 1.00)$; $c{=}0.60$ on W_c input only (other 45 sites $c{=}1$)

but excludes model building, CUDA graph capture, and the 8 warm-up requests before each batch-size measurement. We average these run-level throughputs for each method and batch size. DFlash and EAGLE-3 use 3 repeats for every configuration. DSpark uses 8 repeats for the 16-bit draft and W4A16 GPTQ, 6 for Aligned PTQ, and 3 for FLASHLight. The standard deviation across repeats for each table entry is reported in Appendix H.6. All measurements are performed on a shared host with concurrent jobs on other GPUs.

Latency, TPOT, and peak memory are recorded in the same runs (Appendix H.5). These runs do not record a per-cycle breakdown of drafting, verification, and other costs. When evaluating Equation 51 from these runs, AL and cycle time are taken from the same run. Backend details and interpretation of the runtime results are provided in Appendix H.

B Full-Precision Equivalence and Fusion

This section derives the full-precision equivalence of the basis transformations in Section 3.1, accounting for RMSNorm affine scales and weight fusion along each feature path. We then describe the architecture-specific fusion paths and report numerical checks.

B.1 Why Unaligned PTQ Fails

Unaligned PTQ applies the original projection W_c directly to the target-provided feature $\widetilde{\mathbf{H}} = \mathbf{H}\overline{R}_T$. The resulting error relative to the original projection output is

$$\widetilde{\mathbf{H}}W_c - \mathbf{H}W_c = \mathbf{H}(\overline{R}_T - I)W_c \tag{8}$$

which is generally nonzero even in full precision. Unaligned PTQ therefore does not isolate rounding error. When target features and recurrent draft features use different bases, each path must incorporate its corresponding transformation. The derivations below apply to Aligned PTQ, which satisfies this condition.

B.2 RMSNorm Affine-Scale Reparameterization

Let $\mathbf{x}$ be the d-dimensional row-vector output of the projection. We write RMSNorm (Zhang & Sennrich, 2019) as

$$\begin{aligned} \mathrm{RMSNorm}(\mathbf{x}) &= \mathcal{N}(\mathbf{x})\Gamma, \qquad \Gamma = \mathrm{diag}(\boldsymbol{\gamma}_{\mathrm{rms}}), \\ \mathcal{N}(\mathbf{x}) &= \frac{\mathbf{x}}{\sqrt{\|\mathbf{x}\|_2^2/d + \epsilon_{\mathrm{rms}}}}. \end{aligned} \tag{9}$$

Because $\Gamma R_D \neq R_D\Gamma$ in general, we handle the affine scale separately. Let W_{next} be the downstream weight applied to the rotated feature $\mathcal{N}(\mathbf{x})\Gamma R_D$. The following reparameterization preserves the downstream output:

$$\begin{aligned} W_{\mathrm{next}}^{\Gamma} &= R_D^{\top}\Gamma R_D W_{\mathrm{next}}, \\ \big(\mathcal{N}(\mathbf{x})\Gamma R_D\big)W_{\mathrm{next}} &= \big(\mathcal{N}(\mathbf{x})R_D\big)W_{\mathrm{next}}^{\Gamma}. \end{aligned} \tag{10}$$

Let W_{next}^0 denote the original weight applied to the input in its original basis. Since $W_{\mathrm{next}} = R_D^{\top}W_{\mathrm{next}}^0$, Equation 10 can also be written as $W_{\mathrm{next}}^{\Gamma} = R_D^{\top}\Gamma W_{\mathrm{next}}^0$. Thus, the affine scale is absorbed into the downstream weight.

In DFlash, the hidden RMSNorm affine scale is absorbed into the context-side weights of the K/V projections. When an original weight is shared with the draft-token path, we construct the equivalent weight required by each path. The interface sources, fusion locations, and corresponding weight blocks for EAGLE-3 are specified in Appendix B.6.

B.3 Exchanging RMS Normalization and Rotation

After the affine scale is absorbed into downstream weights, orthogonality gives $\|\mathbf{x}R_D\|_2 = \|\mathbf{x}\|_2$, so

$$\begin{aligned} \underbrace{\mathcal{N}(\mathbf{x})R_D}_{\text{RMSNorm}\to R_D} &= \frac{\mathbf{x}R_D}{\sqrt{\|\mathbf{x}\|_2^2/d + \epsilon_{\text{rms}}}} \\ &= \frac{\mathbf{x}R_D}{\sqrt{\|\mathbf{x}R_D\|_2^2/d + \epsilon_{\text{rms}}}} \\ &= \underbrace{\mathcal{N}(\mathbf{x}R_D)}_{R_D\to\text{RMSNorm}} . \end{aligned} \tag{11}$$

Thus, R_D can be moved before the affine-free normalization and fused into the preceding projection as $W_c R_D$.

B.4 Fusing Independent Draft Rotation into Existing Projections

Consider the *reparameterized full-precision graph* with affine scales absorbed into downstream weights. Let $\mathbf{Z}_{\text{run}}$ and $\mathbf{Z}_{\text{fuse}}$ denote the interface outputs of the explicit runtime-rotation and fused paths, respectively. The concatenated input dimension is $d_{\text{in}} = \sum_i d_i$, and the draft residual dimension is d_D. The block-diagonal matrix of per-source target rotations is $\overline{R}_T \in \mathbb{R}^{d_{\text{in}}\times d_{\text{in}}}$, with $W_c \in \mathbb{R}^{d_{\text{in}}\times d_D}$ and $R_D \in \mathbb{R}^{d_D\times d_D}$. Writing the weight in Equation 3 as $W_c^{\text{base}} = \overline{R}_T^\top W_c R_D$, we obtain

$$\begin{aligned} \mathbf{Z}_{\text{run}} &= \mathcal{N}\Big((\widetilde{\mathbf{H}}\overline{R}_T^\top)W_c\Big)\, R_D \\ &= \mathcal{N}\Big((\widetilde{\mathbf{H}}\overline{R}_T^\top)W_c R_D\Big) \\ &= \mathcal{N}\Big(\widetilde{\mathbf{H}}W_c^{\text{base}}\Big) = \mathbf{Z}_{\text{fuse}}. \end{aligned} \tag{12}$$

Combined with Equation 10, this fusion also preserves the original downstream output with the affine scale included. Source scaling is incorporated through cancellation between S applied to activations and S^{-1} applied to weights. These transformations fuse the input and output basis changes into the existing projection; RMS normalization, activation quantization, and architecture-specific online Hadamard transforms remain in the execution path. The equivalence holds before clipping and quantization. After quantization, in general,

$$Q_A(xR)Q_W(R^\top W) \neq Q_A(x)Q_W(W). \tag{13}$$

The derivation establishes the equivalence of the reparameterized full-precision computations, not identical low-bit predictions.

B.5 Basis Consistency under SpinQuant

The basis mismatch in Equation 8 changes the projection output even before quantization. All W4A4 draft configurations in this paper therefore start from Aligned PTQ, which incorporates the target rotation into the feature projection weight. Table 17 in Appendix F.1 reports ablations of source scaling, clipping, and rotation optimization for the remaining acceptance gap. These evaluations use the same prompts as Tables 10 and 23.

B.6 Architecture-Specific Fusion: EAGLE-3

This subsection and the next describe fusion along the feature paths and prediction heads of EAGLE-3 and DSpark. Table 3 lists the checkpoints, quantizers, data, and execution settings for all three architectures.

EAGLE-3 concatenates hidden features from three target layers and projects them to the draft hidden dimension. The projected target feature and the token embedding are separately normalized with RMSNorm, then concatenated and passed directly to the Q/K/V projections. Let W_f denote the target feature projection, corresponding to W_c in the main text:

$$\begin{aligned} \mathbf{H}_E &= [\mathbf{h}_{\ell_1}, \mathbf{h}_{\ell_2}, \mathbf{h}_{\ell_3}], \\ \mathbf{f}_T &= \mathbf{H}_E W_f, \\ \mathbf{u} &= [\mathrm{RMSNorm}_e(\mathbf{e}), \mathrm{RMSNorm}_h(\mathbf{f})]. \end{aligned} \tag{14}$$

The first forward pass uses the target feature, $\mathbf{f} = \mathbf{f}_T$; recurrent passes instead use the hidden feature from the preceding draft forward pass. The projection W_f that combines the three target features is distinct from the Q/K/V projections that consume the concatenated embedding and hidden feature. Our configuration extracts target features at h_2, h_{16}, h_{29}.

Let $\overline{R}_T = \mathrm{blkdiag}(R_T, R_T, R_T)$ denote the block-diagonal rotation of the concatenated target features, and let $S_E = \mathrm{blkdiag}(\alpha_1 I, \alpha_2 I, \alpha_3 I)$ denote their block-diagonal source scaling. Writing $\widetilde{\mathbf{H}}_E = \mathbf{H}_E \overline{R}_T$, we apply

$$\begin{aligned} W_f^{\mathrm{FL}} &= S_E^{-1} \overline{R}_T^{\top} W_f R_D, \\ (\widetilde{\mathbf{H}}_E S_E) W_f^{\mathrm{FL}} &= \mathbf{f}_T R_D \end{aligned} \tag{15}$$

to the target feature projection. The source scales act on this projection's three feature blocks. The QSD configuration in Table 4 uses $\alpha^\star = (1.5162, 1.1204, 0.5887)$ ($\prod_j \alpha_j^\star = 1$), selected at the initial Hadamard basis $R_{D,0}$. A shared clipping ratio $c^\star = 0.70$ is applied to the 8 A4 sites: the feature projection, Q/K/V, O, and gate/up/down. At the same basis, this grid value minimizes final hidden-state NMSE relative to the 16-bit draft. The source scales and clipping ratio are fixed before rotation optimization and are not reselected afterward.

The embedding and recurrent hidden feature share the draft residual rotation R_D. Let $\mathbf{u}_0 = [\mathcal{N}_e(\mathbf{e}), \mathcal{N}_h(\mathbf{f})]$ denote their concatenation without affine scales, and define $B_D = \mathrm{blkdiag}(R_D, R_D)$ and $\Gamma_{eh} = \mathrm{blkdiag}(\Gamma_e, \Gamma_h)$. The Q/K/V *input-basis transformation* is

$$\begin{aligned} \widetilde{\mathbf{u}} &= \mathbf{u}_0 B_D, \\ W_j^{\text{in-fuse}} &= B_D^{\top} \Gamma_{eh} W_j, \qquad j \in \{Q, K, V\}, \\ \widetilde{\mathbf{u}} W_j^{\text{in-fuse}} &= \mathbf{u}_0 \Gamma_{eh} W_j \end{aligned} \tag{16}$$

and is combined with R_2 on V/O and a fixed Hadamard transform at the MLP down-projection input. The learned residual rotation R_D is incorporated consistently into the target feature projection, embedding, first and recurrent paths, and draft-side weights.

B.7 Architecture-Specific Fusion: DSpark

Architecture and prediction heads. Our DSpark configuration uses Qwen3-4B as the target, with W4A4 quantization and SpinQuant rotations (Liu et al., 2025b). The draft basis is initialized with Adapt Rotation (msModelSlim; Huawei, 2026). DSpark combines a parallel backbone that consumes target hidden features with a lightweight sequential head (Cheng et al., 2026). With the Markov head, the draft distribution is

$$q_k(v \mid x_{k-1}, H) = \mathrm{softmax}_v(U_k + B(x_{k-1}, \cdot)), \qquad B(x_{k-1}, \cdot) = W_1[x_{k-1}] W_2, \tag{17}$$

where U_k denotes the backbone's base logits, H the target hidden feature, W_1 and W_2 the two factors of the rank-256 Markov head, and x_0 the target-provided anchor. The confidence head estimates an acceptance score from the backbone hidden state and the previous token's Markov embedding (Cheng et al., 2026). This score is distinct from both the QSD distribution overlap $a_{i,d}$ and the empirical conditional acceptance A_k in Table 13. Our main comparison uses fixed-length proposals with threshold 0 and no confidence-based pruning.

Consistent rotation across draft prediction paths. We apply the input and output basis transformations from Section 3.1 to the target feature projection. Any additional consumer of the

hidden state must account for the same basis change. For example, if the confidence score is $s_k = \sigma(h_k w_h + e_{k-1} w_e)$ and the backbone hidden state is represented as $h'_k = h_k R_D$, then $w'_h = R_D^\top w_h$ preserves the original score. The hidden-space rotation is not applied directly to the vocabulary-space transition bias. Numerical checks of the target-rotation fold, interface feature projection, and context K/V fusion for DSpark are reported in Appendix B.8.

B.8 Numerical Fusion Audit

Folding the DSpark target rotation in fp32 yields a logit relative error of 4.7×10^{-6}, top-1 agreement of 1.000, and matching greedy generations in 8/8 tested cases. For the interface feature projection and context K/V, fp64 comparisons that include the initial draft basis $R_{D,0}$ yield a relative error of at most 3.7×10^{-14}. These checks assess numerical parity between explicit transformations and fused weights.

For DSpark, rotations, the reciprocal scale $1/S$, and norm gains are folded offline, and fusion reduces the draft GEMM count from 46 to 26. The existing online O/down Hadamard transforms are integrated into the A4 quantization kernel.

For DFlash and EAGLE-3, we fold the seed-0, step-400 rotations learned with range-first QSD and audit them on CPU. DFlash stores R_D and its five R_2 matrices in fp32. The orthogonality error $\max|R^\top R - I|$ is 1.1×10^{-6} for R_D and at most 1.8×10^{-6} for the R_2 matrices. Folding $R_C{=}R_D$ into the output side of W_c agrees with explicitly applying the rotation after the projection, with an fp64 relative error of 1.0×10^{-15}. Using the deployment fp32 fold yields an output difference of 2.2×10^{-7}. Deviations from the original computation due to folding are limited by the rotation's orthogonality error; across the 45 folds in W_c, the context K/V views, and the draft layers, the fp64 relative error is at most 1.5×10^{-6}.

For EAGLE-3, R_D has $\max|R^\top R - I| = 1.4 \times 10^{-6}$. The feature-projection fold has an fp64 identity error of 8.9×10^{-16} and an output error of 2.3×10^{-4} under the deployment order of fp64 folding followed by fp16 storage. Across the 11 folds in Q/K/V (embedding and hidden halves), O, gate/up/down, and the LM head, the fp16 storage error is at most 2.1×10^{-4}, and the fp64 difference from the original computation is at most 1.6×10^{-6}.

These checks validate the numerical consistency of the reported full-precision transformations and their offline weight fusion.

For both architectures, rotations are folded offline, while activation-side source scaling and clipping are handled within the existing A4 quantization kernel. No kernel launches are added relative to Aligned PTQ: for speculative cycles with the same workload, the Aligned PTQ/FLASHLight counts are 632/632 for DFlash and 1005/1005 for EAGLE-3. These counts were measured on an earlier checkpoint. Rotation and range values are folded into weights or passed as kernel arguments, so launch counts are determined by the execution graph; the reported counts also apply to the range-first checkpoint. The DFlash draft's 46 W4A4 linear operators execute as 26 GEMM launches per draft step. Unlike in DSpark, the online R_4 Hadamard transform remains in separate launches, with the same launch count as in Aligned PTQ.

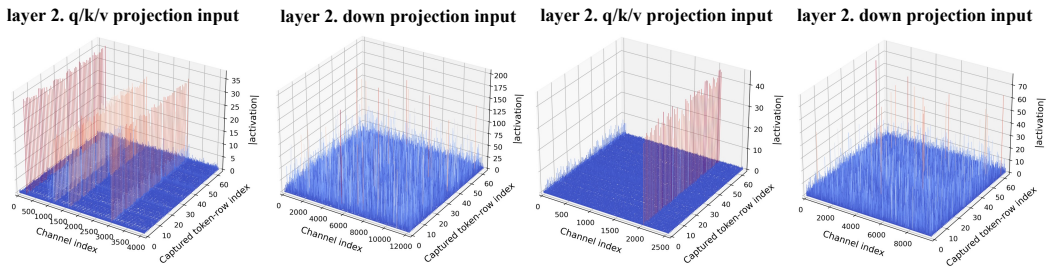
C Interface Range Calibration

This section first examines draft activation outliers before rotation, motivating draft rotation while highlighting the effects of large-norm tokens that rotation alone may not resolve. We then describe the shared A4 grid of the concatenated interface and the calibration procedures, selected values, and diagnostics for source scaling and clipping. FLASHLight selects these range parameters at the initial draft basis and fixes them before rotation optimization.

C.1 Activation Outliers in the Quantized Draft

Like target LLMs, drafts exhibit channel-wise outliers and very large activations (massive activations) (Figure 6). These values can widen the quantization range and reduce the resolution available for the remaining values. Rotation mitigates this effect by redistributing activations concentrated in

individual channels (Ashkboos et al., 2024; Liu et al., 2025b). However, rotation preserves token norms, so the effects of large-norm tokens may persist even after channel concentration is reduced.



(a) DFlash Draft (Llama-3.1-8B-Instruct) (b) DSpark Draft (Qwen3-4B)

Figure 6: Draft activation distributions before rotation. (a) Q/K/V- and down-projection inputs in layer 2 of the DFlash draft (Llama-3.1-8B-Instruct). (b) The corresponding inputs in layer 2 of the DSpark draft (Qwen3-4B). Both drafts exhibit channel-wise outliers and massive activations, highlighting the need to address activation distributions for low-bit quantization.

C.2 Shared A4 Grid of the Concatenated Interface

Features from different target layers have heterogeneous ranges, but per-token A4 quantization assigns the concatenated sources a shared scale and, for asymmetric quantization, a shared zero point. For a token t with quantization range $[\ell_t, u_t]$, asymmetric A4 has grid spacing

$$\Delta_t = \frac{u_t - \ell_t}{2^4 - 1} \tag{18}$$

whereas DSpark's symmetric A4 uses $\Delta_t = \max_j |x_{t,j}|/(2^3-1)$, with $x_{t,j}$ denoting the elements of token t. In both cases, all sources for a token share the same grid. When a large-range source or a few extreme values determine the overall range, smaller-range sources occupy only a few quantization levels (Figure 4).

C.3 Source Scaling: Calibration and Selected Scales

We select one positive scalar α_i per source, shared across all tokens. Table 5 lists the selected $S^\star$ and the reconstruction objective values recorded for each architecture. The scales are selected at the initial draft basis $R_{D,0}$ with $c = 1$ and fixed during rotation optimization and deployment. For DFlash and DSpark, $S^\star$ lies on the $2^{\nu/4}$ lattice, with $\nu = (8, 8, 6, 2, -1)$ and $(13, 11, 7, 6, 0)$, respectively. On the DSpark range-calibration data, per-source RMS values at the feature projection input span 0.25–9.97, indicating substantial differences across sources.

Table 5: Source scales selected by FLASHLight at the initial basis $R_{D,0}$ and the corresponding reconstruction objectives. For DFlash and DSpark, $\mathcal{J}_{\text{proj}}$ in Equation 19 is evaluated on calibration rows excluding attention-sink positions: positions 0 and 1 for DFlash and position 0 for DSpark (68,990 rows from 178 prompts and 39,754 rows from 96 prompts, respectively). EAGLE-3 reports feature-projection output NMSE after per-token RMS normalization and application of the hidden-norm gain, evaluated on 15 validation rows (1,882 tokens). The last column holds $S^\star$ fixed and applies the deployment clipping ratio $c^\star$; it is not reported for EAGLE-3, whose $c^\star$ is selected using final hidden-state NMSE.

Architecture	Scaling factors $S^\star$ ($\alpha^\star$)	$S = I$ ($\alpha = 1$)	$S^\star$ ($\alpha^\star$)	$S^\star, c^\star$
DFlash	(4.0000, 4.0000, 2.8284, 1.4142, 0.8409)	0.0524	0.0162	0.0122
EAGLE-3	(1.5162, 1.1204, 0.5887)	0.0927	0.0388	–
DSpark	(9.5137, 6.7272, 3.3636, 2.8284, 1.0000)	0.1803	0.0554	0.0365

Let X denote the calibration activation matrix and $W_{b,0} = \overline{R}_T^\top W_c R_{D,0}$ the projection weight at the initial basis. The scaling objective is

$$\mathcal{J}_{\mathrm{proj}}(S) = \frac{\left\| Q_A^{(1)}(XS) Q_W(S^{-1} W_{b,0}) - X W_{b,0} \right\|_F^2}{\|X W_{b,0}\|_F^2 + \epsilon_{\mathrm{cal}}}. \tag{19}$$

The reference $XW_{b,0}$ is independent of the candidate, and the objective jointly captures activation and weight quantization errors. For candidates evaluated on the same tensor, normalization by the fixed reference norm leaves the minimizer unchanged relative to tensor-level MSE. DSpark searches this objective using coordinate descent (2 passes) on the $2^{\nu/4}$ lattice, starting from $S = I$. For DFlash, we re-evaluate 160 fixed S candidates on 178 prompts. These candidates include the result of the same coordinate-descent procedure on a separate 96-prompt pool and half-octave lattice points. The deployment choice $S^\star$ has the lowest evaluated objective in both searches. EAGLE-3 uses Adam to minimize feature-projection output NMSE after per-token RMS normalization and application of the hidden-norm gain, subject to $\prod_i \alpha_i = 1$.

Inputs for distribution analysis and calibration are collected at the corresponding fusion and quantization sites. For EAGLE-3, α is calibrated using up to 128 tokens from each of 97 rows in the 112-row corpus (12,330 tokens) and validated on the remaining 15 rows (1,882 tokens). The distribution figures use all 112 rows.

C.4 Source Scaling versus Per-Channel Calibration

At DFlash's initial draft basis $R_{D,0}$, we compare 5-block source scaling with SmoothQuant-style per-channel scaling (Xiao et al., 2023). For feature-projection input channel j, the per-channel scale is $s_j = \max_t |X_{tj}|^\lambda / \max_o |(W_{b,0})_{jo}|^{1-\lambda}$, where t ranges over tokens at position 2 or later and λ denotes the SmoothQuant migration strength. We multiply activation channel j by $1/s_j$ and row j of $W_{b,0}$ by s_j. Table 6 reports $\mathcal{J}_{\mathrm{proj}}$ at $c = 1$ on the 96-prompt calibration pool, which is disjoint from the 178-prompt range-search pool.

Table 6: $\mathcal{J}_{\mathrm{proj}}$ by scaling parameterization at the initial draft basis $R_{D,0}$ with $c = 1$. "Sink excluded" uses 36,521 tokens after excluding positions 0 and 1; "All tokens" uses all 36,713 tokens. The SmoothQuant-style rows do not benchmark the official SmoothQuant implementation.

Scaling configuration	Scalars	Sink excluded	All tokens
None ($S = I$)	–	0.0521	0.0330
SmoothQuant-style per-channel, $\lambda = 0.50$	20,480	0.0160	0.1092
SmoothQuant-style per-channel, $\lambda = 0.75$	20,480	0.0154	0.1794
SmoothQuant-style per-channel, $\lambda = 0.85$	20,480	0.0158	0.2333
Source scaling (5-block, $S^\star$)	5	0.0162	0.0454

On the sink-excluded selection population, per-channel scaling achieves an objective 1.1–4.9% lower than that of $S^\star$; source scaling therefore does not outperform it on this objective. Both reduce the objective by 69–70% relative to $S = I$, with the 5-block parameterization achieving this reduction using only 5 scalars. When attention-sink tokens are included, however, the per-channel objective is 2.4–5.1 times that of $S^\star$; both procedures select scales using sink-excluded tokens. Because unconstrained per-channel scaling can represent block scaling, this comparison concerns the transformations selected by the respective procedures, not the superiority of either parameter space. We do not measure acceptance for per-channel scaling and therefore draw no conclusion about the end-to-end superiority of either parameterization or the effect of verifier-guided rotation from this comparison.

C.5 Before and After Scaling

To assess source-wise range allocation, we fix the initial draft basis $R_{D,0}$ and compare $S = I$ with the deployment choice $S^\star$. Both conditions use the same activation samples and start from the same projection weight, without clipping ($c = 1$). Figures 7 (DFlash) and 8 (EAGLE-3) show both the activation distributions and the weight distributions after reciprocal scaling.

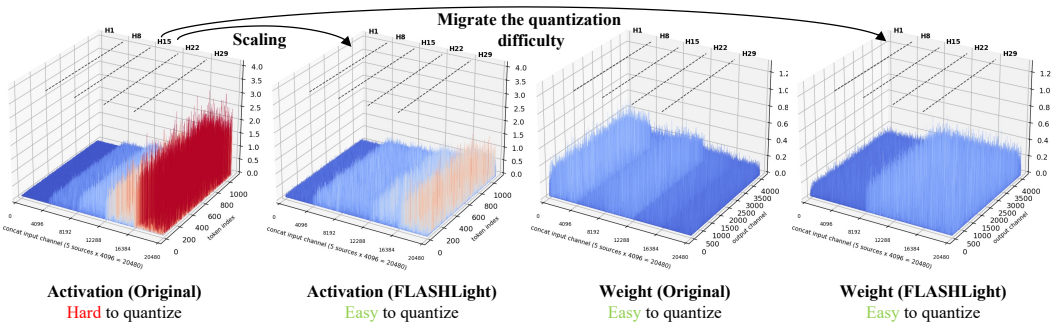


Figure 7: Activation and weight magnitudes at the DFlash feature projection (Llama-3.1-8B-Instruct), before ($S = I$) and after ($S^\star$) source scaling at the initial draft basis $R_{D,0}$. Activations are from positions 2 and later in the 96-prompt calibration pool; weights are the full-precision inputs to the W4 quantizer. Source scaling preserves the full-precision projection output while adjusting range allocation on the low-bit grid shared by heterogeneous sources.

On the data used in these figures, with attention-sink positions (positions 0 and 1) excluded, source scaling reduces the W4A4-versus-FP projection NMSE from 0.0521 to 0.0162 for DFlash (36,521 tokens) and from 0.0881 to 0.0365 for EAGLE-3 (13,988 tokens). Including sink positions reverses this trend: the all-token NMSE increases from 0.0330 to 0.0454 for DFlash and slightly from 0.0109 to 0.0111 for EAGLE-3. These all-token values are dominated by sink tokens, which account for less than 2% of tokens and incur greater error after source scaling. DFlash and DSpark exclude sink positions from the range-search objective, so projection errors on sink tokens do not influence the selection of $S^\star$.

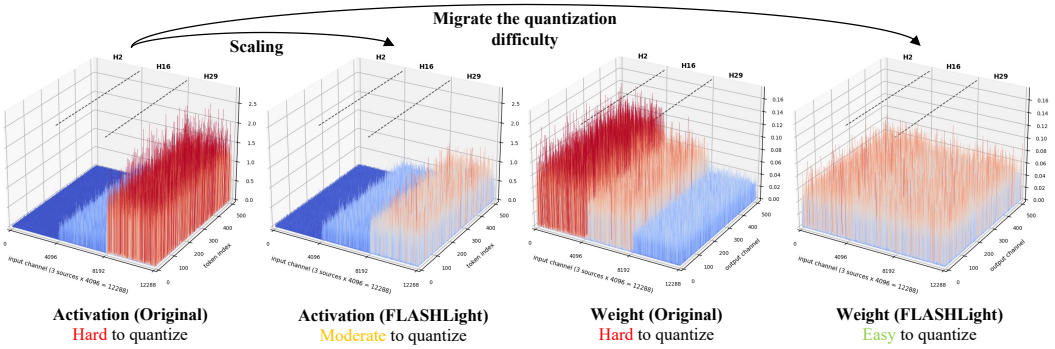


Figure 8: Distributions at the EAGLE-3 feature projection before ($\alpha = 1$) and after ($\alpha^\star$) source scaling at the initial Hadamard draft basis. Activation magnitudes for the three feature blocks are shown alongside weights after reciprocal scaling. Because the block-wise magnitude ordering is reversed between activations and weights, $\alpha^\star$ reduces the imbalance across blocks on both sides.

C.6 Clipping: Per-Token Activation Quantization

Let $\mathbf{x} = (x_1, \ldots, x_N)$ denote an activation row and $c \in (0, 1]$ the clipping ratio. We set the token's lower and upper bounds to

$$\ell = c \min\Big(\min_i x_i, 0\Big), \qquad u = c \max\Big(\max_i x_i, 0\Big) \tag{20}$$

The choice $c = 1$ retains the original range, whereas $c < 1$ moves both bounds toward the origin. A single clipping ratio does not impose $\ell = -u$; the quantizer remains asymmetric.

DFlash and EAGLE-3 use 16 unsigned 4-bit codes, $0, \ldots, 15$. For a row with $u > \ell$, quantization and reconstruction are computed as

$$\Delta = \frac{u-\ell}{15}, \qquad z = \mathrm{round}\left(-\frac{\ell}{\Delta}\right), \\ q_i = \mathrm{clip}\left(\mathrm{round}\left(\frac{x_i}{\Delta}\right) + z, 0, 15\right), \quad \widehat{x}_i = \Delta(q_i - z) \tag{21}$$

All channels in a row share Δ, z. For an all-zero row, the implementation's zero-range guard returns reconstructed values of 0. Rounding the zero point gives actual reconstruction endpoints of $-\Delta z$ and $\Delta(15-z)$. Thus, for DFlash and EAGLE-3, $Q_A^{(c)}$ in the main text includes the integer saturation and dequantization in Equation 21.

DSpark uses symmetric per-token quantization for both its W4A4 target and draft. With row scale $s = c\max_i |x_i|/7$, it computes $q_i = \mathrm{clip}(\mathrm{round}(x_i/s), -8, 7)$ and $\widehat{x}_i = sq_i$ using 16 signed codes. Here too, the clipping ratio scales the range by c.

C.7 Clipping Calibration and Deployment Scope

Table 7 lists the clipping settings selected at the initial draft basis and fixed for rotation optimization and deployment. The subsequent diagnostics use the same settings, and the remaining QSD training settings follow Table 4. For each architecture, the target clipping settings remain unchanged whether or not draft clipping is applied. Projections sharing an activation, such as Q/K/V, reuse the same quantized activation.

Table 7: Draft clipping settings for FLASHLight. Sites are counted by logical linear-operator inputs.

Architecture	$c^\star$	Clipped draft inputs
DFlash	0.70	Feature projection input only (the other 45 draft sites use $c = 1$)
EAGLE-3	0.70	8 sites: feature projection, Q/K/V, O, and gate/up/down
DSpark	0.60	Feature projection input only (the other 45 draft sites use $c = 1$)

With $S^\star$ fixed, DFlash and DSpark select the feature-projection input ratio c that minimizes $\mathcal{J}_{\mathrm{proj}}$ over a 10-point grid (1.00–0.60). For DFlash, $\mathcal{J}_{\mathrm{proj}}$ decreases from 0.0162 at $c = 1$ to 0.0122 at $c = 0.70$, then increases to 0.0124 at $c = 0.65$. For DSpark, the best tested value, $c = 0.60$ (0.0365), is the grid's lower bound; smaller values of c were not explored.

EAGLE-3 selects the clipping ratio shared across its 8 sites using the full draft's final hidden-state NMSE relative to the 16-bit draft. On the 11-point grid (0.50–1.00), the NMSE values at $c = 0.65, 0.70, 0.75$ are $0.22183, 0.20368, 0.20616$ on the calibration rows and $0.21804, 0.20008, 0.20239$ on the validation rows. Both sets attain their lowest NMSE at $c = 0.70$, which is used during range-first rotation optimization and deployment. Clipping is selected by reconstruction error, not by maximizing the code entropy $H(Q)$ or efficiency η defined in Appendix C.8.

C.8 Code Occupancy, Entropy, and Efficiency

We compute the code histogram by quantizing each token with its own Δ, z (scale s for DSpark), then pooling the codes of the analyzed activation elements. For DSpark, signed codes are reindexed as $k = q+8$, mapping $q = -8, \ldots, 7$ to $k = 0, \ldots, 15$. Let N_{all} denote the total number of elements and n_k the count for code k. Then

$$p_k = \frac{n_k}{N_{\mathrm{all}}} = \frac{1}{N_{\mathrm{all}}} \sum_i \mathbf{1}[q_i = k], \qquad \sum_{k=0}^{15} p_k = 1. \tag{22}$$

Each bar in Figure 9 represents p_k.

The Shannon entropy (Shannon, 1948) of the code-occupancy distribution and its normalized efficiency are

$$H(Q) = -\sum_{k=0}^{15} p_k \log_2 p_k, \qquad 0 \le H(Q) \le 4,$$
$$\eta = \frac{H(Q)}{\log_2 16} = \frac{H(Q)}{4}, \quad \eta(\%) = 100\frac{H(Q)}{4} \tag{23}$$

We use $0 \log_2 0 = 0$. If all activations map to one code, $H = 0$; uniform use of all 16 codes gives $H = 4$ and $\eta = 1$. For a general b-bit quantizer, $\eta = H/b$.

The *effective number of codes*, corresponding to a uniform distribution with the same entropy, is

$$N_{\text{eff}} = 2^{H(Q)} \tag{24}$$

This differs from the number of nonempty codes. For the uniform distribution $U_k = 1/16$, we also have

$$D_{\text{KL}}^{(2)}(p\|U) = 4 - H(Q), \qquad \eta = 1 - \frac{D_{\text{KL}}^{(2)}(p\|U)}{4}, \tag{25}$$

where $D_{\text{KL}}^{(2)}$ uses base-2 logarithms. A higher η indicates more uniform code usage. Because quantization parameters vary across rows (Δ, z for the asymmetric quantizer), pooled entropy is not interpreted as the amount of preserved feature information.

C.9 Observed Code-Utilization Changes

Table 8 reports the effect of changing only the clipping ratio from $c = 1$ to its deployment value $c^\star$. The initial draft basis $R_{D,0}$, deployment scales $S^\star$, input samples, and W4 weights are fixed. Figure 9 shows the corresponding proportions of the 16 codes. For each architecture, we collect 8,192 feature-projection input rows (2,048 per dataset) from the first 40 prompts of each of the four evaluation datasets, pooling prompt-ingestion and decode events. These captures are used only for diagnostics, not for range calibration. Attention-sink rows are excluded: positions 0 and 1 of each prompt-turn for DFlash, the first row of each prompt-turn for EAGLE-3, and position 0 for DSpark (74, 35, and 36 rows, respectively). Sites and ratios are specified in Appendix C.7.

Table 8: Code utilization before and after clipping on identical activation samples. Arrows indicate changes from $c = 1$ to the listed deployment ratio $c^\star$; attention-sink rows are excluded.

Projection	$c^\star$	$H(Q)$ (bits)	η (%)	N_{eff}
DFlash interface	0.70	$2.8009 \to 3.2511$	$70.0 \to 81.3$	$6.97 \to 9.52$
EAGLE-3 interface	0.70	$2.5437 \to 2.9492$	$63.6 \to 73.7$	$5.83 \to 7.72$
DSpark interface	0.60	$2.3310 \to 3.0202$	$58.3 \to 75.5$	$5.03 \to 8.11$

Normalized efficiency increases by 11.3, 10.1, and 17.2 percentage points for DFlash, EAGLE-3, and DSpark, respectively. Entropy increases by approximately 0.41–0.45 bits for DFlash and EAGLE-3 and by 0.69 bits for DSpark. Only 0.42%, 0.48%, and 0.89% of activations are clipped in DFlash, EAGLE-3, and DSpark, respectively, all below 1%. These observations are consistent with allowing saturation in a small fraction of the tails to improve resolution in the central region.

C.10 Reconstruction Quality under Controlled Clipping

The clipping rate is $N_{\text{all}}^{-1} \sum_i \mathbf{1}[x_i < \ell_i \text{ or } x_i > u_i]$, whereas the boundary-code rate is $p_0 + p_{15}$ (the combined proportion of codes -8 and 7 for DSpark). Both are distinct from entropy. Stronger clipping can improve code utilization while increasing tail saturation error, so reconstruction error and AL are evaluated separately.

To isolate clipping's effect on the projection output, we fix the deployment scales $S^\star$ and initial draft basis $R_{D,0}$. Within this subsection, let $X = \widetilde{\mathbf{H}} S^\star$ denote the scaled input and $W_c^{\text{range}} =$

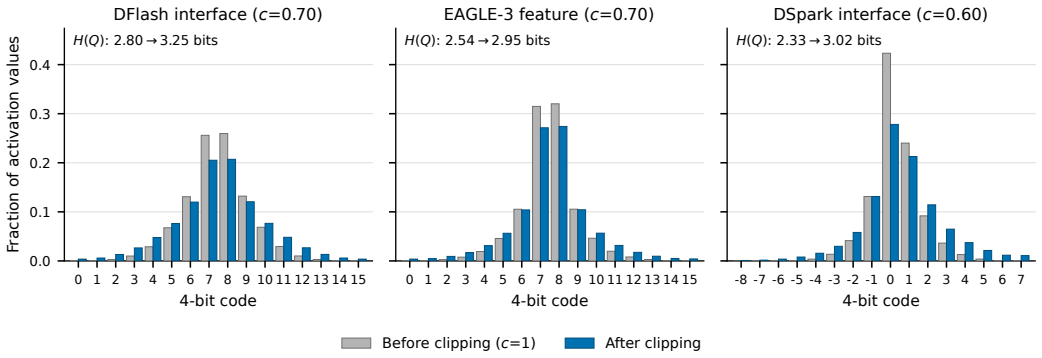


Figure 9: Pooled code occupancy under controlled clipping. Each panel compares the 16-code proportions obtained from identical activation samples with $c = 1$ (gray) and the deployment ratio $c^\star$ (blue), at the initial draft basis and deployment source scales ($S^\star$; $\alpha^\star$ for EAGLE-3). Attention-sink rows are excluded. DFlash and EAGLE-3 show asymmetric per-token codes $0, \ldots, 15$; DSpark shows symmetric per-token codes $-8, \ldots, 7$.

$(S^\star)^{-1}\overline{R}_T^\top W_c R_{D,0}$ the corresponding weight. Holding $\overline{W} = Q_W(W_c^{\text{range}})$ fixed in both conditions, we compute

$$\begin{aligned} \text{NMSE}_A(c) &= \frac{\|Q_A^{(c)}(X) - X\|_F^2}{\|X\|_F^2 + \epsilon_{\text{cal}}}, \\ \text{NMSE}_{\text{proj}|W4}(c) &= \frac{\|Q_A^{(c)}(X)\overline{W} - X\overline{W}\|_F^2}{\|X\overline{W}\|_F^2 + \epsilon_{\text{cal}}} \end{aligned} \tag{26}$$

The second metric uses full-precision activations passed through the same W4 weight as its reference, isolating the effect of activation quantization with the quantized weight fixed. By contrast, the reference for the full W4A4 error is XW_c^{range}.

Table 9: Activation and projection NMSE before and after clipping (Equation 26). All three rows are clip-only comparisons with the initial draft basis and deployment scales $S^\star$ fixed; only c changes from 1 to $c^\star$. Attention-sink rows are excluded. Reduction denotes the relative decrease in $\text{NMSE}_{\text{proj}|W4}$.

Projection	$c^\star$	NMSE_A	$\text{NMSE}_{\text{proj}\|W4}$	Reduction (%)
DFlash interface	0.70	$0.03182 \to 0.01709$	$0.009559 \to 0.005355$	44.0
EAGLE-3 interface	0.70	$0.03702 \to 0.02203$	$0.032345 \to 0.017188$	46.9
DSpark interface	0.60	$0.04673 \to 0.02041$	$0.034523 \to 0.016791$	51.4

All three rows vary only clipping. Including attention-sink rows substantially changes the absolute values of $\text{NMSE}_{\text{proj}|W4}$ (DFlash $0.030695 \to 0.017090$, EAGLE-3 $0.009077 \to 0.005067$, and DSpark $0.131448 \to 0.048726$), but the relative reductions remain in the same direction at 44.3%, 44.2%, and 62.9%, respectively. Because the prompt/decode mix and sample population differ across rows, we do not directly compare their absolute NMSE values.

D Verifier-Guided Rotation Optimization

This section specifies the distributions, gradients, and implementation of the quantization-induced suffix decay (QSD) loss. Matched comparisons that vary only the objective under the same range-first conditions are reported in Appendix F; calibration-order ablations are reported in Table 21.

D.1 Trainable Parameters and Quantized Forward

Let Θ_T and R_T denote the original target weights and rotation, and Θ_D the original draft weights. Range-first calibration selects source scaling and clipping at the initial draft rotation. With $S^\star, c^\star$ and Θ_T, R_T, Θ_D fixed, we optimize only the orthogonal draft rotations $\mathcal{R}_D = \{R_r\}_{r\in\mathcal{I}}$. The residual rotation R_D is parameterized as

$$R_D = R_{D,0}D, \qquad D = \mathrm{Cayley}(A), \quad A^\top = -A \tag{27}$$

Depending on the architecture, we also optimize R_2 over the attention-head dimension. Writing the rotation-dependent effective weight as

$$W_{\text{eff}}(\mathcal{R}_D) = \mathrm{Fuse}\big(\Theta_D, R_T, S^\star, \mathcal{R}_D\big) \tag{28}$$

we quantize both the current effective weight and the activations using the deployment quantization rules in every forward pass, then compute the draft distribution. The range parameters remain fixed, but the W4 weight errors and the A4 errors of draft-internal activations can change as the rotations are updated.

We also apply $S^\star, c^\star$ along the path from target features to the draft context or K/V, so that rotation optimization accounts for the prediction effects of residual input error after calibration. Frozen target features and logits can be cached for their corresponding prefixes. In contrast, draft context and K/V depend on the rotations being optimized and must be computed using the current rotations. Reusing draft K/V from a fixed earlier rotation would not implement the same forward computation.

D.2 Target–Draft Distributions at Anchor Positions

Let i index the prediction block and d the draft position within it. Multiple anchors may be selected from a single dataset row. DFlash uses the token at sequence position a_i as the anchor in slot 0 and predicts $K = 9$ mask slots; slot 0 is excluded from the loss. Let $\mathbf{l}_{T,i,d}$ and $\mathbf{l}_{D,i,d}(\mathcal{R}_D)$ denote the verifier and draft logits at the compared position. Then

$$p_{i,d} = \mathrm{sg}[\mathrm{softmax}(\mathbf{l}_{T,i,d})], \qquad q_{i,d}(\mathcal{R}_D) = \mathrm{softmax}(\mathbf{l}_{D,i,d}(\mathcal{R}_D)) \tag{29}$$

where sg denotes stop-gradient and the teacher is the deployed quantized verifier. Cached features and logits correspond to the prefixes used to generate them.

For DFlash, we obtain $p_{i,d}$ in Equation 29 by teacher-forcing the target's greedy continuation. The draft computes $q_{i,d}$ in parallel from the anchor, mask block, and target features; preceding draft proposals are not provided as token inputs. When using an autoregressive draft's own proposal path, we compare against the distributions assigned by the verifier along that path. In our EAGLE-3 setup, the current quantized draft generates a 6-node greedy chain at each update using the current rotation and $S^\star, c^\star$. At each node, the teacher is the W4A4 verifier distribution conditioned on that path.

The overlap and TV over the common vocabulary are

$$\begin{aligned} a_{i,d} &= \sum_{v\in\mathcal{V}} \min(p_{i,d}(v), q_{i,d}(v)), \\ \mathrm{TV}(p_{i,d}, q_{i,d}) &= \frac{1}{2}\sum_{v\in\mathcal{V}} |p_{i,d}(v) - q_{i,d}(v)| = 1 - a_{i,d} \end{aligned} \tag{30}$$

For a reduced draft vocabulary, we map token IDs to the target vocabulary and assign zero draft probability to unsupported tokens. This zero extension differs from renormalizing the verifier distribution within the draft subset. Zero-support tokens can make full-vocabulary $\mathrm{KL}(p\|q)$ diverge, so KL-based comparisons must specify how support is handled.

D.3 QSD Loss: Prefix Products and Shared Gradients

At each anchor, the QSD loss in Equation 7 sums the prefix products $P_{i,k} = \prod_{d=1}^{k} a_{i,d}$ and normalizes by the valid length K_i. The same overlap prefix-product structure is used in end-to-end TV loss (Li et al., 2026). When all anchors have the same valid length K, QSD and end-to-end TV loss normalized by draft length differ only by a rotation-independent constant. We use this objective

to optimize rotations of a quantized draft while keeping its original weights frozen. Products are computed within each anchor and then averaged, without a fixed per-position discount.

Numerical computation and masks. Each anchor's valid mask $m_{i,d}$ forms a contiguous prefix. Anchor/seed slots, padding, and positions outside the valid continuation are excluded. Let $\mathcal{I}_+$ denote the anchors with $K_i = \sum_d m_{i,d} > 0$, and let $N = |\mathcal{I}_+|$. Verifier and draft probabilities and their overlaps are computed in fp32. To reduce underflow, we compute prefix products as

$$P_{i,k} = \exp\left(\sum_{d=1}^{k} \log \max(a_{i,d}, 10^{-30})\right) \tag{31}$$

and aggregate only terms with $k \le K_i$. A greedy mismatch does not trigger hard masking of subsequent positions: the objective operates on soft overlaps. Anchors of length 0 are excluded, and every remaining anchor has weight $1/N$, regardless of its length.

Dependence on the preceding prefix. For positive overlaps, let $\ell_{i,d} = -\log a_{i,d}$ and consider the regime in which the numerical clamp is inactive. The derivative of QSD with respect to the negative log-overlap is

$$\frac{\partial \mathcal{L}_{\mathrm{QSD}}}{\partial \ell_{i,d}} = \underbrace{\frac{1}{NK_i} \sum_{k=d}^{K_i} P_{i,k}}_{b_{i,d}} \tag{32}$$

An overlap contributes to every prefix term containing its position, so $b_{i,d}$ depends on both the position and the overlap profile within the anchor. The derivative with respect to the overlap itself is

$$-\frac{\partial \mathcal{L}_{\mathrm{QSD}}}{\partial a_{i,d}} = \frac{1}{NK_i} \left(\prod_{j<d} a_{i,j}\right) \left(1 + \sum_{k=d+1}^{K_i} \prod_{j=d+1}^{k} a_{i,j}\right) \tag{33}$$

where an empty product equals 1. Small preceding overlaps reduce the reward for the same improvement in a later overlap. This differs from assigning independent, fixed weights to later-position losses.

Collecting all rotation generators into θ, we obtain

$$\nabla_\theta \mathcal{L}_{\mathrm{QSD}} = \sum_{i \in \mathcal{I}_+} \sum_{d=1}^{K_i} b_{i,d} \nabla_\theta \ell_{i,d} \tag{34}$$

Gradients from different anchors and positions update the same rotation set. For positive overlaps, $b_{i,d}$ is nonincreasing in d. These coefficients therefore do not uniformly upweight later-position negative-log-overlap terms. Instead, QSD couples improvement at a later position to the overlaps along the preceding prefix. This describes the objective's weighting structure, not an independently established empirical advantage. In the matched DFlash comparison, QSD's three-seed mean Avg. AL exceeds that of $\mathrm{TV}_{\mathrm{an}}$ by 0.023, but the seed t interval $[-0.009, +0.055]$ includes zero. $\mathrm{TV}_{\mathrm{an}}$ is the position-wise overlap baseline without prefix coupling (Appendix D.5); the empirical contribution of the prefix product itself is therefore not established.

Relation to measured acceptance. The identity between $1 - \mathrm{TV}(p, q)$ and expected single-token acceptance requires p, q to be the verifier and actual proposal distributions conditioned on the same prefix, with acceptance rule $\min(1, p(v)/q(v))$. The teacher-forced overlap $a_{i,d}$ used here does not generally equal the overlap along such stochastic proposal paths or the empirical conditional acceptance A_d measured by greedy argmax agreement (Equation 45 in Appendix E). QSD is therefore used as a training surrogate for prefix acceptance. We do not use the mean or product of overlaps as measured AL or as an unbiased estimator of it. We assess later-position acceptance recovery and overall performance through actual decoding.

D.4 LK Objectives as Comparison Baselines

We use LK Losses (Samarin et al., 2026) as objective baselines. Matched comparisons with QSD use $\mathrm{LK}_{\mathrm{an}}$, which is normalized per anchor as in QSD (Appendix D.5). Pooled LK, which lacks per-anchor normalization (Equation 39), is included as a reference row in the matched comparison and

is used by the DFlash and DSpark LK arms of the calibration-order ablation. At a single position, LK-α and the KL/TV family are

$$\begin{aligned} \mathcal{L}_\alpha &= -\log(a_d + \epsilon), \\ \mathcal{L}_\lambda &= \lambda\ \mathrm{KL}(p_d \| q_d) + (1-\lambda)\ \mathrm{TV}(p_d, q_d) \end{aligned} \tag{35}$$

The LK-α comparison uses

$$\mathcal{L}_\alpha^{(i,d)} = -\log(a_{i,d} + \epsilon), \qquad \epsilon = 10^{-6} \tag{36}$$

For fixed p, the gradient is

$$\nabla_\theta \mathcal{L}_\alpha = \frac{1}{a+\epsilon} \nabla_\theta\ \mathrm{TV}(p, q_\theta) \tag{37}$$

Thus, LK-α follows the TV gradient direction while increasing the weight of gradients at positions with smaller overlap. At points where $q(v) \neq p(v)$, let $\mathcal{U} = \{v : q(v) < p(v)\}$ and $s_\mathcal{U} = \sum_{v \in \mathcal{U}} q(v)$. Then

$$\frac{\partial \mathcal{L}_\alpha}{\partial l_D(v)} = -\frac{q(v)}{a+\epsilon} \left(\mathbf{1}[v \in \mathcal{U}] - s_\mathcal{U}\right) \tag{38}$$

At $q(v) = p(v)$, we use the subgradient selected by automatic differentiation.

D.5 Pooled and Anchor-Normalized Comparison Objectives

The DFlash and DSpark LK arms of the calibration-order ablation use the pooled normalization

$$\mathcal{L}_{\mathrm{LK}} = \frac{\sum_i \sum_{d=1}^{K} m_{i,d} \gamma^{d-1} \mathcal{L}^{(i,d)}}{\sum_i \sum_{d=1}^{K} m_{i,d} \gamma^{d-1}} \tag{39}$$

with $\gamma = 0.8$, where $\mathcal{L}^{(i,d)}$ is the LK-α loss in Equation 36. Each anchor's total weight is proportional to the sum of its valid depth weights. QSD instead normalizes by K_i within each anchor, so comparisons with pooled LK differ in both prefix coupling and anchor normalization.

To control for anchor weighting, all matched comparison objectives give valid anchors equal weight, as QSD does. For a generic per-position loss $\ell_{i,d}$, the anchor-normalized objective is

$$\mathcal{L}_{\mathrm{an}}[\ell; \gamma] = \frac{1}{N} \sum_{i \in \mathcal{I}_+} \frac{\sum_{d=1}^{K_i} \gamma^{d-1} \ell_{i,d}}{\sum_{d=1}^{K_i} \gamma^{d-1}} \tag{40}$$

This computes a weighted mean over each anchor's valid positions and then an equally weighted mean over valid anchors. The set $\mathcal{I}_+$ and count N are defined as in Equation 7. $\mathrm{LK}_{\mathrm{an}}$ uses $\ell_{i,d} = -\log(a_{i,d} + \epsilon)$ with $\gamma = 0.8$ or $\gamma = 1$. The remaining objectives use $\gamma = 1$: $\mathrm{TV}_{\mathrm{an}}$ uses $\ell_{i,d} = 1 - a_{i,d}$, $\mathrm{KL}_{\mathrm{an}}$ uses $\ell_{i,d} = \mathrm{KL}(p_{i,d} \| q_{i,d})$ over the full vocabulary, and $\mathrm{CE}_{\mathrm{an}}$ uses $\ell_{i,d} = -\log q_{i,d}(y_{i,d})$ for the W4A4 target's greedy token $y_{i,d}$. $\mathrm{DTV}_{\mathrm{an}}$ and $\mathrm{DKL}_{\mathrm{an}}$ replace the teacher p with $\mathrm{sg}(q^{16})$, the 16-bit distribution of the same draft with quantizers and feature-projection clipping disabled.

Up to an additive constant, $\mathrm{TV}_{\mathrm{an}}$ is obtained by replacing QSD's prefix products with individual overlaps. The QSD–$\mathrm{TV}_{\mathrm{an}}$ comparison therefore targets prefix coupling, whereas $\mathrm{TV}_{\mathrm{an}}$ versus $\mathrm{LK}_{\mathrm{an}}$ ($\gamma = 1$) compares linear and logarithmic overlap losses. The EAGLE-3 LK comparison is normalized per anchor and is identical to $\mathrm{LK}_{\mathrm{an}}$ ($\gamma = 0.8$). All anchor-normalized comparisons use the same initial rotation, range, rows, anchors, 400 updates, and final step-400 checkpoint as QSD. QSD itself does not use γ.

D.6 Cayley Parameterization and Quantization Gradients

For each trainable rotation R_r, we introduce an unconstrained matrix B_r and parameterize

$$\begin{aligned} L_r &= \mathrm{tril}(B_r, -1), & A_r &= L_r - L_r^\top, \\ D_r &= \left(I - \frac{1}{2} A_r\right)^{-1} \left(I + \frac{1}{2} A_r\right), & R_r &= R_{r,0} D_r \end{aligned} \tag{41}$$

Since $A_r^\top = -A_r$, we have $D_r^\top D_r = I$; R_r is therefore orthogonal whenever $R_{r,0}$ is orthogonal. Setting $B_r = 0$ gives $D_r = I$, so the step-0 rotation matches the initial Aligned PTQ rotation. The Cayley map can be implemented with a linear solve rather than an explicit inverse. With $P_r = I - A_r/2$, its differential is

$$\mathrm{d}D_r = \frac{1}{2} P_r^{-1}(\mathrm{d}A_r)(D_r + I) = P_r^{-1}(\mathrm{d}A_r)P_r^{-1} \tag{42}$$

Gradients propagate through R_r, D_r, A_r to B_r. Adam (Kingma & Ba, 2015) updates the generators, while the Cayley map maintains orthogonality in each forward pass.

We use the straight-through estimator (STE) for forward passes involving low-bit rounding. For a quantize–dequantize operator Q, the STE

$$Q_{\mathrm{STE}}(X) = X + \mathrm{sg}[Q(X) - X] \tag{43}$$

uses quantized values in the forward pass and the identity gradient in the backward pass. W4 scales are computed from the current rotated weights using the run's quantization rules, but scale/zero-point search and integer rounding are not differentiated.

D.7 Range-First Optimization Procedure

Algorithm 1 performs tensor-level MSE-based range search followed by QSD rotation optimization.

Algorithm 1 Range-first FLASHLight with QSD rotation optimization

Require: Frozen Θ_T, Θ_D, R_T; initial draft rotations $\mathcal{R}_{D,0}$
Require: Range-calibration data $\mathcal{D}_{\mathrm{cal}}$ and rotation-training data $\mathcal{D}_{\mathrm{rot}}$, disjoint from test data; valid horizon K, update budget T_{upd}, learning rate η
Require: Deployment quantizer Q_{dep}
Ensure: Packed draft $\widehat{\Theta}_D$ and range $S^\star, c^\star$

$\mathcal{B} \leftarrow 0$
$S^\star \leftarrow \textsc{SearchScalingMSE}(\mathcal{R}_{D,0}, c = 1, \mathcal{D}_{\mathrm{cal}})$
$c^\star \leftarrow \textsc{SearchClippingMSE}(\mathcal{R}_{D,0}, S^\star, \mathcal{D}_{\mathrm{cal}})$
Freeze $S^\star, c^\star$ throughout rotation learning
for $t = 1, \ldots, T_{\mathrm{upd}}$ **do**
  $\mathcal{R}_D \leftarrow \textsc{CayleyRotations}(\mathcal{B}, \mathcal{R}_{D,0})$
  $(\mathcal{X}, m) \leftarrow \textsc{SamplePrefixes}(\mathcal{D}_{\mathrm{rot}}, K; \mathcal{R}_D, S^\star, c^\star)$ ▷ EAGLE-3: self-rollout
  $(p, \mathcal{H}) \leftarrow \mathrm{sg}(F_T(\mathcal{X}; \Theta_T, R_T))$
  $W_{\mathrm{eff}} \leftarrow \textsc{Fuse}(\Theta_D, R_T, S^\star, \mathcal{R}_D)$
  $q \leftarrow F_D^{\mathrm{STE}}(\mathcal{X}, \mathcal{H}; W_{\mathrm{eff}}, S^\star, c^\star)$
  $\mathcal{L} \leftarrow \textsc{QSDLoss}(p, q, m)$ ▷ Eq. 7
  $\mathcal{B} \leftarrow \textsc{AdamUpdate}(\mathcal{B}, \nabla_{\mathcal{B}}\mathcal{L}, \eta)$
end for
$\mathcal{R}_D^\star \leftarrow \textsc{CayleyRotations}(\mathcal{B}, \mathcal{R}_{D,0})$; freeze $\mathcal{R}_D^\star$ ▷ final step-T_{upd} checkpoint
$\widehat{\Theta}_D \leftarrow \textsc{FuseAndQuantize}(\Theta_D, R_T, \mathcal{R}_D^\star, S^\star, c^\star, Q_{\mathrm{dep}})$
return $(\widehat{\Theta}_D, S^\star, c^\star)$

The function SamplePrefixes returns architecture-specific token prefixes $\mathcal{X}$ and valid mask m. DFlash and DSpark use teacher forcing on W4A4-target greedy continuations, so their prefixes do not depend on the rotations. EAGLE-3 instead uses greedy chains generated by the quantized draft under the current $\mathcal{R}_D$ and $S^\star, c^\star$; its prefixes therefore depend on the rotations and are regenerated at each update. Discrete proposal selection is not differentiated.

F_T returns quantized target distributions and features for the given prefixes. CayleyRotations and F_D^{STE} use Equations 41 and 43, respectively. Fuse incorporates the rotations into the weights through differentiable operations. F_D^{STE} includes the draft context/K/V path with the calibrated range, and draft tensors that depend on the current rotations are recomputed in that forward pass.

Both range searches use tensor-level reconstruction error at the initial rotation $\mathcal{R}_{D,0}$, selecting source scales before evaluating clipping candidates. DFlash and DSpark use coordinate descent (2 passes) on the $2^{\nu/4}$ lattice, starting from $S = I$. DFlash additionally re-evaluates 160 fixed candidates, including the coordinate-descent result, on a separate set of 178 prompts; the same $S^\star$ has the lowest

Table 10: Per-dataset greedy ($T = 0$) acceptance. Range-only applies FLASHLight's $S^\star, c^\star$ at the initial Aligned PTQ draft basis $R_{D,0}$ without further rotation optimization. Avg. AL is the equally weighted mean of the four dataset-level AL values; recovery is the AL ratio relative to the 16-bit draft paired with the same W4A4 target, expressed as a percentage. Bold marks FLASHLight's Avg. AL and recovery.

Method	MT-Bench	GSM8K	ShareGPT	HumanEval	Avg. AL	Recovery (%)
Llama-3.1-8B-Instruct + DFlash						
16-bit target / 16-bit draft	3.92	4.24	3.82	4.89	4.22	–
W4A4 target / 16-bit draft	3.97	4.01	3.79	5.26	4.26	100.0
Aligned PTQ	3.53	3.52	3.40	4.43	3.72	87.4
Range-only W4A4	3.73	3.79	3.57	4.93	4.00	94.1
FLASHLight	3.85	3.96	3.70	5.12	**4.16**	**97.7**
Qwen3-4B + DSpark						
16-bit target / 16-bit draft	3.95	6.34	3.97	5.62	4.97	–
W4A4 target / 16-bit draft	4.90	5.09	4.75	4.77	4.88	100.0
Aligned PTQ	3.88	4.10	3.69	3.76	3.86	79.1
Range-only W4A4	4.65	4.83	4.48	4.48	4.61	94.5
FLASHLight	4.76	5.02	4.60	4.59	**4.74**	**97.2**

objective among these candidates. EAGLE-3 selects α by minimizing RMS-normalized feature-projection output NMSE with Adam. Effective weights are re-quantized for every candidate and rotation update, but $S^\star, c^\star$ are not recalibrated after rotation optimization. We evaluate the final checkpoint after update $T_{\mathrm{upd}} = 400$.

E Acceptance Diagnostics

This section reports per-dataset acceptance and diagnoses W4A4 draft acceptance loss through precision decomposition, position-wise measurements, controlled precision-site interventions, and proposal-length sensitivity.

E.1 Per-Dataset Acceptance

Table 10 reports per-dataset acceptance with target and draft precision distinguished. Target quantization affects AL differently across datasets: for DSpark, AL increases from 3.95 to 4.90 on MT-Bench but decreases from 6.34 to 5.09 on GSM8K. FLASHLight achieves higher AL than Aligned PTQ on every dataset for both architectures.

E.2 Weight and Activation Precision in the Quantized Draft

We vary draft weight and activation precision separately while pairing the basis-aligned DFlash draft with the same W4A4 target. Table 11 reports acceptance and feature-projection NMSE. This experiment diagnoses quantization degradation; it is not a throughput comparison with native hardware kernels at every precision. Its W16A16 reference AL is 4.1435, distinct from the reference in the main acceptance experiment.

Let $A_{w,a}$ denote AL at each precision setting in this experiment. Define the activation-only and weight-only reductions as $\delta_A = A_{16,16} - A_{16,4}$ and $\delta_W = A_{16,16} - A_{4,16}$, respectively, and the full W4A4 reduction as $\delta_{WA} = A_{16,16} - A_{4,4}$. We compare these reductions using

$$r_A = \frac{\delta_A}{\delta_{WA}}, \qquad r_{A/W} = \frac{\delta_A}{\delta_W}. \tag{44}$$

Table 11: Precision decomposition of basis-aligned DFlash. The target is fixed at W4A4.

Draft precision	Avg. AL	Recovery (%)	Projection NMSE
W16A16	4.1435	100.0	0
W4A16	4.0845	98.58	1.18×10^{-2}
W16A4	3.6017	86.93	1.67×10^{-2}
W4A8	4.0808	98.49	1.18×10^{-2}
W8A8	4.1405	99.93	1.47×10^{-4}
W4A4	3.5182	84.91	2.83×10^{-2}

The relative AL reduction is 1.42% for weight-only W4 and 13.07% for activation-only A4. The ratio r_A is 86.6% (95% CI: [84.2%, 89.0%]), and $r_{A/W}$ is 9.19 (95% CI: [7.56, 11.75]). These ratios compare AL reductions; they do not provide an additive attribution satisfying $\delta_A + \delta_W = \delta_{WA}$. The comparisons with W4A16 and W4A8 associate A4 with the main residual loss, but do not localize that loss to the feature projection alone or to a specific draft position.

E.3 Conditional-Acceptance Measurement and Aggregation

Definition. For a fixed-horizon chain, let E_d denote acceptance of the d-th proposal and $C_{d-1} = E_1 \cap \cdots \cap E_{d-1}$ acceptance of all preceding proposals. The event C_0 includes all cycles. The population conditional acceptance at each position and its sample estimator are

$$A_d = \Pr(E_d \mid C_{d-1}), \qquad \widehat{A}_d = \frac{\sum_r \mathbf{1}[C_{r,d-1} \cap E_{r,d}]}{\sum_r \mathbf{1}[C_{r,d-1}]} \tag{45}$$

where r indexes evaluation cycles. The estimator is undefined at positions with a zero denominator. For tree drafts, acceptance is aggregated by depth along the same accepted path. Dataset aggregation and EOS/output-limit handling follow the rules below.

Measurement and aggregation. The following tables and figures use k to index draft positions. With EOS/output-limit censoring, the count-based form of the estimator in Equation 45 is

$$\widehat{A}_k = \frac{N_k^{\text{accept}}}{N_k^{\text{eligible}}}, \tag{46}$$

where N_k^{eligible} counts cycles in which all preceding positions are accepted and the k-th token is actually verified. The numerator counts eligible cycles in which that token is also accepted. For an uncensored, fixed-length verification horizon, the denominator equals the number of prefixes surviving through length $k-1$. The position-wise tables (Appendix E.4) and Figure 1 pool N_k^{accept} and N_k^{eligible} across all cycles from the 604 test turns, without separating datasets. Positions after a committed EOS or beyond the maximum generation length are excluded from both counts because they cannot contribute to the output. An EOS correction token is counted as an observed rejection. For FLASHLight, we first compute pooled $\widehat{A}_k$ separately for each seed, then average the resulting rates over the three seeds.

Under the fixed-horizon chain definition above, let L denote the accepted draft-prefix length and K the horizon. Since $\{L \geq j\} = C_j$, the A_j in Equation 45 equals $\Pr(L \geq j \mid L \geq j-1)$, giving

$$\Pr(L \geq k) = \prod_{j=1}^{k} A_j, \qquad \mathbb{E}[L] = \sum_{k=1}^{K} \prod_{j=1}^{k} A_j. \tag{47}$$

These identities follow from the chain rule of conditional probability and the tail-sum identity, without assuming independence across positions. The length L excludes correction and bonus tokens. If the scheduler shortens verification, unverified positions are not counted as rejections, and Equation 47 must also account for each position's probability of being verified. The main-text DSpark configuration uses fixed-length proposals with $K = 7$, threshold 0, and no pruning (Table 3).

The pooled position-wise rates and the main-text Avg. AL use different dataset weightings: Avg. AL first computes AL within each dataset and then averages the four values equally. For example,

for DFlash with W4A4 target / 16-bit draft, $1 + \sum_k \prod_{j \le k} \widehat{A}_j$ is 4.12, whereas Table 10 reports an Avg. AL of 4.26. The difference arises mainly because pooled counts give greater weight to datasets with more cycles. We therefore report AL from cycle-level aggregates rather than reconstructing it from rounded position-wise ratios. The correspondence between $\mathbb{E}[L] + 1$ and emitted-token AL also depends on consistent treatment of correction/bonus token emission and termination.

E.4 Recovery across Draft Positions

The FLASHLight curves in Figure 1 show recovery of the later-position loss identified in the motivation. For DFlash, A_9 increases from 0.561 under Aligned PTQ to 0.712, approaching the 16-bit draft reference of 0.741 (Figure 1(a); per-seed values: 0.712, 0.704, and 0.721). The final-position improvement of 15.1 percentage points exceeds the first-position improvement of 1.3 percentage points and closes 84% of the final-position gap between Aligned PTQ and the reference. The remaining reference gap ranges from 0.2–2.8 percentage points across positions and is largest at the final position. For DSpark, A_7 similarly increases from 0.758 to 0.873, approaching the reference of 0.891 (per-seed values: 0.875, 0.873, and 0.872) and closing 87% of the final-position gap. The remaining gaps range from 0.6–1.7 percentage points across positions (Figure 1(b)). These improvements extend beyond the first token to the later predictions needed to sustain long accepted prefixes.

Range calibration accounts for most of this late-position recovery. Range-only applies $S^\star$ and $c^\star$ without further rotation optimization and closes 74% and 83% of the final-position gap on DFlash and DSpark, respectively. With the calibrated range parameters fixed, QSD-trained rotations further increase A_k over Range-only by 0.5–2.4 percentage points on DFlash and 0.5–1.3 percentage points on DSpark. The increment is positive at every position for all three seeds. For DFlash, the mean increment over the final three positions is 1.8 percentage points, compared with 1.0 over the first three. For DSpark, the final-position increment is 0.5 percentage points, so the rotation benefit is not concentrated at later positions. Because rollout A_k is evaluated on different reached-prefix sets across configurations, the main-text control in Section 2.2 and Figure 2c compares proposal distributions under identical prefixes. Objective and calibration-order comparisons are provided in Appendix F.

Tables 12 and 13 include the configurations in Figure 1 and the range-calibration stages before rotation optimization. The Source scaling row applies $S^\star$ at the initial draft basis $R_{D,0}$ with $c = 1$. The Clipping row additionally applies $c^\star$, yielding Range-only: the same calibrated range parameters as FLASHLight, without further rotation optimization. The targets follow Table 10; DSpark's W4A4 target uses SpinQuant. The 16-bit target / W4A4 draft row is measured in a separate harness that rotates BF16 target features into the target-rotation basis before feeding them to the same Aligned PTQ draft. As a control, position-wise acceptance measured with a 16-bit draft in this harness differs from the reported 16-bit target / 16-bit draft row by at most 0.13 percentage points for DFlash and 0.09 for DSpark. Relative to the 16-bit draft measured in the same harness, the 16-bit target / W4A4 draft (Aligned PTQ) has first- and final-position gaps of 1.6 and 12.8 percentage points for DFlash, and 7.4 and 11.2 for DSpark. The quantization-induced gap therefore widens toward later positions even with a 16-bit target (Section 2.2). DFlash's QSD training and calibration settings are given in Table 4.

E.5 Controlled Precision-Site Analysis of QSD

The same-prefix probe in Section 2.2 shows that Aligned PTQ suffers substantially more top-1 flips at later draft positions even when every configuration is evaluated on identical target features and prefixes. We further use controlled precision-site interventions on DFlash to narrow down the source of this position-dependent degradation. For the 16-bit draft reference, the probability of a positive target-token margin falls from 0.82 at $d = 1$ to 0.24 at $d = 9$, indicating that later predictions are already less separated from their strongest competitors before draft quantization. However, the magnitude of the W4A4 representation error does not increase toward the suffix: final hidden-state and mean-centered logit errors remain approximately flat across positions, and the ratio of final hidden-state error to interface error changes from 1.76 at $d = 1$ to 1.69 at $d = 9$. Thus, the observed suffix degradation is not explained by a larger representation-error norm or stronger downstream error amplification at later positions.

We next intervene on the two draft quantization sites while keeping the same target features, prefixes, and evaluation anchors. "FP interface + W4A4 internal" restores only the target–draft

Table 12: Conditional acceptance of DFlash / Llama-3.1-8B-Instruct ($K = 9$, $T = 0$). Values are pooled $\widehat{A}_k$ computed from counts over all cycles of the 604 test turns. All rows from Aligned PTQ downward use the same W4A4 target, with drafts evaluated at W4A4. The Source scaling and Clipping rows cumulatively apply range calibration without further rotation optimization. Position-wise gaps and recovery ratios in the main text and appendix are computed from unrounded values and may therefore differ slightly from values recomputed using the displayed three-decimal values (by at most 0.1 percentage point for position-wise gaps).

Configuration	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
16-bit target / 16-bit draft	.831	.777	.758	.760	.757	.756	.743	.739	.720
W4A4 target / 16-bit draft	.802	.768	.759	.765	.771	.774	.777	.756	.741
16-bit target / W4A4 draft (Aligned PTQ)	.815	.756	.725	.716	.695	.668	.639	.610	.593
Aligned PTQ	.787	.747	.718	.710	.699	.672	.655	.610	.561
+ Source scaling ($S^\star$, $c = 1$)	.789	.751	.736	.743	.744	.742	.733	.697	.685
+ Clipping ($c^\star$; Range-only)	.788	.755	.738	.741	.751	.739	.748	.723	.694
FLASHLight (range-first QSD, 3 seeds)	.799	.760	.751	.761	.758	.759	.758	.747	.712

Table 13: Conditional acceptance of DSpark / Qwen3-4B ($K = 7$, $T = 0$). W4A4-target rows use SpinQuant. Aggregation, row definitions, and rounding follow Table 12.

Configuration	A_1	A_2	A_3	A_4	A_5	A_6	A_7
16-bit target / 16-bit draft	.845	.835	.840	.851	.859	.863	.862
W4A4 target / 16-bit draft	.831	.842	.855	.872	.885	.887	.891
16-bit target / W4A4 draft (Aligned PTQ)	.771	.753	.757	.760	.764	.756	.750
Aligned PTQ	.771	.775	.778	.782	.782	.767	.758
+ Source scaling ($S^\star$, $c = 1$)	.814	.820	.838	.851	.860	.859	.858
+ Clipping ($c^\star$; Range-only)	.815	.824	.839	.856	.865	.864	.869
FLASHLight (range-first QSD, 3 seeds)	.822	.833	.849	.862	.874	.876	.873

feature-projection interface to 16-bit while retaining W4A4 quantization inside the draft. Conversely, "W4A4 interface + FP internal" keeps the quantized interface while restoring the remaining draft-internal quantized operations to 16-bit. Table 14 reports the resulting sign-flip rates, defined as in Figure 2c. At $d = 9$, Aligned PTQ has a sign-flip rate of 0.472, whereas restoring only the interface to 16-bit reduces it to 0.268. Quantizing only the interface yields 0.286. Range-only, which preserves the W4A4 execution structure and changes only the calibrated interface range, also reaches 0.286.

Table 14: Controlled DFlash precision-site intervention in the same-prefix probe. Sign flip is $\Pr(m_X < 0 \mid m_{\mathrm{REF16}} > 0)$, with all configurations evaluated on identical target features and prefixes. The table reports the final draft position $d = 9$. These rates pool shared anchors, whereas Figure 2c reports dataset-macro means, accounting for the small numerical differences from the main-text probe.

Configuration	Sign flip at $d = 9$
Aligned PTQ	0.472
FP interface + W4A4 internal	0.268
W4A4 interface + FP internal	0.286
Range-only	0.286

Relative to Aligned PTQ, the 16-bit-interface intervention improves the $d = 9$ flip rate by $0.472 - 0.268 = 0.204$. Range-only recovers $(0.472 - 0.286)/(0.472 - 0.268) \approx 92\%$ of this intervention gap while retaining the W4A4 interface. Thus, range calibration recovers most of the final-position improvement obtained by restoring the interface to 16-bit. The comparable single-site counterfactual rates do not support attributing the degradation to the interface alone. The two precision-site interventions should not be interpreted as an additive attribution of QSD: sign-flip probability is nonlinear, and quantizing the interface alone and the draft internals alone each produces substantially less degradation than quantizing both simultaneously.

As a descriptive control, matching the REF16 margin distribution reduces but does not eliminate the early-to-late increase in sign flips. Because conditioning on the noisy REF16 margin introduces regression-toward-the-mean effects, we do not use this result as a causal decomposition of QSD; the same-slot precision interventions above provide the primary evidence.

E.6 Sensitivity to Proposal Length

We test whether the acceptance recovery above is specific to the proposal length used during calibration and rotation optimization. We reuse the same frozen range parameters and the same seed-0 FLASHLight checkpoint at every K, without recalibration or rotation retraining. DFlash is evaluated on all 604 test turns at $K \in \{2, 4, 6, 9\}$, where $K = 9$ is the training horizon. DSpark is evaluated on a fixed 160-turn subset at $K \in \{2, 4, 6, 7\}$, where $K = 7$ is the training horizon. These are single-checkpoint sensitivity results; the bootstrap intervals below quantify evaluation uncertainty only and do not include training-seed variability.

For this analysis, L is the number of consecutively accepted draft proposals in a cycle and excludes correction and bonus tokens (Appendix E.3). We report the equally weighted macro average of $\mathbb{E}[L]$ over the four datasets. All confidence intervals are 95% paired prompt-cluster bootstrap intervals from 10,000 resamples.

Table 15: Proposal-length sensitivity of expected accepted draft-prefix length $\mathbb{E}[L]$. REF16 is the 16-bit draft reference, PTQ is Aligned PTQ, and Range is Range-only. The same W4A4 target and the same frozen range and seed-0 FLASHLight rotation are used at every K. DFlash uses all 604 test turns; DSpark uses the fixed 160-turn subset. Brackets give 95% paired prompt-cluster bootstrap intervals (10,000 resamples).

Architecture	K	REF16	PTQ	Range	FLASHLight	FLASHLight − PTQ	FLASHLight − Range	FL/REF16 (%)
DFlash	2	1.523	1.452	1.487	1.504	$+0.053$ $[0.047, 0.058]$	$+0.017$ $[0.012, 0.022]$	98.8
	4	2.415	2.192	2.307	2.362	$+0.170$ $[0.158, 0.182]$	$+0.055$ $[0.045, 0.066]$	97.8
	6	2.903	2.536	2.725	2.806	$+0.270$ $[0.250, 0.290]$	$+0.080$ $[0.065, 0.096]$	96.7
	9	3.241	2.706	2.989	3.130	$+0.424$ $[0.396, 0.455]$	$+0.141$ $[0.124, 0.159]$	96.6
DSpark	2	1.595	1.325	1.527	1.562	$+0.237$ $[0.195, 0.294]$	$+0.035$ $[0.027, 0.044]$	97.9
	4	2.779	2.207	2.634	2.697	$+0.490$ $[0.397, 0.614]$	$+0.064$ $[0.047, 0.080]$	97.1
	6	3.620	2.720	3.417	3.527	$+0.807$ $[0.675, 0.984]$	$+0.110$ $[0.086, 0.134]$	97.4
	7	3.894	2.816	3.612	3.770	$+0.954$ $[0.802, 1.149]$	$+0.159$ $[0.134, 0.183]$	96.8

At every tested K, FLASHLight has higher $\mathbb{E}[L]$ than both Aligned PTQ and Range-only, and the paired intervals for both differences exclude zero. As K increases from 2 to the native training horizon, Aligned PTQ's retention relative to REF16 falls from 95.3% to 83.5% on DFlash and from 83.1% to 72.3% on DSpark. FLASHLight remains at 98.8–96.6% and 97.9–96.8%, respectively. Equivalently, FLASHLight recovers 73.5–79.3% of the PTQ loss in $\mathbb{E}[L]$ across the DFlash sweep and 85.7–89.7% across the DSpark sweep. The absolute FLASHLight gains over both PTQ and Range-only also increase monotonically over the tested horizons for this seed-0 checkpoint. We interpret this as proposal-length robustness of the fixed learned representation, not as evidence that the largest tested K is throughput-optimal or that prefix coupling in the QSD objective has an independently established advantage.

The complete-prefix statistic shows the same pattern. At the native horizon, FLASHLight raises the probability of accepting the entire proposal block from 0.034 to 0.087 on DFlash ($K = 9$), compared with 0.101 for REF16 and 0.074 for Range-only. On the DSpark subset at $K = 7$, it raises the probability from 0.167 to 0.348, compared with 0.373 for REF16 and 0.322 for Range-only. Thus, the gain in $\mathbb{E}[L]$ is accompanied by a substantially higher probability of preserving the longest available accepted prefix.

F Ablations and Learning-Signal Controls

This section examines FLASHLight components individually and in combination, compares rotation-learning objectives and teachers under fixed range-first conditions, and evaluates the effect of calibration order. Unless otherwise stated, all comparisons follow the data splits, checkpoints,

Table 16: Probability of accepting all K draft proposals in the proposal-length sweep. Population and uncertainty follow Table 15.

Architecture	K	REF16	PTQ	Range	FLASHLight	FLASHLight − PTQ	FLASHLight − Range
DFlash	2	.679	.632	.657	.668	$+.036$ [.032, .039]	$+.011$ [.008, .014]
	4	.407	.320	.368	.387	$+.067$ [.062, .073]	$+.019$ [.015, .022]
	6	.238	.144	.200	.217	$+.073$ [.066, .081]	$+.017$ [.014, .020]
	9	.101	.034	.074	.087	$+.053$ [.044, .062]	$+.013$ [.010, .016]
DSpark	2	.742	.581	.702	.721	$+.140$ [.118, .168]	$+.019$ [.014, .024]
	4	.566	.375	.516	.538	$+.162$ [.139, .190]	$+.022$ [.017, .027]
	6	.435	.226	.388	.412	$+.185$ [.162, .212]	$+.024$ [.019, .029]
	7	.373	.167	.322	.348	$+.181$ [.156, .209]	$+.027$ [.022, .031]

Table 17: Component ablation: Avg. AL (recovery relative to the 16-bit draft reference, %). Within each architecture, all rows use the same W4A4 target and evaluation prompts as Tables 10 and 23. $S^\star, c^\star$ are the FLASHLight range settings for that architecture. [a]Mean over three training seeds; [b]exploratory screening with seed 0 only. The bottom two rows report per-seed Avg. AL differences for FLASHLight − Range-only and their mean with the seed t interval (2 degrees of freedom). Bold marks the highest value among W4A4 draft configurations for each architecture.

Configuration	S	c	Rotation learning	DFlash	EAGLE-3	DSpark
W4A4 target / 16-bit draft	–	–	–	4.257 (100.0)	5.206 (100.0)	4.878 (100.0)
Aligned PTQ	I	1	–	3.720 (87.4)	4.342 (83.4)	3.859 (79.1)
Scaling only	$S^\star$	1	–	3.974 (93.4)	4.540 (87.2)	4.571 (93.7)
Clipping only	I	$c^\star$	–	3.813 (89.6)	4.512 (86.7)	4.266 (87.5)
Range-only	$S^\star$	$c^\star$	–	4.004 (94.1)	4.862 (93.4)	4.609 (94.5)
QSD rotation-only	I	1	QSD at $S{=}I, c{=}1$	3.861 (90.7)[a]	4.728 (90.8)[b]	4.156 (85.2)[b]
FLASHLight (range-first QSD)	$S^\star$	$c^\star$	QSD at $S^\star, c^\star$	**4.158 (97.7)**[a]	**5.019 (96.4)**[a]	**4.742 (97.2)**[a]
FLASHLight − Range-only, seed 0/1/2				+0.142/+0.161/+0.160	+0.191/+0.145/+0.137	+0.141/+0.125/+0.133
Mean [seed t interval]				+0.154 [+0.127, +0.181]	+0.158 [+0.086, +0.229]	+0.133 [+0.114, +0.153]

and reporting conventions in Appendix A. Evaluation uncertainty is reported as 95% intervals from a paired prompt-cluster bootstrap (10,000 resamples), with conversations as clusters for MT-Bench and prompt stems for ShareGPT. Training-seed variability is reported separately using t intervals with 2 degrees of freedom over paired differences across the three seeds.

F.1 Component Ablation

Table 17 reports the effects of applying source scaling, clipping, and rotation optimization to Aligned PTQ, individually or in combination. Scaling and clipping use FLASHLight's $S^\star, c^\star$ for each architecture. DFlash and DSpark clip only the feature-projection input ($c^\star = 0.7$ and 0.6, respectively), whereas EAGLE-3 applies a shared $c^\star = 0.7$ to 8 A4 sites. Range-only applies $S^\star, c^\star$ at the initial draft basis $R_{D,0}$ without further rotation optimization. QSD rotation-only optimizes rotations with QSD at $S = I, c = 1$ and evaluates them under the same conditions, corresponding to the training stage of rotation-first. It is therefore a component ablation, not a FLASHLight checkpoint with its range calibration removed. Re-evaluating DFlash Aligned PTQ in the same harness reproduces the outputs, cycle counts, and accepted prefixes of all 604 turns from the run in Table 10.

For all three architectures, AL point estimates increase from QSD rotation-only to Range-only to FLASHLight. Range-only closes 53%, 60%, and 74% of the AL gap between Aligned PTQ and the 16-bit draft reference on DFlash, EAGLE-3, and DSpark, respectively, where the fraction closed is $(\mathrm{AL} - \mathrm{AL}_{\mathrm{PTQ}})/(\mathrm{AL}_{\mathrm{REF}} - \mathrm{AL}_{\mathrm{PTQ}})$. Optimizing rotations with QSD under the same calibrated range parameters increases these fractions to 82%, 78%, and 87%.

With the calibrated range parameters fixed, rotation optimization adds $+0.154$ AL on DFlash, $+0.158$ on EAGLE-3, and $+0.133$ on DSpark. The increment is positive for every seed in all three architectures, and each seed t interval excludes zero. For DSpark, the 95% paired prompt-cluster bootstrap interval (10,000 resamples) for this increment is $[+0.120, +0.146]$. FLASHLight's per-seed Avg. AL values are 4.750, 4.734, and 4.742 (mean 4.742, across-seed standard deviation 0.008).

Table 18: Matched objective comparison on DFlash (range-first). The teacher is the W4A4 verifier distribution (p), the verifier's greedy token (y), or the 16-bit draft distribution (q^{16}). Δ is the three-seed mean of the paired Avg. AL difference (comparison objective minus QSD); intervals are seed t intervals (2 degrees of freedom), and $n_>$ counts seeds with higher AL than QSD. S_9 is the fraction of cycles accepting all 9 draft tokens among those whose verification horizon is not truncated by EOS or the generation limit; $\bar{A}_{7:9}$ is the mean of $\widehat{A}_7$–$\widehat{A}_9$. Both are three-seed means of dataset-macro averages. Train reports mean training time per run on a single RTX 4090. Pooled LK is a reference row with different anchor normalization. Bold marks the maximum in the Mean, Recovery, S_9, and $\bar{A}_{7:9}$ columns.

Objective	Teacher	Seed 0	Seed 1	Seed 2	Mean	Recovery (%)	Δ vs. QSD	Seed t interval	$n_>$	S_9	$\bar{A}_{7:9}$	Train (min)
QSD	p	4.146	4.165	4.164	**4.158**	**97.7**	–	–	–	**.089**	.731	37.4
$\mathrm{TV}_{\mathrm{an}}$	p	4.132	4.127	4.147	4.135	97.1	-0.023	$[-0.055, +0.009]$	0/3	.088	**.733**	37.4
$\mathrm{LK}_{\mathrm{an}}$, $\gamma = 1$	p	4.100	4.125	4.096	4.107	96.5	-0.051	$[-0.088, -0.014]$	0/3	.083	.722	37.4
$\mathrm{LK}_{\mathrm{an}}$, $\gamma = 0.8$	p	4.099	4.117	4.088	4.101	96.3	-0.057	$[-0.099, -0.016]$	0/3	.082	.722	37.6
$\mathrm{KL}_{\mathrm{an}}$	p	4.074	4.083	4.069	4.075	95.7	-0.083	$[-0.111, -0.055]$	0/3	.080	.717	37.4
$\mathrm{CE}_{\mathrm{an}}$	y	3.991	4.065	4.085	4.047	95.1	-0.111	$[-0.208, -0.014]$	0/3	.076	.712	37.4
$\mathrm{DTV}_{\mathrm{an}}$	q^{16}	4.090	4.082	4.082	4.085	96.0	-0.073	$[-0.112, -0.035]$	0/3	.081	.719	40.1
$\mathrm{DKL}_{\mathrm{an}}$	q^{16}	4.086	4.063	4.065	4.071	95.6	-0.087	$[-0.146, -0.028]$	0/3	.079	.715	40.2
Pooled LK, $\gamma = 0.8$ (reference)	p	4.119	4.121	4.068	4.102	96.4	-0.056	$[-0.145, +0.033]$	0/3	.080	.717	37.4

For DFlash and DSpark, scaling alone yields most of the AL increase achieved by Range-only (scaling only: 3.974 and 4.571; Range-only: 4.004 and 4.609). For EAGLE-3, scaling and clipping yield similar improvements when applied individually (4.540 and 4.512), while their combination reaches 4.862.

F.2 Effect of the Rotation Learning Signal

To compare learning signals without changing the range or training conditions, we fix the range-first setup on DFlash and vary only the rotation-learning objective. All configurations use the same initial basis $R_{D,0}$, source scales $S^\star$, feature-projection clipping $c^\star = 0.7$, training rows and anchors within each seed, and 400 Adam updates at a learning rate of 2×10^{-3}. The matched objectives are normalized per anchor, giving valid anchors equal weight as in QSD. $\mathrm{LK}_{\mathrm{an}}$ ($\gamma = 0.8$, 1), $\mathrm{TV}_{\mathrm{an}}$, $\mathrm{KL}_{\mathrm{an}}$, and $\mathrm{CE}_{\mathrm{an}}$, together with $\mathrm{DTV}_{\mathrm{an}}$ and $\mathrm{DKL}_{\mathrm{an}}$, which use the 16-bit draft distribution q^{16} as teacher, are defined in Equation 40 of Appendix D.5. Overlap and KL are computed over the full vocabulary without renormalization. $\mathrm{TV}_{\mathrm{an}}$ uses the same anchor weighting and position-wise overlaps as QSD, but without prefix coupling. $\mathrm{DTV}_{\mathrm{an}}$ and $\mathrm{DKL}_{\mathrm{an}}$ use the draft distribution, rather than verifier distributions or greedy-token labels, as their supervision target. Pooled LK ($\gamma = 0.8$) from Equation 39, which lacks per-anchor normalization, is included as a reference row. LK objectives are used only for comparison, not as FLASHLight's training objective.

QSD achieves the highest mean AL in the matched comparison (Table 18). $\mathrm{LK}_{\mathrm{an}}$ ($\gamma = 0.8$, 1), $\mathrm{KL}_{\mathrm{an}}$, $\mathrm{CE}_{\mathrm{an}}$, and both 16-bit-draft-teacher objectives yield lower AL than QSD in all three seeds, with seed t intervals excluding zero. $\mathrm{TV}_{\mathrm{an}}$, which averages individual overlaps without prefix coupling, also yields lower AL in every seed, but its difference from QSD (-0.023) is not statistically resolved: the seed t interval $[-0.055, +0.009]$ includes zero.

The $\mathrm{TV}_{\mathrm{an}}$ minus $\mathrm{LK}_{\mathrm{an}}$ ($\gamma = 0.8$) difference of $+0.034$ ($[-0.026, +0.095]$) is likewise not resolved with three seeds. At the level of point estimates, this contrast accounts for about 60% of the difference between QSD and $\mathrm{LK}_{\mathrm{an}}$. Thus, although QSD achieves higher AL than the matched LK-, KL-, and CE-family objectives, these comparisons do not establish whether its advantage comes from prefix coupling or from directly optimizing overlap.

On DFlash, the observed difference due to anchor normalization is small: $\mathrm{LK}_{\mathrm{an}}$ ($\gamma = 0.8$) minus pooled LK is -0.002 (bootstrap interval $[-0.010, +0.008]$; seed t interval $[-0.051, +0.048]$). The additional teacher forward pass in the 16-bit-draft-teacher configurations increases per-step training time by a factor of about 1.15 (3.08 s vs. 2.67 s). The remaining matched configurations have nearly identical per-run training times (37.4–37.6 min) and peak memory allocations (15.1–15.3 GiB).

Verifier teacher versus 16-bit-draft teacher. Table 19 presents a 2×2 comparison of teacher (the W4A4 verifier distribution p or the 16-bit draft distribution q^{16}) and divergence (TV or KL). Within each seed, the four configurations share training rows, anchors, and initialization, allowing

paired teacher comparisons. Under TV, the verifier teacher improves Avg. AL by $+0.050$ over the 16-bit-draft teacher; the difference is positive in all three seeds and its seed t interval excludes zero. Under KL, the teacher-difference estimate is $+0.004$, which is not statistically resolved. The difference between the two teacher effects, $+0.046$, is also not established because its seed t interval $[-0.001, +0.093]$ includes zero. With the verifier teacher, TV exceeds KL by $+0.060$; with the 16-bit-draft teacher, the difference between divergences is not resolved. These comparisons support a benefit from verifier supervision under the overlap-based objective, but do not establish such a benefit under KL.

Table 19: Teacher $\times$ divergence comparison on DFlash (range-first, three-seed mean Avg. AL). Differences are means of within-seed paired differences; brackets give seed t intervals (2 degrees of freedom), and parentheses give the number of seeds with a positive difference.

Divergence	Verifier teacher p	16-bit-draft teacher q^{16}	Verifier − 16-bit draft
TV	4.135	4.085	$+0.050$ $[+0.018, +0.082]$ (3/3)
KL	4.075	4.071	$+0.004$ $[-0.036, +0.044]$ (2/3)
TV − KL	$+0.060$ $[+0.019, +0.101]$ (3/3)	$+0.014$ $[-0.007, +0.034]$ (3/3)	$+0.046$ $[-0.001, +0.093]$ (3/3)

DSpark and EAGLE-3. Table 22 reports the three-seed range-first QSD versus LK contrasts for these architectures. On DSpark, QSD exceeds pooled LK in all three seeds ($+0.035$), but the seed t interval includes zero. For EAGLE-3, the comparison uses chain LK, which takes a $\gamma = 0.8$-weighted mean over the 6 positions of each anchor and is therefore equivalent to $\mathrm{LK}_{\mathrm{an}}$ ($\gamma = 0.8$). The three-seed QSD-minus-LK difference is -0.004, with no observed QSD advantage. The structural difference between EAGLE-3 training and evaluation is discussed in Appendix G.

Position-wise analysis. Table 20 reports first-rejection positions and later-position conditional acceptance from the matched evaluation. First-rejection distributions use only cycles whose verification horizon is not truncated by EOS or the generation-length limit. Conditional acceptance $\widehat{A}_k$ is aggregated over actually verified positions according to Equation 46. At every position, the first-rejection probability differs by less than 0.6 percentage points between QSD and each comparison objective. Among the matched objectives, QSD has the highest fraction of cycles accepting all 9 draft tokens, at 8.88% ($\mathrm{TV}_{\mathrm{an}}$: 8.79%; $\mathrm{LK}_{\mathrm{an}}$, $\gamma = 0.8$: 8.19%), between Range-only (7.53%) and the 16-bit draft reference (10.30%). QSD's $\bar{A}_{7:9}$ exceeds that of $\mathrm{LK}_{\mathrm{an}}$ ($\gamma = 0.8$) by $+0.0097$ (bootstrap interval $[+0.0049, +0.0145]$), but its difference from $\mathrm{TV}_{\mathrm{an}}$, -0.0017 ($[-0.0061, +0.0026]$), is not statistically resolved. These values are macro averages of per-dataset ratios and are not directly comparable to the cycle-pooled values in Figure 1 and Table 12.

Table 20: First-rejection position distribution (%) and later-position conditional acceptance for the DFlash matched objectives. Column k gives the fraction of cycles with an untruncated verification horizon whose first rejection occurs at the k-th draft token; "None" gives the fraction accepting all 9 draft tokens. These are macro averages of per-dataset ratios. $\bar{A}_{7:9}$ is the mean of $\widehat{A}_7, \widehat{A}_8, \widehat{A}_9$. The top three rows are single evaluations; the remaining rows are means over three training seeds.

Configuration	k=1	2	3	4	5	6	7	8	9	None	$\bar{A}_{7:9}$	$\widehat{A}_9$
W4A4 target / 16-bit draft	18.30	17.84	14.85	10.99	8.33	6.53	5.06	4.30	3.50	10.30	.751	.732
Aligned PTQ	20.05	19.19	16.72	12.48	9.31	7.16	5.13	3.82	2.64	3.49	.601	.557
Range-only	19.71	18.58	15.64	11.38	8.32	6.70	4.89	3.94	3.30	7.53	.712	.684
QSD	18.56	18.35	15.10	10.96	8.65	6.65	5.17	4.11	3.57	8.88	.731	.705
$\mathrm{TV}_{\mathrm{an}}$	18.78	18.41	15.23	11.04	8.38	6.65	5.09	4.16	3.46	8.79	.733	.712
$\mathrm{LK}_{\mathrm{an}}, \gamma = 1$	18.72	18.48	15.51	11.23	8.40	6.57	5.03	4.23	3.51	8.33	.722	.695
$\mathrm{LK}_{\mathrm{an}}, \gamma = 0.8$	18.68	18.56	15.41	11.17	8.49	6.79	5.16	4.17	3.39	8.19	.722	.700
$\mathrm{KL}_{\mathrm{an}}$	18.91	18.67	15.37	11.24	8.56	6.65	5.14	4.16	3.35	7.96	.717	.694
$\mathrm{CE}_{\mathrm{an}}$	18.96	18.74	15.62	11.19	8.49	6.84	5.17	4.07	3.28	7.65	.712	.692
$\mathrm{DTV}_{\mathrm{an}}$	18.93	18.46	15.54	11.14	8.46	6.64	5.25	4.11	3.39	8.08	.719	.696
$\mathrm{DKL}_{\mathrm{an}}$	18.99	18.58	15.42	11.27	8.35	6.83	5.13	4.09	3.42	7.92	.715	.688
Pooled LK, $\gamma = 0.8$ (reference)	18.57	18.41	15.50	11.25	8.54	6.83	5.20	4.29	3.37	8.03	.717	.695

F.3 Effect of Range Calibration Order

Table 21 crosses calibration order with rotation objective. Range-first selects and fixes $S^\star, c^\star$ at the initial draft basis before optimizing rotations. Rotation-first optimizes rotations at $S = I, c =$

Table 21: Calibration-order and rotation-objective ablation: mean Avg. AL over three training seeds (recovery). Range-first uses scaling–clipping–rotation; rotation-first uses rotation–scaling–clipping. LK denotes pooled LK ($\gamma = 0.8$) for DFlash and DSpark and anchor-normalized chain LK for EAGLE-3. Bold marks the per-architecture maximum at the displayed precision.

Architecture	Rotation-first LK	Rotation-first QSD	Range-first LK	Range-first QSD
DFlash	4.086 (96.0%)	4.113 (96.6%)	4.102 (96.4%)	**4.158 (97.7%)**
DSpark	4.685 (96.0%)	4.683 (96.0%)	4.708 (96.5%)	**4.742 (97.2%)**
EAGLE-3	4.985 (95.7%)	5.021 (96.4%)	**5.023 (96.5%)**	5.019 (96.4%)

1, then performs the same tensor-level range search at the learned basis. The DFlash and DSpark LK configurations use pooled LK ($\gamma = 0.8$) as defined in Appendix D.5; EAGLE-3 uses anchor-normalized chain LK. Rotation-first results here serve only as calibration-order ablations. Table 22 reports paired differences within each seed.

Table 22: Per-seed paired contrasts of calibration order and objective (Avg. AL differences). Intervals are seed t intervals for the mean over three seeds (2 degrees of freedom); $n_{>0}$ counts seeds with a positive difference.

Architecture	Contrast	Seed 0	Seed 1	Seed 2	Mean	Seed t interval	$n_{>0}$
DFlash	QSD: range-first − rotation-first	$+0.029$	$+0.045$	$+0.061$	$+0.045$	$[+0.005, +0.086]$	3/3
	LK: range-first − rotation-first	$+0.032$	$+0.026$	-0.010	$+0.016$	$[-0.039, +0.071]$	2/3
	Range-first: QSD − LK	$+0.027$	$+0.044$	$+0.096$	$+0.056$	$[-0.033, +0.145]$	3/3
	Rotation-first: QSD − LK	$+0.030$	$+0.025$	$+0.025$	$+0.027$	$[+0.019, +0.034]$	3/3
DSpark	QSD: range-first − rotation-first	$+0.085$	$+0.035$	$+0.059$	$+0.060$	$[-0.003, +0.122]$	3/3
	LK: range-first − rotation-first	$+0.008$	$+0.040$	$+0.021$	$+0.023$	$[-0.016, +0.062]$	3/3
	Range-first: QSD − LK	$+0.054$	$+0.009$	$+0.041$	$+0.035$	$[-0.023, +0.092]$	3/3
	Rotation-first: QSD − LK	-0.023	$+0.014$	$+0.004$	-0.002	$[-0.049, +0.045]$	2/3
EAGLE-3	QSD: range-first − rotation-first	-0.008	$+0.041$	-0.039	-0.002	$[-0.103, +0.099]$	1/3
	LK: range-first − rotation-first	$+0.014$	$+0.019$	$+0.082$	$+0.038$	$[-0.057, +0.134]$	3/3
	Range-first: QSD − LK	-0.026	$+0.052$	-0.037	-0.004	$[-0.124, +0.117]$	1/3
	Rotation-first: QSD − LK	-0.004	$+0.029$	$+0.084$	$+0.037$	$[-0.075, +0.148]$	2/3

With QSD, range-first increases DFlash AL by $+0.045$ over rotation-first. The difference is positive in all three seeds, and the seed t interval $[+0.005, +0.086]$ excludes zero. DSpark also improves in all three seeds ($+0.060$), but its t interval $[-0.003, +0.122]$ includes zero. No QSD ordering advantage is established for EAGLE-3 (-0.002; positive in 1/3 seeds). With LK, the ordering contrast has a positive point estimate for every architecture, but all t intervals include zero.

Under range-first, QSD exceeds pooled LK in every seed for DFlash and DSpark ($+0.056$ and $+0.035$), but both t intervals include zero. For EAGLE-3, range-first LK has a higher mean than QSD (5.023 vs. 5.019). The wide DFlash interval reflects the variability of the paired differences ($+0.027$, $+0.044$, $+0.096$). By contrast, in the matched comparison that includes these runs (Table 18), QSD's contrasts with anchor-normalized LK have seed t intervals excluding zero. Range-first QSD has the highest mean AL among the four configurations on DFlash and DSpark, whereas differences among range-first LK, range-first QSD, and rotation-first QSD on EAGLE-3 are not resolved by the current seed comparisons.

The EAGLE-3 rotation-first runs reselect $\alpha^\star$ and $c^\star$ (0.65–0.75) for each seed. For DSpark, rotation-first selects the same range as range-first in 4 of 6 runs. The ordering contrasts for these architectures therefore also include differences in the final range. For DFlash, however, rotation-first reselects the same $S^\star$ and $c^\star = 0.7$ as range-first for all 16 rotations examined. In this DFlash comparison, the ordering contrast concerns whether the calibrated range is present during rotation optimization, rather than a difference in the final selected range. This is consistent with the interpretation in Section 3.2 that optimizing rotations using features and K/V constructed under the deployment range is beneficial, but does not isolate the causal contribution of individual error sources.

G RESULTS FOR TREE-BASED DRAFTING: EAGLE-3

The main text focuses on DFlash and DSpark, which use fixed-horizon block proposals. Here, we evaluate FLASHLight on EAGLE-3 (Li et al., 2025), a tree-based autoregressive drafting architecture. We retain the range-first procedure but use architecture-specific range calibration and QSD rollouts, as described below. The target is the same Llama-3.1-8B-Instruct checkpoint used for DFlash, quantized with the same W4A4 SpinQuant settings. Verification uses a tree depth of 5, top-k of 10, and a verification budget of 60. Draft activations use asymmetric per-token A4 quantization with a zero point.

At the initial Hadamard basis, we calibrate the source scales α with Adam by minimizing RMS-normalized feature-projection output NMSE. We select a shared clipping ratio for the 8 A4 sites—the feature projection, Q/K/V, O, and gate/up/down—by minimizing final hidden-state NMSE relative to the 16-bit draft ($c^\star = 0.70$; Appendices C.3 and C.7).

For rotation optimization, QSD is computed along a 6-node greedy chain generated by the quantized draft under the current rotation and fixed $S^\star, c^\star$. Overlap is computed over the 32k-token draft vocabulary without renormalizing the verifier distribution (Appendix D.2). AL is evaluated using PyTorch fake-quant W4A4 (fp16). Draft-path fusion is described in Appendix B.6; checkpoints, data, and execution settings are listed in Table 3, and QSD optimization and range-calibration settings in Table 4.

Table 23: Greedy ($T = 0$) acceptance of EAGLE-3 / Llama-3.1-8B-Instruct. Row definitions and aggregation follow Table 10; Range-only W4A4 uses the FLASHLight range without further rotation optimization.

Method	MT-Bench	GSM8K	ShareGPT	HumanEval	Avg. AL	Recovery (%)
16-bit target / 16-bit draft	5.36	5.43	4.99	5.66	5.36	–
W4A4 target / 16-bit draft	5.24	5.23	5.06	5.30	5.21	100.0
Aligned PTQ	4.35	4.35	4.28	4.39	4.34	83.4
Range-only W4A4	4.83	4.83	4.75	5.04	4.86	93.4
FLASHLight	5.02	5.06	4.79	5.20	**5.02**	**96.4**

Acceptance. FLASHLight achieves an Avg. AL of 5.02, approximately 15.7% higher than Aligned PTQ (4.34), and retains 96.41% of the AL of the 16-bit draft paired with the same W4A4 target (Table 23). Range-only closes 60% of the AL gap between Aligned PTQ and the 16-bit draft reference, reaching 93.4% recovery. Optimizing rotations with these calibrated range parameters fixed further increases Avg. AL by $+0.158$ (seed t interval $[+0.086, +0.229]$; positive in all three seeds), closing 45.8% of the gap remaining after range calibration (Table 17).

Objective and calibration order. The EAGLE-3 comparisons do not establish an advantage of QSD over LK or a benefit from range-first ordering under QSD. Under range-first, the Avg. AL difference between QSD and anchor-normalized chain LK ($\gamma = 0.8$) is -0.004 (seed t interval $[-0.124, +0.117]$; positive in 1/3 seeds). For QSD, the range-first minus rotation-first difference is -0.002 (seed t interval $[-0.103, +0.099]$; positive in 1/3 seeds; Tables 21 and 22). Differences among range-first LK, range-first QSD, and rotation-first QSD are not resolved by these three-seed comparisons. The ordering contrasts also include differences in the ranges reselected for each seed (Appendix F.3). QSD is computed along a greedy chain over a reduced draft vocabulary, whereas AL is evaluated with tree verification. We do not isolate the effects of these training–evaluation differences on the objective comparison.

Throughput. At $B = 1$, FLASHLight achieves 459.7 tokens/s: 26.3% higher throughput than the 16-bit draft and 16.0% higher than Aligned PTQ, but 1.4% lower than W4A16 GPTQ (466.3 tokens/s; Table 24). Its mean throughput differs from W4A16 GPTQ by less than 0.1% at $B = 2$ and is 0.9%, 2.8%, and 1.7% higher at $B = 4, 8, 16$, respectively.

Relative to the 16-bit draft, FLASHLight achieves 4.4% higher throughput at $B = 8$ but only 0.992 times its throughput at $B = 16$. At the latter batch size, Aligned PTQ has 14.2% lower throughput than the 16-bit draft; it also has 3.0% and 9.9% lower throughput at $B = 4, 8$, respectively. Batch-size saturation and latency are reported in Appendices H.4 and H.5.

Table 24: End-to-end throughput of EAGLE-3 on a single RTX 4090 (tokens/s). The measurement protocol and notation follow Table 2; all values are means over 3 repeated runs.

Architecture	Batch	16-bit draft	W4A16 GPTQ (per-ch., aligned)	Aligned PTQ W4A4	FLASHLight W4A4
EAGLE-3	1	364.0	**466.3 (**$\times$1.281**)**	396.3 ($\times$1.089)	459.7 ($\times$1.263)
	2	628.5	759.8 ($\times$1.209)	652.4 ($\times$1.038)	**760.1 (**$\times$1.209**)**
	4	860.8	958.4 ($\times$1.113)	834.8 ($\times$0.970)	**966.6 (**$\times$1.123**)**
	8	992.0	1007.8 ($\times$1.016)	893.5 ($\times$0.901)	**1035.6 (**$\times$1.044**)**
	16	**1074.8**	1048.4 ($\times$0.975)	922.5 ($\times$0.858)	1065.8 ($\times$0.992)

H System Implementation and Runtime Analysis

This section examines the relationship between draft cost and acceptance, describes the native INT4 backend and roofline accounting, and reports batch-dependent runtime behavior, latency, and run-to-run variability. The measurement protocol is provided in Appendix A.4.

H.1 Draft Cost, Acceptance, and Acceleration Limits

Measured phase shares and throughput opportunity. The core latencies in Figure 2a are measured on an RTX 4090 with the Llama-3.1-8B-Instruct target in 16-bit, W8A8 SmoothQuant (Xiao et al., 2023), or W4A4 SpinQuant, while keeping the draft in 16-bit. Each phase share is computed relative to the sum of the drafting and verification latencies. Figure 2b reports decode-phase throughput at batch size 1, aggregated over repeated measurements on 352 requests: both turns of the 80 MT-Bench questions and 64 prompts each from GSM8K, ShareGPT, and HumanEval, with at most 128 generated tokens per request. DFlash and EAGLE-3 use Llama-3.1-8B-Instruct as the target, whereas DSpark uses Qwen3-4B. This workload and aggregation differ from those in Table 2. Ideal PTQ (T-4, D-4) is an estimate that combines the Aligned PTQ execution cost with the 16-bit draft's AL from the same measurement setting. It represents retaining the 16-bit draft's AL at W4A4 draft execution cost under the same W4A4 target.

Latency model. We derive upper bounds on further acceleration from the W4A4-target, 16-bit-draft configuration in Figure 2a. The workload and speculative configuration are fixed, and draft generation and target verification are assumed to execute sequentially. Let T_D and T_V denote their respective mean latencies per cycle. The core latency and the draft's share of it are

$$T_{\text{core}} = T_D + T_V, \qquad \rho_D = \frac{T_D}{T_D + T_V} \tag{48}$$

The measured (T_D, T_V) are $(5.92, 7.58)\,\text{ms}$ for EAGLE-3 and $(3.85, 6.46)\,\text{ms}$ for DFlash, giving ρ_D of approximately 43.9% and 37.3%, respectively. These workload-specific phase shares also reflect differences in verification budget (60 and 10) and drafting strategy (sequential autoregressive versus parallel diffusion). We first derive bounds with AL held fixed.

Target-only acceleration. If only target verification is accelerated by a factor of $s_T \geq 1$, with draft cost and AL fixed, the core-latency speedup is

$$\begin{aligned} S_T(s_T) &= \frac{T_D + T_V}{T_D + T_V/s_T} = \frac{1}{\rho_D + (1-\rho_D)/s_T}, \\ S_T^{\max} &= \lim_{s_T \to \infty} S_T(s_T) = \frac{1}{\rho_D}. \end{aligned} \tag{49}$$

The remaining draft cost determines the target-only acceleration limit.

Draft optimization opportunity. Keeping the same W4A4 target and accelerating draft generation by a factor of $s_D \geq 1$ gives

Table 25: Theoretical headroom for further acceleration from the current W4A4-target, 16-bit-draft configuration, with AL fixed and only drafting and verification latency included. $S_D(2)$ assumes a twofold acceleration of draft generation; it is not a measured quantization speedup.

Architecture	ρ_D	$S_T^{\max}$	$S_D(2)$	$S_D^{\max}$
EAGLE-3	43.9%	2.28×	1.28×	1.78×
DFlash	37.3%	2.68×	1.23×	1.60×

$$S_D(s_D) = \frac{T_D + T_V}{T_D/s_D + T_V} = \frac{1}{\rho_D/s_D + (1-\rho_D)}, \qquad S_D^{\max} = \lim_{s_D \to \infty} S_D(s_D) = \frac{1}{1-\rho_D}. \tag{50}$$

Here, s_D is the speedup of the entire draft-generation phase. Table 25 reports both limits and an illustrative case with $s_D = 2$.

Break-even condition. For the same target, let r denote the 16-bit-draft configuration and q the quantized-draft configuration. Define $\rho = \mathrm{AL}_q/\mathrm{AL}_r$ and $\tau = T_{\mathrm{cycle},q}/T_{\mathrm{cycle},r}$, and let TPS denote output tokens per second. Under the cycle-cost approximation,

$$\frac{\mathrm{TPS}_q}{\mathrm{TPS}_r} \approx \frac{\rho}{\tau}, \qquad \mathrm{TPS}_q > \mathrm{TPS}_r \iff \rho > \tau \tag{51}$$

Draft quantization therefore accelerates decoding only when the relative reduction in cycle cost exceeds the relative reduction in output tokens per cycle. This relation applies at batch size 1 when each configuration's cycle time and AL are aggregated from the same run over the same set of cycles.

Acceptance and remaining overhead. Let $r_{\mathrm{AL}} = \mathrm{AL}_q/\mathrm{AL}_{16}$ denote the quantized-draft AL relative to the 16-bit draft, equal to ρ in Equation 51. Let $T_O \geq 0$ be the remaining cycle latency, neither included in nor overlapping with drafting and verification, and let $f_D = T_D/(T_D + T_V + T_O)$ denote the draft's share of the full cycle. Assuming verification latency and T_O remain unchanged, the throughput ratio in terms of output tokens per cycle (AL) is

$$S_{\mathrm{decode}} = \frac{\mathrm{AL}_q/(T_D/s_D + T_V + T_O)}{\mathrm{AL}_{16}/(T_D + T_V + T_O)} = \frac{r_{\mathrm{AL}}}{f_D/s_D + (1-f_D)}. \tag{52}$$

The break-even condition is therefore

$$r_{\mathrm{AL}} > \frac{f_D}{s_D} + (1 - f_D) \tag{53}$$

If $T_O > 0$, then $f_D < \rho_D$, reducing the acceleration headroom relative to the core-latency-only model. AL and cycle latency must be aggregated from the same runtime workload. If changes in proposal count, sequence length, or scheduling alter T_V and T_O, the comparison must use the actual cycle latency rather than assume fixed costs.

Hardware opportunity depends on phase and batch. Phase-level arithmetic intensity is the ratio of total operations to total logical bytes. The corresponding roofline is an upper-bound analysis that depends on GEMM shape and batch size, not a measurement incorporating actual memory traffic, kernel launches, attention, or normalization costs. For native INT4 GEMMs, the analysis uses the corresponding backend's INT4 Tensor Core compute roof.

H.2 Native INT4 Custom Backend

DFlash, EAGLE-3, and DSpark use a native INT4 custom backend for W4A4 execution. This subsection describes how asymmetric A4 operations are executed with signed-INT4 GEMMs and how

operands are packed. The numerical parity, kernel-tuning, and SASS results reported here come from backend audits for DFlash and EAGLE-3. DSpark execution settings are listed in Table 3, and the runtime measurement protocol for all three architectures is provided in Appendix A.4.

DFlash and EAGLE-3 use asymmetric unsigned A4 activation quantization with per-token scales and integer zero points, whereas the deployed Tensor Core GEMM accepts signed s4 operands. Rather than replace the quantizer with a symmetric one, the backend uses integer recentering and zero-point correction to compute the same quantized linear operation. DSpark uses symmetric per-token A4, stored as unsigned 4-bit codes with zero point 8, and therefore uses the same kernel path. With zero point 8, the correction term in Equation 54 vanishes. The underlying integer dot-product identity is

$$\sum_k (q_k - z) w_k = \sum_k (q_k - 8) w_k + (8 - z) \sum_k w_k. \tag{54}$$

Here, q_k is an unsigned activation code, z the zero point, and w_k a signed weight code. Multi-row GEMMs use CUTLASS-based native $\mathrm{s4} \times \mathrm{s4}$ MMA with INT32 accumulation, while the single-row path retains specialized GEMV/`dp4a` dispatch.

Exact unsigned-to-signed recentering. Let $q_{ik} \in \{0, \ldots, 15\}$ denote the codes in activation row i, with zero point z_i and scale $s_{a,i}$. Let w_{jk} denote output-major weight codes, $s_{w,j}, b_j$ the scale and bias for output channel j, and K_{in} the input dimension. The original asymmetric integer dot product is

$$a_{ij} = \sum_{k=1}^{K_{\text{in}}} (q_{ik} - z_i) w_{jk} \tag{55}$$

Using signed codes $q^s_{ik} = q_{ik} - 8 \in \{-8, \ldots, 7\}$ gives

$$\begin{aligned} t_{ij} &= \sum_{k=1}^{K_{\text{in}}} q^s_{ik} w_{jk}, \qquad r_j = \sum_{k=1}^{K_{\text{in}}} w_{jk}, \\ a_{ij} &= t_{ij} + (8 - z_i) r_j. \end{aligned} \tag{56}$$

This identity is exact in the integer domain. Native s4 MMA computes t_{ij}, while r_j is precomputed and stored as row-sum metadata associated with the packed weights. Including scales and bias, the multi-row output is

$$y_{ij} = \Big(\big[t_{ij} + (8 - z_i) r_j \big] s_{w,j} \Big) s_{a,i} + b_j. \tag{57}$$

Thus, the unsigned A4 zero point is accounted for exactly rather than removed or approximated. Within the verified input-dimension range, neither integer accumulation nor correction overflows, and the relevant integer terms converted to FP32 in the epilogue are exactly representable. The floating-point scale order is preserved separately to maintain parity.

Packing and fused conversion. The existing INT4 weight packing is compatible with the packed operand layout of the CUTLASS/NVIDIA `m16n8k64.s4` MMA fragment used here. Each `uint32` stores 8 consecutive nibbles along the k dimension, with lower-order nibbles corresponding to smaller k. The backend therefore requires no separate weight repacking.

Flipping the sign bit of a 4-bit pattern and interpreting the result as a two's-complement signed value gives

$$\mathrm{s4}(q \,\mathrm{XOR}\, 8) = q - 8. \tag{58}$$

The XOR result is a bit pattern; s4 specifies its signed 4-bit interpretation. For a byte containing two nibbles, the operation is XOR with `0x88`. We fuse this conversion into the signed output path of the RMSNorm/quantization kernel, producing signed activations without an additional conversion

kernel or launch. Activation-side scaling and clipping are also applied on the fly in the same quantization kernel, without separate scaling/clipping launches or intermediate activation materialization. Buffer pools are separated by signedness to prevent aliasing between unsigned and signed execution paths.

Bit-exact scale order. The reference kernel applies the two FP32 scales in different orders for GEMM and GEMV. Letting a denote the accumulator after integer correction,

$$\begin{aligned} y_{\mathrm{GEMM}} &= ((a\, s_w)\, s_a) + b, \\ y_{\mathrm{GEMV}} &= ((a\, s_a)\, s_w) + b. \end{aligned} \tag{59}$$

These orders are equivalent in real arithmetic but can differ after FP32 rounding. Let M denote the number of activation rows. Using only the GEMM order in the initial CUTLASS epilogue produced a 1-ULP BF16 difference from the $M = 1$ reference, leading to mismatches in 6 of 27 speculative-decoding output comparisons. The final implementation selects the reference scale-multiplication order according to M and the execution path, restoring bit-exactness in those parity checks. These checks concern replacement kernels for the same quantized computation, not output identity between 16-bit and quantized drafts.

Kernel tuning. We evaluated 8 tile configurations and split-K factors $\{1, 2, 4, 8, 16\}$ at 12 deployment sites and 9 representative row counts. Only candidates that passed a bit-exactness check were considered for performance selection. No split-K configuration was selected for the evaluated deployment shapes. Tile selection yielded a median kernel-level improvement of 6.25%, and the native kernel outperformed the reference at every tested site with $M \geq 80$. These kernel microbenchmark results are distinct from the end-to-end improvements in Table 2.

Loaded-binary SASS evidence. A SASS census of the shared library loaded by the benchmark found 640 `IMMA.16864.S4.S4.SAT` instructions and no other MMA forms within the inspected binary scope. This confirms the presence of native signed-INT4 MMA in the loaded binary used for multi-row GEMMs. It does not imply that specialized single-row GEMV, 16-bit operators, attention, or all other components execute on INT4 Tensor Cores.

H.3 Roofline Accounting

Let $\mathcal{G}_\phi$ denote the GEMMs in phase ϕ. For each GEMM, let F_g be its operation count and B_g its logical bytes, calculated from tensor sizes and precision. The phase-level arithmetic intensity and theoretical roofline are

$$I_\phi = \frac{\sum_{g \in \mathcal{G}_\phi} F_g}{\sum_{g \in \mathcal{G}_\phi} B_g}. \tag{60}$$

$$P_\phi^{\mathrm{roof}} = \min(P_{\mathrm{peak}},\, \mathrm{BW}_{\mathrm{peak}} I_\phi)\,. \tag{61}$$

The roofline estimates an upper bound as the smaller of the compute peak at the relevant precision and the throughput permitted by peak memory bandwidth at the modeled arithmetic intensity. The logical bytes B_g in Equation 60 are computed from each GEMM's operand and output sizes. This accounting does not measure cache reuse, device-memory transactions, or inter-kernel traffic, and excludes attention, quantization, launch, and synchronization costs.

Figure 3 uses DFlash / Llama-3.1-8B-Instruct on an RTX 4090 at batch size 16, prompt length 512, and verification budget 10. Native INT4 GEMMs and 16-bit operations use their respective compute roofs. In this modeled configuration, W4A16 drafting and verification are limited by the FP16/BF16 compute ceiling, indicating an opportunity to exploit low-bit arithmetic in addition to reducing weight traffic. This logical-byte roofline does not represent measured traffic or latency; actual speedup also depends on GEMM shape, batch size, backend, and non-GEMM costs.

H.4 Batch Scaling and Runtime Interpretation

Cycle cost and accepted output. Let n_j denote the number of tokens actually emitted in cycle j of a sequence. The mean time per output token over the same evaluation interval is

$$T_{\mathrm{token}} = \frac{\sum_j (T_{\mathrm{draft},j} + T_{\mathrm{verification},j} + T_{\mathrm{other},j})}{\sum_j n_j} = \frac{\overline{T}_{\mathrm{cycle}}}{\mathrm{AL}}, \qquad \mathrm{AL} = \frac{\sum_j n_j}{C}. \tag{62}$$

Here, C is the number of cycles in that interval; this is the aggregated form of Equation 1. In batched execution, emitted tokens and wall time are aggregated over all active sequences. Per-sequence AL therefore cannot be substituted directly into a formula using batch wall time.

At $B = 1$, FLASHLight achieves 442.1, 459.7, and 663.6 tokens/s on DFlash, EAGLE-3, and DSpark, respectively, exceeding the corresponding 16-bit-draft throughput by 29.8%, 26.3%, and 28.5%. For DFlash and EAGLE-3, seed-1 and seed-2 checkpoints were each measured once with the same protocol; their throughputs are within 2% of the seed-0 mean at every batch size. Only seed 0 was measured for DSpark. Acceptance recovery is reported as Avg. AL and recovery in Tables 10 and 23. Because AL evaluation and runtime measurement use separate protocols, we do not combine AL and latency from different runs.

EAGLE-3 throughput in Table 24 saturates near 1,000 tokens/s from $B = 8$ onward. At $B = 16$, FLASHLight's mean throughput is 0.992 times that of the 16-bit draft; the across-repeat standard deviations are 18.8 and 8.4 tokens/s, respectively. Isolating the contributions of autoregressive draft steps, attention, and synchronization would require kernel-level profiling. Likewise, FLASHLight's 1.4% lower throughput than W4A16 GPTQ on EAGLE-3 at $B = 1$ (Appendix G) cannot be attributed to activation precision alone, because the configurations differ in both quantization method and AL. The DFlash 16-bit draft runs out of memory (OOM) at $B = 16$, so no paired speedup is reported for that configuration.

H.5 Latency, TPOT, and Peak Memory

The latency, TPOT, and peak memory in Table 26 are recorded in the same runs as the throughput measurements in Tables 2 and 24, using the same execution settings. P50 and P95 latency are the within-run percentiles of request latency from admission to the last token. For each request, time per output token (TPOT) is (last-token time − first-token time)/(number of committed new tokens -1). Requests emitting a single token are excluded, and the reported run-level TPOT is the P50 over the remaining requests. Peak memory is `torch.cuda.max_memory_allocated` over the measurement interval.

We compute each metric within a run and then average the run-level values: 3 repeats for both configurations of DFlash and EAGLE-3, and 8 repeats for the 16-bit draft and 3 for FLASHLight on DSpark. These metrics are measured directly, not inferred from throughput.

Table 26: Measured latency, TPOT, and peak memory. Arrows indicate 16-bit draft → FLASHLight. Values are means of run-level metrics from the same runs as Tables 2 and 24. Peak memory is peak allocated memory, measured in GiB.

Architecture	Batch	P50 (s)	P95 (s)	TPOT (ms)	Peak memory (GiB)
DFlash	1	0.436 → 0.338	0.984 → 0.759	2.72 → 2.05	18.25 → 15.22
	8	0.790 → 0.685	1.533 → 1.356	4.60 → 4.03	20.33 → 17.35
EAGLE-3	1	0.570 → 0.454	0.812 → 0.643	2.51 → 1.96	11.94 → 11.19
	16	3.055 → 2.957	4.526 → 4.386	13.63 → 13.77	16.63 → 15.89
DSpark	1	0.437 → 0.341	0.716 → 0.550	1.65 → 1.25	6.31 → 4.99
	16	1.139 → 1.025	1.754 → 1.583	4.51 → 4.04	11.29 → 9.97

For EAGLE-3 at $B = 16$, FLASHLight lowers P50/P95 request latency by approximately 3%, but TPOT increases from 13.63 ms to 13.77 ms and throughput is 0.992 times that of the 16-bit draft. We therefore do not interpret these lower latency percentiles as evidence of decoding-throughput acceleration at this batch size.

H.6 Runtime Variability

Table 27 reports the across-repeat standard deviation for each entry in Tables 2 and 24. DSpark at $B = 16$ shows particularly large run-to-run variability across all configurations: 2840.6–3350.3 tokens/s for the 16-bit draft and 3235.8–3620.9 tokens/s for FLASHLight. Throughput ratios at this batch size should be interpreted in light of that variability.

Table 27: Across-repeat standard deviation of throughput (tokens/s) for Tables 2 and 24. All DFlash and EAGLE-3 configurations use 3 repeats; DSpark uses 8/8/6/3 repeats in column order. The DFlash 16-bit draft runs out of memory (OOM) at $B = 16$.

Architecture	Batch	16-bit draft	W4A16 GPTQ	Aligned PTQ	FLASHLight
DFlash	1	0.4	0.4	3.2	3.9
	2	1.1	0.9	7.1	6.6
	4	1.8	1.5	14.2	24.4
	8	10.4	2.7	20.2	22.4
	16	OOM	9.8	52.8	27.4
EAGLE-3	1	0.8	4.8	2.1	4.0
	2	2.9	8.7	5.3	8.1
	4	7.1	16.1	9.6	13.8
	8	9.0	16.2	13.9	19.8
	16	8.4	19.3	15.4	18.8
DSpark	1	3.1	7.9	4.2	2.3
	2	5.3	15.0	9.6	6.4
	4	34.6	55.9	29.9	68.2
	8	102.5	40.9	74.3	90.1
	16	203.2	197.4	178.0	220.7

H.7 Quantization Granularity and Mixed-Precision Alternatives

Group-wise scales. Independent source or group scales can mitigate shared-range errors, but change scale handling along the GEMM reduction dimension. Omitting zero points and bias, per-token activation and per-output-channel weight scales can be applied after integer accumulation, whereas independent group scales generally remain inside the group reduction:

$$\widehat{y}^{\text{token}}_{t,o} = s^A_t s^W_o \sum_j q^A_{t,j} q^W_{j,o},$$

$$\widehat{y}^{\text{group}}_{t,o} = \sum_g s^A_{t,g} s^W_{g,o} \sum_{j \in \mathcal{G}_g} q^A_{t,j} q^W_{j,o}.$$

Here, t, o, j index tokens, output channels, and input channels, and $\mathcal{G}_g$ is a group of input channels. QServe identifies CUDA-core partial-sum dequantization in the GEMM main loop as a bottleneck in the group-wise W4A4 implementations it analyzes (Lin et al., 2025). This does not preclude efficient group-wise kernels or imply a universal slowdown.

Fine-grained mixed precision. Assigning different precisions within a projection likewise entails precision-aware execution. MixLLM uses two-step dequantization and software pipelining (Zheng et al., 2026), while COMET develops mixed-precision layouts, W4Ax kernels, and SM scheduling (Liu et al., 2025a). These systems demonstrate that efficient mixed precision is possible through algorithm–system co-design.

FLASHLight's design scope. Our source scales are fixed across tokens, with reciprocal scaling folded into the weights, rather than independent runtime quantization scales for each source. Equation 4 therefore retains per-token A4 and per-output-channel W4 without group-dependent partial-sum scaling or mixed-precision partitioning of the projection. The usual scale and zero-point corrections remain (Appendix H.2). Keeping the entire projection at INT8 or 16-bit is a distinct operator-level alternative; it does not inherently require fine-grained mixed-precision kernels. Our contribution is acceptance recovery within the specified W4A4 execution structure, not universal performance superiority over other granularities or precisions.

I Draft Retraining as an Alternative

Draft retraining absorbs basis mismatch by adapting the draft weights to transformed target features. Unlike a function-preserving reparameterization, it relearns draft predictions rather than preserving the original computation. However, the constraint imposed by a shared A4 grid remains. Suppose that m source features are concatenated as

$$\mathbf{H} = [\mathbf{H}_1 \mid \cdots \mid \mathbf{H}_m] \tag{63}$$

Under the asymmetric per-token A4 quantization used by DFlash and EAGLE-3, all source blocks of a token share the same lower and upper bounds, scale, and zero point. For a token t with overall range $[\ell_t, u_t]$, the 4-bit grid spacing is

$$\Delta_t = \frac{u_t - \ell_t}{2^4 - 1}, \qquad \ell_t = \min_i \min \mathbf{H}_{i,t}, \qquad u_t = \max_i \max \mathbf{H}_{i,t} \tag{64}$$

Here, $\mathbf{H}_{i,t}$ denotes the transformed source block presented to the quantizer, and the expression describes the min–max grid before clipping. DSpark's symmetric per-token A4 quantization similarly shares a single $\Delta_t = \max_i \max |\mathbf{H}_{i,t}|/7$ across sources, without a zero-point offset. Retraining does not replace this shared grid with separate per-source grids. Adapting the draft to the transformed basis therefore does not guarantee that the resolution loss of smaller-range sources is eliminated; low-bit acceptance must still be evaluated for the retrained draft.

Table 28 compares optimization costs based on RTX 4090 measurements. FLASHLight times sum measured calibration and rotation-optimization costs, whereas retraining totals are extrapolated from measured step times and the corresponding training schedules. Basis alignment in Aligned PTQ fuses given rotations into the weights without additional iterative optimization, although initial PTQ calibration and offline transformations still incur cost. The costs of obtaining the initial draft basis $R_{D,0}$ and the target rotation are shared with Aligned PTQ and excluded from the FLASHLight times. The reported FLASHLight time covers one range search at $R_{D,0}$ and 400-step QSD rotation optimization for one seed.

For DFlash, range calibration includes capturing feature-projection inputs for 178 prompts (401.3 s), evaluating 160 scaling candidates (376.7 s), and evaluating 10 clipping candidates (44.6 s). The deployment choice $S^\star$ is one of these 160 candidates and comes from a $2^{\nu/4}$ lattice search at the same basis on a separate 96-prompt pool. We therefore also include feature-projection input capture for that pool (3,228.0 s) and the lattice search itself (530.6 s). For EAGLE-3, range calibration consists of α calibration (204.6 s) and shared-clipping selection (2.8 min, from execution logs).

The per-seed QSD training times are 2,240.9/2,256.5/2,231.4 s for DFlash and 3,461.8/3,550.0/3,512.0 s for EAGLE-3. These wall-clock times include periodic validation (5 times per run), which is not used for checkpoint selection. For DFlash, validation accounts for approximately half of the training time (1,117.5 s averaged over seeds). Thus, within the stated accounting scope, the reported DFlash time includes monitoring overhead that is not required for checkpoint selection. Under the same accounting convention, obtaining all three-seed results requires one range search and three training runs, totaling 3.14 h for DFlash and 3.03 h for EAGLE-3.

Parenthesized times additionally include offline teacher-data or calibration-corpus construction. For DFlash, constructing quantized-verifier teacher outputs for the 192 rotation-training rows takes 1.76 GPU-h. Constructing outputs for the 88 validation rows used to record learning curves takes a further 0.83 GPU-h, which is excluded from these totals. EAGLE-3 instead executes the target during training without a teacher cache; constructing its 112-row calibration corpus adds 0.12 GPU-h (from execution logs).

The GPU counts and optimization objectives differ, so these wall-clock ratios are not a comparison of the minimum cost required to reach the same acceptance.

We did not measure retraining cost for DSpark. Its FLASHLight procedure requires 94 s for the S, c search at $R_{D,0}$ and 35.1/34.7/35.1 min of QSD training for the three seeds. The three training runs total 1.75 GPU-h, with peak allocated memory of 16.39–16.44 GiB. Offline data construction adds 1.41 GPU-h for the 192 rotation-training teacher rows and 0.65 GPU-h for the rows used by the 96 range-search prompts. The 88 validation rows used to record learning curves require a further 0.62 GPU-h.

Table 28: Optimization costs of draft retraining and FLASHLight, based on RTX 4090 measurements. Retraining updates all draft weights on 8 GPUs; its total wall-clock time is extrapolated from the measured step time and training-step count. For EAGLE-3, the range spans the code's 800K-step learning-rate horizon and the 2.66M steps corresponding to 40 epochs. FLASHLight performs range calibration and 400-step QSD rotation optimization for one seed on a single GPU. Parenthesized times additionally include construction of rotation-training teacher data for DFlash or the calibration corpus for EAGLE-3, excluding validation-teacher data used to record learning curves. Initial draft-basis and target-rotation construction are excluded from both FLASHLight times as costs shared with Aligned PTQ. Retraining parameter counts refer to the draft weights being optimized. For FLASHLight, Optimized params counts the dense entries of the rotation generators B_r; parentheses give the free parameters of the skew-symmetric generators in Equation 41. Time ratio is the retraining time divided by the corresponding FLASHLight time; parenthesized ratios use the parenthesized FLASHLight time.

Architecture	Method	GPUs	Optimized params	Time (h)	Time ratio
DFlash	Retraining	8	1048.6M	134.2	71× (37×)
	FLASHLight	1	16.9M (8.4M)	1.90 (3.66)	1×
EAGLE-3	Retraining	8	424.7M	316.7–1052.9	294–977× (264–879×)
	FLASHLight	1	16.8M (8.4M)	1.08 (1.20)	1×