CSRe: Training-Free Diffusion Transformer Acceleration via Component-Wise Selective Refresh

Anonymous authors
Paper under double-blind review

Abstract

Diffusion Transformers (DiTs) repeatedly evaluate attention and MLP modules across denoising steps, despite temporal redundancy in their internal features. Feature caching reduces this cost, but fine-grained reuse must account for both refresh allocation and execution overhead. We introduce CSRe (Component-Wise Selective Refresh), a training-free inference framework that selectively recomputes output-projected attention-head and MLP hidden-channel contributions. Full anchor steps refresh the caches and measure changes between anchors; these lagged measurements guide component selection for subsequent selective steps. Selected contributions update the cached module outputs through their latest-cache deltas. A shared compute budget coordinates attention and MLP refresh, with the remaining MLP budget distributed across blocks according to anchor-measured hidden-activation changes. Selected channels are executed using hardware-aligned dense submatrices. We evaluate CSRe on DiT-XL/2, PixArt-α, and FLUX.1-dev. On an RTX 5090D, CSRe maintains competitive generation quality while achieving denoiser speedups of $2.292\times$, $1.963\times$, and $1.728\times$ on DiT-XL/2, PixArt-α, and FLUX.1-dev, respectively.

1 Introduction

Diffusion models generate high-fidelity images through iterative denoising (Ho et al., 2020; Song et al., 2021; Rombach et al., 2022). Diffusion Transformers (DiTs) replace the convolutional denoiser with a transformer backbone composed primarily of attention and MLP modules (Peebles & Xie, 2023; Chen et al., 2024a). Standard sampling performs dense evaluation of all transformer blocks at every step, so an S-step sampler with L blocks requires SL block evaluations per image, excluding any additional classifier-free-guidance branch. Few-step distillation and consistency training reduce the number of denoiser evaluations by training models for one- or few-step generation (Salimans & Ho, 2022; Song et al., 2023). Model compression instead reduces the cost of the model's operations; Q-DiT applies post-training quantization to Diffusion Transformers (Chen et al., 2025a). We focus on training-free feature reuse, which leaves the pretrained parameters and sampler intact while reducing repeated backbone computation during inference.

Features at adjacent denoising steps exhibit temporal correlation. Existing caching methods exploit this temporal redundancy at several granularities: they reuse groups of U-Net features or entire DiT modules or blocks (Ma et al., 2024; Selvaraju et al., 2024; Chen et al., 2024b), select token positions for recomputation (Zou et al., 2025; 2026), or update selected activation columns (Silveria et al., 2025). For head- and channel-level refresh, choosing components is only part of the allocation problem: under a fixed compute budget, additional attention refresh leaves less capacity for MLP updates. We study this trade-off within one allocation framework, rather than treating the two module families as independently budgeted paths.

Multi-head attention outputs decompose additively into output-projected head contributions, while two-layer MLP outputs decompose into hidden-channel contributions through the second projection. Along dense denoising trajectories, temporal changes are non-uniform across components but are not dominated by a single head or channel. Reusing an unselected component retains its cached contribution and omits only its subsequent temporal increment.

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

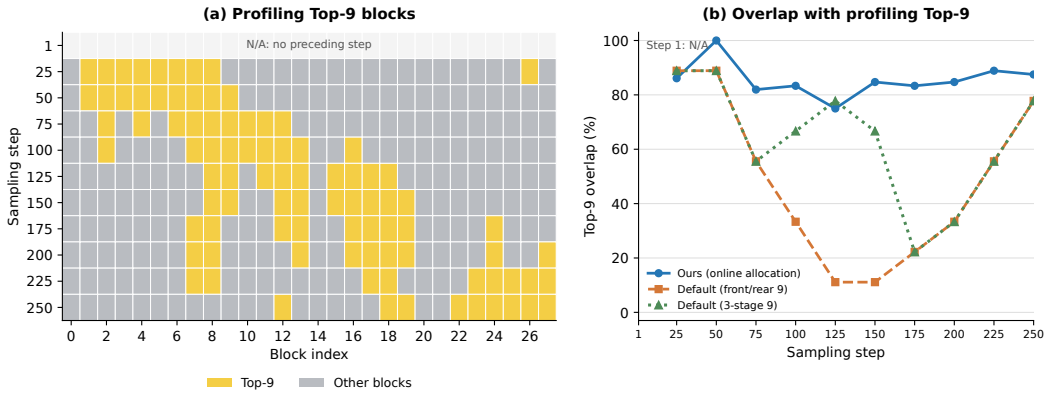


Figure 1: **(a)** Dense-profile Top-9 high-change blocks over depth and denoising time. **(b)** Top-9 overlap of online allocation and two- and three-stage fixed-depth heuristics with the dense profile.

Refresh demand also varies across network depth and sampling stages. Δ-DiT uses stage-dependent block ranges to account for differences in block behavior along denoising (Chen et al., 2024b). Our dense-trajectory measurements (Figure 1) likewise show that MLP hidden-activation change is non-uniform across blocks and that its distribution shifts over time. This motivates redistributing a fixed channel budget, rather than assigning each block the same width throughout sampling.

We introduce CSRe, a training-free framework that alternates *anchor steps* with *selective steps*. Anchor steps perform dense evaluation of all transformer blocks at the current accelerated-trajectory state and refresh the latest-component caches. Their *anchor-to-anchor change* is the difference between module states at the current and preceding completed anchors on that trajectory. This lagged signal ranks the components used in the following selective interval. Selective steps recompute the selected attention heads and add their projected deltas to the cached attention output. MLP channels are updated analogously through their second-projection contributions. Periodic anchors refresh stale contributions. For each model, schedule, and compute target, the allocation protocol uses an independent matched-compute calibration sweep to fix the head count, accounting for the MLP capacity displaced by each additional head. During inference, head identities vary by sample and block based on the preceding anchor-to-anchor change. The remaining budget is assigned to MLP hidden channels, with batch-shared block widths determined by historical hidden-activation changes and sample-specific channel identities selected within each block. More channels are assigned to higher-change blocks while preserving the aggregate width budget. Head and channel scores are not pooled because their scales and costs differ. Hardware-aligned widths retain dense execution. The same formulation covers self-attention, cross-attention, and joint text–image attention.

Our contributions are threefold:

- **We decompose attention and MLP outputs into cacheable projected head and channel contributions and update selected components through latest-cache deltas.** Selected contributions are refreshed by adding their increments relative to their latest cached values to the cached module output. This formulation defines a structured operating point between reusing an entire module output and recomputing it densely.
- **We propose a two-stage allocation protocol that coordinates attention-head and MLP-channel refresh under a shared compute target.** A matched-compute calibration sweep fixes the attention-head count before inference while accounting for the MLP capacity displaced by each additional head; at runtime, head identities are selected dynamically from lagged anchor-to-anchor measurements.
- **We distribute the residual MLP-channel budget across transformer blocks based on anchor-measured hidden-activation changes, allowing per-block widths to adapt to time-varying refresh demand.** This allocation is realized with compact dense submatrices at hardware-aligned channel widths, mapping fine-grained channel selection to dense GPU execution rather than requiring irregular sparse kernels.

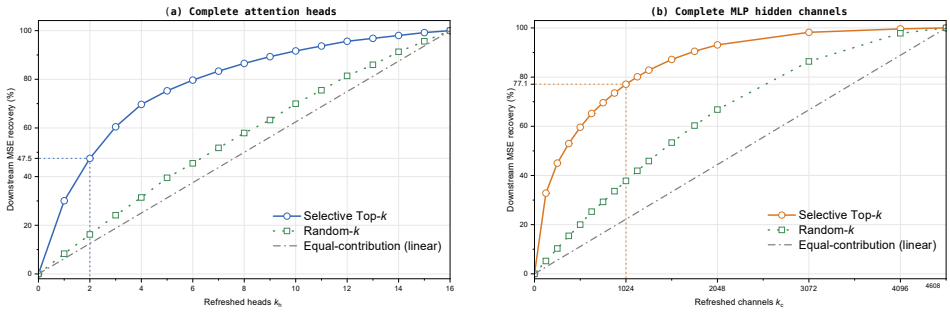


Figure 2: One-transition lagged projected-change rankings recover more downstream error than random subsets. Zero and 100% denote cache-only and dense updates.

2 Related Work

2.1 Diffusion Transformers

DiT replaces convolutional denoisers with transformer blocks dominated by attention and MLP computation (Peebles & Xie, 2023). PixArt-α adds cross-attention for text conditioning, whereas FLUX.1-dev combines double-stream joint text–image attention with subsequent single-stream blocks (Chen et al., 2024a; Black Forest Labs, 2024). These architectural differences determine which attention and MLP components can be cached or selectively refreshed.

Orthogonal approaches reduce sampling steps through solver design or additional training (Song et al., 2021; Lu et al., 2022; 2025; Salimans & Ho, 2022; Song et al., 2023); we focus on training-free methods that reduce per-step backbone computation.

2.2 Training-Free Acceleration

Training-free caching reuses intermediate features across nearby denoising steps. DeepCache reuses groups of high-level U-Net features (Ma et al., 2024). For transformer denoisers, FORA periodically reuses complete attention and MLP outputs (Selvaraju et al., 2024), PAB assigns module-specific broadcast ranges to attention outputs (Zhao et al., 2025), and Δ-DiT uses stage-dependent block ranges rather than within-block components (Chen et al., 2024b). TeaCache uses changes in timestep-modulated inputs to trigger transformer-feature reuse (Liu et al., 2025a), and Scaling-Cache combines dynamically chosen reuse intervals with scaled feature differences (Gu et al., 2026). ECAD searches binary caching decisions over timesteps, blocks, and cacheable modules (Aggarwal et al., 2026). Its offline calibration repeatedly generates samples for candidate schedules, adding setup cost when configurations require re-optimization or each candidate is expensive to evaluate, as may occur in video generation; demonstrated schedule transfer can reduce this cost. ProCache combines non-uniform activation schedules with selective computation over deep blocks and selected tokens (Cao et al., 2026), whereas DPCache selects key timesteps through global path planning based on path-dependent error estimates (Cui et al., 2026).

Feature-correction and forecasting methods differ in how they estimate a missing temporal update. TaylorSeer uses Taylor-series extrapolation, and HiCache replaces that basis with scaled Hermite polynomials (Liu et al., 2025b; Feng et al., 2026). Increment-Calibrated Caching instead derives a low-rank correction from the pretrained model and uses channel-aware SVD to handle activation outliers (Chen et al., 2025b). SpeCa adds verification to feature forecasting and adapts computation to individual samples (Liu et al., 2025c). Relational Feature Caching estimates feature changes and recomputation needs from module input–output relations (Son et al., 2026).

DiTFastAttn reduces attention computation with windowed attention and shares attention results across timesteps and classifier-free-guidance branches (Yuan et al., 2024). TokenCache and ToCa choose token positions within a module for recomputation; ToCa further varies cache ratios with

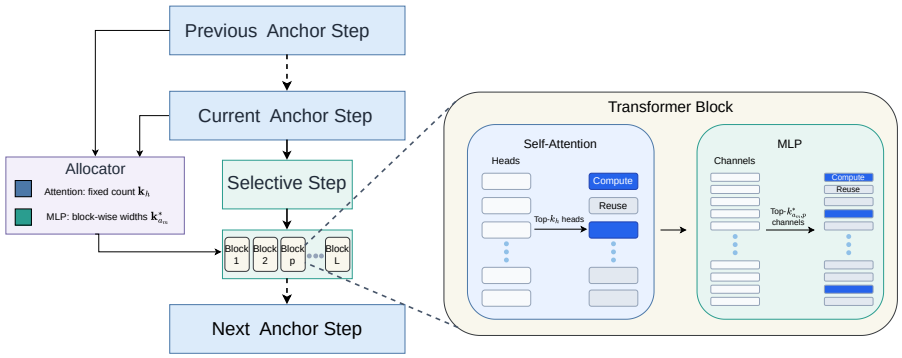


Figure 3: Overview of CSRe. Completed-anchor changes select fixed-count Top-k_h attention heads and allocate block-wise MLP widths $k^*_{a_m,p}$ for the next selective interval. Here p indexes the illustrated block and its MLP refresh module; unselected components reuse cached contributions.

module type, depth, and denoising time (Lou et al., 2024; Zou et al., 2025). DuCa alternates two token-reuse regimes to control accumulated error (Zou et al., 2026). ProCache also recomputes selected tokens within cached segments, but its within-block selection is token-wise rather than head- or channel-wise (Cao et al., 2026). Chipmunk selects changing activation columns and executes them with sparse kernels (Silveria et al., 2025), while SVD-Cache assigns different temporal treatments to principal and residual feature subspaces (Chen et al., 2026). HyCa models heterogeneous dynamics across output feature dimensions and assigns them different ODE-inspired temporal solvers (Zheng et al., 2026).

These methods span several distinct reuse paths. Our experiments select one representative from each path for which aligned implementations are available: FORA represents periodic reuse of complete module outputs, ToCa represents online token-wise recomputation, and ECAD represents offline search over time-, block-, and module-level caching decisions. This set therefore compares CSRe with coarse periodic reuse, fine-grained online selection, and searched non-uniform schedules.

3 METHOD

Figure 3 provides an overview of CSRe. Changes between successive completed anchors select attention-head identities under a shared fixed count and allocate block-wise MLP widths for the next selective interval. At selective steps, CSRe recomputes only the selected components and applies their projected deltas to the cached module outputs.

3.1 PRELIMINARIES

Let $s \in \{0, \dots, S-1\}$ index chronological sampling steps, $\boldsymbol{x}_s$ be the sampler latent, and $\boldsymbol{u}_{s,\ell}$ an activation along the denoiser trajectory. For an optional condition $\boldsymbol{\eta}$, the denoiser prediction advances $\boldsymbol{x}_s$ through the sampler rule Ψ_s; the complete state transition is given in Appendix A.1. Within the denoiser, $\ell \in \{0, \dots, Q-1\}$ indexes one of Q residual submodules, whose update is

$$\boldsymbol{u}_{s,\ell+1} = \boldsymbol{u}_{s,\ell} + \boldsymbol{g}_{s,\ell} \odot f_\ell(\mathrm{Norm}_{s,\ell}(\boldsymbol{u}_{s,\ell}), \boldsymbol{\eta}) \,. \tag{1}$$

Here f_ℓ is the attention or MLP module, whose output is cached before the token-broadcast residual gate $\boldsymbol{g}_{s,\ell}$ ($\mathbf{1}$ when absent). This architectural gate is independent of channel selection; normalization, conditioning, gating, and residual additions remain dense (Appendix A.2). For either an attention or MLP module, suppressing module and architecture-family indices, the output admits the following additive decomposition:

$$\boldsymbol{y}_s = \boldsymbol{b} + \sum_{j=1}^{J} \boldsymbol{c}_{s,j}. \tag{2}$$

For attention, j denotes a head and $J = H$; for an MLP, j denotes a hidden channel and $J = M$. The raw component state $\boldsymbol{r}_{s,j}$ is used for scoring, whereas $\boldsymbol{c}_{s,j}$ is its output-projected contribution

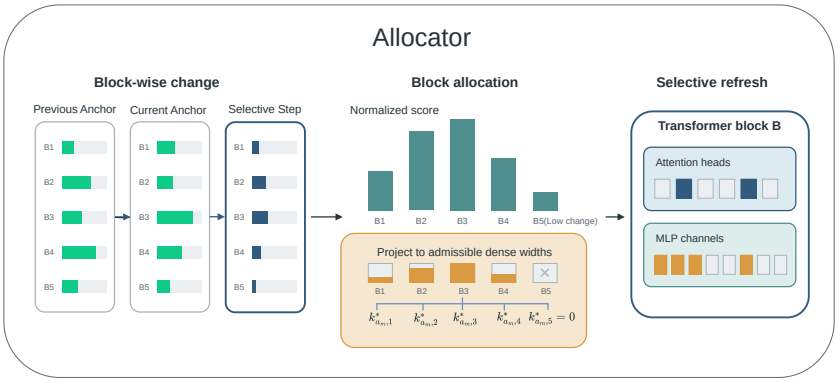


Figure 4: Anchor-guided component selection and block-wise MLP allocation in CSRe.

to the module output. The fixed output bias $\boldsymbol{b}$ is added once. Appendix A.1 gives the attention and MLP expansions, tensor dimensions, projections, and bias handling.

Let $\widetilde{\boldsymbol{c}}_{s,j}$ be the latest cached contribution upon entering step s, and let $\boldsymbol{y}_s^{\text{cache}} = \boldsymbol{b} + \sum_j \widetilde{\boldsymbol{c}}_{s,j}$. Recomputing the selected components updates the module output and its cache. Here $\mathcal{I}_s$ contains all components at an anchor and the scheduled subset at a selective step:

$$\widehat{\boldsymbol{y}}_s = \boldsymbol{y}_{s+1}^{\text{cache}} = \boldsymbol{b} + \sum_j \widetilde{\boldsymbol{c}}_{s+1,j} = \boldsymbol{y}_s^{\text{cache}} + \sum_{j \in \mathcal{I}_s} (\boldsymbol{c}_{s,j} - \widetilde{\boldsymbol{c}}_{s,j}). \tag{3}$$

Selected components advance to their newly computed states; all others retain their latest cached values. Appendix A.1 gives the component-wise cache transition and the corresponding local omitted-delta identity.

3.2 Change-Guided Component-Wise Refresh

Prior work reports coarse depth-dependent specialization in DiTs: front blocks are more closely associated with image outlines, whereas rear blocks affect local details, motivating stage-dependent front/rear block caching in Δ-DiT (Chen et al., 2024b). This suggests that refresh demand may depend on both depth and denoising stage, but not that a fixed depth partition tracks every transition.

We examine this variation on 5K dense 250-step DiT-XL/2 trajectories (Figure 1; Appendix A.3). For each block and adjacent sampling step, block change sums the token-mean absolute MLP hidden change over 4608 channels and then averages over samples; it measures temporal movement, not semantic importance. High-change blocks shift over depth and time. CSRe's online allocation ranking overlaps the dense-profile Top-9 by 85.6%, versus 47.8% for the tested front/rear-nine heuristic. A three-stage heuristic that uses blocks 0–8, 9–17, and 19–27 over $[0, 80)$, $[80, 160)$, and $[160, 250]$, respectively, reaches 63.3%. We compare nine high-change blocks to assess how well coarse, fixed front/rear or front/middle/rear stage partitions track the evolving depth profile. The higher overlap of online allocation shows that adapting the ranking captures this temporal variation more closely than the tested static partitions. Nine is an intermediate comparison size rather than a required refresh count: very small sets emphasize only a few extreme blocks, whereas sets approaching the full depth make overlap less discriminative.

Controlled recovery. We next ask whether a completed-transition change contains useful information for ranking components at the next transition (Appendix A.4). Figure 2 shows that two ranked heads recover 47.5% of the zero-update downstream error, versus $16.2\%\pm0.4\%$ for random selection; 1024 ranked channels recover 77.1%, versus $37.8\%\pm0.1\%$.

3.3 CSRe

Offline configuration. Before evaluation, we fix the anchor schedule, compute target, admissible MLP width set, and shared attention-head count k_h. We screen k_h with dense MLP refresh on a

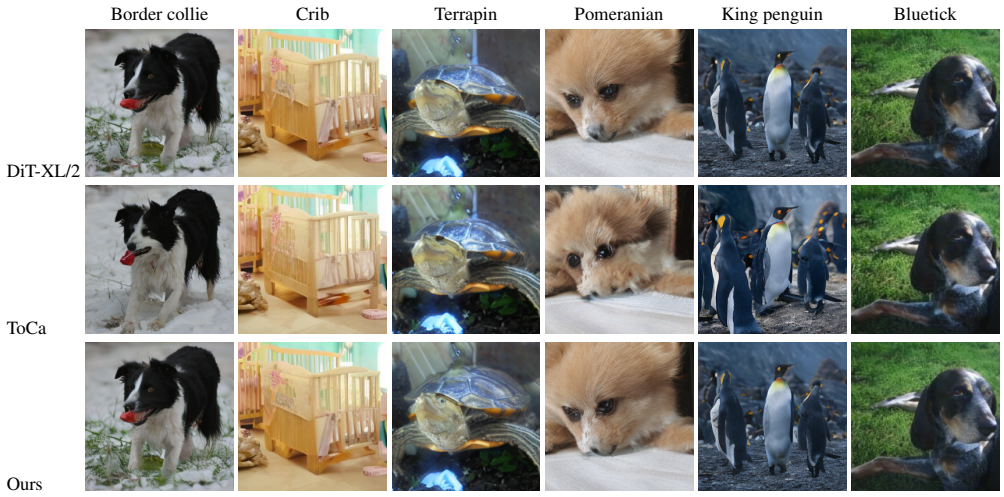


Figure 5: Visualization examples on DiT-XL/2.

calibration split, then refine it under the final compute target and MLP budget (Appendix A.5). It remains fixed across all attention refresh modules throughout inference; only selected head identities vary online. The MLP refresh ratio ρ_c uses the budget remaining after anchor steps, dense operations, and attention refresh. Head and channel scores remain separate because their scales and costs differ.

Runtime protocol. An anchor evaluates the full backbone at the current CSRe-trajectory state, refreshes every component cache, and records the scoring snapshots; it does not retroactively alter earlier sampler states. For successive completed anchors $a_{m-1} < a_m$, the *anchor-to-anchor change* is the difference between corresponding module states observed at these two anchors on the same accelerated trajectory. It determines component identities and batch-shared MLP widths for (a_m, a_{m+1}). Only completed anchors contribute to the scores, and the scoring snapshots are stored separately from the latest caches in Equation (3).

At a selective step, CSRe evaluates only the chosen heads and channels and uses their latest-cache deltas to update the module outputs. All other state in Equation (8) is carried forward. Under the standard initialization, the first two steps serve as anchors, which initialize the caches and provide the first anchor-to-anchor change. Model-specific schedules, including the early full-cache reuse variants, are given in Appendix A.6.

Figure 4 summarizes the resulting runtime procedure: completed-anchor changes determine component identities and block-wise MLP widths for the next selective interval, after which selected component deltas update the cached module output.

Completed-anchor component scores. For sample i, refresh module p, and component j, let $\bar{\boldsymbol{r}}_{i,a_m,p,j}$ denote the raw state recorded at completed anchor a_m. CSRe uses the mean of the absolute change between two successive completed anchors:

$$q_{i,m,p,j} = \mathrm{mean}_{n,d} \left| \bar{\boldsymbol{r}}_{i,a_m,p,j} - \bar{\boldsymbol{r}}_{i,a_{m-1},p,j} \right|. \tag{4}$$

Here n indexes tokens and d indexes features within a head; for an MLP channel, the nonexistent feature dimension d is not averaged. Head identities are selected by the Top-k_h head scores, and channel identities by the channel scores within each MLP module. Only the two completed anchors enter these decisions, which are held over (a_m, a_{m+1}). Scoring snapshots are separate from the latest-component caches used in (3). The family-specific forms are given in Appendix A.1; the reported FLUX score is specified in Appendix A.6.

Attention heads. Head scores use the raw output $\boldsymbol{r}^{\mathrm{attn}}$, while selected cache updates use the projected contribution $\boldsymbol{c}^{\mathrm{attn}}$. Head identities vary by sample, block, attention module, and anchor interval; k_h remains fixed. Under classifier-free guidance, each conditional mask is shared only with its paired unconditional sample.

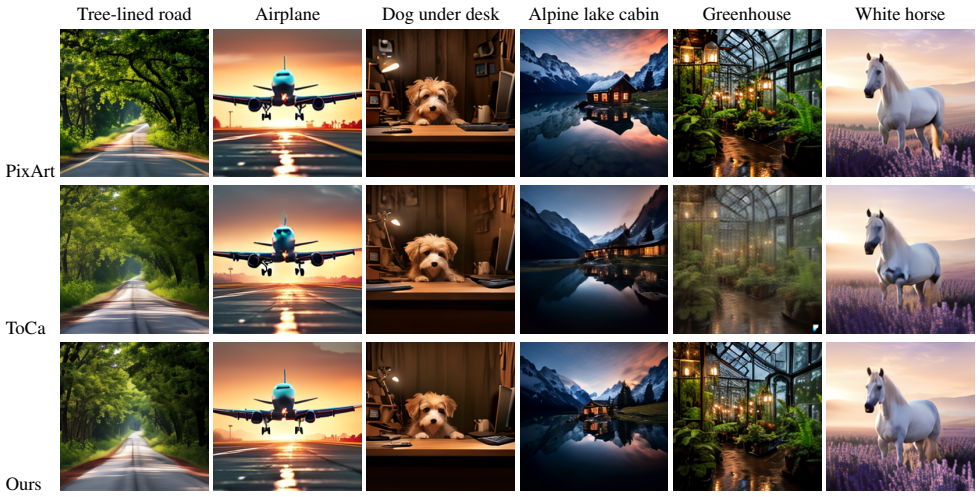


Figure 6: Visualization examples on PixArt-α.

MLP channels and block-wise allocation. Let $p \in \{1, \ldots, P\}$ index MLP refresh modules, with $P = L$ for the L DiT or PixArt blocks. At completed anchor a_m, $k^*_{a_m,p}$ is module p's batch-shared final admissible width after projection; it is fixed over (a_m, a_{m+1}), while selected channel identities are sample-specific. For each sample, MLP module, and hidden channel, CSRe measures the mean of the absolute change of the raw hidden activation between two successive completed anchors. Channel identities are selected within each module from these lagged scores; updates use their projected contributions. To allocate widths across modules, we aggregate the channel scores within each module, normalize the resulting module-level changes across blocks, and average them over the conditional batch. The resulting block-change score measures observed hidden-activation movement on the current accelerated trajectory.

Given the compute target, CSRe first subtracts anchor, dense-operation, and attention-refresh costs to obtain the MLP budget for the following selective interval. It then allocates block-wise width targets in proportion to the completed anchor-to-anchor change scores and projects them onto the admissible dense-width set while exactly preserving the total width budget. If all block-level change scores are zero, the budget is divided uniformly before projection. This unweighted width constraint applies to equal-width, equal-cost modules; heterogeneous branch costs are treated in Appendix A.2. Appendix A.1 details the normalization procedure, stage factors, feasible-budget construction, projection objective, and deterministic tie-breaking rule. Figure 4 illustrates this procedure. At each anchor, the selected columns of the first projection and rows of the second projection are packed into compact dense matrices. Subsequent selective steps reuse these matrices without repacking.

Architecture and cost scope. DiT uses self-attention and MLP refresh; PixArt maintains separate self- and cross-attention caches. FLUX maps its double- and single-stream branches to the same additive update. These architecture mappings are detailed in Appendix A.2. Admissible width tiers, dense break-even points, execution backends, and operator-level costs are implementation details reported in Appendix A.7. Accordingly, we report analytical FLOPs and latency separately.

4 Experiments

4.1 Experimental settings

Models. We evaluate DiT-XL/2 on ImageNet, PixArt-α-XL/2 on MS-COCO, and FLUX.1-dev, all at 256×256. Baselines include dense inference, periodic reuse, ToCa (Zou et al., 2025), and ECAD where available. CSRe requires no retraining; operating points are fixed on disjoint calibration data. Appendix A.8 specifies configurations and data splits.

Table 1: ImageNet results on DiT-XL/2 under ADM 50K evaluation. Latency is evaluated at batch size 18.

Method	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑	TFLOPs↓	Latency (s)↓
DiT-XL/2 (CFG= 1.5)	2.325	4.731	275.115	0.8274	0.5764	118.68 (1.00×)	2.243(1.000×)
FORA (N=3)	2.844	6.172	252.529	0.8049	0.5867	39.95 (2.97×)	0.877(2.558×)
FORA (N=2.8)	2.749	5.973	255.129	0.8054	**0.5890**	43.27 (2.74×)	0.934(2.401×)
ToCa (N=4, R=93%)	2.637	5.838	256.655	0.8084	0.5842	43.42 (2.73×)	1.078(2.081×)
CSRe (k_h=2, ρ_c=0.184)	**2.611**	**5.216**	**260.176**	**0.8115**	0.5842	42.40(2.80×)	0.978(2.292×)

Table 2: MS-COCO results on 20-step PixArt-α. Latency is evaluated at batch size 32.

Method	FID-30K↓	COCO CLIP↑	Parti CLIP↑	TFLOPs↓	Latency (ms)↓
PixArt	27.941	26.581	27.229	11.18 (1.00×)	98.357(1.00×)
FORA (N=2)	29.583	26.545	**27.120**	5.66 (1.98×)	55.771(1.764×)
FORA (N=3)	29.737	26.325	26.850	4.01 (2.79×)	42.924(2.291×)
ECAD (fast)	30.292	26.259	26.899	4.27(2.62×)	42.969(2.289×)
ToCa (N=3, R=60%)	28.050	26.429	26.887	6.60 (1.69×)	71.307(1.379×)
ToCa (N=3, R=70%)	28.281	26.419	26.912	6.10 (1.83×)	65.905(1.492×)
ToCa (N=3, R=80%)	28.772	26.373	26.786	5.44 (2.06×)	60.289(1.631×)
CSRe (k_h=2, ρ_c=0.144)	**27.224**	**26.556**	27.118	5.78(1.93×)	72.197(1.362×)
CSRe (k_h=2, ρ_c=0.125)	28.768	26.454	26.958	4.29(2.61×)	50.108(1.963×)

Evaluation metrics. We report FID-50K, sFID, Inception Score, precision, and recall on ImageNet, and FID-30K and CLIP on MS-COCO, with additional PartiPrompts metrics. We report transformer FLOPs and normalized denoiser latency, hereafter latency. This latency rescales each method's measured latency to the CSRe framework's Full latency for the same model: it equals the CSRe Full latency divided by that method's same-framework speedup (Full latency divided by accelerated latency, both measured in that method's framework). It retains time units and is a rescaled estimate; raw measured latency is reported separately in Appendix A.9. Appendix A.8 specifies the evaluators and data protocols.

4.2 Main results

DiT-XL/2. CSRe uses 250 steps, $k_h = 2$, a base MLP channel ratio $\rho_c = 0.184$, and four-stage anchor periods $(6, 5, 4, 3)$, yielding 58 anchors (Appendix A.6). At 42.40 TFLOPs per image, CSRe obtains FID 2.611, compared with 2.637 for ToCa at 43.42 TFLOPs (Table 1). Its same-framework denoiser speedup is $2.292\times$ on an RTX 5090D (qualitative comparisons in Figure 5).

PixArt-α. CSRe uses 20 steps, $k_h = 2$, $\rho_c = 0.144$, and nominal anchor period $N = 3$ (exact anchor positions are given in Appendix A.6). At 5.78 TFLOPs, it obtains FID-30K 27.224, COCO CLIP 26.556, and PartiPrompts CLIP 27.118. ToCa-R70 uses 6.10 TFLOPs and obtains 28.281, 26.419, and 26.912, respectively (Table 2). The 4.29-TFLOP variant uses $k_h = 2$, $\rho_c = 0.125$, anchor period $N = 3$, and $K_b = 20$. It fully reuses the step-0 cache until the next anchor, then restricts MLP refresh to the top 20 change-ranked blocks while retaining selective head refresh in all blocks (Appendix A.6). It nearly matches ToCa-R80 in FID (28.768 versus 28.772) and gives higher COCO and PartiPrompts CLIP scores (qualitative comparisons in Figure 6).

FLUX.1-dev. CSRe uses 20 steps, anchor period $N = 3$, two refreshed heads in each double- and single-stream block, and an MLP ratio of approximately 0.105, with 512-channel groups capped at 3072 channels. Steps 1 and 2 fully reuse the step-0 attention and MLP caches; subsequent non-anchor steps perform selective refresh, with no top-K_b block restriction (Appendix A.6). This extends component-wise refresh to joint text–image attention. At 80.377 TFLOPs, it obtains FID-30K 26.511 and COCO CLIP 25.905, with a $1.728\times$ same-framework denoiser speedup on an RTX 5090D (Table 3). GPU memory constraints limit our FLUX evaluation to a batch size of two. In principle, larger batches better amortize execution overhead and improve GPU utilization, allowing greater speedups from selective refresh.

Table 3: Results on 20-step FLUX.1-dev at 256×256. Latency is evaluated at batch size 2. The best accelerated result for each quality metric is bold.

Method	FID-30K↓	COCO CLIP↑	TFLOPs↓	Latency (ms)↓
Full FLUX	28.520	25.600	198.70(1.00×)	1822.875(1.000×)
ECAD (fast)	**26.496**	**26.178**	63.02(3.15×)	734.617(2.481×)
FORA (N=3)	26.780	25.950	69.80(2.85×)	704.553 (2.587×)
ToCa (N=4, R=90%)	28.671	25.892	85.92(2.31×)	1073.844(1.698×)
CSRe (k_h=2, ρ_c=0.105)	26.511	25.905	80.377(2.472×)	1054.967(1.728×)

FLOPs and latency. Compute is transformer TFLOPs per image (two FLOPs per multiply–accumulate). Parentheses in the TFLOPs and latency columns report compute reduction and same-framework speedup, respectively. Appendices A.8 and A.9 give the protocol and source-native measurements. All latency measurements are performed on an RTX 5090D. Same-framework speedups are smaller than analytical FLOP reductions because selection, packing, and memory traffic remain, especially against PixArt's fused dense path. Appendix A.9 gives raw RTX 5090D measurements; the main tables show RTX 5090D latency, and Table 6 attributes PixArt operator time.

4.3 Ablation and analysis

Component selection. Table 4 compares component families and selection policies on the same 5K samples at matched compute (42.3648–42.4114 TFLOPs/image; Appendix A.10). Single-family controls reassign the disabled family's budget, whereas random controls preserve refresh counts. Joint change-ranked selection yields the lowest FID-5K (9.2880), compared with 9.4980 for MLP-only refresh, 39.6591 for attention-only refresh, 9.4208 with random heads, and 10.1365 with random heads and channels. These results support joint head–channel refresh and lagged-change ranking.

Table 4: Component-selection ablation.

Variant	Head	Chan.	FID-5K↓
CSRe	✓	✓	**9.2880**
MLP only	×	✓	9.4980
Random H	Random	✓	9.4208
Attn. only	✓	×	39.6591
Random H+C	Random	Random	10.1365

Table 5: Low-to-high-change budget transfer.

MLP block policy	FID-5K↓	COCO CLIP↑
All 28 blocks	34.029	26.559
Top-20	**33.728**	26.490
Bottom-20	34.481	26.374
Random-20	33.921 ± 0.679	26.496 ± 0.059

Block selection. Table 5 tests whether the MLP refresh budget can be concentrated on blocks with larger hidden-state changes. At each anchor, Top-20 ranks blocks by their completed-anchor changes. During the following selective interval, the eight lowest-ranked blocks receive zero MLP refresh width and fully reuse their cached MLP outputs, while the released budget is redistributed among the 20 highest-ranked blocks. Under the same 8.148-TFLOP MLP budget, Top-20 achieves the lowest FID-5K (33.728), compared with All-28 (34.029), Bottom-20 (34.481), and Random-20 (33.921 ± 0.679), while maintaining comparable COCO CLIP. This matched-budget comparison indicates that concentrating refresh computation on high-change blocks is more effective than spreading it across all blocks or selecting blocks without the change signal. Appendix A.10 specifies the prompts, seeds, schedules, and evaluators.

5 Conclusion

CSRe provides training-free diffusion-transformer acceleration through component-wise selective refresh. Its additive formulation updates cached module outputs with the projected deltas of selected attention heads and MLP channels. A shared compute budget coordinates the two component families, while completed-anchor measurements determine component identities and redistribute MLP widths across blocks. Hardware-aligned dense submatrices support execution of the selected components. The framework combines offline profiling and head-count calibration with online allocation and applies to DiT, PixArt, and FLUX architectures.

AI USE STATEMENT

Generative AI tools were used to assist with English translation and language editing, improving the grammar and clarity of author-written text to communicate the paper's central ideas more effectively. They were also used to support literature searches, including surveying the literature and identifying potentially relevant papers during manuscript preparation. All AI-assisted revisions were reviewed by the authors, and candidate references were verified against the original publications before citation. The scientific content of this paper, including the proposed method, experimental results, analyses, and conclusions, was developed and validated by the authors. The authors take full responsibility for the content, claims, and citations in this work.

REPRODUCIBILITY STATEMENT

The method is training-free. The accompanying material records inference flags, schedules, channel-tier sets, evaluation scripts, environment versions, random seeds, prompt/label lists, and table-reproduction commands. Calibration choices and final-evaluation data are kept disjoint. Algorithm steps and model configurations are specified in Appendices A.5 and A.6; implementation details, evaluation protocols, and raw latency measurements are provided in Appendices A.7, A.8, and A.9.

REFERENCES

Anirud Aggarwal, Abhinav Shrivastava, and Matthew Gwilliam. Evolutionary caching to accelerate your off-the-shelf diffusion model. In *International Conference on Learning Representations*, 2026.

Black Forest Labs. FLUX: Reference implementation and model cards. `https://github.com/black-forest-labs/flux`, 2024.

Fanpu Cao, Yaofo Chen, Zeng You, and Wei Luo. Procache: Constraint-aware feature caching with selective computation for diffusion transformer acceleration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(24):19862–19870, 2026. doi: 10.1609/aaai.v40i24.39069.

Guantao Chen, Shikang Zheng, Yuqi Lin, and Linfeng Zhang. Forecast the principal, stabilize the residual: Subspace-aware feature caching for diffusion transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *International Conference on Learning Representations*, 2024a.

Lei Chen, Yuan Meng, Chen Tang, Xinzhu Ma, Jingyan Jiang, Xin Wang, Zhi Wang, and Wenwu Zhu. Q-DiT: Accurate post-training quantization for diffusion transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28306–28315, 2025a.

Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. δ-dit: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024b.

Zhiyuan Chen, Keyi Li, Yifan Jia, Le Ye, and Yufei Ma. Accelerating diffusion transformer via increment-calibrated caching with channel-aware singular value decomposition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025b.

Bowen Cui, Yuanbin Wang, Huajiang Xu, Biaolong Chen, Aixi Zhang, Hao Jiang, Zhengzheng Jin, Xu Liu, and Pipei Huang. Denoising as path planning: Training-free acceleration of diffusion models with dpcache. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Liang Feng, Shikang Zheng, Jiacheng Liu, Yuqi Lin, Qinming Zhou, Peiliang Cai, Xinyu Wang, Junjie Chen, Chang Zou, Yue Ma, and Linfeng Zhang. HiCache: A plug-in scaled-hermite upgrade for taylor-style cache-then-forecast diffusion acceleration. In *International Conference on Learning Representations*, 2026.

Lihui Gu, Jingbin He, Lianghao Su, Kang He, Wenxiao Wang, and Yuliang Liu. ScalingCache: Extreme acceleration of DiTs through difference scaling and dynamic interval caching. In *International Conference on Learning Representations*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025a.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. In *IEEE/CVF International Conference on Computer Vision*, 2025b.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Fei Ren, Shaobo Wang, Kaixin Li, and Linfeng Zhang. SpeCa: Accelerating diffusion transformers with speculative feature caching. In *ACM International Conference on Multimedia*, 2025c.

Jinming Lou, Wenyang Luo, Yufan Liu, Bing Li, Xinmiao Ding, Weiming Hu, Jiajiong Cao, Yuming Li, and Chenguang Ma. Token caching for diffusion transformer acceleration. *arXiv preprint arXiv:2409.18523*, 2024.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22 (4):730–751, 2025. doi: 10.1007/s11633-025-1562-4.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision*, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024.

Austin Silveria, Soham V. Govande, and Daniel Y. Fu. Chipmunk: Training-free acceleration of diffusion transformers with dynamic column-sparse deltas. *arXiv preprint arXiv:2506.03275*, 2025.

Byunggwan Son, Jeimin Jeon, Jeongwoo Choi, and Bumsub Ham. Relational feature caching for accelerating diffusion transformers. In *International Conference on Learning Representations*, 2026.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Zhihang Yuan, Hanling Zhang, Pu Lu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. DiTFastAttn: Attention compression for diffusion transformer models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0037.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. In *International Conference on Learning Representations*, 2025.

Shikang Zheng, Guantao Chen, Qinming Zhou, Yuqi Lin, Lixuan He, Chang Zou, Peiliang Cai, Jiacheng Liu, and Linfeng Zhang. Let features decide their own solvers: Hybrid feature caching for diffusion transformers. In *International Conference on Learning Representations*, 2026.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *International Conference on Learning Representations*, 2025.

Chang Zou, Shikang Zheng, Evelyn Zhang, Runlin Guo, Haohang Xu, Zhengyi Shi, Conghui He, Xuming Hu, and Linfeng Zhang. Rethinking token-wise feature caching: Accelerating diffusion transformers with dual feature caching. *IEEE Transactions on Image Processing*, 35:6211–6220, 2026. doi: 10.1109/TIP.2026.3698363.

A Appendix

A.1 Formal Score and Budget Definitions

Denoiser trajectory and sampler. Let $s \in \{0, \dots, S-1\}$ index chronological sampling steps, $\boldsymbol{x}_s$ be the sampler latent, and $\boldsymbol{u}_{s,\ell}$ an activation inside the denoiser. A standard DiT evaluation followed by a sampler update is

$$\begin{aligned} \boldsymbol{u}_{s,0} &= \mathrm{Embed}(\boldsymbol{x}_s), \qquad \mathcal{D}_\theta(\boldsymbol{x}_s, s, \boldsymbol{\eta}) = \mathrm{Out}(\boldsymbol{u}_{s,Q}), \\ \boldsymbol{x}_{s+1} &= \Psi_s(\boldsymbol{x}_s, \mathcal{D}_\theta(\boldsymbol{x}_s, s, \boldsymbol{\eta}))\,, \end{aligned} \tag{5}$$

Here $\boldsymbol{\eta}$ is an optional condition, Ψ_s is the sampler rule, and Q is the number of residual submodules in the backbone. The residual insertion is defined in (1).

Raw states and projected contributions. We distinguish each component's raw state $\boldsymbol{r}$ from its contribution $\boldsymbol{c}$ to the module output. For attention head h,

$$\boldsymbol{r}^{\mathrm{attn}}_{s,\ell,h} = \boldsymbol{A}_{s,\ell,h}\boldsymbol{V}_{s,\ell,h}, \qquad \boldsymbol{c}^{\mathrm{attn}}_{s,\ell,h} = \boldsymbol{r}^{\mathrm{attn}}_{s,\ell,h}\boldsymbol{W}^{O}_{\ell,h}, \qquad \mathrm{Attn}(\boldsymbol{u}_{s,\ell}) = \boldsymbol{b}^{O}_{\ell} + \sum_{h=1}^{H} \boldsymbol{c}^{\mathrm{attn}}_{s,\ell,h}. \tag{6}$$

Thus $\boldsymbol{r}^{\mathrm{attn}}_{s,\ell,h} \in \mathbb{R}^{N_q \times d_h}$ is the unprojected head output, whereas $\boldsymbol{c}^{\mathrm{attn}}_{s,\ell,h} \in \mathbb{R}^{N_q \times D}$ is its projected contribution. Here $\boldsymbol{A}_{s,\ell,h}$ contains the attention weights, $\boldsymbol{V}_{s,\ell,h}$ the values, $\boldsymbol{W}^{O}_{\ell,h}$ the head's output-projection slice, N_q is the number of query tokens, d_h is the head width, D is the model width, and H is the number of heads. For MLP hidden channel j,

$$\begin{aligned} \boldsymbol{r}^{\mathrm{mlp}}_{s,\ell,j} &= z_{s,\ell,j} = \phi(\boldsymbol{u}_{s,\ell}\boldsymbol{w}^{(1)}_{\ell,j} + b^{(1)}_{\ell,j}), \\ \boldsymbol{c}^{\mathrm{mlp}}_{s,\ell,j} &= \boldsymbol{r}^{\mathrm{mlp}}_{s,\ell,j}\boldsymbol{w}^{(2)}_{\ell,j}, \qquad \mathrm{MLP}(\boldsymbol{u}_{s,\ell}) = \boldsymbol{b}^{(2)}_{\ell} + \sum_{j=1}^{M} \boldsymbol{c}^{\mathrm{mlp}}_{s,\ell,j}. \end{aligned} \tag{7}$$

Here $\boldsymbol{r}^{\mathrm{mlp}}_{s,\ell,j} \in \mathbb{R}^{N}$ is token-wise and $\boldsymbol{c}^{\mathrm{mlp}}_{s,\ell,j} \in \mathbb{R}^{N \times D}$ is its outer-product contribution, where N is the number of input tokens. The vectors $\boldsymbol{w}^{(1)}_{\ell,j}$ and $\boldsymbol{w}^{(2)}_{\ell,j}$ are respectively the first-projection column and second-projection row, $b^{(1)}_{\ell,j}$ is the corresponding hidden bias, ϕ is the activation, and M is the hidden width.

Component-cache transition. Let $\widetilde{\boldsymbol{r}}_{s,j}$ and $\widetilde{\boldsymbol{c}}_{s,j}$ denote component j's latest cached raw state and projected contribution upon entering step s. Evaluating the selected components $\mathcal{I}_s$ advances these states:

$$(\widetilde{\boldsymbol{r}}_{s+1,j}, \widetilde{\boldsymbol{c}}_{s+1,j}) = \begin{cases} (\boldsymbol{r}_{s,j}, \boldsymbol{c}_{s,j})\,, & j \in \mathcal{I}_s, \\ (\widetilde{\boldsymbol{r}}_{s,j}, \widetilde{\boldsymbol{c}}_{s,j})\,, & j \notin \mathcal{I}_s. \end{cases} \tag{8}$$

Equation (8) is the state update and (3) its equivalent incremental form. The fixed output bias is included once: $\boldsymbol{y}^{\mathrm{cache}}_{s} = \boldsymbol{b} + \sum_j \widetilde{\boldsymbol{c}}_{s,j}$. These latest-component caches are distinct from the completed-anchor snapshots used for scoring.

Attention heads. Following Equation (6), $\boldsymbol{r}^{\mathrm{attn}}_{s,\ell,h}$ denotes head h's raw output and $\boldsymbol{c}^{\mathrm{attn}}_{s,\ell,h}$ its projected contribution to the attention output. CSRe ranks heads by changes in $\boldsymbol{r}^{\mathrm{attn}}_{s,\ell,h}$, while selected cache updates use changes in $\boldsymbol{c}^{\mathrm{attn}}_{s,\ell,h}$. For sample i and block ℓ, anchor snapshots $\bar{\boldsymbol{r}}^{\mathrm{attn}}$ define

$$\begin{aligned} q^{\mathrm{attn}}_{i,a_m,\ell,h} &= \mathrm{mean}_{n,d}\left|\bar{\boldsymbol{r}}^{\mathrm{attn}}_{i,a_m,\ell,h} - \bar{\boldsymbol{r}}^{\mathrm{attn}}_{i,a_{m-1},\ell,h}\right|, \\ \mathcal{S}_{i,a_m,\ell} &= \mathrm{TopK}_h\big(\{q^{\mathrm{attn}}_{i,a_m,\ell,h}\}_{h=1}^{H}; k_h\big). \end{aligned} \tag{9}$$

Here n indexes query tokens and $d \in \{1,\dots,d_h\}$ indexes features within a head. Head identities vary by sample, block, attention module, and anchor interval; k_h remains fixed. The attention-module index is suppressed. Under classifier-free guidance, each conditional mask is shared only with its paired unconditional sample.

MLP channels, block-change normalization, and widths. Let $p \in \{1,\dots,P\}$ index MLP refresh modules, with $P = L$ for the L DiT or PixArt blocks. CSRe ranks channels using the raw hidden states $\boldsymbol{r}^{\mathrm{mlp}}$ from Equation (7), while cache updates use their projected contributions $\boldsymbol{c}^{\mathrm{mlp}}$. Anchor snapshots $\bar{\boldsymbol{r}}^{\mathrm{mlp}}$ define

$$\begin{aligned} q^{\mathrm{mlp}}_{i,a_m,p,j} &= \mathrm{mean}_{n}\left|\bar{\boldsymbol{r}}^{\mathrm{mlp}}_{i,a_m,p,n,j} - \bar{\boldsymbol{r}}^{\mathrm{mlp}}_{i,a_{m-1},p,n,j}\right|, \\ \mathcal{T}_{i,a_m,p} &= \mathrm{TopK}_j\big(\{q^{\mathrm{mlp}}_{i,a_m,p,j}\}_{j=1}^{M}; k^*_{a_m,p}\big). \end{aligned} \tag{10}$$

Widths $k^*_{a_m,p}$ are batch-shared, but channel identities are sample-specific. The head count k_h is shared across attention refresh modules, whereas $k^*_{a_m,p}$ varies across MLP modules and completed-anchor intervals. For B_c conditional samples, normalize each sample's module change before averaging:

$$d_{i,a_m,p} = \sum_{j=1}^{M} q^{\mathrm{mlp}}_{i,a_m,p,j}, \qquad D_{a_m,p} = \frac{1}{B_c}\sum_{i=1}^{B_c} \frac{d_{i,a_m,p}}{\sum_{r=1}^{P} d_{i,a_m,r} + \epsilon}, \tag{11}$$

where $\epsilon > 0$ prevents division by zero in the per-sample normalization. This anchor-to-anchor hidden-activation change includes upstream attention effects propagated to the MLP input; no separate attention score is added.

For equal-width, equal-cost modules with hidden width M, let $\mathcal{K}$ contain admissible dense widths and $\mathcal{B} = \{\sum_{p=1}^{P} k_p : (k_1,\dots,k_P) \in \mathcal{K}^P\}$. A normalized stage factor $\gamma_{g(a_m)}$ (Appendix A.6) sets the aggregate width per selective step and the proportional targets. Here $g(a_m)$ is the temporal stage containing the anchor and $\Pi_{\mathcal{B}}$ selects the nearest feasible total in $\mathcal{B}$, breaking ties toward the smaller one:

$$\begin{aligned} B_{a_m} &= \Pi_{\mathcal{B}}\big(PM\rho_c\gamma_{g(a_m)}\big), \\ \widetilde{k}_{a_m,p} &= \begin{cases} B_{a_m}\dfrac{D_{a_m,p}}{\sum_{r=1}^{P} D_{a_m,r}}, & \sum_r D_{a_m,r} > 0, \\ B_{a_m}/P, & \text{otherwise.} \end{cases} \end{aligned} \tag{12}$$

We project these targets to admissible widths while preserving the aggregate budget:

$$\big(k^*_{a_m,1},\dots,k^*_{a_m,P}\big) = \arg\min_{(k_1,\dots,k_P)\in\mathcal{K}^P} \sum_{p=1}^{P}(k_p - \widetilde{k}_{a_m,p})^2 \quad \text{s.t.} \quad \sum_{p=1}^{P} k_p = B_{a_m}. \tag{13}$$

Ties use a fixed index order. At the anchor, selected first-projection columns and second-projection rows are packed into compact dense matrices; selective steps reuse them without repacking.

Omitted temporal increments. Using the pre-update component caches defined in Section 3.1, subtraction of the selective output from the dense module output at the same current input gives

$$\boldsymbol{y}_s - \widehat{\boldsymbol{y}}_s = \sum_{j\notin\mathcal{I}_s} (\boldsymbol{c}_{s,j} - \widetilde{\boldsymbol{c}}_{s,j}). \tag{14}$$

This is a local identity, not an error bound against a separate dense denoising trajectory or on final image quality. Selecting all components recovers dense evaluation of the module, while selecting none reuses its complete cached output.

Anchor-placement proxy. For a fixed few-step sampler, let $d_s \geq 0$ denote profiled average drift and let K be the prescribed anchor count, with $2 \leq K \leq S$. The offline schedule minimizes cache-age-weighted drift:

$$\mathcal{F}^* = \arg \min_{\substack{\mathcal{F} \subseteq \{0,\dots,S-1\} \\ |\mathcal{F}|=K,\ \{0,1\} \subseteq \mathcal{F}}} \sum_{s=0}^{S-1} (s - a_{\mathcal{F}}(s)) d_s, \qquad a_{\mathcal{F}}(s) = \max\{r \in \mathcal{F} : r \leq s\}. \tag{15}$$

Here $a_{\mathcal{F}}(s)$ is the most recent anchor at or before s. This proxy does not optimize final image quality. It describes the profiled schedule with two initial anchors; the periodic early-reuse configurations in Appendix A.6 instead initialize at step 0, reuse the cache at steps 1 and 2, and obtain their first anchor-pair score at step 3.

Attention-specific delta update. Let $\widetilde{\boldsymbol{r}}^{\mathrm{attn}}_{i,s,\ell,h}$ be the latest raw-head cache, $\boldsymbol{o}^{\mathrm{cache}}_{i,s,\ell}$ the cached projected output, and $\mathcal{S}_{i,a_m,\ell}$ the identities selected at anchor a_m. For a selective step in the following interval, Equation (3) becomes

$$\widehat{\boldsymbol{o}}_{i,s,\ell} = \boldsymbol{o}^{\mathrm{cache}}_{i,s,\ell} + \sum_{h \in \mathcal{S}_{i,a_m,\ell}} (\boldsymbol{r}^{\mathrm{attn}}_{i,s,\ell,h} - \widetilde{\boldsymbol{r}}^{\mathrm{attn}}_{i,s,\ell,h}) \boldsymbol{W}^{O}_{\ell,h}. \tag{16}$$

For a selected head, $\widetilde{\boldsymbol{r}}^{\mathrm{attn}}_{i,s+1,\ell,h} = \boldsymbol{r}^{\mathrm{attn}}_{i,s,\ell,h}$; equivalently, Equation (8) advances its projected contribution cache. The module-output cache becomes $\boldsymbol{o}^{\mathrm{cache}}_{i,s+1,\ell} = \widehat{\boldsymbol{o}}_{i,s,\ell}$. Anchor snapshots used for scoring are stored separately.

A.2 Architecture-Specific Extensions

For DiT-XL/2, every block contains one self-attention refresh module and one MLP refresh module. For PixArt-α, Equation (16) is applied independently to self- and cross-attention, each with its own head cache, anchor snapshot, and selected identities. A selected cross-attention head recomputes its image queries against the complete text key/value sequence.

FLUX.1-dev contains double-stream and single-stream blocks. In a double-stream block, image and text queries attend jointly to the concatenated key/value sequence. One selected head therefore includes its complete joint token map; its delta is passed through the image and text output-projection slices and added to the two corresponding cached outputs. The image and text MLPs remain separate refresh modules. In a single-stream block, attention and MLP states are concatenated before the shared second linear layer. Splitting that layer's input columns into attention-head slices and MLP-channel columns gives the two additive component families used by Equations (16) and (3). The concatenated text–image sequence is retained for every selected joint-attention head.

Unequal branch costs. The unweighted constraint in Equation (13) preserves an exact channel-width total, not automatically an exact FLOP total across heterogeneous branches. A double-stream image MLP, a double-stream text MLP, and a single-stream MLP can process different numbers of tokens. Let c_p be the analytical cost per refreshed channel in branch p under the fixed evaluation shape. Its leading two-projection cost is proportional to $N_p(D_{\mathrm{in},p} + D_{\mathrm{out},p})$. The realized selective MLP cost is therefore accounted for as

$$C^{\mathrm{sel}}_{\mathrm{MLP},a_m} = \sum_{p=1}^{P} c_p k_{a_m,p}. \tag{17}$$

An exact width budget must not be described as matched FLOPs unless these costs are equal or the candidate configurations are matched using their realized cost. A cost-constrained extension replaces the equality in Equation (13) by $\sum_p c_p k_{a_m,p} = C_{a_m}$, with C_{a_m} chosen from the feasible cost set.

Input and conditioning projections, positional tensors, normalization and modulation, current residual gates, residual additions, and final projections outside the selective paths remain dense in all three architectures. These architecture mappings change neither the latest-cache recursion nor the lagged information available to the selector.

A.3 Temporal and Depth Observation on DiT

We aggregate adjacent-step post-activation MLP change from 5,000 dense DiT-XL/2 teacher trajectories before applying any selective policy. Figure 7 shows two complementary departures from a uniform cache policy. The four temporal regions have mean changes $1.00\times$, $1.62\times$, $2.56\times$, and $3.72\times$ relative to the first region. Across depth, the mean coefficient of variation is 0.460. Transition-wise ranking assigns the top four blocks 24.35% of total MLP change, compared with 18.51% under one global ranking; at eight blocks, the corresponding coverage is 41.26% versus 33.73%. The three change points are also stable: over 128 independent cumulative profile chunks, their ranges are steps 57–58, 91–92, and exactly 235. This observation determines the stage boundaries and motivates dynamic block placement; it is measured independently of the 50K quality runs. We use four stage-specific MLP budget factors only for DiT-XL/2, whose 250-step sampling trajectory supports a finer temporal allocation. PixArt-α and FLUX.1-dev use only 20 sampling steps, for which such stage-wise refinement offers limited benefit; we therefore use a single base MLP ratio without the four-stage modulation for these models.

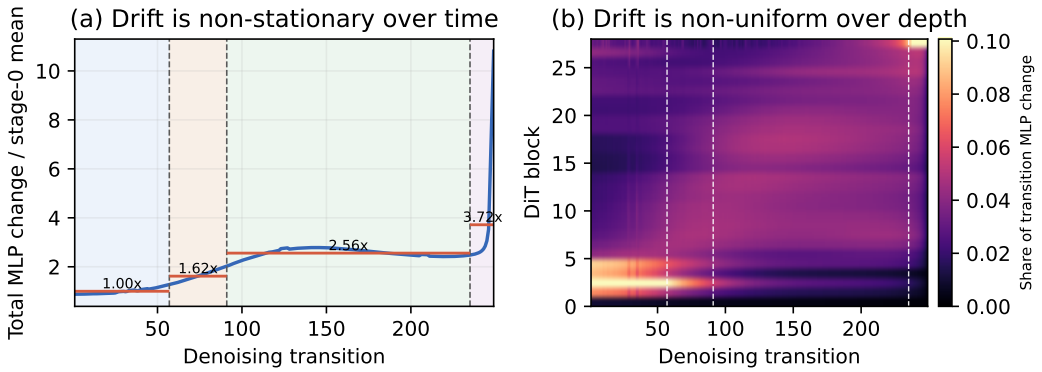


Figure 7: Dense-teacher MLP drift on DiT-XL/2. **(a)** Total change is non-stationary across denoising, with stable profiled stage boundaries. **(b)** Each column normalizes to one and shows where that transition's change lies over transformer depth; a uniform per-block width cannot follow this evolving distribution.

A.4 Diagnostic Details

A.4.1 Controlled downstream recovery

We run dense 250-step DiT-XL/2 trajectories and intervene at 28 steps stratified over early, middle, and late denoising. The Cartesian product of 32 ImageNet classes, two independent noise samples per class (64 trajectories), 28 intervention steps, and 28 blocks gives 50,176 sample–step–block interventions. At step t, component identities are ranked only from the preceding dense transition $(t-2)\to(t-1)$; the current transition $(t-1)\to t$ is not observed by the selector. For attention, the ranked set is

$$\mathcal{S}^{(k)}_{i,t,\ell} = \mathrm{TopK}_h \left\| \left(\boldsymbol{r}^{\mathrm{attn}}_{i,t-1,\ell,h} - \boldsymbol{r}^{\mathrm{attn}}_{i,t-2,\ell,h}\right) \boldsymbol{W}^{O}_{\ell,h} \right\|_F , \tag{18}$$

and the MLP uses the projected-channel score $\|\boldsymbol{r}^{\mathrm{mlp}}_{i,t-1,\ell,j} - \boldsymbol{r}^{\mathrm{mlp}}_{i,t-2,\ell,j}\|_2 \|\boldsymbol{w}^{(2)}_{\ell,j}\|_2$. These projected diagnostic scores include the output projections, whereas Equations (9) and (10) rank the corresponding unprojected state changes. Both use only a completed transition and therefore have the same lagged information pattern. The diagnostic tests whether lagged changes carry ranking information, not equivalence of the projected and unprojected scores. It uses a one-transition lag, whereas CSRe holds completed-anchor decisions over a selective interval.

For one block at a time, we start from its cached output at $t-1$, add the true current deltas of only the ranked components, and evaluate the remaining operations and network suffix densely. Let m index an intervention and let $E_m(k)$ be the MSE between its final CFG noise prediction and the

dense all-component reference. We aggregate by zero-update error mass:

$$R_{\mathrm{w}}(k) = 1 - \frac{\sum_m E_m(k)}{\sum_m E_m(0)}. \tag{19}$$

This form prevents interventions with negligible zero-update error from dominating through per-instance normalization. Random subsets use three fixed selection seeds, and the reported uncertainty is one standard deviation across those seeds. The equal per-unit line is a visual reference, $R_{\mathrm{eq}}(k) = k/H$ for attention and $R_{\mathrm{eq}}(k) = k/M$ for the MLP; it is not an additional selection method.

For attention, selecting 1, 2, 4, 7, and 10 complete heads recovers 30.1%, 47.5%, 69.7%, 83.4%, and 91.7% of the zero-head downstream error. Selecting 128, 512, 1024, and 1792 MLP channels recovers 32.8%, 59.6%, 77.1%, and 90.5%. At the two marked component counts, random selection recovers $16.2\% \pm 0.4\%$ for two heads and $37.8\% \pm 0.1\%$ for 1024 channels.

A.5 Algorithm

1. On a calibration split, first keep MLP refresh dense and decrease the refreshed attention-head count k_h to identify where quality begins to deteriorate. Then refine k_h under the final compute target while adjusting the MLP budget; empirically, one or two heads are usually sufficient. Freeze the choice for the model and anchor schedule, derive the residual MLP ratio, and initialize empty latest-component caches and anchor snapshots.
2. At each denoising step, apply the model-specific schedule. A step can be a full anchor, a full-cache reuse step, or a selective step.
3. At an anchor, evaluate every transformer block densely at the current accelerated-trajectory state. Before replacing the anchor snapshots, measure each head and channel against its preceding anchor snapshot when one is available. The first anchor initializes the snapshots; subsequent completed-anchor pairs provide the scores to select k_h heads, determine the batch-shared MLP widths, and select sample-specific channels for the next interval. Refresh every latest cache to the current anchor value and pack the selected parameter slices.
4. At a full-cache reuse step, retain the attention and MLP outputs while executing the current conditioning, residual skeleton, and final output operations. Early-reuse configurations use this path until the second completed anchor makes change-guided selection available.
5. At a selective step, update the selected attention heads and MLP channels from their own latest cached states, add their projected deltas to the cached module outputs, and advance the sampler. The attention count is fixed for the trajectory; identities and batch-shared MLP widths are held within the selective interval and may change only at its next anchor.

The anchor snapshots used for ranking are read only between anchors. They are separate from the latest-component caches, which are updated whenever their components are selected. This distinction gives all ranking scores a common anchor-to-anchor horizon without discarding the fresher states required by the delta update.

A.6 Anchor Schedules and Model Configurations

For a profiled schedule with a fixed number K of anchors in a few-step sampler, we apply Equation (15) to a small dense profile. The objective is a cache-age-weighted drift proxy, not an optimization of final image quality.

Let n_g be the number of sampling positions in stage g. We normalize the stage factors by $\sum_g n_g \gamma_g / \sum_g n_g = 1$. Equation (12) therefore treats ρ_c as a base channel ratio; anchor exclusion and projection to a model-specific feasible budget can make the realized selective-step average differ slightly. The ratio is derived after subtracting anchor and fixed attention-refresh costs from the compute target. B_{a_m} is held throughout the interval after anchor a_m. For equal-cost modules, (13) preserves the feasible aggregate width exactly. Heterogeneous FLUX branches use the realized-cost accounting in Appendix A.2; FLOPs are audited for each reported configuration.

For DiT-XL/2, the profiled configuration divides 250 sampling steps at normalized boundaries $(0.228, 0.364, 0.94)$ and uses periods $(6, 5, 4, 3)$, producing 58 anchor and 192 selective steps. The

reported DiT operating point uses a base ratio $\rho_c = 0.184$ with normalized stage factors derived from the anchor-to-anchor profile. For the 20-step, $\rho_c = 0.144$ PixArt configuration, the recorded full-anchor positions are $\{0, 1, 4, 7, 10, 12, 15, 18, 19\}$.

Low-compute PixArt configuration. The discrete cost of an anchor depends strongly on the sampling horizon. In a 250-step DiT-XL/2 trajectory, changing the schedule by a few anchors modifies the fraction of dense evaluations only modestly. In a 20-step PixArt trajectory, however, an additional anchor replaces a selective step with a full denoiser evaluation and therefore consumes a substantially larger share of the compute budget. The standard CSRe initialization uses steps 0 and 1 as anchors so that their difference can guide the first selective interval. The low-compute configuration instead retains step 0 as the initial anchor and applies full cache reuse until the next anchor, postponing change-guided selection until an anchor-to-anchor measurement is available. This leaves more of the compute budget for subsequent selective intervals.

The configuration in Table 2 uses anchors $\{0, 3, 6, 9, 12, 15, 18\}$, $k_h = 2$, and $\rho_c = 0.125$; steps 1 and 2 reuse all cached components. Under a low budget, continuous MLP-width targets for low-ranked blocks can reach, or fall below, the smallest nonzero admissible width. At subsequent selective intervals, we therefore restrict MLP refresh to the top $K_b = 20$ blocks ranked by anchor-measured change, set the remaining blocks' refresh widths to zero, and reuse their cached MLP outputs. The channel budget is redistributed within the selected set. Selective attention-head refresh is retained in every block. The realized cost is 4.29 TFLOPs per image.

FLUX configuration. FLUX CSRe uses anchor period $N = 3$, full cache reuse at steps 1 and 2, two double-stream and two single-stream attention heads, and MLP ratio 0.105. Channel ranking is projected-L2. Widths use 512-channel groups capped at 3072 channels, and top-20 block restriction is disabled. Full-cache steps retain timestep conditioning, the block skeleton, and the final output layer; attention and MLP branches reuse their cached outputs. Unselected contributions are retained rather than set to zero.

A.7 Hardware-Aware Implementation and Cost Analysis

For a selected MLP width $k^*_{a_m,p}$, the implementation gathers the first- and second-projection parameter slices associated with the same hidden channels and packs them into compact dense matrices. Restricting $k^*_{a_m,p}$ to a profiled set $\mathcal{K}$ avoids an irregular sparse operator, but selection, gathering, and cache traffic remain real costs. The set is chosen separately for each model and implementation. Equation (13) is solved only over aggregate budgets feasible under that set. If an anchor has no following selective step, no packed matrices are needed for that interval.

For the reported DiT-XL/2 implementation, widths are aligned in multiples of 128 and the partial tiers stop at 1024 because kernel measurements, including per-sample weight packing, place width 1152 near break-even with the dense MLP and make wider partial paths slower. The implementation therefore switches directly from 1024 to the dense width $M = 4608$. Including M lets a high-change block receive a full refresh; its use reflects this discrete execution choice rather than a claim that intermediate widths are algorithmically invalid. FLUX uses its own architecture-specific admissible widths and any configuration-specific caps; neither is a universal CSRe setting.

Triton scoring, packing, or fused selective operators are optional execution backends for the same cache-update equations. Their use does not change the refresh units or the information available to the policy. The reproducibility configuration lists the actual tier set, dense fallback, and enabled kernels separately for DiT, PixArt, and FLUX.

For batch size B, token count N, model width D, k_h selected heads of width d_h, and k_ℓ selected MLP channels, the leading selective costs per block are

$$C_{\text{attn}} = \mathcal{O}(BNDk_hd_h + BN^2k_hd_h), \tag{20}$$

$$C_{\text{MLP}} = \mathcal{O}(BNDk_\ell). \tag{21}$$

For cross-attention, the second term uses the image-query and text-key sequence lengths instead of N^2. These expressions omit selection, packing, cache access, and other dense operations. We therefore report analytical FLOPs and latency separately.

Table 6: PyTorch operator composition for one warmed-up 20-step PixArt batch. Entries are non-overlapping self CUDA time in milliseconds. This traced diagnostic uses a different operating configuration from Table 2.

Method	Total	Dense GEMM	Flash attn.	BMM+softmax	Select/index	MLP pack	Elem./norm	Copy/layout
Full PixArt	1506.1	534.7	113.3	0.0	8.0	0.0	602.0	244.0
ToCa R70	2060.6	298.7	0.0	568.6	102.5	0.0	765.0	250.3
CSRe	1533.6	272.0	73.4	59.3	73.8	64.7	689.7	293.0

PixArt profile details. The profiler run uses one instrumented batch, a base channel ratio of 0.125, and anchors at $\{0, 1, 5, 8, 11, 14, 17\}$. The reported configuration in Table 2 instead uses $\rho_c = 0.144$ and anchors at $\{0, 1, 4, 7, 10, 12, 15, 18, 19\}$ under repeated synchronized wall-clock timing. Profiler instrumentation and the different configuration make the Total column unsuitable as an end-to-end latency estimate; the table is used only to attribute operator composition. ToCa saves 236.0 ms of dense GEMM time, but its released PixArt path disables the memory-efficient attention call in order to return attention maps. It spends 354.5 ms in batched matrix multiplication, 214.1 ms in softmax, and 102.5 ms in selection/indexing. In cross-attention, the released code flattens all CFG streams and materializes a block-diagonal attention tensor of shape $[1, 16, 16384, 988]$, where $16384 = 64 \times 256$ image queries and 988 is the sum of text lengths across streams. The mask removes cross-sample entries only after the concatenated matrix is computed. The timed path also materializes a self-attention map although its final PixArt score uses only cross-attention. These implementation choices explain the local latency without changing ToCa's token-FLOP accounting.

For CSRe, dense GEMM and flash-attention time fall by 262.7 and 39.9 ms, while selection/indexing, dynamic MLP-weight packing, selected-path BMMs, and additional copy and elementwise traffic consume the saving. DiT-XL/2 better amortizes these costs over 250 steps and has no text cross-attention; PixArt uses 20 steps, nine of which remain dense in the reported schedule.

A.8 Evaluation Protocol

ImageNet. All rows use the released DiT-XL/2 256 checkpoint, global seed 0, identical class-label sampling, the 250-step ancestral DDPM sampler, CFG 1.5, and 50,000 uint8 RGB samples. PNG files are sorted by integer sample index and packed into NPZ before the ADM evaluator computes FID, sFID, Inception Score, precision, and recall against the same ImageNet 256 reference statistics.

MS-COCO. All methods use the same ordered 30K COCO-2017 prompts, seed, PixArt checkpoint, 20-step DPM-Solver, CFG 4.5, and JPEG quality 75. FID uses the ToCa-released COCO-2017 reference statistics and `pytorch-fid` 0.3.0 with 2048-dimensional Inception pool3 features. CLIP uses the frozen OpenAI CLIP ViT-L/14 evaluator with identical preprocessing for COCO and the fixed 1,632-prompt PartiPrompts subset.

FLUX. All FLUX rows use FLUX.1-dev at 256×256, 20 denoising steps, guidance scale 5, and the same COCO prompt list. The CSRe run uses BF16, seed 20260904, and quality-generation batch size 2 per GPU. This seed matches the reproduced baselines, but identical initial-noise pairing across frameworks has not been independently verified. The 30,000 PNG outputs are complete and aligned to prompts by global index. FID uses the shared prompt-matched COCO-30K Lanczos reference statistics with `pytorch-fid` Inception pool3 features; COCO CLIP uses the frozen OpenAI CLIP ViT-L/14 evaluator. ImageReward-v1.0 gives 0.641 on 100 frozen calibration prompts with 10 seeds each and 0.921 on 1,632 PartiPrompts images.

Baselines and operating-point alignment. We compare against dense inference, reduced-step sampling on DiT, periodic full-feature reuse following FORA, ToCa (Zou et al., 2025), and ECAD on the architectures for which aligned implementations are available. DiT experiments include the published ToCa setting $N{=}4$, $R{=}93\%$, two FORA operating points, and reduced-step controls. PixArt experiments use ToCa with $N{=}3$ and $R \in \{60\%, 70\%, 80\%\}$. Here, R is ToCa's cache ratio and is not a parameter of CSRe. CSRe operating points are first matched by realized transformer FLOPs; latency is evaluated independently.

The public FORA repository does not provide its announced PixArt implementation. We therefore label the corresponding rows as a reproduction: starting from the public PixArt code, we implement the static periodic attention–MLP reuse rule released for FORA's DiT implementation. We do not substitute ToCa's implementation for FORA. The public DiT implementation also accepts only an integer cache period. For the FLOP-aligned $N = 2.8$ point, we retain FORA's full/cache decision and uniformly place 91 dense evaluations over the 250-step trajectory. The current RTX 5090D measurements and the corresponding framework Full reference are reported in Appendix A.9.

For FLUX, Table 3 compares against ToCa, FORA (N=3), and the released ECAD fast schedule. All COCO-30K rows use the same prompt protocol. KID on smaller samples is used only for operating-point screening and does not replace the frozen COCO-30K evaluation.

Latency and FLOPs. Compute tables use TFLOPs per image, with two FLOPs per multiply–accumulate. The existing FLUX baseline MAC counts are converted by this convention; their transformer operator scope is unchanged. For the CSRe row, the equivalent PyTorch selected-compute path is counted at batch size 1: Full and accelerated counts are 206,953,953,361,920 and 83,715,705,274,368 FLOPs, respectively. Their ratio, 0.404514, is mapped to the established Full scale of 198.70 TFLOPs, giving 80.377 TFLOPs and a $2.472\times$ compute reduction. Equivalent PyTorch operators are counted because the counter does not directly account for the custom Triton kernels. All experiments use NVIDIA GeForce RTX 5090D GPUs. Latency is measured on one RTX 5090D at batch sizes 18, 32, and 2 for DiT-XL/2, PixArt-α, and FLUX.1-dev, respectively. DiT-XL/2 uses its default TF32 matrix multiplication path, whereas all PixArt-α measurements use FP16, as recorded by the inference logs. We use synchronized CUDA events and exclude runs with a concurrent GPU process. Every reported latency uses at least two warm-up iterations followed by five synchronized timed repetitions. Per-image denoiser latency is batch wall time divided by batch size and excludes text encoding, VAE decoding, image encoding, and disk I/O; end-to-end latency is recorded separately. FLOPs count transformer blocks only and include the selection and cache operations represented by the analytical accounting.

Each baseline is executed from its own source tree. All three main tables report each implementation's same-framework speedup and latency,

$$L_m^{\mathrm{norm}} = \frac{L_m^{\mathrm{native}}}{L_{\mathrm{full},m}^{\mathrm{native}}} L_{\mathrm{full}}^{\mathrm{ours}}. \tag{22}$$

This retains each method's native relative speed while expressing it on one dense-latency scale. This latency is a rescaled estimate rather than a raw wall-clock measurement. The raw measurements and each framework Full are reported in Appendix A.9. Every speedup uses an execution forced to Full at every step inside the same framework and measurement harness. The Full row in each main table uses the corresponding CSRe framework Full latency.

On an RTX 5090D at batch size 2, two warm-up iterations and five randomized, CUDA-event-synchronized repeats give mean denoiser latencies of 1822.875 ms/image for framework all-Full and 1054.967 ms/image for CSRe ($1.728\times$). Embeddings are precomputed; model loading, text encoding, VAE decoding, and file output are excluded. Selected-compute and selected-attention Triton kernels are enabled. The native dense diagnostic is 1386.009 ms/image and is not the denominator of the reported speedup. Peak allocated memory is 28.994 GiB for both framework Full and CSRe. The baseline raw timings remain unchanged; the main-table latency values use their original same-framework ratios and the 1822.875-ms CSRe framework Full reference in (22).

A.9 FRAMEWORK-NATIVE RTX 5090D LATENCY

Table 7 reports measurements before normalization. "Framework Full" forces every denoising step to dense evaluation within the same implementation as the accelerated row. The main tables preserve the resulting same-framework speedup and compute latency using Equation (22).

Table 7: Framework-native RTX 5090D denoiser latency (ms/image).

Model	Batch	Method	Framework Full	Accelerated	Speedup
DiT	18	Full	2242.750±3.613	2242.750±3.613	1.000
DiT	18	FORA (N=3)	1766.031±3.187	690.427±1.204	2.558
DiT	18	FORA (N=2.8)	1766.031±3.187	735.500±1.155	2.401
DiT	18	ToCa (N=4, R=93%)	1692.674±4.091	813.282±0.815	2.081
DiT	18	CSRe	2242.750±3.613	978.435±0.813	2.292
PixArt	32	Full	98.357±0.377	98.357±0.377	1.000
PixArt	32	FORA (N=2)	105.635±0.376	59.897±0.138	1.764
PixArt	32	FORA (N=3)	105.635±0.376	46.100±0.068	2.291
PixArt	32	ToCa (R=60%)	473.312±0.635	343.138±0.437	1.379
PixArt	32	ToCa (R=70%)	473.312±0.635	317.147±0.382	1.492
PixArt	32	ToCa (R=80%)	473.312±0.635	290.119±1.056	1.631
PixArt	32	ECAD (fast)	81.634±0.453	35.663±0.047	2.289
PixArt	32	CSRe	98.357±0.377	72.197±0.130	1.362
PixArt	32	CSRe (early reuse, K_b=20)	128.854±0.791	65.645±0.097	1.963
FLUX	2	Full	1822.875±4.658	1822.875±4.658	1.000
FLUX	2	ToCa (N=4, R=90%)	1521.777±2.645	896.469±14.131	1.698
FLUX	2	FORA (N=3)	1375.460±2.320	531.624±0.155	2.587
FLUX	2	ECAD (fast)	1503.929±2.846	606.082±11.887	2.481
FLUX	2	CSRe	1822.875±4.658	1054.967±5.204	1.728

A.10 ABLATION PROTOCOLS

The DiT-XL/2 component-selection study retains the reported anchor schedule, CFG 1.5, and 250 DDPM steps. All variants use the same 5,000 sample indices, class labels, sampling seed, and batch size 32. Each variant targets 42.40 transformer TFLOPs/image within 0.5%. FID-5K is computed against the same ImageNet reference statistics for every variant.

For component selection, the complete method refreshes two score-selected heads and score-selected MLP channels. The MLP-only control assigns the head-refresh budget to MLP channels; the attention-only control assigns the MLP-refresh budget to attention. Random-head and random-head/channel controls retain the respective refresh counts and dynamic width allocator. Random selection uses an independent random-number generator, holds masks within an anchor interval, and shares each conditional mask only with its paired unconditional sample.

The PixArt-α block-selection study uses the same 5,000 COCO prompts, seed 0, and prompt-indexed deterministic noise for every policy. Full steps are $\{0, 3, 6, 9, 12, 15, 18\}$, and steps 1 and 2 fully reuse the step-0 cache. All policies refresh all 16 heads in each self- and cross-attention module and use the same 8.148-TFLOP MLP budget. All-28 spreads this budget over every block. At each anchor, Top-20 and Bottom-20 rank blocks by completed-anchor hidden-state change and assign MLP refresh to the highest- and lowest-ranked 20 blocks, respectively; the other eight blocks receive zero MLP width and reuse their cached outputs. Random-20 selects 20 blocks using six fixed policy seeds, and Table 5 reports the mean and standard deviation. FID-5K uses JPEG quality 75 and the `toca_official_pixart.npz` reference. COCO CLIP uses the frozen OpenAI CLIP ViT-L/14 evaluator.

A.11 QUALITATIVE COMPARISONS ON FLUX.1-DEV

Figures 8–12 show randomly selected qualitative comparisons.



Figure 8: Randomly selected qualitative comparisons on FLUX.1-dev (1/5).

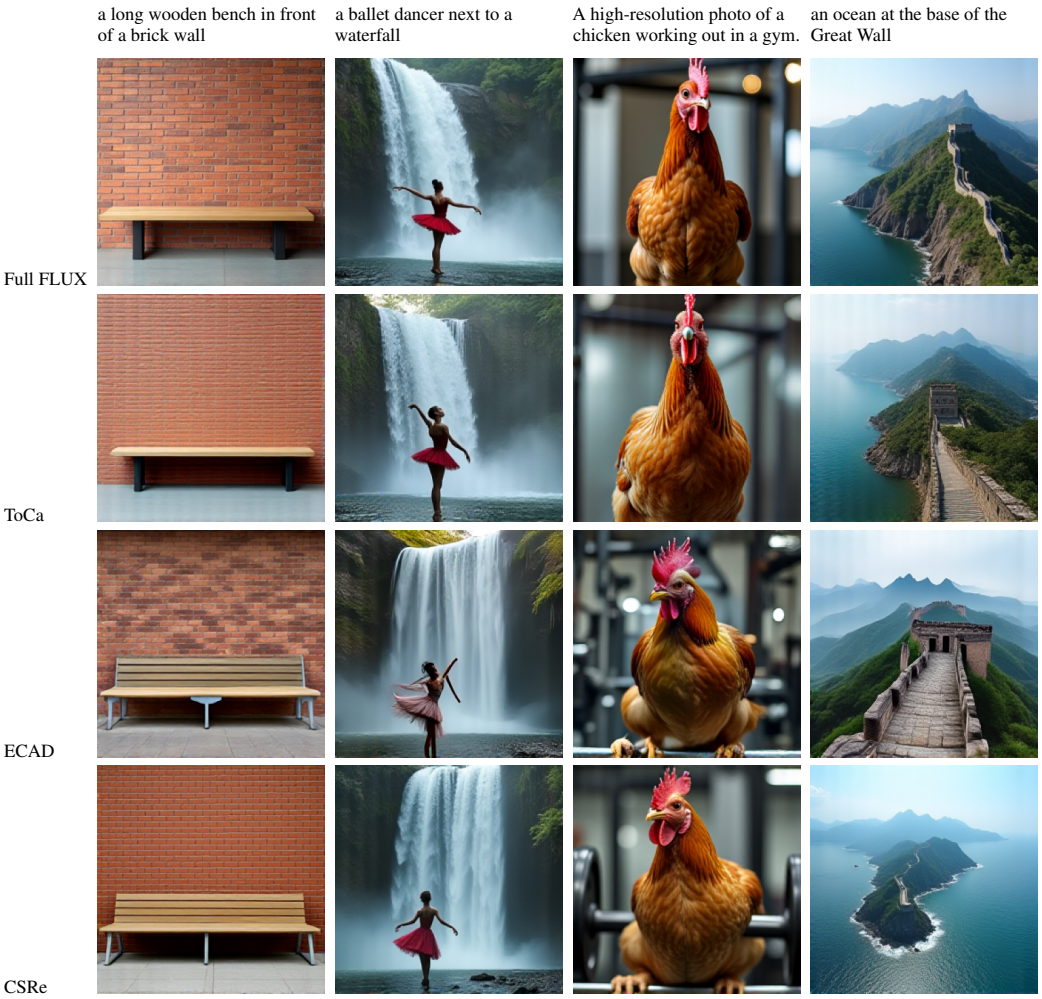


Figure 9: Randomly selected qualitative comparisons on FLUX.1-dev (2/5).

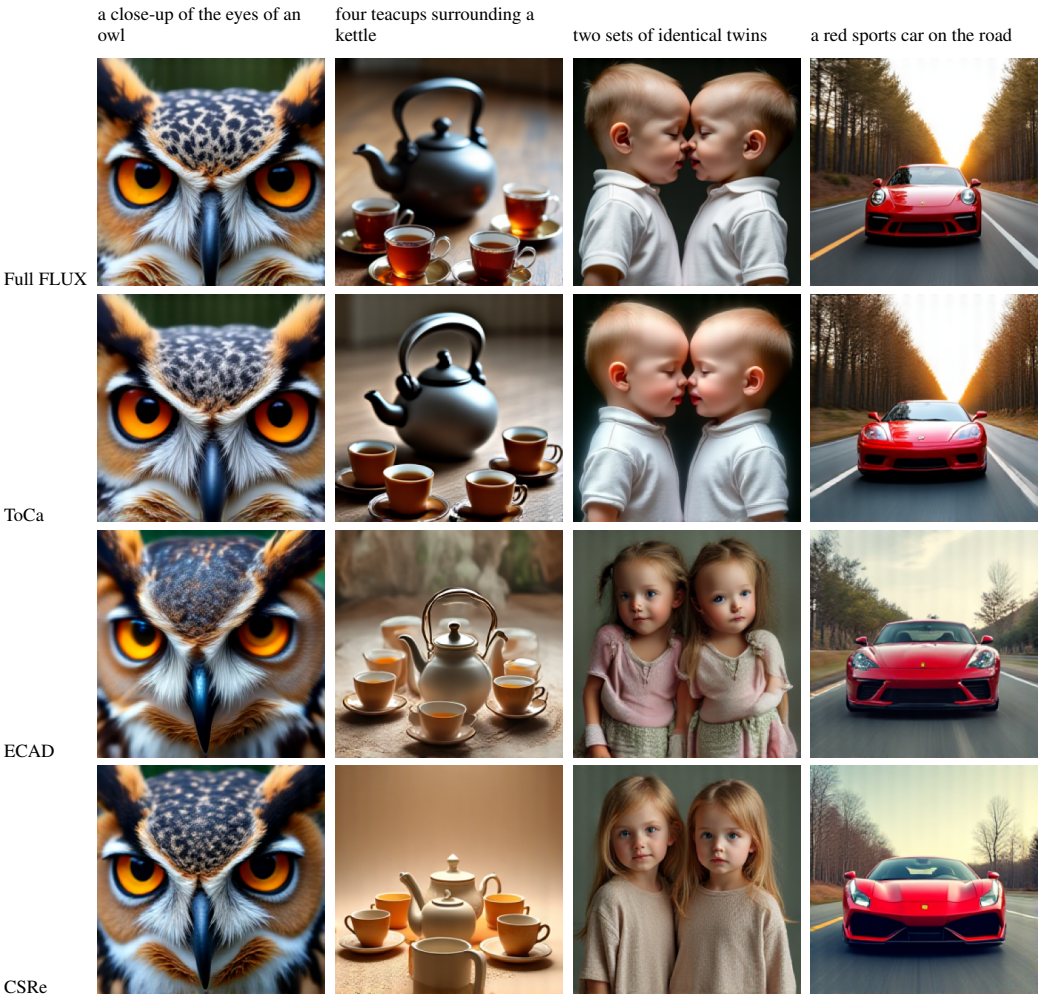


Figure 10: Randomly selected qualitative comparisons on FLUX.1-dev (3/5).

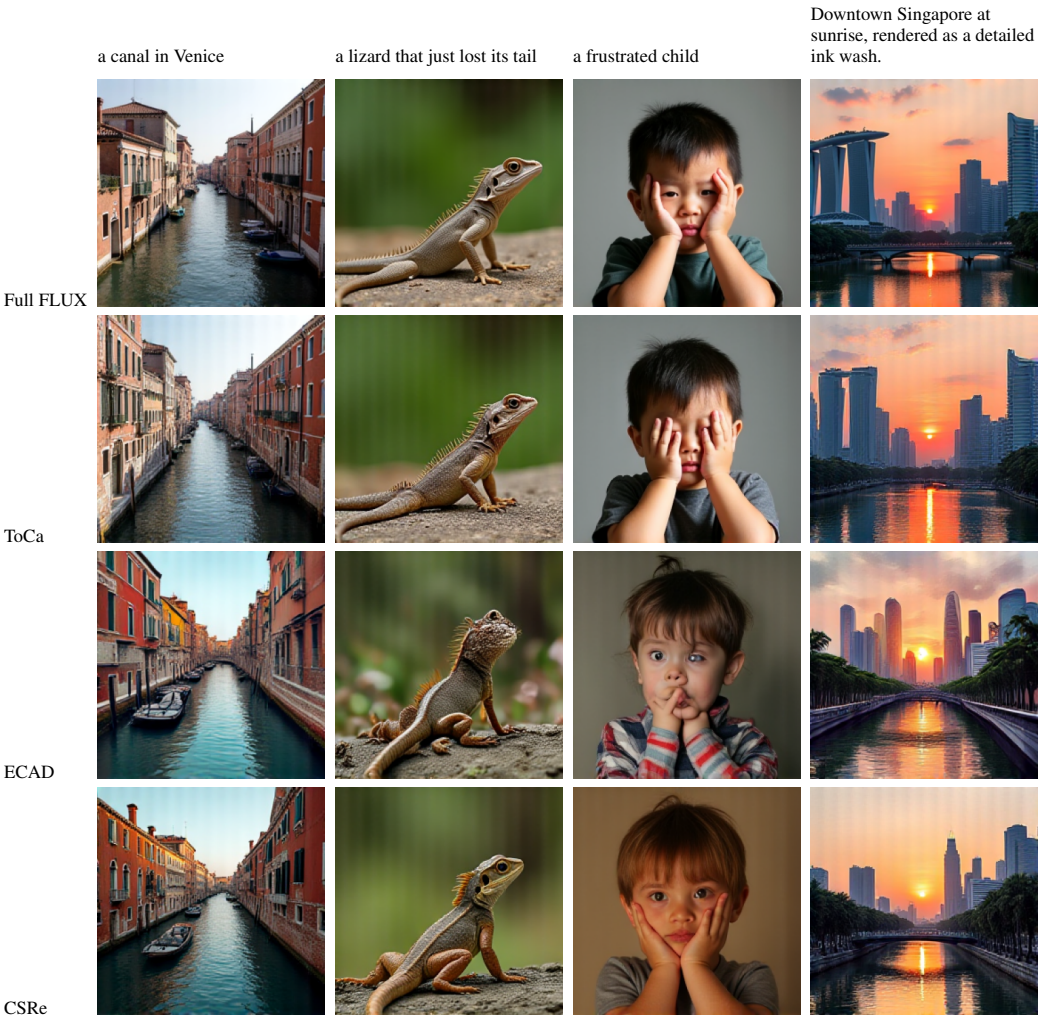


Figure 11: Randomly selected qualitative comparisons on FLUX.1-dev (4/5).



Figure 12: Randomly selected qualitative comparisons on FLUX.1-dev (5/5).