BLOCKDELTA: BLOCK RESIDUAL RECONSTRUCTION FOR FRAME-SPARSE FEW-STEP VIDEO DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-step distillation accelerates video diffusion models, yet each remaining step is still computationally expensive, and on 4-step models, cross-step feature reuse yields limited benefits. Frame-sparse inference alleviates this cost by computing only selected frames (anchors) within each transformer block and reconstructing the skipped frames from them. Its effectiveness hinges on two design choices: (1) what to reconstruct for skipped frames, and (2) how to estimate this target from anchors. We propose BlockDelta, a training-free frame-sparse method that systematically addresses both. For (1), we reconstruct the block residual, the difference between a block's output and input, rather than the full output. Reconstructing the full output implicitly re-estimates the already-known input, introducing unnecessary approximation error. Theoretical analysis and empirical evaluation confirm that residual reconstruction reduces block-wise error and boosts generation quality. For (2), we find that lower local reconstruction error does not necessarily yield higher fidelity to dense inference, i.e., sampling without acceleration. Averaging all anchors has the lowest local error but over-smooths inter-frame differences and leads to larger deviation from dense inference, whereas linear interpolation between the two nearest enclosing anchors preserves temporal structure more faithfully, and BlockDelta uses it. On distilled Wan2.2 and HunyuanVideo-1.5, BlockDelta reduces per-step computation by nearly half and keeps the VBench score within 1.1 points of dense inference, while part of the motion is lost. It also applies to recent audio–video diffusion models.

1 INTRODUCTION

Interactive uses of video diffusion models (VDMs), such as world simulation (He et al., 2025; Team et al., 2026) and iterative content creation (Melnik et al., 2024; Guo et al., 2023), need low generation latency as well as high visual quality. However, the computational cost of VDMs remains a challenge for interactive use: Xi et al. (2025) report that HunyuanVideo (Kong et al., 2024) takes nearly an hour on one A100 GPU to generate a 5-second clip. Distillation techniques (Yin et al., 2025; Huang et al., 2026a; Fan et al., 2026) reduce inference cost by reducing sampling to a few steps, and yet each step still requires a full forward pass of a video DiT over tens of thousands of tokens. To further accelerate these few-step models, we examine existing approaches that reduce computation by using features from previous denoising steps or by computing less within each step.

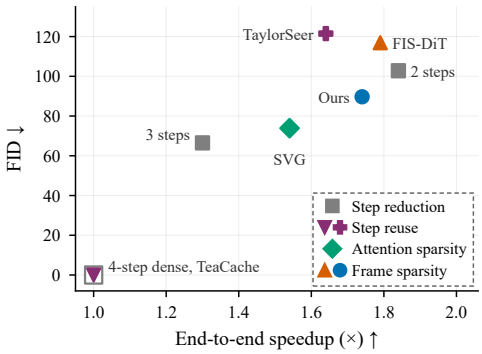


Figure 1: Quality–efficiency comparison on distilled Wan2.2 T2V. Speedup (one H20) and FID (100 prompts) are measured against 4-step dense inference; all methods except step reduction use 4 steps.

On 4-step models, acceleration methods that reuse features from previous steps yield limited gains. Representative examples include TeaCache (Liu et al., 2025a), which reuses cached features across denoising steps, and TaylorSeer (Liu et al., 2025b), which predicts later features from earlier ones. As shown in Figure 1, TeaCache achieves only a 1.00× speedup with an FID of 0 compared to 4-step dense inference (sampling without acceleration), offering no acceleration. TaylorSeer achieves

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

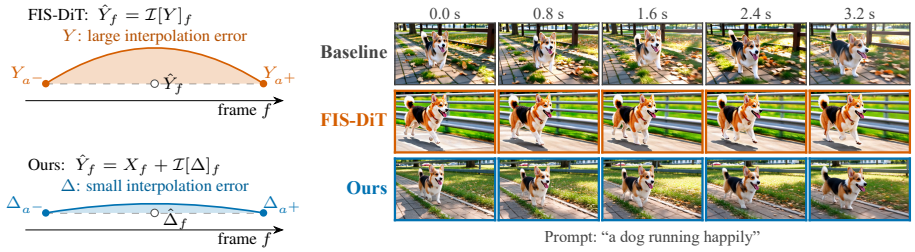


Figure 2: Left: schematic comparison of output and residual interpolation, where X_f, Y_f and $\Delta_f = Y_f - X_f$ denote the block input, output, and residual of frame f. FIS-DiT interpolates anchor outputs to estimate Y_f; we interpolate anchor residuals and add the known input X_f. Right: comparison of video frames from the dense inference baseline, FIS-DiT, and ours, generated with the same seed. Both sparse methods use the same fully computed blocks and anchor-frames.

a 1.64× speedup, whereas 2-step dense sampling is faster (1.84×). Another line of work reduces computation within each step without relying on cross-step feature reuse. SVG (Xi et al., 2025) sparsifies attention but retains the computation in projection and feed-forward layers. FIS-DiT (Tang et al., 2026) employs frame-sparse inference by applying attention, projection, and feed-forward operations only to tokens from selected anchor frames within a DiT block. It reconstructs the outputs of non-anchor frames by interpolating between anchor outputs, achieving a 1.79× speedup. However, output interpolation remains farther from the dense reference than residual interpolation, despite their comparable speedups. This motivates us to investigate whether the fidelity of frame-sparse inference can be improved by revisiting *what* to reconstruct and *how* it is estimated from the anchor frames.

We propose BlockDelta, a training-free frame-sparse method that addresses both questions and specifies which blocks and denoising steps stay dense and which frames serve as anchors. To answer *what to reconstruct*, we establish the block residual, rather than the full output, as the reconstruction target through an exact error comparison. The residual is defined as the difference between a block's output and input. When computation is skipped for a frame, its input remains available, and interpolating the full output as in FIS-DiT re-estimates this known input from the anchors. The reconstructed output then carries the error of this re-estimate in addition to the error of the residual. We instead retain the input and reconstruct only the residual (Figure 2, left), and derive the exact condition under which residual reconstruction has lower error than output reconstruction. This condition requires no modeling assumption and holds on nearly every block of Wan2.2, HunyuanVideo-1.5 and LTX-2.3. With the same dense blocks and anchor frames as FIS-DiT, residual reconstruction also stays closer to dense inference (Figure 2, right).

To answer *how to estimate the missing residual*, we show that lower local reconstruction error does not necessarily lead to higher video fidelity to dense inference. Our theoretical analysis identifies a trade-off: averaging more anchor residuals reduces local error but also suppresses differences across frames. We validate linear interpolation between the two enclosing anchors as an effective balance, achieving higher fidelity than either averaging all anchors or copying a single anchor. The placement of the weights also matters: reversing them or using non-enclosing anchors reduces fidelity.

Our contributions are: (1) We propose BlockDelta, a training-free frame-sparse method that reconstructs the block residual rather than the output of non-anchor frames, justified by an exact, assumption-free error comparison. (2) We show that lowering the local error suppresses the differences across frames and does not necessarily give higher fidelity to dense inference. (3) BlockDelta achieves 1.55–1.74× speedup on distilled Wan2.2 and HunyuanVideo-1.5 within 1.1 VBench points of dense inference, and transfers to LTX-2.3 and MiniMax-H3.

2 Related Work

Methods for accelerating VDMs reduce the number of sampling steps through distillation (Yin et al., 2025; Huang et al., 2026a; Fan et al., 2026), reuse features from previous denoising steps through caching or forecasting (Liu et al., 2025a; Ma et al., 2025; Zhou et al., 2025; Liu et al., 2025b), or reduce computation within each step (Xi et al., 2025; Yang et al., 2025; Tang et al., 2026). For

few-step VDMs, DisCa (Zou et al., 2026) enables cross-step feature reuse through a learned predictor, whereas our method, BlockDelta, reduces computation within each step without additional training. The most closely related work is FIS-DiT (Tang et al., 2026), a frame-sparse method for few-step VDMs that reconstructs block outputs for non-anchor frames by interpolating anchor outputs. BlockDelta instead interpolates block residuals while preserving known inputs, and examines how the reconstruction operator affects fidelity to dense inference. Residual reuse has also been explored in image diffusion by Δ-DiT (Chen et al., 2024), which caches block-group residuals across denoising steps. In contrast, BlockDelta reconstructs block residuals across frames within the same denoising step.

3 PRELIMINARIES

Few-step video diffusion. A VDM generates F latent frames by integrating a learned velocity field from noise level $\sigma = 1$ to 0 over T steps (Lipman et al., 2023; Liu et al., 2023), with few-step distillation reducing T (Yin et al., 2025; Huang et al., 2026a; Fan et al., 2026). The schedule is shifted toward high noise by $\sigma \mapsto \alpha\sigma/[1 + (\alpha - 1)\sigma]$, where $\alpha \geq 1$ (Esser et al., 2024). Each step uses an L-block video DiT with spatio-temporal attention and feed-forward layers (Peebles & Xie, 2023). Wan2.2, HunyuanVideo-1.5 and LTX-2.3 use causal VAEs that encode the first pixel frame separately, while MiniMax-H3 encodes fixed-length clips independently.

Frame-sparse inference. Consider a DiT block $\mathcal{B}$ at one denoising step. The DiT represents each latent frame by S_f spatial tokens. We write X_f for the hidden states of the tokens of latent frame f at the input of $\mathcal{B}$, and call X_f the feature of frame f. The video part of the block input is then $X = \{X_0, \ldots, X_{F-1}\}$; tokens of other modalities, such as text, are not part of X and are always computed in full. Dense execution applies $\mathcal{B}$ to all frames and gives the output $Y = \mathcal{B}(X) = \{Y_0, \ldots, Y_{F-1}\}$. We call $\Delta_f = Y_f - X_f$ the block residual of frame f, which is the change that the block adds to its input. Frame-sparse inference applies the block only to a subset of frames, called anchor frames. We write $\mathcal{A} \subset \{0, \ldots, F-1\}$ for the set of their indices and $\widetilde{Y}_{\mathcal{A}} = \{\widetilde{Y}_a \mid a \in \mathcal{A}\}$ for the outputs that the block computes for them. Note that $\widetilde{Y}_a \neq Y_a$ generally holds, as the tokens of the skipped frames are excluded from the attention computation in $\mathcal{B}$. The outputs of the skipped frames, $f \notin \mathcal{A}$, are then reconstructed from the anchor outputs and the input X by a reconstruction operator $\mathcal{R}$:

$$\hat{Y}_f = \begin{cases} \widetilde{Y}_f, & f \in \mathcal{A}, \\ \mathcal{R}\big(\widetilde{Y}_{\mathcal{A}},\, X\big)_f, & f \notin \mathcal{A}. \end{cases} \tag{1}$$

The inputs of all frames, including the skipped ones, are available before the block runs, and $\mathcal{R}$ can therefore use the full input X. The analysis in Section 4 uses the dense outputs Y_a and does not consider the difference between $\widetilde{Y}_a$ and Y_a; its effect is tested under frame-sparse execution in Sections 5.2 and 5.3, and measured on Wan2.2 in Appendix A.2. Following FIS-DiT (Tang et al., 2026), anchors are selected at a fixed stride along the frame axis (3 unless stated otherwise), the first and last frames are always anchors, and the offset of the stride changes from one block to the next.

4 BLOCKDELTA

The core of BlockDelta is its reconstruction operator $\mathcal{R}$, which is determined by two design choices. The first choice is *what* to reconstruct for a skipped frame: BlockDelta keeps the known input X_f and reconstructs only the block residual Δ_f (Section 4.1). The second choice is *how* to estimate this residual from the anchors: BlockDelta linearly interpolates the residuals of the two anchors that enclose the skipped frame (Section 4.2). Section 4.3 then specifies where and when this reconstruction replaces dense computation, i.e., in which blocks, denoising steps and frames. Figure 3 shows the overall procedure.

4.1 WHAT TO RECONSTRUCT: THE BLOCK RESIDUAL

For a skipped frame, the input X_f is known and only the residual Δ_f is missing. BlockDelta therefore reconstructs Δ_f and adds it to X_f. Output reconstruction, used by FIS-DiT, instead estimates the whole output $Y_f = X_f + \Delta_f$ from the anchors. It thus also estimates X_f, although X_f is

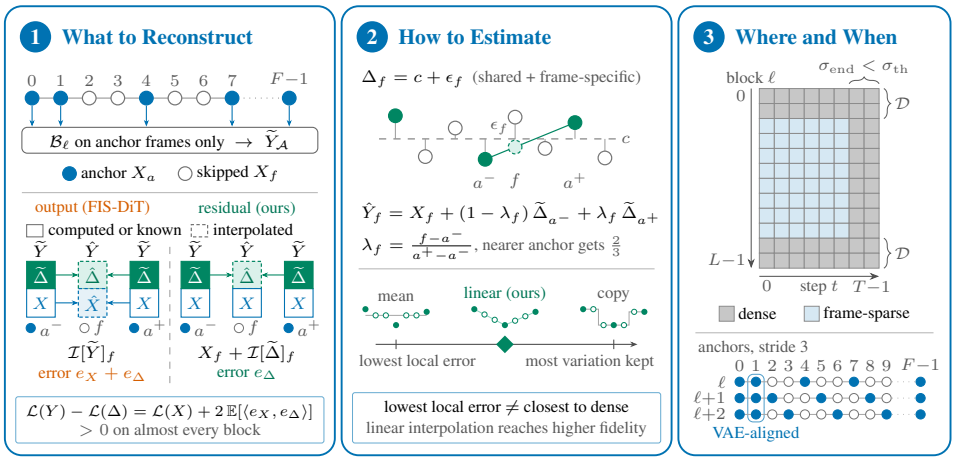


Figure 3: Overview of BlockDelta. (1) Output reconstruction (FIS-DiT) re-estimates the known input X_f of a skipped frame and adds its error e_X; BlockDelta keeps X_f and interpolates only the residual. (2) Averaging more anchors lowers the local error but removes frame-specific variation; BlockDelta interpolates linearly between the two enclosing anchors. (3) Blocks in the dense-block set $\mathcal{D}$ and the σ-aligned dense steps are computed on all frames. In the other blocks, anchors are placed at a fixed stride whose offset changes from block to block. Frames 0 and $F-1$ are anchors in every block, and frame 1 is a VAE-aligned anchor, since frame 0 encodes only the first pixel frame.

already known. The error of this estimate enters the block output and is passed to all later blocks. To compare the two targets, Δ_f and Y_f, we apply the same interpolation to every feature. For an arbitrary frame feature Z, the interpolation $\mathcal{I}$ estimates frame f from the anchors only:

$$\mathcal{I}[Z]_f = \sum_{a \in \mathcal{A}} w_a(f)\, Z_a, \qquad Z \in \{X, Y, \Delta\}, \tag{2}$$

where Z_a is the feature Z of anchor a, and $w_a(f)$ is the weight of anchor a in the estimate of frame f (specified in Section 4.2). Since the weights are the same for all features, the two targets differ only in which feature is interpolated. They give the following two reconstruction operators $\mathcal{R}$ in Eq. (1):

$$\mathcal{R}_{\text{output}}\big(\widetilde{Y}_\mathcal{A}, X\big)_f = \mathcal{I}[\widetilde{Y}]_f, \qquad \mathcal{R}_{\text{residual}}\big(\widetilde{Y}_\mathcal{A}, X\big)_f = X_f + \mathcal{I}[\widetilde{\Delta}]_f, \tag{3}$$

where $\widetilde{\Delta}_a = \widetilde{Y}_a - X_a$ is the residual of anchor a under frame-sparse execution.

We determine the superior target by analyzing the local reconstruction error on dense features. For a skipped frame X_f with $f \notin \mathcal{A}$, let $e_Z = Z_f - \mathcal{I}[Z]_f$ be the reconstruction error of Z. We call $\mathcal{L}(Z) = \mathbb{E}[\|e_Z\|^2]$ the local error of Z, where the expectation is over prompts at a fixed block and denoising step. Since every frame has $Y_f = X_f + \Delta_f$ and $\mathcal{I}$ uses the same weights for all features, interpolation preserves this sum, $\mathcal{I}[Y]_f = \mathcal{I}[X]_f + \mathcal{I}[\Delta]_f$. The output error is therefore the sum of the input error and the residual error, $e_Y = e_X + e_\Delta$. Taking the expected squared norm of both sides gives

$$\mathcal{L}(Y) - \mathcal{L}(\Delta) = \mathcal{L}(X) + 2\,\mathbb{E}[\langle e_X, e_\Delta \rangle], \tag{4}$$

which is exact and requires no assumption on the features. The term $\mathcal{L}(X)$ is the error of estimating the known input X_f from the anchors, which output reconstruction incurs and residual reconstruction avoids. The inner product $\mathbb{E}[\langle e_X, e_\Delta \rangle]$ measures whether the input error and the residual error point in similar directions. Residual reconstruction therefore has the lower local error exactly when the right-hand side is positive. This fails only if the two errors point in largely opposite directions, $\mathbb{E}[\langle e_X, e_\Delta \rangle] \le -\mathcal{L}(X)/2$. The same identity applies under frame-sparse execution, with $\widetilde{Y}_a$ and $\widetilde{\Delta}_a$ in place of Y_a and Δ_a (Appendix A.2). In video DiTs, the right-hand side is positive on almost every block: with stride-3 anchors, on 39 of 40 blocks of Wan2.2, all 54 blocks of HunyuanVideo-1.5 and 47 of 48 blocks of LTX-2.3 (Appendix A.2). On Wan2.2, the interpolation error of the output is 2.17 times that of the residual over all blocks (Figure 4(a)).

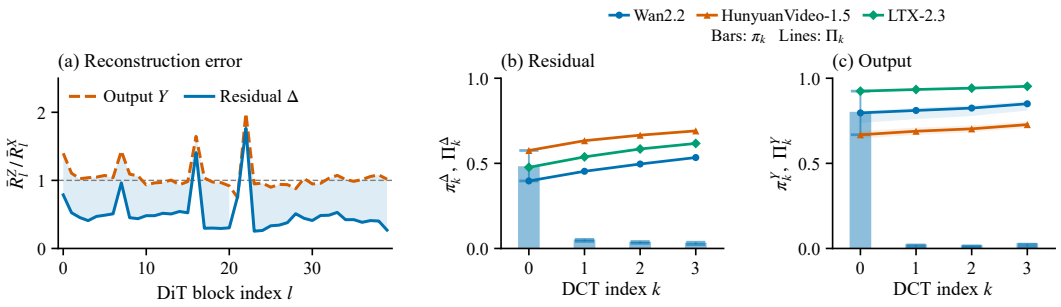


Figure 4: Frame-axis measurements of the output Y and the residual Δ. **(a)** Interpolation error of $Z \in \{Y, \Delta\}$ at block l of Wan2.2, divided by that of the input X. **(b)–(c)** Energy share of each DCT coefficient along the frame axis (π_k^Z) and of the first $k+1$ coefficients (Π_k^Z); coefficient 0 is the component shared by all frames. Definitions are in Appendix A.1.

4.2 How to Estimate the Residual: Interpolation Weights

The weights $w_a(f)$ with the lowest local error do not give the video closest to dense inference, because they also remove the differences between frames. To analyze this trade-off, we decompose the residual Δ_f into a shared component c (common to all frames) and a frame-specific component ϵ_f (panel 2 of Figure 3):

$$c = \frac{1}{F-1} \sum_{g=1}^{F-1} \Delta_g, \qquad \epsilon_f = \Delta_f - c, \tag{5}$$

where both are computed for one prompt at a fixed block and denoising step. By construction, $\Delta_f = c + \epsilon_f$ and $\sum_{f=1}^{F-1} \epsilon_f = 0$. The sums exclude the first latent frame X_0, because the VAE encodes the first pixel frame alone into X_0, and X_0 differs from the other frames, for example in the attention it receives (Figure 6 in Appendix A.1). Since BlockDelta always keeps X_1 as an anchor (Section 4.3), no skipped frame is interpolated from X_0, and X_0 is therefore not used in reconstruction. The analysis below thus uses only the K anchors among $X_1, \ldots, X_{F-1}$, with $F \geq 3$ and $1 \leq K \leq F-2$, and a skipped frame X_f with $f \geq 1$.

Figure 4(b) supports this decomposition. It applies a Discrete Cosine Transform (DCT) to $\Delta_1, \ldots, \Delta_{F-1}$ along the frame axis. The first DCT coefficient carries exactly the energy of c, and the other coefficients carry the energy of ϵ_f. The first coefficient carries 40–58% of the residual energy on Wan2.2, HunyuanVideo-1.5 and LTX-2.3, and the remaining energy is spread almost evenly over the other coefficients. The residual thus has a shared part, which a few anchors can estimate, and a frame-specific part with little structure along the frame axis. The output has an even larger first-coefficient share of 65–92% (Figure 4(c)), yet its interpolation error is larger (Section 4.1). By Eq. (4), the error gap depends on the input error and the cross term, not on this share.

A residual estimate should meet two requirements: a low local error $\mathcal{L}(\Delta)$, and enough frame-specific variation to keep the skipped frames different from each other. With the total weight $s = \sum_{a \in \mathcal{A}} w_a(f)$ and $\Delta_a = c + \epsilon_a$, the estimate is $\mathcal{I}[\Delta]_f = s \cdot c + \mathcal{I}[\epsilon]_f$. With $s = 1$, the shared component is reconstructed exactly, and the error $\hat{\epsilon}_f - \epsilon_f$ comes only from the frame-specific part $\hat{\epsilon}_f = \mathcal{I}[\epsilon]_f$. We call $\nu^2 = \mathbb{E}[\|\epsilon_f\|^2]$ the frame-specific energy and $\hat{\nu}^2 = \mathbb{E}[\|\hat{\epsilon}_f\|^2]$ the retained energy. Weights chosen for only one requirement fail the other (panel 2 of Figure 3), and which one fails depends on how concentrated the weights are. Spreading the weights evenly, e.g., the mean of all K anchors, gives a low local error but a small $\hat{\nu}^2$, and every skipped frame then receives a residual close to c. Concentrating the weights on one anchor, e.g., copying it, keeps the full energy ν^2, but this variation belongs to the anchor, not to frame f, and the local error is high.

To formalize this, we introduce the weight concentration $q_w = \sum_{a \in \mathcal{A}} w_a(f)^2$ for position-only weights, which are fixed nonnegative weights that depend only on the positions of the skipped frame and the anchors. Examples are linear interpolation, the mean of all anchors, and copying the nearest anchor. For $s = 1$, q_w ranges from $1/K$ for the mean of all K anchors to 1 for copying one anchor. To relate q_w to the two requirements, we make the following assumption, which idealizes the observation of Figure 4(b) that the frame-specific part has little structure along the frame axis.

Assumption 1 (Equal energy and equal correlation) *For all analyzed frames f and $g \neq f$: **(A1)** the frame-specific components have equal energy, $\mathbb{E}[\|\epsilon_f\|^2] = \nu^2 > 0$, and are uncorrelated with the shared component, $\mathbb{E}[\langle c, \epsilon_f \rangle] = 0$; **(A2)** every pair of distinct frames has the same correlation, $\mathbb{E}[\langle \epsilon_f, \epsilon_g \rangle] = -\gamma\nu^2$ with $\gamma = 1/(F-2)$.*

Taking the expected inner product of ϵ_f with $\sum_g \epsilon_g = 0$ gives $\nu^2 - (F-2)\gamma\nu^2 = 0$, and this γ is therefore the only value consistent with A2. Under Assumption 1, the frame-specific component has a flat DCT spectrum, as in Figure 4(b). Appendix A.2 measures departures from Assumption 1.

Under Assumption 1, the local error and the retained energy depend on the weights only through s and q_w. For $s = 1$, which reconstructs the shared component exactly, this gives the following result, derived in Appendix B together with the case $s < 1$.

Proposition 1 (Local error and retained energy) *Under Assumption 1, for any anchor index set $\mathcal{A}$ and skipped frame X_f with $f \notin \mathcal{A}$, every position-only estimate with $s = 1$ has $\mathcal{L}(\Delta) = \nu^2(1+\gamma)(1+q_w)$ and $\hat{\nu}^2 = \nu^2[(1+\gamma)q_w - \gamma]$. Hence*

$$\mathcal{L}(\Delta) - \hat{\nu}^2 = \nu^2(1+2\gamma). \tag{6}$$

Since the difference in Eq. (6) is constant, any reduction of the local error removes the same amount of retained energy. Proposition 1 also quantifies the two ends described above. The mean of all anchors, with the smallest $q_w = 1/K$, has the lowest local error and the lowest retained energy $\hat{\nu}^2 = \nu^2[(1+\gamma)/K - \gamma]$. Copying one anchor, with $q_w = 1$, retains the full energy ν^2 but has the highest local error $2\nu^2(1+\gamma)$. The conflict resembles the perception–distortion trade-off in image restoration (Blau & Michaeli, 2018), where a lower distortion can make outputs less similar to natural images. It is also not limited to position-only weights. For the least-squares estimate in any class of estimators closed under scaling, the error and the retained energy sum to ν^2 (Lemma 1 in Appendix B), and this estimate therefore retains less than ν^2 unless it is exact.

A good weighting therefore lies between these two ends. Proposition 1 depends on the weights only through q_w and does not determine which anchors should receive the larger weights. This placement matters through the correlation between the estimate $\hat{\epsilon}_f$ and the true component ϵ_f. With $s = 1$, the local error is $\mathcal{L}(\Delta) = \nu^2 + \hat{\nu}^2 - 2\,\mathbb{E}[\langle \hat{\epsilon}_f, \epsilon_f \rangle]$, and at a fixed retained energy, a larger correlation gives a lower local error. Since $\hat{\epsilon}_f = \sum_{a \in \mathcal{A}} w_a(f)\,\epsilon_a$, the correlation is $\sum_{a \in \mathcal{A}} w_a(f)\,\mathbb{E}[\langle \epsilon_a, \epsilon_f \rangle]$, and it depends on how strongly each anchor a is correlated with frame f. A2 makes all these terms equal, which is why Proposition 1 does not depend on the placement. On Wan2.2, however, the correlation between frame-specific components decreases with the distance between frames (Appendix A.2). For a fixed pair of anchors and a fixed q_w, giving the larger weight to the nearer anchor therefore raises the correlation without changing the retained energy, and lowers the local error.

BlockDelta therefore uses linear interpolation between the two enclosing anchors. Let $a^-, a^+ \in \mathcal{A}$ be the indices of the nearest anchors before and after X_f, with $a^- < f < a^+$. BlockDelta sets

$$\hat{Y}_f = X_f + (1-\lambda_f)\widetilde{\Delta}_{a^-} + \lambda_f \widetilde{\Delta}_{a^+}, \qquad \lambda_f = \frac{f - a^-}{a^+ - a^-}. \tag{7}$$

The two weights sum to $s = 1$, which reconstructs the shared component exactly, and the nearer anchor receives the larger weight. With stride 3, the weights are 1/3 and 2/3, and $q_w = 5/9$ lies between the two ends $1/K$ and 1. Section 5.3 compares this choice with other position-only weights.

4.3 Where and When to Apply Frame Sparsity

Frame sparsity is not applied to every computation. Three fixed sparse execution rules keep some blocks, denoising steps and frames dense: the dense-block set, the σ-aligned dense steps and the VAE-aligned anchors. Algorithm 1 in Appendix C gives the complete procedure.

Dense-block set. The dense-block set $\mathcal{D}$ is a fixed set of blocks that are computed on all frames at every step. Following FIS-DiT, $\mathcal{D}$ contains the first and last few blocks (Appendix D).

σ-aligned dense steps. Errors made in the last denoising steps are carried almost unchanged into the final video, and these steps are therefore computed densely in every block. A step is dense when the noise level at its end is below a threshold σ_{th}. Since the models use different noise schedules,

the rule is defined by the noise level rather than by the number of steps, and one threshold applies to all models (Appendix D).

VAE-aligned anchors. Anchors are also placed where the VAE encodes frames separately, and no skipped frame is then interpolated across such a boundary. The causal VAEs of Wan2.2, HunyuanVideo-1.5 and LTX-2.3 encode the first pixel frame alone into the first latent frame X_0, whereas each other latent frame encodes several pixel frames (Section 3). X_0 also receives 23–93% more attention than the uniform share $1/F$, while every other frame stays within 21.1% of it (Figure 6 in Appendix A.1). Interpolating a skipped frame from X_0 would carry this difference into the estimate. On these models, X_1 is therefore an anchor in every sparse block. MiniMax-H3 encodes fixed-length clips separately, and its anchors are placed at the clip boundaries (Appendix E.2).

Optional corrector. To reduce the remaining difference from dense inference, we also evaluate an optional learned corrector: a low-rank adapter on the last dense blocks that is trained to match the dense velocity prediction while the base weights stay frozen. The corrector is not part of the training-free configuration; Appendix F reports its performance and training details.

5 Experiments

5.1 Experimental Settings

Baseline and dataset. We evaluate BlockDelta on the Wan2.2-A14B text-to-video (T2V) and image-to-video (I2V) models (Wan et al., 2025), distilled to 4 denoising steps with the LightX2V Seko-V2.0 and Seko-V1.0 LoRAs, respectively[1], and on the 8-step-distilled HunyuanVideo-1.5 I2V model (Wu et al., 2025), whose released step-distilled checkpoint supports only I2V. The main comparison follows the full VBench protocol (Huang et al., 2023), with 6,220 T2V and 5,590 I2V videos per method. Ablations use a fixed subset of 100 VBench prompts, unless stated otherwise. Results of BlockDelta on LTX-2.3 (HaCohen et al., 2026) and MiniMax-H3 (MiniMax, 2026) are in Appendix E.2. Visualizations are provided in Appendix K.

Compared methods. We compare BlockDelta with dense inference and with two existing within-step acceleration methods: SVG (Xi et al., 2025), which sparsifies attention, and FIS-DiT (Tang et al., 2026), which reconstructs the outputs of skipped frames by interpolating anchor outputs. Step reuse is not included in Table 1. On the 4-step Wan2.2 models, it is either no faster than dense inference or further from dense inference than using fewer steps (Figure 1 and Appendix G), and we therefore compare only within-step methods; Appendix G compares BlockDelta with step reduction.

Metrics. We measure efficiency, video quality and fidelity to dense inference. Efficiency is measured by the DiT PFLOPs per video and by the DiT time (DiT) and end-to-end time (E2E) per video in seconds, all on one NVIDIA H20. Video quality is measured by the VBench total score with the official T2V or I2V aggregation. Fidelity to dense inference is measured by FID and FVD, which compare the distributions of generated and dense videos, and by LPIPS to the dense video with the same prompt and seed. The ablations add two pixel-level metrics: Head-flash, the largest luminance jump in the first frames, and Dyn., the frame-to-frame change relative to the same-seed dense video (dense inference at 100), which flicker also raises. All methods of a model share prompts, seeds, resolution and sampling settings. Appendices D and D.3 give the definitions.

5.2 Main Results

Attention sparsity *vs.* frame sparsity. We first compare the attention-sparse method (SVG) with the frame-sparse methods in Table 1. Frame sparsity reduces the DiT PFLOPs per video from 5.97–6.71 to 2.70–3.46 and reaches 1.55–1.80× E2E speedup, compared with 1.31–1.54× for SVG. SVG computes every block on every frame and has lower FID and FVD than every frame-sparse method. On Wan2.2, however, BlockDelta has nearly the same LPIPS as SVG (0.5916 vs. 0.5907, 0.2398 vs. 0.2405); the gap lies in FID and FVD. The gap is relatively larger in FVD, which also measures temporal change, than in single-frame FID, suggesting that frame sparsity tends to lose temporal detail, since interpolation cannot recover the frame-specific residual component (Section 4.2).

[1] https://huggingface.co/lightx2v/Wan2.2-Lightning

Table 1: Results of few-step video diffusion models on full VBench. Gray rows are dense inference; the reseed row uses different seeds and shows the change under a seed change alone. Dashes mark values that are trivial or equal to those of dense inference. DiT and E2E are times per video in seconds, with speedups over dense inference in parentheses. VBench is multiplied by 100. *PFLOPs averaged over inputs, as SVG selects its masks per input. †Our reproduction (Appendix C), as the official code is not public. **Bold** marks the best frame-sparse value of each model.

Sparsity Type	Method	Efficiency			Quality	Fidelity		
		PFLOPs ↓	DiT (s) ↓	E2E (s) ↓	VBench ↑	FID ↓	FVD ↓	LPIPS ↓
Wan2.2-A14B (2025) (T2V, 81 frames, 480p, T=4)								
None	Dense inference	6.71	69.4 (1.00×)	76.5 (1.00×)	82.36	—	—	—
	Dense inference (reseed)	—	—	—	82.32	3.45	24.4	0.6671
Attention	SVG (2025)	4.30*	42.3 (1.64×)	49.6 (1.54×)	83.20	4.98	45.4	0.5907
Frame	FIS-DiT† (2026)	**3.32**	**35.4** (1.96×)	**42.7** (1.79×)	81.38	22.22	470.1	0.6599
	BlockDelta (FIS-DiT settings)	**3.32**	35.8 (1.94×)	42.9 (1.78×)	**82.59**	9.83	174.6	0.5973
	BlockDelta	3.46	36.9 (1.88×)	44.0 (1.74×)	82.49	**8.31**	**152.6**	**0.5916**
Both	BlockDelta + SVG	2.47*	25.9 (2.68×)	33.1 (2.31×)	83.41	11.72	190.4	0.6351
Wan2.2-A14B (2025) (I2V, 81 frames, 480p, T=4)								
None	Dense inference	6.38	65.9 (1.00×)	76.6 (1.00×)	88.05	—	—	—
	Dense inference (reseed)	—	—	—	88.10	0.58	4.7	0.2248
Attention	SVG (2025)	4.12*	40.4 (1.63×)	51.3 (1.49×)	88.34	2.42	27.0	0.2405
Frame	FIS-DiT† (2026)	**3.17**	**33.8** (1.95×)	44.8 (1.71×)	70.76	167.48	2667.5	0.6355
	BlockDelta (FIS-DiT settings)	**3.17**	**33.8** (1.95×)	**44.7** (1.71×)	86.96	5.37	202.0	0.2581
	BlockDelta	3.29	35.2 (1.87×)	46.1 (1.66×)	**87.16**	**4.54**	**157.0**	**0.2398**
Both	BlockDelta + SVG	2.37*	24.9 (2.65×)	35.7 (2.15×)	86.94	6.42	346.3	0.2719
HunyuanVideo-1.5 (2025) (I2V, 81 frames, 480p, T=8)								
None	Dense inference	5.97	62.5 (1.00×)	74.6 (1.00×)	90.17	—	—	—
	Dense inference (reseed)	—	—	—	90.08	0.75	4.3	0.2385
Attention	SVG (2025)	3.98*	45.0 (1.39×)	57.0 (1.31×)	90.16	0.48	2.2	0.0525
Frame	FIS-DiT† (2026)	**2.70**	**29.3** (2.13×)	**41.5** (1.80×)	69.42	168.58	2514.7	0.6261
	BlockDelta (FIS-DiT settings)	**2.70**	29.9 (2.09×)	42.1 (1.77×)	88.57	7.13	384.9	0.2344
	BlockDelta	3.27	35.8 (1.75×)	48.0 (1.55×)	**89.11**	**4.01**	**66.6**	**0.2029**
Both	BlockDelta + SVG	2.25*	27.4 (2.28×)	39.5 (1.89×)	89.16	3.57	56.9	0.2028

Analysis of frame-sparse methods. We next compare the frame-sparse methods in pairs to separate the effect of the reconstruction target from that of the sparse execution rules. First, we isolate the effect of residual reconstruction by comparing BlockDelta (FIS-DiT settings) with FIS-DiT. Both use the same dense blocks, dense steps and anchors, and they have the same PFLOPs and nearly the same DiT and E2E times. Residual reconstruction improves every quality and fidelity metric on all three models. On Wan2.2 T2V, Wan2.2 I2V and HunyuanVideo-1.5 I2V, FID decreases from 22.22, 167.48 and 168.58 to 9.83, 5.37 and 7.13, FVD decreases from 470.1, 2667.5 and 2514.7 to 174.6, 202.0 and 384.9, and VBench increases by 1.21, 16.20 and 19.15 points. Reconstructing the residual thus improves quality at the same cost, as the lower local error of the residual in Section 4.1 predicts. We then evaluate the sparse execution rules by replacing the dense steps and anchors of FIS-DiT with the σ-aligned dense steps and VAE-aligned anchors of Section 4.3. With these two rules, FID and FVD decrease on all three models, most on HunyuanVideo-1.5 I2V, by 3.12 and 318.3, at 0.04–0.22× lower E2E speedup.

Combining frame and attention sparsity. We then test whether frame and attention sparsity can be combined. BlockDelta + SVG uses the sparse attention masks of SVG in BlockDelta and is the fastest method on all three models, with E2E speedups of 2.31×, 2.15× and 1.89×. Its VBench is at most 0.22 points below that of BlockDelta. Its FVD decreases on HunyuanVideo-1.5 I2V but increases on Wan2.2, from 152.6 to 190.4 on T2V and from 157.0 to 346.3 on I2V. The two kinds of sparsity can thus be combined for a larger speedup at similar VBench, with a fidelity cost on Wan2.2, where the skipped frames likely inherit the change that SVG makes to their anchors (Appendix E.1).

5.3 Ablation Studies

We conduct two ablations. The first is a leave-one-out ablation on HunyuanVideo-1.5 I2V (Wan2.2 in Appendix H) that removes each of the four components: residual interpolation, the dense-block set, the σ-aligned dense steps and the VAE-aligned anchors. The second changes only the interpolation weights.

Table 2: Ablation on HunyuanVideo-1.5 I2V. Each w/o row removes one component from BlockDelta (last row) and keeps the other three. Gray rows are dense inference methods. FIS-DiT keeps the dense-block set but has none of the other three components. All rows use the same 100 prompts.

Configuration	FID ↓	FVD ↓	Head-flash ↓	Dyn. (%)	DiT s ↓
Dense inference	—	—	0.63	100	62.5
Dense inference (reseed)	16.4	200.2	0.58	101.1	—
FIS-DiT	219.4	2812.1	12.86	220.6	29.3
w/o residual interpolation (output interpolation)	120.9	1394.7	1.09	75.9	35.4
w/o dense-block set (no dense blocks)	202.3	1852.4	13.21	103.2	30.3
w/o σ-aligned dense steps (last step only)	27.1	634.3	1.01	108.2	31.3
w/o VAE-aligned anchors	20.7	314.2	1.99	95.4	34.5
BlockDelta	19.9	283.5	0.90	93.3	35.8

Components of BlockDelta. We remove each component from BlockDelta in turn (Table 2). (1) Without residual interpolation, FID rises from 19.9 to 120.9 at almost the same DiT time. (2) Without the dense-block set, FID rises to 202.3, close to the 219.4 of FIS-DiT, and FVD to 1852.4; Head-flash rises to 13.21, and the rise of Dyn. to 103.2 thus comes from flicker, not recovered motion. (3) Without the σ-aligned dense steps, FVD rises from 283.5 to 634.3 and Dyn. to 108.2. (4) Without the VAE-aligned anchors, Head-flash in the first frames rises from 0.90 to 1.99, against 0.63 for dense inference. Every component thus improves fidelity to dense inference; residual interpolation and the dense-block set matter most, and neither suffices alone.

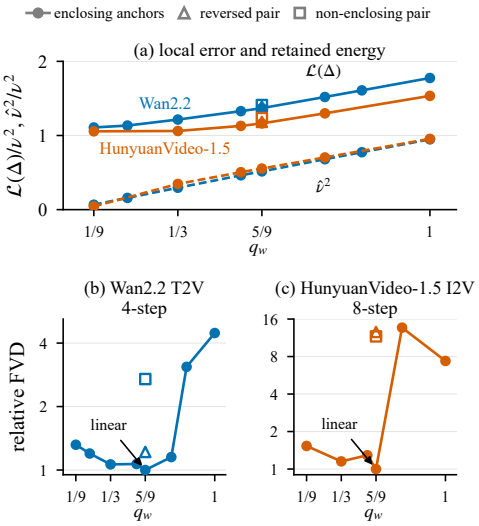


Figure 5: Position-only weights on Wan2.2 T2V and HunyuanVideo-1.5 I2V ($s = 1$). (a) Local error $\mathcal{L}(\Delta)$ and retained energy $\hat{\nu}^2$ over ν^2. (b) and (c) FVD over the lowest FVD per model (log scale). Filled circles go from the mean of all nine anchors ($q_w = 1/9$) to copying the nearest anchor ($q_w = 1$).

Interpolation weights. We change only the weights, with total weight $s = 1$ (Section 4.2), on Wan2.2 T2V and HunyuanVideo-1.5 I2V (Figure 5). As the weight concentration q_w decreases, the local error and the retained energy $\hat{\nu}^2$ decrease together, as Proposition 1 predicts (Figure 5(a)). The mean of all nine anchors ($q_w = 1/9$) has the lowest local error, but its FVD is 1.32 and 1.53 times that of linear interpolation on the two models. Copying the nearest anchor ($q_w = 1$), which retains the most energy, has 4.46 and 7.37 times. Neither end thus gives the lowest FVD. At the same $q_w = 5/9$, weight placement also matters, as Section 4.2 predicts: reversing the two weights, giving 2/3 to the farther anchor, gives 1.22 and 12.60 times the FVD of linear interpolation, and moving them to two non-enclosing anchors gives 2.70 and 11.61 times. Linear interpolation thus has the lowest FVD on both models but not the lowest local error.

6 Conclusion

We propose BlockDelta, a training-free frame-sparse method for few-step video diffusion that keeps the known input of each skipped frame and interpolates only its block residual between the two enclosing anchors. An exact error comparison establishes the residual as the better reconstruction target, and our analysis shows that fidelity to dense inference depends on retaining differences across frames, and the weights with the lowest local error are therefore not the most faithful. On distilled Wan2.2 and HunyuanVideo-1.5, BlockDelta gives 1.55–1.74× end-to-end speedup within 1.1 VBench points of dense inference, and transfers to LTX-2.3 and MiniMax-H3; recovering the frame-specific variation that interpolation cannot reconstruct is the main direction for future work.

7 AI USE STATEMENT

In this work, we used generative AI tools to implement methods and experiment code, to give feedback on experiment design, to assist in writing proofs and refining hypotheses, and to help interpret results. We have not used generative AI tools to generate synthetic data sets or datasets, and translation is not applicable to this work. Additionally, we used generative AI tools to edit the paper for readability and create figures. We have reviewed all AI-assisted work: all derivations were checked by the authors, all hypotheses were tested against measurements, and LLM-generated code was verified and tested for correctness by 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

8 REPRODUCIBILITY STATEMENT

Appendix C gives pseudocode for both reconstruction operators and for the frame-sparse forward pass. Section 5 and Appendix D list the checkpoints, dense-block sets, noise threshold, prompt sets, seeds, metric implementations and timing protocol, and Appendix D.3 defines the pixel-level metrics. Appendix A.1 defines the frame-axis measurements. The supplementary material contains the videos of Appendix K and Figure 7. All methods of a model share prompts, seeds, resolution and sampling settings.

REFERENCES

Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6228–6237. IEEE, 2018.

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*, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Xiangyu Fan, Zesong Qiu, Zhuguanyu Wu, Fanzhou Wang, Zhiqian Lin, Tianxiang Ren, Dahua Lin, Ruihao Gong, and Lei Yang. Phased dmd: Few-step distribution matching distillation via score matching within subintervals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41667–41676, 2026.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023.

Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, et al. Ltx-2: Efficient joint audio-visual foundation model. *arXiv preprint arXiv:2601.03233*, 2026.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, et al. Matrix-game 2.0: An open-source real-time and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *Advances in Neural Information Processing Systems*, 38:167283–167308, 2026a.

Xuyue Huang, Zhe Chen, Wang Shen, and Xiao-Ping Zhang. Learnibridge: Learnable calibration of feature caching for diffusion models acceleration. *arXiv preprint arXiv:2606.26778*, 2026b.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models, 2023. URL https://arxiv.org/abs/2311.17982.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7353–7363, June 2025a.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers, 2025b. URL https://arxiv.org/abs/2503.06923.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. Magcache: Fast video generation with magnitude-aware cache. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 34348–34380. Curran Associates, Inc., 2025. doi: 10.52202/085713-1153. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/311207bb626e36a8f1d3eb92aa67af22-Paper-Conference.pdf.

Andrew Melnik, Michal Ljubljanac, Cong Lu, Qi Yan, Weiming Ren, and Helge Ritter. Video diffusion models: A survey. *arXiv preprint arXiv:2405.03150*, 2024.

MiniMax. MiniMax H3: An open model breaking the boundaries between tasks and modalities. https://www.minimax.io/blog/minimax-h3, July 2026. Accessed September 22, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Jian Tang, Jiawei Fan, Qingbin Liu, and Zheng Wei. Fis-dit: Breaking the few-step video inference barrier via training-free frame interleaved sparsity, 2026. URL https://arxiv.org/abs/2605.11869.

DreamX Team, Yancheng Bai, Rui Chen, Xiangxiang Chu, Rujing Dang, Hao Dou, Bingjie Gao, Qiwen Gu, Siyu Hong, Jiachen Lei, et al. Dreamx-world 1.0: A general-purpose interactive world model. *arXiv preprint arXiv:2606.16993*, 2026.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Bing Wu, Chang Zou, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Jack Peng, Jianbing Wu, Jiangfeng Xiong, Jie Jiang, et al. Hunyuanvideo 1.5 technical report. *arXiv preprint arXiv:2511.18870*, 2025.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse video-gen: Accelerating video diffusion transformers with spatial-temporal sparsity. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68208–68224. PMLR, 13–19 Jul 2025.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, Jianfei Chen, Song Han, Kurt Keutzer, and Ion Stoica. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 96965–96991. Curran Associates, Inc., 2025. doi: 10.52202/085713-3243. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/8bd5148caced2d73cea7b6961a874a49-Paper-Conference.pdf.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22963–22974. IEEE, 2025.

Xin Zhou, Dingkang Liang, Kaijin Chen, Tianrui Feng, Xiwu Chen, Hongkai Lin, Yikang Ding, Feiyang Tan, Hengshuang Zhao, and Xiang Bai. Less is enough: Training-free video diffusion acceleration via runtime-adaptive caching, 2025. URL https://arxiv.org/abs/2507.02860.

Chang Zou, Changlin Li, Yang Li, Patrol Li, Jianbing Wu, Xiao He, Songtao Liu, Zhao Zhong, Kailin Huang, and Linfeng Zhang. Disca: Accelerating video diffusion transformers with distillation-compatible learnable feature caching, 2026. URL https://arxiv.org/abs/2602.05449.

Appendix Overview

The appendix provides the following details:

A. Frame-Axis Measurements and Tests of the Analysis: Definitions of the frame-axis measurements of Section 4 and tests of Assumption 1 and of the target comparison.

B. Optimal Shrinkage within the Linear Class: The derivation of Proposition 1, the case of total weight $s < 1$, and the comparison with linear interpolation.

C. Algorithms and Implementation Fidelity: Pseudocode of the frame-sparse forward pass and of the two reconstruction operators, and checks that our implementation of BlockDelta and FIS-DiT matches the algorithms.

D. Experimental Setup: Models, dense-block sets, noise threshold, hardware and timing protocol, prompt sets and metric definitions.

E. Further Results: The composition of BlockDelta with SVG and the results on LTX-2.3 and MiniMax-H3.

F. Corrector: The optional learned corrector, its training setup and its measured recovery.

G. Step Reduction and Step Reuse: The comparison of fewer dense steps, TeaCache and TaylorSeer on the 4-step Wan2.2 models and the 8-step HunyuanVideo-1.5 I2V model, and of these methods with BlockDelta.

H. Component Ablation Suite: The leave-one-out ablation on Wan2.2 and the effect of the dense-block set, anchor stride and dense-step budget.

I. Scope beyond Few-step Models: BlockDelta and step reduction on the undistilled Wan2.2-A14B model.

J. Limitations and Future Work: The discussion of limitations of BlockDelta and future work.

7. The statement of using generative AI tools in this work.

8. The statement of reproducibility.

K. Qualitative Comparisons: Video frames of dense inference, FIS-DiT and BlockDelta on all five models.

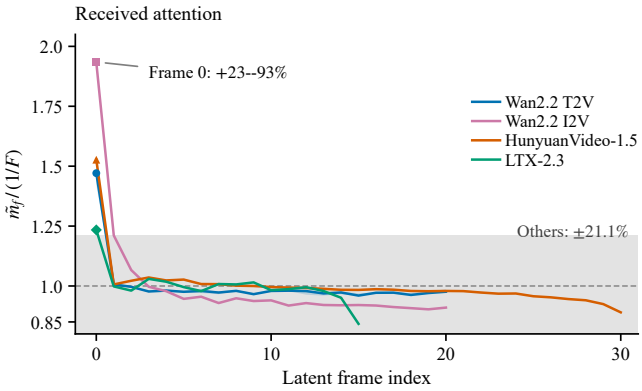


Figure 6: Attention received by each latent frame. $\tilde{m}_f$ is the attention share normalized over video tokens, and F is the number of latent frames. Lines show family medians; shaded envelopes span configurations within each family. The gray band marks the largest relative deviation from $1/F$ among frames other than 0.

A Frame-Axis Measurements and Tests of the Analysis

This appendix supports Section 4. Appendix A.1 defines the measurements of Figures 4 and 6, and Appendix A.2 tests the analysis of Sections 4.1 and 4.2 against measured statistics. Appendix B gives the derivations.

A.1 Definitions of the Frame-Axis Measurements

This subsection defines the measurements of Figures 4 and 6.

Attention mass. Let Attn be the row-normalized attention matrix of a self-attention call over the S video tokens ($S = F \cdot S_f$ for F latent frames of S_f tokens each; text tokens, where present, enter the softmax denominator but belong to no frame bucket). The mass received by frame f is

$$m_f = \frac{1}{S} \sum_{p=1}^{S} \sum_{q \in \text{frame } f} \mathrm{Attn}_{pq},$$

which gives $\sum_f m_f = 1$ in the video-only case. In joint-attention (MMDiT) models, text tokens receive part of the attention of each row (7–8% for HunyuanVideo-1.5), which would shift the whole curve downward relative to models with cross-attention. We therefore normalize each measurement, $\tilde{m}_f = m_f / \sum_{f'} m_{f'}$, and $\tilde{m}_f$ is then the share of frame f in the attention to video tokens for every model. Figure 6 plots $\tilde{m}_f/(1/F)$, the ratio to the uniform share.

(a) Reconstruction error. Anchors are placed at stride 3, and the first and last frames are always anchors. Every skipped frame f between anchors $a < f < b$ is reconstructed by linear interpolation with weight $\lambda_f = (f-a)/(b-a)$, as in Eq. (7). The same interpolation is applied to the input X, the output Y and the residual $\Delta = Y - X$. For $Z \in \{X, Y, \Delta\}$, $R^Z_{l,i}$ is the ℓ_2 norm of the interpolation error of Z over the skipped frames at block l in measurement record i, and $\bar{R}^Z_l = \mathrm{mean}_i\, R^Z_{l,i}$ is its mean over records. Residual reconstruction outputs $\hat{Y} = X + \hat{\Delta}$, and its output error therefore equals the error of Δ, and the errors of the two targets are directly comparable. Panel (a) plots $\bar{R}^Y_l / \bar{R}^X_l$ and $\bar{R}^\Delta_l / \bar{R}^X_l$, which are ratios of mean norms, not means of per-record ratios or squared errors. The division by $\bar{R}^X_l$ removes the growth of the feature norm with depth, which spans a factor of 4 across blocks, and a value of 1 means an error as large as that of re-estimating the input from the anchors. This normalization cancels in the ratio of Y to Δ. Over all blocks, the error of Y is 2.17 times that of Δ, and it is larger on 39 of 40 blocks; the exception is block 21, with a ratio of 0.98.

Panel (a) is measured on 480 points of Wan2.2-A14B, with norms over 256 of the 1,560 sampled spatial tokens.

DCT energy notation for (b)–(c). For a feature sequence $Z \in \{\Delta, Y\}$ with N_Z frames, let $\mathcal{D}_k(Z)$ denote its kth orthogonal DCT-II coefficient along the frame axis. The single-coefficient energy fraction and cumulative energy fraction are

$$\pi_k^Z = \frac{\|\mathcal{D}_k(Z)\|_F^2}{\sum_{j=0}^{N_Z-1} \|\mathcal{D}_j(Z)\|_F^2}, \qquad \Pi_k^Z = \sum_{j=0}^{k} \pi_j^Z.$$

Here $N_\Delta = F - 1$ because panel (b) excludes X_0, whereas $N_Y = F$ in panel (c). The norm sums over spatial tokens and feature channels. These quantities are defined per measurement before the aggregation described below; bars display summaries of π_k^Z and lines display summaries of Π_k^Z.

(b) DCT energy of the residual along the frame axis. For each instance at a fixed block and step, we take the residuals $\Delta_1, \ldots, \Delta_{F-1}$ as in Section 4.2, with X_0 excluded before any centering, apply an orthogonal DCT-II along the frame axis, and report the share of the residual energy carried by each coefficient. The share of coefficient 0 is exactly the energy of the shared component c of Eq. (5), $(F-1)\|c\|^2 / \sum_f \|\Delta_f\|^2$, and the other shares are those of the frame-specific component ϵ. The records are the instance–block–step measurements of Wan2.2, HunyuanVideo-1.5 and LTX-2.3 (3,840, 10,368 and 12,672 records), computed from the residual Gram matrices of Appendix A.2. Lines are medians over the records of each model, and bars are the median of the three model medians with their range. The medians of the coefficient-0 share are 0.397, 0.576 and 0.477, with interquartile ranges $[0.30, 0.53]$, $[0.50, 0.66]$ and $[0.27, 0.76]$. The reference $(k+1)/(F-1)$ is the share of any $k+1$ coefficients when the frame axis has no structure. On Wan2.2, the shares of coefficients 1, 2 and 3 are 0.047, 0.041 and 0.037, against 0.032 for a flat spectrum of the frame-specific component. The energy above the flat spectrum is therefore small and lies in the lowest coefficients; it corresponds to the departure from A2 measured in Appendix A.2. With X_0 included, the coefficient-0 shares are 0.38, 0.56 and 0.47.

(c) DCT energy of the output along the frame axis. We apply the same orthogonal DCT-II along the frame axis of the block output $Y \in \mathbb{R}^{F \times S_f \times D}$ (flattened over the non-frame dimensions), take the squared norm of each coefficient as its energy, and report the single-coefficient fractions π_k^Y and cumulative fractions Π_k^Y through coefficient k over all F frames.

Anchor offset. With anchor stride 3, the anchor grid has three possible offsets along the frame axis. Changing the offset leaves the reconstruction error almost unchanged: across the three offsets, the relative standard deviation of the error is 0.008–0.042 in all measured configurations of the three model families. The offset can therefore change from block to block (Section 3) without a calibration for each offset.

A.2 Tests of the Reconstruction Analysis

This subsection gives the measurements behind the analysis of Section 4: the frame domain of the decomposition, departures from Assumption 1, the choice of interpolation weights, the exact comparison of the two targets and the error added by frame-sparse execution of the anchors. The measurements test Assumption 1 but do not verify all of it. Appendix B gives the derivations.

Centering and frame domain. The shared component is computed for each instance over $X_1, \ldots, X_{F-1}$. With $n = F - 1$ analyzed frames, the residual Gram matrix G of these frames and $P = I - \mathbf{1}\mathbf{1}^\top/n$, the Gram matrix of the frame-specific components is PGP, the shared energy of the instance is $\mathbf{1}^\top G \mathbf{1}/n^2$, and its mean frame-specific energy is $\mathrm{tr}(PGP)/n$. The expectations of Section 4.2 average these quantities over prompts at a fixed block and step. X_0 must be removed before centering; centering all frames and then removing X_0 gives different frame-specific components. The analysis also requires every anchor used in an estimate to lie in this domain. In BlockDelta, X_1 is always an anchor, and no skipped frame is therefore interpolated from X_0 (Section 4.3).

Departures from Assumption 1. On Wan2.2, the frame-specific components depart slightly from equal correlation at short range. With X_0 excluded and moments averaged over prompts before aggregation over block–step strata, the normalized lag covariances at lags 1–4 are 0.050, 0.014, –0.031 and –0.043, whereas A2 gives $-1/(n-1)$ at every lag. The correlation thus decreases with the distance between frames, which Section 4.2 uses for the placement of the weights. For the equal energy of A1, the coefficient of variation of the frame-specific energy over frames has a median of 0.045 over strata on Wan2.2. The medians of the per-instance coefficients of variation are 0.067, 0.126 and 0.240 on Wan2.2, HunyuanVideo-1.5 and LTX-2.3; these two summaries use different aggregation units. The per-frame cross moment between the shared and frame-specific components in A1 was not measured. A1 and A2 are therefore idealizations of the measured statistics, not verified properties.

Exact target comparison. Which target has the lower local error depends on the cross term of the two errors, not on the energy profile along the frame axis. Using the same weights for X, Y and Δ gives $e_Y = e_X + e_\Delta$, which proves Eq. (4) without A1–A2. By the Cauchy–Schwarz inequality,

$$\mathcal{L}(Y) - \mathcal{L}(\Delta) \geq \mathcal{L}(X) - 2\sqrt{\mathcal{L}(X)\mathcal{L}(\Delta)}, \tag{8}$$

and $\mathcal{L}(X) > 4\mathcal{L}(\Delta)$ is therefore a sufficient but not a necessary condition. The comparison uses stride-3 anchors whose offset changes from block to block, with the first and last frames as anchors and without the VAE-aligned anchors, which differs from BlockDelta. Each record first averages the squared errors over the skipped frames, and all records have equal weight. The exact condition holds on 39 of 40, 54 of 54 and 47 of 48 blocks of Wan2.2, HunyuanVideo-1.5 and LTX-2.3, and the sufficient condition on 31, 18 and 43 of these blocks. The ordering reverses on block 21 of Wan2.2 and block 46 of LTX-2.3, where $\mathcal{L}(Y)/\mathcal{L}(\Delta)$ is 0.810 and 0.544 and the correlation between e_X and e_Δ is –0.710 and –0.747.

Frame-sparse execution of the anchors. The analysis uses the dense anchor residuals, whereas frame-sparse execution gives $\widetilde{\Delta}_a = \Delta_a + \eta_a$. Since $\widetilde{Y}_a = X_a + \widetilde{\Delta}_a$, output interpolation receives the same term η_a, and Eq. (4) holds unchanged with e_Δ replaced by the error under frame-sparse execution. The perturbation therefore favors neither target. At a common block input, the local error of residual reconstruction becomes $e_\Delta - \mathcal{I}[\eta]_f$, and

$$\mathbb{E}[\|e_\Delta - \mathcal{I}[\eta]_f\|^2] = \mathcal{L}(\Delta) + \mathbb{E}[\|\mathcal{I}[\eta]_f\|^2] - 2\mathbb{E}[\langle e_\Delta, \mathcal{I}[\eta]_f \rangle]. \tag{9}$$

On Wan2.2 with BlockDelta, the anchor term, the signed cross term and the total error are 0.021, 0.003 and 1.024 times the interpolation term $\mathcal{L}(\Delta)$, measured on 24 prompts, the three sparse steps and the 37 sparse blocks. The frame-sparse execution of the anchors thus adds little to the local error in this setting. This measurement does not bound the error accumulated over the denoising steps.

Estimators outside the linear class. Second moments do not exclude nonlinear or content-dependent estimates of the residual from the anchors. The corrector of Appendix F is such an estimate. It acts on the last dense blocks and is trained against the final velocity; its effect is therefore evaluated by the comparison with and without the corrector, not by the local error of Appendix B.

B Optimal Shrinkage within the Linear Class

This appendix derives Proposition 1 and extends the analysis to any total weight $s \geq 0$ and to signed individual weights; the constraint $s \geq 0$ is relaxed only where stated. Throughout this appendix, $n = F - 1$ is the number of analyzed frames, $n \geq 2$, $1 \leq K \leq n-1$, $C = \mathbb{E}[\|c\|^2] \geq 0$ is the energy of the shared component, and $\nu^2 > 0$; all moments and anchors use $X_1, \ldots, X_{F-1}$ as in Eq. (5). Let $\gamma = 1/(n-1)$, which is the γ of Assumption 1, and $\beta = (n-K)/[K(n-1)] > 0$.

Least-squares estimates retain less energy. The following lemma holds without A1–A2.

Lemma 1 (Energy decomposition) *Assume $\mathbb{E}[\|\epsilon_f\|^2] < \infty$. Let $\hat{\epsilon}_f^*$ minimize $\mathbb{E}[\|\epsilon_f - \hat{\epsilon}_f\|^2]$ over a class of estimators that is closed under scaling, for example the linear span of the anchor residuals. Denote the minimum error by $\mathcal{L}_{\min}$ and the energy of the minimizer by $\hat{\nu}_*^2 = \mathbb{E}[\|\hat{\epsilon}_f^*\|^2]$. Then*

$$\nu^2 = \mathcal{L}_{\min} + \hat{\nu}_*^2, \qquad \hat{\nu}_*^2 \leq \nu^2, \tag{10}$$

with $\hat{\nu}_*^2 = \nu^2$ if and only if $\hat{\epsilon}_f^* = \epsilon_f$ almost surely.

Proof. Because the class is closed under scaling, $h(t) = \mathbb{E}[\|\epsilon_f - t\hat{\epsilon}_f^*\|^2]$ is minimized at $t = 1$. Setting $h'(1) = 0$ gives $\mathbb{E}[\langle \epsilon_f, \hat{\epsilon}_f^* \rangle] = \hat{\nu}_*^2$, and therefore $\mathcal{L}_{\min} = h(1) = \nu^2 - 2\hat{\nu}_*^2 + \hat{\nu}_*^2 = \nu^2 - \hat{\nu}_*^2$. Since $\mathcal{L}_{\min} \geq 0$, $\hat{\nu}_*^2 \leq \nu^2$, with equality if and only if $\mathcal{L}_{\min} = 0$, that is, $\hat{\epsilon}_f^* = \epsilon_f$ almost surely. □

Position-only estimates with $s = 1$ do not form a class closed under scaling, and the lemma therefore does not apply to them directly; Proposition 1 gives the corresponding identity for that class under A1–A2.

The centering constraint. The frame-specific components obey $\sum_g \epsilon_g = 0$ in every instance. Under equal frame-specific component energies this requires

$$\sum_{g \neq f} \mathbb{E}[\langle \epsilon_f, \epsilon_g \rangle] = -\nu^2. \tag{11}$$

A2 assigns the same moment to each off-diagonal pair; centering alone does not require that assignment. Likewise, $\sum_f \langle c, \epsilon_f \rangle = 0$ holds exactly, but individual frame cross moments need not vanish. Their vanishing is a separate part of A1.

Error expansion and fixed-sum optimality. The reconstruction error separates as $(1-s)c + \epsilon_f - \sum_a w_a \epsilon_a$. A1 removes the cross terms between the shared and frame-specific components, and A2 gives

$$\mathbb{E}\left[\left\|\epsilon_f - \sum_a w_a \epsilon_a\right\|^2\right] = \nu^2 \left[1 + q_w - \gamma(s^2 - q_w) + 2\gamma s\right], \tag{12}$$

$$\mathcal{L}(\Delta) = (1-s)^2 C + \nu^2 \left[1 + 2\gamma s + \beta s^2 + (1+\gamma)\left(q_w - \frac{s^2}{K}\right)\right]. \tag{13}$$

Since $q_w \geq s^2/K$, uniform weights uniquely minimize the error at fixed s. Every term after ν^2 in Eq. (13) is nonnegative when $s \geq 0$, which gives $\mathcal{L}(\Delta) \geq \nu^2$ for every linear estimate with $s \geq 0$. Equality requires all weights to vanish and $C = 0$. The stronger bound $\mathcal{L}(\Delta) \geq \nu^2(1 + q_w)$ follows for $0 \leq s \leq 2$ from $2s - s^2 \geq 0$; this derivation does not cover $s > 2$. The retained energy is $\hat{\nu}^2 = \mathbb{E}[\|\sum_a w_a \epsilon_a\|^2] = \nu^2[(1+\gamma)q_w - \gamma s^2]$ under A2. At $s = 1$, Eq. (13) gives $\mathcal{L}(\Delta) = \nu^2(1+\gamma)(1+q_w)$, since $\gamma + \beta = (1+\gamma)/K$, and subtracting the retained energy gives Eq. (6), which proves Proposition 1.

Constrained shrinkage. Uniform weights reduce the risk to a strictly convex scalar quadratic,

$$\mathcal{L}(s) = C + \nu^2 - 2(C - \gamma\nu^2)s + (C + \beta\nu^2)s^2. \tag{14}$$

Write $(x)_+ = \max\{x, 0\}$. The constrained optimum and its risk are

$$s^* = \frac{(C - \gamma\nu^2)_+}{C + \beta\nu^2}, \qquad \mathcal{L}_+^* = C + \nu^2 - \frac{(C - \gamma\nu^2)_+^2}{C + \beta\nu^2} \geq \nu^2, \tag{15}$$

a shrunk all-anchor mean, or the zero estimator when the shared energy is too weak. Projecting the vertex of the quadratic onto $s \geq 0$ proves Eq. (15), with $0 \leq s^* < 1$. Equivalently, for $r = KC/\nu^2$,

$$s^* = \max\left\{0, \frac{r - \gamma K}{r + 1 + \gamma - \gamma K}\right\}. \tag{16}$$

For $C \leq \gamma\nu^2$, the optimum is the zero estimator and its risk is $C + \nu^2$; otherwise it is a positively shrunk mean. In the limit $n \to \infty$ with K, C, ν^2 fixed, the familiar formulas $s^* = r/(1+r)$ and $\mathcal{L}_+^* = \nu^2(1 + s^*/K)$ are recovered. At fixed n, K, the large-C limit of the optimal risk is $\nu^2(1+\gamma)(1+1/K)$, an upper envelope rather than a lower bound.

Why nonnegative total weight matters. Negative total weights can exploit the anticorrelation imposed by centering. Without the constraint, the optimum is $(C - \gamma\nu^2)/(C + \beta\nu^2)$, which can be negative; that sign alone does not imply risk below ν^2. In the boundary case $C = 0$ and $K = n - 1$, the anchors contain every other analyzed frame and $\Delta_f = -\sum_{a\in\mathcal{A}} \Delta_a$ exactly. Negative weights then achieve zero error, whereas the constrained optimum remains zero weights with error ν^2. The lower bound $\mathcal{L}(\Delta) \geq \nu^2$ therefore requires $s \geq 0$, but it does not require each weight to be nonnegative.

Linear-interpolation comparison. An estimator with $s = 1$ has $\mathcal{L}(\Delta) = \nu^2(1 + \gamma)(1 + q_w)$. An interior stride-3 linear interpolation uses weights $(1/3, 2/3)$ or their reverse, giving $q_w = 5/9$, while the unshrunk K-anchor mean has $q_w = 1/K$. Therefore

$$\frac{\mathcal{L}_{\text{interp}}}{\mathcal{L}_{\text{mean}}} = \frac{14K}{9(K+1)}, \qquad \frac{\mathcal{L}_{\text{interp}}}{\mathcal{L}_{+}^{*}} = \frac{\frac{14}{9}(1+\gamma)\nu^2}{\mathcal{L}_{+}^{*}}. \tag{17}$$

These ratios assume that both estimators and their targets lie in the frame domain of Eq. (5) and that the interpolated frame lies between two interior anchors. Frames next to the VAE-aligned anchors or at the ends need their actual weights, and numerical predictions need moments measured on the same domain.

Departures from symmetric correlation. Frame-specific component correlations can change the preferred weighting even when A1 holds. Define $\rho_{gh} = \mathbb{E}[\langle \epsilon_g, \epsilon_h \rangle]/\nu^2 + \gamma$ for $g \neq h$. Relative to Eq. (13), the exact additional term under A1 is

$$\nu^2 \left[\sum_{a\neq b} w_a w_b \rho_{ab} - 2 \sum_{a} w_a \rho_{fa} \right]. \tag{18}$$

Positive target–near-anchor excess correlation can benefit a local linear interpolation, but the anchor–anchor term also contributes. Departures from equal energies or shared–frame-specific component orthogonality introduce further terms. A measured lag correlation therefore does not by itself give a corrected lower bound, and second moments do not exclude nonlinear structure.

C Algorithms and Implementation Fidelity

This appendix gives the frame-sparse forward pass (Algorithm 1) and the two reconstruction operators (Algorithms 2 and 3). FIS-DiT and BlockDelta (FIS-DiT settings) in Table 1 use the same schedule, that is, the same dense-block set, anchor stride, change of anchor offset and dense steps, and differ only in the reconstruction operator applied to the skipped frames after each sparse block. On Wan2.2 T2V, this schedule is the published FIS-DiT setting with anchor stride $\kappa = 3$, dense-block set $\mathcal{D} = \{0, 38, 39\}$ and one dense final step. BlockDelta adds the σ-aligned dense steps and the VAE-aligned anchors, which Algorithm 1 sets through σ_{th} and the number F_{lead} of leading anchor frames.

Both methods call $\mathcal{R}$ at the same place, and they differ only in what $\mathcal{R}$ uses. Output interpolation (FIS-DiT) uses only the anchor outputs, whereas BlockDelta also uses the block input X. The next block receives the reconstructed features, and the two methods therefore have the same features up to the first sparse block and different features after it, although their schedule is the same.

Algorithm 3 applies Eq. (7) between every pair of adjacent anchors: the residual is interpolated along the frame axis, and each skipped frame adds its own input instead of an interpolated one. Since X_f is available before the block runs, the residual is the only unknown. The implementation computes the interpolation in fp32 and casts the reconstructed tokens back to the model dtype. It also has an optional scalar weight on the residual for ablations; all BlockDelta results use Eq. (7) without this weight.

Implementation fidelity. Three checks support that our implementation matches the algorithm. First, with stride 1, every frame is an anchor, the reconstruction operator is the identity, and the output is bitwise identical to dense inference. Second, on Wan2.2 T2V, the end-to-end speedup

Algorithm 1 Frame-sparse forward pass at denoising step k, shared by both methods

Require: block input tokens X, step index k, blocks $\{\mathcal{B}_l\}_{l=0}^{L-1}$, frame count F, anchor stride κ, dense-block set $\mathcal{D}$, noise threshold σ_{th}, number of leading anchor frames F_{lead}, first sparse block l_0, reconstruction operator $\mathcal{R} \in \{\mathcal{R}_{\text{output}}, \mathcal{R}_{\text{residual}}\}$
Ensure: block output tokens X

if $\sigma_k < \sigma_{\text{th}}$ **then**
  $\mathcal{D}_k \leftarrow \{0, 1, \ldots, L-1\}$ — steps ending below σ_{th} stay dense
else
  $\mathcal{D}_k \leftarrow \mathcal{D}$
end if
$f_{\text{lead}} \leftarrow \max(0, F_{\text{lead}} - 1)$ — the offset is counted from the last leading anchor
for $l = 0$ to $L-1$ **do**
  if $l \in \mathcal{D}_k$ **then**
    $X \leftarrow \mathcal{B}_l(X)$
  else
    $o_l \leftarrow (l - l_0) \bmod \kappa$
    $\mathcal{A}_l \leftarrow \{f : f \le f_{\text{lead}} \text{ or } (f - f_{\text{lead}} - o_l) \bmod \kappa = 0 \text{ or } f = F-1\}$
    $\widetilde{Y}_{\mathcal{A}_l} \leftarrow \mathcal{B}_l(\{X_a \mid a \in \mathcal{A}_l\})$ — gather the anchor frames and run the block
    $X \leftarrow \mathcal{R}\big(\widetilde{Y}_{\mathcal{A}_l}, X\big)$
  end if
end for
return X

Algorithm 2 $\mathcal{R}_{\text{output}}$: output interpolation (FIS-DiT)

Require: anchor outputs $\widetilde{Y}_{\mathcal{A}_l}$, anchor indices $\mathcal{A}_l = \{a_0, \ldots, a_M\}$ with $a_0 < \cdots < a_M$, frame count F
Ensure: reconstructed block outputs $\hat{Y}_{0:F-1}$

for $i = 0$ to M **do**
  $\hat{Y}_{a_i} \leftarrow \widetilde{Y}_{a_i}$ — anchor outputs are kept
end for
for $i = 0$ to $M-1$ **do**
  $a^- \leftarrow a_i$, $a^+ \leftarrow a_{i+1}$
  for $f = a^- + 1$ to $a^+ - 1$ **do**
    $\lambda_f \leftarrow (f - a^-)/(a^+ - a^-)$
    $\hat{Y}_f \leftarrow (1 - \lambda_f)\widetilde{Y}_{a^-} + \lambda_f \widetilde{Y}_{a^+}$
  end for
end for
return $\hat{Y}_{0:F-1}$

of our FIS-DiT reproduction on one H20 is 1.79× (Table 1), close to the 1.86× reported by FIS-DiT on the same GPU type. Third, on Wan2.2 T2V, our reproduction changes the VBench score only from 82.36 to 81.38 (Table 1), consistent with the near-lossless result reported by FIS-DiT. FIS-DiT did not evaluate Wan2.2 I2V. On HunyuanVideo-1.5 I2V, it reports a near-lossless result, with the VBench quality score changing from 0.8065 to 0.8060 on 100 VBench prompts paired with reference images. With the same dense blocks $\{0\text{–}5, 51\text{–}53\}$, sparse steps 0–6 and stride 3, our reproduction degrades strongly on the official VBench-I2V list (Table 1). The prompts and reference images of the published evaluation are not given, and the two evaluations are therefore not directly comparable, and implementation details not described in the paper may also contribute.

D Experimental Setup

D.1 Models and Configurations

Step-distilled checkpoints. Of the eleven transformer variants released with HunyuanVideo-1.5, only one is step-distilled, the 480p I2V model, and this model family therefore has no few-step

Algorithm 3 $\mathcal{R}_{\text{residual}}$: residual interpolation (BlockDelta)

Require: block input X, anchor outputs $\widetilde{Y}_{\mathcal{A}_l}$, anchor indices $\mathcal{A}_l = \{a_0, \ldots, a_M\}$ with $a_0 < \cdots < a_M$, frame count F
Ensure: reconstructed block outputs $\hat{Y}_{0:F-1}$

for $i = 0$ to M **do**
  $\hat{Y}_{a_i} \leftarrow \widetilde{Y}_{a_i}$, $\widetilde{\Delta}_{a_i} \leftarrow \widetilde{Y}_{a_i} - X_{a_i}$ anchor outputs are kept
end for
for $i = 0$ to $M - 1$ **do**
  $a^- \leftarrow a_i$, $a^+ \leftarrow a_{i+1}$
  for $f = a^- + 1$ to $a^+ - 1$ **do**
    $\lambda_f \leftarrow (f - a^-)/(a^+ - a^-)$
    $\hat{Y}_f \leftarrow X_f + (1 - \lambda_f)\,\widetilde{\Delta}_{a^-} + \lambda_f\,\widetilde{\Delta}_{a^+}$
  end for
end for
return $\hat{Y}_{0:F-1}$

T2V checkpoint. The frame-axis measurements of Section 4 also include multi-step configurations, whereas all experiments of Section 5 use few-step distilled checkpoints.

Dense computations. The dense-block set is $\mathcal{D} = \{0, 38, 39\}$ of 40 blocks on Wan2.2 and $\mathcal{D} = \{0, \ldots, 5, 51, 52, 53\}$ of 54 blocks on HunyuanVideo-1.5. The noise threshold is $\sigma_{\text{th}} = 0.6$ for all models; it keeps the last of four steps of Wan2.2 and the last two of eight steps of HunyuanVideo-1.5 dense. The settings of LTX-2.3 and MiniMax-H3 are given in Appendix E.2.

Hardware and timing. The videos of Table 1 were generated on PPU accelerators, and their FID and FVD are computed against dense videos generated on the same accelerators. All timings in the paper, on Wan2.2, HunyuanVideo-1.5, LTX-2.3 and MiniMax-H3, were measured on a single NVIDIA H20. Each configuration generates the same 7 prompts, and the reported latency is the mean of the middle three of the seven runs. Speedups are ratios of these latencies to that of dense inference on the same model. E2E time and DiT time are reported separately, and a configuration that appears in several tables has the same measured latency in all of them.

Pseudocode. Appendix C gives pseudocode for both reconstruction operators and shows where they are called in the forward pass.

D.2 Evaluation Protocol

Pairing. LPIPS compares each video with the dense video of the same prompt and seed. FID and FVD compare the feature distribution of a configuration with that of dense inference generated in the same batch. The Dense inference (reseed) rows give the variation under a seed change, not a lower bound on the error. The I2V evaluations use the official VBench-I2V list of 1,118 image–prompt pairs with 5 seeds (5,590 videos per configuration), and the T2V evaluations use the official VBench prompt list (6,220 videos per configuration). Each table states its VBench aggregation.

Prompt sets. The ablations use one fixed set of 100 prompts for T2V and one fixed set of 100 image–prompt pairs for I2V, both drawn from the official VBench lists before any video was generated (Table 3). Wan2.2 T2V uses the T2V set, and Wan2.2 I2V and HunyuanVideo-1.5 I2V share the I2V set. Each prompt is assigned to the first VBench dimension it serves. Each dimension then receives a share of the 100 prompts proportional to its share of the pool, rounded by the largest-remainder method. The T2V pool contains 495 prompts. The prompts that serve dynamic degree are assigned to subject consistency, and this dimension is raised to 15 prompts, since the binary dynamic-degree scores had the largest variance in earlier runs. The I2V pool contains the 1,118 official image–prompt pairs, most of which serve camera motion, and the I2V set keeps this share. The two sets share no prompt, and each video uses seed 42 plus the index of its prompt in the official list.

Table 3: Prompt sets of the ablations: number of prompts assigned to each VBench dimension. Each prompt is counted once, under the first dimension it serves.

T2V dimension	Prompts
Subject consistency	15
Object class	9
Temporal style	9
Overall consistency	9
Scene	9
Spatial relationship	9
Temporal flickering	8
Multiple objects	8
Human action	8
Color	8
Appearance style	8
Total	100

I2V dimension	Pairs
Camera motion	68
I2V subject	22
I2V background	10
Total	100

Distribution metrics. FID uses the standard `pytorch-fid 0.3.0` Inception features (2048 dimensions) of all decoded frames. FVD uses 400-dimensional pre-softmax I3D features of whole videos, resized to a short side of 224 and center-cropped, without temporal resampling. All FID and FVD values use float64 feature moments, unbiased covariance and the same positive-semidefinite square root, with no added regularization. Values from different models or sample sizes are not directly comparable.

D.3 Pixel-Level Metrics

The Head-flash columns of Tables 2, 7 and 5, the Chroma column of Table 8 and the Dyn. columns of Tables 2, 7, 5, 8, 9 and 10 are computed on the decoded pixel frames $v_1, \ldots, v_T \in [0,1]^{H\times W\times 3}$ of each video by a single script shared across models. Let $\bar{\ell}_t = 255 \cdot \mathrm{mean}(v_t)$ be the mean luminance of frame t on the 0–255 scale. Head-flash is the largest luminance jump in the leading frames, $\max_{1\le t\le 7} |\bar{\ell}_{t+1} - \bar{\ell}_t|$, where the first frame, $t = 0$, is excluded because it is always an anchor. Chroma is the per-pixel channel range averaged over the video, $255 \cdot \mathrm{mean}_{t,h,w}\big(\max_c v_{t,h,w,c} - \min_c v_{t,h,w,c}\big)$; a full-clip collapse toward gray lowers it. Both are reported as per-video medians over the 100 videos of a configuration. Dyn. is a motion-retention metric: the per-video motion proxy is the mean absolute frame difference,

$$m(v) = \frac{1}{(T-1)\,HW\cdot 3}\sum_{t=1}^{T-1}\big\|v_{t+1} - v_t\big\|_1,$$

and the column reports $100 \times \mathrm{median}_i\ m(v_i)/m(v_i^{\mathrm{dense}})$ over same-seed pairs i; dense inference is thus 100 by definition and the Dense inference (reseed) row gives the variation of the column under a seed change. We use this proxy rather than VBench `dynamic_degree`, because `dynamic_degree` gives a binary score per video (optical-flow magnitude above a fixed threshold) and is computed only on its own 15 of the 100 prompts, and it therefore cannot show a partial loss of motion. The proxy is continuous and paired on all 100 videos. It measures pixel change, not motion, and flicker and other temporal artifacts also raise it. A value below 100 therefore indicates lost motion, whereas a value at or above 100 does not show that motion is preserved.

E Further Results

E.1 Composition with SVG

Adding SVG to BlockDelta raises FVD on the two Wan2.2 models and lowers it on HunyuanVideo-1.5 I2V (Table 1). A likely cause is that SVG changes the anchor frames, and the skipped frames, which are interpolated from these anchors, contain the same change. The ordering across models agrees with this. Relative to the reseeded dense row, SVG alone has 1.9 times the FVD on

Wan2.2 T2V and 5.7 times on Wan2.2 I2V, where the FVD increase is largest, and 0.5 times on HunyuanVideo-1.5 I2V, where FVD decreases.

E.2 Results on LTX-2.3 and MiniMax-H3

This section evaluates BlockDelta on two models outside the full benchmark. Both models are compared with dense inference generated in the same batch and with a Dense inference (reseed) row, which gives the variation of each column under a seed change. The comparison includes output interpolation with the same settings but not SVG; it thus tests whether BlockDelta transfers, not how it compares with other acceleration methods. Dyn. is the metric of Appendix D.3, with dense inference at 100.

LTX-2.3. LTX-2.3 is the most recent of the three models analyzed in Sections 4.1–4.2 and uses a two-stage distilled pipeline with $8 + 3$ steps. We evaluate BlockDelta at 512×768 and 193 output frames, on 100 T2V prompts with shared seeds. The dense blocks are $\{0, 1, 2, 43\text{–}47\}$ of 48 blocks, and the anchors use stride 3 with the anchor offset changing from block to block. The σ-aligned rule is applied only to the second stage, because this stage adds noise to the upsampled output of the first stage, up to $\sigma = 0.909$, and denoises it again. The rule keeps the last two of the three second-stage steps dense, and all first-stage steps are frame-sparse. The sparse settings remove 26% of the DiT FLOPs (from 0.89 to 0.66 PFLOPs per video). Table 4 reports BlockDelta and output interpolation with the same settings, together with dense inference and Dense inference (reseed). On one H20, with the timing protocol of Appendix D, the E2E speedup is 1.21× and the DiT speedup is 1.32×. The full VBench suite was not run on this model. LTX-2.3 also generates a soundtrack. Frame sparsity is applied only to video tokens, and audio tokens are always computed in full; they change only through the video tokens they attend to. The log-mel distance of residual interpolation to the dense soundtrack is 0.27, against 0.57 for Dense inference (reseed) and 0.31 for output interpolation, and residual interpolation is closer than output interpolation on 69 of 100 paired videos ($p = 2 \times 10^{-4}$). Its FAD in CLAP embedding space is 0.14, against 0.17 for Dense inference (reseed) and 0.23 for output interpolation.

Table 4: Results on LTX-2.3 and MiniMax-H3 with 100 prompts each. VBench-sub is the VBench score on these 100 prompts, multiplied by 100. FID and FVD are computed against dense inference of the same model generated in the same batch. LPIPS is the median over videos of the mean frame-level AlexNet LPIPS to the same-seed dense video, and Dyn. is relative to dense inference (100). Quality values are not comparable to those of Table 1. E2E and DiT times are seconds per video, with speedups relative to dense inference in parentheses; timing follows Appendix D. A dash marks a value that was not measured or does not apply.

Method	Efficiency		Quality				
	E2E (s) ↓	**DiT** (s) ↓	**VBench-sub** ↑	**FID** ↓	**FVD** ↓	**LPIPS** ↓	**Dyn.**
LTX-2.3 (2026) (T2AV, 193 frames, 512×768, T=8+3)							
Dense inference	21.52 (1.00×)	14.79 (1.00×)	76.89	—	—	—	100
Dense inference (reseed)	—	—	76.19	91.85	854.27	0.7335	76.5
Output interpolation	17.57 (1.23×)	11.02 (1.34×)	70.58	144.12	2092.43	0.7008	42.5
BlockDelta	17.85 (1.21×)	11.18 (1.32×)	75.24	99.92	951.89	0.6125	66.5
MiniMax-H3 (2026) (T2AV, 124 frames, 768×1344, T=8)							
Dense inference	300.2 (1.00×)	275.5 (1.00×)	79.27	—	—	—	100
Dense inference (reseed)	—	—	80.44	93.76	928.08	0.684	111.6
Output interpolation	187.1 (1.60×)	162.5 (1.70×)	67.92	129.07	2821.06	0.636	91.9
BlockDelta	188.2 (1.60×)	163.4 (1.69×)	77.61	91.60	1138.19	0.532	63.8

MiniMax-H3. MiniMax-H3 is a joint audio–video model. We use its 8-step distilled checkpoint at 768×1344 and 124 frames, on 100 T2V prompts with shared seeds. The configuration was set once from the Wan2.2 configuration before any run and not changed afterwards: dense blocks $\{0, 1, 44\text{–}49\}$, stride 3 with the anchor offset changing from block to block, and two trailing dense steps.

Its VAE encodes every 17 pixel frames independently into 5 latent frames, which gives 37 latent frames for 124 pixel frames with one seam latent between clips, and the VAE-aligned anchors of Section 4.3 are placed at the clip boundaries. Both sparse rows use these anchors and differ only in the reconstruction target. Residual interpolation is better than output interpolation in VBench-sub, FID, FVD and LPIPS (Table 4), but its Dyn. is lower (63.8 against 91.9). It is closer to dense inference than output interpolation on 98 of 100 paired videos in LPIPS ($p = 8 \times 10^{-27}$), and its FID is below that of Dense inference (reseed). Its VBench-sub is 1.66 points below that of dense inference, more than the 1.17-point difference between dense inference and Dense inference (reseed), and its Dyn. of 63.8 lies in the lower part of the range of the benchmark models. The audio tokens are not skipped. The log-mel distance of residual interpolation to the dense soundtrack is 0.31, against 0.60 for Dense inference (reseed) and 0.42 for output interpolation, and its FAD in CLAP embedding space is 0.13, against 0.17 for Dense inference (reseed). Audio–visual synchrony was not measured.

F Corrector: Instantiations and Measured Recovery

The learned corrector lowers FVD on the two I2V models but increases FID on all three models; it is therefore not part of the training-free BlockDelta of Table 1. Table 5 compares BlockDelta with and without the corrector on the ablation prompts and seeds of each model. The corrector moves the output toward the mean of the dense prediction and lowers per-frame sharpness, and longer training does not change this. This appendix gives the training setup, the relation between the FVD change and the distance to dense inference, two checks of the training, and the runtime cost.

Table 5: Optional corrector on the ablation prompts of each model (same protocol and columns as Tables 2 and 7). BlockDelta is the configuration of Table 1, and the row below it adds the corrector with the same prompts and seeds. Gray rows are dense inference.

Model	Configuration	FID ↓	FVD ↓	Head-flash ↓	Dyn. (%)	DiT s ↓
Wan2.2 T2V	Dense inference	—	—	0.78	100	69.4
	Dense inference (reseed)	81.7	1098.2	0.75	99.7	—
	BlockDelta	89.7	1174.3	0.67	53.6	36.9
	BlockDelta + corrector	95.0	1422.1	0.82	59.3	37.1
Wan2.2 I2V	Dense inference	—	—	0.40	100	65.9
	Dense inference (reseed)	16.1	201.0	0.42	97.3	—
	BlockDelta	21.2	553.3	0.69	80.3	35.2
	BlockDelta + corrector	25.2	338.1	0.40	88.1	35.1
HunyuanVideo-1.5 I2V	Dense inference	—	—	0.63	100	62.5
	Dense inference (reseed)	16.4	200.2	0.58	101.1	—
	BlockDelta	19.9	283.5	0.90	93.3	35.8
	BlockDelta + corrector	27.3	266.1	0.59	74.2	36.1

Training setup. The corrector consists of rank-64 LoRA adapters with scaling $\alpha = 64$ on the attention and MLP projections of the last dense blocks. It is applied at every sparse step and at no dense step. The training inputs are latents collected from the deployed sparse rollout, and each label is the dense model's velocity prediction on the same latent. The training forward pass uses the same sparse execution as deployment, and the base DiT weights stay frozen. All models use AdamW with learning rate 10^{-4}, batch size 1 and gradient clipping at 1.0, and stop early when the held-out error to the dense prediction has not improved for 3 validations. The training prompts cover people, animals, objects and scenes, and do not overlap with any evaluation prompt. The small training set follows LearniBridge (Huang et al., 2026b), which skips all blocks except the last one at cached denoising steps and trains LoRA adapters on that block from 3–5 prompts; it reports little further gain beyond 5 prompts.

On Wan2.2 T2V, the adapters are placed in blocks $\{38, 39\}$ of the high-noise expert and are active at steps $\{0, 1, 2\}$; step 3 is dense and uses the low-noise expert without an adapter. The training set contains 10 prompts and 30 states, with 2 held-out prompts and 6 states. Validation runs every 200 updates, and the selected checkpoint is at update 200.

On HunyuanVideo-1.5 I2V, the adapters are placed in blocks $\{51, 52, 53\}$ and are active at the six sparse steps $\{0, \ldots, 5\}$; steps $\{6, 7\}$ are dense without correction. The training set contains 12 prompts and 72 states, with 4 held-out prompts and 24 states. Validation runs every 50 updates, and the selected checkpoint is at update 700.

FVD change and distance to dense inference. The FVD change depends on how far the configuration without the corrector is from dense inference. Relative to dense inference with a different seed, which gives the FVD of a seed change alone (201.0 on Wan2.2 I2V, 200.2 on HunyuanVideo-1.5 I2V and 1098.2 on Wan2.2 T2V), the FVD without the corrector is 2.75 times larger on Wan2.2 I2V, 1.42 times on HunyuanVideo-1.5 I2V and 1.07 times on Wan2.2 T2V. The corrector gives a large FVD decrease on the first model, a small decrease on the second and an increase on the third.

Training length and training states. On Wan2.2 T2V, we changed one training factor at a time and evaluated each new corrector on the same 100 prompts as Table 5. Training for 3,000 updates without early stopping, 15 times the selected length, gives FID 94.5 and FVD 1485. One round of DAgger, which collects the training states from the corrector's own rollout instead of the rollout without the corrector, gives FID 95.5 and FVD 1409 with early stopping, and FID 99.2 and FVD 1527 with 3,000 updates. All three stay close to the selected corrector (FID 95.0, FVD 1422.1) and above the configuration without the corrector (FID 89.7, FVD 1174.3). The corrector also reduces the spread of the output. The standard deviation of the final latent relative to the same-seed dense latent is 0.96 without the corrector and 0.87 with it, and remains between 0.87 and 0.88 for all four trained correctors. The median per-frame sharpness, measured as the variance of the Laplacian relative to the same-seed dense video, is 1.00 without the corrector and 0.71 with it. The corrector therefore moves the output toward the mean of the dense prediction instead of recovering its details, which agrees with the higher FID. More training or training on the corrector's own states therefore does not change the result on this model.

Steps covered on HunyuanVideo-1.5. An earlier HunyuanVideo-1.5 corrector was active only at steps $\{0, 1, 2\}$, with the same host blocks and training recipe. On the prompts of Table 2, it gives FID 21.6, FVD 236.7 and Dyn. 82.4. Extending the corrector to all six sparse steps lowers its held-out error to the dense prediction from 0.153 to 0.119, but gives FID 27.3, FVD 266.1 and Dyn. 74.2, which are all worse than those of the three-step corrector. As with the interpolation weights in Section 5.3, a lower error to the dense prediction does not give a video closer to dense inference. The comparison above uses the six-step corrector because it matches the definition in Section 4.3.

Runtime cost. The adapters are not merged into the base weights; each sparse step therefore adds two low-rank matrix products in each host projection. Relative to the configuration without the corrector timed in the same window, the DiT time changes by +0.2% on Wan2.2 T2V, +0.1% on Wan2.2 I2V and −0.9% on HunyuanVideo-1.5 I2V; the negative value shows that these changes are within timing variation.

G Step Reduction and Step Reuse

This section compares two ways of reducing computation across denoising steps: using fewer steps and reusing features from previous steps. All methods run on the 4-step distilled Wan2.2 T2V and I2V models of Section 5 at 832×480 and 81 frames, on 100 prompts per task with shared seeds, on one H20. Speedup is the E2E speedup over dense inference with the timing protocol of Appendix D. FID and FVD are computed against dense inference, and LPIPS is paired with the same-seed dense video. The reduced-step rows drop the last one or two of the four steps. TeaCache is run with thresholds 0.2 and 0.6, and TaylorSeer with refresh intervals 5 and 10. At both thresholds on T2V and at threshold 0.2 on I2V, TeaCache never finds two steps similar enough to reuse, and its videos are identical to those of dense inference; at threshold 0.6 on I2V, it reuses one step. Both TaylorSeer intervals are longer than the four-step schedule, and the two settings give identical videos on each task. On the 8-step HunyuanVideo-1.5 I2V model, all methods run at 480p and 81 frames on the 100 prompts of Table 2. The reduced-step rows use 6 or 4 steps of the same noise schedule. TeaCache and TaylorSeer use the implementations in the official HunyuanVideo-1.5 code, which reuse the features of fixed steps instead of testing a threshold; the tested settings skip two or four of the eight steps for TeaCache and two or three for TaylorSeer.

Table 6: Step reduction and step reuse on the 4-step Wan2.2 T2V (top) and I2V (middle) models and on the 8-step HunyuanVideo-1.5 I2V model (bottom). The reduced-step rows use dense inference with fewer denoising steps. On HunyuanVideo-1.5, the number in parentheses is the number of skipped steps, and the BlockDelta + TeaCache row is timed in the same session as the other rows. Gray rows are dense inference with different seeds. TeaCache on T2V and TeaCache (0.2) on I2V give videos identical to dense inference, and the two TaylorSeer settings give identical videos on each task.

	Method	E2E speedup	FID ↓	FVD ↓	LPIPS ↓
Wan2.2 T2V	Dense inference (reseed)	1.00×	81.7	1098.2	0.678
	Dense inference, 3 steps	1.30×	66.5	873.1	0.516
	Dense inference, 2 steps	1.84×	102.8	1494.8	0.632
	TeaCache (0.2 / 0.6)	1.00×	0	0	0
	TaylorSeer (5 / 10)	1.64×	121.5	1778.8	0.654
Wan2.2 I2V	Dense inference (reseed)	1.00×	16.1	201.0	0.217
	Dense inference, 3 steps	1.28×	8.6	101.2	0.083
	Dense inference, 2 steps	1.76×	16.7	188.9	0.121
	TeaCache (0.2)	1.00×	0	0	0
	TeaCache (0.6)	1.27×	26.0	324.1	0.129
	TaylorSeer (5 / 10)	1.58×	22.8	305.9	0.159
HunyuanVideo-1.5 I2V	Dense inference (reseed)	1.00×	16.4	200.2	0.254
	Dense inference, 6 steps	1.31×	5.3	36.0	0.059
	Dense inference, 4 steps	1.79×	11.9	104.1	0.127
	TeaCache (2)	1.30×	1.5	8.1	0.010
	TeaCache (4)	1.79×	9.3	71.8	0.042
	TaylorSeer (2)	1.31×	1.2	7.3	0.011
	TaylorSeer (3)	1.51×	5.7	40.1	0.039
	BlockDelta + TeaCache (2)	1.88×	19.8	281.3	0.178

Step reuse gives limited gains on these 4-step models at the tested settings. TeaCache gives no speedup at both thresholds on T2V and at the lower threshold on I2V, and 1.27× at the higher I2V threshold. TaylorSeer is slower than dense inference with two steps and further from dense inference on both tasks. On these models, using fewer steps therefore gives a better trade-off than reusing features across steps.

On the 8-step HunyuanVideo-1.5 I2V model, step reuse is effective. TeaCache and TaylorSeer are as fast as dense inference with the same number of computed steps and closer to dense inference: TeaCache with four skipped steps reaches 1.79× with FID 9.3 and FVD 71.8, against 11.9 and 104.1 for dense inference with 4 steps. All these rows stay below the FID and FVD of Dense inference (reseed). In the same timing session, BlockDelta reaches 1.61× with FID 19.9 and FVD 283.5 (Table 2), and is therefore further from dense inference than step reduction and step reuse at similar or higher speedups. The two kinds of acceleration can be combined. Adding TeaCache with two skipped steps to BlockDelta raises the speedup to 1.88×, with FID 19.8 and FVD 281.3, almost the same as BlockDelta alone.

Step reduction and BlockDelta differ from dense inference in different ways. In the same timing session as Table 6, BlockDelta reaches 1.75× on Wan2.2 T2V and 1.68× on Wan2.2 I2V. On Wan2.2 T2V, BlockDelta is closer to dense inference than dense inference with 2 steps at 1.84× (FID 89.7 vs. 102.8, FVD 1174.3 vs. 1494.8; Tables 6 and 7). On Wan2.2 I2V, dense inference with 2 steps is faster (1.76× vs. 1.68×) and closer to dense inference (FID 16.7 vs. 21.2, FVD 188.9 vs. 553.3). Figure 7 shows an example. In the dense video, the pan is lifted off the stove and the camera moves up. Dense inference with 2 steps follows this motion but shows two overlapping pans and blurred hands, whereas BlockDelta keeps each frame sharp but leaves the pan on the stove and the camera still.



Figure 7: Frames 0, 35, 50, 65 and 80 of one Wan2.2 I2V video, with the same input image and seed. Rows: dense inference with 4 steps (Baseline), dense inference with 2 steps (2-step), and BlockDelta (Ours).

H Component Ablation Suite

This section reports the leave-one-out ablation of Table 2 on the two Wan2.2 models (Table 7), the role of the dense-block set on all three models (Table 8), and the sensitivity to anchor stride and dense-step budget (Tables 9 and 10). All tables start from BlockDelta, including the VAE-aligned anchors, and change one setting at a time. On the two Wan2.2 models, the σ-aligned rule keeps the same single dense step as FIS-DiT, and Table 7 therefore has no row that removes it. The row without dense blocks in Tables 2 and 7 is the "none" row of Table 8. All configurations of a model share the same 100 prompts (Appendix D.2), seeds and machine. FID and FVD are computed against dense inference generated in the same batch and, with 100 videos, are not comparable to the values of Table 1. In Tables 9 and 10, a larger stride or one fewer dense step raises the speedup and lowers fidelity on all three models, and one more dense step raises fidelity and lowers the speedup. The stride and the noise threshold are the same on all models and are not tuned per model.

VAE-aligned anchors on Wan2.2 T2V. Table 7 shows the same pattern as Table 2, except for the VAE-aligned anchors on Wan2.2 T2V, which has no conditioning image. Removing them leaves Head-flash equal to that of dense inference (0.78) and lowers FID slightly from 89.7 to 87.3, whereas FVD rises from 1174.3 to 1277.3.

Table 7: Leave-one-out ablation on the Wan2.2 models (same protocol and columns as Table 2, with the DiT speedup over dense inference added). Each w/o row removes one component from BlockDelta (last row of each model) and keeps the others; the σ-aligned rule has no row because it coincides with the single dense step of FIS-DiT on these models. Gray rows are dense inference.

Model	Configuration	FID ↓	FVD ↓	Head-flash ↓	Dyn. (%)	DiT s ↓	Speed ↑
Wan2.2-A14B T2V (4 steps)	Dense inference	—	—	0.78	100	69.4	1.00×
	Dense inference (reseed)	81.7	1098.2	0.75	99.7	—	—
	FIS-DiT	116.9	1484.9	0.53	24.7	35.4	1.96×
	w/o residual interpolation (output interpolation)	117.6	1401.2	0.38	23.5	36.5	1.90×
	w/o dense-block set (no dense blocks)	104.4	1450.8	0.54	39.7	34.1	2.04×
	w/o VAE-aligned anchors	87.3	1277.3	0.78	52.0	35.5	1.95×
	BlockDelta	89.7	1174.3	0.67	53.6	36.9	1.88×
Wan2.2-A14B I2V (4 steps)	Dense inference	—	—	0.40	100	65.9	1.00×
	Dense inference (reseed)	16.1	201.0	0.42	97.3	—	—
	FIS-DiT	200.9	2849.0	5.76	148.2	33.8	1.95×
	w/o residual interpolation (output interpolation)	172.2	2465.3	4.75	141.7	34.7	1.90×
	w/o dense-block set (no dense blocks)	57.9	2299.1	2.24	105.9	32.5	2.03×
	w/o VAE-aligned anchors	22.4	598.1	2.70	83.7	33.7	1.95×
	BlockDelta	21.2	553.3	0.69	80.3	35.2	1.87×

Dense-block set. Keeping both ends of the block stack dense gives the best or near-best result on all three models, and the result depends strongly on the position of the dense blocks. Table 8 changes only the set of dense blocks. On HunyuanVideo-1.5 I2V and Wan2.2 T2V, the head blocks carry most of the effect: keeping only the tail blocks dense gives no gain over no dense blocks, whereas keeping only the head blocks lowers FID to 31.3 and 90.0. Without the head blocks, the colors also move away from those of dense inference. On HunyuanVideo-1.5 I2V, Chroma falls from 36.81 to 13.57 with no dense blocks and to 11.64 with only the tail blocks, against 34.21 for dense inference. On Wan2.2 T2V, it rises from 51.15 to 62.86 and 59.37, against 50.33. On Wan2.2 I2V, the head blocks and the tail blocks lower FID from 57.9 to 32.8 and 31.7, respectively. The tail blocks improve on the head blocks on the two I2V models, lowering FVD from 1409.3 to 553.3 on Wan2.2 I2V and from 455.9 to 283.5 on HunyuanVideo-1.5 I2V. Wan2.2 T2V is less sensitive to the tail blocks: adding them changes FID from 90.0 to 89.7 and FVD from 1143.5 to 1174.3. A random set with the same number of blocks is clearly worse than the head and tail set on all three models, with FID 110.3, 38.3 and 117.2, although on Wan2.2 it contains block 1 next to the head.

Table 8: Role of the dense-block set on BlockDelta, including the VAE-aligned anchors; only the dense-block set varies. The head and tail set is the deployed one, inherited from FIS-DiT, and the random set has the same number of blocks. Speed is the DiT speedup over dense inference. Chroma of dense inference is 50.33, 37.03 and 34.21 on the three models.

Model	Dense blocks	FID ↓	FVD ↓	Chroma	Dyn. (%)	Speed ↑
Wan2.2 T2V	none	104.4	1450.8	62.86	39.7	2.04×
	head $\{0\}$	90.0	1143.5	54.22	53.5	1.99×
	tail $\{38, 39\}$	105.4	1379.5	59.37	41.0	1.94×
	head+tail (deployed)	89.7	1174.3	51.15	53.6	1.88×
	random	110.3	1577.6	58.45	37.5	1.89×
Wan2.2 I2V	none	57.9	2299.1	41.91	105.9	2.03×
	head $\{0\}$	32.8	1409.3	38.31	95.0	1.98×
	tail $\{38, 39\}$	31.7	1468.4	38.81	87.2	1.93×
	head+tail (deployed)	21.2	553.3	36.82	80.3	1.87×
	random	38.3	1666.4	39.43	91.3	1.88×
HunyuanVideo-1.5 I2V	none	202.3	1852.4	13.57	103.2	2.06×
	head $\{0$–$5\}$	31.3	455.9	41.34	110.0	1.84×
	tail $\{51$–$53\}$	237.9	2043.4	11.64	107.4	1.94×
	head+tail (deployed)	19.9	283.5	36.81	93.3	1.75×
	random	117.2	1203.8	32.84	78.5	1.74×

Table 9: Sensitivity to the inherited anchor stride on BlockDelta, including the VAE-aligned anchors; only the stride varies. Speed is the DiT speedup over dense inference.

Model	Stride	FID ↓	FVD ↓	Dyn. (%)	Speed ↑
Wan2.2 T2V	3 (deployed)	89.7	1174.3	53.6	1.88×
	4	94.8	1327.6	44.1	2.04×
	5	95.2	1219.9	37.5	2.14×
Wan2.2 I2V	3 (deployed)	21.2	553.3	80.3	1.87×
	4	23.7	854.1	82.8	2.04×
	5	29.5	971.6	83.1	2.14×
HunyuanVideo-1.5 I2V	3 (deployed)	19.9	283.5	93.3	1.75×
	4	24.1	398.9	104.8	1.90×
	5	28.7	511.1	111.7	1.98×

I Scope beyond Few-step Models

This section tests BlockDelta on a model that is not step-distilled. Figure 8 compares dense inference and BlockDelta on the original undistilled Wan2.2-A14B T2V model at 10, 20, 40 and 60 denoising

Table 10: Sensitivity to the dense-step budget on BlockDelta, including the VAE-aligned anchors; the neighboring settings remove or add one dense step. On the two Wan2.2 models, −1 keeps no dense step. Speed is the DiT speedup over dense inference.

Model	Dense-step budget	FID ↓	FVD ↓	Dyn. (%)	Speed ↑
Wan2.2 T2V	−1	93.1	1579.8	50.6	2.68×
	σ-rule (deployed)	89.7	1174.3	53.6	1.88×
	+1	83.1	1073.2	66.2	1.45×
Wan2.2 I2V	−1	40.0	1050.4	86.7	2.68×
	σ-rule (deployed)	21.2	553.3	80.3	1.87×
	+1	18.7	302.8	83.2	1.45×
HunyuanVideo-1.5 I2V	−1	27.1	634.3	108.2	1.99×
	σ-rule (deployed)	19.9	283.5	93.3	1.75×
	+1	16.1	200.6	93.5	1.55×

steps. Wan2.2-A14B uses 40 steps by default, and the 60-step setting extends the tested range beyond this default. All configurations use the same 100 prompts and seeds, and quality is measured against the official 40-step dense configuration.

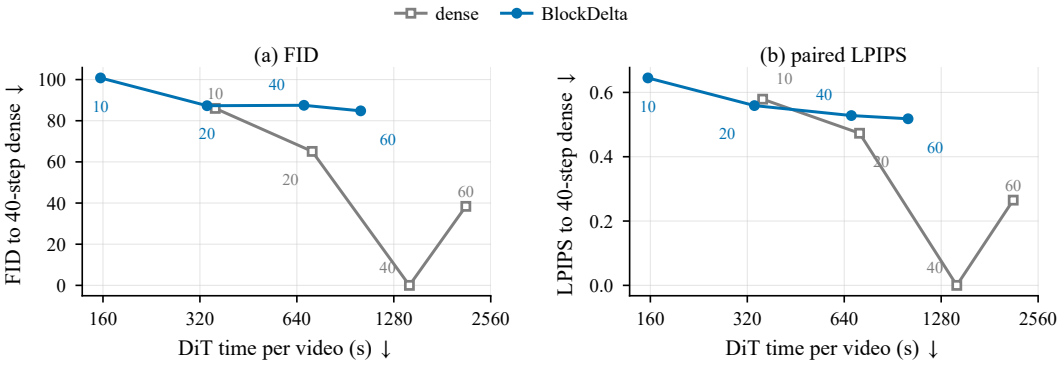


Figure 8: Quality–compute curves on the undistilled Wan2.2-A14B T2V model. Numbers beside the points denote denoising steps; 40 is the official default and 60 is an above-default extension. FID and paired LPIPS are measured against 40-step dense inference on 100 matched prompts and seeds; compute is DiT time per video on one H20. BlockDelta changes little beyond 20 steps, while reducing the number of dense steps gives better fidelity at the higher matched-compute point.

On this undistilled model, BlockDelta is not better than reducing the number of dense steps. At similar compute, 20-step BlockDelta and 10-step dense inference are close: their FID values are 87.3 and 86.0, and their LPIPS values are 0.559 and 0.579. At the next matched-compute point, however, 40-step BlockDelta is worse than 20-step dense inference in both FID (87.5 versus 65.1) and LPIPS (0.528 versus 0.473). Increasing BlockDelta from 20 to 60 steps yields only a small improvement relative to the official 40-step dense configuration.

The small change of BlockDelta beyond 20 steps does not come from a growing difference to dense inference at the same number of steps: paired LPIPS decreases from 0.568 at 10 steps to 0.504 at 60 steps. The result is consistent with interpolation omitting frame-specific residual components that additional sparse steps do not recover, but this sweep does not isolate that mechanism because changing the step count also changes the denoising step size and the number of sparse updates. The generated videos retain only 66–70% of the 60-step dense motion proxy across the tested settings. On the original Wan2.2-A14B schedule, direct step reduction is therefore more effective at the higher matched-compute point. BlockDelta instead targets distilled models with few steps, such as the 4-step Wan2.2 models, on which step reuse gives limited gains (Appendix G).

J Limitations and Future Work

BlockDelta reduces computation within each denoising step and reconstructs skipped frames by interpolating block residuals, which cannot recover the frame-specific part of these residuals. The motion carried by this part is therefore partly lost, most on Wan2.2 T2V (Table 7). Other acceleration methods differ from dense inference in other ways. Attention sparsity computes every frame and stays closer to dense inference in FID and FVD at a smaller speedup, and it can be combined with BlockDelta for a larger speedup (Table 1). Step reduction removes whole steps; on Wan2.2 I2V it is closer to dense inference in FID and FVD, and in the example of Figure 7 it follows the motion of dense inference but renders it with ghosting and blur (Appendix G). On the 8-step HunyuanVideo-1.5 model, step reduction and step reuse are closer to dense inference at similar speedups, and adding step reuse to BlockDelta raises its speedup with almost no change in FID and FVD (Appendix G); combining BlockDelta with fewer steps is not evaluated. On an undistilled many-step model, whose computation lies mainly in the number of steps, reducing the number of dense steps gives better fidelity at the higher matched-compute point (Appendix I). The analysis in Section 4.2 relies on the idealized Assumption 1; it explains why intermediate weightings are needed, and the final weighting is selected by video-level comparisons. Recovering the frame-specific component, for example with a corrector that does not move the output toward the mean of the dense prediction, is the main direction for future work.

K Qualitative Comparisons

Figures 9–16 show frames of videos generated on the 100 ablation prompts of Appendix D.2 (benchmark models) and on the 100 prompts of Appendix E.2 (LTX-2.3 and MiniMax-H3). All rows of a panel use the same prompt and seed, and each row shows five frames at the same positions, evenly spaced from the first to the last frame (frames 0, 20, 40, 60 and 80 of 81 on the benchmark models, 0, 48, 96, 144 and 192 of 193 on LTX-2.3, and 0, 31, 62, 92 and 123 of 124 on MiniMax-H3). The videos are included in the supplementary material. BlockDelta is the training-free configuration of Table 1, without the corrector of Appendix F. On the benchmark models, FIS-DiT and BlockDelta differ in the reconstruction target and in the sparse execution rules of Section 4.3; Table 1 separates the two, and most of the difference comes from the target.

I2V models. On Wan2.2 I2V and HunyuanVideo-1.5 I2V (Figures 11–14), BlockDelta stays close to dense inference in content, color and camera motion. The visible differences are small, for example the lioness closes its mouth earlier than in dense inference. On the 100 ablation prompts, the FID of BlockDelta is 21.2 and 19.9, against 16.1 and 16.4 for Dense inference (reseed) (Tables 7 and 2). FIS-DiT keeps the first frame, which follows the conditioning image, and degrades in the later frames, and the failure differs between the two models. On Wan2.2 I2V, colored high-frequency noise covers the interior frames, and the last frame shows less of it. On HunyuanVideo-1.5 I2V, the colors and contrast drift away from dense inference over time: the scene turns dark blue or black, bright regions saturate, and the structure of the conditioning image is gradually lost. On the same prompts, FIS-DiT has an FID of 200.9 and 219.4 and a Dyn. of 148.2 and 220.6, above the 100 of dense inference, which illustrates why a Dyn. at or above 100 does not show that motion is preserved (Appendix D.3). Output reconstruction replaces the known input of each skipped frame by an estimate, and the error of this estimate is carried to all later blocks (Section 4.1); the figures show the result but do not isolate why it is much larger on the I2V models than on Wan2.2 T2V.

Wan2.2 T2V. On Wan2.2 T2V (Figures 9 and 10), neither sparse method reproduces the same-seed dense video: the layout, the subject and the style often change, for example a woman becomes a boy in “a carrot and an umbrella”. On the full benchmark, the LPIPS of BlockDelta to the same-seed dense video is 0.5916, against 0.6671 for Dense inference (reseed), and its VBench score is 82.49, against 82.36 for dense inference (Table 1). FIS-DiT additionally turns several videos into abstract, saturated patterns, for example the colored tiles of “art gallery”, the colored strokes around the cutlery in “a fork on the right of a knife”, or into a saturated close-up that drops the rest of the prompt, as in “a carrot and an umbrella”. The motion loss of BlockDelta on this model (Dyn. 53.6 in Table 7) is hard to judge from these frames, because the videos differ from the dense videos in content.

LTX-2.3 and MiniMax-H3. On these two models (Figures 15 and 16), BlockDelta also changes the layout of some videos relative to the same-seed dense video, for example the framing of the couple and of the boat, while its LPIPS in Table 4 (0.6125 and 0.532) is below that of Dense inference (reseed) (0.7335 and 0.684). The actions named in the prompts, such as the horse and the zebra lowering their heads to the water and the knife cutting the watermelon, are preserved.



Figure 9: Qualitative comparison on Wan2.2-A14B T2V (4 steps), part 1. Rows: dense inference, FIS-DiT and BlockDelta, with the same prompt and seed.

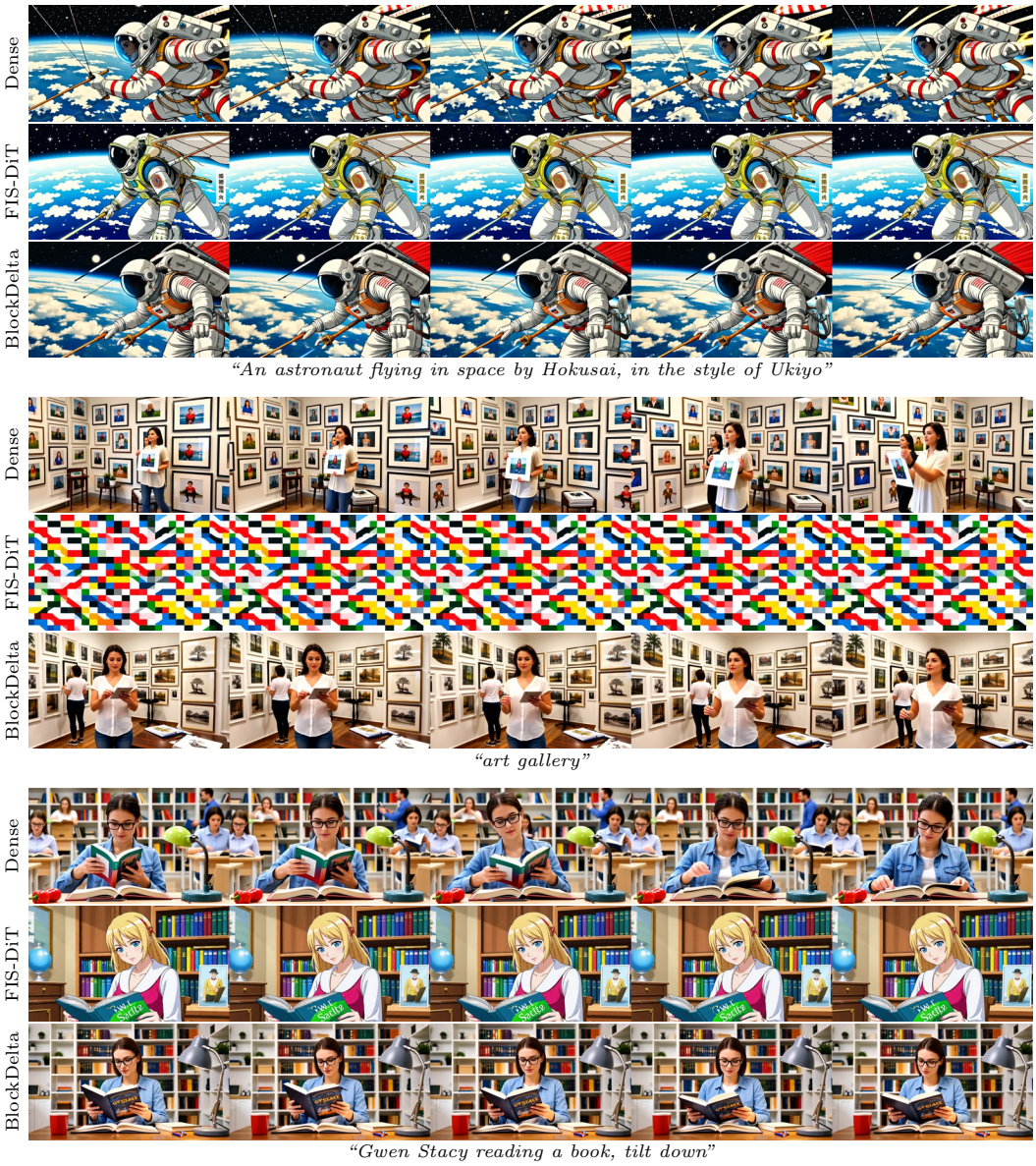


Figure 10: Qualitative comparison on Wan2.2-A14B T2V (4 steps), part 2. Rows as in Figure 9.

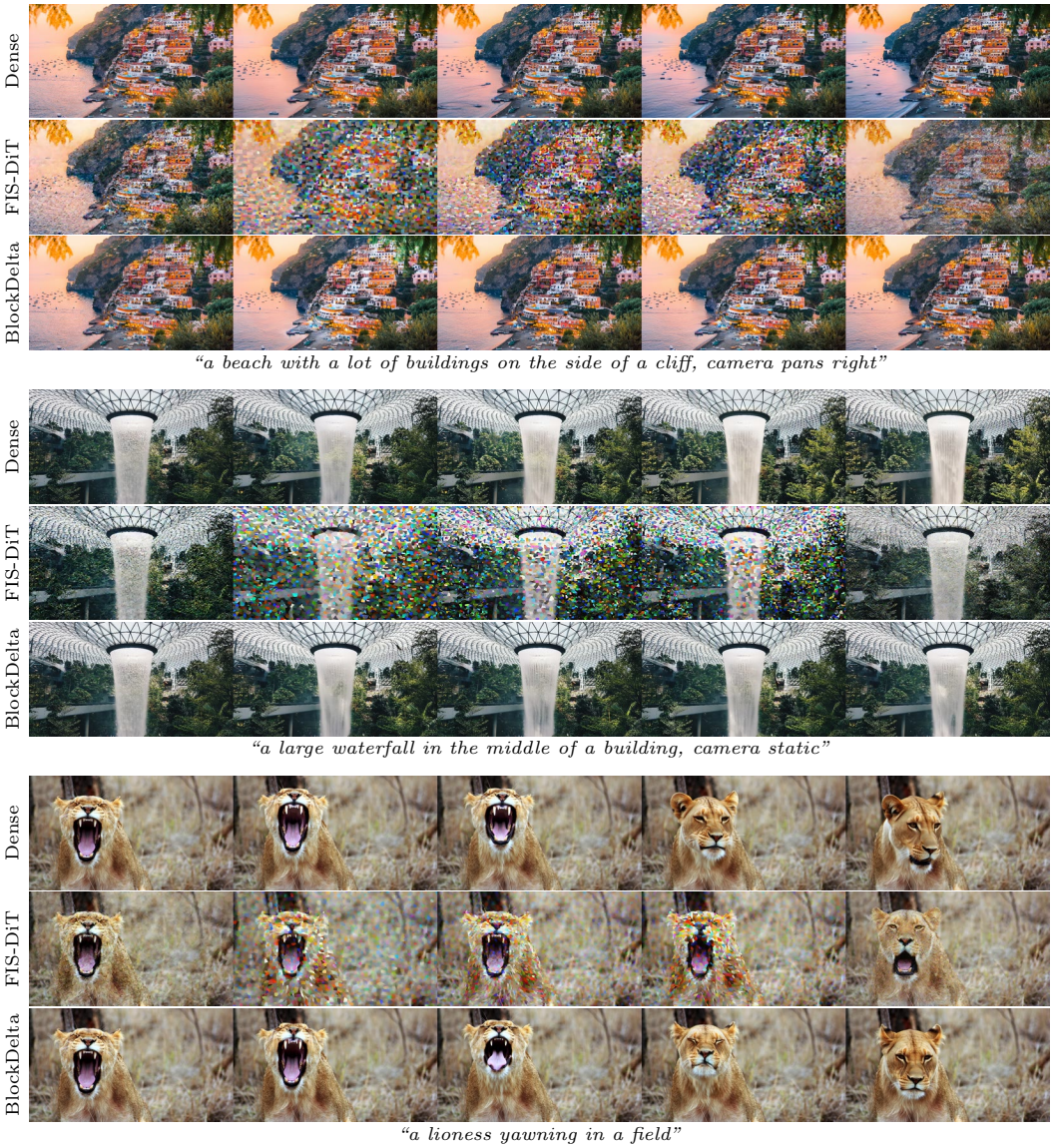


Figure 11: Qualitative comparison on Wan2.2-A14B I2V (4 steps), part 1. Rows as in Figure 9.

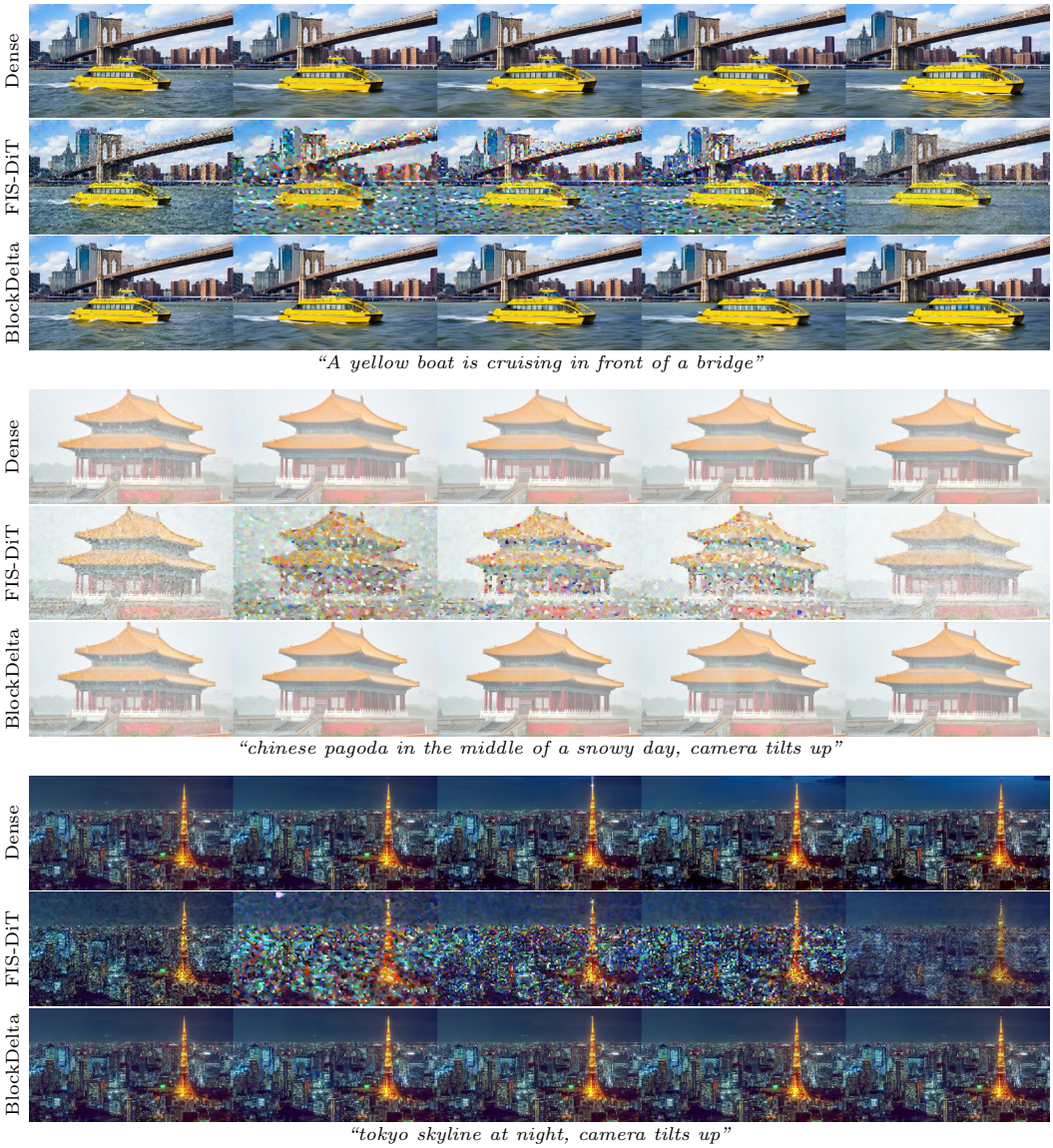


Figure 12: Qualitative comparison on Wan2.2-A14B I2V (4 steps), part 2. Rows as in Figure 9.



Figure 13: Qualitative comparison on HunyuanVideo-1.5 I2V (8 steps), part 1. Rows as in Figure 9.

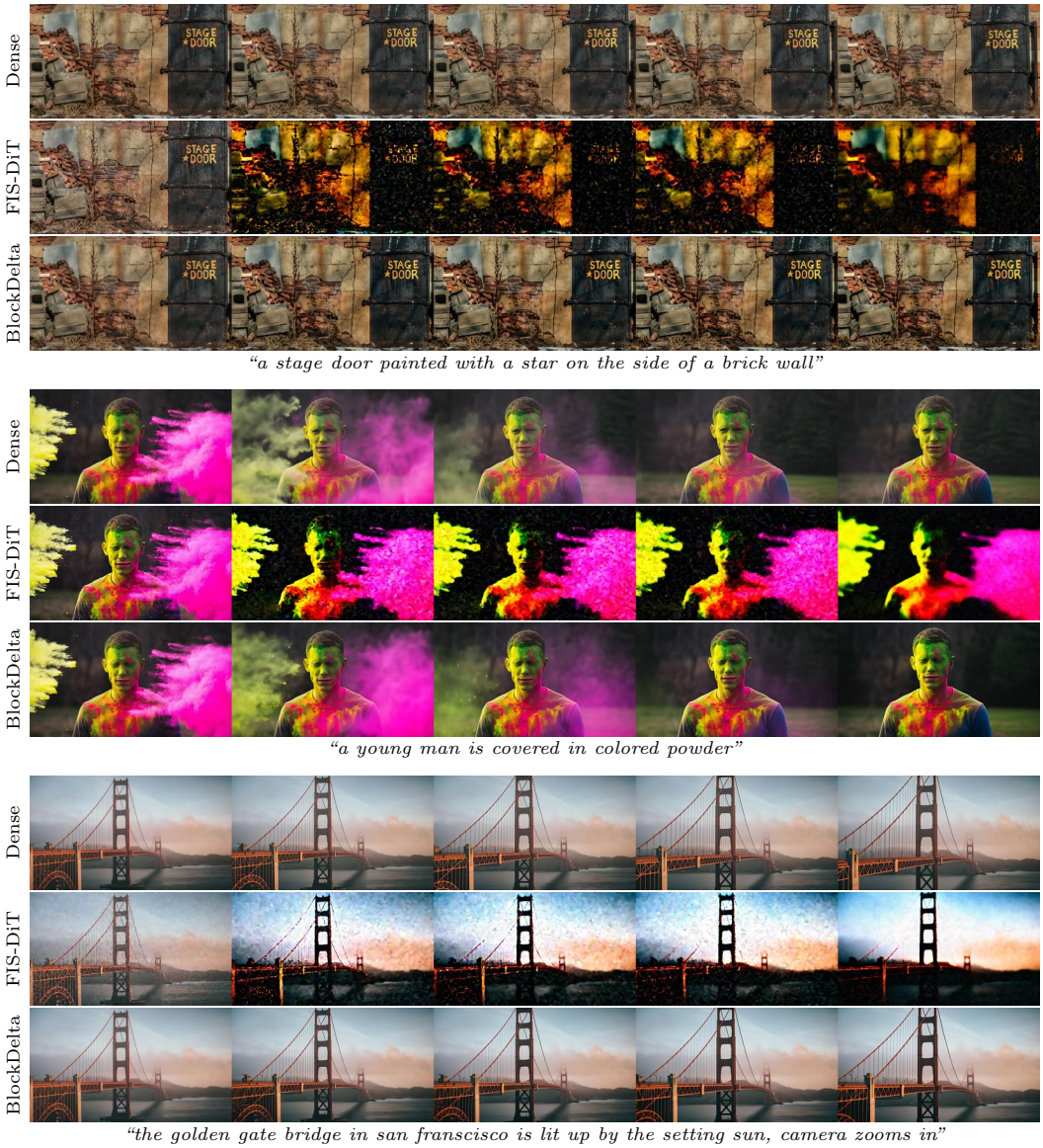


Figure 14: Qualitative comparison on HunyuanVideo-1.5 I2V (8 steps), part 2. Rows as in Figure 9.

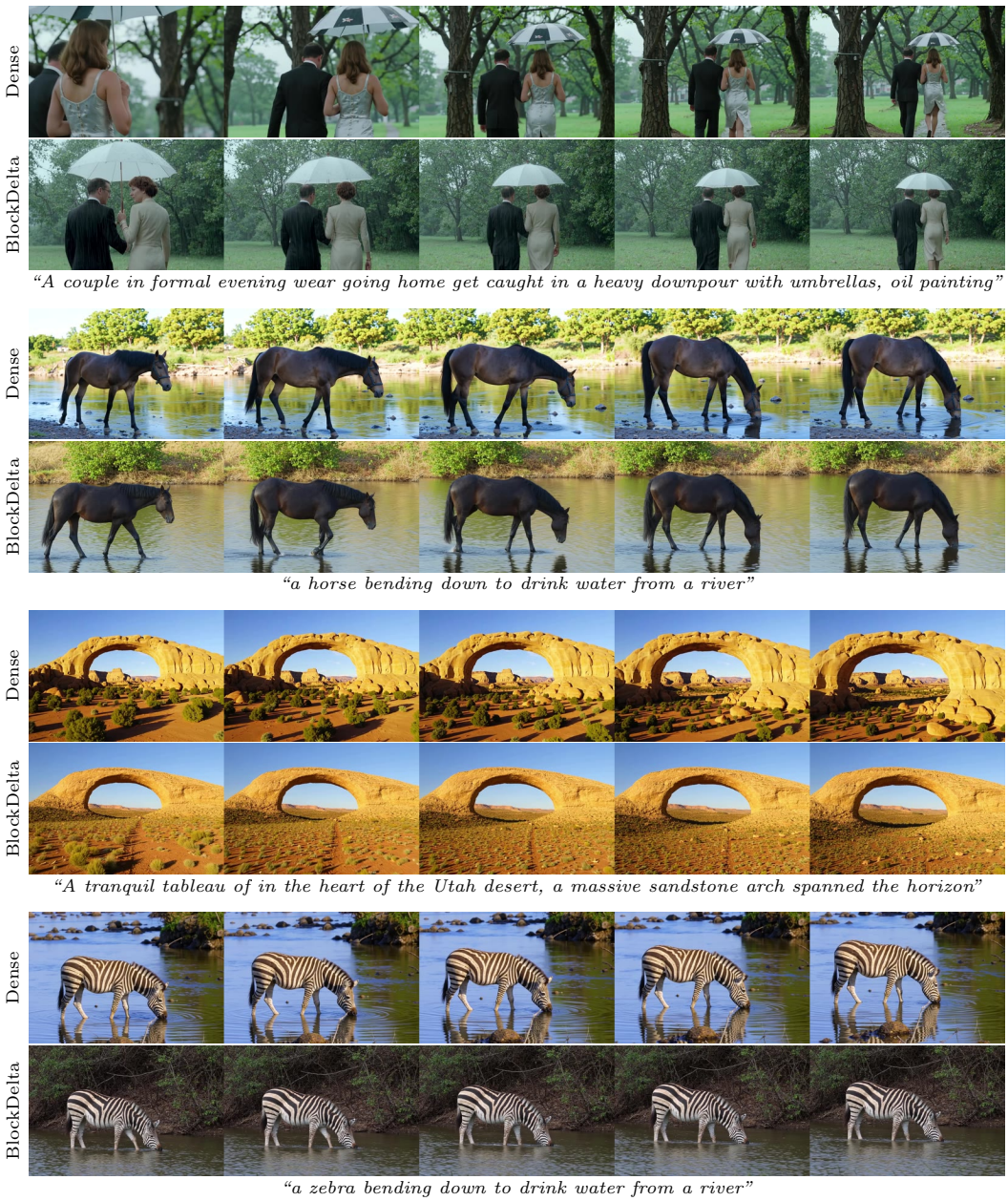


Figure 15: Qualitative comparison on LTX-2.3 (T2AV, 8+3 steps). Rows: dense inference and BlockDelta, with the same prompt and seed.

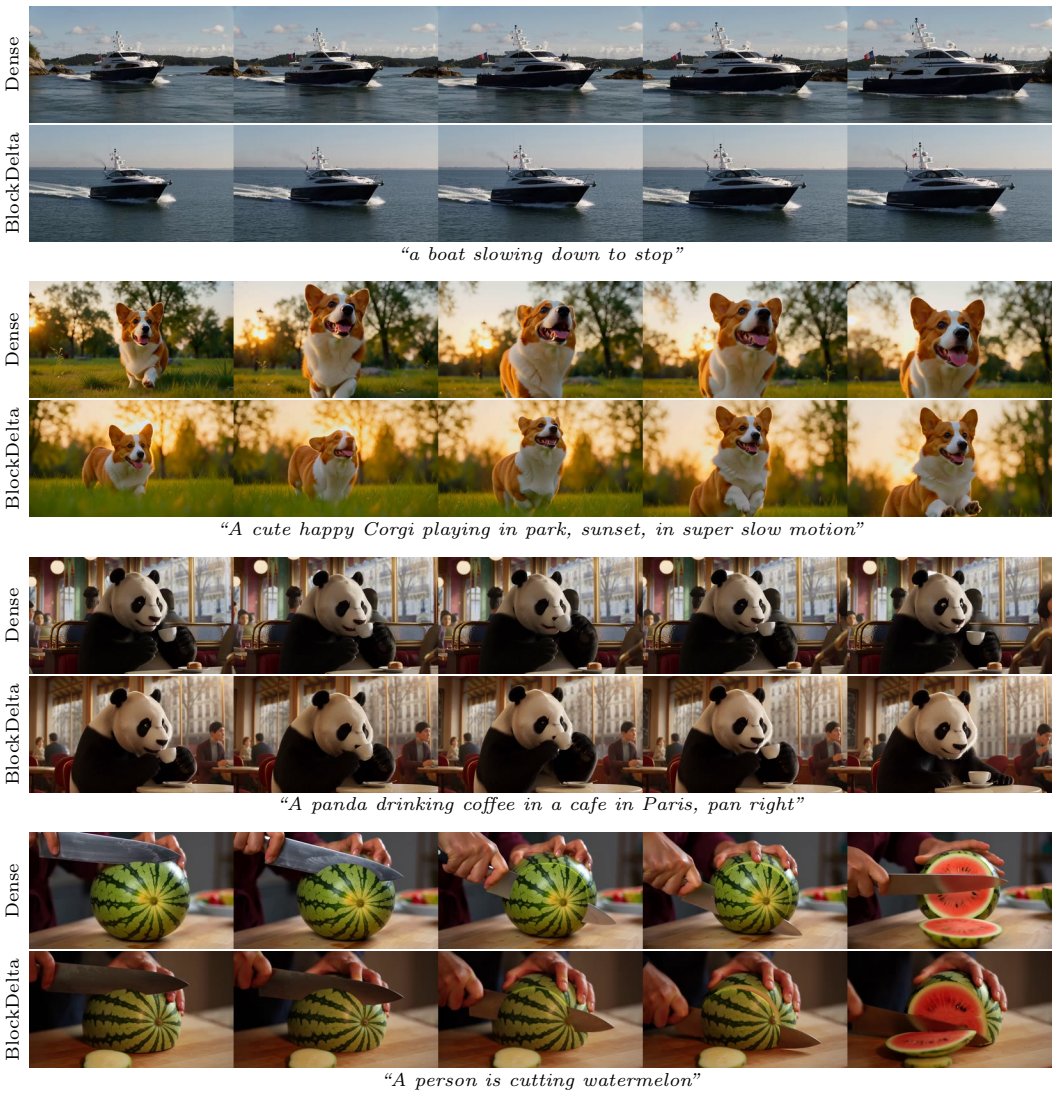


Figure 16: Qualitative comparison on MiniMax-H3 (T2AV, 8 steps). Rows as in Figure 15.