PEBBLEVIDEO: ON-DEVICE VIDEO GENERATION VIA EXTREME MODEL COMPRESSION

Anonymous authors
Paper under double-blind review



Figure 1: **Videos generated by PebbleVideo.** Despite aggressive compression, PebbleVideo produces detailed, temporally coherent videos in four denoising steps.

ABSTRACT

Diffusion Transformers (DiTs) have substantially advanced video generation quality, yet their quadratic bidirectional attention and billion-scale parameter incur significant memory and storage overhead, hindering deployment on resource-constrained devices. Existing approaches improve efficiency through attention redesign or streaming generation, but their restricted global spatiotemporal interactions limit modeling capacity. In contrast, we explore an alternative direction, model compression, and introduce **PebbleVideo**, which aggressively reduces the parameter footprint through structured compression. The core of PebbleVideo is to evaluate compression sensitivity in functional space, preserving generation-critical functions rather than relying on indirect proxies. For the dominant DiT backbone, we measure structural importance through the perturbation induced on the diffusion vector field, directly exposing each module's influence on generation dynamics. For heterogeneous condition encoders, cross-scale representation consistency determines whether compression proceeds through compact-model substitution or in-place structural pruning. Combined with DMD-based distillation, PebbleVideo achieves 512×320 video generation at 28 FPS with only 7.45 GB peak memory. Further incorporating model quantization, we deploy the quantized PebbleVideo on a Samsung mobile NPU, achieving a VBench score of 87.46.

1 INTRODUCTION

Since diffusion models were extended to video generation, they have substantially improved visual fidelity, temporal coherence, and controllability (Ho et al., 2022b;a; Bai et al., 2025; Team Wan et al., 2025). Modern video generation systems typically comprise two functional components: a diffusion backbone that parameterizes the denoising vector field, and condition encoders that map multimodal inputs into representations guiding generation. Among backbone architectures, Diffusion Transformers (DiTs) (Peebles & Xie, 2023) have become increasingly dominant due to their strong scaling behavior and full bidirectional attention, enabling global spatial-temporal interactions to capture long-range dependencies. However, scaling toward higher fidelity has rapidly increased the parameter footprints of both DiT backbones and condition encoders, while full bidirectional attention further incurs memory overhead. Consequently, video generation remains costly to deploy: even the lightweight Wan2.1 (Team Wan et al., 2025) combines a 1.4B-parameter DiT with a billion-parameter condition encoder and requires 19.68 GB of GPU memory to generate a 49-frame 320p video, exceeding the memory budgets of most edge devices.

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.

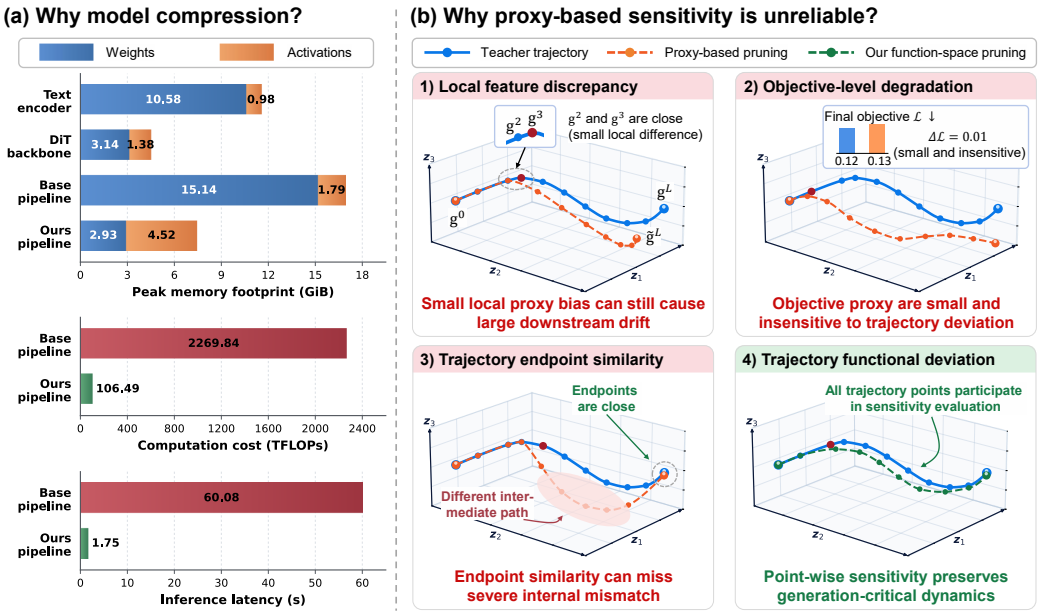


Figure 2: **Motivation.** (a) Model weights dominate the peak memory footprint of the base pipeline, motivating structural compression to reduce memory, computation, and inference latency. (b) The contributions of the L DiT blocks at a given timestep are represented as a trajectory. Red points indicate pruned blocks, while dashed curves denote the corresponding pruned trajectories. Local feature similarity can obscure directional trajectory deviations; objective-level degradation is too coarse to distinguish block sensitivity; and trajectory endpoint similarity overlook severe intermediate mismatch. In contrast, our trajectory-aligned functional deviation evaluates blocks over the full generation trajectory, better preserving the underlying generation dynamics. All results and observations are obtained on ReCamMaster (Bai et al., 2025), with real examples provided in Appendix H.

A straightforward remedy is to train a smaller video generator from scratch. However, foundation-scale pretraining is expensive, while the training data are often inaccessible due to proprietary or privacy constraints. Recent efforts therefore adapt pretrained models for efficient inference through distillation, redesigning attention or generation topology: causal attention enables autoregressive decoding with KV caching (Yin et al., 2025), linear attention replaces quadratic softmax with more efficient token mixing (Chen et al., 2025b), and streaming generation operates on temporally local chunks (Henschel et al., 2025). While effective in reducing computation and runtime memory, these approaches trade unrestricted global spatiotemporal interaction for efficiency, limiting the modeling capacity. More importantly, as shown in Fig. 2(a), for 320p video generation, a common on-device setting, model weights dominate the inference memory footprint over attention activations. Yet, such attention-centric optimization largely leaves the parameter footprint unchanged.

Orthogonal to redesigning token interactions, model compression has been widely adopted as a complementary approach to improving efficiency. Among compression techniques, structured pruning is particularly appealing, as it simultaneously reduces storage, computation, and memory footprint. Recent studies have extended structured pruning to video DiTs. Neodragon (Karnewar et al., 2026) ranks candidate blocks for pruning by the cosine discrepancy between their input and output features, while FastLightGen (Shao et al., 2026) evaluates importance through the objective-level (ELBO) degradation caused by block removal. Although effective, both rely on indirect proxies of structural importance. As illustrated in Fig. 2(b), the trajectory points g^2 and g^3 are close, indicating similar input and output features for Block 3. However, pruning Block 3 overlooks its directional effect on the vector field, leading to substantial downstream trajectory deviation. Objective-level degradation, in contrast, summarizes the effect of block removal with a single scalar, which remains similarly small across different blocks and thus provides limited discrimination for sensitivity estimation. These limitations reveal a common issue: ***indirect proxies capture only partial effects of pruning, rather than its perturbation to the diffusion vector field along the full denoising trajectory***. Under aggressive compression, such sensitivity errors can accumulate across pruning steps, severely distorting the original generation dynamics and hindering post-pruning recovery.

Motivated by this observation, we introduce **PebbleVideo**, an aggressively compressed video generator that preserves full bidirectional attention for on-device deployment. Our key insight is to evaluate compression sensitivity in a function-aligned space by measuring perturbations associated with generation-critical functions, rather than relying on indirect proxies. For the dominant DiT backbone, functional perturbation manifests in the diffusion vector field, enabling structural importance to be directly measured by its impact on denoising dynamics. For condition encoders, it manifests in conditioning representations, where cross-scale representation consistency determines whether to adopt compact-model substitution or In-place pruning. By aligning sensitivity evaluation with the functional objectives of post-compression distillation, our strategy reduces the initial functional gap, enabling more aggressive pruning and facilitating knowledge transfer.

To achieve extreme end-to-end compression through functional-space sensitivity, PebbleVideo introduces three key techniques. First, we propose *trajectory-aligned functional deviation* (TAFD) to assess DiT block sensitivity in function space. Specifically, TAFD maps each block's denoising contribution to a depth-wise trajectory induced by block skipping and aligns it with the intact teacher trajectory to quantify the residual discrepancy. This alignment removes the skip-induced depth shift, thereby isolating the resulting perturbation to the diffusion vector field. Second, to stabilize aggressive pruning, we couple TAFD with progressive pruning and recovery. We prune one TAFD-selected block at a time through staged head/channel pruning, second-order compensation, and vector-field recovery, followed by re-scoring the remaining blocks. This closed loop mitigates the accumulation of functional gaps across pruning stages. Third, for condition encoders, we use linear centered kernel alignment (CKA) (Kornblith et al., 2019) to assess teacher–candidate representation consistency across scales. CKA favors compact-model substitution for text encoder and in-place structural pruning for image encoder, and output distillation then aligns DiT-facing conditioning functions.

PebbleVideo reduces footprint from 8.13B to 1.57B parameters. Combined with four-step DMD-based distillation, PebbleVideo generates a 512×320×49 video in 1.75 s (28 FPS) on NVIDIA RTX 5090, using 7.45 GiB peak memory. After quantization, our Snapdragon 8 Elite Gen5 deployment requires only 1.8 GiB peak memory while retaining a VBench score of 87.46.

2 Related Work

We review studies on on-device video generation, while extended discussions of attention redesign, DiT pruning, encoder compression, step distillation, and quantization are provided in Appendix O.

Efficient Token Interaction. Some studies redesign attention mechanisms or generation topology to reduce the computational and memory costs of token interactions. CausVid (Yin et al., 2025) converts pretrained bidirectional diffusion models into causal autoregressive generators for efficient streaming inference. S^2DiT (Zhao et al., 2026) combines linear-convolutional hybrid attention with strided self-attention for mobile streaming video generation. MobileWan (Ghafoorian et al., 2026) combines recurrent generation with causal linear attention for memory-efficient chunk-wise inference on mobile devices. These approaches improve deployment efficiency through token-interaction redesign, but compromise the global modeling capacity of full bidirectional attention.

Diffusion Model Compression. Another line of work directly reduces model footprint for resource-constrained deployment. For DiT-based video generator, Neodragon (Karnewar et al., 2026) compresses the text encoder, VAE decoder, and MMDiT backbone via encoder distillation, asymmetric decoder distillation, and block pruning, enabling video generation on mobile NPUs. MVDiT (Wu et al., 2025a) combines a highly compressed VAE, sensitivity-aware multi-granularity pruning, and few-step distillation for mobile deployment. FastLightGen (Shao et al., 2026) jointly reduces transformer depth and sampling steps for lightweight video generation.

3 Preliminaries

3.1 Conditional Video Diffusion in Function Space

Denoising functions. Let $\mathbf{z}_0 \in \mathbb{R}^{C\times T\times H\times W}$ denote a clean video latent and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ denote Gaussian noise. A continuous noising path can be written as:

$$\mathbf{z}_t = \alpha_t \mathbf{z}_0 + \sigma_t \boldsymbol{\epsilon}, \qquad t \in [0, 1], \tag{1}$$

where (α_t, σ_t) specifies the noise schedule. A conditional video DiT learns the velocity field $\mathbf{v}_\theta(\mathbf{z}_t, t, \mathbf{c})$ using the flow-matching objective:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\mathbf{z}_0, \boldsymbol{\epsilon}, t}\left[\left\|\mathbf{v}_\theta(\mathbf{z}_t, t, \mathbf{c}) - \dot{\alpha}_t \mathbf{z}_0 - \dot{\sigma}_t \boldsymbol{\epsilon}\right\|_2^2\right]. \tag{2}$$

At inference, a numerical solver generates a video by integrating the probability-flow ODE:

$$\frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t} = \mathbf{v}_\theta(\mathbf{z}_t, t, \mathbf{c}), \qquad \mathbf{z}_t = \mathbf{z}_1 + \int_1^t \mathbf{v}_\theta(\mathbf{z}_s, s, \mathbf{c})\,\mathrm{d}s. \tag{3}$$

Thus, for fixed initial noise $\mathbf{z}_1$, condition $\mathbf{c}$, and solver, the vector field uniquely determines the denoising trajectory. Conversely, along a differentiable denoising trajectory, the vector field is given by $\mathbf{v}_\theta = \mathrm{d}\mathbf{z}_t/\mathrm{d}t$. Measuring a change in the generated latent trajectory is therefore a direct functional probe of the vector field that performs denoising.

Conditioning functions. The text and image encoders do not denoise latents; their essential functions are semantic transformations from their native input domains to the condition space consumed by the DiT. Given a text prompt $\mathbf{p}$ and an input image $\mathbf{I}$, we write these transformations as:

$$\mathbf{c}^{\mathrm{txt}} = f_{\mathrm{txt}}(\mathbf{p}) := A_{\mathrm{txt}}(E_{\mathrm{txt}}(\mathbf{p}))\,, \qquad \mathbf{c}^{\mathrm{img}} = f_{\mathrm{img}}(\mathbf{I}) := A_{\mathrm{img}}(E_{\mathrm{img}}(\mathbf{I}))\,, \tag{4}$$

where E_{txt} and E_{img} are the encoder backbones, and A_{txt} and A_{img} are the native DiT condition projections (`txt_emb` and `img_emb`, respectively). Depending on the generation task, the DiT is conditioned on text, image, or both, collectively denoted by $\mathbf{c} = \{\mathbf{c}^{\mathrm{txt}}, \mathbf{c}^{\mathrm{img}}\}$. Accordingly, the function spaces relevant to compression are:

$$\mathcal{F}_{\mathrm{DiT}} : (\mathbf{z}_t, t, \mathbf{c}) \mapsto \mathbf{v}_t, \quad \mathcal{F}_{\mathrm{txt}} : \mathbf{p} \mapsto \mathbf{c}^{\mathrm{txt}}, \quad \mathcal{F}_{\mathrm{img}} : \mathbf{I} \mapsto \mathbf{c}^{\mathrm{img}}. \tag{5}$$

Projecting DiT blocks into the denoising trajectory. At diffusion time t, a DiT transforms the patched latent $\mathbf{h}_t^0$ through L residual blocks and predicts the velocity from the final state:

$$\mathbf{h}_t^0 = P(\mathbf{z}_t, t), \qquad \mathbf{h}_t^\ell = B_\ell(\mathbf{h}_t^{\ell-1}; t, \mathbf{c}), \qquad \mathbf{v}_\theta(\mathbf{z}_t, t, \mathbf{c}) = D_t(\mathbf{h}_t^L). \tag{6}$$

To expose the functional contribution of each block in a shared function-aligned space, we apply the same output head D_t and sampler update Ψ_t to every intermediate state:

$$\mathbf{g}_t^\ell := \Pi_t(\mathbf{h}_t^\ell) = \Psi_t\big(\mathbf{z}_t, D_t(\mathbf{h}_t^\ell)\big)\,, \qquad \boldsymbol{\Gamma}_t = (\mathbf{g}_t^0, \ldots, \mathbf{g}_t^L). \tag{7}$$

$\boldsymbol{\Gamma}_t$ is a *projected denoising trajectory* through network depth, with $\mathbf{g}_t^\ell$ as its trajectory points. The teacher trajectory in Fig. 2(b) provides an intuitive illustration. For an Euler solver, $\mathbf{g}_t^\ell = \mathbf{z}_t + \Delta t\, D_t(\mathbf{h}_t^\ell)$, so its deviation reflects the perturbation after projecting intermediate states through the native velocity readout. Note that network depth is distinct from diffusion time; it indexes successive refinements of the denoising function at a fixed timestep t. This representation enables the block-sensitivity measure introduced in Sec. 4.1.

3.2 Why Indirect Proxies Can Fail

Existing structured-pruning methods commonly evaluate a block using either a local feature discrepancy or the change of an aggregate generation objective. Both are useful heuristics, but neither defines a distance in the function spaces of Eq. 5.

Local feature discrepancy. Cosine-based criteria, as used by Neodragon (Karnewar et al., 2026), assign block B_ℓ the score:

$$s_\ell^{\mathrm{feat}} = \mathbb{E}\left[1 - \frac{\langle \mathbf{h}^{\ell-1}, \mathbf{h}^\ell \rangle}{\|\mathbf{h}^{\ell-1}\|_2 \|\mathbf{h}^\ell\|_2}\right]. \tag{8}$$

Let $R_{>\ell} = D_t \circ B_L \circ \cdots \circ B_{\ell+1}$ denote the downstream mapping. Skipping B_ℓ induces the local perturbation $\delta\mathbf{h}\ell = \mathbf{h}^\ell - \mathbf{h}^{\ell-1}$, whose effect on the predicted velocity denotes as:

$$\delta\mathbf{v}_\ell = R_{>\ell}(\mathbf{h}^\ell) - R_{>\ell}(\mathbf{h}^{\ell-1}) = J_{R_{>\ell}}\,\delta\mathbf{h}_\ell + \mathcal{O}(\|\delta\mathbf{h}_\ell\|_2^2), \tag{9}$$

where the second equality follows from a first-order Taylor expansion. Eq. 8 captures only the local angular discrepancy, ignoring both perturbation magnitude and the anisotropic downstream Jacobian. Consequently, even a small local perturbation can align with directions strongly amplified

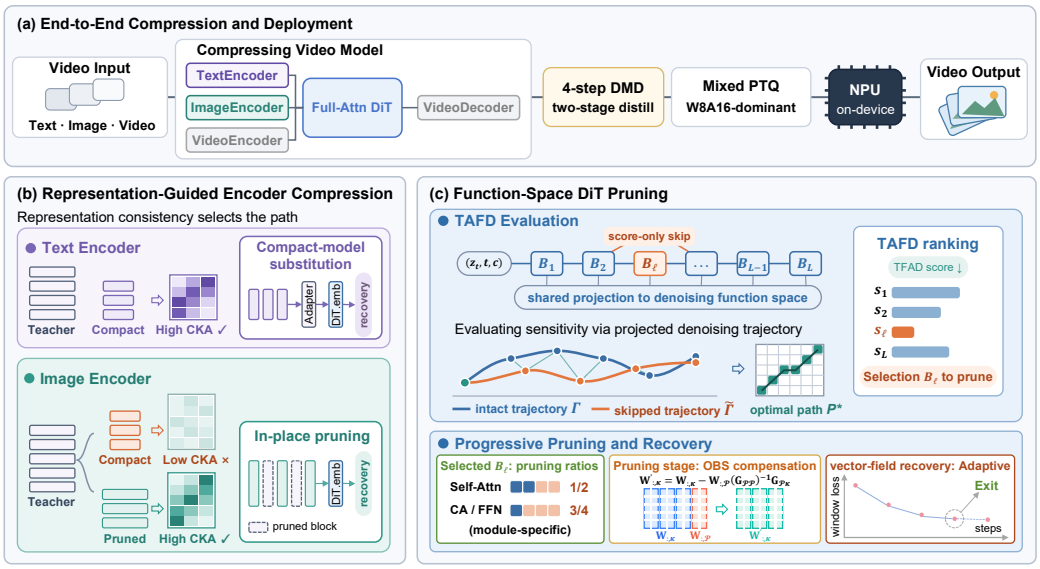


Figure 3: **Overview of PebbleVideo.** (a) Starting from a pretrained video generator, we compress its conditioning encoders and full-attention DiT, reduce denoising to four steps through two-stage DMD, and apply W8A16-dominant mixed-precision quantization for on-device NPU execution. (b) CKA favors compact-model substitution for the text encoder and in-place pruning for the image encoder, followed by output alignment at the DiT-facing outputs. (c) For each candidate DiT block, TAFD aligns the intact and skipped trajectories in a shared denoising-function space and ranks blocks by their optimal-path TAFD scores. The selected block is then progressively pruned using module-specific ratios, OBS compensation, and adaptive vector-field recovery.

by $J_{R_{>\ell}}$, leading to substantial downstream trajectory deviation, as shown in Fig. 2(b). Therefore, local feature similarity does not reliably imply functional redundancy.

Aggregate objective response. FastLightGen (Shao et al., 2026) evaluates the model with block ℓ removed through a Tweedie/ELBO-style error:

$$\mathcal{E}_\ell = \mathbb{E}_{t,\mathbf{z}_t,\mathbf{z}_0} \left\| \mathbf{z}_t - t\mathbf{v}_\theta^{-\ell}(\mathbf{z}_t, t, \mathbf{c}) - \mathbf{z}_0 \right\|_2^2, \tag{10}$$

where $\mathbf{v}_\theta^{-\ell}$ is the pruned vector field. To see the information lost by this scalar, let $\mathbf{e} = \mathbf{z}_t - t\mathbf{v}_\theta - \mathbf{z}_0$ and $\delta\mathbf{v}_\ell = \mathbf{v}_\theta^{-\ell} - \mathbf{v}_\theta$. As derived in Appendix A, the objective change denotes as:

$$\Delta\mathcal{E}_\ell = \mathbb{E}\left[-2t\,\mathbf{e}^\top \delta\mathbf{v}_\ell + t^2 \|\delta\mathbf{v}_\ell\|_2^2\right]. \tag{11}$$

It is an expectation of a residual-dependent projection and a weighted norm, not a metric between vector fields. Errors at different samples, timesteps, or spatial-temporal regions can be averaged away, and perturbations with different trajectory geometry can produce similar objective changes, as shown in Fig. 2(b). Thus, the objective response cannot accurately reflect functional redundancy.

Our experiments also show that endpoint-only evaluation in function space is insufficient to characterize pruning sensitivity, as shown in Fig. 2(b). These observations motivate evaluating each component in the space of function: projected denoising trajectories for the DiT, and the geometry of final conditioning representations for the encoders.

4 Method

The overall compression and deployment pipeline of PebbleVideo is illustrated in Fig. 3(a). Specifically, we first introduce the sensitivity estimator and recovery loop for progressive DiT pruning (Sec. 4.1). We then present representation-guided selection and interface alignment for compressing encoders (Sec. 4.2). Few-step distillation and quantized deployment are detailed in Appendix B.

4.1 FUNCTION-SPACE DIT PRUNING

We evaluate candidate blocks $\mathcal{C}$ on cached denoising states $\mathcal{S} = \{(\mathbf{z}_{t_k}, t_k, \mathbf{c}_k)\}_{k=1}^{K}$, where K denotes the total number of cached states across samples and diffusion timesteps. A standard forward pass records the projected trajectory in Eq. 7. For a candidate block B_ℓ, let $a_1 < \cdots < a_m$ denote its active downstream blocks. We define its pruning sensitivity by the perturbation induced on the projected denoising trajectory when B_ℓ is removed, as shown in Fig. 3 (c). Specifically, we first record the intact suffix trajectory $\mathbf{\Gamma}_{k,\ell}$, then bypass B_ℓ by forwarding its input $\mathbf{h}^{\ell-1}$ directly to the next active block, yielding the corresponding skip trajectory $\widetilde{\mathbf{\Gamma}}_{k,\ell}$:

$$\mathbf{\Gamma}_{k,\ell} = (\mathbf{g}^{\ell-1}, \mathbf{g}^{\ell}, \mathbf{g}^{a_1}, \ldots, \mathbf{g}^{a_m}), \qquad \widetilde{\mathbf{\Gamma}}_{k,\ell} = (\mathbf{g}^{\ell-1}, \widetilde{\mathbf{g}}^{a_1}, \ldots, \widetilde{\mathbf{g}}^{a_m}). \tag{12}$$

where $\widetilde{\mathbf{g}}$ denotes the projected state on the skip trajectory after bypassing B_ℓ. The two trajectories share the same starting anchor $\mathbf{g}^{\ell-1}$, while the skip trajectory contains one fewer refinement stage. Since all states are obtained through the D_t and Ψ_t, they lie in a shared denoising-function space, where their deviations capture how pruning-induced perturbations propagate to the final prediction.

However, removing B_ℓ changes not only the downstream states but also their depth-wise correspondence, shifting each subsequent state by one position. A same-index distance would therefore conflate functional deviation with this skip-induced depth shift. To disentangle the two effects, we propose *trajectory-aligned functional deviation* (TAFD), which first aligns the intact and skip trajectories and then measures their residual discrepancy. Specifically, for local cost $C_{ij} = \|\widetilde{\mathbf{g}}_i - \mathbf{g}_j\|_2$, we use a DTW-style monotonic alignment (Sakoe & Chiba, 1978):

$$D_{ij} = C_{ij} + \min\{D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}\}. \tag{13}$$

Here, D_{ij} is the minimum cumulative cost for aligning the two trajectory prefixes ending at $\widetilde{\mathbf{g}}_i$ and $\mathbf{g}_j$. Let $\mathcal{P}^{\star}_{k,\ell}$ denote the resulting optimal alignment path; we define the pruning sensitivity of B_ℓ as:

$$s_\ell^{\mathrm{TAFD}} = \frac{1}{K}\sum_{k=1}^{K} \frac{D^{k,\ell}_{m,m+1}}{|\mathcal{P}^{\star}_{k,\ell}|}, \qquad \ell^{\star} = \arg\min_{\ell \in \mathcal{C}} s_\ell^{\mathrm{TAFD}}, \tag{14}$$

where m denotes the number of downstream stages in the skip trajectory. The alignment absorbs the one-block depth shift, while path-length normalization mitigates bias from varying suffix lengths. Thus, s_ℓ^{TAFD} measures the functional perturbation induced by removing B_ℓ, with a lower score indicating lower pruning sensitivity. After each pruning-and-recovery phase, we recompute the scores on the current model to capture interactions among removed structures. The pseudocode for block sensitivity evaluation is provided in Alg. 1 of Appendix G.

To stabilize the aggressive pruning process, we introduce three designs for the recovery phase:

Progressive pruning. We perform progressive pruning at both the block and intra-block levels. Existing methods typically estimate block importance once and prune multiple blocks accordingly, overlooking sensitivity shifts induced by preceding structural changes. To address this issue, at outer iteration p, we evaluate all active candidates $\mathcal{C}_p$ on the current student, select $\ell_p^{\star}$ by Eq. 14, compress only $B_{\ell_p^{\star}}$, and recover the model before the next evaluation. Within the selected block, we determine module-specific target pruning ratios based on sensitivity analysis and progressively reach them through cumulative pruning stages. For example, an FFN with a target pruning ratio of $3/4$ is successively pruned to $(1/2, 5/8, 3/4)$, with recovery after each stage. The module sensitivities, target pruning ratios, and detailed pruning schedules are provided in Appendix F.

OBS initialization and compensation. At each pruning stage, we use Optimal Brain Surgeon (OBS) (Hassibi & Stork, 1992) to identify low-saliency structures and initialize the pruned module. For an output weight matrix $\mathbf{W}$, we collect cached inputs $\mathbf{X}$ to the corresponding attention output projection or second FFN linear layer. We compute the input curvature as $\mathbf{H} = \mathbf{X}^{\top}\mathbf{X}$ and apply damping: $\overline{\mathbf{H}} = \mathbf{H} + \lambda_{\mathrm{damp}}\,\mathrm{mean}(\mathrm{diag}(\mathbf{H}))\mathbf{I}$. Let $\mathbf{G} = \overline{\mathbf{H}}^{-1}$. The group saliency $\omega_{\mathcal{G}}$ and OBS compensation for the retained weights are given by:

$$\omega_{\mathcal{G}} = \sum_{j \in \mathcal{G}} \frac{\|\mathbf{W}_{:,j}\|_2^2}{\mathbf{G}_{jj}}, \qquad \mathbf{W}'_{:,\mathcal{K}} = \mathbf{W}_{:,\mathcal{K}} - \mathbf{W}_{:,\mathcal{P}}(\mathbf{G}_{\mathcal{P}\mathcal{P}})^{-1}\mathbf{G}_{\mathcal{P}\mathcal{K}}. \tag{15}$$

Here, $\mathcal{G}$ denotes an attention head or an individual FFN channel, while $\mathcal{P}$ and $\mathcal{K}$ denote the pruned and retained columns, respectively. We remove the lowest-saliency groups and initialize the narrower projections with the compensated retained weights. This compensation reduces pruning-induced output perturbations and facilitates subsequent gradient-based recovery.

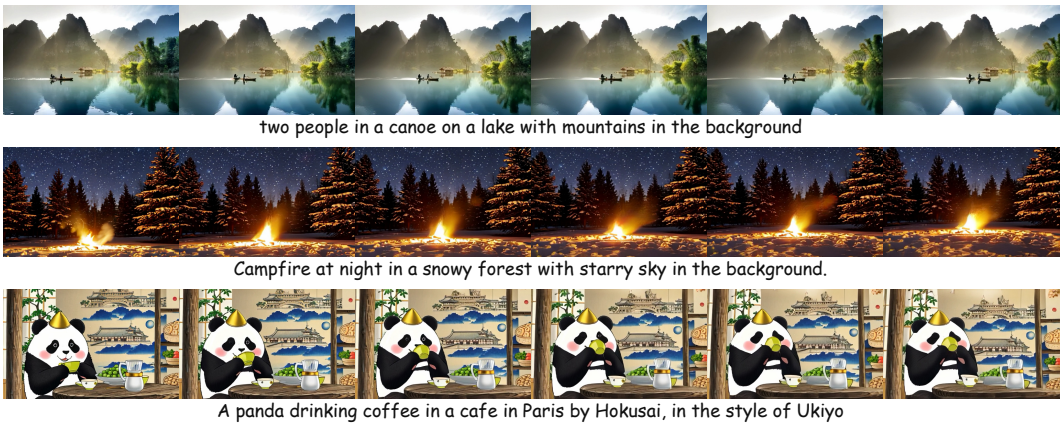


Figure 4: **Qualitative results of PebbleVideo.** Videos generated by PebbleVideo.

Adaptive recovery and plateau exit. After each pruning stage, we recover the student from a frozen teacher by matching their diffusion vector fields:

$$\mathcal{L}_{\text{DiT}-\text{rec}} = \mathbb{E}_{\mathbf{z}_t,t,\mathbf{c}}\left[\|\mathbf{v}_{\theta_s}(\mathbf{z}_t,t,\mathbf{c}) - \mathbf{v}_{\theta_T}(\mathbf{z}_t,t,\mathbf{c})\|_2^2\right], \tag{16}$$

We perform R recovery updates after each pruning stage. In the final stage, recovery continues until either the loss converges or the training budget is exhausted. Convergence is determined when the relative decrease in the window-averaged loss $\overline{\mathcal{L}}_u$ falls below τ: $(\overline{\mathcal{L}}_{u-1} - \overline{\mathcal{L}}_u)/\max(|\overline{\mathcal{L}}_{u-1}|) < \tau$, where $\tau = 5\%$ in all experiments. Any unused recovery budget is reallocated to subsequent blocks and the final joint recovery. The complete pipeline is provided in Alg. 2 of Appendix G.

4.2 Representation-Guided Encoder Compression

Prior work often compresses text encoders by substituting pretrained compact models (Karnewar et al., 2026; Wang et al., 2025). However, applying the same strategy to image encoders causes severe degradation. This discrepancy motivates us to select encoder compression strategies based on how well they preserve the original input-to-condition mapping in representation space.

For n calibration inputs, let $\mathbf{R}_T \in \mathbb{R}^{n\times d_T}$ and $\mathbf{R}_S \in \mathbb{R}^{n\times d_S}$ denote the representations induced by the teacher and compression candidate, respectively. For compact notation, let $q \in \{T, S\}$, We measure their consistency using linear centered kernel alignment (CKA) (Kornblith et al., 2019):

$$\mathbf{H}_n = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top, \quad \mathbf{K}_q = \mathbf{H}_n\mathbf{R}_q\mathbf{R}_q^\top\mathbf{H}_n, \quad \text{CKA}(\mathbf{R}_T,\mathbf{R}_S) = \frac{\langle \mathbf{K}_T, \mathbf{K}_S\rangle_F}{\|\mathbf{K}_T\|_F\|\mathbf{K}_S\|_F}. \tag{17}$$

CKA compares the relational geometry induced by the same inputs and is widely used for cross-network representation comparison (Kornblith et al., 2019). We compare compression candidates using block-wise CKA heatmaps, as shown in Fig. 3(b). The compact text encoder exhibits high CKA with its teacher, indicating that substitution preserves the semantic transformation. In contrast, substituting a compact image encoder changes both channel width and patch tokenization, leading to low CKA. In-place pruning retains the original tokenization and representation dimensionality, yielding substantially higher CKA. We therefore adopt compact substitution for text encoder and in-place pruning for image encoder, selecting the compression operation by representation preservation rather than modality-specific heuristics. The actual CKA heatmaps are provided in Appendix K.

Given the selected compression strategy, we recover the conditioning function at the DiT interface defined in Eq. 18. For modality $m \in \{\text{txt}, \text{img}\}$, let $\mathbf{x}^{\text{txt}} = \mathbf{p}$, $\mathbf{x}^{\text{img}} = \mathbf{I}$, and $q \in \{T, S\}$, $\mathbf{c}_q^m = f_q^m(\mathbf{x}^m)$ denote the teacher and compressed condition outputs. We optimize:

$$\mathcal{L}^m_{\text{enc}-\text{rec}} = \mathbb{E}_{\mathbf{x}^m}\left[\lambda_{\text{mse}}\|\mathbf{c}_S^m - \mathbf{c}_T^m\|_2^2 + \lambda_{\cos}\left(1 - \frac{\langle \mathbf{c}_S^m, \mathbf{c}_T^m\rangle}{\|\mathbf{c}_S^m\|_2\|\mathbf{c}_T^m\|_2}\right)\right]. \tag{18}$$

Following Neodragon (Karnewar et al., 2026), we set $\lambda_{\text{mse}} = 1.0$ and $\lambda_{\cos} = 0.1$ for all experiments. The MSE and cosine terms respectively preserve the element-wise response and global orientation of the DiT-facing condition. Since $\mathbf{c}_q^m$ is the output of the complete conditioning function f_q^m, this objective directly aligns $\mathcal{F}_{\text{txt}}$ or $\mathcal{F}_{\text{img}}$ rather than an intermediate feature proxy.

Table 1: Results of on-device models. VBench Score ↑ is the average of the six reported VBench dimensions ↑. The encoder size includes the text encoder, image encoder, and VAE when available.

Method	Params (B) ↓		VBench	Background	Imaging	Temporal	Aesthetic	Motion	Subject
	Backbone	Encoder	Score	Consistency	Quality	Flickering	Quality	Smoothness	Consistency
AMD-Hummingbird	0.95	0.44	86.70	96.31	71.17	94.83	68.43	93.93	95.53
SnapGen-V	0.60	–	–	–	–	99.37	62.19	99.29	–
On-device Sora	1.21	4.98	81.83	97.00	53.00	99.00	47.00	99.00	96.00
MobileVD	–	0.73	84.46	95.95	67.46	96.00	53.19	97.96	96.18
NeoDragen	2.97	1.97	86.02	96.62	63.27	99.01	61.05	99.43	96.72
MVDiT-Mobile	0.92	–	86.22	96.65	63.41	98.11	64.16	99.23	95.73
S^2DiT	1.80	–	–	–	70.57	98.20	65.64	–	–
MobileWan	4.96	5.85	87.09	95.61	71.79	97.12	62.58	98.59	96.82
Ours-Mobile	0.92	0.65	87.46	97.76	73.08	96.93	60.26	98.65	98.08

Table 2: Results of server-side models. Following NeoDragen, we unify server-side evaluation at 512×320×49 whenever possible. Unlike Ours-Mobile, Ours-Server does not employ model quantization. Peak memory (GiB) ↓, Inference latency (s) ↓, and FPS ↑ are all measured end-to-end.

Method	Params (B) ↓		Peak	Inference	FPS	VBench	Background	Imaging	Temporal	Aesthetic	Motion	Subject
	Backbone	Encoder	Memory	Latency		Score	Consistency	Quality	Flickering	Quality	Smoothness	Consistency
ReCamMaster	1.69	6.44	16.93	60.08	0.82	88.23	97.71	73.56	96.97	65.01	98.84	97.31
Wan2.1-1.3B	1.42	5.81	19.68	28.85	1.70	87.98	98.45	71.04	98.88	62.12	99.21	98.20
Wan2.1-Camera	1.62	6.44	16.79	24.27	2.02	88.44	97.79	74.11	97.12	65.49	98.81	97.33
PyramidalFlow	1.97	5.22	17.09	12.93	3.79	87.70	98.07	68.39	99.44	62.88	98.58	98.84
CogVideoX-2B	1.69	4.98	21.57	24.89	1.97	82.44	95.28	58.33	97.30	50.91	98.78	94.05
Open-Sora-1.2	1.21	5.16	22.89	20.58	2.48	82.08	97.59	51.60	98.91	48.31	99.19	96.88
Open-Sora-1.3	1.21	5.02	OOM	–	–	85.35	97.67	62.79	97.99	57.16	98.71	97.75
LTX-Video	1.92	5.32	15.67	0.48	101.34	85.97	98.02	65.42	99.29	54.28	99.59	99.23
CausVid	1.42	5.81	16.64	2.04	23.99	87.12	97.36	70.02	97.96	59.60	99.27	98.53
SANA-Video	2.06	2.74	17.85	81.60	0.60	86.12	95.62	69.99	96.06	62.13	98.24	94.69
Rectified SpaAttn	1.42	5.81	17.27	26.77	1.83	85.51	96.89	67.47	98.71	53.60	99.23	97.17
MVDiT-Server	2.00	5.68	–	18.90	6.40	87.05	97.32	65.85	98.74	64.72	99.21	96.48
FastLightGen	9.80	7.00	–	28.30	2.86	–	–	71.70	97.50	65.60	98.30	95.60
Ours-Server	0.92	0.65	7.45	1.75	28.00	87.74	97.44	73.16	96.95	63.16	98.56	97.17

5 EXPERIMENTS

Baselines. We compare with state-of-the-art server-side and on-device video generation models, and further evaluate individual components against existing DiT and encoder compression methods. Detailed reproduction protocols and the provenance of all results are provided in Appendix C.5.

Metrics. We evaluate video quality using VBench (Huang et al., 2024)(Appendix C.2). For efficiency, we report backbone and encoder parameter counts. For server-side models, we additionally report peak memory, inference latency, and generation throughput in frames per second (FPS). Runtime metrics are omitted for on-device models due to heterogeneous hardware platforms.

Implementation. We build PebbleVideo on ReCamMaster (Bai et al., 2025) and sequentially apply encoder compression, DiT pruning, four-step DMD, and hardware-aware deployment. Detailed implementation and training configurations are provided in Appendix C.3 and C.4.

5.1 QUALITY EVALUATION

We deploy the quantized PebbleVideo on a mobile device with a Qualcomm Snapdragon 8 Elite Gen5. As shown in Tab. 1, structural compression reduces the ReCamMaster backbone and encoders from 1.69B and 6.44B parameters to 0.92B and 0.65B, respectively. With full bidirectional attention preserved, PebbleVideo achieves a VBench Score of 87.46. On an NVIDIA RTX 5090, PebbleVideo generates a 49-frame video in 1.75 s with only 7.45 GiB peak memory, while achieving a VBench score of 87.74, as shown in Tab. 2. These results demonstrate that PebbleVideo substantially improves end-to-end efficiency while preserving competitive generation quality. Since these methods operate under different generation settings and conditioning modalities, we report them as system-level references rather than strictly controlled head-to-head comparisons. Controlled comparisons of the DiT pruning and encoder compression strategies are provided in Appendices I and L, respectively. Additional analysis of the overall compression pipeline is provided in Appendix E.

Table 3: Hardware efficiency breakdown of PebbleVideo across platforms; peak memory is measured on Snapdragon 8 Elite Gen5, and end-to-end latency covers the complete generation pipeline.

Platform / Measurement	Text Enc.	Image Enc.	VAE Enc.	VAE Dec.	DiT Backbone	Pipeline
PebbleVideo / Params (M)	301.8	336.9	1.5	9.8	920.7	1570.7
Snapdragon 8 Elite Gen5 / Memory (G)	1.3	1.0	0.8	0.8	1.8	1.8
Snapdragon 8 Elite Gen5 / Latency (ms)	32.9	29.9	379.4	145.0	145442.7	146800.0
Apple Silicon Mac / Latency (ms)	12.6	9.5	58.8	155.5	26011.0	26591.0
NVIDIA RTX 5090 / Latency (ms)	10.6	4.3	29.1	43.3	1518.7	1750.0

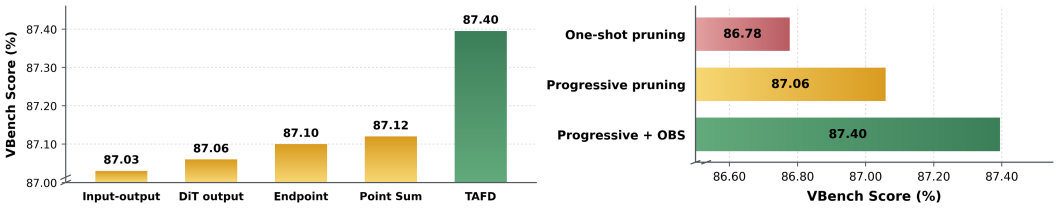


Figure 5: **Ablations on DiT sensitivity metrics and pruning strategies.** Left: Input-output denotes the criterion used by Neodragon, while DiT-output approximates the recovery objective of FastLightGen. Endpoint and Point Sum measure function-space deviations at the trajectory endpoint and across all trajectory points. Right: Ablation of the pruning strategy components, where one-shot pruning follows NeoDragen. All variants prune 8 blocks from ReCamMaster.

5.2 Efficiency Evaluation

We analyze the parameter, memory, and latency breakdown of PebbleVideo across different platforms. As shown in Tab. 3, compression reduces the full pipeline to 1.57B parameters. Combined with mixed-precision quantization, this reduces peak memory to 1.8 GiB on Snapdragon 8 Elite Gen5, enabling mobile video generation with full bidirectional attention.

5.3 Ablation Study

Ablation on DiT Pruning. As shown in Fig. 5, our TAFD-based sensitivity metric outperforms existing proxy criteria and simple trajectory-deviation baselines. Moreover, our proposed pruning strategy further stabilizes training.

Ablation on Generation Resolution. As shown in Tab. 4, PebbleVideo maintains stable generation quality when scaling from 320p to 480p, while achieving a $35.1\times$ inference speedup at 480p, demonstrating strong robustness across resolutions.

Human Preference Evaluation. We conduct an Arena-style anonymous pairwise study with 30 participants to evaluate human preference for PebbleVideo. The complete protocol and results are provided in Appendix N. Fig. 4 presents qualitative results of our method.

Table 4: Results on the 480p, 81-frame video generation task, with ReCamMaster as the baseline.

Method	Params (B) ↓		FPS	VBench	Background	Imaging	Temporal	Aesthetic	Motion	Subject
	Backbone	Encoder		Score	Consistency	Quality	Flickering	Quality	Smoothness	Consistency
Baseline	1.69	6.44	0.16	88.33	96.86	73.48	96.44	67.92	98.78	96.52
Ours	0.92	0.65	5.45	87.97	96.69	72.09	97.36	66.50	98.66	96.53

6 Conclusion

In this paper, we introduce PebbleVideo, an end-to-end compression framework for on-device video generation that preserves full bidirectional attention. By measuring DiT sensitivity through trajectory-aligned perturbations of the diffusion vector field, PebbleVideo identifies generation-critical structures more faithfully than indirect proxies. Cross-scale representation consistency further guides compression of heterogeneous condition encoders, followed by recovery of their conditioning functions. Combined with progressive pruning, few-step distillation, and hardware-aware quantization, PebbleVideo achieves a favorable trade-off between generation quality and deployment efficiency. Our results demonstrate that preserving generation-critical functions provides a practical and complementary path toward efficient on-device video generation.

AI USE STATEMENT

In this work, we used generative AI tools for no tasks with required disclosure. We have not used generative AI tools for any other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for no tasks with recommended disclosure. We have reviewed all AI-assisted work. No AI-assisted work was involved in this work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Jianhong Bai, Menghan Xia, Xiao Fu, Xintao Wang, Lianrui Mu, Jinwen Cao, Zuozhu Liu, Haoji Hu, Xiang Bai, Pengfei Wan, and Di Zhang. ReCamMaster: Camera-controlled generative rendering from a single video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14834–14844, 2025.

Haitam Ben Yahia, Denis Korzhenkov, Ioannis Lelekas, Amir Ghodrati, and Amirhossein Habibian. Mobile video diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19450–19460, 2025.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Ollin Boer Bohan. Taehv: Tiny autoencoder for hunyuan video. https://github.com/madebyollin/taehv, 2025.

Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon Kim. COYO-700M: Image-text pair dataset. GitHub repository, 2022. URL https://github.com/kakaobrain/coyo-dataset.

Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Song Han, and Enze Xie. SANA-Sprint: One-step diffusion with continuous-time consistency distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16185–16195, 2025a.

Junsong Chen, Yuyang Zhao, Jincheng Yu, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, Haozhe Liu, Hongwei Yi, Hao Zhang, Muyang Li, Yukang Chen, Han Cai, Sanja Fidler, Ping Luo, Song Han, and Enze Xie. Sana-video: Efficient video generation with block linear diffusion transformer, 2025b. URL https://arxiv.org/abs/2509.24695.

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

Yu Cheng and Fajie Yuan. LeanVAE: An ultra-efficient reconstruction VAE for video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15692–15702, 2025.

Weichen Fan, Chenyang Si, Junhao Song, Zhenyu Yang, Yinan He, Long Zhuo, Ziqi Huang, Ziyue Dong, Jingwen He, Dongwei Pan, Yi Wang, Yuming Jiang, Yaohui Wang, Peng Gao, Xinyuan Chen, Hengjie Li, Dahua Lin, Yu Qiao, and Ziwei Liu. Vchitect-2.0: Parallel transformer for scaling up video diffusion models. *arXiv preprint arXiv:2501.08453*, 2025.

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.

Gongfan Fang, Kunjun Li, Xinyin Ma, and Xinchao Wang. TinyFusion: Diffusion transformers learned shallow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18144–18154, 2025.

Jiaqi Feng, Justin Cui, Yuanhao Ban, and Cho-Jui Hsieh. One-Forcing: Towards stable one-step autoregressive video generation. *arXiv preprint arXiv:2605.23458*, 2026.

Weilun Feng, Chuanguang Yang, Haotong Qin, Xiangqi Li, Yu Wang, Zhulin An, Libo Huang, Boyu Diao, Zixiang Zhao, Yongjun Xu, and Michele Magno. Q-VDiT: Towards accurate quantization and distillation of video-generation diffusion transformers. In *International Conference on Machine Learning*, pp. 16956–16976, 2025.

Mohsen Ghafoorian, Denis Korzhenkov, Adil Karjauv, Ioannis Lelekas, Noor Fathima, Spyridon Stasis, Hanno Ackermann, Boris van Breugel, Markus Nagel, Fatih Porikli, Animesh Karnewar, and Amirhossein Habibian. MobileWan: Closing the quality gap for mobile video diffusion. *arXiv preprint arXiv:2607.06173*, 2026.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. LTX-Video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. Hanson, J. Cowan, and C. Giles (eds.), *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992.

Roberto Henschel, Levon Khachatryan, Hayk Poghosyan, Daniil Hayrapetyan, Vahram Tadevosyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. StreamingT2V: Consistent, dynamic, and extendable long video generation from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2568–2577, 2025.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David Fleet. Video diffusion models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 8633–8646. Curran Associates, Inc., 2022b. doi: 10.52202/068431-0628.

Yushi Huang, Xingtong Ge, Ruihao Gong, Chengtao Lv, and Jun Zhang. LinVideo: A post-training framework towards O(n) attention in efficient video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23398–23408, 2026.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *International Conference on Learning Representations*, 2025.

Animesh Karnewar, Denis Korzhenkov, Ioannis Lelekas, Noor Fathima, Adil Karjauv, Mohsen Ghafoorian, and Amirhossein Habibian. Neodragon: Mobile video generation using diffusion transformer. In *International Conference on Learning Representations*, 2026.

Bosung Kim, Kyuhwan Lee, Isu Jeong, Jungmin Cheon, Yeojin Lee, and Seulki Lee. On-device Sora: Enabling training-free diffusion-based text-to-video generation for mobile devices. *arXiv preprint arXiv:2502.04363*, 2025.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. HunyuanVideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529, 2019.

Zhiteng Li, Hanxuan Li, Junyi Wu, Kai Liu, Haotong Qin, Linghe Kong, Guihai Chen, Yulun Zhang, and Xiaokang Yang. DVD-Quant: Data-free video diffusion transformers quantization. In *International Conference on Learning Representations*, 2026.

Zongjian Li, Bin Lin, Yang Ye, Liuhan Chen, Xinhua Cheng, Shenghai Yuan, and Li Yuan. WF-VAE: Enhancing video VAE by wavelet-driven energy flow for latent video diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17778–17788, 2025.

Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion adversarial post-training for one-step video generation. In *International Conference on Machine Learning*, pp. 37959–37974, 2025.

Xuewen Liu, Zhikai Li, Jing Zhang, Mengjuan Chen, and Qingyi Gu. Rectified SpaAttn: Revisiting attention sparsity for efficient video generation. *arXiv preprint arXiv:2511.19835*, 2025.

Weili Nie, Julius Berner, Nanye Ma, Chao Liu, Saining Xie, and Arash Vahdat. Transition matching distillation for fast video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4645–4655, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Haonan Qiu, Shikun Liu, Zijian Zhou, Zhaochong An, Weiming Ren, Zhiheng Liu, Jonas Schult, Sen He, Shoufa Chen, Yuren Cong, Tao Xiang, Ziwei Liu, and Juan-Manuel Perez-Rua. HiStream: Efficient high-resolution video generation via redundancy-eliminated streaming. *arXiv preprint arXiv:2512.21338*, 2025.

H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022. URL https://arxiv.org/abs/2202.00512.

Shitong Shao, Yufei Gu, and Zeke Xie. Fastlightgen: Fast and light video generation with fewer steps and parameters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2104–2114, June 2026.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Lifu Wang, Daqing Liu, Xinchen Liu, and Xiaodong He. Scaling down text encoders of text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18424–18433, 2025.

Yushu Wu, Yanyu Li, Anil Kag, Ivan Skorokhodov, Willi Menapace, Ke Ma, Arpit Sahni, Ju Hu, Aliaksandr Siarohin, Dhritiman Sagar, Yanzhi Wang, and Sergey Tulyakov. Taming diffusion transformer for efficient mobile video generation in seconds. *arXiv preprint arXiv:2507.13343*, 2025a.

Yushu Wu, Zhixing Zhang, Yanyu Li, Yanwu Xu, Anil Kag, Yang Sui, Huseyin Coskun, Ke Ma, Aleksei Lebedev, Ju Hu, Dimitris N. Metaxas, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. SnapGen-V: Generating a five-second video within five seconds on a mobile device. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2479–2490, 2025b.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, 2025.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, 2024.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22963–22974, 2025.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. VSA: Faster video diffusion with trainable sparse attention. *arXiv preprint arXiv:2505.13389*, 2025a.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. In *International Conference on Machine Learning*, pp. 74714–74731, 2025b.

Yang Zhang, Er Jin, Wenzhong Liang, Yanfei Dong, Ashkan Khakzar, Philip Torr, Johannes Stegmaier, and Kenji Kawaguchi. Learnable sparsity for vision generative models. In *International Conference on Learning Representations*, 2026.

Lin Zhao, Yushu Wu, Aleksei Lebedev, Dishani Lahiri, Meng Dong, Arpit Sahni, Michael Vasilkovsky, Hao Chen, Ju Hu, Aliaksandr Siarohin, Sergey Tulyakov, Yanzhi Wang, Anil Kag, and Yanyu Li. S2DiT: Sandwich diffusion transformer for mobile streaming video generation. *arXiv preprint arXiv:2601.12719*, 2026.

Kaiwen Zheng, Yuji Wang, Qianli Ma, Huayu Chen, Jintao Zhang, Yogesh Balaji, Jianfei Chen, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Large scale diffusion distillation via score-regularized continuous-time consistency. In *International Conference on Learning Representations*, 2026.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-Sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

Junhan Zhu, Hesong Wang, Mingluo Su, Zefang Wang, and Huan Wang. OBS-Diff: Accurate pruning for diffusion models in one-shot. In *International Conference on Learning Representations*, 2026.

Ya Zou, Jingfeng Yao, Siyuan Yu, Shuai Zhang, Wenyu Liu, and Xinggang Wang. Turbo-VAED: Fast and stable transfer of video-VAEs to mobile devices. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(16):14086–14094, 2026.

PebbleVideo: Supplementary Materials

A Derivation of the Aggregate Objective Response

We provide the derivation of Eq. 11. For a candidate block B_ℓ, the aggregate objective after removing the block is

$$\mathcal{E}_\ell = \mathbb{E}_{t,\mathbf{z}_t,\mathbf{z}_0}\left[\left\|\mathbf{z}_t - t\mathbf{v}_\theta^{-\ell}(\mathbf{z}_t,t,\mathbf{c}) - \mathbf{z}_0\right\|_2^2\right]. \tag{19}$$

The corresponding objective of the intact model is

$$\mathcal{E}_{\text{full}} = \mathbb{E}_{t,\mathbf{z}_t,\mathbf{z}_0}\left[\left\|\mathbf{z}_t - t\mathbf{v}_\theta(\mathbf{z}_t,t,\mathbf{c}) - \mathbf{z}_0\right\|_2^2\right]. \tag{20}$$

Define the residual of the intact model as

$$\mathbf{e} = \mathbf{z}_t - t\mathbf{v}_\theta(\mathbf{z}_t,t,\mathbf{c}) - \mathbf{z}_0, \tag{21}$$

and the vector-field perturbation induced by removing B_ℓ as

$$\delta\mathbf{v}_\ell = \mathbf{v}_\theta^{-\ell}(\mathbf{z}_t,t,\mathbf{c}) - \mathbf{v}_\theta(\mathbf{z}_t,t,\mathbf{c}). \tag{22}$$

It follows that

$$\mathbf{v}_\theta^{-\ell}(\mathbf{z}_t,t,\mathbf{c}) = \mathbf{v}_\theta(\mathbf{z}_t,t,\mathbf{c}) + \delta\mathbf{v}_\ell. \tag{23}$$

Substituting Eq. 23 into Eq. 19 gives

$$\mathcal{E}_\ell = \mathbb{E}_{t,\mathbf{z}_t,\mathbf{z}_0}\left[\left\|\mathbf{z}_t - t\left(\mathbf{v}_\theta(\mathbf{z}_t,t,\mathbf{c}) + \delta\mathbf{v}_\ell\right) - \mathbf{z}_0\right\|_2^2\right] = \mathbb{E}_{t,\mathbf{z}_t,\mathbf{z}_0}\left[\left\|\mathbf{e} - t\delta\mathbf{v}_\ell\right\|_2^2\right].$$

Using

$$\|\mathbf{a}-\mathbf{b}\|_2^2 = \|\mathbf{a}\|_2^2 - 2\mathbf{a}^\top\mathbf{b} + \|\mathbf{b}\|_2^2,$$

we obtain

$$\mathcal{E}_\ell = \mathbb{E}_{t,\mathbf{z}_t,\mathbf{z}_0}\left[\|\mathbf{e}\|_2^2 - 2t\,\mathbf{e}^\top\delta\mathbf{v}_\ell + t^2\|\delta\mathbf{v}_\ell\|_2^2\right]. \tag{24}$$

From Eqs. 20 and 21, we have

$$\mathcal{E}_{\text{full}} = \mathbb{E}_{t,\mathbf{z}_t,\mathbf{z}_0}\left[\|\mathbf{e}\|_2^2\right]. \tag{25}$$

Therefore, the objective change induced by removing B_ℓ is

$$\Delta\mathcal{E}_\ell = \mathcal{E}_\ell - \mathcal{E}_{\text{full}} = \mathbb{E}_{t,\mathbf{z}_t,\mathbf{z}_0}\left[-2t\,\mathbf{e}^\top\delta\mathbf{v}_\ell + t^2\|\delta\mathbf{v}_\ell\|_2^2\right],$$

which gives Eq. 11 in the main paper.

B Few-Step Distillation and Quantized Deployment

Few-step distillation. To further accelerate inference after structural compression, we employ distribution matching distillation (DMD) (Yin et al., 2024) to reduce sampling to four denoising steps. To preserve generation quality under simultaneous model and step compression, we adopt a two-stage distillation scheme. We first initialize the student via multi-step ODE trajectory regression, then apply DMD with adversarial supervision on multi-layer fake-score features, following One-Forcing (Feng et al., 2026). The second-stage generator objective is:

$$\mathcal{L}_G = \mathcal{L}_{\text{DMD}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}}, \qquad \mathcal{L}_{\text{adv}} = \mathbb{E}\left[\frac{1}{F}\sum_{f=1}^{F}\text{softplus}(-D_f(\hat{\mathbf{z}}_\tau,\tau,\mathbf{c}))\right]. \tag{26}$$

Here, $\mathcal{L}_{\text{DMD}}$ denotes the standard distribution-matching loss. The noised latent is $\hat{\mathbf{z}}_\tau = (1-\sigma_\tau)\hat{\mathbf{z}} + \sigma_\tau\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ and $\hat{\mathbf{z}}$ is the clean latent prediction at a randomly sampled step of the student's four-step denoising trajectory. D_f denotes the discriminator logit for latent frame f, computed from spatially pooled and concatenated multi-layer fake-score features conditioned on $\mathbf{c}$. F denotes the number of latent frames, and we set $\lambda_{\text{adv}} = 0.08$.

Quantized deployment. For mobile deployment, we develop an NPU-aware quantization pipeline that primarily uses W8A16 computation while retaining higher precision for quantization-sensitive operators. For multi-step generation, we calibrate over the full sampling trajectory and enforce cross-backend consistency to mitigate quantization-error accumulation. Finally, we compile each module into an independent execution graph and orchestrate cross-step reuse, on-demand residency, and final-step decoding for end-to-end inference.

C Implementation Details

C.1 Model Selection

We adopt the V2V model of ReCamMaster as the baseline for three main reasons.

First, V2V incorporates a richer set of heterogeneous condition encoders, jointly modeling textual, image, and video conditions. This unified setting enables a systematic study of how different encoder types respond to compression, and which compression strategies are better suited to different representation spaces, without relying on comparisons across separate generative models.

Second, compared with the higher generation freedom of T2V or I2V task, V2V is jointly constrained by text, images, and reference videos, providing more explicit correspondence in semantics, spatial structure, and motion. This yields a more controlled evaluation setting, reducing the influence of generation stochasticity and allowing performance degradation caused by model compression to be isolated more reliably.

Third, V2V provides a more demanding setting for on-device deployment. Video conditioning introduces additional spatiotemporal tokens, lengthening the sequence processed by global bidirectional attention and further increasing memory overhead. This directly aligns with our goal of reducing the computational and storage costs of video generation through aggressive structured compression, without restricting global interactions. Practical deployment under this heavier V2V setting therefore provides a stringent test of this direction.

C.2 Evaluation Protocol

Since our base model, ReCamMaster, performs video-to-video (V2V) generation, each text prompt requires a corresponding reference video. We therefore use the publicly released reference videos from the official VBench benchmark, generated by Wan2.1-TI2V from the corresponding VBench prompts. This keeps the evaluation inputs aligned with the VBench benchmark distribution. These reference videos are external to our method and remain fixed throughout evaluation. For T2V and I2V evaluation, we construct the corresponding inputs from the same text–image–video triplets to ensure consistent evaluation content across methods.

Because V2V generation is conditioned on the Wan2.1-TI2V reference videos, the generated videos are already tightly constrained in semantic content. We therefore focus on quality dimensions that are more sensitive to compression-induced degradation. Specifically, we report six VBench metrics: background consistency, imaging quality, temporal flickering, aesthetic quality, motion smoothness, and subject consistency. For compact reporting, we additionally report their arithmetic mean as the *VBench Score*, which is distinct from the official aggregate VBench score.

C.3 Implementation Setup

Encoder compression. We replace UMT5-XXL with UMT5-base and attach a two-layer MLP adapter that maps its 768-dimensional features to the 4096-dimensional teacher space. The adapted features are then processed by the original two-layer DiT text projector to obtain 1536-dimensional conditioning tokens. For the image branch, we directly prune the original CLIP ViT-H/14 encoder: among its 31 active transformer blocks, we retain the 16 even-indexed blocks $\{0, 2, \ldots, 30\}$ and remove the other 15. This preserves the feature width and tokenization, allowing us to reuse the native DiT image projector without an additional adapter.

DiT pruning. We cache the 50 denoising startpoints of a fixed calibration video, one for each step of the teacher trajectory. For every candidate block, we compute TAFD at all startpoints and use the mean score for selection; the remaining candidates are re-scored on the recovered student after each pruning round. TAFD supports multi-GPU parallel evaluation. Based on Eq. 12, it also only compares the suffix trajectories after their shared starting anchor. As a result, each TAFD evaluation takes less than 3 minutes on average. We prune 20 blocks in total. The first 8 selected blocks use a target pruning ratio of $7/8$ for cross-attention and FFN, whereas the remaining 12 use $3/4$; self-attention is capped at $1/2$ in both groups. These targets are reached through the three-stage schedules in Table 10. Recovery uses 100-update loss windows and stops when the relative improvement falls below 5%, subject to a budget of 500 updates per block and 10,000 updates overall.

Few-step distillation. We distill the recovered DiT into a four-step generator in two stages. For ODE initialization, a frozen 50-step Euler teacher with classifier-free guidance 3.0 and timestep shift 5.0 provides trajectory states. We sample an anchor at step 0, 12, 25, or 37 and regress the student's predicted clean latent to the teacher endpoint. We then optimize the ODE-initialized generator with DMD and a score-feature adversarial loss weighted by $\lambda_{\text{adv}} = 0.08$. The real-score network is the frozen unpruned teacher, while the trainable fake-score network is initialized from the recovered pruned DiT. The discriminator aggregates fake-score features from layers 9, 20, and 29 through a 512-dimensional head; real and generated latents are perturbed with shared noise at timesteps sampled from $[0.02, 0.5]$ of the diffusion trajectory. We activate this adversarial branch after five outer iterations, update the fake-score network and discriminator every iteration, and update the generator once every five iterations.

The implementation details for quantization and on-device deployment are provided in Appendix D.

C.4 Training Details and Data

We train the text and image encoders separately, then keep them fixed while pruning and fine-tuning the DiT. The compressed DiT is subsequently distilled into a four-step generator through ODE initialization and distribution matching. Tab. 5–8 summarize the training settings. Each stage uses 16 NVIDIA A800 GPUs. Batch sizes are reported per GPU.

Text encoder. We initialize the student from a pretrained compact UMT5 encoder and distill it using text prompts. We jointly train the student and the projections that map its outputs to the DiT's text-conditioning features, while keeping the UMT5-XXL teacher fixed. Distillation matches the final conditioning features using MSE and cosine-distance losses with weights 1.0 and 0.1.

Table 5: Training settings for text-encoder distillation.

Setting	Value
Teacher / student	UMT5-XXL / UMT5-base
Training data	Vchitect-T2V-Dataverse (Fan et al., 2025)
Optimizer	AdamW (weight decay 0.01)
Learning rate	5×10^{-4}
LR schedule	Cosine; 100 warmup updates; minimum LR 10^{-6}
Batch size per GPU	16
Training length	8 epochs
Precision	BF16
Seed	42

Image encoder. We form the image student by retaining a subset of the CLIP ViT-H/14 teacher's transformer blocks and their pretrained weights. We jointly fine-tune the retained encoder and its projection into the DiT's image-conditioning features. The teacher remains frozen, and we use the same distillation loss and weights as for the text encoder.

Table 6: Training settings for image-encoder distillation.

Setting	Value
Teacher / student	CLIP ViT-H/14 / 16 uniformly retained blocks†
Training data	COYO (Byeon et al., 2022)
Optimizer	AdamW (weight decay 0.01)
Learning rate	5×10^{-4}
LR schedule	Cosine; 50 warmup updates; minimum LR 10^{-6}
Batch size per GPU	64
Training length	8 epochs
Precision	BF16
Seed	42

DiT pruning and recovery. After each pruning stage, we fine-tune the remaining DiT parameters to match the frozen teacher's velocity predictions. The text and image encoders remain fixed. We perform at least 100 updates per stage. After the final pruning stage of a block, recovery stops when the average loss decreases by less than 5% between consecutive 100-update windows, or when the block's allocated training budget is exhausted.

Table 7: Training settings for DiT recovery after pruning.

Setting	Value
Training data	MultiCamVideo (Bai et al., 2025)
Video resolution / frames	512×320 / 49
Optimizer	AdamW (weight decay 0.01)
Learning rate	2×10^{-5}
LR schedule	One-third of the learning rate for the first 5 updates of each recovery stage; constant at 2×10^{-5} thereafter
Batch size per GPU	4
Training budget	10,000 updates
Weight precision	BF16
Seed	42

ODE initialization. We initialize the pruned DiT using trajectories generated by a frozen copy of the model with a 50-step Euler solver. For each minibatch, we sample an intermediate latent at step 0, 12, 25, or 37 and minimize the MSE between the predicted clean latent and the trajectory endpoint.

Distribution matching distillation. We initialize the generator from ODE-trained weights and train it with distribution matching and an adversarial loss ($\lambda_{\text{adv}} = 0.08$). The unpruned teacher remains frozen. A trainable fake-score model, initialized from the pruned DiT, estimates scores for generated samples and provides features to the discriminator. We update this model every iteration and the generator once every five iterations.

Table 8: Training settings for ODE initialization and DMD. Both stages use AdamW with $\beta = (0, 0.999)$ and zero weight decay.

Setting	ODE initialization	DMD
Training data	MultiCamVideo (Bai et al., 2025)	MultiCamVideo (Bai et al., 2025)
Video resolution / frames	512×320 / 49	512×320 / 49
Learning rates	Generator: 2×10^{-6}	Generator: 2×10^{-6}; fake score: 4×10^{-7}; discriminator: 5×10^{-6}
Batch size per GPU	16	2
Training length	240 updates (30 epochs)	100 outer iterations
Teacher guidance scale	3.0	3.0
Timestep shift	5.0	5.0
Precision	BF16	BF16

C.5 Baseline Reproduction Details

For the end-to-end comparisons, we obtain baseline results in one of two ways. If the required metrics are reported in the original paper, we directly use the published values. Otherwise, when an official implementation is publicly available, we download the released code and checkpoint and run inference only. Except for the output resolution and number of frames specified below, we retain all default inference settings from the official implementations.

Server-side baselines. We run the official code and checkpoints of Wan2.1-1.3B, Wan2.1-Camera, PyramidalFlow-T2V, CogVideoX-2B, Rectified SpaAttn, and ReCamMaster at 512×320 resolution for 49 frames. For LTX-Video (HaCohen et al., 2024), we use the official

`ltxv-2b-0.9.6-distilled` checkpoint with the same 512×320, 49-frame setting. For CausVid, we use the released autoregressive checkpoint, and for SANA-Video, the released SANA-Video-2B checkpoint; both are evaluated at 512×320 resolution for 49 frames. Open-Sora-1.2 and Open-Sora-1.3 (Zheng et al., 2024) are evaluated under their official 360p, 51-frame settings. The MVDiT and FastLightGen results are taken directly from their respective papers.

On-device baselines. For AMD-Hummingbird, we use the VBench results of its best-performing 0.9B, 26-frame configuration reported in the technical report. The results for SnapGen-V, MobileVD, MVDiT, and S^2DiT are taken directly from their respective papers. For On-device Sora (Kim et al., 2025), we run the official code and checkpoint using its default 256×256, 68-frame setting. We similarly evaluate NeoDragen with its official code and checkpoint at 512×320 resolution for 49 frames, and MobileWan at 832×480 resolution for 81 frames.

DiT-pruning baselines. We adapt four representative strategies to the same ReCamMaster backbone and training and evaluation protocol. OBS-Diff (Zhu et al., 2026) constructs a timestep-aware Hessian with logarithmically decaying weights over the denoising trajectory, applies second-order saliency to prune self- and cross-attention heads and FFN channels, and compensates the retained weights through OBS updates. NeoDragen (Karnewar et al., 2026) ranks complete DiT blocks by their input–output representation discrepancy, progressively removes the least influential blocks, and recovers the compact model through two-stage fine-tuning. MVDiT (Wu et al., 2025a) performs KD-guided, sensitivity-aware tri-level pruning over blocks, attention heads, and FFN channels, selecting structures under a fixed parameter budget according to the loss increase per parameter removed. FastLightGen (Shao et al., 2026) ranks blocks by the diffusion-objective degradation induced by skipping each block, estimated using Tweedie's formula, and then trains with stochastic skipping of the selected blocks before permanently removing them at inference. All methods are trained for the same total number of steps as ours.

Encoder-compression baselines. We compare with NeoDragen (Karnewar et al., 2026), which replaces UMT5-XXL with frozen UMT5-base, use a two-layer adapter to map 512-D features to 4096 dimensions, and replace the DiT text projector with a four-layer MLP. Because NeoDragen is text-conditioned, we extend this design to images: OpenCLIP ViT-H/14 is replaced by ViT-B/32, followed by token/channel alignment and a 16-linear-layer projector with seven two-layer residual MLP blocks. Both paths match teacher conditioning tokens using MSE and cosine losses.

D Quantization and On-Device Deployment Details

In this section, we provide detailed quantization and on-device deployment settings to facilitate reproducibility and offer practical guidance for deploying video model on mobile hardware. We first export the compressed video generation network into static subgraphs, including the text encoder, image encoder, video encoder, control adapter, DiT backbone, and video decoder. Tensor shapes, data layouts, and the four-step sampling schedule are explicitly fixed across module boundaries. This decomposition preserves the original conditional generation pipeline while enabling independent quantization, compilation, and runtime scheduling of individual modules. The first three sampling steps update only the latent states, while video decoding is performed only after the final step.

We adopt post-training quantization using representative video, text, and camera-trajectory samples. For the DiT backbone, calibration covers all timesteps retained after distillation, allowing activation statistics to capture the distribution shift of latent states along the full generation trajectory rather than at a single timestep or under random inputs. Computation primarily uses INT8 weights and 16-bit fixed-point activations. Higher weight precision is retained for quantization-sensitive projection layers, while FP16 is preserved only for a small subset of dynamic self-attention activations in the deployed model, yielding a predominantly fixed-point graph with sparse high-precision operations. We further explicitly quantize attention-related matrix multiplications, scaling operations, and Softmax-associated tensors, and audit quantization coverage to prevent implicit floating-point fallback from masking missing quantization boundaries.

The deployment pipeline proceeds through floating-point ONNX export, QDQ-based quantization simulation, and device graph compilation. Identical inputs are reused across stages to verify both module-wise outputs and the complete multi-step sampling trajectory. Only quantized graphs that pass both per-module and closed-loop multi-step consistency checks are converted into persistent ex-

ecution graphs for the Qualcomm HTP. We additionally record the quantization configuration, compiler settings, and model hashes to ensure traceability across the training model, quantized graph, and device binary.

At runtime, modules are loaded, executed, and released sequentially following the generation pipeline, avoiding simultaneous residency of all device graphs. The DiT execution graph is reused across the four sampling steps, with its inputs explicitly converted to fixed-point representations at the application boundary according to the compile-time quantization parameters. After the final sampling step, the DiT graph is released and the decoder is loaded to produce the final video.

E Ablation of the Overall Compression Pipeline

As shown in Tab. 9, components are added cumulatively to isolate their marginal effects. Encoder compression removes 71.25% of the parameters and lowers peak memory from 16.93 to 8.83 GiB with only a 0.47-point VBench Score drop, but provides limited acceleration because the encoders run only once. DiT pruning targets the repeatedly executed backbone, further reducing latency by 35.50%, from 59.00 to 38.05 s, at a 0.42-point quality cost. DMD distillation then delivers the largest speed gain, reducing latency by a further 95.40% to 1.75 s while recovering 0.40 VBench Score points. Overall, Ours-Server reduces the parameter count and peak memory by 80.69% and 56.00%, respectively, and achieves a $34.33\times$ end-to-end speedup while remaining within 0.49 VBench points of the baseline. These results demonstrate the complementary roles of encoder compression, DiT pruning, and few-step distillation in balancing model footprint, runtime, and generation quality.

Table 9: **Ablation of the overall compression pipeline.** Components are added cumulatively from top to bottom. Peak memory and latency are measured end-to-end on an NVIDIA RTX 5090.

Method	Efficiency			Quality
	Params (B)	Peak Memory (GiB)	Latency (s)	VBench Score
Baseline	8.13	16.93	60.08	88.23
+ Encoder compression	2.34	8.83	59.00	87.76
+ DiT pruning	1.57	7.46	38.05	87.34
+ DMD distillation (Ours-Server)	1.57	7.45	1.75	87.74

F Module Sensitivity and Intra-Block Pruning Schedules

This section quantifies the functional sensitivity and parameter return of each DiT module family, then converts this trade-off into module-specific target ratios and the stage-wise schedules used for progressive pruning.

Module-wise sensitivity analysis. We separately probe self-attention (SA), cross-attention (CA), and feed-forward (FFN) modules on the cached states $\mathcal{S}$ from Sec. 4.1, using pruning ratios r. For module family $m \in \{\mathrm{SA}, \mathrm{CA}, \mathrm{FFN}\}$, let $\theta^{(m,r)}$ denote the model obtained by structurally pruning a fraction r of module m in *every* DiT block while leaving the other module families intact. We apply the same head- or channel-wise OBS selection and compensation used by our pruning pipeline, so the probe reflects the actual structural operation.

For cached state k, the intact and module-pruned models produce equal-length projected trajectories:

$$\mathbf{\Gamma}_k = (\mathbf{g}_k^0, \ldots, \mathbf{g}_k^L), \qquad \widetilde{\mathbf{\Gamma}}_k^{m,r} = (\widetilde{\mathbf{g}}_{k,m,r}^0, \ldots, \widetilde{\mathbf{g}}_{k,m,r}^L). \tag{27}$$

Unlike block removal, module pruning retains every block endpoint; hence both trajectories contain $L+1$ points and share the anchor $\widetilde{\mathbf{g}}_{k,m,r}^0 = \mathbf{g}_k^0$. Using the raw-latent local cost $C_{ij}^{k,m,r} = \|\widetilde{\mathbf{g}}_{k,m,r}^i - \mathbf{g}_k^j\|_2$ and the recurrence in Eq. 13, we define the module-level TAFD as:

$$s_m^{\mathrm{TAFD}}(r) = \frac{1}{K} \sum_{k=1}^{K} \frac{D_{L,L}^{k,m,r}}{|\mathcal{P}_{k,m,r}^{\star}|}, \tag{28}$$

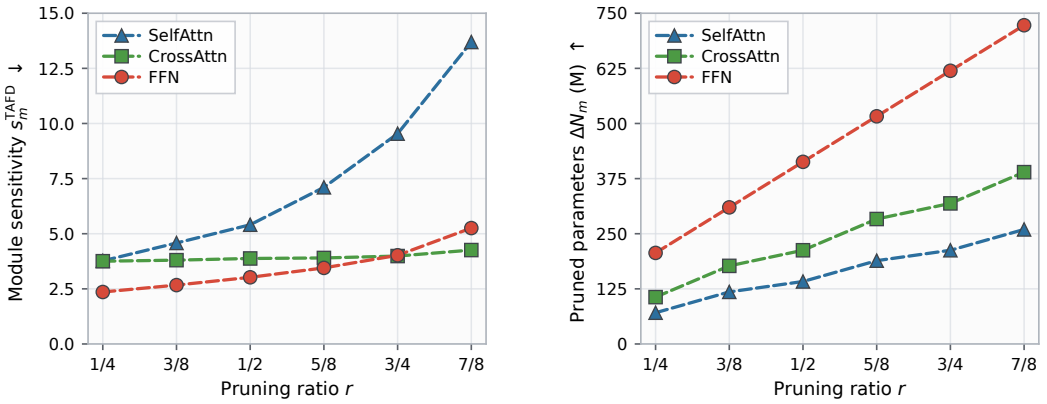


Figure 6: **Module-wise sensitivity–compression trade-off.** Left: module-level TAFD s_m^{TAFD} when pruning module family m across all DiT blocks at ratio r. Right: the corresponding number of structurally removed parameters ΔN_m. The rapidly increasing sensitivity of SelfAttn motivates its $1/2$ pruning cap, whereas CrossAttn and FFN support the more aggressive targets of $3/4$ and $7/8$.

where $D_{L,L}^{k,m,r}$ and $\mathcal{P}_{k,m,r}^{\star}$ are the cumulative cost and optimal path between the two trajectories. Averaging over all K cached sample–timestep states yields a system-level measure of jointly compressing module m across the DiT. These probes determine intra-block budgets only; block selection still uses TAFD on the current student.

Sensitivity-aware target pruning ratios. Sensitivity alone does not capture compression utility because the three module families contain different numbers of parameters. We therefore pair $s_m^{\mathrm{TAFD}}(r)$ with the true structural saving:

$$\Delta N_m(r) = N_{\mathrm{param}}(\theta) - N_{\mathrm{param}}(\theta^{(m,r)}), \tag{29}$$

where $N_{\mathrm{param}}(\cdot)$ counts stored parameters after deleting all coupled projection rows and columns, biases, and normalization weights of each pruned head or FFN channel. The resulting sensitivity–benefit curves plot r against both $s_m^{\mathrm{TAFD}}(r)$ and $\Delta N_m(r)$. As shown in Fig.6, SA enters a steeper-error regime beyond $r = 1/2$ while removing fewer parameters than CA or FFN at the same ratio, motivating a conservative cap of $1/2$. CA remains comparatively stable under aggressive pruning, whereas FFN provides the largest parameter return with a gradual TAFD increase. We consequently choose $3/4$ as the conservative operating point and $7/8$ as the aggressive operating point for CA and FFN. Formally, for the p-th TAFD-selected block,

$$q_p \in \{3/4, 7/8\}, \qquad r_{p,m}^{\star} = \begin{cases} 1/2, & m = \mathrm{SA}, \\ q_p, & m \in \{\mathrm{CA}, \mathrm{FFN}\}. \end{cases} \tag{30}$$

The global budget determines how many selected blocks use each operating point; earlier, less-sensitive blocks receive $7/8$, and later blocks receive $3/4$.

Stage-wise structural cuts. Directly pruning a module to its target ratio creates a large, abrupt function gap. We instead approach each target through three cumulative cuts, as summarized in Tab. 10. Every ratio is measured against the original module width, rather than the width remaining at that stage. For example, an FFN with target $q_p = 7/8$ is first reduced by $1/2$, then to cumulative ratios $3/4$ and $7/8$. Each cut is initialized and compensated by OBS and followed by at least R recovery updates before the next cut. These schedules replace a single aggressive cut with smaller, recoverable function perturbations.

Table 10: Cumulative intra-block removal ratios. Each tuple gives the three successive structural cuts relative to the original module width.

Target q_p	Self-attention	Cross-attention / FFN
$7/8$	$(1/6,\ 1/3,\ 1/2)$	$(1/2,\ 3/4,\ 7/8)$
$3/4$	$(1/6,\ 1/3,\ 1/2)$	$(1/2,\ 5/8,\ 3/4)$

G FUNCTION-SPACE DIT PRUNING ALGORITHMS

We detail the two procedures used in Sec. 4.1. Alg. 1 evaluates all candidate blocks on the current student, and Alg. 2 alternates this evaluation with stage-wise structural pruning and recovery.

For the p-th selected block, let $\mathcal{R}(q_p) = \{\mathbf{r}_{p,j}\}_{j=1}^{3}$ denote the cumulative schedule in Tab. 10, where $\mathbf{r}_{p,j} = (r_{p,j}^{\mathrm{SA}}, r_{p,j}^{\mathrm{CA}}, r_{p,j}^{\mathrm{FFN}})$ gives the three module-wise removal ratios at stage j. Here, $\mathcal{D}_{\mathrm{obs}}$ provides module inputs for the OBS statistics in Eq. 15, whereas $\mathcal{D}_{\mathrm{rec}}$ provides denoising tuples for Eq. 16. For compactness, $\mathrm{OBSCut}(\theta_s, B_\ell, \mathbf{r})$ collects fresh module-wise statistics on $\mathcal{D}_{\mathrm{obs}}$, structurally prunes B_ℓ to the cumulative ratios $\mathbf{r}$, and applies the compensation in Eq. 15. Likewise, $\mathrm{Recover}(\theta_s, n, \mathcal{H})$ optimizes the current student for n steps on $\mathcal{D}_{\mathrm{rec}}$ by matching its vector field to that of the frozen teacher using Eq. 16. After each update, it reapplies the structural masks and appends the resulting loss to the phase history $\mathcal{H}$.

Algorithm 1 Trajectory-Aligned Block Sensitivity Evaluation

Require: Current student θ_s, candidate blocks $\mathcal{C}$, and cached states $\mathcal{S} = \{(\mathbf{z}_{t_k}, t_k, \mathbf{c}_k)\}_{k=1}^{K}$
Ensure: TAFD scores $\{s_\ell^{\mathrm{TAFD}}\}_{B_\ell \in \mathcal{C}}$ and least-sensitive index $\ell^\star$

$S_\ell \leftarrow 0$ for each $B_\ell \in \mathcal{C}$ ▷ score accumulators
for $k = 1$ to K **do**
  Run θ_s once at $(\mathbf{z}_{t_k}, t_k, \mathbf{c}_k)$ and cache $\{\mathbf{h}_{t_k}^d, \Pi_{t_k}(\mathbf{h}_{t_k}^d)\}_{d=0}^{L}$ ▷ Eq. 7
  for $B_\ell \in \mathcal{C}$ **do**
    Let $a_1 < \cdots < a_m$ be the active downstream blocks of B_ℓ
    Form $\mathbf{\Gamma}_{k,\ell}$ from the intact cache ▷ Eq. 12
    From $\mathbf{h}_{t_k}^{\ell-1}$, bypass B_ℓ and replay the downstream suffix
    Project the states at $a_1, \ldots, a_m$ to form $\widetilde{\mathbf{\Gamma}}_{k,\ell}$ ▷ Eq. 12
    Form the raw-latent costs $C_{ij}^{k,\ell} = \|\widetilde{\mathbf{g}}_i - \mathbf{g}_j\|_2$
    Compute $D_{m,m+1}^{k,\ell}$ and $\mathcal{P}_{k,\ell}^\star$ using Eq. 13
    Remove the shared start-anchor match from $\mathcal{P}_{k,\ell}^\star$
    $S_\ell \leftarrow S_\ell + D_{m,m+1}^{k,\ell} / |\mathcal{P}_{k,\ell}^\star|$
  end for
end for
$s_\ell^{\mathrm{TAFD}} \leftarrow S_\ell / K$ for each $B_\ell \in \mathcal{C}$
$\ell^\star \leftarrow \arg\min_{\ell: B_\ell \in \mathcal{C}} s_\ell^{\mathrm{TAFD}}$ ▷ Eq. 14
Return $\{s_\ell^{\mathrm{TAFD}}\}_{B_\ell \in \mathcal{C}}, \ell^\star$

Algorithm 2 Progressive DiT Pruning and Recovery

Require: Frozen teacher θ_T, initial student $\theta_s \leftarrow \theta_T$, candidates $\mathcal{C}_1$, cached states $\mathcal{S}$, and number of blocks N_b
Require: Targets $\{q_p\}_{p=1}^{N_b}$, schedules $\{\mathcal{R}(q_p)\}_{p=1}^{N_b}$, and data $\mathcal{D}_{\text{obs}}, \mathcal{D}_{\text{rec}}$
Require: Recovery budget $\mathcal{B}_{\text{rec}} \geq N_b(3R+W)$, minimum updates R, window width W, plateau threshold τ
Ensure: Structurally compressed student θ_s

$b \leftarrow 0$ ▷ recovery updates consumed
for $p = 1$ to N_b **do**
  $(\{s_\ell^{\text{TAFD}}\}_{B_\ell \in \mathcal{C}_p}, \ell_p^\star) \leftarrow \text{TAFDEvaluate}(\theta_s, \mathcal{C}_p, \mathcal{S})$ ▷ Algorithm 1
  $\{\mathbf{r}_{p,j}\}_{j=1}^{3} \leftarrow \mathcal{R}(q_p)$
  $U_p \leftarrow \left\lfloor \dfrac{\mathcal{B}_{\text{rec}} - b}{N_b - p + 1} \right\rfloor$ ▷ current block budget
  $n_p \leftarrow 0, \quad \mathcal{H}_p \leftarrow \emptyset$
  for $j = 1$ to 3 **do**
    $\theta_s \leftarrow \text{OBSCut}(\theta_s, B_{\ell_p^\star}, \mathbf{r}_{p,j})$ ▷ Eq. 15
    $(\theta_s, \mathcal{H}_p) \leftarrow \text{Recover}(\theta_s, R, \mathcal{H}_p)$ ▷ Eq. 16
    $n_p \leftarrow n_p + R$
  end for
  while $n_p < U_p$ **do**
    $(\theta_s, \mathcal{H}_p) \leftarrow \text{Recover}(\theta_s, 1, \mathcal{H}_p)$
    $n_p \leftarrow n_p + 1$
    if $n_p \bmod W = 0$, $n_p \geq 3R + W$, and two loss windows exist **then**
      $\overline{\mathcal{L}}_{u-1}, \overline{\mathcal{L}}_u \leftarrow$ means of the latest two W-update windows in $\mathcal{H}_p$
      $\rho_u \leftarrow \dfrac{\overline{\mathcal{L}}_{u-1} - \overline{\mathcal{L}}_u}{\max(|\overline{\mathcal{L}}_{u-1}|}$
      if $\rho_u < \tau$ **then**
        break ▷ plateau exit
      end if
    end if
  end while
  $\mathcal{C}_{p+1} \leftarrow \mathcal{C}_p \setminus \{B_{\ell_p^\star}\}, \quad b \leftarrow b + n_p$ ▷ inherit unused budget
end for
if $b < \mathcal{B}_{\text{rec}}$ **then**
  $(\theta_s, \cdot) \leftarrow \text{Recover}(\theta_s, \mathcal{B}_{\text{rec}} - b, \emptyset)$ ▷ final joint recovery
end if
Return θ_s

H Pruning Trajectories Visualization

Fig. 7 complements the conceptual illustration in Fig. 2(b) with trajectories obtained from real model activations. The input–output criterion regards Block 17 as insensitive because $\mathbf{g}^{16}$ and $\mathbf{g}^{17}$ are close, yet its removal causes substantial downstream drift from the intact trajectory. Similarly, the endpoints in Fig. 7(b) are close, while the intermediate post-pruning trajectory deviates markedly. These cases confirm the motivation that local or endpoint closeness can obscure large perturbations to the full denoising path. In contrast, TAFD evaluates the aligned trajectory as a whole and selects Block 23, whose post-pruning trajectory closely follows the intact trajectory in Fig. 7(c). The supplementary `Visualization_Trajectory` folder includes trajectories for all denoising timesteps, with each timestep containing the intact trajectory and the skip trajectory of every candidate block, enabling comprehensive inspection of pruning-induced deviations.

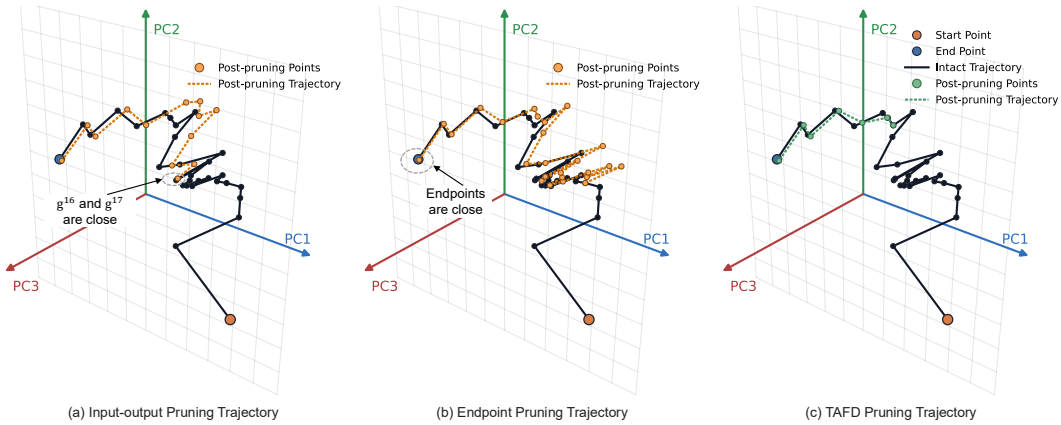


Figure 7: **Real pruning trajectories from ReCamMaster.** All trajectories are computed from real ReCamMaster activations at the 25th denoising timestep and projected into a three-dimensional PCA space. (a) Pruning Block 17 selected by input–output similarity. (b) Pruning Block 5 selected by endpoint similarity. (c) Pruning Block 23 selected by TAFD. Solid black and colored dotted curves denote the intact and post-pruning trajectories, respectively.

I Comparison of Pruning Strategies

We compare our complete DiT pruning pipeline with four representative pruning strategies adapted to the same ReCamMaster backbone. To make the comparison controlled, every method targets a comparable backbone size and follows the same training and evaluation protocol, including the same total number of training steps. Appendix C.5 describes how each baseline is reproduced.

As shown in Tab. 11, our method achieves the highest VBench score of 87.17 while using the smallest backbone, with 0.92B parameters compared with 0.94B–1.02B for the baselines. It outperforms the strongest baseline, MVDiT, by 0.43 points and ranks first on five of the six reported quality dimensions, while remaining within 0.02 on motion smoothness. These results indicate that TAFD-based block selection and progressive pruning preserve generation quality more effectively under a comparable compression budget.

Table 11: Comparison with existing DiT pruning methods. Each baseline applies its pruning strategy to ReCamMaster under a matched compression target and is trained for the same steps.

Method	Backbone Params (B)	VBench Score	Background Consistency	Imaging Quality	Temporal Flickering	Aesthetic Quality	Motion Smoothness	Subject Consistency
OBS-Diff	1.02	86.11	96.18	70.96	96.31	59.88	98.24	95.09
NeoDragen	0.94	86.38	95.98	71.76	96.50	59.92	98.40	95.72
MVDiT	0.95	86.74	96.23	72.29	96.56	60.87	98.48	96.02
FastLightGen	0.94	86.66	95.88	72.89	96.63	60.88	98.30	95.40
Ours	0.92	87.17	96.81	72.90	96.66	61.65	98.46	96.55

J Robustness of TAFD to Random Seeds and Calibration Data

Our default TAFD configuration uses random seed 42 and one calibration prompt–video pair. The prompt is *"a beach with a lot of buildings on the side of a cliff"*, and the trajectory generated from its paired video provides the startpoints used to evaluate candidate blocks.

To assess whether TAFD depends on the random seed or calibration set, we vary the seed while holding the calibration set fixed, and then increase the number of calibration prompt–video pairs while holding the seed fixed.

As shown in Tab. 12, every setting produces the same sequence of selected blocks. The complete pruning order of the 20 blocks is {23, 22, 13, 11, 12, 15, 14, 10, 20, 21, 24, 16, 6, 8, 9, 7, 5, 4, 17,

3}. This invariance across seeds and calibration-set sizes demonstrates that TAFD provides a robust block ranking rather than one tied to a particular random seed or a single calibration example.

Table 12: **Robustness of TAFD to random seeds and calibration data.** The calibration count is the number of prompt–video pairs whose trajectories provide startpoints for TAFD evaluation. All settings produce the same complete 20-block pruning order reported in Appendix J; the first eight selected blocks are shown for compactness, and the highlighted row denotes the default setting.

Random Seed	# Calibration Prompt–Video Pairs	TAFD-Selected Block Sequence (Prefix)
42	1	$\{23, 22, 13, 11, 12, 15, 14, 10, \ldots\}$
1234	1	$\{23, 22, 13, 11, 12, 15, 14, 10, \ldots\}$
0	1	$\{23, 22, 13, 11, 12, 15, 14, 10, \ldots\}$
42	2	$\{23, 22, 13, 11, 12, 15, 14, 10, \ldots\}$
42	10	$\{23, 22, 13, 11, 12, 15, 14, 10, \ldots\}$

K CKA Heatmap Visualization for Encoder Strategy Selection

We compute block-wise CKA between each initialized compression candidate and its original encoder before applying the recovery objective in Eq. 18. As shown in Fig. 8, this visualization directly exposes the initial representation gap and provides a simple criterion for selecting the compression strategy with lower recovery difficulty.

UMT5-Base exhibits structured high-similarity regions with UMT5-XXL, reaching approximately 0.6–0.8 CKA despite its reduced depth and width. We therefore adopt compact-model substitution for the text encoder. In contrast, replacing OpenCLIP ViT-H/14 with ViT-B/32 yields predominantly low CKA (0.2–0.4), whereas in-place pruning preserves substantially stronger correspondence (0.6–0.9) by retaining the original width, tokenization, and weights. We consequently use in-place pruning for the image encoder. Selecting the better-aligned initialization reduces the representation gap that subsequent DiT-interface recovery must close.

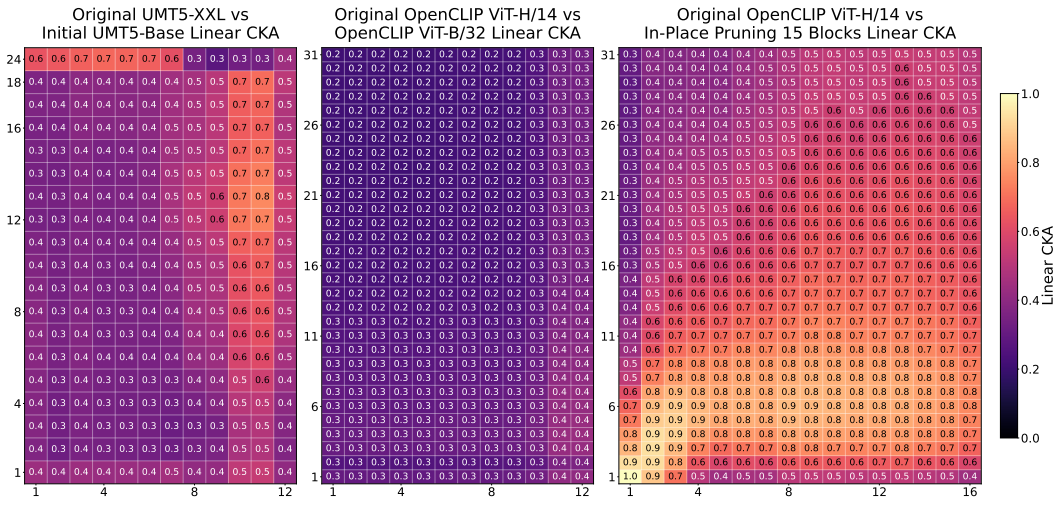


Figure 8: **Block-wise linear CKA of initialized encoder compression candidates.** Rows and columns index blocks of the original and compressed encoders, respectively. Left: UMT5-XXL vs UMT5-Base. Middle: OpenCLIP ViT-H/14 vs ViT-B/32. Right: ViT-H/14 vs in-place pruning that retains 16 of its 31 active blocks. Lighter cells indicate higher representation consistency.

L Comparison of Encoder Compression Strategies

We compare encoder compression strategies on text- and image-conditioned generation using ReCamMaster and Wan2.1. CKA measures representation consistency at the DiT interface, while VBench evaluates the resulting video quality.

As shown in Tab. 13, our compressed encoders consistently achieve higher CKA than NeoDragen across all four generation settings, with gains of 0.12–0.16 for text conditioning and 0.11–0.82 for image conditioning. These improvements are accompanied by higher VBench scores under matched or smaller parameter budgets, while the more aggressive Ours-Small variant remains better than NeoDragen on both metrics. These results validate selecting the encoder compression operation according to representation consistency.

Table 13: Comparison with existing encoder compression methods. CKA ↑ measures linear similarity between the compressed and original DiT-facing conditioning representations, while Params (B) counts the encoder and DiT condition projection. VBench Score is evaluated at 320p-49 frames for ReCamMaster and 480p-81 frames for Wan2.1. Ours-Small adopts a more aggressive encoder compression, replacing UMT5-XXL with UMT5-Small for the text encoder while retaining only 12 blocks for the image encoder.

Method	Text Encoder Compression						Image Encoder Compression					
	ReCamMaster-V2V			Wan2.1-1.3B-T2V			ReCamMaster-V2V			Wan2.1-14B-I2V		
	Params	CKA	VBench	Params	CKA	VBench	Params	CKA	VBench	Params	CKA	VBench
Baseline	5.69	1.00	88.23	5.69	1.00	86.86	0.64	1.00	88.23	0.64	1.00	86.52
NeoDragen	0.32	0.74	87.54	0.32	0.73	78.81	0.34	0.15	87.18	0.35	0.36	86.25
Ours	0.31	0.86	87.77	0.31	0.89	85.22	0.34	0.97	88.22	0.35	0.47	86.51
Ours-Small	0.18	0.81	87.75	0.18	0.82	84.15	0.26	0.62	88.15	0.27	0.45	86.40

M Peak Memory Breakdown

Fig. 9 compares the peak GPU memory of the ReCamMaster baseline and PebbleVideo on an NVIDIA RTX 5090 under the same $512 \times 320 \times 49$ generation setting.

Encoder compression reduces the text-encoder peak from 11.56 to 0.69 GiB, while DiT pruning lowers the backbone peak from 4.52 to 2.94 GiB. Consequently, the end-to-end peak memory decreases by 56.0%, from 16.93 to 7.45 GiB. Although parallel VAE decoding raises the pipeline activation peak from 1.79 to 4.53 GiB, structural compression reduces weight residency from 15.14 to 2.93 GiB and remains the dominant source of the overall memory saving.

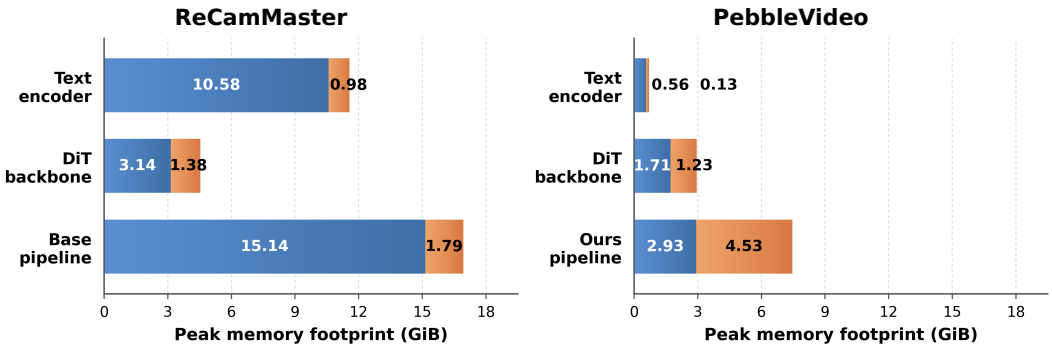


Figure 9: **Peak memory breakdown of ReCamMaster and PebbleVideo.** Blue and orange denote memory occupied by model weights and peak activations, respectively.

N HUMAN PREFERENCE EVALUATION

To evaluate whether PebbleVideo preserves perceptual video quality under aggressive model compression, we conduct a human preference study. We adopt an Arena-style anonymous voting protocol, in which participants compare two videos without knowing the corresponding methods, as shown in Fig. 10. The evaluated samples are drawn from the ReCamMaster baseline, together with PebbleVideo, On-device Sora, and NeoDragen, resulting in 94 video groups and 376 videos in total. The study involves 30 participants, including 17 researchers from the National University of Singapore, University of Toronto, University of Illinois, Tsinghua University, Peking University, Nanjing University, Beihang University, Beijing Institute of Technology, Renmin University of China, Huazhong University of Science and Technology, Wuhan University, South China University of Technology, and the Institute of Automation at the Chinese Academy of Sciences, as well as 13 non-research participants.

The Arena collected 355 valid pairwise votes in total. As shown in Fig. 11, when compared with the uncompressed ReCamMaster baseline, PebbleVideo achieves a preference rate of 38.6% versus 61.4%, demonstrating competitive perceptual quality despite its substantial reduction in model size and inference cost. Against On-device Sora, PebbleVideo obtains a substantially higher preference rate of 76.5% versus 23.5%. The advantage remains pronounced against NeoDragen, where PebbleVideo is preferred in 68.4% of comparisons versus 31.6%. These results indicate that PebbleVideo retains competitive human-perceived quality relative to its uncompressed teacher while providing markedly better perceptual quality than competing efficient video generation methods.

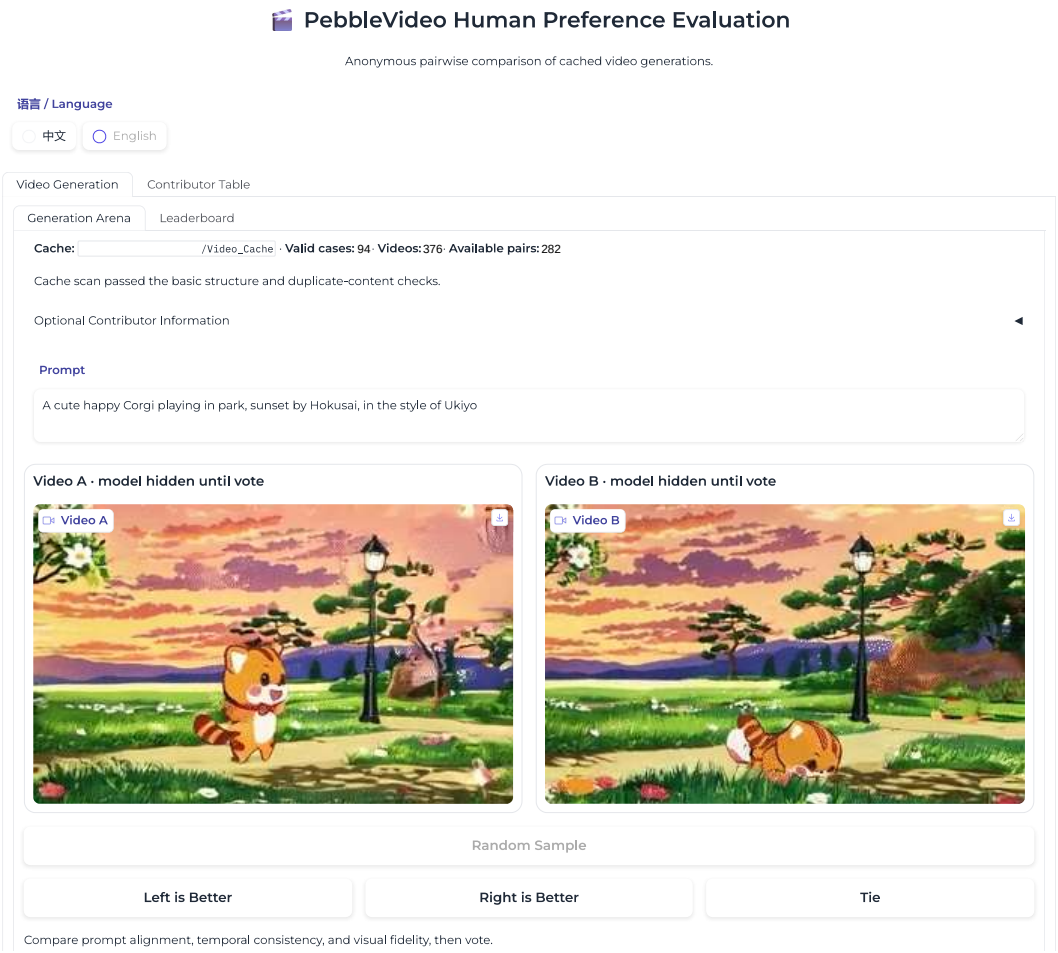


Figure 10: Arena-style anonymous human preference evaluation interface.

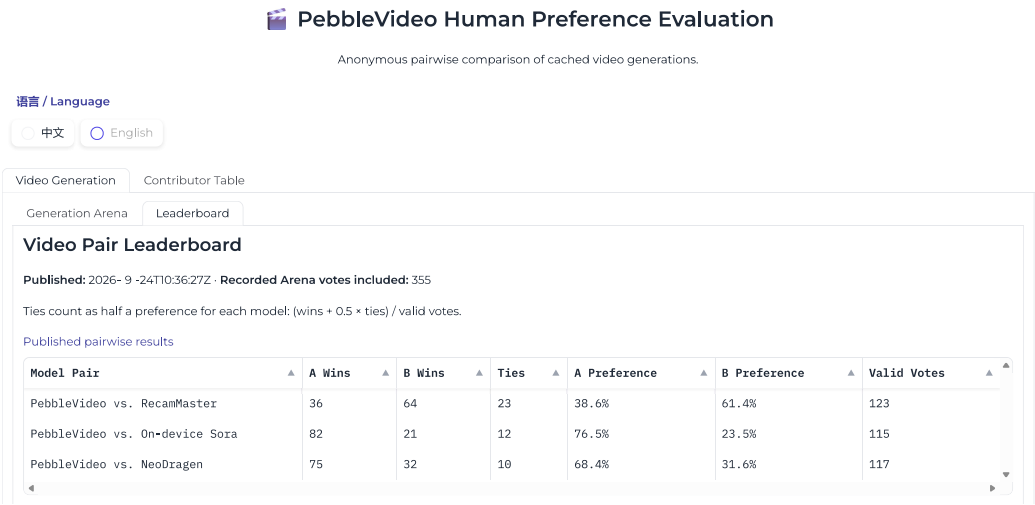


Figure 11: Arena-style anonymous human preference evaluation results.

O Extended Related Work

Video Generation Models. Diffusion models have become a dominant paradigm for video generation, evolving from spatiotemporal U-Net architectures (Ho et al., 2022b; Blattmann et al., 2023) to scalable diffusion transformers. Representative models, including CogVideoX (Yang et al., 2025), HunyuanVideo (Kong et al., 2024), and Wan (Team Wan et al., 2025), employ full bidirectional attention to enable global interactions across spatial and temporal tokens, improving long-range dependency modeling, motion coherence, and visual fidelity. Beyond conventional full-sequence denoising, Pyramidal Flow (Jin et al., 2025) explores hierarchical autoregressive generation to reduce inference cost. Meanwhile, video diffusion has expanded from text-to-video to image-to-video and video-to-video generation, further broadening its applicability. However, quadratic-complexity bidirectional attention and billion-scale parameter counts incur substantial computational, memory, and storage overhead, hindering deployment on resource-constrained devices.

Efficient Attention and Streaming Generation. The quadratic complexity of full self-attention is particularly costly for DiTs due to long spatiotemporal token sequences. Recent work alleviates this bottleneck through causal generation, sub-quadratic attention, sparse token interactions, and recurrent computation. Beyond CausVid, S^2DiT, and MobileWan discussed in the main text, SANA-Video (Chen et al., 2025b) employs block-wise linear attention with recurrent states to enable efficient long-video generation. LinVideo (Huang et al., 2026) adapts pretrained video DiTs to linear-complexity attention through post-training adaptation. Complementary methods retain softmax attention while exploiting structured sparsity: Rectified SpaAttn (Liu et al., 2025) adopts sparse attention to reduce token-interaction costs while preserving generation quality; Trainable Sparse Attention (Zhang et al., 2025a) learns sparse interaction patterns, whereas Sliding Tile Attention (Zhang et al., 2025b) restricts attention to scheduled spatiotemporal tiles. Streaming methods further reduce runtime overhead through chunk-wise generation. Pyramidal Flow (Jin et al., 2025) organizes generation into spatial-temporal pyramids, enabling efficient autoregressive generation with compressed full-resolution history. StreamingT2V (Henschel et al., 2025) propagates short- and long-term context across autoregressive chunks, while HiStream (Qiu et al., 2025) exploits spatial, temporal, and timestep redundancy with a fixed-size anchor cache. Collectively, these methods improve efficiency by redesigning spatiotemporal token interactions or generation topology.

Pruning Diffusion Transformers. Structured pruning has recently emerged as an effective approach to reducing the parameter and computational costs of DiTs without training compact generators from scratch. For image DiTs, TinyFusion (Fang et al., 2025) formulates depth reduction as a learnable pruning problem and explicitly optimizes the recoverability of the resulting shallow model. Learnable Sparsity (Zhang et al., 2026), originally released as Effortless Efficiency, learns structural sparsity for vision generative models. OBS-Diff (Zhu et al., 2026) estimates one-shot structural importance using second-order information, enabling diffusion model pruning with limited recovery

cost. For video DiTs, MobileVD (Ben Yahia et al., 2025) combines channel and temporal-block pruning with multi-scale temporal processing for mobile image-to-video generation. MVDiT (Wu et al., 2025a) performs sensitivity-aware tri-level pruning over transformer components, attention heads, and blocks, followed by knowledge distillation. Neodragon (Karnewar et al., 2026) ranks MMDiT blocks by the local feature discrepancy, removing low-importance blocks before two-stage distillation. FastLightGen (Shao et al., 2026) evaluates block sensitivity through diffusion-objective degradation. MobileWan (Ghafoorian et al., 2026) instead prunes self-attention heads by learning binary per-head gates with sparsity regularization and subsequent distillation. These studies reveal substantial structural redundancy in DiTs. In contrast, we evaluate structural sensitivity through the induced perturbation to the diffusion vector field, directly aligning pruning decisions with the denoising function.

Condition Encoder and VAE Compression. Large encoders and video VAEs contribute substantially to the memory footprint of diffusion pipelines. For text conditioning, DistillT5 (Wang et al., 2025) distills large T5 encoders into compact variants while preserving conditioning quality. Neodragon (Karnewar et al., 2026) extends this approach to video generation by replacing the T5 encoder with a lightweight DistillT5 and a learned context adapter. In contrast, image-encoder compression for diffusion models remains largely unexplored. Video VAEs constitute another major memory and computational bottleneck. WF-VAE (Li et al., 2025) exploits wavelet decomposition for efficient spatiotemporal compression, while LeanVAE (Cheng & Yuan, 2025) and Turbo-VAED (Zou et al., 2026) develop lightweight, hardware-friendly decoders for efficient reconstruction. Neodragon (Karnewar et al., 2026) and MobileWan (Ghafoorian et al., 2026) further compress video decoders through distillation for mobile deployment. TAEHV (Boer Bohan, 2025) provides compact drop-in autoencoders for video diffusion models, substantially reducing decoding latency and memory. Given the maturity of video VAE compression, we directly replace the original VAE with TAEHV and focus our compression design on condition encoders, where cross-scale representation consistency determines whether to adopt compact-model substitution or in-place structural pruning.

Sampling-Step Distillation. Sampling-step distillation reduces diffusion inference cost by distilling multi-step generation into a few-step student. Progressive Distillation (Salimans & Ho, 2022) introduced iterative trajectory compression, while Distribution Matching Distillation (DMD) (Yin et al., 2024) matches the student distribution to that of a pretrained teacher. Few-step distillation has since been extended to large-scale image and video generators. APT (Lin et al., 2025) uses adversarial post-training for one-step video generation, while SANA-Sprint (Chen et al., 2025a) combines consistency and adversarial distillation for one-to-four-step image synthesis. CausVid (Yin et al., 2025) adapts DMD to causal students, SnapGen-V (Wu et al., 2025b) searches hardware-efficient spatial-temporal architectures and couples the resulting compact model with adversarial step distillation, Neodragon (Karnewar et al., 2026) extends DMD to pyramidal flow matching, and FastLightGen (Shao et al., 2026) jointly reduces model depth and sampling steps. Further variants include rCM (Zheng et al., 2026) with score-regularized consistency training, Phased DMD (Fan et al., 2026) with subinterval-wise distribution matching, and Transition Matching Distillation (Nie et al., 2026) with conditional transition-flow distillation. Our system applies DMD-based distillation after structural compression to further reduce denoising steps.

Quantization and Hardware Deployment. Quantization further reduces model storage, memory bandwidth, and computational cost for diffusion deployment. Recent DiT quantization methods, such as Q-DiT (Chen et al., 2025c) and Q-VDiT (Feng et al., 2025), address activation variations across timesteps, tokens, and channels. DVD-Quant (Li et al., 2026) further explores aggressive W4A4 post-training quantization for video DiTs. Despite their effectiveness, most low-bit methods are evaluated on GPUs, whereas practical mobile deployment typically relies on hardware-aware PTQ/QAT and mixed-precision execution. MobileWan (Ghafoorian et al., 2026), for example, adopts 8-bit weights, mixed-precision activations, and distillation-based QAT for mobile NPU deployment. Our work similarly applies quantization after structural and sampling-step compression to enable practical on-device execution.

P QUALITATIVE COMPARISONS

Fig. 12–15 compare ReCamMaster (Teacher), On-device Sora, NeoDragen, and PebbleVideo. Each row shows six frames sampled uniformly over the corresponding video. The supplementary `Visualization_Video` folder further provides video examples.



Figure 12: **Prompt:** windmill.



Figure 13: **Prompt:** A cute happy Corgi playing in park, sunset by Hokusai, in the style of Ukiyo.



Figure 14: **Prompt:** A beautiful coastal beach in spring, waves lapping on sand, pan right.



Figure 15: **Prompt:** a flock of birds flying over a tree at sunset.