LIGHTVAE: TOWARDS COMPACT AND EFFICIENT VIDEO AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent diffusion models enable high-quality video generation, but video VAE encoding and decoding remain computationally expensive. Existing methods use stage-specific lightweight operators, yet either constrain spatial and temporal computation jointly or remove temporal computation altogether, without distinguishing their compression sensitivities in pretrained Conv3D. To address these limitations, we present LightVAE, a compression framework that exploits redundancy in temporal modeling and network depth. LightVAE exactly reparameterizes eligible causal Conv3D layers into an aggregate spatial response and a difference-driven temporal correction. Based on their distinct compression sensitivities, temporal low-rank compression (TLR) keeps the aggregate spatial kernel dense and applies layer-specific low-rank approximation only to the temporal correction. An algebraically equivalent folded implementation converts the arithmetic savings into practical acceleration. LightVAE further uses width-preserving block pruning to reduce depth and probe-guided feature alignment, guided by feedback from a frozen teacher suffix, to recover quality after compression. The framework also extends to video encoders. Across three video VAE backbones and two GPU platforms, LightVAE consistently accelerates decoding. For the Wan VAE family, it achieves up to $2.24\times$ and $1.11\times$ the decoding throughput of state-of-the-art efficient VAE decoders on NVIDIA H100 and RTX 5090, respectively, with higher DAVIS PSNR and lower LPIPS. On MiniMax-H3, it achieves up to $2.97\times$ the original decoding throughput while retaining 99.51% of the original model's overall VBench-2.0 score. Our code is available at `https://anonymous.4open.science/r/LightVAE_ICLR_27-9243/`.

1 INTRODUCTION

Recent video generation models have advanced in visual fidelity, motion coherence, and prompt alignment (Yang et al., 2025; HaCohen et al., 2024; Team Wan et al., 2025). Many operate in a latent space (Rombach et al., 2022), where a video variational autoencoder (VAE) maps between pixels and compact spatiotemporal representations, while a diffusion Transformer (DiT) (Peebles & Xie, 2023) generates video latents through iterative denoising. Although this design reduces generation cost, both denoising and VAE inference remain expensive for high-resolution or long videos. Most acceleration methods target denoising. Diffusion distillation reduces the number of sampling steps (Salimans & Ho, 2022; Yin et al., 2024b;a; 2025; Fan et al., 2026); caching reuses computation across steps (Liu et al., 2025); and efficient attention, quantization, and optimized kernels reduce per-step cost (Zhang et al., 2025; LightX2V Contributors, 2025). As denoising becomes faster, VAE inference can account for a larger fraction of end-to-end latency (Zou et al., 2026; Zhu et al., 2026). In convolutional video VAEs, causal Conv3D becomes particularly expensive in high-resolution decoder stages, where dense spatiotemporal kernels operate on large feature maps. We therefore focus primarily on decoder compression.

A practical approach is to compress a pretrained decoder while preserving its latent interface and recovering quality through teacher supervision. Turbo-VAED (Zou et al., 2026) replaces dense Conv3D with depthwise-separable Conv3D in low-resolution stages while retaining dense Conv3D at higher resolutions. Flash-VAED (Zhu et al., 2026) further combines depthwise-separable Conv3D with channel-pruned Conv2D in high-resolution stages. Flash-VAED's stage-wise replacement experiments show that substituting Conv3D with Conv2D in later, higher-resolution stages yields

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.

greater latency reductions with smaller quality losses, providing an empirical basis for reducing temporal computation in these stages. However, the depthwise-separable Conv3D used by both methods constrains joint spatial filtering and channel mixing, limiting its ability to reproduce pretrained dense-convolution responses. Flash-VAED's pruned Conv2D removes local temporal interactions while also narrowing spatial representations, restricting the spatial mappings inherited from the teacher. Conv2D replacement fixes the affected layers to spatial-only processing rather than providing adjustable temporal capacity. These observations motivate examining whether the spatial and temporal contributions of pretrained Conv3D have different compression sensitivities.

Motivated by this question, we introduce *LightVAE*, a compression framework that combines two complementary forms of structural compression: temporal low-rank compression (TLR) within eligible convolutions and width-preserving pruning across network blocks (Figure 1). TLR reduces temporal computation within retained layers, whereas block pruning reduces network depth without narrowing the retained feature representations. LightVAE further uses an algebraically equivalent folded implementation for efficient TLR execution and probe-guided feature alignment for reconstruction recovery.

TLR begins with an exact reparameterization of pretrained causal Conv3D that expresses its output as an aggregate spatial response plus a temporal correction driven by historical–current feature differences. Under the same cumulative singular-value retention criterion, our branch-specific ablations show that temporal-only factorization produces smaller decoder-output errors than spatial-only or joint factorization, while joint-factorization errors remain close to those of spatial-only factorization. These results lead to TLRConv2D, a stateful operator that keeps the aggregate spatial kernel dense and applies truncated SVD only to the temporal correction. Layer-specific ranks control temporal capacity and computational cost; Figure 2 illustrates the fidelity benefit of retaining the correction. To translate the reduced arithmetic into wall-clock gains, we derive an algebraically equivalent folded implementation that absorbs the current-frame correction into the spatial kernel and caches low-dimensional history projections. This implementation avoids explicit online feature differences, reduces convolution calls and intermediate tensors without additional rank truncation, and improves measured throughput on both evaluated GPUs.

At the depth level, high shortcut–output cosine similarity suggests potential block redundancy (Figure 9). We therefore complement TLR with width-preserving block pruning, adapting TinyFusion's recoverability-aware search (Fang et al., 2025) to convolutional and Transformer blocks while retaining required shortcuts, projections, and resampling operations. This depth reduction is complementary to TLR because it acts across blocks rather than within retained temporal convolutions. For reconstruction recovery, uniform feature MSE weights coordinates equally without accounting for their downstream effects. We instead introduce probe-guided feature alignment, which combines gradients from a frozen teacher suffix with signed student–teacher feature discrepancies to construct detached weights. These weights suppress matching directions that locally increase the probe loss in feature space, while the probe loss itself is excluded from the recovery objective.

LightVAE-Pro applies TLR at the original decoder depth, whereas LightVAE-Lite additionally applies width-preserving block pruning using a recoverability-aware search adapted from TinyFusion (Fang et al., 2025). For Transformer-based autoencoders, the framework uses block pruning without TLR. We summarize our contributions as follows:

- We introduce *TLR*, exploiting the different compression sensitivities of spatial and temporal computation in pretrained causal Conv3D. TLR preserves dense spatial modeling while compressing only the temporal component; an equivalent folded implementation reduces convolution calls and intermediate tensors for practical acceleration.
- We introduce a general structural compression-and-recovery framework that works for both convolutional and Transformer autoencoders. Its probe-guided recovery uses a frozen teacher suffix to weight feature alignment according to how feature discrepancies affect the final reconstruction.
- Across three backbones and two GPUs, LightVAE consistently accelerates decoding. It reaches $2.24\times$/$1.11\times$ the throughput of state-of-the-art efficient Wan decoders (Team Wan et al., 2025; Wan Team, 2025) on H100/RTX 5090 with higher DAVIS PSNR/lower LPIPS, and $2.97\times$ speedup on MiniMax-H3 (MiniMax, 2026) with 99.51% VBench-2.0 retention.

2 Related Work

Efficient video autoencoders and operator design. Video autoencoder acceleration explores lightweight architectures (Cheng & Yuan, 2025) and compression of pretrained decoders (Zou et al., 2026; Zhu et al., 2026), the latter preserving the latent interface of an existing diffusion backbone. Turbo-VAED (Zou et al., 2026) uses depthwise-separable Conv3D in low-resolution stages and dense Conv3D at higher resolutions. Flash-VAED (Zhu et al., 2026) combines depthwise-separable Conv3D with channel-pruned Conv2D, demonstrating that later stages tolerate reduced temporal computation. Depthwise-separable Conv3D constrains joint spatial filtering and channel mixing, while channel-pruned Conv2D removes local temporal interactions and narrows spatial representations. Prior work also factorizes spatiotemporal convolutions (Qiu et al., 2017; Tran et al., 2018) or applies low-rank kernel approximation (Lebedev et al., 2015). TLR instead exactly separates an aggregate spatial response from a difference-driven temporal correction, preserving the dense spatial kernel at conversion while controlling temporal capacity through layer-specific ranks.

Structured pruning and knowledge distillation. Channel pruning reduces network width (He et al., 2017), whereas block pruning reduces depth while preserving the feature widths of retained blocks. TinyFusion (Fang et al., 2025) jointly learns structural selection and parameter adaptation to identify diffusion Transformers with strong post-pruning recoverability. LightVAE adapts this search principle to convolutional and Transformer blocks in video autoencoders. Reconstruction recovery commonly uses output or intermediate-feature distillation (Hinton et al., 2015; Romero et al., 2015). CrossKD (Wang et al., 2024) passes student features through a teacher detection head and directly distills the resulting cross-head predictions. Gradient-Guided Knowledge Distillation (Lan & Tian, 2024) uses task gradients to weight task-relevant features during distillation. LightVAE instead uses gradients from a frozen teacher-suffix probe, together with signed student–teacher feature discrepancies, to construct detached feature-alignment weights. These weights suppress matching directions that locally increase the probe loss in feature space. Unlike direct cross-head prediction distillation, the probe loss itself is excluded from the recovery objective.

3 Preliminaries

A video variational autoencoder (VAE) maps a video V to a latent representation Z through an encoder $\mathcal{E}$ and reconstructs it as Y through a decoder $\mathcal{D}$. Latent video diffusion models generate in this representation space. We focus on decoder compression, but the framework also extends to encoders that use causal Conv3D. Experiments on reference-conditioned encoding are reported in Appendix G.2. Scalars are italic, vectors bold lowercase, matrices bold uppercase, and tensors bold sans-serif uppercase. Subscripts T, S, and P denote teacher, student, and probe quantities. We use $\mathrm{sg}(\cdot)$ for stop-gradient and $\mathrm{vec}(\cdot)$ for vectorization, omitting batch dimensions throughout. Consider a dense causal Conv3D with K temporal taps and unit temporal stride and dilation:

$$\mathsf{Y}_t = \sum_{j=0}^{K-1} \mathsf{W}_j * \mathsf{X}_{t-j} + \boldsymbol{b}, \tag{1}$$

where $*$ denotes spatial convolution, $\mathsf{X}_t \in \mathbb{R}^{C_{\mathrm{in}} \times H \times W}$, $\mathsf{W}_j \in \mathbb{R}^{C_{\mathrm{out}} \times C_{\mathrm{in}} \times k_h \times k_w}$, and $\boldsymbol{b} \in \mathbb{R}^{C_{\mathrm{out}}}$ is broadcast over spatial positions. The output has shape $C_{\mathrm{out}} \times H_{\mathrm{out}} \times W_{\mathrm{out}}$. The index $j = 0$ denotes the current frame, independently of kernel storage order, and preceding inputs follow the source layer's temporal padding rule. For the matrix form, let $d = C_{\mathrm{in}} k_h k_w$ and $P = H_{\mathrm{out}} W_{\mathrm{out}}$. Reshape each kernel into $\boldsymbol{W}_j \in \mathbb{R}^{C_{\mathrm{out}} \times d}$ and arrange spatial input patches as columns of $\boldsymbol{X}_t \in \mathbb{R}^{d \times P}$, using the source convolution's spatial stride, dilation, and padding. Then

$$\boldsymbol{Y}_t = \sum_{j=0}^{K-1} \boldsymbol{W}_j \boldsymbol{X}_{t-j} + \boldsymbol{b}\mathbf{1}_P^\top, \tag{2}$$

where $\boldsymbol{Y}_t \in \mathbb{R}^{C_{\mathrm{out}} \times P}$ contains the output features and $\mathbf{1}_P \in \mathbb{R}^P$ is the all-ones vector. Patch matrices are used only for derivation and are not materialized during inference.

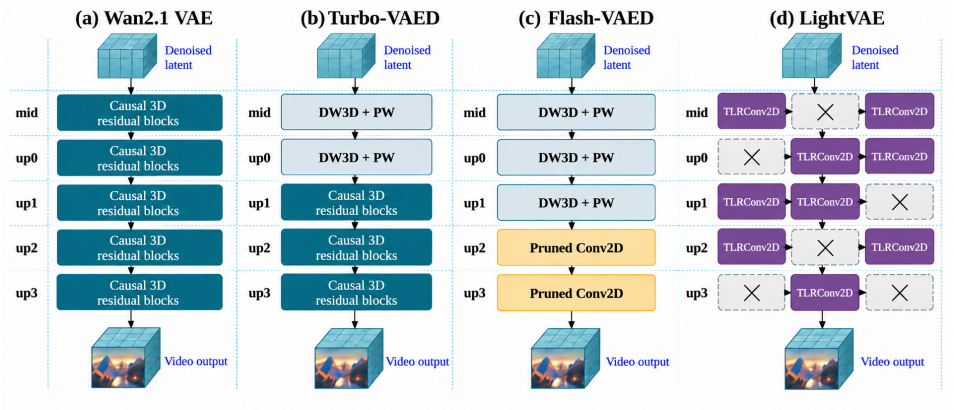


Figure 1: **Architectural comparison.** LightVAE replaces eligible causal Conv3D layers with TLRConv2D and optionally removes network blocks. TLRConv2D retains the dense aggregate spatial response while applying a layer-specific low-rank approximation only to the temporal correction. Width-preserving block pruning independently reduces network depth, complementing TLR's reduction of per-layer temporal computation.

4 LightVAE

LightVAE combines temporal low-rank compression with width-preserving block pruning (Figure 1). For pruned models, block selection and LoRA merging precede TLR conversion. Probe-guided feature alignment then supports reconstruction recovery, followed by weight folding and projected-history caching for inference.

4.1 Temporal Low-Rank Compression

Temporal low-rank compression (TLR) approximates a pretrained causal convolution while preserving its aggregate spatial response. Below, we derive the resulting TLRConv2D operator. Consider the three-tap convolution in Equation (2), with unit temporal stride and dilation.

Spatial response and temporal correction. Define the aggregate spatial kernel $\boldsymbol{S} = \boldsymbol{W}_0 + \boldsymbol{W}_1 + \boldsymbol{W}_2$, concatenate the historical kernels as $\boldsymbol{M}_{\text{temp}} = [\boldsymbol{W}_1, \boldsymbol{W}_2] \in \mathbb{R}^{C_{\text{out}} \times 2d}$, and stack the temporal patch differences:

$$\Delta \boldsymbol{X}_t = \begin{bmatrix} \boldsymbol{X}_{t-1} - \boldsymbol{X}_t \\ \boldsymbol{X}_{t-2} - \boldsymbol{X}_t \end{bmatrix} \in \mathbb{R}^{2d \times P}. \tag{3}$$

Adding and subtracting the historical kernels applied to the current input yields the exact decomposition

$$\boldsymbol{Y}_t = \underbrace{\boldsymbol{S}\boldsymbol{X}_t}_{\text{spatial response}} + \underbrace{\boldsymbol{M}_{\text{temp}} \Delta \boldsymbol{X}_t}_{\text{temporal correction}} + \boldsymbol{b}\mathbf{1}_P^\top. \tag{4}$$

The first term applies the sum of all temporal kernel slices to the current feature. The second term corrects this response using differences between the historical and current features.

Approximation allocation. TLR retains the aggregate spatial kernel and approximates only the temporal correction. A spatial-kernel error acts on $\boldsymbol{X}_t$, whereas a temporal-correction error acts on $\Delta \boldsymbol{X}_t$. Preserving $\boldsymbol{S}$ therefore removes the spatial approximation term. Temporal continuity motivates this allocation, and the stage-wise comparison in Figure 5 evaluates it under the selected truncation rule. The branch-error decomposition and bound are given in Appendix B.

Temporal-correction factorization. For a prescribed rank r, the best rank-r approximation of $\boldsymbol{M}_{\text{temp}}$ in the Frobenius norm is given by its truncated singular value decomposition (SVD):

$$\begin{aligned} \boldsymbol{M}_{\text{temp},r} &= \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^\top = \boldsymbol{A}[\boldsymbol{B}_1, \boldsymbol{B}_2], \\ \boldsymbol{A} &= \boldsymbol{U}_r \boldsymbol{\Sigma}_r, \qquad [\boldsymbol{B}_1, \boldsymbol{B}_2] = \boldsymbol{V}_r^\top. \end{aligned} \tag{5}$$

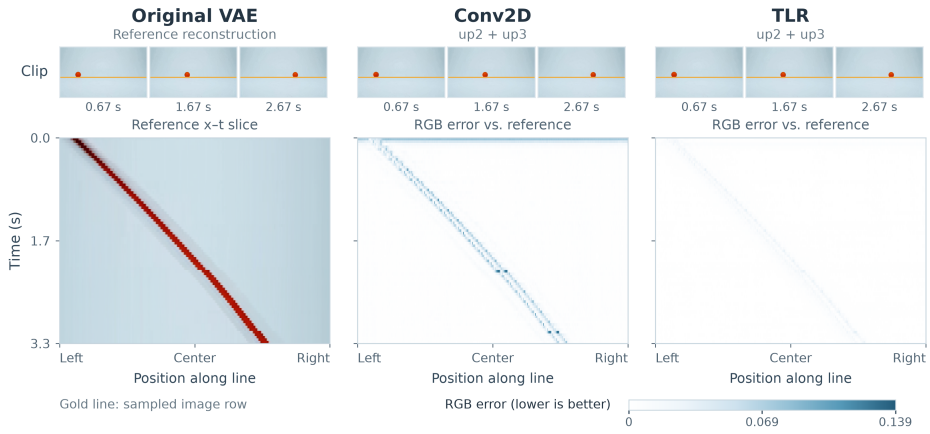


Figure 2: **Effect of retaining the temporal correction.** Only residual convolutions in Wan2.1's `up2`/`up3` stages are replaced by Conv2D or TLRConv2D ($\eta = 0.6$), without recovery. **Top:** Reconstructed frames with the sampled row marked in gold. **Bottom:** The original VAE's x–t slice and RGB error maps for Conv2D and TLR on a shared scale.

Here, each $\boldsymbol{B}_j \in \mathbb{R}^{r\times d}$ is implemented as a spatial Conv2D with r output channels, and $\boldsymbol{A} \in \mathbb{R}^{C_{\text{out}}\times r}$ is implemented as a pointwise Conv2D. The first and last d columns of $\boldsymbol{V}_r^\top$ form $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$, respectively. This partition introduces no approximation beyond the SVD and allows the two historical kernels to share the output projection $\boldsymbol{A}$. The resulting operator is

$$\widehat{\boldsymbol{Y}}_t = \boldsymbol{S}\boldsymbol{X}_t + \boldsymbol{b}\boldsymbol{1}_P^\top + \boldsymbol{A}[\boldsymbol{B}_1, \boldsymbol{B}_2]\Delta\boldsymbol{X}_t. \tag{6}$$

The resulting stateful operator, TLRConv2D, implements TLR by imposing a low-rank constraint on $\boldsymbol{M}_{\text{temp}}$. At conversion, $\boldsymbol{S}$ is preserved exactly, and the temporal factors are initialized from pretrained weights by SVD. The shared threshold η specifies the minimum fraction of each layer's temporal singular-value sum to retain, yielding layer-specific ranks according to their singular-value spectra. Layer ranks remain fixed during recovery and inference. For identical layer inputs, SVD initialization gives $\|\widehat{\boldsymbol{Y}}_t - \boldsymbol{Y}_t\|_{\text{F}} \le \sigma_{r+1}\|\Delta\boldsymbol{X}_t\|_{\text{F}}$. For a fixed truncation, smaller historical–current feature differences yield a smaller bound on the layer-output error. Appendix B provides the local error analysis and rank-selection details. Figure 2 compares Conv2D and TLRConv2D replacements applied only to the residual convolutions in the `up2`/`up3` stages of the Wan2.1 VAE decoder (Team Wan et al., 2025), without recovery. Relative to the original VAE reconstruction, TLR produces smaller x–t slice errors in this example, illustrating the benefit of retaining the temporal correction.

Computational cost. The convolutional multiply–accumulate (MAC) counts per output feature frame are

$$\begin{aligned} c_{\text{3D}} &= 3PC_{\text{out}}d, \\ c_{\text{TLR}} &= P(C_{\text{out}}d + 2rd + C_{\text{out}}r). \end{aligned} \tag{7}$$

TLR reduces the MAC count when $r < 2C_{\text{out}}d/(2d + C_{\text{out}})$. Appendix C.1 provides the derivation and counting conventions.

4.2 Weight-Folded Execution with Projected-History Caching

The difference form in Equation (6) requires three spatial-convolution calls and explicitly constructs two temporal differences. As illustrated in Figure 3, we reduce operator-call and intermediate-tensor overhead by folding the current-frame correction into the spatial kernel and computing the current response and low-rank features from the same current-frame input.

Weight folding. After recovery, the fixed parameters allow the current-frame terms to be collected as

$$\boldsymbol{S}_{\text{eff}} = \boldsymbol{S} - \boldsymbol{A}(\boldsymbol{B}_1 + \boldsymbol{B}_2). \tag{8}$$

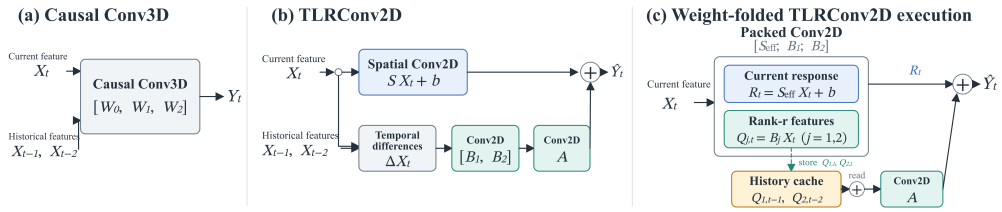


Figure 3: Causal Conv3D, TLRConv2D, and its weight-folded execution.

The product contracts the intermediate channel dimension and is reshaped to the same spatial-kernel dimensions as $\boldsymbol{S}$. For each incoming frame, define

$$\mathbf{Q}_{j,t} = \boldsymbol{B}_j \boldsymbol{X}_t \in \mathbb{R}^{r \times P}, \qquad j \in \{1, 2\}. \tag{9}$$

Equation (6) then becomes

$$\widehat{\boldsymbol{Y}}_t = \underbrace{\boldsymbol{S}_{\text{eff}} \boldsymbol{X}_t + \boldsymbol{b}\mathbf{1}_P^\top}_{\mathbf{R}_t} + \boldsymbol{A}(\mathbf{Q}_{1,t-1} + \mathbf{Q}_{2,t-2}). \tag{10}$$

The current-frame correction is contained in $\mathbf{R}_t$, leaving only the lagged projections in the history term.

Joint convolution and projected history. Because $\boldsymbol{S}_{\text{eff}}$, $\boldsymbol{B}_1$, and $\boldsymbol{B}_2$ share the same input, their output channels can be concatenated into one Conv2D:

$$\begin{bmatrix} \mathbf{R}_t \\ \mathbf{Q}_{1,t} \\ \mathbf{Q}_{2,t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_{\text{eff}} \\ \boldsymbol{B}_1 \\ \boldsymbol{B}_2 \end{bmatrix} \boldsymbol{X}_t + \begin{bmatrix} \boldsymbol{b} \\ \mathbf{0}_r \\ \mathbf{0}_r \end{bmatrix} \mathbf{1}_P^\top. \tag{11}$$

This joint convolution has $C_{\text{out}} + 2r$ output channels and retains the source spatial stride, dilation, and padding. The cached projections $\mathbf{Q}_{1,t-1}$ and $\mathbf{Q}_{2,t-2}$ are summed, projected by $\boldsymbol{A}$, and added to $\mathbf{R}_t$. The current projections are cached for outputs at $t+1$ and $t+2$, respectively, with the required history read before any cache entry is overwritten. Cache initialization follows the source temporal-padding rule; the state persists across chunks of the same video and is reset between videos. Folding preserves the MAC count in Equation 7 while reducing convolution calls and intermediate tensors.

4.3 Block Selection and Recovery

Block selection. We adapt TinyFusion's recoverability-aware search (Fang et al., 2025) to video VAE blocks under fixed group-wise retention quotas. Required shortcuts, projections, and resampling operations are retained. Selection logits and shared LoRA parameters (Hu et al., 2022) are trained with frozen base weights using an $\mathcal{L}_1$ reconstruction term and LPIPS (Zhang et al., 2018). After search, LoRA updates are merged into the retained weights before TLRConv2D initialization. The search applies to convolutional and compatible Transformer blocks. Appendix D summarizes the grouped search, pruning units, and export procedure.

Probe-guided recovery. At a compatible feature boundary, a frozen teacher suffix maps the student's feature to a reconstruction, which is compared with the detached teacher output. Let $\mathbf{g}$ denote the probe loss's input gradient and $\boldsymbol{\delta}$ the signed student–teacher feature discrepancy. Ordinary feature MSE updates coordinate i along $-\delta_i$, which locally increases the probe loss when $\delta_i g_i < 0$. We use $\mathbf{g}$ and $\boldsymbol{\delta}$ to construct nonnegative alignment weights w_i that suppress these conflicting updates. The weighted feature loss is $\mathcal{L}_{\text{feature}} = N_f^{-1} \sum_i \text{sg}(w_i)\delta_i^2$, where N_f is the number of feature elements. Appendix E defines the weighting rule and analyzes its local feature-space alignment. The student is trained with

$$\begin{aligned} \mathcal{L}_{\text{recon}} &= \mathcal{L}_1(\mathbf{Y}_{\text{S}}, \mathbf{V}) + \lambda_{\text{perc}} \,\text{LPIPS}(\mathbf{Y}_{\text{S}}, \mathbf{V}) + \lambda_{\text{adv}} \mathcal{L}_{\text{adv}}, \\ \mathcal{L}_{\text{restore}} &= \mathcal{L}_{\text{recon}} + \lambda_{\text{F}} \mathcal{L}_{\text{feature}}. \end{aligned} \tag{12}$$

Here, $\mathcal{L}_1$, LPIPS, and $\mathcal{L}_{\text{adv}}$ denote reconstruction, perceptual, and adversarial losses. Loss coefficients and training settings are given in Appendix F. We cycle through compatible group boundaries, evaluating one probe per iteration. Its gradients determine detached feature weights; its loss is excluded from $\mathcal{L}_{\text{restore}}$. Teacher and probe computations are used only during recovery.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

For Wan2.1/2.2 (Team Wan et al., 2025; Wan Team, 2025), Pro retains all 14 residual blocks, while Lite keeps one residual main branch per stage ($14 \rightarrow 5$). MiniMax-H3 (MiniMax, 2026) uses block pruning only, reducing decoder Transformer blocks ($36 \rightarrow 12$) and encoder residual main branches ($12 \rightarrow 6$). Throughput is measured on NVIDIA H100 and GeForce RTX 5090 using 81-frame 480×832/704×1280 clips for Wan2.1/Wan2.2 and 124-frame 768×1344 clips for MiniMax-H3. Reconstruction uses 33 consecutive native-resolution frames on UCF-101 (Soomro et al., 2012) and all frames at 720×1280 on the DAVIS 2019 Challenge test set (Caelles et al., 2019). Where required, last-frame repetition yields $4n + 1$ inputs; repeated frames are excluded from scoring. Ablations use 33-frame 480×832 clips unless specified otherwise. Video-level RGB PSNR and frame-averaged SSIM (Wang et al., 2004) and LPIPS-VGG (Zhang et al., 2018; Simonyan & Zisserman, 2015) are averaged equally across videos. For VBench-2.0 (Zheng et al., 2025), LightX2V (LightX2V Contributors, 2025) uses the throughput-test resolutions with 81/121/124 frames for Wan2.1/Wan2.2/MiniMax-H3, respectively. Within each backbone, only the decoder varies; prompts, seeds, and sampling settings remain fixed. Appendix F provides compression hyperparameters, training settings, loss coefficients, and software versions.

5.2 MAIN RESULTS

Reconstruction and generation quality. Tables 1 and 2 summarize LightVAE's speed–quality trade-offs. On Wan2.1, Lite achieves $1.12\times$/$1.11\times$ the throughput of Flash-VAED on H100/RTX 5090, with higher DAVIS PSNR and lower LPIPS, while Pro closely preserves the original reconstruction quality. On Wan2.2, Pro surpasses Turbo-VAED on all reconstruction metrics across both datasets while achieving higher H100 throughput (38.84 vs. 29.24 FPS). Across the Wan VAEs, Lite reaches up to $4.06\times$/$4.20\times$ the original throughput on H100/RTX 5090. On RTX 5090, Wan2.2 Lite reaches 31.87 FPS, whereas Turbo-VAED runs out of memory under our 81-frame protocol, compared with the standard 17-frame evaluation in its original paper (Zou et al., 2026). Both LightVAE variants use $r = 0$ in the converted `up2`/`up3` residual convolutions, consistent with evidence that later stages tolerate reduced temporal computation (Zhu et al., 2026). The ablations in Section 5.3 separately show that TLR gives the lowest matched-MAC error among the tested allocations, block pruning reaches higher recovered PSNR than channel pruning at similar pre-recovery throughput, probe-guided recovery improves over reconstruction-only recovery, and folding increases throughput on both GPUs. On MiniMax-H3, Lite achieves $2.95\times$/$2.97\times$ the original decoding throughput on H100/RTX 5090 while retaining 99.51% of the original VBench-2.0 overall score. Reference encoding is accelerated by up to $2.68\times$, with jointly compressed encoders and decoders producing visually similar outputs. Appendices G.1 and G.2 provide detailed scores and reference-conditioned comparisons.

5.3 ABLATION STUDIES

Spatial–temporal approximation and rank selection. At a shared retention threshold, temporal-only approximation yields the smallest output errors, while joint errors closely track spatial-only errors (Figure 5). Temporal errors decrease from `up1` to `up3`, consistent with Flash-VAED's stage-wise findings (Zhu et al., 2026). TLR also achieves the lowest error among the tested matched-MAC allocations without recovery (Figure 6). Near its 70% MAC floor, spatial-only compression leaves little spatial capacity; joint compression allocates more budget to S and reduces error. At 80%–95%, spatial-only compression regains spatial capacity while retaining dense M_{temp}, outperforming the joint allocation that keeps both branches low-rank. These results support preserving S and prioritizing compression of M_{temp}. Appendix F.2 details the protocols and allocation constraints. Figure 7 shows increasing fidelity with higher η at reduced throughput; we use $\eta = 0.6$ as the default balance.

Operator quality and inference efficiency. Figure 4 compares TLR with summed-slice Conv2D and DW3D+PW/Conv2D hybrids using Flash-VAED's layout (Zhu et al., 2026). Teacher-SVD initialization, adapted from network decoupling (Guo et al., 2018), improves the hybrids over random-DW initialization, but TLR retains higher PSNR before and after matched recovery. This supports

Table 1: VAE reconstruction quality and decoding throughput on NVIDIA H100 and RTX 5090. OOM denotes out of memory. Best compressed results per backbone are **bold**.

Model	FPS↑		UCF-101			DAVIS-2019		
	NVIDIA H100	RTX 5090	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Wan2.1 VAE	52.48	30.92	33.5021	0.94087	0.05343	34.2021	0.92363	0.08627
Flash-VAED	191.04	116.55	30.7241	0.91399	0.09013	31.9097	0.90491	0.12768
LightVAE-Pro	104.02	61.88	**33.4373**	**0.94339**	**0.05585**	**33.9764**	**0.92071**	**0.09478**
LightVAE-Lite	**213.28**	**129.94**	30.9058	0.91453	0.09027	32.4032	0.90302	0.11575
Wan2.2 VAE	23.95	10.18	33.9193	0.94532	0.05134	34.2463	0.92134	0.08315
Turbo-VAED	29.24	OOM	31.8020	0.93116	0.06627	30.0160	0.89384	0.13801
LightVAE-Pro	38.84	18.58	**33.3065**	**0.94110**	**0.05776**	**33.5184**	**0.91585**	**0.09774**
LightVAE-Lite	**65.41**	**31.87**	30.8672	0.91330	0.08969	31.5560	0.89150	0.13570
MiniMax-H3 VAE	18.15	9.85	31.6361	0.91876	0.09328	32.2907	0.89980	0.14918
LightVAE-Lite	**53.52**	**29.23**	**28.8349**	**0.87828**	**0.13696**	**29.3048**	**0.85393**	**0.21310**

Table 2: VBench-2.0 generation quality (0–100; higher is better).

Generation backbone	Decoder	VBench-2.0↑					
		Human Fidelity	Controllability	Creativity	Physics	Commonsense	Overall
Wan2.1-1.3B	Original decoder	85.3388	21.7736	52.3006	43.8093	61.7110	52.9867
	Flash-VAED	83.1481	21.4708	49.6575	**46.7925**	61.4237	52.4985
	LightVAE-Pro	**84.8408**	**21.8701**	**51.9213**	45.8570	61.1363	**53.1251**
	LightVAE-Lite	84.1943	21.8338	48.4021	44.8614	**61.7110**	52.2005
Wan2.2-5B	Original decoder	86.3657	22.6547	56.6134	53.5895	65.1593	56.8765
	Turbo-VAED	86.3387	**23.2704**	**58.1631**	54.8722	**66.0214**	**57.7332**
	LightVAE-Pro	**86.4226**	23.0824	57.6016	**55.3117**	65.1593	57.5155
	LightVAE-Lite	85.9445	23.1687	55.7466	54.6638	65.1593	56.9366
MiniMax-H3	Original decoder	83.9687	54.4288	45.6439	78.3034	72.0710	66.8832
	LightVAE-Lite	**82.6056**	**55.2702**	**45.4439**	**75.3712**	**74.0825**	**66.5546**

preserving dense spatial mappings while approximating the temporal correction. Folding improves throughput on both GPUs. Retaining these mappings and the low-rank correction costs more than the lighter operators, while combining TLR with block pruning provides the favorable speed–quality trade-offs in Table 1.

Joint operator and depth compression. Table 3 uses the Wan input settings of Table 1. Pruning retains 5 of 14 residual main branches, while unstarred TLR uses $\eta = 0.6$ throughout. Depth pruning alone achieves $1.85\times$–$2.35\times$ the original throughput, and folded TLR alone achieves $1.14\times$–$1.41\times$. Crucially, applying folded TLR after pruning provides a further $1.17\times$–$1.40\times$ throughput increase over pruning alone. This incremental gain demonstrates their complementarity: pruning reduces network depth, whereas TLR reduces the temporal-convolution cost within the retained network. TLR* sets $r = 0$ in `up2`/`up3` and uses $\eta = 0.6$ elsewhere. Combined with pruning, it matches LightVAE-Lite in Table 1 and achieves $1.27\times$–$1.30\times$ the throughput of the unstarred combination.

Block pruning and recovery. High shortcut–output similarity suggests depth redundancy (Figure 9, left; Appendix D.2). For comparable inference cost, block pruning retains 6 of 14 residual main branches, while channel pruning removes approximately 41.7% of channels. Both retain dense Conv3D, reaching 45.58/42.69 FPS on RTX 5090 before recovery. Block pruning converges faster and reaches higher PSNR under matched recovery (right), supporting width-preserving depth reduction. Separately, Figure 8 fixes the architecture at 5 of 14 branches: uniform feature MSE underperforms reconstruction alone, whereas probe-guided alignment improves recovery. Appendix E analyzes the local alignment of feature updates with the probe objective.

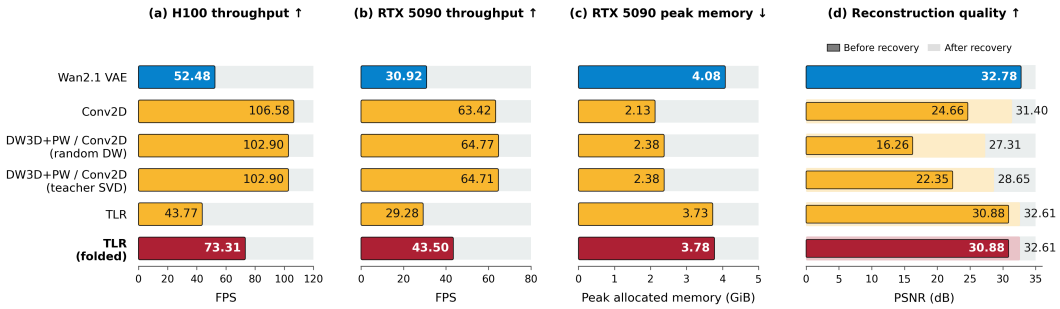


Figure 4: Wan2.1 VAE operator comparison at the original depth and width. (a,b) H100/RTX 5090 throughput; (c) RTX 5090 peak allocated memory; (d) DAVIS PSNR before (dark) and after (light) matched recovery.

Table 3: Decoding throughput (FPS↑) under individual and joint compression. Parentheses indicate speedup over the original decoder on the same GPU and backbone.

Configuration	NVIDIA H100		RTX 5090	
	Wan2.1 VAE	Wan2.2 VAE	Wan2.1 VAE	Wan2.2 VAE
Original	52.48 (1.00×)	23.95 (1.00×)	30.92 (1.00×)	10.18 (1.00×)
Depth pruning only	123.20 (2.35×)	44.19 (1.85×)	71.57 (2.31×)	19.73 (1.94×)
TLR only	73.31 (1.40×)	27.24 (1.14×)	43.50 (1.41×)	12.83 (1.26×)
Depth pruning + TLR	165.54 (3.15×)	51.65 (2.16×)	100.08 (3.24×)	24.51 (2.41×)
Depth pruning + TLR*	**213.28** (**4.06**×)	**65.41** (**2.73**×)	**129.94** (**4.20**×)	**31.87** (**3.13**×)

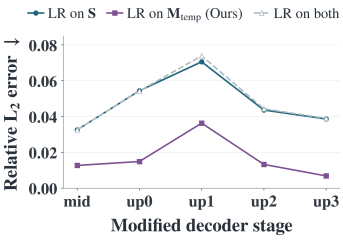


Figure 5: Mean relative L_2 error under stage-wise approximation.

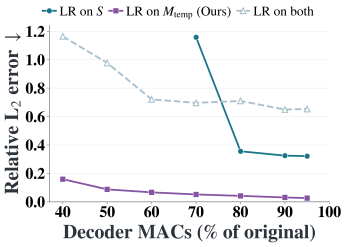


Figure 6: Mean relative L_2 error at matched MAC budgets.

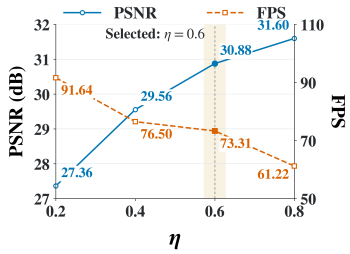


Figure 7: PSNR and throughput versus retention threshold η.

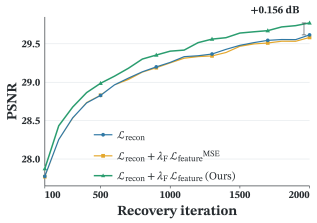


Figure 8: DAVIS PSNR with different recovery losses.

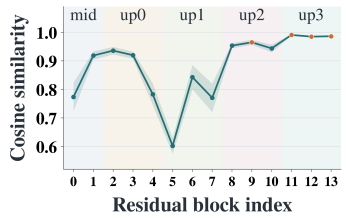
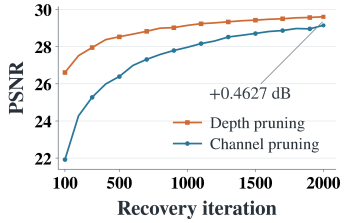


Figure 9: Shortcut–output cosine similarity (left) and DAVIS PSNR under depth and channel pruning (right).



6 Conclusion

We presented LightVAE, combining temporal low-rank compression, folded execution, width-preserving block pruning, and probe-guided recovery. It achieves up to 2.24×/1.11× the throughput of evaluated efficient Wan decoders on H100/RTX 5090 with higher DAVIS PSNR and lower LPIPS. On MiniMax-H3, block pruning provides up to 2.97× decoding throughput while retaining 99.51% of the VBench-2.0 score, and extends to efficient reference encoding.

AI USE STATEMENT

In this work, generative AI tools were used solely to improve the language, readability, and presentation of the manuscript. They were not used for any task requiring disclosure under the ICLR 2027 Policy on the Use of AI by Authors. The authors reviewed all AI-assisted revisions to ensure that the original technical meaning was preserved and take full responsibility for the final content of the manuscript, including all AI-assisted language edits.

REFERENCES

Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. The 2019 DAVIS challenge on VOS: Unsupervised multi-object segmentation. *arXiv preprint arXiv:1905.00737*, 2019.

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 (ICCV)*, pp. 15692–15702, 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 (CVPR)*, pp. 18144–18154, 2025.

Jianbo Guo, Yuxi Li, Weiyao Lin, Yurong Chen, and Jianguo Li. Network decoupling: From regular to depthwise separable convolutions. In *British Machine Vision Conference (BMVC)*, 2018.

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.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1389–1397, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

Qizhen Lan and Qing Tian. Gradient-guided knowledge distillation for object detectors. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 424–433, 2024.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. In *International Conference on Learning Representations*, 2015.

LightX2V Contributors. LightX2V: Light video generation inference framework. GitHub repository, 2025. URL `https://github.com/ModelTC/LightX2V`.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7353–7363, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

MiniMax. MiniMax H3. GitHub repository, 2026. URL https://github.com/MiniMax-AI/MiniMax-H3.

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

Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with Pseudo-3D residual networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 5533–5541, 2017.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015.

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

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Team Wan et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6450–6459, 2018.

Wan Team. Wan2.2: Open-source video generation models. GitHub repository, 2025. URL https://github.com/Wan-Video/Wan2.2.

Jiabao Wang, Yuming Chen, Zhaohui Zheng, Xiang Li, Ming-Ming Cheng, and Qibin Hou. CrossKD: Cross-head knowledge distillation for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16520–16530, 2024.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861.

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, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis. In *Advances in Neural Information Processing Systems*, volume 37, pp. 47455–47487, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo 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 (CVPR)*, pp. 6613–6623, 2024b.

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 (CVPR)*, pp. 22963–22974, 2025.

Jintao Zhang, Kaiwen Zheng, Kai Jiang, Haoxu Wang, Ion Stoica, Joseph E Gonzalez, Jianfei Chen, and Jun Zhu. Turbodiffusion: Accelerating video diffusion models by 100-200 times. *arXiv preprint arXiv:2512.16093*, 2025.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and Ziwei Liu. VBench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

Lunjie Zhu, Yushi Huang, Xingtong Ge, Yufei Xue, Zhening Liu, Yumeng Zhang, Zehong Lin, and Jun Zhang. Flash-VAED: Plug-and-play VAE decoders for efficient video generation. *arXiv preprint arXiv:2602.19161*, 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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 14086–14094, 2026.

APPENDIX

A Limitations **13**

B Analysis of Temporal Low-Rank Compression **13**

B.1 Local Error and Comparison with Conv2D 14

B.2 Rank Selection and Input Dependence 15

C Efficient TLRConv2D: Cost and Execution **15**

C.1 Computational Cost and Rank Condition 15

C.2 Folding Equivalence and History Indexing 16

D Block-Pruning Implementation **17**

D.1 Selection, Adaptation, and Export 17

D.2 Feature Visualization 17

E Teacher-Suffix Probe and Feature Alignment **18**

F Experimental Details **19**

F.1 Training and Recovery 19

F.2 Compression and Ablation Settings 20

F.3 Inference Benchmarks 21

G Additional Downstream Evaluation **21**

G.1 Fine-Grained VBench-2.0 Results 21

G.2 Reference-Conditioned Generation 22

A LIMITATIONS

The convolutional evaluation of TLR currently focuses on the Wan VAE family; the MiniMax-H3 experiments evaluate block pruning rather than TLR. Temporal ranks are selected from weight spectra, block retention follows predefined group-wise quotas, and measured latency depends on hardware and implementation. In addition, the probe weighting provides a local feature-space alignment property rather than a parameter-space descent guarantee. Future work could learn ranks and retention counts jointly from activation and latency statistics and evaluate TLR across additional video-autoencoder families.

B ANALYSIS OF TEMPORAL LOW-RANK COMPRESSION

Unless otherwise stated, we analyze a three-tap dense causal convolution with unit temporal stride and unit dilation, identical layer inputs, and identical bias and boundary handling. The SVD error results describe local conversion at initialization, whereas the folding identity in Appendix C.2 remains exact after recovery. For LightVAE-Lite, the source weights include the LoRA updates merged after block selection.

B.1 Local Error and Comparison with Conv2D

Exact decomposition and branch errors. Substituting $\boldsymbol{S} = \boldsymbol{W}_0 + \boldsymbol{W}_1 + \boldsymbol{W}_2$ into Equation (4) gives

$$\begin{aligned} &\boldsymbol{S}\boldsymbol{X}_t + \boldsymbol{M}_{\text{temp}}\Delta\boldsymbol{X}_t \\ &= (\boldsymbol{W}_0 + \boldsymbol{W}_1 + \boldsymbol{W}_2)\boldsymbol{X}_t + \boldsymbol{W}_1(\boldsymbol{X}_{t-1} - \boldsymbol{X}_t) + \boldsymbol{W}_2(\boldsymbol{X}_{t-2} - \boldsymbol{X}_t) \\ &= \boldsymbol{W}_0\boldsymbol{X}_t + \boldsymbol{W}_1\boldsymbol{X}_{t-1} + \boldsymbol{W}_2\boldsymbol{X}_{t-2}. \end{aligned} \tag{B.1}$$

Applying the source bias once recovers the original response. If the spatial and temporal-correction matrices are approximated by $\widetilde{\boldsymbol{S}}$ and $\widetilde{\boldsymbol{M}}_{\text{temp}}$, respectively, while inputs and bias remain unchanged, the output error is

$$\widetilde{\mathbf{E}}_t = (\widetilde{\boldsymbol{S}} - \boldsymbol{S})\boldsymbol{X}_t + (\widetilde{\boldsymbol{M}}_{\text{temp}} - \boldsymbol{M}_{\text{temp}})\Delta\boldsymbol{X}_t. \tag{B.2}$$

The triangle inequality and $\|\mathbf{AB}\|_{\text{F}} \le \|\mathbf{A}\|_2\|\mathbf{B}\|_{\text{F}}$ give

$$\begin{aligned} \|\widetilde{\mathbf{E}}_t\|_{\text{F}} \le \|\widetilde{\boldsymbol{S}} - \boldsymbol{S}\|_2\|\boldsymbol{X}_t\|_{\text{F}} \\ + \|\widetilde{\boldsymbol{M}}_{\text{temp}} - \boldsymbol{M}_{\text{temp}}\|_2\|\Delta\boldsymbol{X}_t\|_{\text{F}}. \end{aligned} \tag{B.3}$$

Preserving $\boldsymbol{S}$ eliminates the spatial approximation term, leaving only the temporal-correction error.

Fixed-rank weight approximation. Let $q = \text{rank}(\boldsymbol{M}_{\text{temp}})$ and write its SVD as $\boldsymbol{M}_{\text{temp}} = \sum_{i=1}^{q} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$, where $\sigma_1 \ge \cdots \ge \sigma_q > 0$ and $\sigma_{q+1} = 0$. For $0 \le r \le q$, truncated SVD minimizes the rank-r weight-approximation error in the Frobenius norm:

$$\|\boldsymbol{M}_{\text{temp},r} - \boldsymbol{M}_{\text{temp}}\|_{\text{F}}^2 = \sum_{i=r+1}^{q} \sigma_i^2. \tag{B.4}$$

The corresponding output error is input-dependent, as quantified below.

Local output error. Because $\boldsymbol{S}$ is preserved exactly, the conversion error at SVD initialization is

$$\mathbf{E}_t = \widehat{\boldsymbol{Y}}_t - \boldsymbol{Y}_t = (\boldsymbol{M}_{\text{temp},r} - \boldsymbol{M}_{\text{temp}})\Delta\boldsymbol{X}_t. \tag{B.5}$$

The SVD residual has spectral norm σ_{r+1}, so submultiplicativity gives

$$\|\mathbf{E}_t\|_{\text{F}} \le \sigma_{r+1}\|\Delta\boldsymbol{X}_t\|_{\text{F}}. \tag{B.6}$$

For a fixed truncation, the bound scales with the historical–current feature differences. More precisely,

$$\mathbf{E}_t = -\sum_{i=r+1}^{q} \sigma_i \mathbf{u}_i(\mathbf{v}_i^\top \Delta\boldsymbol{X}_t). \tag{B.7}$$

Orthogonality of the left singular vectors gives

$$\|\mathbf{E}_t\|_{\text{F}}^2 = \sum_{i=r+1}^{q} \sigma_i^2 \|\mathbf{v}_i^\top \Delta\boldsymbol{X}_t\|_2^2. \tag{B.8}$$

Thus, the actual output error depends on both the discarded singular values and the feature-difference directions.

Rank monotonicity. For fixed source weights, input patches, and ordered SVD components, any $0 \le r_1 \le r_2 \le q$ satisfies

$$\|\mathbf{E}_t^{(r_1)}\|_{\text{F}}^2 - \|\mathbf{E}_t^{(r_2)}\|_{\text{F}}^2 = \sum_{i=r_1+1}^{r_2} \sigma_i^2 \|\mathbf{v}_i^\top \Delta\boldsymbol{X}_t\|_2^2 \ge 0. \tag{B.9}$$

Taking the spatial-only reference $\widehat{\boldsymbol{Y}}_t^{(0)} = \boldsymbol{S}\boldsymbol{X}_t + \boldsymbol{b}\mathbf{1}_P^\top$ gives

$$\|\widehat{\boldsymbol{Y}}_t^{(r)} - \boldsymbol{Y}_t\|_{\text{F}}^2 \le \|\widehat{\boldsymbol{Y}}_t^{(0)} - \boldsymbol{Y}_t\|_{\text{F}}^2. \tag{B.10}$$

Each added singular component can only reduce the local conversion error; rank q recovers the source layer exactly. At $r = 0$, the operator reduces to the summed-slice Conv2D initialized with $\boldsymbol{S}$.

B.2 Rank Selection and Input Dependence

Singular-value retention. For layers using spectral rank selection, a nonzero temporal-correction matrix and shared threshold $\eta \in (0, 1]$ determine the smallest rank satisfying

$$r = \min \left\{ k \in \{1, \dots, q\} : \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{q} \sigma_i} \geq \eta \right\}. \tag{B.11}$$

The rule retains a fraction η of the singular-value sum and yields a layer-specific rank from each spectrum. Unless otherwise specified, $\eta = 0.6$; ranks are fixed after conversion, and a zero temporal-correction matrix uses no temporal factors. The matched-MAC ablation instead allocates ranks by retained squared singular-value energy per added MAC under a prescribed decoder budget (Appendix F.2).

Dependence on video features. Let $\mathbf{R}_{\text{temp}} = \boldsymbol{M}_{\text{temp},r} - \boldsymbol{M}_{\text{temp}}$ and $\mathbf{C}_\Delta = \mathbb{E}[\Delta \boldsymbol{X}_t \Delta \boldsymbol{X}_t^\top]$, where the expectation is over input videos and temporal positions. Then

$$\mathbb{E}\|\mathbf{E}_t\|_{\text{F}}^2 = \operatorname{tr}\big(\mathbf{R}_{\text{temp}} \mathbf{C}_\Delta \mathbf{R}_{\text{temp}}^\top\big). \tag{B.12}$$

Equation (B.12) makes the dependence on the feature-difference covariance explicit. Figures 5 and 6 evaluate the spectral and matched-MAC protocols, respectively; Figure 7 motivates the default $\eta = 0.6$.

C Efficient TLRConv2D: Cost and Execution

C.1 Computational Cost and Rank Condition

Counting convention. One multiply–accumulate (MAC) denotes an operation of the form $a \leftarrow a + wx$. Counts are reported for one output feature frame, with $d = C_{\text{in}} k_h k_w$ and $P = H_{\text{out}} W_{\text{out}}$. Standalone additions, feature differences, bias additions, nonlinearities, and memory transfers are excluded. Counting multiplication and addition separately gives approximately two floating-point operations per MAC. For N videos with T_{out} output frames, the counts scale by NT_{out}; network-level totals sum the actual feature shapes and ranks of all layers. The subscript TLR identifies the computational cost of the TLRConv2D operator.

Source and low-rank operators. Each temporal kernel slice costs $PC_{\text{out}}d$ MACs, so

$$c_{\text{3D}} = \sum_{j=0}^{2} PC_{\text{out}}d = 3PC_{\text{out}}d. \tag{C.1}$$

The exact spatial-response/temporal-correction rewriting alone does not change this count. TLRConv2D retains the dense spatial response, computes two rank-r temporal projections, and applies one shared pointwise output projection:

$$\begin{aligned} c_{\text{TLR}} = \; & \underbrace{PC_{\text{out}}d}_{\text{spatial response}} + \underbrace{2Prd}_{\text{temporal projections}} \\ & + \underbrace{PC_{\text{out}}r}_{\text{pointwise projection}}. \end{aligned} \tag{C.2}$$

The two projected temporal terms are summed before applying $\boldsymbol{A}$; applying $\boldsymbol{A}$ to them separately would require two pointwise projections. For $P > 0$,

$$\begin{aligned} c_{\text{TLR}} < c_{\text{3D}} &\iff C_{\text{out}}d + 2rd + C_{\text{out}}r < 3C_{\text{out}}d \\ &\iff r(2d + C_{\text{out}}) < 2C_{\text{out}}d. \end{aligned} \tag{C.3}$$

Thus, TLRConv2D uses fewer convolutional MACs when

$$r < \frac{2C_{\text{out}}d}{2d + C_{\text{out}}}, \tag{C.4}$$

and its arithmetic ratio is

$$\frac{c_{\mathrm{TLR}}}{c_{\mathrm{3D}}} = \frac{1}{3} + \frac{2r}{3C_{\mathrm{out}}} + \frac{r}{3d}. \tag{C.5}$$

The spatial cost remains fixed, while the temporal cost grows with rank. These expressions quantify convolutional arithmetic; measured GPU latency also depends on execution overhead and memory traffic.

C.2 Folding Equivalence and History Indexing

Collecting current-frame terms. Expanding Equation (6) gives

$$\begin{aligned}\widehat{\boldsymbol{Y}}_t = \boldsymbol{S}\boldsymbol{X}_t + \boldsymbol{b}\mathbf{1}_P^\top + \boldsymbol{A}\boldsymbol{B}_1\boldsymbol{X}_{t-1} + \boldsymbol{A}\boldsymbol{B}_2\boldsymbol{X}_{t-2} \\ - \boldsymbol{A}\boldsymbol{B}_1\boldsymbol{X}_t - \boldsymbol{A}\boldsymbol{B}_2\boldsymbol{X}_t.\end{aligned} \tag{C.6}$$

Collecting the terms involving $\boldsymbol{X}_t$ yields

$$\begin{aligned}\widehat{\boldsymbol{Y}}_t = [\boldsymbol{S} - \boldsymbol{A}(\boldsymbol{B}_1 + \boldsymbol{B}_2)]\boldsymbol{X}_t + \boldsymbol{b}\mathbf{1}_P^\top \\ + \boldsymbol{A}(\boldsymbol{B}_1\boldsymbol{X}_{t-1} + \boldsymbol{B}_2\boldsymbol{X}_{t-2}).\end{aligned} \tag{C.7}$$

The bracketed term is $\boldsymbol{S}_{\mathrm{eff}}$. Because $\boldsymbol{A}$ is pointwise and all factors are linear, each product $\boldsymbol{A}\boldsymbol{B}_j$ contracts only the intermediate channel dimension and does not enlarge the spatial kernel. Folding is performed after recovery and introduces no additional approximation or rank truncation.

Shared-input joint convolution. Equation (11) stacks $\boldsymbol{S}_{\mathrm{eff}}$, $\boldsymbol{B}_1$, and $\boldsymbol{B}_2$ along the output-channel dimension. The kernels share the same spatial stride, dilation, and padding; their output widths are C_{out}, r, and r, respectively, and only the first output group receives the source bias. The joint Conv2D and pointwise history projection cost $(C_{\mathrm{out}} + 2r)dP$ and $C_{\mathrm{out}}rP$ MACs, respectively, whose sum equals c_{TLR}.

History indexing. For an incoming frame at time s, $\mathbf{Q}_{j,s} = \boldsymbol{B}_j\boldsymbol{X}_s$ records both the projection kernel j and the source time s. Therefore,

$$\boldsymbol{B}_1\boldsymbol{X}_{t-1} = \mathbf{Q}_{1,t-1}, \qquad \boldsymbol{B}_2\boldsymbol{X}_{t-2} = \mathbf{Q}_{2,t-2}. \tag{C.8}$$

Away from the sequence boundary, the current projections are used only by subsequent outputs because their negative current-frame contribution is already folded into $\boldsymbol{S}_{\mathrm{eff}}$:

$$\begin{aligned}\widehat{\boldsymbol{Y}}_{t+1} = \mathbf{R}_{t+1} + \boldsymbol{A}(\mathbf{Q}_{1,t} + \mathbf{Q}_{2,t-1}), \\ \widehat{\boldsymbol{Y}}_{t+2} = \mathbf{R}_{t+2} + \boldsymbol{A}(\mathbf{Q}_{1,t+1} + \mathbf{Q}_{2,t}).\end{aligned} \tag{C.9}$$

Thus, $\mathbf{Q}_{1,t}$ is required one step later and $\mathbf{Q}_{2,t}$ two steps later; $\mathbf{Q}_{2,t-1}$ also remains available for the next output.

At each step, execution proceeds as follows:

1. Compute $\mathbf{R}_t$, $\mathbf{Q}_{1,t}$, and $\mathbf{Q}_{2,t}$ using the joint Conv2D.
2. Read $\mathbf{Q}_{1,t-1}$ and $\mathbf{Q}_{2,t-2}$ according to the source temporal-padding rule.
3. Apply $\boldsymbol{A}$ to their sum and add the result to $\mathbf{R}_t$.
4. Update the cache after all entries required by the current output have been read.

The cache persists across chunks of the same video and is reset between independent videos. At sequence start, zero padding creates zero virtual entries. Under replicated-first-frame padding, projections of the first input frame populate the virtual history before the first output is formed.

Execution efficiency. Folding replaces three shared-input spatial-convolution calls with one joint call while retaining the pointwise history projection and the TLR MAC count. With matching bias placement and temporal boundary handling, it is algebraically equivalent up to floating-point evaluation order. Figure 4 reports the resulting throughput and peak memory.

D BLOCK-PRUNING IMPLEMENTATION

We adapt TinyFusion's recoverability-aware search (Fang et al., 2025) to video autoencoders, using grouped block selection, shared LoRA adaptation, and straight-through Gumbel-Softmax estimation.

Search objective. For decoder search on a video $\mathbf{V}$, the teacher encoder supplies posterior-mean latents $\mathbf{Z} = \mathcal{E}_{\mathrm{T}}(\mathbf{V})$, and a sampled structure produces $\mathbf{Y}_{\mathrm{search}}$. Selection logits and shared LoRA parameters are optimized with frozen base weights using

$$\mathcal{L}_{\mathrm{search}} = \mathbb{E}_{\mathbf{V},\boldsymbol{\xi}}\big[\mathcal{L}_1(\mathbf{Y}_{\mathrm{search}}, \mathbf{V}) + \lambda_{\mathrm{perc}}^{\mathrm{search}} \,\mathrm{LPIPS}(\mathbf{Y}_{\mathrm{search}}, \mathbf{V})\big], \tag{D.1}$$

where $\boldsymbol{\xi}$ denotes structure-sampling noise and $\lambda_{\mathrm{perc}}^{\mathrm{search}}$ weights the search-time perceptual loss.

D.1 SELECTION, ADAPTATION, AND EXPORT

Legal pruning units. For a convolutional residual block, a binary retention gate m_ℓ gives

$$\mathbf{H}_{\ell+1} = \mathcal{P}_\ell(\mathbf{H}_\ell) + m_\ell \mathcal{F}_\ell(\mathbf{H}_\ell), \tag{D.2}$$

where $\mathcal{P}_\ell$ includes any required shortcut projection. Setting $m_\ell = 0$ removes the residual computation while retaining the shortcut path. Compatible Transformer blocks may be replaced by the identity. Resampling layers and network input/output layers remain fixed, preserving feature dimensions at retained-block interfaces.

Grouped selection. For a group of n_g blocks with retention quota k_g, the legal masks are $\mathcal{M}_g = \{\mathbf{m} \in \{0,1\}^{n_g} : \sum_j m_j = k_g\}$. Each candidate mask has a learned selection logit. Straight-through Gumbel-Softmax (Jang et al., 2017) with temperature $\tau > 0$ samples one legal mask in the forward pass and uses relaxed probabilities for the backward pass. All groups are optimized jointly using Equation (D.1).

Adaptation and export. The pretrained base weights remain frozen while selection logits and shared LoRA parameters (Hu et al., 2022) are optimized. After search, each group retains the highest-scoring mask according to the exponential moving average of its logits. LoRA updates are merged into retained weights as

$$\boldsymbol{W}_{\mathrm{merge}} = \boldsymbol{W} + \frac{\alpha_{\mathrm{L}}}{r_{\mathrm{L}}} \boldsymbol{L}\boldsymbol{R}, \tag{D.3}$$

where $\boldsymbol{L}$ and $\boldsymbol{R}$ are the learned LoRA factors, with shapes $m \times r_{\mathrm{L}}$ and $r_{\mathrm{L}} \times n$ for a reshaped weight $\boldsymbol{W} \in \mathbb{R}^{m \times n}$. The symbols r_{L} and α_{L} denote the LoRA rank and scaling coefficient, respectively. Eligible merged weights are then converted using TLR, followed by probe-guided reconstruction recovery and folded export. Block selection preserves the widths of retained representations, while TLR controls temporal capacity through layer-specific ranks. Depth selection and temporal-rank allocation are performed sequentially.

Applicability. Block search applies to compatible convolutional and Transformer blocks in both encoders and decoders. In our experiments, TLR is applied to the Wan convolutional VAEs, whereas MiniMax-H3 uses block pruning only. Decoder reconstruction uses teacher-encoder latents; encoder compression is evaluated in downstream reference-conditioned generation.

D.2 FEATURE VISUALIZATION

Figure D.1 visualizes the shortcut output, full block output, and residual response for two high-similarity blocks from one video. The maps show channel-wise RMS at feature-frame index 8, with a shared scale within each row. These examples illustrate the small residual response of high-similarity blocks. Pruning decisions use aggregate search scores over the calibration data.

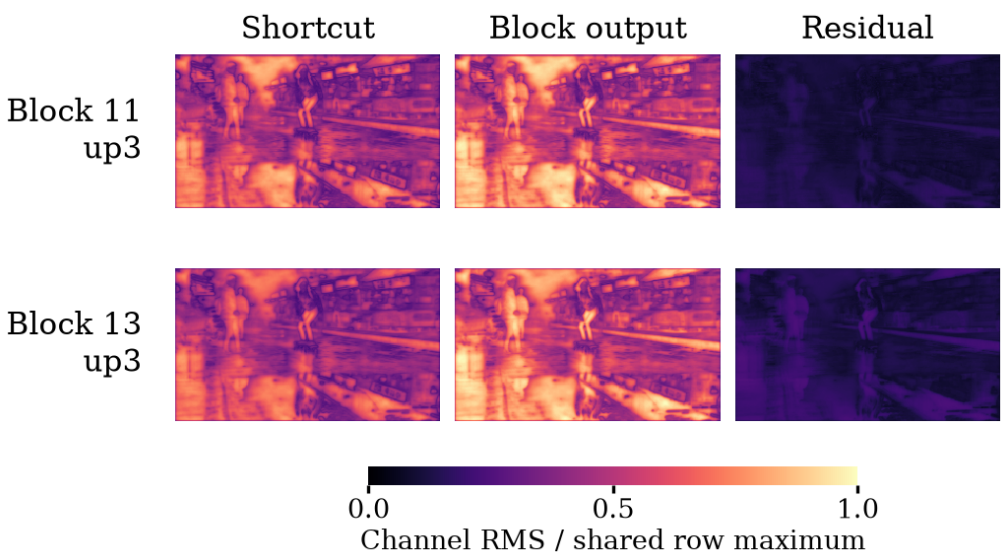


Figure D.1: **Single-video feature example.** Channel-wise RMS of the shortcut, block output, and residual for blocks 11 and 13 at feature-frame index 8; each row uses a shared scale. Blocks are selected using aggregate search scores rather than this video's features.

E Teacher-Suffix Probe and Feature Alignment

During recovery, we cycle through compatible group boundaries and probe one boundary per iteration.

Teacher-suffix probe. For the same latent $\mathbf{Z}$, let $\mathbf{Y}_q = \mathcal{D}_q(\mathbf{Z})$ for $q \in \{\mathrm{T}, \mathrm{S}\}$. At a compatible intermediate boundary, write $\mathcal{D}_q = \mathcal{D}_q^{>} \circ \mathcal{D}_q^{\leq}$ and $\mathbf{H}_q = \mathcal{D}_q^{\leq}(\mathbf{Z})$. The teacher and student features have compatible shapes at this boundary. The frozen teacher suffix probes the student feature as $\mathbf{Y}_{\mathrm{P}} = \mathcal{D}_{\mathrm{T}}^{>}(\mathbf{H}_{\mathrm{S}})$. With $\mathbf{h}_q = \mathrm{vec}(\mathbf{H}_q) \in \mathbb{R}^{N_f}$, define

$$\mathcal{L}_{\mathrm{P}} = \frac{1}{N_y} \left\| \mathrm{vec}\big(\mathbf{Y}_{\mathrm{P}} - \mathrm{sg}(\mathbf{Y}_{\mathrm{T}})\big) \right\|_2^2, \qquad \mathbf{g} = \nabla_{\mathbf{h}_{\mathrm{S}}} \mathcal{L}_{\mathrm{P}}, \qquad \boldsymbol{\delta} = \mathbf{h}_{\mathrm{S}} - \mathrm{sg}(\mathbf{h}_{\mathrm{T}}). \tag{E.1}$$

Here, N_f and N_y are the numbers of feature and output elements, respectively. The suffix parameters are frozen while gradients remain enabled at its input.

Probe-guided feature alignment. Ordinary feature matching moves coordinate i along $-\delta_i$. When $\delta_i g_i < 0$, this coordinate-wise direction locally increases the probe loss. We suppress such directions and weight the remaining feature errors by

$$w_i = \mathrm{clip}_{[0, w_{\max}]} \left(\frac{[\delta_i (N_f g_i / 2)]_+}{\delta_i^2 + \rho} \right), \qquad \mathcal{L}_{\mathrm{feature}} = \frac{1}{N_f} \sum_{i=1}^{N_f} \mathrm{sg}(w_i) \delta_i^2. \tag{E.2}$$

Here, w_i is the nonnegative alignment weight for coordinate i, $[a]_+ = \max(a, 0)$, $\rho > 0$ stabilizes the denominator, and $w_{\max} > 0$ caps the weights. The factor $N_f/2$ accounts for the normalization of the feature-MSE gradient. The weights are detached during student optimization, so no second-order differentiation is required.

Comparison with ordinary feature MSE. The unweighted feature-matching baseline is

$$\mathcal{L}_{\mathrm{feature}}^{\mathrm{MSE}} = \frac{1}{N_f} \sum_{i=1}^{N_f} \delta_i^2. \tag{E.3}$$

Using the reconstruction objective $\mathcal{L}_{\text{recon}}$ defined in Equation (12), the three recovery objectives are

$$\begin{aligned} &\mathcal{L}_{\text{recon}}, \\ &\mathcal{L}_{\text{recon}} + \lambda_{\text{F}} \mathcal{L}^{\text{MSE}}_{\text{feature}}, \\ &\mathcal{L}_{\text{restore}} = \mathcal{L}_{\text{recon}} + \lambda_{\text{F}} \mathcal{L}_{\text{feature}}. \end{aligned} \tag{E.4}$$

The second objective assigns equal weights to feature coordinates; the third uses detached probe-guided weights. All three objectives use the same architecture with 5 of 14 residual main branches retained, initialization, data, recovery budget, and feature boundaries. Both feature objectives use the fixed coefficient $\lambda_{\text{F}} = 0.01$. Under this controlled comparison, Figure 8 shows that uniform feature MSE underperforms reconstruction alone, whereas probe-guided alignment improves recovery. The probe loss is used only to construct detached weights.

Feature-space alignment. At a selected boundary, let $\mathbf{u} = \nabla_{\mathbf{h}_{\text{S}}} \mathcal{L}_{\text{feature}}$. Because the weights are detached during student optimization,

$$u_i = \frac{2}{N_f} w_i \delta_i. \tag{E.5}$$

If $\delta_i g_i \leq 0$, then $w_i = 0$. Otherwise, u_i and g_i have the same sign. Consequently,

$$\mathbf{g}^\top \mathbf{u} = \frac{2}{N_f} \sum_i w_i \delta_i g_i \geq 0. \tag{E.6}$$

Thus, the feature-loss gradient is nonnegatively aligned with the probe-loss gradient in feature space.

Damping and clipping. For a coordinate satisfying $\delta_i g_i > 0$, before upper clipping,

$$u_i = g_i \frac{\delta_i^2}{\delta_i^2 + \rho}. \tag{E.7}$$

Damping attenuates updates for small discrepancies, and clipping limits their magnitude without changing the sign relation. In the zero-damping, unclipped limit, aligned coordinates pass the probe gradient and all other coordinates receive zero weight.

Local feature-space alignment. For the hypothetical feature-space update $\mathbf{h}'_{\text{S}} = \mathbf{h}_{\text{S}} - \epsilon \mathbf{u}$, differentiability gives

$$\mathcal{L}_{\text{P}}(\mathbf{h}'_{\text{S}}) = \mathcal{L}_{\text{P}}(\mathbf{h}_{\text{S}}) - \epsilon \mathbf{g}^\top \mathbf{u} + o(\epsilon). \tag{E.8}$$

Equation (E.6) therefore guarantees a nonpositive first-order change when $\mathbf{h}_{\text{S}}$ is updated directly. For a parameter update with student-prefix Jacobian $\mathbf{J} = \partial \mathbf{h}_{\text{S}} / \partial \boldsymbol{\theta}$, the gradient interaction becomes

$$(\mathbf{J}^\top \mathbf{g})^\top (\mathbf{J}^\top \mathbf{u}) = \mathbf{g}^\top \mathbf{J} \mathbf{J}^\top \mathbf{u}. \tag{E.9}$$

Because $\mathbf{J}\mathbf{J}^\top$ couples feature coordinates, the feature-space guarantee need not hold in parameter space; Figure 8 evaluates the resulting training effect.

F Experimental Details

F.1 Training and Recovery

Search and optimization. Block search follows TinyFusion (Fang et al., 2025), using LoRA (Hu et al., 2022) with $r_{\text{L}} = 16$ and $\alpha_{\text{L}} = 32$. Search and recovery share AdamW (Loshchilov & Hutter, 2019) with $(\beta_1, \beta_2) = (0.9, 0.999)$, zero weight decay, and a per-GPU batch size of one without gradient accumulation. The learning rate warms up to 10^{-5} over 50 steps and then follows cosine decay to 10^{-6}. We clip the gradient norm to 1.0 and use an exponential moving average decay of 0.999.

Recovery loss coefficients. In Equation (12), we use fixed coefficients $\lambda_{\text{perc}} = 0.8$, $\lambda_{\text{adv}} = 0.05$, and $\lambda_{\text{F}} = 0.01$ throughout recovery. Uniform feature MSE and probe-guided alignment use the same feature coefficient and feature boundaries. The coordinate weights w_i are computed from probe gradients and signed feature discrepancies, as defined in Appendix E.

F.2 Compression and Ablation Settings

Model configurations. For the Wan VAEs, LightVAE-Pro retains all 14 residual blocks, while LightVAE-Lite retains one residual main branch per stage ($14 \to 5$). Both use $r = 0$ in the converted residual convolutions of `up2`/`up3`; elsewhere, TLR selects layer-specific ranks using $\eta = 0.6$. MiniMax-H3 uses block pruning only: $36 \to 12$ decoder Transformer blocks and $12 \to 6$ encoder residual main branches. Required shortcuts, projections, and resampling operations are retained.

Operator comparisons. Operator ablations preserve the original Wan2.1 decoder depth and width. Conv2D uses summed temporal kernel slices. The DW3D+PW/Conv2D hybrids follow Flash-VAED's operator layout (Zhu et al., 2026), with random-DW or teacher-SVD initialization, excluding its channel pruning and training recipe. For teacher-SVD initialization, we apply a rank-one SVD to each input channel's pretrained Conv3D kernel, extending the DW+PW decomposition of Guo et al. (2018) to three dimensions. All compressed variants use the same recovery protocol; Figure 4 reports reconstruction quality, throughput, and peak memory.

Spatial–temporal approximation protocols. Figures 5 and 6 use 1,024 and 256 Wan2.1-generated videos, respectively, retaining the original decoder depth and width without recovery. For each converted convolution, $\boldsymbol{S} = \boldsymbol{W}_0 + \boldsymbol{W}_1 + \boldsymbol{W}_2$ acts on current features, while $\boldsymbol{M}_{\text{temp}} = [\boldsymbol{W}_1, \boldsymbol{W}_2]$ acts on historical–current feature differences. The stage-wise comparison modifies one stage's residual convolutions at a time using the same singular-value retention threshold $\eta = 0.6$; other stages remain unchanged. The matched-MAC comparison uses 81-frame 480×832 clips and includes the same 28 residual convolutions in the Wan2.1 decoder. All candidates use identical original-encoder latents. We report the equal-video mean of $\|\hat{y}_i - y_i^{\text{orig}}\|_2 / \|y_i^{\text{orig}}\|_2$, measuring final-output deviation from the original decoder before clamping.

Budget-dependent rank allocation. For matched-MAC comparisons, each compressed branch starts at rank one. Additional ranks are allocated greedily by the gain in retained squared singular-value energy per added MAC, independently for each budget using source weights only. Spatial-only compression retains dense $\boldsymbol{M}_{\text{temp}}$, while TLR retains dense $\boldsymbol{S}$; both configurations allow individual compressed branches to return to dense execution. For joint compression, we divide the required absolute MAC savings equally between the spatial and temporal branches and allocate ranks within each branch budget independently.

MAC limits and curve interpretation. The original decoder costs approximately 137.68 TMAC per video. Fixed operations, dense $\boldsymbol{S}$, and dense $\boldsymbol{M}_{\text{temp}}$ account for 8.86%, 30.38%, and 60.76%, respectively. Counting convolutions and attention matrix products, the rank-one minimum yields MAC floors of 69.86% for spatial-only compression, 39.70% for TLR, and 39.72% for the equal-saving joint baseline. Consequently, spatial-only compression is evaluated only at 70%, 80%, 90%, and 95%; TLR and joint compression additionally cover 40%, 50%, and 60%. The plot uses actual MAC ratios. At a 70% total budget, spatial-only and joint compression retain only 1.26% and 50.63% of the original spatial MACs, respectively, consistent with their errors of 1.1585 and 0.6965. At 80%, spatial-only compression restores its spatial budget to 34.17% while retaining the full temporal branch, and its error drops below joint compression. Under this allocation, TLR yields the lowest error at every evaluated budget. This advantage is consistent with preserving the dense current-frame spatial branch $\boldsymbol{S}$ while concentrating approximation on the temporal correction.

Joint compression. Table 3 evaluates block pruning and TLR separately and jointly on NVIDIA H100 and RTX 5090 using the Wan input settings of Table 1. Pruning retains 5 of 14 residual main branches. Unstarred TLR uses $\eta = 0.6$ in all stages; TLR* sets $r = 0$ in `up2`/`up3` and uses $\eta = 0.6$ elsewhere. The original and Depth pruning + TLR* rows match the original and LightVAE-Lite configurations in Table 1, respectively.

Depth and channel pruning. For the pruning comparison, both variants retain dense Conv3D: block pruning keeps 6 of 14 residual main branches (57.1% removal), while channel pruning removes approximately 41.7% of channels. Their pre-recovery RTX 5090 throughput is 45.58 and 42.69 FPS, respectively, enabling comparison at similar inference cost under the same recovery

Table G.1: Fine-grained VBench-2.0 generation quality (0–100; higher is better). Metrics continue in the lower panel. Dashed lines separate original and compressed decoders. Official Overall uses the benchmark aggregation.

Generation backbone	Decoder	Camera Motion	Complex Landscape	Complex Plot	Composition	Diversity	Dynamic Attribute	Dynamic Spatial Relationship	Human Anatomy	Human Clothes	Human Identity
Wan2.1-1.3B	Original	16.0494	16.8889	10.4722	41.9706	62.6306	13.5531	21.2560	88.7595	94.5701	72.6868
	Flash-VAED	14.1975	17.7778	9.5556	42.3375	56.9775	12.8205	21.7391	86.2864	93.6652	69.4928
	LightVAE-Pro	16.0494	18.6667	8.7778	42.0755	61.7671	13.5531	23.1884	87.8389	94.5701	72.1133
	LightVAE-Lite	15.4321	19.1111	9.0556	41.1321	55.6720	13.9194	20.7729	87.0935	94.0909	71.3987
Wan2.2-5B	Original	10.8025	20.4444	11.3611	46.0901	67.1367	10.9890	25.1208	87.4368	97.6744	73.9860
	Turbo-VAED	11.1111	21.1111	12.9167	46.2683	70.0579	12.0879	25.1208	87.2452	98.1481	73.6227
	LightVAE-Pro	10.8025	20.4444	12.2500	47.6520	67.5512	11.3553	25.1208	87.0102	98.1308	74.1267
	LightVAE-Lite	10.8025	22.0000	11.9167	47.0231	64.4702	10.9890	25.6039	86.9995	98.1395	72.6945
MiniMax-H3	Original	70.9877	26.4444	21.0370	60.8700	30.4177	82.4176	52.6570	95.4118	94.6429	61.8515
	LightVAE-Lite	72.5309	26.4444	21.6204	61.9497	28.9380	82.0513	54.1063	92.6465	93.3333	61.8369

Generation backbone	Decoder	Human Interaction	Instance Preservation	Material	Mechanics	Motion Order Understanding	Motion Rationality	Multi-View Consistency	Thermotics	Official Overall
Wan2.1-1.3B	Original	54.6667	91.8129	28.9157	53.9683	19.5286	31.6092	39.7984	52.5547	52.9867
	Flash-VAED	53.6667	91.8129	30.5882	57.6000	20.5387	31.0345	46.4271	52.5547	52.4985
	LightVAE-Pro	54.0000	91.8129	29.0698	54.0984	18.8552	30.4598	48.0858	52.1739	53.1251
	LightVAE-Lite	53.3333	91.8129	29.4118	57.1429	21.2121	31.6092	41.7962	51.0949	52.2005
Wan2.2-5B	Original	60.0000	91.8129	39.7436	59.1241	19.8653	38.5057	57.3342	58.1560	56.8765
	Turbo-VAED	59.3333	91.8129	38.9610	57.6642	21.2121	40.2299	63.5777	59.2857	57.7332
	LightVAE-Pro	61.3333	91.8129	38.4615	59.5588	20.2703	38.5057	65.3694	57.8571	57.5155
	LightVAE-Lite	60.6667	91.8129	38.4615	58.6957	20.2020	38.5057	62.0777	59.4203	56.9366
MiniMax-H3	Original	81.6667	93.5673	89.6552	78.7234	45.7912	50.5747	63.8827	80.9524	66.8832
	LightVAE-Lite	83.0000	93.5673	89.6552	80.5755	47.1380	54.5977	51.9436	79.3103	66.5546

protocol. The separate feature-loss ablation fixes the architecture at 5 of 14 retained residual main branches (Appendix E).

F.3 Inference Benchmarks

The decoding benchmarks use NVIDIA H100 and GeForce RTX 5090 GPUs with Python 3.12.3, PyTorch `2.11.0+cu130`, CUDA 13.0, and cuDNN 9.19.0. Table 1 uses 81-frame clips at 480×832 for Wan2.1 and 704×1280 for Wan2.2, and 124-frame 768×1344 clips for MiniMax-H3. The joint-compression ablation in Table 3 uses both GPUs and the same Wan input settings as Table 1. These decoding-throughput settings differ from VBench-2.0 generation in Section 5.1, where Wan2.2 uses 121 frames. OOM denotes an out-of-memory failure and is not assigned a numerical throughput or speedup.

G Additional Downstream Evaluation

G.1 Fine-Grained VBench-2.0 Results

Table G.1 reports all 18 VBench-2.0 dimensions for Wan2.1, Wan2.2, and MiniMax-H3 on a 0–100 scale. Videos follow the matched generation protocol in Section 5.1: only the decoder varies within each backbone. On Wan2.1, Pro scores 53.1251 versus 52.9867 for the original decoder, while Lite retains 98.52% of the original overall score. On Wan2.2, Pro and Lite score 57.5155 and 56.9366, respectively, compared with 56.8765 for the original decoder and 57.7332 for Turbo-VAED. On MiniMax-H3, Lite retains 99.51% of the original overall score. The evaluated efficient decoders maintain aggregate generation scores close to the originals despite their reconstruction differences, suggesting limited sensitivity of the overall score under these compression settings. Individual dimensions can still change substantially: H3 Lite improves Motion Rationality but decreases Multi-View Consistency.

G.2 Reference-Conditioned Generation

We evaluate joint encoder–decoder compression in MiniMax-H3 reference-conditioned generation. LightVAE-Lite reduces the encoder from 12 to 6 residual main branches and the decoder from 36 to 12 Transformer blocks, while retaining required shortcuts, projections, and resampling operations. We compare the original encoder and decoder with their compressed counterparts under five reference-input configurations: one, three, six, or nine reference images, and nine images together with one reference video.

Table G.2 reports encoding and decoding latency separately. Encoding latency covers visual-VAE encoding of all reference images and the reference video when present, excluding other reference-conditioning modules. Decoding latency covers VAE decoding of the generated latents. The original and compressed pipelines use identical reference inputs, prompts, random seeds, diffusion backbones, and sampling settings. Output resolution and video duration are fixed across configurations. Speedup is computed as the original latency divided by the corresponding LightVAE-Lite latency.

Reference encoding is accelerated by $2.12\times$–$2.68\times$, and decoding by $2.68\times$–$2.75\times$ in this pipeline. Figure G.1 compares the jointly compressed encoder–decoder pipeline with the original pipeline across four examples. The selected frames remain visually close in overall appearance and scene content, supporting the practical extension to reference encoding.

Table G.2: **Encoding and decoding efficiency for MiniMax-H3 reference-conditioned generation.** Original denotes the uncompressed encoder and decoder, whereas Lite uses their compressed counterparts. All reference images, reference-video frames, and decoded output frames have a spatial resolution of 768×1344. Both the input reference video and the decoded output video contain 124 frames. Latency is measured in seconds on a single NVIDIA H100 GPU. Speedup is computed as the Original latency divided by the corresponding Lite latency.

Reference input	Encoding			Decoding		
	Original (s)↓	Lite (s)↓	Speedup↑	Original (s)↓	Lite (s)↓	Speedup↑
1 image	1.6925	0.6838	2.48×	8.6724	3.1511	2.75×
3 images	5.1713	1.9299	2.68×	8.6762	3.1553	2.75×
6 images	10.3361	4.0597	2.55×	8.6828	3.1932	2.72×
9 images	14.8763	5.9142	2.52×	8.5608	3.1933	2.68×
9 images + 1 video	28.8845	13.6354	2.12×	8.6418	3.1761	2.72×



(a) A fluffy tabby-and-white cat meows, then lowers its head and rests its chin on its front paws beside a sunlit window.

(b) A woman beside a wooden fence smiles and says, "What a peaceful morning," as the camera slowly moves closer.



(c) Small waves wash around rocks and recede across a golden-hour beach as the camera glides forward along the shoreline.



(d) A hummingbird hovers with rapidly beating wings, gently approaches the flowers, and pauses in midair as the camera follows.

Figure G.1: **Qualitative comparison in MiniMax-H3 reference-conditioned generation.** Original (top) uses the original encoder and decoder; LightVAE (bottom) compresses both. Each example shows three frames in temporal order with its prompt summary below. The displayed outputs remain visually close in overall appearance and scene content.