PyramidalWan:
On Making Pretrained Video Model Pyramidal for Efficient Inference

Supplementary Material

This supplementary material is structured as follows. Sec. A1 provides our theoretical contribution: generalization of transition between stages to a broader class of resizing operations. In Sec. A2 we report additional experimental results: we apply recent Jenga [19] method to our Wan-DMD checkpoint and compare its latency with our pyramidal approach. Sec. A3 contains training and evaluation details necessary for reproducing our research. We provide the extended version of Tab. 3 of the main text in Tabs. A2 and A3. We encourage the readers to review the attached videos and compare the outputs produced by different models qualitatively.

A1. Transition between pyramidal stages

In this section we provide a generalization for nearest-neighbour upsampling and average pooling used for transition between stages in PyramidalFlow. While we describe our apporach in terms of wavelets, note that the same derivations are valid for any resizing operation based on orthogonal transforms.

A1.1. Downsampling

Consider a clean video tensor x_0 at the desired output resolution. For the purpose of this section we treat it as a single-channel tensor, flattened in a column vector $\mathrm{x}_0 \in \mathbb{R}^{T\cdot H\cdot W}$. We use a forward diffusion process and denote the noisy version of this video as $\mathrm{x}_\sigma = (1-\sigma)\,\mathrm{x}_0 + \sigma\epsilon$ for some independently sampled $\epsilon \sim \mathcal{N}(0, I)$, *i.e.* $p(\mathrm{x}_\sigma \mid \mathrm{x}_0) = \mathcal{N}\left((1-\sigma)\,\mathrm{x}_0, \sigma^2 I\right)$.

For a single-level orthogonal wavelet decomposition with analysis matrix $\mathcal{W}$ we can decompose x_σ into low- and high-frequency bands $\mathcal{W}\mathrm{x}_\sigma = \left(\hat{\mathrm{x}}_{\sigma,\mathrm{lo}}^T,\, \hat{\mathrm{x}}_{\sigma,\mathrm{hi}}^T\right)^T \in \mathbb{R}^{T\cdot H\cdot W}$, where $\hat{\mathrm{x}}_{\sigma,\mathrm{lo}}^T$ has a dimension of $\frac{T}{2}\cdot\frac{H}{2}\cdot\frac{W}{2}$. To "extract" the low-frequency part, we can use the projector matrix Π_{lo} and write $\hat{\mathrm{x}}_{\sigma,\mathrm{lo}} = \Pi_{\mathrm{lo}}\mathcal{W}\mathrm{x}_\sigma$. Due to the properties of linear operators, the distribution of $p(\hat{\mathrm{x}}_{\sigma,\mathrm{lo}} \mid \mathrm{x}_0)$ is still Gaussian, and we can compute its mean vector and covariance matrix. For covariance we have

$$\begin{aligned}\mathrm{Cov}[\hat{\mathrm{x}}_{\sigma,\mathrm{lo}} \mid \mathrm{x}_0] &= \Pi_{\mathrm{lo}}\mathcal{W}\cdot\sigma^2 I\cdot(\Pi_{\mathrm{lo}}\mathcal{W})^T \\ &= \sigma^2\Pi_{\mathrm{lo}}\mathcal{W}\mathcal{W}^T\Pi_{\mathrm{lo}}^T \\ &= \sigma^2\Pi_{\mathrm{lo}}\Pi_{\mathrm{lo}}^T \\ &= \sigma^2 I\end{aligned}$$

thanks to the orthogonality of matrix $\mathcal{W}$. For the mean,

$$\mathbb{E}[\hat{\mathrm{x}}_{\sigma,\mathrm{lo}} \mid \mathrm{x}_0] = (1-\sigma)\,\Pi_{\mathrm{lo}}\mathcal{W}\mathrm{x}_0 = (1-\sigma)\,U\mathrm{x}_0,$$

where U is a low-frequency part of the matrix $\mathcal{W} = \left(U^T,\, V^T\right)^T$.

In general, row sums for entries of U are not equal to 1. Therefore, if x_0 is a constant vector with all entries being equal to each other, pixel values in $U\mathrm{x}_0$ will differ from those in x_0. To compensate that, for the given wavelet $\mathcal{W}$ we introduce a scaling constant $\omega \in \mathbb{R}_{>0}$ such that $\frac{1}{\omega}U$ preserves the pixel values in this specific case.

We define the downsampling operation $\mathfrak{R}^{\downarrow}(\mathrm{x}_\sigma)$ as

$$\mathfrak{R}^{\downarrow}(\mathrm{x}_\sigma) = \frac{1}{\omega}\Pi_{\mathrm{lo}}\mathcal{W}\mathrm{x}_\sigma = \frac{1}{\omega}\hat{\mathrm{x}}_{\sigma,\mathrm{lo}}. \tag{1}$$

Introduced scaling keeps the range of pixel values after this operation similar to natural signals. As an example, for Haar wavelet downsampling $\mathfrak{R}^{\downarrow}(\cdot)$ is just equal to average pooling.

From the calculations above,

$$\begin{aligned}p\left(\mathfrak{R}^{\downarrow}(\mathrm{x}_\sigma) \mid \mathrm{x}_0\right) &= \mathcal{N}\left((1-\sigma)\cdot\frac{1}{\omega}U\mathrm{x}_0,\, \frac{\sigma^2}{\omega^2}I\right) \\ &= \mathcal{N}\left((1-\sigma)\cdot\mathfrak{R}^{\downarrow}(\mathrm{x}_0),\, \frac{\sigma^2}{\omega^2}I\right),\end{aligned} \tag{2}$$

and, consequently, this distribution can be reparametrized as

$$\mathfrak{R}^{\downarrow}(\mathrm{x}_\sigma) = (1-\sigma)\cdot\mathfrak{R}^{\downarrow}(\mathrm{x}_0) + \frac{\sigma}{\omega}\varepsilon \tag{3}$$

for ε sampled from $\mathcal{N}(0, I)$.

For the noisy signal x_σ at the original resolution, we can calculate the signal-to-noise ratio as

$$\mathrm{SNR}[\mathrm{x}_\sigma \mid \mathrm{x}_0] = \frac{\left\|(1-\sigma)\,\mathrm{x}_0\right\|^2}{\mathbb{E}\left\|\sigma\epsilon\right\|^2} = \left(\frac{1-\sigma}{\sigma}\right)^2\frac{\left\|\mathrm{x}_0\right\|^2}{\mathbb{E}\left\|\epsilon\right\|^2}. \tag{4}$$

For the downsampled noisy signal, from Eq. (3)

$$\begin{aligned}\mathrm{SNR}\left[\mathfrak{R}^{\downarrow}(\mathrm{x}_\sigma) \mid \mathrm{x}_0\right] &= \frac{\left\|(1-\sigma)\,\mathfrak{R}^{\downarrow}(\mathrm{x}_0)\right\|^2}{\mathbb{E}\left\|\frac{\sigma}{\omega}\varepsilon\right\|^2} \\ &= \omega^2\left(\frac{1-\sigma}{\sigma}\right)^2\frac{\left\|\mathfrak{R}^{\downarrow}(\mathrm{x}_0)\right\|^2}{\mathbb{E}\left\|\varepsilon\right\|^2} \\ &\approx \omega^2\cdot\mathrm{SNR}[\mathrm{x}_\sigma \mid \mathrm{x}_0].\end{aligned} \tag{5}$$

Now, if we were to start a new forward diffusion process at the lower resolution, we would parametrize it as

$(1-\tau)\,\mathfrak{R}^{\downarrow}(\mathrm{x}_0)+\tau\eta$ for Gaussian random noise η and noise level τ. To match signal to noise ratio of this process with Eq. (3), we need to solve the equation

$$\left(\frac{1-\tau}{\tau}\right)^2 = \omega^2 \left(\frac{1-\sigma}{\sigma}\right)^2 \tag{6}$$

which results in $\sigma = \frac{\omega\tau}{1+(\omega-1)\tau}$. This coincides with the results obtained in the prior works [2, 8, 10] in context of high-resolution diffusion models in pixel space.

If we denote by i the number of downsampling operations applied one after another, we can rewrite the above equation as the relation between *global noise levels* at scales i and $i+1$, namely,

$$\sigma^{(i)} = \frac{\omega\sigma^{(i+1)}}{1+(\omega-1)\,\sigma^{(i+1)}}. \tag{7}$$

By 'global' we mean that this noise levels are used in the forward diffusion equation. We also will refer to the global noise level at highest resolution $\sigma^{(0)}$ as the *natural noise level* ς. Relation between global level $\sigma^{(i)}$ and natural noise level ς is obtained with recursive application of the above equation.

A1.2. Upsampling

Since we used the wavelet analysis matrix to define downsampling, we can now use the inverse operation to upsample the noisy signal $\mathrm{x}_\sigma \sim p(\mathrm{x}_\sigma \mid \mathrm{x}_0) = \mathcal{N}\big((1-\sigma)\mathrm{x}_0, \sigma^2 I\big)$. For that purpose, we treat x_σ as a low-frequency band and sample high-frequency bands independently from zero-centered Gaussian distribution $\mathcal{N}\big(0, \nu^2 I\big)$. The concatenated vector with all bands is then distributed as $\mathcal{N}\Big(\begin{pmatrix}(1-\sigma)\mathrm{x}_0\\ 0\end{pmatrix}, \begin{pmatrix}\sigma^2 I & 0\\ 0 & \nu^2 I\end{pmatrix}\Big)$. After synthesis matrix $\mathcal{W}^T$ is applied, the resulting vector $\mathrm{x}^{\uparrow}_{\sigma,\nu}$ is distributed as

$$p\big(\mathrm{x}^{\uparrow}_{\sigma,\nu} \mid \mathrm{x}_0\big) = \mathcal{N}\left(\mathcal{W}^T \begin{pmatrix}(1-\sigma)\mathrm{x}_0\\ 0\end{pmatrix}, \mathcal{W}^T \begin{pmatrix}\sigma^2 I & 0\\ 0 & \nu^2 I\end{pmatrix} \mathcal{W}\right).$$

Since $\mathcal{W} = \big(U^T,\ V^T\big)^T$, we can rewrite the mean and covariance as

$$\begin{aligned}\mathbb{E}\big[\mathrm{x}^{\uparrow}_{\sigma,\nu} \mid \mathrm{x}_0\big] &= (1-\sigma)\,\mathcal{W}^T \begin{pmatrix}\mathrm{x}_0\\ 0\end{pmatrix}\\ &= (1-\sigma)\,\big(U^T,\, V^T\big) \begin{pmatrix}\mathrm{x}_0\\ 0\end{pmatrix}\\ &= (1-\sigma)\,U^T\mathrm{x}_0,\end{aligned} \tag{8}$$

$$\begin{aligned}\mathrm{Cov}\big[\mathrm{x}^{\uparrow}_{\sigma,\nu} \mid \mathrm{x}_0\big] &= \mathcal{W}^T \left(\nu^2 I + \begin{pmatrix}(\sigma^2-\nu^2)I & 0\\ 0 & 0\end{pmatrix}\right) \mathcal{W}\\ &= \nu^2 I + \big(U^T,\, V^T\big) \begin{pmatrix}(\sigma^2-\nu^2)I & 0\\ 0 & 0\end{pmatrix} \begin{pmatrix}U\\ V\end{pmatrix}\\ &= \nu^2 I + \big(\sigma^2-\nu^2\big)\,U^T U.\end{aligned} \tag{9}$$

To keep the magnitude of the mean similar to natural signals, we need to scale it by ω – the opposite of what was required for downsampling. We would like to match the obtained distribution with that of the forward diffusion process starting with $\omega U^T\mathrm{x}_0$ and parametrized as $(1-\tau)\cdot\omega U^T\mathrm{x}_0 + \tau\varepsilon$. To achieve this, we additionally multiply the upscaled signal by a non-negative constant r and solve the system of equations w.r.t. r, ν, and τ,

$$1-\tau = (1-\sigma)\,r, \tag{10}$$

$$\tau^2 I = r^2\omega^2\left(\nu^2 I + \big(\sigma^2-\nu^2\big)\,U^T U\right). \tag{11}$$

The last equation can be rewritten as

$$\big(\tau^2 - r^2\omega^2\nu^2\big)\,I = r^2\omega^2\,\big(\sigma^2-\nu^2\big)\,U^T U. \tag{12}$$

Note that U is a 'wide' rectangular matrix, and therefore the rank of RHS is not greater than rank of LHS. Thus, the equality can be fulfilled only if both sides are equal to 0, which leads to $\tau = r\omega\nu$ and $\nu = \sigma$. Consequently,

$$r = \frac{1}{1+(\omega-1)\,\sigma}, \tag{13}$$

$$\tau = \frac{\omega\sigma}{1+(\omega-1)\,\sigma}. \tag{14}$$

We define two new functions of a noisy signal x_σ, upsampling $\mathfrak{R}^{\uparrow}$ and upsampling-and-renoising $\mathfrak{R}^{\uparrow}_{\mathcal{N}}$ as

$$\mathfrak{R}^{\uparrow}(\mathrm{x}_\sigma) = \omega\mathrm{x}^{\uparrow}_{\sigma,0}, \tag{15}$$

$$\mathfrak{R}^{\uparrow}_{\mathcal{N}}(\mathrm{x}_\sigma) = r\omega\mathrm{x}^{\uparrow}_{\sigma,\sigma} = \frac{\omega}{1+(\omega-1)\,\sigma}\mathrm{x}^{\uparrow}_{\sigma,\sigma}. \tag{16}$$

From the derivations above,

$$\mathfrak{R}^{\uparrow}_{\mathcal{N}}(\mathrm{x}_\sigma) = (1-\tau)\,\mathfrak{R}^{\uparrow}_{\mathcal{N}}(\mathrm{x}_0) + \tau\varepsilon.$$

With the scale index i and global noise level notation introduced above, we can reformulate Eq. (17) as

$$\sigma^{(i)} = \frac{\omega\sigma^{(i+1)}}{1+(\omega-1)\,\sigma^{(i+1)}}. \tag{17}$$

Note that this coincides with Eq. (7). This leads to an important observation: both downsampling operation $\mathfrak{R}^{\downarrow}(\mathrm{x}_{\sigma^{(i)}})$ and upsampling $\mathfrak{R}^{\uparrow}_{\mathcal{N}}(\mathrm{x}_{\sigma^{(i)}})$ applied to the noisy sample at scale i do not change the natural noise level ς associated with the global noise level $\sigma^{(i)}$.

A1.3. Downsampling + upsampling

Consider the forward diffusion process at scale $i+1$ that goes from $\mathfrak{R}^{\downarrow}(\mathrm{x}_0)$ to the global noise level $\sigma^{(i+1)}$. Then, after $\mathfrak{R}^{\uparrow}_{\mathcal{N}}$ being applied to this noisy sample, the result has the same marginal distribution as another diffusion process at scale i with noise level $\sigma^{(i)}$ that started with $\mathfrak{R}^{\uparrow}_{\mathcal{N}}\circ\mathfrak{R}^{\downarrow}(\mathrm{x}_0)$. But for the clean signal x_0 at scale i composition of introduced operations $\mathfrak{R}^{\uparrow}_{\mathcal{N}} \circ \mathfrak{R}^{\downarrow}$ is equivalent to wavelet-based low-pass filtering.

This means that if the noise level $\sigma^{(i)}$ is high enough to turn the high-frequency part of $\mathrm{x}_{\sigma^{(i)}}$ into i.i.d. noise, then with this (or larger) amount of noise the discrepancy between processes starting with x_0 and $\mathfrak{R}^{\uparrow}_{\mathcal{N}} \circ \mathfrak{R}^{\downarrow}(\mathrm{x}_0)$ is negligible. Therefore, part of the backward diffusion process can run at lower resolution until the level $\sigma^{(i+1)}$, followed by upsampling $\mathfrak{R}^{\uparrow}_{\mathcal{N}}$ and backward process starting from $\sigma^{(i)}$.

In practice, noise levels suitable for switching between resolutions are calculated based on the average spectrum of the dataset, and therefore it is not guaranteed that for *any* x_0 the noise level $\sigma^{(i)}$ is high enough to destroy its high-frequency content. Thus, it is more convenient to train a denoising model on inputs sampled from the downsampling-upsampling process rather than the original one.

A2. Additional experiments

Both PyramidalFlow and PPF can be viewed as special cases of token reduction strategies during inference. Token reduction techniques are well established in the community, with several recent methods proposed, such as ToMe [1], ADAPTOR [15], RLT [4], and vid-TLDR [3]. However, these approaches typically rely on heuristics based on mutual token similarity. Consequently, their speed-up procedure is *dynamic*, depending on factors such as prompt complexity and initial latent noise. In contrast, pyramidal approaches can be represented as *static* computational graphs (one per stage), which greatly simplifies deployment, particularly in resource-constrained environments [11].

Despite this fundamental difference, we include a comparison with the recent Jenga method [19]. Jenga employs a dynamic sparse attention mask based on token similarity and leverages space-filling curves for GPU-friendly implementation. We apply Jenga-Base (without progressive resolution) and Jenga-Turbo (with half of the steps performed on downsampled latents) on top of our Wan-DMD checkpoint and measure performance and latency using 2 sampling steps. For Jenga-Turbo, we tried downsampling factors of 0.75 and 0.5. Note that the original Jenga implementation does not apply downsampling along the temporal dimension, even when using progressive resolution. We found that adding it to spatial downsampling indeed leads to noticeable drop of quality. Our default compiler settings (see Sec. A3) did not work for Jenga. For that reasons, we adjusted the settings to `fullgraph=False` and `mode='max-autotune-no-cudagraphs'`. Results are reported in Tab. A1. For details on *AttenCarve* hyperparameters, please refer to the work of Zhang et al. [19]. TeaCache [12] was not used in these experiments.

While Jenga obtains good VBench and VBench-2.0 scores, we found that videos produced by these models often suffer from abrupt scene changes, incoherent motion, and 'episodic' behavior. These artifacts are more pronounced than in case of original Wan-DMD model or our pyramidal modifications. Please, refer to the supplementary videos for comparison. Overall, our experiments demonstrate that pyramidization results in better quality-efficiency trade-off.

A3. Additional details

In this section we provide training and evaluation details useful for reproducing our experiments.

Handling spatiotemporal resolution. In the default setting, Wan model operates with 21 latent frame at stage $i = 0$. This requires special treatment for upsampling and downsampling along the temporal axis. For our final experiments on PyramidalWan, we opted for separate handling of the first frame. Namely, during the forward diffusion process its global noise level σ_{first} is different from the global noise level of the rest of the frames σ_{rest}, but both of them correspond to the same natural level ς. This is due to the fact that relation between global and natural levels depends on the scaling factor ω (see Eq. (17)), which differs for 2D and 3D cases. Similarly, during the upscaling operation $\mathfrak{R}^{\uparrow}_{\mathcal{N}}$ first frame is upsampled only spatially, while the rest of the frames — spatiotemporally, both according to Eq. (16). The derivative of the noised signal w.r.t. the global level is computed separately for the first and the other frames. Since RoPE positional embeddings [9] used in Wan could potentially make it challenging for the network to handle this special nature of the first frame, we added a special learnable embedding vector to all the tokens of the first frame after the patchification layer.

For PPF, increased kernel size of the patchification layer also introduces inconsistency with the original shape of the latent video tensor. Therefore, for stages $i = 1$ and $i = 2$ we apply trilinear interpolation to upsample the original tensor to the minimal shape that allows to apply the patchifier. *E.g.*, for stage 2 we resized from $21 \times 60 \times 104$ to $24 \times 64 \times 104$, thus enabling patchification with kernel size $4 \times 8 \times 8$. After the unpatchification layer, the last operation of VideoDiT, trilinear interpolation is applied again to resize the tensor to the original shape. Note that for PPF we treat the first frame in the same way as other frames.

Wan-DMD. For DMD distillation of the original model, fake score network was updated twice per each update of the student model. The learning rate of AdamW optimizer for the student was 1×10^{-5}, and for fake score 5×10^{-6}. Noise levels for the student's inputs were selected uniformly from the set $\{0.0050, 0.7149, 0.9092, 1\}$, corresponding to uniform selection of *timesteps* between 1 and 1000 with the consequent application of *shift* [8] equal to 5. For fake score network, noise levels were selected uniformly, with further application of shift of 5. For teacher's outputs, classifier-free guidance was applied with scale value of 5. We used the negative prompt [7] *"Bright tones, overexposed, static,*

Table A1. **Jenga results.** We measure the total latency of Video DiT calls for Jenga [19] applied to Wan-DMD checkpoint with 2 sampling steps. For comparison, our Wan-PPF-DMD model with 2-2-1 schedule achieves a total transformer latency of 810 ms (standard deviation < 10 ms) at the same output video resolution. Beyond offering a better quality–efficiency trade-off, pyramidal models benefit from a static computational graph, which greatly simplifies deployment.

Model	AttenCarve		ProRes of 1st step		Latency, ms ↓	Total score ↑	
	k	p	factor	temporal downsampling		VBench	VBench-2.0
Jenga-Base	0.05	0.3	1×	✗	$1{,}211 \pm 199$	81.25	54.30
	0.1	0.3	1×	✗	$1{,}278 \pm 205$	83.52	52.84
	0.15	0.3	1×	✗	$1{,}297 \pm 304$	83.54	53.28
	0.15	0.9	1×	✗	$1{,}680 \pm 201$	83.13	57.19
Jenga-Turbo	0.1	0.3	0.75×	✗	$1{,}089 \pm 145$	83.38	53.94
	0.1	0.3	0.5×	✗	932 ± 114	82.98	53.46
	0.1	0.3	0.5×	✓	865 ± 350	81.23	52.30

blurred details, subtitles, style, works, paintings, images, static, overall gray, worst quality, low quality, JPEG compression residue, ugly, incomplete, extra fingers, poorly drawn hands, poorly drawn faces, deformed, disfigured, misshapen limbs, fused fingers, still picture, messy background, three legs, many people in the background, walking backwards".

Wan-Adv. The discriminator head comprises two branches: a *spatial* branch and a *temporal* branch. Both branches share a common processing pipeline: input reshaping, initial convolution, one or more ResNet blocks, SiLU activation, and a final convolution, followed by restoring the original video layout. The spatial branch operates on frame-level structure by flattening the temporal dimension into the batch, $(b, t, c, h, w) \mapsto (b \cdot t, c, h, w)$, and applies 2D convolutions and 2D ResNet blocks to capture intra-frame details. The temporal branch focuses on temporal dynamics by flattening spatial dimensions, $(b, t, c, h, w) \mapsto (b \cdot h \cdot w, c, t, 1, 1)$, and uses 3D convolutions with kernel size $3 \times 1 \times 1$ and temporal ResNet blocks to model inter-frame relationships. Weights are initialized with Xavier normal. The final convolution is zero-initialized for more stable adversarial training. For adversarial distillation, the discriminator head (with the backbone feature extractor kept frozen) is updated four times more frequently than the generator (student model). We use the AdamW optimizer with a learning rate of 1×10^{-4} for the discriminator head and 1×10^{-5} for the student model. Noise levels for student model distillation are sampled uniformly from the set $\{0.25, 0.5, 0.75, 1\}$.

PyramidalWan. To split the natural noise scale between stages, we conducted the spectral analysis of latents produced by the encoder of WanVAE. Our analysis follows the same procedure as that described, among others, by Dieleman [5], Starodubcev et al. [17]. Based on the results we set the 'cleaner' natural leves as follows: $\varsigma_c^{(1)} = 0.5858$, $\varsigma_c^{(2)} = 0.9412$.

For training of our pyramidal flow matching model, for each stage i we sampled the natural noise level uniformly,

$$u \sim \mathrm{Uni}(0, 1), \tag{18}$$

$$\varsigma^{(i)} = \varsigma_c^{(i)} + u \cdot \left(\varsigma_n^{(i)} - \varsigma_c^{(i)}\right), \tag{19}$$

where natural levels $\varsigma_c^{(i)}$ and $\varsigma_n^{(i)}$ correspond to the global boundary levels $\sigma_c^{(i)}$ and $\sigma_n^{(i)}$ with 3D scaling factor ω, see Eq. (17).

For DMD-PT pipeline, student's u (see Eq. (19)) was selected uniformly from the set $\{0.25, 0.5, 0.75, 1\}$, while for the fake score it was sampled according to Eq. (18). In DMD-OT training, we sampled u w.r.t. Eq. (18) and applied shift 5 afterwards. For the fake score (since it is initialized with the original Wan model), natural noise level ς' was sampled from $\mathrm{Uni}(0, 1)$, and shift 5 was applied after that.

PyramidalWan has been trained with learning rate of 1×10^{-5}. For PyramidalWan-DMD, optimizers had the same hyperparameters as for Wan-DMD. To mix videos of different spatiotemporal resolution in the same batch, we flattened all the tokens into a single 1D sequence and applied sparse self-attention and cross-attention masks using FlexAttention [6].

For adversarial distillation of PyramidalWan (Adv-OD and Adv-PD), we follow the same hyperparameters as Wan-Adv. The only difference is that the noise levels for the student model are adjusted at each stage according to Eq. (19).

Wan-PPF. For PPF models, we used the same hyperparameters as for the training of the full Wan models. Noise levels were split between stages in the same way as for PyramidalWan.

Sampling from trained models. For all the models, both multi-step and few-step, we used `FlowMatchEulerDiscreteScheduler` from the diffusers library for sampling [18, v0.33.0]. For Wan-DMD, noise levels for 4-step sampling were taken from the set $\{1, 0.9097, 0.7173, 0.0244\}$, and for 2-step

sampling from $\{1, 0.8347\}$. For PyramidalWan-DMD with 2-2-1 schedule, natural noise levels were selected as follows: $\varsigma^{(2)} \in \{1, 0.9863\}$, $\varsigma^{(1)} \in \{0.9412, 0.8645\}$, $\varsigma^{(0)} \in \{0.5858\}$. Same schedule was used for Wan-PPF-DMD. For the multi-step flow matching models, we used classifier-free guidance scale of 5.

Measurements. To estimate the computational cost of various models, we calculated FLOPs using DeepSpeed library [16, v0.14.2]. For latency measurements, models were compiled separately for each resolution with the PyTorch compiler [14, v2.7.0] on H100 GPU. The configuration of the compiler was set as follows: `dynamic=False`, `fullgraph=True`, `mode="max-autotune"`. For VBench and VBench-2.0 evaluation of all models mentioned in the paper, generated videos were saved using TorchVision's [13, v0.22.0] `torchvision.io.write_video` function with default parameters. We used *GPT*[1] set of extended prompts to evaluate VBench and *Wanx*[2] set for VBench-2.0.

[1] https://github.com/Vchitect/VBench/blob/7bcb8691c49426ac30544456d19a234d971722e6/prompts/augmented_prompts/gpt_enhanced_prompts/all_dimension_longer.txt

[2] https://github.com/Vchitect/VBench/blob/8270c9e54eb56de9a589ec351f4ff3c4e0ab3dfd/VBench-2.0/prompts/prompt_aug/Wanx_full_text_aug.txt

Table A2. **VBench scores.**

Models	Subject Consistency	Background Consistency	Temporal Flickering	Motion Smoothness	Dynamic Degree	Aesthetic Quality	Imaging Quality	Object Class	Multiple Objects	Human Action
Wan2.1-1.3B (50 steps)	93.07	95.21	99.35	98.03	69.17	65.20	65.07	88.78	72.07	96.40
Wan2.1-1.3B (25 steps)	92.06	95.43	99.30	97.13	69.44	62.91	62.28	85.89	66.40	96.99
PyramidalWan (20-20-10)	97.02	97.38	99.44	98.68	44.44	66.13	65.69	95.27	84.47	95.60
Wan-Adv (4 steps)	95.62	95.44	98.47	98.87	71.39	63.37	65.79	91.33	71.91	95.00
Wan-Adv (2 steps)	95.97	95.31	98.58	98.93	64.17	63.68	66.24	90.35	69.91	93.80
Wan-Adv (1 step)	95.04	94.71	98.59	98.80	39.44	63.23	66.08	88.26	68.61	92.60
Wan-DMD (4 steps)	94.57	94.34	97.91	97.78	85.83	66.66	67.43	92.94	77.76	96.20
Wan-DMD (2 steps)	95.81	94.92	98.27	98.28	64.44	66.87	68.42	95.11	84.45	97.40
Wan-DMD (1 step)	92.95	92.94	98.55	98.44	46.94	59.96	66.89	84.43	64.65	95.60
PyramidalWan-Adv-OD (2-2-1)	97.63	97.17	98.96	98.49	45.83	66.48	69.94	94.21	79.80	94.40
PyramidalWan-Adv-PD (2-2-1)	96.39	96.71	99.23	98.79	56.94	65.08	67.82	93.94	72.84	91.80
Wan-PPF-DMD (2-2-1)	96.18	94.34	98.37	98.80	49.72	65.30	69.35	94.95	85.40	94.60
PyramidalWan-DMD-OT (2-2-1)	97.56	96.74	97.66	98.04	43.89	69.96	71.14	95.97	84.24	95.00
PyramidalWan-DMD-PT* (2-2-1)	98.06	96.98	99.34	99.07	31.39	68.07	69.24	95.44	85.18	94.20

Models	Color	Spatial Relationship	Scene	Appearance Style	Temporal Style	Overall Consistency	Quality Score	Semantic Score	Total Score
Wan2.1-1.3B (50 steps)	83.20	75.46	54.56	22.82	25.78	26.99	83.47	78.57	82.49
Wan2.1-1.3B (25 steps)	83.11	67.94	49.64	23.69	24.90	26.28	82.09	76.02	80.87
PyramidalWan (20-20-10)	88.51	77.37	55.12	21.96	24.92	26.48	83.36	80.70	82.83
Wan-Adv (4 steps)	84.70	75.78	51.21	22.01	24.22	26.19	84.06	77.39	82.72
Wan-Adv (2 steps)	86.22	74.16	51.80	21.71	24.15	26.08	83.74	76.82	82.35
Wan-Adv (1 step)	84.74	75.98	50.51	21.30	23.85	25.86	81.38	75.85	80.28
Wan-DMD (4 steps)	80.93	71.82	51.90	21.60	25.24	26.59	84.71	77.86	83.34
Wan-DMD (2 steps)	86.04	78.51	51.95	21.65	25.32	26.82	84.00	80.41	83.28
Wan-DMD (1 step)	84.99	68.52	47.51	22.15	24.60	26.07	80.63	74.75	79.45
PyramidalWan-Adv-OD (2-2-1)	85.82	80.77	51.73	20.96	24.34	25.66	83.94	78.74	82.90
PyramidalWan-Adv-PD (2-2-1)	80.99	70.83	54.17	20.10	24.19	26.05	84.20	76.07	82.57
Wan-PPF-DMD (2-2-1)	88.90	79.48	51.85	20.46	24.88	26.17	83.04	79.80	82.39
PyramidalWan-DMD-OT (2-2-1)	83.96	82.18	50.77	22.22	24.09	25.90	83.63	79.80	82.86
PyramidalWan-DMD-PT* (2-2-1)	81.26	82.14	52.54	21.35	24.93	26.35	83.46	79.75	82.72

Table A3. **VBench-2.0 scores.**

Models	Camera Motion	Complex Landscape	Complex Plot	Composition	Diversity	Dynamic Attribute	Dynamic Spatial Relationship	Human Anatomy	Human Clothes	Human Identity	Human Interaction	Instance Preservation
Wan2.1-1.3B (50 steps)	49.08	72.44	31.99	61.70	48.42	40.80	97.99	11.31	25.12	85.96	63.50	16.44
Wan2.1-1.3B (25 steps)	96.10	47.62	10.91	73.13	62.90	25.12	19.11	83.04	4.02	79.33	48.65	71.33
PyramidalWan (20-20-10)	99.06	18.68	12.85	57.26	74.39	31.88	17.78	95.91	37.40	82.71	44.79	72.00
Wan-Adv (4 steps)	49.07	14.44	9.54	50.34	51.59	38.46	27.54	88.14	96.73	71.88	67.67	83.63
Wan-Adv (2 steps)	44.14	17.11	9.28	48.75	45.39	39.56	26.57	87.12	95.85	65.18	60.67	82.46
Wan-Adv (1 step)	40.43	17.56	9.65	45.51	31.51	41.39	27.05	86.07	85.52	67.20	59.00	78.36
Wan-DMD (4 steps)	99.48	51.28	10.92	76.34	60.84	35.27	16.67	79.53	24.51	88.08	48.53	71.67
Wan-DMD (2 steps)	37.96	20.22	13.58	48.70	45.30	43.96	31.40	87.25	100.00	54.94	75.00	85.96
Wan-DMD (1 step)	89.81	54.21	10.23	74.22	73.15	30.92	16.89	80.12	2.21	69.53	47.13	62.00
PyramidalWan-Adv-OD (2-2-1)	23.77	16.22	7.87	39.79	49.82	13.92	24.15	61.12	100.00	84.09	62.00	93.57
PyramidalWan-Adv-PD (2-2-1)	28.40	15.56	9.20	42.99	47.02	21.61	28.99	88.56	98.60	75.78	59.33	89.47
Wan-PPF-DMD (2-2-1)	31.79	16.67	10.36	48.62	31.46	12.45	24.64	88.23	100.00	86.61	66.67	89.47
PyramidalWan-DMD-OT (2-2-1)	24.69	15.56	9.00	45.56	54.52	25.64	30.92	45.52	100.00	86.98	73.67	91.23
PyramidalWan-DMD-PT* (2-2-1)	22.53	13.56	10.39	42.81	26.80	21.98	29.95	63.55	100.00	80.60	75.00	95.32

Models	Material	Mechanics	Motion Order Understanding	Motion Rationality	Multi-View Consistency	Thermotics	Creativity Score	Commonsense Score	Controllability Score	Human Fidelity Score	Physics Score	Total Score
Wan2.1-1.3B (50 steps)	9.62	80.63	71.67	69.44	49.05	32.10	48.73	63.38	33.96	80.71	53.30	56.02
Wan2.1-1.3B (25 steps)	41.95	67.57	50.34	36.70	34.26	64.08	49.49	62.50	35.01	79.44	52.20	55.73
PyramidalWan (20-20-10)	32.76	59.22	44.50	23.91	21.60	53.68	44.64	64.33	28.39	85.38	51.89	54.93
Wan-Adv (4 steps)	64.66	75.74	21.21	31.03	0.00	61.87	50.96	57.33	32.56	85.58	50.57	55.40
Wan-Adv (2 steps)	62.07	73.08	23.23	34.48	18.90	63.38	47.07	58.47	31.51	82.72	54.36	54.82
Wan-Adv (1 step)	63.16	73.88	14.81	30.46	0.00	60.14	38.51	54.41	29.99	79.60	49.29	50.36
Wan-DMD (4 steps)	36.78	71.30	53.20	32.32	36.42	64.75	50.87	58.16	36.36	82.80	59.23	57.48
Wan-DMD (2 steps)	68.50	72.99	37.37	39.66	12.58	68.79	47.00	62.81	37.07	80.73	55.72	56.67
Wan-DMD (1 step)	38.51	72.95	30.21	30.98	42.28	70.63	38.67	59.31	35.36	77.49	55.00	53.17
PyramidalWan-Adv-OD (2-2-1)	60.61	58.33	12.12	28.16	36.12	49.64	44.80	60.86	22.86	81.74	51.17	52.29
PyramidalWan-Adv-PD (2-2-1)	66.67	69.53	17.85	34.48	9.79	58.04	45.00	61.98	25.85	87.65	51.01	54.30
Wan-PPF-DMD (2-2-1)	59.62	61.90	10.77	36.21	12.78	57.64	40.04	62.84	24.76	91.61	47.98	53.45
PyramidalWan-DMD-OT (2-2-1)	64.49	62.32	23.91	35.63	39.33	60.99	50.04	63.43	29.05	77.50	56.78	55.36
PyramidalWan-DMD-PT* (2-2-1)	62.07	62.99	25.93	31.03	17.67	60.96	34.81	63.18	28.48	81.38	50.92	51.75

References

[1] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. *CVPRW*, 2023. A3

[2] Ting Chen. On the Importance of Noise Scheduling for Diffusion Models, 2023. A2

[3] Joonmyung Choi, Sanghyeok Lee, Jaewon Chu, Minhyuk Choi, and Hyunwoo J. Kim. vid-TLDR: Training free token merging for light-weight video transformer. In *CVPR*, 2024. A3

[4] Rohan Choudhury, Guanglei Zhu, Sihan Liu, Koichiro Niinuma, Kris M. Kitani, and Laszlo Attila Jeni. Don't look twice: Faster video transformers with run-length tokenization. In *NeurIPS*, 2024. A3

[5] Sander Dieleman. Diffusion is spectral autoregression, 2024. A4

[6] Juechu Dong, BOYUAN FENG, Driss Guessous, Yanbo Liang, and Horace He. Flexattention: A programming model for generating fused attention variants. In *MLSys*, 2025. A4

[7] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. In *NeurIPS*, 2020. A3

[8] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. A2, A3

[9] Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary Position Embedding for Vision Transformer. In *ECCV*, Cham, 2025. A3

[10] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *ICML*, 2023. A2

[11] Animesh Karnewar, Denis Korzhenkov, Ioannis Lelekas, Noor Fathima, Adil Karjauv, Mohsen Ghafoorian, and Amir Habibian. Neodragon: Mobile video generation using diffusion transformer. In *ICLR*, 2026. A3

[12] 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 *CVPR*, 2025. A3

[13] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016. A5

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. A5

[15] Elia Peruzzo, Adil Karjauv, Nicu Sebe, Amir Ghodrati, and Amir Habibian. Adaptor: Adaptive token reduction for video diffusion transformers. In *CVPRW*, 2025. A3

[16] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *SIGKDD*, 2020. A5

[17] Nikita Starodubcev, Ilya Drobyshevskiy, Denis Kuznedelev, Artem Babenko, and Dmitry Baranchuk. Scale-wise distillation of diffusion models. In *ICLR*, 2026. A4

[18] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022. A4

[19] Yuechen Zhang, Jinbo Xing, Bin Xia, Shaoteng Liu, Bohao Peng, Xin Tao, Pengfei Wan, Eric Lo, and Jiaya Jia. Training-free efficient video generation via dynamic token carving. In *NeurIPS*, 2025. A1, A3, A4