

PyramidalWan: On Making Pretrained Video Model Pyramidal for Efficient Inference

Denis Korzhenkov* Adil Karjauv* Animesh Karnewar Mohsen Ghafoorian Amirhossein Habibian
 Qualcomm AI Research[†]

{dkorzhen, akarjauv, karnewar, mghafoor, ahabibia}@qti.qualcomm.com

Abstract

Recently proposed pyramidal models decompose the conventional forward and backward diffusion processes into multiple stages operating at varying resolutions. These models handle inputs with higher noise levels at lower resolutions, while less noisy inputs are processed at higher resolutions. This hierarchical approach significantly reduces the computational cost of inference in multi-step denoising models. However, existing open-source pyramidal video models have been trained from scratch and tend to underperform compared to state-of-the-art systems in terms of visual plausibility. In this work, we present a pipeline that converts a pretrained diffusion model into a pyramidal one through low-cost finetuning, achieving this transformation without degradation in quality of output videos. Furthermore, we investigate and compare various strategies for step distillation within pyramidal models, aiming to further enhance the inference efficiency. Our results are available at <https://qualcomm-ai-research.github.io/PyramidalWan>.

1. Introduction

Recent video diffusion models have achieved remarkable generative quality [12, 20, 27, 39]. However, these impressive capabilities come at a cost: multi-step inference remains computationally expensive. The principal strategies for reducing inference overhead are step distillation and architectural optimization [6, 11, 25, 42, 43].

Beyond these approaches, several recent works have proposed training diffusion models that process inputs with different noise levels at different resolutions [4, 7, 34, 35, 44]. This method is motivated by the observation known as *spectral autoregression*: in the spectral decomposition of natural signals, higher-frequency components tend to have lower

Table 1. **Computational costs.** Schedule is the number of steps per each of three stages, from the lowest spatiotemporal resolution to the highest. For diffusion models, the cost is doubled due to the usage of classifier-free guidance.

Inference method	Schedule	TFLOPs ↓
Diffusion	0-0-50	$2 \times 12,592$
Pyramidal diffusion	20-20-10	$2 \times 2,821$
Step distillation	0-0-4	1,007
	0-0-2	504
Pyramidal step distillation	2-2-2	534
	2-2-1	282
	1-1-1	267

Table 2. **Latency of a single denoiser forward pass.** The savings obtained due to the reduced number of tokens at stages 1 and 2 lead to 43% speedup for 2-2-1 schedule in comparison with 0-0-2 while being only 13% slower than 0-0-1. Video DiT was compiled for each stage-wise resolution separately for this measurement.

Method	Stage	Latency, ms ↓
PyramidalWan $81 \times 448 \times 832$	0 (hi-res)	631.77
	1 (mid-res)	33.76
	2 (lo-res)	7.62
Wan-PPF $81 \times 480 \times 832$	0 (hi-res)	713.76
	1 (mid-res)	39.85
	2 (lo-res)	8.26

magnitudes and thus are eliminated earlier during the forward diffusion process [8, 9, 30]. This insight can be exploited to make generation more efficient. In the beginning, the generator starts with pure low-resolution Gaussian noise, synthesizes (still noisy) coarse structure at the same resolution, and then progressively increases the resolution simultaneously with further denoising. Jin et al. [18] proposed the formalization of this approach in a form of *PyramidalFlow* framework. However, they only demonstrated training of such a model from scratch under limited computational resources. In our work, we show that pretrained state-of-the-art diffusion models can be made pyramidal via low-cost finetuning without loss in visual quality.

In detail, we begin with the pretrained Wan2.1-1.3B model [36] and decompose its forward and backward dif-

*Equal contribution

[†]Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

fusion processes into three spatiotemporal stages, operating at resolutions of $81 \times 448 \times 832$, $41 \times 224 \times 416$, and $21 \times 112 \times 208$, respectively, see Fig. 1. We finetune the model using the pyramidal flow matching loss [18], demonstrating that this approach substantially reduces inference cost while maintaining near-original quality. Furthermore, we conduct a study of various step distillation strategies within the pyramidal setup, both for conventional and pyramidal teacher diffusion models. We also demonstrate for the first time that recently proposed Pyramidal Patchification models (an alternative to PyramidalFlow) [21] can be successfully trained for few-step video generation.

In addition to this empirical study, we present a theoretical generalization of the resolution transition operations introduced in PyramidalFlow. Specifically, we extend these operations to arbitrary upsampling and downsampling functions based on orthogonal transforms. Notably, average pooling and nearest-neighbor upsampling, employed in the original work, can be interpreted as scaled instances of the Haar wavelet operator, fitting within our generalized framework.

In summary, our contributions are as follows:

1. We show that a conventional video diffusion transformer can be effectively converted into a spatiotemporal pyramidal diffusion model with minimal finetuning cost and without compromising quality.
2. We conduct a systematic study of step distillation techniques within the pyramidal setup, offering practical insights for various training scenarios.
3. We extend the procedure of transition between stages in the PyramidalFlow framework to a broader class of upsampling functions.

2. Related works

Pyramidal models. The observation that Gaussian noise degrades information across all frequency components at a uniform rate has been well established in the literature [8, 9]. Simultaneously, natural signals such as images and videos are known to exhibit relatively low magnitudes in their high-frequency components [14, 15]. Together, these insights motivate a multi-resolution denoising strategy. An early implementation of this idea was the cascaded diffusion model, which employed multiple denoising networks, each operating at a distinct resolution [13]. Subsequent works aimed to unify this approach within a single network, introducing different mechanisms for transitioning between resolutions [4, 7, 44]. PyramidalFlow [18] proposed a mathematically grounded framework based on flow matching, offering a coherent view of the forward diffusion process and spatial resolution changes. Building on this, TPDiff extended the methodology to the video frame rate varying per stage [29]. The practical viability of pyramidal

models has also been demonstrated through deployment on resource-constrained devices such as mobile chips [19]. In our work, we adopt the PyramidalFlow framework to convert a pretrained video model into a pyramidal pipeline, enabling efficient inference with relatively cheap training.

Patch-pyramidal models. An alternative to modifying input resolution is adjusting the kernel size of the patchification and unpatchification layers in the diffusion transformer (DiT) based on the noise level. This allows the most computationally intensive transformer blocks to operate on fewer tokens for noisier inputs, achieving similar efficiency gains to PyramidalFlow. An advantage of this approach is that it avoids the need for mathematical derivations to handle stage transitions. FlexiDiT [2] combines this strategy with learnable per-stage LoRA adapters [16], which may pose challenges for inference on resource-constrained devices. More recently, Pyramidal Patchification Flow (PPF) [21] demonstrated that such adapters are not essential, neither for training from scratch nor for finetuning pretrained text-to-image diffusion models. In our experiments, we find that under limited training budgets, PyramidalFlow outperforms PPF for diffusion-style finetuning. Nevertheless, we show that patch-pyramidal models remain a strong candidate for distillation into few-step inference models.

Step distillation of diffusion models. Since multi-step inference in diffusion models is computationally intensive for many applications, step distillation has become a key area of research. The most widely adopted methods include adversarial distillation [32], distribution matching distillation (DMD) [28, 40, 41], and consistency models [5, 10, 31]. While most efforts in this field have focused on image generation, several recent papers have extended these techniques to video models [3, 25, 37, 47]. In our work, we explore DMD and adversarial approaches to distill a few-step pyramidal version of a pretrained video diffusion model, whether originally pyramidal or not. Notably, the concurrent SwD [33] and Neodragon [19] works also studied pyramidal step distillation to reduce inference costs. However, SwD did not consider the case of a pyramidal teacher model, while Neodragon has not explored PPF-based training. Our work fills these gaps.

3. Preliminaries

To reduce the computational cost of video generation inference, we adopt the PyramidalFlow framework [18], splitting the forward and backward diffusion processes into S stages indexed by i . Each stage corresponds to a specific spatiotemporal resolution, where stage $i = 0$ operates at the original video tensor size, and stage $i = S - 1$ at the lowest resolution. In practice, we use $S = 3$. The upsampling operation $\mathfrak{R}^\uparrow$ transitions from stage $i + 1$ to i by doubling the number of frames, height, and width using nearest-neighbor

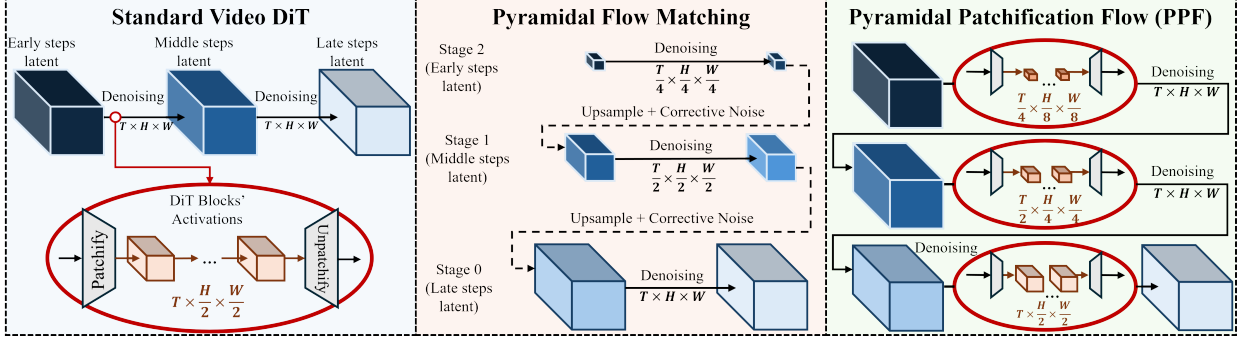


Figure 1. **Inference of different types of models.** Left: input and output tensors of a standard DiT always have the same size, and the number of tokens in transformer blocks does not depend on the noise level. Center: in pyramidal flow matching, higher noise levels are processed at smaller spatiotemporal resolution. For transition between stages special corrective noise should be added after upsampling. Right: in PPF framework instead of changing the resolution, kernel size of patchifier is adjusted for each stage. This keeps the number of tokens equal to that in pyramidal flow matching.

upsampling. Its counterpart, $\mathfrak{R}^\downarrow$, is implemented via 3D average pooling. Unlike prior works of Jin et al. [18] and Ran and Shou [29], which applied stage-wise generation to either spatial or temporal dimensions, we apply $\mathfrak{R}^\uparrow$ and $\mathfrak{R}^\downarrow$ across all three video axes.

3.1. Stage-wise definition of clean signal

Most recent video generation models operate in the latent space of a pretrained autoencoder (VAE). In this setting, the target signal x_0 is defined as the output of the VAE encoder $\mathcal{E}$ applied to an input RGB video $\mathcal{V}$, *i.e.*, $x_0 = \mathcal{E}(\mathcal{V})$. Therefore, there are two options for constructing the clean signal $x_0^{(i)}$ at stage $i > 0$. The first approach, adopted by Jin et al. [18], defines it as a latent signal, downsampled several times:

$$x_0^{(i)} = \mathfrak{R}^\downarrow \circ \dots \circ \mathfrak{R}^\downarrow(x_0). \quad (1)$$

The second option performs downsampling in the original RGB space before encoding with VAE,

$$x_0^{(i)} = \mathcal{E}(\text{Down}(\mathcal{V}, i)), \quad (2)$$

where Down denotes a suitable resizing operation, *e.g.* trilinear interpolation [33]. As we show below, the choice between these definitions depends on the specific training setup.

3.2. Stage-wise forward process

For each stage i , we define two boundary noise levels: a cleaner level $\sigma_c^{(i)}$ and a noisier level $\sigma_n^{(i)}$, such that $0 \leq \sigma_c^{(i)} < \sigma_n^{(i)} \leq 1$. Here, $\sigma = 0$ corresponds to a clean signal, and $\sigma = 1$ to i.i.d. Gaussian noise. Given the clean signal $x_0^{(i)}$ at the appropriate resolution, we compute two boundary

samples using a shared noise tensor $\epsilon \sim \mathcal{N}(0, I)$:

$$y_c^{(i)} = (1 - \sigma_c^{(i)}) x_0^{(i)} + \sigma_c^{(i)} \epsilon, \quad (3)$$

$$y_n^{(i)} = (1 - \sigma_n^{(i)}) \mathfrak{R}^\uparrow(x_0^{(i+1)}) + \sigma_n^{(i)} \epsilon. \quad (4)$$

For the noise level σ such that $\sigma_c^{(i)} < \sigma \leq \sigma_n^{(i)}$, the noised signal $x_\sigma^{(i)}$ is defined via linear interpolation between boundary samples

$$x_\sigma^{(i)} = (1 - \rho) y_c^{(i)} + \rho y_n^{(i)} \quad (5)$$

$$= (1 - \rho) (1 - \sigma_c^{(i)}) x_0^{(i)} + \rho (1 - \sigma_n^{(i)}) \mathfrak{R}^\uparrow(x_0^{(i+1)}) + \sigma \epsilon, \quad (6)$$

where $\rho = \frac{\sigma - \sigma_c^{(i)}}{\sigma_n^{(i)} - \sigma_c^{(i)}}$ is the *local* noise level, $0 < \rho \leq 1$. For clarity, in contrast to local level ρ we refer to σ as the *global* noise level, since it determines the total amount of added noise, as evident from Eq. (4). At the target stage $i = 0$ global noise levels are also called *natural* noise levels and denoted by ς . Although, strictly speaking Eq. (4) describes a generalized stochastic interpolant [1] rather than diffusion, we will still use the term ‘diffusion’ for simplicity.

Importantly, in PyramidalFlow framework, resampling operations $\mathfrak{R}^\uparrow$ and $\mathfrak{R}^\downarrow$, stage-wise clean signals, and boundary noise levels are defined to satisfy the following equivalence of probabilistic distributions

$$\mathfrak{R}_\mathcal{N}^\uparrow(y_c^{(i+1)}) \stackrel{d}{=} y_n^{(i)}, \quad (7)$$

where $\stackrel{d}{=}$ means equality in distribution, and $\mathfrak{R}_\mathcal{N}^\uparrow$ is the up-sampling operation $\mathfrak{R}^\uparrow$ followed by the addition of some

amount of non-i.i.d. noise. This noise aims to decorrelate the adjacent pixels after upsampling, and its parameters have been derived by Jin et al. [18]. In Supplementary, we generalize the functions $\mathfrak{R}^\downarrow$, $\mathfrak{R}^\uparrow$, and $\mathfrak{R}_\mathcal{N}^\uparrow$ from simple resampling operations (average pooling and nearest neighbor interpolation) to any resizing methods based on orthogonal transforms, *e.g.* wavelets. Our generalization involves sampling the missing high-frequency components before upscaling from Gaussian noise. This enables decorrelation even when the upscaling operation involves interaction between pixels.

Equation (5) implies that the ‘cleaner’ boundary sample from stage $i + 1$ after upsampling with $\mathfrak{R}_\mathcal{N}^\uparrow$ becomes a valid ‘noisier’ boundary sample of stage i . This establishes a relation between noise levels across two consecutive stages. This relation, if applied recursively, allows to map any global noise level $\sigma^{(i)}$ at stage i to a corresponding natural noise level ς at stage 0. In Supplementary, we show that natural noise levels corresponding to different stages do not overlap. This ensures that the natural level ς is a unique conditioning value for the denoising network in the pyramidal setup, independent of the number of stages S .

The ‘noisier’ bounds $\{\sigma_n^{(i)}\}_i$ are selected in practice via spectral analysis of noised samples at each stage. Specifically, $\sigma_n^{(i)}$ should be large enough that high-frequency components become indistinguishable from scaled Gaussian noise, *i.e.* $p(y_n^{(i)} | x_0^{(i)}) \approx \mathcal{N}\left(\left(1 - \sigma_n^{(i)}\right) x_0^{(i)}, \left(\sigma_n^{(i)}\right)^2 I\right)$. This allows to downsample and proceed with the forward process at stage $i + 1$ without loss of information [8, 33].

4. Method

4.1. Pyramidal finetuning

To convert a pretrained conventional diffusion model to its pyramidal version, we apply finetuning with the dedicated loss function.

Flow matching loss. We start with the pretrained open-sourced Wan2.1-1.3B model F . For brevity, hereinafter we omit the conditioning text prompt c in the notation. Since this model was trained with flow matching loss [36], it approximates the derivative of the noised signal w.r.t. the noise level [26],

$$F(x_\sigma, \sigma) \approx \mathbb{E} \left[\frac{dx_\sigma}{d\sigma} \mid x_\sigma \right]. \quad (6)$$

To preserve this property during pyramidal finetuning, at each stage i we define the objective for the student network F_θ as the derivative w.r.t. the global noise level, *i.e.*

$$\frac{dx_\sigma^{(i)}}{d\sigma} = \frac{dx_\sigma^{(i)}}{d\rho} \frac{d\rho}{d\sigma} = \frac{y_n^{(i)} - y_c^{(i)}}{\sigma_n^{(i)} - \sigma_c^{(i)}}.$$

The pyramidal loss is defined then as

$$L_{\text{pyr}}(\theta) = \sum_i \mathbb{E}_{x_0^{(i)}} \mathbb{E}_\epsilon \mathbb{E}_\rho \left\| F_\theta(x_\sigma^{(i)}, \varsigma) - \frac{dx_\sigma^{(i)}}{d\sigma} \right\|^2, \quad (7)$$

for uniformly distributed local level $\rho \sim \text{Uni}(0, 1)$.

Distillation loss. In addition to flow matching, we apply a distillation loss to align student’s partially denoised latents with the teacher’s predictions. For stage i and global noise level σ , $\sigma_c^{(i)} < \sigma \leq \sigma_n^{(i)}$, we first map σ to the natural noise level ς as discussed in Sec. 3.2. We sampled the high-resolution noise $\epsilon^{(0)}$ at stage 0, and construct the teacher’s prediction for the noisy input $x_\varsigma = (1 - \varsigma) x_0 + \varsigma \epsilon^{(0)}$ as

$$\tilde{x}_{\varsigma^{(i)}} = x_\varsigma - \left(\varsigma - \varsigma_c^{(i)} \right) \cdot F(x_\varsigma, \varsigma), \quad (8)$$

where $\varsigma_c^{(i)}$ is the natural noise level corresponding to the ‘cleaner’ bound of the denoising process at stage i .

Next, we downsample the noise to stage i and scale by a constant to preserve unit variance, $\epsilon^{(i)} \propto \mathfrak{R}^\downarrow \circ \dots \circ \mathfrak{R}^\downarrow(\epsilon^{(0)})$. Using $\epsilon^{(i)}$, we construct boundary samples $y_c^{(i)}$ and $y_n^{(i)}$ for the student’s noised input $x_\sigma^{(i)}$. Student’s single-step prediction of the cleaner boundary value equals

$$\tilde{y}_c^{(i)} = x_\sigma^{(i)} - \left(\sigma - \sigma_c^{(i)} \right) \cdot F(x_\sigma^{(i)}, \varsigma). \quad (9)$$

The distillation loss is then defined as

$$L_{\text{dist}}(\theta) = \sum_i \mathbb{E}_{x_0^{(i)}, x_0} \mathbb{E}_{\epsilon^{(0)}} \mathbb{E}_\sigma \left\| \tilde{y}_c^{(i)} - \tilde{x}_{\varsigma^{(i)}} \right\|^2. \quad (10)$$

In early experiments, we found that training with latents downsampled in pixel space, *e.g.* using Definition ②, yields significantly better visual results. This observation aligns with findings of Starodubcev et al. [33]. We refer to the resulting model F_θ as *PyramidalWan*.

4.2. Pyramidal step distillation

While pyramidal diffusion alone reduces computational cost by 78% (see Tab. 1), we aim to further decrease latency through step distillation. Below, we describe how distribution matching distillation (DMD) and adversarial technique are adapted to the pyramidal framework.

4.2.1. DMD with original teacher

Since in case of Wan2.1 model we have access to the original pretrained non-pyramidal model F , we can use it as a teacher in a DMD pipeline [40]. In this method, the student model F_ξ is trained to predict the clean signal at the i -th stage in a single step, $\hat{x}_0^{(i)} = x_{\sigma^{(i)}} - \sigma^{(i)} \cdot F_\xi(x_{\sigma^{(i)}}, \varsigma)$. Notably, to construct the input $x_{\sigma^{(i)}}$ we follow the rollout

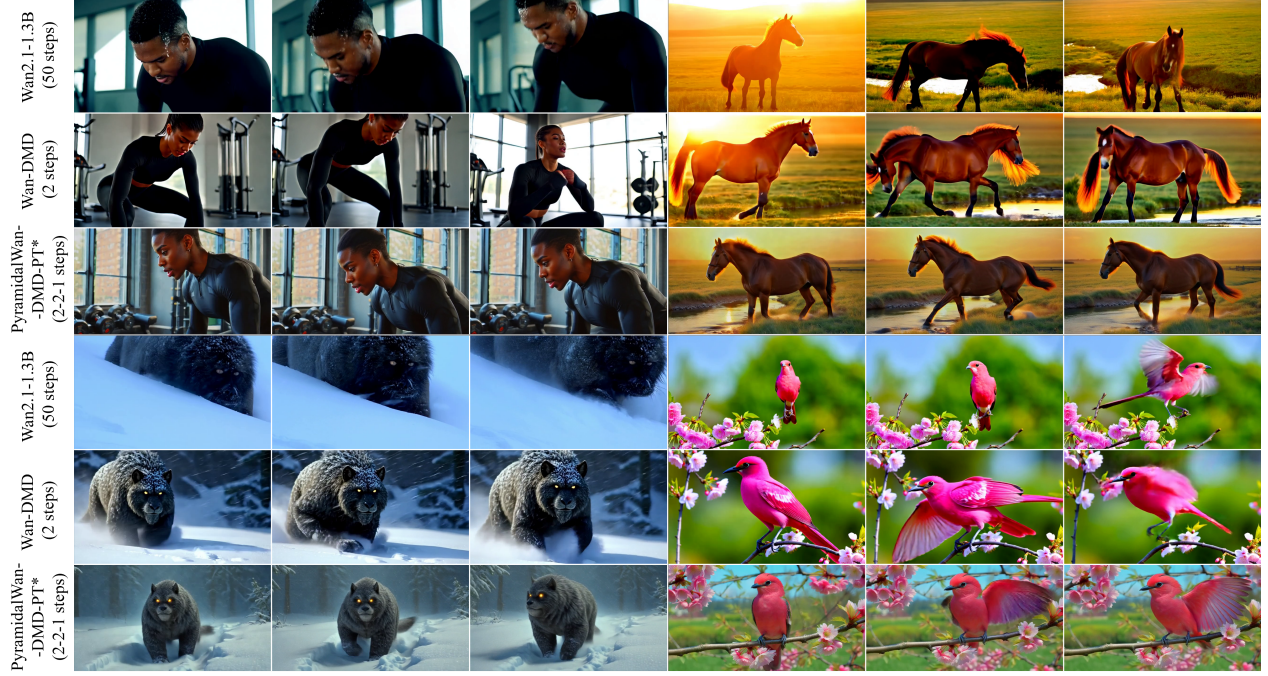


Figure 2. **Examples of video generations.** Videos produced by our pyramidal step-distilled model are similar in quality to outputs of more computationally expensive baselines.

strategy proposed by Yin et al. [40], using a detached output of the student:

$$\mathbf{x}_{\sigma^{(i)}} = \mathbf{y}_n^{(i)} + \left(\sigma^{(i)} - \sigma_n^{(i)} \right) \cdot \text{stopgrad} \left[F_{\xi} \left(\mathbf{y}_n^{(i)}, \varsigma_n^{(i)} \right) \right]. \quad (11)$$

Once the clean signal $\hat{\mathbf{x}}_0^{(i)}$ is predicted, it is re-noised according to the teacher’s forward process with a noise level σ' , $0 < \sigma' \leq 1$,

$$\hat{\mathbf{x}}_{\sigma'}^{(i)} = (1 - \sigma') \hat{\mathbf{x}}_0^{(i)} + \sigma' \epsilon.$$

The *fake score* network F_{φ} is trained using regular flow matching loss, aiming to denoise the student’s predictions,

$$L_{\text{fm}}(\varphi) = \sum_i \mathbb{E}_{\hat{\mathbf{x}}_{\sigma'}^{(i)}} \left\| F_{\varphi} \left(\hat{\mathbf{x}}_{\sigma'}^{(i)}, \sigma' \right) - \frac{d\hat{\mathbf{x}}_{\sigma'}^{(i)}}{d\sigma'} \right\|^2.$$

The gradient of DMD loss is defined as follows,

$$\nabla_{\xi} L_{\text{dmd}}(\xi) = \left(F_{\varphi} \left(\hat{\mathbf{x}}_{\sigma'}^{(i)}, \sigma' \right) - F \left(\hat{\mathbf{x}}_{\sigma'}^{(i)}, \sigma' \right) \right) \cdot \nabla_{\xi} F_{\xi}(\mathbf{x}_{\sigma^{(i)}}, \varsigma). \quad (12)$$

For each training sample, weight of the loss L_{dmd} is set equal to w_{dmd} ,

$$w_{\text{dmd}} = \sigma^{(i)} \cdot \left\| F \left(\hat{\mathbf{x}}_{\sigma'}^{(i)}, \sigma' \right) - \frac{d\hat{\mathbf{x}}_{\sigma'}^{(i)}}{d\sigma'} \right\|_1^{-1}, \quad (13)$$

giving greater weight to samples whose re-noised versions can be well ‘denoised’ by the teacher F . To stabilize training, we also include a supervised loss that encourages the

student’s output to be similar to the teacher’s one [31], $L_{\text{teach}}(\xi) = \|F_{\xi}(\mathbf{x}_{\sigma^{(i)}}, \varsigma) - F(\mathbf{x}_{\sigma^{(i)}}, \sigma^{(i)})\|^2$. This term is added to the DMD loss with a weight of 0.01.

We found experimentally that such training does not work for the original pretrained Wan2.1-1.3B teacher due to its inability to generate videos at the lowest spatiotemporal resolution, *i.e.* for $i = S - 1$. While this may seem to contradict the results recently reported by Starodubcev et al. [33], we note that their SwD method gradually upscales the video tensor by a fractional factor after each step, likely reaching the resolution compatible with the teacher model earlier in the generation process. In our case, to remain consistent with PyramidalFlow framework, we briefly finetuned the teacher using flow matching loss on a dataset of videos with varying resolution. Clean latent tensors were generated using Definition 2, which was also used for training. In the beginning of step distillation, both the student and fake score network were initialized from the finetuned teacher’s checkpoint. At inference time, we use the same sampling algorithm as in pyramidal diffusion, but with only a few steps per stage [24]. The resulting few-step generator is referred to as *PyramidalWan-DMD-OT*.

4.2.2. DMD with pyramidal teacher

The DMD pipeline described above requires modifications when employing a pyramidal flow matching model as the teacher. This setup is increasingly relevant given the reported training efficiency of pyramidal diffusion models [21, 34]. The key difference arises from the fact that

pyramidal teacher has been trained with stage-wise inputs. Therefore, student’s prediction of the clean signal $\hat{x}_0^{(i)}$ should be re-noised similarly to Eqs. (1) and (2),

$$\hat{y}_c^{(i)} = \left(1 - \sigma_c^{(i)}\right) \hat{x}_0^{(i)} + \sigma_c^{(i)} \varepsilon, \quad (14)$$

$$\hat{y}_n^{(i)} = \left(1 - \sigma_n^{(i)}\right) \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow \left(\hat{x}_0^{(i)}\right) + \sigma_n^{(i)} \varepsilon, \quad (15)$$

$$\hat{x}_{\sigma'}^{(i)} = (1 - \rho') \hat{y}_c^{(i)} + \rho' \hat{y}_n^{(i)}. \quad (16)$$

This definition of $\hat{y}_n^{(i)}$ assumes that per-stage clean signals follow Definition ①, in contrast to how the pyramidal diffusion model has been trained (see Sec. 4.1). Using Definition ② would require both VAE decoder and encoder to compute $\hat{x}_0^{(i+1)}$, which is computationally expensive for video models. To address this, before applying the DMD pipeline, we finetuned our pyramidal diffusion model F_θ according to this definition, resulting in the model F_{θ_1} . We found that this approach works better than training with Definition ① from the beginning. Importantly, using the original F_θ as a teacher leads to unsatisfactory quality of few-step generations.

The fake score network F_φ is trained with pyramidal flow matching loss $L_{\text{pyr}}(\varphi)$ from Eq. (7), but with re-noised predicted signals $\hat{x}_0^{(i)}$ instead of ground-true noised signals $x_0^{(i)}$.

The original DMD formulation relies on estimating the score function, or equivalently, the added Gaussian noise ε [1, 28, 41]. To construct such an estimator, we define $\Delta^{(i)}$, a linear combination of $\hat{y}_c^{(i)}$ and $\hat{y}_n^{(i)}$,

$$\Delta^{(i)} = \sigma_n^{(i)} \hat{y}_c^{(i)} - \sigma_c^{(i)} \hat{y}_n^{(i)} \quad (17)$$

$$= \sigma_n^{(i)} \left(1 - \sigma_c^{(i)}\right) \hat{x}_0^{(i)} - \sigma_c^{(i)} \left(1 - \sigma_n^{(i)}\right) \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow \left(\hat{x}_0^{(i)}\right). \quad (18)$$

Applying $\mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow$ to both LHS and RHS and using the fact that $\mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow \circ \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow = \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow$, gives

$$\mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow \left(\Delta^{(i)}\right) = \left(\sigma_n^{(i)} - \sigma_c^{(i)}\right) \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow \left(\hat{x}_0^{(i)}\right). \quad (19)$$

Together with Eq. (15), this yields a closed-form expression for ε given $\hat{y}_n^{(i)}$ and $\hat{y}_c^{(i)}$.

For a pretrained pyramidal teacher or pyramidal fake score model F_ν , where $\nu \in \{\theta_1, \varphi\}$, the stage boundary samples are estimated as $\hat{x}_{\sigma'}^{(i)} + \left(\sigma_n^{(i)} - \sigma'\right) F_\nu \left(\hat{x}_{\sigma'}^{(i)}, \varsigma'\right)$ and $\hat{x}_{\sigma'}^{(i)} + \left(\sigma_c^{(i)} - \sigma'\right) F_\nu \left(\hat{x}_{\sigma'}^{(i)}, \varsigma'\right)$ respectively.

The noise estimator $\hat{\varepsilon}_\nu$ equals then (we omit arguments of F_ν for brevity)

$$\begin{aligned} \hat{\varepsilon}_\nu &= \frac{1}{\sigma_n^{(i)}} \left(\hat{x}_{\sigma'}^{(i)} + \left(\sigma_n^{(i)} - \sigma'\right) F_\nu\right) \\ &\quad - \frac{1 - \sigma_n^{(i)}}{\sigma_n^{(i)}} \left(\mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow \left(\hat{x}_{\sigma'}^{(i)}\right) - \sigma' \cdot \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow (F_\nu)\right). \end{aligned}$$

The difference between the estimators provided by the teacher and the fake score model is proportional to

$$\hat{\varepsilon}_\varphi - \hat{\varepsilon}_{\theta_1} \propto \beta_1 \cdot (F_\varphi - F_{\theta_1}) + \beta_2 \cdot \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow (F_\varphi - F_{\theta_1}),$$

where $\beta_1 = \sigma_n^{(i)} - \sigma'$, $\beta_2 = \sigma' \left(1 - \sigma_n^{(i)}\right)$. We normalize these weights as $\tilde{\beta}_k = \frac{\beta_k}{\beta_1 + \beta_2}$ for $k = 1, 2$. Similarly, the gradient of the re-noised prediction is proportional to the following weighted sum

$$-\nabla_\xi \hat{x}_{\sigma'}^{(i)} \propto \gamma_1 \nabla_\xi F_\xi + \gamma_2 \nabla_\xi \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow (F_\xi), \quad (20)$$

with the weights $\gamma_1 = \left(\sigma_n^{(i)} - \sigma'\right) \left(1 - \sigma_c^{(i)}\right)$, $\gamma_2 = \left(\sigma' - \sigma_c^{(i)}\right) \left(1 - \sigma_n^{(i)}\right)$. We normalize these as $\tilde{\gamma}_k = \frac{\gamma_k}{\gamma_1 + \gamma_2}$ and define the gradient of pyramidal DMD loss as

$$\begin{aligned} \nabla_\xi L_{\text{dmd-pyr}}(\xi) &= \left(\tilde{\beta}_1 (F_\varphi - F_{\theta_1}) + \tilde{\beta}_2 \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow (F_\varphi - F_{\theta_1})\right) \\ &\quad \cdot \nabla_\xi \left(\tilde{\gamma}_1 F_\xi + \tilde{\gamma}_2 \mathfrak{R}^\uparrow \circ \mathfrak{R}^\downarrow (F_\xi)\right). \end{aligned} \quad (21)$$

Each sample in $L_{\text{dmd-pyr}}$ loss is weighted with

$$w_{\text{dmd-pyr}} = \rho \cdot \left\| F_{\theta_1} \left(\hat{x}_{\sigma'}^{(i)}, \varsigma'\right) - \frac{d\hat{x}_{\sigma'}^{(i)}}{d\sigma'} \right\|_1^{-1}. \quad (22)$$

We call the resulting model *PyramidalWan-DMD-PT*.

In addition, we explored the simplified version by setting $\tilde{\beta}_1 = \tilde{\gamma}_1 = 1$ and $\tilde{\beta}_2 = \tilde{\gamma}_2 = 0$ in Eq. (21). This reduces $L_{\text{dmd-pyr}}$ to the formulation similar to Eq. (12). Although such a variant has insufficient theoretical grounding, we found that it performs marginally better in practice. This modification is referred to as *-PT**.

4.2.3. Adversarial distillation

As an alternative to DMD, we explore pyramidal adversarial distillation. In this setting, student F_ξ similarly predicts a ‘cleaner’ boundary sample $\hat{y}_c^{(i)}$ of the current stage i in a single step, while a discriminator attempts to distinguish between features extracted from generated and ground-true samples. The discriminator consists of two components:

1. Frozen feature extractor $F^\dagger$, based on a pretrained diffusion model. We denote the model as *PyramidalWan-Adv-OD* when using the backbone from the original Wan model F , and *-PD* when using the pyramidal backbone of F_θ instead.
2. Trainable discriminator head D_φ attached to the final block of $F^\dagger$, comprises *spatial* and *temporal* branches implemented with lightweight convolutional layers and residual blocks.

The discriminator minimizes an adversarial Hinge loss [22] over features at each stage,

$$\begin{aligned} L_D(\varphi) &= \sum_i \left[\mathbb{E}_{y_c^{(i)}} \left[\max(0, 1 - D_\varphi(F^\dagger(y_c^{(i)})) \right] \right. \\ &\quad \left. + \mathbb{E}_{y_c^{(i)}} \left[\max(0, 1 + D_\varphi(F^\dagger(\hat{y}_c^{(i)})) \right] \right]. \end{aligned}$$

The student optimizes a combined objective balancing adversarial and reconstruction terms, weighted by λ_{adv} and λ_{rec} , similarly to the approach of Zhang et al. [45]. Empirically, we find that $\lambda_{\text{adv}} = 1$ and $\lambda_{\text{rec}} = 2$ yield the highest visual quality.

$$L_G(\xi) = \sum_i \mathbb{E} \left[-\lambda_{\text{adv}} \cdot D_{\varphi}(F^{\dagger}(\hat{y}_c^{(i)})) + \lambda_{\text{rec}} \cdot \|\hat{y}_c^{(i)} - y_c^{(i)}\|_2^2 \right]$$

4.3. Patch-pyramidal training

An alternative approach for varying per-stage computational cost is Pyramidal Patchification Flow (PPF) [21]. PPF does not alter the resolution of the denoiser transformer’s inputs or outputs. Instead, it introduces stage-wise patchification and unpatchification layers, as Fig. 1 shows. For earlier stages, kernel size of patchifier is accordingly increased, and therefore, the transformers blocks in PPF operate with exactly the same number of tokens as for PyramidalFlow. Importantly, diffusion training, step distillation, and inference can be performed within the PPF framework in the same way as for the original pretrained Wan model.

5. Experiments

5.1. Training setup

Step distillation of the original model. As a natural baseline for reducing inference cost, we trained a step-distilled version of the original Wan model. Following the DMD pipeline described in Sec. 4.2.1, we adopted the one-step rollout strategy [40] but omitted the adversarial loss, as we found it unnecessary. This training setup does not require visual data and relies solely on text prompts; we used prompts from the 350K subset of the dataset provided by Lin et al. [23]. Training was conducted for 31K iterations on 16 H100 GPUs. As an alternative, we performed adversarial distillation on the original Wan model, following the procedure similar to the one outlined in Sec. 4.2.3. The model was trained for 30K iterations on a single H100 GPU using 80K synthetic videos generated by Wan2.1-14B, a larger variant of the original model.

Pyramidal flow matching and step distillation. For these experiments, we used the same synthetic dataset of 80K videos generated by the Wan2.1-14B model. We observed that models trained on the synthetic data produce visually superior results compared to those trained on real video samples. To ensure compatibility with Wan’s patchification layer at the lowest stage ($i = 2$), we slightly reduced the spatial resolution of videos from the default 480×832 to 448×832 , making thus both height and width divisible by 64. All models mentioned in Secs. 4.1 and 4.2 were fine-tuned for 5K iterations with a batch size of 6 per GPU (2 samples per stage) on two H100 GPUs. For DMD-PT and DMD-PT* we conducted training with LoRA adapters, since we found that otherwise training might diverge. In other cases, we finetuned all the weights.

Patch-pyramidal training. To train the diffusion-based Wan-PPF model, we used the same setup as for PyramidalWan. Training details of patch-pyramidal DMD model were kept consistent with those used for DMD distillation of the original Wan model, except for the smaller training budget. This experiment was conducted for 5K steps on 8 GPUs, with a batch size of 3 per GPU (one sample per stage). We did not observe improvements with longer training.

5.2. Results

Main results. We evaluate the quality of generated videos using the VBench and VBench-2.0 toolkits [17, 46]. Our results are summarized in Tab. 3. First, we observe that PyramidalWan, *i.e.* the pyramidized diffusion model, achieves scores comparable to the original Wan model sampled with 50 steps, while being approximately 4.5 times more efficient in terms of FLOPs. Notably, it also achieves the highest Semantic score among all evaluated models, resembling the findings of the concurrent work of Zhang et al. [44].

While Li et al. [21] successfully demonstrated finetuning of pretrained text-to-image models within the PPF framework, we found that extending this approach to video generation is challenging. Using the same training budget and dataset as in our pyramidal training setup, our PPF-based text-to-video diffusion model failed to converge, and the quality of generated clips remained unsatisfactory. Notably, increasing the compute budget to 8 GPUs did not improve results. Step distillation using DMD also failed when the new patchification and unpatchification layers were initialized according to the scheme proposed in the original PPF work. Surprisingly, however, the DMD pipeline could still be applied successfully when the student model was initialized with the pretrained PPF diffusion model — even though that checkpoint itself produced poor-quality generations. We attribute this to the mode-seeking behavior of the reverse-KL objective used in DMD [38].

Among few-step models, the step-distilled baseline Wan-DMD provides very strong performance combined with efficient inference. Despite the significant gains in test-time efficiency (see Tab. 1), it surpasses the original diffusion model in both total scores even when sampled with only two steps. However, single-step generation remains infeasible as the quality of videos drops substantially. We fill this gap with our few-step pyramidal models. They are evaluated in a scenario with only one step at highest resolution, *i.e.* at stage $i = 0$, and a few steps at lower-resolution stages. Given the much lower computational cost and latency of these stages, as reported in Tabs. 1 and 2, we adopt a 2-2-1 inference schedule (steps per stage in resolution-increasing order).

All models under this schedule achieve total score of VBench comparable to Wan diffusion model, with only a minor drop relative to Wan-DMD sampled with 2 steps.

Table 3. **Model comparison.** Step-distilled versions of the original model, Wan-DMD and Wan-Adv, provide the quality of generation in the few-step mode on par with the diffusion model. However, they cannot unlock the satisfactory single-step inference. Pyramidal models with a single step at highest resolution fill the gap and demonstrate good performance, as measured quantitatively.

Model	VBench $\uparrow$			VBench-2.0 $\uparrow$					
	Total	Quality	Semantic	Total	Creativity	Commonsense	Controllability	Human Fidelity	Physics
Wan2.1-1.3B (50 steps)	82.49	83.47	78.57	56.02	48.73	63.38	33.96	80.71	53.30
Wan2.1-1.3B (25 steps)	80.87	82.09	76.02	55.73	49.49	62.50	35.01	79.44	52.20
PyramidalWan (20-20-10)	82.83	83.36	80.70	54.93	44.64	64.33	28.39	85.38	51.89
Wan-Adv (4 steps)	82.72	84.06	77.39	55.40	50.96	57.33	32.56	85.58	50.57
Wan-Adv (2 steps)	82.35	83.74	76.82	54.82	47.07	58.47	31.51	82.72	54.36
Wan-Adv (1 step)	80.28	81.38	75.85	50.36	38.51	54.41	29.99	79.60	49.29
Wan-DMD (4 steps)	83.33	84.71	77.86	57.48	50.87	58.16	36.36	82.80	59.23
Wan-DMD (2 steps)	83.28	84.00	80.41	56.67	47.00	62.81	37.07	80.73	55.72
Wan-DMD (1 step)	79.45	80.63	74.75	53.17	38.69	59.31	35.36	77.49	55.00
PyramidalWan-Adv-OD (2-2-1)	82.90	83.94	78.74	52.29	44.80	60.86	22.86	81.74	51.17
PyramidalWan-Adv-PD (2-2-1)	82.57	84.20	76.07	54.30	45.00	61.98	25.85	87.65	51.01
Wan-PPF-DMD (2-2-1)	82.39	83.04	79.80	53.45	40.04	62.84	24.76	91.61	47.99
PyramidalWan-DMD-OT (2-2-1)	82.86	83.63	79.80	55.36	50.04	64.34	29.05	77.50	56.78
PyramidalWan-DMD-PT* (2-2-1)	82.72	83.46	79.75	51.75	34.81	63.18	28.48	81.38	50.92

VBench-2.0, however, indicates some degradation, in particular for Creativity and Controllability dimensions. Although distillation with the original teacher, PyramidalWan-DMD-OT, gets the best metrics within this set of models, we observed that its outputs often exhibit oversaturated colors and cartoon-alike appearance. Please refer to Supplementary for visual examples. Among all models, we found that PyramidalWan-DMD-PT* produced the most visually appealing results, and therefore selected it for the user study.

In the study, the assessors were shown pairs of videos generated from identical prompts and asked to choose the preferred one or select “no preference”. Each pair included one video generated with step-distilled pyramidal model, and one from either Wan with 50 steps (first study) or Wan-DMD with 2 steps (second). In total, we collected 700 responses. We conducted the binomial test for the hypothesis ‘Baseline is strictly preferred with probability 0.5’ and the one-sided ‘less’ alternative. As shown in Tab. 4, in both comparisons the hypothesis should be rejected. This indicates that the participants found the quality of this model on par with more computationally expensive baselines despite its lower VBench-2.0 score. Examples of videos are provided in Fig. 2.

Ablations. To check the impact of distillation losses in both pyramidal finetuning and step distillation experiments, we conducted experiments by removing this terms. For PyramidalWan, this reduced VBench-2.0 total score from 54.93 to 54.02. Notably, for DMD with pyramidal teacher effect was the opposite (see Tab. 5), however at the expense of noticeable reduction of Dynamic Degree. As mentioned in Sec. 4.2.2, the simplified version of $L_{\text{dmd-pyr}}$ objective yields improved scores despite its theoretical weakness. We leave further investigation of this phenomenon for the future.

Table 4. **User study.** We evaluate our pyramidal model with 2-2-1 inference schedule against two baselines. As the results show, participants did not find significant difference in visual quality.

Baseline	Preference, % $\uparrow$			p -value
	Ours	No preference	Baseline	
Wan (50 steps)	29.1	29.1	41.7	< 0.001
Wan-DMD (2 steps)	33.1	35.4	31.4	< 0.001

Table 5. **Ablation study.** Training without distillation loss improves the VBench-2.0 score but reduces the amount of motion in generated videos. Simplified version of DMD objective leads to better empirical results.

Model	Total score $\uparrow$	
	VBench	VBench-2.0
PyramidalWan-DMD-PT*	82.72	51.75
PyramidalWan-DMD-PT* w/o L_{teach}	82.44	52.36
PyramidalWan-DMD-PT	82.56	50.67

6. Conclusion

In this work we explored pyramidization — a strategy of reducing inference costs of video diffusion models that is complementary to other architectural innovations. We presented a pipeline of converting a pretrained *conventional* diffusion model into a pyramidal one, both for multi-step and few-step inference regimes. Further, we demonstrated step distillation of a pretrained *pyramidal* diffusion model: an important milestone for future research given the reported training efficiency of such systems. In addition, we made a theoretical contribution by extending the procedure of switching between resolutions to a broader class of upsampling operations. The resulting models occupy the practical niche enabling few-step generation with only a single step at the target resolution. While demonstrating results comparable to more costly baselines in the human preference study, our models still lag behind in certain quantitative metrics. Addressing this gap remains a promising direction for future work.

References

- [1] Michael Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025. 3, 6
- [2] Sotiris Anagnostidis, Gregor Bachmann, Yeongmin Kim, Jonas Kohler, Markos Georgopoulos, Arsiom Sanakoyeu, Yuming Du, Albert Pumarola, Ali Thabet, and Edgar Schönfeld. Flexidit: Your diffusion transformer can easily generate high-quality samples with less compute. In *CVPR*, 2025. 2
- [3] Haitam Ben Yahia, Denis Korzhnikov, Ioannis Lelekas, Amir Ghodrati, and Amirhossein Habibi. Mobile video diffusion. In *ICCV*, 2025. 2
- [4] Andrew Campbell, William Harvey, Christian Dietrich Weibach, Valentin De Bortoli, Tom Rainforth, and Arnaud Doucet. Trans-dimensional generative modeling via jump diffusion models. In *NeurIPS*, 2023. 1, 2
- [5] Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Song Han, and Enze Xie. Sana-sprint: One-step diffusion with continuous-time consistency distillation. In *ICCV*, 2025. 2
- [6] Junsong Chen, Yuyang Zhao, Jincheng YU, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, Haozhe Liu, Hongwei Yi, Hao Zhang, Muiyang Li, Yukang Chen, Han Cai, Sanja Fidler, Ping Luo, Song Han, and Enze Xie. SANA-video: Efficient video generation with block linear diffusion transformer. In *ICLR*, 2026. 1
- [7] Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow, 2025. 1, 2
- [8] Sander Dieleman. Diffusion is spectral autoregression, 2024. 1, 2, 4
- [9] Fabian Falck, Teodora Pandevara, Kiarash Zahirnia, Rachel Lawrence, Richard Turner, Edward Meeds, Javier Zazo, and Sushrut Karmalkar. A fourier space perspective on diffusion models, 2025. 1, 2
- [10] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *NeurIPS*, 2025. 2
- [11] Mohsen Ghafoorian, Denis Korzhnikov, and Amirhossein Habibi. Attention surgery: An efficient recipe to linearize your video diffusion transformer. In *CVPR*, 2026. 1
- [12] 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, 2024. 1
- [13] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022. 2
- [14] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *ICML*, 2023. 2
- [15] Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion: 1.5 fid on imagenet512 with pixel-space diffusion. In *CVPR*, 2025. 2
- [16] 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 *ICLR*, 2022. 2
- [17] Ziqi Huang, Yanan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024. 7
- [18] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong MU, and Zhouchen Lin. Pyramidal Flow Matching for Efficient Video Generative Modeling. In *ICLR*, 2025. 1, 2, 3, 4
- [19] Animesh Karnewar, Denis Korzhnikov, Ioannis Lelekas, Noor Fathima, Adil Karjauv, Mohsen Ghafoorian, and Amir Habibi. Neodragon: Mobile video generation using diffusion transformer. In *ICLR*, 2026. 2
- [20] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojuan Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xincheng Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. Hunyuanvideo: A systematic framework for large video generative models, 2025. 1
- [21] Hui Li, Baoyou Chen, Li jiaye, Jingdong Wang, and Siyu Zhu. Pyramid patchification flow for visual generation. In *ICLR*, 2026. 2, 5, 7
- [22] Jae Hyun Lim and Jong Chul Ye. Geometric gan, 2017. 6
- [23] Bin Lin, Yunyang Ge, Xinhua Cheng, Zongjian Li, Bin Zhu, Shaodong Wang, Xianyi He, Yang Ye, Shenghai Yuan, Lihuan Chen, Tanghui Jia, Junwu Zhang, Zhenyu Tang, Yatian Pang, Bin She, Cen Yan, Zhiheng Hu, Xiaoyi Dong, Lin Chen, Zhang Pan, Xing Zhou, Shaoling Dong, Yonghong Tian, and Li Yuan. Open-sora plan: Open-source large video generation model, 2024. 7
- [24] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *WACV*, 2024. 5
- [25] Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion adversarial post-training for one-step video generation. In *ICML*, 2025. 1, 2
- [26] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023. 4
- [27] Dongyang Liu, Shicheng Li, Yutong Liu, Zhen Li, Kai Wang, Xinyue Li, Qi Qin, Yufei Liu, Yi Xin, Zhongyu Li, Bin Fu, Chenyang Si, Yuewen Cao, Conghui He, Ziwei Liu,

- Yu Qiao, Qibin Hou, Hongsheng Li, and Peng Gao. Lumina-video: Efficient and flexible video generation with multi-scale next-dit, 2025. 1
- [28] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-Instruct: A Universal Approach for Transferring Knowledge From Pre-trained Diffusion Models. In *NeurIPS*, 2023. 2, 6
- [29] Lingmin Ran and Mike Zheng Shou. TPDiff: Temporal pyramid video diffusion model. In *ICLR*, 2026. 2, 3
- [30] Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. In *ICLR*, 2023. 1
- [31] Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. In *NeurIPS*, 2025. 2, 5
- [32] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation, 2024. 2
- [33] Nikita Starodubcev, Ilya Drobyshevskiy, Denis Kuznedelev, Artem Babenko, and Dmitry Baranchuk. Scale-wise distillation of diffusion models. In *ICLR*, 2026. 2, 3, 4, 5
- [34] MiniMax Team. MiniMax Hailuo 02, World-Class Quality, Record-Breaking Cost Efficiency - MiniMax News, 2025. <https://www.minimax.io/news/minimax-hailuo-02>. 1, 5
- [35] Jiayan Teng, Wendi Zheng, Ming Ding, Wenyi Hong, Jianqiao Wangni, Zhuoyi Yang, and Jie Tang. Relay diffusion: Unifying diffusion process across resolutions for image synthesis. In *ICLR*, 2024. 1
- [36] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Fei Wu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wenten Wang, Wenting Shen, Wenyan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. 1, 4
- [37] Yushu Wu, Zhixing Zhang, Yanyu Li, Yanwu Xu, Anil Kag, Yang Sui, Huseyin Coskun, Ke Ma, Aleksei Lebedev, Ju Hu, Dimitris N. Metaxas, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapgen-v: Generating a five-second video within five seconds on a mobile device. In *CVPR*, 2025. 2
- [38] Yilun Xu, Weili Nie, and Arash Vahdat. One-step Diffusion Models with f -Divergence Distribution Matching, 2025. 7
- [39] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Wei Han Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer. In *ICLR*, 2025. 1
- [40] 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 *NeurIPS*, 2024. 2, 4, 5, 7
- [41] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step Diffusion with Distribution Matching Distillation. In *CVPR*, 2024. 2, 6
- [42] Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric P. Xing, and Hao Zhang. Faster video diffusion with trainable sparse attention. In *NeurIPS*, 2025. 1
- [43] Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. In *ICML*, 2025. 1
- [44] 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. 1, 2, 7
- [45] Zhixing Zhang, Yanyu Li, Yushu Wu, Yanwu Xu, Anil Kag, Ivan Skorokhodov, Willi Menapace, Aliaksandr Siarohin, Junli Cao, Dimitris N. Metaxas, Sergey Tulyakov, and Jian Ren. SF-v: Single forward video generation model. In *NeurIPS*, 2024. 7
- [46] Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yanan He, Fan Zhang, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, et al. Vbench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness, 2025. 7
- [47] Kaiwen Zheng, Yuji Wang, Qianli Ma, Huayu Chen, Juntao Zhang, Yogesh Balaji, Jianfei Chen, Ming-Yu Liu, Jun Zhu, and Qingsheng Zhang. Large scale diffusion distillation via score-regularized continuous-time consistency. In *ICLR*, 2026. 2