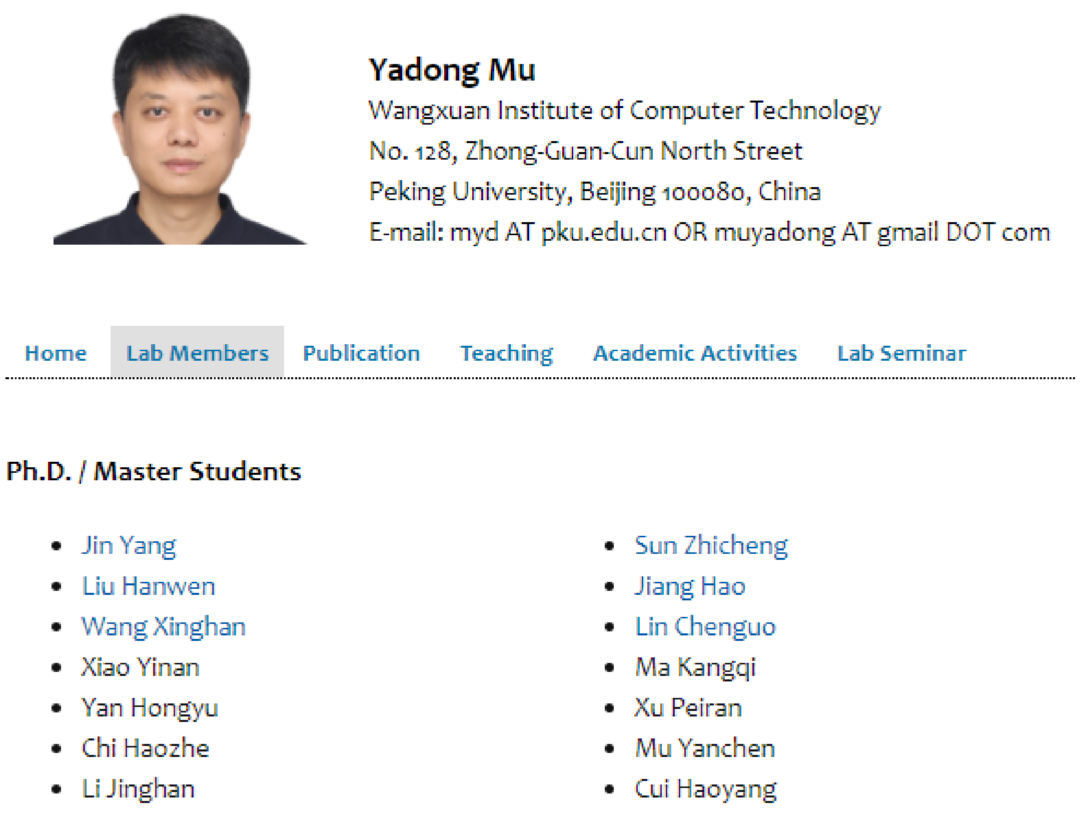
PYRAMIDAL FLOW MATCHING FOR EFFICIENT VIDEO GENERATIVE MODELING

Yang Jin[1], **Zhicheng Sun**[1], **Ningyuan Li**[3], **Kun Xu**[2], **Kun Xu**[2], **Hao Jiang**[1], **Nan Zhuang**[2], **Quzhe Huang**[2], **Yang Song**, **Yadong Mu**[1]*, **Zhouchen Lin**[1]

[1]Peking University, [2]Kuaishou Technology, [3]Beijing University of Posts and Telecommunications

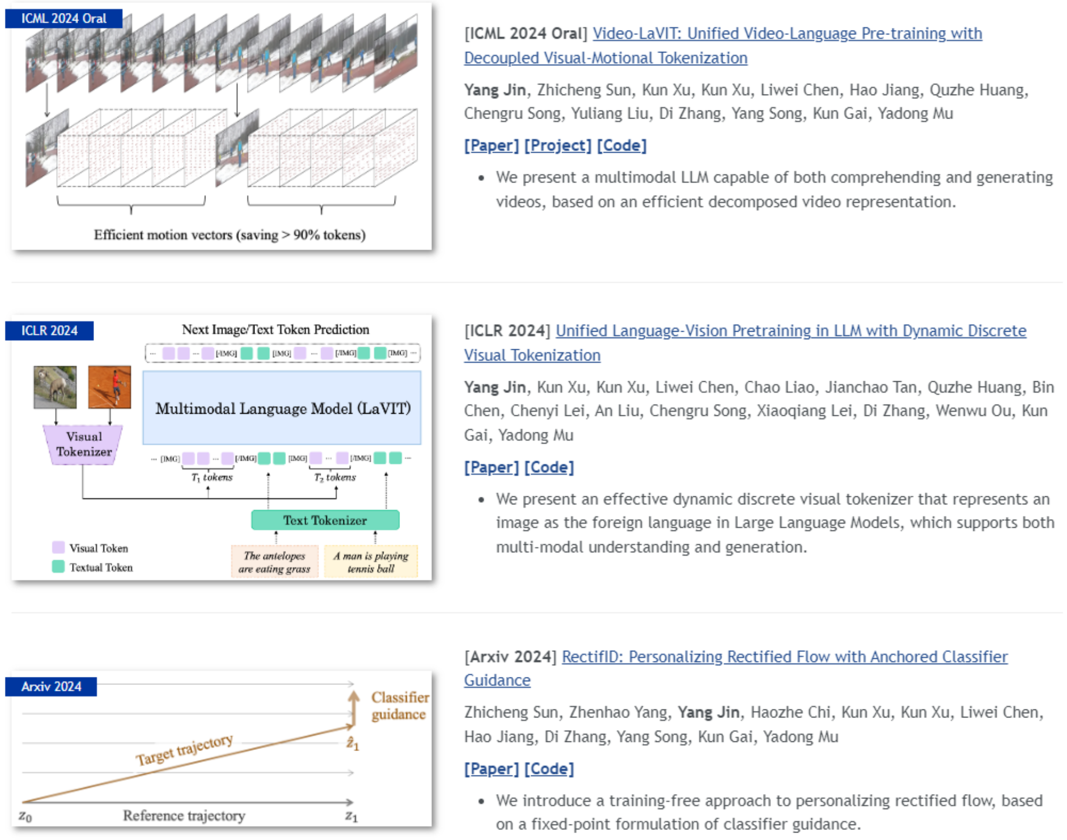
`jiny@stu.pku.edu.cn`, `{sunzc,myd}@pku.edu.cn`



Yadong Mu
Wangxuan Institute of Computer Technology
No. 128, Zhong-Guan-Cun North Street
Peking University, Beijing 100080, China
E-mail: myd AT pku.edu.cn OR muyadong AT gmail DOT com

Ph.D. / Master Students

- Jin Yang
- Liu Hanwen
- Wang Xinghan
- Xiao Yinan
- Yan Hongyu
- Chi Haozhe
- Li Jinghan
- Sun Zhicheng
- Jiang Hao
- Lin Chenguo
- Ma Kangqi
- Xu Peiran
- Mu Yanchen
- Cui Haoyang



[ICML 2024 Oral] Video-LaVIT: Unified Video-Language Pre-training with Decoupled Visual-Motional Tokenization

Yang Jin, Zhicheng Sun, Kun Xu, Kun Xu, Liwei Chen, Hao Jiang, Quzhe Huang, Chengru Song, Yuliang Liu, Di Zhang, Yang Song, Kun Gai, Yadong Mu

[Paper] [Project] [Code]

- We present a multimodal LLM capable of both comprehending and generating videos, based on an efficient decomposed video representation.

ICLR 2024
Next Image/Text Token Prediction
Multimodal Language Model (LaVIT)
Visual Tokenizer
T_1 tokens
T_2 tokens
Text Tokenizer
Visual Token
Textual Token
The antelopes are eating grass
A man is playing tennis ball

[ICLR 2024] Unified Language-Vision Pretraining in LLM with Dynamic Discrete Visual Tokenization

Yang Jin, Kun Xu, Kun Xu, Liwei Chen, Chao Liao, Jianchao Tan, Quzhe Huang, Bin Chen, Chenyi Lei, An Liu, Chengru Song, Xiaoqiang Lei, Di Zhang, Wenwu Ou, Kun Gai, Yadong Mu

[Paper] [Code]

- We present an effective dynamic discrete visual tokenizer that represents an image as the foreign language in Large Language Models, which supports both multi-modal understanding and generation.

[Arxiv 2024] RectifID: Personalizing Rectified Flow with Anchored Classifier Guidance

Zhicheng Sun, Zhenhao Yang, **Yang Jin**, Haozhe Chi, Kun Xu, Kun Xu, Liwei Chen, Hao Jiang, Di Zhang, Yang Song, Kun Gai, Yadong Mu

[Paper] [Code]

- We introduce a training-free approach to personalizing rectified flow, based on a fixed-point formulation of classifier guidance.

Yang Jin

Peking University

I am Yang Jin (金阳), a Ph.D. student at Wangxuan Institute of Computer Technology (WICT) in Peking University, advised by Prof. Yadong Mu. Before that, I obtained my B.S. degree in Computer Science and Engineering from Beihang University. I have been a research intern at ByteDance and Kuaishou Technology.

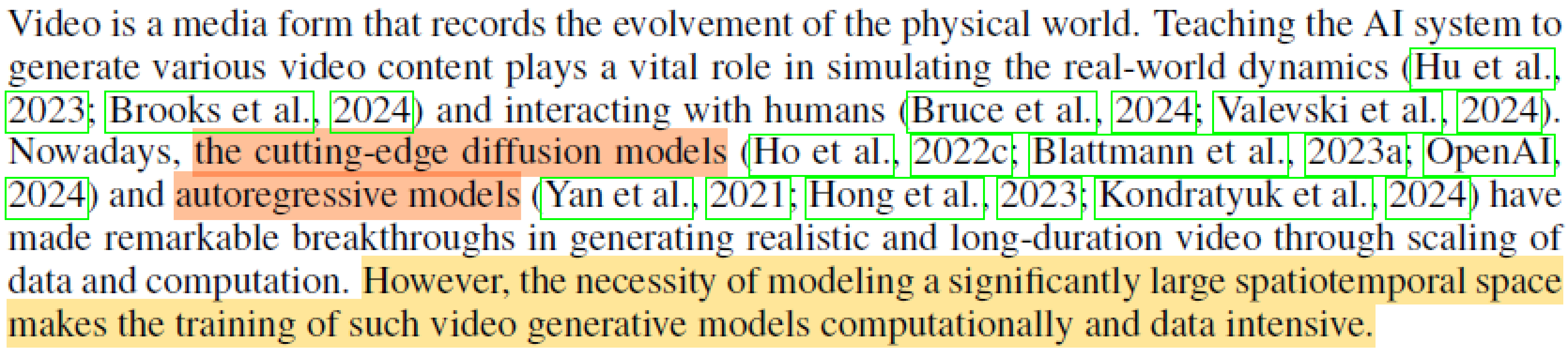
My research interests cover (1) **Multi-modal Large Language Model** and large-scale vision-language pre-training (2) **AIGC** including image, video, and personalized generation with both diffusion and autoregressive models (3) **Video Understanding** covers both recognition and localization tasks. I have published several papers and been reviewers at many conferences such as CVPR, ECCV, ICCV. If you are interested in my research, feel free to contact me through e-mail.

https://github.com/jy0205
https://jy0205.github.io/

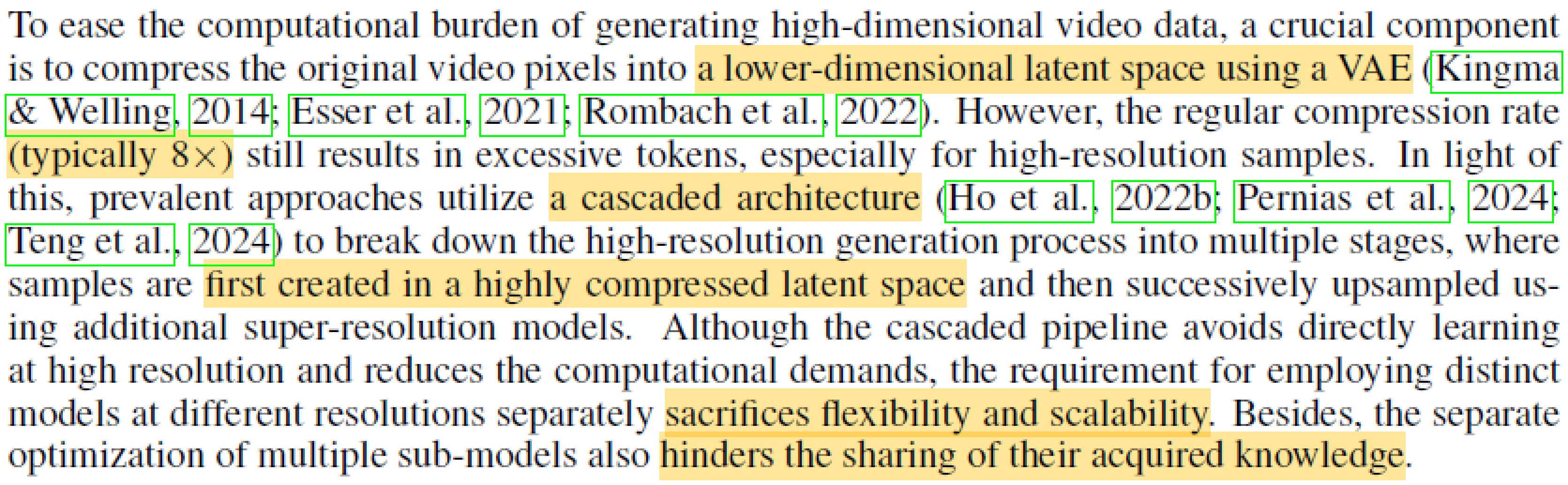
ABSTRACT

Video generation requires modeling a vast spatiotemporal space, which demands significant computational resources and data usage. To reduce the complexity, the prevailing approaches employ a cascaded architecture to avoid direct training with full resolution. Despite reducing computational demands, the separate optimization of each sub-stage hinders knowledge sharing and sacrifices flexibility. This work introduces a unified pyramidal flow matching algorithm. It reinterprets the original denoising trajectory as a series of pyramid stages, where only the final stage operates at the full resolution, thereby enabling more efficient video generative modeling. Through our sophisticated design, the flows of different pyramid stages can be interlinked to maintain continuity. Moreover, we craft autoregressive video generation with a temporal pyramid to compress the full-resolution history. The entire framework can be optimized in an end-to-end manner and with a single unified Diffusion Transformer (DiT). Extensive experiments demonstrate that our method supports generating high-quality 5-second (up to 10-second) videos at 768p resolution and 24 FPS within 20.7k A100 GPU training hours. All code and models will be open-sourced at `https://pyramid-flow.github.io`.

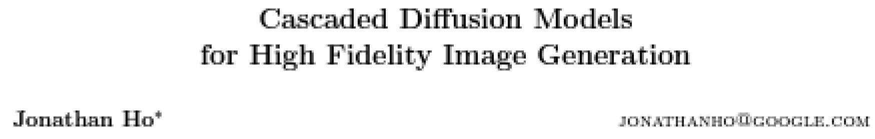
1 INTRODUCTION



Video is a media form that records the evolvement of the physical world. Teaching the AI system to generate various video content plays a vital role in simulating the real-world dynamics (Hu et al., 2023; Brooks et al., 2024) and interacting with humans (Bruce et al., 2024; Valevski et al., 2024). Nowadays, the cutting-edge diffusion models (Ho et al., 2022c; Blattmann et al., 2023a; OpenAI, 2024) and autoregressive models (Yan et al., 2021; Hong et al., 2023; Kondratyuk et al., 2024) have made remarkable breakthroughs in generating realistic and long-duration video through scaling of data and computation. However, the necessity of modeling a significantly large spatiotemporal space makes the training of such video generative models computationally and data intensive.

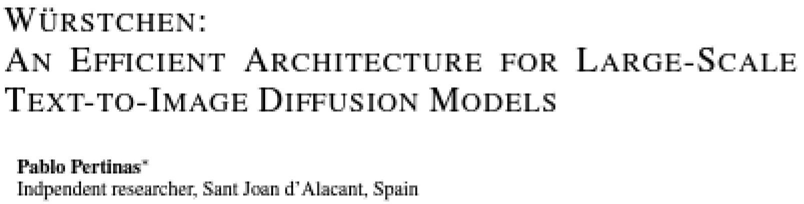


To ease the computational burden of generating high-dimensional video data, a crucial component is to compress the original video pixels into a lower-dimensional latent space using a VAE (Kingma & Welling, 2014; Esser et al., 2021; Rombach et al., 2022). However, the regular compression rate (typically 8×) still results in excessive tokens, especially for high-resolution samples. In light of this, prevalent approaches utilize a cascaded architecture (Ho et al., 2022b; Pernias et al., 2024; Teng et al., 2024) to break down the high-resolution generation process into multiple stages, where samples are first created in a highly compressed latent space and then successively upsampled using additional super-resolution models. Although the cascaded pipeline avoids directly learning at high resolution and reduces the computational demands, the requirement for employing distinct models at different resolutions separately sacrifices flexibility and scalability. Besides, the separate optimization of multiple sub-models also hinders the sharing of their acquired knowledge.



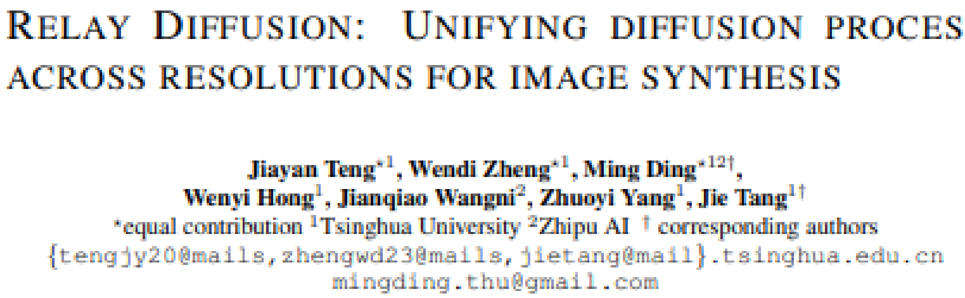
Cascaded Diffusion Models
for High Fidelity Image Generation

Jonathan Ho* JONATHANHO@GOOGLE.COM



WÜRSTCHEN:
AN EFFICIENT ARCHITECTURE FOR LARGE-SCALE
TEXT-TO-IMAGE DIFFUSION MODELS

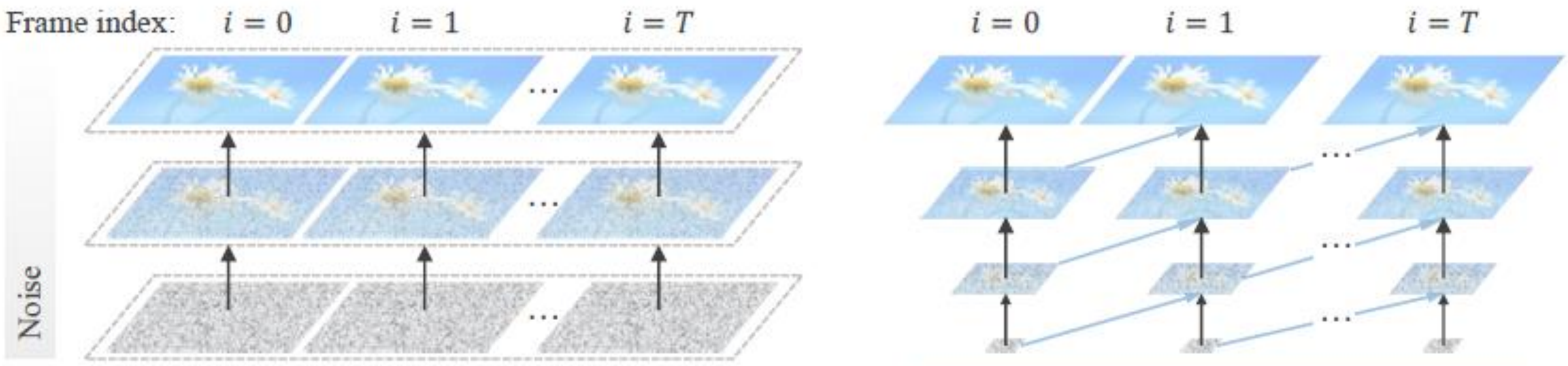
Pablo Pertinas*
Indpendent researcher, Sant Joan d'Alacant, Spain



RELAY DIFFUSION: UNIFYING DIFFUSION PROCESS
ACROSS RESOLUTIONS FOR IMAGE SYNTHESIS

Jiayan Teng[*1], Wendi Zheng[*1], Ming Ding[*12†],
Wenyi Hong[1], Jianqiao Wangni[2], Zhuoyi Yang[1], Jie Tang[1†]
[*]equal contribution [1]Tsinghua University [2]Zhipu AI [†] corresponding authors
{tengjy20@mails,zhengwd23@mails,jietang@mail}.tsinghua.edu.cn
mingding.thu@gmail.com

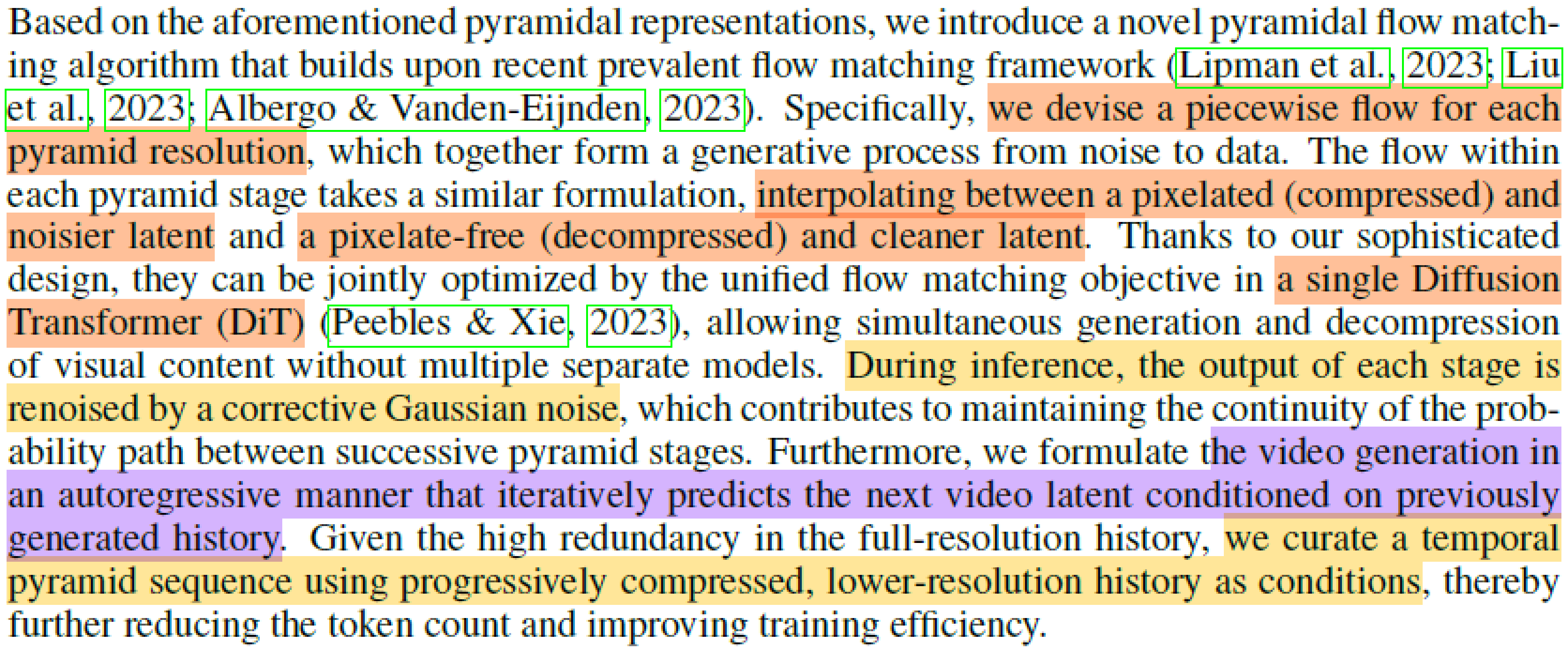
This work presents an efficient video generative modeling framework that transcends the limitations of the previous cascaded approaches. Our motivation stems from the observation in Fig. 1a that the initial timesteps in diffusion models are quite noisy and uninformative. This suggests that operating at full resolution throughout the entire generation trajectory may not be necessary. To this end, we reinterpret the original generation trajectory as a series of pyramid stages that operate on compressed representations of different scales. Notably, the efficacy of image pyramids (Adelson et al., 1984) has been widely validated for discriminative neural networks (Lin et al., 2017; Wang et al., 2020) and more recently for diffusion models (Ho et al., 2022b; Pernias et al., 2024; Teng et al., 2024)and multimodal LLMs (Yu et al., 2023; Tian et al., 2024). Here, we investigate two types of pyramids: the spatial pyramid within a frame and the temporal one between consecutive frames (as illustrated in Fig. 1b). In such a pyramidal generation trajectory, only the final stage operates at full resolution, drastically reducing redundant computations in earlier timesteps. The main advantages are twofold: (1) The generation trajectories of different pyramid stages are interlinked, with the subsequent stage continuing to generate from the previous ones. This eliminates the need for each stage to regenerate from pure noise in some cascade models. (2) Instead of relying on separate models for each image pyramid, we integrate them into a single unified model for end-to-end optimization, which admits drastically-expedited training with more elegant implementation as validated by experiments.



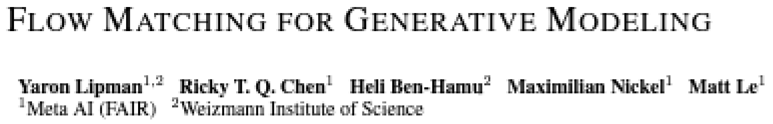
(a) Video diffusion model like Sora (OpenAI, 2024)

(b) Our proposed pyramidal flow matching

Figure 1: A motivating example for pyramidal flow matching: (a) Existing diffusion models operate at full resolution, spending a lot of computation on very noisy latents. (b) Our method harnesses the flexibility of flow matching to interpolate between latents of different resolutions. This allows for simultaneous generation and decompression of visual content with better computational efficiency. Note that the black arrows indicate the denoising trajectory, and the blue ones are temporal condition.

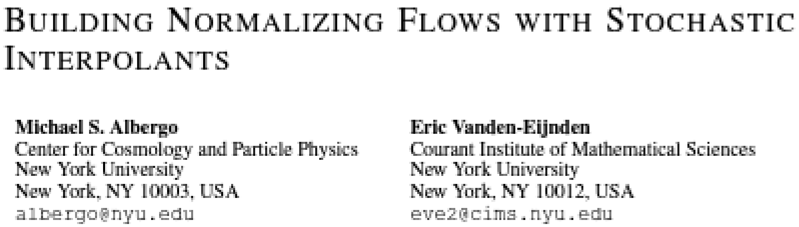


Based on the aforementioned pyramidal representations, we introduce a novel pyramidal flow matching algorithm that builds upon recent prevalent flow matching framework (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023). Specifically, we devise a piecewise flow for each pyramid resolution, which together form a generative process from noise to data. The flow within each pyramid stage takes a similar formulation, interpolating between a pixelated (compressed) and noisier latent and a pixelate-free (decompressed) and cleaner latent. Thanks to our sophisticated design, they can be jointly optimized by the unified flow matching objective in a single Diffusion Transformer (DiT) (Peebles & Xie, 2023), allowing simultaneous generation and decompression of visual content without multiple separate models. During inference, the output of each stage is renoised by a corrective Gaussian noise, which contributes to maintaining the continuity of the probability path between successive pyramid stages. Furthermore, we formulate the video generation in an autoregressive manner that iteratively predicts the next video latent conditioned on previously generated history. Given the high redundancy in the full-resolution history, we curate a temporal pyramid sequence using progressively compressed, lower-resolution history as conditions, thereby further reducing the token count and improving training efficiency.



FLOW MATCHING FOR GENERATIVE MODELING

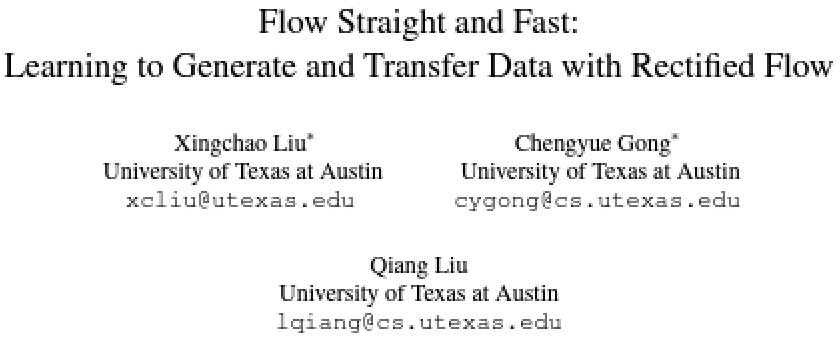
Yaron Lipman[1,2] Ricky T. Q. Chen[1] Heli Ben-Hamu[2] Maximilian Nickel[1] Matt Le[1]
[1]Meta AI (FAIR) [2]Weizmann Institute of Science



BUILDING NORMALIZING FLOWS WITH STOCHASTIC INTERPOLANTS

Michael S. Albergo
Center for Cosmology and Particle Physics
New York University
New York, NY 10003, USA
albergo@nyu.edu

Eric Vanden-Eijnden
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA
eve2@cims.nyu.edu



Flow Straight and Fast:
Learning to Generate and Transfer Data with Rectified Flow

Xingchao Liu*
University of Texas at Austin
xcliu@utexas.edu

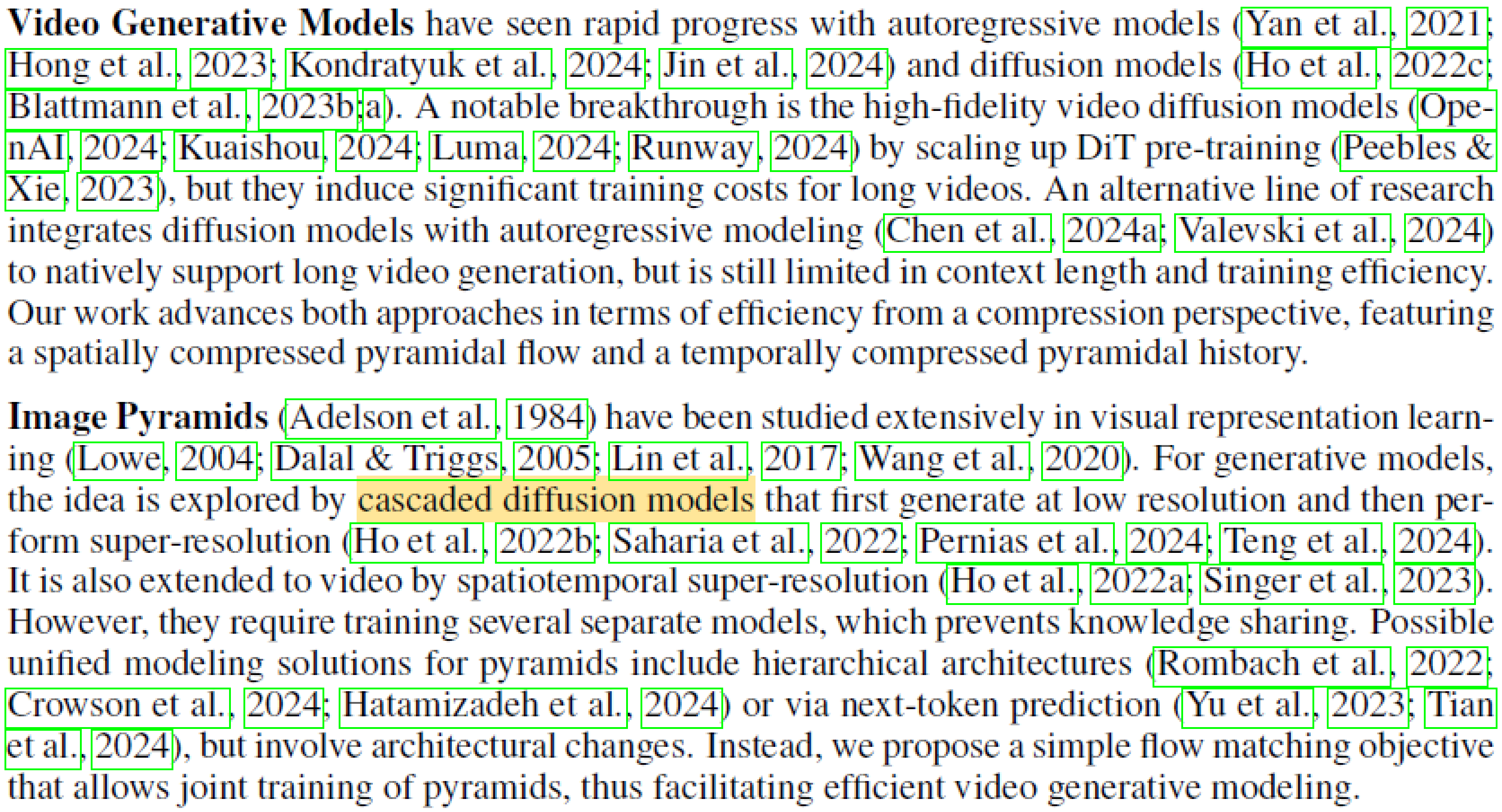
Chengyue Gong*
University of Texas at Austin
cygong@cs.utexas.edu

Qiang Liu
University of Texas at Austin
lqiang@cs.utexas.edu

The collaboration of the spatial and temporal pyramids results in remarkable training efficiency for video generation. Compared to the commonly used full-sequence diffusion, our method significantly reduces the number of video tokens during training (e.g., $\leq$15,360 tokens versus 119,040 tokens for a 10-second, 241-frame video), thereby reducing both computational resources required and training time. By training only on open-source datasets, our model generate high-quality 10-second videos at 768p resolution and 24 fps. The core contributions of this paper are summarized as follows:

- We present pyramidal flow matching, a novel video generative modeling algorithm that incorporates both spatial and temporal pyramid representations. Utilizing this framework can significantly improve training efficiency while maintaining good video generation quality.
- The proposed unified flow matching objective facilitates joint training of pyramid stages in a single Diffusion Transformer (DiT), avoiding the separate optimization of multiple models. The support for end-to-end training further enhances its simplicity and scalability.
- We evaluate its effectiveness on VBench (Huang et al., 2024) and EvalCrafter (Liu et al., 2024), with highly competitive performance among video generative models trained on public datasets.

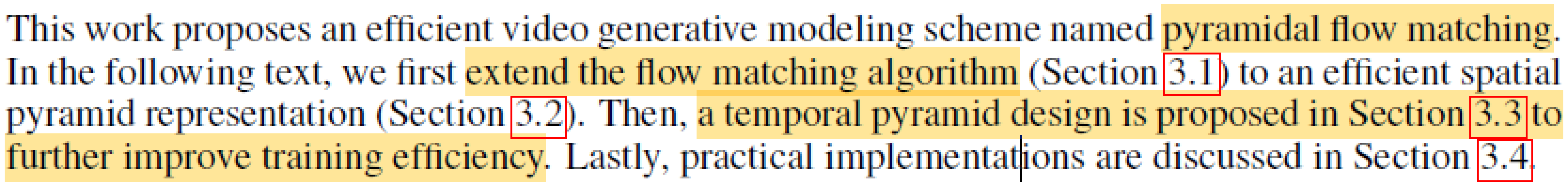
2 Related Work



Video Generative Models have seen rapid progress with autoregressive models (Yan et al., 2021; Hong et al., 2023; Kondratyuk et al., 2024; Jin et al., 2024) and diffusion models (Ho et al., 2022c; Blattmann et al., 2023b;a). A notable breakthrough is the high-fidelity video diffusion models (OpenAI, 2024; Kuaishou, 2024; Luma, 2024; Runway, 2024) by scaling up DiT pre-training (Peebles & Xie, 2023), but they induce significant training costs for long videos. An alternative line of research integrates diffusion models with autoregressive modeling (Chen et al., 2024a; Valevski et al., 2024) to natively support long video generation, but is still limited in context length and training efficiency. Our work advances both approaches in terms of efficiency from a compression perspective, featuring a spatially compressed pyramidal flow and a temporally compressed pyramidal history.

Image Pyramids (Adelson et al., 1984) have been studied extensively in visual representation learning (Lowe, 2004; Dalal & Triggs, 2005; Lin et al., 2017; Wang et al., 2020). For generative models, the idea is explored by cascaded diffusion models that first generate at low resolution and then perform super-resolution (Ho et al., 2022b; Saharia et al., 2022; Pernias et al., 2024; Teng et al., 2024). It is also extended to video by spatiotemporal super-resolution (Ho et al., 2022a; Singer et al., 2023). However, they require training several separate models, which prevents knowledge sharing. Possible unified modeling solutions for pyramids include hierarchical architectures (Rombach et al., 2022; Crowson et al., 2024; Hatamizadeh et al., 2024) or via next-token prediction (Yu et al., 2023; Tian et al., 2024), but involve architectural changes. Instead, we propose a simple flow matching objective that allows joint training of pyramids, thus facilitating efficient video generative modeling.

3 METHOD



This work proposes an efficient video generative modeling scheme named pyramidal flow matching. In the following text, we first extend the flow matching algorithm (Section 3.1) to an efficient spatial pyramid representation (Section 3.2). Then, a temporal pyramid design is proposed in Section 3.3 to further improve training efficiency. Lastly, practical implementations are discussed in Section 3.4.

3.1 Preliminaries on Flow Matching

Similar to diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020), flow generative models (Papamakarios et al., 2021; Song et al., 2021; Xu et al., 2022; Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023) aim to learn a velocity field $\boldsymbol{v}_t$ that maps random noise $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to data samples $\boldsymbol{x}_1 \sim q$ via an ordinary differential equation (ODE):

$$\frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{v}_t(\boldsymbol{x}_t). \tag{1}$$

Recently, Lipman et al. (2023); Liu et al. (2023); Albergo & Vanden-Eijnden (2023) proposed the flow matching framework, which provides a simple simulation-free training objective for flow generative models by directly regressing the velocity $\boldsymbol{v}_t$ on a conditional vector filed $\boldsymbol{u}_t(\cdot|\boldsymbol{x}_1)$:

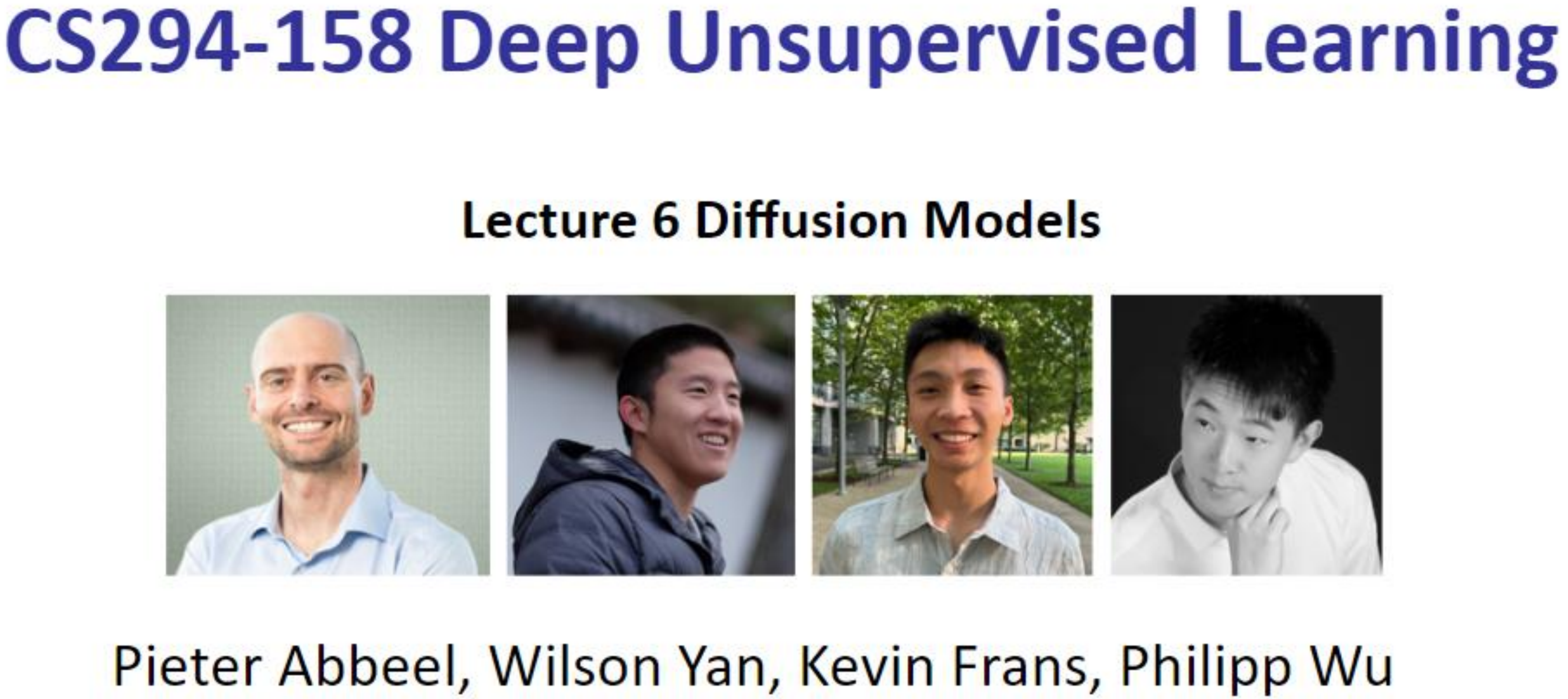
$$\mathbb{E}_{t,q(\boldsymbol{x}_1),p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)} \left|\left| \boldsymbol{v}_t(\boldsymbol{x}_t) - \boldsymbol{u}_t(\boldsymbol{x}_t|\boldsymbol{x}_1) \right|\right|^2, \tag{2}$$

where $\boldsymbol{u}_t(\cdot|\boldsymbol{x}_1)$ uniquely determines a conditional probability path $p_t(\cdot|\boldsymbol{x}_1)$ toward data sample $\boldsymbol{x}_1$. An effective choice of the conditional probability path is linear interpolation of data and noise:

$$\boldsymbol{x}_t = t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_0, \tag{3}$$

$$\boldsymbol{x}_t \sim \mathcal{N}(t\boldsymbol{x}_1, (1-t)^2\boldsymbol{I}), \tag{4}$$

and $\boldsymbol{u}(\boldsymbol{x}_t|\boldsymbol{x}_1) = \boldsymbol{x}_1 - \boldsymbol{x}_0$. Notably, flow matching can be flexibly extended to interpolate between distributions other than standard Gaussians. This enables us to devise a sophisticated flow matching algorithm that specializes in reducing the computational cost of video generative modeling.

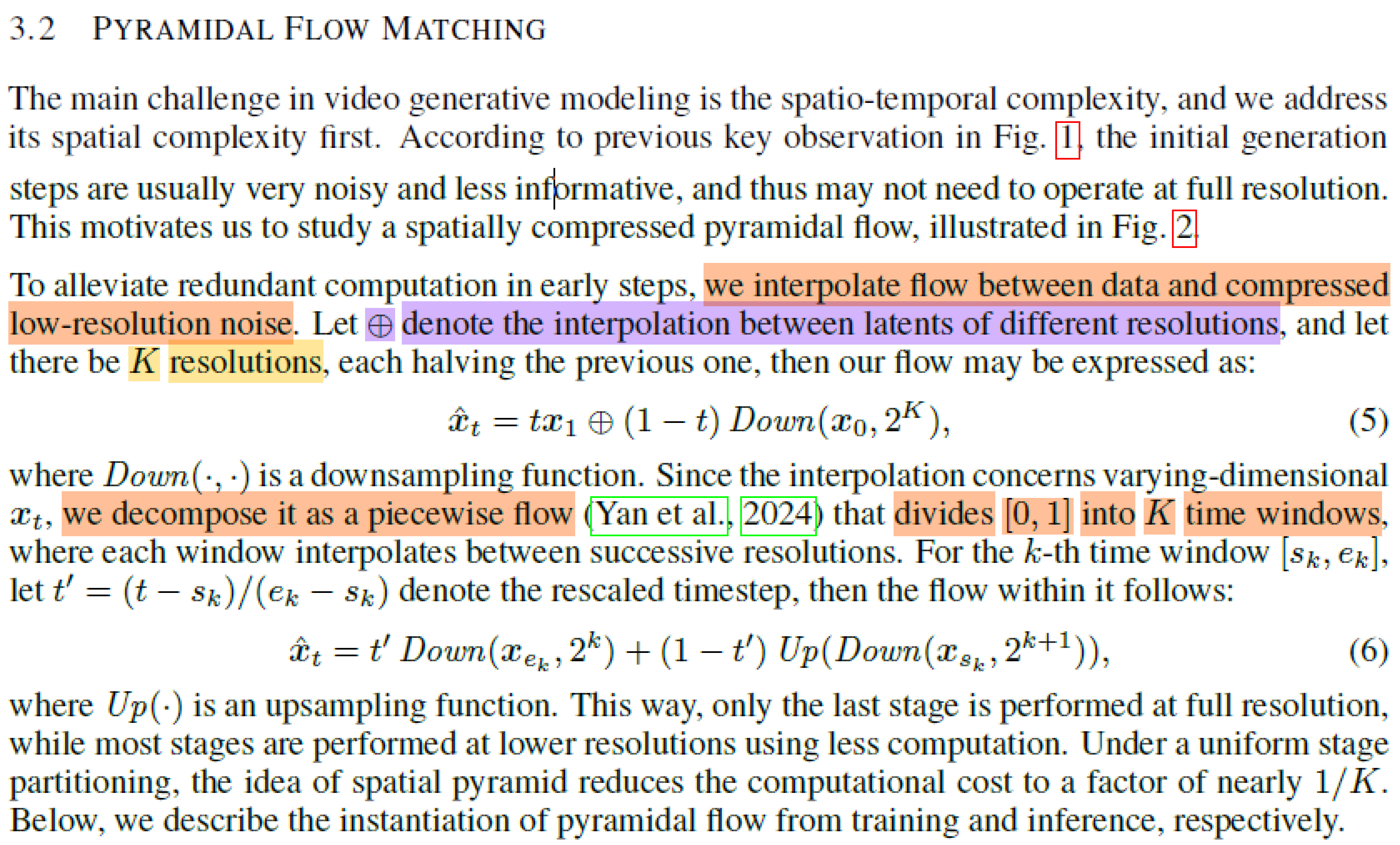


CS294-158 Deep Unsupervised Learning

Lecture 6 Diffusion Models

Pieter Abbeel, Wilson Yan, Kevin Frans, Philipp Wu

https://docs.google.com/presentation/d/18Tkjp6sExgbLF2Fz7PW9gMbUxHrybFY5pEJkhiraLxU/edit#slide=id.g2bb12cd8dfb_0_702



3.2 Pyramidal Flow Matching

The main challenge in video generative modeling is the spatio-temporal complexity, and we address its spatial complexity first. According to previous key observation in Fig. 1, the initial generation steps are usually very noisy and less informative, and thus may not need to operate at full resolution. This motivates us to study a spatially compressed pyramidal flow, illustrated in Fig. 2.

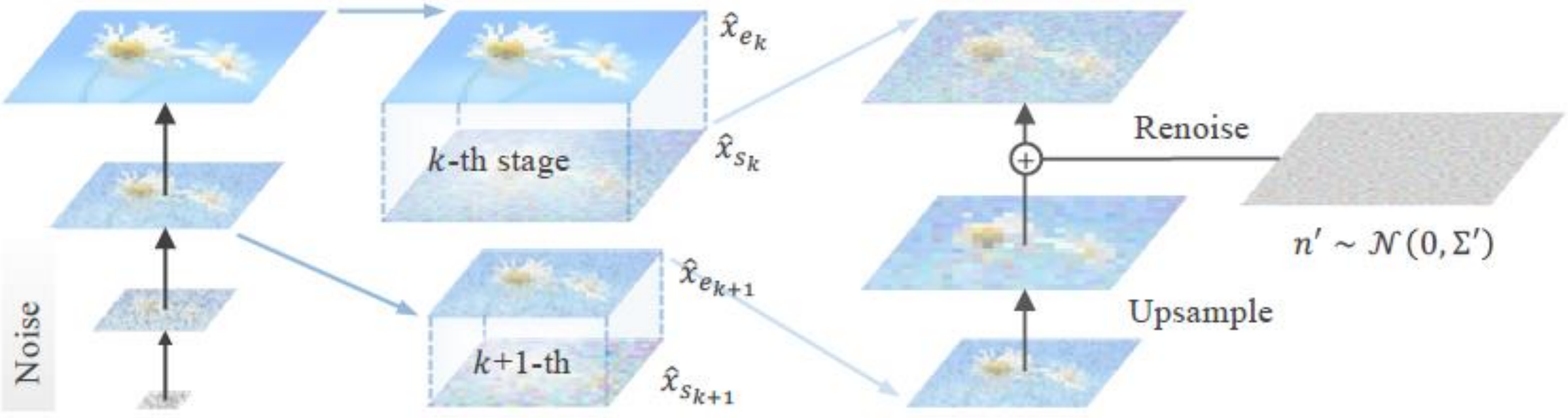
To alleviate redundant computation in early steps, we interpolate flow between data and compressed low-resolution noise. Let $\oplus$ denote the interpolation between latents of different resolutions, and let there be K resolutions, each halving the previous one, then our flow may be expressed as:

$$\hat{\boldsymbol{x}}_t = t\boldsymbol{x}_1 \oplus (1-t)\, Down(\boldsymbol{x}_0, 2^K), \tag{5}$$

where $Down(\cdot,\cdot)$ is a downsampling function. Since the interpolation concerns varying-dimensional $\boldsymbol{x}_t$, we decompose it as a piecewise flow (Yan et al., 2024) that divides $[0,1]$ into K time windows, where each window interpolates between successive resolutions. For the k-th time window $[s_k, e_k]$, let $t' = (t - s_k)/(e_k - s_k)$ denote the rescaled timestep, then the flow within it follows:

$$\hat{\boldsymbol{x}}_t = t'\, Down(\boldsymbol{x}_{e_k}, 2^k) + (1-t')\, Up(Down(\boldsymbol{x}_{s_k}, 2^{k+1})), \tag{6}$$

where $Up(\cdot)$ is an upsampling function. This way, only the last stage is performed at full resolution, while most stages are performed at lower resolutions using less computation. Under a uniform stage partitioning, the idea of spatial pyramid reduces the computational cost to a factor of nearly $1/K$. Below, we describe the instantiation of pyramidal flow from training and inference, respectively.



(a) Unified modeling of pyramidal flow

(b) Handling jump points via renoising

Figure 2: Illustration of spatial pyramid. (a) The pyramidal flow is divided into multiple stages, each from a pixelated and noisy starting point to a pixelate-free and cleaner result. (b) During inference, we add a corrective noise at jump points across stages to ensure continuity of the proabability path.

3.2.1 Unified Training

In the construction of pyramidal flow, our main concern is unified modeling of different stages, as previous works (Ho et al., 2022b; Pernias et al., 2024; Teng et al., 2024) all require training multiple models for separate generation and super-resolution, which hinders knowledge sharing.

To unify the objectives of generation and decompression/super-resolution, we curate the probability path by interpolating between different noise levels and resolutions. It starts with a more noisy and pixelated latent upsampled from a lower resolution, and yields cleaner and fine-grained results at a higher resolution, as illustrated in Fig. 2a. Formally, the conditional probability path is defined by:

$$\text{End:} \qquad \hat{\boldsymbol{x}}_{e_k} \sim \mathcal{N}(e_k\, Down(\boldsymbol{x}_1, 2^k), (1-e_k)^2 \boldsymbol{I}), \tag{7}$$

$$\text{Start:} \qquad \hat{\boldsymbol{x}}_{s_k} \sim \mathcal{N}(s_k\, Up(Down(\boldsymbol{x}_1, 2^{k+1})), (1-s_k)^2 \boldsymbol{I}), \tag{8}$$

where $s_k < e_k$, and the upsampling and downsampling functions for the clean $\boldsymbol{x}_1$ are well defined, *e.g.*, by nearest or bilinear resampling. In addition, to enhance the straightness of the flow trajectory, we couple the sampling of its endpoints by enforcing the noise to be in the same direction. Namely, we first sample a noise $\boldsymbol{n} \sim \mathcal{N}(0, \boldsymbol{I})$ and then jointly compute the endpoints $(\hat{\boldsymbol{x}}_{e_k}, \hat{\boldsymbol{x}}_{s_k})$ as:

$$\text{End:} \qquad \hat{\boldsymbol{x}}_{e_k} = e_k\, Down(\boldsymbol{x}_1, 2^k) + (1-e_k)\boldsymbol{n}, \tag{9}$$

$$\text{Start:} \qquad \hat{\boldsymbol{x}}_{s_k} = s_k\, Up(Down(\boldsymbol{x}_1, 2^{k+1})) + (1-s_k)\boldsymbol{n}. \tag{10}$$

Thereafter, we can regress the flow model $\boldsymbol{v}_t$ on the conditional vector field $\boldsymbol{u}_t(\hat{\boldsymbol{x}}_t|\boldsymbol{x}_1) = \hat{\boldsymbol{x}}_{e_k} - \hat{\boldsymbol{x}}_{s_k}$ with the following flow matching objective to unify generation and decompression:

$$\mathbb{E}_{k,t,(\hat{\boldsymbol{x}}_{e_k}, \hat{\boldsymbol{x}}_{s_k})} \left\| \boldsymbol{v}_t(\hat{\boldsymbol{x}}_t) - (\hat{\boldsymbol{x}}_{e_k} - \hat{\boldsymbol{x}}_{s_k}) \right\|^2. \tag{11}$$

3.2.2 Inference with Renoising

During inference, standard sampling algorithms can be applied within each pyramid stage. However, we must carefully handle the jump points (Campbell et al., 2023) between successive pyramid stages of different resolutions to ensure continuity of the probability path.

To ensure continuity, we first upsample the previous low-resolution endpoint with nearest or bilinear resampling. The result, as a linear combination of the input, follows a Gaussian distribution:

$$Up(\hat{\boldsymbol{x}}_{e_{k+1}}) \sim \mathcal{N}(e_{k+1}\, Up(Down(\boldsymbol{x}_1, 2^{k+1})), (1-e_{k+1})^2\, \boldsymbol{\Sigma}), \tag{12}$$

where $\boldsymbol{\Sigma}$ is a covariance matrix depending on the upsampling function. Comparing Eqs. (8) and (12), we find it possible to match the Gaussian distributions at each jump point by a linear transformation of the upsampled result. Specifically, the following rescaling and renoising scheme would suffice:

$$\hat{\boldsymbol{x}}_{s_k} = \frac{s_k}{e_{k+1}}\, Up(\hat{\boldsymbol{x}}_{e_{k+1}}) + \alpha \boldsymbol{n}', \quad \text{s.t. } \boldsymbol{n}' \sim \mathcal{N}(0, \boldsymbol{\Sigma}'), \tag{13}$$

where the rescaling coefficient s_k/e_{k+1} allows matching the means of these distributions, and the corrective noise $\boldsymbol{n}'$ with a weight of α allows matching their covariance matrices.

To derive the corrective noise and its covariance, we consider a simplest scenario with nearest neighbor upsampling. In this case, $\boldsymbol{\Sigma}$ has a blockwise structure with non-zero elements only in the 4×4 blocks along the diagonal (corresponding to those upsampled from the same pixel). Then, it can be inferred that the corrective noise's covariance matrix $\boldsymbol{\Sigma}'$ also has a blockwise structure:

$$\boldsymbol{\Sigma}_{block} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \Rightarrow \boldsymbol{\Sigma}'_{block} = \begin{pmatrix} 1 & \gamma & \gamma & \gamma \\ \gamma & 1 & \gamma & \gamma \\ \gamma & \gamma & 1 & \gamma \\ \gamma & \gamma & \gamma & 1 \end{pmatrix}, \tag{14}$$

where $\boldsymbol{\Sigma}'_{block}$ contains negative elements $\gamma \in [-1/3, 0]$[1] to reduce the correlation within each block, as illustrated in Fig. 2b. Since it is desirable to maximally preserve the signals at each jump point, we opt to add a small amount of noise with $\gamma = -1/3$ such that it is most specialized for decorrelation. Substituting this into the above gives the update rule at jump points (see Appendix A for derivations):

$$\hat{\boldsymbol{x}}_{s_k} = \frac{1+s_k}{2}\, Up(\hat{\boldsymbol{x}}_{e_{k+1}}) + \frac{\sqrt{3}(1-s_k)}{2} \boldsymbol{n}', \tag{15}$$

with $e_{k+1} = 2s_k/(1+s_k)$. The resulting inference process with renoising is shown in Algorithm 1.

$$\begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix}$$

$$\begin{pmatrix} A & P_1 & B & P_2 & C \\ P_3 & P_4 & P_5 & P_6 & P_7 \\ D & P_8 & E & P_9 & F \\ P_{10} & P_{11} & P_{12} & P_{13} & P_{14} \\ G & P_{15} & H & P_{16} & I \end{pmatrix} \quad \begin{pmatrix} A & A & B & B & C & C \\ A & A & B & B & C & C \\ D & D & E & E & F & F \\ D & D & E & E & F & F \\ G & G & H & H & I & I \\ G & G & H & H & I & I \end{pmatrix}$$

$$P_4 = A \cdot 0.25 + B \cdot 0.25 + D \cdot 0.25 + E \cdot 0.25$$

Step 2: Eigenvalue Equation

To find the eigenvalues λ, we use the characteristic equation:

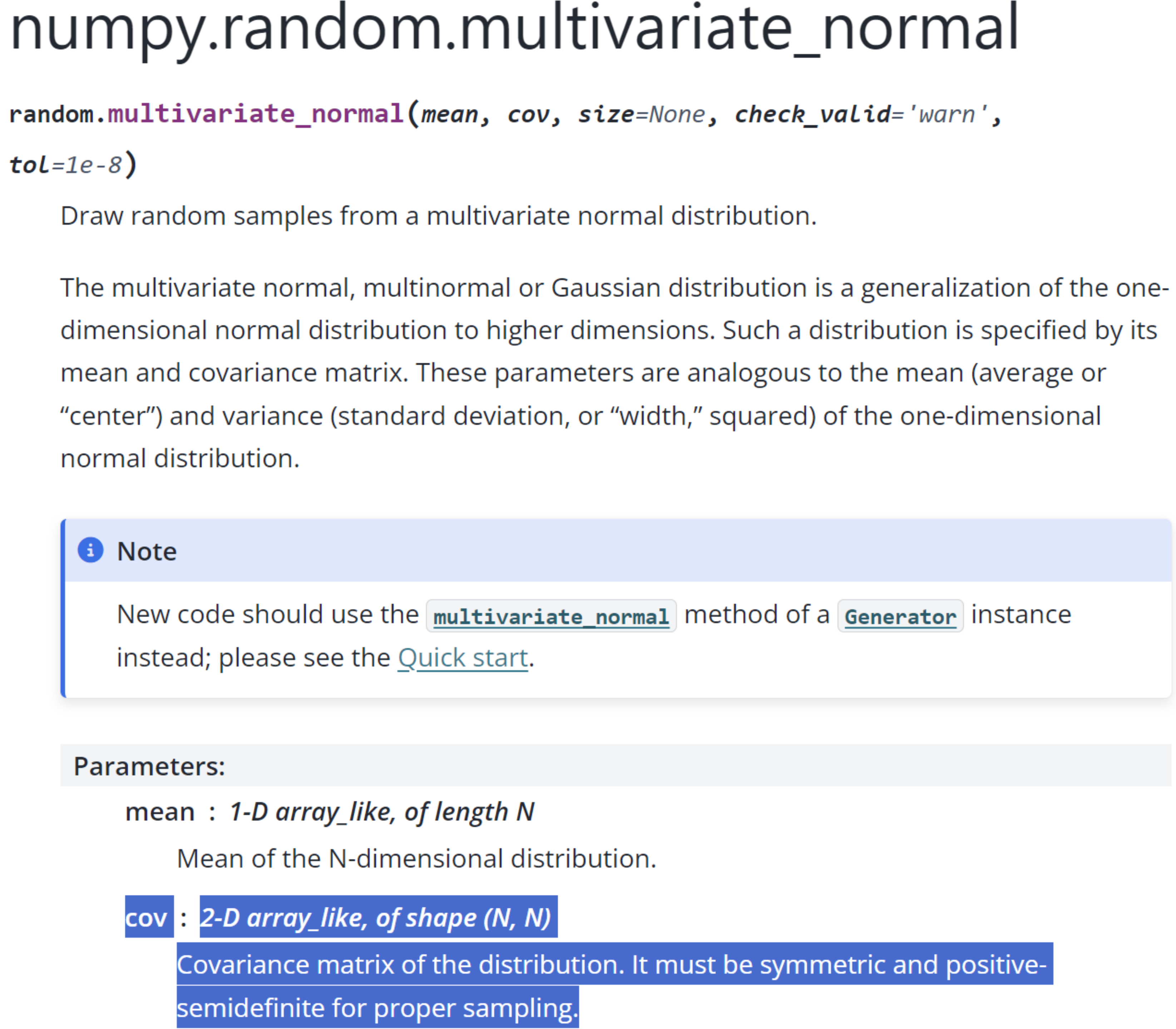
$$\det(\Sigma_{\text{block}} - \lambda I) = 0$$

where I is the identity matrix, and λ is an eigenvalue. So we need to solve the determinant of the matrix $\Sigma_{\text{block}} - \lambda I$, which is:

$$\Sigma_{\text{block}} - \lambda I = \begin{pmatrix} 1-\lambda & \gamma & \gamma & \gamma \\ \gamma & 1-\lambda & \gamma & \gamma \\ \gamma & \gamma & 1-\lambda & \gamma \\ \gamma & \gamma & \gamma & 1-\lambda \end{pmatrix}$$

Final Determinant Formula:

$$\boxed{\det(\Sigma_{\text{block}}) = (1+3\gamma)(1-\gamma)^3}$$



numpy.random.multivariate_normal

```
random.multivariate_normal(mean, cov, size=None, check_valid='warn',
tol=1e-8)
```

Draw random samples from a multivariate normal distribution.

The multivariate normal, multinormal or Gaussian distribution is a generalization of the one-dimensional normal distribution to higher dimensions. Such a distribution is specified by its mean and covariance matrix. These parameters are analogous to the mean (average or “center”) and variance (standard deviation, or “width,” squared) of the one-dimensional normal distribution.

Note

New code should use the `multivariate_normal` method of a `Generator` instance instead; please see the Quick start.

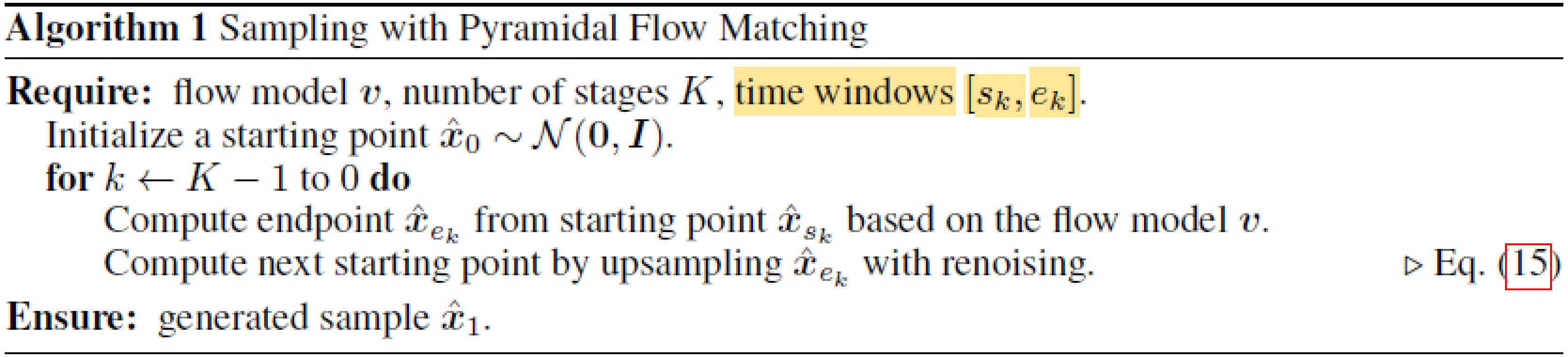
Parameters:

mean : *1-D array_like, of length N*

Mean of the N-dimensional distribution.

cov : *2-D array_like, of shape (N, N)*

Covariance matrix of the distribution. It must be symmetric and positive-semidefinite for proper sampling.



Algorithm 1 Sampling with Pyramidal Flow Matching

```
Require: flow model $v$, number of stages $K$, time windows $[s_k, e_k]$.
  Initialize a starting point $\hat{\boldsymbol{x}}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
  for $k \leftarrow K - 1$ to $0$ do
      Compute endpoint $\hat{\boldsymbol{x}}_{e_k}$ from starting point $\hat{\boldsymbol{x}}_{s_k}$ based on the flow model $v$.
      Compute next starting point by upsampling $\hat{\boldsymbol{x}}_{e_k}$ with renoising.          ▷ Eq. (15)
Ensure: generated sample $\hat{\boldsymbol{x}}_1$.
```

3.3 Pyramidal Temporal Condition

Beyond the spatial complexity addressed in above sections, video presents another significant challenge due to its temporal length. The prevailing full-sequence diffusion methods generate all video frames simultaneously, restricting them to fixed-length generation (consistent with training). In contrast, the autoregressive video generation paradigm supports flexible-length generation during inference. Recent advancements (Chen et al., 2024a; Valevski et al., 2024) have also demonstrated its effectiveness in creating long-duration video content. However, their training is still severely limited by the computational complexity arising from the full-resolution long-history condition.

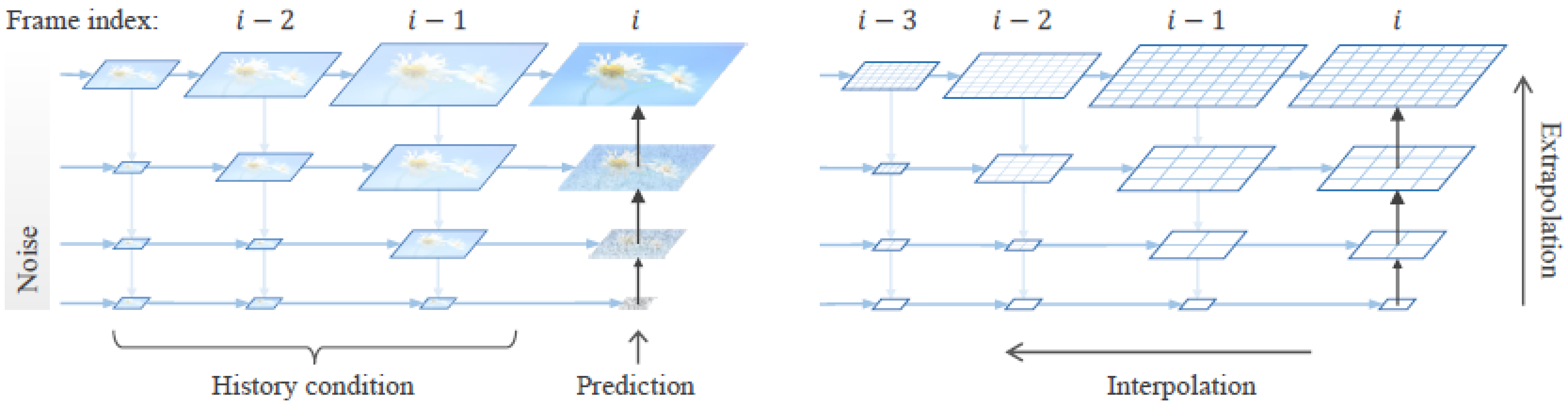
We observe that there is a high redundancy in full-resolution history conditions. For example, earlier frames in a video tend to provide high-level semantic conditions and are less related to appearance details. This motivates us to use compressed, lower-resolution history for autoregressive video generation. As shown in Fig. 3a, we adopt a history condition of gradually increasing resolutions:

$$\underbrace{\ldots \to Down(x_{t'}^{i-2}, 2^{k+1}) \to Down(x_{t'}^{i-1}, 2^{k})}_{\text{History condition}} \to \underset{\text{Training}}{\underset{\uparrow}{\hat{x}_t^i}} \quad , \tag{16}$$

where the superscripts are the history latent index, and the subscript t' indicates small noise added to history latents in training to mitigate error accumulation with autoregressive generation, as in (Chen et al., 2024a; Valevski et al., 2024). After training, we use clean generated frames for inference:

$$\underbrace{\ldots \to Down(x_{1}^{i-2}, 2^{k+1}) \to Down(x_{1}^{i-1}, 2^{k})}_{\text{History condition}} \to \underset{\text{Prediction}}{\underset{\uparrow}{\hat{x}_t^i}} \quad . \tag{17}$$

The above design significantly reduces the computational and memory overhead of video generative pre-training. Let there be T history latents over K lower resolutions, then most frames are computed at the lowest resolution of $1/2^K$, which reduces the number of training tokens by up to $1/4^K$ times. As a result, training efficiency is improved by up to $16^K/T$ times.



(a) Temporal pyramids rearranged in rows

(b) Position encoding in spatial and temporal pyramid

Figure 3: Illustration of temporal pyramid. (a) At each pyramid stage, the generation is conditioned on a compressed, lower-resolution history to improve training efficiency of the autoregressive model, as indicated by the rows. (b) A compatible position encoding scheme is devised that extrapolates in the spatial pyramid but interpolates in the temporal pyramid to allow spatial alignment of conditions.

3.4 Practical Implementation

In this section, we show that the above pyramid designs can be easily implemented using standard Transformer architecture (Vaswani et al., 2017) and pipelines. This is crucial for efficient and scalable video generative pre-training based on existing acceleration frameworks.

Unlike previous methods (Ma et al., 2024) that utilize factorized spatial and temporal attention to reduce computational complexity, we directly employ full sequence attention, thanks to much fewer tokens required by our pyramidal representation. Furthermore, blockwise causal attention is adopted in each transformer layer, ensuring that each token cannot attend to its subsequent frames. The ablation results in Appendix C.2 illustrate that such casual attention design is crucial for autoregressive video generation. Another important design choice is the position encoding, as the pyramid designs introduce multiple spatial resolutions. As shown in Fig. 3b, we extrapolate position encoding in the spatial pyramid for better fine-grained detail (Yang et al., 2024), while interpolating it in the temporal pyramid input to spatially align the history conditions.

During training, different pyramidal stages are uniformly sampled in each update iteration. The autoregressive nature of our method inherently supports joint training of images and videos, since the first frame in a video acts as an image. We pack training samples with varying token counts together to form the length-balanced training batch following Patch n' Pack (Dehghani et al., 2023). After training, our method natively possesses the capability of text-to-video and text-conditioned image-to-video generation. During inference sampling, the classifier-free guidance strategy can be employed to enhance temporal consistency and motion smoothness of the generated video.

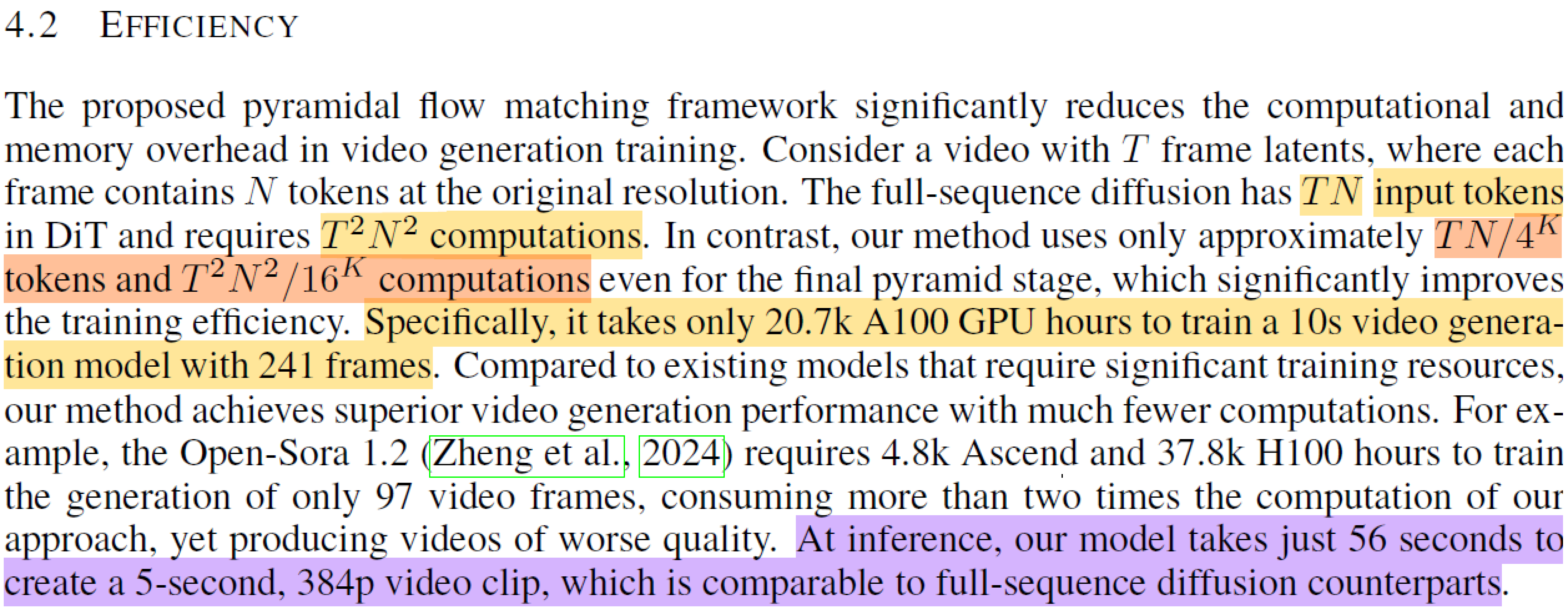
4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Training Dataset. Our model is trained on a mixed corpus of open-source image and video datasets. For images, we utilize a high-aesthetic subset of LAION-5B (Schuhmann et al., 2022), 11M from CC-12M (Changpinyo et al., 2021), 6.9M non-blurred subset of SA-1B (Kirillov et al., 2023), 4.4M from JourneyDB (Sun et al., 2023), and 14M publicly available synthetic data. For video data, we incorporate the WebVid-10M (Bain et al., 2021), OpenVid-1M (Nan et al., 2024), and another 1M high-resolution non-watermark video primarily from the Open-Sora Plan (PKU-Yuan Lab et al., 2024). After postprocessing, around 10M single-shot videos are available for training.

Evaluation Metrics. We utilize the VBench (Huang et al., 2024) and EvalCrafter (Liu et al., 2024) for quantitative performance evaluation. VBench is a comprehensive benchmark that includes 16 fine-grained dimensions to systematically measure both motion quality and semantic alignment of video generative models. EvalCrafter is another large-scale evaluation benchmark including around 17 objective metrics for assessing video generation capabilities. In addition to automated evaluation metrics, we also conducted a study with human participants to measure the human preference for our generated videos. The compared baselines are summarized in Appendix B.

Implementation Details. We utilize the prevailing MM-DiT architecture from SD3 Medium (Esser et al., 2024) as the base model, with 2B parameters in total. It employs sinusoidal position encoding (Vaswani et al., 2017) in the spatial dimensions. As for the temporal dimension, the 1D Rotary Position Embedding (RoPE) (Su et al., 2024) is added to support flexible training with different video durations. In addition, we use a 3D Variational Autoencoder (VAE) to compress videos both spatially and temporally with a downsampling ratio of $8 \times 8 \times 8$. It shares a similar structure with MAGVIT-v2 (Yu et al., 2024) and is trained from scratch on the WebVid-10M dataset (Bain et al., 2021). The number of pyramid stages is set to 3 in all the experiments. Following Valevski et al. (2024), we add some corruptive noise of strength uniformly sampled from $[0, 1/3]$ to the history pyramid conditions, which is critical for mitigating the autoregressive generation degradation.



4.2 EFFICIENCY

The proposed pyramidal flow matching framework significantly reduces the computational and memory overhead in video generation training. Consider a video with T frame latents, where each frame contains N tokens at the original resolution. The full-sequence diffusion has TN input tokens in DiT and requires T^2N^2 computations. In contrast, our method uses only approximately $TN/4^K$ tokens and $T^2N^2/16^K$ computations even for the final pyramid stage, which significantly improves the training efficiency. Specifically, it takes only 20.7k A100 GPU hours to train a 10s video generation model with 241 frames. Compared to existing models that require significant training resources, our method achieves superior video generation performance with much fewer computations. For example, the Open-Sora 1.2 (Zheng et al., 2024) requires 4.8k Ascend and 37.8k H100 hours to train the generation of only 97 video frames, consuming more than two times the computation of our approach, yet producing videos of worse quality. At inference, our model takes just 56 seconds to create a 5-second, 384p video clip, which is comparable to full-sequence diffusion counterparts.

Table 1: Experimental results on VBench (Huang et al., 2024). In terms of total score and quality score, our model even outperforms CogVideoX-5B (Yang et al., 2024) with twice the model size. In the following tables, we use blue to denote the highest scores among models trained on public data.

Model	Public Data	Total Score	Quality Score	Semantic Score	Motion Smoothness	Dynamic Degree
Gen-2	×	80.58	82.47	73.03	**99.58**	18.89
Pika 1.0	×	80.69	82.92	71.77	99.50	47.50
CogVideoX-2B	×	80.91	82.18	75.83	97.73	59.86
CogVideoX-5B	×	81.61	82.75	**77.04**	96.92	**70.97**
Kling	×	81.85	83.38	75.68	99.40	46.94
Gen-3 Alpha	×	**82.32**	84.11	75.17	99.23	60.14
Open-Sora Plan v1.1	✓	78.00	80.91	66.38	98.28	47.72
Open-Sora 1.2	✓	79.76	81.35	73.39	98.50	42.39
VideoCrafter2	✓	80.44	82.20	73.42	97.73	42.50
T2V-Turbo	✓	81.01	82.57	74.76	97.34	49.17
Ours	✓	81.72	**84.74**	69.62	99.12	64.63

Table 2: Experimental results on EvalCrafter (Liu et al., 2024). See Appendix C.1 for raw metrics.

Model	Public Data	Final Sum Score	Visual Quality	Text-Video Alignment	Motion Quality	Temporal Consistency
Pika 1.0	×	250	63.05	66.97	**56.43**	63.81
Gen-2	×	**254**	**69.09**	63.92	55.59	**65.40**
ModelScope	✓	218	53.09	54.46	52.47	57.80
Show-1	✓	229	52.19	62.07	53.74	60.83
LaVie	✓	234	57.99	**68.49**	52.83	54.23
VideoCrafter2	✓	243	63.98	63.16	54.82	61.46
Ours	✓	244	67.94	57.01	55.31	63.41

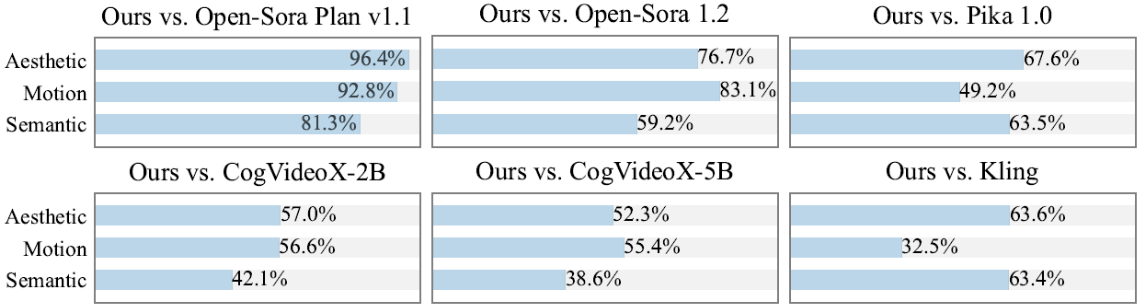


Figure 4: User preference on sampled VBench prompts. Our videos are generated at 5s, 768p, 24fps.

User study. While quantitative evaluation scores reflect the video generation capability to some extent, they may not align with human preferences for visual quality. Hence, an additional user study is conducted to compare our performance with six baseline models, including CogVideoX (Yang et al., 2024) and Kling (Kuaishou, 2024). We utilized 50 prompts sampled from VBench and asked 20+ participants to rank each model according to the aesthetic quality, motion smoothness, and semantic alignment of the generated videos. As seen in Fig. 4, our method is preferred over open-source models such as Open-Sora and CogVideoX-2B especially in terms of motion smoothness. This is due to the substantial token savings achieved by pyramidal flow matching, enabling generation of 5-second (up to 10-second) 768p videos at 24 fps, while the baselines usually support video synthesis of similar length only at 8 fps. The detailed user study settings are presented in Appendix B.

Image-to-Video Generatetion. Thanks to the autoregressive property of our model and the causal attention design, the first frame of each video acts similarly to an image condition during the training. Consequently, although our model is optimized solely for text-to-video generation, it naturally accommodates text-conditioned image-to-video generation during inference. Given an image and a textual prompt, it is able to animate the static input image by autoregressively predicting the future frames without further fine-tuning. In Fig. 6, we illustrate qualitative examples of its image-to-video generation performance, where each example consists of 120 newly synthesized frames spanning a duration of 5 seconds. As can be seen, our model successfully predicts reasonable subsequent motion, endowing the images with rich temporal dynamic information. More generated video examples are best viewed on our project page at `https://pyramid-flow.github.io`.

4.4 ABLATION STUDY

In this section, we conduct ablation studies to validate the crucial component of our methods, including the spatial pyramid in denoising trajectory and the temporal pyramid in history condition. Due to limited space, the ablations for other design choices are provided in Appendix C.2.

Effectiveness of spatial pyramid. In the generation trajectory of the proposed spatial pyramid, only the final stage operates at full resolution, which significantly reduces the number of tokens for most denoising timesteps. With the same computational resources, it can handle more samples per training batch, greatly enhancing the convergence rate. To validate its efficiency, we designed a baseline that employs the standard flow matching objective for training text-to-image generation in our early experiments. This baseline is optimized using the same training data, number of tokens per batch, hyperparameter configurations, and model architecture to ensure fairness. The performance comparison is illustrated in Fig. 7. It can be observed that the variant using pyramidal flow demonstrates superior visual quality and prompt-following capability. We further quantitatively evaluate the FID metric of these methods on the MS-COCO benchmark (Lin et al., 2014) by randomly sampling 3K prompts. The FID performance curve over training steps is presented on the right of Fig. 7. Compared to standard flow matching, the convergence rate of our method is significantly improved.

Effectiveness of temporal pyramid. As mentioned in Section 4.2, the temporal pyramid design can drastically reduce the computation demands compared to traditional full-sequence diffusion. Similar to the spatial pyramid, we also established a full-sequence diffusion baseline under the same experimental setting to investigate its training efficiency improvement. The qualitative comparison with the baseline is presented in Fig. 8, where the generated videos of our pyramidal variant demonstrate much better visual quality and temporal consistency under the same training steps. In contrast, the full-sequence diffusion baseline is far from convergence. It fails to produce coherent motion, leading to fragmented visual details and severe artifacts in the generated videos. This performance gap clearly highlights the training acceleration achieved by our method in video generative modeling.

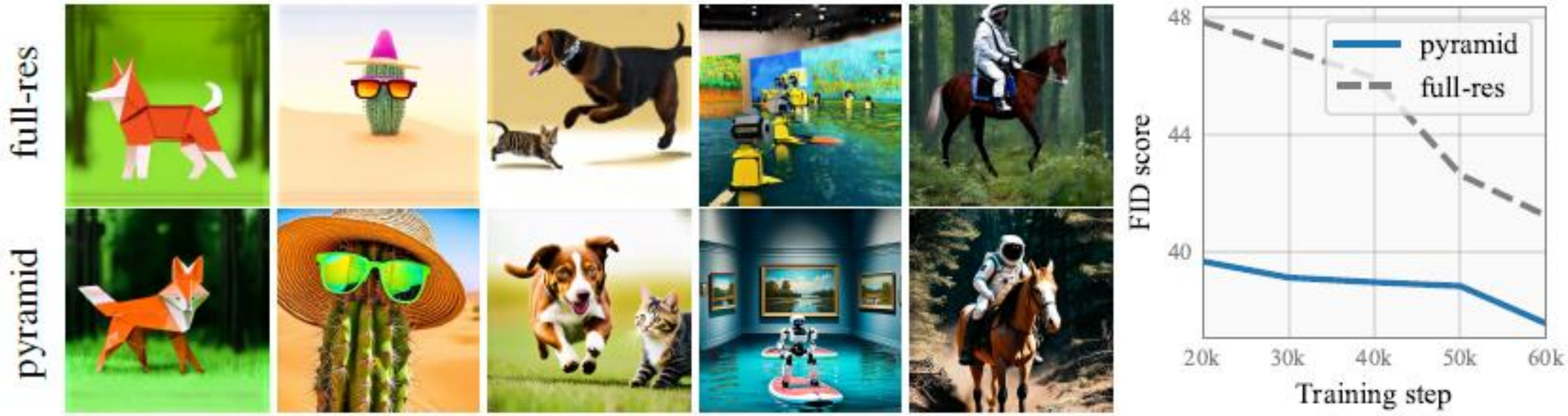


Figure 7: Ablation study of spatial pyramid at 50k image training step. On the right is a quantitative comparison of the FID results, where our method achieves almost three times the convergence speed.

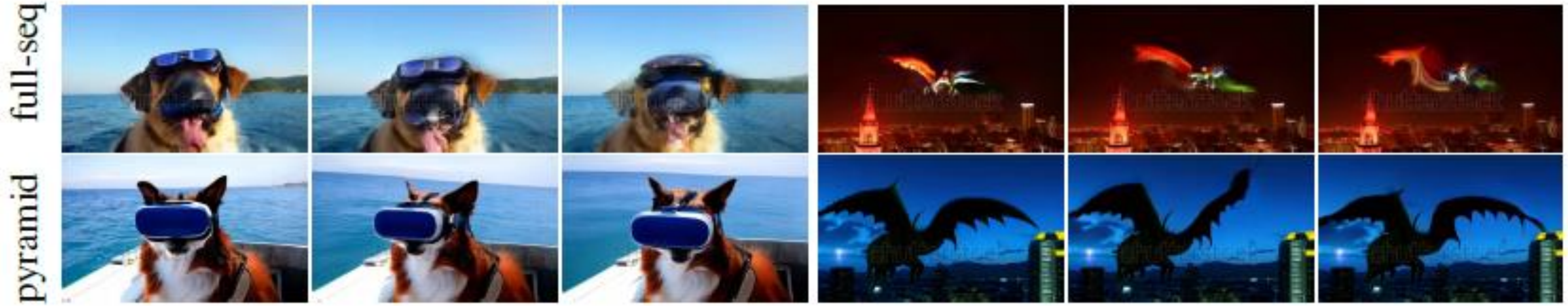
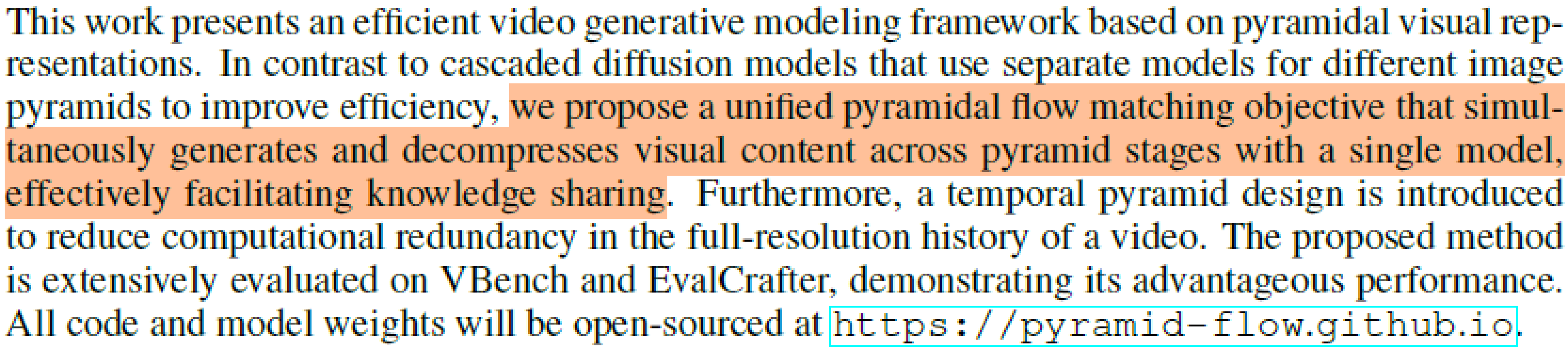


Figure 8: Ablation study of temporal pyramid at 100k low-resolution video training step.

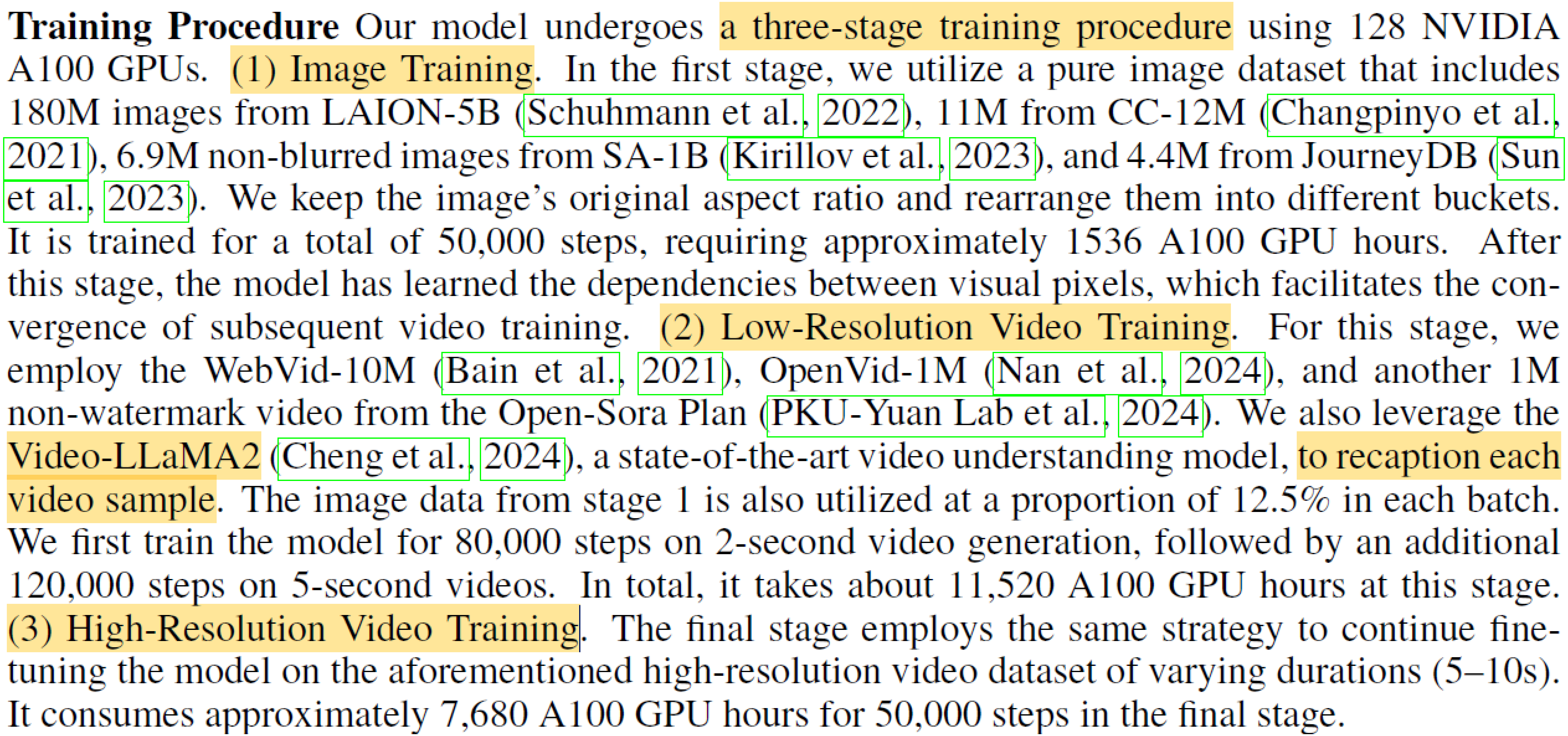
5 CONCLUSION



This work presents an efficient video generative modeling framework based on pyramidal visual representations. In contrast to cascaded diffusion models that use separate models for different image pyramids to improve efficiency, we propose a unified pyramidal flow matching objective that simultaneously generates and decompresses visual content across pyramid stages with a single model, effectively facilitating knowledge sharing. Furthermore, a temporal pyramid design is introduced to reduce computational redundancy in the full-resolution history of a video. The proposed method is extensively evaluated on VBench and EvalCrafter, demonstrating its advantageous performance. All code and model weights will be open-sourced at `https://pyramid-flow.github.io`.

B Experimental Settings

Model Implementation Details. We adopt the MM-DiT architecture, based on SD3 Medium (Esser et al., 2024), which comprises 24 transformer layers and a total of 2B parameters. The weights of the MM-DiT are initialized from the SD3 medium. Following the more recent FLUX.1 (Black Forest Labs, 2024), both T5 (Raffel et al., 2020) and CLIP (Radford et al., 2021) encoders are employed for prompts embedding. To address the redundancy in video data, we have designed a 3D VAE that compresses videos both spatially and temporally into a latent space. The architecture of this VAE is similar to MAGVIT-v2 (Yu et al., 2024), employing 3D causal convolution to ensure that each frame depends only on the preceding frames. It features an asymmetric encoder-decoder with Kullback-Leibler (KL) regularization applied to the latents. Overall, the 3D VAE achieves a compression rate of 8 × 8 × 8 from pixels to the latent. It is trained on WebVid-10M and 6.9M SAM images from scratch. To support the tokenization of very long videos, we scatter them into multiple GPUs to distribute computation like CogVideoX (Yang et al., 2024).



Training Procedure Our model undergoes a three-stage training procedure using 128 NVIDIA A100 GPUs. (1) Image Training. In the first stage, we utilize a pure image dataset that includes 180M images from LAION-5B (Schuhmann et al., 2022), 11M from CC-12M (Changpinyo et al., 2021), 6.9M non-blurred images from SA-1B (Kirillov et al., 2023), and 4.4M from JourneyDB (Sun et al., 2023). We keep the image's original aspect ratio and rearrange them into different buckets. It is trained for a total of 50,000 steps, requiring approximately 1536 A100 GPU hours. After this stage, the model has learned the dependencies between visual pixels, which facilitates the convergence of subsequent video training. (2) Low-Resolution Video Training. For this stage, we employ the WebVid-10M (Bain et al., 2021), OpenVid-1M (Nan et al., 2024), and another 1M non-watermark video from the Open-Sora Plan (PKU-Yuan Lab et al., 2024). We also leverage the Video-LLaMA2 (Cheng et al., 2024), a state-of-the-art video understanding model, to recaption each video sample. The image data from stage 1 is also utilized at a proportion of 12.5% in each batch. We first train the model for 80,000 steps on 2-second video generation, followed by an additional 120,000 steps on 5-second videos. In total, it takes about 11,520 A100 GPU hours at this stage. (3) High-Resolution Video Training. The final stage employs the same strategy to continue fine-tuning the model on the aforementioned high-resolution video dataset of varying durations (5–10s). It consumes approximately 7,680 A100 GPU hours for 50,000 steps in the final stage.

C.2 ABALTION STUDY

In this section, we conduct additional ablation studies of two important design details in our proposed pyramidal flow matching, including the corrective noise added during inference of the spatial pyramid and the blockwise causal attention used for autoregressive video generation.

Role of corrective noise. To study its efficacy in the spatial pyramid, we curate a baseline method that inferences without adding this corrective Gaussian noise. The detailed comparative results of our method against this variant are shown in Fig. 10. While the baseline method has a correct global structure, it fails to produce a fine-grained, high-resolution image with rich details and instead produces a blurred image that suffers from block-like artifacts (better observed when zooming in). This is because applying the upsampling function at the jump points between different pyramid stages of varying resolutions results in excessive correlation between spatially adjacent latent values. In comparison, our generated images have rich details and vivid colors, confirming that the adopted corrective renoising scheme effectively addresses this artifact problem in the spatial pyramid.

Effectiveness of causal attention. In Fig. 11, we study the effect of blockwise causal attention by comparing it to the bidirectional attention used in full-sequence diffusion. While an intuitive understanding might be that bidirectional attention promotes information exchange and increases model capacity, it is understudied for autoregressive video generation. In an early experiment, we trained a baseline model using bidirectional attention across different latent frames, the results of which are visualized in Fig. 11. As can be seen from the sampled keyframes of the 1-second videos, this model suffers from a lack of temporal coherence as the subject in the generated video is constantly changing in shape and color. Meanwhile, our model shows good temporal coherence with reasonable motion. We infer that this is because the history condition in bidirectional attention is influenced by the ongoing generation and thus deviates, whereas the history condition in causal attention is fixed, serving as a predetermined condition and stabilizing the autoregressive generative process.

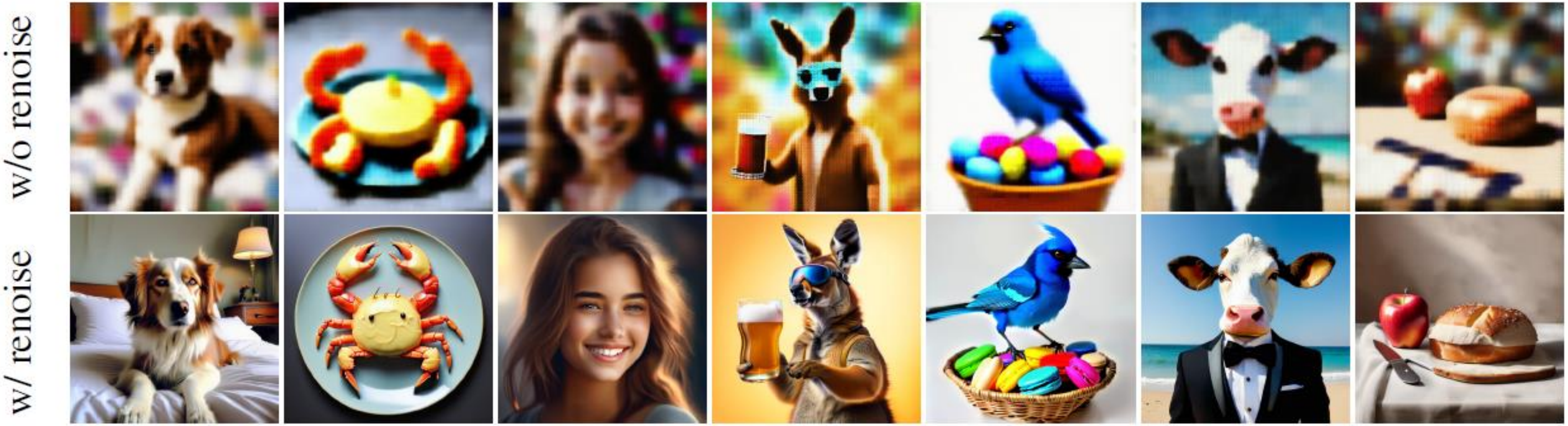


Figure 10: Ablation study of corrective renoising during the inference stage.

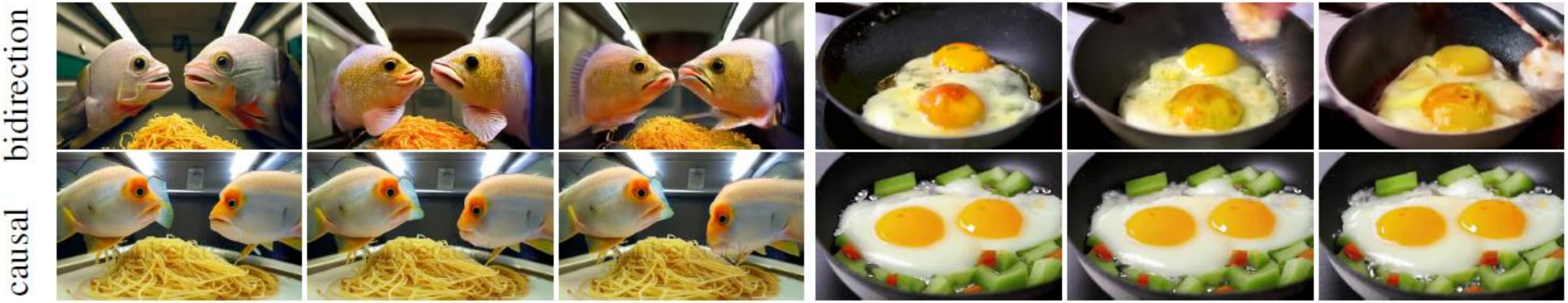


Figure 11: Ablation study of blockwise causal attention at 100k training step.

D LIMITATIONS

Our method only supports autoregressive generation and cannot be extended to keyframe interpolation or video interpolation. In addition, we noticed that the temporal pyramid designs to improve training efficiency can sometimes lead to subtle subject inconsistency, especially over the long term. While this is not a prevalent problem, we believe that developing more sophisticated temporal compression methods is critical to the broader applicability of our video generative model.

There are also several issues related to the training data. Since we did not include a prompt rewriting procedure in the data curation, the experimental results are focused on relatively short prompts. Also, due to the data filtering procedure, our model did not learn scene transitions during training. This may be overcome by introducing an additional model as the scene director (Lin et al., 2024).