Reconstructing Visual Appearance and Process by Repurposing Pretrained Diffusion Models

Bowei Chen

A dissertationsubmitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2026

Reading Committee:

Steven M. Seitz, Chair

Brian Curless, Chair

Ira Kemelmacher-Shlizerman

Program Authorized to Offer Degree:
Allen School of Computer Science & Engineering

University of Washington

Abstract

Reconstructing Visual Appearance and Process by Repurposing Pretrained Diffusion Models

Bowei Chen

Co-Chairs of the Supervisory Committee:
Steven M. Seitz
Allen School of Computer Science & Engineering

Brian Curless
Allen School of Computer Science & Engineering

Visual generation problems often arise in regimes where the available observations are incomplete or indirect. Inputs may capture only fragments of visual appearance, or omit the intermediate processes that produced the final result, yet models are expected to synthesize outputs that are visually complete, coherent, and plausible. This setting places strong demands on generative models, requiring them to infer missing information, integrate fragmented evidence, and reconstruct underlying structure or dynamics consistent with the observed outcome.

This thesis investigates how pretrained diffusion models can be repurposed to address visual generation tasks arising under partial or indirect observation. I study a set of representative applications in which this tension manifests in different forms, spanning both appearance reconstruction and process reconstruction. These include synthesizing complete human appearance and motion from a small number of casually captured selfies, selectively editing portraits while preserving fine-grained identity features, and reconstructing plausible painting processes from a single finished artwork. Across these settings, the desired outputs are visually complete, while the inputs provide only sparse, incomplete, or indirect constraints.

Although recent diffusion models learn powerful visual priors through large-scale pre-

training, they are primarily designed for generic generation tasks such as text-to-image or text-to-video synthesis, and are not directly suited to these partial-observation scenarios. Mismatches in input structure, supervision, and data availability make naive fine-tuning or prompt-based adaptation ineffective. This thesis addresses the central question: *how can pretrained diffusion models be repurposed to support novel visual generation tasks under partial observation?*

I explore repurposing strategies across different supervision regimes. In settings where paired training data is unavailable, I develop methods that exploit weakly aligned observations or synthetically constructed supervision to enable appearance reconstruction from fragmented inputs. In settings with limited but well-aligned paired data, I show that composing multiple pretrained models into cascaded pipelines can amplify scarce supervision and enable complex process reconstruction, such as inferring plausible sequences of painting actions consistent with a final artwork.

Beyond pipeline-level design, the effectiveness of reconstructing visual appearance and process also depends critically on the quality of visual priors learned during diffusion pretraining. Stronger priors lead to more robust adaptation and higher generation quality across tasks. This observation motivates the final part of the thesis, which explores repurposing at a more fundamental level. I propose *AlignTok*, a framework that repurposes pretrained visual encoders as tokenizers for diffusion models, enabling diffusion to operate in a semantically rich latent space. This design simplifies training and improves efficiency, scalability, and generation quality, resulting in stronger visual priors that better support downstream reconstruction tasks.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I first want to thank my advisors, Steven M. Seitz, Brian Curless, and Ira Kemelmacher-Shlizerman, for their unwavering support, guidance, and trust throughout my Ph.D. Their vision and mentorship shaped not only this dissertation but also the way I approach problems and pursue new ideas. I am deeply grateful for the freedom they gave me to explore ambitious ideas and for the high standards they consistently encouraged me to pursue. I am also grateful to my committee member, Jeffrey A. Bilmes, for his time and support.

I would like to sincerely thank my mentors, Tiancheng Zhi and Kai Zhang, for their guidance and support during my internships. Their technical depth, clarity of thinking, and high standards pushed me to grow quickly and take on more ambitious challenges. I also thank my collaborators during the internships, Sai Bi, Hao Tan, He Zhang, Tianyuan Zhang, Zhengqi Li, Yuanjun Xiong, Jianming Zhang, Peihao Zhu, Shen Sang, Jing Liu, and Linjie Luo, for the stimulating discussions, thoughtful feedback, and collaborative spirit. Working with them was both rewarding and inspiring.

I am especially grateful to my labmates, Yuanhao Wang, Johanna Karras, Baback Elmieh, Susung Hong, Mengyi Shan, Jingwei Ma, Xiaojuan Wang, Vivek Jayaram, Luyang Zhu, Alice Gao, Benlin Liu, Meng-Li Shih, and Yifan Wang, for the inspiring discussions, collaboration, and daily support. It was a privilege to grow alongside such thoughtful and driven peers. I also want to thank my friends, Yuqun Wu, Heng Yu, and Tianyuan Zhang, for their encouragement and companionship. Their presence made these years richer and more balanced.

I am deeply grateful to my parents, Jiansong Chen and Bingjun Lin, for their unconditional love, sacrifice, and constant support. Everything I have achieved rests on the foundation they built for me. I also thank my brother, Bohan Chen, and his wife, Xiaoyan Huang, for their encouragement and care. Thanks to my cat, Tutu, for the quiet compan-

ionship, comfort, and small daily moments of joy that made the long writing nights a little lighter.

I am especially thankful to my grandmother, Saizhen Shi, whose love and presence shaped my childhood and whose memory continues to guide me.

Finally, I thank my wife, Jingyi Lin, my true love. Your unwavering love, support, patience, and belief in me carried me through every high and low. None of this would have been possible without you.

DEDICATION

to my dear wife, Jingyi

Chapter 1

INTRODUCTION

Much of the visual world is never fully observed. Cameras glimpse only fragments of a scene, sensors capture incomplete evidence, and the processes that give rise to visual outcomes often unfold beyond our view. Yet visual generation systems are routinely asked to produce results that appear complete, coherent, and intentional. This tension between partial observation and complete visual synthesis lies at the heart of visual intelligence, and addressing it requires generative models that can infer not only missing visual content, but also the structure and dynamics of the processes that could plausibly have produced it.

Motivated by this gap between partial observation and complete visual synthesis, this thesis investigates several representative visual generation applications in which this tension arises in distinct forms. I first study *total selfie* (Fig. 1.1) and *fit-check video generation*, where the available inputs are limited to a small number of casually captured selfies that each observe only a fragment of a person's appearance. The challenge in these settings lies in inferring missing viewpoints, resolving inconsistencies across observations, and reconstructing a globally coherent and identity-preserving representation from sparse visual evidence. I then consider *feature-preserving portrait editing*, where the input provides only partial specification of the desired outcome: certain attributes are intended to change, while others must remain unchanged. This requires reconstructing a complete image that selectively modifies targeted factors while faithfully preserving fine-grained appearance details that are not explicitly constrained. Finally, I examine *Inverse Painting* (Fig. 1.2), a setting in which the observation is reduced to a single final artwork, while the sequence of intermediate steps that produced it is entirely unobserved. Here, the task is to infer a plausible generative process—capturing ordering, layering, and incremental refinement—that is consistent with the observed outcome. Together, these applications span both appearance and process reconstruction, and collectively illustrate how partial or indirect observations impose fun-

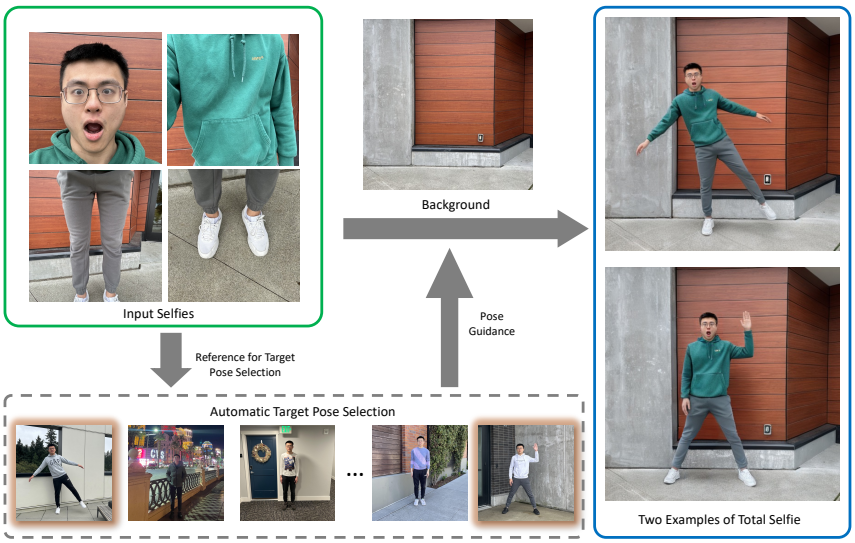


Figure 1.1: We generate full-body selfies of you (right), from self-captured images of your face and body (top left) and background.

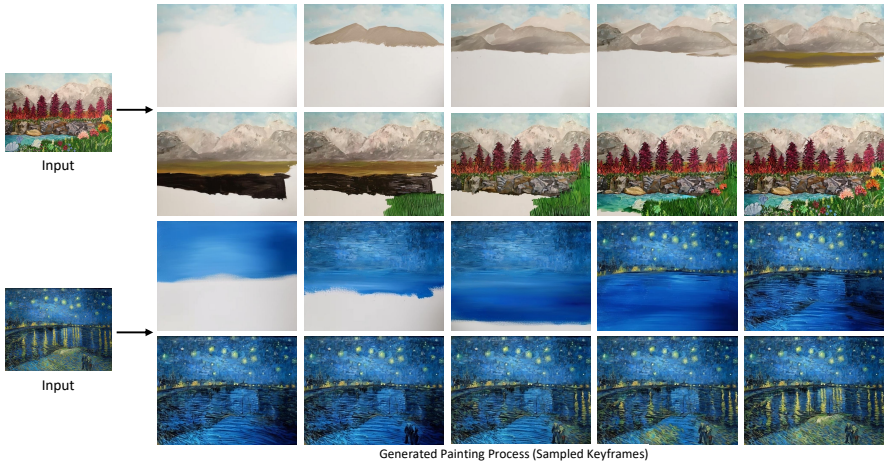


Figure 1.2: We present Inverse Painting, a diffusion-based method to generate time-lapse videos of the painting process from a target painting. Images courtesy Catherine Kay Greenup and Rawpixel.

damental challenges for generative models tasked with producing complete, coherent visual results.

Addressing these challenges requires generative models with strong visual priors and the ability to infer missing structure from limited evidence. Recent years have witnessed remarkable progress in visual generative models. Images and videos that once required hours of careful manual creation can now be synthesized automatically from simple descriptions or reference examples. In particular, diffusion models trained on large-scale image and video datasets have demonstrated an unprecedented ability to generate high-quality, realistic visual content from text prompts or input images, capturing fine appearance details, complex structure, and natural motion [216, 20, 282, 304, 273, 12, 271, 156, 336, 202, 251, 102, 258, 39, 222] (examples in Fig. 1.3). These advances are reshaping how visual media is created and are motivating a new class of applications that were previously difficult or impossible to realize.

Figure 1.3: **Text-to-image generation with diffusion models.** Given a text prompt, a pretrained diffusion model can synthesize diverse, high-quality images with rich structure and appearance. Image credit [65].

However, many creative applications arising in partial-observation settings cannot be supported directly by existing pretrained diffusion models. While such models excel at general tasks, such as text-to-image generation, text-to-video generation, or general image editing, they are not designed to handle the task-specific inputs and outputs required by emerging creative applications. For example, the inputs of the creative task may consist of

several selfies that each capture only a partial view of a person's body, while the desired output may be a full-body image that coherently integrates these observations and preserves the subject's identity, appearance, and pose. Such transformations require strong appearance consistency and global geometric reasoning, and are not directly or fully supported by standard pretrained diffusion models, even when using current state-of-the-art models [82].

Enabling pretrained diffusion models to support these creative applications requires principled design. Paired training data that directly specifies the desired input–output mapping may be limited in scale or entirely unavailable, and the target distributions may differ significantly from those encountered during pretraining. As a result, naive fine-tuning or prompt-based adaptation is insufficient, leading to failures in controllability, appearance consistency, identity preservation, or geometric coherence.

This thesis investigates a central question: *how can pretrained models be repurposed to support novel visual generation tasks?* I explore choices of data construction, supervision design, and pipeline architecture. Across a series of projects, this thesis demonstrates that carefully designed repurposing strategies can unlock new capabilities from pretrained diffusion models, even in settings with limited paired data or without paired data at all.

- **No Paired Data Available.** I first consider scenarios in which task-specific paired supervision is unavailable. I study two complementary strategies. The first exploits weakly aligned training pairs derived from loosely related observations, enabling adaptation to novel generation tasks in *Total Selfie* (Fig. 1.1) and *Fit Check Video Generation*, which synthesize full-body images and videos from a small number of selfie photos. The second constructs synthetic paired data using complementary pretrained models, showing that high-quality synthetic supervision can effectively guide image editing in *Feature-Preserving Portrait Editing*.

- **Limited Paired Supervision.** I then study settings in which paired supervision is well aligned but scarce. By composing multiple pretrained models into cascaded pipelines, I show that scarce supervision can be amplified to enable complex reconstruction tasks, demonstrated by recovering plausible painting processes from final artworks in *Inverse Painting* (Fig. 1.2).

While the above approaches focus on repurposing pretrained diffusion models through data construction and pipeline design, their effectiveness ultimately depends on the quality of the visual priors learned during pretraining. Stronger pretrained representations consistently lead to more robust adaptation and higher generation quality, suggesting that improving the priors themselves is a promising direction for enabling broader and more reliable repurposing.

- **Improving Visual Priors.** I investigate repurposing at a more fundamental level by improving the visual priors learned during diffusion pretraining. I propose *AlignTok* (Fig. 1.4), which repurposes pretrained visual encoders as tokenizers for diffusion models, allowing diffusion to operate directly in a semantically rich latent space. This design simplifies training and improves efficiency, scalability, and generation quality, leading to stronger visual priors for downstream adaptation.

This thesis is organized as follows. Chapter 2 reviews background on diffusion models and visual generative modeling. Chapters 3–6 each present one project that explores a dif-

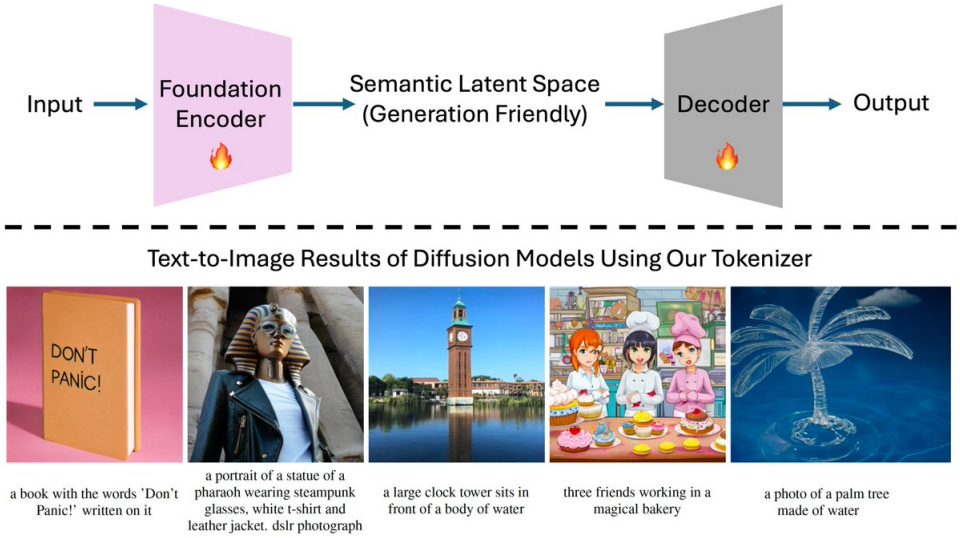


Figure 1.4: Pretrained visual foundation encoder can be aligned to serve as tokenizers for latent diffusion models in image generation.

ferent repurposing strategy: Chapter 3 introduces *Total Selfie*, Chapter 4 presents *Fit Check Video Generation*, Chapter 5 studies *Feature-Preserving Portrait Editing*, and Chapter 6 describes *Inverse Painting*. Chapter 7 introduces *AlignTok*, which investigates repurposing at the level of diffusion pretraining to improve visual priors. Finally, Chapter 8 concludes with a discussion of future directions.

Chapter 2

FOUNDATION OF DIFFUSION MODELS AND FLOW MATCHING

Diffusion models are a class of generative models that synthesize realistic images or videos by progressively transforming random noise into coherent content through a learned denoising process. Despite their popularity, understanding the foundational framework of diffusion models can be challenging. This is largely due to the wide variety of noise schedulers, network parametrization, time sampling strategies, training objectives, and inconsistent notations adopted across different papers [65, 244, 242, 103, 126]. Moreover, recent advances in image and video generation [86, 65] have adopted the flow matching objective [160] as an alternative way to train diffusion models. Since diffusion and flow matching were originally presented as two distinct classes of generative models, this development has led to some confusion in the community.

In this section, we provide a deeper analysis of various design choices in diffusion frameworks, such as different noise schedules, and discuss how these factors influence both the training and inference of diffusion models. We also clarify the connection between diffusion models and flow matching, enabling a unified understanding and the exchange of methodologies – for example, training diffusion models with a flow matching objective. Since flow matching is often conceptually more intuitive, we begin by introducing its core principles. We then present diffusion models through the lens of flow matching, making their underlying mechanisms easier to understand.

2.1 Flow Matching

Flow matching is an elegant approach to generative modeling that defines a continuous transformation between probability distributions. The goal of flow matching is to learn a velocity field that transforms a simple source distribution into a complex target distribution. We start with a source distribution $X_0 \sim \pi_0$ (typically a standard Gaussian) and aim to

transform it into a target distribution X_1 (our data distribution) through a continuous flow (*i.e.*, path). *Throughout our discussion, we will use the terms "flow" and "path" interchangeably to refer to the continuous transformation trajectory between distributions.* Formally, we define this transformation via the ordinary differential equation (ODE):

$$\frac{dX_t}{dt} = v_t(X_t), \quad \forall t \in [0,1], \quad \text{starting from } X_0 \sim \pi_0 \tag{2.1}$$

Here, v_t is the time-dependent velocity field that we aim to learn, and X_t represents the state at time t along the flow path. In practice, the velocity field v_t is parameterized using a neural network $v_\theta(X_t, t)$ with parameters θ, allowing it to learn complex transformations that map between distributions.

To generate new samples given the trained velocity field, we perform inference by numerically solving the ODE:

$$\frac{dX_t}{dt} = v_\theta(X_t, t), \quad X_0 \sim \pi_0 \tag{2.2}$$

This integration is typically performed using numerical ODE solvers such as Euler's method, Runge-Kutta methods, or more advanced adaptive solvers. For example, with Euler's method, we can discretize time into N steps and iteratively update:

$$X_{t+\Delta t} = X_t + v_\theta(X_t, t) \cdot \Delta t \tag{2.3}$$

where $\Delta t = 1/N$. Higher-order methods like Runge-Kutta or adaptive step-size solvers can provide more accurate solutions with fewer steps, thereby yielding higher-quality samples.

2.1.1 Training through Straight Path

The key insight of training flow matching is that we can directly supervise the neural network velocity field by defining explicit paths (*i.e.*, flows) between randomly sampled data points from source and target distribution. Given independently sampled pairs (X_0, X_1) from the source and target distributions, we choose to define a straight-line path between them:

$$X_t = tX_1 + (1-t)X_0 \tag{2.4}$$

The straight-line path is chosen for its simplicity and computational efficiency, though other path choices are possible. Taking the time derivative of this path yields the ground truth velocity:

$$\dot{X}_t = \frac{dX_t}{dt} = X_1 - X_0 \tag{2.5}$$

This constant velocity vector provides a clear supervision signal, allowing us to train the velocity field model v_θ using a simple mean squared error loss:

$$\mathcal{L} = \min_\theta \mathbb{E}\left[\|\dot{X}_t - v_\theta(X_t, t)\|^2\right] \tag{2.6}$$

In practice, we sample $t \sim \mathcal{U}(0, 1)$ for each training point, compute X_t and $\dot{X}_t$, and update the neural network parameters θ to minimize the discrepancy between predicted and ground truth velocities. This direct supervision approach avoids the need to solve ODEs during training, making flow matching computationally efficient compared to other flow-based methods [138].

2.1.2 Learned Flow is not Straight

A common misunderstanding is that the learned velocity will induce straight path (*i.e.*, flow) because it is supervised by straight-line parameterization, thus fewer sampling steps would be needed due to the straightness of the path with smaller discretization error.

However, despite training with straight-line paths, the trained model often produces curved trajectories. This occurs because at intersection points between different trajectories, the model averages multiple directions, leading to smoothly varying paths that better respect the data manifold's geometry. For example, if we have two training pairs that cross paths:

$$X_t^{(1)} = tX_1^{(1)} + (1-t)X_0^{(1)} \tag{2.7}$$

$$X_t^{(2)} = tX_1^{(2)} + (1-t)X_0^{(2)} \tag{2.8}$$

At their intersection, the model must compromise between the two directions, resulting in curved trajectories during inference that don't precisely match either training flow but create a smooth transformation overall. Figure 2.1 illustrates this phenomenon.

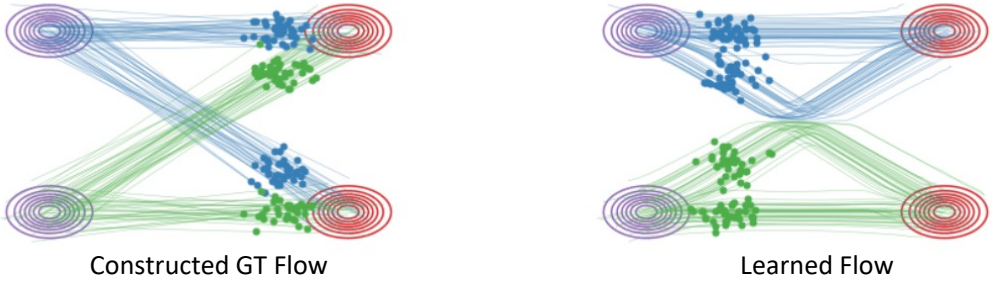


Figure 2.1: Visualization of constructed ground truth flows used for training (left) and the learned flows (right). The model is trained on straight-line paths, but the produced paths are rewired on the intersecting points, creating a smoother overall flow through the data manifold. Image credit [169].

2.2 Diffusion Models

Diffusion models, similar to flow matching, can be understood through the concept of transforming a simple distribution into a complex data distribution. We can relate diffusion models to the generalized affine gaussian path definition used in flow matching, which is given by $X_t = \alpha_t X_1 + \beta_t X_0$ where $X_0 \sim \mathcal{N}(0, I)$ is the source (noise) distribution and X_1 is the target (data) distribution.

In the context of a diffusion model such as Denoising Diffusion Probabilistic Models (DDPM) [103], the process defining the path between data and noise (*i.e.*, forward process) can be written in a form analogous to our flow matching path. In particular, the state X_t in DDPM at a given time t along this path is defined as:

$$X_t = \sqrt{\bar{\alpha}_t} X_1 + \sqrt{1 - \bar{\alpha}_t} X_0 \tag{2.9}$$

Here, the terms $\sqrt{\bar{\alpha}_t}$ and $\sqrt{1 - \bar{\alpha}_t}$ are time-dependent coefficients corresponding to α_t and β_t in the general affine path formulation, and $\bar{\alpha}_t$ is computed based on the time t. The choice of these coefficients as functions of time (*i.e.*, α_t and β_t) is referred to as the "scheduler."

A key distinction often highlighted is that diffusion models are typically formulated using Stochastic Differential Equations (SDEs), whereas the flow matching framework we've

discussed so far is based on Ordinary Differential Equations (ODEs). The SDE for a diffusion process can be written as:

$$dX_t = v_t(X_t)dt + \sigma(X_t, t)dW_t \tag{2.10}$$

where $v_t(X_t)$ is the drift coefficient (*i.e.*, velocity field), $\sigma(X_t, t)$ is the diffusion coefficient, and dW_t represents Brownian motion. To simplify the comparison and unify the understanding, we can initially ignore the stochastic term ($\sigma(X_t, t)dW_t$) in the SDE (we will revisit SDE later). This effectively treats the diffusion process as an ODE:

$$dX_t = v_t(X_t)dt \tag{2.11}$$

By making this simplification, diffusion models can be conceptualized as defining different Gaussian ground truth (GT) paths for the transformation from noise to data, where flow matching is a special case that defines straight GT paths. This sets the stage for a unified view where both flow matching and diffusion models aim to learn a vector field to transform distributions, with the primary initial difference being the specific form of the path (scheduler) and whether the underlying dynamic is purely deterministic (ODE) or stochastic (SDE) [1].

2.2.1 Properties of the Unified Framework: Schedulers and Inference

With the initial simplification of viewing diffusion processes as ODEs, we can explore the properties of this unified framework. A crucial aspect is understanding how different choices of "schedulers" or ground truth (GT) paths relate to each other.

As established, both flow matching and diffusion models (under the ODE assumption) define a path X_t between a noise distribution $X_0 \sim \mathcal{N}(0, I)$ and a data distribution X_1. A common way to define these paths is through affine interpolations:

$$X_t = \alpha_t X_1 + \beta_t X_0 \tag{2.12}$$

[1] Not all diffusion models rely on an SDE formulation—for instance, DDIM [242] is based on a deterministic ODE. However, most diffusion models do incorporate stochasticity, so in this report we primarily focus on the SDE-based formulation.

where (α_t, β_t) are scalar functions of time $t \in [0, 1]$ defining the specific "scheduler," X_1 represents a sample from the data distribution, and X_0 represents a sample from the noise distribution. For instance, the straight path in our initial flow matching discussion had $\alpha_t = t$ and $\beta_t = (1 - t)$. Diffusion models like DDPM, as formulated in Equation 2.9, use $X_t = \sqrt{\bar{\alpha}_t} X_1 + \sqrt{1 - \bar{\alpha}_t} X_0$.

An important insight [160] is that different affine interpolations (i.e., different schedulers) are fundamentally related. Given two such interpolations, $X_t = \alpha_t X_1 + \beta_t X_0$ and $X'_t = \alpha'_t X_1 + \beta'_t X_0$, that start and end at the same distributions, they are pointwise transformations of each other. Specifically, there exist scalar functions τ_t (a time warping function) and ω_t (a scaling function) such that:

$$X'_t = \frac{1}{\omega_t} X_{\tau_t} \tag{2.13}$$

where τ_t and ω_t are determined by the ratios of the scheduler coefficients:

$$\frac{\alpha_{\tau_t}}{\beta_{\tau_t}} = \frac{\alpha'_t}{\beta'_t}; \quad \omega_t = \frac{\alpha_{\tau_t}}{\alpha'_t} = \frac{\beta_{\tau_t}}{\beta'_t} \tag{2.14}$$

with boundary conditions $\tau_0 = 0, \tau_1 = 1$ and $\omega_0 = \omega_1 = 1$. This means that all affine schedulers are, in essence, exploring the same underlying space of paths, just at different "speeds" and "scales" at corresponding warped times. Fig. 2.2 provides a visualization to illustrate the point.

An interesting consequence of this properties is the flexibility at inference time. If a model is trained using a particular scheduler (defining specific α_t, β_t), we are not strictly bound to use the exact same path definition or discretization during inference. We can often switch to a different scheduler or a different number of discretization steps. For example, if we train a model with scheduler A, we can potentially use scheduler B or C for inference. Ideally, if the network were perfectly trained and the discretization steps were infinitely fine, sampling with different schedulers would produce identical results. However, in practice, this is rarely achievable due to model imperfections and the finite number of discretization steps.

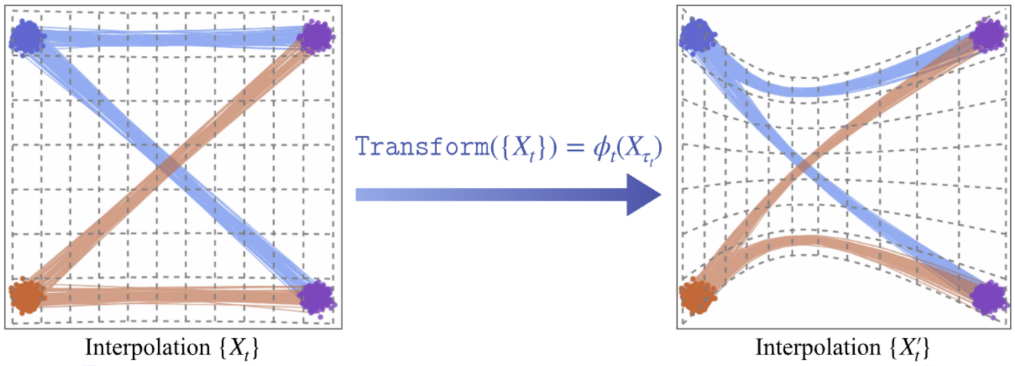


Figure 2.2: Visualization of relationship between paths defined by different schedulers. Image credit [170].

2.2.2 Properties of the Unified Framework: Training

In the unified framework, diffusion models and flow matching can have different schedulers, network parameterizations, and time sampling strategies. In the following sections, we will examine how these design choices affect model training and highlight the key differences between various diffusion frameworks proposed in recent works. This deeper analysis will help demystify the underlying distinctions among methods, despite the often complex mathematical derivations presented in the literature [65, 244, 242, 103, 126].

Training with Different Schedulers. The pointwise relationship between different schedulers (discussed in Sec. 2.2.1) has direct implications for training a velocity model $v_\theta(X_t, t)$. If we train a model using one affine path X_t with a loss function like:

$$\mathcal{L}(\theta) = \mathbb{E}[w(t)\|\dot{X}_t - v_\theta(X_t, t)\|^2] \tag{2.15}$$

where $w(t)$ is the time-related weighting function. Then we have another model with a different affine path X'_t and loss $\mathcal{L}'(\theta)$:

$$\mathcal{L}'(\theta) = \mathbb{E}[w'(t)\|\dot{X}'_t - v'_\theta(X_t, t)\|^2] \tag{2.16}$$

then these two training procedures are equivalent up to a re-weighting of the loss $w'(t) = \frac{\omega_t^2}{\dot{\tau}_t} w(\tau_t)$ and a reparameterization of the velocity network $v'_\theta(X_t, t) = \frac{\dot{\tau}_t}{\omega_t} v_\theta(\omega_t X_t; \tau_t) - \frac{\dot{\omega}_t}{\omega_t} X_t$, where ω_t is defined in Eq. 2.14.

In simpler terms, training with different affine GT paths (schedulers) is largely equivalent to applying a different time-dependent weighting function in the loss and adjusting the network's input and output scaling. Among these changes, the rescaling of the network has minimal impact on the training dynamics.

Training with Different Network Output Parameterizations. Beyond the choice of scheduler, another critical design aspect in diffusion and flow models is the parameterization of the neural network's output. The network can be trained to predict various quantities. Common choices include:

- The noise component $\hat{X}_0$ (this is often referred to as noise prediction or ϵ-prediction).
- The clean data sample $\hat{X}_1$ (representing the network's direct prediction of the clean sample).
- The velocity field $\hat{v}$ (representing the network's prediction of the true velocity, an approach used by flow matching, where $\hat{v} = v_\theta(X_t, t)$).

When these different types of predictions are adopted and optimized using a Mean Squared Error (MSE) loss, they can be seen as applying different time-dependent weightings to a noise prediction loss. Specifically, if we take the MSE loss for predicting the noise, $\|\hat{X}_0 - X_0\|_2^2$ (where X_0 is the true noise component), as a baseline, the losses for other parameterizations can be related as follows. Given $X_t = \alpha_t X_1 + \beta_t X_0$ and let $\lambda_t = \log(\alpha_t^2/\beta_t^2)$, we can derive the following:

- Predicting clean data: $\|\hat{X}_1 - X_1\|_2^2 = e^{-\lambda_t}\|\hat{X}_0 - X_0\|_2^2$.
- Predicting velocity: $\|\hat{v} - v_t\|_2^2 = \alpha_t^2(e^{-\lambda_t} + 1)^2\|\hat{X}_0 - X_0\|_2^2$.

This illustrates that various common objectives in diffusion and flow models can be expressed in a unified noise-prediction form with a different time-related function:

$$\mathcal{L}(\theta) = \mathbb{E}\left[w(t)\|\hat{X}_0 - X_0\|_2^2\right] \tag{2.17}$$

Here, $w(t)$ is a time-related weighting function determined by the specific scheduler (α_t, β_t) and the chosen network output parameterization.

Training with Different Time Sampling Strategies. During training, a random timestep is sampled for each training example to construct X_t. Some methods, such as DDPM [103], sample timesteps uniformly, while others, like EDM [126], use a non-uniform distribution. In theory, the choice of timestep sampling distribution does not affect the expected loss in the limit of infinite training iterations, as the expectation remains unchanged. However, in practice, training is finite, and the sampling distribution directly impacts optimization. This is closely related to Monte Carlo sampling, and is effectively equivalent to applying a timestep-dependent weighting to the loss function.

Unifying Takeaway for Training. The key takeaway from this analysis is that using different schedulers, network output parameterizations, and time sampling strategies is primarily equivalent to changing the time-related weighting $w(t)$ in the training loss. Theoretically, if training converges perfectly, these different formulations should lead to models that can represent the same underlying predicted noise or velocity field. However, in practice, this weighting $w(\lambda_t)$ can significantly influence the training dynamics and the final performance of the model. For example, the good results sometimes achieved with flow matching objectives might not solely be due to the "straightness" of the paths but rather due to the implicit time weighting these objectives apply, which may better suit certain data characteristics or network architectures.

2.2.3 Reintroducing Stochasticity: SDE Samplers

So far, for unification, we've largely considered the ODE formulation. However, diffusion models often benefit from being formulated as SDEs. We now reintroduce stochasticity into our unified framework. Importantly, this does not alter any of the training formulations or conclusions discussed earlier; the only difference arises during inference, specifically in the sampling process. That is, if we can show that an SDE sampler – as introduced in diffusion models – can be used in place of an ODE sampler to generate samples from any trained model (including those trained with flow matching objectives) without changing sampling

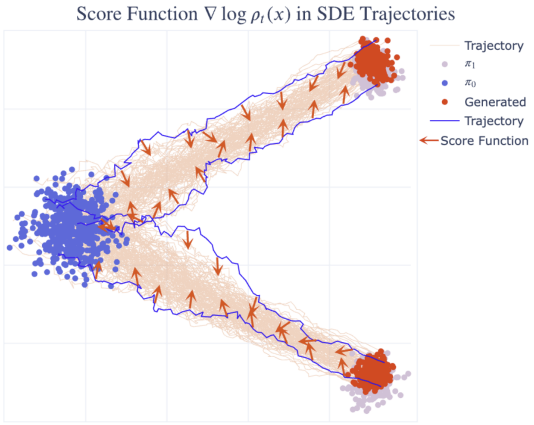


Figure 2.3: Visualization of error correction ability of the SDE sampler against ODE sampler. Image credit [170].

distribution, then we can naturally incorporate SDE-based samplers into our framework. This allows us to claim that our unified framework encompasses both diffusion models and flow matching, even in the presence of stochasticity as found in the SDE formulation of diffusion models.

Luckily, the theory of Langevin dynamics provides the theoretical grounding for this. It can be achieved by formulating an SDE sampler in a specific form that leverages the learned velocity field, given by:

$$dX_t = v_\theta(X_t, t)dt + \sigma(X_t, t)^2 \nabla \log \rho_t(X_t)dt + \sqrt{2}\sigma(X_t, t)dW_t, \quad X_0 \sim p_0 \tag{2.18}$$

where $\rho_t(X_t)$ is the marginal distribution of X_t. This SDE combines the learned velocity $v_\theta(X_t, t)$ with a stochastic term and a marginal distribution term inspired by Langevin dynamics. Langevin dynamics naturally drives samples towards regions of higher probability density, and adding these terms helps guide the process. Crucially, sampling with this SDE results in the same marginal distribution at each time step as the deterministic ODE path, ensuring consistency with the model's training objective. $\nabla \log \rho_t(X_t)$ in this SDE can be computed from the learned velocity $v_\theta(X_t, t)$ and the known parameters of sampling schedulers. In this way, we can flexibly switch between ODE and SDE schedulers as needed, which unifies our framework.

One question one may ask is: Why do we need an SDE sampler? Here, we discuss the advantage of SDE samplers over deterministic ODE solvers:

1. **Error Correction:** The stochastic term helps correct for accumulated errors from time discretization and neural network inaccuracies, making sampling more robust. Fig. 2.3 visualizes this process.

2. **Improved Sample Quality:** The added noise can aid in exploring the data manifold, potentially leading to higher-quality and more diverse samples, especially with fewer steps.

The ability to use SDE samplers highlights a key connection, reinforcing the unification of diffusion models and flow matching inference processes.

Chapter 3

TOTAL SELFIE

This chapter presents the research project *Total Selfie: Generating Full Body Images from Selfies* [33], conducted with Brian Curless, Ira Kemelmacher-Shlizerman, and Steven M. Seitz. The subsequent analysis and comparisons to related studies in this chapter are based on the prevailing state of the art at the time of this work.

The prevalence of selfies has skyrocketed in recent years, with an estimated 93 million taken each day. Despite their popularity, they suffer from multiple shortcomings: (1) they capture only the upper portion of the subject, (2) the close-up camera viewpoint distorts faces and requires awkward poses (e.g., with arm reaching out), and (3) it is difficult to compose a shot that optimally captures both the subject and the scene.

Instead, what if you could capture the full-body image that *someone else would take of you* in the scene? We call this a *total selfie*. As input, we require four selfies to cover different parts of your body, and a photo of the background that you would like to be composited into (Fig. 3.1). Based on this information, we generate convincing full-body photos of you in a specified target pose in the desired scene (Fig. 3.1 right). In practice, we automatically select candidate reference photos from your photo collection, allowing you to choose one or more of them to determine the target pose.

Solving this problem requires addressing a number of challenges. First, we must render a complete and accurate image of your body, piecing together separate close-up images of your face, upper torso, legs, and shoes. Second, we must reproject you to a virtual viewpoint from several feet away – far enough to compose your full body within the scene. And third, we need to render you with a desired target pose (where you're not holding the camera), which can be completely different from the one you used to take the selfies. The target pose can be specified by any full-body image from your photo collection. To facilitate target pose selection, we auto-detect photos from your collection where you are wearing similar clothing

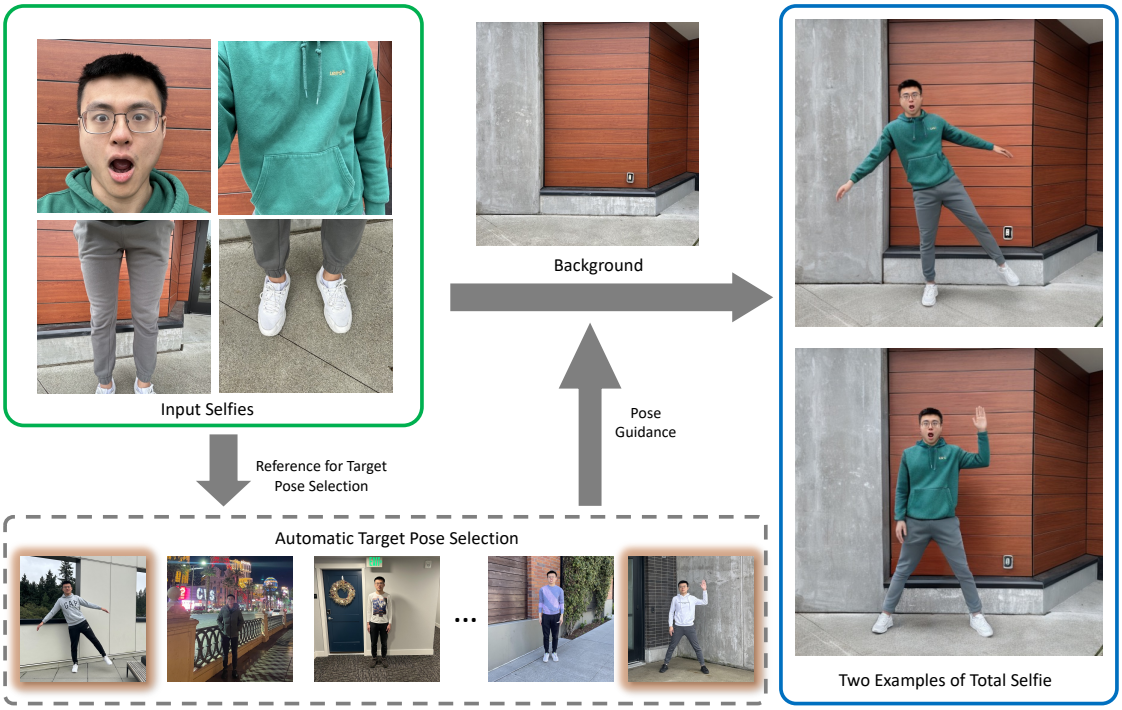


Figure 3.1: We generate full-body selfies of you (right), from self-captured images of your face and body (top left) and background. You can choose any target pose from a reference photo — we auto-select a set of good candidates from your photo collection (bottom).

to the input selfie set, leading to results with more accurate body shapes for a given type of clothing. Most importantly, the resulting composite must retain your identity, expression, and clothing, but be composited realistically into the target scene with the desired pose and appearance.

One approach to this problem would be to collect a paired dataset of selfies and full-body images of many people, and train a generative model on it. However, acquiring such a dataset would be time and cost intensive. Instead, we train a selfie to full-body model using a paired *synthetic* dataset, and further perform per-capture fine-tuning to narrow the gap between real and synthetic data. Specifically, we first introduce a diffusion-based inpainting model trained in a self-supervised manner. This model takes four selfies as input and inpaints a full-body subject into a masked background. Given a set of input images, we

first remove perspective distortion of the face selfie, and then fine-tune the trained model on these images to enhance the fidelity of generated full-body photo further.

Our contributions can be summarized as follows:

- We introduce a novel type of self-captured photo – *total selfie* – that captures your entire body, as if some one was taking a photo of you in the scene.

- We propose a diffusion-based full-body generation model, followed by per-capture fine-tuning techniques, to generate *total selfie* from four selfies covering the body, a background image, and an auto-selected reference image as target pose.

- We demonstrate results for twelve individuals in various scenes (*e.g.*, indoor and sunny outdoor) and clothing (*e.g.*, skirts) with a wide range of poses and expressions. Our experimental results outperform existing methods in generating realistic and accurate full-body images.

3.1 Related Work

Full-Body Image Generation. Extensive research has been dedicated to generating full-body images, either without specific conditions [68, 70, 281, 71] or with conditions such as pose [303], shape [223], or text prompts [119]. One related area of research to our task is human reposing [211, 66, 137, 115, 265, 43, 326, 339, 212, 54, 173, 213, 125, 15, 90, 149, 176, 224]. These methods transform a full-body (or partial-body) human image from one pose to another, with a target pose provided. For example, DisCo [265] designed a diffusion-based framework to achieve this by using a person image (in any pose), a background image, and a target pose. However, these methods are tailored for single image input and fall short when applied to our task due to the inherent limitation of a single selfie in capturing the full view of the person.

Another related line of work is Virtual Try-On [341, 47, 146, 289, 297, 154, 46, 38, 95], where the goal is to generate a visualization of how a garment might appear on a person, given the person image (in different clothing) and the garment image. For instance, LaDI-VTON [185] introduced the first Latent Diffusion textual Inversion-Enhanced model to syn-

thesize an image of the person wearing a specified clothing item. However, these approaches usually assume simple backgrounds, posing challenges in generating realistic shading within complex backgrounds. They also cannot alter footwear and facial expressions, crucial for our task.

Despite these challenges, both streams of work assume input from third-person view images, not selfies.

Selfie-Related Techniques. Numerous studies have explored selfies for applications like reposing [175], face recognition [142, 23], style transfer [151, 252], novel view synthesis [16, 194, 195, 8, 124], relighting [28], and video stabilization [311, 310]. For example, [175] proposed a coordinate-based method to transform a typical selfie, mainly focusing on the face, into a neutral-pose portrait. Nevertheless, their approach was limited to upper body selfies and could not generate full-body selfies capturing both the subject and the surroundings. Our work is the first to propose and generate *total selfie* from arm-captured selfies.

Diffusion Models. Diffusion models have recently demonstrated their success in various tasks such as text-to-image [209, 221, 217] and image-to-image translation [298, 89]. DreamBooth [219] was proposed to personalize a text-to-image model by fine-tuning the model on a few reference images. RealFill [248] introduced an image completion technique to outpaint an input photo using reference images from the same scene. It fine-tuned a text-to-image inpainting model using reference images and applied it to complete the input photo. However, RealFill assumed third-person view images as input and mainly focused on generating scene content, rather than full-body subjects. Another relevant work, Paint-By-Example [298], introduced an image-conditioned inpainting model to inpaint masked scenes with content specified in a reference photo. Similarly, we frame our problem as an exemplar-based inpainting problem. Thus we adapt this model to suit our settings.

3.2 Total Selfie

We refer to our pipeline as Total Selfie, and define our task setting more formally. As input, a user captures four selfies, including face I_f, upper body I_u, lower body I_ℓ, and shoes I_s, as well as the background image I_b. Total Selfie inpaints the full-body individual into I_b, with

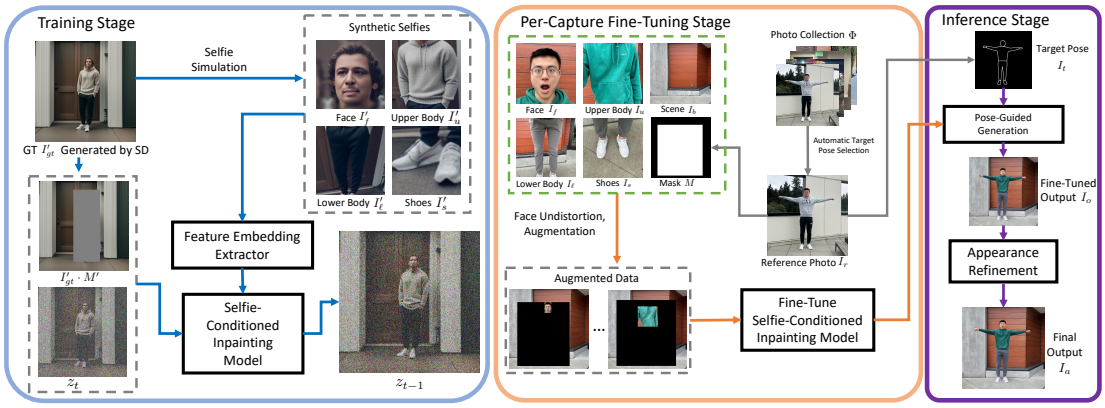


Figure 3.2: **Overview of Total Selfie.** First, we train a selfie-conditioned inpainting model based on a synthetic selfie to full-body dataset (blue box). Second, we fine-tune the trained model on a specific capture (orange box), and use it to produce a full-body selfie with the help of modified ControlNet (for pose) and appearance refinement (for face and shoes), visualized in the purple box. Note that, images in the green dashed box (inside the orange box) serve as input and conditional signals to the inpainting model, arrows omitted for simplicity.

the target pose I_t and inpainting mask M (where 1 indicates the region to be inpainted) specified by a reference image I_r. Here, I_r is automatically selected from the user's photo collection Φ.

Total Selfie has two main steps. As depicted in Fig. 3.2 left, we first generate a large paired dataset comprising four selfies as input and a corresponding full-body image as ground truth. Then we train a selfie-conditioned inpainting model on this dataset. Given an input capture, we perform preprocessing on the input images, including face undistortion and automated pose selection. These preprocessed images are then used to fine-tune the trained model (Fig. 3.2 middle). The fine-tuned model is employed to generate an initial output I_o, which is further refined to produce the final output I_a (Fig. 3.2 right).

3.2.1 Training Selfie-Conditioned Inpainting Model

Training the selfie-conditioned inpainting model involves two steps: (1) generating a large paired dataset and (2) training an image-conditioned diffusion model on this dataset.

Dataset Generation. We define one training pair as $\{(S', I'_{gt} \cdot M', M'), I'_{gt}\}$, where $S' = \{I'_f, I'_u, I'_\ell, I'_s\}$ is a set of four synthetic selfies for face, upper body, lower body, and shoes respectively. I'_{gt} is the ground-truth full-body image, and M' is the inpainting mask.

To create a pair, we start by generating I'_{gt} using Stable Diffusion [217], with the pose guided by OpenPose ControlNet [322]. The person's bounding box in I'_{gt} is then scaled up (following [298]) to generate the inpainting mask M'. We choose a bounding box representation instead of a more detailed shape for the mask since we anticipate changes in nearby regions, like those affected by shadows, when integrating the individual into the scene. Simulating selfie set S' from the third-person view I'_{gt} is non-trivial. One possible idea is to estimate 3D geometry of I'_{gt} using depth estimation [208, 14, 19] or human reconstruction [291] methods, and then re-render it from the perspective of a selfie camera. However, this is not practical due to inaccuracy of the estimated 3D geometry. Instead we propose a simpler yet effective way for obtaining S' using homography transformation. To create I'_u, we follow these steps. (1) Identify the *typical* positions of upper body OpenPose keypoints from pre-captured real upper body selfies (see supplementary). (2) Detect the upper body OpenPose keypoints from I'_{gt}. (3) Compute a homography transformation that maps keypoints in I'_{gt} to their typical positions. (4) Use this transformation to warp I'_{gt} and obtain I'_u. We apply the same approach to generate I'_ℓ and I'_s. Finally, we use FFHQ face alignment [129] to extract the face selfie I'_f. The resultant selfie set S' is visualized in the top right part of Fig. 3.2 blue box. We repeat this process to create multiple training pairs with I'_{gt} in diverse poses. Although the selfie simulation process primarily focuses on viewpoint correction and does not consider pose differences (*e.g.*, using one arm to hold camera), we observed that this is sufficient for training an inpainting model and obtaining a reasonable selfie-to-full-body prior.

Training. We start with an existing image-conditioned, diffusion-based inpainting model called Paint-By-Example [298] and tailor our specific task. We refer to our adapted model

as selfie-conditioned inpainting model.

To train our model, we first apply the forward diffusion process to I'_{gt}. Starting from a clean latent $z_0 = E(I'_{gt})$, the forward process yields a noisy latent $z_t = \alpha_t z_0 + (1 - \alpha_t)\epsilon$, where E is the encoder of Variational AutoEncoder used in Latent Diffusion [217]. Here, t is a randomly sampled timestep, ϵ represents Gaussian noise, and α_t is a weight parameter determined by t. We then use a diffusion model to denoise z_t and update the model parameters by minimizing the following loss function:

$$\mathcal{L} = E_{t,z_0,\epsilon}||\epsilon_\theta(z_t, I'_{gt} \cdot M', c', t) - \epsilon||_2^2, \tag{3.1}$$

where c' is the condition for the diffusion model, which, in our case, is the image embeddings extracted from S'. In Paint-By-Example, c' is implemented as $L \cdot F(I)$, where I is a single image, $F(\cdot)$ is a pretrained CLIP image encoder [206] followed by a multi-layer perceptron (MLP), and $L(\cdot)$ is a linear layer. In our model, we extend this idea and implement it as:

$$c' = L([F(I'_f), F(I'_u), F(I'_\ell), F(I'_s)]), \tag{3.2}$$

where $[\cdot]$ is the operation of concatenation. L is modified to adapt to the dimension of the concatenated embeddings. One advantage of $F(\cdot)$ is that it transforms an image into a highly compressed vector. This forces the network to understand the clothing type and color in S', preventing it from reaching optimal results by simply applying homography transformation to S' in training. The training process is visualized in the blue box in Fig. 3.2. In practice, we train our model by updating parameters (θ, weights in F's MLP and L) using Eq. 3.1. We initialize weights (excluding those in layer L) with pretrained weights from Paint-By-Example. This allows us to leverage the generative prior in this pretrained model and significantly reduce our training time.

3.2.2 Per-Capture Preprocessing and Fine-Tuning

We start by presenting two test time preprocessing steps: *face undistortion* and *automatic target pose selection*. Then we introduce how to generate results aligning with the target pose, and discuss the issue of generated results to motivate the per-capture fine-tuning.

At test time, the set of user-captured selfies $S = \{I_f, I_u, I_\ell, I_s\}$ has a different distribution from the simulated selfies S' in the training set. We note two particular differences: (1) real selfie I_f often exhibits significant face distortion, which is not present in the simulated selfie I'_f (obtained from full-body image), (2) upper body selfies I_u and I'_u have different poses, arising from the need to hold the camera out front (using an upper body arm) when taking a real selfie (see Fig. 3.2). To address the first issue, we propose a face undistortion strategy as a preprocessing step to help reduce the domain gap. We address the second issue, pose differences, with a fine-tuning step which we discuss later, as it is non-trivial to resolve during preprocessing.

Face Undistortion. Existing methods alleviate selfie distortion by either optimizing on a single image [265, 233] or by training a model on a combination of an unrealistic synthetic dataset and a small real dataset [331]. For test-time efficiency, we follow the latter idea. We render a large paired dataset using a method that generates realistic, textured 3D heads using 3D GANs [30]. Then we fine-tune a talking-head synthesis network [266] to perform perspective undistortion using the rendered dataset. See supplementary for more details and results. Additionally, we roughly align the shoes selfie I_s with I'_s by cropping it based on the bounding box of the shoes. For simplicity, we reuse I_f and I_s to represent corrected face selfie and cropped shoes selfie, respectively.

Automatic Target Pose Selection. Another preprocessing step is to obtain the target pose I_t to guide the full-body selfie generation. We require I_t to convey two types of information: (1) the desired pose and (2) the actual body shape. To achieve this, we represent I_t as the contour of the user's body, derived from a reference photo I_r of the *same* person. We first discuss how to obtain I_r and address the process of deriving I_t from I_r in the next section.

We develop an automatic selection strategy to help obtain I_r from the users' photo collection Φ. The selection criteria are based on the similarity between the clothing types in the input selfies and a candidate image in Φ. This is because the more similar the clothing type is, the more accurately the body shape (in this particular type of outfit) can be extracted from I_r. Specifically, we first use a pretrained human parsing model [301] to detect the clothing types (*e.g.*, hoodie) in the selfies I_u, I_ℓ, and I_s. We then apply the

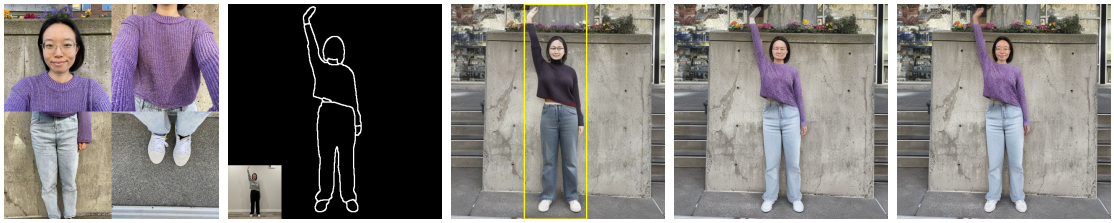
detection to each full-body photo in Φ. A list of candidate reference photos is suggested based on the number of matched clothing types (higher is preferred) between the selfies and the image from Φ. Then the user can choose a reference image I_r from the list. Finally, the inpainting mask M is obtained using the scaled bounding box of the person in I_r.

Pose-Guided Generation. After preprocessing, to generate a full-body selfie, we follow the standard diffusion denoising process with the guidance of ControlNet [322] to ensure that the generated image aligns with the pose in I_r. The key difference is that we modify the ControlNet architecture to enable it to possess similar guiding capabilities for our image-conditioned diffusion model as it does for text-conditioned models. Specifically, we replace the text embeddings with c, which is computed using Eq. 3.2 by replacing simulated selfie I'_k with real selfie I_k, where $k \in \{f, u, \ell, s\}$. This modification allows any pretrained ControlNet to be plugged into our model, providing the desired guidance. To meet our requirements, we specifically opt for a Canny Edge ControlNet with target pose I_t as the control signal. To obtain I_t, we segment various body parts in I_r using [301], producing a semantic map. The Canny Edge I_t is then detected from this semantic map (Fig. 3.2 top right), not directly from I_r itself. This ensures that the pipeline is not influenced by the outfit details in I_r, such as cloth texture. Note that, as we will discuss, one can always use OpenPose ControlNet, and obtain I_t (skeleton) from any human image, albeit sacrificing accurate body shape.

To obtain the full-body selfie, starting from a random noise z_T at timestep T, we iteratively perform one-step denoising using $\epsilon_\theta(z_t, I_b \cdot M, c, t)$ with modified ControlNet for T steps. This yields the clean latent z_0, which is further decoded by decoder D to generate the full-body selfie I_n. However, as shown in Figure 3.3 (c), directly using trained model in Sec. 3.2.1 produces inaccurate outfit and identity due to the gap between synthetic and real data.

Fine-Tuning. To improve results, we fine-tune the selfie-conditioned inpainting model on the specific capture. By leveraging the full-body generative prior in the model, this strategy is robust to distribution differences (*e.g.*, pose variations in the upper body) between S and S', enhancing the preservation of clothing texture and generating pose-specific details (wrinkles on upper clothing in Fig. 3.3 (d)).

To implement this strategy, we generate a "ground truth" for fine-tuning by resizing



(a) Input Selfies (b) Target Pose (c) Output w/o FT (d) FT Output (e) Full Pipeline

Figure 3.3: **Results for different modules of our pipeline.** Background image omitted due to space; regions inside bounding box (c) are to be inpainted. The Canny Edge image in (b) is detected from the reference image, inset. Generating without fine-tuning (c) produces inaccurate outfit and identity. Through fine-tuning, the pipeline (d) generates correct outfit with reasonable shading and clothing details (*e.g.*, wrinkles on upper cloth), but with wrong identity. With appearance refinement, the full pipeline (e) yields high-quality full-body selfies.

and placing a randomly selected selfie image from the set S into the masked area of the image I_b (see supplementary for details). The bottom left part of Fig. 3.2 orange box visualizes examples of two augmented images produced using I_f and I_u. We create 200 augmented images and employ them to fine-tune $\epsilon_\theta(z_t, I_b \cdot M, c, t)$ supervised by the loss computed similar to Eq. 3.1. While fine-tuning on close-up selfies, the full-body generative prior in the trained model helps prevent overfitting, especially for I_u with significant pose differences. Finally, we utilize the fine-tuned model to produce full-body selfie I_o using the pose-guided generation. Fig. 3.3 (d) shows an example of fine-tuned output, which has reasonable outfit and shading but wrong identity. The identity problem arises because the VAE used in Latent Diffusion fails to generate details of the small face in the full-body photo, a well-known limitation. Similar challenges arise with shoes because they are too small in I_o.

Appearance Refinement. To further enhance the details of I_o, we employ several post-processing strategies. First, for refining face and shoes, we augment I_f and I_s by resizing them to various resolutions and zero-padding the border. Subsequently, we train a Dream-

Booth model [219] with two concepts (shoes and face) using these augmented images. Finally, we crop the face region from I_o and then employ SDEdit [179], based on the trained DreamBooth, to refine the face. We subsequently compose the refined face back into I_o. A similar operation is applied to the shoes region. Second, we also apply the similar operation to hands region, but with a pretrained Stable Diffusion model [217] since hands are usually not shown in the input selfies. As shown in Fig. 3.3 (e) and Fig. 3.2 purple box, these post-processing steps culminate in the creation of the final output, denoted as I_a. This final output faithfully preserves the identity and outfit, and exhibits realistic shading, along with the desired pose.

At the end, the user can switch to a new reference photo and use the current fine-tuned model for generation, provided the person in the new reference photo is within the inpainting mask M. If not, fine-tuning for the new reference photo is required. See the supplementary for execution time and implementation details, including strategy to help preserve background content inside the inpainting mask.

3.3 Experiments

Results. We begin by showcasing the results of Total Selfie across five different captures, as illustrated in Figure 3.4. The goal is to retain the user's facial expression and clothing, but realistically retargeted onto the background with the desired pose; these results demonstrate this capability.

In particular, the results show compelling full-body selfies with a wide range of poses, even if the desired poses significantly deviate from those in the input selfies (row 2 to 4). Importantly, the method is able to add realistic wrinkles in the clothing to fit the new pose, with consistent shading (*e.g.*, raised arm in rows 2 to 4). Observe that the technique works with a range of clothing, from short sleeves to jackets, and both pants and skirts. Finally, the method is able to convincingly fill in missing details (*e.g.*, hands, which are often missing in selfies), that fit the individual and are realistically shaded in the target scene. More results are in the supplementary.

Evaluation Data. For evaluation, we collected a dataset of twelve people wearing various outfits with selfies taken in a variety of scenes and lighting conditions, resulting in a total of

(a) Input Selfies (b) Target Pose (c) Background (d) Total Selfie

Figure 3.4: **Results**. The second column shows the Canny Edge images detected from reference images (shown as insets). Regions inside yellow box of (c) are the masked regions.

Method	LPIPS ↓	SSIM ↑	PSNR ↑	FID ↓
Paint-By-Example	0.252	0.621	12.37	184.6
DisCo	0.287	0.586	12.13	245.7
LaDI-VTON	0.263	0.563	10.35	172.8
DreamBooth	0.224	0.663	13.61	159.2
Ours-FT-AR	0.218	0.691	14.19	154.1
Ours-AR	0.190	0.703	16.89	143.4
Ours-FU	0.188	0.706	16.98	141.6
Ours	**0.187**	**0.708**	**17.01**	**139.4**

Table 3.1: **Comparison with baselines and our ablation variants.** The metrics are evaluated only in the foreground. We employ ground truth Canny Edge to define the target pose for all rows.

seventeen captures. Additionally, we captured real, "ground truth" full-body photos of each subject, maintaining the same clothing, nearly identical facial expressions and background.

Ablation Study. We study the effects of different parts of the pipeline and design three variants: (1) *Ours-FT-AR*: full pipeline without per-capture fine-tuning and appearance refinement. (2) *Ours-AR*: full pipeline without appearance refinement, (3) *Ours-FU*: full pipeline without face undistortion. As discussed in Sec. 3.2, Fig. 3.3 and Table. 3.1 show that the full pipeline outperforms all tested variants.

Comparison to Baseline. To the best of our knowledge, there are no existing papers solving the same task. We therefore adapt four existing methods to create four baselines. (1) *Paint-By-Example* [298] was designed to inpaint a masked image using a single source image. Here, we concatenate four input selfies vertically as the source since this works better than only using one selfie. We use Canny Edge ControlNet [322] to guide the pose. (2) *DisCo* [265] reposes an input human image using diffusion models, given a background photo and a target pose. We opt for this over other reposing approaches because it can handle complex backgrounds. Similar as before, we concatenate four input selfies vertically to create

(a) Input Selfies (b) Background (c) LaDI-VTON (d) Dreambooth (e) Ours (f) Real Photo

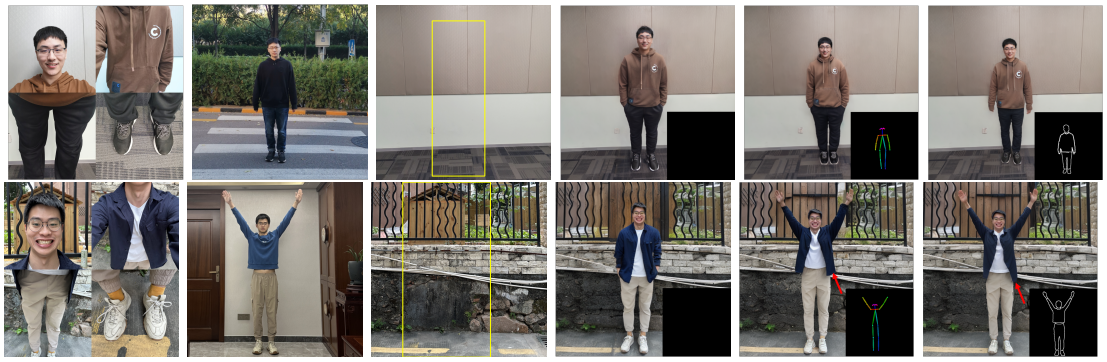
Figure 3.5: **Qualitative comparison with two best-performing baselines.** For this comparison, we used Canny Edge of the real photo as target pose (inset of (f)). Our pipeline clearly outperforms baselines in terms of photorealism and faithfulness (zoom in for details, including faces and shoes). Note that, while the selfies, background image, and real photo were captured in the same session, variations in lighting conditions, auto exposure, white balance, and other factors may result in intensity and color tone differences.

the input human image for better performance. Following the official implementation, we use OpenPose Skeleton as target pose since their provided model does not work well with Canny Edge input. (3) *LaDI-VTON* [185] is a diffusion-based Virtual Try-On method. We opt for this method over other similar methods since it can handle both upper- and lower-body garments, and the code is available. This method specifically requires a full-body RGB image as input. Thus we use Stable Diffusion and Canny Edge ControlNet to generate the full-body input aligning with target pose. Then we apply garment editing on this input image using upper and lower body selfies, excluding the face and shoes, as this method does not support editing those elements. Finally, we blend the edited image with the input background for the final output. (4) *DreamBooth* [219] customizes a pretrained text-to-image diffusion model by fine-tuning with a few reference images. Here, we use four selfies (with augmentation) as reference images and generate outputs using Canny Edge ControlNet.

Fig. 3.5 and Table. 3.1 show the qualitative and quantitative results of the compared methods, respectively. Please note that real photos may have contrast and color tone variations (e.g., due to auto-exposure and white balance), leading to differences between real and generated outputs. Among the compared methods, *DisCo* performs worst because it is guided by skeleton and assumes a third-person view input. *Paint-By-Example* is also ineffective since it is not designed specifically for full-body generation. Further, both methods can only consider one reference image, leading to sub-optimal results. *LaDI-VTON* exhibits artifacts on the clothes (Fig. 3.5 (c)), which may be attributed to the reference garment images being selfies, distinct from those in its training set. *DreamBooth* produces inaccurate outfits (Fig. 3.5 (d)). This is due to the pretrained text-to-image model lacking strong priors for understanding clothes in selfies and linking them together as a coherent full-body image. Conversely, Total Selfie, which is initially trained on a synthetic dataset and then fine-tuned per capture, excels in realistically generating full-body selfies.

Discussion of Target Pose Options. We explore the trade-offs of different target pose options, as shown in Fig. 3.6. *No Condition* is the simplest option, requiring no target pose input. Our pipeline automatically determines pose, body shape, and scale from the masked background and input selfies. However, it generates inaccurate body shape and scale (row 1(d)), and lacks pose controllability. *OpenPose Skeleton* enables users to specify the target pose (skeleton) using any human photo. Our pipeline integrates this control signal using a modified OpenPose ControlNet. While this option produces reasonable poses, it struggles with correct body shapes (row 1(e)). *Canny Edge* offers accurate body shape and pose but may yield sub-optimal results when the reference pose photo has different clothing than the input selfies. For instance, the hem of the jacket in row 2(f) appears less natural. In such cases, *OpenPose Skeleton* (row 2(e)) can do a better job with clothing contours. In summary, each option has its advantages and limitations, and there is no one-size-fits-all choice. The selection of the target pose type depends on users' needs and available data.

Limitations and Future Work. Fig. 3.7 shows two limitations of Total Selfie: (1) While our method generally yields a harmonized output (b), the shading of the body may not precisely align with that in the actual photo (c). (2) Our method cannot accurately generate hard shadows of a person under strong sunlight since inferring the sun's direction



(a) Input Selfies (b) Ref Photo (c) Background (d) No Cond. (e) Skeleton (f) Canny Edge

Figure 3.6: **Discussion of target pose options.** Insets of (d)-(f) show the conditional signals. In the first row, Canny Edge (f) exhibits the closest body shape to the reference photo (b) compared to (d) and (e). In the second row, when the reference photo has a different clothing type, Canny Edge (f) produces an unnatural hem of the jacket (highlighted by a red arrow). In such cases, the OpenPose skeleton (e) may offer a more natural result despite a slight fattening of the waist compared to reference.

(a) Input Selfies (b) Ours (c) Real Photo

Figure 3.7: **Failure case of shadow generation under strong sunlight.** Masked regions are shown in (b), and target pose is from (c).

and scene geometry solely from the background is difficult.

Topics of future work include exploring how to effectively infer body shape and scale from input selfies, and automatically suggesting good target poses.

Conclusions. We introduce a new selfie type called *total selfie*, and propose a diffusion-based framework to generate it from four selfies, a background image, and a target pose. Our method generates faithful and realistic full-body selfies, outperforming existing techniques.

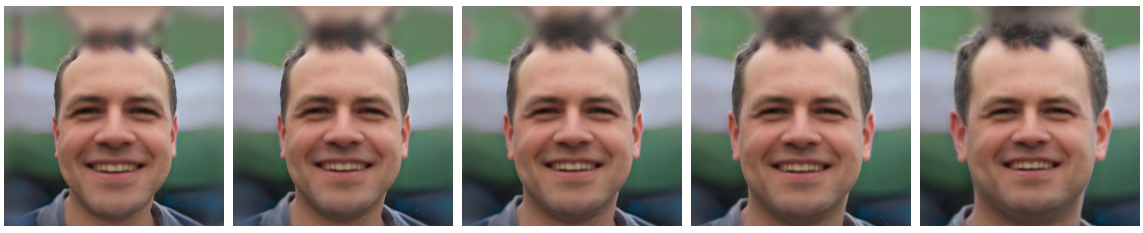
SUPPLEMENTARY MATERIAL

3.4 Face Undistortion

A common problem with face selfies is perspective distortion. This is caused by the camera being too close to the subject, resulting in facial features closer to the camera appearing larger and those farther appearing smaller, thereby creating an unnatural and distorted appearance.

Previous studies have addressed this issue either through single-image optimization [265, 233] or training on a combined dataset of real and unrealistic synthetic images [331]. For test-time efficiency, we follow the idea of large dataset training. This includes two steps: (1) generate high-quality paired dataset with distorted and undistorted face images. (2) Train a network on this dataset.

The goal of the first step is to render a pair of images with small and large camera-subject distance. To implement this, we adopt EG3D [30], a state-of-the-art textured 3D head generation method. EG3D utilizes a random noise vector and camera parameters to generate tri-planes, which can then be employed for volumetric rendering to produce color images and meshes. One straightforward idea for pair generation is to fix the random noise



(a) $d = 1$ (b) $d = 1.3$ (c) $d = 1.6$ (d) $d = 1.9$ (e) $d = 10$ (GT)

Figure 3.8: Example of 4 pairs of training data rendered from one textured mesh. The left 4 columns are the input images, and the last column is their ground truth.

vector and adjust the camera parameters to directly render desired RGB images. However, this is not feasible as EG3D is pre-trained on a dataset with a specific camera-subject distance. Consequently, rendering images with out-of-distribution camera-subject distances results in noticeable artifacts.

Instead, we create a textured 3D head mesh using EG3D and render head images at varying distances through traditional rendering. Specifically, we employ EG3D to create tri-planes, which are utilized to sample the volume, producing a cube with dimensions $H \times W \times C$ containing density and color values. The surface of the head (including background) is then extracted as a mesh using the Marching Cubes algorithm [171]. For each 3D surface vertex, the vertex color is determined by assigning the color value of the nearest point on the cube. With the textured mesh in place, we proceed to render images at varying distances using conventional rendering techniques. The camera rotation matrix is fixed, and only the camera distance d is adjusted. To maintain consistent eye positions across different images of the same mesh, the focal length f is computed based on the camera distance, given by:

$$f = df_0, \tag{3.3}$$

where $f_0 = 2.9$ represents the pre-defined focal length for rendering images without invalid pixels (*i.e.*, ensuring that all camera rays can hit the mesh) when $d = 1$. We use PyTorch3D to render 4 input images with severe distortion by setting d to 1, 1.3, 1.6, and 1.9. Additionally, a shared ground-truth image is rendered with d set to 10. For better alignment, all rendered images are processed using the FFHQ face alignment technique proposed by [129]. Fig. 3.8 shows 4 training pairs derived from a single textured mesh. In total, we generate 10,000 textured meshes, each yielding four training pairs, resulting in a dataset comprising 40,000 training pairs.

The next step is to train an undistortion network using the rendered dataset. For this, we adapt an existing method called facevid2vid [266]. This method uses a source image and a driving image to synthesize a talking-head image with appearance and head pose derived from the source and driving images respectively. For our task, both the source and driving images are the image with severe distortion, and the output image is the undistorted image, which will be supervised by our rendered ground-truth. The facevid2vid consists of

a couple of face feature extractors that can be applied to any face image regardless of the downstream task. In order to harness this power, we choose to fine-tune the pretrained model on our dataset instead of training from scratch.

During inference, given a face selfie I_f, we initially align it and subsequently utilize the fine-tuned network for perspective undistortion. Fig. 3.9 shows results of perspective undistortion on the face selfies. Following [266], we set learning rate as 2e-4, and batch size as 8 for training.

3.5 Implementation Details

We present the implementation details, and the code will be publicly available after acceptance. Following Stable Diffusion [146], all images in our pipeline are square and share a consistent resolution of 512.

3.5.1 Dataset Generation

We define one training pair as $\{(S', I'_{gt} \cdot M', M'), I'_{gt}\}$, where $S' = \{I'_f, I'_u, I'_\ell, I'_s\}$ is a set of four synthetic selfies for face, upper body, lower body, and shoes respectively. I'_{gt} is the ground-truth full-body image, and M' is the mask indicating the region to be inpainted.

We employ RealisticVision [228] as the pretrained Stable Diffusion with OpenPose ControlNet v1.1 [322] to generate I'_{gt}. The guidance scale is set to 7.5, the denoising step is 20, and the ControlNet scale is 1.0. OpenPose Skeleton images, used for guidance, are detected from a subset of the Human Bodies in the Wild dataset [48]. Fig. 3.10 illustrates three examples of these pose images. The text prompt used is "a [gender], [place], [upper], [lower], [shoes], standing, front-facing, RAW photo, full body shot, 8k uhd, high quality, film grain". Here, [gender] can be man or woman, [place] includes common indoor and outdoor locations such as beach, park, street, restaurant, cafe, shopping mall, *etc.* [upper] comprises shirt, hoodie, sweater, jacket, *etc.*, while [lower] includes jeans, leggings, shorts, *etc.*, and [shoes] covers sneakers, heels, boots, flats, etc. Additionally, we use CodeFormer [338] to enhance the details of face regions in the generated I'_{gt}.

After obtaining the face-refined I'_{gt}, we utilize the human parsing network [301] to generate the semantic map of I'_{gt}. This map is then employed to extract the bounding box of

Face Selfie | Aligned Face Selfie | Undistored Face Selfie

Figure 3.9: Results of our trained perspective undistortion network. Given a face selfie (left), we first align it (middle), and then correct the perspective distortion (right).

the person. The bounding box is scaled up following the strategy outlined in [298] to obtain M'.

For the real selfies utilized in detecting typical keypoints for selfie simulation, we gather

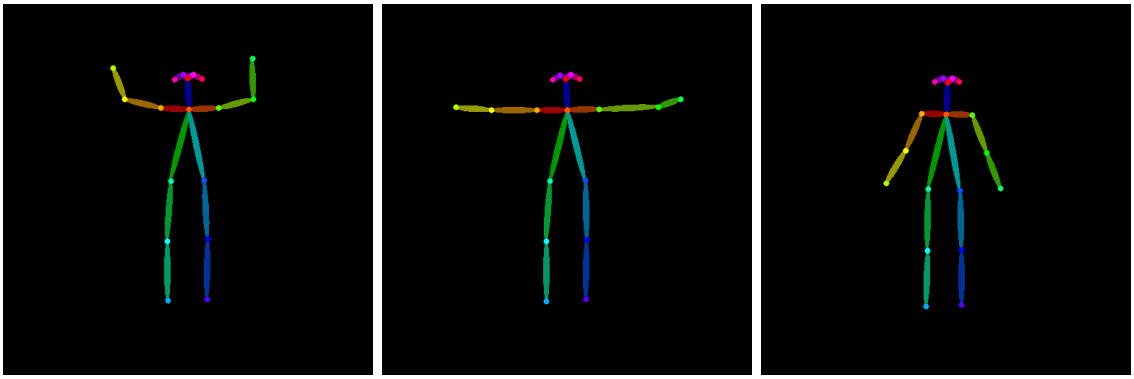


Figure 3.10: Examples of OpenPose skeleton images we use to generate ground truth full body image.

sets of upper body, lower body, and shoes selfies, each comprising 10 examples. Fig. 3.11 illustrates one example from each pre-captured selfie category.

In total, we create a training dataset consisting of 39,816 pairs, encompassing diverse individuals, clothing types, poses, and backgrounds.

3.5.2 Training

We initialize all network weights using the pretrained model provided by Paint-By-Example [298], except for the adapted linear layer L (zero-initialized). For training, we set the learning rate as 1e-5 and batch size as 12. We train the model for 9 epochs, taking around 36 hours on 3 NVIDIA A100 GPUs.

3.5.3 Automatic Target Pose Selection

We develop an automatic selection strategy to help obtain I_r from the users' photo collection Φ. The selection criteria are based on the similarity between the clothing types in the input selfies and a candidate image in Φ. This is because the more similar the clothing type is, the more accurately the body shape (in this particular type of outfit) can be extracted from I_r.

Specifically, we begin by utilizing the pretrained human parsing model [301] to obtain

Upper Body Lower Body Shoes

Figure 3.11: Examples of in-the-wild selfies used to detect typical keypoints.

the semantic map of selfies I_u. We filter out semantic labels that occupy less than 21 pixels and labels that do not belong to upper cloth (*e.g.*, pants, face). This results in a set of upper body labels, denoted as P_u. The same process is applied to obtain lower body labels P_ℓ and shoes labels P_s. For a full-body reference photo in Φ, we use the same network to obtain its semantic map. We filter out semantic labels that occupy less than 5 pixels and labels that do not belong to the upper body, lower body, and shoes. The resulting set is denoted as P_r. The matching score of each photo in Φ is computed by $P_r \cap (P_u \cup P_\ell \cup P_s)$. Then we rank the candidate references based on the matching score, with higher scores indicating better matches. Once reference photo I_r is selected, we obatin inpainting mask M using the bounding box of I_r, scaled up by 1.1 times by default.

Fig. 3.12 (a) to (d) visualizes an example of the semantic map of selfies and the selected reference photo.

3.5.4 Pose-Guided Generation

For generation, we set $T = 50$ and use DDIM scheduler for denoising. The ControlNet scale is set to 1.0. In cases where the target pose involves spreading arms, the mask M might sometimes become too large, potentially leading to the failure of preserving background content in I_b. To help alleviate this issue, we apply the following strategy during the

(a) Upper Body (b) Lower Body (c) Shoes (d) Ref Photo

Figure 3.12: Visualization of automatic target pose selection. The top row shows the input selfies and candidate reference photos, and the bottom row shows their segmentation results.

denoising process. We dilate the foreground mask (the finer mask containing only the human body) by 21 pixels, denoted as $\bar{M}$. Then, at each denoising timestep t, we compute the denoised latent as:

$$z_{t-1} = \begin{cases} z^f_{t-1}, & \text{if } t \leq sT \\ z^f_{t-1} \cdot \bar{M} + z^b_{t-1} \cdot (1 - \bar{M}), & \text{if } t > sT \end{cases}, \tag{3.4}$$

where z^f_{t-1} is the foreground latent obtained using the selfie-conditioned inpainting model (following the same process discussed in the main paper). z^b_{t-1} is the background latent obtained by adding noise to I_b by $t-1$ steps using DDIM scheduler. We set $s = 0.4$. This enables the generation of details in the surrounding area (*e.g.*, shadows) based on the inpainting model in later timesteps (smaller t), while reasonably preserving background content in the earlier timesteps (larger t).

3.5.5 Fine-Tuning

We generate a "ground truth" for fine-tuning by resizing and placing a randomly selected selfie image from the set S into the mask region M of the background image I_b. The full-body inpainting prior learned by the trained model is used to determine where and how to place the selected selfie image into the masked I_b.

Specifically, we first generate full body selfie I_n without any pose as condition. This is achieved by using the same process as Pose-Guided Generation but omitting the modified ControlNet. Suppose the upper body selfie I_u is selected for augmentation. We extract the bounding box of the upper body in I_n based on the semantic map of I_n detected by the human parsing network [301]. Then we resize I_u to have the same height as this bounding box. The resize operation keeps the aspect ratio of I_u unchanged to avoid using an image with the wrong scale. Finally, we paste this resized I_u to the masked I_b, ensuring the center of resized I_u and the bounding box are the same.

In practice, we generate 20 different I_n as the candidate pool for augmentation. We then repeat the above augmentation process (resizing and pasting) 200 times, each time with I_n randomly chosen from the candidate pool, resulting in a dataset of 200 augmented images. For fine-tuning, we set the learning rate to 5e-6 and the batch size to 4. The model is fine-tuned for 400 steps, taking around 10 minutes on a single NVIDIA A40 GPU.

3.5.6 Appearance Refinement

To train the DreamBooth (with two concepts) using only one image for each concept, we need to augment them to avoid overfitting. Specifically, we randomly resize the face selfie I_f from a resolution of 350 to 450 and apply random zero-padding to create the augmented image with a resolution of 512. The same operation is performed for the shoes selfie I_s, but with resizing resolution from 400 to 500. We generate 50 augmented images for I_f and I_s respectively. Then, we train a DreamBooth with two concepts using these two kinds of augmented images. Specifically, we set the training text prompt for face and shoes as "a sks face" and "a hta shoes" respectively. We set the learning rate as 5e-6, batch size as 4, and fine-tune the RealisticVision Stable Diffusion model for 300 epochs, taking around 5

minutes on a single NVIDIA A40 GPU.

To perform refinement, we use the pretrained human parsing model [301] to obatin the face (shoes) region in I_o, and use SDEdit based on trained DreamBooth to edit the face (shoes). We then paste the cropped, edited image back onto I_o. The same process is applied (using the pretrained Stable Diffusion model) for the hands regions (left and right hands) since hands are often invisible or incomplete in selfies.

3.6 Experiments

3.6.1 Dataset

We provide details on the data capture process for our pipeline. In a specific site, the user is requested to capture five square images: face, upper body, lower body, shoes, and background. Firstly, the user takes a face and upper body photos using the front camera, holding the camera with either one or two hands. Subsequently, the user switches to the rear camera to capture the lower body, shoes, and background photos. The entire process usually takes less than 20 seconds.

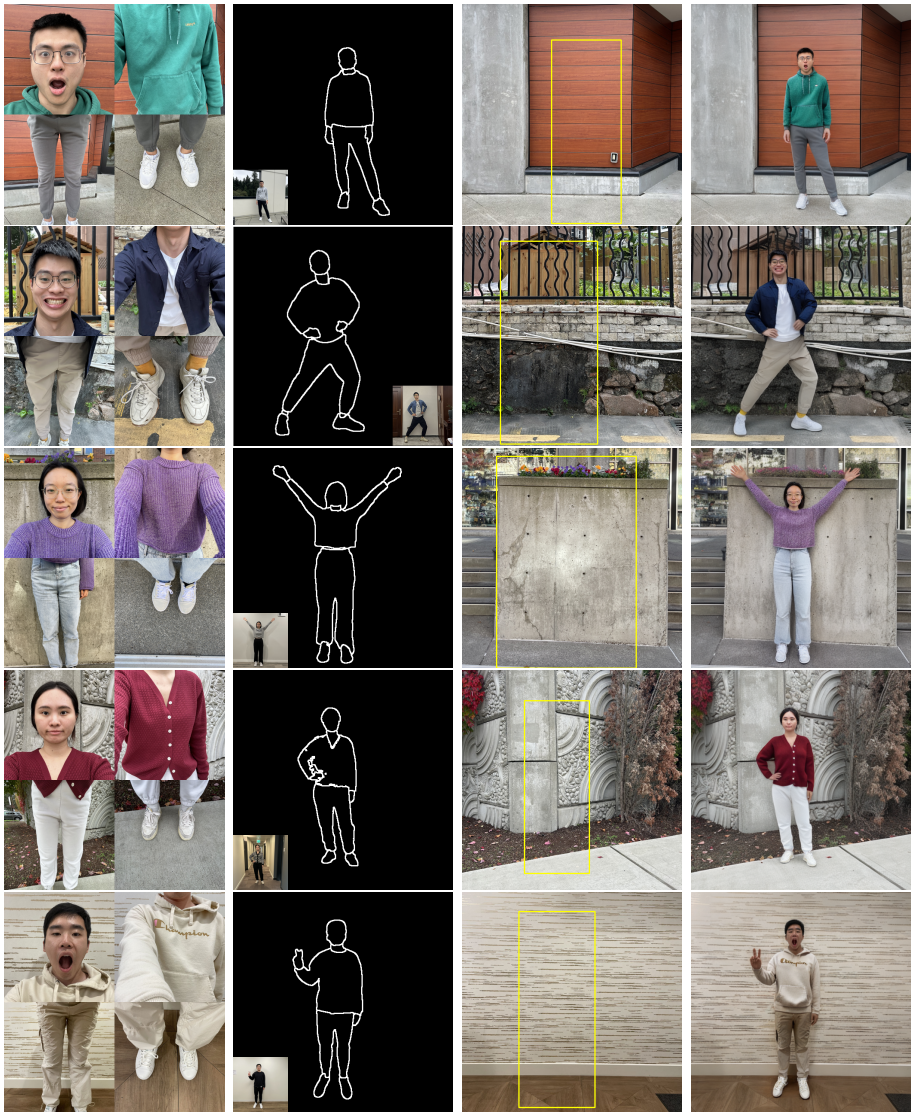
To obtain the real photo as the "ground truth," we have another person take a full-body photo of the user in a desired pose. This process should ensure that the real photos maintain the same clothing, nearly identical facial expressions, and background.

3.6.2 Results

We show more results of Total Selfie in Fig. 3.13. Total Selfie can produce high-quality full-body shots in diverse backgrounds, poses, outfits, and expressions, all while maintaining reasonable shading and composition.

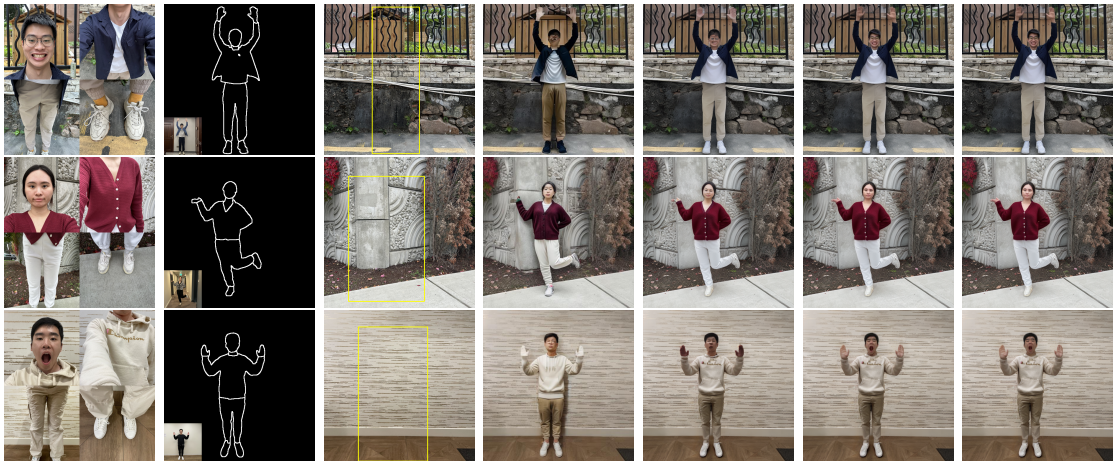
3.6.3 Ablation Study

Fig. 3.14 shows additional results of the ablation study, further demonstrating the effectiveness of our final design.



(a) Input Selfies (b) Target Pose (c) Background (d) Total Selfie

Figure 3.13: Results. The second column shows the Canny Edge images detected from reference images (shown as insets). Regions inside yellow box of (c) are the masked regions. Total Selfie generates realistic, full-body images of different individuals with diverse poses and expressions against a variety of backgrounds, while preserving facial expression and clothing.



(a) Input Selfies (b) Target Pose (c) Background (d) Ours-FT-AR (e) Ours-AR (f) Ours-FU (g) Ours

Figure 3.14: Results for different modules of our pipeline. The Canny Edge image in (b) is detected from the reference image, inset. Regions inside the bounding box (c) are to be inpainted. Generating without fine-tuning and appearance refinement (d) produces an inaccurate outfit and identity. Through fine-tuning, the pipeline (e) generates the correct outfit with reasonable shading but with the wrong identity. Without face undistortion, (f) generates a face with more perspective distortion (*i.e.*, exaggerated facial features), zoom in for details. In contrast, the full pipeline (g) yields high-quality full-body selfies.

3.6.4 Baseline Comparison

Fig. 3.15 shows a comparison with all baselines. Total Selfie can produce high-quality full-body shots in diverse backgrounds, poses, outfits, and expressions, all while maintaining reasonable shading and composition.

(a) Input Selfies (b) Background (c) PBE (d) DisCo (e) LaDI-VTON (f) DreamBooth (g) Ours (h) Real Photo

Figure 3.15: Qualitative comparison with all baselines. For all methods (except for DisCo), we used the Canny Edge of the real photo as the target pose (inset of (h)). For DisCo, we used OpenPose Skeleton of the real photo as the target pose. Our pipeline clearly outperforms baselines in terms of photorealism and faithfulness (zoom in for details, including faces and shoes). Note that, while the selfies, background image, and real photo were captured in the same session, variations in lighting conditions, auto exposure, white balance, and other factors may result in intensity and color tone differences.

Chapter 4

GENERATING FIT CHECK VIDEOS WITH A HANDHELD CAMERA

This chapter presents the research project *Generating Fit Check Videos with a Handheld Camera* [35], conducted with Brian Curless, Ira Kemelmacher-Shlizerman, and Steven M. Seitz. The subsequent analysis and comparisons to related studies in this chapter are based on the prevailing state of the art at the time of this work.

With the rise in popularity of video platforms like Tiktok and Instagram Reels, self-captured video has become a major industry. However, taking high quality *full-body* videos of yourself is not straightforward. Anyone who has tried propping up their phone on a table for a selfie, knows how difficult it is to frame a shot this way. For social media professionals, the most common solution is to use a tripod-mounted camera, but this introduces its own friction: creators must carry extra gear, guess at their framing without a reliable viewfinder, and often retake videos multiple times to achieve a desirable result [7, 6, 200].

It would be much easier to stand in front of a full-length mirror and capture video with your mobile phone. However, this introduces other problems: the camera is visible, the framing is awkward (Fig. 4.1 top left), you have limited ability to move around, and the viewpoint moves with you. These limitations highlight the need for a more convenient and user-friendly solution for full-body video self-capture.

Our goal is to enable full-body selfie video capture with a handheld mobile phone. We employ a full-length mirror to capture front and back selfie photos, for a range of applications but with a particular focus on "fit check" videos (influencers trying on outfits) popular on social media. Specifically, we provide the following capabilities:

- full-body self-capture with a handheld mobile phone that appears as if the camera was mounted on a tripod

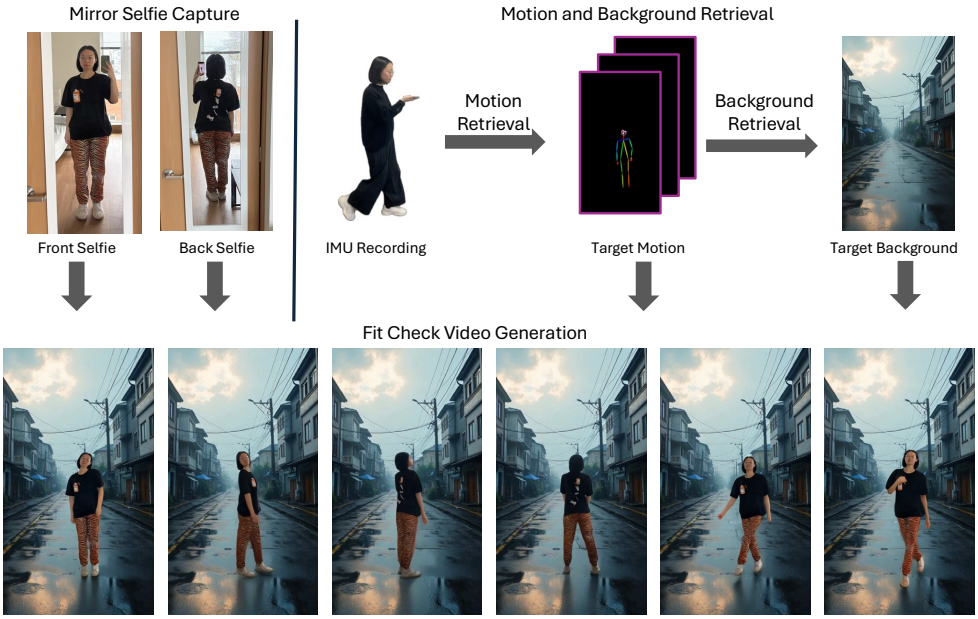


Figure 4.1: From two mirror selfie photos (top-left), we generate a photorealistic video of you performing a desired motion against a compatible desired background (bottom), with realistic shading, shadows, and reflections. The motion is captured using your mobile phone's IMU sensors (top-right). A target use case is self-captured "fit check" videos of you showing off an outfit.

- video output generated from image captures (*i.e.*, mirror stills) – eliminating the need to record full-body video
- accurate rendering of both the subject's front and back
- virtual backgrounds with consistent shading

In short, we seek to provide the video you would have achieved with a tripod-mounted camera and lots of practice in your desired environment.

Our solution is to separately capture your appearance (in a mirror), your movement (from phone sensors), and a compatible background, and then *compose* them together into

a desired video (Fig. 4.1). All steps are designed to be easy and quick to achieve with a standard handheld mobile phone.

Solving this problem involves several challenges. First, we need to generate a video from two mirror stills (selfies) and motion information; the video should maintain temporal consistency, support human poses that differ from those in the input selfies, and accurately preserve the subject's outfit and face – all rendered from a stationary third-person viewpoint. Second, the motion capture process must be simple and achievable using only a handheld phone – this is especially challenging, as your body pose is high-dimensional and not fully observable from the phone that you are carrying in your hand. Third, you should be composited into the new background with realistic shading, shadows, and reflections.

Our solution for motion self-capture is to leverage the inertial sensor (IMU) present on today's mobile phones. Because your full body pose and motion are not typically fully observable from your handheld phone, we rely on a database of pose sequences and seek to retrieve the best match from IMU data. Once a motion is selected, we retrieve the top-k backgrounds with the closest ground plane orientation to that of the selected motion's background, ensuring motion–background compatibility (Fig. 4.1, top right).

Given the motion and background, a straightforward approach to generating fit check videos is to design a video diffusion-based human animation model that takes two mirror selfies, a target motion, and a background image as input. However, adapting existing human animation methods [327, 254] faces two challenges: (1) they do not support additional reference image inputs (*i.e.*, back selfies), and simple modifications to include them lead to outfit inaccuracies; and (2) they often produce low-quality frames with blurriness and poor human-scene composition (*e.g.*, weak reflections on reflective floors). To overcome these issues, we introduce a novel, parameter-free frame generation strategy and a multi-reference attention mechanism that jointly enable effective use of all input selfies. Furthermore, we propose an effective image-based fine-tuning strategy using a synthesized, task-specific image dataset to enhance sharpness, reflections, and shadow rendering while maintaining temporal consistency. Our contributions are as follows:

- We demonstrate a novel application to capture full-body, fixed-viewpoint fit-check

videos from handheld mobile phones. Our approach supports customized motion capture and matching, and virtual background with reasonable shading and shadows.

- We design a parameter-free frame generation strategy, as well as a multi-reference attention mechanism, to effectively integrate multiple reference images into human animation methods based on video diffusion models.

- We introduce an image-based fine-tuning strategy that enhances frame quality, improving sharpness, reflections, and shadow rendering. This architecture-agnostic strategy can be applied to other human animation models.

4.1 Related Work

Diffusion Models for Human Image Animation. Human image animation aims to animate a single static human image to generate a human video. In this section, we review diffusion-based approaches, as they have shown superior generation quality compared to GAN-based methods [29, 213, 234, 235, 272, 292, 315, 324, 329].

Diffusion-based approaches for human image animation can be categorized into two main types. The first category [343, 106, 134, 193, 163, 296, 295, 31, 108, 109, 276, 275] builds on top of an image diffusion model [216], where additional temporal layers, appearance and pose encoding strategy are designed based on the image diffusion model to ensure the temporal consistency, appearance and pose fidelity. For example, Animate Anyone [106] introduced ReferenceNet to inject reference image features into diffusion models and employed a pose guider (skeleton-based) to ensure accurate motion. Training occurred in two stages: first, diffusion model was fine-tuned on image pairs without temporal constraints; second, the added temporal layers (initialized by AnimateDiff [84]) were optimized using multi-frame videos. Champ [343] extended this approach by incorporating SMPL-based pose signals [198] for improved alignment. However, these methods suffer from jitter and temporal inconsistencies due to their reliance on image diffusion models, requiring extra training for temporal layers.

The second category [254, 327, 262, 245, 267, 229, 156, 268, 260, 270, 92, 61, 337,

269] builds on video diffusion models [20, 304, 259] with stronger temporal consistency. StableAnimator [254] proposed an video diffusion-based pipeline with an identity-aware appearance controller to generate videos with improved identity preservation, conditioned on skeleton input. MimicMotion [327] introduced a confidence-aware pose guidance mechanism (in skeleton format) and regional loss amplification to enhance video quality and reduce distortions. However, these methods tend to produce frames that are slightly blurrier than image diffusion-based approaches.

In summary, both methods assume a single front-facing reference image without motion customization. We build on MimicMotion for our task due to its strong appearance and temporal consistency and its wide adoption in the community [55, 133]. We redesign the frame generation strategy and attention mechanism and introduce a fine-tuning stage, enabling the use of multiple reference images and producing sharper, higher-quality frames. While based on MimicMotion, our pipeline can be readily adapted to other video diffusion-based methods with minimal changes.

Personalization Technique. This line of work personalizes foundation models for image [219, 73, 143, 220, 307, 263, 153, 97] and video generation [274, 278, 203, 96, 316, 141, 186]. Personalized video synthesis methods generate videos from reference images, primarily using text prompts to control motion, background, and appearance. In contrast, we enable finer control by incorporating a spatially aligned per-frame pose sequence and a specified background.

Selfie-Related Applications. Numerous studies have studied selfies for tasks like reposing [175], face recognition [23, 142], style transfer [151], novel view synthesis [8, 16, 124, 34], but none address mirror selfies for full-body animation.

4.2 Method

Task Definition. As input, the user captures two full-body mirror selfies – front view I_{fr} and back view I_{bk} – and records motion using a mobile phone's IMU. This motion is used to retrieve a target pose sequence, $P_{1:N}$, where N is the sequence length. Finally, a target background I_{bg} is retrieved. Given these inputs, we generate a fit check video, $\hat{V}_{1:N}$, with the person following the target poses within the target background (Fig. 4.2 right).

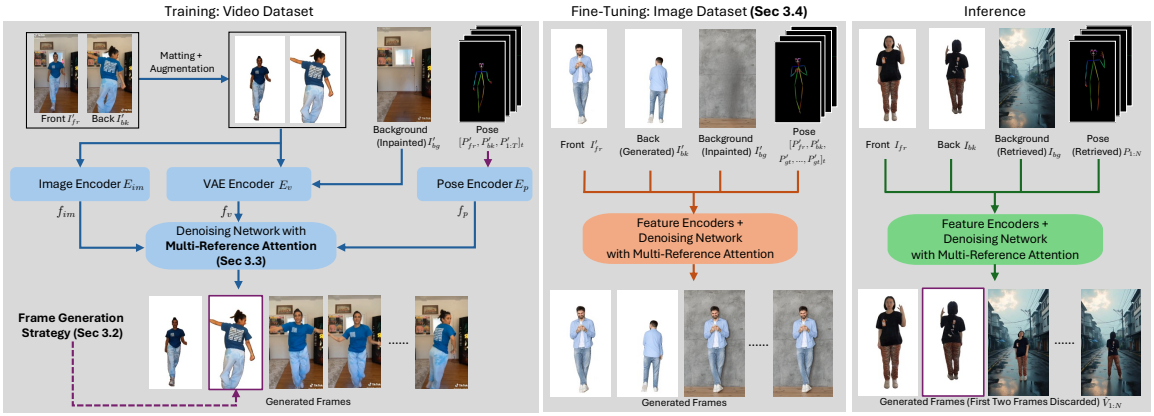


Figure 4.2: **Method Overview.** *Left*: We train our model on a fit-check video dataset using pairs of front and back images, GT video frames, an inpainted background, and a pose sequence.[1] Our frame generation strategy and multi-reference attention effectively encode features of multiple reference images. *Middle*: We fine-tune the trained model on a high-quality image dataset, supervising the generated frames using the input front or back image to enhance frame quality. *Right*: During inference, the method takes front and back selfies, a retrieved pose sequence, and a retrieved background as input, generating a video with the first two frames removed. The VAE decoders are omitted.

Training Data for Human Animation. As no dataset contains paired mirror selfies and fit check videos, we build our dataset using fit check videos taken from third-person viewpoints. We found this data sufficient for training the model to handle selfie inputs effectively. Each training pair consists of an input set $\{I'_{fr}, I'_{bk}, I'_{bg}, P'_{1:T}\}$ and its corresponding ground-truth (GT) video $V'_{1:T}$. Here, $V'_{1:T}$ comprises T consecutive frames sampled from a fit check video $V'_{1:N}$ $(N \geq T)$, while $P'_{1:T}$ represents the corresponding extracted poses in DWPose [302] format. The background image I'_{bg} is extracted under the assumption of a static camera, with occluded regions (not visible across all frames) inpainted using Stable Diffusion [216]. The front and back view images, I'_{fr} and I'_{bk}, are randomly sampled from $V'_{1:N}$ and have their backgrounds removed using [210], as the original background is not needed during testing.

Next, we first discuss the human animation component, assuming poses and background have been retrieved. We begin with a naive animation method and its limitations (Sec. 4.2.1), which motivate our frame generation strategy (Sec. 4.2.2) and multi-reference attention (Sec. 4.2.3). Next, we introduce an image-based fine-tuning strategy to enhance frame quality (Sec. 4.2.4) and conclude with retrieval (Sec. 4.2.5). Fig. 4.2 provides an overview of our method.

4.2.1 Naive Method

Our naive method builds on MimicMotion [327], a video-diffusion human animation framework that integrates pose and appearance features. The original model assumes one reference image and a driving pose sequence, which is incompatible with our setting. We thus extend its feature-integration mechanism to (1) aggregate appearance features from multiple reference images, and (2) condition generation on a target background image. We will first describe the model training before detailing feature integration.

Model Training. We adopt the EDM diffusion framework [127] for model training. We use the ground-truth sequence $V = [I'_{fr}, V'_{1:T}]_t$ as supervision, where $[\cdot,\cdot]_t$ is temporal concatenation. Here, I'_{fr} is treated as the first ground-truth frame. This follows MimicMotion's base model, an image-to-video method trained to generate the first frame identical to the input image. We use I'_{fr} instead of I'_{bk} as MimicMotion assumes a front-facing input, and we continue training from their pretrained model. During training, we update the pose encoder (introduced later) and the denoising network. Please refer to the left box "Training: Video Dataset" in Fig. 4.2 for visualization. Note that, the naive method passes only the poses of $V = [I'_{fr}, V'_{1:T}]_t$ through the solid purple arrows, and does not generate the back image (bottom, purple border) in the output.

Feature Integration. Three features are integrated into the denosing network. First, VAE feature $f_v = [E_v(I'_{fr}), E_v(I'_{bk}), E_v(I'_{bg})]_c$ is obtained by passing I'_{fr}, I'_{bk}, and I'_{bg} through the VAE encoder E_v, where $[\cdot,\cdot]_c$ is channel-wise concatenation. This feature is duplicated along the temporal axis, channel-concatenated with the noisy latent, and then fed into the denoising network. Second, the image features are extracted as $f_{im} = E_{im}(I'_{fr}) + E_{im}(I'_{bk})$, where we use addition instead of concatenation, as it improves results in practice. Here,

E_{im} is implemented as a CLIP encoder [206], and its output feature f_{im} is integrated into the denoising network via cross-attention layers. Third, pose features are extracted as $f_p = E_p([P'_{fr}, P'_{1:T}]_t)$, where E_p is implemented the same as the pose encoder in MimicMotion and P'_{fr} is the DWPose of I'_{fr}. Finally, f_p is integrated into the denoising network by adding it to the output of denoising network's first layer. All these modifications are natural extensions of design of MimicMotion. **Reference Image Augmentation**. During training, I'_{fr} and I'_{bk} are sampled from the same video as the GT sequence, resulting in similar lighting and shading. At test time, we use selfies and new backgrounds with different lighting. To bridge this gap, we augment training data by adjusting the color tone of I'_{fr} and I'_{bk} to match random backgrounds using a pretrained image harmonization network [42] (details in supplementary). This approach outperforms image relighting-based augmentation [323], which tends to overmodify shading and distort clothing patterns.

Model Inference. We replace $\{I'_{fr}, I'_{bk}, I'_{bg}, P'_{1:T}\}$ with $\{I_{fr}, I_{bk}, I_{bg}, P_{1:N}\}$ during inference. Starting from a random noise, the model progressively denoises it to obtain a clean latent, which is then decoded by the VAE decoder to generate frames. We remove the first frame because it is the same as I_{fr}. Since this process only produces $T+1$ frames, we adopt the progressive latent fusion strategy from [327] to generate N frames ($N \geq T$). This method segments the sequence into overlapping segments with $T+1$ frames, denoises each segment, and fuses them to generate a N-length video $\hat{V}^{1:N}$. To better preserve identity, we refine the face in generated frames using a pretrained face refinement method [50], with I_{fr} as the reference. Fig. 4.2 (right) visualizes the process, where the naive method also does not produce the back-view frame with the purple border.

Fig. 4.3 (b) shows a frame generated by our naive design, which struggles to reconstruct patterns on the upper clothing in the back view. This shows that simply injecting features from feature encoders into the network is insufficient for effectively utilizing the additional reference image I_{bk}.

---

[1] This example is for illustration only and is not used for training.

(a) Inputs (b) Naive (c) Naive + RefNet (d) Naive + FG (e) Naive+ FG+MRA (f) Ours: (e) + FT

Figure 4.3: **Model Ablations.** The inputs are shown in (a). The naive method (b) fails to render accurate back views. ReferenceNet (c) improves back-view generation but introduces hood artifacts and blurs text. Additionally, it requires extra parameters, reducing model efficiency. Our frame generation strategy (d) produces better back views than (b) without extra parameters, though text remains blurry. Multi-reference attention (e) enhances back view patterns, and adding the fine-tuning stage (f) delivers sharp, recognizable text.

4.2.2 Frame Generation Strategy

The goal is to improve the naive model by effectively integrating I_{bk} for accurate back-view reconstruction. One option is to add a ReferenceNet [106, 262], which extracts features from the front and back images and injects them into the denoising network via self-attention. As shown in Fig. 4.3(c), this design captures back text (though blurry) but introduces unwanted white patterns around the hood. Moreover, ReferenceNet increases model parameters and training time by about 1.5×, making it computationally inefficient.

We observe that the front-view clothing pattern is well reconstructed, likely because I_{fr} is always generated as the first frame, allowing its features to propagate through the video due to strong frame-to-frame consistency. Motivated by this, we propose a simple, parameter-free frame generation strategy (Fig. 4.2 left). Our insight is that enforcing the front and back views as the first frames leverages the video model's consistency, ensuring key visual details, such as clothing patterns, remain coherent throughout the sequence. Specifically, we treat the back-view image as the second ground-truth frame, enhancing the naive approach by setting $V = [I'_{fr}, I'_{bk}, V'_{1:T}]_t$ and $f_p = E_p([P'_{fr}, P'_{bk}, P'_{1:T}]_t)$, where P'_{bk}

represents the DWPose of I'_{bk}. During inference, the model generates $T+2$ frames at once (through multiple denoising steps), and we discard the first two frames.

As shown in Fig. 4.3 (d), this substantially improves back-view accuracy over the naive method (b) and achieves results comparable to ReferenceNet (c), but with fewer parameters, reduced training time, and no hood artifacts. However, the text in (d) remains blurry and unreadable.

4.2.3 Multi-Reference Attention

To better reproduce patterns from input images, we design a multi-reference attention mechanism, building on top of our frame generation design. Our key idea is to modify the denoising network's spatial self-attention layers to better integrate features from multiple reference images into other frames (see Fig. 4.4). Specifically, we define a feature map from a self-attention layer as $x_{1:(T+2)} \in \mathbb{R}^{(T+2)\times H\times W\times C}$, where H, W, and C represent the height, width, and channel size of the feature map, respectively. Based on our frame generation design, we extract the reference image features as $x_{fr} = x_1$ and $x_{bk} = x_2$, both having dimensions $1 \times H \times W \times C$. Then we duplicate x_{fr} and x_{bk} along temporal axis T times. Next, we augment $x_{3:(T+2)}$ (*i.e.*, feature maps of other frames, excluding x_1 and x_2) by concatenating them width-wise with x_{fr} and x_{bk}. Further, we concatenate x_1 with $[x_{fr}, x_{fr}]_w$ and x_2 with $[x_{bk}, x_{bk}]_w$, both along the width dimension, for batch processing, where $[\cdot,\cdot]_w$ is width-wise concatenation. Finally, we apply self-attention and extract the first third of the output feature map along the width dimension as the output, similar to [106]. The key difference from [106, 262] is that those methods apply self-attention using reference features from ReferenceNet, while ours directly uses features from the denoising network. Our design offers two advantages: (1) it ensures that x_{fr}, x_{bk}, and $x_{3:T+2}$ share the same feature space, enabling effective fusion, and (2) it is more efficient, avoiding extra parameters by removing the need for ReferenceNet.

As shown in Fig. 4.3 (e), adding multi-reference attention sharpens text details compared to (d), though the word "HARMONY" remains a bit hard to recognize.

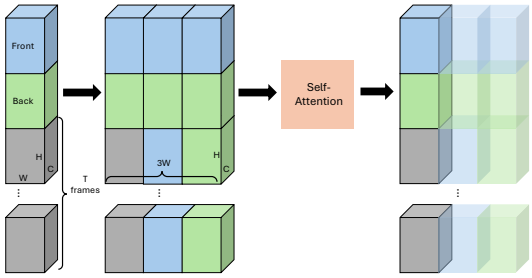


Figure 4.4: **Multi-Reference Attention**. Given a pre-attention feature map (left), we duplicate and concatenate the front (blue) and back (green) features with all frame features (gray) (middle). For batch processing, the front and back features are also concatenated with themselves. The combined features then pass through self-attention layers, and we extract the first third of the output along the width axis as the final result (right).

4.2.4 Image-Based Fine-Tuning

To improve frame quality, we introduce an image-based fine-tuning strategy consisting of two steps: (1) constructing a task-specific, high-quality image dataset, and (2) fine-tuning the model on this dataset while maintaining temporal coherence. We did not apply high-quality video fine-tuning due to two challenges: (1) it is difficult to obtain fit check videos (with both front and back views) of higher quality than our training data, and (2) motion blur in videos often degrades sharpness. Fig. 4.2 (middle) shows a visualization.

Constructing Image Dataset for Fine-Tuning. Our goal is to construct training pairs, each consisting of an input set $\{I'_{fr}, I'_{bk}, I'_{bg}, P'_{gt}\}$ and a corresponding ground-truth image I'_{gt}. Obtaining real-world images I'_{fr}, I'_{bk}, and I'_{gt} of the same person within identical backgrounds is challenging; thus, we opt for synthetic dataset creation. To construct each training pair, we first collect real front-view images (I'_{fr}) from the web, prioritizing those with visible reflections or shadows to improve the model's ability to render realistic lighting effects. Next, we employ a pretrained image reposing method (the first stage of [106]) to synthesize the corresponding back-view images (I'_{bk}). We set the ground-truth I'_{gt} as either I'_{fr} or I'_{bk}, obtaining I'_{bg} by inpainting the foreground and shadow or reflection regions of I'_{gt}. P'_{gt} is derived from the selected I'_{gt}. In addition, I'_{fr} and I'_{bk} undergo matting when used as input.

Fine-Tuning Video Models on Image Dataset. We fine-tune our trained model using constructed image pairs, where the ground-truth image I'_{gt} and its corresponding pose P'_{gt} are duplicated T times to match the required input sequence length. In practice, this duplication strategy leads to better temporal consistency compared to fine-tuning with non-duplicated sequences (*i.e.*, setting $T = 1$). Inspired by [41], we fine-tune only the spatial layers of the model to preserve its temporal coherence. During fine-tuning, we select I'_{fr} as GT 80% of the time, and I'_{bk} 20% of times. This balance is chosen because I'_{bk} is synthetically generated and may contain artifacts, while completely omitting it would lead to model collapse, causing the model to generate only front views, even for back-facing input poses. As part of data augmentation, we randomly shift and scale I'_{gt}, P'_{gt} and I'_{bg}, filling undefined regions with white pixels.

Fine-tuning on high-quality images with shadows and reflections for just 30 minutes on a single NVIDIA A100 GPU improves frame sharpness and human-scene composition (*i.e.*, more realistic shadows and reflections). As shown in Fig. 4.3 (f), it also sharpens text. Our fine-tuning strategy is architecture-agnostic and applicable to other human animation models. See supplementary for examples of shadow and reflection improvements.

4.2.5 Motion and Background Retrieval from IMU

Motion Acquisition. A motion generation model could map IMU data to motion, but existing approaches have key drawbacks: (1) they require multiple IMUs or motion capture devices [256, 308, 64, 118, 317], while we use only a single phone, or (2) they suffer from foot sliding and jittering with sparse input [183, 294, 250, 320]. Given these limitations, we adopt motion retrieval instead. However, prior retrieval methods [201, 17, 72, 230, 152] rely mainly on text queries, which do not suit our setting. Training an IMU-to-motion retrieval model is also challenging due to the lack of large-scale paired mobile IMU and motion datasets, and synthetic data [178] introduce a domain gap, as real phone IMUs are noisy and optimized for non-motion tasks.

Instead, we propose an IMU-based motion retrieval approach that matches recorded orientation and translation to a database of fit check motions. The idea is to retrieve the top-k closest matches by computing dynamic time warping (DTW) distances between the

recorded and candidate motions' orientation and translation (details in supplementary).

Background Acquisition. The retrieved motion must align naturally with the virtual background's ground plane. Given the retrieved motion, we select the top-k backgrounds whose ground plane orientation best matches that of the motion's original background (details in supplementary).

4.3 Experiments

In this section, we evaluate the human animation component. Implementation details and motion retrieval evaluation are provided in the supplementary.

Datasets. We collect 1,590 fit check videos from the web, each featuring a single person with both front and back views. The average length is 9.5 seconds, and subsampling every 4 frames yields about 72 frames per video. The dataset is split into 1,441 training and 149 test videos. Body orientation is estimated per frame using a pretrained SMPL-based detector [155], classifying frames as front (330°–30°) or back (150°–210°). For fine-tuning, we collect 122 front-view web images to form 122 fine-tuning pairs. In addition to our test set, we also evaluate on widely used datasets, TikTok [113] and UBC Fashion [318].

To evaluate our selfie task, we create a self-capture dataset with front and back mirror selfies from 8 individuals wearing various outfits, totaling 24 captures. We record a "ground truth" fit check video for each capture, maintaining the same clothing and a nearly identical background image.

Our Results. We present the results of our method in Fig. 4.5. First, our method generates realistic fit-check videos with diverse poses while consistently preserving identity, even from selfie inputs. Second, our method effectively leverages both the front and back selfies to reconstruct a plausible outfit appearance across various clothing types (*e.g.*, dresses, pants, shorts). Observe the detailed rendering of the person holding the pants in row 3, column 7. Third, our method naturally composites the person into diverse indoor and outdoor backgrounds with realistic lighting, reflections, and shadows. Please see more results in supplementary.

Fig. 4.6 shows results of the same capture composited into various virtual backgrounds. Despite the input selfie having strong left-sided lighting, our method successfully relights

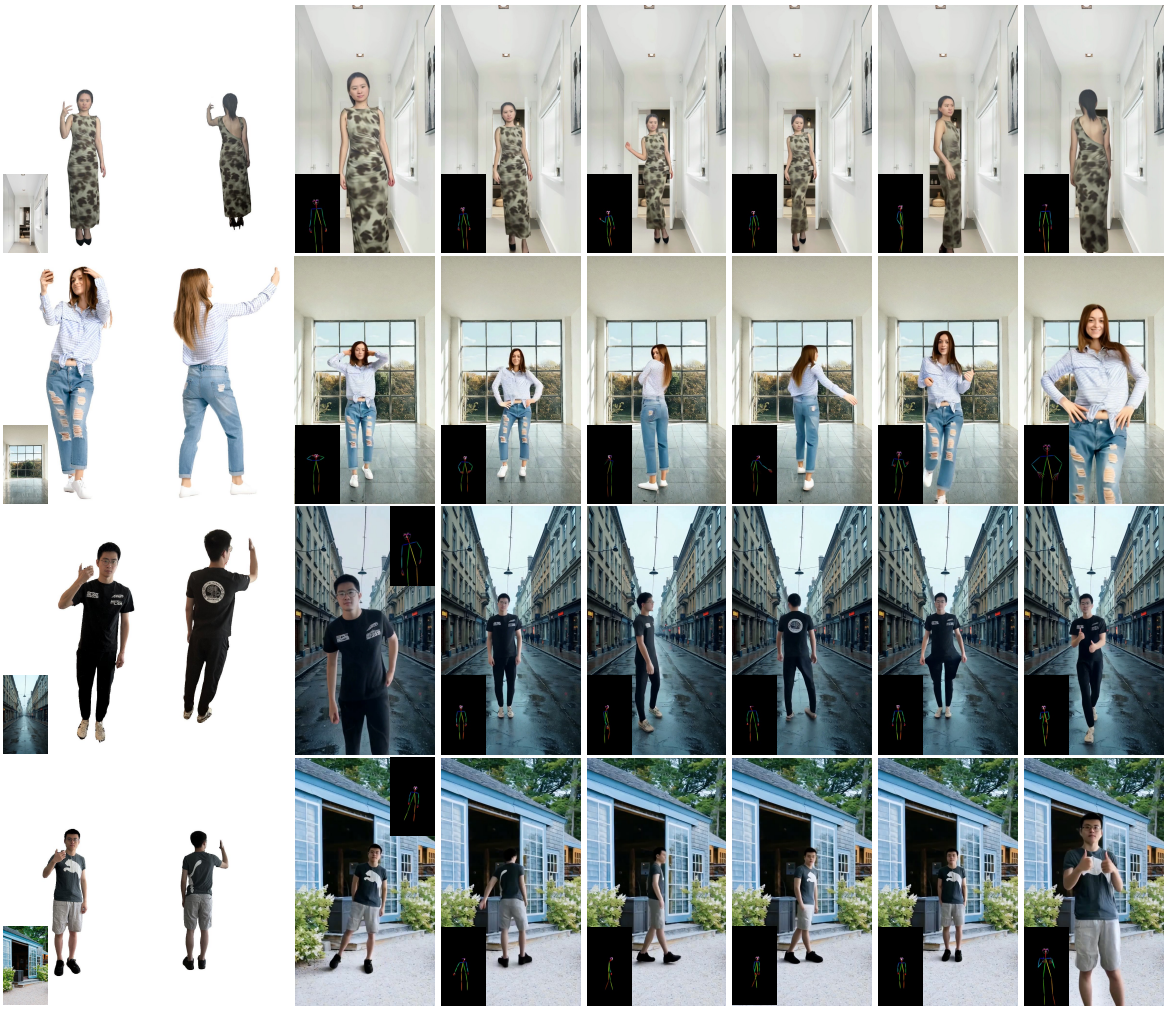


Figure 4.5: **Our Results.** The left two columns show the input selfies and background, while the right six columns display the generated results (inset: pose input, locations adjusted to avoid occlusion). Given mirror selfies with various outfits and lighting conditions, our method generates realistic fit-check videos, accurately capturing appearance across diverse poses. Additionally, it generates reflections (rows 1–3) and shadows (row 4) on the ground, ensuring natural integration with both indoor and outdoor backgrounds.

the person to match each background's illumination, enabled by our reference image augmentation strategy and the learned prior of the video diffusion model.

Baselines. To our knowledge, no prior work directly addresses our task. So we adapt four

Figure 4.6: **Results with the Same Capture under Different Virtual Backgrounds.** The first column shows the input selfies and target pose; the others show results under different virtual backgrounds (insets: input backgrounds). Despite strong left lighting in the selfies, our method adapts shading to each background.

human animation methods as baselines: (1) Animate Anyone [106] uses ReferenceNet to extract features from a single reference image. We extend it to extract features from both front and back selfies and integrate them into the denoising network. The background image is encoded by the VAE and incorporated into the denoising network in the same manner as noisy latents, aligning with our design. (2) Champ [343] builds on Animate Anyone with additional pose signals (*e.g.*, SMPL). We adapt it in the same way as (1) with additional pose inputs. (3) StableAnimator [254]. (4) MimicMotion [327]. Both (3) and (4) are video diffusion-based methods adapted in the same way as our naive method, except without reference image augmentation. All baselines share our input setup (except Champ with additional pose input). For fair comparison, we initialize each model with its pretrained checkpoint and continuing training them on our dataset. More details are in supplementary.

Comparison with Baselines. First, we present a qualitative comparison with baselines on our selfie dataset in Fig. 4.7. Image diffusion-based methods, Animate Anyone (b) and Champ (c), leverage back selfies via ReferenceNet to improve back-view generation but introduce artifacts and temporal inconsistency, causing background jitter. All baselines fail to produce realistic ground reflections, reducing realism, while our method generates reason-

(a) Inputs (b) Animate Anyone (c) Champ (d) StableAnimator (e) MimicMotion (f) Ours (g) Real Frame

Figure 4.7: **Qualitative comparison with baselines.** Inputs and real frame are shown in (a) and (g), respectively. The input pose is detected from the real frame. Image diffusion-based methods (b) and (c) fail to generate accurate back views and exhibit poor temporal consistency, causing background jitter and artifacts (see top-left blue text in the first row of (b)). Video diffusion-based baselines (d) and (e) generate completely wrong back views. Our method (f) achieves accurate outfit rendering and realistic ground reflections. Note that background images and real videos, though captured in the same session, may vary in intensity and color tone due to auto-exposure and white balance.

able reflections, demonstrating the effectiveness of the shadow and reflection enhancement in the fine-tuning stage. Overall, our method delivers accurate outfits, realistic reflections, and strong temporal consistency.

Tab. 4.1 presents the quantitative results. Champ achieves a better LPIPS score than other baselines due to its sharp image diffusion-based frame generation. However, both Champ and Animate Anyone perform poorly on video metrics (FID-VID, FVD), indicating temporal inconsistency. Our model outperforms all baselines across all metrics. Additional qualitative and quantitative comparisons on selfie captures, datasets (our test set, UBC Fashion, TikTok), and videos are in the supplementary.

Human Study. We conducted a user study with 21 participants, each rating four identical videos (on a 1–5 scale, higher is better) across three aspects: *body size accuracy*, *garment accuracy*, and *realism*. Our method achieved average scores of 3.88, 4.08, and 3.75, outper-

Table 4.1: **Quantitative Comparisons on Self-Captured Dataset.** All methods are evaluated without face refinement as post-processing and the target poses are detected from the captured real videos. Our method outperforms all baselines and variants.

Method	LPIPS ↓	SSIM ↑	PSNR ↑	FID ↓	FID-VID ↓	FVD ↓
Animate Anyone [106]	0.428	0.429	14.12	128.9	65.33	1158
Champ [343]	0.410	0.402	13.86	121.8	61.17	1108
StableAnimator [254]	0.422	0.454	14.70	131.5	60.09	982.8
MimicMotion [327]	0.413	0.458	14.69	123.6	59.63	947.8
Ours-FG-MRA-FT	0.411	0.459	14.67	123.5	60.41	925.4
Naive + RefNet	0.403	0.463	14.77	123.4	59.58	922.2
Ours-MRA-FT	0.398	0.468	14.98	123.4	57.00	921.6
Ours-FG	0.404	0.462	14.92	124.2	53.45	895.8
Ours-MRA	0.386	0.478	16.17	118.6	53.92	871.0
Ours-RIA	0.384	0.491	16.36	119.0	52.98	866.6
Ours-FT	0.395	0.471	15.29	121.9	55.87	910.8
Ours-FT + Joint Training	0.391	0.481	16.12	120.2	53.42	891.3
Ours-FT + Full FT	0.383	0.489	16.41	118.3	53.12	924.1
Ours	**0.381**	**0.497**	**16.80**	**116.6**	**51.69**	**854.9**

forming the best baselines – 3.20 (MimicMotion), 2.88 (Champ), and 2.82 (MimicMotion), respectively. See supplementary for more quantitative results for the three aspects.

Ablation Study. We test the following variants: (1) *Ours-FG-MRA-FT*: naive implementation. (2) *Naive+RefNet*: naive implementation with ReferenceNet. (3) *Ours-MRA-FT*: our model without multi-reference attention and fine-tuning, same as *Naive+FG* in Fig. 4.3. (4) *Ours-FG*: our model without frame generation strategy, where multi-reference attention is modified to use only the first frame rather than the first two. (5) *Ours-MRA*: our model without multi-reference attention. (6) *Ours-RIA*: our model without reference image augmentation. (7) *Ours-FT*: our model without fine-tuning, same as *Naive+FG+MRA* in Fig. 4.3. (8) *Ours-FT+Joint Training*: our model replacing image-only fine-tuning in Sec. 4.2.4 with joint fine-tuning on image and video data. We use the same image data as

(a) Input selfies and pose (b) Background (c) Ours

Figure 4.8: **Limitations.** (a) and (b) show the inputs; in (b), the sun direction is not from the front. Our method (c) misestimates the sun direction, causing incorrect shadow.

in Sec. 4.2.4 together with our fit check training videos. (9) *Ours-FT+Full FT*: our model replacing spatial-layer-only fine-tuning in Sec. 4.2.4 with full fine-tuning of the denoising network.

Tab. 4.1 and Fig. 4.3 show that our final model outperforms all variants. Beyond the discussion in Sec. 4.2, we observe: (1) *Ours* surpasses *Ours-RIA*, validating the benefit of reference image augmentation. (2) *Ours* outperforms *Ours-FT + Joint Training* and *Ours-FT + Full FT*, demonstrating the advanatge of fine-tuning only the spatial layers. Please see supplementary for additional results.

Limitations. Our method has several limitations. First, it fails to generate realistic shadows when compositing subjects into sunlit backgrounds (Fig. 4.8). Second, constrained by the base model of MimicMotion, it cannot faithfully render fine details such as hands and intricate textures. Third, the background in generated frames might have slight color shift compared to the input background due to the use of VAE and the color shift between frames in training videos. These issues could be mitigated by adopting a more advanced video diffusion model. Moreover, our motion capture method does not model arm movements due to the single-phone IMU setup; integrating additional IMUs (*e.g.*, from a smartwatch) could

address this. Lastly, while the model performs well with the current dataset, scaling up training data can further improve the model.

SUPPLEMENTARY MATERIAL

4.4 Motion and Background Retrieval

Motion Acquisition. For motion retrieval, the key idea is to retrieve the top-k closest matches by computing dynamic time warping (DTW) distances between the recorded and candidate motions' orientation and translation. For orientation, we focus on the horizontal direction (yaw) since a mobile phone's IMU provides a more robust estimate in this axis compared to roll and pitch. Moreover, yaw captures spinning, which, along with translation, effectively represents most of the motions. For simplicity, we define orientation as yaw throughout the rest of the paper. Below we introduce the details of motion retrieval.

First, the user performs a motion while holding a phone, recording IMU data to obtain orientation $O_{1:\tilde{N}}$ and global translation $\mathcal{T}_{1:\tilde{N}}$, where $\tilde{N}$ is the motion length. Here, the orientation is directly derived from the device's motion sensors, while the global translation is computed through a multi-step process. First, we filter the acceleration data to reduce noise. Then, we integrate the filtered acceleration to obtain velocity, followed by a second integration to derive translation over time.

For each candidate motion i in the database (stored as RGB video), we extract the SMPL [198] sequence $S^i_{1:\bar{N}}$ and global translation $\mathcal{T}^i_{1:\bar{N}}$ using a pretrained human pose estimator [155], where $\bar{N}$ is the motion length. The candidate's orientation $O^i_{1:\bar{N}}$ is obtained from the yaw component of the global rotation of the root joint in $S^i_{1:\bar{N}}$. We compute the distance between the recorded motion and each candidate as:

$$D^i = D_{dtw}(O_{1:\tilde{N}}, O^i_{1:\bar{N}}) + \alpha D_{dtw}(\mathcal{T}_{1:\tilde{N}}, \mathcal{T}^i_{1:\bar{N}}), \tag{4.1}$$

where D_{dtw} is dynamic time warping (DTW) to handle sequences of different lengths. $\alpha = 0.1$ is a constant that balances orientation and translation importance. All candidates are ranked based on D^i, and the top-k motions are retrieved for user selection. After selection, we extract the DWPose sequence $P_{1:N}$ as target pose sequence.

In practice, we set our motion database to the same as our training set, as the training videos consist of fit check videos, which provide well-suited motions.

Background Acquisition. The retrieved motion must be compatible with the new background, ensuring natural alignment with the ground plane. One approach is to rotate the SMPL sequence of the retrieved motion to align with the ground plane of any given background, and then extract DWPose from the rendered motion. However, this fails due to foot sliding in the estimated SMPL sequence. Instead, we opt for background retrieval, selecting backgrounds where the ground plane is closely aligned with the retrieved motion. Specifically, we begin by utilizing a pretrained depth estimation model [264] and a pretrained image segmentation model [114] to estimate the ground plane normal for both the original background – where the motion was captured – and all candidate backgrounds. Then, we retrieve the top-k backgrounds from our database with the closest ground plane normals. Once a background is selected, we automatically scale and translate $P_{1:N}$ to keep foot keypoints on the ground.

For our background database, we curate a diverse database of indoor and outdoor environments, consisting of 800 images, including both AI-generated and real images.

4.5 Implementation Details

We will introduce the implementation details below, and all the images and videos are operated in the resolution of height 1024 and width 576, same as the MimicMotion. *We will release the code upon acceptance.*

4.5.1 Our Method

Reference Image Augmentation. During training, we apply a pretrained image harmonization network [42] to adjust the color tone of the front and back reference images, I'_{fr} and I'_{bk}. This network takes a composite (unharmonized) image and a foreground mask as input, then produces a harmonized image where the foreground seamlessly blends with the background.

For each reference image, we first apply a pretrained image matting method [210] to obtain the foreground mask. We then randomly select a background image from our back-

ground database and composite it with the extracted human foreground to create a composite image. The composite image and its corresponding foreground mask are fed into the image harmonization network, which adjusts the foreground color to better match the background. After obtaining the harmonized output images, we remove their backgrounds and use the resulting images as training data. We apply this process independently to both I'_{fr} and I'_{bk}, using different background images for each. This is to accommodate the natural color tone variations between front and back selfies at test time, even when captured almost simultaneously.

Model Architecture and Parameters. We adopt the 3D denoising UNet, image encoder, pose encoder, VAE encoder, and VAE decoder architectures from MimicMotion [327]. Due to computational constraints, we set T to 6, which means each training batch contains 8 frames (including front and back reference image). Larger T can be used if more computational resource is available. While we train on 8 video frames per batch, we find that the model can be extended to generate 16-frame or 24-frame sequences at test time, improving efficiency without a noticeable loss in quality.

Model Training. We initialize the model using the pretrained checkpoint "MimicMotion_1.pth" from MimicMotion. We did not apply regional loss amplification in MimicMotion because we found that this does not improve the results in our experiments. During training, we randomly sample the front and back view images, I'_{fr} and I'_{bk}, from the training video $V'_{1:N}$. To sample a T-length sequence $V'_{1:T}$ from the training video, we apply the following strategy:

- Randomly select frames from the video (20% of the time)
- Select a sequence containing at least one front-facing frame (not necessarily I'_{fr}) (40% of the time)
- Select a sequence containing at least one back-facing frame (not necessarily I'_{bk}) (40% of the time)

Additionally, we apply reference image augmentation to both I'_{fr} and I'_{bk} for 50% of time.

During training, each conditioning feature – f_v, f_{im} and f_p – is randomly dropped (set to zero) 10% of the time, following the classifier-free guidance method [104]. This allows us to control the strength of each conditional signal during inference. Training runs on a single NVIDIA A100 GPU with a batch size of 1 and a learning rate of 1e-5, for 220K steps (around 112 hours).

Model Fine-Tuning. We fine-tune the trained model on a high-quality image dataset. The shadow and reflection regions are manually annotated. During fine-tuning, we omit reference image augmentation, as we observe that the model becomes confused by color tone shifts, resulting in frames with unnatural colors. We use the same strategy as the training time to drop the conditioning feature. We apply a weighted loss strategy during fine-tuning. Specifically, we assign a higher weight $\beta = 2$ to the loss computed in the shadow and reflection regions, while maintaining a weight of 1 for the rest of the region. For optimization, we use a learning rate of 1e-6 and fine-tune for 1K steps, which takes approximately 30 minutes.

Model Inference. At inference, we set the guidance scale to 2 and the number of overlapping frames to 4. The denoising time step is set to 25, and we use the Euler scheduler [127].

4.5.2 Baseline Details

Human Animation Baselines. We initialize all baselines from their pretrained checkpoints and train them on our dataset on a single NVIDIA A100 GPU for a fair comparison. Additionally, we set the frame length for all baselines to 8, aligning with our settings.

For Animate Anyone[106], since the official code is unavailable, we choose to use a widely-adopted unofficial implementation[184]. We follow the same hyperparameter settings as this codebase. The first stage of training runs for 100K steps, taking approximately 40 hours to converge. The second stage runs for 40K steps, requiring around 24 hours to complete. We observe that further training degrades performance.

For Champ [343], we use the official code. The first stage of training is conducted for 100K steps, taking approximately 35 hours to converge. The second stage runs for 40K steps, requiring about 20 hours to complete. Similar to Animate Anyone, we find that

additional training negatively impacts performance.

For StableAnimator [254], we follow the official implementation and train the model for 220K steps with a learning rate of 1e-5, taking approximately 132 hours to complete.

For MimicMotion [327], since no training code is provided, we implement our own training procedure. We train the model for 220K steps with a learning rate of 1e-5, which takes around 99 hours to complete.

Motion Retrieval Baseline. As there are no existing IMU-to-motion retrieval baselines, we adopt the text-to-motion retrieval method TMR[201] as our baseline. We use the official implementation and run the pretrained model (trained on the HumanML3D dataset[83]) on the recorded motion and our motion database to retrieve the top-k matching motions.

4.6 Experiments

4.6.1 More Results for Selfie Input

Fig. 4.10 presents additional results of our method on real selfie captures. Our approach generates high-quality fit check videos featuring diverse outfits in both indoor and outdoor settings, accurately capturing a wide range of poses with realistic shading, reflections, and shadows.

4.6.2 Comparison with Baseline

Datasets

In addition to evaluating our model on the self-captured real selfie dataset presented in the main paper, we conduct further analysis on three additional test sets: (1) the test set from our self-collected dataset, (2) the test set of UBC Fashion dataset [318], and (3) the TikTok dataset [113]. We filter out videos that do not include a back view. After filtering, our dataset test set contains 149 videos, with an average of 68 frames per video. The UBC Fashion dataset consists of 100 videos, averaging 98 frames per video, while the TikTok dataset includes 19 videos, with an average of 115 frames per video.

For evaluation, we randomly sample a front and back image as reference inputs, while the input pose sequence is extracted from the corresponding ground truth (GT) video. The

Figure 4.9: **Results with the Same Capture under Different Virtual Backgrounds.** The first column shows the input selfies and target pose; the others show results under different virtual backgrounds (insets: input backgrounds). Despite strong left lighting in the selfies, our method adapts shading to each background.

input backgrounds in our test set and the TikTok dataset are obtained using the same inpainting strategy as in our training set. However, for the UBC Fashion dataset, we use a plain white background instead, as inpainting a nearly white background with Stable Diffusion introduces artifacts. Finally, we compare the generated video with the GT video.

Notably, the input reference images in these datasets are captured from a third-person perspective rather than as selfies. This setup enables us to evaluate model performance on non-selfie inputs. Furthermore, since these images are sampled from the GT video, they share similar lighting conditions with the GT. This also differs from our selfie setup, but we still evaluate on these datasets for a more comprehensive evaluation.

Please note that all methods are trained solely on our training set, with baselines initialized from their official checkpoints.

Qualitative Comparison

Fig. 4.9 shows more results of the same captures under different virtual backgrounds.

Fig. 4.11, Fig. 4.12, and Fig. 4.13 present additional comparisons on real selfie captures.

We observe the following: (1) Animate Anyone and Champ exhibit artifacts such as inaccurate clothing patterns and artifacts around the shoes (row 3, column 3 in Fig. 4.12). (2) MimicMotion and StableAnimator fail to accurately reconstruct the appearance of the back view, demonstrating that simple modifications to these methods do not effectively utilize the additional reference image input. Additionally, they struggle to capture fine details in the front view (*e.g.* missing logo in row 5 in Fig. 4.13) and produce blurry patterns, whereas our method preserves accurate and sharp patterns due to the fine-tuning stage. (3) Our method surpasses all baselines in both appearance and pose fidelity while also generating more realistic reflections and shadows on the floor.

Fig. 4.14 presents a qualitative comparison on the UBC Fashion dataset. We observe the following: (1) Image diffusion-based methods (Animate Anyone and Champ) exhibit noticeable background color shifts, indicating their limited generalization ability to unseen backgrounds. Additionally, they introduce visible artifacts on faces and bodies and fail to accurately capture body shape. (2) Video diffusion-based baselines (MimicMotion and StableAnimator) struggle to reconstruct the appearance of the back view accurately, highlighting that simple modifications to these methods do not effectively utilize the additional reference image input. (3) Our method outperforms all baselines in both appearance and pose fidelity, producing more realistic and coherent results.

Fig. 4.15 presents a qualitative comparison on the TikTok dataset. We observe the following: (1) Animate Anyone and Champ exhibit noticeable artifacts, such as missing body parts (e.g., row 1, column 4, and row 3, column 4) and inaccurate clothing patterns (rows 2 to 5). (2) MimicMotion and StableAnimator struggle to accurately reconstruct the appearance of the back view, demonstrating that simple modifications to these methods do not effectively utilize the additional reference image input. Additionally, they produce blurry patterns (e.g., shorts in row 1), whereas our method generates sharper details due to the design of the fine-tuning stage. (3) Our method surpasses all baselines in both appearance and pose fidelity, delivering more realistic and coherent results.

Table 4.2: **Quantitative Comparisons on Our Test Set.** All methods are evaluated without face refinement as post-processing. For each metric, the best and second-best methods are highlighted in bold and underline, respectively. Our method outperforms all tested baselines across all metrics. The ablation variant, *Ours-RIA*, achieves results comparable to *Ours* on this dataset because the input front and back images share the same background as the ground truth (GT) frames, making reference image augmentation less necessary in this case.

Method	SSIM ↑	LPIPS ↓	PSNR ↑	FID ↓	FVD-VID ↓	FVD ↓
Animate Anyone [106]	0.733	0.242	19.60	128.8	57.43	421.7
Champ [343]	0.763	0.231	20.60	91.72	32.40	372.9
StableAnimator [254]	0.771	0.219	21.03	97.26	33.33	394.7
MimicMotion [327]	0.779	0.211	21.34	88.02	30.51	366.1
Ours-FG-MRA-FT	0.776	0.205	21.20	89.19	29.30	356.6
Naive+RefNet	0.781	0.202	22.38	87.52	29.40	342.4
Ours-MRA-FT	0.782	0.198	22.64	86.74	28.16	308.7
Ours-FG	0.781	0.203	22.40	87.64	25.76	311.0
Ours-MRA	<u>0.796</u>	0.186	23.08	82.82	23.71	292.2
Ours-RIA	0.795	<u>0.184</u>	<u>23.15</u>	<u>79.07</u>	**22.44**	<u>281.5</u>
Ours-FT	0.786	0.196	22.81	86.68	26.00	298.0
Ours-FT + Joint Training	0.789	0.188	23.01	83.22	26.42	289.3
Ours-FT + Full FT	0.792	0.186	23.07	81.52	25.42	324.1
Ours	**0.799**	**0.183**	**23.61**	**79.02**	<u>23.11</u>	**279.3**

Quantitative Comparison

Tab. 4.2, 4.3, and 4.4 present the quantitative results of our model compared to the baselines across the three datasets. We observe that Champ performs competitively among the baselines on our test set in terms of video-related metrics, FVD-VID and FVD, but performs worse on the other two datasets. This indicates that this image diffusion-based method achieves better temporal consistency when the input reference images and background are in-distribution. However, its performance degrades significantly for out-of-distribution inputs, demonstrating poor generalization ability.

As discussed in the main paper, our method outperforms all baselines on all metrics by

Table 4.3: **Quantitative Comparisons on the UBC Fashion Dataset.** All methods are evaluated without face refinement as post-processing. For each metric, the best and second-best methods are highlighted in bold and underline, respectively. Our method outperforms all tested baselines across all metrics. The ablation variant, *Ours-RIA*, achieves results comparable to *Ours* on this dataset because the input front and back images share the same background as the ground truth (GT) frames, making reference image augmentation less necessary in this case.

Method	SSIM ↑	LPIPS ↓	PSNR ↑	FID ↓	FVD-VID ↓	FVD ↓
Animate Anyone [106]	0.902	0.130	17.08	60.27	51.94	480.0
Champ [343]	0.888	0.130	16.43	59.34	56.06	363.7
StableAnimator [254]	0.914	0.071	21.61	58.47	31.67	191.5
MimicMotion [327]	0.918	0.069	22.05	56.27	28.04	185.7
Ours-FG-MRA-FT	0.922	0.065	21.76	56.55	27.88	181.7
Naive+RefNet	0.922	0.064	22.47	49.18	24.64	183.1
Ours-MRA-FT	0.924	0.061	22.58	48.99	21.02	170.7
Ours-FG	0.921	0.068	21.94	53.17	28.13	169.4
Ours-MRA	0.928	<u>0.055</u>	<u>23.68</u>	47.41	13.64	149.8
Ours-RIA	<u>0.932</u>	<u>0.055</u>	23.59	<u>46.42</u>	<u>12.70</u>	<u>144.2</u>
Ours-FT	0.925	0.057	23.39	48.68	19.31	166.0
Ours-FT + Joint Training	0.923	0.059	23.54	48.32	20.14	148.7
Ours-FT + Full FT	0.928	0.057	23.66	47.39	18.95	174.2
Ours	**0.937**	**0.052**	**23.73**	**45.33**	**12.36**	**138.7**

employing a novel frame generation strategy with multi-reference attention and a fine-tuning approach, leading to enhanced appearance fidelity and frame quality.

4.6.3 Body Size, Garment Accuracy, Realism

Tab. 4.5 reports the full results of our human study, which evaluates body size, garment accuracy, and overall realism. Our method consistently outperforms all baselines across all criteria.

Tab. 4.6 presents the corresponding automatic evaluation results for body size, garment accuracy, and realism. Detailed descriptions and discussions are provided below.

Table 4.4: **Quantitative Comparisons on the TikTok Dataset.** All methods are evaluated without face refinement as post-processing. For each metric, the best and second-best methods are highlighted in bold and underline, respectively. Our method outperforms all tested baselines across all metrics. The ablation variant, *Ours-RIA*, achieves results comparable to *Ours* on this dataset because the input front and back images share the same background as the ground truth (GT) frames, making reference image augmentation less necessary in this case.

Method	SSIM ↑	LPIPS ↓	PSNR ↑	FID ↓	FVD-VID ↓	FVD ↓
Animate Anyone [106]	0.779	0.236	18.78	81.02	44.55	551.9
Champ [343]	0.774	0.243	18.24	90.89	53.81	669.2
StableAnimator [254]	0.784	0.242	18.48	90.43	46.20	473.4
MimicMotion [327]	0.787	0.235	18.67	87.22	38.14	433.5
Ours-FG-MRA-FT	0.790	0.234	18.64	87.36	37.54	435.6
Naive+RefNet	0.795	0.226	18.72	87.11	37.15	433.9
Ours-MRA-FT	0.802	0.224	18.98	85.19	37.22	436.3
Ours-FG	0.799	0.229	18.39	81.96	36.11	399.8
Ours-MRA	0.805	0.217	19.42	78.60	32.35	387.5
Ours-RIA	**0.808**	<u>0.216</u>	**19.71**	<u>76.70</u>	**30.85**	<u>384.5</u>
Ours-FT	0.803	0.221	19.33	83.58	35.77	390.8
Ours-FT + Joint Training	0.794	0.221	19.13	79.48	34.21	390.2
Ours-FT + Full FT	0.801	0.219	19.51	78.79	33.85	428.5
Ours	<u>0.807</u>	**0.215**	<u>19.65</u>	**76.65**	<u>31.21</u>	**382.1**

For quantitative body size evaluation, we used the SMPL-based estimator CLIFF [155] to estimate shape parameters on predicted and real frames from our self-captured dataset. We calculated shape difference by averaging the absolute differences between shape parameters of predicted and real frames. Our method achieved a shape difference of 0.053, outperforming the best baseline, MimicMotion (0.062), by 17%.

We evaluated garment accuracy via region alignment and appearance similarity. For region alignment, we used SAM2 [210] to segment garment regions in predicted and ground-truth frames from self-captured dataset and computed the average IoU. Our method achieved 0.923, better than the best baseline, MimicMotion (0.895). For appearance similarity, we

Table 4.5: **Results of Human Study.** Our method outperforms all baselines.

Method	Body Shape Accuracy ↑	Garment Accuracy ↑	Realism ↑
Animate Anyone [106]	2.55	2.45	2.03
Champ [343]	3.02	2.88	2.32
StableAnimator [254]	3.10	2.18	2.76
MimicMotion [327]	3.20	2.05	2.82
Ours	**3.88**	**4.08**	**3.75**

Table 4.6: **Results of Quantitative Evaluation of Body Size, Garment Accuracy, and Realism.** Our method outperforms all baselines.

Method	Shape Parameter Difference ↓	Garment Region IoU ↑	Garment Appearance LPIPS ↓	VLM Realism Score ↑
Animate Anyone [106]	0.073	0.853	0.539	5.8
Champ [343]	0.075	0.861	0.512	5.6
StableAnimator [254]	0.071	0.884	0.616	6.2
MimicMotion [327]	0.062	0.895	0.601	6.7
Ours	**0.053**	**0.923**	**0.485**	**7.5**

calculated LPIPS within the segmented garment regions, with our method scoring 0.485, outperforming the best baseline, Champ (0.512).

To evaluate realism, we used a VLM [11] to rate generated videos on a 1–10 scale (higher is better). Our method scored 7.5, outperforming the best baseline, MimicMotion (6.7).

4.6.4 Comparison to Models Trained on Other Datasets

We further compare our method with models trained on different datasets: Veo 3.1 (References to Video) [57] and Phantom (built on Wan2.1 [259]) [164]. These models take one or multiple reference images and a text prompt as input to generate a video. In our experiments, we provide three reference images – the front selfie, back selfie, and background image – along with a text description of motion to generate fit-check videos.

We use the following text prompt for Veo 3.1 (References to Video): *"Generate a fit-check video. The person [specify the motion]. The person's appearance must remain consistent*

with the first and second images (front and back selfies), and the background should match the third image. The camera should remain static and the viewpoint must not change. "

We additionally include a background description in the text input for Phantom, as it tends to ignore the provided background image. The text prompt is: *"Generate a fit-check video. The person [specify the motion]. The person's appearance must remain consistent with the first and second images (front and back selfies), and the background should match the third image. The background is [description of background]. The camera should remain static and the viewpoint must not change. "*

Note that, unlike our method, which takes a motion sequence as an explicit input, these models rely solely on text to determine motion. Therefore, their generated motions may differ from ours.

Fig. 4.16, Fig. 4.17, and Fig. 4.18 show qualitative comparisons. Both Veo 3.1 (References to Video) and Phantom only support landscape-mode video generation. Phantom generates inaccurate outfit (see Fig. 4.16). It also struggles to maintain consistency with the provided background image, often leading to incorrect scene composition. Veo 3.1 may produce inaccurate visual details (see the red arrow in Fig. 4.16 and Fig. 4.17).

4.6.5 Ablation Study

Qualitative Comparison

Fig. 4.19 shows additional comparisons of our variants with the full method. We observe the following: (1) *Ours-FG-MRA-FT (naïve)* fails to render accurate back views, indicating that simple modifications do not effectively encode features from additional reference images. (2) *Naïve+RefNet* incorporates ReferenceNet, improving back views but introducing artifacts (rows 1 to 2) and failing in the case of a white background (rows 3 to 4). This demonstrates that using ReferenceNet reduces the model's generalization ability to unseen backgrounds. Additionally, it introduces extra parameters for training, which negatively impacts training efficiency. (3) *Ours-MRA-FT (Naïve + FG)* applies our frame generation strategy, producing better back views than naive methods, but still suffers from blurry patterns. (4) *Ours-FT (Naïve + FG + MRA)* further integrates multi-reference attention,

enhancing back view patterns.

Importantly, all the variants fail to produce realistic reflections or generate weaker, blurry reflections on the ground (rows 1 and 2). In contrast, our method effectively achieves this by leveraging a fine-tuning strategy specifically designed to enhance shadows and reflections. In summary, our full model produces sharper results with more accurate patterns while effectively generating realistic shadows and reflections.

Quantitative Comparison

Tab. 4.2, 4.3, and 4.4 present the quantitative results of our model compared to its variants across the three datasets. As discussed in the main paper, our method outperforms most variants across all metrics by employing a novel frame generation strategy, a multi-reference attention, and a fine-tuning approach, resulting in improved appearance fidelity and temporal consistency.

The ablation variant, *Ours-RIA*, achieves results comparable to *Ours* on this dataset because the input front and back images share the same background as the ground truth (GT) frames, making data augmentation less essential in this scenario.

Inaccurate Segmentation

Our method relies on pre-segmentation of the input selfies. Therefore, we evaluate its robustness to segmentation inaccuracies. In practice, we found the adopted SAM2 [210] robust for segmenting mirror selfies. For study purposes, we dilated and eroded segmentation masks in self-captured dataset by 5% and 10%, feeding these inaccurately segmented selfies to our model. The resulting LPIPS scores – 0.383 (5% dilation), 0.386 (10% dilation), 0.384 (5% erosion), and 0.388 (10% erosion) – were only slightly worse than with accurate masks (0.381).

4.6.6 Evaluation on Motion Retrieval

We collect a test set by asking five participants to perform fit-check motions (*e.g.*, walks, twirls) while using a phone to record IMU data. Simultaneously, we capture video recordings

of these motions with an external camera. From each video, we extract the corresponding SMPL sequence as ground truth (GT), which is later used for evaluation, constructing a dataset of 20 IMU inputs and GT SMPL sequences.

Since there are no existing IMU-to-motion retrieval methods, we compare our approach against TMR [201], a text-to-motion retrieval baseline. To provide input for this baseline, we manually annotate textual descriptions for the motions in our test set. An example text annotation for a motion is: "A person walks away from the camera with their back facing it, then turns left by half a circle."

We compare the top-k retrieved motions with the GT motion using a pretrained motion encoder [249] to evaluate their similarity. This motion encoder takes as input a sequence of motion parameters in the HumanML3D format [83], which can be obtained from an SMPL sequence (details in [249]), and outputs a motion embedding. For evaluation, we input both the retrieved and GT motions into the encoder and compute their similarity based on the dot product of their motion embeddings. Our method achieves a similarity score of 0.848 for $k = 1$ and 0.716 for $k = 5$, outperforming TMR's 0.641 ($k = 1$) and 0.522 ($k = 5$). These results demonstrate the effectiveness of our approach, highlighting that IMU-based retrieval provides more fine-grained motion guidance than text-based retrieval.

Figure 4.10: **More Results of Our Method.** The left two columns display the input selfies and background, while the right six columns show the generated results (inset: pose input, adjusted to prevent occlusion). Given mirror selfies with various outfits and lighting conditions, our method produces realistic fit-check videos, accurately capturing appearance from diverse poses. It also generates reflections and shadows on the ground, ensuring seamless integration between the subject and both indoor and outdoor backgrounds.

Figure 4.11: **Qualitative Comparison with Baselines on Selfie Inputs.** The left two columns display the input selfies, pose, and background, while the right five columns showcase the generated results from various methods. The inset of Champ illustrates its corresponding SMPL pose input. All results are post-processed using face refinement. Our method surpasses all tested baselines by more accurately reconstructing appearance across diverse poses while also producing more realistic shadows and reflections on the floor.

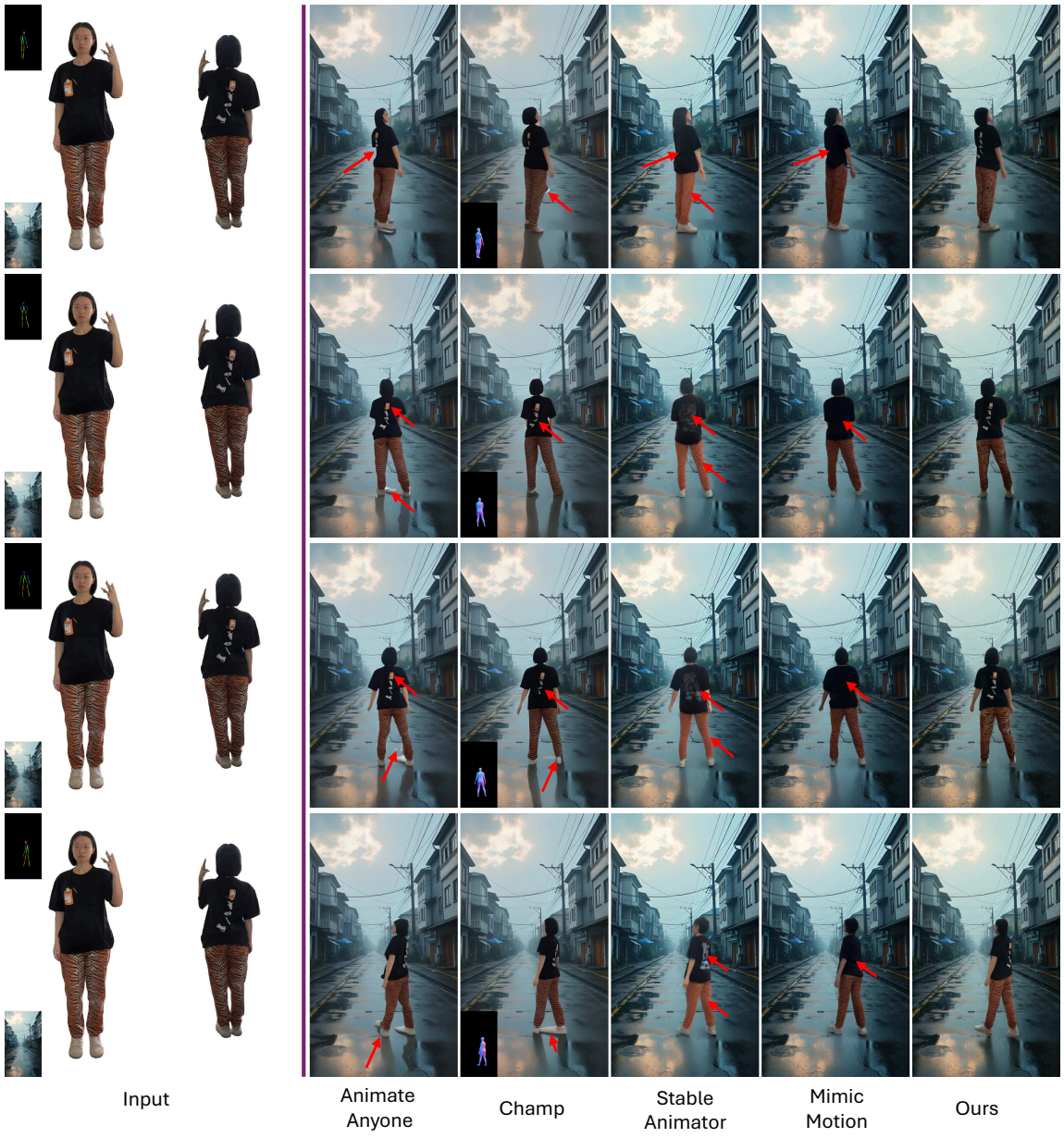


Figure 4.12: **Qualitative Comparison with Baselines on Selfie Inputs.** The left two columns display the input selfies, pose, and background, while the right five columns showcase the generated results from various methods. The inset of Champ illustrates its corresponding SMPL pose input. All results are post-processed using face refinement. Our method surpasses all tested baselines by more accurately reconstructing appearance across diverse poses while also producing more realistic shadows and reflections on the floor.

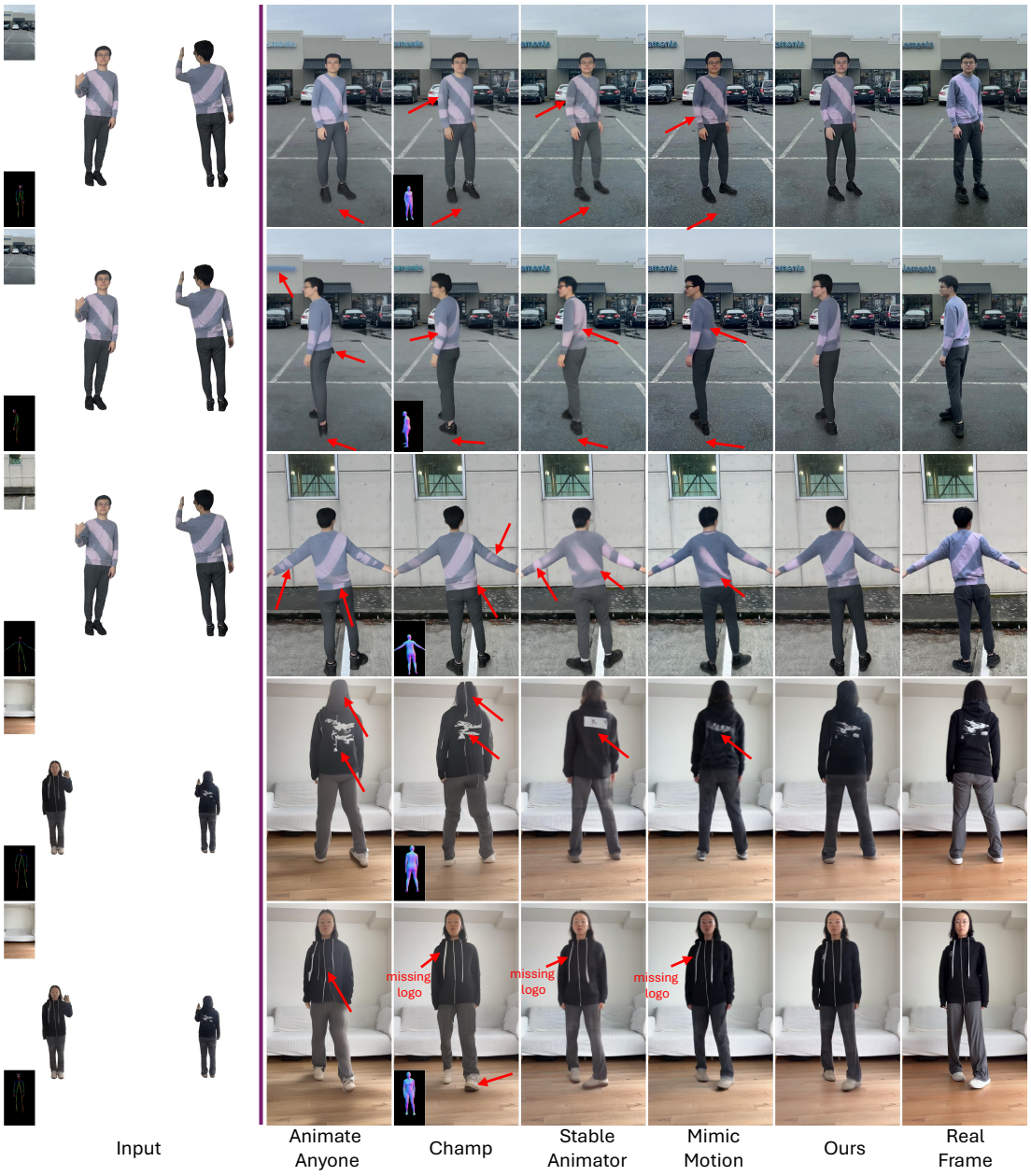


Figure 4.13: **Qualitative Comparison with Baselines on Self-Captured Dataset.** The left two columns display the input selfies, pose, and background, while the right six columns showcase the generated results from various methods alongside the real frame. The target poses are detected from the real frames. The inset of Champ illustrates its corresponding SMPL pose input. All results are post-processed using face refinement.

Figure 4.14: **Qualitative Comparison with Baselines on UBC Fashion Dataset.** The left two columns present the input reference images, pose, and background, while the right six columns showcase the generated results from various methods alongside the ground truth (GT). The target poses are detected from the GT. The inset of Champ displays its corresponding SMPL pose input. All results are post-processed using face refinement.

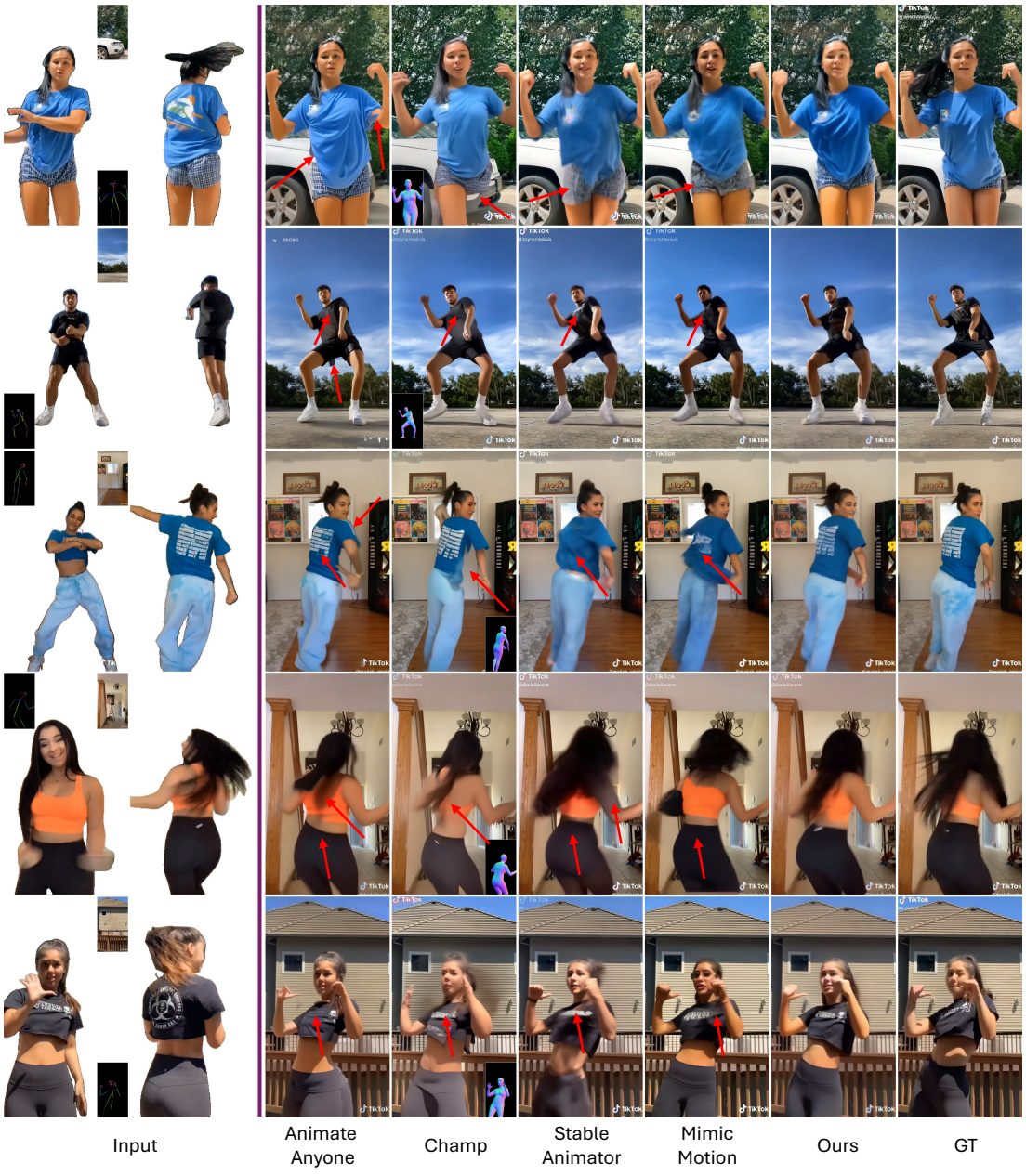


Figure 4.15: **Qualitative Comparison with Baselines on TikTok Dataset.** The left two columns present the input reference images, pose, and background, while the right six columns showcase the generated results from various methods alongside the ground truth (GT). The target poses are detected from the GT. The inset of Champ displays its corresponding SMPL pose input. Note that none of the methods were trained on the TikTok Dataset. All results are post-processed using face refinement.

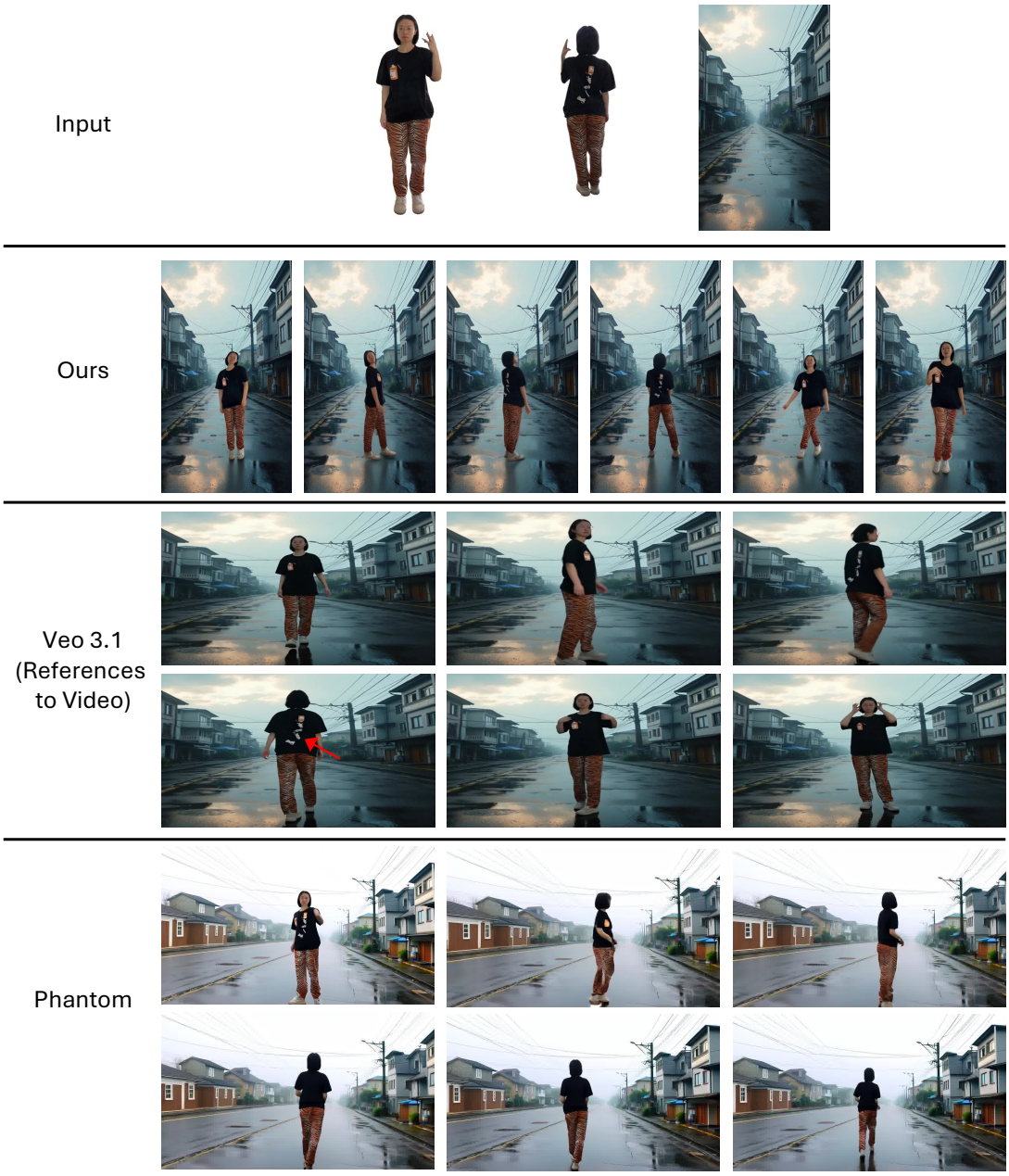


Figure 4.16: **Qualitative Comparison with Methods Trained on Other Datasets.** The first row shows the input: front selfie, back selfie, and background image. The remaining rows present the outputs from different methods. Both Veo 3.1 (References to Video) and Phantom take these three images as input along with a text description of motion. Also, they only support landscape-mode video generation. In addition, Phantom fails to effectively utilize the input background image, leading to inconsistent scene composition.

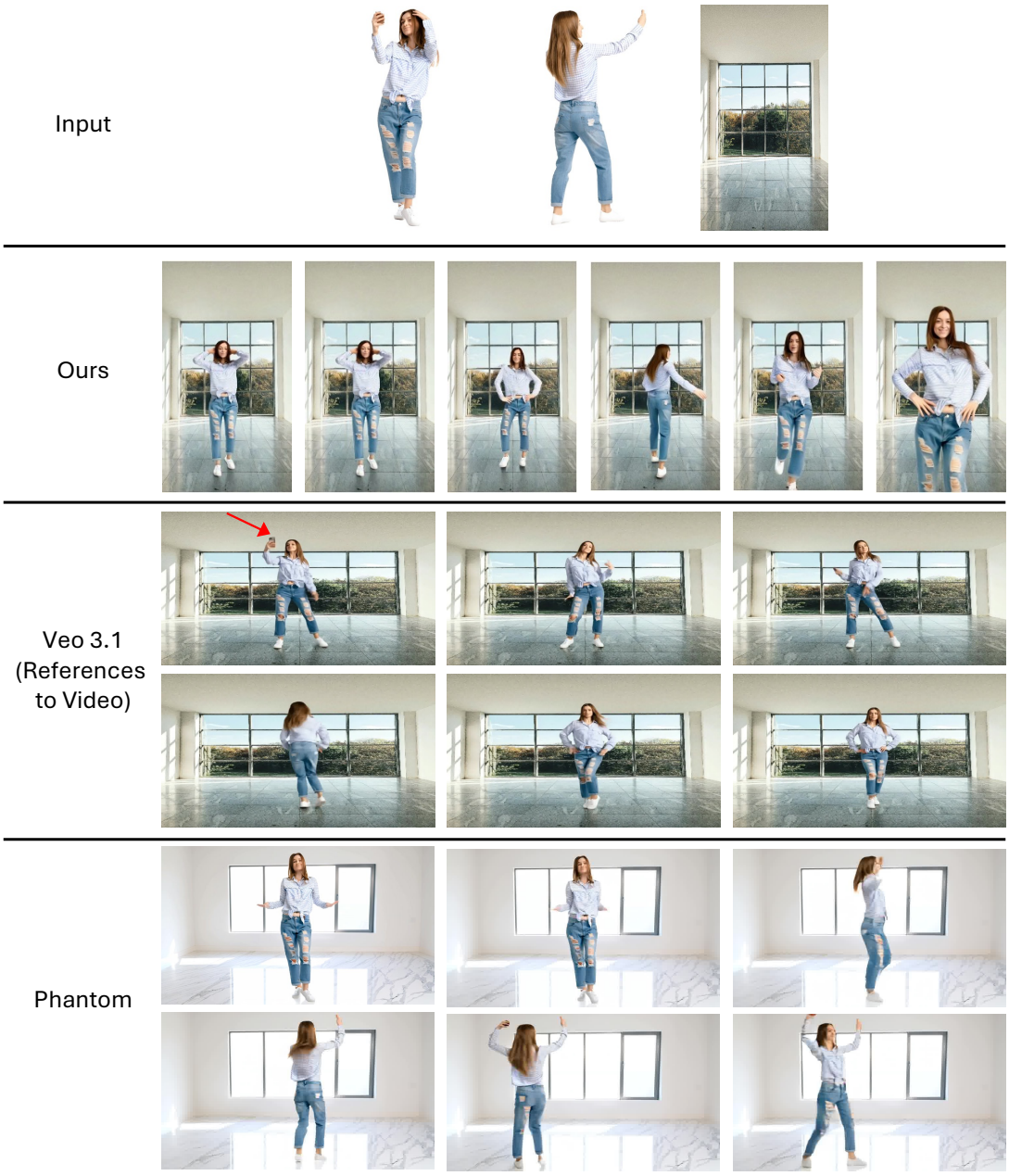


Figure 4.17: **Qualitative Comparison with Methods Trained on Other Datasets.** The first row shows the input: front selfie, back selfie, and background image. The remaining rows present the outputs from different methods. Both Veo 3.1 (References to Video) and Phantom take these three images as input along with a text description of motion. Also, they only support landscape-mode video generation. In addition, Phantom fails to effectively utilize the input background image, leading to inconsistent scene composition.

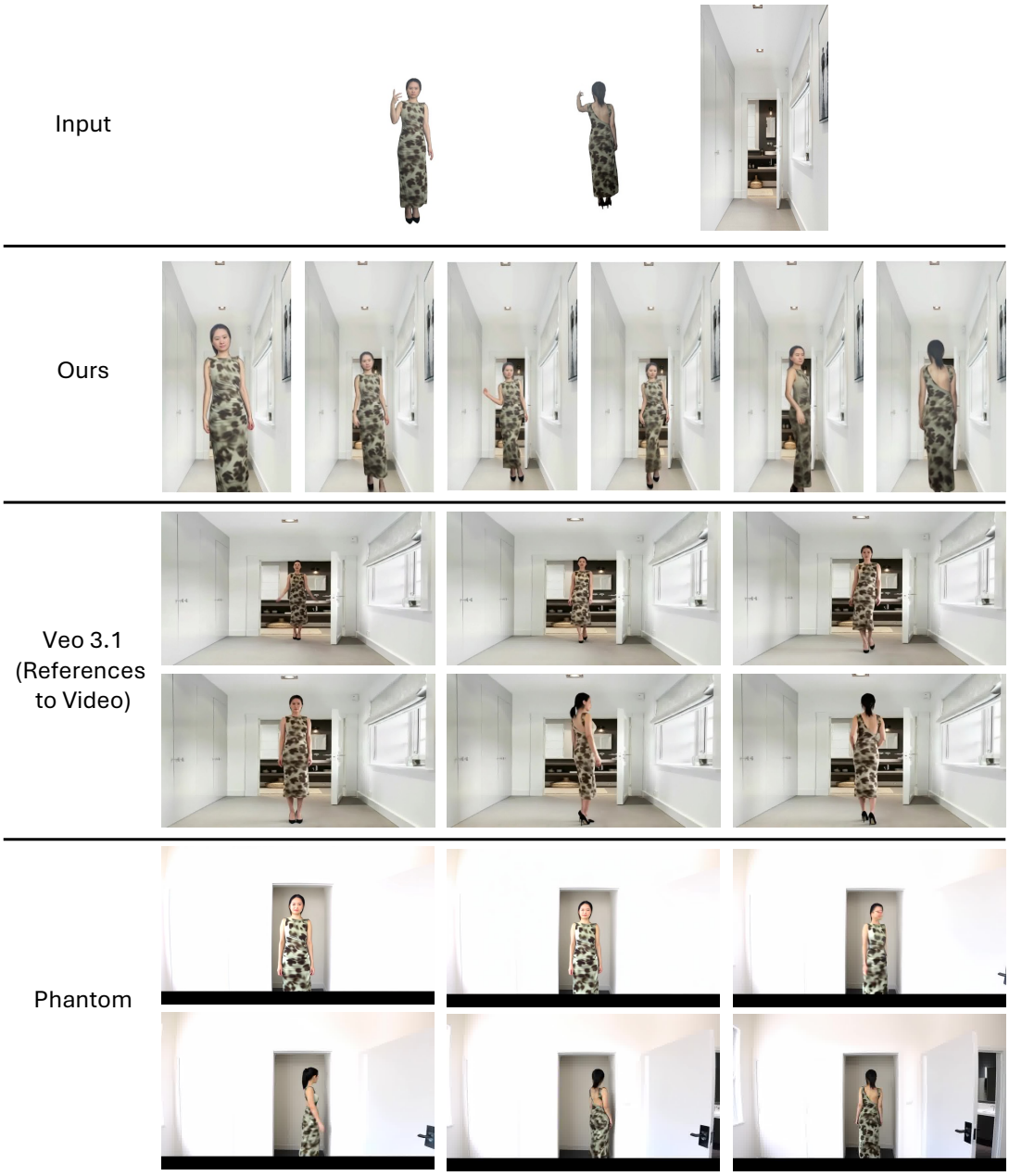


Figure 4.18: **Qualitative Comparison with Methods Trained on Other Datasets.** The first row shows the input: front selfie, back selfie, and background image. The remaining rows present the outputs from different methods. Both Veo 3.1 (References to Video) and Phantom take these three images as input along with a text description of motion. Also, they only support landscape-mode video generation. In addition, Phantom fails to effectively utilize the input background image, leading to inconsistent scene composition.

Figure 4.19: **Model Ablations.** The inputs are shown in the left two columns. The right five columns showcase the generated results from various variants and our full model. All results are post-processed using face refinement. *Ours-FG-MRA-FT (naive)* fails to render accurate back views. *Naive+RefNet* incorporates ReferenceNet, improving back views but creates artifacts (row 1 to 2) and fails in the case of a white background (row 3 to 4). *Ours-MRA-FT (Naïve + FG)* uses our frame generation strategy and produces better back views than naive methods. *Ours-FT (Naïve + FG + MRA)* additionally uses multi-reference attention and enhances back view patterns.

Chapter 5

LEARNING FEATURE-PRESERVING PORTRAIT EDITING FROM GENERATED PAIRS

This chapter presents the research project *Learning Feature-Preserving Portrait Editing from Generated Pairs* [37], conducted with Tiancheng Zhi, Peihao Zhu, Shen Sang, Jing Liu, and Linjie Luo. The subsequent analysis and comparisons to related studies in this chapter are based on the prevailing state of the art at the time of this work.

Portrait editing is increasingly favored in photo and social applications. In many of these applications, users can select from a set of pre-defined editing options and then apply their chosen edits to their own photos. In practice, the key requirement of portrait editing is to deliver outcomes that achieve selected editing while *strictly* preserving the features of subjects intended to remain unaltered (*e.g.*, identity and clothing for expression editing). Nevertheless, meeting this requirement poses a considerable challenge, as even slight deviations in these features can markedly affect the perceived quality of the outcome. Therefore, the goal of this paper is to design a portrait editing pipeline that can achieve superior editing outcomes for a specific editing task favored by users.

Existing image editing approaches fail to satisfy the requirements of portrait editing tasks. They can be categorized into two types. The first one is training-free methods, which mainly rely on a pretrained diffusion model [218] to perform editing guided by a text prompt. However, they suffer from two limitations. (1) They struggle to achieve desired editing as they depend on inversion techniques to reverse the input image into a denoising process, which may hurt editability. (2) They fail to preserve detailed subject features as little prior knowledge for invariance is enforced. Figure 5.1 (left) shows outputs of a training-free method Prompt2Prompt [98]. Another stream of work is training-based methods, aiming to learn the editing direction for desired changes, and also preserve untargeted subject features, with a training set. However, these methods require extremely high-quality training dataset,

Figure 5.1: Our method takes a portrait image as input, and applies advanced editing effects with our proposed framework. We can handle both real human portraits (1st row) as well as cartoon characters (2nd row). Our approach obtains superior aesthetic quality while at the same time preserving key features from the input subject. Compared with baseline approaches (left), we achieve better subject feature preservation (*e.g.*, identity), structural alignment, and fewer artifacts.

which is usually hard to collect. Figure 5.1 (left) shows outputs of a recent training-based method BBDM [148].

In this paper, we opt for training using a synthetic dataset generated automatically at low cost, thereby eliminating the necessity of manually collecting datasets. Our framework generates a synthetic dataset for any user-defined editings and uses this dataset to effectively learn the editing directions, fulfilling the aforementioned requirements, and upholding high image quality. Specifically, we first design a conditional dataset generation strategy to produce diverse paired data given text prompts, which has better identity and layout alignment than existing data generation strategy. Given these pairs, we design a Multi-Conditioned Diffusion Model (MCDM) to effectively learn editing direction and preserve the subject features. This is achieved by injecting the conditional signals from input image and text prompt into the diffusion model through different ways. Finally, we demonstrate that the trained MCDM can *explicitly* identify regions expected to change (*e.g.*, face regions for expression editing), producing an editing mask. This provides guidance for our inference

process to further keep subject features untouched.

As shown in Figure 5.1, our editing results achieve expressive styles while preserving subject features, in both real person costume and cartoon expression editing cases. The effectiveness is further substantiated by quantitative analysis and user study (Sec. 5.3), which collectively attest to its superiority over existing baseline methods.

Contributions: (1) A data generation technique providing paired data with better identity and layout alignment; (2) A Multi-Conditioned Diffusion Model producing feature-preserving results and accurate editing masks for inference guidance; (3) State-of-the-art portrait editing results. *The code will be released.*

5.1 Related Work

Image generation and editing have seen significant advancements with generative models like GANs [81], VAEs [135], and normalizing flows [214], leading to highly realistic outputs [130, 131]. Recent breakthroughs in diffusion models [103, 241, 244, 242], such as Imagen [221], GLIDE [190], DALL-E2 [207], and Stable Diffusion [218], have further revolutionized this field. They can generate a wide variety of images from mere textual descriptions and has spurred research into their applications in image editing.

Training-Free Approaches: Prevalent editing methods rely on inverting images into a model's latent space [1, 2, 286, 342, 340] and editing by manipulating latent codes [2, 231, 93] or model weights [75, 215, 13, 5], without new model training. They are known as training-free methods. Text-to-image diffusion models, akin to GANs, use Gaussian noise as latent input, combined with textual guidance, to generate images. Methods like SDEdit [179] add noise to the input image for a fixed number of steps, and then initiate a text-guided denoising process for repainting. However, these methods apply global editing, failing to preserve details in areas not targeted for modification. To overcome this issue, some studies [189, 10, 9] use user-provided masks to define editing regions, thus allowing for partial edits. Yet, obtaining precise masks for editing is non-trivial, and mask-based inpainting methods often result in the loss of image information within the masked area, disrupting the consistency between the pre- and post-edit images.

For controlled, local editing, Prompt2Prompt [98] and DiffEdit [51] have been devel-

oped. The former preserves layout and subject geometry through cross-attention maps, while the latter generates an editing mask through contrasting predictions from different text conditions. Both methods employ DDIM inversion [242, 60] to encode input images. However, DDIM inversion, especially with classifier-free guidance, often leads to unsatisfactory reconstruction and editing outcomes. Null-text Inversion [182] improves inversion reconstruction while retaining the editing capabilities. Pix2pix-zero [197] improves DDIM inversion through noise regularization [131] and introduces cross-attention loss during the denoising process. However, this method may pose difficulties in terms of control and could lead to unexpected outcomes, especially for portrait editing.

Training-Based Approaches: Training-based methods learn editing direction from a large dataset. Li et al. [148] and Sheynin et al. [232] train diffusion models for image-to-image translation and local semantic editing without inversion, but their expressiveness and quality lag behind current large-scale diffusion models. InstructPix2Pix [24] uses GPT-3 [26] and Prompt2Prompt [98] to create text edited pairs and distills a diffusion model, generally producing more controlled edits and showing robustness with real image inputs. The effectiveness of training-based methods depends on the quality of the constructed pairs. Our method, which falls into this category, achieves greater consistency and superior editing results by using Composable Diffusion [165] to generate better pairs. Relying on our condition injection mechanism and network design, we are capable of producing edits which are less affected by data imperfection, and thus better preserving input features.

Diffusion-based editing also relates to concept embedding [74], model fine-tuning [219], and controlled generation [322, 187], but they are outside our discussion scope.

5.2 Our Pipeline

Given an input portrait image x_A in the source domain A, our goal is to synthesize a high-quality portrait image $\hat{x}_B$ in domain B. A well-edited image $\hat{x}_B$ should: (1) retain the untargeted subject features (*e.g.*, identity) and rough layout from x_A, (2) ensure editing fidelity (*i.e.*, $\hat{x}_B \in B$) and maintain high image quality.

To this end, we design a diffusion-based image editing pipeline with three stages. (1) We first introduce an automated data generation strategy to create reasonably good but

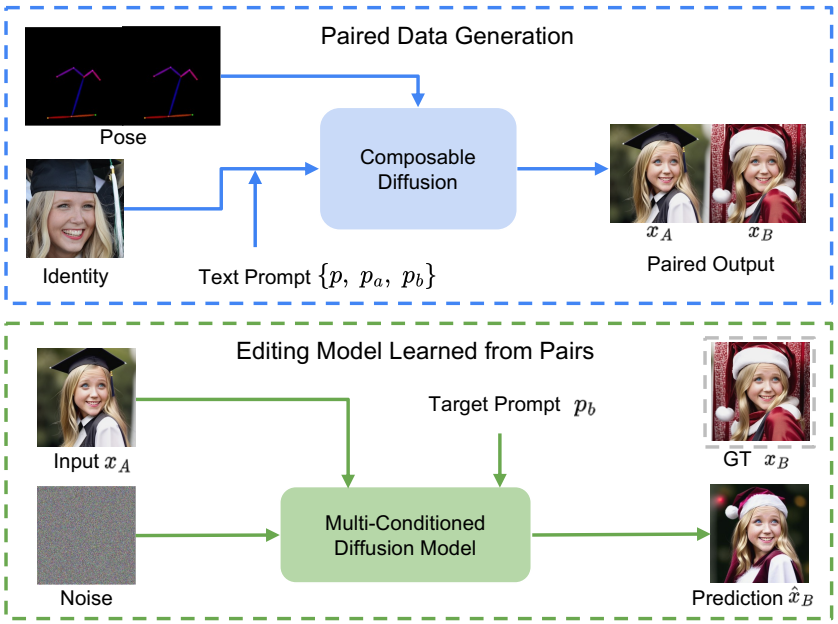


Figure 5.2: Overview of our pipeline. *Paired Data Generation* (blue dashed box) first constructs training pairs using Composable Diffusion [165] conditioning on pose and identity information. *Multi-Conditioned Diffusion Model* (green dashed box) encodes multiple condition signals to learn the editing direction and preserve subject features based on the generated pairs. The multi-condition design enhances the robustness in handling imperfections within training pairs.

not perfect pairs of input x_A and ground truth x_B (Figure 5.2 left). (2) Then we design and train a Multi-Conditioned Diffusion Model (MCDM) (Figure 5.2 right) on this generated dataset. By leveraging multiple conditions in different ways, MCDM can effectively learn the editing direction from the training pairs, while preserving detailed subject features that are not supposed to be changed. (3) During inference, we generate edited results using the trained MCDM with an automatically generated editing mask to further preserve subject details in x_A.

5.2.1 Preliminary

We start with a quick overview of Latent Diffusion [218] and establish notations that we use throughout. Latent Diffusion has two components: (1) a Variational Autoencoder, including an encoder E to transform an image x into a latent code $z = E(x)$, and a decoder D to map z back to an image $x' = D(z)$, (2) a U-Net $\epsilon_\theta(z_t, t, C)$ which predicts added noise given a noisy latent. z_t is the noisy latent code at timestep t and C is a *tuple* of conditional signals.

To generate an image, a noisy latent z_T is randomly sampled and processed through denoising by the U-Net over a fixed number of timesteps, denoted as T. The iterative denoising process transforms z_T into a clean latent z_0, which is subsequently utilized by the decoder D to generate the image. Specifically, at timestep t, the denoised latent z_{t-1} is sampled based on z_t and $\tilde{\epsilon}_\theta(z_t, t, C)$, which is computed using classifier-free guidance [104]. Here is an example with two elements in C, given by:

$$\begin{aligned}\tilde{\epsilon}_\theta(z_t, t, C) =& \epsilon_\theta(z_t, t, \{\varnothing, \varnothing\}) \\ &+ s_1(\epsilon_\theta(z_t, t, \{c_1, \varnothing\}) - \epsilon_\theta(z_t, t, \{\varnothing, \varnothing\})) \\ &+ s_2(\epsilon_\theta(z_t, t, \{c_1, c_2\}) - \epsilon_\theta(z_t, t, \{c_1, \varnothing\})),\end{aligned} \tag{5.1}$$

where c_1 and c_2 denote two conditional signals, with $\varnothing$ representing a null value (*e.g.*, a black image for image condition). s_1 and s_2 are the weights for c_1 and c_2, respectively. Eq. 5.1 can also be easily rewritten to suit the case of one or three conditional signals.

For clarity, we primarily use the task of costume editing to illustrate the pipeline. The goal is to transform a person with a regular outfit into a Santa Claus costume.

5.2.2 Paired Data Generation

The goal is to design a data generation strategy that can produce paired exemplars aligned with a specified editing direction (*e.g.*, from regular to Santa Claus costumes) defined by text prompts. However, generating pairs with perfect spatial and identity alignment is very challenging. Thus we seek to design a strategy (Figure 3.2 left) that can generate reasonably good pairs, meeting these essential criteria: (1) the user identity in input x_A and ground truth x_B should match as closely as possible; (2) x_A and x_B should have rough

(a) Prompt-to-Prompt Strategy

(b) Our Strategy w/o Pose

(c) Our Strategy w/o ID

(d) Our Strategy

Figure 5.3: Examples of pairs generated by different strategies. Prompt-to-Prompt (a) fails to produce pairs with consistent identity. Without pose condition, (b) produces pairs with significant spatial misalignment. Without identity conditions, (c) results in pairs with face shapes difference. Our strategy (d) significantly improves these issues.

spatial alignment; (3) the data should cover a diverse range of user appearances (for better generalization).

One straightforward idea suggested by InstructPix2Pix [25] is to use GPT-3 [26] for generating a pair of text prompts in the source and target domains. These generated prompts are then employed to create x_A and x_B using a pretrained Stable Diffusion model [218] and the Prompt2Prompt image editing technique [98]. However, this method often results in unsatisfactory x_B as it fails to preserve the identity in x_A, as depicted in Figure 5.3 (a).

Instead, we build a conditional pair generation strategy on top of Composable Diffusion [165] to meet the three requirements. Key designs include: (1) Following [165], we generate x_A and x_B within a single image achieved through a single denoising process. This helps generate consistent identities in x_A and x_B (criterion 1). (2) We incorporate pose information to improve spatial alignment (criterion 2). (3) We extract identity information from real photos and use this information to ensure criterion 1 and 3.

To implement design (1), we employ a pretrained Stable Diffusion in conjunction with Composable Diffusion [165] to generate an image $x = [x_A, x_B] \in \mathbb{R}^{H\times 2W\times 3}$, where the operator $[\cdot,\cdot]$ represents the horizontal concatenation of two images. Here, H and W denote the height and width of x_A and x_B. Further, the design (2) and (3) are implemented as conditions to guide the denoising process of x.

Specifically, we begin by randomly initializing a latent code $z_T \in \mathbb{R}^{h\times 2w\times 4}$, where $h = H/8$, $w = W/8$, and 4 represents the feature dimension of the latent code. At each timestep t, we compute the predicted noise by combining three classifier-free guidance results:

$$\begin{aligned}\bar{\epsilon} =& s'_d \cdot \tilde{\epsilon}_{\theta'}(z_t, t, \{c_p, c_{id}\}) + \\ & s'_a \cdot M'_a \odot \tilde{\epsilon}_{\theta'}(z_t, t, \{c_{p_a}, c_{id}\}) + \\ & s'_b \cdot M'_b \odot \tilde{\epsilon}_{\theta'}(z_t, t, \{c_{p_b}, c_{id}\}),\end{aligned} \tag{5.2}$$

where c_p, c_{p_a}, and c_{p_b} represent text embeddings computed from the shared prompt p, the source prompt p_a, and the target prompt p_b, respectively. In the example of Figure 5.2, p is "the same woman on the left and right", p_a is "a woman, normal costume", and p_b is "a woman, santa claus costume". c_{id} denotes identity embeddings (design (3)) extracted from a real-world portrait image using a variant of CLIP-based identity encoder [279], trained on the FFHQ dataset [130]. This encoder translates an image into multiple textual word embeddings, thus can be combined with c_p, c_{p_a}, and c_{p_b} to provide identity information for the denoising process. See supplementary for further details.

The matrices M'_a and M'_b are defined as $[\mathbf{1}, \mathbf{0}]$ and $[\mathbf{0}, \mathbf{1}]$ respectively, both belonging to $\mathbb{R}^{h\times 2w\times 4}$. Here, $\mathbf{1}$ ($\mathbf{0}$) represents a matrix in the dimension $h \times w \times 4$ with all values set to one (zero). Additionally, the variables s'_d, s'_a, and s'_b signify the strengths associated with each predicted noise. Furthermore, the denoising process is guided by a pose image (design

(a) Input (b) Output w/o ID (c) Output with ID

Figure 5.4: Training on a dataset with less diverse identities (b) results in inconsistent identity with the input (a). Conversely, training on a dataset with diverse identities yields the desired editing outcome (c), demonstrating its better generalization ability.

(2)) using the OpenPose [27] ControlNet [322], as shown in Figure 5.2 top left. This pose image ensures alignment by featuring the same pose in both the left and right parts of the image. The pair generated by our approach is depicted in Figure 5.2 on the left.

Notably, both design (2) (for pose) and design (3) (for identity) play a crucial role in generating good pairs. Figure 5.3 illustrates this point. Dropping one of them results in considerable spatial misalignment (b) and noticeable differences in facial shape (c). In addition, design (3) also contributes to generating diverse individuals across different pairs. This is crucial for enhancing generalization ability, as shown in Figure 5.4.

5.2.3 Training Multi-Conditioned Diffusion Model

Although the generated pairs are reasonably good, they are still not perfect. For example, in Figure 5.2, the face in x_B is slightly wider than that in x_A. The imperfection can potentially confuse the model and harm the performance.

Therefore, given these imperfect pairs, we design an image editing model to effectively learn pertinent information, such as editing direction and preservation of untargeted subject

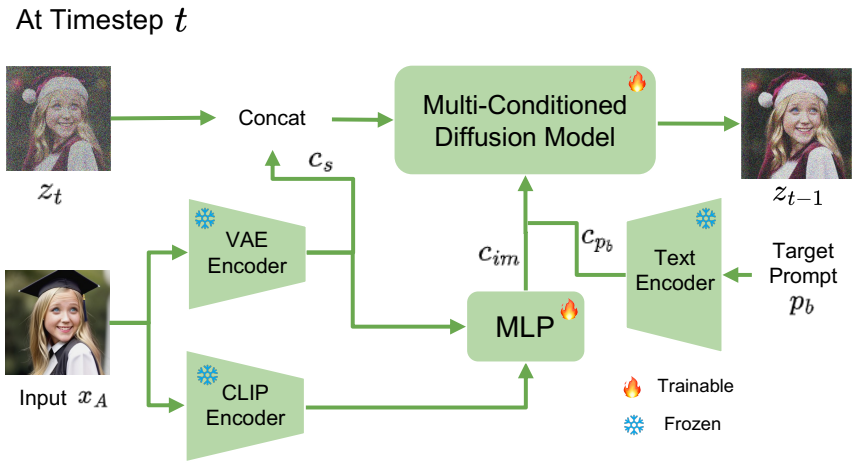


Figure 5.5: **Illustration of Multi-Conditioned Diffusion Model**. Both image and text embeddings are injected into the model through different ways to effectively learn the editing direction and preserve subject features.

features, from the generated pairs while simultaneously filtering out unexpected noise – specifically, small variations in identity and layout. Inspired by [125], the key design of our model is to integrate various conditions into the Stable Diffusion architecture in distinct ways. We call our model Multi-Conditioned Diffusion Model (MCDM). We will first define these conditions, and later elaborate how they help learn pertinent information from imperfect data through different injection ways. The details of the MCDM are shown in Figure 5.5.

Our model $\epsilon_\theta(z_t, t, \{c_s, c_{im}, c_{p_b}\})$ considers three pathways of conditional signals: (1) spatial embeddings $c_s = E(x_A)$, (2) text embeddings c_{p_b}, extracted by pretrained Stable Diffusion text encoder with target text prompt p_b as input, (3) image embeddings $c_{im} = MLP([E(x_A), CLIP_{im}(x_A)])$, where $CLIP_{im}(\cdot)$ denotes embeddings extracted from the pretrained CLIP image encoder [206]. $MLP(\cdot)$ is a multi-layer perceptron that projects image embeddings to the space of text embeddings.

To incorporate these embeddings into our model, we make modifications to the Stable Diffusion architecture as follows. (1) To prevent the imperfections in x_B from misleading

the model into generating an output $\hat{x}_B$ that alters the layout and identity in x_A, we concatenate the spatial embeddings c_s with the noisy latent z_t (input of U-Net). The resulting concatenation is then utilized as the input for the U-Net. Architecturally, the first layer of the U-Net encoder is adjusted to accommodate an additional 4 channels (for c_s), increasing the total to 8 channels. (2) c_{p_b} and c_{im} are concatenated and fed into the cross-attention layer, akin to the Stable Diffusion architecture. Functionally, c_{p_b} includes crucial information about the target domain as instructed by the text prompt, steering the output $\hat{x}_B$ towards the desired domain B. Simultaneously, c_{im} contributes visual information derived from the input image to the cross-attention layer, offering visual guidance in the attention mechanism. This prevents $\hat{x}_B$ from strictly adhering to the text instruction, ensuring that the output remains connected to the visual context of x_A and preventing undue deviation.

We initializes network weights with pretrained Stable Diffusion [218]. The training scheme is similar to Stable Diffusion, but with several differences: (1) we replace c_{p_b} with c_{p_a} and x_B with x_A by 5% of time. This enables the model to reconstruct input images (*i.e.*, perform identical editing), which will be utilized during the inference phase for mask generation. (2) Inspired by [125], we implement a dropout mechanism for multiple signals for classifier-free guidance. Specifically, with a 20% probability, we drop any combination of the following: c_s, c_{im}, c_p, or even all of them.

Figure 5.6 illustrates the ablation of these design choices, underscoring the effectiveness of employing all conditional signals simultaneously, as previously discussed.

5.2.4 Mask-Guided Editing using Trained Model

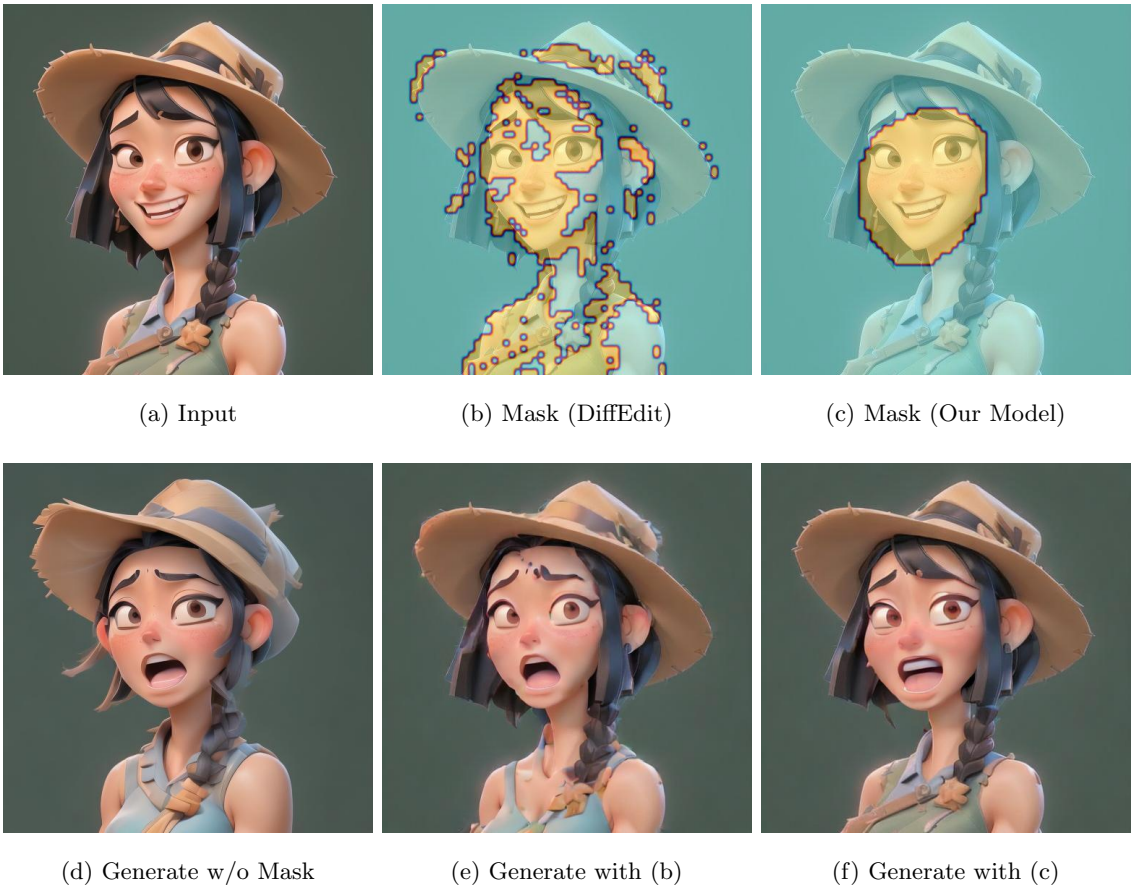
After training, the standard approach for generating predictions $\hat{x}_B$ from x_A involves denoising a random latent z_T over T iterations using trained model (with classifier-free guidance). While the generated $\hat{x}_B$ successfully accomplishes the desired edits while preserving identity and layout, challenges may persist in retaining specific details of the subject's features. For example, in Figure 5.7, an illustration of expression editing (to a shocked expression) depicts the standard generation output (d), where the hat and upper clothing patterns differ from those in the input image (a).

(a) Input (b) Ours w/o Prt (c) Ours w/o Spt

(d) Ours w/o Iemb (e) Ours w/o CFG (f) Ours

Figure 5.6: **Ablation Study of Design Choice of MCDM**. The goal is to have the person in (a) wear a royal costume. Training from scratch (b) yields the poorest image quality due to the absence of image generation priors and text prompt interpretation. Dropping spatial embeddings (c) fails to preserve spatial layout and the person's hairstyle. Excluding image embeddings (d) causes "over-editing" towards the target domain, compromising fidelity (*e.g.*, the golden face in (d)). Without classifier-free guidance, less expressive edits emerge (e) (*e.g.*, incomplete crown). In contrast, our full pipeline (f) produces the best editing results.

To enhance the preservation of these details, a mask can be derived from the trained MCDM, providing explicit guidance for the denoising process. This mask indicates areas



(a) Input (b) Mask (DiffEdit) (c) Mask (Our Model)

(d) Generate w/o Mask (e) Generate with (b) (f) Generate with (c)

Figure 5.7: **Comparison of Mask-Guided Editing on Cartoon Expression Editing (to Shocked Expression).** Standard generation (d) alters details (*e.g.*, patterns on hats and upper clothing) in the input image (a). Applying the mask generation strategy to our model improves the accuracy of the generated mask (c) compared to the one generated by DiffEdit (b). When guided by the mask in (c), the edited image (f) effectively preserves details (e.g., clothing) compared to the one (see (e)) guided by (b).

for editing and those to be left untouched. We adapt DiffEdit [51] to automatically generate such a mask. The key difference between our and DiffEdit's mask generation strategy is that, instead of relying on a pretrained Stable Diffusion model, we leverage our trained MCDM with its reconstruction capabilities to achieve more precise mask generation. By

applying DiffEdit to our trained MCDM instead of the original Stable Diffusion model, we can achieve more precise mask generation due to MCDM's reconstruction capability.

Figure 5.7 (c) shows an example of editing mask generated by our trained model, which is more accurate than the one produced by the DiffEdit used to produce pairs (Figure 5.7 (b)). This demonstration underscores the MCDM's capacity to discern the types of content that should be edited, even by training on an imperfect dataset.

Once we have the mask M, at each timestep t, we calculate the mask-guided predicted noise by:

$$\hat{\epsilon} = \tilde{\epsilon}_\theta(z_t, t, \{c_s, c_{im}, c_{p_b}\}) \odot M \; + \; \tilde{\epsilon}_\theta(z_t, t, \{c_s, c_{im}, c_{p_a}\}) \odot (1 - M).$$

It implies that we denoise for target editing (using p_b) within the mask, and preserve the original image content (using p_a) outside the mask. Figure 5.7 (e) shows the result with mask guidance. See implementation details in supplementary.

5.3 Experiments

Datasets: We evaluate the performance of our pipelines in two distinct portrait editing tasks: costume editing and cartoon expression editing. For each task, we define four different editing directions for input in a specific domain. For costume editing, the input image is a realistic portrait image with everyday costume, and the output is the same person with flower, sheep, Santa Claus, or royal costume. For cartoon expression editing, the input image is a cartoon portrait with a neutral expression, while the output is the same cartoon character with four different expressions: angry, shocked, laughing, or crying. For each task, we generate a dataset of 69,900 image pairs (17475 for each editing direction) for training. See details in supplementary.

Baselines: We choose 6 state-of-the-art image editing baselines for comparison. In particular, Prompt2Prompt (P2P for short) [98], pix2pix-zero [197], DiffEdit [51], SDEdit [179] are training-free diffusion methods with editing direction guided by text prompt. Since SDEdit is sensitive to a strength parameter, we test two different parameters of it, namely SDEdit 0.5 and SDEdit 0.8. Larger strength produces outputs that obeys the editing directions but deviates from the input images. SPADE [196] and BBDM [148] are training-based image

	Costume Editing			Expression Editing		
	SSIM	LPIPS	FID	SSIM	LPIPS	FID
Prompt2Prompt	0.764	0.694	83.94	0.936	0.426	33.13
pix2pix-zero	0.695	0.812	120.8	0.897	0.517	46.65
DiffEdit	0.722	0.721	93.52	0.922	0.423	32.20
SDEdit 0.8	0.681	0.766	93.17	0.914	0.461	38.59
SDEdit 0.5	0.713	0.712	68.85	0.925	0.445	38.99
SPADE	0.761	0.651	67.96	0.936	0.427	31.02
BBDM	0.776	0.676	64.17	0.937	0.432	27.26
Ours w/o Prt	0.734	0.672	84.58	0.940	0.420	28.82
Ours w/o Spt	0.767	0.665	78.38	0.735	0.430	24.56
Ours w/o Iemb	0.785	0.636	65.74	0.941	0.408	23.07
Ours w/o Mask	0.788	0.639	63.52	0.940	0.421	26.61
Ours	**0.791**	**0.633**	**52.56**	**0.943**	**0.400**	**22.09**

Table 5.1: **Quantitative Results of All Tested Methods**. Our method outperforms all tested baselines and variants over all metrics. Please note, there are no metrics that can accurately describe the performance of models because the ground truths we used are not real and unique truths.

editing framework building on top of Generative Adversarial Networks [52] and diffusion model [53], respectively.

Real-World Applications: We showcase the practical applications of models trained on two datasets through two distinct scenarios. The first application revolves around real portrait costume editing, wherein the inputs are in-the-wild portrait images. As shown in top 4 rows in Figure 5.8, both training-based and training-free methods yield unsatisfactory results; the former exhibits noticeable artifacts, while the latter often fails to align with the provided prompts.

The second application is sticker pack generation. Here, the objective is to generate a

Figure 5.8: **Results Comparison.** existing methods either fail to apply desired edits (*e.g.*, SDEdit in first 4 rows) or struggle to figure out which region to apply the edits.

(a) Input (b) P2P (c) DiffEdit (d) SDEdit (e) BBDM (f) Ours (g) GT

Figure 5.9: **Comparison on Validation Set.** In row 1 (sheep costume), the training-free baselines (b) to (d) fall short of achieving the intended edits , while the training-based methods (e) exhibit noticeable artifacts on eyes. In row 2 (angry), all baselines change the subject identity (*e.g.*, missing glasses, wrong hair color and clothing). In contrast, our method produces high-quality editing results while preserving the identity. Note that validation set input is from generated pairs, so baseline results look better than Figure 5.8.

cartoon sticker pack based on an in-the-wild portrait image. To achieve this, we initially perform data augmentation, incorporating processes such as cropping and homography, on the real input image. These augmented data are then employed to train a DreamBooth [219]. Subsequently, the trained DreamBooth is utilized to generate a cartoonized portrait image of the subject, guided by a meticulously crafted text prompt. Finally, our model is applied to the cartoonized image to produce outputs featuring four distinct trained expressions. Please note, directly utilizing DreamBooth to generate images with various expressions doesn't yield satisfactory results due to the layout change and overfitting issues. As shown in Figure 5.8 (bottom 4 rows), training-free baselines outperform their training-based counterparts. This is because the training-based baselines are not robust enough to handle imperfect training pairs. In contrast, our method outperforms all baselines in both editing fidelity and the preservation of the subject's features, while maintaining high image quality.

User Study: We conducted a user study on two real-world applications, each with 12 examples. Participants were presented with inputs and outputs generated by DiffEdit, SDEdit 0.5, SPADE, BBDM, and our proposed pipeline, randomly shuffled. The 32 participants were asked to give a rating from 1 to 5 (higher means better) for each output. We normalized the rating of each example and user to remove the user bias. In costume editing task, our method achieves the highest average rating, surpassing DiffEdit by 3.3 times, SDEdit 0.5 by 1.8 times, SPADE by 2.1 times, and BBDM by 2.5 times. Similarly, for the expression editing, our method receives the best rating, outperforming DiffEdit by 1.7 times, SDEdit 0.5 by 1.4 times, SPADE by 2.9 times, and BBDM by 1.6 times. These results demonstrate that our method consistently produces superior visual outcomes compared with baselines in both tasks.

Comparison on Validation Set: For quantitative evaluation, we create a validation dataset for each task by generating 1,000 image pairs in two distinct ways. The first approach involves generating paired data following the same methodology described before, resulting in 100 pairs. For the second method, we adopt a different strategy aimed at introducing subjects not present in the FFHQ dataset. We exclude identity embeddings and add detailed text descriptions of individuals (generated by ChatGPT) to p, p_a, and p_b. This yields an additional 900 pairs for evaluation. We believe a more comprehensive evaluation can be conducted by combining these two types of pairs. Figure 5.9 and Table 5.1 show that our method outperforms all tested baselines.

Ablation Study: We conduct experiments to assess the effectiveness of each component of our model, resulting in four variants: (1) *Ours w/o Prt*, training our model from scratch, (2) *Ours w/o Spt*, removing spatial embeddings c_s, (3) *Ours w/o Iemb*, excluding the image embeddings c_{im}, and (4) *Ours w/o mask*, eliminating mask guidance during inference. We did not test variants without text conditions since we trained 4 editing directions using one model in the evaluation, and text conditions are used to determine which types of editing to perform at the test time. As discussed in sec. 5.2, Table 5.1, Figure 5.6, and Figure 5.7 show that our final design outperforms these variants.

Limitations and Future Work: The dataset generation strategy assumes Stable Diffusion can generate images in the source and target domains, which might not always be the case.

The editing performance is compromised when handling datasets with most of the pairs with significant noise, such as substantial layout and identity differences.

In the future, we will (1) move away from the constraint of paired data and explore methods for handling unpaired data effectively, (2) reduce the required amount of training data, making the pipeline more efficient and scalable.

5.4 Conclusion

In this paper, we aim for portrait editing such as changing costumes and expressions while preserving the untargeted features. We introduce a novel multi-conditioned diffusion model, trained on training pairs generated by our proposed dataset generation strategy. During inference, our model produces an editing mask and uses it to further preserve details of subject features. Our results on two editing tasks demonstrates superiority over existing state-of-the-art methods both quantitatively and qualitatively.

SUPPLEMENTARY MATERIAL

5.5 Implementation Details

We present implementation details here, and the code will be publicly available after acceptance.

5.5.1 Paired Data Generation

Different pretrained Stable Diffusion models are employed for generating two datasets. The costume editing dataset is created using RealisticVision v4.0 [227]. For the cartoon expression editing dataset, fine-tuning is performed on RealisticVision v1.3 [226] using 400 images generated by the Samaritan 3D cartoon pretrained Stable Diffusion model [204]. The fine-tuned model is then used as base model to generate the cartoon expression editing dataset.

For text prompt, we set p to "the same [X] on the left and right", where [X] corresponds to the gender (either man or woman) detected from the portrait image used to extract identity information. p_a is "a [X], [Y]", where [Y] refers to the description of the source domain (*i.e.*, normal costume or neutral expression). p_b is "a [X], [Z]", where [Z] is the description of the target domain. In costume editing task, [Z] are "cute flower costume", "cute white sheep costume", "Santa Claus costume", or "traditional golden palace costume". In expression editing task, [Z] are "angry", "shocked", "laughing", or "crying". In addition, we set $s'_d = 0.1$ and $s'_a = s'_b = 0.9$.

For the CLIP-based identity encoder, we train it on the FFHQ dataset [132]. As a variant of the image encoder and global mapper in [279], our architecture consists of a CLIP image encoder [206] followed by a Decoder layer [257] and a multi-layer perceptron. This identity encoder translates an input image to multiple textual word embeddings, which can be regarded as "new words" in the textual word embedding space. The resultant embeddings c_{id} have a dimension of 2×768, which can be regarded as using two words to describe

the identity. Finally, c_{id} is inserted into the position between "[X]" and its previous word in c_{p_a}. In other words, the "new words" are inserted right before "[X]". Same operation is also performed for c_p and c_{p_b}. Images from the FFHQ dataset are utilized for identity information extraction due to the dataset's broad diversity of identities across its entirety.

The pose image is generated using Stable Diffusion with the text prompt "a person, head shot", resulting in 1000 candidate poses. From these candidates, one pose is randomly selected to serve as the pose condition for generating a pair.

For both datasets, the image size is set to $H = W = 512$. The denoising timestep is set to $T = 20$, and the guidance scale of the classifier-free guidance is set to 7.5. We use Stochastic Karras VE scheduler [126] for denosing.

5.5.2 Training Multi-Conditioned Diffusion Model

In the Multi-Conditioned Diffusion Model (MCDM), we implement $MLP(\cdot)$ following the structure described in the adapter in [125]. During training, the weights in the first layer of the U-Net related to spatial embeddings c_s are zero-initialized. For other weights, we use the same pretrained Stable Diffusion model, which was adopted for paired data generation, as the base model for initialization.

The training setup includes a batch size of 4 and a learning rate of 5e-6. All models are trained for 200,000 steps, and the training process takes approximately 3 days on a single A100 GPU.

5.5.3 Mask-Guided Editing using Trained Model

During the inference phase, we set the timestep T to 20 and utilize the UniPC scheduler [330] for all tasks. Regarding the guidance scale for the classifier-free guidance, we configure $s_1 = 3$, $s_2 = 3$, and $s_3 = 5$. Here, s_1, s_2, and s_3 correspond to image embeddings, input embeddings, and text embeddings, respectively.

For detailed implementation of mask computation, we first add random noise to the input image x_A for 10 steps using the UniPC scheduler, resulting in z_t. Subsequently, we perform one-step denoising processes twice using the trained model, each with a distinct

prompt – p_a and p_b. This yields z'_{t-1} (for reconstruction) and z_{t-1} (for target editing), respectively. The distinction between z_{t-1} and z'_{t-1} is interpretable as the contrast between applying editing and not applying it. This difference is then utilized to compute the editing mask. Noteworthy is the use of p_a for reconstruction, which, as opposed to utilizing a user-defined source prompt in DiffEdit, ensures exact reconstruction and facilitates superior mask computation. Now, we can compute a binary editing mask M by $N(|z_{t-1} - z'_{t-1}|) > \lambda$, where regions with a mask value of 1 indicate areas related for editing, while those with a value of 0 remain unaltered. λ serves as a constant threshold, where a larger value results in fewer regions being marked for editing. We set λ to 0.2 and 0.35 for costume and cartoon expression editing, respectively. The operation $N(\cdot)$ denotes the normalization of the values to a range from 0 to 1, followed by a Gaussian blur with a kernel size of 5.

In practice, following the approach of DiffEdit [51], the above process is repeated with a set of 10 different z_{t-1} and z'_{t-1} (generated with different random seeds) to enhance the stability of the mask computation. This helps mitigate the potential impact of variations introduced by different random seeds during the noise addition step.

5.6 Experiments

5.6.1 Datasets

For dataset generation details, please refer to section 5.5.1 for details.

5.6.2 Real-World Applications

Fig. 5.10 and Fig. 5.11 show more results of costume editing task. Fig. 5.12 and Fig. 5.13 showcase more expression editing results. In summary, our method outperforms all baselines in both editing fidelity and the preservation of the subject's features, while maintaining high image quality.

5.6.3 Comparison on Validation Set

Fig. 5.14 and Fig. 5.15 show the comparison between our method and all baselines on costume editing and expression editing dataset, respectively. Our method outperforms all

Figure 5.10: Costume editing comparison with baselines on in-the-wild images. Editing directions from top to bottom are to flower, sheep, Santa, and royal costume. Our method produces high-quality editing results while preserving the subject features.

Figure 5.11: Costume editing comparison with baselines on in-the-wild images. Editing directions from top to bottom are to flower, sheep, Santa, and royal costume. Our method produces high-quality editing results while preserving the subject features.

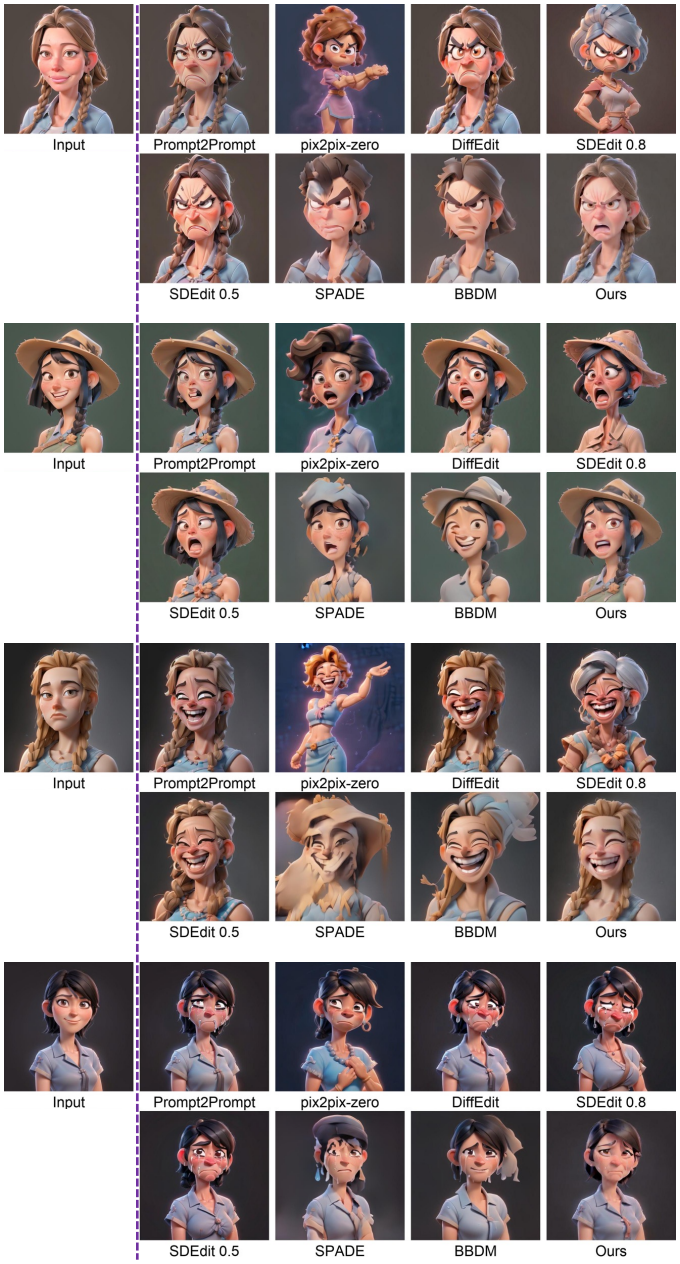


Figure 5.12: Expression editing comparison with baselines on cartoon input produced by DreamBooth. Editing directions from top to bottom are to angry, shocked, laughing, and crying expression. Our method produces high-quality editing results while preserving the subject features.

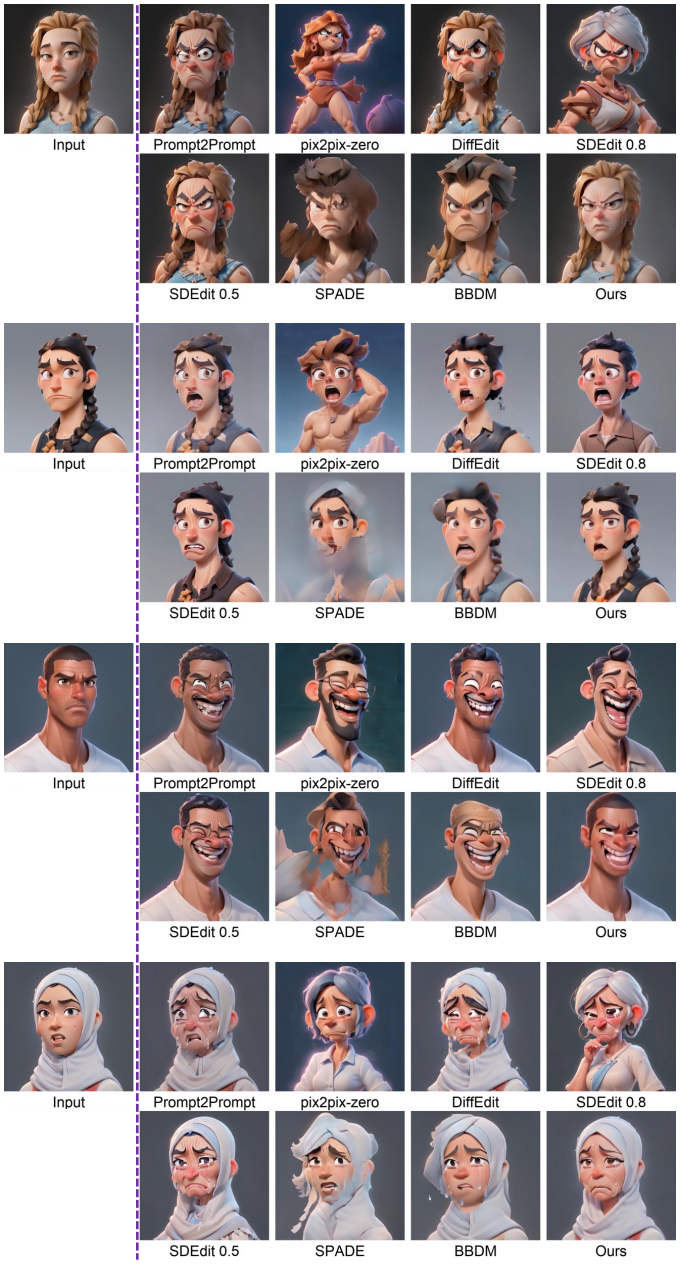


Figure 5.13: Expression editing comparison with baselines on cartoon input produced by DreamBooth. Editing directions from top to bottom are to angry, shocked, laughing, and crying expression. Our method produces high-quality editing results while preserving the subject features.

baselines in generating high-quality editing outputs.

5.6.4 Ablation Study

Fig. 5.16 shows additional results of ablation of different condition for MCDM, demonstrating the effectiveness of our final design.

Figure 5.14: Costume editing comparison on validation set. Editing directions from top to bottom are to flower, sheep, Santa, and royal costume. Our method produces high-quality editing results while preserving the subject feature. Note that validation set input is from generated pairs, so baseline results look better than those in real-world applications.

Figure 5.15: Expression editing comparison on validation set. Editing directions from top to bottom are to angry, shocked, laughing, and crying expression. Our method produces high-quality editing results while preserving the subject feature. Note that validation set input is from generated pairs, so baseline results look better than those in real-world applications.

Figure 5.16: Ablation Study of design choice of different conditions of MCDM. Training from scratch (b) yields the poorest image quality due to the absence of image generation priors and text prompt interpretation. Dropping spatial embeddings (c) fails to preserve spatial layout. Excluding image embeddings (d) fails to learn precise editing direction, leading to artifacts in the outputs. In contrast, our full pipeline (e) produces the best editing results.

Chapter 6

INVERSE PAINTING: RECONSTRUCTING THE PAINTING PROCESS

This chapter presents the research project *Inverse Painting: Reconstructing The Painting Process* [36], conducted with Yifan Wang, Brian Curless, Ira Kemelmacher-Shlizerman, and Steven M. Seitz. The subsequent analysis and comparisons to related studies in this chapter are based on the prevailing state of the art at the time of this work.

When we look at a painting, we see only the final outcome of the artist's creative process. Leonardo da Vinci worked on the Mona Lisa for 16 years – it would be fascinating to see a time-lapse video of the Mona Lisa's creation. While no such video exists for Leonardo, there are many videos online in which artists have filmed the creation of entire paintings [121, 3, 4, 116]. These visualizations are fascinating in the way they show hidden layers, structures, and provide insight into the artistic creation process. While such visualizations currently exist for only a tiny subset of paintings in the world, we propose to train machine learning models on such data to predict how a wide variety of other paintings could have been made. While the resulting videos should not be treated as accurate reconstructions of any specific painting's creation, they are nonetheless intriguing and insightful as *plausible* reconstructions, in capturing rules that many painters employ, such as layering, back-to-front-ordering, and focusing on objects/regions in stages[63, 3, 21, 121]. While our approach was trained only on acrylic paintings of landscapes, as shown in Fig. 6.2, we believe that future models could produce plausible visualizations for almost any piece of art.

Previous approaches [111, 239, 76, 238, 56, 110, 344, 167] rely on hand-crafted painting principles instead of learning them from the real painting processes. The most relevant work, Timecraft [328], generates time-lapse videos by learning from actual painting videos. However, it operates only on low-resolution (50x50 pixels) patches and therefore lacks holistic semantic context.

Figure 6.1: We present Inverse Painting, a diffusion-based method to generate time-lapse videos of the painting process from a target painting. By training on acrylic paintings with a similar artistic style to that of the first example in this figure, our method is capable of handling a diverse range of styles (e.g., oil painting, above bottom). The resulting videos resemble how human artists typically paint, for example, from back to front, focusing on semantic objects or regions at a time, and employing layering techniques. Image courtesy Catherine Kay Greenup.

Figure 6.2: **How a real artist paints.** Time lapse from a real painting video, representative of the training painting style. The artist uses a back-to-front order with layering techniques, starting with the sky, then clouds, mountains, and other elements. The artist typically focuses on one semantic region at a time.

We define this task as an autoregressive image generation problem. It begins with a blank canvas, which is iteratively updated based on its current state and the target painting.

This sequential updating continues until the artwork is completed. To implement this, one possible approach, similar to Timecraft [328], is to train a network that takes the current and target images as inputs and outputs the updated canvas at each step. However, we've found that a purely pixel-based approach struggles to produce reasonable results in practice, and thus we incorporate additional semantic analysis.

In particular, we draw inspiration from the techniques used by human artists. Consider the second painting in Fig. 6.1. Initially, an artist decides on the semantic content to depict – such as the sky – and selects appropriate areas of the canvas for this content, such as the upper portion. Subsequently, the artist applies specific paints to these regions, using blue for the sky, while leaving contents such as clouds for later stages. Building on this observation, we design a two-stage method that decomposes the instruction generation and canvas rendering. In the first stage, we generate a textual instruction and a corresponding region mask from the current and target images. The textual instruction provides high-level guidance on the order of painting, and its corresponding region mask directs the focus area. In the second stage, a diffusion-based renderer is proposed to leverage the textual instruction and region mask, in conjunction with the current and target images, to effectively update the canvas.

Our method can handle in-the-wild landscape paintings across diverse artistic styles (*e.g.*, Impressionism and Realism) and color themes, ranging from dark to bright. We demonstrate that our approach surpasses current state-of-the-art methods in creating high-quality, human-like painting videos, supported by both qualitative and quantitative results, as well as human studies.

6.1 Related Work

6.1.1 Painting Process Generation

Stroke-based rendering is a computer graphics technique that creates non-photorealistic images by placing discrete elements like brush strokes instead of traditional pixels [87, 99, 246, 67, 161, 85, 76, 287, 168, 117, 69]. The painting process can be obtained by visualizing the sequence of element placement. However, many studies mainly focus on generating

images in various artistic styles [344, 161, 287, 168, 139]. They often overlook the order and position of brush stroke placement, resulting in a non-human-like painting process. To produce a more human-like painting process, recent work constrains the placement of brush strokes based on predefined painting principles using techniques such as reinforcement learning (RL) [111, 239, 76, 238, 56, 110, 344] or Transformers [167]. For example, [167] developed a Transformer-based [257] feed-forward painter that applies a coarse-to-fine painting strategy. Initially, the painter works from a blurry (downsampled) version of the target painting, progressively refining the canvas using higher-resolution versions until the painting is complete.

However, these techniques face two limitations: (1) They rely on hand-crafted painting principles that do not accurately mimic the actual human painting process, whereas our method learns from real-world painting data. (2) Their parameterized brush strokes fail to capture the full variation observed in real painting processes and result in an approximate version of the target painting. In contrast, our diffusion-based renderer effectively handles these variations and recreates the target painting more accurately.

Another stream of work, pixel-based generation, generates the painting process by directly updating pixels on the canvas [328, 147, 277]. [277] used a Vision Transformer (ViT) [62] to map a target image to a series of 9 intermediate images via a non-autoregressive approach. However, this method is specifically tailored for traditional Chinese painting. The work most related to ours is Timecraft [328], which generates time-lapse videos from paintings by learning from actual painting videos. However, their method is limited to low-resolution (50x50 pixels) crops, neglecting the full artwork's context. Our method handles full paintings with higher resolution.

6.1.2 Diffusion Models

Diffusion models have recently demonstrated their success in various tasks such as text-to-image [209, 221, 217], image-to-image translation [33, 298], and image-to-video synthesis [107, 20]. [20] developed a latent video diffusion model capable of generating videos from an initial frame. [107] proposed a diffusion framework to synthesize character animation

videos from a reference image and a sequence of target poses. This framework includes a component called ReferenceNet to inject features from the reference image directly into the denoising UNet of the diffusion model. In this paper, we adopt the ReferenceNet to inject the target image into our diffusion-based renderer.

6.2 Our Method

In a real painting process, an artist continually applies changes to the current state of the canvas. Our method formulates this process as an autoregressive image generation problem. Given a target painting I_T as input, we reconstruct a time-lapse video consisting of T *keyframes* $\{\hat{I}_1, ..., \hat{I}_T\}$, starting from a blank canvas I_0 progressively towards the target I_T. Each *keyframe* transition represents a fixed time interval, a feature of time-lapse videos. We start by presenting a one-step canvas rendering approach for each canvas update and discussing its limitations in Sec. 6.2.1. This motivates our two-stage design, which incorporates additional semantic analysis. The details of the training process are described in Sec. 6.2.2 and 6.2.3, while the testing is covered in Sec. 6.2.4. Fig. 6.3 shows an overview of the method.

6.2.1 One-Stage Canvas Rendering Approach

For clarity, we will refer to step $t-1$ as the current step and step t as the next step. The goal is to render next image $\hat{I}_t$ based on $\hat{I}_{t-1}$ and I_T. We approach this as an image translation problem, and design a diffusion-based renderer based on the denoising UNet g_u from Stable Diffusion [217]. This renderer leverages image priors to enhance image quality by initializing g_u with pretrained weights from Stable Diffusion.

We train this diffusion-based renderer with inputs $\{I_{t-1}, I_T, \Delta_t\}$, where I_{t-1} is the ground-truth (GT) current image, and Δ_t is the actual time interval between I_{t-1} and GT next image I_t. By incorporating Δ_t, we model the time spent on each update in the painting process, enabling the generation of videos with time-informed keyframes during testing. The output $\hat{I}_t$, which predicts the next image, is supervised using I_t. For supervision, we start with a clean latent $z_0 = E_I(I_t)$, where E_I is the pretrained VAE encoder in Stable Diffusion. We then apply the forward process to produce a noisy latent $z_s = \beta_s z_0 + (1 - \beta_s)\epsilon$, where

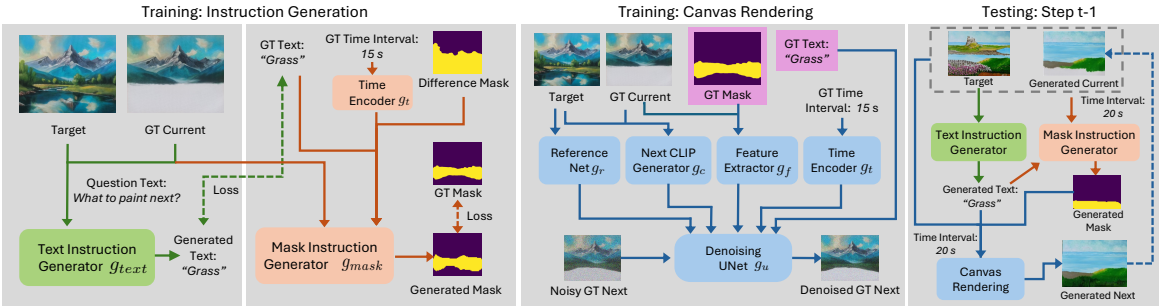


Figure 6.3: **Method overview.** The training has two stages. The instruction generation stage (left two gray boxes) includes the text instruction generator (green) and the mask instruction generator (light orange). These generators produce the text and mask instructions essential for updating the canvas in the next stage. The second stage is canvas rendering (third gray box), where a diffusion-based renderer generates the next image based on multiple conditional signals, such as text and mask instructions. Omitting the text and mask (purple boxes) yields the one-stage method described in Sec. 6.2.1. To simplify the figure, we omit the VAE encoder, CLIP encoder, and text encoder. During testing at step $t-1$ (last gray box), we first generate a text instruction (green arrows), which is then used to create a region mask (orange arrows). Both are then provided to the canvas rendering stage to produce the next image (blue arrows). Image courtesy Catherine Kay Greenup.

s denotes a randomly sampled denoising timestep, ϵ represents Gaussian noise, and β_s is a weighting parameter dependent on s. The denoising UNet g_u is then employed to denoise z_s. We update the parameters of the renderer by minimizing the following loss function:

$$\mathcal{L} = \mathbb{E}_{s,z_0,\epsilon}||g_u(z_s, c, s) - \epsilon||_2^2, \tag{6.1}$$

where c represents the conditional signals of g_u, consisting of features of $\{I_T, I_{t-1}, \Delta_t\}$. We extract these features using *ReferenceNet* g_r, *feature encoder* g_f, *time encoder* g_t and *next CLIP generator* g_c. During the training, we jointly update g_u, g_r, g_f, g_t, and g_c using Eq. 6.1. Please refer to the gray box "Training: Canvas Rendering" in Fig. 6.3 (excluding the mask and text in the purple box).

For ReferenceNet g_r, the target image I_T is integrated into g_u through g_r, as introduced

in [107]. The ReferenceNet g_r utilizes the same UNet architecture and pretrained weights (for initialization) from Stable Diffusion [217], ensuring feature extraction within the same feature space as g_u. It takes the target image I_T as input, and extracts intermediate feature maps from its self-attention layers. These feature maps are then fused into g_u by concatenating them with corresponding feature maps from the same layers in g_u. We opt for the design of ReferenceNet because it fuses features more effectively than other designs, such as ControlNet [322], in practice.

For Feature encoder g_f, it consists of a shallow convolutional neural network (CNN) to encode I_{t-1}, denoted as $g_f(I_{t-1})$. We found that this shallow network effectively learns features of I_{t-1} while saving computational time. The encoded feature map has the same spatial resolution of z_s (noisy latent of I_t), and is concatenated with z_s along channel dimension as spatial input to the g_u during training. The first layer of g_u is modified to adjust to the new channel dimension.

For time encoder g_t, it applies positional encoding – same as that used in NeRF [180] – to the Δ_t, followed by a multi-layer perceptron (MLP) that maps the positional encoding feature dimension from 21 to 768. This results in the time embeddings $g_t(\Delta_t)$. The advantage of positional encoding lies in its ability to adapt to varying Δt across different training samples. This continuous encoding enhances interpolation capabilities, useful for handling specific time intervals during testing. Moreover, to provide additional painting guidance for g_u, we introduce the next CLIP generator g_c, implemented as an MLP. This module inputs the CLIP embeddings of I_{t-1} and I_T, and outputs the predicted CLIP feature of the next image. This predicted CLIP feature and $g_t(\Delta_t)$ are concatenated and fed into the cross-attention layers of g_u.

Fig. 6.5 (b) illustrates the results of using this model without incorporating Δ_t. In this scenario, significant content is added in a single update, which does not effectively represent the progressive nature of the painting process. When Δ_t is included through g_t (Fig. 6.5 (c)), the new content volume per update is better controlled; however, this approach still results in unnatural rendering of mountains, as the yellow layers prematurely appear before the completion of the green mountain base. This indicates that a purely pixel-based network struggles to fully capture the semantics of the painting, prompting the need for more explicit

semantic controls.

6.2.2 Training: Instruction Generation

To address the aforementioned issues of the purely pixel-based method, we draw inspiration from typical artistic practices, where artists decide what to paint and where, then apply paint accordingly. Based on this observation, we propose a two-stage method (Fig. 6.3) where we first generate text and mask instructions as semantic guidance. These instructions are then used to steer the diffusion-based renderer. In this section, we describe the training of a text and mask generator. These two generators are used to produce two types of instructions for each canvas update: a text instruction $\hat{p}_t$, describes what semantic content to paint, and a region mask $\hat{M}_t$, specifies the focus regions corresponding to the text instruction.

Text Instruction Generator

Like an artist who envisions a target image and compares it to the current state of the work to decide what to paint next, our text instruction generator must discern visual content and generate appropriate textual instructions. This provides high-level guidance for the painting order. We implement this generator using the architecture of a large vision-language model LLaVA [162]. LLaVA accepts a single image combined with a question text prompt and produces a response text prompt. For training, as the generator requires a single image input, we horizontally concatenate the target image, I_T, and the ground-truth current image, I_{t-1}, to form the input. The text instruction $\hat{p}_t$ is generated as follows:

$$\hat{p}_t = g_{text}([I_T, I_{t-1}], p), \tag{6.2}$$

where g_{text} is the generator, and $[\cdot, \cdot]$ is horizontal concatenation of two images. p is the question text prompt (details in supplementary).

To enhance the model's performance and reduce training time, we initialize the text generator, g_{text} with pretrained weights of LLaVA 1.5 [162]. The text generator is fine-tuned with full supervision of the ground-truth text instructions p_t. The leftmost gray box in Fig. 6.3 visualizes this process, where "grass" is generated as the text instruction for the next step.

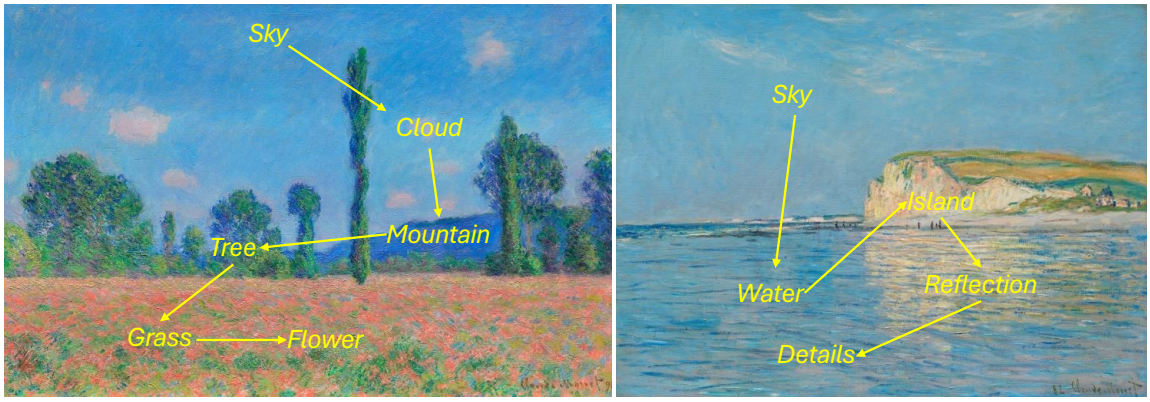


Figure 6.4: **Generated text instructions.** The sequence of generated text instructions (yellow text and arrows) demonstrate a natural painting order, arranging elements from back to front such as clouds over the sky, flowers over grass, and reflections over water. The "Details" in the right image refers to water texture and small details on the island. Each text instruction may repeat over multiple frames but is displayed only once for simplification[1].

Fig. 6.4 shows the text instructions generated during the painting process of in-the-wild paintings at test time. These instructions guide our renderer to use layering techniques, painting from back to front. This demonstrates that g_{text} not only captures the semantic essence of the target paintings but also effectively learns the painting order from the dataset.

Mask Instruction Generator

On top of what to paint, an artist also decides where on the canvas to apply the content. Our method formulates this as a binary region mask, $\hat{M}_t$. This mask provides instructions that specify focus regions for each step of the painting process. For training, as visualized in the second gray box in Fig. 6.3, our mask instruction generator considers 4 factors to determine the intended painting region: (1) The GT current image I_{t-1} and target image I_T. (2) Text instruction p_t. During training, we use the ground-truth instruction p_t to ensure that the training is guided by accurate instructional data. (3) Difference mask M_d,

[2]Image courtesy Claude Monet, Poppy Field (Giverny), 1890, The Art Institute of Chicago. Image courtesy Cleveland Museum of Art.

which represents areas of the current canvas that are still incomplete relative to the target image. This binary mask M_d is derived by computing the difference map between I_T and I_{t-1} using perceptual distance [325]. This difference map is then binarized using a threshold $\alpha = 0.2$, setting pixels with values above α to 1 in M_d. Formally, we define the operations of computing M_d as $D(I_T, I_{t-1}, \alpha)$. (4) Time interval Δ_t, which can influence the size of the intended painting regions, as less area may be covered when less time is available.

Considering all these factors, we design our mask instruction generator based on the UNet proposed in Stable Diffusion (but without noise as input). The cross-attention design of this UNet architecture allows us to seamlessly incorporate textual and time interval information as conditional signals.

The mask generator g_{mask} has two inputs. The first one is spatial input, which includes I_T, I_{t-1}, and M_d. Specifically, we begin by encoding I_T and I_{t-1} using the pretrained VAE image encoder E_I. This encoder reduces the spatial resolution by a factor of 8 and converts the channel dimension from 3 to 4. We then concatenate these encoded images with the downsampled M_d (notation remains unchanged for simplicity) along the channel axis to form the composite spatial input: $[E_I(I_T), E_I(I_{t-1}), M_d]$. This spatial input is fed into the UNet similarly to Sec. 6.2.1.

The second one is the conditional input, which encodes p_t and Δ_t. We first use the pretrained text encoder g_p in Stable Diffusion to encode p_t, producing a text embedding with 77 tokens, each of dimension 768. For Δ_t, similar to Sec. 6.2.1, we use g_t (same architecture but different weights) to compute time embeddings with dimension 1 x 768. The time embedding is treated as an additional token and concatenated with the text embeddings to form the 78-token conditional input $[g_p(p_t), g_t(\Delta_t)]$ for cross-attention layers. Formally, we denote the generation of $\hat{M}_t$ as follows:

$$\hat{M}_t = g_{mask}([E_I(I_T), E_I(I_{t-1}), M_d], [g_p(p_t), g_t(\Delta_t)]). \tag{6.3}$$

During the training, we keep g_p and E_I frozen, and update weights in g_{mask} and g_t. The training is supervised by the downsampled GT mask $M_t = D(I_t, I_{t-1}, \alpha)$ using the binary cross-entropy loss.

(a) Inputs (b) CLIP Embed (CE) (c) CE + Time Embed (TE) (d) CE + TE + Text (e) CE + TE + Mask (f) CE + TE + Text + Mask

Figure 6.5: **Effects of conditional signals.** (a) shows the current canvas and target image (inset). With only predicted CLIP embeddings of the next image (b), the model generates excessive content per update. Including time intervals (c) properly limits new content volume but results in unnatural mountain rendering. Omitting the mask instruction (d) causes the renderer to complete the mountain area in one step, relying heavily on the text instruction "mountain". Omitting the text instruction (e) results in generating some of the green lake (red arrow) before completing the mountain (mask shown in the inset). The full pipeline (f) updates the canvas at a reasonable pace, drawing the top of the mountain in green, before layering on the yellow region. Image courtesy Catherine Kay Greenup.

6.2.3 Training: Canvas Rendering

Canvas rendering aims to generate $\hat{I}_t$ using the current and target images, alongside text and mask instructions, within a specified time interval. We modify the diffusion renderer introduced in Sec. 6.2.1 to integrate text and mask instructions while keeping other components unchanged, as shown in Fig. 6.3 (third gray box). For training, we use the ground-truth text and mask. Specifically, we replace $g_f(I_{t-1})$ with $g_f([I_{t-1}, M_t])$. Here $[\cdot, \cdot]$ refers to concatenation along channel axis. We modify first layer of g_f to handle this change. The text p_t is encoded by g_p, concatenated with predicted CLIP embeddings and time embeddings, and then input into the cross-attention layers. We found these CLIP embeddings provide semantic features beyond what is captured by explicit text and mask instructions alone. For instance, consider column 2 in row 1 of Fig. 6.6, where the text "mountain" and its corresponding mask serve as the instructions. These instructions do not specify whether the mountain should be painted with intricate details or in a rough, preliminary form, with

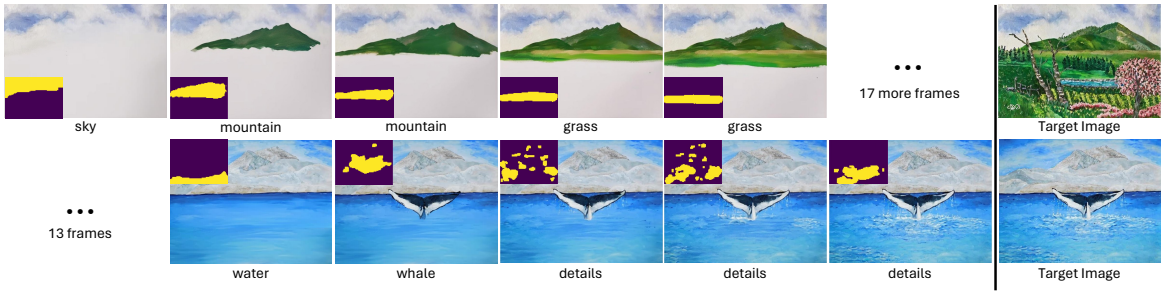


Figure 6.6: **Qualitative results with generated texts and masks.** We show five consecutive generated keyframes (left) given in-the-wild target images (far-right). The text (caption) and mask (inset) for each keyframe are the generated instructions used to produce that keyframe. The first row illustrates the early stages of the process, where our method paints sequentially from back (sky) to front (grass), typically focusing on one subject per keyframe based on the provided text and mask. For instance, keyframes 4 and 5 focus exclusively on grass, disregarding other elements like trees (layered on later) that occupy the same region of the target image. The second row depicts the final stages of the painting process for a different painting. Here, our method paints the water before layering on the whale. The process concludes with the addition of final details scattered throughout the painting (as guided by text and mask), mimicking techniques used by human artists. The resulting keyframe sequence reflect human-like decisions in painting order, semantic attention, and layering techniques. Images courtesy Catherine Kay Greenup.

finer details to be added later. Without CLIP embeddings, the model completes the mountain in full detail, deviating from the painting style of the training set. Incorporating the embeddings helps guide the model to adhere to the painting style.

Fig. 6.5 demonstrates the impact of excluding various conditional signals. By omitting the mask, variant (d) generates the entire semantic content (*i.e.*, mountain) at once, underscoring the importance of pixel-level mask guidance. Without text for high-level semantic guidance, variant (e) unexpectedly paints some of the green lake on the canvas. In contrast, the full pipeline (f) achieves the most natural result by integrating all these conditions.

6.2.4 Test-Time Generation

At inference time, we begin with a blank (white) canvas I_0, and update this canvas autoregressively using our trained pipeline to approximate a target painting I_T. We use a fixed time interval $\hat{\Delta}_t$ across all steps, following our definition of keyframes. This process is terminated when minimal updates are applied (see supplementary). We visualize an update in a specific step $t-1$ in Fig. 6.3 (the rightmost gray box). Given the current image $\hat{I}_{t-1}$ predicted by previous step, we first generate the text instruction $\hat{p}_t$ by employing Eq.6.2 and substituting I_{t-1} with $\hat{I}_{t-1}$. Then we compute the mask $\hat{M}_t$ using Eq.6.3 by replacing I_{t-1} with $\hat{I}_{t-1}$, M_d with $\hat{M}_d = D(I_T, \hat{I}_{t-1}, \alpha)$, p_t with $\hat{p}_t$, and Δ_t with $\hat{\Delta}_t$. Finally, to render $\hat{I}_t$, we start with random noise z_S and perform S denoising steps using $\{\hat{I}_{t-1}, I_T, \hat{p}_t, \hat{M}_t, \hat{\Delta}_t\}$ through diffusion renderer. This results in z_0 which is decoded by the VAE decoder from Stable Diffusion, producing $\hat{I}_t$. Please refer to the supplementary for details on training and test-time generation, including hyperparameters, execution time, and GT text annotations.

Fig. 6.6 visualizes consecutive keyframes of the painting process generated by our method, guided by the text and mask instructions. In the early stages of the process (row 1), the method typically focuses on a single semantic class per update. In latter stages of the process (row 2), once the background is completed, the focus switches to completing foreground and refining details. Here, "details" refer to fine elements typically painted in the later stages such as the ocean spray. In the second row of Fig. 6.6, columns 4-6 illustrate the finishing of these elements using the same text instruction "details" but different mask instructions over three frames.

6.3 Experiments

6.3.1 Datasets

We collected a dataset of 294 videos from the web [3], showcasing typical landscape acrylic painting processes featuring common themes like mountains, trees, flowers, and lakes. These paintings depict various times of the day and weather conditions. Each video averages 9 minutes, often sped up for quicker viewing. The footage comprehensively captures the complete painting process, including views of the canvas and palette as well as continuous

hand and paintbrush movements throughout the video.

For data preprocessing, we employed a pretrained segmentation method [166] to segment all video frames, cropping them to focus solely on the canvas areas. Further, the cropped frames showing hands, paint strokes, or minimal changes were excluded. After preprocessing, we divided the dataset into 265 training and 29 validation paintings. Both subsets have an average of 27 frames per artwork, with time intervals averaging around 23.6 seconds in training and 22.0 seconds in validation between frames. The training and validation sets have 7261 and 783 pairs. *During testing, the time interval is set to 20 seconds unless stated otherwise.*

6.3.2 Our Results

Fig. 6.7 shows the outcomes of our pipeline on in-the-wild paintings. First, the generated painting process resembles human-like painting orderings, typically painting from back to front, saving foreground objects and fine details for the last stages. For example, in row 2, the sequence starts with the sky, moves to the mountain, trees and water, and finishes with foreground objects and details such as the reflections, deck, and people. Second, each phase of the painting process focuses on specific semantic objects or areas. For instance, transitions from columns 2 to 6 in row 7 mainly focus sequentially on the distant bush, grass, and the foreground tree. Third, the underlayering contents are reliably rendered in the intermediate images when applying the layering techniques. For example, the base of the mountain is painted in row 1, column 2, with additional details added in column 3. Similarly, clouds are layered across the sky in row 6, columns 1 to 2. Finally, our method effectively handles artworks of various aspect ratios and artistic styles, including Impressionism and Realism. It also accommodates paintings with varied color themes. More results are in supplementary.

6.3.3 Baselines

We consider three baselines in the main paper (more baselines in supplementary). (1) *Time-craft* [328] employs a conditional variational autoencoder to create time-lapse videos from paintings. It trains on real painting videos to emulate the human painting process. How-

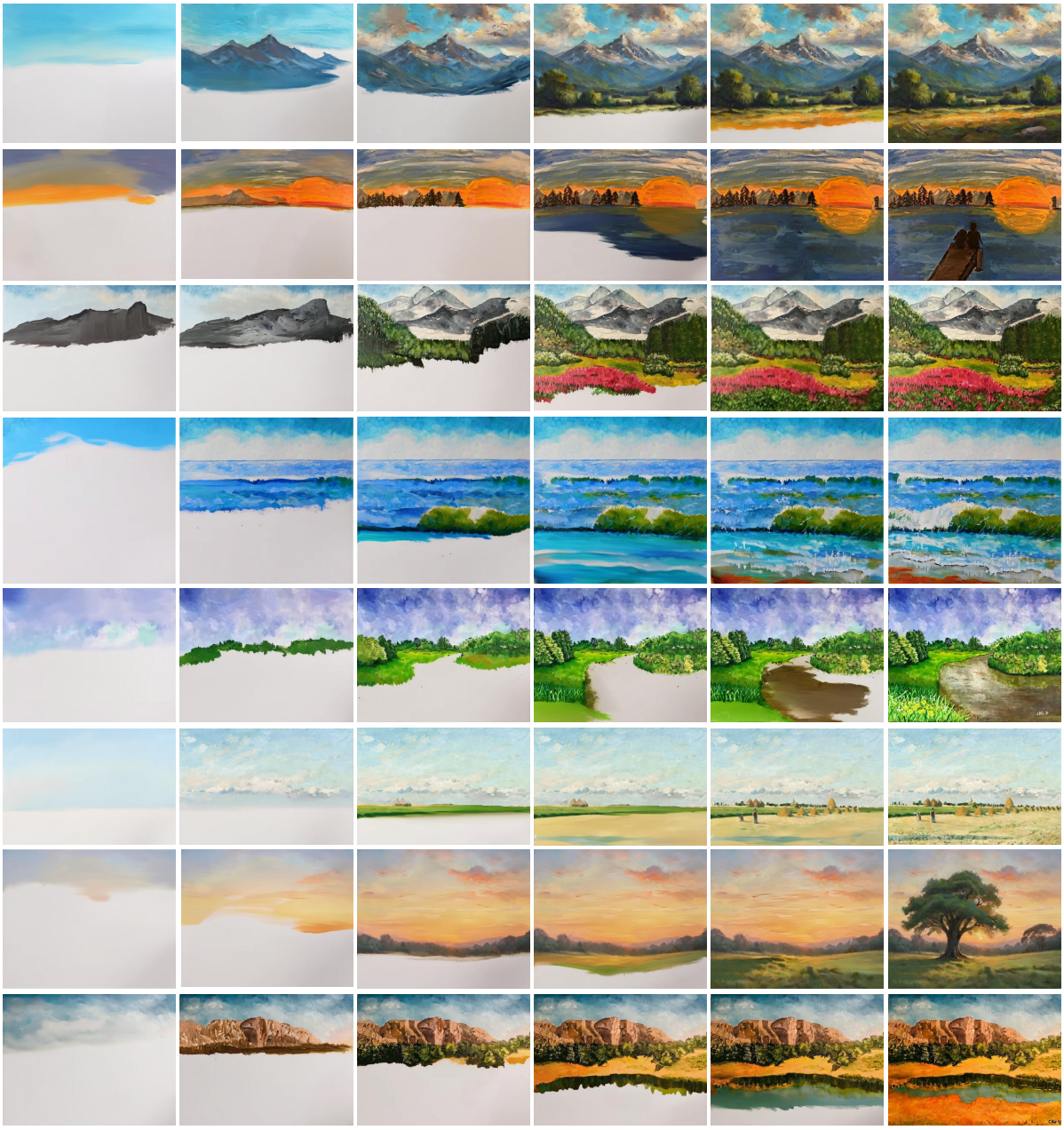


Figure 6.7: **Qualitative results.** We show results on in-the-wild paintings. Images courtesy Catherine Kay Greenup, National Gallery of Art, Washington, and Michelle Shlizerman.

ever, it handles only low-resolution 50x50 crops from downscaled paintings (126x168) due to computational limits. We trained the model on our dataset following their training strategy. For evaluation, we resized the target painting to 50x50 as input and scale the output videos back to the original painting's resolution. See the supplementary for evaluation on cropped paintings. (2) *Paint Transformer* [167] generates a stroke-level painting process from an input painting. This self-supervised method does not rely on real painting videos, but instead employs a hand-crafted coarse-to-fine strategy to mimic human painting process. We used the pretrained model provided by the authors for our comparisons. Please see supplementary for more stroke-based rendering baselines [110, 344]. (3) *Stable Video Diffusion (SVD)* [20] produces a 14- or 25-frame video given the first frame as input. We fine-tuned a 14-frame model on our dataset using LoRA[105]. During this fine-tuning, we sampled one frame from our training sequences as input and its previous 13 frames as ground truth, padding with white images where necessary. For inference, the target painting was input into the fine-tuned model to generate 14 frames. The final frame from this sequence initiated the next set of 14 frames, continuing until a white canvas is achieved.

6.3.4 Metrics

We aim to evaluate the methods in 5 aspects. (1) Human-likeness of the painting order. This measures whether the generated videos mimic human painting order. (2) Focus consistency in canvas updates. This checks if each update targets specific, reasonable areas. (3) Convergence speed towards the target painting. This evaluates the speed at which the generated video progresses towards the target painting. (4) Adherence to specified time intervals between keyframes. This evaluates whether the temporal progression between generated keyframes aligns with predefined intervals, reflecting the dynamics of an actual painting process. (5) Video quality. We design 5 metrics to cover these aspects.

- LPIPS. This addresses aspects (1) and (4). It calculates the perceptual distance between each frame in the real painting video and the closest generated keyframe based on the time. For example, a real frame at 1:17 is compared with the generated frame at 1:20, assuming a 20-second time interval.

- IoU (Intersection over Union). It evaluates aspect (2) without considering painting order. We first compute difference masks for consecutive frames in both real and generated videos using the function D (defined in Sec. 6.2.2). For each generated difference mask, we then calculate the IoU with all real difference masks, selecting the highest value as the final IoU.

- DDC (Difference of Distance Curve). It evaluates aspects (3) and (4) by comparing the distance curves of the generated and real sequences. The distance curve of a sequence depicts its progression towards the target painting, plotting time (x-axis, minutes) against LPIPS distance (y-axis) between target and current images. We use Dynamic Time Warping (DTW) [188] to compute the difference between generated and real curves, accommodating time shifts.

- DTS (Difference of Time Spent). Evaluating aspect (4), DTS measures the temporal difference in minutes between the durations of the real and generated painting videos.

- FID (Fréchet Inception Distance) [101] compares frames in generated and GT videos for aspect (5).

To compute these metrics, for *SVD* and *Timecraft* (trained on our dataset), we set the time interval to 23.6 seconds, matching our training set's average. For the self-supervised *Paint Transformer*, we set it to 2.8 seconds, based on the average training video duration (561 seconds) divided by the number of frames (200). All metrics are calculated for each painting and averaged across the validation set to derive overall performance metrics.

6.3.5 Comparison with Baseline

First, we present a qualitative comparison with baselines, as illustrated in Fig. 6.8. *SVD* generates artifacts such as the red blocks in the sky regions of column 1. This issue occurs because the target image has a tree that obscures this area, and *SVD* fails to realistically render the obscured regions. It also results in unreasonable painting orders and often stalls during the painting process, leading to extended convergence times (columns 5 and

Figure 6.8: **Comparison with baselines.** Displayed frames are evenly sampled from GT (top row) and generated videos (other rows). The last GT frame (top-right) is used as input. *SVD* produces noticeable artifacts such as strange color blocks and truncated tree trunks, as highlighted by the red arrows in columns 1 and 3. Additionally, it produces unreasonable painting orders and tends to get stuck during the process (e.g., minimal change between columns 5 and 6). *Paint Transformer* displays a non-human-like painting order and can only produce a "stylized" version of the target painting due to the limitations of parameterized paintstrokes. *Timecraft* results in blurry outputs with noticeable artifacts. In contrast, our method better mimics the human-like painting process and achieves higher visual quality.

6). *Paint Transformer* demonstrates a non-human-like painting order in which strokes are added in parallel to different grid cells of a canvas. *Timecraft* produces visible artifacts in low-resolution videos, likely because its conditional variational autoencoder does not match the image generation quality of diffusion models. Besides, this pure pixel-based method also produces unreasonable painting orders. Our model outperforms these baselines. Finally, we present the quantitative comparisons in Table. 6.1. Our pipeline surpasses all tested baselines across all metrics. For additional comparisons and analysis, including the visualization

Method	LPIPS ↓	IoU ↑	DDC ↓	DTS ↓	FID ↓
Paint Transformer	0.643	0.104	94.61	6.057	337.3
Timecraft	0.602	0.251	153.2	9.964	289.8
SVD	0.500	0.197	135.5	8.577	168.3
Ours-TE-TG-MG	0.468	0.128	88.13	6.204	203.3
Ours-TG-MG	0.447	0.139	62.79	4.913	187.4
Ours-TE	0.413	0.375	58.81	4.153	167.5
Ours-MG	0.435	0.175	61.09	3.972	182.5
Ours-TG	0.399	0.400	39.41	2.120	161.1
Ours-RN	0.416	0.396	46.71	1.542	174.2
Ours-CE	0.371	0.402	34.16	1.346	158.4
Ours 10	0.369	0.349	35.27	1.693	158.7
Ours 30	0.387	0.353	36.26	1.933	151.7
Ours	**0.364**	**0.418**	**32.66**	**1.273**	**150.6**

Table 6.1: **Comparison with baselines and our ablation variants.** Our full model (with a time interval of 20) outperforms all of them.

of distance curves and the distribution of their slopes, please refer to the supplementary.

6.3.6 Human Study

We conducted a human study on the generated painting processes of 8 in-the-wild paintings. The 33 participants were first presented with 3 examples of the real painting process as reference. Then for each painting, the participants were presented with 4 randomly shuffled painting processes (1 for our method and 3 for baselines), and were asked to give a rating from 1 to 5 (higher means better) for each process. We normalized the rating of each example and user to remove the user bias. Our method achieves the highest average rating, surpassing *SVD* by 1.9 times, *Paint Transformer* by 2.1 times, *Timecraft* by 3.0 times.

These show that our method can produce a better painting process than baselines. Please see supplementary for more details.

6.3.7 Ablation Study

We perform two ablation studies. The first one evaluates the different components in the pipeline using 7 variants. (1) *Ours-TE-TG-MG*: full model excluding time embeddings, text, and mask generators. (2) *Ours-TG-MG*: full model without text and mask generators. (3) *Ours-TE*: full model without time embeddings, omitting the time interval in the pipeline. (4) *Ours-MG*: full model without the mask generator. (5) *Ours-TG*: full model without the text generator, omitting text in the pipeline. (6) *Ours-RN*: full model without ReferenceNet, where the target image is inputted to the feature encoder in a manner similar to the current image. (7) *Ours-CE*: full model without the next CLIP generator. Results shown in Table. 6.1 and Fig. 6.5 indicate that the full model surpasses all variants. In addition to the discussion in Sec. 6.2, we observe that: (1) *Ours* outperforms *Ours-TE*, especially on *DDC* and *DTS*, highlighting the importance of time embeddings. (2) *Ours-MG* exhibits poor performance in IoU, illustrating the effectiveness of mask guidance for focus regions. (3) *Ours* outperforms *Ours-RN*, showing the effectiveness of using ReferenceNet to inject target image features into the diffusion model. (4) *Ours* works better than *Ours-CE*, indicating that the next CLIP generator offers beneficial complementary guidance.

The second ablation study evaluates the effect of different time intervals during testing. We test two variants, *Ours 10* and *Ours 30*, with time intervals set to 10 and 30 seconds respectively. Table. 6.1 shows that the IoU for these variants is lower than *Ours*, which might be because the average time interval in the validation set is around 22 seconds, closer to *Ours* (20 seconds). This suggests that time embeddings effectively guide the mask generator to produce reasonable size of region masks. Additionally, Fig. 6.9 visualizes the impact of different time intervals on a single canvas update. A larger time interval leads to a larger region mask and a more substantial update on the canvas, demonstrating how time intervals can be used to regulate the number of frames required to complete a painting. Different time intervals can also be used at various stages of the painting process based on

(a) Current Image (b) 10 seconds (c) 20 seconds (d) 30 seconds (e) Target Image

Figure 6.9: **Ablation of time interval.** (a) and (e) show the current and target image, respectively. (b) to (d) show the predicted next images and their masks (inset) given different time intervals. As the time interval increases, the size of the predicted mask expands, leading to more extensive updates on the canvas. Image courtesy Cleveland Museum of Art.

user needs.

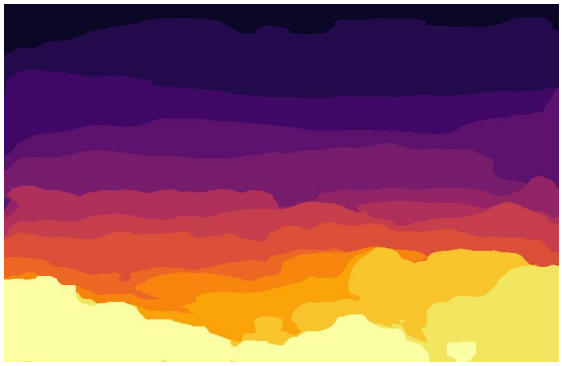
6.3.8 Analysis of the Text and Mask Generators

First, we compare the entire sequences of our generated text instructions with the sequences of GT instructions in two ways: (a) without considering order, 91% of our instructions appear in the GT instructions, and (b) taking order into account by computing the longest common subsequence (LCS) between GT and our instructions, 78% of our instructions are included in the LCS. These show that our text generator provides reasonable instructions for the painting order.

Second, instead of evaluating entire text instruction sequences, we evaluate a single canvas update with GT current image and time interval as input. This enables the use of GT text and masks for each update during evaluation. Our text generator produces text instructions that align with GT text 72% of the time, outperforming the pretrained LLaVA model used to initialize it (31%). When providing the predicted text as input to the mask generator, the predicted mask's IoU is 0.64, slightly lower than using GT text (0.70). These demonstrate the text generator's effectiveness.

Finally, we also evaluate our mask generator based on a single canvas update. Generating masks by a pretrained segmentation method [166] using GT text yields an IoU of 0.42, lower

(a) Input



(b) Time Map

Figure 6.10: **Emergent property**. The time map (b), derived from the difference masks, is similar to the (inverse) depth map of the target painting (a). This highlights our method's effectiveness in capturing the typical painting principle of layering from back to front, as learned from the training videos. Image courtesy Simon Berger.

than our mask generator's 0.70. Removing the text condition in our mask generator yields a suboptimal IoU of 0.58. These results demonstrate the effectiveness of our mask generator design.

6.4 Discussions

6.4.1 Emergent Property

Our method showcases an emergent property related to the painting principle, as depicted in Fig. 6.10. The time map (b) is derived from difference masks $\hat{M}_t$ between consecutive keyframes in the generated videos. Each mask is weighted by its sequence position t and merged into a single image, with overlapping areas selecting the highest value to form the time map. This map is similar to a depth map, indicating that the back-to-front painting principle – reflective of the artists' styles in our training dataset – has been effectively captured by our method.

Figure 6.11: **Failure cases.** Trained only on landscapes, our method produces unnatural results on portraits. Image courtesy the MET.

6.4.2 Limitations and Future Work

Our method has several limitations. First, it is trained on landscape paintings and struggles to generalize to other types of paintings like portraits (Fig. 6.11). Future work will involve extending its applicability to different genres. Second, our method currently exhibits limited painting styles due to the training data. Training on a larger and more diverse dataset could help the model learn various painting styles. Third, our method fails when encountering paintings with large objects (*e.g.*, colosseum) that are uncommon in the landscape. In such cases, the model can still paint the objects with the help of predicted CLIP embeddings, but in an incorrect painting order. Incorporating a semantic map could potentially address this issue.

SUPPLEMENTARY MATERIAL

6.5 Implementation Details

In this section, we present the implementation details.

6.5.1 One-Step Canvas Rendering Approach

Please refer to Sec. 6.5.3 for implementation details.

6.5.2 Training: Instruction Generation

For text instruction generator g_{text}, we set the question prompt p to: "There are two images side by side. The left image is an intermediate stage in a painting process of the right image. Please tell me what content should be painted next? The answer should be less than 2 words."

We obtain the ground-truth text instructions p_t with the assistance of the pretrained LLaVA 1.5 model "LLaVA-v1.5-7B[2]". Specifically, we horizontally concatenate the ground-truth current image I_{t-1} with the ground-truth next image I_t. We then input this concatenated image alongside the modified question prompt p', which reads: "There are two images side by side. The right image is the next step of the left image in the painting process of a painting. Please tell me what is added to right image? The answer should be less than 2 words." The LLaVA model then outputs the text instructions p_t, where we manually correct any inaccuracies in p_t.

For fine-tuning g_{text}, we employ LoRA and train for 10 epochs, using a learning rate of 1e-4 and a batch size of 16. The fine-tuning process takes approximately 5 hours on a single NVIDIA A100 GPU.

For the mask generator g_{mask}, we implement this as the UNet architecture proposed by Stable Diffusion [217]. The input layer of the UNet is modified to accept a 9-channel input,

[2]https://huggingface.co/liuhaotian/llava-v1.5-7b

tailored for the spatial input components $[E_I(I_T), E_I(I_{t-1}), M_d]$. For the time encoder g_t, we implement 3 fully connected layers that progressively increase the input feature dimension from 21 to 256, 512, and finally 768. A ReLU activation function follows each fully connected layer, with the exception of the last one to allow for linear output transformation. We train g_{mask} and g_t for 80k steps with a learning rate of 1e-5 and a batch size of 1. The training process takes approximately 13 hours on a single NVIDIA A100 GPU.

6.5.3 Training: Canvas Rendering

The feature extractor g_f takes as input the ground-truth current image I_{t-1} and region mask M_t, outputs an encoded feature of them. This feature extractor is implemented as a shallow network containing 9 convolutional layers, which scales the input spatial resolution by 8. The next CLIP generator, denoted as g_c, accepts the CLIP embeddings of both the ground-truth current image, $CLIP(I_{t-1})$, and the target image, $CLIP(I_T)$, as inputs. It then outputs a prediction for the CLIP embedding of the next image. Within g_c, the embeddings $CLIP(I_{t-1})$ and $CLIP(I_T)$ are concatenated along the feature dimension. This concatenated vector is subsequently processed through a three-layer multi-layer perceptron (MLP). The MLP's layers map the features from dimensions of 1536 to 768, 384, and back to 768 respectively. The ReLU activation function is employed at each layer except the final one, where no activation is applied. The time encoder g_t consists of 3 fully connected layers, same as that introduced in Sec.6.5.2. For more details on the implementation of ReferenceNet g_r, please refer to [107].

We initialize g_u using RealisticVision V5.1 [228]. The models within canvas rendering, namely g_u, g_r, g_f, g_t, and g_c, are jointly trained using a learning rate of 1×10^{-5} and a batch size of 1. Each conditional signal is dropped (set to zero) 10 percent of the time, in accordance with the classifier-free guidance method [104]. This enables us to control the strength of each conditional signal at the test time inference. We train the models in 200k steps, taking around 34 hours on a single NVIDIA A100 GPU.

For the one-stage approach outlined in Sec. 3.1 of the main paper, we use the same training strategy but exclude text and mask instructions.

6.5.4 Test-Time Generation

At a specific step $t-1$, we render the subsequent image $\hat{I}_t$ using the trained pipeline. The denoising process of the diffusion renderer employs a scheduler based on ancestral sampling, specifically utilizing the Euler method steps [127]. The denoising timestep S is set to 25. For classifier-free guidance, we assign guidance scales of 5 for text, mask, and time interval, and a scale of 2 for the next CLIP embeddings. Each update in this process takes approximately 4 seconds on a single NVIDIA A100 GPU. The generation process is halted if the perceptual distances between $\hat{I}_{t-2}$ and $\hat{I}_{t-1}$, and between $\hat{I}_{t-1}$ and $\hat{I}_t$, are both less than 1×10^{-3}.

6.5.5 Baselines Details

For *Timecraft*, we train the model on our dataset using the default settings provided in the official code. The training consists of two stages: pairwise optimization and sequence optimization. In the first stage, we train the model for 500K steps, which takes approximately 4 hours on 2 TITAN XP GPUs. In the second stage, we train the model for 78K steps, which takes around 25 hours on 2 TITAN XP GPUs. We observed that training with more steps will degrade the model performance.

For *Stable Video Diffusion (SVD)*, we fine-tune a 14-frame model on our dataset using LoRA [105]. During fine-tuning, we sample one frame from our training sequences as input and use its previous 13 frames as ground truth, padding with white images when necessary. The target image is used as the input frame 40% of the time, while other images are randomly selected otherwise. We fine-tune the model for 2K steps, which takes around 1.5 hours on 4 NVIDIA A100 GPUs. Fine-tuning for 2K steps yields the best performance; more steps cause the model to produce painting videos that get stuck, while fewer steps result in underfitting, causing the camera viewpoint to shift.

For *Paint Transformer*, we use the pretrained model provided by the authors for our comparisons. This model generates 200 frames given an input painting. We additionally evaluate two stroke-based rendering baselines [344, 110] and an amodal segmentation baseline [192] in Sec. 6.6, please refer to Sec. 6.6.2 for their implementation details.

6.6 Experiments

6.6.1 More Results

In Fig. 6.14, we show more results of our method. As discussed in the main paper, our method can handle paintings with different styles and generate human-like painting process in terms of painting order, focal region and layering techniques.

6.6.2 More Baselines

We compare our method with two additional stroke-based rendering baselines and an amodal segmentation baseline.

Stylized Neural Painting [344] employs an optimization-based approach featuring a novel neural renderer that mimics vector renderer behavior. Here, the stroke prediction is framed as a parameter search process aiming to maximize the similarity between the input image and the rendered output. We utilize the pretrained network "the oil-paint brush" provided by the authors for comparison. This method generates 499 frames given a target painting.

Compositional Neural Painter [110] utilizes a phased RL strategy for predicting paint regions and a painter network to determine stroke parameters. A neural stroke renderer is then trained to apply the strokes onto the canvas based on the predicted stroke parameters. We use the authors' pretrained networks for comparison. This method generates 50 frames from a target painting.

pix2gestalt [192] completes a partially visible object in the image given the partial segment mask of the object. We adapt it to out task as follows: (1) segment the target image using [166], (2) complete each segment using the pretrained model of *pix2gestalt*, (3) place completed segments on the canvas by depth (farther first). The depth of each segment is determined by the average depth of its pixels, where the depth is estimated using a pretrained depth estimation model [300]. Please note that we define the painting order heuristically, as the baseline does not support learning this order.

Similar to the strategy used for *Paint Transformer* in the main paper, we set the time intervals for these three baselines based on the average training video duration (561 seconds) divided by the number of frames. This results in time intervals of 1.12 seconds for *Stylized*

Neural Painting and 11.22 seconds for *Compositional Neural Painter*. The time interval of *pix2gestalt* varies for different target images, depending on the number of detected segments in the target image.

6.6.3 More Metrics

We also evaluate the quality of the generated videos using the Fréchet Video Distance (FVD) [79]. While FVD might not be ideally suited for assessing time-lapse painting process videos, we include it for the sake of comprehensiveness.

6.6.4 Baseline Comparison

We present more comparisons with baseline methods on both full and cropped paintings.

Full Paintings

We provide qualitative comparisons with all baselines for full paintings in Fig. 6.17 and Fig. 6.18. The three stroke-based rendering baselines – *Stylized Neural Painting*, *Compositional Neural Painter*, and *Paint Transformer* – apply brushstrokes in a non-human-like manner, as they are not trained on actual painting videos. Furthermore, due to the limitations of parameterized brushstroke constraints, they produce only a "stylized" version of the target painting. *pix2gestalt*'s results are non-human-like and exhibited visual artifacts. This is due to inaccurate predefined painting order, imperfect segmentation, and unnecessary paints on the canvas to complete unoccluded segments. *SVD* often gets stuck in the painting process, evident in columns 1 to 2 of Fig. 6.17, and produces visual artifacts, such as the unreasonable colors in columns 1 and 2 of Fig. 6.17 and columns 1 to 5 of Fig. 6.18. Moreover, despite being trained on a real painting dataset, it still fails to mimic the human painting order reasonably. *Timecraft* generates only very low-resolution sequences and introduces noticeable visual artifacts. In contrast, our method significantly outperforms all baselines in mimicking human-like painting sequences, focusing on focal areas, employing layering techniques, and achieving good video quality.

Table. 6.2 presents the quantitative comparisons across all baselines and metrics. Our

Evaluation on Full Paintings						
Method	LPIPS ↓	IoU ↑	DDC ↓	DTS ↓	FID ↓	FVD ↓
Stylized Neural Painting	0.669	0.031	122.2	8.782	358.3	1457
Compositional Neural Painter	0.680	0.049	91.93	7.507	374.8	1505
pix2gestalt	0.609	0.214	126.3	9.089	341.9	1576
Paint Transformer	0.643	0.104	94.61	6.057	337.3	1616
Timecraft	0.602	0.251	153.2	9.964	289.8	1582
SVD	0.500	0.197	135.5	8.577	168.3	1594
Ours-TE-TG-MG	0.468	0.128	88.13	6.204	203.3	1591
Ours-TG-MG	0.447	0.139	62.79	4.913	187.4	1468
Ours-TE	0.413	0.375	58.81	4.153	167.5	1319
Ours-MG	0.435	0.175	61.09	3.972	182.5	1471
Ours-TG	0.399	0.400	39.41	2.120	161.1	1418
Ours-RN	0.416	0.396	46.71	1.542	174.2	1464
Ours-CE	0.371	0.402	34.16	1.346	158.4	1326
Ours 10	0.369	0.349	35.27	1.693	158.7	1347
Ours 30	0.387	0.353	36.26	1.933	151.7	1279
Ours	**0.364**	**0.418**	**32.66**	**1.273**	**150.6**	**1273**
Evaluation on Cropped Paintings						
Timecraft	0.647	0.165	166.29	6.743	363.0	1627
Ours	**0.452**	**0.296**	**56.62**	**2.545**	**197.2**	**1034**

Table 6.2: **Comparison with baselines and our ablation variants on the full and cropped paintings.** Our full model (with a time interval of 20) outperforms all of them.

method outperforms all baselines in every evaluated metric.

Cropped Paintings

We compare with *Timecraft* by randomly selecting 3 low-resolution crops from every downsampled full painting in the validation set, following the cropping strategy presented in the paper of *Timecraft*. Fig. 6.12 provides a qualitative comparison of cropped paintings with *Timecraft*, where our approach yields a more authentic painting process in terms of painting order, focus regions, and overall video quality. The quantitative results in Table. 6.2 further demonstrate that our method outperforms *Timecraft* across all metrics.

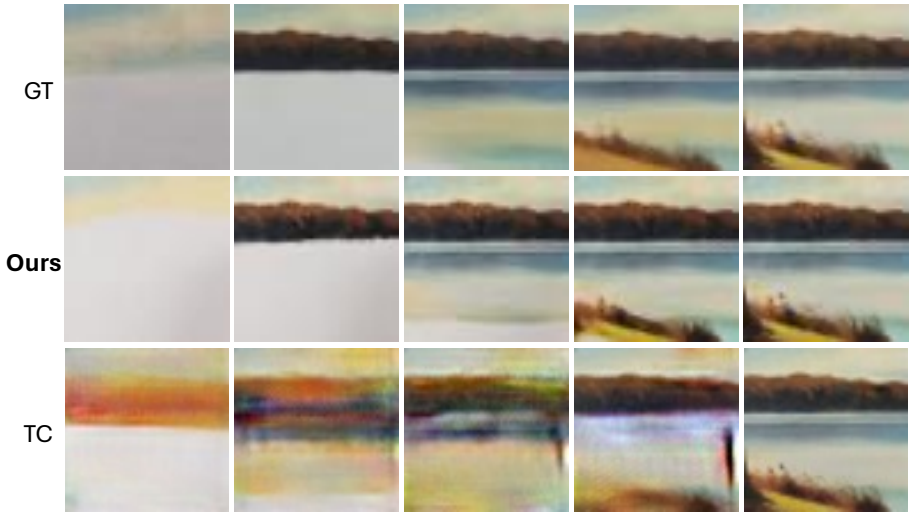
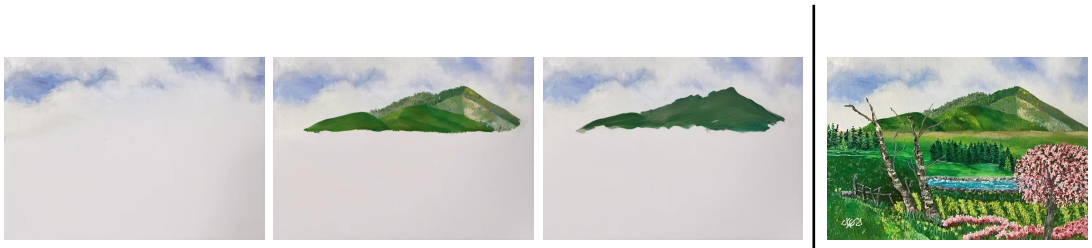


Figure 6.12: **Qualitative comparison on low-resolution painting crops.** We compare with *Timecraft* (TC) on low-resolution painting crops. *Timecraft* produces artifacts and fails to produce a human-like painting process. In contrast, our method delivers a more realistic painting video with better quality.

6.6.5 Ablation Study

In Fig. 6.15, we present the ablation studies for variants omitting different conditional signals. Both one-step variants, *Ours-TE-TG-MG* and *Ours-TG-MG*, yield unsatisfactory results, underscoring the significance of the two-stage design. *Ours-MG* heavily depends on text instructions and tends to complete an entire semantic class with each update, unexpectedly accelerating the generation process excessively. *Ours-TG* struggles to comprehend the semantic contents of the target paintings, consequently painting the grass and flowers simultaneously without employing layering techniques. We further present qualitative results of the variants without considering predicted CLIP embedding (*Ours-CE*) in Fig. 6.13. It completes details of the mountain at an early stage, which fails to mimic the painting style of artists in our training set. In contrast, *Ours* delivers the most human-like painting process, characterized by a logical painting order, targeted focal regions, and proper use of layering techniques.



(a) Current Image (b) Ours-CE (c) Ours (d) Target Image

Figure 6.13: **Ablation of the predicted CLIP embeddings.** The current and target images are shown in (a) and (d), respectively. (b) and (c) represent the outputs generated by different models at the same timestep, utilizing the same time interval, current image, and target image. Without incorporating predicted CLIP embeddings (b), the model completes the mountain in full detail. The process involves frequent switching of color brushes, deviating from the painting style observed in the training set. Our full model (c) paints the base layer of the mountain first and leaves the details for latter stages, which follows the artistic techniques presented by the artists in our training data. Please see Fig. 6 in the main paper for more keyframes generated by the full model. Image courtesy Catherine Kay Greenup.

6.6.6 Human Study

As described in the main paper, we normalized the ratings to remove user bias. Specifically, we divided each participant's rating for a specific painting sequence by the sum of their ratings for all 4 painting sequences (of the same target painting). We then averaged these normalized ratings across all paintings and participants for each method.

Our method achieved the highest average normalized rating, surpassing *SVD* by 1.9 times, *Paint Transformer* by 2.1 times, and *Timecraft* by 3.0 times (reported in the main paper). Without normalization, our method received the highest rating at 4.21, compared to *SVD* (2.96), *Paint Transformer* (2.11), and *Timecraft* (1.52).

6.6.7 Influence of Random Seeds

In Fig. 6.16, we illustrate the outcomes of utilizing different random seeds for the diffusion model in our methods. Although different seeds are used, the generated painting processes generally follow a similar order, with minor variations in the sequence of painting foreground objects. These variations are reflective of those observed in the training data, demonstrating that our method can learn the general painting order from these slightly varied sequences and capture the variability. This adaptability allows for the use of different random seeds at inference time to achieve diverse results.

6.6.8 Error Accumulation

Our approach of training the diffusion models with GT inputs helps alleviate error accumulation. Specifically, training with GT texts and masks avoids errors propagated from the trained text and mask generators, achieving better LPIPS score of 0.364 compared to training with predicted texts and masks (0.438). Additionally, training with GT current frames enhances both stability and efficiency. Furthermore, during inference, using the target image as input helps the convergence of the painting process and further alleviates error accumulation.

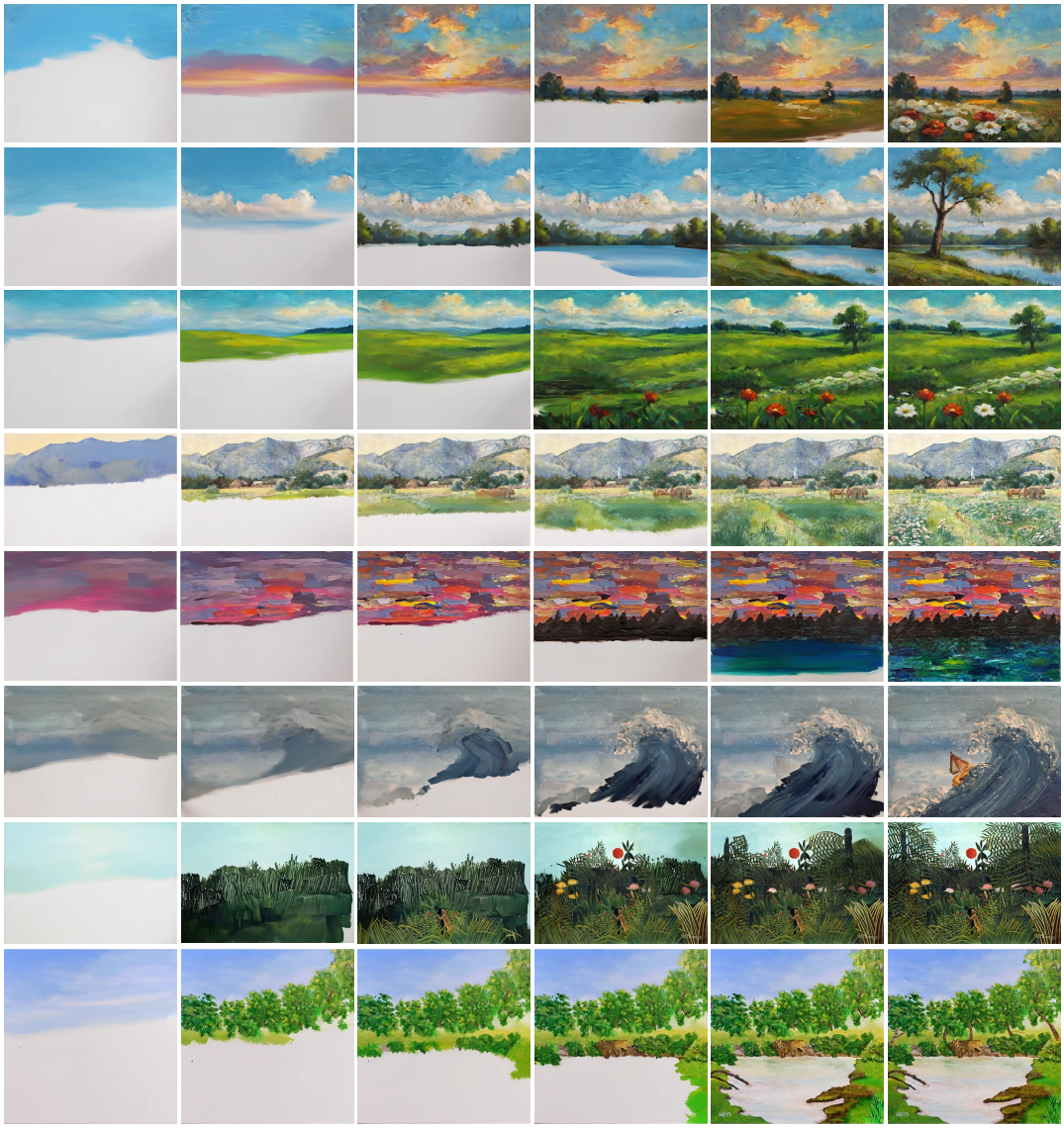


Figure 6.14: **More qualitative results on our method.** We show our results on in-the-wild paintings, where the left 5 columns are sampled frames from the generated painting process, and the rightmost column is the target image. Our method effectively handles paintings across various artistic styles, color themes, and aspect ratios. The generated sequences showcase human-like painting orders, maintain reasonable focal regions during different phases, and employ layering painting techniques. Image courtesy Julius Zorkoczy, Landscape with a Blooming Meadow, Slovenska narodna galeria, SNG, `https://www.webumenia.sk/dielo/SVK:SNG.K_1710`. Image courtesy Henri Rousseau, Virgin Forest with Sunset, Kunstmuseum Basel Museum. Image courtesy Catherine Kay Greenup. Images courtesy Michelle Shlizerman.

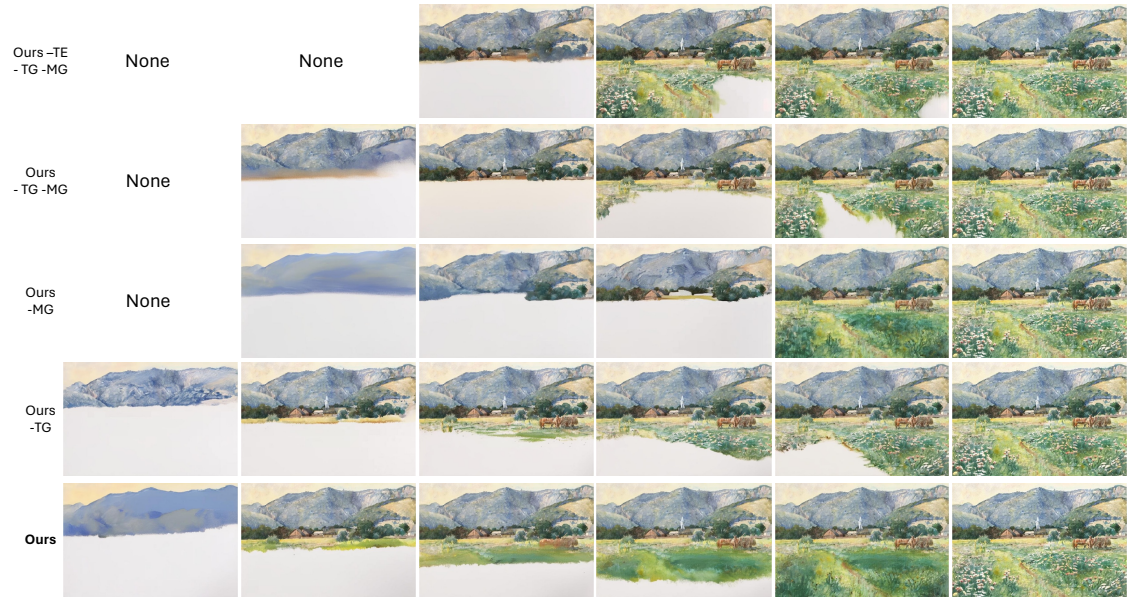


Figure 6.15: **Ablation study of different conditional signals.** The row with "None" means there are no enough keyframes for display. For instance, the first row has only four keyframes, leaving the first two columns "None". *Ours-TE-TG-MG* generates excessive content for each update, resulting in only four keyframes throughout the entire painting process. *Ours-TG-MG* slows generation slightly but still converges quickly. Besides, it also produces unreasonable rendering, as shown in column 2. *Ours-TG*, relying heavily on text instructions, tends to complete entire semantic classes in each update, promoting swift convergence. *Ours-MG* leverages region masks to moderate the speed of generation. Yet, in the absence of textual guidance, it struggles to adequately differentiate between semantic classes. For example, in columns 4 to 6, the flower and grass are painted simultaneously, rather than using layering techniques that complete the grass first and add the flower afterward. In contrast, *Ours* achieves a more logical and orderly painting process. Image courtesy Julius Zorkoczy, Landscape with a Blooming Meadow, Slovenska narodna galeria, SNG, `https://www.webumenia.sk/dielo/SVK:SNG.K_1710`.

Figure 6.16: **Comparison of using different random seeds for the diffusion model in our method.** Each row displays the results of a specific random seed. Despite using different random seeds, the painting orders are generally similar (i.e., back to front) with slight differences in painting foreground objects. For example, rows 1 and 2 tend to address the far ground and house in the final stages, whereas row 3 completes the far ground immediately after finishing the background nearby. Both styles of painting order are observed in the training set, and our method successfully captures these variations. Image courtesy National Gallery of Art.

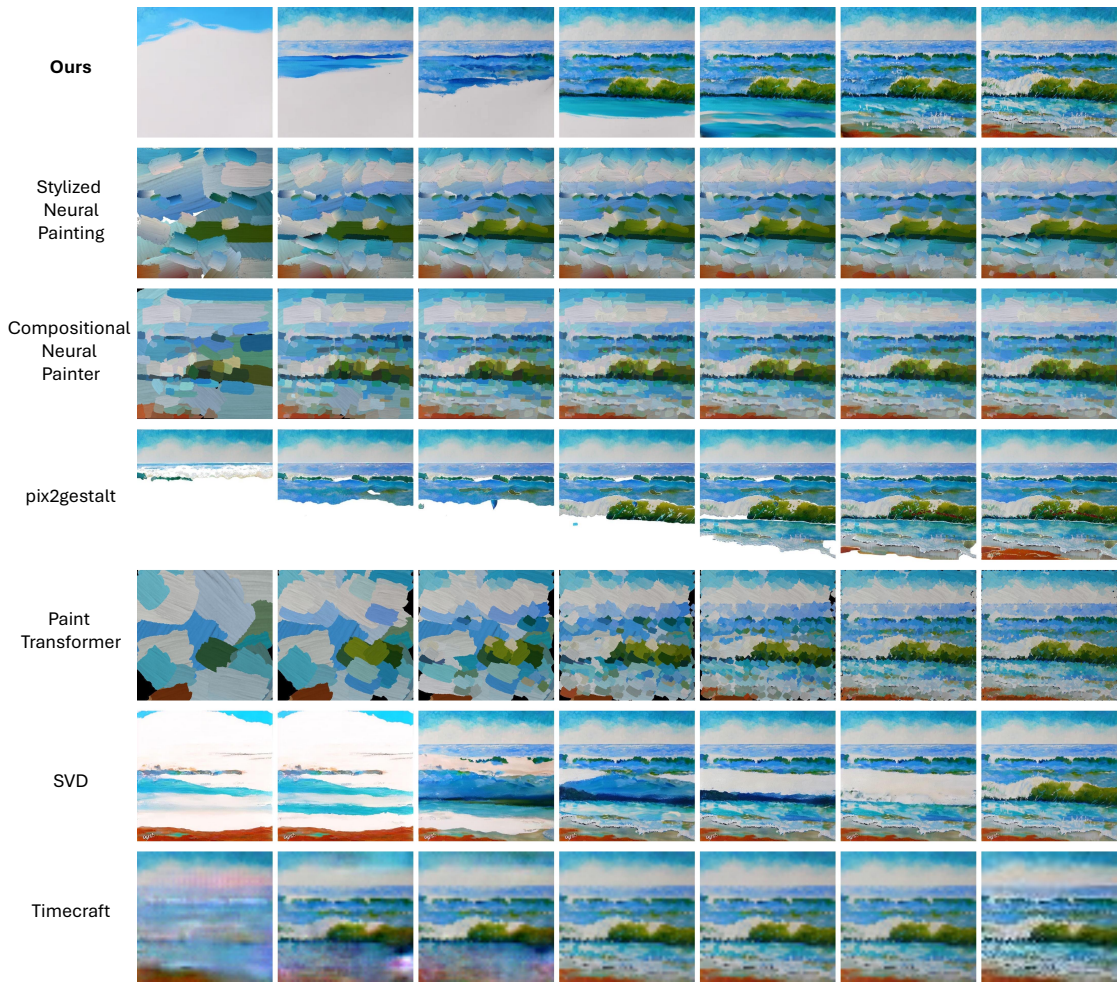


Figure 6.17: **Comparison with baselines on in-the-wild images.** Our method significantly outperforms these baselines in generating a more human-like painting video with better visual quality. Image courtesy Catherine Kay Greenup.

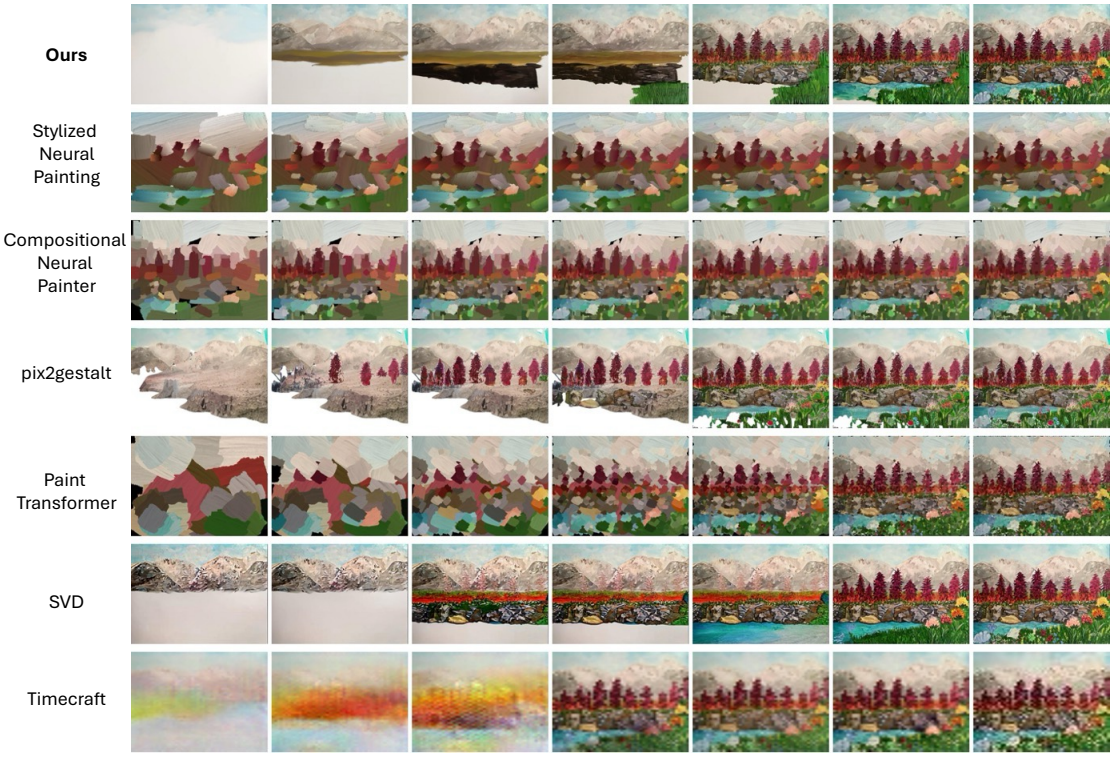


Figure 6.18: **Comparison with baselines on in-the-wild images.** Our method significantly outperforms these baselines in generating a more human-like painting video with better visual quality. Image courtesy Catherine Kay Greenup.

Chapter 7

ALIGNING VISUAL FOUNDATION ENCODERS TO TOKENIZERS FOR DIFFUSION MODELS

This chapter presents the research project *Aligning Visual Foundation Encoders to Tokenizers for Diffusion Models* [32], conducted with Sai Bi, Hao Tan, He Zhang, Tianyuan Zhang, Zhengqi Li, Yuanjun Xiong, Jianming Zhang, Kai Zhang. The subsequent analysis and comparisons to related studies in this chapter are based on the prevailing state of the art at the time of this work.

Diffusion models have recently emerged as the leading method for high-fidelity image generation. A crucial component of training image diffusion models is the *continuous* visual tokenizer, which defines the latent space where diffusion operates [216]. Training such a tokenizer involves two tasks: (1) the encoder must learn a diffusion-friendly latent space, often referred to as the *diffusability* of the latent space [240]; and (2) the decoder must learn to reconstruct the input signal. A common practice for training a continuous visual tokenizer is to adopt a variational autoencoder (VAE), optimized with reconstruction loss and a lightly weighted KL regularization term. Since the KL term typically has only a small weight, training is dominated by reconstruction loss, making the two tasks asymmetric: the decoder's reconstruction learning is direct and well-supervised, while the encoder's representation learning is indirect – the latent space is shaped largely as a byproduct of reconstruction and only weakly regularized by the KL prior. As a result, the latent space often develops an unpredictable structure dominated by low-level details, limiting its diffusability [306].

Recent work on tokenizers for image diffusion models has explored adding constraints such as semantic regularization (Fig. 7.1 left), which adds a loss term to encourage the latent space to align with representations from large pretrained encoders [306, 40]. These studies demonstrate that semantically grounded latents exhibit better diffusability, leading

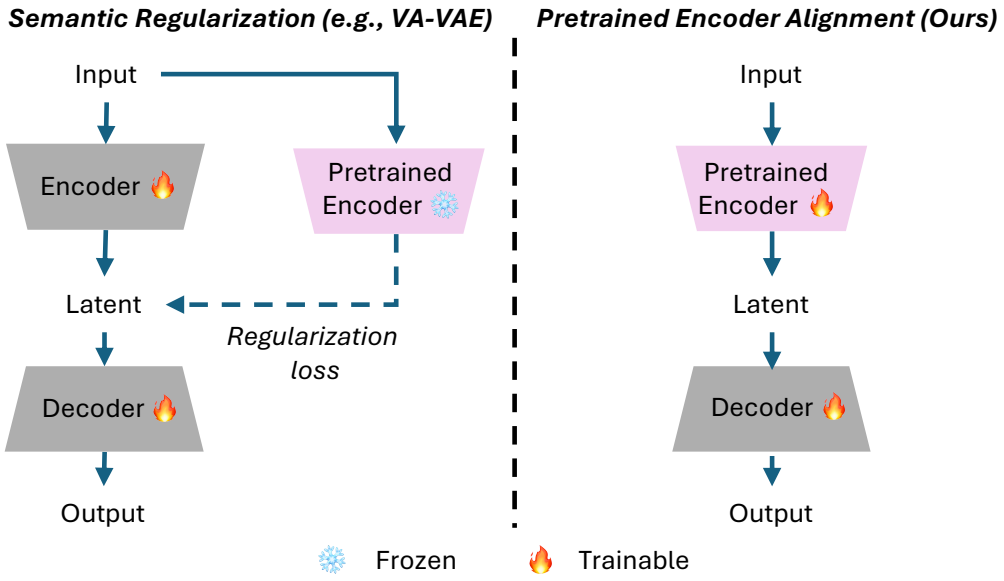


Figure 7.1: **Regularization vs. Alignment.**

to improved generation quality in diffusion models. However, these methods still fall short, as the encoder must still learn semantic structure *from scratch* via the regularization loss while simultaneously managing the competing reconstruction objective.

Our goal is to design an image tokenizer with stronger semantic grounding – hence better diffusability – with competitive reconstruction ability. Our intuition is that learning semantic is inherently more difficult than learning reconstruction. Thus, we take a different perspective: rather than forcing the encoder to learn semantics from scratch, we *align* a pretrained visual foundation encoder (*e.g.*, DINOv2 [191]) to a visual tokenizer (Fig. 7.1 right). Since the encoder already captures rich semantic structure, our alignment makes the first task – learning a diffusion-friendly latent space (*i.e.*, achieving strong diffusability) – much easier. Training can then focus on equipping the tokenizer with reconstruction ability, avoiding the difficulties of learning semantic from scratch.

We implement this idea through a three-stage alignment procedure. First, we freeze the pretrained encoder and train a lightweight adapter and decoder with reconstruction loss, establishing a semantically grounded latent space. Second, we jointly optimize all components

with an additional semantic preservation loss, enabling the encoder to capture fine-grained perceptual details essential for both reconstruction and generation, while maintaining its underlying semantic structure. Finally, we fine-tune only the decoder to enhance reconstruction fidelity while keeping the latent space fixed. This progressive alignment preserves a semantically rich, diffusion-friendly latent space that also retains the details necessary for accurate reconstruction and high-quality generation.

We demonstrate the effectiveness of our method on both the ImageNet 256×256 dataset [59] and the LAION dataset [225] by training diffusion models using our continuous tokenizers. On ImageNet, our semantic tokenizer accelerates the convergence of diffusion models and improves generation quality over previous methods, both with and without classifier-free guidance (CFG), as well as in unconditional generation settings. To further validate scalability, we train a 2B-parameter text-to-image diffusion model on LAION and show that it converges significantly faster than the FLUX VAE [144]. We conjecture that these benefits arise from a well-grounded semantic latent space, which provides a more structured representation.

In summary, we propose a new paradigm for training visual tokenizers by aligning pretrained visual encoders. Our approach is simple, scalable, and effective, and we believe it can open new directions in tokenizer design while inspiring future research in generative modeling.

7.1 Related Work

7.1.1 Foundation Encoder for Representation Learning

Large-scale foundation visual encoders enable the extraction of rich, transferable representations from diverse visual data. Models such as CLIP [206], SigLIP [319], SigLIP 2 [253], MAE [94], Perception Encoder [22], DINOv2 [191], and DINOv3 [237] demonstrate that pretraining on massive datasets enables encoders to capture meaningful visual features that generalize effectively across downstream visual understanding tasks. In this paper, we adopt DINOv2 as our visual foundation encoder by default, since we empirically find that it is more effective for diffusion modeling than alternatives such as SigLIP 2 and MAE.

7.1.2 Image Tokenizers for Generative Models

Image tokenizers are essential for scaling visual generation. They use an encoder–decoder framework to map images into compact latent spaces where generative models can operate more effectively and efficiently [216]. A common practice in training tokenizers is to rely on reconstruction losses such as L1 loss, perceptual loss, and adversarial loss [216, 140, 280, 283, 313, 240, 91, 236, 255]. Depending on the design, tokenizers are either continuous, where the latent distribution is regularized by a KL loss, or discrete, where quantization with a codebook is enforced via a VQ loss. While these methods achieve faithful reconstruction fidelity, the resulting latent space is often dominated by fine-grained details because they mainly rely on reconstruction loss, which can hinder generative performance.

Semantic Regularization. Recent methods introduce semantic regularization strategies to enrich the image tokenizers with higher-level semantics [290, 40, 306, 44, 145, 299, 49, 172]. VA-VAE [306] introduces a continuous tokenizer for diffusion models by regularizing its latent space to be close to a pretrained encoder, whereas GigaTok [290] proposes a discrete tokenizer for autoregressive models by imposing semantic constraints on decoder features. Instead of using semantic regularization, we align a pretrained encoder that is already capable of extracting high-level semantic representations, and show that this leads to a more diffusion-friendly latent space. We conduct extensive comparisons with VA-VAE, as both methods target diffusion models, and demonstrate that our alignment strategy outperforms semantic regularization.

Pretrained Encoders as Discrete Tokenizers. The use of pretrained encoders in tokenizers has also been studied, but primarily in the context of *discrete* tokenizers for *autoregressive models*. One line of work treats the pretrained encoder as a fixed feature extractor without fine-tuning it to encode perceptual details [333, 45, 288, 261]. Another line of work fine-tunes pretrained encoders with additional architectural designs, such as introducing extra encoders [205, 120] or decoders [88], or splitting encoder features into separate vocabularies [243]. A third direction leverages contrastive learning, fine-tuning the pretrained encoder with image–text supervision to capture semantic alignment [285, 174, 332, 157]. However, this strategy primarily assumes that the encoder is a language-aligned

model (*e.g.*, SigLIP 2).

In contrast to these works on *discrete* tokenizers, we focus on *continuous* tokenizers for diffusion models. Rather than introducing additional architectural design or requiring image–text supervision, we keep the simple architecture of autoencoder and directly align a pretrained encoder with a self-supervised semantic preservation loss. Our method can generalize to any visual encoders, offering a simple and scalable path toward semantically rich tokenizers for generative modeling.

Concurrent Work. [247] explores enabling pretrained CLIP with reconstruction ability. The key distinction is that we target diffusion-based generation, providing extensive experiments showing that aligning a pretrained encoder yields latent space with better diffusability than using semantic regularization. In contrast, their work focuses on unified multimodal understanding, generation, and editing within a hybrid architecture that combines multimodal large language models (MLLMs) with diffusion. Another difference is that we study different foundation visual encoders and identify DINOv2 as particularly well suited for latent diffusion models, whereas their focus remains on CLIP in the context of unified modeling.

7.2 Method

We aim to build a semantic, diffusion-friendly visual tokenizer by aligning a pretrained visual encoder. We begin with a review of latent diffusion models, followed by an introduction of our method.

7.2.1 Background of Latent Diffusion Models in Image Generation

Latent diffusion models (LDMs) [216] operate by learning a denoising process in the compressed latent space produced by a continuous visual tokenizer. The tokenizer consists of an encoder E and a decoder D. The encoder maps an input image $x \in \mathbb{R}^{H\times W\times 3}$ to a latent code $z_0 = E(x) \in \mathbb{R}^{\frac{H}{f}\times\frac{W}{f}\times d}$, where H is image height, W is image width, f is the downsampling factor and d is the channel dimension. The decoder reconstructs $\hat{x} = D(z_0)$. The tokenizer is trained with a reconstruction objective combining pixel-level L1, perceptual,

and adversarial losses:

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\ell_1}(x, \hat{x}) + w_p \mathcal{L}_{\text{perceptual}}(x, \hat{x}) + w_g \mathcal{L}_{\text{GAN}}(x, \hat{x}), \tag{7.1}$$

where w_p and w_g are the weights for the perceptual and adversarial loss, respectively. In addition, a KL regularization term is often included alongside the reconstruction loss to encourage a well-structured latent space. After training the tokenizer, a diffusion model learns to reverse a forward noising process in this learned latent space. A common formulation is flow matching, where we define an interpolating path:

$$z_t = (1-t)\, z_0 + t\, z_1, \quad z_1 \sim \mathcal{N}(0, I),\ t \in [0, 1], \tag{7.2}$$

where t is the diffusion timestep. The velocity field is given by $u_t = \frac{d}{dt} z_t = z_1 - z_0$. The diffusion model v_θ is trained to predict this velocity, with the loss $\mathcal{L}_{\text{FM}} = \mathbb{E}_{z_0, z_1, t}\left[\|v_\theta(z_t, t) - u_t\|_2^2\right]$.

7.2.2 Aligning Pretrained Encoder to Visual Tokenizer

Our method leverages a pretrained visual encoder, which offers rich semantic representations, and progressively adapts it into a diffusion-friendly visual tokenizer. This is implemented through three alignment stages, as illustrated in Fig. 7.2. First, we align the encoder's semantic space to a generative latent space by training a lightweight adapter and decoder (*Latent Alignment*). Second, we jointly optimize all components to enhance generation and reconstruction fidelity while preserving semantic structure (*Perceptual Alignment*). Finally, we fine-tune the decoder to further improve reconstruction quality while keeping the latent space untouched (*Decoder Refinement*).

Stage 1: Latent Alignment. As the first stage, the goal is to adapt the pretrained encoder to create a latent space suitable for generation (Fig. 7.2 top). This requires that the latent representations contain semantic information and can be mapped back to the image domain for reconstruction.

Given an input image x, we extract its embedding using a frozen pretrained encoder E_p. Since embeddings from encoders trained for representation learning tasks are typically very high-dimensional (*e.g.*, 1024 channels for *DINOv2-L/14*), they are not directly suitable for diffusion models, which usually operate in lower dimensions (*e.g.*, 32 or 64). To address

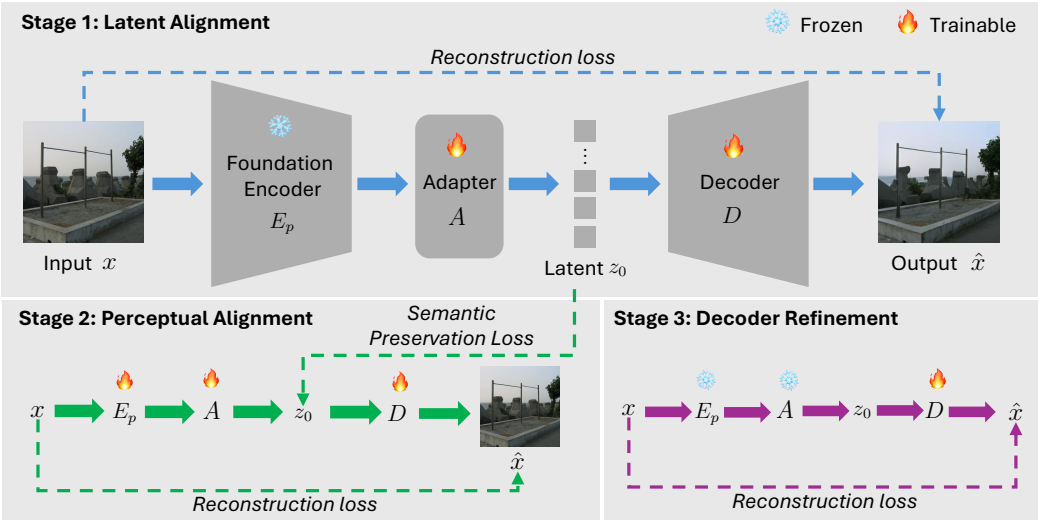


Figure 7.2: **Method Overview.** *Stage 1: Latent Alignment (top).* The pretrained encoder is kept frozen, while the adapter and decoder are trained with reconstruction loss to align its output into a semantically grounded latent space for generation. *Stage 2: Perceptual Alignment (bottom left).* All components are optimized jointly to enrich the latent space with low-level details, while a semantic preservation loss ensures that it retains high-level semantics. *Stage 3: Decoder Refinement (bottom right).* Only the decoder is fine-tuned with reconstruction loss to enhance reconstruction fidelity.

this, we introduce an adapter A that projects the high-dimensional features into a compact latent code of dimension d (32 by default):

$$z_0 = A(E_p(x)). \tag{7.3}$$

To complete the tokenizer, a decoder D is then introduced to reconstruct the input image from z_0. During this stage, only the adapter A and decoder D are trained with the reconstruction loss in Eq. 7.1, while the pretrained encoder E_p remains frozen to ensure a semantically-rich latent space. We omit the KL term because we found that it does not provide benefits and only imposes unnecessary distributional constraints, which can distort the encoder's semantics in the latent space.

Although this stage yields a semantically grounded latent space, it does not achieve

high-fidelity reconstruction (see a noticeable color shift in top right image of Fig. 7.2 and the orange point in Fig. 7.3 left), as the frozen encoder cannot capture the fine-grained perceptual details required for precise image reconstruction and high-quality generation.

Stage 2: Perceptual Alignment. The goal of this stage is to adapt the pretrained encoder so that it can capture fine-grained, low-level image details while still preserving semantic features, as shown in Fig. 7.2 (bottom left). Starting from the checkpoints of the previous stage, we jointly optimize E_p, A, and D using the same reconstruction loss as in Eq. 7.1. This encourages E_p to encode richer details, thereby improving reconstruction quality, as indicated by the green curve in Fig. 7.3 (left). However, while reconstruction improves rapidly, this optimization simultaneously causes the latent space to catastrophically lose its semantic structure, as reflected by the sharp drop in linear probing accuracy (green curve in Fig. 7.3 right). To address this issue, we introduce a simple yet effective semantic preservation loss, which constrains the latent codes produced in the current stage to remain close to those obtained in the previous stage. Formally, we define this loss as an L2 loss:

$$\mathcal{L}_{\text{sp}} = L_{\ell_2}(z_0^*, z_0), \tag{7.4}$$

where z_0^* and z_0 are the latent codes produced by E_p and A in the current stage (being updated) and the previous stage (kept frozen), respectively. An alternative is to apply this loss directly on the output of E_p, providing more flexibility by leaving the adapter A unregularized. However, we found this variant degrades generation quality (see ablation study). The overall loss for this stage is:

$$\mathcal{L}_{\text{pa}} = \mathcal{L}_{\text{rec}} + w_{sp}\mathcal{L}_{\text{sp}}, \tag{7.5}$$

where $w_{sp} = 1$ balances the semantic preservation loss. As shown in Fig. 7.3 (blue curve), this strategy maintains a semantically rich latent space while achieving comparable reconstruction performance.

Stage 3: Decoder Refinement. The previous two stages already align the pre-trained encoder into a visual tokenizer that significantly boosts generation performance. The goal of this stage is to further refine the decoder to improve reconstruction quality, as shown in Fig. 7.2 (bottom right). The key motivation is that, although the latent space is semantically

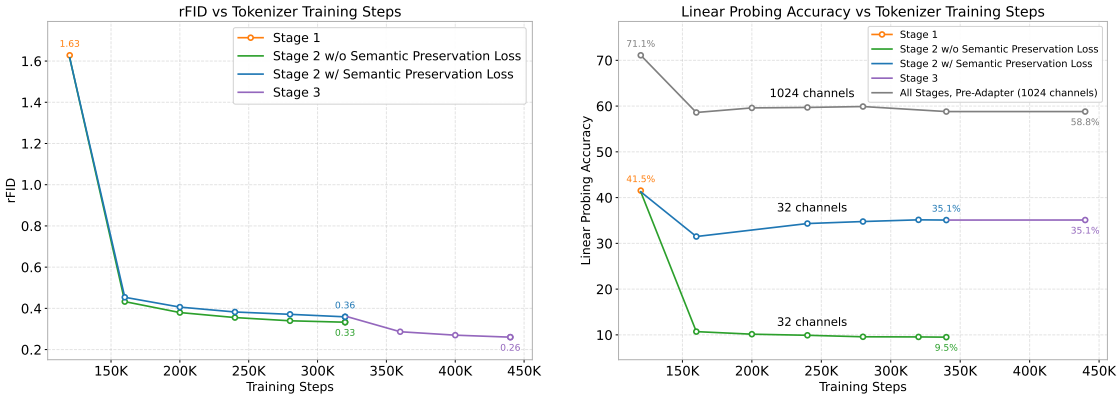


Figure 7.3: **Reconstruction vs. Semantic Preservation in Tokenizer Training.** *Left*: reconstruction FID (rFID) across training steps. *Right*: linear probing accuracy across training steps. Linear probing accuracy is evaluated on the latent code (32 channels), except for *All Stages, Pre-Adapter (1024 channels)*, which is reported only for reference. In this case, linear probing accuracy is measured on the feature before the adapter, using the same checkpoint as *Stage 2 w/ Semantic Preservation Loss*.

aligned, the decoder may still underfit because the latent space kept changing throughout the previous two stages. Fine-tuning the decoder alone allows it to better exploit the existing latent representation without disturbing its semantic structure. Specifically, we continue training from the second stage but only update the decoder. This preserves the learned latent space and can even be applied after training the downstream generative model. As shown in Fig. 7.3 left (purple curve), this stage improves reconstruction performance.

7.3 Experiments

We evaluate on two datasets: ImageNet 256×256 [59] and a subset of LAION-2B [225]. Most ablation studies and baseline comparisons are conducted on ImageNet (Sec. 7.3.1–7.3.3), followed by larger scale text-to-image experiments on LAION (Sec. 7.3.4). For ImageNet, we set the downsampling ratio to $f = 16$, the latent channel dimension to $d = 32$, the semantic preservation loss weight to $w_{sp} = 1$, and the sampling step to 30, unless otherwise specified. Same as VA-VAE [306], we use LightningDiT (∼673M parameters) as generative

Table 7.1: **Ablation study.** Evaluated on ImageNet 256×256 at 80K training steps with 30 sampling steps, using the CFG scale that yields the lowest generation FID (gFID). Our full three-stage model achieves the best balance between reconstruction and generation quality.

Configuration	rFID↓	PSNR↑	LPIPS↓	L. P. Acc.↑	gFID↓	IS↑	Prec↑	Recall↑
Semantic Preservation Loss								
Weight = 0	0.33	**26.29**	**0.110**	9.50%	3.05	215.1	0.832	0.550
Weight = 5	0.49	23.70	0.163	40.55%	2.48	244.4	0.836	0.566
Weight = 10	0.59	22.63	0.189	40.89%	2.59	243.5	0.839	0.562
Applied Pre-Adapter	0.34	25.78	0.118	15.61%	2.83	226.9	0.835	0.553
Cosine Loss	0.37	25.23	0.129	37.99%	2.23	248.6	0.819	0.585
Training Strategy for Stage 2								
LoRA Fine-Tuning	1.35	26.18	0.121	18.56%	2.97	243.3	0.815	0.557
w/o EMA	0.33	25.70	0.122	35.04%	2.24	246.0	0.809	0.587
High-Level Design								
Larger Decoder	0.36	26.27	0.121	27.24%	2.52	248.3	0.808	0.571
Stage 1 only	1.63	17.34	0.323	**41.53%**	3.00	246.4	**0.843**	0.529
Stage 1 + Stage 2	0.36	25.62	0.121	35.09%	2.19	248.6	0.811	0.591
Full Model	**0.26**	25.83	0.117	35.09%	**2.17**	**249.3**	0.811	**0.599**

model for ImageNet experiments.

Metrics. For ImageNet, we use reconstruction FID (rFID), PSNR, and LPIPS to evaluate reconstruction quality; linear probing accuracy to assess the semantic structure of latent space; and generation FID (gFID), Inception Score (IS), Precision (Prec), and Recall to measure generative performance.

7.3.1 Ablation Study

We test different variants of our design and summarize them in Tab. 7.1. *All variants we test do not employ the third stage, thus we mainly compare them to Stage 1 + Stage 2 for analysis.*

Semantic Preservation Loss. We first analyze the effect of varying the weight of the

semantic preservation loss in Eq. 7.5. Without this loss (weight = 0), the model achieves slightly better reconstruction quality, but the linear probing accuracy and generative metrics degrade, indicating that the latent space collapses toward low-level details at the expense of semantic structure. Increasing the weight (5 or 10) enforces stronger preservation, which improves generative performance over the one with weight = 0, but comes with a notable drop in reconstruction fidelity. These results reveal a clear trade-off: too little preservation leads to semantic drift, whereas too much preservation overly constrains the encoder and harms pixel-level fidelity. In the *Stage 1 + Stage 2* variant, a moderate weight of 1 achieves the best balance between generation and reconstruction.

We also test a variant *Applied Pre-Adapter* where the semantic preservation loss is applied directly on the outputs of the pretrained encoder (before adapter). This approach partially preserves semantics, and the generation quality lags behind *Stage 1 + Stage 2* variant. This may be because leaving the adapter unregularized gives it too much flexibility, leading to a loss of semantic structure.

Training Strategy. We compare different optimization strategies for the perceptual alignment stage against the *Stage 1 + Stage 2* variant. *LoRA Fine-Tuning* performs noticeably worse, showing that restricting updates to low-rank adapters is insufficient for balancing semantics and reconstruction. Removing EMA [128] slightly decreases generation performance. The *Stage 1 + Stage 2* variant uses EMA to stabilize training and thereby produce a more stable latent space.

High-Level Design. We also evaluate alternative architectural and training-stage designs. Employing a larger decoder (∼351M parameters), implemented as a hybrid transformer–CNN architecture following [290], yields slight improvements in reconstruction while degrading generative performance. This suggests that increasing decoder capacity may not yield additional benefits when training on a dataset of ImageNet's scale. Using only the latent alignment stage (stage 1) preserves high-level semantics but causes poor reconstruction, rendering this configuration unsuitable for generative modeling. Introducing the perceptual alignment stage (stage 2) resolves this issue, substantially improving both reconstruction and generation, thereby validating the need to fine-tune the encoder. Extending to the three-stage design with decoder refinement further strengthens reconstruction and slightly

Table 7.2: **Comparison of Various Pretrained Encoders.** ImageNet 256×256 at 80K steps; 30 sampling steps; CFG tuned for best gFID. Stage 3 is not applied.

Config.	rFID↓	gFID↓	IS↑	Prec↑	Recall↑
MAE	**0.29**	3.12	216.5	**0.834**	0.543
SigLIP 2	0.35	2.22	246.1	0.816	0.576
DINOv2	0.36	**2.19**	**248.6**	0.811	**0.591**

improves generative quality, showing the importance of progressive alignment.

Other Pretrained Encoders. We compare different pretrained encoders as the backbone of our tokenizer. As shown in Tab. 7.2, *MAE* achieves the strongest reconstruction fidelity but performs the worst in generation, likely due to its reconstruction objective. *DINOv2* achieves the best balance, delivering superior generation quality while maintaining competitive reconstruction performance.

7.3.2 Comparison with Other Tokenizers

We provide a comprehensive comparison with two baselines: the vanilla VAE and VA-VAE [306], which is a representative method that applies semantic regularization using a pretrained DINOv2 model to the latent space. Unless otherwise noted, all baseline checkpoints are taken from the official VA-VAE implementation.

Class-Conditional Generation

Sampling Step. In Fig. 7.4 left, we show that our method achieves better performance than VA-VAE with fewer sampling steps. While VA-VAE requires more than 120 steps to approach its best FID, our tokenizer reaches near-optimal performance with 80 steps. Remarkably, our 50-step generations even surpass the quality of VA-VAE's outputs at 250 steps – the default in its official implementation. We attribute this to the smoother latent space, where discretization errors cause only minor variations, rather than drastic shifts in color, semantics, or overall composition. This robustness allows the model to maintain

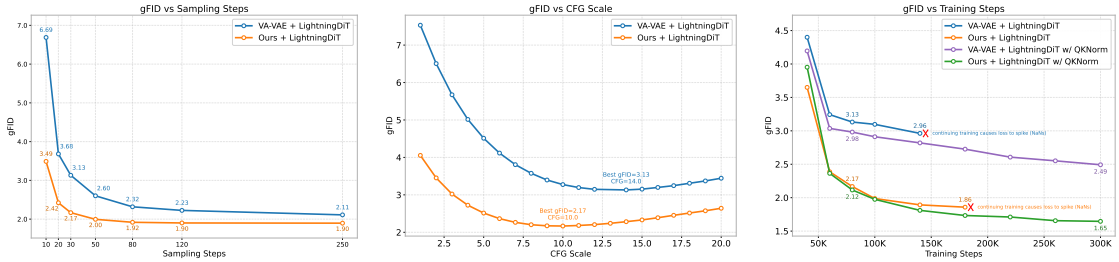


Figure 7.4: **Comparison of Sampling Steps, CFG Scales, and Convergence Speed.** Evaluated on ImageNet 256×256. *Left*: effect of sampling steps versus gFID at 80K training steps. *Middle*: effect of CFG scale versus gFID at 80K training steps with 30 sampling steps. *Right*: effect of training steps versus gFID with 30 sampling steps. QKNorm is enabled during extended training to ensure stability. All gFIDs in the left and right figures are reported using the best-searched CFG scale.

higher fidelity with fewer sampling steps.

CFG Scale. In Fig. 7.4 middle, we plot gFID versus CFG scale, where our tokenizer consistently outperforms VA-VAE across the entire range. Notably, our method achieves strong performance even at low CFG values, whereas VA-VAE relies on larger guidance scales to reach comparable fidelity. This shows that our latent space already encodes well-separated class semantics, reducing the dependence on aggressive guidance and yielding better generations with smaller CFG.

Convergence Speed. We compare the convergence speed of generative model training using our tokenizer against VA-VAE. As shown in Fig. 7.4 right, our approach consistently achieves better gFID across training iterations. By aligning with a pretrained encoder rather than relying on regularization, our method provides a more semantically structured latent space. This leads to roughly 5x faster training, requiring only ∼60K steps compared to VA-VAE's 300K steps for comparable quality, when evaluated with 30 sampling steps.

Different Channel Dimensions. In Tab. 7.3, we compare Vanilla VAE, VA-VAE, and our method using latent spaces with 32 and 64 channels. Key observations include: (1) Vanilla VAE performs worst in terms of gFID. This is because its latent space primarily encodes low-level details, which diffusion models struggle to exploit effectively. (2) Our

Table 7.3: **Comparison with Other Tokenizers.** Evaluated on ImageNet 256×256 at 80K training steps with 30 sampling steps. The checkpoints for both the Vanilla VAE and VA-VAE with CNN encoders are taken from the official VA-VAE repository. *VA-VAE*† denotes the VA-VAE model we trained, using a ViT encoder that matches our architecture but initialized from scratch.

Tokenizer	Enc. Arch.	rFID↓	L. P. Acc.↑	gFID (uncond)↓	gFID w/o CFG↓	gFID w/ CFG↓
f16d32 (downsampling factor 16, latent dimension 32)						
Vanilla VAE	CNN	**0.26**	6.04%	29.12	10.17	3.31
VA-VAE	CNN	0.28	22.96%	19.12	7.79	3.13
VA-VAE†	ViT	0.37	33.57%	18.27	8.21	3.16
Ours	ViT	**0.26**	**35.09%**	**11.80**	**4.05**	**2.17**
f16d64 (downsampling factor 16, latent dimension 64)						
Vanilla VAE	CNN	0.17	5.09%	36.41	16.99	4.03
VA-VAE	CNN	**0.14**	19.72%	26.70	12.23	3.20
VA-VAE†	ViT	0.18	43.53%	19.81	7.92	3.19
Ours	ViT	0.17	**46.99%**	**14.40**	**5.24**	**2.34**

method consistently outperforms VA-VAE in terms of gFID, regardless of whether CFG is used. (3) When VA-VAE employs the same ViT encoder and is trained for the same number of steps as ours, it achieves comparable linear probing accuracy but still falls short in generative performance. This suggests that our semantic structure provides advantages beyond class separation, yielding a more semantically organized latent space that boosts the generative performance of diffusion models.

Unconditional Generation

We also evaluate our method in the unconditional setting (class number = 1), as shown in Tab. 7.3. Our semantic latent space consistently outperforms baseline tokenizers, producing higher-quality generations without relying on class information. It is worth noting that all models are trained for only 80K steps, so the results may not reflect fully optimized performance.

Table 7.4: **System-Level Comparison.** We compare with VAR [251], MagViT-v2 [312], MAR [150], l-DeTok [299], MaskDiT [335], DiT [199], SiT [177], FasterDiT [305], MDT [77], MDTv2 [78], REPA [314], CausalFusion [58], MAETok [40], and VA-VAE [306]. Gray and purple regions refer to LightningDiT trained for 64 epochs (80K training steps, no QKNorm) and 800 (1M training steps, with QKNorm) epochs, respectively. Bold numbers indicate the best result in each color block.

Method	**Tokenizer**	**# token × # dim**	**rFID↓**	**Training Epochs**	**Gen w/o CFG**				**Gen w/ CFG**			
					gFID ↓	**IS ↑**	**Prec ↑**	**Recall ↑**	**gFID ↓**	**IS ↑**	**Prec ↑**	**Recall ↑**
AutoRegressive (AR)												
VAR-d30	-	256 × 32	-	350	-	-	-	-	1.92	323.1	0.82	0.59
MagViT-v2	-	256 × 5	-	1080	3.65	200.5	-	-	1.78	319.4	-	-
MAR	LDM	256 × 16	0.53	800	2.35	227.8	0.79	0.62	1.55	303.7	0.81	0.62
MAR-L	l-DeTok	256 × 16	0.68	800	1.86	238.6	0.82	0.61	1.35	304.1	0.81	0.62
Diffusion Transformer Using SD-VAE												
MaskDiT	SD-VAE	1024 × 4	0.61	1600	5.69	177.9	0.74	0.60	2.28	276.6	0.80	0.61
DiT				1400	9.62	121.5	0.67	0.67	2.27	278.2	0.83	0.57
SiT				1400	8.61	131.7	0.68	0.67	2.06	270.3	0.82	0.59
FasterDiT				400	7.91	131.3	0.67	0.69	2.03	264.0	0.81	0.60
MDT				1300	6.23	143.0	0.71	0.65	1.79	283.0	0.81	0.61
MDTv2				1080	-	-	-	-	1.58	314.7	0.79	0.65
REPA				800	5.90	-	-	-	1.42	305.7	0.80	0.65
CausalFusion				800	3.61	180.9	0.75	0.66	1.77	282.3	0.82	0.61
LightningDiT without QKNorm												
LightningDiT	MAETok	128 × 32	0.48	320	2.21	208.3	-	-	1.73	308.4	-	-
LightningDiT	VA-VAE	256 × 32	0.28	800	2.17	205.6	0.77	0.65	1.35	295.3	0.79	0.65
LightningDiT with QKNorm												
LightningDiT	VA-VAE	256 × 32	0.28	800	2.50	**206.2**	**0.76**	0.65	1.52	286.6	**0.79**	**0.65**
LightningDiT	Ours	256 × 32	**0.26**	800	**2.04**	**206.2**	**0.76**	**0.67**	**1.37**	**293.6**	**0.79**	**0.65**
LightningDiT without QKNorm												
LightningDiT	VA-VAE	256 × 32	0.28	64	5.14	130.2	0.76	**0.62**	2.11	252.3	0.81	0.58
LightningDiT	Ours	256 × 32	**0.26**	64	**3.71**	**148.9**	**0.77**	**0.62**	**1.90**	**260.9**	**0.81**	**0.61**

Reconstruction

As shown in Tab. 7.3, our method achieves competitive reconstruction with 32 channels but lags behind VA-VAE (CNN encoder) at 64 channels. Lowering the semantic preservation loss weight and increasing learning rate in stage 2 improves reconstruction to match VA-VAE, with only a slight drop in generation performance (still surpassing VA-VAE).

7.3.3 Comparison with Other Systems

We conduct a systematic comparison with other systems, as shown in Tab. 7.4. Both VA-VAE and our method are sampled using 250 sampling steps. When comparing our method to VA-VAE at 64 epochs (80K training steps), we surpass it in both reconstruction and generation quality, highlighting our superior convergence speed. For the 800-epoch setting (1M training steps), we retrain LightningDiT of VA-VAE using the official repository with QKNorm enabled – necessary to avoid loss spikes, but slightly degrade generative performance, as noted by the authors. Our method (with QKNorm) achieves a gFID of 1.37, outperforming VA-VAE's 1.52 (with QKNorm) and comparable to the original VA-VAE result of 1.35 (without QKNorm) reported in their paper.

7.3.4 Scale-Up Experiments on Text-to-Image Generation

We conduct a system-level comparison of our tokenizer with the widely adopted FLUX VAE on the text-to-image generation task. We train our tokenizer on images resized so that their shortest edge is 256 pixels, preserving aspect ratios. The diffusion model is trained for 200K steps at 256 resolution and fine-tuned for 90K steps at 512 resolution, both with variable aspect ratios.

For comparison, Fig. 7.5 presents results from generative models trained with our tokenizer and with FLUX VAE for 100K steps at 256×256 resolution. Our method produces images with better coherence, stronger text alignment, and competitive visual quality. Quantitative results (Tab. 7.5) further confirm the advantage of our approach. Fig. 7.6 presents our generated results at 512 resolution. Notably, our tokenizer is trained only on 256-resolution images, demonstrating its ability to generalize to unseen resolutions. These results suggest that our approach scales effectively and can potentially serve as a strong

Table 7.5: **Quantitative Comparison on Text-to-Image (T2I) Generation with FLUX VAE.** Compared on *COCO Prompt 6K*, which has 6K captions sampled from the COCO validation set. Each 2B-parameter T2I model is trained for 100K steps and evaluated at 256×256 resolution with CFG. rFID is computed using 200K randomly sampled images from the COYO-700M dataset [181].

Tokenizer	rFID	gFID	KID	HPSv2	PickScore	ImageReward	Aesthetic Scores	CLIP Scores	VQA Scores
FLUX VAE	**0.102**	35.78	0.018	0.242	0.397	0.162	5.411	31.21	0.775
Ours	0.443	**30.27**	**0.015**	**0.249**	**0.603**	**0.564**	**5.573**	**32.21**	**0.849**

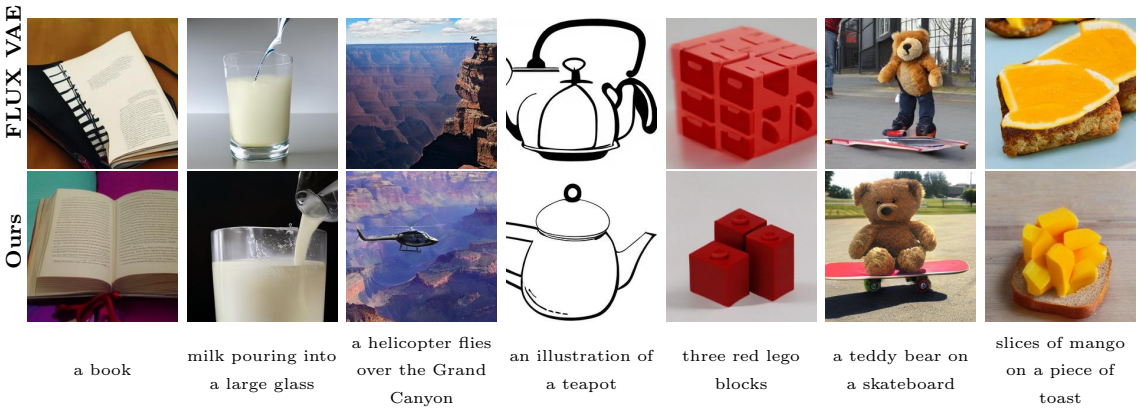


Figure 7.5: **Qualitative Comparison on Text-to-Image Generation with FLUX VAE.** Input text prompts are shown below the images and results (256×256 resolution) are generated from generative models trained for 100K steps. Our method (bottom row) produces images with better coherence and prompt alignment compared to the one using FLUX VAE (top row).

alternative to FLUX VAE in large-scale training.

7.4 Limitations and Discussions

Our method, while effective, has several limitations. First, although the semantic latent space improves generative quality, its reconstruction ability still lags behind FLUX VAE.

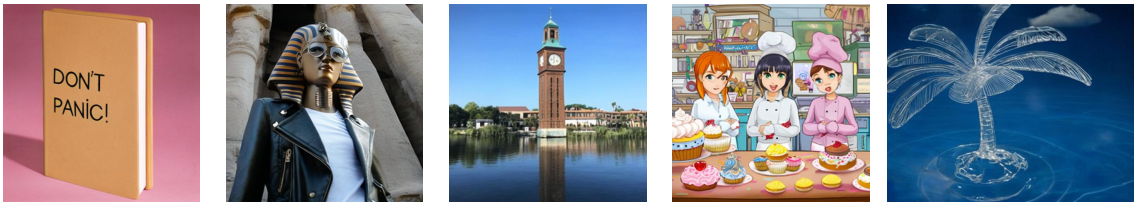


Figure 7.6: **Qualitative Results of Our Method on Text-to-Image Generation at 512 Resolution.** The input text prompts are shown below the images. Results are obtained from diffusion models trained for 290K steps. The first four are square images, and the final one has a 4:5 aspect ratio.

This gap could be narrowed with stronger decoders, larger channel dimensions, longer training, or scaling to larger models and compute budgets. Second, our evaluation is limited to images up to 512 resolution. Exploring higher resolutions is an interesting future direction, and the recently introduced DINOv3 – showing strong capability for variable resolutions – could be leveraged for this purpose.

Our study highlights a key insight: aligning a pretrained semantic encoder yields a more generation-friendly latent space than learning semantic from scratch. Although our work focuses on tokenizers for image diffusion, extending this approach to video tokenization, discrete tokenizers for autoregressive generation, and unified representations for multi-modal models is a promising direction for future work. We hope our findings inspire a rethinking of tokenizer design in generative modeling.

SUPPLEMENTARY MATERIAL

7.5 Implementation Details

Tokenizer. For ImageNet, we set the downsampling ratio to $f = 16$, the latent channel dimension to $d = 32$. We use the pretrained DINOv2 checkpoint *vit_large_patch14_dinov2.lvd142m* (∼304M parameters) [191] as encoder. The adapter is implemented as a two-layer MLP that projects the 1024-dimensional encoder output into a 32-dimensional latent space. The decoder is a convolutional network with ∼42M parameters, same as VA-VAE [306]. The training images are first resized so that the shorter edge is 256 pixels while preserving the aspect ratio, and then randomly cropped to 256×256. Since the DINOv2 encoder uses a patch size of 14, we first resize the input image to 224×224 resolution before feeding it into the encoder. This produces a 16×16 latent feature map, matching the downsampling ratio of $f = 16$ for 256-resolution images. For training, we use a batch size of 64 and train on 8 NVIDIA H100 GPUs. The learning rate is set to 1e-4 for the first and third stages, and 1e-5 for the second stage. Following [306], the generator loss in L_{GAN} is rescaled to match the magnitude of $L_{\ell_1} + w_p L_{\text{perceptual}}$ based on the ratio of their gradient norms at the last layer of the decoder, where $w_p = 1.0$. Similarly, we rescale L_{sp} according to the gradient ratio at the last layer of the encoder. We set $w_g = 0.5$ and $w_{sp} = 1$, which are applied after loss rescaling. We enable L_{GAN} after 5K training steps in the first stage, and keep it active throughout the second and third stages. We enable Exponential Moving Average (EMA) [128] during training and apply them in the second and third stages during inference. Training runs for 6 epochs in the first stage (∼120K steps, ∼7 hours), 11 epochs in the second stage (∼220K steps, ∼24 hours), and 5 epochs in the third stage (∼100K steps, ∼6 hours).

For LAION dataset, we use the pretrained DINOv2 checkpoint *vit_large_patch14_reg4_dinov2* (∼304M parameters) [225] as encoder. The tokenizer is trained with a batch size of 256 and a learning rate of 1e-4 for all stages, using 32 NVIDIA H100 GPUs. We set w_{sp} to 3. All

other settings follow those of the ImageNet experiments. Training images are resized such that the shortest edge is 256 pixels while preserving the aspect ratio, without restricting them to square shapes. Similar to before, the input images are resized to multiples of 14 so that the encoder output aligns with the downsampling ratio of $f = 16$. We train for 300K steps in the first stage (∼30 hours), 100K steps in the second stage (∼20 hours), and 50K steps in the last stage (∼5 hours).

Generative Models. For ImageNet experiments, we use LightningDiT [306] as the generative model (∼673M parameters) and adopt the same hyperparameters as in its official implementation. We set the batch size to 1024, the learning rate to 2e-4, and the transformer patch size to 1. Unless otherwise specified, all experiments are trained for 80K steps (64 epochs), which takes approximately 12 hours on 8 NVIDIA H100 GPUs. We do not enable QKNorm for models trained with fewer than 200K steps, following the official implementation. When training models for more than 200K steps, we enable QKNorm from the start of the training to stabilize optimization and prevent loss spikes. We use the Euler sampler with 30 sampling steps unless otherwise specified. Following LightningDiT, we apply CFG to only the first three latent channels for a fair and consistent comparison, unless otherwise specified.

For LAION experiments, we employ a diffusion transformer (2B parameters) following the FLUX implementation [144, 321, 122, 123]. We use a learning rate of 1e-4 with weight decay and a transformer patch size of 1. Training begins on 256-resolution images with varying aspect ratios for 200K steps using a batch size of 2048, which requires roughly 140 hours on 32 H100 GPUs. We then continue on 512-resolution images with varying aspect ratios for an additional 90K steps using a batch size of 512, requiring about 135 hours on 32 H100 GPUs. For inference, we apply EMA checkpoints and use a DDIM sampler [242] with 50 steps, setting the classifier-free guidance (CFG) scale to 5.

Datasets. For our text-to-image training dataset LAION-2B, we apply a series of pre-filters to remove low-quality or undesired samples: we exclude images that are NSFW, watermarked, low-aesthetic, invalid, or from blocked domains, resulting in a subset of about 616M images.

Linear Probing. For linear probing, we follow the standard ImageNet protocol widely

used to assess the semantic quality of latent representations. Specifically, we freeze both the VAE encoder and the adapter, and train a single linear classifier on top of the adapter's output features to predict ImageNet-1K classes. We use linear probing because it directly measures how much semantic information is linearly accessible in the latent space, making it a widely adopted proxy for semantic quality. The classifier is trained for 3 epochs using Adam optimizer with a learning rate of 0.001 and a batch size of 256 on the ImageNet-1K training set, and evaluated on the ImageNet validation set.

Ablation Study. All methods, except for the *Full Model*, are trained without the third stage (*i.e.*, decoder refinement). For the *Larger Decoder* variant, we replace the CNN decoder with a larger decoder composed of a ViT architecture followed by a CNN module (totaling ∼351M parameters), adapted from [290]. For the variant *LoRA Fine-Tuning*, we apply LoRA to both the attention and MLP layers, using a rank of 16, an alpha of 32, and a dropout of 0.1. All variants (except for *LoRA Fine-Tuning*) are trained with the same hyperparameters and training iterations for fair comparison: 6 epochs in the first stage and 11 epochs in the second stage. The *Full Model* is additionally trained with a third stage of 6 epochs. These settings follow the default configuration described in the main paper.

Comparison with Other Pretrained Encoders. All methods are evaluated without training Stage 3. We use "vit_large_patch16_224.mae" and "google/siglip2-large-patch16-256" as the pretrained encoders for the *MAE* and *SigLIP 2* variants, respectively. For the *SigLIP 2* variant, we set $w_{sp} = 2$, as this yields better results. Apart from this adjustment, all methods are trained with identical hyperparameters and iterations, following the default configuration outlined in the main paper.

Comparison with Other Tokenizers. Our method with 32 channels is trained following the default configuration described in the main paper. For the 64-channel version, we train for 6 epochs (∼120K steps) in the first stage, 17 epochs (∼340K steps) in the second stage, and 11 epochs (∼220K steps) in the third stage. For the *Vanilla VAE* and *VA-VAE* with a CNN encoder (∼28M parameters), we use the pretrained checkpoints from [306] for both the 32- and 64-channel experiments. All pretrained checkpoints were trained with a batch size of 256 for 50 epochs, except the 32-channel VA-VAE model, which was trained for 125 epochs. For *VA-VAE* with a ViT encoder, we adopt the same architecture as our method

Table 7.6: **Training Cost Comparison.** All measurements are obtained at an input resolution of 256×256 on 8 NVIDIA A100 GPUs (batch size 8 per GPU). Values marked with ∗ denote GPU-hour estimates computed from the training epochs reported in the official implementation of [306].

Tokenizer	Enc. #Param	Trainable Params	Frozen Params	Total GPU Memory	Time Per Training Step	Total A100 GPU Hours
Vanilla VAE	28.41 M	72.60 M	14.71 M	148.2 GB	0.360 s	800.0*
VA-VAE		72.63 M	319.0 M	203.0 GB	0.534 s	2966*
Our Stage 1	304.4 M	44.25 M	319.0 M	157.5 GB	0.402 s	107.2
Our Stage 2		348.6 M	319.0 M	238.2 GB	0.772 s	377.4
Our Stage 3		44.18 M	319.1 M	159.0 GB	0.412 s	91.55

but without initializing it from DINOv2 weights, and use the learning rate of 1e-4. The 32-channel version is trained for 22 epochs, whereas the 64-channel version is trained for 34 epochs, matching the number of epochs used in our model training.

Scale-Up Experiments on Text-to-Image Generation. To compute rFID, we use 200K images randomly sampled from the COYO-700M [181] dataset. The gFID and KID [18] on *COCO Prompt 6K* are computed using 6K generated images and 6K corresponding real images from the sampled captions.

For completeness, we provide the reference for the metrics we used for evaluation: FID [101], KID [18], HPSv2 [284], PickScore [136], ImageReward [293], Aesthetic Scores [225], CLIP Scores [100], VQA Scores [159], LPIPS [325].

7.6 More Experiments

7.6.1 Other Pretrained Encoders

Tab. 7.9 reports additional reconstruction metrics comparing different pretrained encoders. The one using DINOv2 achieves the best balance, delivering superior generation quality while maintaining competitive reconstruction performance.

Table 7.7: **Inference Memory Consumption Comparison.** We evaluate the peak GPU memory consumption under different image resolutions and batch sizes.

Tokenizer	Image Resolution	Encoder #Params	Encoder Peak GPU Memory (GB)			Decoder #Params	Decoder Peak GPU Memory (GB)		
			batch size=1	batch size=4	batch size = 8		batch size=1	batch size=4	batch size = 8
VA-VAE	256 × 256	28.41 M	0.301	0.838	1.543	41.42 M	0.310	0.713	1.250
Ours	256 × 256	304.4 M	1.363	1.372	1.385				
VA-VAE	512 × 512	28.41 M	0.841	2.959	5.794	41.42 M	0.713	2.324	4.472
Ours	512 × 512	304.4 M	1.284	1.450	1.674				
VA-VAE	1024 × 1024	28.41 M	2.958	11.45	22.78	41.42 M	2.324	8.768	17.36
Ours	1024 × 1024	304.4 M	1.450	2.121	3.015				

Table 7.8: **Inference Compute Cost Comparison.** We measure the GFLOPS and Latency under different image resolutions, using a single NVIDIA A100 GPU with a batch size of 1.

Tokenizer	Image Resolution	Encoder			Decoder		
		#Params	GFLOPS	Latency	#Params	GFLOPS	Latency
VA-VAE	256 × 256	28.41 M	69.21	6.548 ms	41.42 M	126.5	13.91 ms
Ours	256 × 256	304.4 M	77.85	13.42 ms			
VA-VAE	512 × 512	28.41 M	279.2	19.44 ms	41.42 M	509.3	42.31 ms
Ours	512 × 512	304.4 M	310.4	18.92 ms			
VA-VAE	1024 × 1024	28.41 M	1155	73.92 ms	41.42 M	2089	173.9 ms
Ours	1024 × 1024	304.4 M	1241	121.2 ms			

7.6.2 Reconstruction

Fig. 7.9 shows a qualitative comparison of reconstruction quality on the ImageNet 256×256 dataset. The variant *Ours w/o Stage 2 + 3* (fourth column) fails to reconstruct the input accurately, while the other methods show similar reconstruction quality.

Tab.7.10 reports additional reconstruction metrics for different tokenizers. While *Ours* (the version presented in the main paper) outperforms all baselines in generative performance, it remains quantitatively weaker in reconstruction quality. To address this, we report a variant of our method trained with a larger learning rate (increased from 1e-5 to

1e-4) and a smaller semantic preservation weight (reduced from 1.0 to 0.5) in stage 2. This hyperparameter setting pushes the tokenizer to learn perceptual details more aggressively. For this variant, we train the first stage for 6 epochs, second stage for 11 epochs, the final stage for 10 epochs. The resulting variant achieves competitive reconstruction performance (on par with VA-VAE using ViT encoder) while still surpassing VA-VAE in generation. This demonstrates that our method can improve reconstruction with only a minor trade-off in generative quality through simple hyperparameter adjustments. Other strategies for improvement include extending stage 2 or stage 3 with additional training iterations, or using a larger batch size. A more aggressive approach is to replace the decoder with a stronger architecture during stage 3 and train the new decoder. While this requires additional training resources, it offers flexibility without compromising the learned semantic space.

7.6.3 Convergence Speed

Figs. 7.10, Fig. 7.11, and Fig. 7.12 present additional comparisons between our method and VA-VAE on the ImageNet 256×256 dataset. Our method converges faster, indicating that aligning and preserving semantics in the pretrained encoder is more effective than semantic regularization.

Fig. 7.13, Fig. 7.14, Fig. 7.15, and Fig. 7.16 compare the convergence speed of our method with FLUX VAE. Our method converges significantly faster, demonstrating the advantage of the learned semantically rich latent space.

7.6.4 More Quantitative Results

Tab. 7.15 reports quantitative results on two additional prompt sets, showing that our tokenizer consistently outperforms FLUX VAE in generation performance.

7.6.5 More Qualitative Results

Fig. 7.19 shows our sampled results of class-conditioned image generation on ImageNet 256×256 dataset, trained with 800 epochs.

Table 7.9: **Comparison with Other Pretrained Encoders.** Evaluated on ImageNet 256×256 at 80K training steps with 30 sampling steps, using the CFG scale that yields the lowest gFID. Decoder refinements (stage 3) are not applied. While the model with MAE yields the strongest reconstruction performance, it performs the worst in generation quality. The model with DINOv2 achieves the best overall generation quality, with a slight drop in reconstruction quality compared to MAE.

Configuration	**rFID↓**	**PSNR↑**	**LPIPS↓**	**gFID↓**	**IS↑**	**Prec↑**	**Recall↑**
MAE	**0.29**	**26.12**	**0.113**	3.12	216.5	**0.834**	0.543
SigLIP 2	0.35	25.29	0.129	2.22	246.1	0.816	0.576
DINOv2	0.36	25.62	0.121	**2.19**	**248.6**	0.811	**0.591**

Fig. 7.20 shows our sampled results of text-to-image generation at resolution 256×256, trained with 200K steps.

Fig. 7.21 and Fig. 7.22 show our sampled results of text-to-image generation at resolution 512×512. Fig. 7.23 and Fig. 7.24 further show results at different aspect ratios, including portrait mode and landscape mode. All these results are generated using the model trained with 290K steps.

7.6.6 Failure Case

Fig. 7.8 shows the failure case of our method. Issues include inaccurate rendering of clock numerals (*e.g.*, the 12), incorrect object counts, incorrect long text generation, and difficulties in rendering fine details such as hands.

7.7 LLM Usage

In preparing this manuscript, we used large language models (LLMs) as general-purpose writing assistants for grammar corrections, rephrasing, and clarity/concision edits. All LLM-suggested edits were reviewed and verified by the authors, who take full responsibility for the final manuscript.

Table 7.10: **Comparison of Other Tokenizers with Different Configurations.** Evaluated on ImageNet 256×256 at 80K training steps with 30 sampling steps. The checkpoints for both the Vanilla VAE and VA-VAE with CNN encoders are taken from the official VA-VAE repository. *VA-VAE*† denotes the VA-VAE model we trained, using a ViT encoder that matches our architecture but initialized from scratch. *Ours* refers to the version of our method presented in the main paper. *Ours** indicates a variant trained with a larger learning rate and smaller semantic preservation weight, which achieves reconstruction quality comparable to VA-VAE but with slightly reduced generation performance.

Tokenizer	**Enc. Arch.**	**rFID↓**	**PSNR↑**	**LPIPS↓**	**L. P. Acc.↑**	**gFID w/ CFG↓**
f16d32 (downsampling factor 16, latent dimension 32)						
Vanilla VAE	CNN	**0.26**	**27.14**	**0.097**	6.04%	3.31
VA-VAE	CNN	0.28	26.31	0.104	22.96%	3.13
VA-VAE†	ViT	0.37	25.66	0.130	33.57%	3.16
Ours	ViT	**0.26**	25.83	0.117	**35.09%**	**2.17**
f16d64 (downsampling factor 16, latent dimension 64)						
Vanilla VAE	CNN	0.17	**29.38**	**0.061**	5.09%	4.03
VA-VAE	CNN	**0.14**	29.13	0.062	19.72%	3.20
VA-VAE†	ViT	0.18	29.12	0.075	43.53%	3.19
Ours	ViT	0.17	27.41	0.089	**46.99%**	**2.34**
Ours*	ViT	**0.14**	28.91	0.070	45.22%	2.50

Table 7.11: **Quantitative Comparison on Text-to-Image (T2I) Generation with FLUX VAE.** We report metrics on two additional prompt sets for T2I models trained with our VAE and FLUX VAE, each for 100K steps, evaluated at 256×256 resolution. The rFID is computed using 200K randomly sampled images from the COYO-700M dataset [181].

Tokenizer	rFID	HPSv2	PickScore	ImageReward	Aesthetic Scores	CLIP Scores	VQA Scores
Parti Prompt [309] (with CFG)							
FLUX VAE	**0.102**	0.239	0.391	0.235	5.292	31.49	0.705
Ours	0.443	**0.246**	**0.609**	**0.594**	**5.389**	**32.56**	**0.782**
HPSv2 Prompt [284] (with CFG)							
FLUX VAE	**0.102**	0.224	0.373	0.090	5.478	31.59	0.737
Ours	0.443	**0.231**	**0.626**	**0.366**	**5.604**	**33.12**	**0.792**

Table 7.12: **Quantitative Comparison on Text-to-Image (T2I) Generation with FLUX VAE (Without CFG).** We report metrics on diverse prompt sets for 2B-parameter T2I models trained with our tokenizer and the FLUX VAE. Each T2I model is trained for 100K steps and evaluated at 256 × 256 resolution. Our tokenizer is trained on LAION dataset. The rFID is computed using 200K randomly sampled images from the COYO-700M dataset [181].

Tokenizer	rFID	gFID	KID	HPSv2	PickScore	ImageReward	Aesthetic Scores	CLIP Scores	VQA Scores
Coco Prompt 6K [158] (without CFG)									
FLUX VAE	**0.102**	48.76	0.025	0.183	0.345	-1.127	4.205	27.05	0.534
Ours	0.443	**33.46**	**0.014**	**0.210**	**0.655**	**-0.581**	**4.963**	**28.87**	**0.658**
Parti Prompt [309] (without CFG)									
FLUX VAE	**0.102**	-	-	0.186	0.378	-1.176	4.003	26.85	0.538
Ours	**0.443**	-	-	**0.206**	**0.622**	**-0.654**	**4.751**	**28.59**	**0.618**
HPSv2 Prompt [284] (without CFG)									
FLUX VAE	**0.102**	-	-	0.170	0.372	–1.206	4.142	26.59	0.567
Ours	0.443	-	-	**0.192**	**0.628**	**-0.764**	**4.955**	**28.20**	**0.645**

Table 7.13: **GenEval Quanlitative Comparison on Text-to-Image (T2I) Generation with FLUX VAE (with and without CFG).** We report metrics on GenEval [80] for 2B-parameter T2I models trained with our tokenizer and FLUX VAE. Each T2I model is trained for 100K steps and evaluated at 256×256 resolution. Our tokenizer is trained on LAION dataset. The rFID is computed using 200K randomly sampled images from the COYO-700M dataset [181].

Tokenizer	rFID	Overall	Single object	Two object	Counting	Colors	Position	Color attribution
with CFG								
FLUX VAE	**0.102**	0.476	0.978	0.477	0.346	0.813	0.107	0.135
Ours	0.443	**0.556**	**0.984**	**0.702**	**0.446**	**0.824**	**0.120**	**0.257**
without CFG								
FLUX VAE	**0.102**	0.230	0.746	0.101	0.090	0.406	0.020	0.015
Ours	0.443	**0.329**	**0.850**	**0.303**	**0.234**	**0.508**	**0.037**	**0.040**

Table 7.14: **Quantitative Comparison on Text-to-Image (T2I) Generation with VA-VAE (with and without CFG).** We report metrics on diverse prompt sets for 1B-parameter T2I models trained with our tokenizer and the VA-VAE. Each T2I model is trained for 50K steps and evaluated at 256×256 resolution. Both our tokenizer and VA-VAE are trained on ImageNet. The rFID is computed using 200K randomly sampled images from the COYO-700M dataset [181].

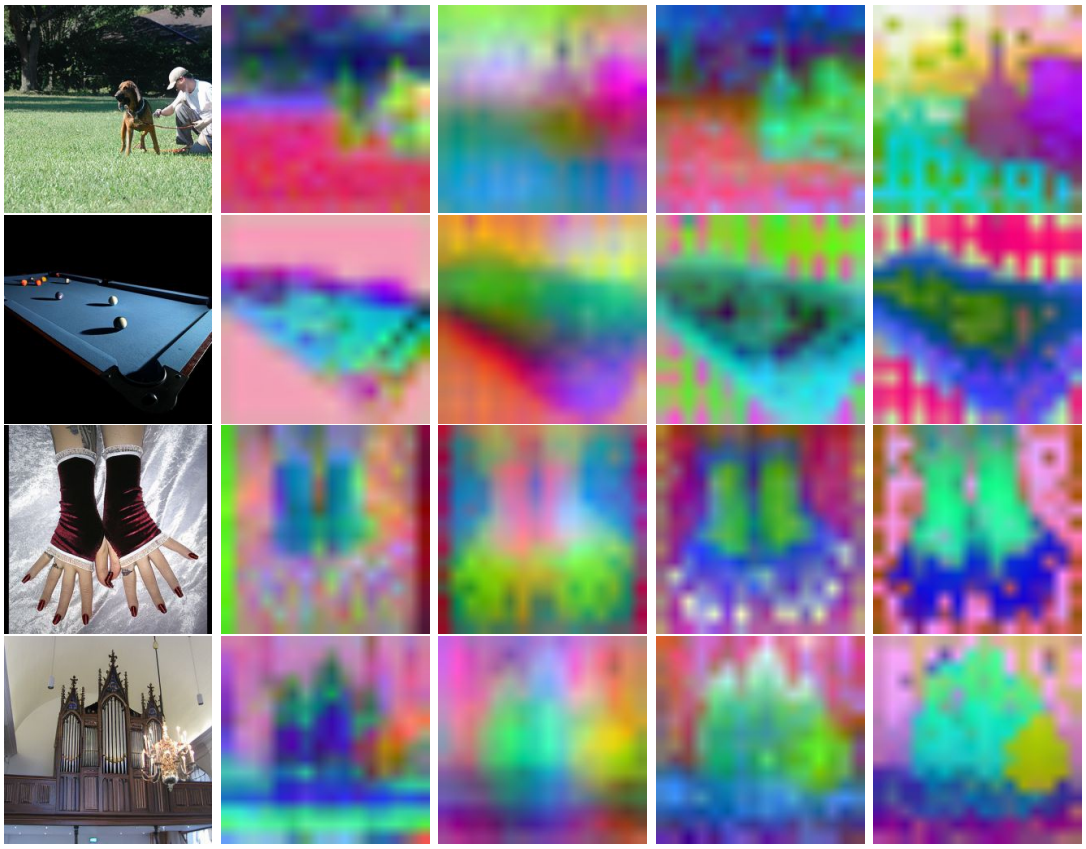
Tokenizer	rFID	gFID	KID	HPSv2	PickScore	ImageReward	Aesthetic Scores	CLIP Scores	VQA Scores
Coco Prompt 6K [158] (with CFG)									
VA-VAE	0.241	34.13	0.016	0.221	0.426	-0.129	5.088	31.50	0.740
Ours	**0.191**	**31.19**	**0.015**	**0.230**	**0.574**	**0.088**	**5.215**	**31.68**	**0.781**
Parti Prompt [309] (with CFG)									
VA-VAE	0.241	-	-	0.220	0.444	-0.146	4.893	31.40	0.677
Ours	**0.191**	-	-	**0.228**	**0.556**	**0.062**	**5.015**	**31.73**	**0.706**
HPSv2 Prompt [284] (with CFG)									
VA-VAE	0.241	-	-	0.201	0.435	-0.287	5.042	31.63	0.708
Ours	**0.191**	-	-	**0.210**	**0.565**	**-0.131**	**5.120**	**32.05**	**0.737**
Coco Prompt 6K [158] (without CFG)									
VA-VAE	0.241	42.35	0.019	0.179	0.448	-1.292	4.352	26.89	0.497
Ours	**0.191**	**39.86**	**0.017**	**0.187**	**0.552**	**-1.121**	**4.493**	**27.53**	**0.548**
Parti Prompt [309] (without CFG)									
VA-VAE	0.241	-	-	0.179	0.457	-1.396	4.163	26.23	0.493
Ours	**0.191**	-	-	**0.186**	**0.543**	**-1.213**	**4.320**	**26.99**	**0.524**
HPSv2 Prompt [284] (without CFG)									
VA-VAE	0.241	-	-	0.162	0.455	-1.378	4.256	26.05	0.521
Ours	**0.191**	-	-	**0.170**	**0.545**	**-1.251**	**4.451**	**26.48**	**0.551**

Table 7.15: **GenEval Quanlitative Comparison on Text-to-Image (T2I) Generation with VA-VAE (with and without CFG).** We report metrics on GenEval [80] for 1B-parameter T2I models trained with our tokenizer and VA-VAE. Each T2I model is trained for 50K steps and evaluated at 256×256 resolution. Both our tokenizer and VA-VAE are trained on ImageNet. The rFID is computed using 200K randomly sampled images from the COYO-700M dataset [181].

Tokenizer	rFID	Overall	Single object	Two object	Counting	Colors	Position	Color attribution
with CFG								
VA-VAE	0.241	0.411	0.944	0.338	0.275	0.782	0.050	0.078
Ours	**0.191**	**0.454**	**0.978**	**0.434**	**0.338**	**0.795**	**0.073**	**0.110**
without CFG								
VA-VAE	0.241	0.194	0.653	0.071	0.088	0.335	0.010	0.005
Ours	**0.191**	**0.243**	**0.734**	**0.136**	**0.125**	**0.418**	**0.033**	**0.013**

Table 7.16: **Quantitative Analysis of Latent Space.** We report metrics for latent space of three tokenizers (Vanilla VAE, VA-VAE, and Ours), using the output space of the pretrained DINOv2 encoder as a reference. CKNNA [112] measures the similarity between each tokenizer's latent space and the DINOv2 output. For the other metrics, we additionally report the percentage indicating how close each tokenizer's results are to those of DINOv2. All metrics are computed using 10K samples from the ImageNet validation set.

Tokenizer	CKNNA	Spatial Variance	Total Variation	Density CV	Gini Coefficient	Normalized Entropy	gFID
Vanilla VAE	0.023	11.41 (+432.9%)	957.6 (+96.3%)	0.310 (+24.0%)	0.173 (+24.4%)	0.994 (-0.2%)	3.31
VA-VAE	0.233	13.74 (+541.7%)	806.1 (+65.1%)	**0.215 (-14.0%)**	**0.122 (-12.2%)**	**0.997 (+0.1%)**	3.13
Ours	**0.282**	**1.205 (-43.7%)**	**302.3 (-38.0%)**	0.295 (+18.0%)	0.160 (+15.1%)	0.994 (-0.2%)	**2.17**
DINOv2	1.000	2.141	487.9	0.250	0.139	0.996	-



(a) Input (b) Vanilla VAE (c) VA-VAE (d) Ours (e) DINOv2

Figure 7.7: **PCA Visualization of Latent Space.** Vanilla VAE can produce latents that are either over-smooth or overly sharp, and VA-VAE tends to generate consistently over-smooth representations. In contrast, our latent space most closely matches the characteristics of DINOv2 outputs, demonstrating that our tokenizer preserves richer semantics.

a clock on a brick building wall of some sort

ten red apples

the words 'KEEP OFF THE GRASS' written on a brick wall

a grandmother reading a book to her grandson and granddaughter

Figure 7.8: **Failure Case of Our Method on Text-to-Image Generation at 512×512 Resolution.** The input text prompts are shown below the images. Results are obtained from generative models trained for 290K steps. Common issues include inaccurate rendering of clock numerals (*e.g.*, the 12), errors in object counting, inconsistencies in generating longer text, and difficulties in producing fine details such as hands.

Input Vanilla VA-VAE Stage 1 Ours

Figure 7.9: **Qualitative Comparison of Reconstruction Quality on ImageNet 256×256 Validation Set.** The variant *Ours w/o Stage 2 + 3* (fourth column) fails to accurately reconstruct the input, as the pretrained visual encoder does not capture sufficient perceptual details in the latent space. All the other methods achieve comparable reconstruction quality qualitatively.

Figure 7.10: **Qualitative Comparison of Convergence Speed on ImageNet 256×256.** We compare with VA-VAE. Results are reported with the best CFG scale, using EMA for sampling except at 20K and 40K steps.

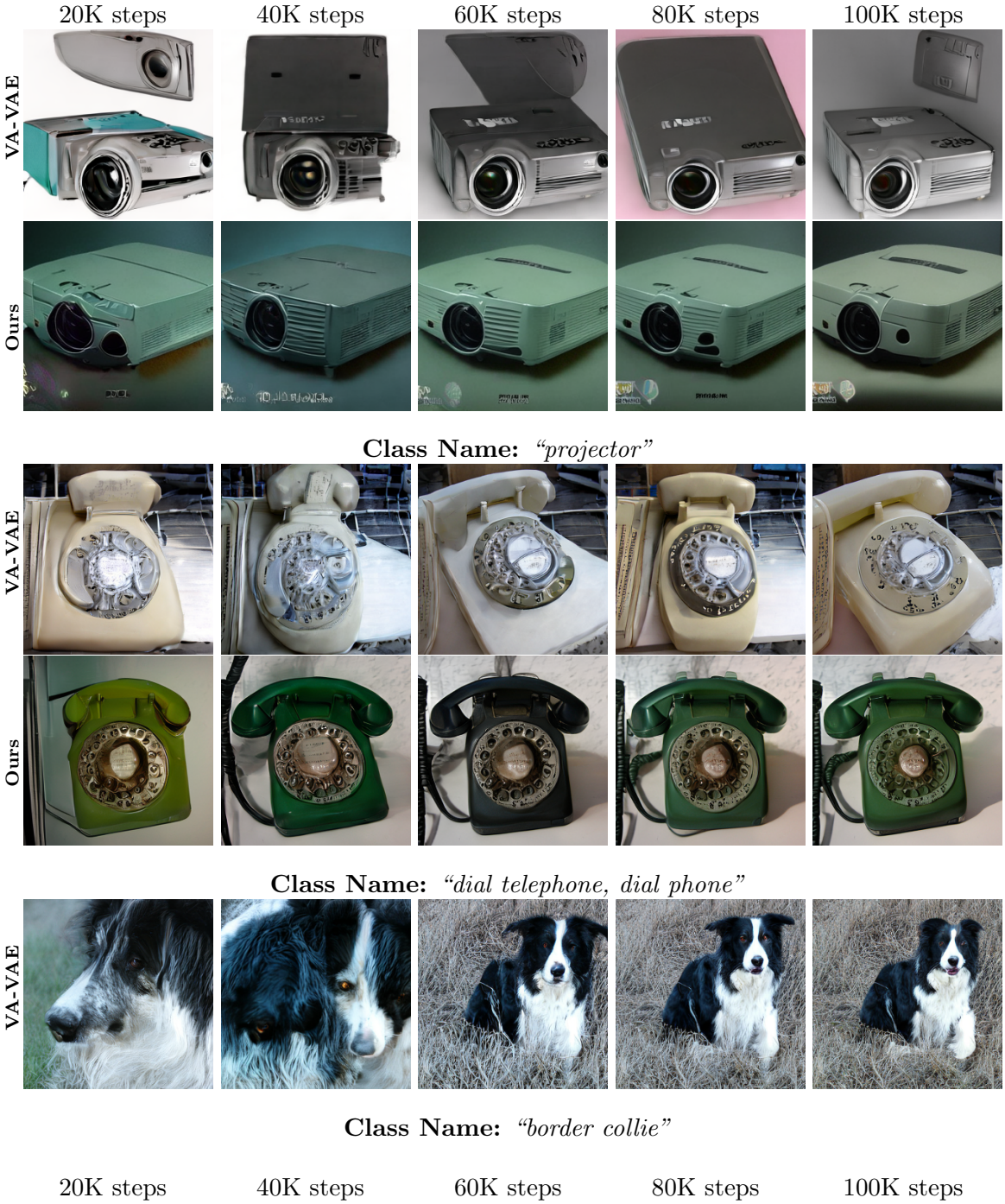


Figure 7.11: **Qualitative Comparison of Convergence Speed on ImageNet 256×256.** We compare with VA-VAE. Results are reported with the best CFG scale, using EMA for sampling except at 20K and 40K steps.

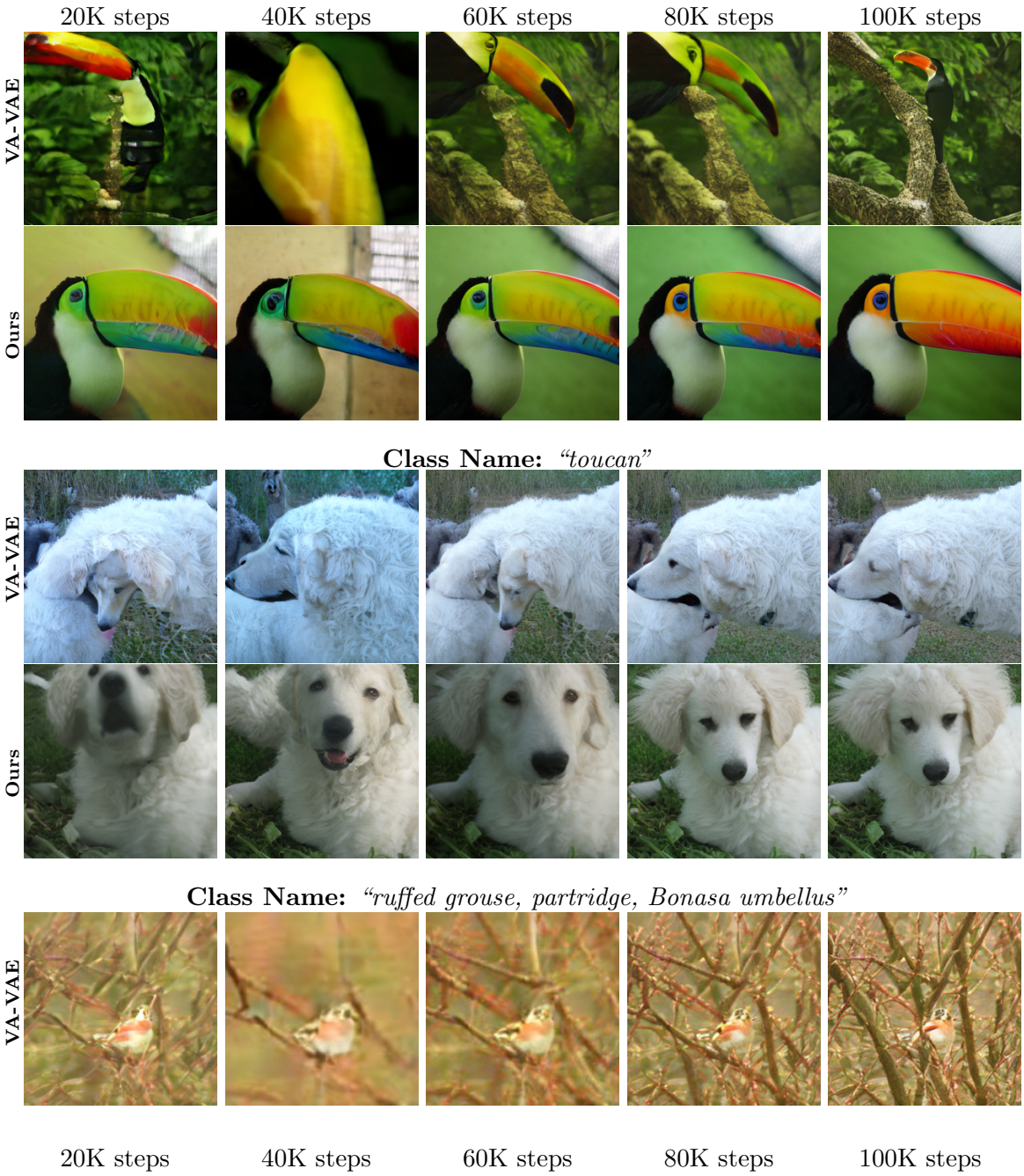


Figure 7.12: **Qualitative Comparison of Convergence Speed on ImageNet 256×256.** We compare with VA-VAE. Results are reported with the best CFG scale, using EMA for sampling except at 20K and 40K steps.

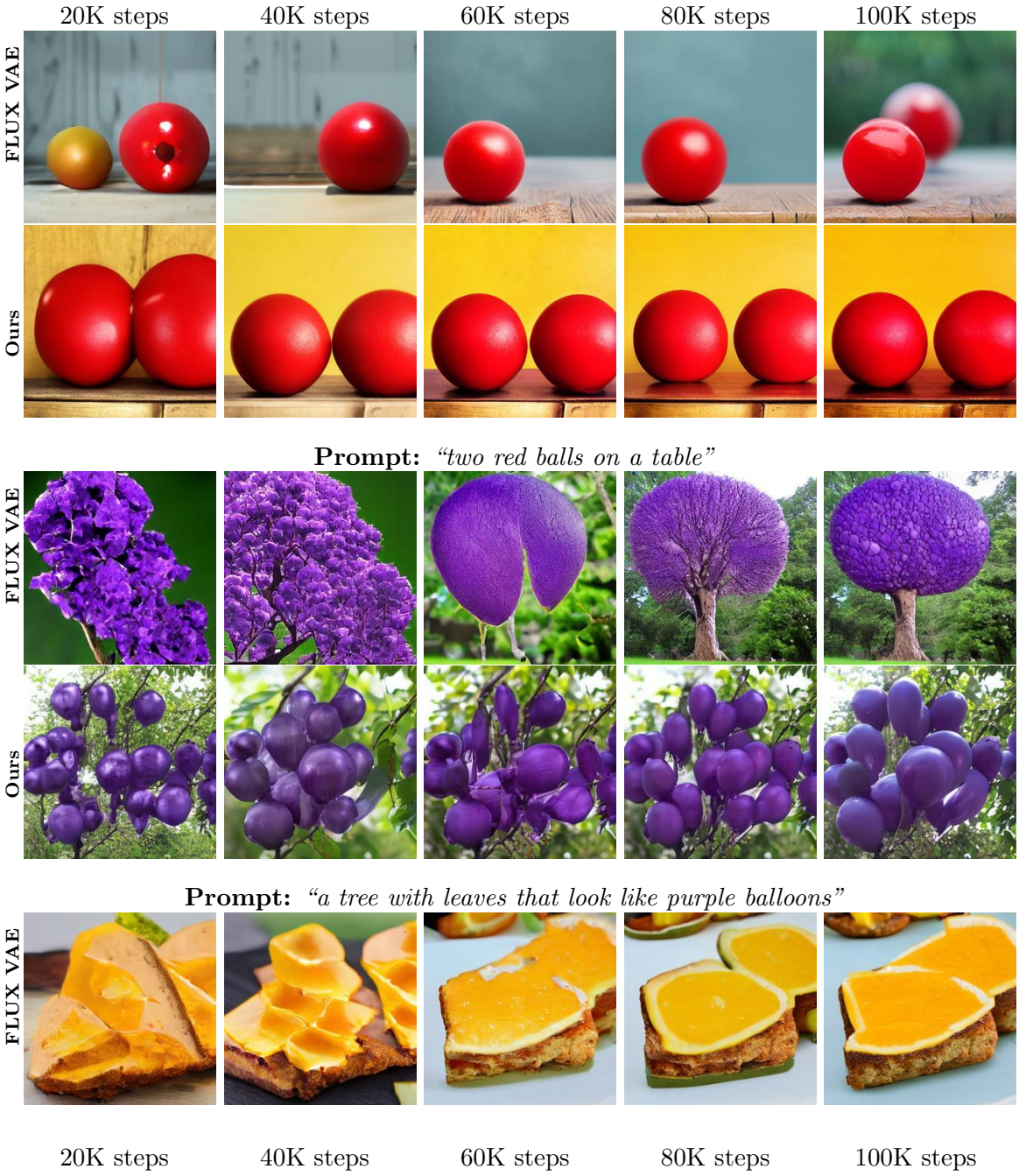


Figure 7.13: **Qualitative Comparison of Convergence Speed on Text-to-Image Generation.** We compare with FLUX VAE. Results are reported with CFG scale set to 5, using images generated at 256×256 resolution.

Figure 7.14: **Qualitative Comparison of Convergence Speed on Text-to-Image Generation.** We compare with FLUX VAE. Results are reported with CFG scale set to 5, using images generated at 256×256 resolution.

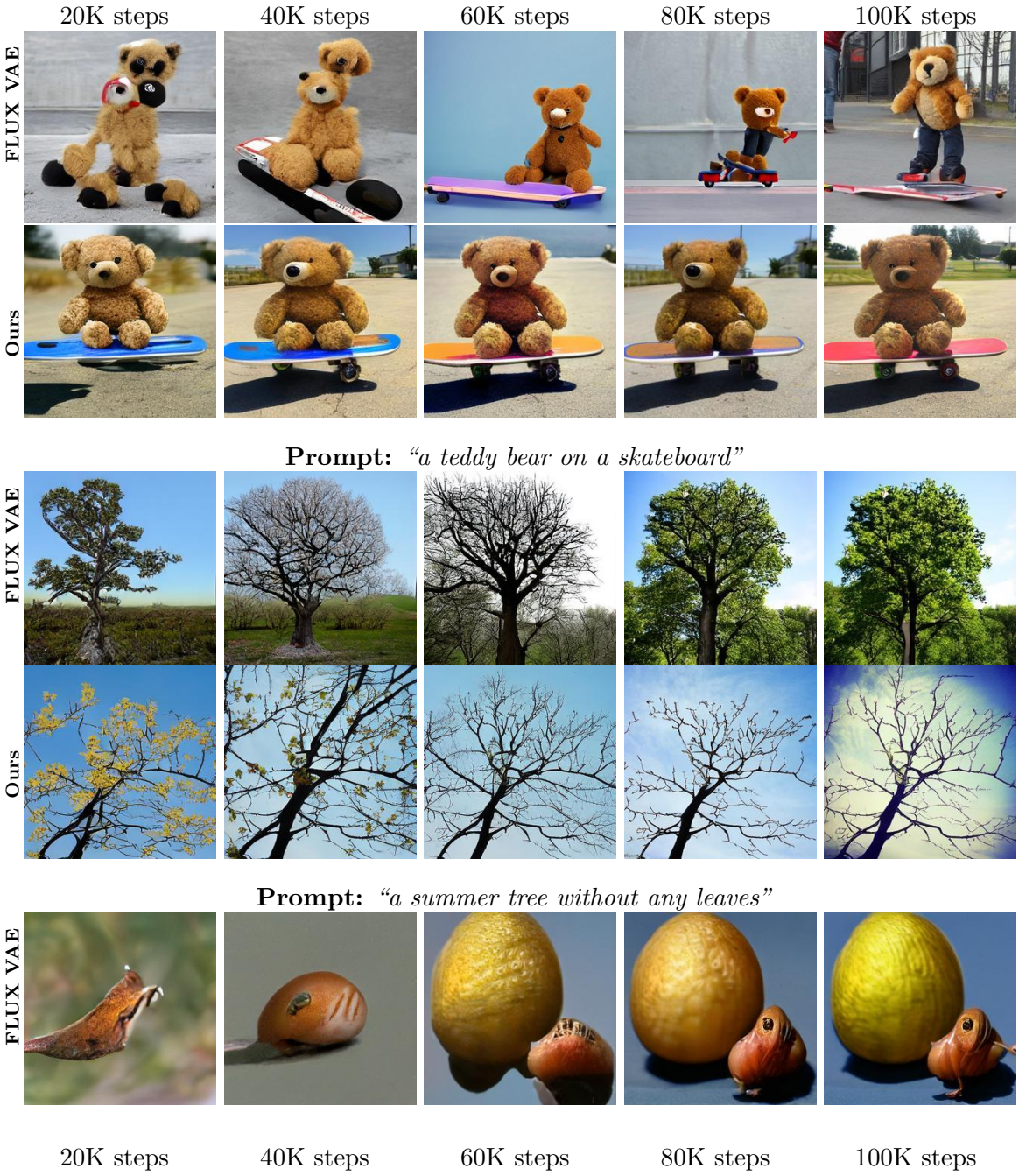


Figure 7.15: **Qualitative Comparison of Convergence Speed on Text-to-Image Generation.** We compare with FLUX VAE. Results are reported with CFG scale set to 5, using images generated at 256×256 resolution.

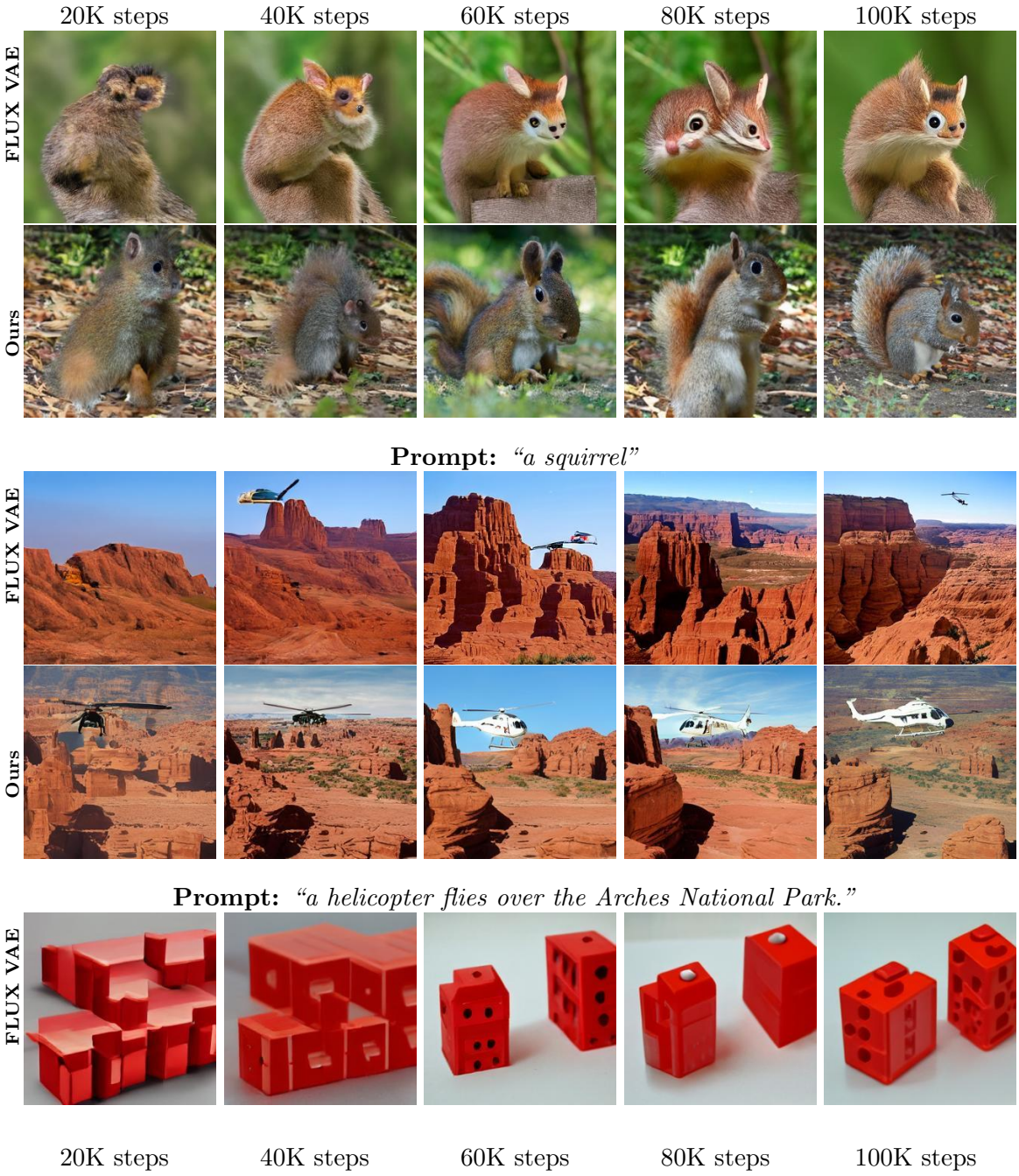


Figure 7.16: **Qualitative Comparison of Convergence Speed on Text-to-Image Generation.** We compare with FLUX VAE. Results are reported with CFG scale set to 5, using images generated at 256×256 resolution.

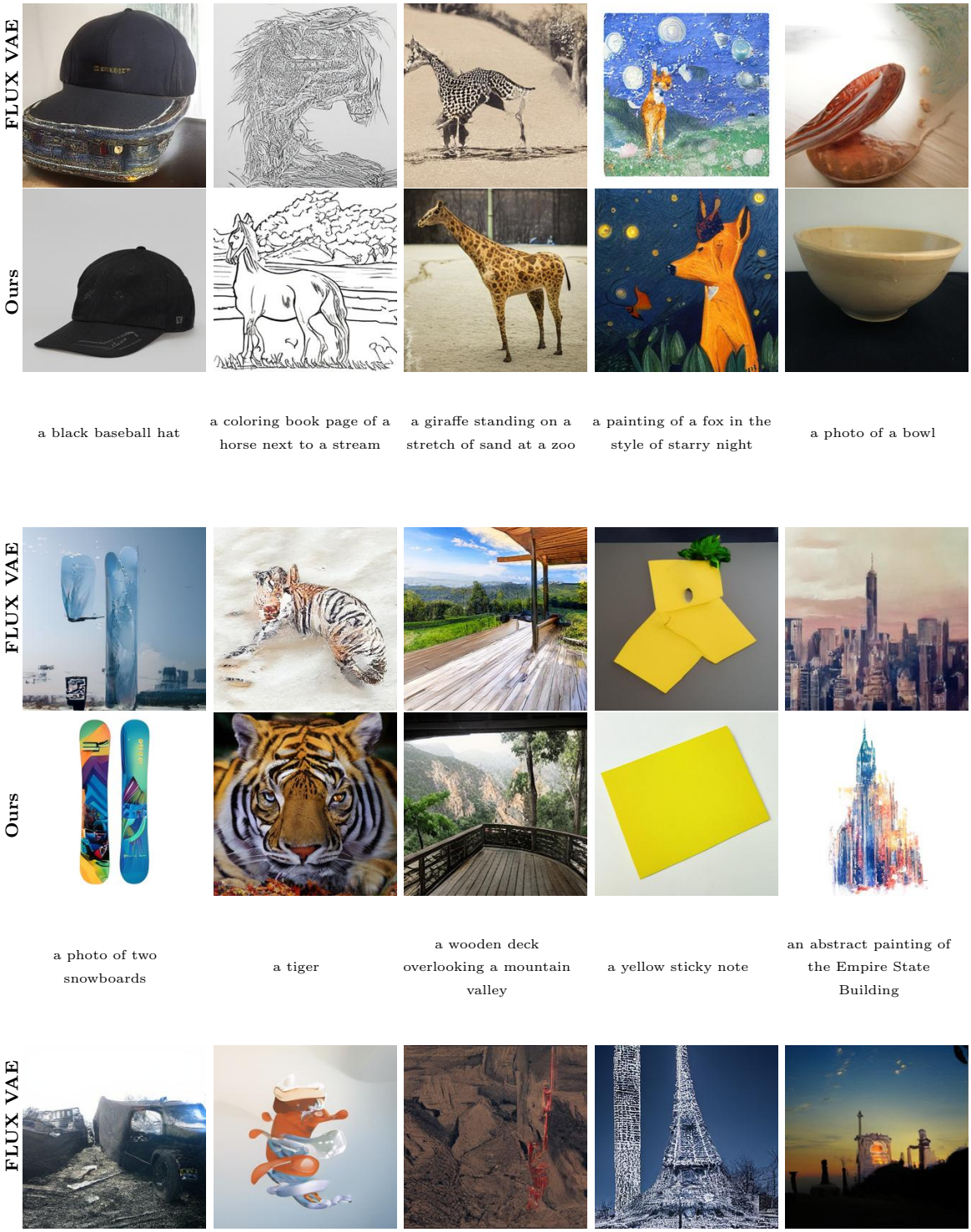


Figure 7.17: **Qualitative Comparison on Text-to-Image Generation with FLUX VAE (without CFG).** Input text prompts are shown below the images and results (256×256 resolution) are generated from 2B-parameter diffusion models trained for 100K steps, without using CFG.

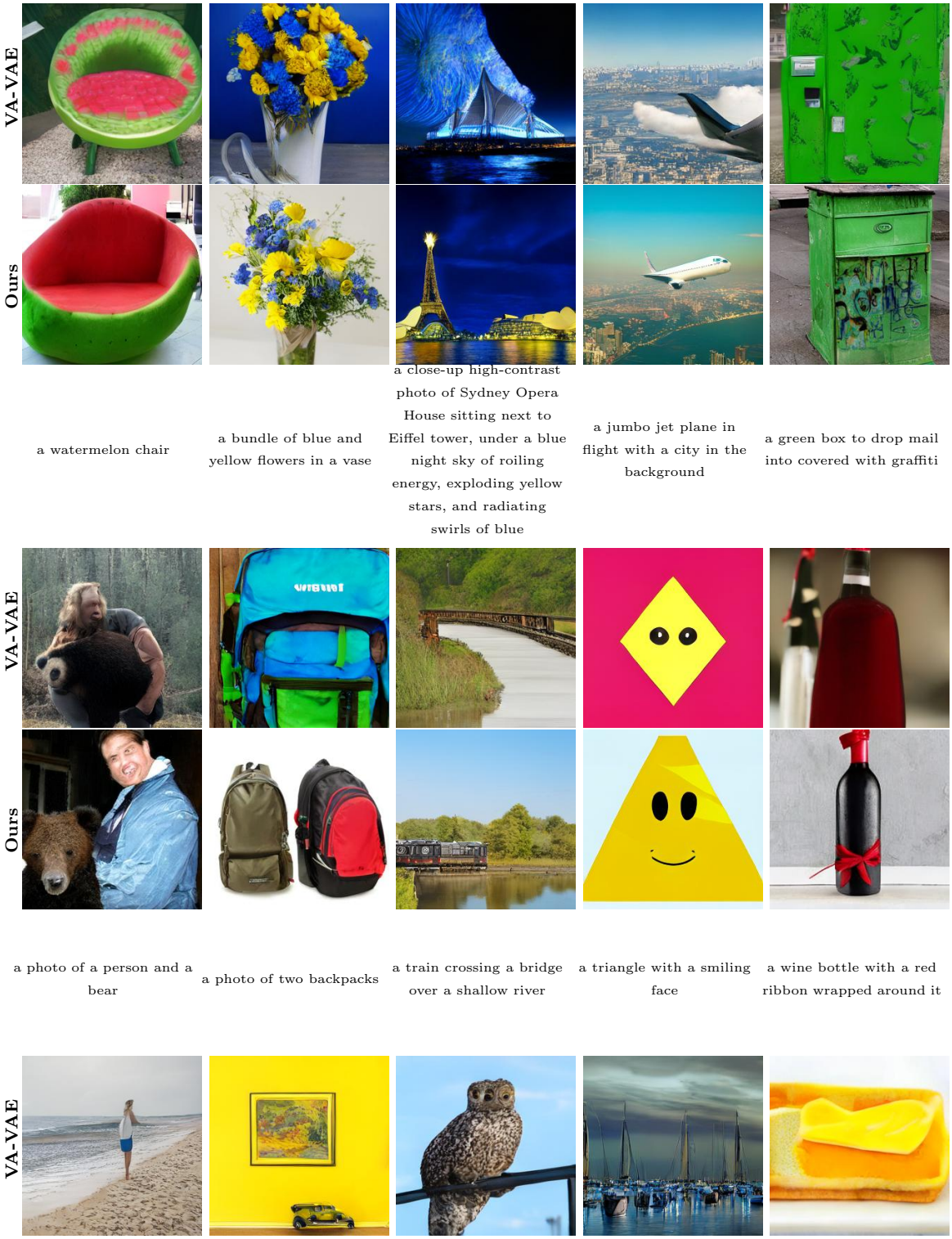


Figure 7.18: **Qualitative Comparison on Text-to-Image Generation with VA-VAE.** Input text prompts are shown below the images and results (256×256 resolution) are generated from 1B-parameter diffusion models trained for 50K steps, using CFG scale of 5. Note that, both our tokenizer and VA-VAE are trained on ImageNet.

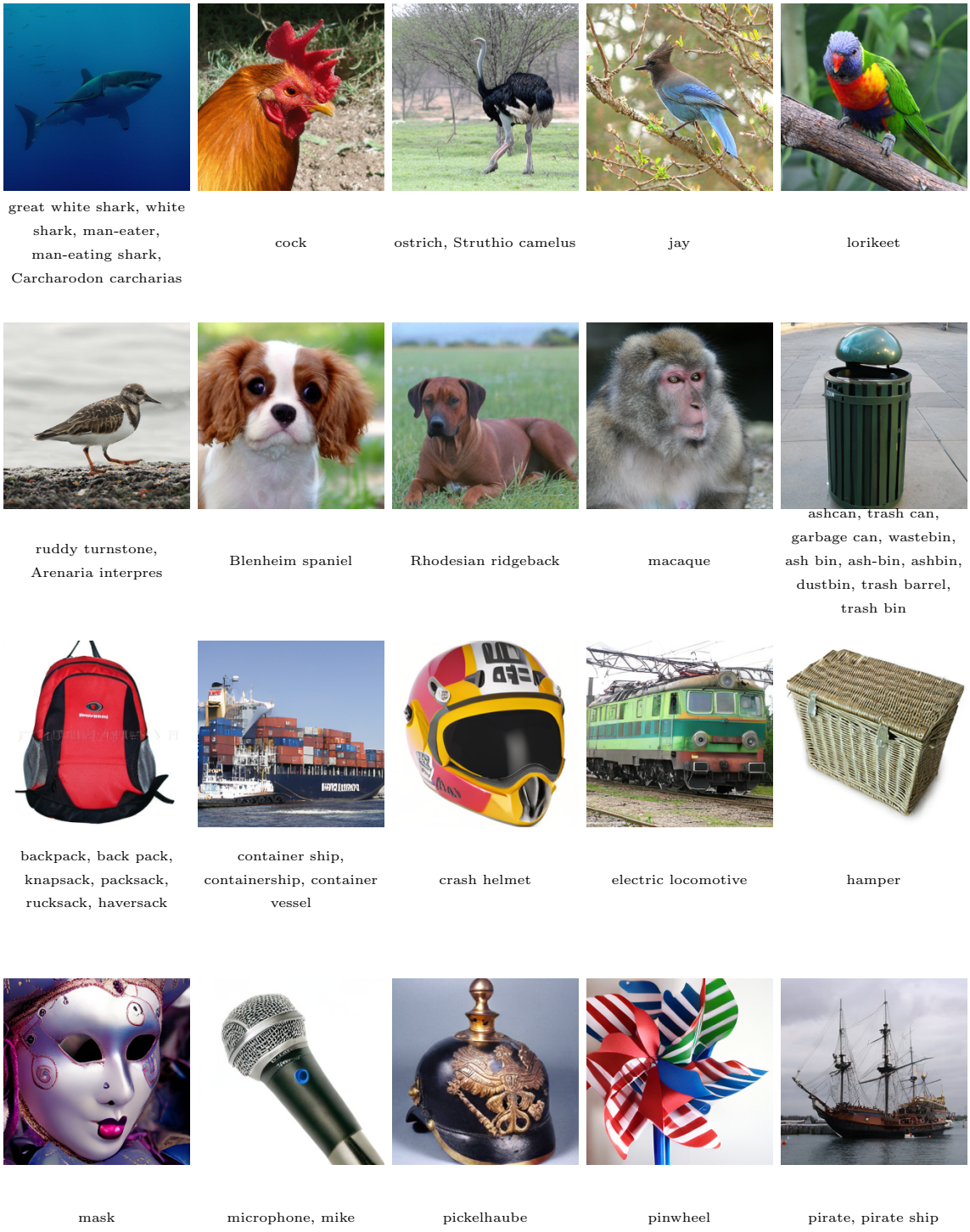


Figure 7.19: **Qualitative Results of Our Method on ImageNet Class-Conditional Generation.** ImageNet class names are shown below the images and results are from diffusion models trained with 800 epochs. We set the CFG to 5 and apply it to all latent channels.

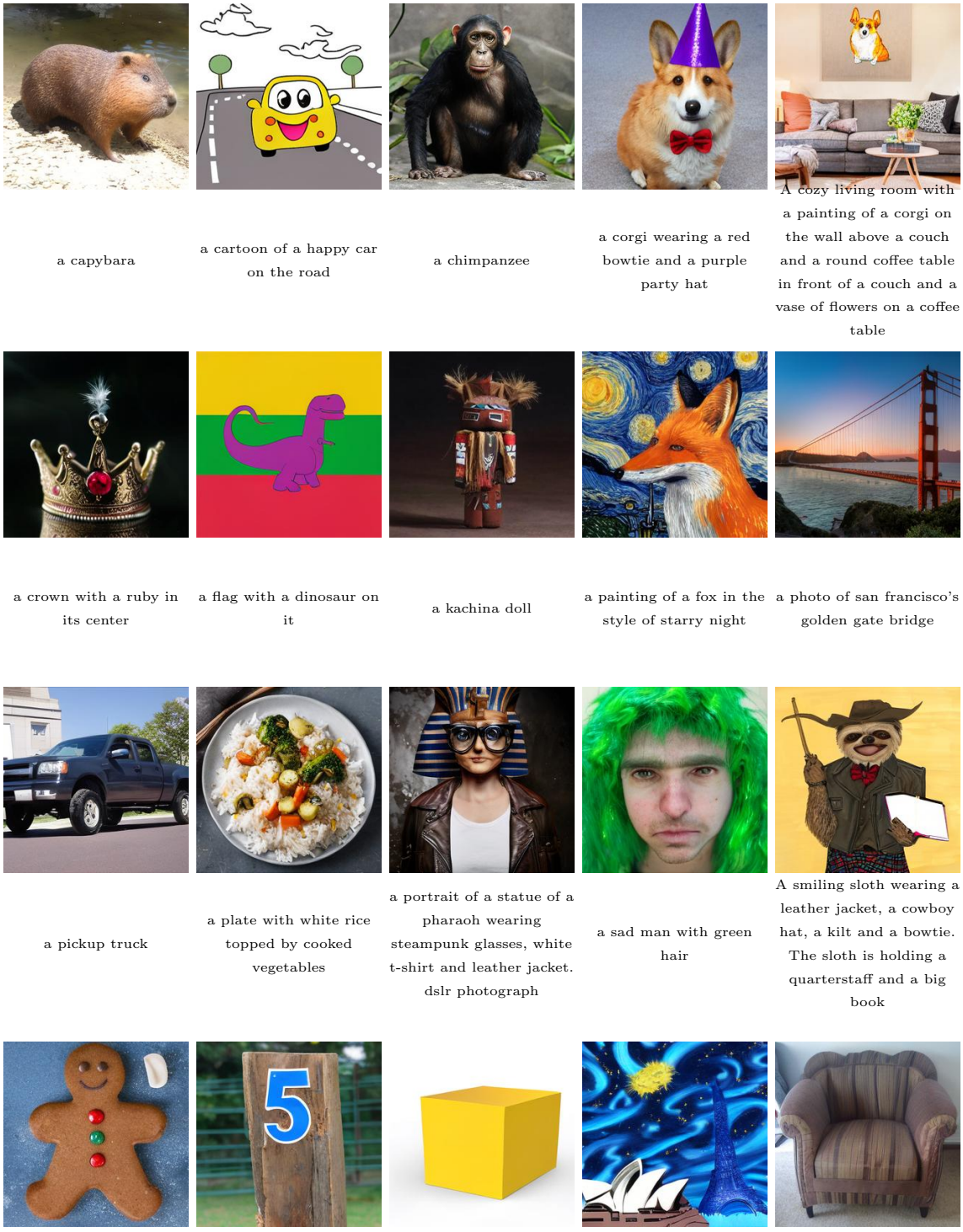


Figure 7.20: **Qualitative Results of Our Method on Text-to-Image Generation at 256×256 Resolution.** The input text prompts are shown below the images. Results are obtained from generative models trained for 200K steps.

a lemon character wearing sunglasses on the beach

a little boy flying through space eating pizza and cheese among candy planets in a comic book style drawing

a painting of a pokemon resembling a phone booth wearing clothes walking on two legs in a cobblestone street in a magical city

a plant with orange flowers shaped like stars

a plushy tired owl sits on a pile of antique books in a humorous illustration

SpongeBob in Dragon Ball style

assortment of colorful flowers in glass vase on table

an elephant is standing outside in the dirt

a young boy wearing a suit and smiling

a yellow wall with a large framed oil painting of a car

a West Highland white terrier holding a "Hug me!" sign

a truck that has spray paint on it

Figure 7.21: **Qualitative Results of Our Method on Text-to-Image Generation at 512×512 Resolution.** The input text prompts are shown below the images. Results are obtained from generative models trained for 290K steps.

A blue-haired girl with soft features stares directly at the camera in an extreme close-up Instagram picture

A cute anthropomorphic bear knight wearing a cape and crown in pale blue armor

A dog wearing a business suit smoking a cigar in a cinematic style

A fox wearing a yellow dress

A man carrying a yellow container filled with lemons

a store front that has the word 'openai' written on it

Multiple baskets of recently picked grapefruits on display

the word 'START' written above the word 'SMILING'

a toy car in front of a teddy bear

a wooden toy horse with a mane made of rope

A bike rider traveling down a road, in the desert

a crowd of people watching fireworks by a park

Figure 7.22: **Qualitative Results of Our Method on Text-to-Image Generation at 512×512 Resolution.** The input text prompts are shown below the images. Results are obtained from generative models trained for 290K steps.

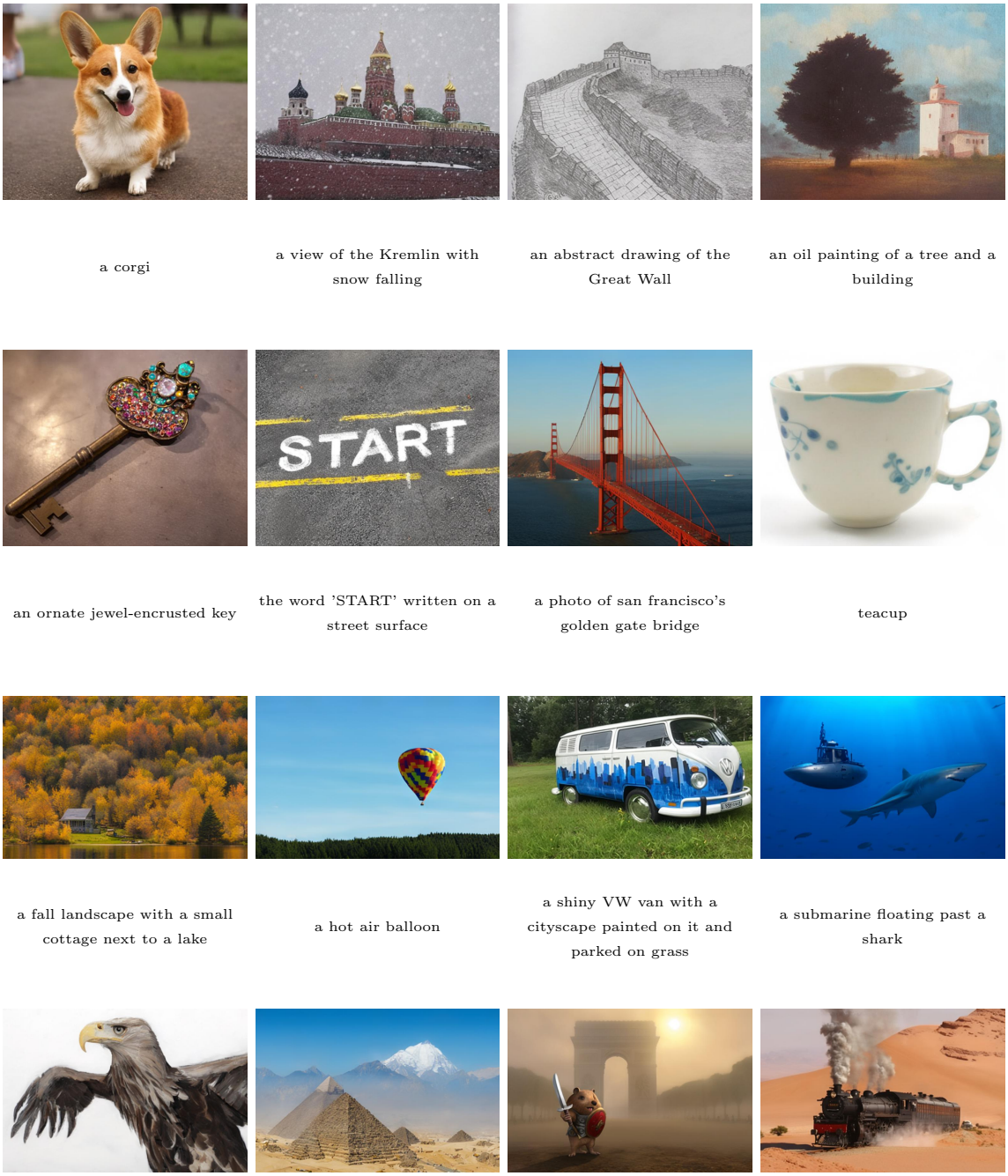


Figure 7.23: **Qualitative Results of Our Method on Text-to-Image Generation at Various Aspect Ratios with the Shortest Edge Equal to 512.** The input text prompts are shown below the images. Results are obtained from generative models trained for 290K steps. Rows 1 and 2 show 512 × 640 (3:4), row 3 and 4 show 512 × 768 (2:3), and row 5 shows 512 × 910 (9:16).

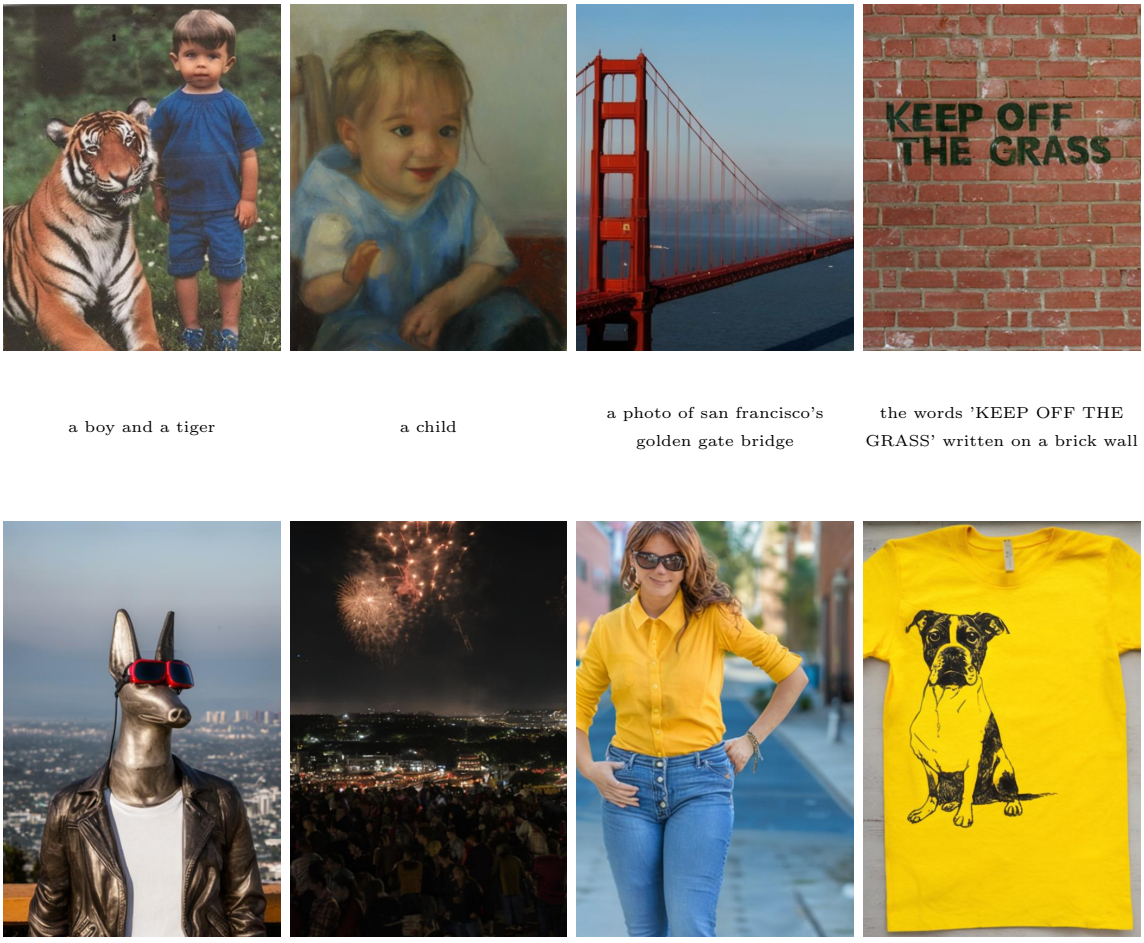


Figure 7.24: **Qualitative Results of Our Method on Text-to-Image Generation at Various Aspect Ratios with the Shortest Edge Equal to 512.** The input text prompts are shown below the images. Results are obtained from generative models trained for 290K steps. Rows 1 and 2 show 640 × 512 (4:3), rows 3 and 4 show 768 × 512 (3:2), and row 5 shows 910 × 512 (16:9).

Chapter 8

CONCLUSION

Across the projects presented in this thesis, a common challenge emerges: many compelling visual generation applications must be solved from observations that are fundamentally incomplete. Whether reconstructing full human appearance from a handful of selfies, editing portraits while preserving identity, or inferring a plausible painting process from a single final artwork, the available inputs underspecify the desired output. These tasks are inherently ill-posed, requiring models to hallucinate missing appearance, structure, or process while remaining consistent with limited evidence.

The results of Chapters 3–6 demonstrate that such applications are not beyond the reach of pretrained diffusion models, but they cannot be addressed by these models in their standard form. Direct application or naive fine-tuning consistently fails due to mismatches in input structure, supervision, and task requirements. Instead, success depends on carefully designed system-level interventions. By reformulating tasks, introducing weakly aligned or synthetic supervision, and composing pretrained models into cascaded pipelines, the proposed methods enable diffusion models to operate effectively in regimes far from their original training distribution. These designs make it possible to reconstruct globally coherent appearance, preserve identity under selective edits, and recover plausible generative processes, even when paired data is scarce or entirely absent.

A key empirical insight from these applications is that repurposing occurs most effectively at the interface between the model and the task. Pretrained diffusion models are not asked to directly solve ill-posed problems; rather, the surrounding pipeline reshapes the problem into a form compatible with the model's learned priors. This perspective reframes many challenging visual generation problems as system-design problems, where appropriate decomposition and supervision can unlock capabilities already latent in pretrained models.

At the same time, the the effectiveness of the methods depends critically on the quality

and flexibility of the visual priors learned during pretraining. Stronger base models consistently yield better identity preservation, robustness to distribution shift, and perceptual quality, while weaker priors amplify artifacts and failure modes. This observation motivates the final contribution of the thesis, AlignTok (Chapter 7), which addresses repurposing at a more fundamental level by improving the priors themselves. By repurposing pretrained visual foundation encoders as tokenizers for diffusion models, AlignTok enables diffusion to operate in a semantically rich latent space, leading to improved training efficiency and stronger representations for downstream adaptation.

Taken together, the contributions of this thesis suggest that partial-observation visual generation problems can be addressed through repurposing at multiple levels. System- and pipeline-level designs expand the range of applications that pretrained diffusion models can support under challenging data regimes, while prior-level repurposing improves the representational foundations that make such adaptation reliable. These approaches are complementary: stronger priors reduce the burden on system design, while well-structured pipelines extend the practical reach of even imperfect models. Viewed in this light, the thesis advances both practical methods and conceptual understanding for deploying pretrained diffusion models in complex, real-world visual generation settings where complete observation is the exception rather than the norm.

8.1 Future Work

Building on the findings of this thesis, several promising directions emerge for future research.

8.1.1 Future Directions for Selfie-Based Appearance Reconstruction

The selfie-based applications studied in this thesis, including Total Selfie and Fit Check Video Generation, highlight both the promise and the limitations of reconstructing full human appearance from sparse, user-captured inputs. A natural direction for future work is to improve robustness under even more constrained capture conditions, such as a single selfie, severe occlusions, or challenging clothing and lighting variations. Addressing these

cases may require stronger geometric reasoning, improved pose and body-shape priors, or tighter integration between diffusion models and parametric human representations.

8.1.2 Future Directions for Fit Check Video Generation

Fit check video generation opens several avenues for future research beyond the current formulation. One direction is to expand the diversity and controllability of generated motions, enabling users to specify motion styles, camera trajectories, or interaction with the environment. Achieving this reliably will likely require tighter coupling between motion priors and appearance synthesis, as well as improved disentanglement between body motion, camera motion, and clothing deformation.

Another promising direction is improved physical plausibility. While diffusion models can synthesize visually convincing motion, enforcing constraints related to contact, gravity, and clothing dynamics remains difficult. Incorporating lightweight physics priors or learned physical constraints into the generation pipeline could improve realism, especially for longer sequences or more complex motions.

8.1.3 Future Directions for Feature-Preserving Portrait Editing

For feature-preserving portrait editing, future work could focus on expanding the range of editable attributes while maintaining strict identity preservation. As the number of editable factors increases, controlling unintended entanglement between attributes becomes increasingly challenging. Developing more explicit representations of editable versus non-editable features, or introducing stronger disentanglement objectives during training, could improve robustness.

Another direction is interactive and iterative editing. While the current approach supports single-step edits, real-world use cases often involve multiple rounds of refinement. Enabling consistent multi-step editing without cumulative identity drift remains an open problem, and may require memory mechanisms or explicit constraints that preserve previously fixed attributes across editing iterations.

8.1.4 Future Directions for Inverse Painting and Process Reconstruction

Inverse Painting represents an early step toward reconstructing generative processes from final visual outcomes. A natural extension is to broaden the class of processes that can be reconstructed, including other artistic media such as watercolor, charcoal, or digital illustration, each of which exhibits distinct structural and temporal characteristics. Extending the framework to handle these media would require richer representations of tool usage, layering, and material interaction.

More broadly, inverse painting raises fundamental questions about the identifiability of generative processes from final outcomes. Future work could explore uncertainty-aware generation, producing multiple plausible process hypotheses rather than a single reconstruction, and studying how users interpret and interact with these alternatives.

8.1.5 Leveraging Video Priors for Image-Based Tasks

One intriguing direction is to more deeply exploit video priors for image editing and reconstruction tasks, particularly when paired training data is scarce. For instance, in Total Selfie, a set of input selfies can be reinterpreted as the initial frames of a video sequence. By conditioning a video generation model on a virtual camera motion that gradually reveals the subject, the system could synthesize a coherent full-body sequence with strong appearance consistency, leveraging the temporal priors learned by large video diffusion models. Such approaches blur the boundary between image and video generation, suggesting that video priors may serve as a powerful regularizer even for fundamentally image-based tasks.

8.1.6 Accelerating Diffusion-Based Pipelines

Despite their strong visual fidelity, diffusion-based models remain computationally expensive due to their iterative denoising process. This inference latency poses a major challenge for interactive or on-device applications. Recent advances in diffusion model distillation offer a promising path forward, enabling multi-step diffusion models to be distilled into few-step or even single-step generators while preserving perceptual quality. Applying these techniques to task-specific diffusion pipelines—such as inverse painting or fit check video generation—

could dramatically reduce inference time. However, ensuring robustness and generalization across diverse conditioning signals and pipeline designs remains an open problem.

8.1.7 Toward More General Repurposing of Pretrained Models

This thesis opens broader questions about how pretrained models can be repurposed beyond individual applications. While AlignTok explores repurposing at the tokenizer level to improve visual priors, future work could investigate how other pretrained models can be systematically integrated into diffusion pipelines. Developing principled frameworks for such reuse could further reduce the reliance on task-specific data and enable faster adaptation to new visual generation problems.

8.1.8 Combining High-Dimensional Latent Diffusion with Aligned Tokenizers

A promising future direction is to combine recent advances in high-dimensional latent diffusion models with the alignment-based tokenizer proposed in this thesis. While the current approach compresses pretrained visual features into a low-dimensional latent space to ensure stable diffusion training, this compression inevitably limits representational capacity. Historically, diffusion models were believed to operate reliably only on low-dimensional latents, which motivated this design choice.

Recent work such as RAE [334] demonstrates that diffusion models can be trained directly on high-dimensional latent spaces, challenging this long-standing assumption. Operating in a higher-dimensional latent space provides increased representational capacity and has the potential to improve both reconstruction fidelity and semantic expressiveness.

I see strong complementarity between RAE and AlignTok. RAE focuses on enabling diffusion in high-dimensional latent spaces, whereas AlignTok emphasizes semantic alignment and representation quality through encoder adaptation. Integrating high-dimensional latent diffusion with the alignment-based training strategy explored in this thesis could lead to tokenizers that simultaneously offer high representational capacity and strong semantic structure, further improving the adaptability and performance of diffusion models across diverse visual generation tasks.

8.1.9 Simplifying Data Capture

One important direction is to further reduce the complexity of the data capture process required by current systems. For example, the Total Selfie pipeline relies on multiple selfie inputs to reconstruct full-body appearance. A natural extension is to explore whether a single input image could suffice, potentially by leveraging wide field-of-view cameras or capturing selfies from tilted or oblique angles. Such designs could allow a single capture to encode richer geometric cues, reducing user burden while maintaining reconstruction quality. More broadly, this direction aligns with the goal of making repurposed diffusion pipelines accessible to casual users in everyday scenarios.

BIBLIOGRAPHY

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4432–4441, 2019.

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8296–8305, 2020.

[3] AhamArt. Aham art channel. `https://www.youtube.com/channel/UC2n_iZPXdEN59Vs4aHIkcQA`, 2024.

[4] AhmadArt. Ahmadart channel. `https://www.youtube.com/@AhmadArt/videos`, 2024.

[5] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18511–18521, 2022.

[6] Allison Anderson. 10 tips for photographing yourself with a tripod. `https://hertravelstyle.com/photography/2018/6/4/10-tips-for-photographing-yourself-with-a-tripod`, 2018. Blog post on Her Travel Style.

[7] Allison Anderson. How i take travel photos of myself (with a tripod). `https://www.youtube.com/watch?v=1WiSs_3hJWM`, 2018. YouTube video.

[8] ShahRukh Athar, Zhixin Shu, and Dimitris Samaras. Flame-in-nerf: Neural control of radiance fields for free view face animation. In *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*, pages 1–8. IEEE, 2023.

[9] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *arXiv preprint arXiv:2206.02779*, 2022.

[10] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022.

[11] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[12] Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, et al. Lumiere: A space-time diffusion model for video generation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[13] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *arXiv preprint arXiv:2005.07727*, 2020.

[14] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023.

[15] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Jorma Laaksonen, Mubarak Shah, and Fahad Shahbaz Khan. Person image synthesis via denoising diffusion model. *arXiv preprint arXiv:2211.12500*, 2022.

[16] Jia-Wang Bian, Huangying Zhan, and Ian Reid. Nvss: High-quality novel view selfie synthesis. In *2021 International Conference on 3D Vision (3DV)*, pages 1085–1094. IEEE, 2021.

[17] Ioana Bica, Anastasija Ilić, Matthias Bauer, Goker Erdogan, Matko Bošnjak, Christos Kaplanis, Alexey A Gritsenko, Matthias Minderer, Charles Blundell, Razvan Pascanu, et al. Improving fine-grained understanding in image-text pre-training. *arXiv preprint arXiv:2401.09865*, 2024.

[18] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[19] Reiner Birkl, Diana Wofk, and Matthias Müller. Midas v3.1 – a model zoo for robust monocular relative depth estimation. *arXiv preprint arXiv:2307.14460*, 2023.

[20] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[21] Ross Bob. *Bob Ross Experience The Joy of Painting Volume X 10 Ten.* Bob Ross; First Edition, 1987.

[22] Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, et al. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv preprint arXiv:2504.13181*, 2025.

[23] Cristian Botezatu, Mathias Ibsen, Christian Rathgeb, and Christoph Busch. Fun selfie filters in face recognition: Impact assessment and removal. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 5(1):91–104, 2022.

[24] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18392–18402, June 2023.

[25] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

[26] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[27] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[28] Nicola Capece, Francesco Banterle, Paolo Cignoni, Fabio Ganovelli, Roberto Scopigno, and Ugo Erra. Deepflash: Turning a flash selfie into a studio portrait. *Signal Processing: Image Communication*, 77:28–39, 2019.

[29] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5933–5942, 2019.

[30] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *arXiv*, 2021.

[31] Di Chang, Yichun Shi, Quankai Gao, Jessica Fu, Hongyi Xu, Guoxian Song, Qing Yan, Xiao Yang, and Mohammad Soleymani. Magicdance: Realistic human dance video generation with motions & facial expressions transfer. *CoRR*, 2023.

[32] Bowei Chen, Sai Bi, Hao Tan, He Zhang, Tianyuan Zhang, Zhengqi Li, Yuanjun Xiong, Jianming Zhang, and Kai Zhang. Aligning visual foundation encoders to tokenizers for diffusion models. *arXiv preprint arXiv:2509.25162*, 2025.

[33] Bowei Chen, Brian Curless, Ira Kemelmacher-Shlizerman, and Steve Seitz. Total selfie: Generating full-body selfies. *arXiv preprint arXiv:2308.14740*, 2023.

[34] Bowei Chen, Brian Curless, Ira Kemelmacher-Shlizerman, and Steven M Seitz. Total

selfie: Generating full-body selfies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6701–6711, 2024.

[35] Bowei Chen, Brian Curless, Ira Kemelmacher-Shlizerman, and Steven M Seitz. Generating fit check videos with a handheld camera. *arXiv preprint arXiv:2505.23886*, 2025.

[36] Bowei Chen, Yifan Wang, Brian Curless, Ira Kemelmacher-Shlizerman, and Steven M Seitz. Inverse painting: Reconstructing the painting process. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[37] Bowei Chen, Tiancheng Zhi, Peihao Zhu, Shen Sang, Jing Liu, and Linjie Luo. Learning feature-preserving portrait editing from generated pairs. *arXiv preprint arXiv:2407.20455*, 2024.

[38] Chieh-Yun Chen, Yi-Chung Chen, Hong-Han Shuai, and Wen-Huang Cheng. Size does matter: Size-aware virtual try-on via clothing-oriented transformation try-on network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7513–7522, 2023.

[39] Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Juncheng Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengchen Ma, et al. Skyreels-v2: Infinite-length film generative model. *arXiv preprint arXiv:2504.13074*, 2025.

[40] Hao Chen, Yujin Han, Fangyi Chen, Xiang Li, Yidong Wang, Jindong Wang, Ze Wang, Zicheng Liu, Difan Zou, and Bhiksha Raj. Masked autoencoders are effective tokenizers for diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

[41] Haoxin Chen, Yong Zhang, Xiaodong Cun, Menghan Xia, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7310–7320, 2024.

[42] Jianqi Chen, Yilan Zhang, Zhengxia Zou, Keyan Chen, and Zhenwei Shi. Dense pixel-to-pixel harmonization via continuous image representation. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2023.

[43] Junyang Chen, Xiaoyu Xian, Zhijing Yang, Tianshui Chen, Yongyi Lu, Yukai Shi, Jinshan Pan, and Liang Lin. Open-world pose transfer via sequential test-time adaption. *arXiv preprint arXiv:2303.10945*, 2023.

[44] Junyu Chen, Dongyun Zou, Wenkun He, Junsong Chen, Enze Xie, Song Han, and Han Cai. Dc-ae 1.5: Accelerating diffusion model convergence with structured latent space. *arXiv preprint arXiv:2508.00413*, 2025.

[45] Zisheng Chen, Chunwei Wang, Xiuwei Chen, Hongbin Xu, Runhui Huang, Jun Zhou, Jianhua Han, Hang Xu, and Xiaodan Liang. Semhitok: A unified image tokenizer via semantic-guided hierarchical codebook for multimodal understanding and generation. *arXiv preprint arXiv:2503.06764*, 2025.

[46] Seunghwan Choi, Sunghyun Park, Minsoo Lee, and Jaegul Choo. Viton-hd: High-resolution virtual try-on via misalignment-aware normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14131–14140, 2021.

[47] Zheng Chong, Xujie Zhang, Fuwei Zhao, Zhenyu Xie, and Xiaodan Liang. Fashion matrix: Editing photos by just talking. *arXiv preprint arXiv:2307.13240*, 2023.

[48] Vasileios Choutas, Lea Muller, Chun-Hao P. Huang, Siyu Tang, Dimitris Tzionas, and Michael J. Black. Accurate 3d body shape regression using metric and semantic attribute. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[49] Xiangxiang Chu, Renda Li, and Yong Wang. Usp: Unified self-supervised pretraining for image generation and understanding. *arXiv preprint arXiv:2503.06132*, 2025.

[50] FaceFusion Contributors. Facefusion: Ai-powered face swapping. `https://github.com/facefusion/facefusion`, 2024.

[51] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022.

[52] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[53] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[54] Aiyu Cui, Daniel McKee, and Svetlana Lazebnik. Dressing in order: Recurrent person image generation for pose transfer, virtual try-on and outfit editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14638–14647, October 2021.

[55] Datou. Mimic motion (animate anyone killer) — comfyui workflow. `https://openart.ai/workflows/datou/mimic-motion-animate-anyone-killer/RBcspPWvu8RCQB5ev031`, 2024.

[56] Manuel Ladron de Guevara, Matthew Fisher, and Aaron Hertzmann. Segmentation-based parametric painting. *arXiv preprint arXiv:2311.14271*, 2023.

[57] Google DeepMind. Veo 3.1 — video meets audio: advanced creative controls. `https://deepmind.google/models/veo/`, 2025. Accessed: YYYY-MM-DD.

[58] Chaorui Deng, Deyao Zhu, Kunchang Li, Shi Guang, and Haoqi Fan. Causal diffusion transformers for generative modeling. *arXiv preprint arXiv:2412.12095*, 2024.

[59] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[60] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[61] Yanbo Ding. Mtvcrafter: 4d motion tokenization for open-world human image animation. *arXiv preprint arXiv:2505.10238*, 2025.

[62] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[63] Annette Dozier. *Painting peaceful country landscapes*. North Light Books, 2007.

[64] Yuming Du, Robin Kips, Albert Pumarola, Sebastian Starke, Ali Thabet, and Artsiom Sanakoyeu. Avatars grow legs: Generating smooth human motion from sparse tracking inputs with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 481–490, 2023.

[65] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[66] Haipeng Fang, Zhihao Sun, Ziyao Huang, Fan Tang, Juan Cao, and Sheng Tang. Dance your latents: Consistent dance generation through spatial-temporal subspace attention guided by motion flow. *arXiv preprint arXiv:2310.14780*, 2023.

[67] Kevin Frans and Chin-Yi Cheng. Unsupervised image to sequence translation with canvas-drawer networks. *arXiv preprint arXiv:1809.08340*, 2018.

[68] Anna Frühstück, Krishna Kumar Singh, Eli Shechtman, Niloy J Mitra, Peter Wonka, and Jingwan Lu. Insetgan for full-body image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7723–7732, 2022.

[69] Hongbo Fu, Shizhe Zhou, Ligang Liu, and Niloy J Mitra. Animated construction of line drawings. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 1–10, 2011.

[70] Jianglin Fu, Shikai Li, Yuming Jiang, Kwan-Yee Lin, Chen Qian, Chen Change Loy, Wayne Wu, and Ziwei Liu. Stylegan-human: A data-centric odyssey of human generation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVI*, pages 1–19. Springer, 2022.

[71] Jianglin Fu, Shikai Li, Yuming Jiang, Kwan-Yee Lin, Wayne Wu, and Ziwei Liu. Unitedhuman: Harnessing multi-source data for high-resolution human generation. *arXiv preprint*, arXiv:2309.14335, 2023.

[72] Kent Fujiwara, Mikihiro Tanaka, and Qing Yu. Chronologically accurate retrieval for temporal grounding of motion-language models. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024.

[73] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.

[74] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022.

[75] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *arXiv preprint arXiv:2108.00946*, 2021.

[76] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. In *International Conference on Machine Learning*, pages 1666–1675. PMLR, 2018.

[77] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer, 2023.

[78] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Mdtv2: Masked diffusion transformer is a strong image synthesizer. *arXiv preprint arXiv:2303.14389*, 2023.

[79] Songwei Ge, Aniruddha Mahapatra, Gaurav Parmar, Jun-Yan Zhu, and Jia-Bin Huang. On the content bias in fréchet video distance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[80] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152, 2023.

[81] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[82] Google DeepMind. Gemini 2.5 flash image: State-of-the-art image generation and editing model. Technical report, Google, 2025. Formerly known as "Nano Banana".

[83] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5152–5161, June 2022.

[84] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning, 2023.

[85] David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.

[86] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

[87] Paul Haeberli. Paint by numbers: Abstract image representations. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 207–214, 1990.

[88] Jiaming Han, Hao Chen, Yang Zhao, Hanyu Wang, Qi Zhao, Ziyan Yang, Hao He, Xiangyu Yue, and Lu Jiang. Vision as a dialect: Unifying visual understanding and generation via text-aligned representations. *arXiv preprint arXiv:2506.18898*, 2025.

[89] Ligong Han, Yinxiao Li, Han Zhang, Peyman Milanfar, Dimitris Metaxas, and Feng Yang. Svdiff: Compact parameter space for diffusion fine-tuning. *arXiv preprint arXiv:2303.11305*, 2023.

[90] Xiao Han, Xiatian Zhu, Jiankang Deng, Yi-Zhe Song, and Tao Xiang. Controllable person image synthesis with pose-constrained latent diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22768–22777, 2023.

[91] Philippe Hansen-Estruch, David Yan, Ching-Yao Chung, Orr Zohar, Jialiang Wang, Tingbo Hou, Tao Xu, Sriram Vishwanath, Peter Vajda, and Xinlei Chen. Learnings from scaling visual tokenizers for reconstruction and generation. *arXiv preprint arXiv:2501.09755*, 2025.

[92] Zhao Haoyu, Qi Zhongang, Wang Cong, Zheng Qingping, Lu Guansong, Chen Fei, Xu Hang, and Wu Zuxuan. Dynamictrl: Rethinking the basic structure and the role of text for high-quality human image animation. *arXiv:2503.21246*, 2025.

[93] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *Advances in neural information processing systems*, 33:9841–9850, 2020.

[94] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick.

Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[95] Sen He, Yi-Zhe Song, and Tao Xiang. Style-based global appearance flow for virtual try-on. In *CVPR*, 2022.

[96] Xuanhua He, Quande Liu, Shengju Qian, Xin Wang, Tao Hu, Ke Cao, Keyu Yan, and Jie Zhang. Id-animator: Zero-shot identity-preserving human video generation. *arXiv preprint arXiv:2404.15275*, 2024.

[97] Zecheng He, Bo Sun, Felix Juefei-Xu, Haoyu Ma, Ankit Ramchandani, Vincent Cheung, Siddharth Shah, Anmol Kalia, Harihar Subramanyam, Alireza Zareian, et al. Imagine yourself: Tuning-free personalized image generation. *arXiv preprint arXiv:2409.13346*, 2024.

[98] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

[99] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460, 1998.

[100] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

[101] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

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

[103] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[104] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[105] 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. *arXiv preprint arXiv:2106.09685*, 2021.

[106] Li Hu, Xin Gao, Peng Zhang, Ke Sun, Bang Zhang, and Liefeng Bo. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. *arXiv preprint arXiv:2311.17117*, 2023.

[107] Li Hu, Xin Gao, Peng Zhang, Ke Sun, Bang Zhang, and Liefeng Bo. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. *arXiv preprint arXiv:2311.17117*, 2023.

[108] Li Hu, Guangyuan Wang, Zhen Shen, Xin Gao, Dechao Meng, Lian Zhuo, Peng Zhang, Bang Zhang, and Liefeng Bo. Animate anyone 2: High-fidelity character image animation with environment affordance. *arXiv preprint arXiv:2502.06145*, 2025.

[109] Shoukang Hu, Takuya Narihira, Kazumi Fukuda, Ryosuke Sawata, Takashi Shibuya, and Yuki Mitsufuji. Humangif: Single-view human diffusion with generative prior. *arXiv preprint arXiv:2502.12080*, 2025.

[110] Teng Hu, Ran Yi, Haokun Zhu, Liang Liu, Jinlong Peng, Yabiao Wang, Chengjie Wang, and Lizhuang Ma. Stroke-based neural painting and stylization with dynamically predicted painting region. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 7470–7480, 2023.

[111] Zhewei Huang, Wen Heng, and Shuchang Zhou. Learning to paint with model-based deep reinforcement learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8709–8718, 2019.

[112] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

[113] Yasamin Jafarian and Hyun Soo Park. Learning high fidelity depths of dressed humans by watching social media dance videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12753–12762, June 2021.

[114] Jitesh Jain, Jiachen Li, MangTik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One Transformer to Rule Universal Image Segmentation. *arXiv*, 2022.

[115] Rishabh Jain, Mayur Hemani, Duygu Ceylan, Krishna Kumar Singh, Jingwan Lu, Mausoom Sarkar, and Balaji Krishnamurthy. Umfuse: Unified multi view fusion for human editing applications. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7182–7191, 2023.

[116] JayLeePainting. Jay lee painting channel. `https://www.youtube.com/@Joonyart`, 2024.

[117] Biao Jia, Chen Fang, Jonathan Brandt, Byungmoon Kim, and Dinesh Manocha. Paintbot: A reinforcement learning approach for natural media painting. *arXiv preprint arXiv:1904.02201*, 2019.

[118] Yifeng Jiang, Yuting Ye, Deepak Gopinath, Jungdam Won, Alexander W Winkler, and C Karen Liu. Transformer inertial poser: Real-time human motion reconstruction from sparse imus with simultaneous terrain generation. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[119] Yuming Jiang, Shuai Yang, Haonan Qiu, Wayne Wu, Chen Change Loy, and Ziwei Liu. Text2human: Text-driven controllable human image generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–11, 2022.

[120] Yang Jiao, Haibo Qiu, Zequn Jie, Shaoxiang Chen, Jingjing Chen, Lin Ma, and Yu-Gang Jiang. Unitoken: Harmonizing multimodal understanding and generation through unified visual encoding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3600–3610, 2025.

[121] JoonyArt. Joony art channel. `https://www.youtube.com/c/Joonyart`, 2024.

[122] Zhang Kai, Wang Peng, Bi Sai, Zhang Jianming, and Xiong Yuanjun. Knapformer. `https://github.com/Kai-46/KnapFormer/`, 2025.

[123] Zhang Kai, Wang Peng, Bi Sai, Zhang Jianming, and Xiong Yuanjun. minfm. `https://github.com/Kai-46/minFM/`, 2025.

[124] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. Conerf: Controllable neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18623–18632, 2022.

[125] Johanna Karras, Aleksander Holynski, Ting-Chun Wang, and Ira Kemelmacher-Shlizerman. Dreampose: Fashion image-to-video synthesis via stable diffusion. *arXiv preprint arXiv:2304.06025*, 2023.

[126] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022.

[127] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

[128] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24174–24184, 2024.

[129] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[130] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for

generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[131] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020.

[132] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1867–1874, 2014.

[133] Kijai. Mimicmotion — human motion video generation. `https://www.runcomfy.com/comfyui-workflows/comfyui-mimicmotion-workflow-human-motion-video-generation`, 2024.

[134] Jeongho Kim, Min-Jung Kim, Junsoo Lee, and Jaegul Choo. Tcan: Animating human images with temporally consistent pose guidance using diffusion models. In *European Conference on Computer Vision*, pages 326–342. Springer, 2024.

[135] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[136] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. 2023.

[137] Markus Knoche, István Sárándi, and Bastian Leibe. Reposing humans by warping 3d features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1044–1045, 2020.

[138] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[139] Dmytro Kotovenko, Matthias Wright, Arthur Heimbrecht, and Bjorn Ommer. Rethinking style transfer: From pixels to parameterized brushstrokes. In *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12196–12205, 2021.

[140] Theodoros Kouzelis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. EQ-VAE: Equivariance regularized latent space for improved generative image modeling. In *Forty-second International Conference on Machine Learning*, 2025.

[141] Max Ku, Cong Wei, Weiming Ren, Harry Yang, and Wenhu Chen. Anyv2v: A tuning-free framework for any video-to-video editing tasks. *arXiv preprint arXiv:2403.14468*, 2024.

[142] Laxman Kumarapu, Shiv Ram Dubey, Snehasis Mukherjee, Parkhi Mohan, Sree Pragna Vinnakoti, and Subhash Karthikeya. Wsd: Wild selfie dataset for face recognition in selfie images. *arXiv preprint arXiv:2302.07245*, 2023.

[143] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1931–1941, 2023.

[144] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[145] Junho Lee, Jeongwoo Shin, Hyungwook Choi, and Joonseok Lee. Latent diffusion models with masked autoencoders. *arXiv preprint arXiv:2507.09984*, 2025.

[146] Sangyun Lee, Gyojung Gu, Sunghyun Park, Seunghwan Choi, and Jaegul Choo. High-resolution virtual try-on with misalignment and occlusion-handled conditions. In *European Conference on Computer Vision*, pages 204–219. Springer, 2022.

[147] Alexander Leiser and Tim Schlippe. Ai in art: simulating the human painting process. In *International Conference on ArtsIT, Interactivity and Game Creation*, pages 295–308. Springer, 2021.

[148] Bo Li, Kaitao Xue, Bin Liu, and Yu-Kun Lai. Bbdm: Image-to-image translation

with brownian bridge diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1952–1961, 2023.

[149] Nannan Li, Kevin J Shih, and Bryan A Plummer. Collecting the puzzle pieces: Disentangled self-driven human pose transfer by permuting textures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7126–7137, 2023.

[150] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[151] Yuanming Li, Youngsaeng Jin, Jeonggi Kwak, Dongsik Yoon, David Han, and Hanseok Ko. Adaptive content feature enhancement gan for multimodal selfie to anime translation. 2021.

[152] Zhe Li, Weihao Yuan, Yisheng He, Lingteng Qiu, Shenhao Zhu, Xiaodong Gu, Weichao Shen, Yuan Dong, Zilong Dong, and Laurence T Yang. Lamp: Language-motion pretraining for motion generation, retrieval, and captioning. *arXiv preprint arXiv:2410.07093*, 2024.

[153] Zhen Li, Mingdeng Cao, Xintao Wang, Zhongang Qi, Ming-Ming Cheng, and Ying Shan. Photomaker: Customizing realistic human photos via stacked id embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8640–8650, June 2024.

[154] Zhi Li, Pengfei Wei, Xiang Yin, Zejun Ma, and Alex C Kot. Virtual try-on with pose-garment keypoints guided inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22788–22797, 2023.

[155] Zhihao Li, Jianzhuang Liu, Zhensong Zhang, Songcen Xu, and Youliang Yan. Cliff: Carrying location information in full frames into human pose and shape estimation. In *ECCV*, 2022.

[156] Gaojie Lin, Jianwen Jiang, Jiaqi Yang, Zerong Zheng, Chao Liang, Yuan Zhang, and Jingtuo Liu. Omnihuman-1: Rethinking the scaling-up of one-stage conditioned

human animation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13847–13858, 2025.

[157] Haokun Lin, Teng Wang, Yixiao Ge, Yuying Ge, Zhichao Lu, Ying Wei, Qingfu Zhang, Zhenan Sun, and Ying Shan. Toklip: Marry visual tokens to clip for multimodal comprehension and generation. *arXiv preprint arXiv:2505.05422*, 2025.

[158] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[159] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. *arXiv preprint arXiv:2404.01291*, 2024.

[160] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[161] Peter Litwinowicz. Processing images and video for an impressionist effect. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 407–414, 1997.

[162] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[163] Jinlin Liu, Kai Yu, Mengyang Feng, Xiefan Guo, and Miaomiao Cui. Disentangling foreground and background motion for enhanced realism in human video generation. *arXiv preprint arXiv:2405.16393*, 2024.

[164] Lijie Liu, Tianxiang Ma, Bingchuan Li, Zhuowei Chen, Jiawei Liu, Gen Li, Siyu Zhou, Qian He, and Xinglong Wu. Phantom: Subject-consistent video generation via cross-modal alignment. *arXiv preprint arXiv:2502.11079*, 2025.

[165] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compo-

sitional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pages 423–439. Springer, 2022.

[166] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[167] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Ruifeng Deng, Xin Li, Errui Ding, and Hao Wang. Paint transformer: Feed forward neural painting with stroke prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6598–6607, 2021.

[168] Xiao-Chang Liu, Yu-Chen Wu, and Peter Hall. Painterly style transfer with learned brush strokes. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[169] Xingchao Liu, Chengyue Gong, and Qiang Liu. Rectified flow. `https://www.cs.utexas.edu/~lqiang/rectflow/html/intro.html`, 2022.

[170] Xingchao Liu, Chengyue Gong, and Qiang Liu. Rectified flow. `https://rectifiedflow.github.io`, 2022.

[171] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[172] Jiasen Lu, Liangchen Song, Mingze Xu, Byeongjoo Ahn, Yanjun Wang, Chen Chen, Afshin Dehghan, and Yinfei Yang. Atoken: A unified tokenizer for vision, 2025.

[173] Zhengyao Lv, Xiaoming Li, Xin Li, Fu Li, Tianwei Lin, Dongliang He, and Wangmeng Zuo. Learning semantic person image generation by region-adaptive normalization. 2021.

[174] Chuofan Ma, Yi Jiang, Junfeng Wu, Jihan Yang, Xin Yu, Zehuan Yuan, Bingyue Peng, and Xiaojuan Qi. Unitok: A unified tokenizer for visual generation and understanding. *arXiv preprint arXiv:2502.20321*, 2025.

[175] Liqian Ma, Zhe Lin, Connelly Barnes, Alexei A Efros, and Jingwan Lu. Unselfie: Translating selfies to neutral-pose portraits in the wild. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pages 156–173. Springer, 2020.

[176] Liyuan Ma, Tingwei Gao, Haitian Jiang, Haibin Shen, and Kejie Huang. Waveipt: Joint attention and flow alignment in the wavelet domain for pose transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7215–7225, 2023.

[177] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pages 23–40. Springer, 2024.

[178] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, pages 5442–5451, October 2019.

[179] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

[180] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[181] Byeon Minwoo, Park Beomhee, Kim Haecheon, Lee Sungjun, Baek Woonhyuk, and Kim Saehoon. Coyo-700m: Image-text pair dataset. `https://github.com/kakaobrain/coyo-dataset`, 2022.

[182] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6038–6047, 2023.

[183] Vimal Mollyn, Riku Arakawa, Mayank Goel, Chris Harrison, and Karan Ahuja. Imuposer: Full-body pose estimation using imus in phones, watches, and earbuds. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2023.

[184] MooreThreads. Moore animate anyone. `https://github.com/MooreThreads/Moore-AnimateAnyone`, 2024.

[185] Davide Morelli, Alberto Baldrati, Giuseppe Cartella, Marcella Cornia, Marco Bertini, and Rita Cucchiara. Ladi-vton: Latent diffusion textual-inversion enhanced virtual try-on. *arXiv preprint arXiv:2305.13501*, 2023.

[186] Chong Mou, Mingdeng Cao, Xintao Wang, Zhaoyang Zhang, Ying Shan, and Jian Zhang. Revideo: Remake a video with motion and content control. *Advances in Neural Information Processing Systems*, 37:18481–18505, 2024.

[187] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023.

[188] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[189] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[190] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models, 2022.

[191] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[192] Ege Ozguroglu, Ruoshi Liu, Dídac Surś, Dian Chen, Achal Dave, Pavel Tokmakov, and Carl Vondrick. pix2gestalt: Amodal segmentation by synthesizing wholes. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[193] Yatian Pang, Bin Zhu, Bin Lin, Mingzhe Zheng, Francis EH Tay, Ser-Nam Lim, Harry Yang, and Li Yuan. Dreamdance: Animating human images by enriching 3d geometry cues from 2d poses. *arXiv preprint arXiv:2412.00397*, 2024.

[194] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021.

[195] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

[196] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[197] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023.

[198] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019.

[199] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[200] Chuck Peters. 8 tips on how to use a tripod. `https://www.videomaker.com/videonews/2013/09/8-tips-on-how-to-use-a-tripod/`, 2013. Accessed: 2025-05-22.

[201] Mathis Petrovich, Michael J. Black, and Gül Varol. TMR: Text-to-motion retrieval using contrastive 3D human motion synthesis. In *International Conference on Computer Vision (ICCV)*, 2023.

[202] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[203] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

[204] PromptSharingSamaritan. Samaritan 3d cartoon v1.0, 2023.

[205] Liao Qu, Huichao Zhang, Yiheng Liu, Xu Wang, Yi Jiang, Yiming Gao, Hu Ye, Daniel K Du, Zehuan Yuan, and Xinglong Wu. Tokenflow: Unified image tokenizer for multimodal understanding and generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2545–2555, 2025.

[206] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learn-

ing transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[207] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[208] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.

[209] Royi Rassin, Shauli Ravfogel, and Yoav Goldberg. Dalle-2 is seeing double: flaws in word-to-concept mapping in text2image models. *arXiv preprint arXiv:2210.10606*, 2022.

[210] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[211] Jian Ren, Menglei Chai, Oliver J Woodford, Kyle Olszewski, and Sergey Tulyakov. Flow guided transformable bottleneck networks for motion retargeting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10795–10805, 2021.

[212] Yurui Ren, Xiaoqing Fan, Ge Li, Shan Liu, and Thomas H Li. Neural texture extraction and distribution for controllable person image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13535–13544, 2022.

[213] Yurui Ren, Ge Li, Shan Liu, and Thomas H Li. Deep spatial transformation for pose-guided person image generation and animation. *IEEE Transactions on Image Processing*, 29:8622–8635, 2020.

[214] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[215] Daniel Roich, Ron Mokady, Amit H. Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Transactions on Graphics (TOG)*, 2022.

[216] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

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

[218] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[219] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[220] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Wei Wei, Tingbo Hou, Yael Pritch, Neal Wadhwa, Michael Rubinstein, and Kfir Aberman. Hyperdreambooth: Hypernetworks for fast personalization of text-to-image models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6527–6536, 2024.

[221] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[222] Sand-AI. Magi-1: Autoregressive video generation at scale, 2025.

[223] Soubhik Sanyal, Partha Ghosh, Jinlong Yang, Michael J Black, Justus Thies, and Timo Bolkart. Sculpt: Shape-conditioned unpaired learning of pose-dependent clothed and textured human meshes. *arXiv preprint arXiv:2308.10638*, 2023.

[224] Soubhik Sanyal, Alex Vorobiov, Timo Bolkart, Matthew Loper, Betty Mohler, Larry S Davis, Javier Romero, and Michael J Black. Learning realistic human reposing using cyclic self-supervision with 3d shape, pose, and appearance consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11138–11147, 2021.

[225] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

[226] SG_161222. Realistic vision v1.3, 2023.

[227] SG_161222. Realistic vision v4.0, 2023. Face and Gesture submission ID 324. Supplied as supplemental material `fg324.pdf`.

[228] SG_161222. Realistic vision v5.1, 2023.

[229] Ruizhi Shao, Youxin Pang, Zerong Zheng, Jingxiang Sun, and Yebin Liu. 360-degree human video generation with 4d diffusion transformer. *ACM Transactions on Graphics (TOG)*, 43(6):1–13, 2024.

[230] Kalakonda Sai Shashank, Shubh Maheshwari, and Ravi Kiran Sarvadevabhatla. Morag–multi-fusion retrieval augmented generation for human motion. *arXiv preprint arXiv:2409.12140*, 2024.

[231] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, 2020.

[232] Shelly Sheynin, Oron Ashual, Adam Polyak, Uriel Singer, Oran Gafni, Eliya Nachmani, and Yaniv Taigman. Knn-diffusion: Image generation via large-scale retrieval. *arXiv preprint arXiv:2204.02849*, 2022.

[233] YiChang Shih, Wei-Sheng Lai, and Chia-Kai Liang. Distortion-free wide-angle portraits on camera phones. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[234] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2377–2386, 2019.

[235] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in neural information processing systems*, 32, 2019.

[236] Gianluigi Silvestri and Luca Ambrogioni. Covae: Consistency training of variational autoencoders. *arXiv preprint arXiv:2507.09103*, 2025.

[237] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025.

[238] Jaskirat Singh, Cameron Smith, Jose Echevarria, and Liang Zheng. Intelli-paint: Towards developing human-like painting agents. *arXiv preprint arXiv:2112.08930*, 2021.

[239] Jaskirat Singh and Liang Zheng. Combining semantic guidance and deep reinforcement learning for generating human level paintings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16387–16396, 2021.

[240] Ivan Skorokhodov, Sharath Girish, Benran Hu, Willi Menapace, Yanyu Li, Rameen Abdal, Sergey Tulyakov, and Aliaksandr Siarohin. Improving the diffusability of autoencoders. *arXiv preprint arXiv:2502.14831*, 2025.

[241] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[242] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[243] Wei Song, Yuran Wang, Zijia Song, Yadong Li, Haoze Sun, Weipeng Chen, Zenan Zhou, Jianhua Xu, Jiaqi Wang, and Kaicheng Yu. Dualtoken: Towards unifying visual understanding and generation with dual visual vocabularies. *arXiv preprint arXiv:2503.14324*, 2025.

[244] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[245] Shuai Tan, Biao Gong, Xiang Wang, Shiwei Zhang, Dandan Zheng, Ruobing Zheng, Kecheng Zheng, Jingdong Chen, and Ming Yang. Animate-x: Universal character image animation with enhanced motion representation. *arXiv preprint arXiv:2410.10306*, 2024.

[246] Fan Tang, Weiming Dong, Yiping Meng, Xing Mei, Feiyue Huang, Xiaopeng Zhang, and Oliver Deussen. Animated construction of chinese brush paintings. *IEEE transactions on visualization and computer graphics*, 24(12):3019–3031, 2017.

[247] Hao Tang, Chenwei Xie, Xiaoyi Bao, Tingyu Weng, Pandeng Li, Yun Zheng, and Liwei Wang. Unilip: Adapting clip for unified multimodal understanding, generation and editing. *arXiv preprint arXiv:2507.23278*, 2025.

[248] Luming Tang, Nataniel Ruiz, Chu Qinghao, Yuanzhen Li, Aleksander Holynski, David E Jacobs, Bharath Hariharan, Yael Pritch, Neal Wadhwa, Kfir Aberman, and

Michael Rubinstein. Realfill: Reference-driven generation for authentic image completion. *arXiv preprint arXiv:2309.16668*, 2023.

[249] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.

[250] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023.

[251] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

[252] Dmitrii Torbunov, Yi Huang, Haiwang Yu, Jin Huang, Shinjae Yoo, Meifeng Lin, Brett Viren, and Yihui Ren. Uvcgan: Unet vision transformer cycle-consistent gan for unpaired image-to-image translation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 702–712, 2023.

[253] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[254] Shuyuan Tu, Zhen Xing, Xintong Han, Zhi-Qi Cheng, Qi Dai, Chong Luo, and Zuxuan Wu. Stableanimator: High-quality identity-preserving human image animation. *arXiv preprint arXiv:2411.17697*, 2024.

[255] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[256] Tom Van Wouwe, Seunghwan Lee, Antoine Falisse, Scott Delp, and C Karen Liu. Diffusionposer: Real-time human motion reconstruction from arbitrary sparse sen-

sors using autoregressive diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2513–2523, 2024.

[257] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[258] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

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

[260] Boyuan Wang, Xiaofeng Wang, Chaojun Ni, Guosheng Zhao, Zhiqin Yang, Zheng Zhu, Muyang Zhang, Yukun Zhou, Xinze Chen, Guan Huang, et al. Humandreamer: Generating controllable human-motion videos via decoupled generation. *arXiv preprint arXiv:2503.24026*, 2025.

[261] Chunwei Wang, Guansong Lu, Junwei Yang, Runhui Huang, Jianhua Han, Lu Hou, Wei Zhang, and Hang Xu. Illume: Illuminating your llms to see, draw, and self-enhance. *arXiv preprint arXiv:2412.06673*, 2024.

[262] Qilin Wang, Zhengkai Jiang, Chengming Xu, Jiangning Zhang, Yabiao Wang, Xinyi

Zhang, Yun Cao, Weijian Cao, Chengjie Wang, and Yanwei Fu. Vividpose: Advancing stable video diffusion for realistic human image animation. *arXiv preprint arXiv:2405.18156*, 2024.

[263] Qixun Wang, Xu Bai, Haofan Wang, Zekui Qin, Anthony Chen, Huaxia Li, Xu Tang, and Yao Hu. Instantid: Zero-shot identity-preserving generation in seconds. *arXiv preprint arXiv:2401.07519*, 2024.

[264] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision, 2024.

[265] Tan Wang, Linjie Li, Kevin Lin, Chung-Ching Lin, Zhengyuan Yang, Hanwang Zhang, Zicheng Liu, and Lijuan Wang. Disco: Disentangled control for referring human dance generation in real world. *arXiv preprint arXiv:2307.00040*, 2023.

[266] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[267] Xiang Wang, Shiwei Zhang, Changxin Gao, Jiayu Wang, Xiaoqiang Zhou, Yingya Zhang, Luxin Yan, and Nong Sang. Unianimate: Taming unified video diffusion models for consistent human image animation. *arXiv preprint arXiv:2406.01188*, 2024.

[268] Xiang Wang, Shiwei Zhang, Longxiang Tang, Yingya Zhang, Changxin Gao, Yuehuan Wang, and Nong Sang. Unianimate-dit: Human image animation with large-scale video diffusion transformer. *arXiv preprint arXiv:2504.11289*, 2025.

[269] Xiang Wang, Shiwei Zhang, Longxiang Tang, Yingya Zhang, Changxin Gao, Yuehuan Wang, and Nong Sang. Unianimate-dit: Human image animation with large-scale video diffusion transformer. *arXiv preprint arXiv:2504.11289*, 2025.

[270] Xiang Wang, Shiwei Zhang, Hangjie Yuan, Yujie Wei, Yingya Zhang, Changxin Gao, Yuehuan Wang, and Nong Sang. Taming consistency distillation for accelerated human image animation. *arXiv preprint arXiv:2504.11143*, 2025.

[271] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

[272] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. G3an: Disentangling appearance and motion for video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5264–5273, 2020.

[273] Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, et al. Lavie: High-quality video generation with cascaded latent diffusion models. *International Journal of Computer Vision*, 133(5):3059–3078, 2025.

[274] Zhao Wang, Aoxue Li, Lingting Zhu, Yong Guo, Qi Dou, and Zhenguo Li. Customvideo: Customizing text-to-video generation with multiple subjects. *arXiv preprint arXiv:2401.09962*, 2024.

[275] Zhao Wang, Hao Wen, Lingting Zhu, Chenming Shang, Yujiu Yang, and Qi Dou. Anycharv: Bootstrap controllable character video generation with fine-to-coarse guidance. *arXiv preprint arXiv:2502.08189*, 2025.

[276] Zhenzhi Wang, Yixuan Li, Yanhong Zeng, Yuwei Guo, Dahua Lin, Tianfan Xue, and Bo Dai. Multi-identity human image animation with structural video diffusion. *arXiv preprint arXiv:2504.04126*, 2025.

[277] Zunfu Wang, Fang Liu, Zhixiong Liu, Changjuan Ran, and Mohan Zhang. Intelligentpaint: a chinese painting process generation method based on vision transformer. *Multimedia Systems*, 30(2):1–17, 2024.

[278] Yujie Wei, Shiwei Zhang, Zhiwu Qing, Hangjie Yuan, Zhiheng Liu, Yu Liu, Yingya Zhang, Jingren Zhou, and Hongming Shan. Dreamvideo: Composing your dream videos with customized subject and motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6537–6549, 2024.

[279] Yuxiang Wei, Yabo Zhang, Zhilong Ji, Jinfeng Bai, Lei Zhang, and Wangmeng Zuo. Elite: Encoding visual concepts into textual embeddings for customized text-to-image generation. *arXiv preprint arXiv:2302.13848*, 2023.

[280] Xin Wen, Bingchen Zhao, Ismail Elezi, Jiankang Deng, and Xiaojuan Qi. " principal components" enable a new language of images. *arXiv preprint arXiv:2503.08685*, 2025.

[281] Zhenzhen Weng, Laura Bravo-Sánchez, and Serena Yeung. Diffusion-hpc: Generating synthetic images with realistic humans. *arXiv preprint arXiv:2303.09541*, 2023.

[282] Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv preprint arXiv:2509.20328*, 2025.

[283] Pingyu Wu, Kai Zhu, Yu Liu, Longxiang Tang, Jian Yang, Yansong Peng, Wei Zhai, Yang Cao, and Zheng-Jun Zha. Alitok: Towards sequence modeling alignment between tokenizer and autoregressive model. *arXiv preprint arXiv:2506.05289*, 2025.

[284] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

[285] Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. Vila-u: a unified foundation model integrating visual understanding and generation. *arXiv preprint arXiv:2409.04429*, 2024.

[286] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12863–12872, 2021.

[287] Ning Xie, Hirotaka Hachiya, and Masashi Sugiyama. Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting. *IEICE TRANSACTIONS on Information and Systems*, 96(5):1134–1144, 2013.

[288] Rongchang Xie, Chen Du, Ping Song, and Chang Liu. Muse-vl: Modeling unified vlm through semantic discrete encoding. *arXiv preprint arXiv:2411.17762*, 2024.

[289] Zhenyu Xie, Zaiyu Huang, Xin Dong, Fuwei Zhao, Haoye Dong, Xijin Zhang, Feida Zhu, and Xiaodan Liang. Gp-vton: Towards general purpose virtual try-on via collaborative local-flow global-parsing learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23550–23559, 2023.

[290] Tianwei Xiong, Jun Hao Liew, Zilong Huang, Jiashi Feng, and Xihui Liu. Gigatok: Scaling visual tokenizers to 3 billion parameters for autoregressive image generation. *arXiv preprint arXiv:2504.08736*, 2025.

[291] Yuliang Xiu, Jinlong Yang, Xu Cao, Dimitrios Tzionas, and Michael J. Black. ECON: Explicit Clothed humans Optimized via Normal integration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

[292] Chao Xu, Jiangning Zhang, Yue Han, Guanzhong Tian, Xianfang Zeng, Ying Tai, Yabiao Wang, Chengjie Wang, and Yong Liu. Designing one unified framework for high-fidelity face reenactment and swapping. In *European conference on computer vision*, pages 54–71. Springer, 2022.

[293] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: learning and evaluating human preferences for text-to-image generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 15903–15935, 2023.

[294] Vasco Xu, Chenfeng Gao, Henry Hoffmann, and Karan Ahuja. Mobileposer: Real-time full-body pose estimation and 3d human translation from imus in mobile consumer devices. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–11, 2024.

[295] Zhongcong Xu, Jianfeng Zhang, Jun Hao Liew, Hanshu Yan, Jia-Wei Liu, Chenxu

Zhang, Jiashi Feng, and Mike Zheng Shou. Magicanimate: Temporally consistent human image animation using diffusion model. In *arXiv*, 2023.

[296] Jingyun Xue, Hongfa Wang, Qi Tian, Yue Ma, Andong Wang, Zhiyuan Zhao, Shaobo Min, Wenzhe Zhao, Kaihao Zhang, Heung-Yeung Shum, et al. Follow-your-pose v2: Multiple-condition guided character image animation for stable pose control. *arXiv preprint arXiv:2406.03035*, 2024.

[297] Keyu Yan, Tingwei Gao, Hui Zhang, and Chengjun Xie. Linking garment with person via semantically associated landmarks for virtual try-on. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17194–17204, 2023.

[298] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by example: Exemplar-based image editing with diffusion models. *arXiv preprint arXiv:2211.13227*, 2022.

[299] Jiawei Yang, Tianhong Li, Lijie Fan, Yonglong Tian, and Yue Wang. Latent denoising makes good visual tokenizers. *arXiv preprint arXiv:2507.15856*, 2025.

[300] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024.

[301] Lu Yang, Wenhe Jia, Shan Li, and Qing Song. Deep learning technique for human parsing: A survey and outlook. *arXiv preprint arXiv:2301.00394*, 2023.

[302] Zhendong Yang, Ailing Zeng, Chun Yuan, and Yu Li. Effective whole-body pose estimation with two-stages distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4210–4220, 2023.

[303] Zhuoqian Yang, Shikai Li, Wayne Wu, and Bo Dai. 3dhumangan: 3d-aware human image generation with 3d pose mapping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23008–23019, 2023.

[304] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

[305] Jingfeng Yao, Cheng Wang, Wenyu Liu, and Xinggang Wang. Fasterdit: Towards faster diffusion transformers training without architecture modification. *Advances in Neural Information Processing Systems*, 37:56166–56189, 2024.

[306] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[307] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

[308] Xinyu Yi, Yuxiao Zhou, Marc Habermann, Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Feng Xu. Physical inertial poser (pip): Physics-aware real-time human motion tracking from sparse inertial sensors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[309] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

[310] Jiyang Yu and Ravi Ramamoorthi. Selfie video stabilization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 551–566, 2018.

[311] Jiyang Yu, Ravi Ramamoorthi, Keli Cheng, Michel Sarkis, and Ning Bi. Real-time selfie video stabilization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12036–12044, 2021.

[312] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.

[313] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. *Advances in Neural Information Processing Systems*, 37:128940–128966, 2024.

[314] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.

[315] Wing-Yin Yu, Lai-Man Po, Ray CC Cheung, Yuzhi Zhao, Yu Xue, and Kun Li. Bidirectionally deformable motion modulation for video-based human pose transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7502–7512, 2023.

[316] Shenghai Yuan, Jinfa Huang, Xianyi He, Yunyuan Ge, Yujun Shi, Liuhan Chen, Jiebo Luo, and Li Yuan. Identity-preserving text-to-video generation by frequency decomposition. *arXiv preprint arXiv:2411.17440*, 2024.

[317] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[318] Polina Zablotskaia, Aliaksandr Siarohin, Bo Zhao, and Leonid Sigal. Dwnet: Dense warp-based network for pose-guided human video generation. *arXiv preprint arXiv:1910.09139*, 2019.

[319] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023.

[320] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Shaoli Huang, Yong Zhang, Hongwei Zhao, Hongtao Lu, and Xi Shen. T2m-gpt: Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[321] Kai Zhang, Fujun Luan, Sai Bi, and Jianming Zhang. Ep-cfg: Energy-preserving classifier-free guidance. *arXiv preprint arXiv:2412.09966*, 2024.

[322] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.

[323] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Scaling in-the-wild training for diffusion-based illumination harmonization and editing by imposing consistent light transport. In *The Thirteenth International Conference on Learning Representations*, 2025.

[324] Pengze Zhang, Lingxiao Yang, Jian-Huang Lai, and Xiaohua Xie. Exploring dual-task correlation for pose guided person image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7713–7722, 2022.

[325] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[326] Yahui Zhang, Shaodi You, Sezer Karaoglu, and Theo Gevers. Pose guided human motion transfer by exploiting 2d and 3d information. In *2022 International Conference on 3D Vision (3DV)*, pages 587–595. IEEE, 2022.

[327] Yuang Zhang, Jiaxi Gu, Li-Wen Wang, Han Wang, Junqi Cheng, Yuefeng Zhu, and Fangyuan Zou. Mimicmotion: High-quality human motion video generation with confidence-aware pose guidance. *arXiv preprint arXiv:2406.19680*, 2024.

[328] Amy Zhao, Guha Balakrishnan, Kathleen M Lewis, Frédo Durand, John V Guttag, and Adrian V Dalca. Painting many pasts: Synthesizing time lapse videos of paintings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8435–8445, 2020.

[329] Jian Zhao and Hui Zhang. Thin-plate spline motion model for image animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3657–3666, 2022.

[330] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *arXiv preprint arXiv:2302.04867*, 2023.

[331] Yajie Zhao, Zeng Huang, Tianye Li, Weikai Chen, Chloe LeGendre, Xinglei Ren, Ari Shapiro, and Hao Li. Learning perspective undistortion of portraits. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7849–7859, 2019.

[332] Yue Zhao, Fuzhao Xue, Scott Reed, Linxi Fan, Yuke Zhu, Jan Kautz, Zhiding Yu, Philipp Krähenbühl, and De-An Huang. Qlip: Text-aligned visual tokenization unifies auto-regressive multimodal understanding and generation. *arXiv preprint arXiv:2502.05178*, 2025.

[333] Anlin Zheng, Xin Wen, Xuanyang Zhang, Chuofan Ma, Tiancai Wang, Gang Yu, Xiangyu Zhang, and Xiaojuan Qi. Vision foundation models as effective visual tokenizers for autoregressive image generation. *arXiv preprint arXiv:2507.08441*, 2025.

[334] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

[335] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *arXiv preprint arXiv:2306.09305*, 2023.

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

[337] Jingkai Zhou, Yifan Wu, Shikai Li, Min Wei, Chao Fan, Weihua Chen, Wei Jiang, and Fan Wang. Realisdance-dit: Simple yet strong baseline towards controllable character animation in the wild. *arXiv preprint arXiv:2504.14977*, 2025.

[338] Shangchen Zhou, Kelvin C.K. Chan, Chongyi Li, and Chen Change Loy. Towards robust blind face restoration with codebook lookup transformer. In *NeurIPS*, 2022.

[339] Xinyue Zhou, Mingyu Yin, Xinyuan Chen, Li Sun, Changxin Gao, and Qingli Li. Cross attention based style distribution for controllable person image synthesis. *arXiv preprint arXiv:2208.00712*, 2022.

[340] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[341] Luyang Zhu, Dawei Yang, Tyler Zhu, Fitsum Reda, William Chan, Chitwan Saharia, Mohammad Norouzi, and Ira Kemelmacher-Shlizerman. Tryondiffusion: A tale of two unets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2023.

[342] Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, and Peter Wonka. Improved stylegan embedding: Where are the good latents?, 2021.

[343] Shenhao Zhu, Junming Leo Chen, Zuozhuo Dai, Zilong Dong, Yinghui Xu, Xun Cao, Yao Yao, Hao Zhu, and Siyu Zhu. Champ: Controllable and consistent human image animation with 3d parametric guidance. In *European Conference on Computer Vision*, pages 145–162. Springer, 2024.

[344] Zhengxia Zou, Tianyang Shi, Shuang Qiu, Yi Yuan, and Zhenwei Shi. Stylized neural painting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15689–15698, 2021.