

A Implementation details

Table 7: Architecture and training configurations of Blockwise Flow Matching

	Vanilla BFM-S (Table 1)	BFM-S _{SF} (Table 1,4,6,7)	BFM-S _{SF} -RA (Table 1)	BFM-XL _{SF} (Table 2,3,5,7,8)	BFM-XL _{SF} -RA (Table 2,8)
Num. segments M	6	6	6	6	6
Each Velocity Block $v_{\theta}^{(m)}$					
Num. layers	8	4	4	5	5
Hidden dims	384	384	384	1152	1152
Num. heads	6	6	6	16	16
Feature Alignment Network f_{ϕ}					
Num. layers	—	6	6	20	20
Hidden dims	—	384	384	1152	1152
Num. heads	—	6	6	16	16
Feature Residual Network f_{η}					
Num. layers	—	—	2	—	4
Hidden dims	—	—	384	—	1152
Num. heads	—	—	6	—	16
Training Config.					
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	1e-4	1e-4	1e-4	1e-4	1e-4
Batch size	256	256	256	256	256
λ	—	0.5	—	0.5	—
Visual Encoder \mathcal{E}	—	DINOv2-B	DINOv2-B	DINOv2-B	DINOv2-B
$d(\cdot, \cdot)$	—	cosine sim.	cosine sim.	cosine sim.	cosine sim.

We provide the overall hyperparameter setup in Table 7.

Training. We implement our model based on the original SiT implementation [1] with recent improvements such as SwiGLU [20] activations and Rotary Positional Embeddings (RoPE) [21]. We use the pre-computed latent vectors from raw pixels via stable diffusion VAE without data augmentation following [17]. For feature projection, we employ a three-layer MLP with SiLU activation functions. We use uniform partitioning where $t_m = \frac{m}{M}$ for simplicity, though our preliminary experiments suggest that non-uniform partitioning optimized for data-specific dynamics could yield further improvements. We use 8 NVIDIA RTX3090 or H200 GPUs for experiments. Full training pseudo-code is illustrated in Algorithm 1 and 2.

Inference. For inference, we use the Euler solver with the ODE (Tables 1,3,4,5,6 in the main paper) or the Euler-Maruyama solver [1] with the SDE (Table 2 in the main paper). We keep the 246 solver steps (41 solver steps for each segment) except for Table 3 in the main paper. Given an initial noise input $x_{t_0} = x_0 \sim p_{\text{noise}}$, the generation proceeds sequentially through each temporal segment $[t_{m-1}, t_m)$. At the beginning of the m -th segment, we compute the semantic feature $f_{t_{m-1}}$ using the feature alignment network f_{ϕ} . For subsequent timesteps $t \in [t_{m-1}, t_m)$, we approximate the semantic feature either using FRN f_{η} or the full alignment network f_{ϕ} , depending on the desired trade-off between speed and fidelity. Then, the velocity block $v_{\theta}^{(m)}$ predicts the velocity: $\hat{v}_t^{(m)} = v_{\theta}^{(m)}(x_t, t, f_t)$. The intermediate state is updated iteratively via a numerical solver using the predicted velocity. This process repeats until reaching $t = 1$, yielding the final sample $x_1 \sim p_{\text{data}}$. A full pseudocode implementation of the inference process is provided in Algorithm 3 and 4.

B Analysis details

Component analysis (Table 1). We provide additional implementation details for the experimental configurations reported in Table 1 of main paper. For the vanilla Flow Matching model, we train the original SiT-S model (12 transformer blocks) with architectural improvements (SwiGLU and Rotary Positional Embeddings). For our vanilla BFM-S model, we train the same transformer architecture with six temporal segments. We design each velocity block with 8 transformer blocks. When adding a feature alignment network (BFM-S_{SF}), we reduce the number of layers of each velocity block, as shown in Table 7.

Algorithm 1 Training Blockwise Flow Matching (BFM) with Semantic Feature Guidance

```
1: repeat
2:   Sample data and noise:  $x_0 \sim p_{\text{data}}, x_1 \sim p_{\text{noise}}$ 
3:   Initialize total losses:  $\mathcal{L}_{\text{BFM}} \leftarrow 0, \mathcal{L}_{\text{align}} \leftarrow 0$ 
4:   for each segment index  $m = 1, \dots, M$  do
5:     Compute segment endpoints:
6:        $x_{t_{m-1}} = (1 - t_{m-1})x_0 + t_{m-1}x_1$ 
7:        $x_{t_m} = (1 - t_m)x_0 + t_mx_1$ 
8:     Sample intermediate timestep:  $t \sim \mathcal{U}[t_{m-1}, t_m]$ 
9:     Compute intermediate point:  $x_t = (1 - t')x_{t_{m-1}} + t'x_{t_m}$ , where  $t' = \frac{t - t_{m-1}}{t_m - t_{m-1}}$ 
10:    Compute semantic features:  $h^* = \mathcal{E}(x_1)$ ,  $f_t = f_\phi(x_t, t, c)$ 
11:    Compute ground-truth velocity:  $v_t^{(m)} = \frac{x_{t_m} - x_{t_{m-1}}}{t_m - t_{m-1}}$ 
12:    Compute losses:
13:       $\mathcal{L}_{\text{BFM}} += \|v_\theta^{(m)}(x_t, t, f_t) - v_t^{(m)}\|^2$ 
14:       $\mathcal{L}_{\text{align}} += d(h_\psi(f_t), h^*)$ 
15:    end for
16:    Update parameters  $\theta, \phi, \psi$  with gradient descent:
       $\nabla_{\theta, \phi, \psi} (\mathcal{L}_{\text{BFM}} + \lambda \mathcal{L}_{\text{align}})$ 
17: until convergence
```

Algorithm 2 Training Feature Residual Network (FRN)

```
1: Freeze parameters  $v_\theta, f_\phi, h_\psi$ , and introduce FRN parameters  $f_\eta$ 
2: repeat
3:   Sample  $m, x_0, x_1, t$  as Algorithm 1
4:   Compute initial semantic feature at segment start:  $f_{t_{m-1}} = f_\phi(x_{t_{m-1}}, t_{m-1}, c)$ 
5:   Compute initial semantic feature at intermediate step  $t$ :  $f_t = f_\phi(x_t, t, c)$ 
6:   Approximate semantic feature at intermediate step  $t$ :  $\hat{f}_t = f_{t_{m-1}} + t' \cdot f_\eta(x_t, t, c)$ 
7:   Compute residual approximation loss  $\mathcal{L}_{\text{FRN}} = d(h_\psi(\hat{f}_t), h_\psi(f_t))$ 
8:   Update FRN parameters  $\psi$  via gradient descent:
      $\nabla_\psi \mathcal{L}_{\text{FRN}}$ 
9: until convergence
```

30 **Spectral Entropy and High-Frequency Ratio (Figure 3).** To analyze the frequency characteristics
31 of generated images across timesteps, we compute two key metrics: Spectral Entropy (SE) and the
32 High-Frequency Ratio (HFR). Spectral Entropy quantifies the distributional complexity of a signal in
33 the frequency domain. Specifically, we treat the normalized 2D power spectrum of an image as a
34 probability distribution and compute its Shannon entropy. A higher SE indicates a more uniform,
35 less structured distribution of spectral energy (i.e., more randomness), while a lower SE reflects a
36 concentration of energy in fewer frequencies, indicating more structured signals. High-Frequency
37 Ratio measures the proportion of total spectral energy in the high-frequency range. It captures the
38 relative contribution of fine-grained details in the image.

39 To compute both metrics, we randomly sample 10,000 real images from the ImageNet set and
40 transform them to noisy versions at specific timesteps by interpolation formulation. Each image
41 is converted to the frequency domain via a 2D Fourier Transform. We then calculate the power
42 spectrum and apply azimuthal integration to obtain the spectral distribution. SE is computed as the
43 Shannon entropy of the normalized spectrum, while HFR is obtained by summing the energy above a
44 predefined frequency threshold (=0.5) and dividing by the total energy.

45 **Fourier power spectrum (Figure 5).** We perform a frequency-domain analysis to evaluate how
46 well the spectral characteristics of generated images match those of real images. This experiment
47 tests whether our blockwise modeling strategy improves spectral fidelity by allowing different blocks
48 to specialize in capturing distinct frequency components along the generative trajectory. Specifically,
49 we randomly sample 100 images from each model (BFM-XL_{SF} and SiT-XL [11]) and 100 real images
50 from ImageNet. We compute the 2D Fourier power spectrum for each image and apply azimuthal

Algorithm 3 Inference

```
1: Initialize:  $x_{t_0} = x_0 \sim p_{\text{noise}}$ 
2: for  $m = 1, 2, \dots, M$  do
3:   Define  $K$  solver steps within segment:  $t_{m-1} = t^{(0)} < \dots < t^{(K)} = t_m$ 
4:   for  $k = 0, 1, \dots, K - 1$  do
5:     Compute semantic feature:  $f_{t^{(k)}} = f_\phi(x_{t^{(k)}}, t^{(k)}, c)$ 
6:     Compute velocity:  $\hat{v}_{t^{(k)}} = v_\theta^{(m)}(x_{t^{(k)}}, t^{(k)}, f_{t^{(k)}})$ 
7:     Update state with solver  $\Phi$  step size  $\Delta t$ :
8:        $x_{t^{(k+1)}} = \Phi(x_{t^{(k)}}, \hat{v}_{t^{(k)}}, \Delta t)$ 
9:   end for
10: end for
11: Output final generated sample:  $x_1 \sim p_{\text{data}}$ 
```

Algorithm 4 Efficient inference with Feature Residual Network

```
1: Initialize:  $x_{t_0} = x_0 \sim p_{\text{noise}}$ 
2: for  $m = 1, 2, \dots, M$  do
3:   Compute semantic feature once at segment start:
4:      $f_{t_{m-1}} = f_\phi(x_{t_{m-1}}, t_{m-1}, c)$ 
5:   Define  $K$  solver steps within segment:  $t_{m-1} = t^{(0)} < \dots < t^{(K)} = t_m$ 
6:   for  $k = 0, 1, \dots, K - 1$  do
7:     Approximate semantic feature efficiently:  $f_{t^{(k)}} = f_{t_{m-1}} + f_\eta(x_{t^{(k)}}, t^{(k)}, c)$ 
8:     Compute velocity:  $\hat{v}_{t^{(k)}} = v_\theta^{(m)}(x_{t^{(k)}}, t^{(k)}, f_{t^{(k)}})$ 
9:     Update state with solver  $\Phi$  step size  $\Delta t$ :
10:       $x_{t^{(k+1)}} = \Phi(x_{t^{(k)}}, \hat{v}_{t^{(k)}}, \Delta t)$ 
11:   end for
12: end for
13: Output final generated sample:  $x_1 \sim p_{\text{data}}$ 
```

51 integration to obtain a 1D mean spectral power distribution, following the procedure in [22]. We then
52 calculate the Fréchet Distance [23] between the spectral distributions of generated and real images to
53 quantify their similarity.

54 **Discrepancy between $f_{t_{m-1}}$ and f_t (Figure 4).** For each segment m , we randomly sample 50
55 images from the ImageNet dataset and convert them into their corresponding noisy versions $x_{t_{m-1}}$
56 and x_t at timesteps t_{m-1} and $t \in [t_{m-1}, t_m)$, respectively. We then evaluate the temporal consistency
57 of the semantic features extracted by the alignment network by computing the mean squared error
58 (MSE) between $f_{t_{m-1}} = f_\phi(x_{t_{m-1}}, t_{m-1}, c)$ and $f_t = f_\phi(x_t, t, c)$. A lower MSE indicates that the
59 feature alignment network learns temporally stable, time-invariant representations across the segment.

60 **Quantitative comparison between REPA [17] and SemFeat (Table 6).** We provide additional
61 implementation details for the experimental configurations reported in Table 6. All models are trained
62 for 100K iterations using DINOv2-B [24] as the target representation, with a loss weight of $\lambda = 0.5$
63 and negative cosine similarity as the alignment objective. Feature projection layers are implemented
64 with identical configurations across all methods for a fair comparison. (a): We implement REPA
65 using the SiT-S architecture (12 transformer layers), aligning the hidden representations at the 8th
66 transformer layer to the DINOv2-B targets. (b): We incorporate our proposed SemFeat into the
67 standard FM framework by decomposing the SiT-S model into two modules: a feature alignment
68 network composed of 8 transformer layers, and a velocity model composed of 4 transformer layers.
69 (c): This setting corresponds to our BFM-S_{SF} in Table 7.

70 **PCA results (Figure 7).** For the PCA analysis, we extract semantic features from noisy inputs x_t
71 using both REPA-XL and BFM-XL_{SF}. In the case of REPA-XL, we use the hidden representations
72 from the 8th transformer layer, as this layer is aligned with DINO features during training. For
73 BFM-XL_{SF}, we directly use the outputs of the feature alignment network, $f_t = f_\phi(x_t, t, c)$. We
74 perform Principal Component Analysis across features collected from multiple timesteps to capture

75 their dominant variations. The top three principal components are then mapped to the RGB channels
76 for visualization, allowing us to assess the semantic consistency and structure of features across time.

77 C Additional results

78 **Different model scales.** We provide evaluation results of different models with different scales. As
demonstrated in Table 8, our model consistently outperforms baselines.

Table 8: **Comparison of methods at different scales** on ImageNet 256×256 without classifier-free guidance. \downarrow indicates that lower values are better.

Method	Iterations	#Params.	NFE	FID \downarrow
DiT-S [16]	400K	33M	250	66.3
SiT-S [1]	400K	33M	250	57.5
REPA-S [17]	400K	33M	250	49.7
BFM-S_{SF} (Ours)	400K	70M	246	44.6
DiT-XL [16]	400K	675M	250	19.5
SiT-XL [1]	400K	675M	250	17.2
SiT-XL [1]	7M	675M	250	8.3
REPA-XL [17]	400K	675M	250	7.9
REPA-XL [17]	4M	675M	250	5.9
BFM-XL_{SF} (Ours)	400K	942M	246	6.4
BFM-XL_{SF} (Ours)	1M	942M	246	5.6

79

80 **Additional PCA results.** In Figure 9, we present additional PCA visualizations of semantic
81 features across varying noise levels. SemFeat consistently produces semantically coherent and
82 temporally stable representations, maintaining structural consistency even under significant noise. In
83 contrast, REPA exhibits greater variability and noise in its feature representations, with noticeable
84 inconsistencies across the same object regions. These observations further support the effectiveness of
85 SemFeat’s modular conditioning in capturing robust and meaningful semantic information throughout
the generative trajectory.

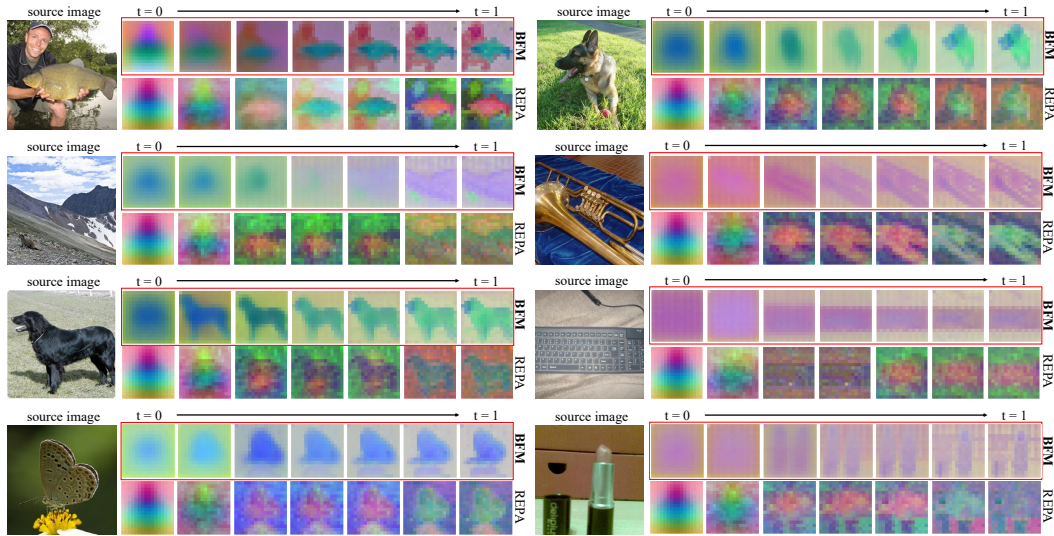


Figure 9: **Semantic features over timesteps.** We visualize the PCA of features from BFM and REPA for the source images. Compared to REPA, the semantic features extracted from BFM are more consistent across timesteps.

86

87 **D Qualitative results**



Figure 10: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “fox squirrel” (335).



Figure 11: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “Cacatua galerita” (89).



Figure 12: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “shield, buckler” (787).



Figure 13: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “snow leopard” (289).



Figure 14: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “American robin” (246).



Figure 15: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “volcano” (980).



Figure 16: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “Great Dane” (246).

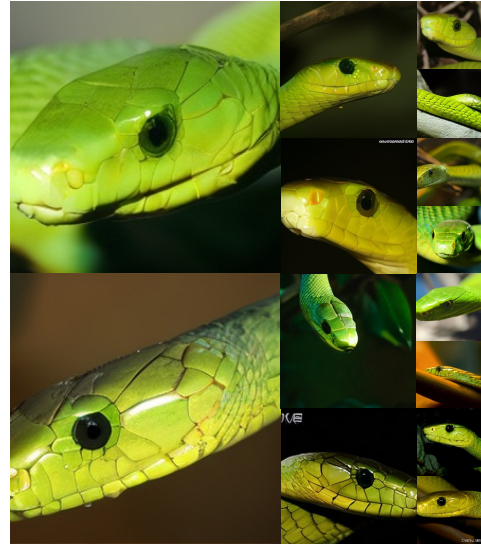


Figure 17: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “green mamba” (64).

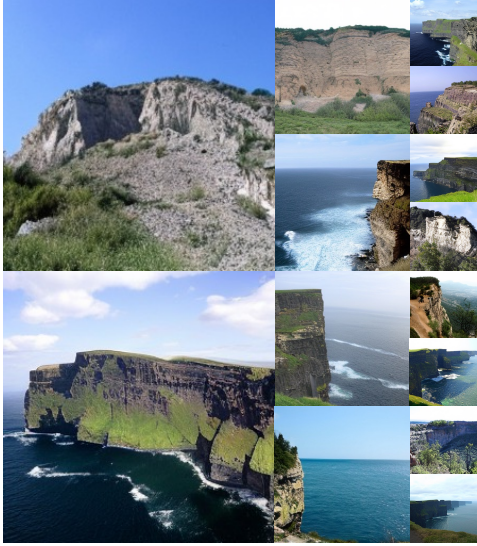


Figure 18: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “Cliff” (972).



Figure 19: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “Coral Reef” (973).

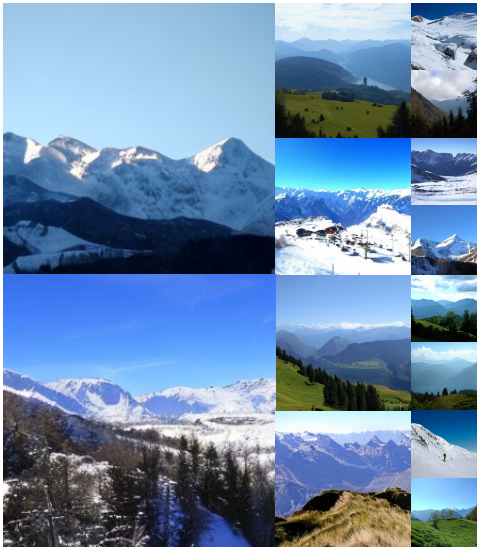


Figure 20: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “Alp” (970).

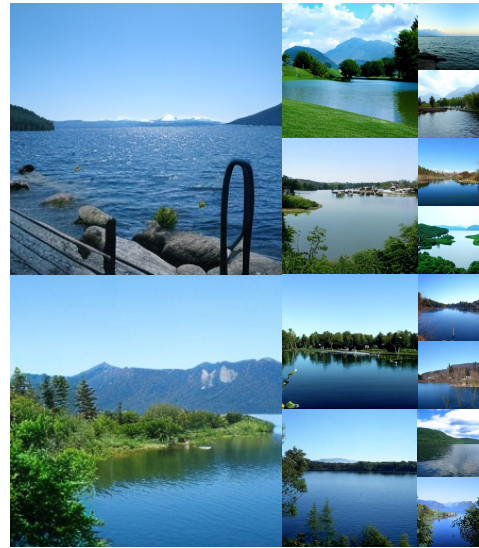


Figure 21: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “Lakeside” (975).



Figure 22: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “bald eagle, American eagle, *Haliaeetus leucocephalus*” (22).



Figure 23: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “Norwegian elkhound, elkhound” (174).



Figure 24: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “hippopotamus, hippo, river horse, *Hippopotamus amphibious*” (246).



Figure 25: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “lesser panda, red panda, panda, bear cat, cat bear, *Ailurus fulgens*” (387).

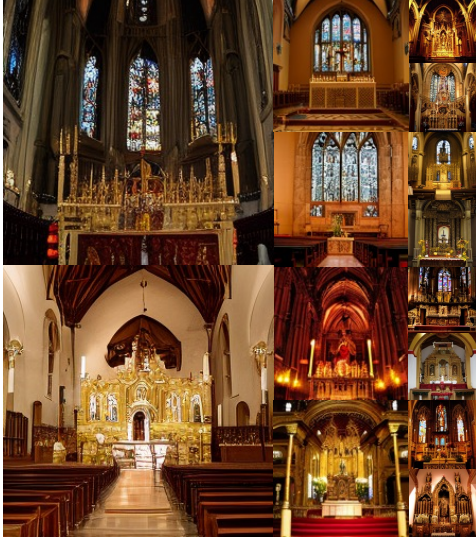


Figure 26: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “altar” (406).



Figure 27: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “castle” (483).



Figure 28: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “coffee mug” (504).



Figure 29: **Uncurated generation results of BFM-XL_{SF}**. We use classifier-free guidance with $w = 4.0$. Class label = “library” (624).

E Limitations and Future work

Reliance on external features. Our method relies on semantic features extracted from a pretrained visual encoder (e.g., DINOv2) to guide the velocity models through SemFeat. While this enables stronger semantic conditioning and improves generation quality, it introduces an external dependency and limits the model’s generality to domains where such pretrained encoders are available. In future work, we plan to explore self-supervised alternatives that jointly learn semantic representations during training, potentially eliminating the need for external networks.

Multi-modal large-scale datasets. Our current evaluation focuses on single-modality image generation using ImageNet. While BFM scales well in this setting, applying the same framework to multi-modal, large-scale datasets (e.g., text-to-image or video datasets) remains an open challenge. Future research could investigate how to adapt blockwise modeling and semantic conditioning to settings that require more complex cross-modal reasoning and longer temporal consistency, such as text-driven video generation.

Non-uniform partitioning schedule. In this work, we adopt a uniform temporal segmentation strategy for simplicity. However, the generative trajectory exhibits varying complexity across timesteps, suggesting that certain regions (e.g., early noisy stages or late refinement phases) may benefit from finer-grained modeling. Future work could explore non-uniform or learnable partitioning schedules that allocate computational resources adaptively based on local signal complexity, potentially improving both performance and efficiency.

F Broader impact

The goal of this work is to improve the efficiency and scalability of generative models. By introducing Blockwise Flow Matching (BFM), we reduce the computational burden of high-quality generation, making diffusion-based models more accessible for researchers and practitioners with limited resources. This has potential societal benefits in democratizing access to generative AI, enabling smaller labs, startups, and educational institutions to experiment with cutting-edge models without requiring extensive computational infrastructure. However, improved efficiency also lowers the barrier for misuse. Generative models can be exploited to produce content such as deepfakes or misinformation at scale. We acknowledge that its capabilities could be repurposed in harmful ways. Mitigating such misuse requires responsible downstream deployment, as well as efforts from the broader community to develop and enforce ethical standards around generative AI.

References

- [1] 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, ECCV*, 2024.
- [2] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *arXiv preprint arXiv:2207.12598*, 2022.
- [3] Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. In *arXiv preprint arXiv:2404.07724*, 2024.
- [4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems, NeurIPS*, 2021.
- [5] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 2022.
- [6] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2022.
- [7] Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. In *Advances in Neural Information Processing Systems, NeurIPS*, 2023.

- [8] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2023.
- [9] Ali Hatamizadeh, Jiaming Song, Guilin Liu, Jan Kautz, and Arash Vahdat. Diffit: Diffusion vision transformers for image generation. In *European Conference on Computer Vision, ECCV*, 2024.
- [10] 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.
- [11] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. In *Transactions on Machine Learning Research, TMLR*, 2024.
- [12] Shuai Wang, Zexian Li, Tianhui Song, Xubin Li, Tiezheng Ge, Bo Zheng, and Limin Wang. Exploring DCN-like architecture for fast image generation with arbitrary resolution. In *Advances in Neural Information Processing Systems, NeurIPS*, 2024.
- [13] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning, ICML*, 2023.
- [14] Rui Zhu, Yingwei Pan, Yehao Li, Ting Yao, Zhenglong Sun, Tao Mei, and Chang Wen Chen. Sd-dit: Unleashing the power of self-supervised discrimination in diffusion transformer. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2024.
- [15] ZiDong Wang, Zeyu Lu, Di Huang, Cai Zhou, Wanli Ouyang, et al. Fitv2: Scalable and improved flexible vision transformer for diffusion model. In *arXiv preprint arXiv:2410.13925*, 2024.
- [16] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2023.
- [17] 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 *arXiv preprint arXiv:2410.06940*, 2024.
- [18] Minglei Shi, Ziyang Yuan, Haotian Yang, Xintao Wang, Mingwu Zheng, Xin Tao, Wenliang Zhao, Wenzhao Zheng, Jie Zhou, Jiwen Lu, et al. Diffmoe: Dynamic token selection for scalable diffusion transformers. *arXiv preprint arXiv:2503.14487*, 2025.
- [19] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dynamic diffusion transformer. *arXiv preprint arXiv:2410.03456*, 2024.
- [20] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. In *arXiv preprint arXiv:2302.13971*, 2023.
- [21] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. In *Neurocomputing*, 2024.
- [22] Ricard Durall, Margret Keuper, and Janis Keuper. Watch your up-convolution: CNN based generative deep neural networks are failing to reproduce spectral distributions. In *Conference on Computer Vision and Pattern Recognition, CVPR*, 2020.
- [23] Efrat, Guibas, Sarel Har-Peled, and Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. In *Discrete & Computational Geometry*, 2002.
- [24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. In *arXiv preprint arXiv:2304.07193*, 2023.