Supplementary Material

A. Extension to Conditional Generation

The COT-FM framework naturally extends to conditional generation settings. In unconditional generation, we use unsupervised clustering to partition the target distribution. In conditional generation, the conditions themselves serve as natural cluster labels. For instance, in text-to-image generation, images corresponding to the same text prompt (e.g., "cat") naturally form a cluster. Similarly, in robot manipulation tasks, different observations correspond to different action distributions, providing an implicit cluster.

The key difference is that conditional generation may encounter unseen conditions at test time, such as input observation for robot policy. Unlike unconditional generation where we can precompute mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$ for each cluster k, we cannot enumerate all possible conditions in advance. To address this, we learn a conditional model ϕ that dynamically predicts the noise distribution for any given condition c_k:

$$\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_\phi(c_k), \boldsymbol{\sigma}_\phi^2(c_k)I), \tag{1}$$

where $\boldsymbol{\mu}_\phi(c_k)$ and $\boldsymbol{\sigma}_\phi(c_k)$ are the predicted mean and standard deviation conditioned on c_k. Since learning a full covariance matrix in high-dimensional space is prohibitively difficult and unstable, we approximate the covariance using only the predicted standard deviation. This allows COT-FM to generate samples from appropriate cluster-wise noise distributions at inference time, maintaining the benefits of reduced trajectory curvature while generalizing to novel conditions. We formulate the problem of training a conditional model ϕ that predicts mean $\boldsymbol{\mu}_\phi(c_k)$ and standard deviation $\boldsymbol{\sigma}_\phi(c_k)$ for a given condition c_k as a reinforcement learning problem within a one-step Markov Decision Process (MDP).

The one-step MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, R)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, and R the reward function. Since our framework terminates after one step, at timestep $t = 0$, the conditional model ϕ observes a state $s_0 \in \mathcal{S}$, outputs an action $a_0 \in \mathcal{A}$ and earns a reward $R(s_0, a_0)$. The objective is to maximize the expected return over all states in $\mathcal{S}$. We adopt Proximal Policy Optimization (PPO) [37] algorithm for this reinforcement learning problem.

As shown in Algorithm 1, given an offline dataset $\mathcal{D}$ with $\{(c_k, X_{1,k})\}_{k=1}^K$, our goal is to learn a conditional distribution that best matches the distribution of $X_{1,k}$ under condition c_k. We therefore treat the condition c_k as state s_0 and specify the action as drawing an initial point $\boldsymbol{x}_0$ according to c_k

$$\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_\phi(c_k), \boldsymbol{\sigma}_\phi^2(c_k)I), \tag{2}$$

Algorithm 1 Training conditional model for non-fixed condition in robot policy

Input: Trained FM model $\boldsymbol{v}_\theta$, initialized conditional model ϕ, offline dataset $\mathcal{D}$ with $\{(c_k, X_{1,k})\}_{k=1}^K$, batch size M, training steps L.
for $l = 1$ to L **do**
  Initialize minibatch: $B \leftarrow \{\}$
  for $m = 1$ to M **do**
    Sample $(c_k, \boldsymbol{x}_1)$ from dataset $\mathcal{D}$
    $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_\phi(c_k), \boldsymbol{\sigma}_\phi^2(c_k)I)$
    Compute log-likelihood $\log p_\phi(\boldsymbol{x}_0 \mid c_k)$
    Roll out FM: $\hat{\boldsymbol{x}}_1 \leftarrow \mathrm{ODESolver}(\boldsymbol{v}_\theta, \boldsymbol{x}_0, c_k)$
    $V \leftarrow \exp(-\mathrm{MSE}(\hat{\boldsymbol{x}}_1, \boldsymbol{x}_1))$
    $\hat{A} \leftarrow V - \mathbb{E}_{p_k}[V]$
    Append to minibatch: $B \leftarrow B \cup \{(c_k, \boldsymbol{x}_0, \hat{A}, \log p_\phi(\boldsymbol{x}_0 \mid c_k)\}$
  end for
  Optimize ϕ according to minibatch with PPO.
end for
Output: Trained conditional model ϕ.

as $a \in \mathcal{A}$. The sampled $\boldsymbol{x}_0$ is then propagated through the ODE to obtain a terminal point $\hat{\boldsymbol{x}}_1$. We define the reward for this transition as

$$R(s_0, a_0) = \exp(-\mathrm{MSE}(\hat{\boldsymbol{x}}_1, \boldsymbol{x}_1)). \tag{3}$$

where MSE denotes mean squared error between the generated and target terminal points. Because the MDP contains only a single transition, the value equals the reward:

$$V = \exp(-\mathrm{MSE}(\hat{\boldsymbol{x}}_1, \boldsymbol{x}_1)), \tag{4}$$

We compute the advantage as

$$\hat{A} = V - \mathbb{E}_{p_k}[V], \tag{5}$$

where p_k is the distribution from condition c_k. We then use these data to update ϕ with PPO.

During finetuning of the pretrained flow model in Algorithm 2 and during sampling in Algorithm 3, the procedure follows our original algorithm, except that the predefined $(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)$ are replaced with the conditional outputs $(\boldsymbol{\mu}_\phi(c_k), \boldsymbol{\sigma}_\phi(c_k))$.

B. Computational Cost Analysis

Theoretical Analysis. Let n be the dataset size, b the batch size, and K the number of clusters. Exact OT costs $O(n^3)$, while batch OT (used by OT-CFM) costs $O(b^3 \cdot (n/b)) = O(nb^2)$ per epoch [29]. COT-FM computes exact OT within K clusters each epoch: $O((n/K)^3 \cdot K) = O(n(n/K)^2)$. Our focus is the speed–quality tradeoff: as shown in Tab. 1, COT-FM achieves better FID at matched per-epoch OT wall-clock time.

Algorithm 2 Fine-tuning FM model for non-fixed condition in robot policy

Input: FM model $\boldsymbol{v}_\theta$, conditional model ϕ, batch size M, training steps L, offline dataset $\mathcal{D} = \{(c_k, X_{1,k})\}_{k=1}^K$.
for $l = 1$ to L **do**
  Initialize minibatch: $B \leftarrow \{\}$
  for $m = 1$ to M **do**
    Sample batch data $(c_k, \boldsymbol{x}_1) \sim \mathcal{D}$
    $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_\phi(c_k), \boldsymbol{\sigma}_\phi^2(c_k)I)$
    Append to minibatch: $B \leftarrow B \cup \{(\boldsymbol{x}_0, \boldsymbol{x}_1)\}$
  end for
  $\mathcal{L}_{\mathsf{CFM}}(\theta) = \mathbb{E}_{t,(\boldsymbol{x}_0,\boldsymbol{x}_1)\sim B}\|\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\|_2^2$
  $\theta \leftarrow \theta - \gamma\nabla_\theta\mathcal{L}_{\mathsf{CFM}}(\theta)$
end for
Output: Updated FM model $\boldsymbol{v}_\theta$.

Algorithm 3 Sampling for non-fixed condition in robot policy

Input: Conditional model ϕ, FM model $\boldsymbol{v}_\theta$, condition c_k, number of sampling steps T.
Draw source sample $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_\phi(c_k), \boldsymbol{\sigma}_\phi^2(c_k)I)$
for $t = 0$ to $T - 1$ **do**
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}_\theta(\boldsymbol{x}, \frac{t}{T}) \cdot \frac{1}{T}$
end for
Output: Generated target sample $\boldsymbol{x}$.

Table 1. Speed–quality tradeoff comparison on CIFAR-10. OT-CFM with batch size $b = 512$ and COT-FM with cluster size $n/K = 500$ yield comparable per-epoch OT computation time, but COT-FM achieves significantly better FID.

OT-CFM (Batch OT)			COT-FM (Ours)		
b	Time (sec.)	FID	n/K	Time (sec.)	FID
512	6.2	4.78	500 (50000/100)	6.3	**3.97**

Empirical Time Breakdown. We measure the wall-clock time of each preprocessing stage on CIFAR-10. As shown in Tab. 2, the one-time preprocessing takes 445 seconds in total, dominated by the reverse ODE computation. The per-epoch OT computation is approximately 15 seconds.

Table 2. Wall-clock time breakdown for one-time preprocessing on CIFAR-10.

One-time Preprocessing	Time Cost (sec.)
Reverse ODE	420
DINO feature extraction	15
K-means clustering	10

Overhead of Reverse ODE vs. Training Longer. On CIFAR-10, the reverse ODE preprocessing takes 420 seconds, equivalent to approximately 5 training epochs. Training RF for these additional epochs increases FID from 4.45 to 4.49 due to overfitting, while OT-CFM shows no improvement—FID remains at 4.78. In contrast, COT-FM achieves FID **3.97**, confirming that the preprocessing overhead is well justified by the quality gains.

C. Comparison with 2-Rectified Flow

2-Rectified Flow (2-RF) [23] is a standard rectification method that also involves a two-stage pipeline: (1) train 1-RF, then (2) generate synthetic couplings via forward ODE and retrain the model. We provide a direct comparison on CIFAR-10 to demonstrate that COT-FM is complementary to 2-RF.

As shown in Tab. 3, applying COT-FM on top of 2-RF yields additional gains: 2-RF achieves 12.21 FID at 1-step generation, while 2-RF + COT-FM achieves **10.91** FID. This demonstrates that the two approaches address different aspects of the problem—2-RF straightens trajectories via retraining on synthetic couplings, while COT-FM further reduces path crossings through cluster-wise optimal transport—and can be combined for cumulative improvement.

Table 3. 1-step FID (↓) comparison on CIFAR-10 between 2-Rectified Flow and 2-Rectified Flow enhanced with COT-FM.

Method	FID (↓)
2-Rectified Flow [23]	12.21
2-Rectified Flow [23] + COT-FM (Ours)	**10.91**

D. Additional Metrics and Ablation on K

We report FID, Inception Score (IS), Precision, and Recall on CIFAR-10 with 50 NFE steps in Tab. 4. COT-FM improves FID, IS, and Recall compared to Rectified Flow while maintaining the same Precision, indicating that COT-FM achieves broader coverage of the target distribution without sacrificing sample fidelity.

Table 4. Additional metrics on CIFAR-10 (50-step) with $K = 100$.

Method	FID ↓	IS ↑	Precision ↑	Recall ↑
Rectified Flow [23]	4.45	9.11	0.78	0.65
COT-FM (Ours)	**3.97**	**9.36**	0.78	**0.67**

We also ablate the number of clusters K in Tab. 5. The results show that COT-FM is robust to the choice of K: even with as few as $K = 5$ clusters, COT-FM (FID 4.56) outperforms OT-CFM (FID 4.78). Performance peaks around

$K = 100$ and slightly degrades at $K = 120$, suggesting that overly fine-grained clustering can reduce the number of samples per cluster and diminish the effectiveness of intra-cluster OT. In practice, we recommend applying the elbow method [43] to select K.

Table 5. Ablation on number of clusters K (CIFAR-10, 50-step FID ↓).

Method	$K = 5$	$K = 50$	$K = 100$	$K = 120$
COT-FM	4.56	4.10	**3.97**	4.06

E. Discussion on Clustering Quality

A natural concern is whether COT-FM is sensitive to the quality of the clustering. We argue that the method degrades gracefully and is robust in practice for the following reasons.

First, $K = 1$ reduces to the full-dataset setting where exact OT is intractable, necessitating batch OT as in OT-CFM. Any $K > 1$ improves over this baseline because cluster-wise OT still reduces path crossings compared to batch OT, even with imperfect cluster assignments. As demonstrated in the ablation study (Sec. D), even $K = 5$ outperforms OT-CFM.

Second, for conditional generation tasks such as robot manipulation, COT-FM does not require explicit clustering. Instead, the conditional model learns source distributions directly from task embeddings, as described in Sec. A.

Third, for unconditional generation, self-supervised features from DINO [4, 28] achieve 78.3% k-NN accuracy on ImageNet, demonstrating that semantically meaningful clusters naturally emerge and are well-separated in real-world data. This is precisely the property that COT-FM relies on: as long as the clustering captures broad semantic structure, the intra-cluster OT assignment will produce straighter transport paths than global batch OT.

F. Convergence of Alternating Optimization

Table 4(a) in the main paper shows that FID improves across iterations of the alternating optimization: 4.45 (0 iterations) → 4.23 (1 iteration) → 3.97 (2 iterations), with diminishing returns at iteration 3. Fig. 1 shows that the training loss decreases monotonically across iterations.

Intuitively, the convergence behavior is stable because the reverse ODE produces non-intersecting paths by construction, and the subsequent cluster-wise OT reduces local transport cost within each cluster. Each iteration of the alternating procedure therefore starts from a better-aligned coupling than the previous one, leading to monotonic improvement. A formal convergence analysis remains an interesting direction for future work.

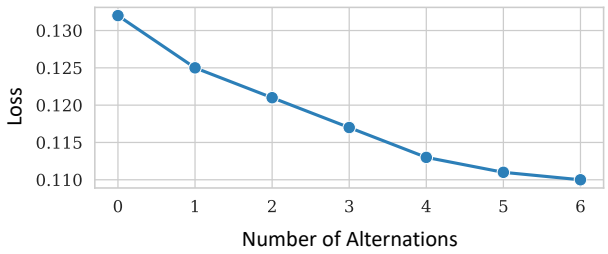

Figure 1. Training loss across alternating optimization iterations, showing monotonic decrease.

G. Related work

Diffusion Models and Flow Matching. Diffusion models [14, 17, 33, 39–41] have emerged as a powerful framework for generative modeling. These models learn a transformation that maps a prior distribution to a target distribution by progressively adding noise to the target distribution and then learning the reverse process. The reverse process is typically formulated as a stochastic differential equation (SDE) or an ordinary differential equation (ODE) [15, 41].

In contrast, Flow Matching (FM) [2, 21], learns a time-dependent velocity field that directly transports samples from the prior distribution to the target distribution. By modeling this velocity field, FM reduces the need to solve differential equations during training, providing a more direct and efficient way to learn the generative mapping.

Acceleration Strategies. Diffusion models and flow models sample slowly because of the recursive model evaluations required when solving the underlying ODE. To accelerate sampling while preserving generation quality, existing works can be broadly categorized into three directions: distilling knowledge from a pretrained model, learning a straighter velocity field, and directly learning a solution map of the ODE.

For distillation approaches [10, 24, 34, 36, 46, 47], the central idea is to distill the behavior of multi-step diffusion or score-based model into a few-step model. These methods supervise the student model using teacher trajectories, enabling efficient sampling without significantly sacrificing generation quality.

For learning straighter flows, a key challenge in training FM with a straight flow path is constructing a better coupling between the source and target distributions. The original FM formulation [21] uses random couplings for simplicity. However, random couplings induce high curvature of flow trajectories and degrade generation quality. Recent works [6, 18, 30, 38, 44] address this issue by improving the couplings using Optimal Transport (OT). Another line of work explores coupling with generated samples [23, 48].

They propose an iterative training framework that couples the source distribution with the corresponding target distribution predicted by the pre-trained model.

To directly learn the solution map of ODE [3, 9, 11, 16, 42], Consistency Model (CM) [42] learns a mapping from any point on a trajectory to its corresponding end point. Consistency Trajectory Model (CTM) [16] generalizes this idea by learning a mapping between any two points on the trajectory. Shortcut Model [9] learns discrete shortcut transitions along the trajectory, while MeanFlow [11] extends to a continuous formulation by learning the average velocity field between two points on the trajectory.

Optimized Noise Prior. Many works have adopted the strategy of optimizing the noise prior to improve the performance of diffusion or flow model. In the domain of image generation, prior works [26, 27, 35] demonstrate that a specific noise prior can induce specific pattern through diffusion model. Building on this idea, subsequent works [8, 12] enhance diffusion model performance by optimizing the initial noise using signals from reward models. Moreover, [5, 20] employ reinforcement learning (RL) and Direct Preference Optimization (DPO) [31] frameworks to modify the initial noise.

In the context of robot policy learning, several works [1, 45] show that a better prior can lead to improved performance or reduce the number of function evaluations (NFEs) required to achieve the same performance.

H. Experimental Configuration

H.1. Compute Environment

For the LIBERO robotics experiments, we use a single NVIDIA GeForce RTX 4090 GPU. For the CIFAR-10 and ImageNet experiments, we use the AMD AI & HPC Clusters Taiwan, equipped with AMD Instinct MI300X GPUs (192 GB HBM3) and AMD EPYC 9684X 96-Core Processors. GPUs within each node are interconnected via AMD Infinity Fabric (XGMI). CIFAR-10 experiments are conducted on a single MI300X GPU, while ImageNet experiments use 8 MI300X GPUs on a single node. All AMD experiments use ROCm 7.2.0.

H.2. 2D Point Cloud

We evaluate several flow-based generative models: Rectified Flow [23], OT-CFM [44], MeanFlow [11], and our COT-FM on three 2D benchmarks: Mixture of 5 Gaussians, Two Moons, and a Checkerboard Grid. The source distribution is a single Gaussian with mean $(0, 0)$ and standard deviation 0.6, while the target distribution follows the corresponding toy structure.

All methods use the same network architecture for fair comparison. We train each model using the Adam optimizer with learning rate 1e-3, batch size 512, and run 500 epochs for each task. In each iteration, source–target mini-batches are sampled, and the corresponding coupling strategy is applied. For evaluation, we measure: (1) Wasserstein distance between generated and target samples using exact 2-Wasserstein distance; (2) trajectory curvature, computed by discretizing the integrated trajectories, normalizing tangents, and measuring second-order directional change. Lower curvature corresponds to straighter probability paths.

H.3. CIFAR-10

We experiment with unconditional generation on CIFAR-10 [19], where the input to the model is $32 \times 32 \times 3$ in pixel space. We evaluate Fréchet Inception Distance (FID) [13] using 50K generated images. Optimal Transport is computed using the publicly released implementation from [44].

To ensure a fair comparison with Rectified Flow [23], our implementation follows their setting. We train the models for 600 epochs with a batch size of 128 using Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, and no weight decay. The learning rate is set to 2×10^{-4} and we apply gradient clipping with a maximum norm of 1.0. The backbone is an NCSN++-style UNet, and detailed parameters are provided in Table 6. We maintain exponential moving average (EMA) weights with decay rate of 0.999999.

For comparison with MeanFlow [11], we adopt their experimental configuration. We train the models for 240 epochs with a batch size of 512. We used Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and no weight decay. The learning rate is set to 0.0001. The UNet architecture follows [40], with model parameters summarized in Table 6. We maintain EMA weights with a decay rate of 0.99999.

H.4. ImageNet

We experiment with conditional generation on ImageNet [7] 256×256 with full dataset. The input to the model is $32 \times 32 \times 4$ in latent space obtained from the VAE. We evaluate Fréchet Inception Distance (FID) [13] using 50K generated images. Optimal Transport is computed using the publicly released implementation from [44].

Our implementation is slightly modified from the setting of a non-official PyTorch implementation [49] to 1-RF. The images are augmented through horizontal flipping. Both 1-RF and COT-FM are trained for 190 epochs in total; COT-FM is fine-tuned from a pre-trained RF model after 160 epochs, requiring only 30 additional epochs of training. We use a batch size of 256 with the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1 \times 10^{-8}$ and no weight decay. We set the learning rate to 1×10^{-4} and apply gradient clipping with a maximum norm of 1.0. The backbone is Scalable Interpolant Transformers(SiT) [25]. We maintain EMA weights with a decay rate of 0.9999. We set ω to 2.5 and 2 for SiT-B/4 and SiT-B/2, respectively. All other hyperparameters remain

identical to the original configuration.

H.5. LIBERO

The LIBERO benchmark [22] uses a Franka Emika Panda robot with end-effector control and camera control. We evaluate on Spatial for spatial relationships and Long for long horizon tasks. Each task starts with 50 different initial points and we record the average success rate for each task with 3 different seeds. The hyperparameters are the same as FLOWER [32], shown in Table 8.

We train the reinforcement learning model over 15 passes of the offline dataset. After collecting 51,200 samples in each pass, we perform Proximal Policy Optimization (PPO) [37] training for the conditional model, where condition c_k contains language instructions and the current visual observations. During PPO training, the discount factor γ is set to 0.99, the clipping parameter ϵ to 0.2, and the learning rate to 1×10^{-5}. Each PPO training session runs for 10 epochs.

For finetuning FLOWER, we follow the setting reported in FLOWER, as summarized in Table 7. To ensure training stability, we only finetune the Action Encoders, Action Heads, and Flow Transformer.

I. Additional Generated Results

We present additional generated images for CIFAR-10 in this section. We use the same noise for those sampled from $\mathcal{N}(0, I)$ and same noises for those sampled from our method for consistency.

Component	Rectified Flow	MeanFlow
Channels	128	128
Channel multiple	(1, 2, 2, 2)	(2, 2, 2)
Residual blocks per resolution	4	4
Normalization	GroupNorm	GroupNorm
Nonlinearity	Swish	Swish
Attention resolution	16	16
Dropout	0.15	0.2
Embedding type	Fourier	Positional

Table 6. UNet architecture for CIFAR-10 experiments.

Components	Number of Parameters
ViT	360M
VLM	205M
Action Encoders	3.2M
Action Heads	31.8K
Global-AdaLN	28.3M
Cond Linear Proj.	1.0M
Timestep Embedder	1.3M
Cond Norm	1.0K
FreqEmbedder	1.3M
Action Space Embedder	1.3M
Flow Transformer	339M
ϕ (ours)	17M
Total Parameters FLOWER	**964M**

Table 7. The parameters of all model components in FLOWER.

Settings	LIBERO
Action Space Encoders	2-layered MLP
Action Space Decoders	Linear Attention
Number of Flow-T Layers	18
Latent Dimension	1024
Number of Heads	16
Position Embedding	1D Rope
Sampling Distribution	Uniform
Attention Dropout	0.1
MLP Dropout	0.1
Residual Dropout	0.1
Act Seq Length	10
Denoising Steps	4
Multistep	4
Camera Views	[Primary Static, Wrist]
Use Proprio	False
Action Space	Delta EEF
Frequency	10

Table 8. LIBERO hyperparameters of FLOWER.

Figure 2. Visualization of CIFAR-10 for different flow-based methods under 10-step and 50-step generation settings.

Figure 3. Visualization of CIFAR-10 for MeanFlow under 1-step generation settings.

Acknowledgment

This work was supported in part by the AMD–ITRI Joint Laboratory, which provided MI300X high-performance computing resources and technical support for the execution and validation of this research. This work was also supported by the AMD University Program AI & HPC Cluster. We further acknowledge Kuo-Guang Tsai for his technical support on the AMD system cluster infrastructure. This research was also supported by the National Science and Technology Council (NSTC), Taiwan, under Grants 114-2222-E-002-008, 114-2221-E-002-182-MY3, 113-2221-E-002-201, and 115-2634-F-002-001.

References

[1] Tomoharu Aizu, Takeru Oba, Yuki Kondo, and Norimichi Ukita. Robot motion planning using one-step diffusion with noise-optimized approximate motions. *arXiv preprint arXiv:2504.19652*, 2025. 4

[2] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations (ICLR)*, 2023. 3

[3] Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research (TMLR)*, 2024. 4

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021. 3

[5] Changgu Chen, Libing Yang, Xiaoyan Yang, Lianggangxu Chen, Gaoqi He, Changbo Wang, and Yang Li. Find: Fine-tuning initial noise distribution with policy optimization for diffusion models. In *ACM International Conference on Multimedia (ACM MM)*, 2024. 4

[6] Aram Davtyan, Leello Tadesse Dadi, Volkan Cevher, and Paolo Favaro. Faster inference of flow-based generative models via improved data-noise coupling. In *International Conference on Learning Representations (ICLR)*, 2025. 3

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 4

[8] Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 4

[9] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations (ICLR)*, 2025. 4

[10] Zhengyang Geng, Ashwini Pokle, and J Zico Kolter. One-step diffusion distillation via deep equilibrium models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3

[11] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 4

[12] Xiefan Guo, Jinlin Liu, Miaomiao Cui, Jiankai Li, Hongyu Yang, and Di Huang. Initno: Boosting text-to-image diffusion models via initial noise optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 4

[13] 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. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 4

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[15] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[16] Dongjun Kim, AI Sony, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *International Conference on Learning Representations (ICLR)*, 2024. 4

[17] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 3

[18] Nikita Kornilov, Petr Mokrov, Alexander Gasnikov, and Aleksandr Korotin. Optimal flow matching: Learning straight trajectories in just one step. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 3

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. University of Toronto. 4

[20] Zeming Li, Xiangyue Liu, Xiangyu Zhang, Ping Tan, and Heung-Yeung Shum. Noisear: Autoregressing initial noise prior for diffusion models. *arXiv preprint arXiv:2506.01337*, 2025. 4

[21] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. 3

[22] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 5

[23] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023. 2, 3, 4

[24] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021. 3

[25] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 4

[26] Jiafeng Mao, Xueting Wang, and Kiyoharu Aizawa. Guided image synthesis via initial image editing in diffusion model. In *ACM International Conference on Multimedia (ACM MM)*, 2023. 4

[27] Jiafeng Mao, Xueting Wang, and Kiyoharu Aizawa. The lottery ticket hypothesis in denoising: Towards semantic-driven initialization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 4

[28] 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. *Transactions on Machine Learning Research (TMLR)*, 2024. 3

[29] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 2019. 1

[30] Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky TQ Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning (ICML)*, 2023. 3

[31] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2024. 4

[32] Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. FLOWER: Democratizing generalist robot policies with efficient vision-language-flow models. In *Conference on Robot Learning (CoRL)*, 2025. 5

[33] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, 2015. 3

[34] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022. 3

[35] Dvir Samuel, Rami Ben-Ari, Simon Raviv, Nir Darshan, and Gal Chechik. Generating images of rare concepts using pre-trained diffusion models. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2024. 4

[36] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 3

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 1, 5

[38] Andreas Sochopoulos, Nikolay Malkin, Nikolaos Tsagkas, João Moura, Michael Gienger, and Sethu Vijayakumar. Fast flow-based visuomotor policies via conditional optimal transport couplings. In *Conference on Robot Learning (CoRL)*, 2025. 3

[39] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015. 3

[40] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 4

[41] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2020. 3

[42] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023. 4

[43] Robert L. Thorndike. Who belongs in the family? *Psychometrika*, 1953. 3

[44] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research (TMLR)*, 2024. 3, 4

[45] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2025. 4

[46] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[47] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *International Conference on Machine Learning (ICML)*, 2024. 3

[48] Huminhao Zhu, Fangyikang Wang, Tianyu Ding, Qing Qu, and Zhihui Zhu. Analyzing and mitigating model collapse in rectified flow models. *arXiv preprint arXiv:2412.08175*, 2025. 3

[49] Yu Zhu. Meanflow: Pytorch implementation. https://github.com/zhuyu-cs/MeanFlow, 2025. PyTorch implementation of Mean Flows for One-step Generative Modeling. 4