ALIGNFLOW: IMPROVING FLOW-BASED GENERATIVE MODELS WITH SEMI-DISCRETE OPTIMAL TRANSPORT

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-based Generative Models (FGMs) effectively transform noise into a data distribution, and coupling the noise and data in the training of FGM by Optimal Transport (OT) improves the straightness of the flow paths. However, existing OT-based couplings are difficult to combine with modern models and/or to scale to large datasets due to the curse of dimensionality in the sample complexity of (batch) OT. This paper introduces AlignFlow, a new approach using Semi-Discrete Optimal Transport (SDOT) to enhance FGM training by establishing explicit alignment between noise and data pairs. SDOT computes a transport map by partitioning the noise space into Laguerre cells, each mapped to a corresponding data point. During the training of FGM, i.i.d.-sampled noise is matched with corresponding data by the SDOT map. AlignFlow bypasses the curse of dimensionality and scales effectively to large datasets and models. Our experiments demonstrate that AlignFlow improves a wide range of state-of-the-art FGM algorithms and can be integrated as a plug-and-play solution with negligible additional cost.

1 INTRODUCTION

The generative model is a machine learning task that generates new data that resembles the given dataset. This task is important and has seen great progress over the past decades, e.g., ChatGPT (Achiam et al., 2023) for natural language and Stable Diffusion (Rombach et al., 2022) for image generation. In addition to autoregressive models that dominate language modeling, other backbone algorithms for generative modeling include GANs (Goodfellow et al., 2020), normalizing flows (Rezende & Mohamed, 2015), regression models (e.g., GPT (Radford et al., 2018), LLaMA (Touvron et al., 2023)), diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021), and Flow-based Generative Models (FGMs) (Liu et al., 2022; Lipman et al., 2022; Albergo et al., 2023).

This work will focus on improving a wide range of FGMs, including flow matching (Lipman et al., 2022), shortcut model (Frans et al., 2024), MeanFlow (Geng et al., 2025), Live Reflow Frans et al. (2024), but excluding continuous normalizing flows (CNF) (Chen et al., 2018; Albergo & Vanden-Eijnden, 2022) (see Sec. 3.1 for more specifications). FGMs focus on learning a time-dependent vector field, approximated by a neural network whose integration is a trajectory that transits a randomly sampled noise to newly generated data.

Despite their ability to generate high-quality samples, a major drawback of FGMs is the high computational cost associated with sampling. The inference process involves integrating an ODE, where each integration step requires a forward pass through the neural network. As a result, generating a single sample requires multiple neural network evaluations. This cost is measured by the Number of Function Evaluations (NFE), which refers to the number of forward passes. For vanilla flow matching, the NFE is typically greater than 100.

FGMs' training procedure generally consists of the following three steps: (1) randomly sample noise and data points; (2) compute the target vector field that the neural network aims to approximate; and (3) update the model parameters via an optimization step. For further details, refer to Algorithms 1 and 4. In this work, we focus on improving the first step. While many state-of-the-art methods have

proposed sophisticated designs for the target vector field, they often rely on independently sampling noise and data pairs. However, this independent pairing has been shown to inherently induce curved trajectories (Liu et al., 2022; Hertrich et al., 2025), leading to high NFE. In other words, the random matching of data and noise inherently encourages non-straight generative paths.

To address the limitations of random noise–data matching, we propose AlignFlow, a method that aligns noise and data using semi-discrete optimal transport (SDOT) to guide the target vector fields learned by the neural network. SDOT computes the optimal transport (OT) plan from a continuous noise distribution to a discrete dataset. This results in a fixed mapping that defines the shortest and most direct connection from any sampled noise point to a corresponding data point. Our training procedure adopts a two-stage approach: the first stage computes the SDOT map, and the second stage trains the FGM using any target vector field of choice. However, instead of randomly pairing i.i.d. noise with uniformly sampled data, we match each noise sample to a data point as prescribed by the SDOT map. A high-level overview is provided in Algorithm 3. For implementation details such as class-conditioned generation and data augmentation, please refer to Section 3.5. We outline the key benefits below, with further discussion in Section 4.

- AlignFlow is a *plug-and-play method*, making it easy to integrate into existing FGMs. It can be readily combined with state-of-the-art training techniques to further enhance performance.
- AlignFlow *bypasses the curse of dimensionality* (Sec. 4.1), thus scales well to large-scale models and datasets in high-dimensional spaces.
- The SDOT map defines a deterministic and optimal transport path without randomness; that is, each noise sample is consistently matched to a fixed data point, independent of batch size. This property makes AlignFlow particularly suitable for large models that require small-batch training and, intuitively, also contributes to faster convergence (see Sec. 4.2).
- The extra cost for computing the SDOT map is low (less than 1% extra cost) (Sec. 4.3).

AlignFlow improves FGMs and bypasses the curse of dimensionality problem in OT. Consider the task of estimating an OT plan between the unknown data distribution $\tilde{p}_1$ and a known noise distribution p_0. This task is challenging not merely due to computational limitations, but rather stems from the inherent statistical limitations imposed by the finite size of the dataset. (For further discussion, see Sec. 4.1.)

Theorem 1 (Sample complexity in OT (Informal version for Thm. 1 in Fournier & Guillin (2015)))**.** *In a d-dimensional space, the error in estimating the p-Wasserstein distance $W_p(\tilde{p}_1, p_0)$ between a known distribution p_0 and an unknown distribution $\tilde{p}_1$ with only access to $|I|$ samples is of order $\sim |I|^{-p/d}$.*

Thm. 1 shows that the number of samples required grows exponentially w.r.t. dimensionality, which is known as curse of dimensionality. Importantly, this limitation is fundamental: the only way to reduce this error is to increase the dataset size, which is often infeasible in practice. Therefore, any approach that attempts to estimate an OT plan between the population distribution $\tilde{p}_1$ and p_0 cannot scale effectively to modern, high-dimensional generative modeling tasks (see Sec. E).

Bypassing the Curse of Dimensionality. OT is widely regarded as highly relevant to generative modeling, as it characterizes transformations between distributions via straight and efficient transport paths, a desirable property in FGMs. Unlike prior (batch) OT-based approaches, AlignFlow bypasses the curse of dimensionality by computing a Semi-Discrete OT (SDOT) plan between the empirical dataset (a discrete distribution) and the noise distribution (a known continuous distribution). Since both distributions are explicitly known, this formulation allows for an accurate and tractable transport plan, even in high-dimensional settings.

2 More Related Works

A central factor in reducing the NFE during sampling in FGMs is the straightness of the generative path. Since the inference process involves integrating a learned vector field, straighter trajectories are easier to integrate accurately, thereby requiring fewer NFEs. Numerous methods have been proposed to encourage straighter paths, and we categorize these efforts into three main approaches:

	Algorithm	No Extra tuning	Scale to large models	No Curse of Dim	OT based
Coupling	Tong et al. (2023)	✓	✗	✗	✓
	Liu et al. (2022)	✓	✓	-	✗
	Kornilov et al. (2024)	✗	✗	✗	✓
Noise-data alignment	**AlignFlow (ours)**	✓	✓	✓	✓

Table 1: A comparison between coupling methods and noise-data alignment.

Target vector field: This class of methods aims to straighten generative paths by guiding the neural network to learn a smoother or more linear target vector field. Approaches such as consistency training and the Shortcut model (Frans et al., 2024) adopt techniques from diffusion models (Yang et al., 2024), enforcing penalties on inconsistencies between the forward and backward segments of the path. MeanFlow (Geng et al., 2025) introduces a stronger regularization term based on the Jacobian-vector product. These advanced loss functions have proven effective, reducing the NFE to as low as 4 and even 1 for class-conditioned generation on the ImageNet dataset.

Distillation: Recent works aim to reduce NFE by distilling a trained FGM into a more efficient model (Boffi et al., 2025; Dao et al., 2025). Distillation techniques have also shown success in diffusion models (Salimans & Ho, 2022; Song et al., 2023; Kim et al., 2023). However, distillation involves an additional training stage built upon a pre-trained model, and thus can be viewed as complementary to our approach.

Coupling: FGMs noise and data independently (Algo. 1). Many pioneers improve FGMs by better coupling, i.e., choosing the noise and data from a carefully designed joint distribution:

- Kornilov et al. (2024) trains an Input Convex Neural Network (ICNN) (Amos et al., 2017) to serve as the Brenier potential (Peyré et al., 2019, Thm. 2.1), thereby providing the OT map between the real data distribution and the noise distribution. In this framework, the ICNN and the FGM are trained jointly. However, the inclusion of the ICNN introduces significant computational overhead, making it challenging to scale to large models and high-dimensional tasks.
- Tong et al. (2023) employs Sinkhorn iterations (Peyré et al., 2019, Sec. 4.2) to compute the OT plan between i.i.d. sampled noise and data at each training step. However, this approach is sensitive to batch size: large batches incur high computational cost, as Sinkhorn is an iterative algorithm with per-minibatch complexity of $\mathcal{O}(n^2)$; meanwhile, small batches limit the quality of the estimated coupling. Empirical results suggest that this method struggles to generalize to class-conditioned generation[1].
- Liu et al. (2022) proposes to disentangle crossing trajectories in the learned vector field to promote straighter generative paths. This approach has recently been scaled to larger models by Esser et al. (2024), although the scaling process is non-trivial. While the method does not directly solve an OT problem, its underlying formulation is closely related to OT, as discussed in Theorem 3.5 of their paper.

3 METHODOLOGY

Mathematically, the generative modeling task can be formulated as follows: given a dataset $\{x_i\}_{i\in I}$, assumed to consist of i.i.d. samples from an unknown probability distribution $\tilde{p}_1$ on a space $\mathcal{X}$,[2] the goal is to generate new samples that follow the same distribution $\tilde{p}_1$ using only the observed dataset $\{x_i\}_{i\in I}$.

Notation Throughout this paper, we will use p_0 to denote the noise distribution (some distribution that is easy to sample from, e.g. normal distribution) and p_1 is the Dirac distribution corresponding

[1] https://github.com/atong01/conditional-flow-matching/issues/117

[2] In many settings, $\mathcal{X}$ is taken to be a latent space, and the data is obtained via encoding through a VAE.

to the dataset, i.e., $p_1 = \sum_{i \in I} b_i \delta_{x_i}$ where the weights are uniform in our setting, i.e., $b_i \equiv \frac{1}{|I|}$. $i \in I$ is the index for the dataset, while $j \in J$ is the index for the samples in the minibatch. $\tilde{p_1}$ is the unknown real data distribution. In all cases, we will use superscripts for the dataset index and subscripts for time.

3.1 Flow-based Generative Models (FGMs)

We first summarize a general FGM framework in Algo. 1. In each iteration of the training process, we first sample a set of noise x_0, data x_1 and time t. Then we compute x_t as the interpolation between x_0 and x_1. Finally, we let a NN $u(x_t, t; \theta)$ to approximate a target vector field $\mathrm{TargetVectorField}$ and update the NN parameters.

Algorithm 1: Flow-based Generative Model (Training)

Input: Source distribution p_0, dataset $\{x_1^i\}_{i \in I}$, neural network $u(x, t; \theta)$
Output: Learned velocity field $u(\cdot, \cdot; \theta)$

1 Hyperparameters: batch size B, number of steps K
▷ Training loop
2 for $k = 1 \dots K$ do
3 Sample i.i.d. $\{x_0^j\}_{j=1}^B \sim p_0$ ▷ sample noise
4 Sample minibatch $\{x_1^{m_j}\}_{j=1}^B$ from dataset $\{x_1^i\}_{i \in I}$ ▷ sample data
5 $t^j \sim \mathcal{U}(0, 1)$ for $j = 1, \dots, B$ ▷ sample time
6 for $j = 1 \dots B$ do
7 $x_t^j \leftarrow (1 - t^j) x_0^j + t^j x_1^{m_j}$
8 $v^j = \mathrm{TargetVectorField}(x_0^j, x_1^{m_j})$
9 $\hat{v}^j = u(x_t^j, t^j; \theta)$
10 $L(\theta) = \mathrm{Loss}\left(\{\hat{v}^j, v^j\}_{j=1}^B\right)$ ▷ objective function
11 Update θ using $\nabla_\theta L$ ▷ optimization

(Vanilla) Flow Matching Lipman et al. (2022) chooses $\mathrm{TargetVectorField}(x_0, x_1) = x_1 - x_0$ and $\mathrm{Loss}\left(\{\hat{v}^j, v^j\}_{j=1}^B\right) = \frac{1}{B} \sum_j \left\|\hat{v}^j - v^j\right\|_2^2$. The Shortcut Model (Frans et al., 2024), MeanFlow (Geng et al., 2025), consistency training (Frans et al., 2024), Live Reflow Frans et al. (2024) et al. all fit into the above FGM framework.

In line 3 and 4 of Algo. 1, noise and data points are sampled independently, leading to curved trajectories intrinsically Liu et al. (2022); Hertrich et al. (2025). To address this limitation, we sample noise-data pairs from the joint distribution computed by SDOT.

3.2 Coupling between noise and data

In fact, the loss function Algo. 1 optimizes estimates the following expectation [3]:

$$L(\theta) \approx \mathcal{L}(\theta) = \mathbb{E}_{t \sim \mathrm{Unif}[0,1], x_0 \sim p_0, x_1 \sim p_1} \|u(x_t, t; \theta) - \mathrm{TargetVectorField}(x_0, x_1)\|_p^p \quad (1)$$

This means x_0 (sampled from p_0) and x_1 (sampled from p_1) are independent in Algo. 1, i.e., $(x_0, x_1) \sim p_0 \times p_1$. Recent works, such as (Pooladian et al., 2023; Liu et al., 2022; Tong et al., 2023), are trying to find more powerful joint distributions, and smaple (x_0, x_1) from any $\gamma \in \Gamma(p_0, p_1)$:

$$\mathcal{L}_\gamma(\theta) = \mathbb{E}_{t \sim \mathrm{Unif}[0,1], (x_0, x_1) \sim \gamma} \|u(x_t, t; \theta) - \mathrm{TargetVectorField}(x_0, x_1)\|_p^p \quad (2)$$

where $\Gamma(p_0, p_1)$ is the set of all possible joint distribution of p_0 and p_1:

$$\Gamma := \left\{ \gamma \in \mathcal{P}(\mathcal{X} \times \mathcal{X}) : \int \gamma(x_0, x_1)\, \mathrm{d}x_0 = p_1(x_1) \quad \forall x_1, \int \gamma(x_0, x_1)\, \mathrm{d}x_1 = p_0(x_0) \quad \forall x_0 \right\} \quad (3)$$

[3] We here choose the loss function to be $\mathrm{Loss}\left(\{\hat{v}_t^i, v_t^i\}_{i=1}^B\right) := \frac{1}{B} \sum_i \left\|\hat{v}_t^i - v_t^i\right\|_p^p$ for simplicity

Each element in Γ is referred to as a coupling between p_0 and p_1. As evident from the definition, Γ is a vast set. Although training with any valid coupling theoretically yields a correct vector field, the straightness of the resulting trajectories and the efficiency of the training process can vary significantly depending on the choice of coupling. This naturally raises the question: which coupling should we choose? OT is widely believed to provide meaningful guidance in addressing this question. In the next section, we will derive a specific coupling γ based on OT theory. [4]

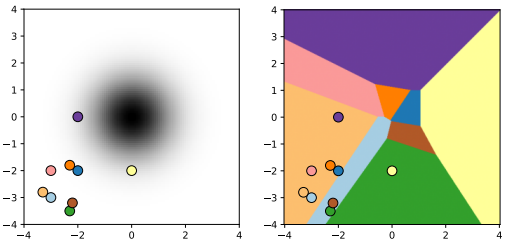
3.3 Semi-Discrete Optimal Transport

The Optimal Transport (OT) problem seeks to compute the optimal coupling between two probability distributions by minimizing a given cost function $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, (see, e.g., Peyré et al. (2019) for a comprehensive overview):

$$\gamma_* := \arg\min_{\gamma \in \Gamma(q_1, q_2)} \left(\int_{\mathcal{X}\times\mathcal{X}} c(y_1, y_2)\, \mathrm{d}\gamma(y_1, y_2) \right), \tag{4}$$

where we choose $c(y_1, y_2) := \|y_1 - y_2\|^2$ throughout the paper. [5]

A **discrete distribution** is the opposite of a continuous distribution (e.g., normal distribution), meaning that the random variable only takes finite (or countable) values, e.g., the dataset distribution $p_1 = \frac{1}{|I|}\sum_{i\in I} \delta_{x_i}$. A OT problem between a continuous distribution and a discrete distribution is called **Semi-Discrete Optimal Transport (SDOT)** (Peyré et al., 2019, Sec. 5), and this will become our main tool for noise-data alignment.



(a) Dataset and noise distribution (b) Laguerre cells

Figure 1: Visualization of Laguerre cells in 2-dim. The noise distribution is the normal distribution (dark shadow in the left figure), and the dataset is the points in the lower left corner. The whole space is partitioned into cells using SDOT, and each region is mapped to the data point with same color by the SDOT map. The integral of the probability of the noise distribution in each Laguerre cell equals the probability of the corresponding data point.

Unlike general OT problems, the transport plan of SDOT problems can be represented by a $|I|$-dim vector $\mathbf{g} = [g_i]_{i\in I}$ called **dual weight**, where $|I|$ is the number of points in the discrete distribution p_1. Given the dual weight $\mathbf{g}$, the SDOT plan is $\varphi(\cdot; \mathbf{g}) : \mathcal{X} \to I$

$$\varphi(x_0; \mathbf{g}) := \arg\min_{i\in I} \; c(x_0, x_1^i) - g_i \tag{5}$$

where $\mathbf{g}$ will be omited when no ambiguity.

In fact, $\mathcal{X}$ is partitioned into cells by the SDOT map in the sense that cell L_i contains the points transported to the i-th point in the dataset. Such a partition is called Laguerre cells L_i (Fig. 1).

$$\mathsf{L}_i(\mathbf{g}) := \{x \in \mathcal{X} : c(x, y_i) - g_i \le c(x, y_j) - g_j, \forall j\} \tag{6}$$

Now we need to discuss how to compute the dual weight g. SDOT, same as general OT problems in Eq. (4), is a minimization problem. By analyzing its dual problem, the dual weight can be solved by maximizing the following objective function utilizing the Laguerre cell (see e.g., Eq. 5.7 in Peyré et al. (2019)):

$$\mathcal{E}(\mathbf{g}) := \sum_{i\in I} \int_{\mathsf{L}_i(g)} (c(x, y_j) - g_j)\, \mathrm{d}p_0(x) + \langle \mathbf{g}, \mathbf{b} \rangle \tag{7}$$

whose gradient is given by $\nabla\mathcal{E}(\mathbf{g})_i = -\int_{\mathsf{L}_i(\mathbf{g})} \mathrm{d}p_0 + b_i$, where b_i is the probability for each point. In our cases, $b_i \equiv \frac{1}{|I|}$.

[4] Formally, any joint distribution over two marginals is a coupling. However, in the context of FGMs, we distinguish between two types: we refer to the joint distribution between noise p_0 and the dataset p_1 as noise–data alignment, while we use coupling to refer specifically to the joint distribution between p_0 and the unknown data distribution $\tilde{p}_1$.

[5] The minimum value is $W_2^2(q_1, q_2) := \min_{\gamma\in\Gamma(q_1,q_2)} \left(\int_{\mathcal{X}\times\mathcal{X}} \|y_1 - y_2\|^2 \mathrm{d}\gamma(y_1, y_2) \right)$.

To solve this maximization problem, we use Adam (Kingma & Ba, 2014) to maximize it. Although such a solution for the SDOT problem is not new (e.g., Peyré et al. (2019)), we newly propose an efficient EMA estimation for MRE and L1, which will be helpful when tuning the hyperparameters ϵ, β, lr, and justify the performance of the output dual weight. For more discussion, see Sec. A.

Algorithm 2: Dual weight computation for SDOT map

Input: Source distribution p_0, dataset $\{x^i\}_{i\in I}$ and the corresponding probabilities $\mathbf{b} = [b^i]_{i\in I}$, entropic regularization strength ϵ, EMA parameter β, batch size B, cost function c

Output: Dual weight $\mathbf{g} = [g_i]_{i\in I}$

Initialization: $\nabla\mathcal{E}_{\texttt{ema}} = \mathbf{0}$, $\mathbf{g} = \mathbf{0}$, $\mathbf{g}_{\texttt{ema}} = \mathbf{0}$

for *step* $= 1, 2\ldots$ do

  Sample i.i.d. $\{x_0^j\}_{j=1}^B \sim p_0$ ▷ `sample noise`

  for $j = 1, \ldots, B$ do

    if $\epsilon \neq 0$ then

      $h_j = \mathrm{SoftMax}_{i\in I}\left(-\frac{c(x_0^j, x_1^i) - g_i}{\epsilon}\right)$ ▷ `SDOT map with current` $\mathbf{g}$

    else

      $\varphi(x_0^j; \mathbf{g}) = \arg\min_{i\in I} \;\; c(x_0^j, x_1^i) - g_i$

      $h_j = \mathbb{1}_{\varphi(x_0^j;\mathbf{g})}$

  $\nabla\mathcal{E}(\mathbf{g}) = \frac{1}{B}\sum_j h_j - \mathbf{b}$

  $\nabla\mathcal{E}_{\texttt{ema}} = \beta\nabla\mathcal{E}_{\texttt{ema}} + (1-\beta)\nabla\mathcal{E}(\mathbf{g})$ ▷ `Smoothen by EMA`

  Update $\mathbf{g}$ using $\nabla\mathcal{E}_\epsilon(\mathbf{g})$ ▷ `optimization`

  $\mathbf{g}_{\texttt{ema}} = \beta\mathbf{g}_{\texttt{ema}} + (1-\beta)\mathbf{g}$

Return: dual weight $\mathbf{g}_{\texttt{ema}}$

After the dual weight $\mathbf{g}$ is computed, the SDOT map $\mathcal{X} \to I$ by Eq. (5).

3.4 Main algorithm

Having established the necessary technical framework, we now proceed to the natural derivation of the AlignFlow methodology. As noted in the introduction, computing the OT plan between the unknown true data distribution $\tilde{p_1}$ and the noise distribution p_0 is infeasible because the OT sample complexity is severely hampered by the curse of dimensionality (Thm. 1). This core difficulty arises from the fact that we have only limited empirical samples available from $\tilde{p_1}$, whose underlying form remains unknown.

The critical question then becomes: How can we effectively bypass the sample complexity limitations imposed by the curse of dimensionality?

Idea for AlignFlow The critical insight behind AlignFlow is to circumvent the challenge posed by the unknown true data distribution $\tilde{p_1}$ by focusing on the known empirical distribution p_1, which is characterized by a Dirac distribution over the dataset samples. The OT plan between p_1 and p_0 retains the desirable properties of OT, including a straight path. Furthermore, since the empirical distribution p_1 is inherently discrete, the OT problem can be computed efficiently by Algo. 2.

This leads to the AlignFlow (Algo. 3), which uses the SDOT map to compute the noise-data alignment. Note that in the derivation of Algo. 3, we never use the unknown $\tilde{p_1}$. This is a theoretical benefit for AlignFlow: we do not require any assumption about the real data distribution $\tilde{p_1}$. And this is the key to bypassing the curse of dimensionality.

3.5 Additional techniques

Remark 1 (Noise storage)**.** *In line 7 and 9 in Algo. 3, we need to sample a large amount of noise ahead before we start training the FGM model, Simply saving them in memory, even on disks, will be almost impossible.* [6]*. Our solution is to only save the random seed generating the noise, i.e., each noise-data pair is (seed, index).*

[6] According to our tests, 10 epochs of noise for ImageNet training in latent space (Sec. 5.3 and 5.2) will take terabytes of disk space. Besides disk space, IO will be a huge problem.

Algorithm 3: AlignFlow: noise-data alignment by SDOT (Training)

Input: Source distribution p_0, dataset $\{x_1^i\}_{i\in I}$, neural network $u(x,t;\theta)$
Output: Learned velocity field $u(\cdot,\cdot;\theta)$
Hyperparameters: batch size B, number of steps K
* Stage 1: compute SDOT map *
Run Algo. 2 to get dual weight $\mathbf{g}$.
* Stage 2: Train flow-based generative model *
Let $M = K \cdot B$
Sample i.i.d. $\{x_0^j\}_{j=1}^M \sim p_0$ ▷ sample noise
$m_j = \varphi(x_0^j)$ for $j = 1,\dots,M$ ▷ match noise to data
$t^j \sim \mathcal{U}(0,1)$ for $j = 1,\dots,M$ ▷ sample time
$\{m_j\}_{j=1}^M = \text{Rebalance}\left(\{m_j\}_{j=1}^M\right)$ ▷ Only if needed. Sec. F
▷ Training loop
for $k = 1 \dots K$ do
  for $l = 1 \dots B$ do
    $j = (k-1)\cdot B + l$
    $x_t^j \leftarrow (1-t^j)x_0^j + t^j x_1^{m_j}$
    $v^j = \text{TargetVectorField}(x_0^j, x_1^{m_j})$
    $\hat{v}^j = u(x_t^j, t^j; \theta)$
  $L(\theta) = \text{Loss}\left(\{\hat{v}^j, v^j\}_{j=(k-1)\cdot B+1}^{k\cdot B}\right)$ ▷ objective function
  Update θ using $\nabla_\theta L$ ▷ optimization

Such an approach will require a map from seed to random matrices (supported by Jax) and loading the whole ImageNet latent to memory for random fetch (automatically optimized by the PyTorch dataloader). By such an approach, the (seed, index) pairs for 500 epochs of ImageNet training will only cost $\sim 1GB$ disk space.

Remark 2 (Data augmentation)**.** *Data augmentation is a critical component for achieving optimal performance in image-related tasks. However, incorporating complicated augmentation techniques, such as random cropping or random rotation, directly into the SDOT map formulation can be challenging.*

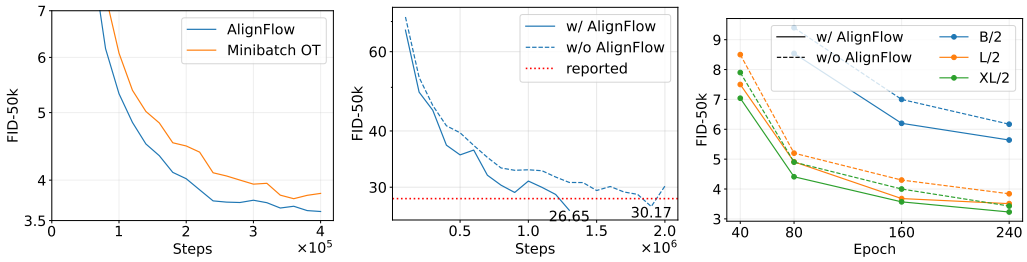
Fortunately, for most state-of-the-art image generation tasks, the necessary data augmentation is often limited to a random horizontal flip for peak performance. This specific case can be elegantly managed without complex modification to the SDOT framework: we can simply redefine the dataset as the union of two subsets: the original images and their horizontally flipped counterparts.

Remark 3 (Class-conditioned generation)**.** *For class-conditioned tasks, such as those discussed in Sections 5.2 and 5.3, we assume the data follows a class-specific distribution, denoted $p_{1,c}$, for the c-th class. The procedure involves computing the SDOT map between the base noise distribution p_0 and each class-specific data distribution $p_{1,c}$. Subsequently, after the noise is generated, the method proceeds to create (noise, data) pairs and performs the requisite rebalance operation independently for each class.*

4 Advantage of AlignFlow

4.1 Bypass Curse of dimensionality

The curse of dimensionality (Thm. 1) originates from the difficulty of empirically estimating the true, unknown data distribution. The AlignFlow framework, however, effectively bypasses this challenge by focusing solely on the SDOT plan between p_1 (the known, finite empirical dataset itself) and p_0. Since p_1 is fully defined by the dataset and p_0 is easily sampled, the SDOT plan can, theoretically, be solved with zero estimation error. We emphasize that this approach makes no assumption regarding the quality of the SDOT plan (between p_1 and p_0) as an approximation of the classical OT plan (between $\tilde{p_1}$ and p_0).



(a) CIFAR 10 on Unet, generated by adaptive integrator DOPRI5 (Sec. 5.1)

(b) DiT-B/2 on ImageNet256 with shortcut model, generated by 4-step forward Euler.

(c) SiT on ImageNet256 with Meanflow, generated by 1-step forward Euler.

Figure 2: Training curves for AlignFlow on different tasks. Each figure illustrates the FID-50k score against the number of training steps. The results demonstrate that AlignFlow provides a consistent and simultaneous improvement over all baseline algorithms shown, enhancing both final performance and training convergence speed.

4.2 Deterministic alignment

The Stochastic Denoising Optimal Transport (SDOT) map is theoretically fully deterministic: a given sample from the noise distribution is consistently mapped to a fixed data point.

Intuitively, this determinism confers a significant advantage in terms of convergence speed: To determine the target vector field v_t at some t and x_t, the standard approach using the random coupling (as in Algorithm 1) requires iterating across the entire dataset:

$$u(x_t, t; \theta) = \mathbb{E}_{x_1 \sim p_1} \, \text{TargetVectorField}(x_0, x_1) p_0(x_0), \quad x_0 = x_1 - (x_1 - x_t)/t \tag{8}$$

However, the fixed coupling in AlignFlow avoids this estimation process, and the target vector field for the neural network to learn is provided by

$$u(x_t, t; \theta) = \text{TargetVectorField}(x_0, x_1) p_0(x_0), \quad x_0 = x_1 - (x_1 - x_t)/t, x_1 = \varphi(x_t) \tag{9}$$

This crucial difference demonstrates that fixed coupling significantly simplifies the estimation of the target vector field, thereby leading to the accelerated convergence observed empirically with AlignFlow.

4.3 Low computational cost

AlignFlow directly computes the SDOT map in stage 1. Compared to other pioneers using indirect approaches that estimate the OT plan by samples (e.g., Reflow operation in Liu et al. (2022), ICNN in Kornilov et al. (2024), and Sinkhorn iteration in Tong et al. (2023)), the computation of the SDOT map in Stage 1 is more accurate and efficient. Empirically, stage 1 takes negligible cost ($< 1\%$ extra time). More details are in Sec. A.

Upon completion of Stage 1, SDOT map is fully computed. Consequently, the only additional overhead in Stage 2 stems from the generation of the training noise-data pairs (Lines 7 and 9 in Algorithm 3). This process is highly efficient and executes rapidly on modern GPUs, incurring an almost negligible cost (typically $<0.1\%$ of the total training time).

5 Experiments

5.1 CIFAR10 unconditional generation on Unet

Following the methodology of (Tong et al., 2023, Section 5.3), we trained a U-Net model unconditionally on the CIFAR10 dataset. In this setup, the Flow Generative Model (FGM) was trained directly in the pixel space. The comparative training curve is displayed in Figure 2(a), and the FID-50k scores for various ODE integrators are detailed in Table 2. Compared to the coupling estimated via the standard minibatch Optimal Transport (OT) algorithm, our AlignFlow approach

Algorithm	AlignFlow?	NFE=4	Difference	NFE=1	Difference
Flow Matching	✓	93.16	↓ 32.46	276.18	↓ 28.86
	✗	125.62		305.04	
Consistency Training	✓	103.14	↓ 8.70	64.33	↓ 12.04
	✗	111.84		76.37	
Live Reflow (Frans et al., 2024)	✓	60.23	↓ 34.52	47.06	↓ 12.81
	✗	94.75		59.87	
Shortcut Models (Frans et al., 2024)	✓	**30.31**	↓ 2.80	**43.92**	↓ 2.73
	✗	33.11		46.65	

Table 3: Evaluation of DiT-B/2 on ImageNet 256 using FID-50k demonstrates that AlignFlow significantly enhances performance across all tested NFE configurations.

Backbone	# params	w/ AlignFlow	w/o AlignFlow	Difference
SiT-B/4	131M	**13.75**	15.53	↓ 1.78
SiT-B/2	131M	**5.60**	6.17	↓ 0.57
SiT-L/2	459M	**3.51**	3.84	↓ 0.33
SiT-XL/2	676M	**3.23**	3.43	↓ 0.20

Table 4: FID-50k on ImageNet256 by Meanflow (NFE=1). AlignFlow improves MeanFlow in all model sizes.

demonstrates faster convergence and achieves better FID scores across all tested ODE integrators. All experiments utilized the official code provided by Tong et al. (2023).

	Euler (100 steps)	Euler (1000 steps)	DOPRI5
Minibatch OT (Tong et al., 2023)	4.80	3.92	3.82
AlignFlow (ours)	4.72	3.79	**3.71**

Table 2: Comparing FID-50k score for Unet trained on CIFAR10 between minibatch OT and AlignFlow with different ODE integrators. The reported results are the average of 5 independent runs. AlignFlow outperforms minibatch OT under different ODE integrators.

5.2 ImageNet256 on DiT with Shortcut Model

AlignFlow can be easily combined with modern SOTA models and scales to large datasets. We train it with DiT as NN on the class-conditioned ImageNet with 256×256 resolution (ImageNet256). The FGM operates in the latent space with shape $28 \times 28 \times 4$ generated by a pretrained VAE. All model hyperparameters were adopted directly from (Frans et al., 2024, Table 1 and 3) without any modification or tuning. The comparison of the training curve for the shortcut model with and without AlignFlow is presented in Fig. 2(b). The improvement of more models by AlignFlow is shown in Tab. 3.

5.3 ImageNet256 on SiT with MeanFlow

AlignFlow further improves the one-step generation model MeanFlow (Geng et al., 2025). MeanFlow uses SiT as NN and is trained on class-conditioned ImageNet256. The FGM is trained in the latent space with shape $28 \times 28 \times 4$ generated by a pretrained VAE. The code is a non-official PyTorch implementation (Zhu, 2025), since it has proven to be able to reproduce the reported results on GPU. All the hyperparameters are identical to the official setting in Sec. A in Geng et al. (2025) without further tuning. The training curve in Fig. 2(c) and the FID score is in Tab. 4. AlignFlow improves both performance and convergence speed in all cases, showing that AlignFlow scales to large models. Image samples are shown in Fig. 4 in the appendix.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Jason M Altschuler, Jonathan Niles-Weed, and Austin J Stromme. Asymptotics for semidiscrete entropic optimal transport. *SIAM Journal on Mathematical Analysis*, 54(2):1718–1741, 2022.

Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International conference on machine learning*, pp. 146–155. PMLR, 2017.

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*, 2025.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Quan Dao, Hao Phung, Trung Tuan Dao, Dimitris N Metaxas, and Anh Tran. Self-corrected flow distillation for consistent one-step and few-step image generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 2654–2662, 2025.

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.

Nicolas Fournier and Arnaud Guillin. On the rate of convergence in wasserstein distance of the empirical measure. *Probability theory and related fields*, 162(3):707–738, 2015.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Johannes Hertrich, Antonin Chambolle, and Julie Delon. On the relation between rectified flows and optimal transport. *arXiv preprint arXiv:2505.19712*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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*, 37:104180–104204, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Huidong Liu, Ke Ma, Lei Zhou, and Dimitris Samaras. Efficient semi-discrete optimal transport using the maximum relative error between distributions.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

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. *arXiv preprint arXiv:2304.14772*, 2023.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

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*, pp. 10684–10695, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

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. *arXiv preprint arXiv:2302.00482*, 2023.

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. *arXiv preprint arXiv:2302.13971*, 2023.

Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

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

A EFFICIENT SDOT ALGORITHM FOR LARGE DATASETS

A.1 INDICATOR FOR PERFORMANCE OF ALGO. 2

Before diving into details, let's define Maximum Relative Error (MRE) and L1 (Liu et al.), which help to judge the quality of the dual weight:

$$\mathrm{MRE}(\mathbf{g}) = \max_{i\in I} \frac{|p_i - b_i|}{b_i}, \quad \mathrm{L}_1(\mathbf{g}) = \sum_{i\in I} |p_i - b_i|, \quad p_i := \int \mathbb{L}_i(\mathbf{g})\, \mathrm{d}p_0 \tag{10}$$

In Algo. 2, they can be estimated efficiently by $\widetilde{\mathrm{MRE}} = \|\nabla \mathcal{E}_{\mathtt{ema}}\|_\infty$ and $\widetilde{\mathrm{L1}} = \|\nabla \mathcal{E}_{\mathtt{ema}}\|_1$.

Mathematically, MRE and L_1 are simply estimating $\|\nabla \mathcal{E}\|_\infty$ and $\|\nabla \mathcal{E}\|_1$. However, the reason why MRE is important is because it measures the unbalance between each target. $\mathrm{MRE} = 0$ means the SDOT is computed perfectly, both because it perfectly minimizes the objective function Eq. (4) and also because each target is mapped to with equal probability. See also Sec. F.

A.2 ENTROPIC REGULARIZATION FOR SDOT

Discrete OT problems are known to be non-smooth and people add regularization terms to smoothen the landscape. For SDOT problems, similar techniques can also be applied. Instead of solving the problem Eq. (4), SDOT with entropic regularization is solving (Altschuler et al., 2022)

$$\min_{\gamma\in\Gamma(p_0,p_1)} \int_{\mathcal{X}\times\mathcal{X}} c(x_0,x_1)\mathrm{d}\gamma(x_0,x_1) + \epsilon\, \mathrm{KL}(\gamma||p_0\otimes p_1) \tag{11}$$

Although introducing the extra term leads to bias, it significantly improves smoothness.

A.3 HYPERPARAMETERS TUNING FOR COMPUTATION OF SDOT

Algo. 2 has mainly three hyperparameters to be tuned: entropic regularization strength ϵ, EMA parameter β, and learning rate lr in Adam. During the iteration in Algo. 2, $\mathrm{L1}$ will be continuously decaying if the hyperparameters are correctly tuned.

- Entropic regularization strength ϵ balances the bias and difficulties of the optimization problem. Large ϵ will introduce bias, while 0 or a small ϵ leads to harder optimization.
- ϵ should be fixed during the optimization procedure. When optimizing the problem, consider increasing the batch size and/or decreasing the learning rate when $\mathrm{L1}$ plateaus.
- The optimization stops when MRE meets your requirement. We recommend ensuring it is below 0.2 for good performance in the downstream task of FGM.
- Usually, the learning rate for Adam should be relatively large. Unlike in modern machine learning problems that 0.001 learning rate is recommended, in the computation of SDOT map Algo. 2, 10 is a good starting point for tuning the learning rate.
- For large datasets, please use a larger batch size and/or increase the EMA parameter β (e.g., change 0.99 to 0.999)

A.4 HYPERPARAMETERS AND COMPUTATIONAL COST

For CIFAR 10, we use the training set for training (50000 images with shape $32 \times 32 \times 3$).

- Normalize the whole dataset with mean = $(0.5, 0.5, 0.5)$ and std = $(0.5, 0.5, 0.5)$.
- Concatenate the dataset with the augmented (horizontally flipped) dataset.
- Compute the SDOT map with Algo. 2 and hyperparameters in Table 5

For ImageNet, we use the training set with 1281167 images separated into 1000 classes. In both shortcut model in Sec. 5.2 and Meanflow in Sec. 5.3, the SDOT map was done in the latent space with each latent representation $28 \times 28 \times 4$. Each image was augmented by horizontal flipping, making each class have around 2600 images in total after augmentation.

# step	learning rate	batch size	EMA parameter β	entropic reg ϵ	MRE	L1
0-1000	10	1024	0.99	1	3.4	0.29
1000-6000	0.1	4096	0.999	1	0.27	0.045
6000-11000	0.1	16384	0.999	0.01	0.11	0.019

Table 5: CIFAR 10 (unconditional) SDOT hyperparameters. It costs 8 min 30 s on L40S.

# step	learning rate	batch size	EMA parameter β	entropic reg ϵ	MRE	L1
3000	10	4096	0.99	0.01	~ 0.08	~ 0.016

Table 6: ImageNet256 (class conditioned, latent space) SDOT hyperparameters. It costs $<$10 s on L40S for each class.

Remark 4. *Although ImageNet is larger than CIFAR, the SDOT map computation is cheaper for ImageNet then CIFAR due to the following reasons:*

- *ImageNet experiments perform SDOT in the latent space, which has similar dimension to CIFAR in pixel space.*
- *ImageNet is class-conditioned, makes each class has only $\sim$ 2600 images. Since OT problem scales quadratically w.r.t. number of targets , although the ImageNet dataset is larger, it becomes 1000 easier SDOT problems.*

B More details for Flow-Based Generative Models

B.1 More examples for FGM framework in Algo. 1

Shortcut model In this model, an auxiliary input d for the neural network, i.e., $u = u(x, t, d; \theta)$ and d^j is i.i.d. sampled from $\mathcal{D}(\cdot|t^j)$. Given hyperparameter κ, choose $\mathrm{TargetVectorField}(x_0^j, x_1^j) = x_1^j - x_0^j$ for $j = 1, ..., \kappa$, and $\mathrm{TargetVectorField}(x_0^j, x_1^j) = \mathrm{StopGrad}(s_t^j + s_{t+d}^j)$ for $j = \kappa + 1, ..., B$, where $s_t^j := u(x_t^j, t^j, d^j)$, $x_{t+d}^j := x_t^j + s_t d^j$, $s_{t+d}^j := u(x_{t^j+d^j}^j, t^j + d^j, d^j)$. Together with $\mathrm{Loss}\left(\left\{\hat{v}^j, v^j\right\}_{j=1}^B\right) := \frac{1}{B}\sum_j \left\|\hat{v}^j - v^j\right\|_2^2$, Algo. 1 recovers the shortcut model in Frans et al. (2024).

Meanflow In this model, an extra r input for the neural network, i.e., $u = u(x, t, r; \theta)$. By choosing $\mathrm{TargetVectorField}(x_0, x_1) = \mathrm{StopGrad}(v_t - (t-r)v_t\partial_x u + \partial_t u)$ and $\mathrm{Loss}\left(\left\{\hat{v}^i, v^i\right\}_{i=1}^B\right) := \frac{1}{B}\sum_i \left\|\hat{v}^i - v^i\right\|_p^p$, Algo. 1 recovers Meanflow in Geng et al. (2025).

B.2 Sampling/inference process for FGM

To better see the benefits of the specially-designed target vector field, let's see the inference process for FGM in Algo. 4. The inference process is a process integrating the ODE $\partial_t x_t = u(t, x_t; \theta)$ with initial condition x_0 sampled from p_0, and x_1 is a new sample. Most inference processes for FGM are the same, except the ODE integrator may be different. However, the difficulty for such integration varies by the learned $u(t, x; \theta)$: intuitively, let's imagine two trajectories of x_t, one is complicated, while the other is a straight line. Then the straight line can be easily computed by one-step forward Euler $x_1 = x_0 + u(0, x_0; \theta)$, while the complicated one requires a complicated ODE integrator. This means the straightness of the integrated trajectory is the key: the straighter the path, the easier integration and thus, less NFE of NN (Liu et al., 2022). Algorithm trained by fancy target vector field will benefit from the extra straightness, e.g., Meanflow is able to generate high-quality samples with NFE=1, while vanilla flow matching requires > 100 NFE.

Algorithm 4: FGM / AlignFlow (Sampling)

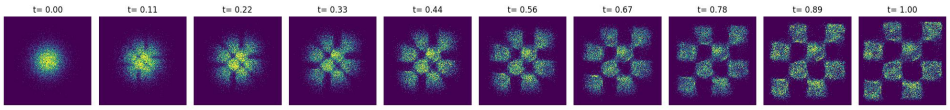
Input: Noise distribution p_0, neural network $u = u(x, t; \theta)$, ODEIntegrator
Output: A new sample from the data distribution
Sample $\{x(0)\} \sim p_0$ ▷ sample noise
$v(t) := u(x(t), t; \theta)$
$x(1) = \text{ODEIntegrator}(v(t))$
Return: a new sample $x(1)$

C SYNTHETIC EXPERIMENT: CHECKERBOARD

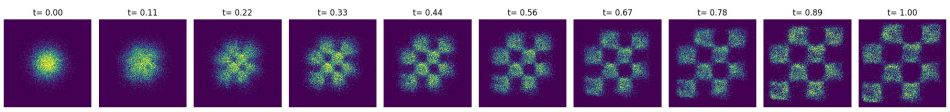
In this section, we visualize different learned trajectories by training the FGM on a synthetic 2-dimensional data distribution. Following (Lipman et al., 2022, Fig. 4), we set the data distribution $\tilde{p_1}$ as the checkerboard in $[-2, 2] \times [-2, 2]$. p_0 is chosen as the widely used normal distribution. However, different from their setting, which assumes accessibility to an infinite amount of training data [7], we fix the training set at the beginning instead. Although our setting leads to less smoothness of the learned distribution than the infinite data setting, it simulates real ML tasks where data is limited.

Fig. 3 plots the density changes from normal distribution to checkerboard when time evolves from 0 to 1, showing that AlignFlow gives a straighter path compared to minibatch OT and vanilla flow matching in Fig. 3. Identical hyperparameters are used in all cases.

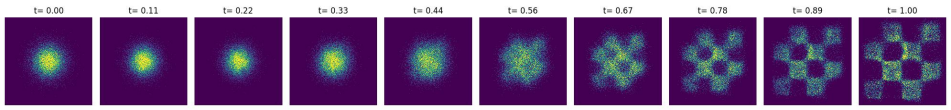
For AlignFlow, we compare the density evolution for FGM trained with different couplings between noise and data.



(a) AlignFlow (Alog. 3)



(b) Minibatch OT (Tong et al., 2023)



(c) Vanilla flow matching (Lipman et al., 2022)

Figure 3: Comparison of the trajectory of FGM between different methods. AlignFlow has a straighter trajectory compared to vanilla flow matching and has a clearer boundary compared to Minibatch OT (e.g., at $t = 0.22$).

D CAPABILITY OF GENERALIZATION

Experts may notice that Eq. (5) is a map from noise to data, which is already $\mathcal{X} \to \mathcal{X}$. Since FGM also gives a vector field whose integration is $\mathcal{X} \to \mathcal{X}$, why do we still need to train the FGM model in stage 2? The answer is, the SDOT map φ in Eq. (5) only remembers the dataset and cannot generalize, i.e., every noise is mapped to a data point in the dataset, and no new data is generated. As a result, a powerful NN is required to be trained in Stage 2 in Algo. 3 on top of the SDOT map.

[7] In the checkerboard experiment in Lipman et al. (2022); Tong et al. (2023), new training data is drawn in each minibatch.

Another interesting thing is: The SDOT map is a fixed map from noise to the dataset. Will this noise-data alignment hurt the capability of generalization compared to random noise-data alignment? The answer is no. Here are some explanations from different aspects:

- In traditional FGM (1), the (noise, data) pair is also generated from the $p_0 \times p_1$, rather than $p_0 \times \tilde{p_1}$, and the capability of generalization is widely proven. Algo. 3 only changes the coupling, but not the marginal distribution.
- Experimentally, we are generating the images that do not exist in the original dataset (Fig. 4)
- Theoretical guarantee that any coupling will lead to the correct push forward vector field in Eq. (2) in by Sec. 3.2.
- We also provide an intuition: All FGMs try to learn the map from noise to a discrete dataset, since there is no access to the real data distribution $\tilde{p_1}$. However, the model still generalizes. This is because the generalization ability comes from the regularity of the neural network that approximates the vector field, but does not come from the randomness of the matching between noise and data.

E More about Curse of Dimensionality

Modern generative model tasks are usually high-dimensional with limited data. For example,

- On unconditional CIFAR dataset in pixel space (Sec. 5.1), the dimension is $32 \times 32 \times 3 = 3072$, but we only have 100k training data in total (50k images, 100k after horizontal flip augmentation)
- In ImageNet256 task (Sec. 5.2 and 5.3), we have only approximately 2600 images (1300 images in each class, 2600 after horizontal flip) for each class, and the latent space we perform FGM algorithm has dimension $28 \times 28 \times 4 = 3136$.

Don't forget that we require the number of samples to be of order exponential dimension in Thm. 1, which is definitely not enough based in the cases above. And this phenomenon is more severe in the class-conditional generation setting, since we require the number of samples in each class to be of order exponential dimension.

To judge if an algorithm will suffer from the curse of dimensionality, we can use the following criterion: if a method uses samples to estimate an OT plan between $\tilde{p_1}$ and p_0, then it must suffer from the curse of dimensionality. The reason is that estimating the OT plan by samples needs an important assumption: the OT plan between p_1 and p_0 approximates the OT plan between $\tilde{p_0}$ and p_0, which is generally not true as discussed above. According to this criterion, Tong et al. (2023); Kornilov et al. (2024) will suffer from the curse of dimensionality. Although rectified flow (Liu et al., 2022) does not estimate the OT plan directly, it claims the algorithm is also trying to approximate the OT plan between $\tilde{p_1}$ and p_0.

F Rebalance: handling non-perfect SDOT map

Instead of sampling i.i.d. data, Algo. 3 only samples noise and feeds the NN with data generated by the SDOT map (line 7). This may lead to the problem of the SDOT map not being computed perfectly, and the data feeding the NN is biased. Roughly speaking, when $\nabla \mathcal{E}$ is not 0, then $\varphi(x_0),\ x_0 \overset{\text{i.i.d.}}{\sim} p_0$ is biased from p_1 (See more discussions in Sec. A, especially MRE defined in Eq. (10)).

If you find running Algo. 2 until MRE converges to 0 is too hard and expensive, especially for large datasets with a huge number of data without class condition, rebalance is here to help. A non-zero MRE leads to biased sampled data, meaning that the model is training with some of the data seen more often, while others are seen less often. To address this difficulty, we introduce the rebalance

operation in Line 9, defined by

$$\text{rebalance}\left(\{m_j\}_{j=1}^M\right) := \arg\max_{\{\tilde{m}_j\}} \left\{ \sum_j \mathbb{1}(\tilde{m}_j = m_j) : \left| \max_{i\in I} \sum_{j=1}^M \mathbb{1}_i(\tilde{m}_j) - \min_{i\in I} \sum_{j=1}^M \mathbb{1}_i(\tilde{m}_j) \right| \leq 1 \right\} \tag{12}$$

Intuitively, rebalance calculates the minimum modification of the targets s.t. the frequency of each data point is the same. Such an operation forces the model to see the correct unbiased dataset even SDOT map is not fully converged, at the cost of introducing some randomness in the coupling.

In our research, we didn't find a difference in the CIFAR10 experiment in Sec. 5.1 between with and without the rebalance operation, since Algo. 2 learns SDOT map pretty well (more than 85% of the data are unchanged in the rebalance operation). However, we use it throughout all our experiments to ensure that the data fed into the FGM model is the same as random coupling for a fair comparison.

G IMAGE SAMPLES GENERATED BY ALIGNFLOW

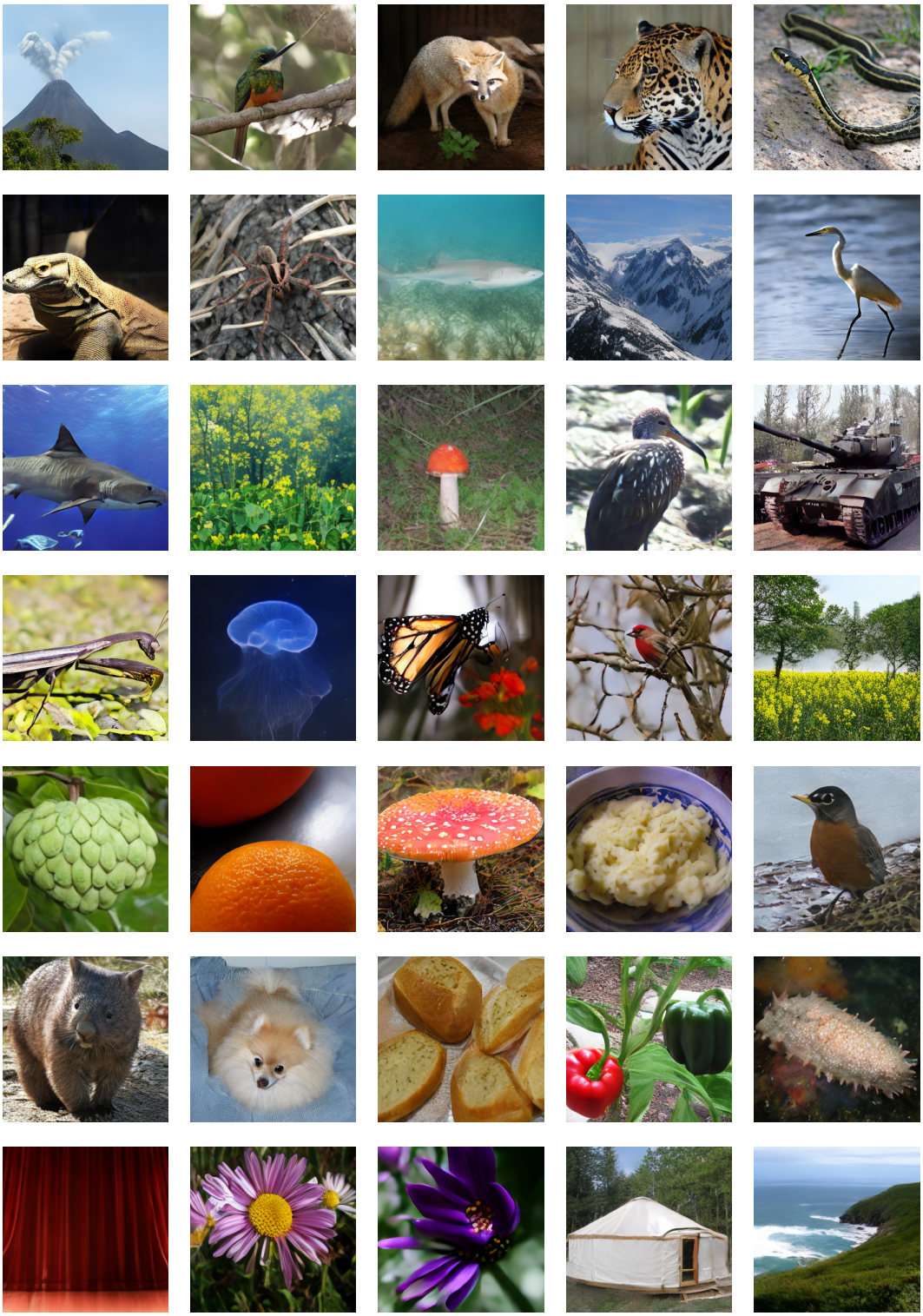


Figure 4: Images generated by MeanFlow+AlignFlow trained on ImageNet256 (FID=3.23, NFE=1).