MIXFLOW: MIXED SOURCE DISTRIBUTIONS IMPROVE RECTIFIED FLOWS

Nazir Nayal **Christopher Wewer** **Jan Eric Lenssen**
Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
{nnayal,cwewer,jlenssen}@mpi-inf.mpg.de

ABSTRACT

Diffusion models and their variations, such as rectified flows, generate diverse and high-quality images, but they are still hindered by slow iterative sampling caused by the highly curved generative paths they learn. An important cause of high curvature, as shown by previous work, is independence between the source distribution (standard Gaussian) and the data distribution. In this work, we tackle this limitation by two complementary contributions. First, we attempt to break away from the standard Gaussian assumption by introducing κ−FC, a general formulation that conditions the source distribution on an arbitrary signal κ that aligns it better with the data distribution. Then, we present MixFlow, a simple but effective training strategy that reduces the generative path curvatures and considerably improves sampling efficiency. MixFlow trains a flow model on linear mixtures of a fixed unconditional distribution and a κ−FC-based distribution. This simple mixture improves the alignment between the source and data, provides better generation quality with less required sampling steps, and accelerates the training convergence considerably. On average, our training procedure improves the generation quality by 12% in FID compared to standard rectified flow and 7% compared to previous baselines under a fixed sampling budget. Code available at: https://github.com/NazirNayal8/MixFlow

1 INTRODUCTION

Generative modeling, the problem of fitting and sampling from data distributions, is a heavily explored topic with remarkable success in recent years, mostly driven by progress in image generation Ho et al. (2020); Song et al. (2021a); Lipman et al. (2023). Existing generative models offer trade-offs between sampling speed, diversity, and the quality of the generated samples, referred to as the generative learning trilemma Xiao et al. (2022). Diffusion models Song et al. (2021b); Ho et al. (2020); Song et al. (2021a) and their variations have pushed the performance considerably in terms of diversity and quality. However, a single inference requires several forward passes to obtain high quality samples. Therefore, several works have explored ways to reduce the number of function evaluations required for sampling. Rectified Flow Liu et al. (2023) and Flow Matching Lipman et al. (2023) tackled the problem from the perspective of straightening the generative paths by replacing the diffusion schedulers with optimal transport displacement interpolations McCann (1997) between the source and target distributions. Even though their formulations provide theoretical guarantees for requiring a fewer number of sampling steps, the amount of required steps still remains high in practice. In this work, we tackle this problem by introducing an effective training strategy for flow models to reduce the amount of steps required to generate high quality samples.

Flow models learn to iteratively transform a simple source distribution, usually a standard Gaussian, to a complex data distribution. For Flow Matching, a recent line of work shows that the sampling speed is strongly influenced by the assumptions on the forward coupling Tong et al. (2024); Pooladian et al. (2023); Lee et al. (2023). The forward coupling is the joint distribution of the source and the target, which encodes their dependence relation. Optimizing the forward coupling leads to straighter generative paths by exposing the model to source-target pairs that are more aligned Lee et al. (2023).

Inspired by this, we propose κ-Forward Coupling (κ−FC), a general formulation for learnable forward couplings that can utilize an arbitrary guiding signal κ to align the source distribution with the target.

The more informative κ is of the data distribution, better alignment is achieved. Nevertheless, we show that naively optimizing the forward coupling with κ introduces a difficult trade-off with a regularization hyperparameter that can lead to issues like the prior hole problem Hao & Shafto (2023). To counter this, we introduce MixFlow, a technique that uses a linear mixture of two distributions as the source distribution, one of which is fixed and the other a learnable distribution that is learned using κ-FC. The mixing encourages that samples on the interpolation path map to similar regions in the target distribution, transporting structure from the conditional to the unconditional (Gaussian) source distribution. MixFlow demonstrates overall improvement in sampling quality and requires fewer sampling steps. Furthermore, we show that, given a conditioning signal that is sufficiently informative, our formulation allows for controlling the speed-quality trade-off at test-time.

To verify our findings, we present exhaustive results on common image generation benchmarks, showing that our approach improves FID by 12% compared to standard Rectified Flow and by 7% compared to the best previous method for trajectory straightening, with a comparable number of sampling steps. In contrast to previous works, our trade-off does not depend on parameters that need to be set during training. We provide an analysis of different design choices and their effect on the source distribution, generation quality, and sampling speed.

In summary, our contributions are:

- We propose κ-FC, a general formulation for learnable forward couplings that can be conditioned on arbitrary variables for obtaining better source distributions.
- We introduce MixFlow, a method for training Rectified Flows with a linear mixture of two distributions as a source distribution, which leads to less sampling steps required to generate high quality samples.

2 Related work

In general, the lines of work that study the sampling speed problem in diffusion models can be categorized into the following groups, depending on which part of the design space is examined.

Distillation. One direction explores linearizing the mapping between the source and the target distribution through distillation Salimans & Ho (2022); Liu et al. (2023); Berthelot et al. (2023); Zhou et al. (2024); Luhman & Luhman (2021); Xie et al. (2024), consistency constraints Song et al. (2023); Song & Dhariwal (2024); Silvestri et al. (2025); Geng et al. (2025); Yang et al. (2024), or Reflow Liu et al. (2023). While these methods are able to achieve reasonable generation quality with a single sampling step, they require retraining a model multiple times, and often degrade the model's performance for higher number of sampling steps Guo & Schwing (2025). On the other hand, we show that training MixFlow once can improve the performance for all choices of sampling steps, and can also considerably reduce the required training budget. A branch in this direction attempts to improve the Reflow operation by generalizing it to arbitrary schedules Wang et al. (2025a), or enhancing its design components Kim et al. (2025). Furthermore, we highlight that this direction is orthogonal to our approach and these methods can be applied to any model trained with MixFlow.

Faster Solvers. Another line of work focuses on developing faster samplers by utilizing better numerical ODE solvers Dockhorn et al. (2022); Karras et al. (2022b); Lu et al. (2022); Song et al. (2021a). Despite these improvements, the sampling speed remains bounded by the curvature of the generative trajectories induced by the flow models. In this work, we tackle the same problem but from the orthogonal perspective of source distribution optimization. Hence, our method can be combined with any ODE solver to achieve faster sampling.

Path Straightness. Rectified Flow Lee et al. (2023) shows that the intersections of the paths constructed by the source and target distribution samples affect the straightness of the generative paths. Flow-Matching Lipman et al. (2023) defines the probability paths as an optimal transport interpolation to straighten the trajectories, which leads to a similar formulation to the one used by Rectified Flow. Variational Rectified Flow Matching Guo & Schwing (2025) was also proposed to improve trajectory straightness by explicitly modeling the multiple possible paths that cross a certain point using a VAE. A recent method QAC Liang et al. (2024) conditions the flow model on learnable

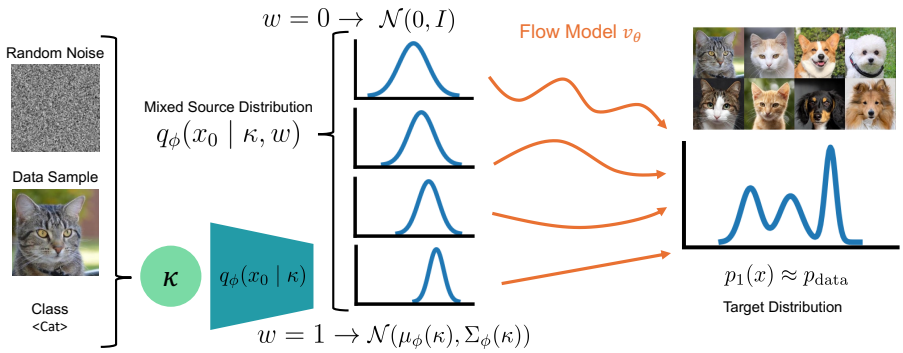


Figure 1: **Method overview.** We propose training rectified flows with mixed source distributions, obtained by interpolating a conditional and a simple unconditional distribution. The conditional distribution is predicted from a signal κ, which is possibly informative, e.g. a specific data example, a class label, or entirely independent, e.g. random noise. The learned conditional distribution provides a trajectory structure that minimizes the degree of intersections, which is inherited by the mapping from the unconditional source distribution.

representation in order to reduce the trajectory curvatures. Despite the improvements these methods achieve, they still assume an independent coupling between the source and target distribution. In this work, we aim to reduce the curvatures by optimizing this coupling.

Optimized Forward Coupling. Most related to our method are the works that explore the impact of the forward coupling on trajectory curvature. Some methods improve the coupling through approximating the optimal transport plan between the source and target distributions Tong et al. (2024); Pooladian et al. (2023). However, it is computationally infeasible to solve an optimal transport problem on an entire dataset, so they approximate on the mini-batch level. Fast-ODE Lee et al. (2023) explores parameterizing the coupling as a neural network as a function of the data sample and optimizing it jointly with the flow model, which is shown to minimize the forward intersections and hence leads to faster sampling. However, the unavailability of the data samples at in inference time restricts the representation of the learned coupling, since it cannot deviate significantly from the independent coupling. We propose a general formulation that subsumes Fast-ODE and allows for larger deviation from the independent coupling assumption.

3 Background

We first introduce necessary background and notations of Rectified Flows in Sec. 3.1 and the degree of trajectory intersection with its impact on sampling speed in Sec. 3.2.

3.1 Rectified Flow

We assume a d-dimensional space $\mathbb{R}^d$ where the data points lie. The aim of Rectified Flow Liu et al. (2023) is to learn a mapping between samples of a tractable source distribution $p_0(x)$ and a complex target distribution $p_1(x)$. We define $q(x_0, x_1)$ as their joint coupling whose marginal preserves their respective densities, and is by default assumed to be an independent coupling $q(x_0, x_1) = p_0(x)p_1(x)$. Given samples $x_0 \sim p_0(x), x_1 \sim p_1(x)$, an intermediate representation $x_t \sim p_t$ on the straight path between x_0, x_1 is defined as $x_t = tx_1 + (1-t)x_0$ for $t \in [0, 1]$, which represents a time-differentiable forward coupling between $p_0(x), p_1(x)$. Rectified Flow proposes to learn a vector field $v_\theta(x_t, t)$ parametrized by θ, which approximates the velocity required to flow in straight paths from x_0 to x_1, passing through x_t, defined as the time derivative $dx_t = v_t(x)dt = (x_1 - x_0)dt$ of the intermediate representation. The parameters θ of the learned vector field are found by minimizing

$$\mathcal{L}_{\text{RF}}(\theta) := \mathbb{E}_{x_0, x_1 \sim q(x_0, x_1)} \left[l(x_0, x_1) \right], \quad l(x_0, x_1) := \int_0^1 \|x_1 - x_0 - v_\theta(x_t, t)\|^2 \, dt. \tag{1}$$

3.2 DEGREE OF INTERSECTION

Previous works Lee et al. (2023); Wang et al. (2025b) have shown the effect of choosing the forward coupling $q(x_0, x_1)$ on the curvature of the generative trajectories. When the paths constructed between pairs (x_0, x_1) in the forward process are highly intersecting, the vector-field model learns to estimate the mean direction, which causes the generative paths to be highly curved.

The optimal θ^* in Eq. 1 is achieved when $v_{\theta^*}(x_t, t) = \mathbb{E}_{x_t}[x_1 - x_0 \mid x_t]$ as an estimator for the mean-squared error. Assuming we obtain an optimal model, and given a forward coupling $q(x_0, x_1)$, the degree of intersection of the forward trajectories can be estimated as:

$$I(q) = \mathop{\mathbb{E}}_{x_0, x_1 \sim q(x_0, x_1)} \left[\int_0^1 \left\| x_1 - x_0 - v_{\theta^*}(x_t, t) \right\|^2 dt \right], \tag{2}$$

which is minimized for the same values as Eq. 1 Lee et al. (2023). With a fixed independent coupling $q(x_0, x_1) = p_0(x)p_1(x)$, $I(q)$ remains fixed. Therefore, in order to straighten the generated trajectories and improve sampling speed, previous methods Tong et al. (2024); Lee et al. (2023); Pooladian et al. (2023) attempt to optimize $q(x_0, x_1)$ in order to minimize $I(q)$.

4 STRAIGHTENED TRAJECTORIES VIA DISTRIBUTION MIXING

This section introduces MixFlow, a method for training rectified flow from mixtures of (un)conditional source distributions. Fig. 1 provides an overview of our approach. In Sec. 4.1, we propose κ−FC, a general formulation of learnable forward couplings that can depend on arbitrary variables to optimize source distributions for a lower degree of intersection. Moreover, we discuss the limitations of a naively constructed conditional source distribution relying on a simple Gaussian assumption. To this end, Sec.4.2 formally introduces MixFlow and highlights its effect in overcoming these limitations.

4.1 LEARNABLE FORWARD COUPLING (κ−FC)

Let κ be a generic random variable that lies in $\mathbb{R}^n$. It can be an informative signal related to the data distribution $p_1(x)$, such as a class label, or entirely independent. In practice, κ can represent class labels, captions of an image, or any correlated or uncorrelated signal. Our general formulation subsumes the parametrization in Fast-ODE Lee et al. (2023) as a special case, where the conditioning is the data sample itself $\kappa = x_1$.

Abstracting from the choice of κ, we assume it to be a common cause for x_0 and x_1, i.e., x_0, x_1 are conditionally independent given κ. With this assumption, the forward coupling can be written as:

$$q(x_0, x_1) = \int q(x_0, x_1 \mid \kappa)\, p(\kappa) d\kappa = \int q(x_0 \mid \kappa) q(x_1 \mid \kappa) q(\kappa) d\kappa = \int q(x_0|\kappa) q(x_1, \kappa) d\kappa \tag{3}$$

Given this factorization, we propose a learnable coupling with additional parameters ϕ: $q_\phi(x_0, x_1) = \int q_\phi(x_0 \mid \kappa) q(x_1, \kappa) d\kappa$. that can be jointly optimized with the vector field $v_\theta(x_t, t)$ to minimize the following loss

$$\mathcal{L}_{\kappa\text{-FC}}(\theta, \phi) = \mathbb{E}_{x_1, \kappa \sim q(x_1, \kappa), x_0 \sim q_\phi(x_0|\kappa)} l(x_0, x_1), \tag{4}$$

which we obtain by sampling $x_0 \sim q_\phi(x_0 \mid \kappa)$ in Eq. equation 1. One open design choice is the construction of $q_\phi(x_0 \mid \kappa)$. A straightforward option is a Gaussian $q_\phi(x_0 \mid \kappa) = \mathcal{N}(\mu_\phi(\kappa), \Sigma_\phi(\kappa))$ with learnable mean and covariance, enforced by adding a regularization term $\beta D_{KL}\left(q_\phi(x_0|\kappa) \| \mathcal{N}(0, I)\right)$ to Eq. equation 4. However, we argue and empirically show (see Tab. 4) that this design strongly depends on the choice of the regularization weight β that controls how close the conditional distribution is to the standard Gaussian. The case of $\beta \to 0$ results in the prior hole problem Hao & Shafto (2023), well-known in the context of VAEs, that prevents sampling at inference time without a given κ. For $\beta \to \infty$, $q_\phi(x_0 \mid \kappa)$ becomes almost a standard Gaussian and therefore independent of κ which removes any advantage of the learnable forward coupling. Even worse, β is a hyperparameter determined prior training with strong influence on the performance during inference.

Therefore, we propose an alternative design of $q_\phi(x_0 \mid \kappa)$ via distribution mixing that is robust to train and achieves lower curvature (cf. Sec. 3.2), resulting in higher quality samples with small numbers of network evaluations.

4.2 Flowing from a Mixture of Two Distributions

We propose to train the vector-field model to flow from linear interpolations of two distributions: (1) the parameterized Gaussian $\mathcal{N}(\mu_\phi(\kappa), \Sigma_\phi(\kappa))$, and (2) a standard Gaussian $\mathcal{N}(0, I)$. As linear interpolations of two Gaussians with scalar weight w, the resulting source distributions $q_\phi(x \mid \kappa, w) = \mathcal{N}(w\mu_\phi(\kappa), w\Sigma_\phi(\kappa) + (1-w)I)$ are themselves Gaussian. Training the vector field with a conditional source distribution, the unconditional standard normal, and everything in between enables the network to learn to effectively utilize κ while enforcing full coverage of the Gaussian space. Thus, during training, the efficient coupling between conditional source distribution and target distribution is transferred to the unconditional source distribution as well, allowing inference without conditioning. We now introduce training and sampling algorithms.

Algorithm 1: Training

Input : $q(x_1, \kappa), \mu_\phi, \Sigma_\phi, v_\theta, \beta, N$

for $i \leftarrow 1$ **to** N **do**
- $x_1, \kappa \sim q(x_1, \kappa)\ \ t, w \sim \mathcal{U}(0, 1)$
- $\mu_\kappa \leftarrow \mu_\phi(\kappa), \Sigma_\kappa \leftarrow \Sigma_\phi(\kappa)$
- $\mu_w \leftarrow w\mu_\kappa\ ,\ \Sigma_w \leftarrow w\Sigma_\kappa + (1-w)I$
- $x_0 \sim \mathcal{N}(\mu_w, \Sigma_w)$
- $x_t \leftarrow tx_1 + (1-t)x_0$
- $\mathcal{L} \leftarrow ||x_1 - x_0 - v_\theta(x_t, t)||^2 + \beta D_{KL}\left(\mathcal{N}(\mu_\kappa, \Sigma_\kappa) \| \mathcal{N}(0, I)\right)$
- Update (θ, ϕ) with $\nabla\mathcal{L}$

end

Return : μ_{ϕ^*}, Σ_{ϕ^*}, v_{θ^*}

Algorithm 2: Sampling

Input : μ_ϕ, Σ_ϕ, v_θ, w, κ (optional), `ODESolver`

if κ *is given* **then**
- $\mu_\kappa \leftarrow \mu_\phi(\kappa), \Sigma_\kappa \leftarrow \Sigma_\phi(\kappa)$
- $\mu_w \leftarrow w\mu_\kappa\ ,\ \Sigma_w \leftarrow w\Sigma_\kappa + (1-w)I$
- $x_{\text{init}} \sim \mathcal{N}(\mu_w, \Sigma_w)$

else
- $x_{\text{init}} \sim \mathcal{N}(0, I)$

end

$x_{\text{sampled}} \leftarrow$ `ODESolver`$(x_{\text{init}}, v_\theta)$

Return : x_{sampled}

Training. During training as outlined in Alg. 1, we sample $w \sim \mathcal{U}(0, 1)$ independently for each example such that $v_\theta(x_t, t)$ learns to flow from a mixture of distributions, by minimizing

$$\mathcal{L}_{\text{ours}}(\theta, \phi) = \mathop{\mathbb{E}}_{\substack{x_1, \kappa \sim q(x_1, \kappa) \\ w \sim \mathcal{U}(0,1) \\ x_0 \sim q_\phi(x_0 \mid \kappa, w)}} \Big[l(x_0, x_1) + \beta R\left(x_0, \kappa, w\right) \Big]. \tag{5}$$

where $R\left(x_0, \kappa, w\right) = D_{KL}\left(q_\phi(x_0 \mid \kappa, w) \| \mathcal{N}(0, I)\right)$ is the KL divergence loss. This design has some important benefits. In Sec. 6.2, we show empirically that with this formulation, we can choose the regularization weight β in Eq. equation 5 to be very small (order of 10^{-5}) without losing training stability and coverage of the Gaussian prior for sampling. This essentially allows for a larger deviation of q_ϕ from the standard Gaussian distribution and therefore more complexity in its coverage of the data distribution modes. More importantly, the ability to use lower β values allows for obtaining lower trajectory curvatures. See Tab. 1 and Supplementary A.2 for discussion and empirical evidence.

Model	β	Curvature (↓)
Rectified Flow	∞	0.0467
Fast-ODE	20	0.0388
MixFlow (ours)	10^{-5}	**0.0366**

Table 1: **Trajectory curvature.** We compare the generative trajectory of MixFlow with Rectified Flow and Fast-ODE. With lower β coefficient for the KL divergence loss, MixFlow achieves $\sim 5\%$ improved curvature over Fast-ODE.

Sampling. For sampling as detailed in Alg. 2, the source distribution can be chosen to be a Gaussian interpolant like during training, if κ is available, or the standard normal as fallback. Furthermore, with a given κ, we can freely choose w for sampling the initialization of the ODE. In Sec. 6.1, we show that w provides control over the speed-quality tradeoff at inference time, which eliminates the need to retrain in order to push the performance with both low and high sampling budget. Even without κ at inference time, e.g., in the case of $\kappa = x_1$ being the data sample itself, our method still achieves straightened sampling trajectories starting from the standard Gaussian source distribution.

5 EXPERIMENTS

Our experimental evaluation of MixFlow comprises multiple established benchmark datasets for unconditional image generation. We further consider both low and high sampling budget settings using different ODE solvers to assess the achieved trade-offs between sampling quality and speed.

5.1 UNCONDITIONAL GENERATION ON CIFAR10

We demonstrate the effectiveness of our approach by comparing with previous methods on the CIFAR10 dataset Krizhevsky (2009). We train a rectified flow model with distribution mixing, where we define κ as the data sample itself and choose $\beta = 10^{-5}$ in the loss equation 5. We follow the exact configuration of Fast-ODE Lee et al. (2023) for fair comparison.

Method	Solver	NFE	FID (↓)
Rectified Flow	RK45	127	2.58
FM - OT		142	6.36
Minibatch-OT		133.9	3.58
Fast-ODE		118	2.45
QAC		-	2.43
Ours		124.7	**2.27**
Fast-ODE	Heun's 2^{nd} order	5	24.40
QAC			19.68
Ours			**19.29**
Fast-ODE	Heun's 2^{nd} order	9	9.96
QAC			10.28
Ours			**8.97**

Table 2: **Speed-quality trade-off on CIFAR10.** We evaluate our method on CIFAR10 in terms of FID using different ODE solvers. In the top, we fully simulate the ODE trajectory using the RK45 adaptive solver. MixFlow achieves notable improvements in FID with a comparable number of function evaluations (NFEs). Our method further improves sample quality with a small number of NFE as can be seen for 5 and 9 function evaluations with Heun's 2^{nd} order solver.

5.1.1 CURVATURE EVALUATION

We first evaluate the curvatures of the generative trajectories induced by the model trained with MixFlow and compare it with that of Fast-ODE and Rectified Flow. We generate 10K trajectories using an Euler sampler with 128 inference steps. Further details on the curvature computation can be found in the supplementary. We show the results in Tab. 1. MixFlow achieves $\sim 22\%$ improvement compared to Rectified Flow and $\sim 5\%$ improvement compared to Fast-ODE. Due to our mixture formulation, we are able to train the vector-field model using a learned source distribution $q_\phi(x_0 \mid \kappa)$ with a much lower KL divergence weight (β), and hence it can deviate further from the standard Gaussian distribution, thereby achieving lower curvature in its generative paths.

5.1.2 GENERATION EVALUATION

The evaluation w.r.t. Fréchet Inception Distance (FID) score covers both low and high sampling budget. We sample via full ODE simulation using the RK45 adaptive step-size solver for the high sampling budget. For the low number of sampling steps, we use the Heun 2^{nd} order solver to generate samples using 5 and 9 number of function evaluations (NFEs). The results are shown in Tab. 2.

Full Simulation. We compare with trajectory curvature minimizing methods: Rectified Flow Liu et al. (2023), flow matching Lipman et al. (2023), QAC Liang et al. (2024), and also with methods that optimize the forward coupling:Tong et al. (2024), Fast-ODE Lee et al. (2023). Our method shows a reduction of $\sim 12\%$in FID compared to the standard Rectified Flow model, and $\sim 7\%$ improvement compared to the Fast-ODE with a comparable NFE. This shows that our method can better capture the diversity of the data distribution compared to previous methods.

Low Sampling Budget. We compare with the most related and recent baselines Fast-ODE Lee et al. (2023) and QAC Liang et al. (2024). When NFE $= 5$, our method achieves $\sim 20\%$ improvement compared to Fast-ODE in FID, and $\sim 2\%$ improvement compared to QAC. For NFE $= 9$, our method improves FID by $\sim 10\%$ compared to Fast-ODE and $\sim 12.7\%$ improvement in comparison with QAC. This highlights the effectiveness of our approach in the low sampling budget regime.

	Model / NFE	β	4	5	10	20	32	64	128
FFHQ	Fast-ODE	10	**32.58**	25.33	13.21	8.85	7.54	6.91	7.01
		20	38.23	29.12	14.03	8.78	7.08	5.95	5.72
		30	41.16	30.75	14.37	8.76	6.90	5.45	4.93
	Ours	5×10^{-5}	33.72	**25.04**	**12.23**	**7.52**	**5.31**	**4.01**	**3.75**
	Model / NFE	β	4	5	10	20	32	64	128
AFHQ	Fast-ODE	10	21.80	18.04	11.80	9.05	8.22	7.47	7.21
		20	25.73	20.11	10.56	6.89	5.74	4.92	4.55
		30	30.84	23.08	11.17	6.66	5.37	4.40	3.96
	Ours	5×10^{-5}	**19.72**	**15.57**	**7.95**	**5.05**	**4.30**	**3.65**	**3.33**

Table 3: **Comparison with Fast-ODE.** using with different KL divergence weights β w.r.t FID-10k on the FFHQ and AFHQv2 64×64 datasets. Our model outperforms Fast-ODE for different β choices on almost all NFEs. MixFlow provides overall the best trade-off between sampling speed and quality without the need to retrain with a different β parameter.

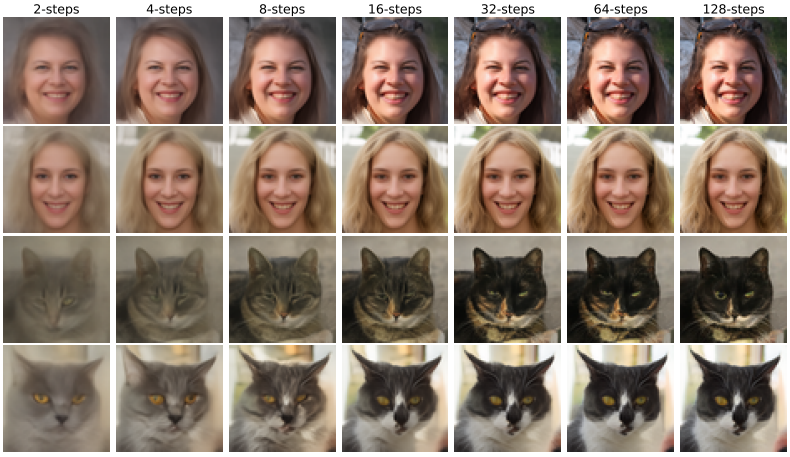


Figure 2: **Qualitative Results.** We show our method's generation on FFHQ (rows 1-2) and AFHQv2 (rows 3-4) datasets using different steps. MixFlow requires few steps to generate reasonable outputs.

5.2 Unconditional Generation on FFHQ & AFHQ

We further train and evaluate our method on higher resolution datasets FFHQ 64×64 Karras et al. (2019) and AFHQv2 64×64 Choi et al. (2020). We train the models with κ as the data sample, and with $\beta = 5 \times 10^{-5}$. We evaluate the performance with FID-10K using the Euler solver across different sampling steps. We compare against Fast-ODE Lee et al. (2023) which reports several results that differ in the choice β in Tab. 3. In both datasets, MixFlow outperforms their models with different β values for almost all NFEs. Despite the comparability of $\beta = 10$ at low NFEs with MixFlow, its performance degrades with higher NFEs. On the other hand, our method achieves the best trade-off between speed and quality and improves the FID for all NFEs without the need to retrain with different β values. In addition, we present qualitative examples for both datasets in Figure 2. With very few steps (<10), the generations are of already of a reasonable quality.

6 Analysis

6.1 Choice of Conditioning for Source Distribution

In addition to our default choice of $\kappa = x_1$ as the data sample itself, we explore two additional instantiations of κ. The first choice is defining κ as the class label assigned to each sample of $p_1(x)$, we call it κ_c. While this choice seems to violate the assumptions of unconditional generation, our

β / NFE	2	4	10	20	32	64	128
∞	171.7	54.5	13.16	6.90	5.02	3.63	3.04
1	168.35	53.64	13.29	6.86	4.97	3.58	3.02
10^{-3}	148.36	46.20	12.10	6.55	4.91	3.59	3.02
10^{-5}	99.30	29.64	**9.02**	**5.23**	**3.90**	**2.95**	**2.52**
10^{-6}	93.45	27.62	9.20	5.80	4.59	3.61	3.21
5×10^{-7}	**89.34**	**27.61**	10.07	6.64	5.39	4.39	3.92

Table 4: **Effect of regularization weight β.** Lower β values enabled by MixFlow result in improved FID. However, regularization is still required, as for smaller values in the order of 10^{-8}, we observe that the source distribution collapses. We choose $\beta = 10^{-5}$ as a default.

Input / NFE	2	4	10	20	32	64	128
Rectified Flow	171.7	54.5	13.16	6.90	5.02	3.63	3.04
Noise (κ_n)	157.43	49.83	11.40	5.86	4.31	3.15	2.79
Label (κ_c)	160.17	48.65	11.35	5.89	4.37	3.27	2.82
Sample	**99.30**	**29.64**	**9.02**	**5.23**	**3.90**	**2.95**	**2.52**

Table 5: **Effect of conditioning signal κ.** κ_n denotes $\kappa \sim \mathcal{N}(0, I)$ and κ_c denotes the class label assumption. We sample from the κ_n and κ_c models with $w = 0$. All choices of κ lead to improvements over rectified flow (first row). Choosing κ as the sample is best because it is maximally informative of the data distribution.

goal is to simply demonstrate the possibility of using a signal that is available during inference with our framework, which can motivate its effectiveness in more complex conditional generation tasks. Each class label is represented with a learnable embedding. The second choice explores the opposite of $\kappa = x_1$, where we assume $\kappa \sim \mathcal{N}(0, I)$ is a noise sample from a standard Gaussian distribution, referred to as κ_n, representing a case uncorrelated with the data distribution.

Effect on Performance. We explore the impact of choosing κ in comparison with the standard Rectified Flow model in Table 5. For fair comparison, both models are evaluated with $w = 0$, i.e, with standard Gaussian. We see that the all choices of κ improve FID compared to the baseline. Interestingly, we observe that an uninformative κ_n (second row) can still improve FID for all sampling steps. This is due to the flexibility of the learnable forward coupling, which, through optimization in Eq. equation 5, learns to map samples of $\mathcal{N}(0, I)$ to a sub-region that is more aligned with the data distribution. Even comparing κ_n with κ_c, κ_n slightly outperforms it. However, this advantage drops as w increases (see supplementary). Nevertheless, defining $\kappa = x_1$ (last row) gives the best FID values across all steps, which reflects the importance of a maximally informed distribution.

6.2 Distance Between Mixed Distributions

We explore the effect of the KL divergence weight hyperparameter β in Eq. equation 5. β controls the deviation of the conditioned distribution from the standard Gaussian distribution. Therefore, it defines the width of the continuous range of that the model is exposed to. We train different models with different β values on CIFAR10 and evaluate them using an Euler solver across several choices of sampling steps. Table 4 shows the results compared to the standard Rectified Flow model. We see that for all steps, the FID improves as β decreases, achieving clear improvements at $\beta = 10^{-5}$. As β goes below 10^{-5} the performance at low NFEs improves, while it deteriorates for high NFEs. Hence, we find $\beta = 10^{-5}$ to provide considerable improvements while maintaining stability. Our results show that allowing the conditional distribution to deviate sufficiently from the standard Gaussian is essential to achieve faster sampling, which is uniquely enabled by our proposed MixFlow.

7 Conclusion

We addressed in this work the sampling efficiency problem of in flow models through the lens of curvature minimization. We presented κ-FC, a general formulation of learnable forward couplings for rectified flows that can leverage arbitrary signals. We highlighted the limitations by naively training with κ-FC and as a solution proposed MixFlow a training strategy that mixes conditional and unconditional distributions while training flow models. MixFlow successfully minimizes the trajectory curvature, improves performance under fixed sampling budget, and leads to faster convergence.

Limitations and Future Work. As our κ-FC formulation abstracts from a generic conditioning variable, we are excited to apply our method to other instances like text prompts, besides the current set of noise, labels, and data samples. Furthermore, while MixFlow reduces the regularization of the learnable forward coupling (low KL divergence weight), which minimizes curvature and speeds up sampling, it still requires a Gaussian assumption. Therefore, we will investigate further relaxations while maintaining performance in future work.

REFERENCES

David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbot, and Eric Gu. Tract: Denoising diffusion models with transitive closure time-distillation. *ArXiv*, abs/2303.04248, 2023. URL https://api.semanticscholar.org/CorpusID:257404979.

Ricky T. Q. Chen. torchdiffeq, 2018. URL https://github.com/rtqichen/torchdiffeq.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. GENIE: Higher-order denoising diffusion solvers. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=LKEYuYNOqx.

Zhengyang Geng, Ashwini Pokle, Weijian Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xQVxo9dSID.

Pengsheng Guo and Alex Schwing. Variational rectified flow matching. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2025. URL https://openreview.net/forum?id=ZLL6SYNptz.

Xiaoran Hao and Patrick Shafto. Coupled variational autoencoder. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 12546–12555. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/hao23b.html.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577. Curran Associates, Inc., 2022a.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577. Curran Associates, Inc., 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a98846e9d9cc01cfb87eb694d946ce6b-Paper-Conference.pdf.

Beomsu Kim, Yu-Guan Hsieh, Michal Klein, marco cuturi, Jong Chul Ye, Bahjat Kawar, and James Thornton. Simple reflow: Improved techniques for fast flow models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=fpvgSDKXGY.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

Alex Krizhevsky. Learning multiple layers of features from tiny images. pp. 32–33, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Sangyun Lee, Beomsu Kim, and Jong Chul Ye. Minimizing trajectory curvature of ODE-based generative models. In *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023.

Yuchen Liang, Yuchan Tian, Lei Yu, Huaao Tang, Jie Hu, Xiangzhong Fang, and Hanting Chen. Learning quantized adaptive conditions for diffusion models. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXXXI*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=2uAaGwlP_V.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *ArXiv*, abs/2101.02388, 2021. URL https://api.semanticscholar.org/CorpusID:230799531.

Robert J. McCann. A convexity principle for interacting gases. *Advances in Mathematics*, 128(1): 153–179, 1997. ISSN 0001-8708. doi: https://doi.org/10.1006/aima.1997.1634. URL https://www.sciencedirect.com/science/article/pii/S0001870897916340.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 23–29 Jul 2023.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Gianluigi Silvestri, Luca Ambrogioni, Chieh-Hsin Lai, Yuhta Takida, and Yuki Mitsufuji. VCT: Training consistency models with variational noise coupling. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=CMoX0BEsDs.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WNzy9bRDvG.

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*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:257280191.

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*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Fu-Yun Wang, Ling Yang, Zhaoyang Huang, Mengdi Wang, and Hongsheng Li. Rectified diffusion: Straightness is not your need in rectified flow. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=nEDToD1R8M.

Zibin Wang, Zhiyuan Ouyang, and Xiangyun Zhang. Block flow: Learning straight flow on data blocks, 2025b.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=JprM0p-q0Co.

Sirui Xie, Zhisheng Xiao, Diederik P. Kingma, Tingbo Hou, Ying Nian Wu, Kevin Patrick Murphy, Tim Salimans, Ben Poole, and Ruiqi Gao. Em distillation for one-step diffusion models. *ArXiv*, abs/2405.16852, 2024. URL https://api.semanticscholar.org/CorpusID:270062581.

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. *CoRR*, abs/2407.02398, 2024. URL https://doi.org/10.48550/arXiv.2407.02398.

Zhenyu Zhou, Defang Chen, Can Wang, Chun Chen, and Siwei Lyu. Simple and fast distillation of diffusion models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 40831–40860. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/47ee3941a6f1d23c39b788e0f450e2a7-Paper-Conference.pdf.

Supplementary Materials

A Trajectory Curvature

A.1 Computation Details

In order to demonstrate the impact of our method MixFlow on the curvatures of the generative paths, we follow the procedure in Fast-ODE Lee et al. (2023) to estimate the curvature of the generated trajectories. We generate 10000 trajectories using an Euler solver with 128 steps, and compute the average curvature for an optimized vector-field model v_θ as:

$$C(v_\theta) = \mathbb{E}_{t,x_0}\left[\|x_0 - x_1 - v_\theta(x_t, t)\|^2\right] \tag{6}$$

where $t \sim \mathcal{U}(0,1)$, $x_0 \sim p_0(x) = \mathcal{N}(0, I)$, and x_1 is sampled deterministically with an ODE solver: $x_1 = \texttt{ODESolver}(x_0, v_\theta)$.

The curvature definition here closely resembles the degree of intersection $I(q)$ defined in eq. (4) in the main paper, where the differences are: (1) $I(q)$ is a function of the coupling $q(x_0, x_1)$ for the source and target distributions, whereas $C(v_\theta)$ is a function of an optimized vector field model, (2): x_1 in $I(q)$ is sampled from the coupling, whereas x_1 in $C(v_\theta)$ is computed from x_0 deterministically with the ODE solver.

A.2 Effect of β

As we argue in the main paper, a main advantage of MixFlow is that it allows training our learnable forward coupling (κ-FC) with much lower KL Divergence weight β compared to previous work, FastODE. Here, we show empirically that lower β values correlate with lower curvatures of the generative trajectories. We train multiple MixFlow models with different β values ranging from $5 \times 10\hat{}-7$ to 1, and then evaluate the curvature of each model in order to see the trend. In Figure 3, we see that as β decreases (goes right on the x-axis), the curvature values (in the y-axis) tend to decrease.

B Implementation Details

We share in this section the details about the models used in the experiments, as well as the training hyperparameters.

Vector Field Model $v_\theta(x_t, t)$. We use the UNet architecture used in Fast-ODE Lee et al. (2023) for fair comparison, which follows the DDPM++ implementation of EDM Karras et al. (2022a). You can see in Table 6 the configuration of the network for used each dataset in the top part. In the bottom part, we show the corresponding training hyperparameters. All models optimized with Adam Kingma & Ba (2014) with a learning rate that linearly increases until 2×10^{-4} and then remains constant for the rest of the iterations. We also use Exponential Moving Average (EMA) on the model weights with ratio 0.9999, and we find it to be a critical factor for convergence. The CIFAR10 experiments all were trained on a single A100 80GB GPU. The FFHQ and AFHQv2 experiments each were trained on 4 A100 40GB GPUs, where the effective batch size shown in the tables was distributed equally among the GPUs.

Source Prediction Network $q_\phi(x_0 \mid \kappa)$. We use a small UNet by adapting the `UNet2DModel` implementation from the `diffusers` library, version `0.32.2`. Its hyperparameters are shown in Table 7, where the parameters are aligned with the library's implementation for ease of reproduceability. The input to the UNet is expected to be $(3 \times H \times W)$, where H, W can differ depending on the dataset used. However, depending on the input κ, the first layer might differ slightly. When κ is the data sample (as in the default experiments) or a noise sample (κ_n), there is no change applied to the network. When κ is the class label (κ_c), an embedding layer is appended before, which maps the

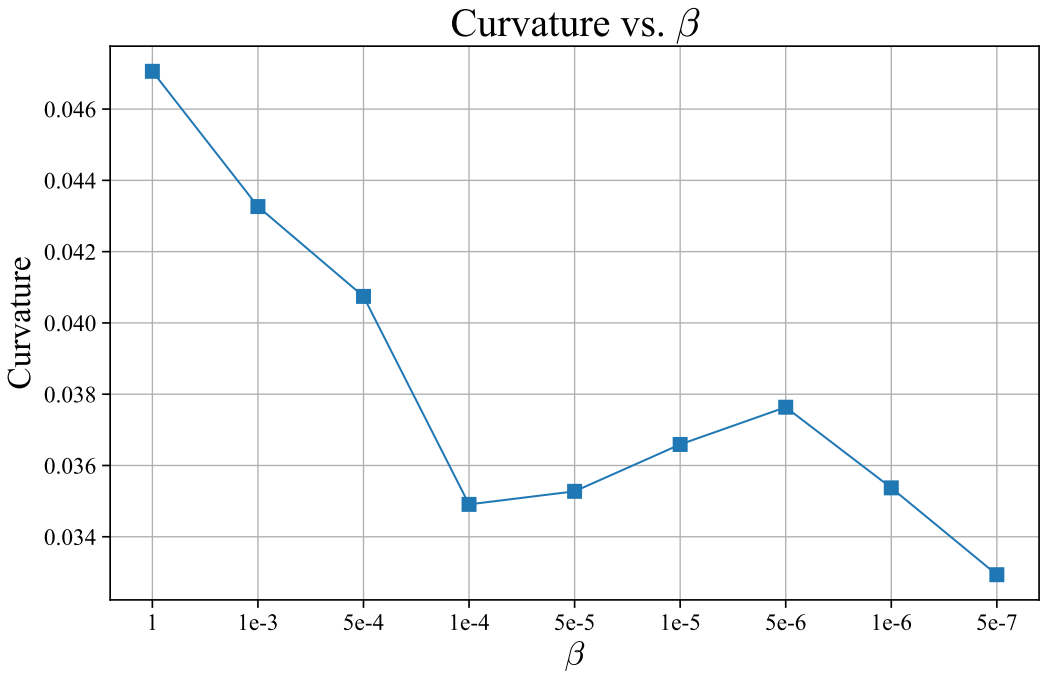


Figure 3: **Curvature vs. β.** We show how the curvature of the generative trajectories changes with different β values. We can see a clear trend of lower curvature with a lower β value

	Parameters	CIFAR10	FFHQ	AFFHQv2
UNet Parameters	Channel Size	128	128	128
	Channel Multiplier	[2,2,2]	[1,2,2,2]	[1,2,2,2]
	Blocks per Layer	4	4	4
	Attention Resolution	16	16	16
	Dropout Probability	0.13	0.05	0.25
	Embedding type	positional	positional	positional
	Model Size	55.7M	61.8M	61.8
Training Setup	EMA ratio	0.9999	0.9999	0.9999
	Iterations	500K	500K	300K
	Batch Size	128	256	256
	Optimizer	Adam	Adam	Adam
	Learning Rate (LR)	2×10^{-4}	2×10^{-4}	2×10^{-4}
	LR Scheduling	constant	constant	constant
	LR Warmup steps	5000	39060	39060

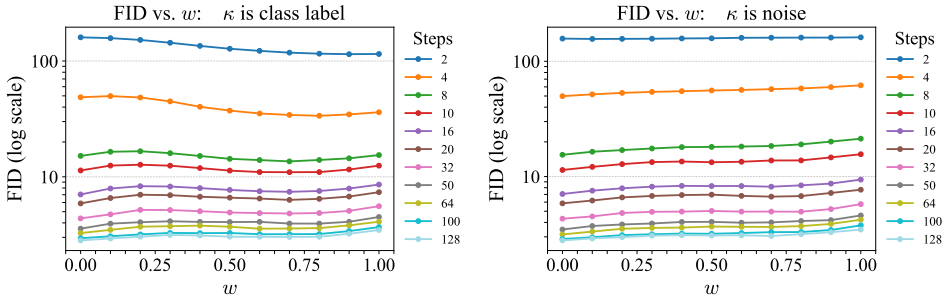
Table 6: **Model and Experiments Configurations.** In the upper part, we show the UNet configuration for the vector field model v_θ. In the lower part we show the training hyperparameters. Each column show the configuration for a specific dataset.

Parameters	
Time Embedding Type	positional
Flip Sin to Cos	true
Down Block Types	[`DownBlock2D`, `DownBlock2D`, `DownBlock2D`, `AttnDownBlock2D`]
Up Block Types	[`AttnUpBlock2D`, `UpBlock2D`, `UpBlock2D`, `UpBlock2D`]
Block Out Channels	[32, 64, 64, 64]
Layers per Block	2
Activation Function	`silu`
Attention Head Dim	8
Model Size	$\sim$ 2M

Table 7: **Source Prediction Network $q_\phi(x_0 \mid \kappa)$ Configuration.** We show the UNet configurations for the source prediction parametrization. The params follow the `diffusers` library definition of the UNet model.

class labels into embeddings of size $3HW$, which are reshaped into $3 \times H \times W$ and then fed to the UNet. The network outputs the mean and the log variance of the distribution. We assume that the covariance is diagonal.

Evaluation. We use the Euler ODE solver from the `torchdiffeq` Chen (2018) library, and its `scipy` Virtanen et al. (2020) library wrapper for the RK45 solver, where we set the `rtol` and `atol` parameters both to 10^{-5}. As for Heun's 2^{nd} solver, we follow the manual implementation as in Fast-ODE Lee et al. (2023).



(a) κ_c: Class label conditioning

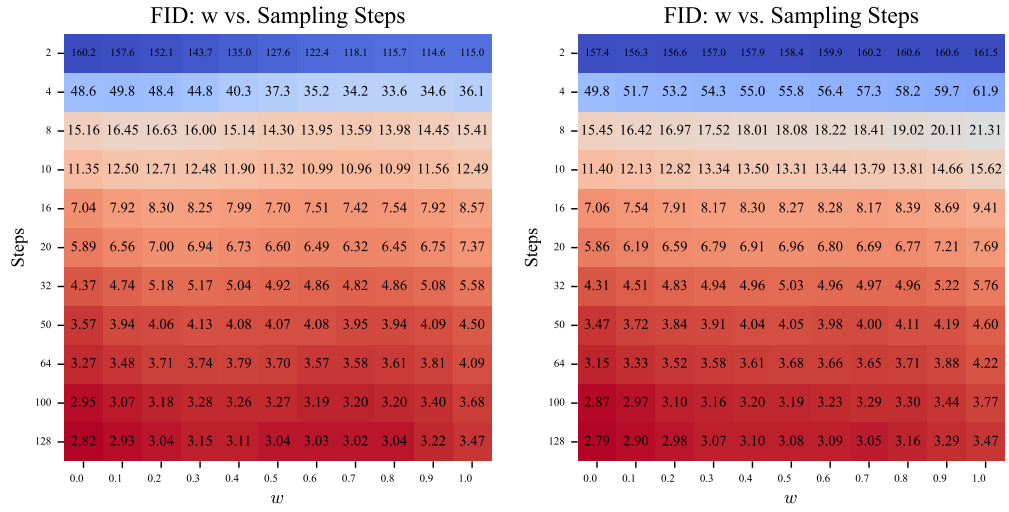
(b) κ_n: Noise conditioning

Figure 4: **Effect of interpolation weight w.** We visualize the effect of varying w (x-axis) during sampling on FID (y-axis) across different numbers of sampling steps (different lines). **(a)** Sampling with a few steps benefits from a larger weight of the source distribution conditioned on class label κ_c, while for many steps, the unconditional standard Gaussian is better suited. **(b)** With conditioning on uncorrelated Gaussian noise κ_n, the best FID is achieved for $w = 0$, i.e., not using the conditional distribution at all during sampling. However, note that even in this case training the vector field on a mixture of distributions still improves performance as shown in Tab. 5.

C Additional Analysis

C.1 Training Efficiency

We highlight the training efficiency as another notable advantage of our formulation. Fig. 6 shows the FID during training compared to the final performance of one of our strongest baselines Fast-ODE Lee et al. (2023). Note that MixFlow achieves approximately the same performance as Fast-ODE using only 60% of the full training iterations. Therefore, our method not only accelerates sampling by straightening flow trajectories, but also training in terms of convergence.



(a) κ_c: Class label conditioning (b) κ_n: Noise conditioning

Figure 5: **FID for Sampling Steps vs. weight w.** We show the FID across different sampling step choices for both κ_c and κ_n as the interpolation parameter w changes. This is the numerical version of Figure 3 in the main paper, which provides a more accurate look into the change of FID values, where red indicates lower FID and blue indicates higher FID.

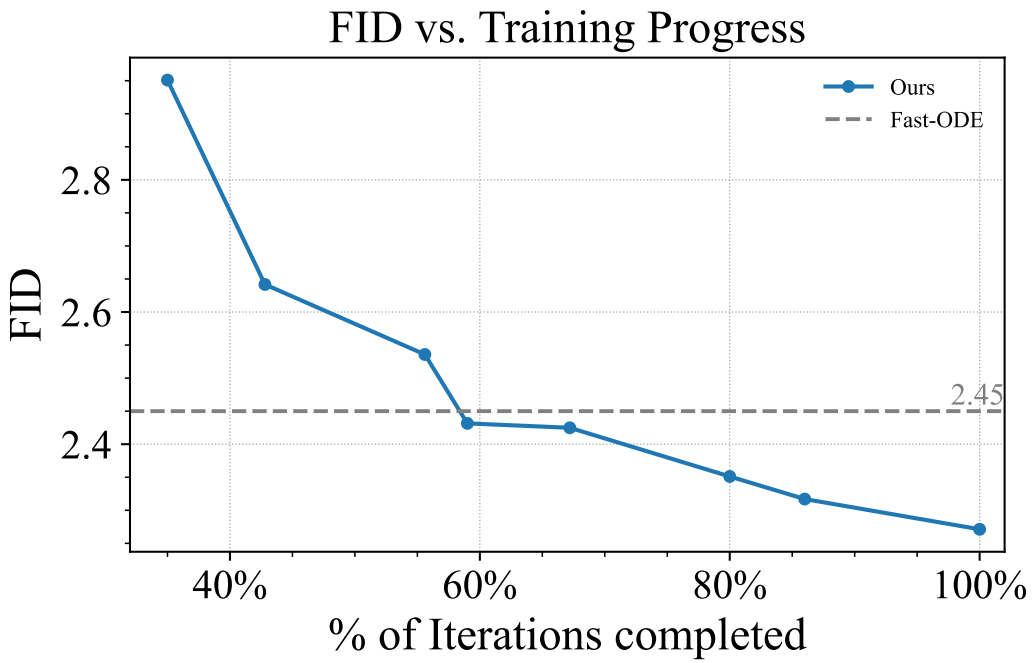


Figure 6: **FID vs. training progress.** Samples are generated with the RK45 solver across different step of the training process. Our method achieves the same performance as Fast-ODE (gray dotted line) with only 60% of the training budget.

C.2 Effect of w

In the cases where the conditioning signals are available during inference, we can use them to vary the mixture parameter w while sampling. So to understand the importance of choosing w, we train two different models with κ_c, κ_n. Figure 4b shows the results for κ_n. We can see that the FID (y-axis) achieves the best value at $w = 0$ and then degrades as w (x-axis) increases from 0 to 1, showing that when the signal is not very informative, flowing from the standard Gaussian distribution where $w = 0$

results in better generation. As for κ_c, as shown in Figure 4a, an interesting pattern appears. For a low number of sampling steps (2,4), we notice that the FID tends to improve as w increases, showing that the distribution learned from the informative signal has a positive effect with few sampling steps. As the number of sampling steps increases, we see that the FID is best at $w = 0$ and starts dropping as w increases. Hence, we conclude that, with a sufficiently informative signal, w can control the quality-speed tradeoff during inference. So, depending on the sampling budget available, w can be tuned at inference to provide the best FID accordingly.

We also provide in Figure 5 a more fine-grained version of the figure to clearly see the difference in FID values between the two choices: κ_c: class label, and κ_n: standard Gaussian noise. The histograms show the FID values for the choice of sampling steps (y-axis) against the mixture parameter w (x-axis). Stronger redness indicates lower FID, and stronger blueness indicates higher FID. When $w = 0$, we notice that κ_n performs slightly better than κ_c, as shown in Table 3 in the main paper. As w increases, however, the performance of κ_c becomes better for all sampling steps compared to κ_n, which reflects the effect of class labels as a conditioning signal.

C.3 Sampling Steps

In order to highlight the effectiveness of MixFlow for improving sampling speed, we show in Figure 7 some qualitative generations across different sampling step choice in comparison with Rectified Flow. Notice that with low sampling steps (2,4), MixFlow generates higher quality samples in comparison with those of Rectified Flow.

D Additional Qualitative Results

We include more qualitative examples that are generated with MixFlow when κ is the data sample, and $\beta = 10^{-5}$. All the images are generated with Euler solver with 64 sampling steps. We show the generations for CIFAR10 in Figure 8, FFHQ 64×64 in Figure 9, and AFHQv2 64×64 in Figure 10.

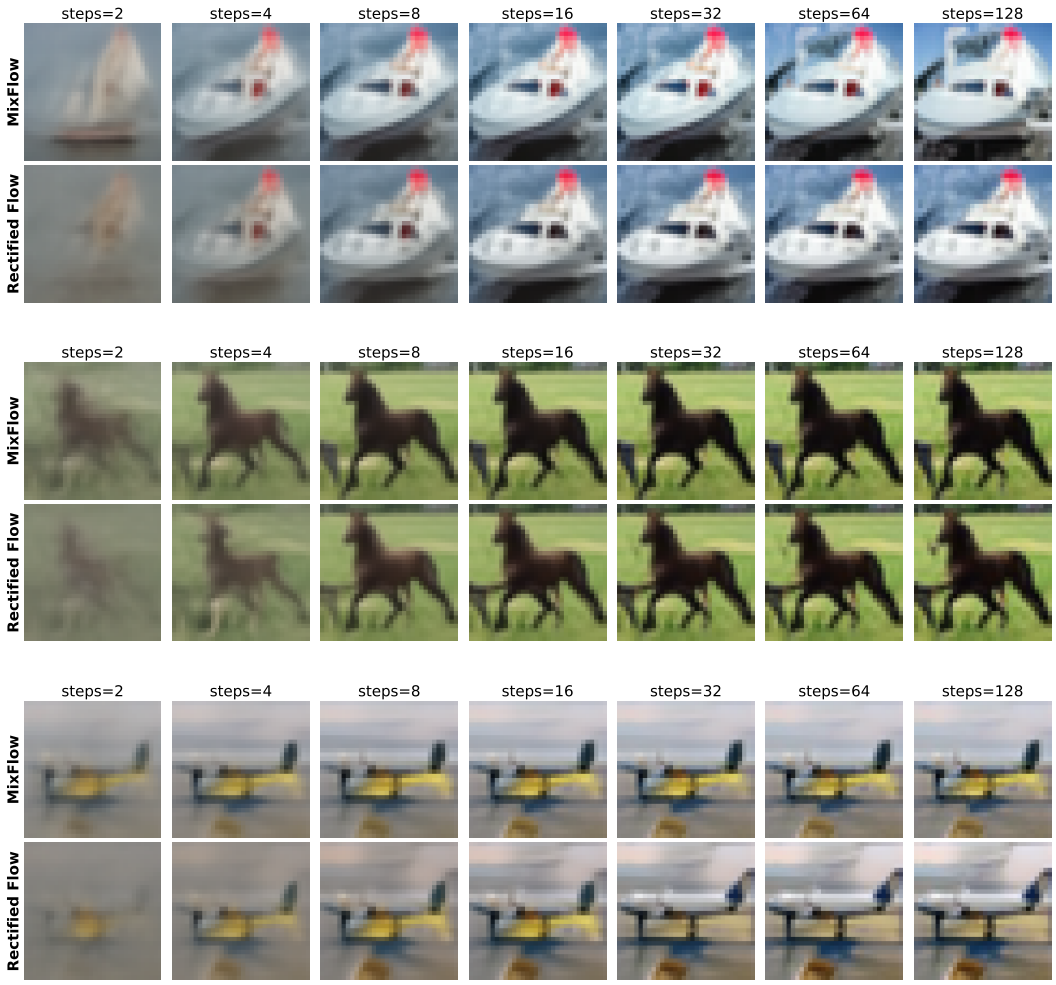


Figure 7: **Comparison against Rectified Flow**. We show multiple examples for generated images with different sampling steps and compare against Rectified Flow. We highlight that for a low number of sampling steps (2,4), MixFlow generates much clearer images compared to Rectified Flow.

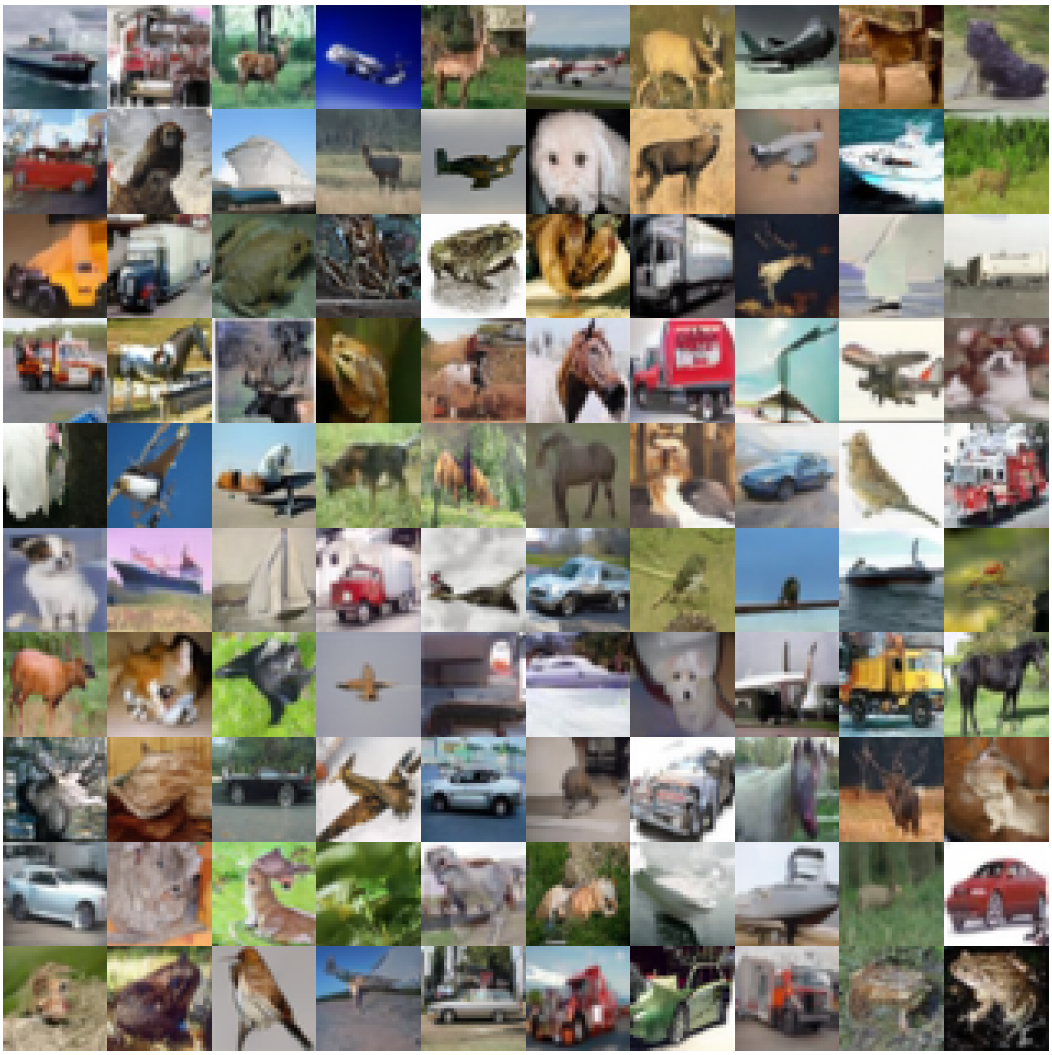


Figure 8: **Qualitative Results on CIFAR10**

Figure 9: **Qualitative Results on FFHQ** 64×64

Figure 10: **Qualitative Results on AFHQv2** 64×64