Straighten Viscous Rectified Flow via Noise Optimization

Jimin Dai[1], Jiexi Yan[2], Jian Yang[1], Lei Luo[1]
[1]PCA Lab, Nanjing University of Science and Technology, [2]Xidian University
{jimindai, csjyang}@njust.edu.cn, {jxyan1995, luoleipitt}@gmail.com

Abstract

The Reflow operation aims to straighten the inference trajectories of the rectified flow during training by constructing deterministic couplings between noises and images, thereby improving the quality of generated images in single-step or few-step generation. However, we identify critical limitations in Reflow, particularly its inability to rapidly generate high-quality images due to a distribution gap between images in its constructed deterministic couplings and real images. To address these shortcomings, we propose a novel alternative called Straighten Viscous Rectified Flow via Noise Optimization (VRFNO), which is a joint training framework integrating an encoder and a neural velocity field. VRFNO introduces two key innovations: (1) a historical velocity term that enhances trajectory distinction, enabling the model to more accurately predict the velocity of the current trajectory, and (2) the noise optimization through reparameterization to form optimized couplings with real images which are then utilized for training, effectively mitigating errors caused by Reflow's limitations. Comprehensive experiments on synthetic data and real datasets with varying resolutions show that VRFNO significantly mitigates the limitations of Reflow, achieving state-of-the-art performance in both one-step and few-step generation tasks.

1. Introduction

Diffusion models (DMs) [9, 23, 25] have attracted significant attention due to their remarkable generative capabilities and stable optimization process. DMs have outperformed traditional models such as generative adversarial networks (GANs) [1, 4] and Variational autoencoders (VAEs) [30] in various generation tasks. DMs work by designing a forward noise-adding scheme, which progressively transforms the target distribution into the initial distribution (usually a Gaussian distribution), thereby constructing a Probability Flow (PF) from complex to simple distributions. DMs learn the inverse process of PF, and then they can sample Gaussian noises and generate samples that approximate the target distribution through multiple inference steps. Currently, various forward noise-adding schemes have been developed for DMs. Among them, the most straightforward approach is the Rectified Flow (RF) [14, 15, 22]. The core idea of RF is to add noise to the data in a linear manner, constructing a deterministic PF from the Gaussian distribution to the target distribution. The model only needs to learn the velocity field of the straight-line interpolation trajectories between the noises and the samples, Once trained, the model can efficiently generate samples that approximate the target distribution.

Generating samples with DMs typically requires multiple inference steps that consume substantial computational resources and time. RF constructs straight-line interpolation trajectories between noises and samples as the reference, and it theoretically learns a straight flow from the Gaussian distribution to the target distribution. The straight flow implies that RF learns a constant velocity field, which enables few-step or even single-step sampling in an ideal scenario. However, during actual training, the RF struggles to learn a constant velocity field, and the resulting PF trajectories remain curved. Therefore, RF still relies on multiple steps to generate the desired images. RF attributes the curvature to the crossing of reference trajectories during training and employs a Reflow operation to mitigate this issue.

In this paper, we observe that during the actual training process, the probability of crossing between the straight-line interpolation trajectories of randomly matched noises and images at a given time step t is $P \sim O(e^{-c(n\times n)})$, which is very small in high-dimensional data space (Theorem 1). Therefore, we are more inclined to believe that the curvature arises because the neural network model is not an exact solver, intermediate states along different trajectories may become "approximately crossing" due to their high similarity in statistical property, thereby interfering with the model predictions. Furthermore, we conducted an in-depth analysis of the factors that contribute to the success of Reflow, as well as its limitations, detailed in Section 2.2.

To retain the advantages of Reflow while mitigating its drawbacks, we propose a new method called Straighten Viscous Rectified Flow via Noise Optimization (VRFNO). VRFNO introduces a historical velocity term (HVT) as input to the neural velocity field, allowing the model to distinguish the direction of the RF in the presence of approximate crossing. By incorporating historical velocity information, the model can more accurately predict the velocity. Moreover, VRFNO directly uses the original dataset to avoid the distribution gap and achieves more appropriate matches between the noises and the images (we refer to it as *optimized coupling*) by noise optimization. Specifically, we propose a joint training framework for an encoder and a neural velocity field. The encoder first encodes images from randomly matched noise-image pairs and then outputs the corresponding mean and variance matrices. The mean and variance matrices are combined with the noise using the reparameterization technique to generate a new noise, which is then fed into the neural velocity field for training. This joint training approach enables the encoder to adjust the mean and variance of the noise based on the image, thereby optimizing the matches between the noises and the images, which in turn achieves *optimized couplings*. By using these *optimized couplings* to train the velocity field, the model can more efficiently straighten the inference trajectory during training, enabling one-step or few-step generation.

In summary, our contributions are as follows:

- We propose a new method VRFNO to straighten the inference trajectories for higher quality few-step and single-step generation.
- We introduce an HVT as the auxiliary information to enhance the model's accuracy in velocity prediction when the flow approximates crossing.
- We construct *optimized couplings* by optimizing the noise to train the neural velocity field. This avoids relying on *deterministic couplings* used in Reflow, which will constrain the model's generation quality.

2. Preliminary

2.1. Rectified Flow

RF is an ordinary differential equations (ODE)-based DM that aims to learn the mapping between two distributions, π_0 and π_1. In image generation tasks, π_0 typically represents a standard Gaussian distribution, while π_1 corresponds to the target image distribution. For empirical observations $X_0 \sim \pi_0$ and $X_1 \sim \pi_1$ on time $t \in [0, 1]$, RF is defined as:

$$dX_t = v(X_t, t)dt, \tag{1}$$

where intermediate state $X_t = tX_1 + (1 - t)X_0$ represents a time-differentiable interpolation between X_0 and X_1, $v : \mathbb{R}^{n\times n} \times [0, 1] \to \mathbb{R}^{n\times n}$ denotes the velocity field defined on the data-time domain. Since RF uses simple straight-line interpolation to connect X_0 and X_1, its trajectory has a constant velocity field $v_{ref} = X_1 - X_0$. Thus, the training process optimizes the model by solving a least-squares regression problem, i.e., fitting v_{ref} to neural velocity field v_θ:

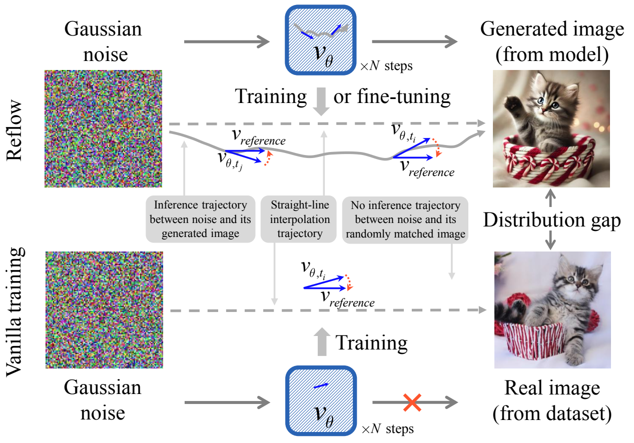

Figure 1. **Comparison between Reflow and Vanilla training mode.** The top illustrates the Reflow training mode, where noises generate images via a pre-trained model, forming *deterministic couplings* for training. These couplings are reused during training by randomly sampling different intermediate states along the trajectories. The bottom describes the vanilla training mode, where noises and images are randomly sampled to form *arbitrary couplings* for training, without reusing data during training.

$$\min \mathbb{E}_{X_0, X_1 \sim \gamma, t \in p(t)} \left[\| v_{ref} - v_\theta(X_t, t) \|^2 \right], \tag{2}$$

where γ represents the coupling of (X_0, X_1), $p(t)$ denotes the time distribution defined on $[0, 1]$. During the inference process, the ODE usually needs to be discretized and solved via the Euler method, expressed as:

$$X_{t+\Delta t} = X_t + \Delta t \cdot v_\theta(X_t, t), t \in \{0, \Delta t, \dots, (N - 1)\Delta t\}, \tag{3}$$

where $\Delta t = \frac{1}{N}$, N represents the total number of steps. Generally, a larger N results in higher-quality generated images, but it requires more computational resources for sampling. The reference trajectory of RF is a straight flow. If RF can converge to the straight flow through training, it can significantly minimize numerical errors in the ODE solver, enabling few-step or single-step generation: as the state moves along the trajectory with uniform linear motion, the velocity at a single time step equals the average velocity over the entire trajectory. As long as the cumulative sum of all time increments throughout the motion equals unity, the final endpoint of the trajectory will remain consistent.

2.2. Analysis of Reflow

The inference trajectory of RF exhibits a curved pattern when trained using the vanilla method. While RF attributes this phenomenon to the crossing of reference trajectories [13] and proposes Reflow as a solution (with implementation details provided in Appendix C), our experiments (Fig.2) and analysis suggest that the effectiveness

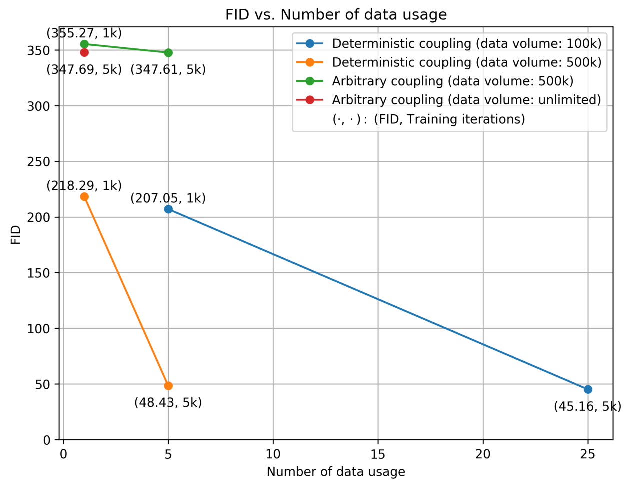

Figure 2. **Impact of coupling type and data reuse on RF's performance.** Training RF from scratch with different coupling types, we control the data volume to construct different data reuse scenarios and observe the impact of coupling type and data reuse on model performance under the same training iterations. Each iteration samples 500 data pairs, ensuring that the total data volume seen by the model remains the same across different reuse scenarios under the same training iterations.

of Reflow cannot be solely explained by its ability to mitigate trajectory-crossing issues. This is particularly evident given that trajectory crossings are extremely rare in high-dimensional spaces, as we will elaborate in Section 3.1.

Through experimental and analytical investigations, the success of Reflow can also be attributed to the following two factors: (1) the images in the training data for Reflow are generated by the pre-trained model, i.e., *deterministic couplings*. This implies the existence of a deterministic trajectory between a coupling that can be learned and inferred by the model, as shown in the first row of Fig.1. Training/fine-tuning with Reflow is analogous to learning this trajectory during training and progressively aligning it with a straight-line trajectory. In contrast, randomly matched data pairs (*arbitrary couplings*) lack explicit trajectory relationships, requiring the model to infer unknown trajectories from noises and constrain them to straight lines, making the training more complex and unstable. As demonstrated in Fig.2, under identical data volume and training iterations, RF trained with *deterministic couplings* achieves superior image quality (green line vs. orange line). (2) Reflow reuses data by sampling intermediate states from different time steps of the same trajectory for training, as depicted in the first row of Fig.1. This training strategy resembles the multi-time-scale optimization methods employed in the distillation, both of which aim to enhance model performance by leveraging multi-level information. Fig.2 demonstrates that more data reuse iterations lead to better performance under *deterministic couplings* (orange line vs. blue line). More details and analyses can be found in Appendix B.

Additionally, the drawbacks of Reflow are evident. As shown in Fig.1, there is a distribution gap between the generated images and the real images. Each iteration of training utilizes images generated by the previous model as training data, leading to the accumulation of errors over time. Consequently, the quality of the images generated by the subsequent model tends to degrade compared to the previous one. To balance high image quality with computational efficiency, it is typically recommended to limit the iterative training to 2 or 3 cycles. Furthermore, while data reuse is effective, its effectiveness is influenced by the volume of data. If the volume of reused data is small, it may not adequately represent the true probabilistic distribution of the data, thereby affecting the model's learning of the PF [14]. On the other hand, excessive reused data will cause significant storage pressure, which increases the consumption of computational resources and reduces training efficiency.

3. Method

We aim to develop an advanced training framework for RF as a superior alternative to Reflow, further improving the quality of generated images in single-step and few-step generation tasks. To achieve this, we propose a novel method called VRFNO and the overview of VRFNO is shown in Fig.3. Specifically, VRFNO introduces an HVT to effectively improves the model's capability to differentiate inference trajectories, thereby substantially mitigating prediction disturbances arising from intermediate state similarities. Furthermore, to circumvent the use of pre-trained model-generated images as training data during the training process, as employed in Reflow, and to ensure that randomly matched noise-image data pairs (X_0, X_1) exhibit desirable characteristics akin to deterministic coupling. We define a new concept "*optimized coupling*" and propose to achieve *optimized coupling* by optimizing noise.

Definition 1. (X_0, X_1) *is called an optimized coupling if it satisfies the following condition:*

$$\|v_\theta(tX_1 + (1-t)X_0) - (X_1 - X_0)\| \leq \varepsilon,$$

where $\varepsilon > 0$ *is a small positive constant.*

3.1. Viscous Rectified Flow

RF attributes the curvature of inference trajectories to the frequent crossings of reference trajectories. Specifically, when noise-image pairs are randomly matched, multiple reference trajectories may cross at the same point X_t at time step t. During training, this causes different reference trajectories to provide identical inputs to the model at time step t, confusing the model and degrading its prediction accuracy. However, our calculations (Theorem 1) show that the probability of trajectory crossings occurring during actual

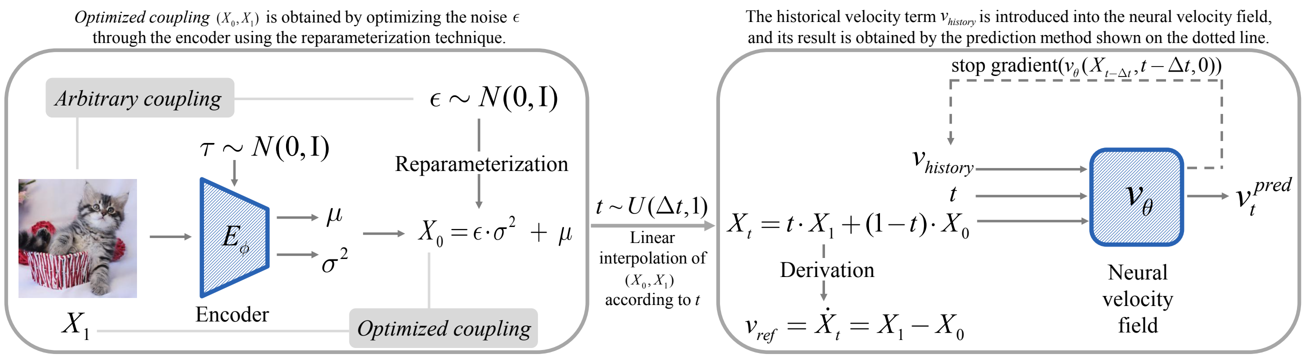
Optimized coupling (X_0, X_1) is obtained by optimizing the noise ϵ through the encoder using the reparameterization technique.
Arbitrary coupling
$\epsilon \sim N(0,\mathrm{I})$
$\tau \sim N(0,\mathrm{I})$
Reparameterization
E_ϕ
μ
σ^2
$X_0 = \epsilon \cdot \sigma^2 + \mu$
X_1
Encoder
Optimized coupling
$t \sim U(\Delta t, 1)$
Linear interpolation of (X_0, X_1) according to t
The historical velocity term $v_{history}$ is introduced into the neural velocity field, and its result is obtained by the prediction method shown on the dotted line.
stop gradient($v_\theta(X_{t-\Delta t}, t-\Delta t, 0)$)
$v_{history}$
t
$X_t = t \cdot X_1 + (1-t) \cdot X_0$
v_θ
v_t^{pred}
Derivation
$v_{ref} = \dot{X}_t = X_1 - X_0$
Neural velocity field

Figure 3. **Overview of VRFNO.** The encoder and the neural velocity field form a joint training framework: randomly matched noise ϵ and image X_1 (called *arbitrary coupling*) are optimized by the encoder to obtain *optimized coupling* (X_0, X_1), which are then used to train the neural velocity field. The introduction of the HVT $v_{history}$ enhances the distinction of the trajectories. Both components work together during training to straighten the inference trajectories, thereby improving the model's predictive accuracy.

training is $P \sim O(e^{-c(n\times n)})$, which is extremely rare in high-dimensional space. More commonly, since the initial states are independently sampled from a standard Gaussian distribution, their statistical properties are very similar. In the early stages of the interpolation trajectory, the intermediate states contain limited image information, causing the differences between the intermediate states to remain small. Since RF learns the mapping relationship between distributions, this high similarity in statistical properties may hinder the model's ability to distinguish between states, leading to blurred predictions. Ideally, RF has a constant velocity field, which means that the intermediate state X_t moves in a straight line from noise X_0 to image X_1 at uniform velocity v_{ref}, forming a straight inference trajectory.

Theorem 1. *In $(n \times n)$-dimensional space, for straight-line interpolation trajectories $X^{(\cdot)} = \left\{X_t^{(\cdot)} : t \in [0,1]\right\}$, the probability of $X^{(i)}$ and $X^{(j)}$ crossing at the point X_t at time step t is $P \sim O(e^{-c(n\times n)}), c > 0$.*

As shown in Theorem 2, the velocity difference $\Delta({v_{ref}}^{(i)}, {v_{ref}}^{(j)})$ between each X_t is more significant than their state difference $\Delta({X_t}^{(i)}, {X_t}^{(j)})$, and this more significant difference enables the model to better identify the currently inferred trajectory, thus improving prediction accuracy. Therefore, we introduce an HVT $v_{history}$ as an approximation of the constant velocity, which is fed into the model at each time step, resulting in the Viscous Rectified Flow (VRF):

$$dX_t = v(X_t, t, v_{history})dt, \quad t \in [\Delta t, 1], \tag{4}$$

where Δt is the time interval between the current state and the adjacent history state. During the inference process, the HVT input to the model at the first time step is 0. In subsequent sampling iterations, each iteration inputs the velocity predicted by the model at the previous time step into the model as an HVT, providing the model with auxiliary information on the flow direction and enhancing the model's ability to make correct predictions.

Algorithm 1 Training of VRFNO.

```
**Input:** image $X_1$, noise $\epsilon$, learning rate $\eta$, time interval $\Delta t$, hyperparameter $\alpha$, encoder $E_\phi$, velocity field $v_\theta$.
**repeat**
    $\mu, \sigma^2 = E_\phi(X_1)$
    $X_0 = \epsilon \cdot \sigma^2 + \mu$
    $v_{ref} = X_1 - X_0$
    Sample $t \sim U(\Delta t, 1)$
    $X_t = tX_1 + (1-t)X_0$
    $X_{t-\Delta t} = (t-\Delta t)X_1 + (1-t+\Delta t)X_0$
    $v_{history} = stopgrad(v_\theta(X_{t-\Delta t}, t-\Delta t, 0))$
    $v_t = v_\theta(X_t, t, v_{history})$
    $L(\theta, \phi) \leftarrow$
        $\mathbb{E}[d(v_{ref}, v_t)] + \frac{\alpha}{2}(\sigma^2 + \mu^2 - 1 - \log(\sigma^2))$
    $\theta \leftarrow \theta - \eta \nabla_\theta L(\theta, \phi)$
    $\phi \leftarrow \phi - \eta \nabla_\phi L(\theta, \phi)$
**until** convergence
```

Theorem 2. *For each intermediate state X_t along a PF trajectory, the velocity difference between X_t are greater than their state difference:*

$$\Delta({v_{ref}}^{(i)}, {v_{ref}}^{(j)}) \geq \Delta({X_t}^{(i)}, {X_t}^{(j)}),$$

where $\Delta(\cdot,\cdot) = \mathbb{E}[\|\cdot - \cdot\|_F^2]$ and $\mathbb{E}[\cdot]$ is the expectation.

3.2. Noise Optimization for Optimized Coupling

We propose using noise reparameterization technique to optimize the noise representation. Specifically, we design a joint training framework for an encoder and a neural velocity field (as shown in Fig.3), where the encoder optimizes the noises to achieve *optimized couplings* and uses these optimized data pairs to train the neural velocity field. The complete training process is shown in Algorithm 1. We adopt the working mode of the encoder in the VAE architecture [8]: the encoder E_ϕ takes the image X_1 as input and outputs the corresponding mean matrix μ and variance matrix σ^2, then generates the optimized noise X_0 by combining μ and σ^2 with randomly matched noise ϵ using the reparameterization technique:

$$X_0 = \epsilon \cdot \sigma^2 + \mu. \tag{5}$$

Given the limited number of images in the dataset, to prevent the model from developing a memory effect and ensure diversity in generated images, we introduce random perturbation $\tau \sim N(0, \mathrm{I})$ at the intermediate layers of the encoder, as shown in Fig.3. This ensures that the reparameterized mean and variance matrices still exhibit differences even when multiple Gaussian noises are matched to the same image. Furthermore, after these Gaussian noises matched the same image are reparameterized, they tend to concentrate in a smaller subspace. During training, each image is matched with noise from its corresponding subspace and participates in the training. This approach is similar to the data reuse strategy in Reflow, and it avoids the issue of limiting the model's generalization ability due to the reuse of limited data, thereby helping to improve the performance of the model. After obtaining the optimized noise X_0, we perform straight-line interpolation between it and the corresponding image X_1 to obtain X_t and $X_{t-\Delta t}$:

$$\begin{array}{c} X_t = tX_1 + (1-t)X_0 \\ X_{t-\Delta t} = (t-\Delta t)X_1 + (1-t+\Delta t)X_0 \end{array}, \quad t \in [\Delta t, 1]. \tag{6}$$

We use the $stopgrad$ operator to calculate the HVT $v_{history} = stopgrad(v_\theta(X_{t-\Delta t}, t-\Delta t, 0))$. The neural velocity field v_θ takes the intermediate state X_t, time step t, and HVT as inputs, and learns a constant velocity field by fitting the reference velocity v_{ref}:

$$VCL(\theta, \phi) = \mathbb{E}_{t \in p(t)} \left[d(v_{ref}, v_\theta(X_t, t, v_{history})) \right], \tag{7}$$

where $d(\cdot,\cdot)$ indicates distance metric. This process simultaneously optimizes the encoder E_ϕ to ensure that the reparameterized noises and the corresponding images satisfy the property of *optimized couplings*. Meanwhile, to prevent the encoder from overfitting during training, which could cause the reparameterized noises to rely too heavily on image information from datasets and thus constrain the diversity of generated images, we introduce KL divergence regularization to constrain the mean and variance of the encoder's output near the standard Gaussian distribution:

$$KLL(\phi) = \frac{1}{2}\left(\sigma^2 + \mu^2 - 1 - \log(\sigma^2)\right). \tag{8}$$

Therefore, the total loss of our joint training framework is:

$$L(\theta, \phi) = VCL(\theta, \phi) + \alpha KLL(\phi), \tag{9}$$

Algorithm 2 Sampling of VRFNO.

```
Input: image in dataset $X_1$, sampling steps $N$, encoder $E_\phi$, velocity field $v_\theta$.
Output: generated image $\tilde{X}_1$.
Sample $\epsilon \sim N(0, \mathrm{I})$
$\mu, \sigma^2 = E_\phi(X_1)$
$X_0 = \epsilon \cdot \sigma^2 + \mu$
$\Delta t = 1/N$
$v_{history} = 0$
for $i = 0$ to $N-1$ do
    $v_{pred} = v_\theta(X_{i\Delta t}, i\Delta t, v_{history})$
    $v_{history} = v_{pred}$
    $X_{(i+1)\Delta t} = X_{i\Delta t} + \Delta t \cdot v_{pred}$
end for
$\tilde{X}_1 = X_{N\Delta t}$
```

where α is a hyperparameter that controls the strength of regularization. By using the *optimized couplings* optimized through the encoder to train the neural velocity field, it can gradually learn how to infer nearly straight trajectories, thereby enabling few-step and single-step sampling. Furthermore, our encoder-neural velocity field joint generation framework still satisfies the marginal preserving property (Theorem 3), with a detailed definition and proof provided in the Appendix A.

Theorem 3. *Assume X is rectifiable and Z is its viscous rectified flow. The marginal law of Z_t equals that of X_t at every time t, i.e., Law(Z_t) = Law(X_t), $\forall t \in [0,1]$.*

Noise optimization. In previous studies, noise optimization typically treats noise as a learnable parameter and performs small-scale iterative updates over multiple iterations using gradient information provided by the model, until it converges to an approximate optimal solution [6, 35]. While it can generate high-quality images, its limitations are also evident: it requires personalized optimization for each noise and relies on pre-trained models. In contrast, our method utilizes the reparameterization technique to optimize the noise through linear transformations directly. This approach allows for significant adjustments to the noise in a single optimization step, enabling rapid and efficient approximation to the optimal solution, significantly reducing the number of iterations and computational overhead, while also eliminating the dependence on pre-trained models.

Two-stage training. In the training process, we adopt a two-stage strategy based on empirical observations. In the first stage, we set $d(\cdot,\cdot)$ to the MSE loss, and once convergence is achieved, we proceed to the second stage. In this stage, we incorporate the LPIPS loss [32] for joint training on top of the first-stage setup, continuing until convergence is reached again. Notably, our method does not rely on distillation or adversarial training mechanisms.

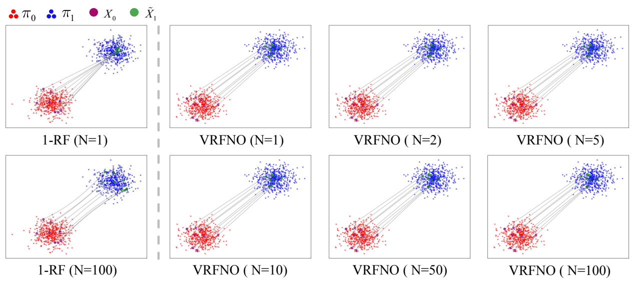

Figure 4. **Visualization of inference trajectories on synthetic data.** The first column on the left shows the inference trajectories of 1-RF with N steps, and the three columns on the right show the inference trajectories of VRFNO with N steps.

Sampling. After training the encoder and the neural velocity field, we can generate images using VRF ODE introduced in Eq.(4), the discrete sampling process is shown in Algorithm 2. For single-step generation, the HVT is set to 0 during sampling, while in few-step generation, the HVT is updated at each step. Additionally, we integrate the dataset into the sampler, meaning that the sampled noise is also reparameterized during the generation process. Since we introduced perturbations in the encoder, this will not affect the diversity of the generated images but can further enhance the quality of the generated images.

4. Experiment

We evaluate the performance of VRFNO in various scenarios, including synthetic data and real data. In Section 4.1, we compare the performance of RF and VRFNO by studying the inference trajectories between two 2D Gaussian distributions, which clearly demonstrates the effectiveness of our method. In Section 4.2, we extend the experiments to real image datasets, CIFAR-10 [12] and AFHQ [3], at different resolutions to validate the performance of VRFNO in one-step and few-step generation. Additionally, we conducted further analysis on VRFNO, including ablation studies, as well as quantitative evaluations of its trajectory straightness and inference efficiency.

Baselines and evaluation. We evaluated VRFNO and compared it with state-of-the-art diffusion models [10, 18, 24, 27, 28] and rectified flow models (RF [14], Constant Acceleration Flow (CAF) [21], TraFlow [29], Shortcut Model (SM) [7]) in single-step and few-step generation. Among them, RF serves as the baseline for our method, while CAF is the most similar approach to ours. Therefore, these two methods are the primary focus of our comparison. We use Fréchet Inception Distance (FID) and Kernel Inception Distance (KID) to assess the quality of generated images, and use Inception Score (IS) to measure diversity. Lower FID and KID indicate higher generation quality, while a higher IS suggests greater diversity in the generated images.

Table 1. Quantitative results on CIFAR-10.

Methods	NFE	IS ($\uparrow$)	FID ($\downarrow$)	KID($\times 10^{-3}$) ($\downarrow$)
Diffusion/Consistency Models				
VP ODE	1	1.20	451	-
(+distill)		(8.73)	16.23)	
sub-VP ODE	1	1.21	451	-
(+distill)		(8.80)	(14.32)	
DDIM distillation	1	8.36	9.36	-
Progressive	1	-	9.12	-
CT	1	8.49	8.71	-
EDM	5	-	37.75	-
Rectified Flow Models				
1-RF	1	1.13	379	428
2-RF	1	8.15	11.97	8.66
CAF	1	8.32	4.81	-
TraFlow	1	-	4.50	-
SM	1	-	4.93	-
VRFNO (Ours)	1	**9.59**	**4.50**	**2.73**
1-RF	5	7.12	34.81	32.26
2-RF	5	9.01	4.36	2.25
CAF	5	8.67	4.03	-
VRFNO (Ours)	5	**9.42**	**4.03**	**2.13**
1-RF	10	8.44	12.70	11.50
2-RF	10	9.13	3.83	1.63
CAF	10	9.12	3.77	-
VRFNO (Ours)	10	**9.51**	**3.36**	**1.31**

4.1. Synthetic Experiments

We demonstrate the trajectory straightening effect of VRFNO through 2D synthetic data experiments. For the neural velocity field, we use a simple multi-layer perceptron (MLP) architecture with three hidden layers, each containing 64 units. For the encoder, we adopt the encoder architecture from the VAE framework, consisting of two hidden layers, each with 16 units. Additionally, a fully connected layer is used to produce the mean and variance separately.

The visualization results are shown in Fig.4, where we compared VRFNO with the 1st generation RF (1-RF). Additionally, in Appendix B, we provide the inference trajectories of 1-RF and the 2nd generation RF (2-RF) for different numbers of steps. In the multi-step generation, the inference trajectory of VRFNO is closer to a straight line compared to 1-RF, which provides a guarantee for accurate single-step and few-step generation. The single-step and few-step inference trajectories of VRFNO are visually straight, indicating that the neural velocity field of VRFNO better converges to a constant velocity field, making its predictions of the trajectory direction more stable. In addition, the error between the endpoints of the single-step and the few-step inference trajectories and the endpoints of the multi-step inference trajectories is small, indicating that VRFNO is able

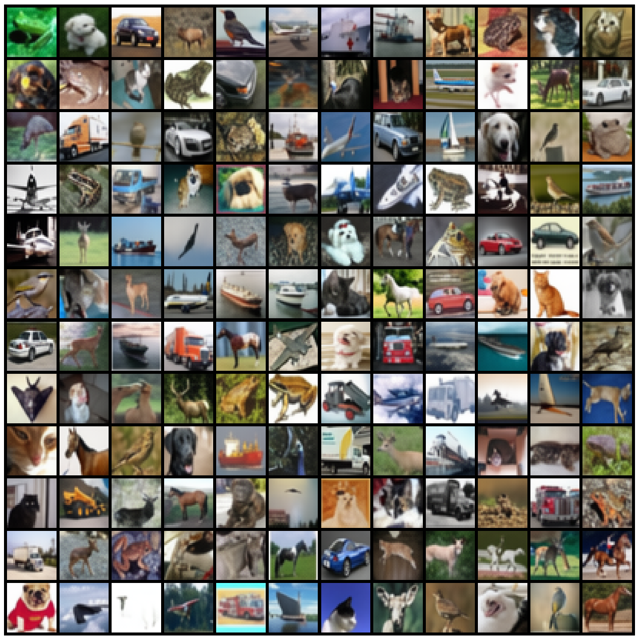
Figure 5. **Qualitative results on CIFAR-10.** Visualization of one-step generation of VRFNO

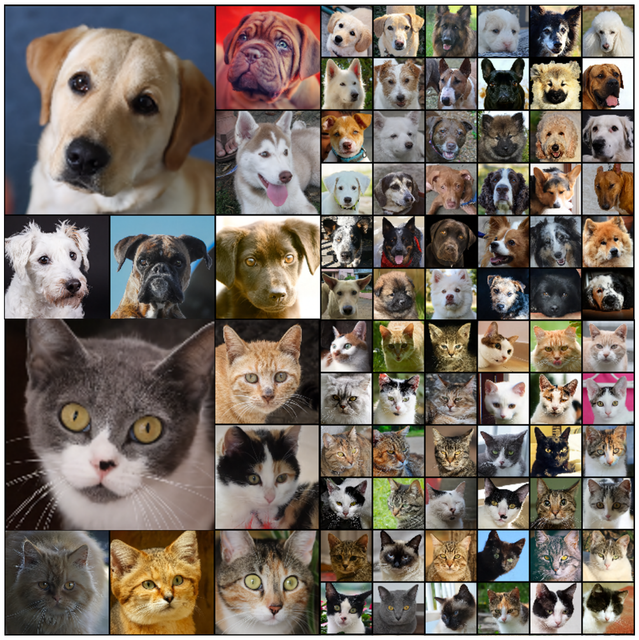
Figure 6. **Qualitative results on AFHQ.** Visualization of one-step generation of VRFNO at different resolutions

to accurately map samples from the initial distribution to the target distribution in both single-step and few-step generation. In contrast, when 1-RF is generated in a single step, the endpoint of the trajectory is usually concentrated at the mean point of the target distribution, while the samples near this point are usually meaningless in practical applications, as shown in Fig.II (a) in Appendix B.

4.2. Real-data Experiments

To further validate the effectiveness of our method, we evaluate VRFNO on real image datasets, namely CIFAR-10 with a resolution of 32×32 and AFHQ with different resolutions (including 64×64, 128×128, 256×256). For the neural velocity field, we adopt the UNet architecture from DDPM++ [27] to train on CIFAR-10 and the NCSNv2 [26] to train on AFHQ; for the encoder, we refer to the encoder architecture from [8], and its details can be found in Appendix E. Specifically, the encoder requires very few parameters, taking the train of CIFAR-10 as an example, the number of parameters in the encoder is less than $1/20$ of those in the neural velocity field.

Tab.1 presents the quantitative evaluation of VRFNO on CIFAR-10. Compared to other diffusion models, our approach achieves superior performance in single-step generation, with FID = 4.53, KID = 2.73, and IS = 10.59. Compared to previous rectified flow models, VRFNO attains state-of-the-art performance in both single-step and few-step generation (our training does not involve distillation). Fig.5 illustrates the qualitative evaluation of VRFNO in single-step generation on CIFAR-10. Further comparative qualitative evaluations between VRFNO and RF in N-step generation can be found in Appendix B. Tab.2 provides the quantitative evaluation of VRFNO's generated images on the AFHQ cat and dog datasets, focusing on the image quality of single-step generation at different resolutions. Our method is designed as a better alternative to Reflow, so the comparison primarily highlights the performance difference with 2-RF. From the tab.2, it is evident that VRFNO outperforms 2-RF in image generation quality across different resolutions.The qualitative evaluation of AFHQ at different resolutions can be found in Fig.6.

Table 2. Quantitative results on AFHQ with different resolutions.

Methods	Datasets	NFE	Resolution	FID(↓)
2-RF (+distill)	AFHQ -CAT	1	64×64	181.93(31.57)
			128×128	172.66(29.81)
			256×256	171.84(29.33)
	AFHQ -DOG	1	64×64	200.77(33.64)
			128×128	192.30(32.70)
			256×256	189.82(30.27)
VRFNO (Ours)	AFHQ -CAT	1	64×64	28.69
			128×128	27.56
			256×256	27.04
	AFHQ -DOG	1	64×64	44.64
			128×128	27.21
			256×256	27.37

4.3. Further Analysis

Ablation Study. We conduct an ablation study to evaluate the effectiveness of each component in our framework

Table 3. Ablation study of HVT and noise optimization.

Config	NFE	HVT	Noise optimization	FID(↓)
A	1	✗	✗	379
B	1	✓	✗	332
C	1	✗	✓	4.72
D	1	✓	✓	4.53
E	5	✗	✗	34.81
F	5	✓	✗	32.50
G	5	✗	✓	4.28
H	5	✓	✓	4.03
I	10	✗	✗	12.70
J	10	✓	✗	9.34
K	10	✗	✓	4.75
L	10	✓	✓	3.40

Table 4. Comparison of flow straightness.

Dataset	2-RF	3-RF	CAF	VRFNO(Ours)
2D	0.067	0.053	0.058	0.054
CIFAR-10	0.058	0.056	0.035	0.026

Table 5. Comparison of the inference time (sec) in N-step.

NFE	RF	CAF	VRFNO(Ours)
1	0.172	0.181	0.305
10	1.404	1.415	1.646

for single-step and few-step generation. Specifically, we investigate the improvements brought to RF by (1) the introduction of the HVT and (2) the noise optimization operation. The configurations and results are shown in Tab.3. The two components have both contributed to the improvement of model performance. Regarding the introduction of the HVT, even with one-step sampling, the model performance improved, indicating that it facilitates trajectory rectification during the training process. As for the noise optimization operation, it results in a significant performance boost, demonstrating the effectiveness of training the neural velocity field with our defined *optimized coupling*.

Flow Straightness. To evaluate the straightness of the inference trajectories, we introduce the Normalized Flow Straightness Score (NFSS) [14, 15]. A smaller NFSS indicates that the inference trajectory is closer to a straight line. We compare VRFNO with RF and CAF on synthetic datasets and CIFAR-10, with the experimental results shown in Tab.4. Our method generates inference trajectories that are straighter compared to the other methods.

Time Efficiency. We also compare the inference time of our method with RF and CAF. Tab.5 presents the time required to generate 512 images in both one-step and ten-step settings. Our method incurs a slightly higher time cost than RF due to the additional encoder. In contrast, CAF consists of two neural velocity fields, it necessitates an additional computation (its total NFE equals $N + 1$), resulting in significantly higher time consumption.

5. Related Work

The inference process of DMs can be viewed as an iterative solution of ODE [25] or stochastic differential equations [2, 9], and ODE-based methods [5, 31] are more effective in few-step sampling due to their determinism. A widely used first-order ODE solver is the Euler solver, but its large local truncation error necessitates more iteration steps for high-quality image generation. To reduce iteration steps, higher order ODE solvers such as DEIS [31] and DPM solvers [16] exploit the semilinear structure of diffusion ODEs, deriving exact solutions for high-quality image generation with fewer steps. DPM-solver++ [17] enhances DPM-solver by improving stability in higher-order solvers. UniPC [33] builds a unified predictor-corrector framework for DPM solver, enhancing the quality of sampling in a few steps. DC-solver [34] uses dynamic compensation to correct misalignment in UniPC, thus improving image quality.

Another mainstream method for accelerated sampling is distillation. Progressive distillation [24] iteratively halves the steps of a DDIM until it enables image generation in just 4 steps. Diff-Instruct [20] accelerates sampling by minimizing Integra Kullback-Leibler divergence to merge knowledge from multiple time steps of a pre-trained model. Consistency Model (CM) [28] enables direct mapping from noise to data by minimizing the differences between the final states when adjacent ODE trajectory points are used as inputs. Latent CM [19] treats the guided reverse diffusion process as solving an augmented PF ODE, directly predicting the ODE solution in the latent space, enabling fast and high-fidelity sampling. Consistency Trajectory Model [11] further optimizes CM by allowing the model to learn mappings between arbitrary initial and final times along the ODE trajectory during the diffusion process.

6. Conclusion

This paper proposes a novel method VRFNO, which is a joint training framework combining an encoder (for noise optimization) and a neural velocity field (for velocity prediction). VRFNO utilizes the encoder to optimize noises to transform randomly matched noises and images into *optimized couplings*, which are then used to train the neural velocity field for precise prediction of straight inference trajectories. To further straighten the inference trajectories during training, we introduce the HVT into the neural velocity field to enhance its prediction accuracy. Extensive experiments conducted on both synthetic and real world datasets demonstrate the effectiveness and scalability of VRFNO, and the results show that it achieves state-of-the-art performance across multiple datasets and varying resolutions.

Acknowledgments

This work was supported in part by the National Science Fund of China (No.62276135, U24A20330 and 62361166670).

References

[1] Jonas Adler and Sebastian Lunz. Banach wasserstein gan. *Advances in neural information processing systems*, 31, 2018. 1

[2] Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. *arXiv preprint arXiv:2201.06503*, 2022. 8

[3] 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*, pages 8188–8197, 2020. 6

[4] Tianyi Chu, Wei Xing, Jiafu Chen, Zhizhong Wang, Jiakai Sun, Lei Zhao, Haibo Chen, and Huaizhong Lin. Attack deterministic conditional image generative models for diverse and controllable generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1362–1370, 2024. 1

[5] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Genie: Higher-order denoising diffusion solvers. *Advances in Neural Information Processing Systems*, 35:30150–30166, 2022. 8

[6] Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. *Advances in Neural Information Processing Systems*, 37:125487–125519, 2025. 5

[7] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024. 6

[8] Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017. 5, 7

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 8

[10] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 6

[11] 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. 8

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[13] Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022. 2

[14] 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*, 2023. 1, 3, 6, 8

[15] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 8

[16] 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. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022. 8

[17] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. 8

[18] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021. 6

[19] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023. 8

[20] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 8

[21] Dogyun Park, Sojin Lee, Sihyeon Kim, Taehoon Lee, Youngjoon Hong, and Hyunwoo J Kim. Constant acceleration flow. *arXiv preprint arXiv:2411.00322*, 2024. 6

[22] Saptarshi Roy, Vansh Bansal, Purnamrita Sarkar, and Alessandro Rinaldo. 2-rectifications are enough for straight flows: A theoretical insight into wasserstein convergence. *arXiv e-prints*, pages arXiv–2410, 2024. 1

[23] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022. 1

[24] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 6, 8

[25] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1, 8

[26] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020. 7

[27] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 6, 7

[28] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 6, 8

[29] Zhangkai Wu, Xuhui Fan, Hongyu Wu, and Longbing Cao. Traflow: Trajectory distillation on pre-trained rectified flow, 2025. 6

[30] YoungJoon Yoo and Jongwon Choi. Topic-vq-vae: Leveraging latent codebooks for flexible topic-guided document generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 19422–19430, 2024. 1

[31] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022. 8

[32] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5

[33] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 8

[34] Wenliang Zhao, Haolin Wang, Jie Zhou, and Jiwen Lu. Dc-solver: Improving predictor-corrector diffusion sampler via dynamic compensation. *arXiv preprint arXiv:2409.03755*, 2024. 8

[35] Zikai Zhou, Shitong Shao, Lichen Bai, Zhiqiang Xu, Bo Han, and Zeke Xie. Golden noise for diffusion models: A learning framework. *arXiv preprint arXiv:2411.09502*, 2024. 5