On the comparison of several generative models

Zhijun Wang
Department of Computer Science
University of Toronto

Zewen Shen
Department of Computer Science
University of Toronto

Abstract

In this paper, we compare the image generation quality of the rectified flow algorithm, the score-based SDE diffusion model, and the variational autoencoder algorithm.

1 Introduction

In recent years, the development of generative models for image generation has received significant attention from the research community. Generative models are machine learning models that can generate new data similar to the data on which they were trained. Among the many different types of generative models, three popular approaches are the rectified flow algorithm, the score-based SDE diffusion model, and the variational autoencoder algorithm. These methods have been applied to various tasks, including image generation, inpainting, and super-resolution. In this paper, we compare the image generation quality of these three methods. Specifically, we evaluate the quality of the generated images based on two commonly used metrics: the Fréchet Inception Distance (FID) and the Inception Score. We also consider the cost of generating the samples using each of these methods.

2 Related work

Flow-based generative models are a popular class of generative models capable of generating high-quality samples from complex probability distributions. They are based on the idea of transforming a simple base distribution into a complex target distribution through a series of invertible transformations. One of the most popular flow-based generative models is the Real-valued Non-Volume Preserving (RealNVP) model introduced by Dinh et al. [1]. The RealNVP model is based on a sequence of coupling layers that transform the input distribution in a way that preserves its volume while increasing its complexity. Another popular flow-based generative model is the Glow model proposed by Kingma and Dhariwal [4]. The Glow model is based on a series of invertible 1x1 convolutions, which are used to transform the input distribution into a target distribution with a high degree of complexity. However, these algorithms have restrictive expressibility, as the network has to be invertible. To overcome this limitation, several more flexible models have been proposed, such as FFJORD [2], SDE diffusion models [8] and rectified flows [5]. These models have demonstrated their ability to generate high-quality samples from complex distributions, and the latter two models are the focus of our project.

Variational autoencoder [3] is another important class of method for generative modeling, which is first introduced by Kingma and Welling in 2013 as a probabilistic approach to unsupervised learning. They proposed a novel way to train a neural network to perform both dimensionality reduction and generate new samples from the learned distribution. Recently, several hybrid models have been proposed that combine the advantages of VAEs with flow-based models. These models include the Variational Flow model in [6] and the VAE Householder flow model in [9].

Preprint. Under review.

3 Algorithms

3.1 Rectified Flow

The goal of continuous normalizing flow-based models is to find a drift function $v : \mathbb{R}^N \times [0, 1]$, such that the ordinary differential equation

$$\mathrm{d}x = v(x, t)\,\mathrm{d}t, \quad t \in [0, 1] \tag{1}$$

induces a flow from a data distribution p_0 to another data distribution p_1. In our case, we pick p_0 to be a Gaussian distribution, and p_1 to be our desired distribution. Given a pair of dataset $(X_0, X_1) \sim (p_0, p_1)$, we train a neural network $v_\theta(x, t)$ that approximates $v(x, t)$ using the following objective function:

$$\operatorname*{argmin}_\theta \mathbb{E}[\|X_1 - X_0 - v_\theta(tX_1 + (1 - t)X_0, t)\|_2^2], \tag{2}$$

such that the continuous normalizing flow follows the linear path pointing from X_0 to X_1 as much as possible. The authors [5] prove that the ODE trajectory $x(t)$ and the interpolation $tX_1 + (1 - t)X_0$ has the same marginal distribution, for any $t \in [0, 1]$, from which it follows that the induced flow is indeed a mapping from p_0 to p_1. To generate samples from p_1, one sets the initial value of the ODE (1) to be samples from the Gaussian distribution, then solves the ODE numerically. However, if the function v_θ is non-smooth, it requires extensive evaluation of the neural network v_θ to solve the ODE accurately, which makes the sample generation costly. The authors propose a so-called "reflow" algorithm that alleviates the cost, which we describe as follows. After training the network v_θ, we sample $X_0 \in p_0$, and compute the solution to the ODE at time $t = 1$, i.e., $x(1)$. This gives us a new pair of dataset $(X_0, x(1))$, using which we can train a new rectified flow $\widetilde{v}_\theta$. One can show that the ODE trajectory of the resulting ODE has the same marginal distribution as the previous ODE. Additionally, the function $\widetilde{v}_\theta$ becomes smoother than v_θ, and fewer evaluation of $\widetilde{v}_\theta$ is required to solve the ODE to a desired accuracy. Note that this reflow process will not increase the quality of the generated sample. Since the goal of our project is to study the quality of the generated images, the reflow algorithm is not implemented.

3.2 Score-based generative modeling through stochastic differential equations

A diffusion process is described by a stochastic differential equation (SDE) in the form

$$\mathrm{d}x = f(x, t)\,\mathrm{d}t + g(t)\,\mathrm{d}w, \quad t \in [0, T], \tag{3}$$

where $f(x, t) : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}$ denotes the drift coefficient, $g(t) : \mathbb{R} \to \mathbb{R}$ denotes the diffusion coefficient, and w denotes the standard Brownian motion. Additionally, the reverse of the diffusion process above is also a diffusion process, which is described by the following reverse-time SDE:

$$\mathrm{d}x = [f(x, t) - g(t)^2 \nabla_x \log p_t(x)]\,\mathrm{d}t + g(t)\,\mathrm{d}\bar{w}, \quad t \in [0, T], \tag{4}$$

where $\bar{w}$ is a standard Wiener process when time flows backward from T to 0. For simplicity, we choose $f(x, t) = 0$, $g(t) = \sigma^t$ for some $\sigma > 0$, and set the time domain to be the interval $[0, 1]$. Given a data distribution p_0 and an initial value $x(0) \sim p_0$, the authors [8] show that $x(t)|x(0) \sim \mathcal{N}(x(0), \frac{1}{2\log\sigma}(\sigma^{2t} - 1)I)$. Furthermore, when σ is large, the marginalization of $x(1)|x(0)$ with respect to $x(0)$ follows the distribution $\int p_0(\mathbf{y})\mathcal{N}\left(\mathbf{y}, \frac{1}{2\log\sigma}(\sigma^2 - 1)\mathbf{I}\right)d\mathbf{y} \approx \mathcal{N}\left(\mathbf{0}, \frac{1}{2\log\sigma}(\sigma^2 - 1)\mathbf{I}\right)$. In other words, the SDE (3) maps p_0 to a Gaussian distribution, which is easy to sample from. Therefore, provided that the so-called score function $\nabla_x \log p_t(x)$ is available, one can generate samples from the original data distribution p_0 by solving the reverse-time SDE (4) with an initial data sampled from the Gaussian distribution (see TODO for numerical algorithms for solving SDEs). We train a neural network $s_\theta(x, t)$ that approximates the score function using the following denoising score matching objective:

$$\min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{\mathbf{x}(0)\sim p_0(\mathbf{x})}\mathbf{E}_{\mathbf{x}(t)\sim p_{0t}(\mathbf{x}(t)|\mathbf{x}(0))}[\|s_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))\|_2^2]], \tag{5}$$

where $\mathcal{U}(0, T)$ is a uniform distribution over $[0, T]$, $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ denotes the transition probability from $\mathbf{x}(0)$ to $\mathbf{x}(t)$, and $\lambda(t) \in \mathbb{R}_{>0}$ denotes a positive weighting function. Note that $\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ can be easily computed, as $x(t)|x(0) \sim \mathcal{N}(x(0), \frac{1}{2\log\sigma}(\sigma^{2t} - 1)I)$.

3.3 Variational Autoencoder

A variational autoencoder [3] aims to acquire a low-dimensional representation of high-dimensional data by encoding it into a probability distribution in latent space. This representation can then be utilized to generate new samples similar to the original data. The VAE algorithm involves two neural networks - an encoder $P_\theta : \mathbb{R}^N \to (\mathbb{R}^d \times \mathbb{R}^d)$, where N denotes the dimension of the images and d represents the latent dimensions, and a decoder $Q_\omega : \mathbb{R}^d \to \mathbb{R}^N$. The encoder maps an image to the mean vector and variance vector of a Gaussian distribution, and it is desired that the Kullback-Leibler (KL) divergence between this Gaussian distribution and $\mathcal{N}(0, I)$ is as small as possible. The decoder maps a sample from the Gaussian distribution output by the encoder to an image sample. The reconstruction error between this image sample and the input to the encoder is minimized. Thus, the networks are trained to minimize the loss function given by

$$L(\theta, \omega; x) = \mathbb{E}_x[\|Q_\omega(z) - x\|_2 + KL(\mathcal{N}(P_\theta(x)), \mathcal{N}(0, I))].$$

After training, samples from the decoder can be generated by sampling from $\mathcal{N}(0, I)$, then feeding the sample as input to the decoder.

4 Experiment

4.1 Network architectures and datasets

In all three models, we use the UNet architecture [7] with a number of channels $(128, 256, 512, 1024)$ to parametrize the drift function v_θ and the score function s_θ. In particular, we incorporate the time information via Fourier random features, i.e., we sample a fixed vector $w \in \mathcal{N}(0, I)$, and define the time feature for t as the concatenation of $\sin(2\pi wt)$ and $\cos(2\pi wt)$. In the VAE model, the encoder is a CNN network with four convolutional layers, whose number of channels is $(128, 256, 512, 1024)$, each followed by a GroupNorm layer and a Swish activation function. We linearly map the final CNN feature to the vectors of mean and variance with the latent dimension equal to 128. The decoder network uses four transposed convolutional layers and a Swish activation function to take a latent vector z and produce an output image with higher resolution. The number of channels for each layer has been decreased, and the output reaches the original color channel. Finally, a fully connected linear layer is applied to map the latent vector to the shape of the input data.

We use the fashion MNIST dataset and the CIFAR 10 dataset as our training dataset. We implemented the rectified flow model and the variational autoencoder model from scratch, and implemented the SDE model using `https://colab.research.google.com/drive/120kYYBOVa1i0TD85RjlEkFjaWDxSFUx3?usp=sharing` as the starter code.

4.2 Optimizers and Hyperparameters

We used the Adams optimizer for all models and set the learning rate to $1e-4$ and the batch size to 32 for the Rectified Flow and SDE models. The SDE models also included hyperparameters such as $sigma = 25.0$ and $n\_epochs = 100$, while the rectified flow model used $signal\_to\_noise\_ratio = 0.16$ and $num\_steps = 500$. We determined that the learning rate was the most critical hyperparameter and would adapt it based on the first few runs' outputs. For the VAE model, we recommended a learning rate of either $5e-4$ or $3e-4$. We found that the learning rate and latent dimension were critical variables during experimentation. We observed that a latent size of 64 resulted in a relatively high loss of approximately 4.7 while increasing the latent size to 128 reduced the loss to 3.848.

4.3 Experiment Progress

Based on the model architecture we introduced in the method and UNet section, we conducted our experiment using it to enhance the model's learning ability. Our results were ideal. Specifically, rectified Flow and SDE models outweigh the VAE in the quality of images.

4.4 Hardware and Schedule

To train our rectified flow model and SDE models, we used a Tesla T4 GPU and trained each model for 1 hour using the MNIST dataset and 2 hours using the CIFAR-10 dataset, which resulted in high-quality image results. For our VAE model, we utilized a NVIDIA GeForce RTX 3070 Laptop GPU and trained it for 2 hours.

5 Results

In the table below, we report the Inception score and FID score for the generated images. We also report the generated image samples in the appendix. These two metrics are commonly used in rating the generative model.

Table 1
Test Result for Different Models

	Inception Score	FID Score*	Training Time
Rectified Flow	1.73 ± 0.05	42.50 ± 0.67	$2h$
Score Based Diffusion Model	1.84 ± 0.07	48.17 ± 0.43	$2h$
VAE model*	1.64 ± 0.04	61.27 ± 0.54	$2h$

Note1: the FID score is calculated by getting 1024 random samples and taking the average of the 10 biggest values.
Note2: the VAE model is fine tuned here. But it's not necessary in partical.
Based on the results, we observe that the Rectified Flow model yields the lowest FID score, indicating that it generates images that are more closely related to real images compared to the other models. Additionally, it exhibits the second highest Inception Score, suggesting that the generated images are diverse in nature. On the other hand, the Score Based Diffusion Model achieves the highest Inception Score, but it is associated with a higher FID Score compared to the rectified flow. Lastly, the VAE model has the lowest Inception Score and the highest FID Score, suggesting that the quality of the generated samples is comparatively low.

Despite the inferior quality of the samples generated by the VAE model, it is the least expensive method for generating samples. In contrast, the rectified flow and SDE methods produce much better quality images. However, the cost of generating samples using these methods is two orders of magnitude more expensive than that of the VAE. Additionally, in comparison to the SDE method, the image quality of the rectified flow method is inferior. However, the use of the reflow algorithm significantly accelerates the sample generation process.

In conclusion, the Rectified Flow model appears to be the best option for generating samples with high quality and diversity. However, this method comes with a significantly higher cost compared to the VAE. Ultimately, the choice of the method depends on the specific requirements of the application, such as the required quality, diversity, and cost.

References

[1] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[2] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

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

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

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[8] 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.

[9] Jakub M Tomczak and Max Welling. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

Contribution

Zhijun Wang: Implementation of VAE. Experiments. Writing.
Zewen Shen: Implementation of rectified flow, score-based SDE models. Experiments. Writing.

Appendix: Generated samples

Figure 1
Rectified flow, Fashion MNIST

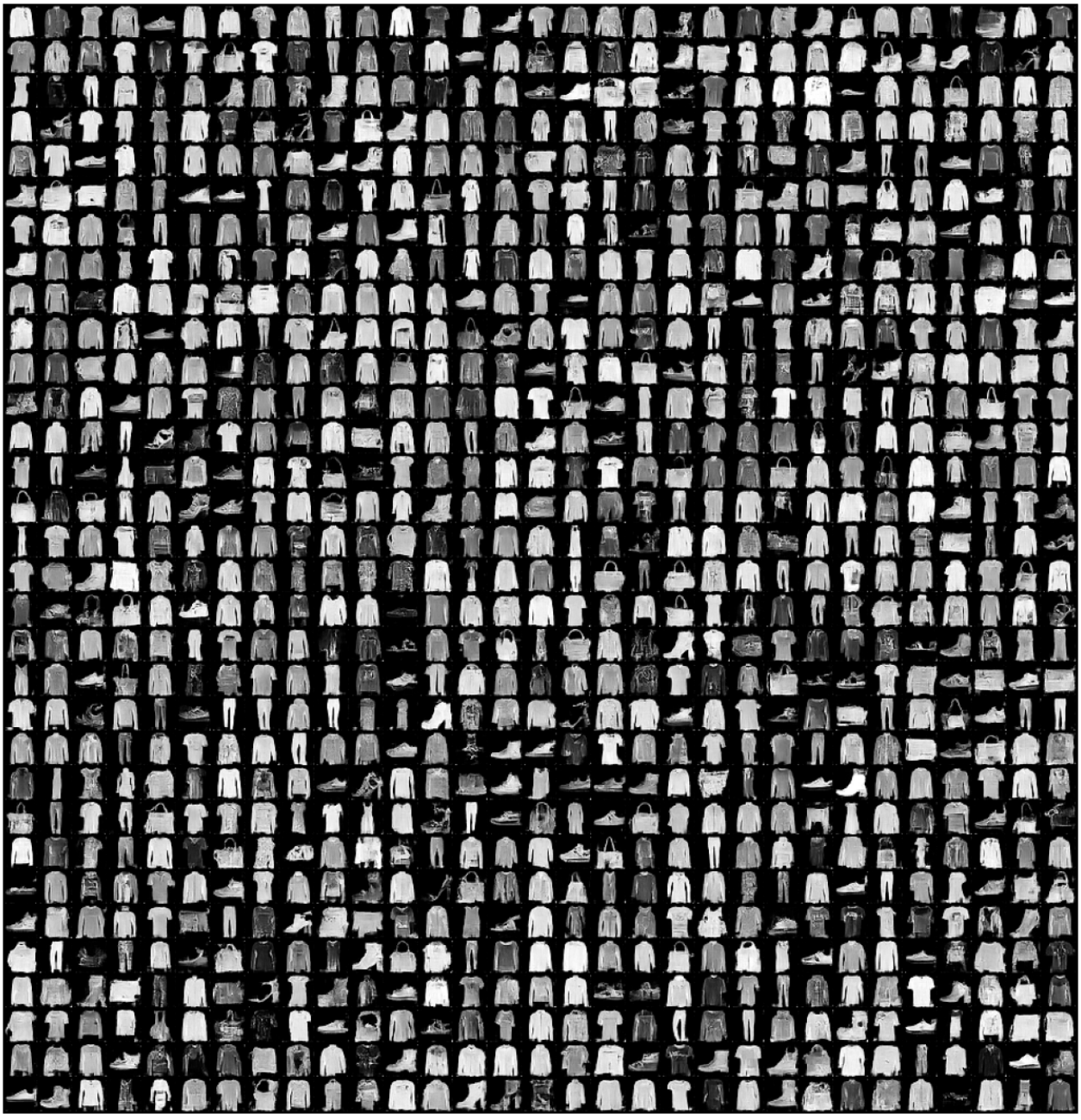

Figure 2
SDE, Fashion MNIST

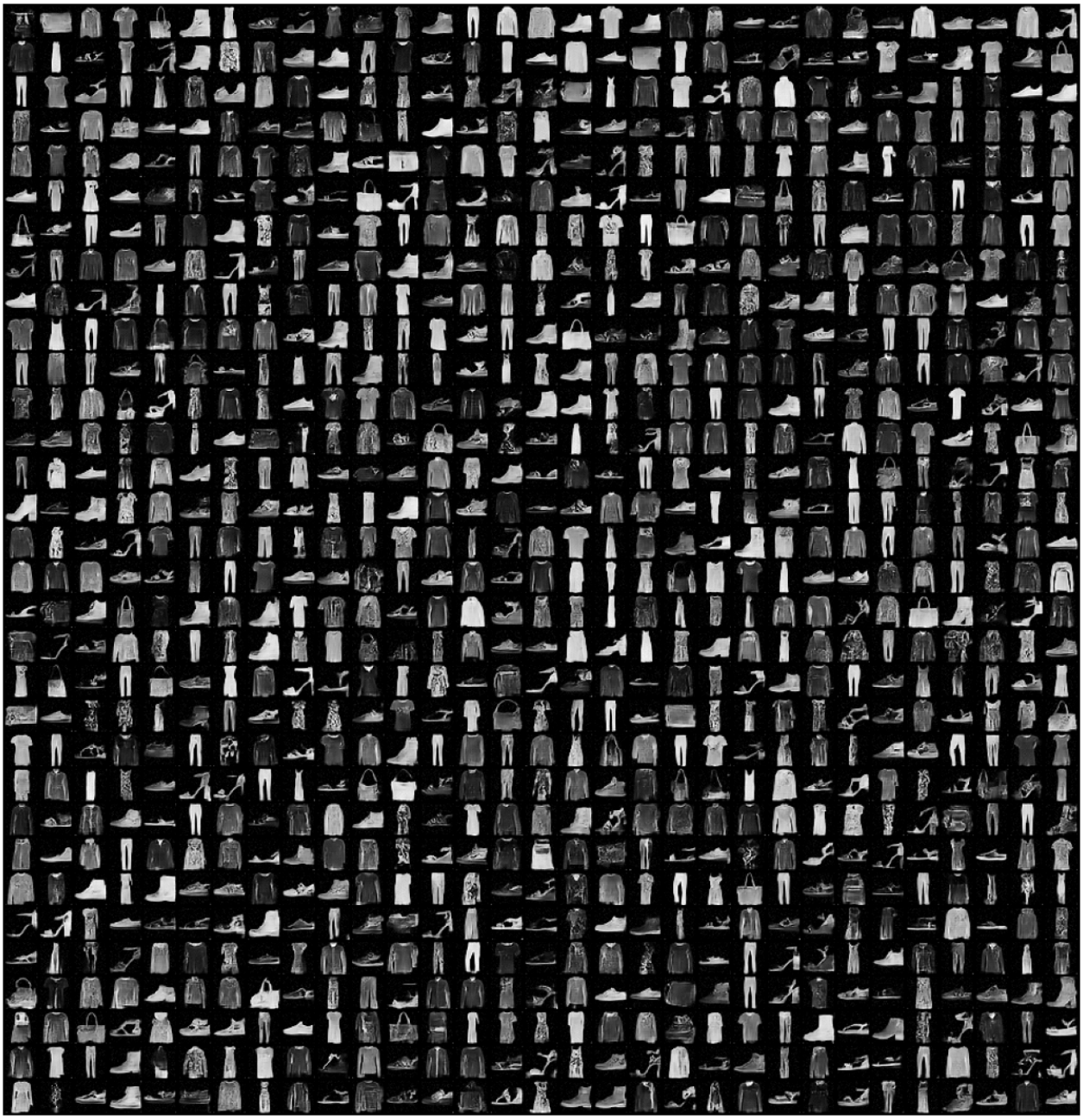

Figure 3
VAE, Fashion MNIST

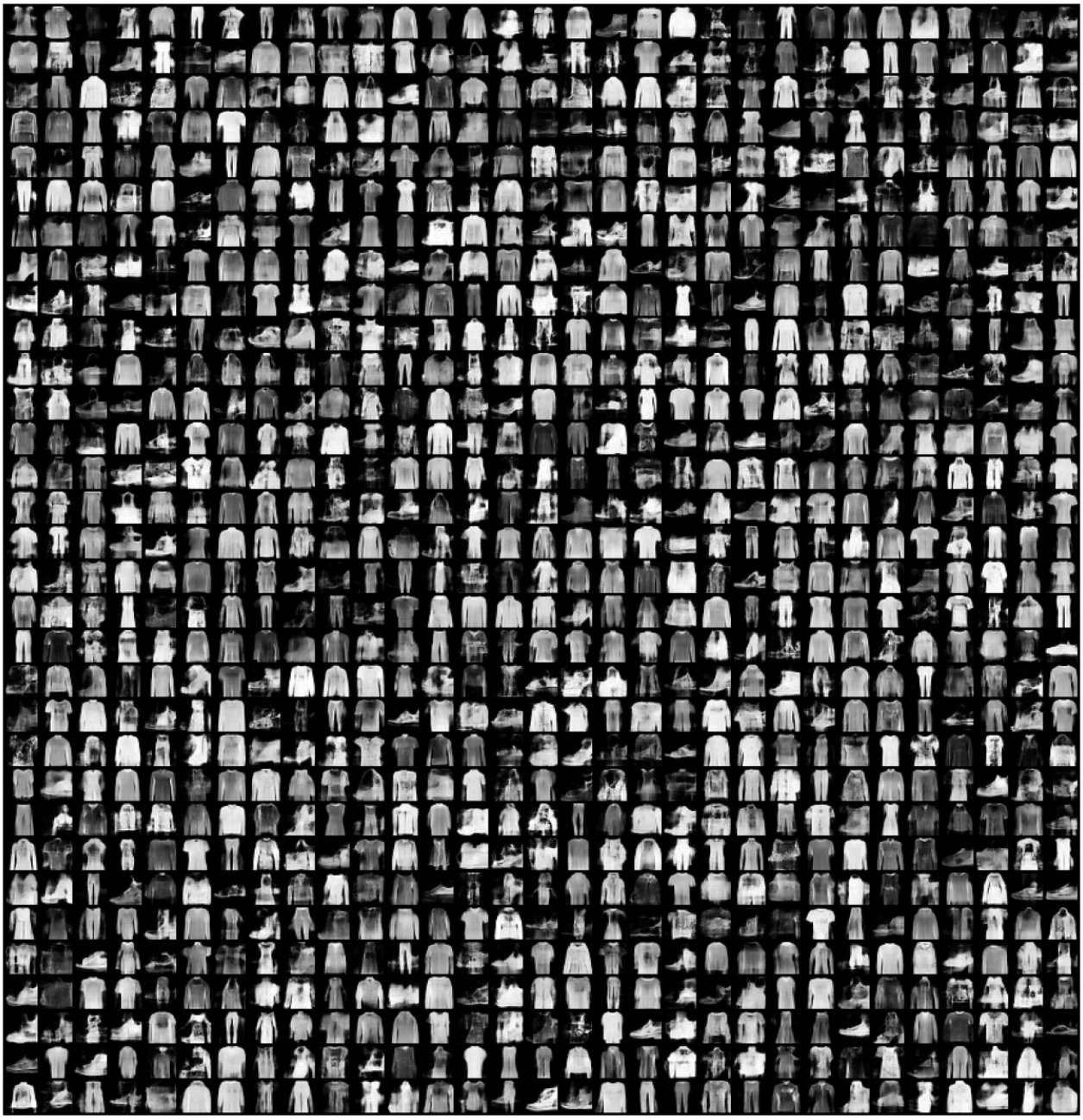

Figure 4
Rectified flow, CIFAR 10

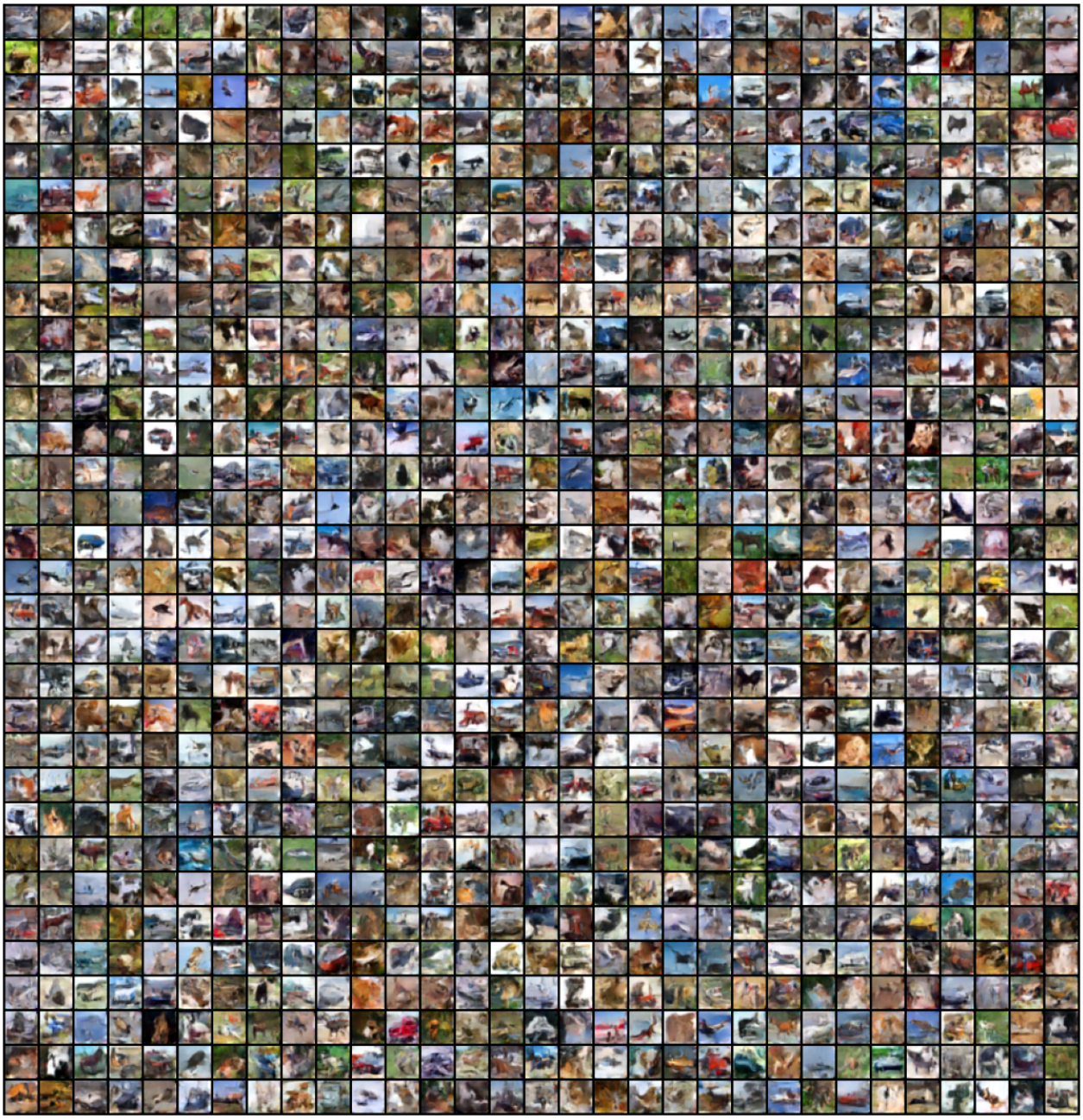

Figure 5
SDE, CIFAR 10

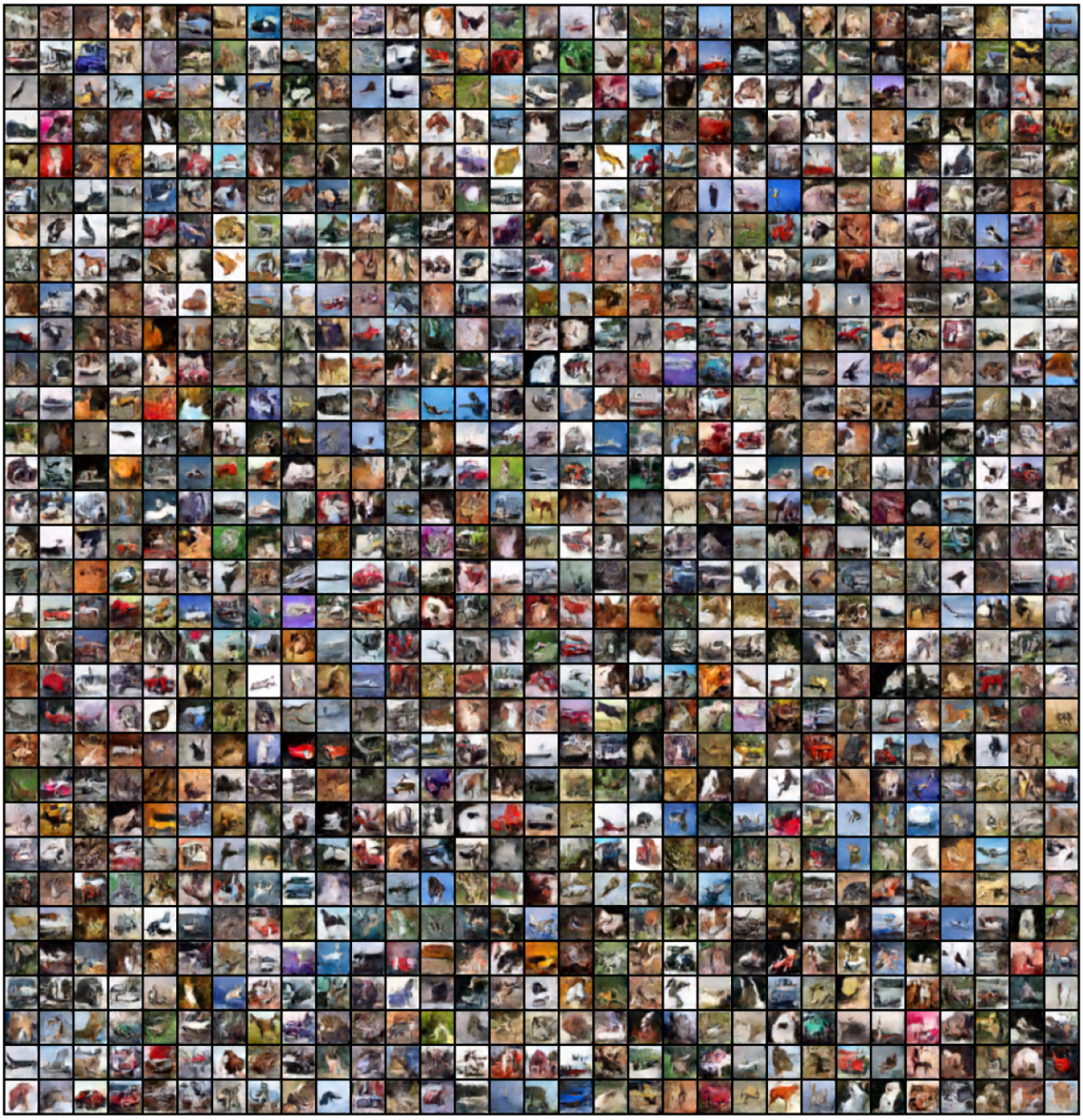

Figure 6
VAE, CIFAR 10