Adaptive Noise Score Networks

Haozhen Shen
University of Toronto
haozhen.shen@mail.utoronto.ca

Abstract

Score-based models have made impressive progress in the past couple of years, and state-of-the-art results are demonstrated in numerous applications. One of the main challenges is that it is difficult to estimate the score function in low-density regions. Previous methods approach this by perturbing the score with multiple levels of noise scales, and later generalizing to the continuous case through stochastic differential equations resulting in an infinite amount of noise. In both cases, they assume the variance of noise has the form of some constant times the identity matrix, and noise is chosen randomly when perturbing. However, this diagonal covariance assumption and random selection of noise have several high dimensional limitations. We outline these limitations and extend this noise perturbing process to be tailored for different density regions. Here, we propose the *Adaptive Noise Score Networks* (ANSN), a novel approach that is inspired by Markov Chain Monte Carlo (MCMC) techniques and leverages variance information from previously seen data to determine what scale of noise perturbation to be used during training, resulting in fewer network evaluations and faster convergence. ANCSN produces samples comparable to NCSN++ and GANs on the MNIST and CIFAR-10 datasets.

1 Introduction

The goal of generative modeling is to learn a data distribution and be able to generate samples from it. However, to cover the distribution faithfully and have high-quality samples generated is not an easy task. Recent approaches in this field can be mainly separated into two categories, namely likelihood-based models [4, 13, 9, 8], and implicit generative models [5]. Where the former is trained through directly maximizing likelihood or some surrogate of the likelihood, and the latter usually involves adversarial training. However, in both cases, there are limitations in their modeling nature. Likelihood-based models often make strong model assumptions before building a probabilistic model, this significantly limits the expressiveness of such models at the very beginning. Typical examples include variational autoencoder [9], autoregressive models [8], and flow-based models [4]. On the other side, the prominent example of implicit generative models is Generative Adversarial networks (GANs) [5] which suffer from unstable training and usually have an intractable generation process.

In this paper, we focus on score-based generative models which instead of directly learning the data distribution we model the stein score [11] of the logarithmic data density, and later sampling can be solely done using this estimated score. In prior work, Song et al. [13] show how this type of model can achieve state-of-the-art results in terms of sample quality and distribution coverage by parameterizing the score function as a Noise Conditioned Score Network (NCSN). The reason to condition on noise is that directly modeling the score leads to inaccurate score estimates in low-density regions and perturbing the data with different levels of noise eases the model learning in such regions. During training, they perturb the score with a noise σ_i randomly selected from a predetermined set of noise scales $\{\sigma_0, ..., \sigma_n\}$ assuming the noise follows $N(0, \sigma_i \mathbf{I})$.

Preprint. Under review.

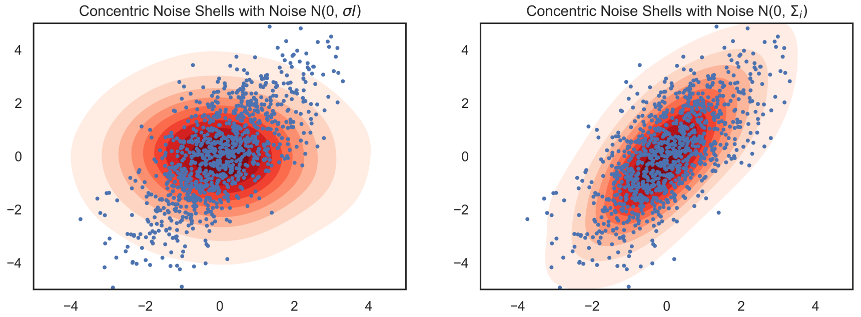

Figure 1: **Left**: Perturbing with a fixed diagonal covariance noise $N(0, \sigma I)$. While gradient does not have diagonal covariance. **Right**:Perturbing with a adapted covariance noise $N(0, \Sigma)$.

Albeit the significant improvement in performance, randomly selecting a noise scale and assuming it is normal and has diagonal covariance has several limitations. When modeling high dimensional data, a high dimensional Gaussian noise is indistinguishable from a sphere with radius σ_i. In other words, more like a soap bubble instead of a ball. Therefore, perturbing with a fixed set of noise scales is analogous to stacking concentric noise shells around each estimated score [3], interpolating the gradient space. This provides another interpretation of why multiple levels of noise scales result in a major improvement. However, in high dimensions, a major limitation of this is that the actual gradient might have a very dynamic variance structure. For example, when some dimensions have low variance and some high, a fixed noise scale with constant variance can not interpolate well the high variance dimensions and in the low variance dimensions, this perturbation has no use to our training process but simply adding noise. Another downfall is that when modeling high dimensional data such as image data these noise shells can be extremely thin, and we might fail to interpolate the gradients among the gaps. In a seminal work, Song et al. [16] generalizes noise scale into the continuous case by describing this perturbating process with a stochastic differential equation (SDE) and which can later be converted to maximum likelihood training. However, this still does not address the potential dynamical structure of the gradient variance at different density regions.

Here we propose the Adaptive Noise Score Networks (ANCSN), a new approach that is inspired by Markov Chain Monte Carlo techniques [6, 7, 1], and leverages variance information from previously seen data to determine what scale of noise perturbation to be used at different density regions during training. We accumulate variance information of previously seen data to provide an empirical estimate of score variance and stochastically update it, we then perform a range of perturbations accordingly. We can consider this model as a score based model with an adaptive noise perturbation process. This approach has several key advantages:

Generalization: With this adaptive scheme, our model is capable of accurately capturing the gradient flows at density region with dynamic variance, obtaining an even more accurate score estimate, and generating even more realistic samples.

Convergence: Since we have tailored noise at different density regions, we can utilize carefully designed covariance of noise to generalize faster at different density regions, therefore resulting in faster convergence.

Avoids Mode Collapse: Previously where we have noise perturbation with constant variance, this might lead to mode collapse since a fixed variance might just smooth out a closely nearby mode that is small enough in some specific dimension. But now we can utilize carefully designed noises to avoid such cases.

2 Background

Here we review score-based generative models, (see [13] for details). Suppose our dataset consists of i.i.d. samples $\{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^N$ from an unknown data distribution $p_{\text{data}}(\mathbf{x})$. The *score* (Stein score) of a probability density $p(\mathbf{x})$ is defined as $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. The *Noise conditioned Score Network*

$\mathbf{s}_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^D$ is a neural network parameterized by $\boldsymbol{\theta}$, which will be trained to approximate the score of $p_{\text{data}}(\mathbf{x})$. The ultimate goal here is to faithfully model $p_{\text{data}}(\mathbf{x})$ and be able to generate samples from it.

2.1 Score Matching

An intuitive formalization to perform score matching is to minimize the following objective,

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}}[||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})||_2^2] \tag{2.1}$$

We can approach this by training a noised conditioned score network $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$ to estimate $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ and obtain samples without modeling $p_{\text{data}}(\mathbf{x})$. Later it can be shown that the objective is equivalent to the following up to a constant

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\operatorname{tr}(\nabla_{\mathbf{x}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})) + \frac{1}{2}||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})||_2^2\right], \tag{2.2}$$

where $\nabla_{\mathbf{x}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$ denotes the Jacobian of $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$.However, there are still computational challenges. When minimizing this objective, the computation of $\operatorname{tr}(\nabla_{\mathbf{x}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}))$ requires $\mathcal{O}(n)$ backpropagations to calculate and is therefore not scalable in high dimensions. Luckily there are two methods to circumvent computation of the trace.

Denoising score matching (DSM) Denoising score matching [17] is score matching technique that completely circumvents $\operatorname{tr}(\nabla_{\mathbf{x}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}))$. By first perturbing the data point $\mathbf{x}$ with a pre-specified noise distribution $q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$ (usually normal) Vincent et al. was able to show that estimating the score of the perturbed data distribution $q_\sigma(\tilde{\mathbf{x}}) \triangleq \int q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})p_{\text{data}}(\mathbf{x})\,\mathrm{d}\mathbf{x}$ which is the following:

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_{\text{data}}(\mathbf{x})}[\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]. \tag{2.3}$$

is equivalent to our score matching objective above when the noise is small enough such that $q_\sigma(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$. Intuitively following the gradient of a perturbed sample should bring us to the clean ones.

Sliced score matching (SSM) In a prior work of Song. He proposed sliced score matching [15] which uses random projections to approximate $\operatorname{tr}(\nabla_{\mathbf{x}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}))$ in score matching. The objective is

$$\mathbb{E}_{p_{\mathbf{v}}}\mathbb{E}_{p_{\text{data}}}\left[\mathbf{v}^\intercal\nabla_{\mathbf{x}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\mathbf{v} + \frac{1}{2}||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})||_2^2\right], \tag{2.4}$$

where $p_{\mathbf{v}}$ is a simple distribution of random vectors, *e.g.*, the multivariate standard normal. Sliced score matching tries to estimate the exact score of data through random projections, but requires around four times more computations since requiring forward mode auto-differentiation.

2.2 Empirical Variance Estimate

During training, we make an empirical estimate of score variance and update it stochastically. This is analogous to Adaptive Markov Chain Monte Carlo in MCMC theory [6, 7], and we leverage theory from [6, 1] and define our empirical estimate of covariance C at iteration t as follows,

$$\widehat{C_t} = \begin{cases} C_0 & t = 0 \\ s_d cov(X_0, ..., X_{t-1}) + s_d \epsilon I_d & t \geqslant 0 \end{cases} \tag{2.5}$$

Where s_d is a parameter that depends only on dimension d and $\epsilon > 0$ is a constant that we may choose very small, and $\mathbf{I}_d$ denotes the d-dimensional identity matrix. And C_0 is an arbitrary, strictly positive definite matrix chosen at the beginning, according to our best prior knowledge. And recall the empirical covariance matrix determined by points $X_0, ...X_n \in \mathbb{R}^d$,

$$cov(X_0, ..., X_n) = \frac{1}{k}\left(\sum_{i=0}^{n} X_i X_i^T - (n+1)\bar{X_n}\bar{X_n}^T\right), \tag{2.6}$$

$$\tag{2.7}$$

where $\bar{X_n} = \frac{1}{n+1}\sum_{i=0}^{n} X_i$. So one obtains in Eq. (2.5) the empirical covariance estimate satisfies the recursion formula,

$$C_{t+1} = \frac{t-1}{t}C_t + \frac{s_d}{t}(t\bar{X_{t-1}}\bar{X_{t-1}}^T) - (t-1)\bar{X_t}\bar{X_t}^T + X_t X_t^T + \epsilon I_d). \tag{2.8}$$

This formulation allows one to calculate C_t without too much computational cost since the mean X_t also satisfies an obvious recursion formula . The role of the parameter is just carried from the formulation of [6] for ergodicity reasons. As a basic choice for the scaling parameter we use $s_d = (2.4)^2/d$ same as [6], where it was shown that in a certain sense this choice optimizes the mixing properties in the case of Gaussian noises.

2.3 Diffusion MCMC for Sampling

Since we only have hands on the score function, we leverage diffusion based MCMC techniques to draw samples using only the score function, namely Langevin dynamics. Given a fixed step size $\epsilon > 0$, and an initial value $\tilde{\mathbf{x}}_0 \sim \pi(\mathbf{x})$ with π being a prior distribution, the Langevin method recursively computes the following

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\,\mathbf{z}_t, \tag{2.9}$$

where $\mathbf{z}_t \sim \mathcal{N}(0, I)$. The distribution of $\tilde{\mathbf{x}}_T$ equals $p(\mathbf{x})$ when $\epsilon \to 0$ and $T \to \infty$, in which case $\tilde{\mathbf{x}}_T$ becomes an exact sample from $p(\mathbf{x})$ when we have a long enough chain.

Note that sampling from Eq. (2.9) only requires the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. Therefore, in order to obtain samples from $p_{\text{data}}(\mathbf{x})$, we can first train our score network such that $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ and then approximately obtain samples with Langevin dynamics using $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$.

3 Adaptive Noise Conditioned Score Networks

In the previous work of Song [13], he found out that by using multiple noise levels we can obtain a sequence of noise-perturbed distributions that converge to the true data distribution. Extending this another step further, we propose to also leverage the variance information previously seen.

Let $\{\Sigma_i\}_{t=1}$ be the empirical variance estimated during each training . Let $q_\Sigma(\mathbf{x}) \triangleq \int p_{\text{data}}(\mathbf{t})\mathcal{N}(\mathbf{x} \mid \mathbf{t}, \Sigma)\,\mathrm{d}\mathbf{t}$ denote the perturbed data distribution. We aim to train a adaptive conditional score network to estimate the scores of the perturbed data distributions, *i.e.*, $\forall \Sigma \in \{\Sigma_i\}_{i=1} : \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log q_\Sigma(\mathbf{x})$. Note that $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathbb{R}^D$ when $\mathbf{x} \in \mathbb{R}^D$.

In terms of model architectures we adapt what Song used in [13] but with slight modification. To summarize, the model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$ combines the architecture design of U-Net [12] with dilated/atrous convolution [2] which are well known and successful architectures in semantic segmentation.

3.1 Training ANCSNs via DSM

We adopt denoising score matching as it is faster and naturally fits the task of estimating scores of noise-perturbed data distributions. We choose the noise distribution to be $q_\Sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \Sigma)$; therefore $\nabla_{\tilde{\mathbf{x}}} \log q_\Sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = -{}^{(\tilde{\mathbf{x}}-\mathbf{x})}/_{\Sigma}$. For a given Σ, the denoising score matching objective Eq. (2.3) is

$$\ell(\boldsymbol{\theta}) \triangleq \frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \Sigma)}\left[|| \frac{1}{\Sigma}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\Sigma} ||_2^2 \right]. \tag{3.1}$$

Where the $1/\Sigma$ factor adjust for the scale in score at different variance levels according to [14]. And empirically, we observe that when the score networks are trained to optimality we approximately have $||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})_{\Sigma}||_2 \propto 1/\Sigma ||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})_{\Sigma_0}||_2$. It is worth highlighting that this objective requires no adversarial training, no surrogate losses, and no sampling from the score network during training unlike contrastive divergence when training energy based model.

3.2 ANSN Sampling via Annealed Langevin dynamics

After we have our trained ANSN $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$, we adopt the sampling approach described by Song in [13] via annealed Langevin dynamics (Alg. 1) to produced samples but now adopting a fixed scale of σ's

Algorithm 1 Annealed Langevin dynamics.

```
Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.
Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to $L$ do
    $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ {$\alpha_i$ is the step size.}
    for $t \leftarrow 1$ to $T$ do
        Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
        $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2\sigma_i}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\alpha_i}\,\mathbf{z}_t$
    end for
    $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$ =0
```

we start annealed Langevin dynamics by initializing the samples from some fixed prior distribution, *e.g.*, uniform noise. Then, we run Langevin dynamics to sample from $q_{\sigma_1}(\mathbf{x})$ with step size α_1. Next, we run Langevin dynamics to sample from $q_{\sigma_2}(\mathbf{x})$, starting from the final samples of the previous simulation and using a reduced step size α_2. We continue in this fashion, using the final samples of Langevin dynamics for $q_{\sigma_{i-1}}(\mathbf{x})$ as the initial samples of Langevin dynamic for $q_{\sigma_i}(\mathbf{x})$, and tuning down the step size α_i gradually with $\alpha_i = \epsilon \cdot \sigma_i^2/\sigma_L^2$. Finally, we run Langevin dynamics to sample from $q_{\sigma_L}(\mathbf{x})$, which is close to $p_{\text{data}}(\mathbf{x})$ when $\sigma_L \approx 0$.

4 Experiments

In this section, we illustrate some of our examples generated from our ANSNs. They are able to produce high quality image samples on MNIST and CIFAR10 image datasets.

Setup We use MNIST and CIFAR-10 [10] datasets in our experiments. All images are rescaled so that pixel values are in $[0, 1]$. When using annealed Langevin dynamics for image generation, we choose $T = 100$ and $\epsilon = 2 \times 10^{-5}$, and use uniform noise as our initial samples.

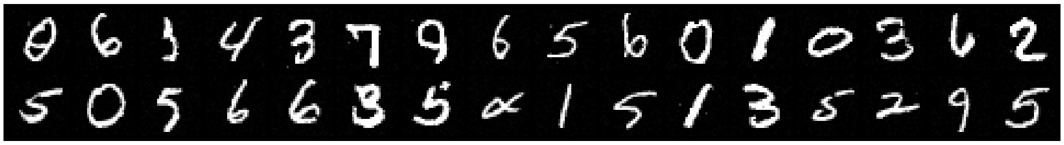

Figure 2: MNIST 1000 ALD Sampling Iterations

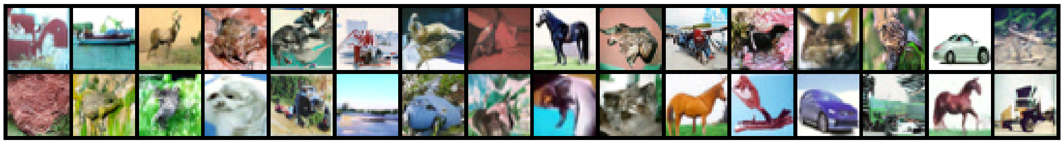

Figure 3: CIFAR 1000 ALD Sampling Iterations

Image generation we see that high uncurated samples can be generated from annealed Langevin dynamics for MNIST and CIFAR-10. As shown by the samples, our generated images have higher or comparable quality to those from modern likelihood-based models and GANs.

5 Conclusion

We propose the Adaptive Noise Score Networks where perform generative modeling not on the data distribution but instead on the gradients of data densities via score matching, and then generate samples via Langevin dynamics. We outlined several limitation of the previous methods and proposed to use a adaptive variance scheme to alleviate. Experimentally, we show that our approach can generate high quality images while being well generalized on the data distribution.

References

[1] Y. F. Atchadé and J. S. Rosenthal. On adaptive markov chain monte carlo algorithms. *Bernoulli*, 11(5):815–828, 2005.

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[3] J. Deasy, N. Simidjievski, and P. Liò. Heavy-tailed denoising score matching. *arXiv preprint arXiv:2112.09788*, 2021.

[4] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[6] H. Haario, E. Saksman, and J. Tamminen. An adaptive metropolis algorithm. *Bernoulli*, pages 223–242, 2001.

[7] H. Haario, E. Saksman, and J. Tamminen. Componentwise adaptation for high dimensional mcmc. *Computational Statistics*, 20(2):265–273, 2005.

[8] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[9] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] A. Krizhevsky and G. Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

[11] Q. Liu, J. Lee, and M. Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284. PMLR, 2016.

[12] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[13] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

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

[15] Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[16] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[17] P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

A Sampled Results

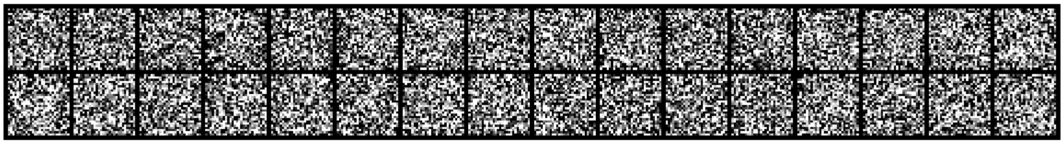

Figure 4: MNIST Iteration 1

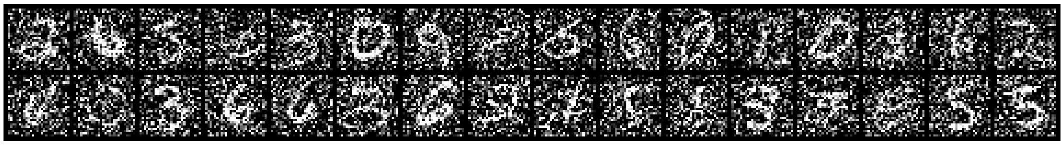

Figure 5: MNIST Iteration 100

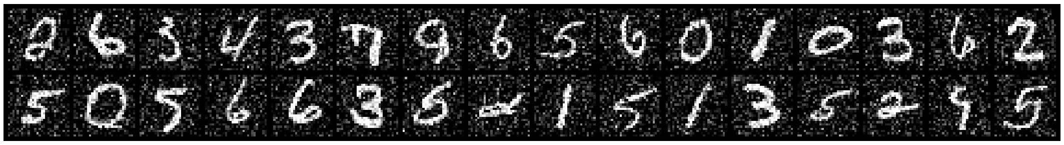

Figure 6: MNIST Iteration 300

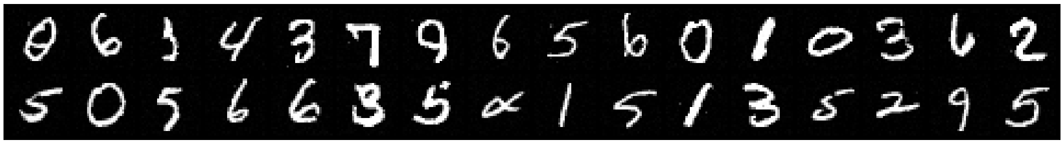

Figure 7: MNIST Iteration 1000

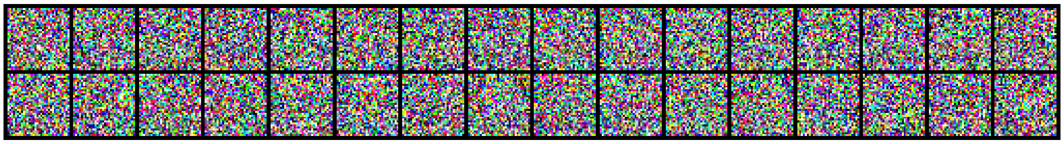

Figure 8: CIFAR10 Iteration 1

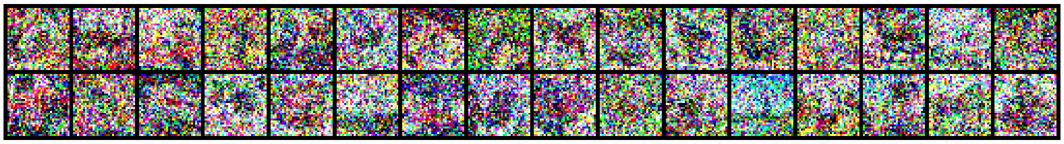

Figure 9: CIFAR10 Iteration 100

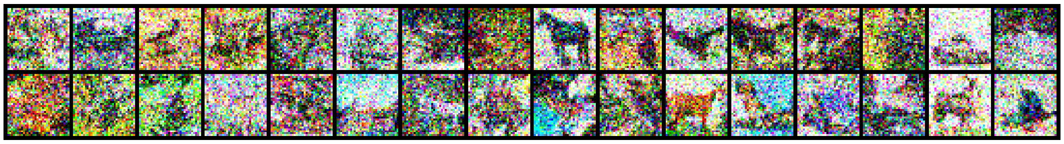

Figure 10: CIFAR10 Iteration 300

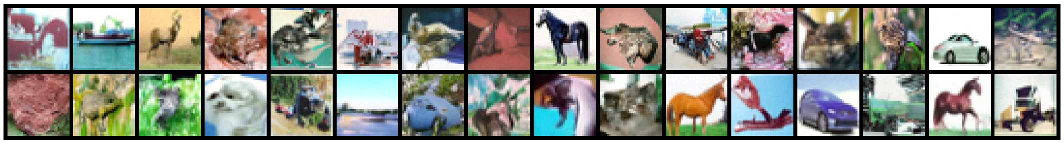

Figure 11: CIFAR10 Iteration 1000