Score-Based Generative Modeling with Multi-Sample Denoiser

Anuj Nagpal
ICME, Stanford University
anujnag@stanford.edu

Abstract

Score-based generative models are gaining a lot of traction recently due to their GAN-level image sampling quality without adversarial training along with the added advantage of exact log-likelihood computation [3, 13]. It has been known for quite some time that denoising autoencoders are equivalent to score-matching models [14]. In this project, we propose that it is possible to extend this equivalence to denoise multiple noisy samples of an image together and get a better score estimate. We experiment this idea on some popular image datasets and show that the denoising model can be trained quicker with lesser error as we increase the number of samples, keeping the architecture and number of parameters same. This opens up the possibility to extend this idea to incorporate image augmentations or importance weighting in score-based generative models.

1. Introduction

Image generative modelling techniques have been broadly grouped into two categories so far. The first class is likelihood based models which directly learn the distribution's probability density function via maximum likelihood such as autoregressive models, variational autoencoders [5], normalizing flow models [8], and energy based models [7]. The other class is implicit generative models where the probability distribution is implicitly represented by a model of its sampling process e.g. generative adversarial networks [4]. Despite their success, both of these techniques have their own limitations. Likelihood based models either require strong restrictions on the model architecture to ensure a tractable normalizing constant for likelihood computation (e.g. energy based models), or must rely on surrogate objectives to approximate maximum likelihood training (e.g. VAEs). Implicit generative models, on the other hand, often require adversarial training (e.g. GANs), which has been observed to be unstable and can lead to mode collapse.

Song et al [11] proposed another way to represent probability density functions via the gradient of its logarithm, known as the score function, which is able to circumvent most of these limitations. Apart from achieving state-of-the-art performance on many applications, score based models do not require approximating the intractable normalizing constant, can be directly learned without adversarial training and also provide a way to compute exact likelihood by stochastic differential equation (SDE) formulation. In this project, we use this SDE formulation and extend it to multiple samples for generating images.

1.1. Score Based Models

Generative modeling tries to learn a model distribution that is as close to the true data distribution as possible, so that we can generate realistic-looking samples from it. For energy based models, this involves modeling a probability density function of the form:

$$p_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta}$$

where f_θ is a real-valued function parametrized by θ and Z_θ is a normalizing constant such that $\int p_\theta(x)dx = 1$. The score function, $s_\theta(x)$ for this distribution is then defined as:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = -\nabla_x f_\theta(x)$$

Score-based models learn the score function by minimizing the fisher divergence between the model and the data distributions, defined as:

$$\mathbb{E}_{p(x)}[||\nabla_x \log p_{\text{data}}(x) - s_\theta(x)||_2^2]$$

Even though we can't directly calculate unknown data score $\nabla_x \log p_{\text{data}}(x)$, there exists a class of methods known as 'score matching' techniques that can minimize this fisher divergence and learn an approximate score function.

1.2. Sampling with Langevin Dynamics

Once we have trained a score function $s_\theta(x) \approx \nabla_x \log p(x)$, we can use an iterative MCMC procedure called Langevin Dynamics to sample from a distribution using only its score function:

$$x_{t+1} \leftarrow x_t + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_t, \qquad t = 0, 1, ..., T$$

where x_0 is drawn from an arbitrary prior distribution $\pi(x)$ and $z_t \sim \mathcal{N}(0, I)$. When $\epsilon \to \infty$ and $T \to \infty$, x_T obtained from this procedure converges to a sample from true data distribution $p(x)$ under some regularity conditions.

Even though this looks good theoretically, real-life datasets usually reside on low-dimensional manifolds in high dimensional spaces and a point sampled from the prior distribution is likely to have an undefined log probability in the high dimensional space, and thus an undefined score function. This can derail the score-based model in the initial iterations of Langevin chain and prohibit us from generating a high quality sample. To overcome this problem, we perturb data points with noise to populate the low density regions and train score based models on the noisy data points instead.

Algorithm 1 Annealed Langevin Dynamics

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$
  Initialize $\widetilde{x}_0$
  $X \leftarrow x$
  $N \leftarrow n$
  for $i \leftarrow 1$ to L **do**
    $\alpha_i \leftarrow \epsilon.\sigma_i^2/\sigma_L^2$ (α_i is the step size)
    for $t \leftarrow 1$ to T **do**
      Draw $z_t \sim \mathcal{N}(0, I)$
      $\widetilde{x}_t \leftarrow \widetilde{x}_{t-1} + \frac{\alpha_i}{2} s_\theta(\widetilde{x}_{t-1}, \sigma_i) + \sqrt{\alpha_i} z_t$
    end for
    $\widetilde{x}_0 \leftarrow \widetilde{x}_T$
  end for
  return $\widetilde{x}_t$

2. Related Work

2.1. Noise Perturbed Score Based Model

To bypass the problem discussed above with vanilla score matching, we perturb data points with noise to populate the low density regions and train score based models on the noisy data points instead. This noise scale can be small or large but there is a tradeoff in using each - larger noise can cover more low density regions for better score estimation, but it over-corrupts the data distribution. On the other hand, smaller noise causes less corruption of the original data distribution but does not cover the low density regions that good. A solution that takes best of both worlds uses multiple scales of noise perturbations simultaneously, known by the name of Annealed Langevin Dynamics.

To make it concrete, the idea is that:

- We draw samples from a noise perturbed distribution $q_{\sigma_i}(\widetilde{x}|x)$ by first sampling $x \sim p(x)$ and then compute $x + \sigma_i z$ with $z \sim \mathcal{N}(0, 1)$ for different noise scales $\sigma_1 < \sigma_2 < ... < \sigma_L$.
- With that, we can estimate the score function of each noise-perturbed distribution by training a Noise Conditional Score Network with score matching such that $s_\theta(x, i) \approx \nabla_x \log p_{\sigma_i}(x)$ for all $i = 1, 2, ..., L$.
- Then we can generate samples using this trained score network $s_\theta(x, i)$ by running 'annealed' Langevin dynamics for $i = L, L-1, ..., 1$ in sequence (see Algorithm 1).

2.2. Denoising Score Matching

In denoising score matching, we perturb a data point x with noise to generate a sample from $q_\sigma(\widetilde{x}|x)$ and then do score matching for the distribution of the perturbed data $q_\sigma(\widetilde{x}) := \int q_\sigma(\widetilde{x}|x) p_{\text{data}}(x) dx$ by optimizing:

$$\frac{1}{2}\mathbb{E}_{x \sim p_{data}(x), \widetilde{x} \sim q_\sigma(\widetilde{x}|x)}[||s_\theta(\widetilde{x}) - \nabla_{\widetilde{x}} \log q_\sigma(\widetilde{x}|x)||_2^2]$$

Doing score matching for perturbed distribution clears out the computationally expensive trace term but the caveat is that this doesn't give us the score of noise-free or clean data and there is some approximation error.

2.3. Denoising Autoencoder as Score Matching

A denoising autoencoder is able to generate a denoised image given a noisy sample and is generally trained with mean square error as the loss objective:

$$\mathcal{L}(y, x; \theta) = ||\hat{x}(y; \theta) - x||_2^2$$

where y here is a noisy sample obtained from original sample x. Pascal Vincent [14] showed that denoising autoencoders can be seen as equivalent to score matching techniques. So if we have a denoised least squares estimate $\hat{x}$ from a perturbed sample y, we can use it to estimate the score function using Tweedie's formula as:

$$\nabla_x \log p(x) = \frac{\hat{x} - y}{\sigma^2}$$

where σ is the noise scale of perturbation.

3. Approach

3.1. Multi-Sample Denoiser

Denoising autoencoders conventionally use a single noisy sample to get a least squares estimate but the model can be easily extended to more than one sample such that given k noisy samples of an input data point from true data distribution, our denoiser can learn to reconstruct the original data point, thus giving us a multi-sample least squares

estimate. Such a denoiser can be trained with a similar loss function, but generalized to k different noisy samples:

$$L(y_{1:k}, x; \theta) = ||\hat{x}(y_{1:k}; \theta) - x||_2^2$$

With this idea in our hand, the missing bridge between our multi-sample denoiser and Langevin sampling is getting a score estimate. To complete this gap, we extend the Tweedie formula to multiple samples as proved in next section.

3.2. Multi-Sample Tweedie Formula

Consider $y_1, y_2, ..., y_k$ as k independent noisy observations of a random variable x. The least-square estimator of x given $y_1, y_2, ..., y_k$ is given by:

$$\begin{aligned}\hat{x}(y_1, y_2, ..., y_k) &= \int x P(x|y_1, y_2, ..., y_k)\, dx \\ &= \int x \frac{P(x, y_1, y_2, ..., y_k)}{P(y_1, y_2, ..., y_k)}\, dx\end{aligned}$$

We can also write that

$$\begin{aligned}P(y_1, y_2, ..., y_k) &= \int P(x, y_1, y_2, ..., y_k) dx \\ &= \int P(y_1|x)P(y_2|x)...P(y_k|x)P(x)dx\end{aligned}$$

where $P(y_i|x) = \mathcal{N}(y_i; x, \sigma^2) \ \ \forall i$.

Now consider:

$$\begin{aligned}&\sigma^2 \frac{\partial P(y_1, y_2, ...y_k)}{\partial y_1} \\ &= \sigma^2 \int \frac{\partial P(y_1|x)}{\partial y_1} P(y_2|x)...P(y_k|x)P(x)dx \\ &= \int (x - y_1)P(y_1|x)P(y_2|x)...P(y_k|x)dx\end{aligned}$$

which can also be written as:

$$\begin{aligned}&\sigma^2 \frac{1}{P(y_1, y_2, ..., y_k)} \frac{\partial P(y_1, y_2, ..., y_k)}{\partial y_1} \\ &= \int (x - y_1) \frac{P(x, y_1, y_2, ..., y_k)}{P(y_1, y_2, ..., y_k)} dx \\ &= \hat{x}(y_1, y_2, ..., y_k) - y_1\end{aligned}$$

This finally gives us:

$$\frac{\partial \log P(y_1, y_2, ..., y_k)}{\partial y_i} = \frac{\hat{x}(y_1, y_2, ..., y_k) - y_i}{\sigma^2}$$

Thus given a denoiser based on k samples, we can get a score estimate and run langevin dynamics to sample from $P(y_1, y_2, ..., y_k)$ as follows:

$$\begin{bmatrix} y_{1,t+1} \\ y_{2,t+1} \\ ... \\ y_{k,t+1} \end{bmatrix} \leftarrow \begin{bmatrix} y_{1,t} \\ y_{2,t} \\ ... \\ y_{k,t} \end{bmatrix} + \epsilon \begin{bmatrix} \nabla_{y_1} \log P(y_1, ..., y_k) \\ \nabla_{y_2} \log P(y_1, ..., y_k) \\ ... \\ \nabla_{y_k} \log P(y_1, ..., y_k) \end{bmatrix} + \sqrt{2\epsilon} \begin{bmatrix} z_1 \\ z_2 \\ ... \\ z_k \end{bmatrix}$$

where $z_1, z_2, ..., z_k \sim \mathcal{N}(0, I)$.

The whole idea can be summarized in below figure:

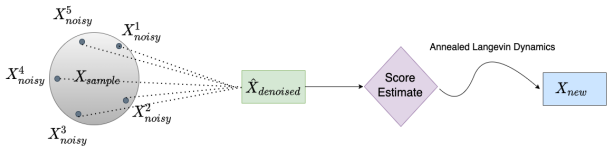

Figure 1. Score Matching with Multi-Sample Denoiser

Why this might be better? - We claim that using multiple samples instead of one gives our score matching model a more sophisticated understanding of the true data distribution.

3.3. Model Design

- The data distribution p_0 is perturbed to the prior distribution p_t as per the stochastic differential equation (SDE):

  $$dx = \sigma^t dw, t \in [0, 1]$$

  The reason for choosing this SDE is governed by the mathematical result that for large σ, the prior distribution we are perturbing towards, $p_{\{t=1\}}$, is given by:

  $$\int p_0(y) \mathcal{N}\left(x, y, \frac{(\sigma^2 - 1)I}{2 \log \sigma}\right) dy \approx \mathcal{N}\left(0, \frac{(\sigma^2 - 1)I}{2 \log \sigma}\right)$$

  which is approximately independent of the data distribution and is easy to sample from.

- Intuitively, this SDE captures a continuum of Gaussian perturbations with variance function $\frac{(\sigma^{2t} - 1)I}{2 \log \sigma}$. This continuum of perturbations allows us to gradually transfer samples from a data distribution p_0 to a simple Gaussian distribution p_1.

- With this formulation, we generate multiple i.i.d. noisy samples by first sampling t in $[0, 1]$, followed by calculating a uniform scaling factor $\lambda = \frac{(\sigma^2 - 1)I}{2 \log \sigma}$ with a large σ, and then sampling different noise $z_i \sim N(0, I)$ for each sample so that a single perturbed sample is given by $\tilde{x}_i = x + \lambda z_i$.

- We also incorporate the noise scale information as Gaussian random features similar to the time scale SDE formulation by Song et al. [13] as they have been proven to learn high frequency functions in low dimensional domains. Specifically, we first sample a fixed non-learnable parameter $\omega \sim \mathcal{N}(0, \sigma^2 I)$ and for a noise scale t, the resulting noise scale embedding will be the concatenated vector $[\sin(2\pi\omega t); \cos(2\pi\omega t)]$.

•

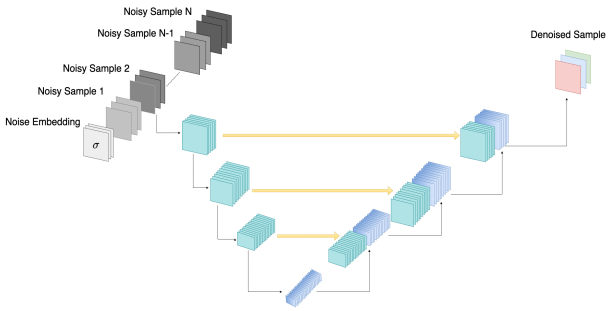

Figure 2. Model Architecture

3.4. Model Architecture and Hyperparameters

- We use a U-net architecture [9] as our multi-sample denoiser where the multiple input noisy samples are concatenated along the channel dimension. For a $(28, 28, 3)$ colored image with k noisy samples, this will be equivalent to an input with dimensions $(28, 28, 3 * k)$ (illustrated in Figure 2).

- We feed the noise scale information given by the vector $[\sin(2\pi\omega t); \cos(2\pi\omega t)]$ through a linear layer to get a noise scale embedding which is further concatenated with all the i.i.d noisy samples for the network input (see Figure 2).

- The U-net downsamples the input data by 2D convolutions with 64, 128, 256 and 512 number of filters, followed by upsampling via transposed convolutions to the original resolution which gives us a denoised estimate.

- We also keep passing the noise scale embedding through a linear layer after each convolution stage and simply add it to our activations from convolutions. The combined state is then passed through Group Normalization and Swish Activation layers before feeding it to the next convolution.

- The hyperparameters used for the experiments discussed ahead had the following values:

  - Batch Size = 256
  - Noise Scale Embedding Dimension = 256
  - Langevin Dynamics Chain Length = 10
  - Number of Epochs = 100
  - Learning Rate = 3e-3
  - Value of σ = 25

4. Experiments on Different Datasets

4.1. Toy Gaussian Mixture Model

To test our proposition, we start our first experiments with a toy dataset of a uniformly weighted mixture of Gaussian distributions with probability density given by:

$$p(x) = \sum_{i=1}^{K} w_i \mathcal{N}(\mu_i, \Sigma_i)$$

where w_i are the mixture weights. The choice of this dataset was driven by the availability of true data distribution so that we can easily compare generated samples with simulated data from true distribution.

After training a denoiser until the loss objective goes down near to zero and running annealed Langevin dynamics for 100,000 iterations from the score estimates, we plot the last 20,000 points for 10 chains with first 80,000 points used as burn-in. Samples from true data distribution are also plotted ahead for visualization:

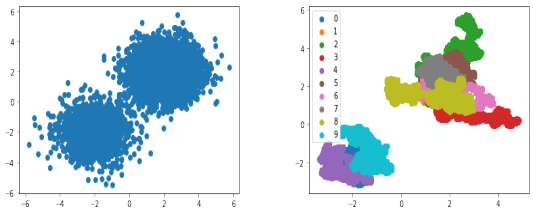

Figure 3. True Distribution vs Denoised Sample Trajectories

It can be observed that the generated data points are close to modes of both Gaussian distributions, showing some promise for the method.

4.2. MNIST Experiments

After getting promising results on the toy distribution, we ramped up the model to 3-dimensional $(H, W, C) = (28, 28, 1)$ images and experimented with the MNIST dataset [2]. Keeping the model architecture and training time same, the denoising model achieved better performance as we increased the number of noisy samples fed in the input:

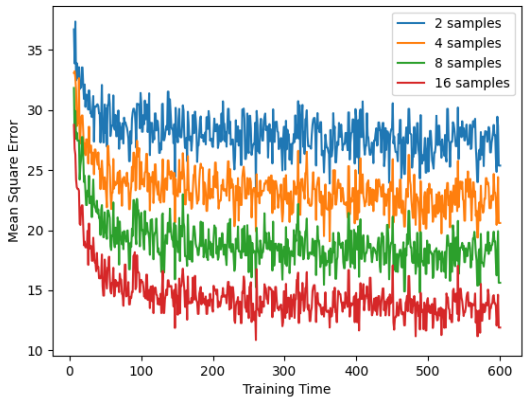

Figure 4. MNIST Training Curves

The model was able to generate realistic looking samples from complete noise as shown in Figure 5 with score estimates from model trained to denoise 8 samples:

Figure 5. MNIST Samples

It is worth noting that not all chains converged to good samples and we were required to change the langevin chain parameters or seed at times.

4.3. CIFAR-10 Experiments

The next step was to experiment with $(H, W, C) = (32, 32, 3)$ colored CIFAR-10 images [6] and noticed a similar pattern on increasing the number of samples. The difference isn't visible on increasing the number of samples from 2 to 4 but becomes clearer as we make it 8 or 16:

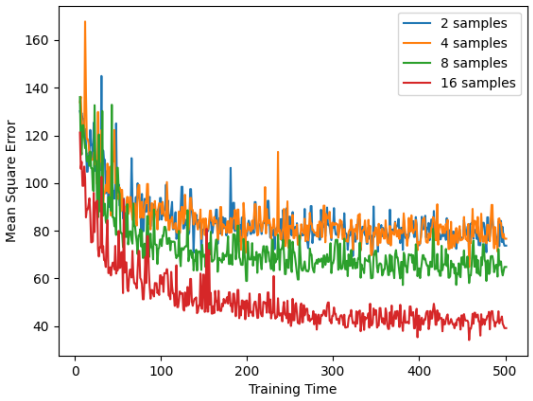

Figure 6. CIFAR-10 Training Curves

The model was able to generate realistic looking samples from complete noise as shown in Figure 7 for model trained to denoise 8 samples but once again, it is worth noting that most of the langevin chains converged to a monochromatic image with nothing meaningful.

Figure 7. CIFAR10 Samples

5. Results and Discussions

There were two clear gains on increasing the number of samples fed to denoising model:

- The denoiser gets trained trained quicker and achieves lesser mean squared error on increasing the number of samples, when trained for same number of epochs.
- Since we have a better least squares estimate, the score estimates were also more accurate and as a result, all the annealed langevin chains converged more quickly to realistic looking samples.

Despite the model's success to finally generate realistic looking samples, it's also worth noting that there are certain issues with the approach:

- Not all the langevin chains converged to meaningful samples. The sampling is tightly dependent on the value of epsilon, noise scales used, and the number of iterations per noise scale.
- Even though we can estimate score values from denoised image using the equivalency proved earlier, it

seems like learning to denoise is a harder task for the model than to estimate the gradient since the former is similar to learning a point whereas the latter is similar to learning a direction.

Since it was really hard to sample 2048 meaningful images from our model trained on CIFAR-10 images, it wasn't fair to compute FID scores and compare it with other benchmarks. The noticeable improvements were in the final denoising error after training for same number of epochs and we report the corresponding mean square error as our quantitative metric for comparison:

#Samples	MSE - MNIST	MSE - CIFAR10
1	29.379	90.929
2	25.383	73.761
4	20.606	76.676
8	15.631	64.834
16	11.904	39.172

Table 1. Number of Samples vs Final Denoising Error

6. Conclusions and Future Work

We were able to show that it is in fact possible to denoise multiple images together, get a score estimate and use that to generate image samples that look similar to the ones in true data distribution. This project serves as a good proof-of-concept for using this multi-sample extension and combine it with ideas from variational autoencoder space to get gains in blooming field of score-based generative models.

One possible idea to explore in future is to give importance weights to each of the samples to get improvements similar to IWAEs. Another direction is to extend the integral for calculating likelihood by Yang et al [12] to multiple samples. Both of these ideas are not obvious from a mathematical point of view and may require some algebraic work.

Another idea is to incorporate consistency regularization for different image augmentations as have been shown for VAEs [10]. The idea we propose is to consider a VAE which has a large number of stochastic layers (just as [1]) such that as the number of layers goes to infinity, we can view the process of going from the image to the deepest noise as a stochastic process, i.e. an SDE. So the correct analogy will be that instead of saying that a single distribution $q_\phi(z||\widetilde{x})$ is similar to $q_\phi(z|x)$, we will want $q_\phi(z(t)||\widetilde{x})$ and $q_\phi(z(t)||x)$ to be similar where $q_\phi(z(t)||\widetilde{x})$ is the distribution induced by SDE. How to carry this out in practice and how to weigh this consistency regularization for various t is a potential research direction.

7. Acknowledgements

We thank Ermon group in Stanford Artificial Intelligence Laboratory (SAIL) for providing access to Atlas GPU cluster for conducting the experiments and Chris Cundy for acting as an external mentor for this project.

References

[1] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020. 6

[2] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 4

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021. 1

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014. 1

[6] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[7] Yann Lecun, Sumit Chopra, and Raia Hadsell. *A tutorial on energy-based learning*. 01 2006. 1

[8] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016. 1

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 4

[10] Samarth Sinha and Adji B Dieng. Consistency regularization for variational auto-encoders. *arXiv preprint arXiv:2105.14859*, 2021. 6

[11] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, 2019. 1

[12] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020. 6

[13] 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. 1, 3

[14] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011. 1, 2