Diffusion Model Theory and Adaptation to MRFs

Mohammed Ehab
ehab02@mit.edu

12/13/2023

Abstract

Diffusion models are a recently-developed powerful way to sample from distributions given only sample access to them. The goal of this project is twofold: first, to understand the theory behind diffusion models and their guarantees, and second, to study them in the context of Markov Random Fields.

1 Introduction

The field of generative modeling has seen a massive breakthrough in the last few years with the invention of denoision diffusion probabilistic models (DDPMs) [HJA20] and score-based models [SE20]. After gaining popularity through their success, it became clear that the two models have some similarities, and they were unified under the framework of stochastic differential equations (SDEs) in [SSDK+21].

The success these models have seen inspires their adaptation for learning structured distributions. In particular, it raises the following question: if we know something about the distribution we're sampling from, P, how can we incorporate it into the algorithm? This project aims to answer this questions for distributions that are well-modeled by Markov Random Fields (MRFs).

In section 2, we will explore how diffusion and score-based models work, as well as their unification. We will also offer background on a task they solve that we'll use to test our algorithm. In section 3, we will discuss the algorithm and its experimental results. In section 4, we will discuss improving a detail about these models (namely, their noise schedule) in order to get faster sampling from MRFs. Finally, in section 5, we will discuss the literature on the mixing time of diffusion models.

2 Background

2.1 Score-Based Generative Modeling

The story starts with the following dynamics for sampling from a distribution p:-

$$x_{t+1} = x_t + \epsilon \nabla \log(p(x_t)) + \sqrt{2\epsilon} Z_t, Z_t \sim \mathcal{N}(0, 1)$$

This algorithm is called Langevin dynamics, and it can be shown that under some regularity conditions, the distribution of x_t converges to p [WT]. This motivates the following idea for generative modeling: since Langevin dynamics only use $\nabla \log(p(x_t))$, called the score function, we can parameterize a neural network $s_\theta(x)$ that predicts the score of x. We can do so by minimizing the following loss function:-

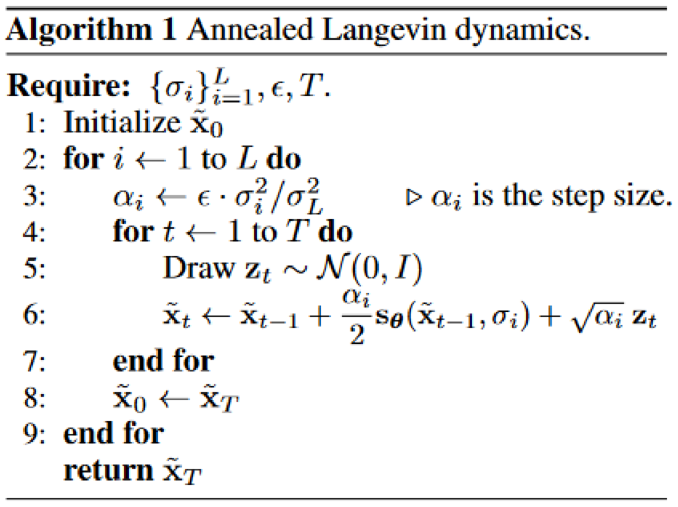
$$L(\theta) = E_x[||\nabla_x \log(p(x)) - s_\theta(x)||^2]$$

Of course, we don't have access to $\nabla_x \log(p(x))$; we only have access to data. But a technique developed as early as 2005 called score matching can be used to mitigate this issue [Hyv]. Once we have that network, we can run Langevin dynamics to sample.

This approach has seen little success in practice. The reason is that the loss function above cares less about regions where p has low density, so the score estimates from these regions are inaccurate. This derails Langevin dynamics and generates low quality samples [SE20].

To mitigate that, one idea is to add noise to the data before estimating the score in order to fill these low density regions. The more noise we add, the better noise estimates get, but the noisier our samples get as well.

To mitigate *that*, we can add multiple levels of noise, and sample from progressively less noisy versions of p. Specifically, in each step of Langevin dynamics, we will use a different step size ϵ_i, and we will use the score function from a noisy distribution $p_{\sigma_i}(x)$. This given the following algorithm called Annealed Langevin dynamics [SE20]:-



Algorithm 1 Annealed Langevin dynamics.

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

Following the same recipe from before, we can learn a noise-conditional neural network $s_\theta(x, \sigma)$ that learns the score of p after being noised with IID Gaussian noise of variance σ. The new loss function will be:-

$$\sum_{i=1}^{L} \lambda_i E_x[||\nabla_x \log(p_{\sigma_i}(x)) - s_\theta(x, \sigma_i)||^2]$$

which is just a weighted sum of the loss functions we had before. The paper demonstrates that this algorithm is much more successful than vanilla Langevin dynamics.

Concurrently, DDPMs were being developed in the work of Ho et. al. [HJA20]. The idea behind them is a little bit different: suppose we have a forward process that gradually adds noise to our data until it becomes completely noise. The process has the following dynamics:-

$$x_{t+1} = \beta_t x_t + \sqrt{1 - \beta_t^2} Z, Z \sim \mathcal{N}(0, 1)$$

Then, we will learn how to reverse this process to generate data from noise. The authors then claim that the reverse of this process is approximately a Gaussian so that the dynamics are the following:-

$$x_{t-1} \sim \mathcal{N}(\mu(x_t, t), \Sigma(x_t, t))$$

They then parameterize the means and covariances as neural networks, and for the loss function, they essentially minimize the KL divergence between the distribution of the path of the backwards process and that of the path of the forwards process.

The models we discussed have some similarities in how they progressively noise and denoise the data, but it's unclear what the exact connection between them is. In particular, how does the score function play a role in DDPMs? Why is the reverse process in a DDPM approximately Gaussian?

A unified framework for both models was developed in [SSDK$^+$21] in the language of stochastic differential equations. Let's take the idea of the forward process in DDPMs but run it in continuous time. We get a stochastic differential equation of the following form:-

$$dx = f(x, t)dt + g(t)dw$$

where f represents what happens to our signal and g represents the noise schedule, and the initial condition is $x_0 = p$. It turns out that it's possible to run this SDE backwards in time using the following SDE:-

$$dx = [f(x, t) - g^2(t)\nabla \log(p_t(x))]dt + g(t)dw$$

This theorem explains two things: first, the score function appears again! Second, this explains why the reverse process in DDPMs is approximately Gaussian: the continuous version of it is the reverse of an SDE, which is another SDE.

Using this theorem, we can do score-based modeling in continuous time. We can use any SDE solver we want for the reverse SDE, and we can minimize a continuous version of the neural network loss:-

$$\int_0^T \lambda(t) E_x[||\nabla_x \log(p_t(x)) - s_\theta(x, t)||^2]$$

Effectively, score-based models and DDPMs are two different interpretations and discretizations of this idea. The SDE is ultimately our choice and part of the model. Score-based models choose an SDE that doesn't attenuate the signal and only injects noise. They were also described in a way that emphasizes the score function and separates it from Annealed Langevin dynamics, which can now be understood as a particular discretization of the SDE. On the other hand, DDPMs were described in a way that emphasizes the existence of a forwards noising process that we want to reverse. However, the score estimation and backwards dynamics were entangled inside the neural networks that estimate the mean and covariance. What remains to understand is the equivalence of the losses. We'll see near the end of the report that by choosing the right weighting function $\lambda(t)$, we minimize the KL divergence just like DDPMs.

2.2 Image Super-Resolution

Image super-resolution is about solving the following task: given a low-resolution image, upsample it to a high-resolution image that has the same features.

Employing diffusion models to solve image super-resolution was explored in [SHC$^+$21]. In this paper, the author proposed the following solution: suppose we have a joint distribution (X, Y) of low-resolution images X and their corresponding high-resolution images Y. Then, we can model the task as sampling from the distribution $Y|X$. This new task can be solved with score-based modeling as follows: while learning the score function, we not only condition on time, but also on the low-resolution image X.

During training, we may only have access to high-resolution images. We would need to turn them into pairs of images (X, Y) from the joint distribution. To do that, we can downsample them with whatever noising operations we want our model to be immune against.

3 Learning Score Functions of MRFs

3.1 Theory

The Hammersley-Clifford theorem tells us that the density of MRFs can be factorized as follows:-

$$p(x) = \prod_C \psi(C)$$

where C iterates over the cliques of the underlying graph G. Using that fact:-

$$\nabla_x \log(p(x)) = \nabla_x \log(\prod_C \psi(C)) = \sum_C \nabla \log(\psi(C))$$

which writes the score function as a sum of smaller score functions. Notice that instead of summing over the cliques, we can equivalently sum over the local neighborhoods of each vertex in the graph.

Within our learning framework, this motivates the following idea: instead of learning one large neural network to estimate the scores, we can learn N small neural networks and sum up their outputs.

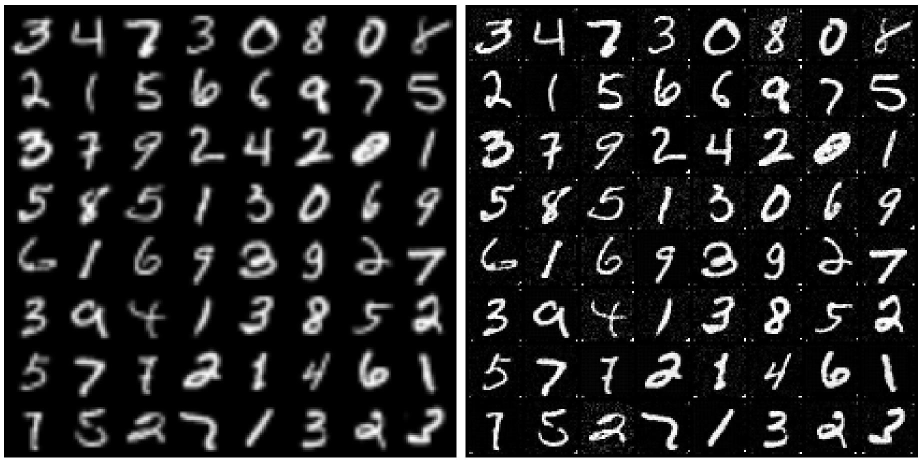
3.2 Experiments

Diffusion models are typically bench-marked on their ability to generate images. However, image distributions do not typically follow a local MRF structure. Instead, we experiment with solving image super-resolution. Recall that in that setting, we wanted to sample from $Y|X$, where X is a low-resolution and Y is a high-resolution image. Due to the conditioning, this distribution is more local and well-represented by an MRF.

The experiment was carried on MNIST. Each small network took as input the 9 pixels around a given pixel + the 9 pixels from the low-resolution image + the time we're conditioning upon, t. It then consisted of 4 fully-connected layers with Swish activations. The code for the project can be found here.

3.3 Initial Results

Surprisingly, the training error saturated at a high value. To verify that the issue was in the model and not anywhere else in the code, I ran a second experiment where the model was replaced with a normal CNN as a baseline. The training worked as expected, and the model was able to upsample low-resolution images:-



3.4 The Time Embedding Issue

As demonstrated in [TSM+20], neural networks fail to learn high-frequency functions in one of their variables. In this setting, the score function changes rapidly as a function of t, which causes a practical failure to learn.

The solution proposed in [TSM+20] is to avoid feeding in t directly. Instead, pick fixed values $\omega_1, \ldots, \omega_D$, and replace t with the feature vector $[sin(\omega_1 t), cos(\omega_1 t), \ldots, sin(\omega_D t), cos(\omega_D t)]$. These functions change with different frequencies as functions of t, enabling neural networks to learn more effectively.

Indeed, this approach is employed in the CNN architecture I used to get the positive results. However, there's a hurdle with employing it for the architecture with the small networks: the embedding dimension, D, depends on the Fourier coefficients of the score function rather than the MRF structure, and the values people succeeded with in practice to train diffusion models are all in the triple-digits. This defies the point of using N small neural networks.

I still attempted to use a low-dimensional embedding of t, but the training error still didn't decrease enough. Similar results were found by conditioning on the noise level instead of time. Finally, I tried discretizing time again and training a different model for each time step, but this approach had a main drawback: the number of parameters was huge, since it scaled linearly with the number of time steps. This means the model requires too much memory for any practical application, and furthermore, requires far longer training time to see any results. Alas, none of the approaches worked out in practice.

3.5 Discussion

While the theory still holds, getting this approach to work in practice requires fixing the time embedding issue. Furthermore, while the goal of using smaller neural networks is achieving higher efficiency, it became clear in hindsight that this wouldn't be the case for image super-resolution and other vision applications. The reason is that people have already been using locality very effectively since they replaced fully-connected networks with CNNs. Perhaps testing the approach on data coming from a less-structured MRF would achieve more interesting results.

4 Noise Schedule that Respects MRFs

Another thing we can study is developing a noise schedule that respects the MRF structure. Namely, as Costis suggested, instead of adding IID noise, maybe it's better to add the Gaussian with the same precission matrix as our data.

One way to formalize this problem is the following: given some metric D for distances between distributions, find a noise schedule such that the path from our initial distribution, P, to some final Gaussian distribution is as small as possible.

4.1 Closest Gaussian in Wasserstein Distance

I focused on studying a simpler problem: what's the closest Gaussian to P in some distance D? We'll start with the Wasserstein distance.

As a reminder, the Wasserstein-2 distance is defined as:-

$$W_2(X,Y)^2 = \inf_{\gamma \in \Gamma(X,Y)} E_{(x,y)\sim\gamma}||x-y||^2$$

where $\Gamma(X,Y)$ is the set of couplings of X and Y. Let T be the optimal transport plan from X to Y. It's known that in 1 dimension, $T(x) = \phi_Y^{-1}(\phi_X(x))$ where ϕ denotes the CDF. In other words, points with the same CDF are associated with each other.

We can use this to compute the closest Gaussian to P in Wasserstein distance. To do this, let T be the optimal transport plan from $\mathcal{N}(0,1)$ to P. The optimal transport plan from P to $\mathcal{N}(\mu,\sigma)$ can be composed into the plan from $\mathcal{N}(0,1)$ to P and from $\mathcal{N}(0,1)$ to $\mathcal{N}(\mu,\sigma)$. The latter is the affine plan $\sigma x + \mu$. Hence:-

$$W_2(D, \mathcal{N}(\mu,\sigma))^2 = \int (\sigma x + \mu - T(x))^2 d\mathcal{N}(0,1)$$

Setting the derivative w.r.t μ to 0:-

$$\int (\sigma x + \mu - T(x)) d\mathcal{N}(0,1) = \mu - \int T(x) d\mathcal{N}(0,1) = 0$$

By a change of variables, $\int T(x) d\mathcal{N}(0,1) = \int x dP = E_P[x]$. So $\mu = E_P[x]$ as expected.
Setting the derivative w.r.t σ to 0:-

$$\int (\sigma x + \mu - T(x)) x d\mathcal{N}(0,1) = \sigma - \int x T(x) d\mathcal{N}(0,1) = 0$$

So $\sigma = E_{x\sim\mathcal{N}(0,1)}[xT(x)]$. Another interpretation of this result is that if γ is the coupling that achieves the infimum in the definition of W_2, then $\sigma = E_{(x,y)\sim\gamma}[xy]$. Notice that this quantity is tractable in one dimension, since the optimal coupling straightforwardly depends on the inverse CDF of P, which we can efficiently estimate from samples.

Notice that if the P is a Rademacher distribution, then the optimal transport plan moves the negative half of the Gaussian to -1 and the positive half to 1. Hence, $\sigma = \sqrt{\frac{2}{\pi}} \int\limits_0^\infty x e^{-x^2/2} dx = \sqrt{\frac{2}{\pi}}$, which is **not** the variance of P.

The issue with generalizing this approach for higher dimensions is that the optimal transport map doesn't compose anymore, so we can't write the initial integral for it. That being said, if we get the same result in high dimensions, it would be interesting but ultimately useless in practice, since computing $T(x)$ is harder than generative modeling to begin with.

4.2 Attempt to Work in Higher Dimensions

The Wasserstein distance is difficult to work with because it's not as nice in higher dimensions, but also because our noising process adds a Gaussian Z to P, and $W_2(P, P+Z)$ is hard to get a handle on. One thing we know behaves nicely under adding random variables is the characteristic function, so I looked into metrics based on that.

Recall that the characteristic function $\phi_P(k) = E_P[e^{ik^Tx}]$. The Fourier-based metric d_2 is defined as follows:-

$$d_2(P,Q) = \sup_{k\in\mathbb{R}^d\setminus\{0\}} \frac{|\phi_P(k) - \phi_Q(k)|}{||k||^2}$$

In [ACG+20], it's shown that this metric is equivalent to the Wasserstein-2 metric.

Computing the closest Gaussian in this metric still proved difficult. The answer depends on all the moments of P, but Gaussians only allow us to control for the first two. On the bright side, using the Gaussian with the same precision matrix makes sense because its characteristic function matches that of P up to third order information, and we can use that to get a bound on the distance:-

$$\sup_{k\in\mathbb{R}^d\setminus\{0\}} \frac{|\phi_P(k) - \phi_{\mathcal{N}(0,\Sigma)}(k)|}{||k||^2} = \sup_{k\in\mathbb{R}^d\setminus\{0\}} \frac{|(\phi_P(k) - 1 + \frac{1}{2}k^T\Sigma k) - (e^{-\frac{1}{2}k^T\Sigma k} - 1 + \frac{1}{2}k^T\Sigma k)|}{||k||^2}$$

$$= \sup_{k\in\mathbb{R}^d\setminus\{0\}} \frac{|\phi_P(k) - 1 + \frac{1}{2}k^T\Sigma k| + |e^{-\frac{1}{2}k^T\Sigma k} - 1 + \frac{1}{2}k^T\Sigma k|}{||k||^2}$$

Since $1 - \frac{1}{2}k^T\Sigma k \leq |\phi_P(k)| \leq 1$, and the same for $\phi_{\mathcal{N}(0,\Sigma)}(k)$, we can remove the absolute values and upper bound the above quantity by:-

$$\sup_{k\in\mathbb{R}^d\setminus\{0\}} \frac{k^T\Sigma k}{||k||^2} = ||\Sigma||_{op}$$

It should also be possible to get a bound that only uses third moments and higher, but I'm not sure how to do so.

4.3 Working with KL instead

Initially, I didn't want to work with KL because it's not a distance, so we can't really talk about shortest paths. However, since we're solving the simpler problem of finding the closest Gaussian, we can compute the closest Gaussian in KL:-

$$KL(P||\mathcal{N}(\mu,\Sigma)) = E_P[\log P(x)] - E_P[\log \mathcal{N}(\mu,\Sigma)]$$

$$= E_P[\log P(x)] + \frac{1}{2(2\pi)^{d/2}} E_P[\log(|\Sigma|) + (x-\mu)^T\Sigma^{-1}(x-\mu)]$$

Setting the derivative w.r.t μ to 0 gives:-

$$E_P[\Sigma^{-1}(x-\mu)] = 0 \Rightarrow \mu = E_P[x]$$

From here on, we can assume $\mu = 0$. It will also be easier to work with the precision matrix. Setting the derivative w.r.t Σ^{-1} to 0 gives:-

$$-\Sigma + E_P[xx^T] = 0 \Rightarrow \Sigma = E_P[xx^T]$$

So the desired Gaussian is indeed the one with the same mean and covariance.

I also tried studying the problem in TV distance. The approach I took is to get a nearby Gaussian with the tournament method. However, I couldn't find a better cover than the naive exponentially-large one.

I didn't get to think about the original problem as much, since the simpler problem proved to be quite difficult. My conjecture is that all the added noise will be a scaled copy of the target Gaussian that comes from the easier problem, so that's why I set on solving that one first.

5 Mixing Time of Diffusion Models

As part of the project, I surveyed the literature on diffusion model mixing time to understand what can be improved for MRFs. Here's a summary of my findings:-

There was an early large body of work that had relatively strong assumptions, like the data lying on a low-dimensional manifold or something about the log-Sobolev constant. In terms of minimal assumption papers, [CCL+23] showed that the iteration complexity is $\tilde{O}(\frac{dL^2}{\epsilon^2})$ under the assumption that the score is L-Lipschitz. Then, [CLL] removed the assumption and showed an $\tilde{O}(\frac{d^2 \log^2(1/\delta)}{\epsilon^2})$ iteration complexity to sample from a distribution whose noise level is δ. Finally, [BDBDD23] showed $\tilde{O}(\frac{d \log^2(1/\delta)}{\epsilon^2})$ under only the assumptions of L^2-accurate score estimation and a finite second moment.

The key ingredient all these recent papers use is Girsanov's theorem. Girsanov's theorem is a change-of-measure theorem that allows us to do the following [Goo]:-

Suppose P is the measure induced by the SDE $dX = dW$ up to time T, and Q is the measure induced by $dX = \alpha dt + dW$. Then:-

$$\frac{dQ}{dP} = exp(\int_0^T a_t dX_t - \frac{1}{2}\int_0^T a_t^2 dt)$$

So it allows us to compute the ratio of the densities of distributions induced by SDEs. And because of the exponential in the above expression as well as the Ito integral that behaves like a martingale, this theorem is particularly nice for computing the KL divergence between the paths of 2 SDEs.

This idea was exploited in early work by Yang Song himself when he was first developing score-based models in [SE20]. He used it to bound the KL divergence between the reverse-SDE that uses the actual score function and the one that uses the estimated score function (see the proof of Theorem 1). Carrying through the proof shows that the KL divergence is given by:-

$$\frac{1}{2}\int_0^T E_{p_t(x)}[g(t)^2 ||\nabla_x \log p_t(x) - s_\theta(x,t)||_2^2]$$

and so choosing $g(t)^2$ as the weighting function minimizes KL divergence, which wraps up the equivalence between diffusion and score-based models.

The literature that followed uses Girsanov's theorem for a more difficult goal; namely, bounding the KL divergence between the continuous SDE and the discretized version of it. The discrete process does not even follow the conditions of Girsanov's theorem, but the authors massage it to make the theorem work.

As for the final step from quadratic down to linear, the authors appeal to the theory of stochastic localization. Stochastic localization was developed as a tool to study the mixing time of Markov chains, but it awfully looks like diffusion models in its use of reverse time SDEs. Recently, [Mon23] showed that the theories are actually equivalent up to a change of variables. The authors of [BDBDD23] leveraged the strong results from stochastic localization theory to push the iteration complexity down to linear.

Finally, I'd like to argue it's unlikely to improve upon linear, even if the underlying distribution is an MRF. Suppose the distribution is actually d i.i.d copies of a random variable X, and suppose the distribution learned by diffusion is d i.i.d copies of a random variable Y. By tensorization of KL, $KL(X^{\otimes d}||Y^{\otimes d}) = dKL(X||Y)$. So to get an error of ϵ in $KL(X^{\otimes d}||Y^{\otimes d})$, we need $KL(X||Y) \leq \frac{\epsilon}{d}$. So in order to get sublinear iteration complexity, we'd have to improve the $O(\frac{1}{\epsilon})$ dependence for learning in KL.

References

[ACG$^+$20] Gennaro Auricchio, Andrea Codegoni, Stefano Gualandi, Giuseppe Toscani, and Marco Veneroni. The Equivalence of Fourier-based and Wasserstein Metrics on Imaging Problems, May 2020. arXiv:2005.06530 [math-ph, stat].

[BDBDD23] Joe Benton, Valentin De Bortoli, Arnaud Doucet, and George Deligiannidis. Linear Convergence Bounds for Diffusion Models via Stochastic Localization, August 2023. arXiv:2308.03686 [cs, stat].

[CCL$^+$23] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R. Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions, April 2023. arXiv:2209.11215 [cs, math, stat].

[CLL] Hongrui Chen, Holden Lee, and Jianfeng Lu. Improved Analysis of Score-based Generative Modeling:User-Friendly Bounds under Minimal Smoothness Assumptions.

[Goo] Jonathan Goodman. Week 10 Change of measure, Girsanov formula.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, December 2020. arXiv:2006.11239 [cs, stat].

[Hyv] Aapo Hyvarinen. Estimation of Non-Normalized Statistical Models by Score Matching.

[Mon23] Andrea Montanari. Sampling, Diffusions, and Stochastic Localization, May 2023. arXiv:2305.10690 [cs].

[SE20] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution, October 2020. arXiv:1907.05600 [cs, stat].

[SHC$^+$21] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Image Super-Resolution via Iterative Refinement, June 2021. arXiv:2104.07636 [cs, eess].

[SSDK$^+$21] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, February 2021. arXiv:2011.13456 [cs, stat].

[TSM$^+$20] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, June 2020. arXiv:2006.10739 [cs].

[WT] Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics.