Reviewing the Paper "Generative modeling by estimating gradients of the data distribution" and Extending the Framework for A Multi-Resolution Diffusion Generative Model

Dominic Padova and Zixuan Liu

05/15/2023

1 Introduction

Machine learning has benefited from having high quality data. But collecting and cleaning data to get high quality can be complicated and expensive in terms of time and monetary cost. Furthermore, when the data generation setting changes, often new data must be collected and cleaned to accommodate the new setting and new constraints; this compounds issues stemming from the data collection process. However, estimating the data generation process from existing data, then simulating new data based on the estimated data generation process could alleviate some of the costs of high quality data collection. Estimating the data generation process often amounts to estimating an unknown and complicated data distribution, say of images of dogs, using a statistical model, called a generative model. The statistical model assigns a probability to data points, like an image of a chihuahua or an image of a blueberry muffin, and builds the model from these assignments. Once this model is estimated, new data can be produced by sampling from the generative model. This is the idea behind generative modeling (see Figure 1).

2 Paper Review: Generative modeling by estimating gradients of the data distribution

2.1 What is generative modeling?

Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning regularities or patterns in input data so that the model can be used to generate or output new examples that may be drawn from the original dataset.

2.2 A key challenge for building complex generative models

One main challenge of building a generative model from data is that our data distribution can be extremely complicated, especially for data with high dimensions. So consider how complicated it might be for the distribution of images, video, and audio. It might have millions of dimensions. So in our result, we need to build a complex model to fit the data

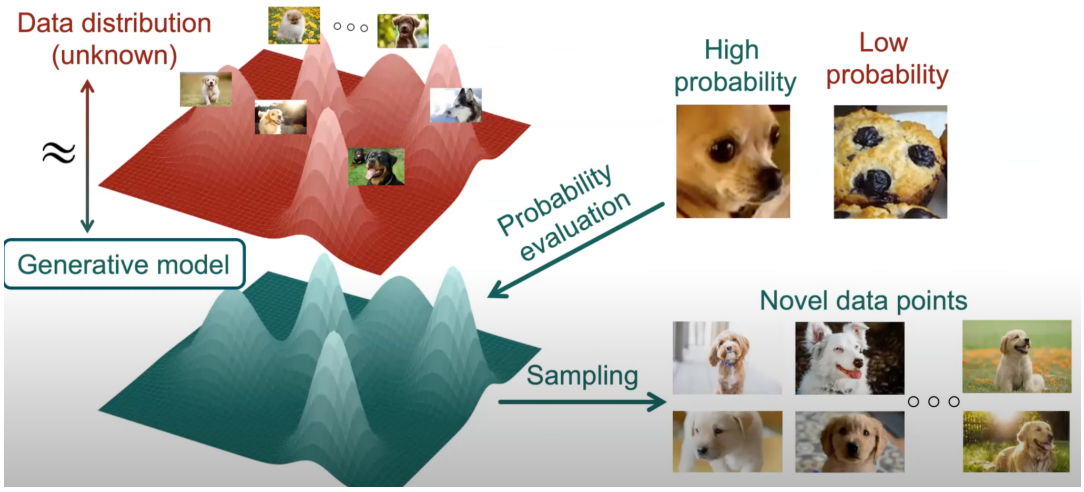


Figure 1: **Generative modeling.** Generative modeling is performed by estimating the data distribution using a statistical model that assigns probabilities to data to build the model, and then produces new data by sampling from the model. Image credit: Song, Y. `https://www.youtube.com/watch?v=nv-WTeKRLl0&t=3174s` (oral presentation).

distribution. Suppose we have a continuous probability distribution where we use $p(x)$ to represent the probability density function, we define the score function as the gradient of $\log p(x)$. Given the density function, we can compute the score function very easily because we can just take the derivative of the logarithm:

$$p(x) = \frac{1}{Z(\theta)} e^{-E(x,\theta)}$$

$$\log(p(x)) = -E(x,\theta) - \log(Z(\theta))$$

$$\underbrace{\nabla_x \log(p(x))}_{\text{Stein score function}} = \underbrace{-\nabla_x E(x,\theta)}_{\substack{\text{Neg. gradient of energy} \\ \text{function w.r.t. data}}}$$

Conversely, with the score function, we can also recover the density function in principle by computing integrals. So mathematically, this function preserves all the information in the density function. But computationally, this score function is much easier to work with compared to the density function. Intuitively, the Stein score function is the vector field that points in the direction where the (log) probability grows the fastest. Therefore, it captures information about the data distribution.

2.3 Score-matching

Mathematically, we are given a bunch of data points which as assumed to be i.i.d. sampled from the data distribution $p(x)$, and our goal is to estimate this score function of the data density. So how can we train this model to be close to our ground truth data score function?

One approach is to minimize an objective function that seeks to bring two vector fields close together: the ground truth score function and an estimate of the score function. This approach is called score matching. How can we compute the score matching objective? Mathematically, we can capture this with the Fisher divergence objective. However, the fisher divergence cannot be directly computed because we do not know the ground truth value of the score function. Still, there is a way to address this challenge, using an two approaches called denoising score matching and sliced score matching:

- Explicit (Fisher Divergence-based) score matching

$$\frac{1}{2}\mathbb{E}_{p_{data}(x)}[\| \underbrace{\nabla_x \log\left(p_{data}(x)\right)}_{\text{Fisher divergence: unknown}} - s_\theta(x)\|_2^2] \tag{1}$$

- Denoising score matching

$$\frac{1}{2}\mathbb{E}_{\underbrace{q_\sigma(\tilde{x} \mid x)p_{data}(x)}_{q_\sigma(\tilde{x})\text{: noised data dist.}}}[\|s_\theta(\tilde{x}) - \underbrace{\nabla_{\tilde{x}} \log\left(q_\sigma(\tilde{x} \mid x)\right)}_{\substack{\text{Gradient of Gaussian noise kernel,} \\ \text{conditioned on original data}}}\|_2^2] \tag{2}$$

- Sliced score matching

$$\mathbb{E}_{p_v}\mathbb{E}_{p_{data}}[\underbrace{v^T \nabla_x s_\theta(x) v}_{\substack{\text{Random projections onto} \\ \text{Jacobian of score model}}} + \frac{1}{2}\|s_\theta(x)\|_2^2] \tag{3}$$

where the sliced score matching objective is achieved by applying the Divergence theorem to the Explicit (Fisher divergence-based) score matching objective to achieve the Implicit score matching objective

$$\mathbb{E}_{p_{data}(x)}[\frac{1}{2}\|s_\theta(x)\|_2^2 + \underbrace{\text{trace }(\nabla_x s_\theta(x))}_{\text{divergence of } s_\theta(x)}]$$

and using an efficient random projection estimator to estimate the trace($\nabla_x s_\theta(x)$) divergence term.

2.4 Langevin dynamics

Langevin dynamics can produce samples from a probability density $p(x)$ using only the score function $\nabla_x \log p(x)$ (See Figure 2). Given a fixed step size $\epsilon > 0$, and an initial value $\tilde{x}_0 \sim \pi(x)$ with π being a prior distribution (e.g. a Uniform distribution), the Langevin method recursively computes the following

$$\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2}\nabla_x \log\left(p(\tilde{x}_{t-1})\right) + \sqrt{\epsilon} z_t, z_t \sim N(0, Id), \epsilon > 0 \tag{4}$$

This procedure can be interpreted as a noisy gradient ascent. The noise here allows the samples to be distributed around the peaks instead of directly on the peaks. This allows better recovery of the true distribution.

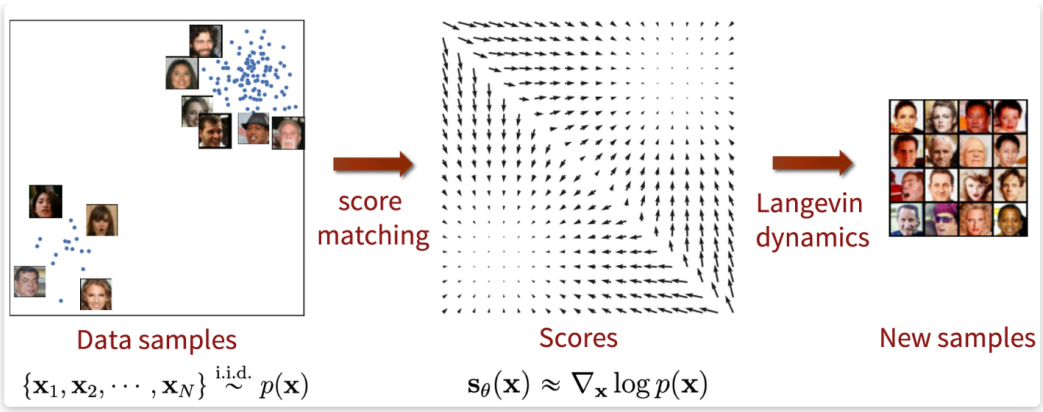


Figure 2: **Langevin dynamics.** New samples are produced using the estimated score model and Langevin dynamics. Image credit: [1].

2.5 Challenges of score-based generative modeling

- The manifold hypothesis

  Since the score function is a gradient taken in the ambient space, it is undefined when x is confined to a low dimensional manifold.
  The score-matching objective provides a consistent score estimator only when the support of the data distribution is the whole space (See Figure 3).

- Inaccurate score estimation with score matching

  In regions of low data density, score matching may not have enough evidence to estimate score functions accurately, due to a lack of data samples (See Figure 4).

- Slow mixing of Langevin dynamics When two modes of the data distribution are separated by low-density regions, Langevin dynamics will not be able to correctly recover the relative weights of these two modes in a reasonable time, and therefore might not converge to the true distribution (See Figure 5).

2.6 Adding noise to the data and annealing the Langevin sampling solves the challenges

Perturbing data with random Gaussian noise makes the data distribution more amenable to score-based generative modeling. When using Langevin dynamics to generate samples, initially use scores corresponding to large noise models, and gradually anneal down the step size based on the noise level. These two additions allow score matching models to estimate the true score function, effectively passing the benefits from estimating large noise models down to the low noise models, and allow the annealed Langevin dynamics process to correctly recover the relative weights of the modes separated by low data density regions (See Figure 6).

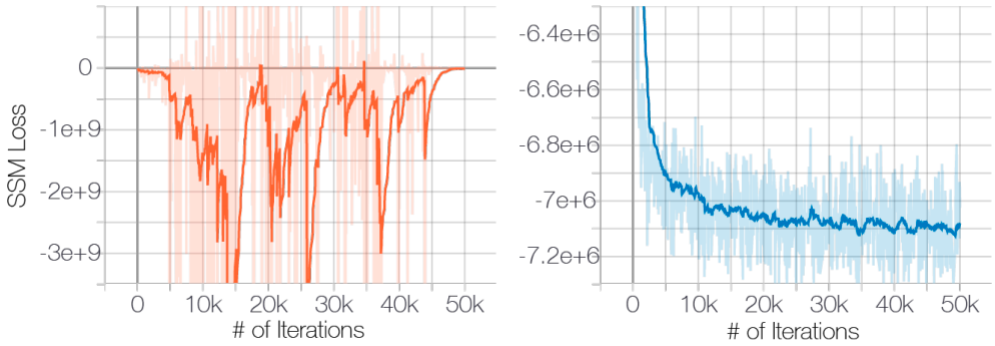


Figure 3: **Manifold Hypothesis Challenge.** Assuming the data lies on a low-dimensional manifold (the Manifold Hypothesis) presents a challenge for a score-based model to estimate the score function, since the score depends on the gradient of the full ambient space, not a gradient along a low-dimensional manifold. Whereas the sliced score matching (SSM) loss function under the Manifold Hypothesis (left) shows the model has trouble learning the score function, the SSM loss after injecting noise into the data to support the data on the full ambient space shows a marked improvement in learning ability. Image credit: [1].

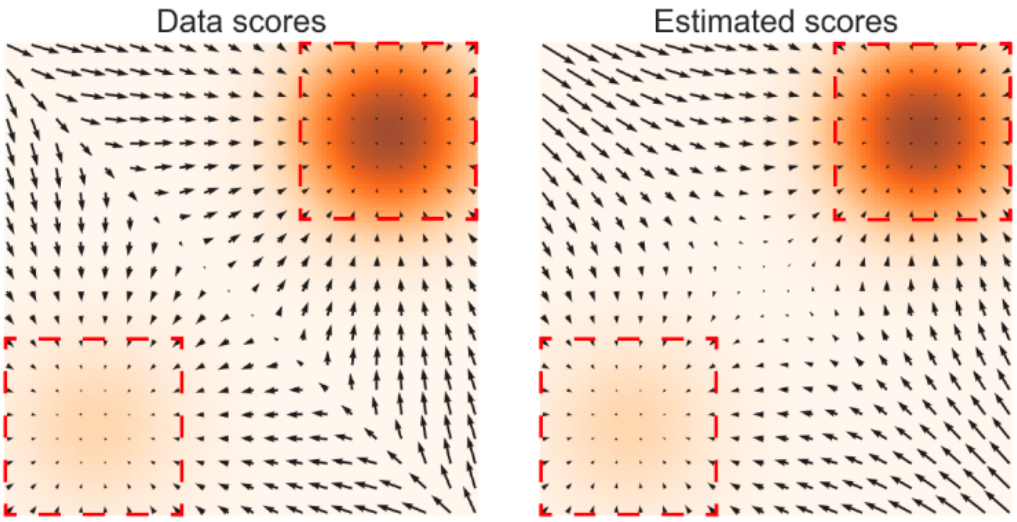


Figure 4: **Inaccurate Score Estimation Challenge.** Score matching suffers in low density regions of the data distribution, due to a dearth of samples. The true data scores (left) in the low data density region (bottom red square) are not recapitulated by the estimated scores (right). Image credit: [1].

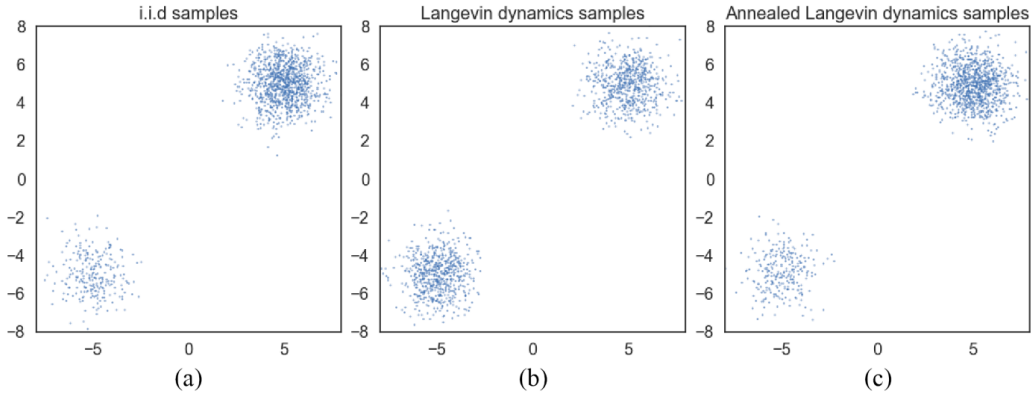


Figure 5: **Slow Mixing of Langevin Dynamics Challenge.** Langevin Dynamics suffers from the slow mixing problem, where the process may not recover the relative weights of modes are separated by low density regions of the data distribution. Sample i.i.d. from an arbitrary distribution (a), use Langevin dynamics to push samples towards modes, where the relative weights are the same, thus not converging to the true data distribution (b), and annealing the Langevin dynamics recovers the proper weights (c). Image credit: [1].

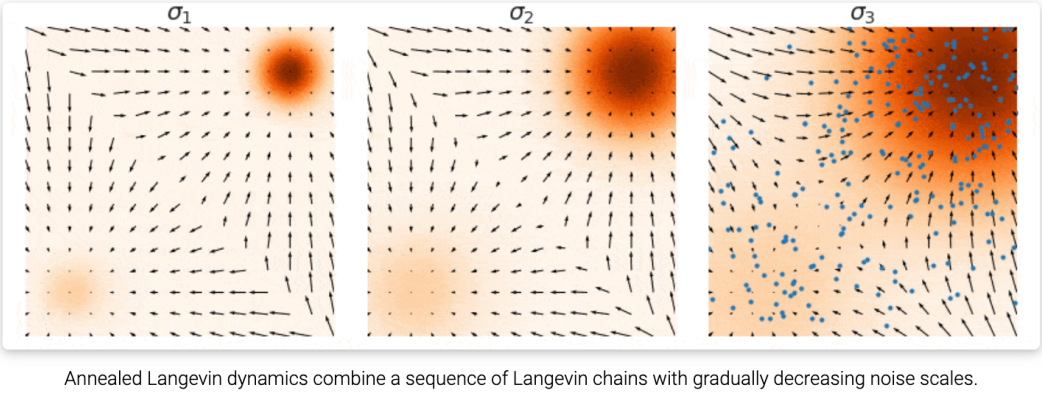


Figure 6: **Annealed Langevin Dynamics.** Learning noise-conditional score models and then using a sequence of Langevin dynamics processes, starting with the large noise score models and sequentially using the lower noise score models solves the three challenges of score based modeling. [1].

Model	Inception	FID
CIFAR-10 Unconditional		
PixelCNN [59]	4.60	65.93
PixelIQN [42]	5.29	49.46
EBM [12]	6.02	40.58
WGAN-GP [18]	$7.86 \pm .07$	36.4
MoLM [45]	$7.90 \pm .10$	**18.9**
SNGAN [36]	$8.22 \pm .05$	21.7
ProgressiveGAN [25]	$8.80 \pm .05$	-
NCSN (Ours)	$\mathbf{8.87} \pm .12$	25.32
CIFAR-10 Conditional		
EBM [12]	8.30	37.9
SNGAN [36]	$8.60 \pm .08$	25.5
BigGAN [6]	**9.22**	**14.73**

Table 1: Inception and FID scores for CIFAR-10

Figure 7: **Inception and FID Scores for CIFAR-10.** The noise conditional score network pipeline achieves the highest Inception score on the unconditional CIFAR-10 image generation task and a low FID score; these metrics are competitive with the best performing GANs and method-of-moments-trained networks (MoLM). Image credit: [1].

2.7 Results

Table 7 clearly shows the comparison results with different models evaluated by Inception and FID two metrics. Inception is a metric to evaluate the model performance and compare the accuracy and training time. FID is a measure of the similarity between two sets of images. It is commonly used as a metric to evaluate the performance of generative models. A lower FID score indicates that the generated images are more similar to the real images, and thus the generative model is considered to perform better.

This paper built models on three different datasets, which are MNIST, CelebA, and CIFAR-10, and Figure 8 shows the new generating samples generated from the generative model. Through the image results, we can clearly see that the generated model has a good effect, which is not much different from the data points in the original data set.

3 Conclusion of Paper Review

Estimating the data distribution, which is unknown and high-dimensional, is hard. It requires calculating the partition function is hard or intractable in general. Rather, estimating a score-based neural network model $s_\theta(x) \approx \nabla_x \log p(x)$ from data avoids this issue and is tractable via score-matching. Samples can be produced using Langevin dynamics, an MCMC procedure that samples from the data distribution $p(x)$ using only its score function with added noise. This process that can be interpreted as noisy gradient ascent. The noise here allows the samples to be distributed around the peaks instead of directly on the peaks. This allows better recovery of the true distribution. Noising the data and annealing (i.e. gradually lowering the learning rate of) the Langevin dynamics greatly assists the score estimation and sample generation process.

Thus, score-based generative modeling avoids calculation of partition function by relying on the Stein score function, benefits from injecting noise into the data, avoids the need to sample from a Markov chain during training, is performed in two, decoupled stages: (1) noising, (2)

(a) MNIST (b) CelebA (c) CIFAR-10

Figure 5: Uncurated samples on MNIST, CelebA, and CIFAR-10 datasets.

Figure 8: **New samples on MNIST, CelebA, and CIFAR-10.** The noise conditional score network and annealed Langevin dynamics pipeline produces new samples the resemble the data on which they were trained. Image credit: [1].

sample generation, and can be used to train energy-based models by using the gradient of an energy-based model as the score model.

4 Extending the Score-Based Generative Modeling Framework with a Multi-Resolution Diffusion Generative Model

Since adding noise improve score-based generative modeling performance, and adding more noise from the same family improves performance even more, then adding an infinite number of noise perturbations from that same family may improve the model performance even more. This idea of appealing to a continuous noising process places score-based generative models into the continuous diffusion with drift regime [2]. These models are referred to as generative diffusion models.

Generative diffusion models perform well, but training them and sampling from them can be slow. One way to improve the speed of training and performance is to introduce a multi-resolution inductive bias into the model. This multi-resolution inductive bias incorporates into the model the idea that images have objects of different scales and have locally varying statistics. Rather than train a single diffusion model on a single resolution of images, which is slow and can get stuck in local optima, we aim to train a single diffusion model for each resolution, where the coarser models inform the learning process of the higher resolution models, speeding up the learning process for higher resolutions and avoiding getting caught in inferior local optima. Furthermore, using a multi-resolution representation based on residual images could further improve the efficiency of the model by reducing redundancy of information at neighboring resolutions. A multi-resolution model could therefore benefit from the simplicity and efficiency of the low-resolution model as well as the efficiency and detailed accuracy of the higher-resolution residual models that provide insight into how the image information changes between resolutions.

5 Methods

5.1 Laplacian Pyramid Representation of Images

Let an image be defined on the domain $X \subset \mathbb{R}^n$ by the mapping $I : X \to \mathbb{R}^{n \times c}$ where n is the dimension of the ambient space (here this means the number of pixels in a channel) and c is the number of channels. In this report, we will work with 3-channel RGB images of size 32×32, and therefore $n = 1024$ and $c = 3$. Given an image $I(x)$, we wish to generate a linear, invertible, multi-resolution image representation as a series of k images of successively lower resolution

$$L : X \to (I_0(x), I_1(x), \ldots, I_k(x)) \in \mathbb{R}^{n \times c} \tag{5}$$

$$X \mapsto (X_0, X_1, \ldots, X_k) \in \mathbb{R}^{n \times c} \tag{6}$$

$$X_k \subset X_{k-1} \subset \ldots \subset X_0 \tag{7}$$

where X_0 is the domain of the finest resolution image and each subsequent X_{i+1} is the blurred and downsampled image of Let $d(\cdot)$ be a downsampling operator that blurs and decimates an input image $I(x)$ of size $\ell \times \ell$ with $\ell \in \mathbb{Z}$ so that $d(I)(x)$ is a new image of size $\ell/2 \times \ell/2$. Also, let $u(\cdot)$ be an upsampling operator which smooths and expands $I(x)$ to be twice the size, so is a new image $u(I)(x)$ of size $2\ell \times 2\ell$. Here, we choose $d(\cdot)$ to be an average pooling (i.e. windowed averaging with a 2×2 kernel and a 2×2 stride) operation operating on each channel individually. We choose $d(\cdot)$ to be a in-plane bilinear interpolation function. Therefore, the pyramid is constructed by

$$\begin{cases} I_0(x) = I(x) \\ r_i(x) = d^{i+1}(I)(x) - u^i(d^{i+1}(I))(x) \text{ for } 0 \leq i \leq k-1 \\ I_k(x) = d^k(I)(x) \end{cases} \tag{8}$$

Thus, the Laplacian pyramid image representation of an image is then given by the series

$$L(x) = \{I_0(x), r_0(x), \ldots, r_{k-1}(x), I_k(x)\} \tag{9}$$

where k is the number of levels (i.e. number of resolutions) of the pyramid, $I_0(x)$ is the highest-resolution image, $I_k(x)$ is the lowest-resolution image, and the sequence of images $r_{0:k-1}(x)$ are residuals. See Figure 9 for an illustration. To achieve the original image from the Laplacian pyramid, we can reverse the recursive process, which yields the linear relation:

$$I_{k-1} = u(I_k) + r_{k-1} \implies I_0 = \sum_{i=0}^{k-1} u^i(I_i) + r_i. \tag{10}$$

There are many advantages to this representation: (i) the process is linear, (ii) it is invertible, (iii) it reduces redundancy since we work not only with the low frequency shape information but also with the and high frequency texture information not present in low resolution images through the residual images, (iv) the residual images are sparse and should be easier to forward and backwards transform.

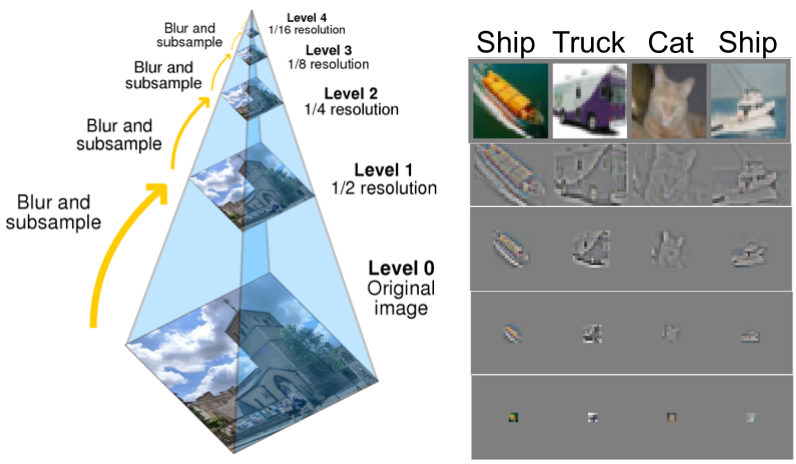


Figure 9: **Laplacian Pyramid Image Representation.** An illustration of the Laplacian pyramid (left) and a grid of Laplacian pyramid images representation of four images sampled from CIFAR-10 (right). Cartoon image credit: `https://en.wikipedia.org/wiki/Pyramid_(image_processing)#/media/File:Image_pyramid.svg`

5.2 Laplacian Pyramid Diffusion

For each of the i.i.d. samples in our training dataset, we generate the k-level Laplacian pyramid representations, then we train k parallel diffusion models. The ith diffusion model is trained on the ith-level images. One can choose the number of scales $k = \log_2 \ell$, which for a $\ell = 32$ image, $k = 5$, but instead we chose $k = 3$. This may produce a floor bias, where there is an implicit assumption that using more levels of the pyramid will not bring useful information to the table.

Following Song et al. 2020 [2], we construct a diffusion process $\{x(t)\}_0^T$ for a data sample x and $t \in [0, T]$ where T is fixed and finite[1] such that the initial data distribution $x(0) \sim p_0$ of i.i.d. samples is flowed forward in time with transition kernel $p_{s,t}(x_t \mid x_s)$ (a Gaussian function) to the final data distribution $x(T) \sim p_T$, which is taken to be a standard Normal distribution. The forward and backward diffusion processes are modeled by forward and backward Ito equations, respectively, where the backward process is achieved by estimating the time-dependent score function.

Now, the forward and backward Ito equations for the level $i \in 0, \ldots, k$ images of the Laplacian pyramid are given by

$$\begin{cases} dx_i &= f_i(x_i, t)dt + g_i(t)dw_i \text{ (Forward Ito SDE)} \\ dx_i &= (f_i(x_i, t) - g_i^2(t)\nabla_{x_i} \log p_t(x_i))dt + g_i(t)d\bar{w}_i \text{ (Reverse Ito SDE)} \end{cases} \tag{11}$$

where $f_i(t, x_i) : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}$ are affine drift coefficients taking d_i-dimensional real vectors for the dimension of the ith level to d_i-dimensional real vectors for any time t, $g_i(t) : [0, T] \to \mathbb{R}$. Since we use affine drift coefficients, we achieve Gaussian perturbation kernels. The authors suspect, but do not provide proof for, the idea that $f_0(t, x_0) \leq f_1(t, x_1) \leq \ldots \leq f_k(t, x_k)$ and $g_0(t, x_0) \leq g_1(t, x_1) \leq \ldots \leq g_k(t, x_k)$, means that the i-th diffusion model is on a faster time

[1]The model has a solution $T \to +\infty$, but we choose a T as a finite real number that is "large enough".

scale and therefore learns faster. The authors suspect the speed of training is related to the number of scales k, and that the smallest resolution should be the fastest.

The time-dependent score model is estimated by finding the optimal neural network parameters θ^* which minimize the continuous denoising score matching loss function:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{x(0)} \mathbb{E}_{x(t)|x(0)} \left[\| s_\theta(t, x(t)) - \nabla_{x(t)} \log \left(p_{0,t}(x(t) \mid x(0)) \right) \|_2^2 \right] \right\}, \tag{12}$$

where $\lambda > 0$ for all t so that $s_{\theta^*}(t, x(t)) \approx \nabla_x \log p_t(x(t)))$.

We want to express the likelihood of each level of the Laplacian pyramid as a function of higher level of the pyramid data (i.e. the lower resolution data x_{i+1}) and the lower level of the pyramid residual (i.e. the higher resolution residual data r_i). Then this will show the likelihood relations for each individual model, which can be used independently or in tandem.

$$p(x_k) = p(r_{k-1}, x_k) \tag{13}$$

$$p(x_{k-1}) = p(r_{k-2}, x_{k-1}) \tag{14}$$

$$\vdots \tag{15}$$

$$p(x_0) = p(r_0, x_1) \tag{16}$$

While the operations to build the Laplacian pyramid are deterministic, and therefore do not produce a random variable for single images, the data distribution for each level of the pyramid is not guaranteed to be deterministic.

Using the likelihood at each level, we can get the likelihood of the original, highest resolution data. The distribution of the highest resolution data is related to the joint distribution over the residual data and lowest resolution data from the Laplacian pyramid. This can be re-expressed using the chain rule of probability as a product of conditional distributions conditioned on the next lowest resolution data x_{i+1} and the higher resolution residual data r_i as follows:

$$p(x_0) = p(r_0, \ldots, r_{k-1}, x_k) \tag{17}$$

$$= p(r_0) p(r_1 \mid r_0) \cdots p(r_{k-1} \mid r_0 \ldots r_{k-2}) p(x_k \mid r_0 \ldots r_{k-1}) \tag{18}$$

$$= p(x_k \mid r_0 \ldots r_{k-1}) \prod_{i=0}^{k-1} p(r_i \mid r_0 \ldots r_{k-2}) \tag{19}$$

Taking the logarithm, we achieve the log-likelihood

$$\log p(x_0) = \log p(x_k \mid r_0 \ldots r_{k-1}) + \sum_{i=0}^{k-1} \log p(r_i \mid r_0 \ldots r_{k-2}) \tag{20}$$

Taking the gradient we now have an expression in terms of Stein scores:

$$\nabla_x \log p(x_0) = \nabla_x \log p(x_k \mid r_0 \ldots r_{k-1}) + \sum_{i=0}^{k-1} \nabla_x \log p(r_i \mid r_0 \ldots r_{k-2}) \tag{21}$$

This last expression gives us a linear function to produce score functions at the original data resolution using the score functions of the other elements of the Laplacian pyramid. The full resolution score model may be recovered from the other pyramid level score models by smoothing and upsampling, i.e. by successively applying the $u(\cdot)$ operator to upsample the gradient vector fields and adding them to the next highest level score model. However, the authors do not provide proof here. Note that the forward and reverse likelihoods can be achieved from one another using Bayes Rule.

5.3 Model Intuition

Intuitively, one can appeal to classical mechanical models to understand the behavior of diffusion under drift. For instance, imagine that you are throwing a ball to a friend when there is no wind. The ball is given an initial velocity and it shoots through the air (following a parabola) and lands in your friend's glove. This trajectory is the marginal ODE: because there is no wind or random forces acting on the ball to change its trajectory, the trajectory is determined completely by the initial velocity.

Now imagine that you throw the ball on a windy day. When there is a steady gust of wind acting on the ball while it travels through the air, the trajectory changes. The wind modifies the velocity of the ball by a constant magnitude and by a change in the direction. This is the marginal ODE with drift.

Now imagine that you throw the ball on a really windy day. Wind is gusting with different strengths in different directions, randomly perturbing the trajectory of the ball as it sails through the air. This is the stochastic ODE: the trajectory of the ball is no longer determined by the initial velocity alone but also on the random forces that act on the ball, changing its trajectory. How will you know where the ball will go if it is subject to randomness? The best you can say is the expected terminal point.

To reverse the process, one must follow the negative trajectory backwards in time.

5.4 Training

Training can be performed in separate stages in parallel or end-to-end using a cascade model. In the separate stage model, the lowest resolution model is trained at the same time as the next highest level model, and so on, so each level is trained for its specific resolution and residual, while the weights of the previous level are fixed and separate from the other models. In the cascade model, the weights are not fixed but update with full passes through the network.

There are three experiments that can achieve these training pipelines:

- Train only the lowest resolution and compare.
- Train each level of the pyramid separately and compare.
- Train each level of the pyramid simultaneously in one model and compare.

However, in the work, we only accomplish the separate training of each level and generate images unconditionally (i.e. not conditioned on the previous levels of the pyramid).

5.5 Image Generation

Given a k-level Laplacian pyramid representation of the data, we can sample from the Gaussian noise distribution at each level and use the reverse Ito SDE to flow from the noise distribution to produce new samples at each level of the pyramid. This approach was done using a black-box ODE sampler. However, the conditional sampling approach could be achieved using a annealed Langevin dynamics conditioned on the previous levels of the pyramid. This approach in theory would sequentially build up a high resolution image by updating the sample state according the pyramid-level score model.

Algorithm 1 Annealed Conditional Langevin MCMC

Require: σ, ϵ, T, k
  Initialize $\tilde{x}_0 \sim U(0, 1)$
  for $i \leftarrow 1$ `to` k **do** ▷ Loop over pyramid levels
    $\alpha_i \leftarrow \epsilon \cdot \sigma^2/2^i$ ▷ Anneal step size based on pyramid level
    for $t \leftarrow 1$ `to` T **do**
      `Draw` $z_t \sim N(0, Id)$
      $\tilde{x}_t \leftarrow \tilde{x}_{t-1} + \frac{\alpha_i}{2} s_\theta(\tilde{x}_{t-1}, t-1) + \sqrt{\alpha_i} z_t$
    end for
    $\tilde{x}_0 \leftarrow \tilde{x}_T$
  end for
  return $\tilde{x}_T$

The sample produced at the final time should have the properties of the last level of the pyramid. In our case, this should be the full resolution image. This should produce high resolution samples quickly, since each iteration over the levels of the pyramid is conditioned on the final state of the sample at the last level of the pyramid.

5.6 Data

- Dataset: CIFAR-10 ($3 \times 32 \times 32$, 10 classes, 60k RGB images)
- Denoising: unconditional; score-matching
- Sampling: unconditional
- Task: image generation

5.7 Results on CIFAR-10: Image Generation

The results of training experiments number 1 and number 2 can be seen in Figures 10 and 11. The loss curves show that the models do learn the score function at each level of the pyramid, as indicated by the rapidly decreasing average loss values. The new samples produced are uncurated images of each level of the pyramid. However, the images are sampled unconditionally to one another. The Google Collab Notebook which modifies Dr. Yang Song's tutorial to achieve these results can be found here: `https://colab.research.google.com/drive/1EK8SpL6OIRbU64iyp47tmARXhmrvSOEq#scrollTo=zX1_hSXpKO9R`.

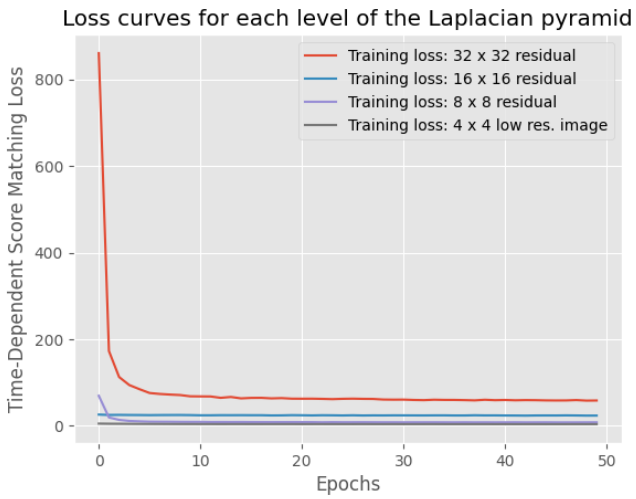
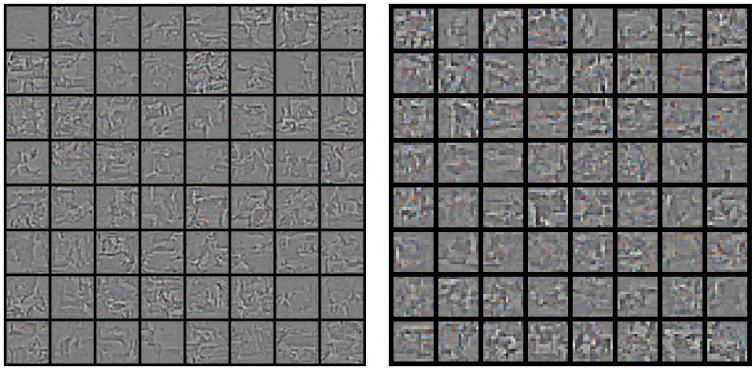
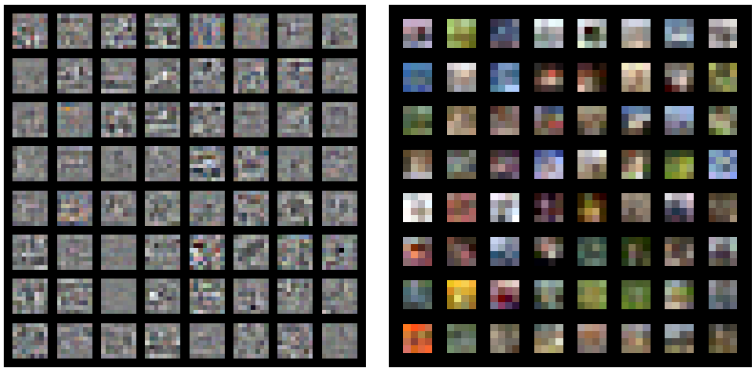


Figure 10: **Training loss curves for the Laplacian pyramid diffusion models.** Four distinct diffusion models, one model per level of the pyramid, were trained separately and the average time dependent score matching loss during training is plotted against the training time (50 epochs).



(a) 32x32 Residual Image Samples (b) 16x16 Residual Image Samples



(c) 8x8 Residual Image Samples (d) 4x4 Low Res. Image Samples

Figure 11: **New Laplacian pyramid samples on CIFAR-10.** New samples generated at each level of the Laplacian pyramid after training 4 distinct diffusion models in parallel and sampling using a black-box ODE solver.

5.8 Discussion and Future Work

Preliminary experiments are inconclusive, as only one distinct model was fit for each Laplacian pyramid image distribution, and sampling was performed unconditionally, i.e. distinct from the other levels of the pyramid. Future work will involve conditional sampling, conditioned on the previous level of the pyramid, and training a cascaded model, where training is conditionally performed as well as conditional sampling. Training the cascaded model may be accomplished by using a sum of denoising score matching terms, summed over the levels of the Laplacian pyramid, and conditional training may be performed by embedding a new layer into the score network for each level of the pyramid. Furthermore, the authors believe this model can be extended for class conditional sampling, by embedding class labels as a layer in the score network. Finally, the authors would like to use optimization and sampling methods from optimal control (e.g. shooting method for training, Hamiltonian Monte Carlo for sampling). These ideas are expounded below:

- Biased and conditional sampling
  Each class should get its own drift which biases the trajectory, which could be considered analogous to the slope term in linear regression, and each class gets its own diffusion term. Also, the diffusion models share a background space and so the drifts and diffusions should be added.

$$\begin{cases} dx &= \sum_{j=1}^{C} f_j(x,t)dt + \sum_{j=1}^{C} g_j(t)dw \text{ (Biased Forward Ito SDE)} \\ dx &= \sum_{j=1}^{C} (f_j(x,t) - g_j^2(t)\nabla_x \log p_t(y \mid x))dt + \sum_{j=1}^{C} g_j(t)d\bar{w} \text{ (Biased Reverse Ito SDE)} \end{cases} \tag{22}$$

- Use Optimization Methods from Optimal Control
  Examples that could be used at least in the marginal setting (i.e. drift only) are the single/multiple shooting methods or the adjoint sensitivity methods. The shooting methods could reduce the dimensionality of the problem by formulating the problem in terms of the initial parameters. This means samples can be generated by sampling from the distribution of initial momenta and shooting the initial sample through time

and space according to the probability flow ODE.

- Hamiltonian Monte Carlo sampling conditioned on class and on images

- Introduce time-optimality condition in the cost function.
  The diffusion problem has a solution on the infinite horizon in time. In practice, one can only approximate this with finite terminal time T. By leaving T unfixed, on can solve for the T that satisfies some time optimality condition in the optimization procedure. This provides a principled tuning knob balancing the trade-off between the accuracy of the solution and the time it takes to train the model.
- Introduce penalty terms on the initial and terminal conditions for tunable, approximate matching.

References

[1] Yang Song and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution". In: *Advances in neural information processing systems* 32 (2019).

[2] Yang Song et al. "Score-based generative modeling through stochastic differential equations". In: *arXiv preprint arXiv:2011.13456* (2020).