CSCI 699 - ProbGen

Probabilistic and Generative Models

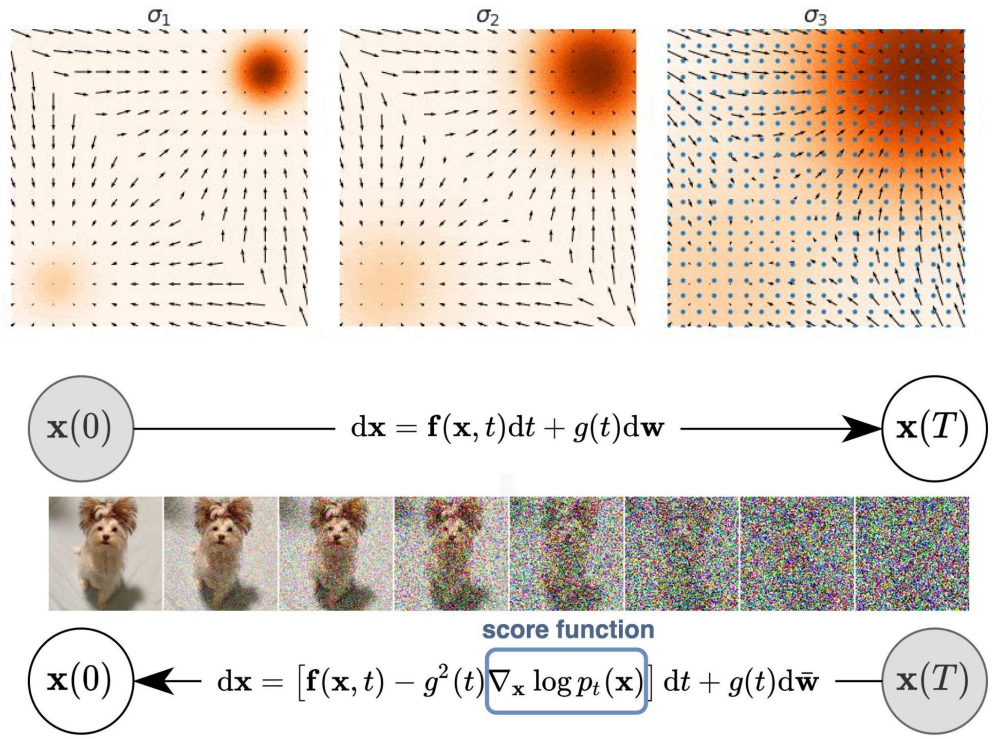
Willie Neiswanger

Lecture 8 - Score-based and Diffusion Generative Models

Today

Today

Lecture: Score-based generative models, and intro to diffusion models.

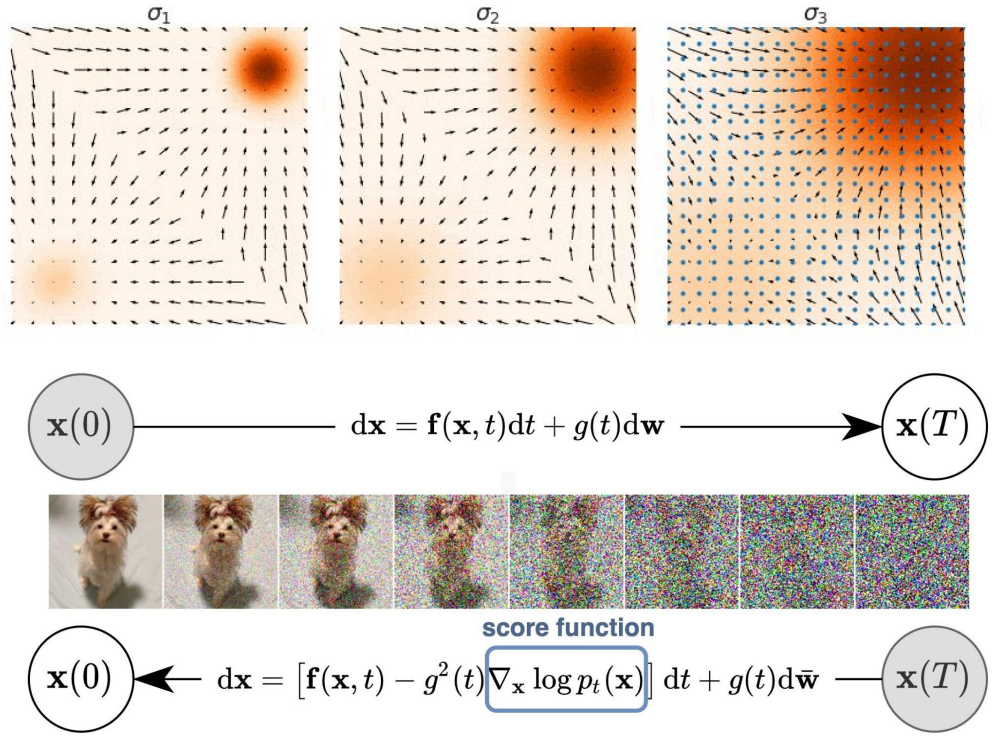


Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

Today

Lecture: Score-based generative models, and intro to diffusion models.

- Score-based Generative Models
  - Modeling the (Stein) Score Function
  - Annealed Langevin Monte Carlo

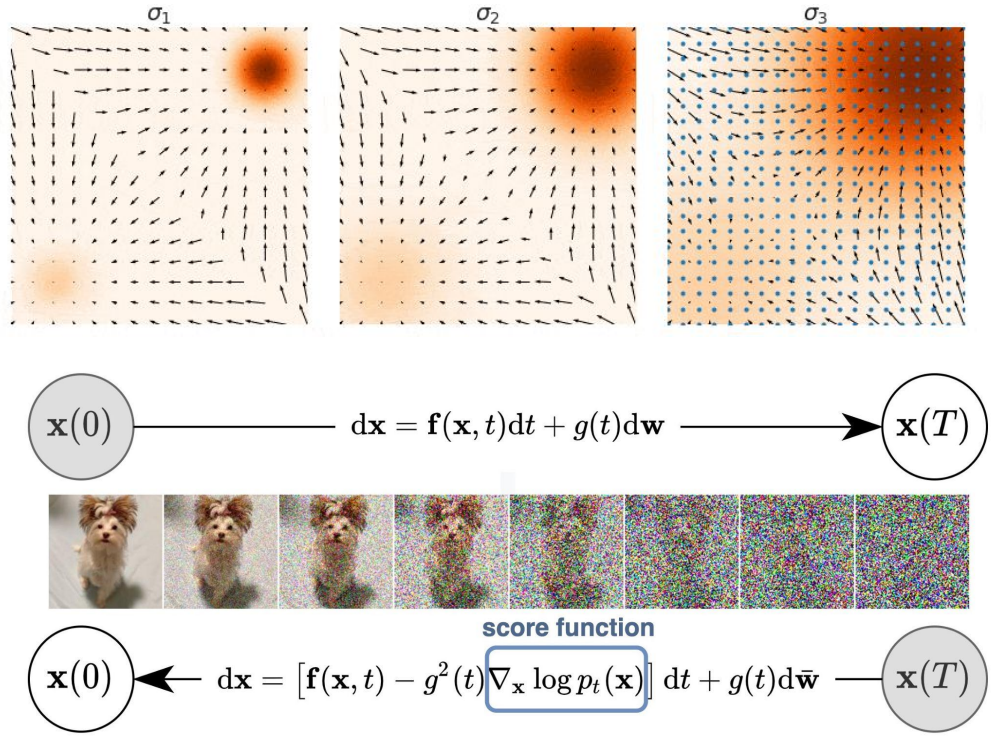


Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

Today

Lecture: Score-based generative models, and intro to diffusion models.

- Score-based Generative Models
  - Modeling the (Stein) Score Function
  - Annealed Langevin Monte Carlo
- Framing as a diffusion model.
- VI for diffusion models.



Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

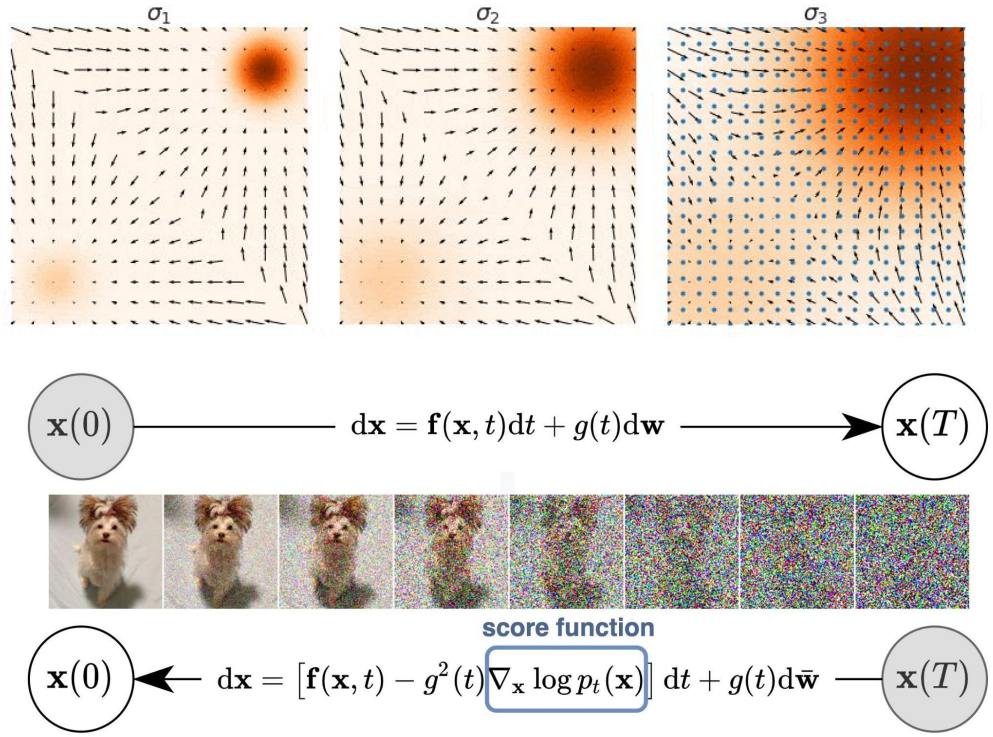
Today

Lecture: Score-based generative models, and intro to diffusion models.

- Score-based Generative Models
  - Modeling the (Stein) Score Function
  - Annealed Langevin Monte Carlo
- Framing as a diffusion model.
- VI for diffusion models.

After:

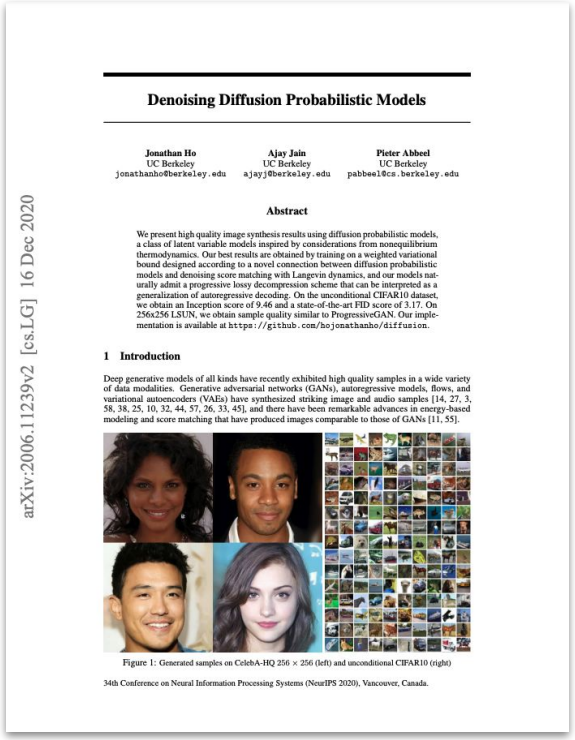
- Second batch of four **paper presentations**.



Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

Today

After: Second batch of four **paper presentations**.



Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Jukebox: A Generative Model for Music

Prafulla Dhariwal [*1] **Heewoo Jun** [*1] **Christine Payne** [*1] **Jong Wook Kim** [1] **Alec Radford** [1] **Ilya Sutskever** [1]

Abstract

We introduce Jukebox, a model that generates music with singing in the raw audio domain. We tackle the long context of raw audio using a multi-scale VQ-VAE to compress it to discrete codes, and modeling those using autoregressive Transformers. We show that the combined model at scale can generate high-fidelity and diverse songs with coherence up to multiple minutes. We can condition on artist and genre to steer the musical and vocal style, and on unaligned lyrics to make the singing more controllable. We are releasing thousands of non cherry-picked samples, along with model weights and code.

1. Introduction

Music is an integral part of human culture, existing from the earliest periods of human civilization and evolving into a wide diversity of forms. It evokes a unique human spirit in its creation, and the question of whether computers can ever capture this creative process has fascinated computer scientists for decades. We have had algorithms generating piano sheet music (Hiller Jr & Isaacson, 1957; Moorer, 1972; Hadjeres et al., 2017; Huang et al., 2017), digital vocoders generating a singer's voice (Bonada & Serra, 2007; Saino et al., 2006; Blaauw & Bonada, 2017) and also synthesizers producing timbres for various musical instruments (Engel et al., 2017; 2019). Each captures a specific aspect of music generation: melody, composition, timbre, and the human voice singing. However, a single system to do it all remains elusive.

The field of generative models has made tremendous progress in the last few years. One of the aims of generative modeling is to capture the salient aspects of the data and to generate new instances indistinguishable from the true data The hypothesis is that by learning to produce the data we can learn the best features of the data[1]. We are surrounded by highly complex distributions in the visual, audio, and text domain, and in recent years we have developed advances in text generation (Radford et al.), speech generation (Xie et al., 2017) and image generation (Brock et al., 2019; Razavi et al., 2019). The rate of progress in this field has been rapid, where only a few years ago we had algorithms producing blurry faces (Kingma & Welling, 2014; Goodfellow et al., 2014) but now we now can generate high-resolution faces indistinguishable from real ones (Zhang et al., 2019b).

Generative models have been applied to the music generation task too. Earlier models generated music symbolically in the form of a pianoroll, which specifies the timing, pitch, velocity, and instrument of each note to be played. (Yang et al., 2017; Dong et al., 2018; Huang et al., 2019a; Payne, 2019; Roberts et al., 2018; Wu et al., 2019). The symbolic approach makes the modeling problem easier by working on the problem in the lower-dimensional space. However, it constrains the music that can be generated to being a specific sequence of notes and a fixed set of instruments to render with. In parallel, researchers have been pursuing the non-symbolic approach, where they try to produce music directly as a piece of audio. This makes the problem more challenging, as the space of raw audio is extremely high dimensional with a high amount of information content to model. There has been some success, with models producing piano pieces either in the raw audio domain (Oord et al., 2016; Mehri et al., 2017; Yamamoto et al., 2020) or in the spectrogram domain (Vasquez & Lewis, 2019). The key bottleneck is that modeling the raw audio directly introduces extremely long-range dependencies, making it computationally challenging to learn the high-level semantics of music. A way to reduce the difficulty is to learn a lower-dimensional encoding of the audio with the goal of losing the less important information but retaining most of the musical information. This approach has demonstrated some success in generating short instrumental pieces restricted to a set of a few instruments (Oord et al., 2017; Dieleman et al., 2018).

In this work, we show that we can use state-of-the-art deep generative models to produce a single system capable of generating diverse high-fidelity music in the raw audio domain, with long-range coherence spanning multiple minutes. Our approach uses a hierarchical VQ-VAE architecture (Razavi

[*]Equal contribution [1]OpenAI, San Francisco. Correspondence to: <jukebox@openai.com>.

[1]Richard Feynmann famously said, "What I cannot create, I do not understand"

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

Abstract

The choice of approximate posterior distribution is one of the core problems in variational inference. Most applications of variational inference employ simple families of posterior approximations in order to allow for efficient inference, focusing on mean-field or other simple structured approximations. This restriction has a significant impact on the quality of inferences made using variational methods. We introduce a new approach for specifying flexible, arbitrarily complex and scalable approximate posterior distributions. Our approximations are distributions constructed through a normalizing flow, whereby a simple initial density is transformed into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained. We use this view of normalizing flows to develop categories of finite and infinitesimal flows and provide a unified view of approaches for constructing rich posterior approximations. We demonstrate that the theoretical advantages of having posteriors that better match the true posterior, combined with the scalability of amortized variational approaches, provides a clear improvement in performance and applicability of variational inference.

1. Introduction

There has been a great deal of renewed interest in variational inference as a means of scaling probabilistic modeling to increasingly complex problems on increasingly larger data sets. Variational inference now lies at the core of large-scale topic models of text (Hoffman et al., 2013), provides the state-of-the-art in semi-supervised classification (Kingma et al., 2014), drives the models that currently produce the most realistic generative models of images (Gregor et al., 2014; 2015; Rezende et al., 2014; Kingma & Welling, 2014), and are a default tool for the understanding of many physical and chemical systems. Despite these successes and ongoing advances, there are a number of disadvantages of variational methods that limit their power and hamper their wider adoption as a default method for statistical inference. It is one of these limitations, the choice of posterior approximation, that we address in this paper.

Variational inference requires that intractable posterior distributions be approximated by a class of known probability distributions, over which we search for the best approximation to the true posterior. The class of approximations used is often limited, e.g., mean-field approximations, implying that no solution is ever able to resemble the true posterior distribution. This is a widely raised objection to variational methods, in that unlike other inferential methods such as MCMC, even in the asymptotic regime we are unable recover the true posterior distribution.

There is much evidence that richer, more faithful posterior approximations do result in better performance. For example, when compared to sigmoid belief networks that make use of mean-field approximations, deep auto-regressive networks use a posterior approximation with an auto-regressive dependency structure that provides a clear improvement in performance (Mnih & Gregor, 2014). There is also a large body of evidence that describes the detrimental effect of limited posterior approximations. Turner & Sahani (2011) provide an exposition of two commonly experienced problems. The first is the widely-observed problem of under-estimation of the variance of the posterior distribution, which can result in poor predictions and unreliable decisions based on the chosen posterior approximation. The second is that the limited capacity of the posterior approximation can also result in biases in the MAP estimates of any model parameters (and this is the case e.g., in time-series models).

A number of proposals for rich posterior approximations have been explored, typically based on structured mean-field approximations that incorporate some basic form of dependency within the approximate posterior. Another potentially powerful alternative would be to specify the approximate posterior as a mixture model, such as those developed by Jaakkola & Jordan (1998); Jordan et al. (1999); Gershman et al. (2012). But the mixture approach limits

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

s1: Simple test-time scaling

Niklas Muennighoff [*134] **Zitong Yang** [*1] **Weijia Shi** [*23] **Xiang Lisa Li** [*1] **Li Fei-Fei** [1] **Hannaneh Hajishirzi** [23]
Luke Zettlemoyer [2] **Percy Liang** [1] **Emmanuel Candès** [1] **Tatsunori Hashimoto** [1]

Abstract

Test-time scaling is a promising new approach to language modeling that uses extra test-time compute to improve performance. Recently, OpenAI's o1 model showed this capability but did not publicly share its methodology, leading to many replication efforts. We seek the simplest approach to achieve test-time scaling and strong reasoning performance. First, we curate a small dataset **s1K** of 1,000 questions paired with reasoning traces relying on three criteria we validate through ablations: difficulty, diversity, and quality. Second, we develop budget forcing to control test-time compute by forcefully terminating the model's thinking process or lengthening it by appending "Wait" multiple times to the model's generation when it tries to end. This can lead the model to double-check its answer, often fixing incorrect reasoning steps. After supervised finetuning the Qwen2.5-32B-Instruct language model on **s1K** and equipping it with budget forcing, our model **s1-32B** exceeds o1-preview on competition math questions by up to 27% (MATH and AIME24). Further, scaling **s1-32B** with budget forcing allows extrapolating beyond its performance without test-time intervention: from 50% to 57% on AIME24. Our model, data, and code are open-source at https://github.com/simplescaling/s1.

Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

1. Introduction

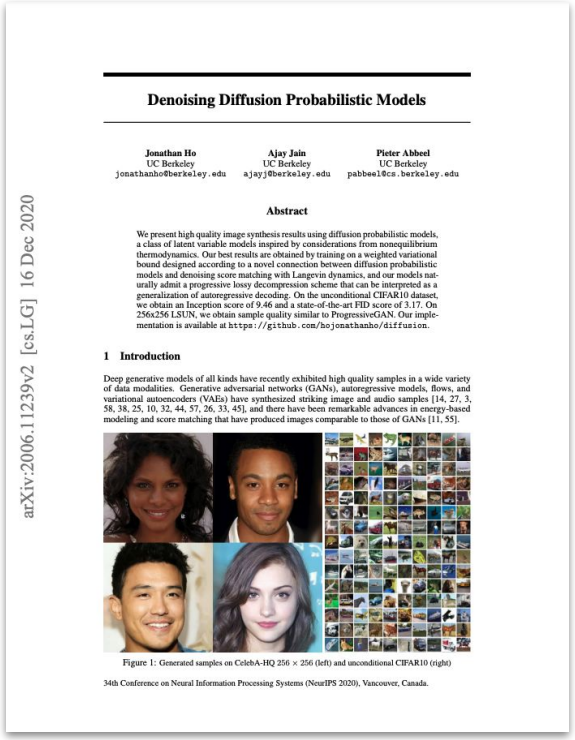
Performance improvements of language models (LMs) over the past years have largely relied on scaling up train-time compute using large-scale self-supervised pretraining (Kaplan et al., 2020; Hoffmann et al., 2022). The creation of these powerful models has set the stage for a new scaling paradigm built on top of them: *test-time scaling*. The aim of this approach is to increase the compute at test time to get better results. There has been much work exploring this idea (Snell et al., 2024; Welleck et al., 2024), and the viability of this paradigm was recently validated by OpenAI o1 (OpenAI, 2024). o1 has demonstrated strong reasoning performance with consistent gains from scaling test-time compute. OpenAI describes their approach as using large-scale reinforcement learning (RL) implying the use of sizable amounts of data (OpenAI, 2024). This has led to various attempts to replicate their models relying on techniques like Monte Carlo Tree Search (Gao et al., 2024b; Zhang et al., 2024a), multi-agent approaches (Qin et al., 2024), and others (Wang et al., 2024a; Huang et al., 2024b; 2025). Among these approaches, DeepSeek R1 (DeepSeek-AI et al., 2025) has successfully replicated o1-level performance, also employing reinforcement learning via millions of samples and multiple training stages. However, despite the large number of o1 replication attempts, none have openly replicated a clear test-time scaling behavior. Thus, we ask: what is the simplest approach to achieve both test-time scaling and strong reasoning performance?

We show that training on only 1,000 samples with next-token prediction and controlling thinking duration via a simple test-time technique we refer to as *budget forcing* leads to a strong reasoning model that scales in performance with more test-time compute. Specifically, we construct **s1K**, which consists of 1,000 carefully curated questions paired with reasoning traces and answers distilled from Gemini Thinking Experimental (Google, 2024). We perform supervised fine-tuning (SFT) of an off-the-shelf pretrained model

[*]Equal contribution. ZY and NM started the project. WS, NM and ZY collected the prompts, XL, ZY and NM, built the data pipeline, LZ and WS proposed using a 1K subset and ZY and NM built budget forcing. [1] Stanford University. [2] University of Washington, Seattle. [3] Allen Institute for AI. [4] Contextual AI.

Today

After: Second batch of four **paper presentations**.



arXiv:2006.11239v2 [cs.LG] 16 Dec 2020

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

arXiv:2005.00341v1 [eess.AS] 30 Apr 2020

Jukebox: A Generative Model for Music

Prafulla Dhariwal [*1] Heewoo Jun [*1] Christine Payne [*1] Jong Wook Kim [1] Alec Radford [1] Ilya Sutskever [1]

Abstract

We introduce Jukebox, a model that generates music with singing in the raw audio domain. We tackle the long context of raw audio using a multi-scale VQ-VAE to compress it to discrete codes, and modeling those using autoregressive Transformers. We show that the combined model at scale can generate high-fidelity and diverse songs with coherence up to multiple minutes. We can condition on artist and genre to steer the musical and vocal style, and on unaligned lyrics to make the singing more controllable. We are releasing thousands of non cherry-picked samples, along with model weights and code.

1. Introduction

Music is an integral part of human culture, existing from the earliest periods of human civilization and evolving into a wide diversity of forms. It evokes a unique human spirit in its creation, and the question of whether computers can ever capture this creative process has fascinated computer scientists for decades. We have had algorithms generating piano sheet music (Hiller Jr & Isaacson, 1957; Moorer, 1972; Hadjeres et al., 2017; Huang et al., 2017), digital vocoders generating a singer's voice (Bonada & Serra, 2007; Saino et al., 2006; Blaauw & Bonada, 2017) and also synthesizers producing timbres for various musical instruments (Engel et al., 2017; 2019). Each captures a specific aspect of music generation: melody, composition, timbre, and the human voice singing. However, a single system to do it all remains elusive.

The field of generative models has made tremendous progress in the last few years. One of the aims of generative modeling is to capture the salient aspects of the data and to generate new instances indistinguishable from the true data The hypothesis is that by learning to produce the data we can learn the best features of the data[1]. We are surrounded by highly complex distributions in the visual, audio, and text domain, and in recent years we have developed advances in text generation (Radford et al.), speech generation (Xie et al., 2017) and image generation (Brock et al., 2019; Razavi et al., 2019). The rate of progress in this field has been rapid, where only a few years ago we had algorithms producing blurry faces (Kingma & Welling, 2014; Goodfellow et al., 2014) but now we now can generate high-resolution faces indistinguishable from real ones (Zhang et al., 2019b).

Generative models have been applied to the music generation task too. Earlier models generated music symbolically in the form of a pianoroll, which specifies the timing, pitch, velocity, and instrument of each note to be played. (Yang et al., 2017; Dong et al., 2018; Huang et al., 2019a; Payne, 2019; Roberts et al., 2018; Wu et al., 2019). The symbolic approach makes the modeling problem easier by working on the problem in the lower-dimensional space. However, it constrains the music that can be generated to being a specific sequence of notes and a fixed set of instruments to render with. In parallel, researchers have been pursuing the non-symbolic approach, where they try to produce music directly as a piece of audio. This makes the problem more challenging, as the space of raw audio is extremely high dimensional with a high amount of information content to model. There has been some success, with models producing piano pieces either in the raw audio domain (Oord et al., 2016; Mehri et al., 2017; Yamamoto et al., 2020) or in the spectrogram domain (Vasquez & Lewis, 2019). The key bottleneck is that modeling the raw audio directly introduces extremely long-range dependencies, making it computationally challenging to learn the high-level semantics of music. A way to reduce the difficulty is to learn a lower-dimensional encoding of the audio with the goal of losing the less important information but retaining most of the musical information. This approach has demonstrated some success in generating short instrumental pieces restricted to a set of a few instruments (Oord et al., 2017; Dieleman et al., 2018).

In this work, we show that we can use state-of-the-art deep generative models to produce a single system capable of generating diverse high-fidelity music in the raw audio domain, with long-range coherence spanning multiple minutes. Our approach uses a hierarchical VQ-VAE architecture (Razavi

[1] Richard Feynmann famously said, "What I cannot create, I do not understand"

arXiv:1505.05770v6 [stat.ML] 14 Jun 2016

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

Abstract

The choice of approximate posterior distribution is one of the core problems in variational inference. Most applications of variational inference employ simple families of posterior approximations in order to allow for efficient inference, focusing on mean-field or other simple structured approximations. This restriction has a significant impact on the quality of inferences made using variational methods. We introduce a new approach for specifying flexible, arbitrarily complex and scalable approximate posterior distributions. Our approximations are distributions constructed through a normalizing flow, whereby a simple initial density is transformed into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained. We use this view of normalizing flows to develop categories of finite and infinitesimal flows and provide a unified view of approaches for constructing rich posterior approximations. We demonstrate that the theoretical advantages of having posteriors that better match the true posterior, combined with the scalability of amortized variational approaches, provides a clear improvement in performance and applicability of variational inference.

1. Introduction

There has been a great deal of renewed interest in variational inference as a means of scaling probabilistic modeling to increasingly complex problems on increasingly larger data sets. Variational inference now lies at the core of large-scale topic models of text (Hoffman et al., 2013), provides the state-of-the-art in semi-supervised classification (Kingma et al., 2014), drives the models that currently produce the most realistic generative models of images (Gregor et al., 2014; 2015; Rezende et al., 2014; Kingma & Welling, 2014), and are a default tool for the understanding of many physical and chemical systems. Despite these successes and ongoing advances, there are a number of disadvantages of variational methods that limit their power and hamper their wider adoption as a default method for statistical inference. It is one of these limitations, the choice of posterior approximation, that we address in this paper.

Variational inference requires that intractable posterior distributions be approximated by a class of known probability distributions, over which we search for the best approximation to the true posterior. The class of approximations used is often limited, e.g., mean-field approximations, implying that no solution is ever able to resemble the true posterior distribution. This is a widely raised objection to variational methods, in that unlike other inferential methods such as MCMC, even in the asymptotic regime we are unable recover the true posterior distribution.

There is much evidence that richer, more faithful posterior approximations do result in better performance. For example, when compared to sigmoid belief networks that make use of mean-field approximations, deep auto-regressive networks use a posterior approximation with an auto-regressive dependency structure that provides a clear improvement in performance (Mnih & Gregor, 2014). There is also a large body of evidence that describes the detrimental effect of limited posterior approximations. Turner & Sahani (2011) provide an exposition of two commonly experienced problems. The first is the widely-observed problem of under-estimation of the variance of the posterior distribution, which can result in poor predictions and unreliable decisions based on the chosen posterior approximation. The second is that the limited capacity of the posterior approximation can also result in biases in the MAP estimates of any model parameters (and this is the case e.g., in time-series models).

A number of proposals for rich posterior approximations have been explored, typically based on structured mean-field approximations that incorporate some basic form of dependency within the approximate posterior. Another potentially powerful alternative would be to specify the approximate posterior as a mixture model, such as those developed by Jaakkola & Jordan (1998); Jordan et al. (1999); Gershman et al. (2012). But the mixture approach limits

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

arXiv:2501.19393v3 [cs.CL] 1 Mar 2025

s1: Simple test-time scaling

Niklas Muennighoff [*1 3 4] Zitong Yang [*1] Weijia Shi [*2 3] Xiang Lisa Li [*1] Li Fei-Fei [1] Hannaneh Hajishirzi [2 3]
Luke Zettlemoyer [2] Percy Liang [1] Emmanuel Candès [1] Tatsunori Hashimoto [1]

Abstract

Test-time scaling is a promising new approach to language modeling that uses extra test-time compute to improve performance. Recently, OpenAI's o1 model showed this capability but did not publicly share its methodology, leading to many replication efforts. We seek the simplest approach to achieve test-time scaling and strong reasoning performance. First, we curate a small dataset **s1K** of 1,000 questions paired with reasoning traces relying on three criteria we validate through ablations: difficulty, diversity, and quality. Second, we develop budget forcing to control test-time compute by forcefully terminating the model's thinking process or lengthening it by appending "Wait" multiple times to the model's generation when it tries to end. This can lead the model to double-check its answer, often fixing incorrect reasoning steps. After supervised finetuning the Qwen2.5-32B-Instruct language model on **s1K** and equipping it with budget forcing, our model **s1-32B** exceeds o1-preview on competition math questions by up to 27% (MATH and AIME24). Further, scaling **s1-32B** with budget forcing allows extrapolating beyond its performance without test-time intervention: from 50% to 57% on AIME24. Our model, data, and code are open-source at https://github.com/simplescaling/s1.

Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

1. Introduction

Performance improvements of language models (LMs) over the past years have largely relied on scaling up train-time compute using large-scale self-supervised pretraining (Kaplan et al., 2020; Hoffmann et al., 2022). The creation of these powerful models has set the stage for a new scaling paradigm built on top of them: *test-time scaling*. The aim of this approach is to increase the compute at test time to get better results. There has been much work exploring this idea (Snell et al., 2024; Welleck et al., 2024), and the viability of this paradigm was recently validated by OpenAI o1 (OpenAI, 2024). o1 has demonstrated strong reasoning performance with consistent gains from scaling test-time compute. OpenAI describes their approach as using large-scale reinforcement learning (RL) implying the use of sizable amounts of data (OpenAI, 2024). This has led to various attempts to replicate their models relying on techniques like Monte Carlo Tree Search (Gao et al., 2024b; Zhang et al., 2024a), multi-agent approaches (Qin et al., 2024), and others (Wang et al., 2024a; Huang et al., 2024b; 2025). Among these approaches, DeepSeek R1 (DeepSeek-AI et al., 2025) has successfully replicated o1-level performance, also employing reinforcement learning via millions of samples and multiple training stages. However, despite the large number of o1 replication attempts, none have openly replicated a clear test-time scaling behavior. Thus, we ask: what is the simplest approach to achieve both test-time scaling and strong reasoning performance?

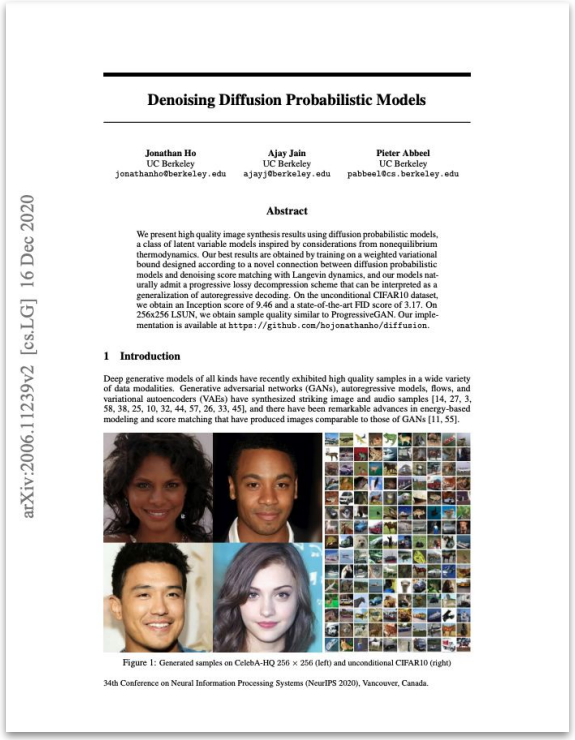
We show that training on only 1,000 samples with next-token prediction and controlling thinking duration via a simple test-time technique we refer to as *budget forcing* leads to a strong reasoning model that scales in performance with more test-time compute. Specifically, we construct **s1K**, which consists of 1,000 carefully curated questions paired with reasoning traces and answers distilled from Gemini Thinking Experimental (Google, 2024). We perform supervised fine-tuning (SFT) of an off-the-shelf pretrained model

[*] Equal contribution. ZY and NM started the project. WS, NM and ZY collected the prompts, XL, ZY and NM, built the data pipeline, LZ and WS proposed using a 1K subset and ZY and NM built budget forcing. [1] Stanford University. [2] University of Washington, Seattle. [3] Allen Institute for AI. [4] Contextual AI.

Canonical diffusion model paper (DDPM)

Today

After: Second batch of four **paper presentations**.



arXiv:2006.11239v2 [cs.LG] 16 Dec 2020

Denoising Diffusion Probabilistic Models

Jonathan Ho UC Berkeley jonathanho@berkeley.edu
Ajay Jain UC Berkeley ajayj@berkeley.edu
Pieter Abbeel UC Berkeley pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

arXiv:2005.00341v1 [eess.AS] 30 Apr 2020

Jukebox: A Generative Model for Music

Prafulla Dhariwal [*1] **Heewoo Jun** [*1] **Christine Payne** [*1] **Jong Wook Kim** [1] **Alec Radford** [1] **Ilya Sutskever** [1]

Abstract

We introduce Jukebox, a model that generates music with singing in the raw audio domain. We tackle the long context of raw audio using a multi-scale VQ-VAE to compress it to discrete codes, and modeling those using autoregressive Transformers. We show that the combined model at scale can generate high-fidelity and diverse songs with coherence up to multiple minutes. We can condition on artist and genre to steer the musical and vocal style, and on unaligned lyrics to make the singing more controllable. We are releasing thousands of non cherry-picked samples, along with model weights and code.

1. Introduction

Music is an integral part of human culture, existing from the earliest periods of human civilization and evolving into a wide diversity of forms. It evokes a unique human spirit in its creation, and the question of whether computers can ever capture this creative process has fascinated computer scientists for decades. We have had algorithms generating piano sheet music (Hiller Jr & Isaacson, 1957; Moorer, 1972; Hadjeres et al., 2017; Huang et al., 2017), digital vocoders generating a singer's voice (Bonada & Serra, 2007; Saino et al., 2006; Blaauw & Bonada, 2017) and also synthesizers producing timbres for various musical instruments (Engel et al., 2017; 2019). Each captures a specific aspect of music generation: melody, composition, timbre, and the human voice singing. However, a single system to do it all remains elusive.

The field of generative models has made tremendous progress in the last few years. One of the aims of generative modeling is to capture the salient aspects of the data and to generate new instances indistinguishable from the true data The hypothesis is that by learning to produce the data we can learn the best features of the data[1]. We are surrounded by highly complex distributions in the visual, audio, and text domain, and in recent years we have developed advances in text generation (Radford et al.), speech generation (Xie et al., 2017) and image generation (Brock et al., 2019; Razavi et al., 2019). The rate of progress in this field has been rapid, where only a few years ago we had algorithms producing blurry faces (Kingma & Welling, 2014; Goodfellow et al., 2014) but now we now can generate high-resolution faces indistinguishable from real ones (Zhang et al., 2019b).

Generative models have been applied to the music generation task too. Earlier models generated music symbolically in the form of a pianoroll, which specifies the timing, pitch, velocity, and instrument of each note to be played. (Yang et al., 2017; Dong et al., 2018; Huang et al., 2019a; Payne, 2019; Roberts et al., 2018; Wu et al., 2019). The symbolic approach makes the modeling problem easier by working on the problem in the lower-dimensional space. However, it constrains the music that can be generated to being a specific sequence of notes and a fixed set of instruments to render with. In parallel, researchers have been pursuing the non-symbolic approach, where they try to produce music directly as a piece of audio. This makes the problem more challenging, as the space of raw audio is extremely high dimensional with a high amount of information content to model. There has been some success, with models producing piano pieces either in the raw audio domain (Oord et al., 2016; Mehri et al., 2017; Yamamoto et al., 2020) or in the spectrogram domain (Vasquez & Lewis, 2019). The key bottleneck is that modeling the raw audio directly introduces extremely long-range dependencies, making it computationally challenging to learn the high-level semantics of music. A way to reduce the difficulty is to learn a lower-dimensional encoding of the audio with the goal of losing the less important information but retaining most of the musical information. This approach has demonstrated some success in generating short instrumental pieces restricted to a set of a few instruments (Oord et al., 2017; Dieleman et al., 2018).

In this work, we show that we can use state-of-the-art deep generative models to produce a single system capable of generating diverse high-fidelity music in the raw audio domain, with long-range coherence spanning multiple minutes. Our approach uses a hierarchical VQ-VAE architecture (Razavi

[*]Equal contribution [1]OpenAI, San Francisco. Correspondence to: <jukebox@openai.com>.

[1]Richard Feynmann famously said, "What I cannot create, I do not understand"

arXiv:1505.05770v6 [stat.ML] 14 Jun 2016

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

Abstract

The choice of approximate posterior distribution is one of the core problems in variational inference. Most applications of variational inference employ simple families of posterior approximations in order to allow for efficient inference, focusing on mean-field or other simple structured approximations. This restriction has a significant impact on the quality of inferences made using variational methods. We introduce a new approach for specifying flexible, arbitrarily complex and scalable approximate posterior distributions. Our approximations are distributions constructed through a normalizing flow, whereby a simple initial density is transformed into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained. We use this view of normalizing flows to develop categories of finite and infinitesimal flows and provide a unified view of approaches for constructing rich posterior approximations. We demonstrate that the theoretical advantages of having posteriors that better match the true posterior, combined with the scalability of amortized variational approaches, provides a clear improvement in performance and applicability of variational inference.

1. Introduction

There has been a great deal of renewed interest in variational inference as a means of scaling probabilistic modeling to increasingly complex problems on increasingly larger data sets. Variational inference now lies at the core of large-scale topic models of text (Hoffman et al., 2013), provides the state-of-the-art in semi-supervised classification (Kingma et al., 2014), drives the models that currently produce the most realistic generative models of images (Gregor et al., 2014; 2015; Rezende et al., 2014; Kingma & Welling, 2014), and are a default tool for the understanding of many physical and chemical systems. Despite these successes and ongoing advances, there are a number of disadvantages of variational methods that limit their power and hamper their wider adoption as a default method for statistical inference. It is one of these limitations, the choice of posterior approximation, that we address in this paper.

Variational inference requires that intractable posterior distributions be approximated by a class of known probability distributions, over which we search for the best approximation to the true posterior. The class of approximations used is often limited, e.g., mean-field approximations, implying that no solution is ever able to resemble the true posterior distribution. This is a widely raised objection to variational methods, in that unlike other inferential methods such as MCMC, even in the asymptotic regime we are unable recover the true posterior distribution.

There is much evidence that richer, more faithful posterior approximations do result in better performance. For example, when compared to sigmoid belief networks that make use of mean-field approximations, deep auto-regressive networks use a posterior approximation with an auto-regressive dependency structure that provides a clear improvement in performance (Mnih & Gregor, 2014). There is also a large body of evidence that describes the detrimental effect of limited posterior approximations. Turner & Sahani (2011) provide an exposition of two commonly experienced problems. The first is the widely-observed problem of under-estimation of the variance of the posterior distribution, which can result in poor predictions and unreliable decisions based on the chosen posterior approximation. The second is that the limited capacity of the posterior approximation can also result in biases in the MAP estimates of any model parameters (and this is the case e.g., in time-series models).

A number of proposals for rich posterior approximations have been explored, typically based on structured mean-field approximations that incorporate some basic form of dependency within the approximate posterior. Another potentially powerful alternative would be to specify the approximate posterior as a mixture model, such as those developed by Jaakkola & Jordan (1998); Jordan et al. (1999); Gershman et al. (2012). But the mixture approach limits

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

arXiv:2501.19393v3 [cs.CL] 1 Mar 2025

s1: Simple test-time scaling

Niklas Muennighoff [*134] **Zitong Yang** [*1] **Weijia Shi** [*23] **Xiang Lisa Li** [*1] **Li Fei-Fei** [1] **Hannaneh Hajishirzi** [23] **Luke Zettlemoyer** [2] **Percy Liang** [1] **Emmanuel Candès** [3] **Tatsunori Hashimoto** [1]

Abstract

Test-time scaling is a promising new approach to language modeling that uses extra test-time compute to improve performance. Recently, OpenAI's o1 model showed this capability but did not publicly share its methodology, leading to many replication efforts. We seek the simplest approach to achieve test-time scaling and strong reasoning performance. First, we curate a small dataset **s1K** of 1,000 questions paired with reasoning traces relying on three criteria we validate through ablations: difficulty, diversity, and quality. Second, we develop budget forcing to control test-time compute by forcefully terminating the model's thinking process or lengthening it by appending "Wait" multiple times to the model's generation when it tries to end. This can lead the model to double-check its answer, often fixing incorrect reasoning steps. After supervised finetuning the Qwen2.5-32B-Instruct language model on **s1K** and equipping it with budget forcing, our model **s1-32B** exceeds o1-preview on competition math questions by up to 27% (MATH and AIME24). Further, scaling **s1-32B** with budget forcing allows extrapolating beyond its performance without test-time intervention: from 50% to 57% on AIME24. Our model, data, and code are open-source at https://github.com/simplescaling/s1.

Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

1. Introduction

Performance improvements of language models (LMs) over the past years have largely relied on scaling up train-time compute using large-scale self-supervised pretraining (Kaplan et al., 2020; Hoffmann et al., 2022). The creation of these powerful models has set the stage for a new scaling paradigm built on top of them: *test-time scaling*. The aim of this approach is to increase the compute at test time to get better results. There has been much work exploring this idea (Snell et al., 2024; Welleck et al., 2024), and the viability of this paradigm was recently validated by OpenAI o1 (OpenAI, 2024). o1 has demonstrated strong reasoning performance with consistent gains from scaling test-time compute. OpenAI describes their approach as using large-scale reinforcement learning (RL) implying the use of sizable amounts of data (OpenAI, 2024). This has led to various attempts to replicate their models relying on techniques like Monte Carlo Tree Search (Gao et al., 2024b; Zhang et al., 2024a), multi-agent approaches (Qin et al., 2024), and others (Wang et al., 2024a; Huang et al., 2024b; 2025). Among these approaches, DeepSeek R1 (DeepSeek-AI et al., 2025) has successfully replicated o1-level performance, also employing reinforcement learning via millions of samples and multiple training stages. However, despite the large number of o1 replication attempts, none have openly replicated a clear test-time scaling behavior. Thus, we ask: what is the simplest approach to achieve both test-time scaling and strong reasoning performance?

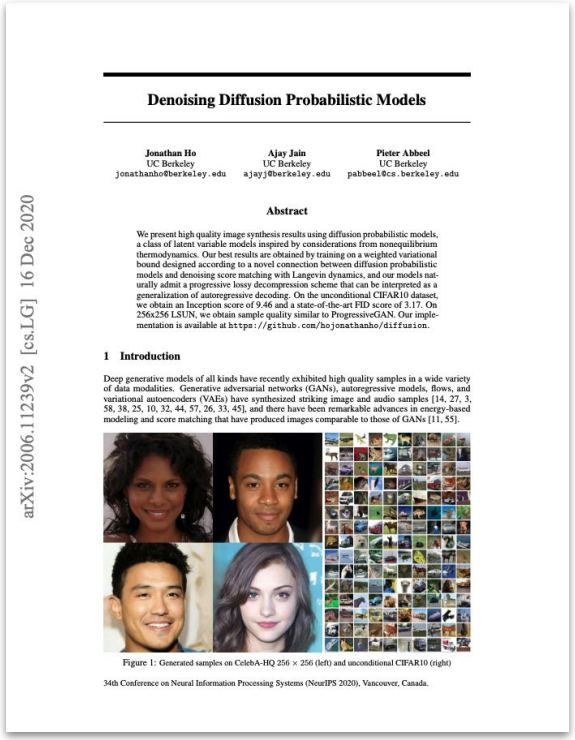
We show that training on only 1,000 samples with next-token prediction and controlling thinking duration via a simple test-time technique we refer to as *budget forcing* leads to a strong reasoning model that scales in performance with more test-time compute. Specifically, we construct **s1K**, which consists of 1,000 carefully curated questions paired with reasoning traces and answers distilled from Gemini Thinking Experimental (Google, 2024). We perform supervised fine-tuning (SFT) of an off-the-shelf pretrained model

[*]Equal contribution. ZY and NM started the project. WS, NM and ZY collected the prompts, XL, ZY and NM, built the data pipeline, LZ and WS proposed using a 1K subset and ZY and NM built budget forcing. [1] Stanford University. [2] University of Washington, Seattle. [3] Allen Institute for AI. [4] Contextual AI.

Multiscale VAE for audio/music generation

Today

After: Second batch of four **paper presentations**.



arXiv:2006.11239v2 [cs.LG] 16 Dec 2020

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

arXiv:2005.00341v1 [eess.AS] 30 Apr 2020

Jukebox: A Generative Model for Music

Prafulla Dhariwal [*1] Heewoo Jun [*1] Christine Payne [*1] Jong Wook Kim [1] Alec Radford [1] Ilya Sutskever [1]

Abstract

We introduce Jukebox, a model that generates music with singing in the raw audio domain. We tackle the long context of raw audio using a multi-scale VQ-VAE to compress it to discrete codes, and modeling those using autoregressive Transformers. We show that the combined model at scale can generate high-fidelity and diverse songs with coherence up to multiple minutes. We can condition on artist and genre to steer the musical and vocal style, and on unaligned lyrics to make the singing more controllable. We are releasing thousands of non cherry-picked samples, along with model weights and code.

1. Introduction

Music is an integral part of human culture, existing from the earliest periods of human civilization and evolving into a wide diversity of forms. It evokes a unique human spirit in its creation, and the question of whether computers can ever capture this creative process has fascinated computer scientists for decades. We have had algorithms generating piano sheet music (Hiller Jr & Isaacson, 1957; Moorer, 1972; Hadjeres et al., 2017; Huang et al., 2017), digital vocoders generating a singer's voice (Bonada & Serra, 2007; Saino et al., 2006; Blaauw & Bonada, 2017) and also synthesizers producing timbres for various musical instruments (Engel et al., 2017; 2019). Each captures a specific aspect of music generation: melody, composition, timbre, and the human voice singing. However, a single system to do it all remains elusive.

The field of generative models has made tremendous progress in the last few years. One of the aims of generative modeling is to capture the salient aspects of the data and to generate new instances indistinguishable from the true data The hypothesis is that by learning to produce the data we can learn the best features of the data[1]. We are surrounded by highly complex distributions in the visual, audio, and text domain, and in recent years we have developed advances in text generation (Radford et al.), speech generation (Xie et al., 2017) and image generation (Brock et al., 2019; Razavi et al., 2019). The rate of progress in this field has been rapid, where only a few years ago we had algorithms producing blurry faces (Kingma & Welling, 2014; Goodfellow et al., 2014) but now we now can generate high-resolution faces indistinguishable from real ones (Zhang et al., 2019b).

Generative models have been applied to the music generation task too. Earlier models generated music symbolically in the form of a pianoroll, which specifies the timing, pitch, velocity, and instrument of each note to be played. (Yang et al., 2017; Dong et al., 2018; Huang et al., 2019a; Payne, 2019; Roberts et al., 2018; Wu et al., 2019). The symbolic approach makes the modeling problem easier by working on the problem in the lower-dimensional space. However, it constrains the music that can be generated to being a specific sequence of notes and a fixed set of instruments to render with. In parallel, researchers have been pursuing the non-symbolic approach, where they try to produce music directly as a piece of audio. This makes the problem more challenging, as the space of raw audio is extremely high dimensional with a high amount of information content to model. There has been some success, with models producing piano pieces either in the raw audio domain (Oord et al., 2016; Mehri et al., 2017; Yamamoto et al., 2020) or in the spectrogram domain (Vasquez & Lewis, 2019). The key bottleneck is that modeling the raw audio directly introduces extremely long-range dependencies, making it computationally challenging to learn the high-level semantics of music. A way to reduce the difficulty is to learn a lower-dimensional encoding of the audio with the goal of losing the less important information but retaining most of the musical information. This approach has demonstrated some success in generating short instrumental pieces restricted to a set of a few instruments (Oord et al., 2017; Dieleman et al., 2018).

In this work, we show that we can use state-of-the-art deep generative models to produce a single system capable of generating diverse high-fidelity music in the raw audio domain, with long-range coherence spanning multiple minutes. Our approach uses a hierarchical VQ-VAE architecture (Razavi

[1] Richard Feynmann famously said, "What I cannot create, I do not understand"

arXiv:1505.05770v6 [stat.ML] 14 Jun 2016

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

Abstract

The choice of approximate posterior distribution is one of the core problems in variational inference. Most applications of variational inference employ simple families of posterior approximations in order to allow for efficient inference, focusing on mean-field or other simple structured approximations. This restriction has a significant impact on the quality of inferences made using variational methods. We introduce a new approach for specifying flexible, arbitrarily complex and scalable approximate posterior distributions. Our approximations are distributions constructed through a normalizing flow, whereby a simple initial density is transformed into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained. We use this view of normalizing flows to develop categories of finite and infinitesimal flows and provide a unified view of approaches for constructing rich posterior approximations. We demonstrate that the theoretical advantages of having posteriors that better match the true posterior, combined with the scalability of amortized variational approaches, provides a clear improvement in performance and applicability of variational inference.

1. Introduction

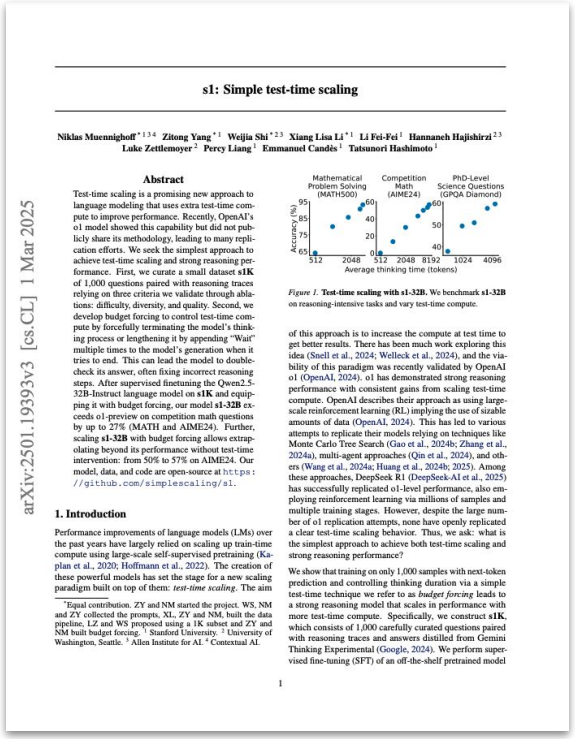
There has been a great deal of renewed interest in variational inference as a means of scaling probabilistic modeling to increasingly complex problems on increasingly larger data sets. Variational inference now lies at the core of large-scale topic models of text (Hoffman et al., 2013), provides the state-of-the-art in semi-supervised classification (Kingma et al., 2014), drives the models that currently produce the most realistic generative models of images (Gregor et al., 2014; 2015; Rezende et al., 2014; Kingma & Welling, 2014), and are a default tool for the understanding of many physical and chemical systems. Despite these successes and ongoing advances, there are a number of disadvantages of variational methods that limit their power and hamper their wider adoption as a default method for statistical inference. It is one of these limitations, the choice of posterior approximation, that we address in this paper.

Variational inference requires that intractable posterior distributions be approximated by a class of known probability distributions, over which we search for the best approximation to the true posterior. The class of approximations used is often limited, e.g., mean-field approximations, implying that no solution is ever able to resemble the true posterior distribution. This is a widely raised objection to variational methods, in that unlike other inferential methods such as MCMC, even in the asymptotic regime we are unable recover the true posterior distribution.

There is much evidence that richer, more faithful posterior approximations do result in better performance. For example, when compared to sigmoid belief networks that make use of mean-field approximations, deep auto-regressive networks use a posterior approximation with an auto-regressive dependency structure that provides a clear improvement in performance (Mnih & Gregor, 2014). There is also a large body of evidence that describes the detrimental effect of limited posterior approximations. Turner & Sahani (2011) provide an exposition of two commonly experienced problems. The first is the widely-observed problem of under-estimation of the variance of the posterior distribution, which can result in poor predictions and unreliable decisions based on the chosen posterior approximation. The second is that the limited capacity of the posterior approximation can also result in biases in the MAP estimates of any model parameters (and this is the case e.g., in time-series models).

A number of proposals for rich posterior approximations have been explored, typically based on structured mean-field approximations that incorporate some basic form of dependency within the approximate posterior. Another potentially powerful alternative would be to specify the approximate posterior as a mixture model, such as those developed by Jaakkola & Jordan (1998); Jordan et al. (1999); Gershman et al. (2012). But the mixture approach limits

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.



arXiv:2501.19393v3 [cs.CL] 1 Mar 2025

s1: Simple test-time scaling

Niklas Muennighoff [*1 3 4] Zitong Yang [*1] Weijia Shi [*2 3] Xiang Lisa Li [*1] Li Fei-Fei [1] Hannaneh Hajishirzi [2 3]
Luke Zettlemoyer [2] Percy Liang [1] Emmanuel Candès [1] Tatsunori Hashimoto [1]

Abstract

Test-time scaling is a promising new approach to language modeling that uses extra test-time compute to improve performance. Recently, OpenAI's o1 model showed this capability but did not publicly share its methodology, leading to many replication efforts. We seek the simplest approach to achieve test-time scaling and strong reasoning performance. First, we curate a small dataset **s1K** of 1,000 questions paired with reasoning traces relying on three criteria we validate through ablations: difficulty, diversity, and quality. Second, we develop budget forcing to control test-time compute by forcefully terminating the model's thinking process or lengthening it by appending "Wait" multiple times to the model's generation when it tries to end. This can lead the model to double-check its answer, often fixing incorrect reasoning steps. After supervised finetuning the Qwen2.5-32B-Instruct language model on **s1K** and equipping it with budget forcing, our model **s1-32B** exceeds o1-preview on competition math questions by up to 27% (MATH and AIME24). Further, scaling **s1-32B** with budget forcing allows extrapolating beyond its performance without test-time intervention: from 50% to 57% on AIME24. Our model, data, and code are open-source at https://github.com/simplescaling/s1.

Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

1. Introduction

Performance improvements of language models (LMs) over the past years have largely relied on scaling up train-time compute using large-scale self-supervised pretraining (Kaplan et al., 2020; Hoffmann et al., 2022). The creation of these powerful models has set the stage for a new scaling paradigm built on top of them: *test-time scaling*. The aim of this approach is to increase the compute at test time to get better results. There has been much work exploring this idea (Snell et al., 2024; Welleck et al., 2024), and the viability of this paradigm was recently validated by OpenAI o1 (OpenAI, 2024). o1 has demonstrated strong reasoning performance with consistent gains from scaling test-time compute. OpenAI describes their approach as using large-scale reinforcement learning (RL) implying the use of sizable amounts of data (OpenAI, 2024). This has led to various attempts to replicate their models relying on techniques like Monte Carlo Tree Search (Gao et al., 2024b; Zhang et al., 2024a), multi-agent approaches (Qin et al., 2024), and others (Wang et al., 2024a; Huang et al., 2024b; 2025). Among these approaches, DeepSeek R1 (DeepSeek-AI et al., 2025) has successfully replicated o1-level performance, also employing reinforcement learning via millions of samples and multiple training stages. However, despite the large number of o1 replication attempts, none have openly replicated a clear test-time scaling behavior. Thus, we ask: what is the simplest approach to achieve both test-time scaling and strong reasoning performance?

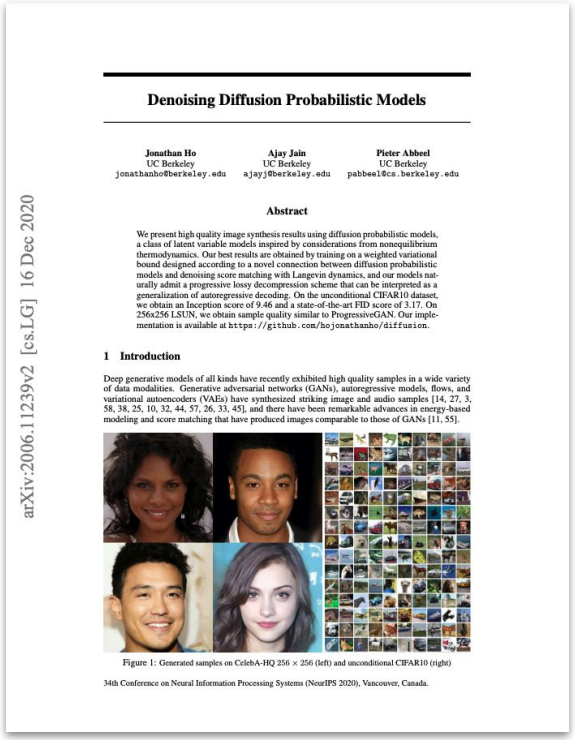
We show that training on only 1,000 samples with next-token prediction and controlling thinking duration via a simple test-time technique we refer to as *budget forcing* leads to a strong reasoning model that scales in performance with more test-time compute. Specifically, we construct **s1K**, which consists of 1,000 carefully curated questions paired with reasoning traces and answers distilled from Gemini Thinking Experimental (Google, 2024). We perform supervised fine-tuning (SFT) of an off-the-shelf pretrained model

[*] Equal contribution. ZY and NM started the project. WS, NM and ZY collected the prompts, XL, ZY and NM, built the data pipeline, LZ and WS proposed using a 1K subset and ZY and NM built budget forcing. [1] Stanford University. [2] University of Washington, Seattle. [3] Allen Institute for AI. [4] Contextual AI.

Normalizing flow models for VI

Today

After: Second batch of four **paper presentations**.



arXiv:2006.11239v2 [cs.LG] 16 Dec 2020

Denoising Diffusion Probabilistic Models

Jonathan Ho UC Berkeley jonathanho@berkeley.edu

Ajay Jain UC Berkeley ajayj@berkeley.edu

Pieter Abbeel UC Berkeley pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

arXiv:2005.00341v1 [eess.AS] 30 Apr 2020

Jukebox: A Generative Model for Music

Prafulla Dhariwal [*1] **Heewoo Jun** [*1] **Christine Payne** [*1] **Jong Wook Kim** [1] **Alec Radford** [1] **Ilya Sutskever** [1]

Abstract

We introduce Jukebox, a model that generates music with singing in the raw audio domain. We tackle the long context of raw audio using a multi-scale VQ-VAE to compress it to discrete codes, and modeling those using autoregressive Transformers. We show that the combined model at scale can generate high-fidelity and diverse songs with coherence up to multiple minutes. We can condition on artist and genre to steer the musical and vocal style, and on unaligned lyrics to make the singing more controllable. We are releasing thousands of non cherry-picked samples, along with model weights and code.

1. Introduction

Music is an integral part of human culture, existing from the earliest periods of human civilization and evolving into a wide diversity of forms. It evokes a unique human spirit in its creation, and the question of whether computers can ever capture this creative process has fascinated computer scientists for decades. We have had algorithms generating piano sheet music (Hiller Jr & Isaacson, 1957; Moorer, 1972; Hadjeres et al., 2017; Huang et al., 2017), digital vocoders generating a singer's voice (Bonada & Serra, 2007; Saino et al., 2006; Blaauw & Bonada, 2017) and also synthesizers producing timbres for various musical instruments (Engel et al., 2017; 2019). Each captures a specific aspect of music generation: melody, composition, timbre, and the human voice singing. However, a single system to do it all remains elusive.

The field of generative models has made tremendous progress in the last few years. One of the aims of generative modeling is to capture the salient aspects of the data and to generate new instances indistinguishable from the true data The hypothesis is that by learning to produce the data we can learn the best features of the data[1]. We are surrounded by highly complex distributions in the visual, audio, and text domain, and in recent years we have developed advances in text generation (Radford et al.), speech generation (Xie et al., 2017) and image generation (Brock et al., 2019; Razavi et al., 2019). The rate of progress in this field has been rapid, where only a few years ago we had algorithms producing blurry faces (Kingma & Welling, 2014; Goodfellow et al., 2014) but now we now can generate high-resolution faces indistinguishable from real ones (Zhang et al., 2019b).

Generative models have been applied to the music generation task too. Earlier models generated music symbolically in the form of a pianoroll, which specifies the timing, pitch, velocity, and instrument of each note to be played. (Yang et al., 2017; Dong et al., 2018; Huang et al., 2019a; Payne, 2019; Roberts et al., 2018; Wu et al., 2019). The symbolic approach makes the modeling problem easier by working on the problem in the lower-dimensional space. However, it constrains the music that can be generated to being a specific sequence of notes and a fixed set of instruments to render with. In parallel, researchers have been pursuing the non-symbolic approach, where they try to produce music directly as a piece of audio. This makes the problem more challenging, as the space of raw audio is extremely high dimensional with a high amount of information content to model. There has been some success, with models producing piano pieces either in the raw audio domain (Oord et al., 2016; Mehri et al., 2017; Yamamoto et al., 2020) or in the spectrogram domain (Vasquez & Lewis, 2019). The key bottleneck is that modeling the raw audio directly introduces extremely long-range dependencies, making it computationally challenging to learn the high-level semantics of music. A way to reduce the difficulty is to learn a lower-dimensional encoding of the audio with the goal of losing the less important information but retaining most of the musical information. This approach has demonstrated some success in generating short instrumental pieces restricted to a set of a few instruments (Oord et al., 2017; Dieleman et al., 2018).

In this work, we show that we can use state-of-the-art deep generative models to produce a single system capable of generating diverse high-fidelity music in the raw audio domain, with long-range coherence spanning multiple minutes. Our approach uses a hierarchical VQ-VAE architecture (Razavi

[*]Equal contribution [1]OpenAI, San Francisco. Correspondence to: <jukebox@openai.com>.

[1]Richard Feynmann famously said, "What I cannot create, I do not understand"

arXiv:1505.05770v6 [stat.ML] 14 Jun 2016

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

Abstract

The choice of approximate posterior distribution is one of the core problems in variational inference. Most applications of variational inference employ simple families of posterior approximations in order to allow for efficient inference, focusing on mean-field or other simple structured approximations. This restriction has a significant impact on the quality of inferences made using variational methods. We introduce a new approach for specifying flexible, arbitrarily complex and scalable approximate posterior distributions. Our approximations are distributions constructed through a normalizing flow, whereby a simple initial density is transformed into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained. We use this view of normalizing flows to develop categories of finite and infinitesimal flows and provide a unified view of approaches for constructing rich posterior approximations. We demonstrate that the theoretical advantages of having posteriors that better match the true posterior, combined with the scalability of amortized variational approaches, provides a clear improvement in performance and applicability of variational inference.

1. Introduction

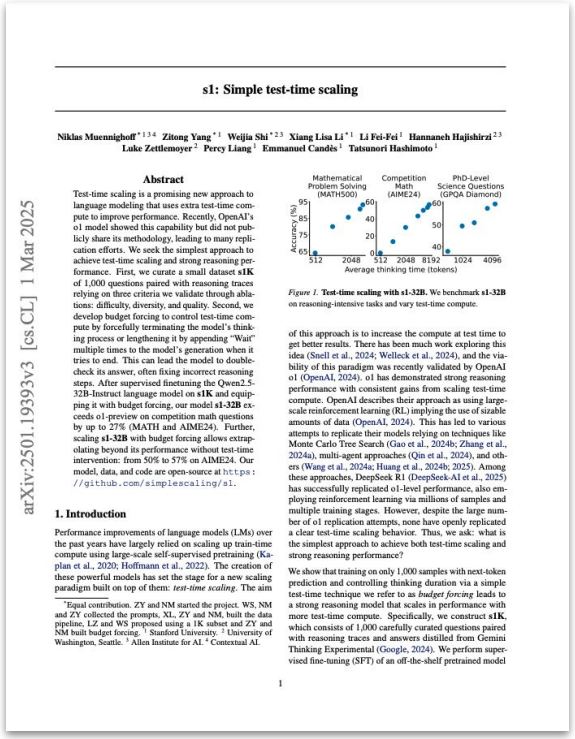
There has been a great deal of renewed interest in variational inference as a means of scaling probabilistic modeling to increasingly complex problems on increasingly larger data sets. Variational inference now lies at the core of large-scale topic models of text (Hoffman et al., 2013), provides the state-of-the-art in semi-supervised classification (Kingma et al., 2014), drives the models that currently produce the most realistic generative models of images (Gregor et al., 2014; 2015; Rezende et al., 2014; Kingma & Welling, 2014), and are a default tool for the understanding of many physical and chemical systems. Despite these successes and ongoing advances, there are a number of disadvantages of variational methods that limit their power and hamper their wider adoption as a default method for statistical inference. It is one of these limitations, the choice of posterior approximation, that we address in this paper.

Variational inference requires that intractable posterior distributions be approximated by a class of known probability distributions, over which we search for the best approximation to the true posterior. The class of approximations used is often limited, e.g., mean-field approximations, implying that no solution is ever able to resemble the true posterior distribution. This is a widely raised objection to variational methods, in that unlike other inferential methods such as MCMC, even in the asymptotic regime we are unable recover the true posterior distribution.

There is much evidence that richer, more faithful posterior approximations do result in better performance. For example, when compared to sigmoid belief networks that make use of mean-field approximations, deep auto-regressive networks use a posterior approximation with an auto-regressive dependency structure that provides a clear improvement in performance (Mnih & Gregor, 2014). There is also a large body of evidence that describes the detrimental effect of limited posterior approximations. Turner & Sahani (2011) provide an exposition of two commonly experienced problems. The first is the widely-observed problem of under-estimation of the variance of the posterior distribution, which can result in poor predictions and unreliable decisions based on the chosen posterior approximation. The second is that the limited capacity of the posterior approximation can also result in biases in the MAP estimates of any model parameters (and this is the case e.g., in time-series models).

A number of proposals for rich posterior approximations have been explored, typically based on structured mean-field approximations that incorporate some basic form of dependency within the approximate posterior. Another potentially powerful alternative would be to specify the approximate posterior as a mixture model, such as those developed by Jaakkola & Jordan (1998); Jordan et al. (1999); Gershman et al. (2012). But the mixture approach limits

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.



arXiv:2501.19393v3 [cs.CL] 1 Mar 2025

s1: Simple test-time scaling

Niklas Muennighoff [*134] **Zitong Yang** [*1] **Weijia Shi** [*23] **Xiang Lisa Li** [*1] **Li Fei-Fei** [1] **Hannaneh Hajishirzi** [23] **Luke Zettlemoyer** [2] **Percy Liang** [1] **Emmanuel Candès** [3] **Tatsunori Hashimoto** [1]

Abstract

Test-time scaling is a promising new approach to language modeling that uses extra test-time compute to improve performance. Recently, OpenAI's o1 model showed this capability but did not publicly share its methodology, leading to many replication efforts. We seek the simplest approach to achieve test-time scaling and strong reasoning performance. First, we curate a small dataset **s1K** of 1,000 questions paired with reasoning traces relying on three criteria we validate through ablations: difficulty, diversity, and quality. Second, we develop budget forcing to control test-time compute by forcefully terminating the model's thinking process or lengthening it by appending "Wait" multiple times to the model's generation when it tries to end. This can lead the model to double-check its answer, often fixing incorrect reasoning steps. After supervised finetuning the Qwen2.5-32B-Instruct language model on **s1K** and equipping it with budget forcing, our model **s1-32B** exceeds o1-preview on competition math questions by up to 27% (MATH and AIME24). Further, scaling **s1-32B** with budget forcing allows extrapolating beyond its performance without test-time intervention: from 50% to 57% on AIME24. Our model, data, and code are open-source at https://github.com/simplescaling/s1.

Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

1. Introduction

Performance improvements of language models (LMs) over the past years have largely relied on scaling up train-time compute using large-scale self-supervised pretraining (Kaplan et al., 2020; Hoffmann et al., 2022). The creation of these powerful models has set the stage for a new scaling paradigm built on top of them: *test-time scaling*. The aim of this approach is to increase the compute at test time to get better results. There has been much work exploring this idea (Snell et al., 2024; Welleck et al., 2024), and the viability of this paradigm was recently validated by OpenAI o1 (OpenAI, 2024). o1 has demonstrated strong reasoning performance with consistent gains from scaling test-time compute. OpenAI describes their approach as using large-scale reinforcement learning (RL) implying the use of sizable amounts of data (OpenAI, 2024). This has led to various attempts to replicate their models relying on techniques like Monte Carlo Tree Search (Gao et al., 2024b; Zhang et al., 2024a), multi-agent approaches (Qin et al., 2024), and others (Wang et al., 2024a; Huang et al., 2024b; 2025). Among these approaches, DeepSeek R1 (DeepSeek-AI et al., 2025) has successfully replicated o1-level performance, also employing reinforcement learning via millions of samples and multiple training stages. However, despite the large number of o1 replication attempts, none have openly replicated a clear test-time scaling behavior. Thus, we ask: what is the simplest approach to achieve both test-time scaling and strong reasoning performance?

We show that training on only 1,000 samples with next-token prediction and controlling thinking duration via a simple test-time technique we refer to as *budget forcing* leads to a strong reasoning model that scales in performance with more test-time compute. Specifically, we construct **s1K**, which consists of 1,000 carefully curated questions paired with reasoning traces and answers distilled from Gemini Thinking Experimental (Google, 2024). We perform supervised fine-tuning (SFT) of an off-the-shelf pretrained model

[*]Equal contribution. ZY and NM started the project. WS, NM and ZY collected the prompts, XL, ZY and NM, built the data pipeline, LZ and WS proposed using a 1K subset and ZY and NM built budget forcing. [1] Stanford University. [2] University of Washington, Seattle. [3] Allen Institute for AI. [4] Contextual AI.

s1: Simple test-time compute/scaling

Today

Quick note on scribe and discussion lead assignments:

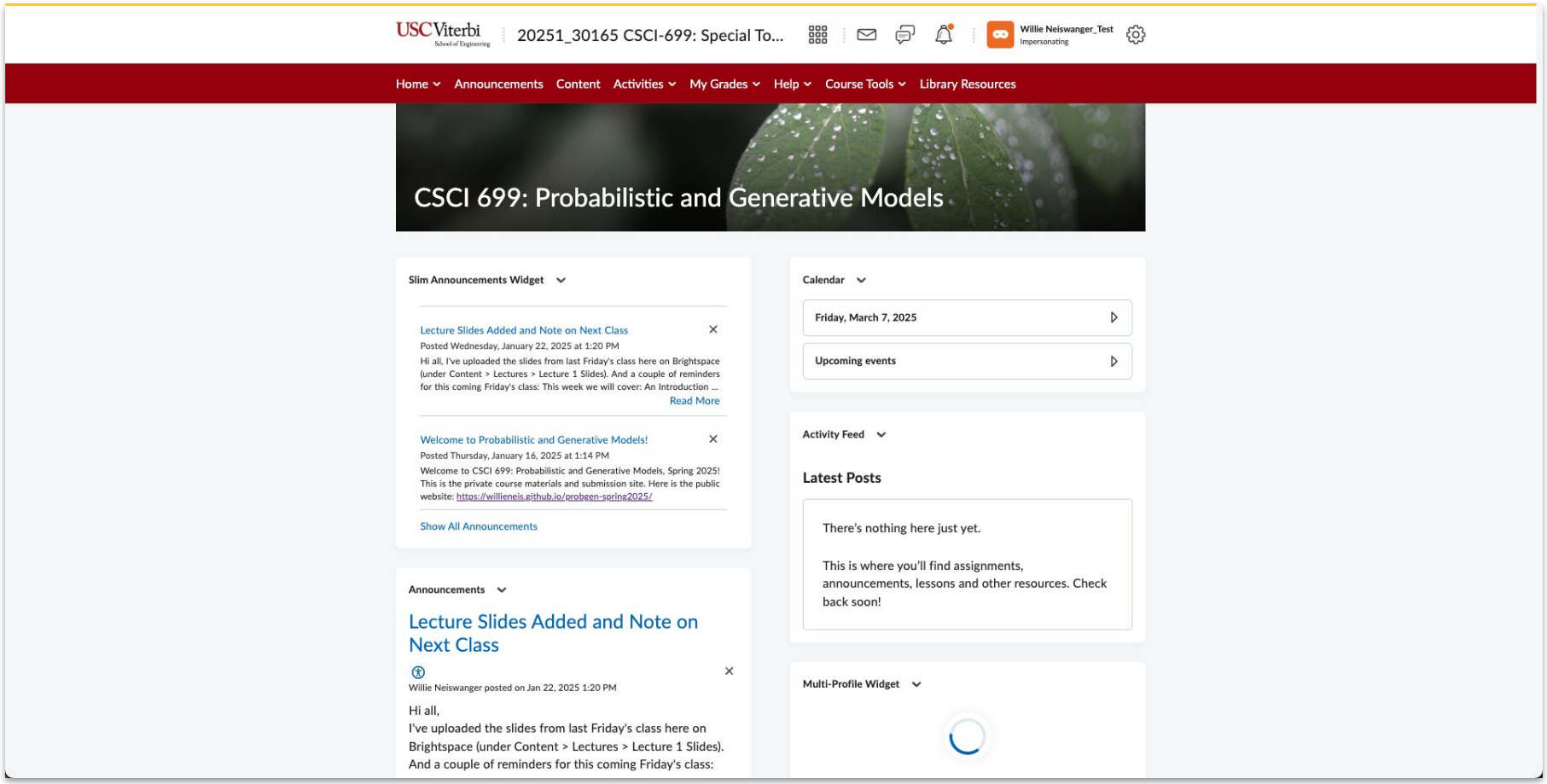
- As you know, discussion leads bring ~5 questions on their assigned paper, and scribes write up ~1 page document.
- Both assignments should be uploaded as a **PDF file** on Brightspace.
- Ideally within a ~week of your scribe/discussion lead duties.

Today

Here’s how to upload:

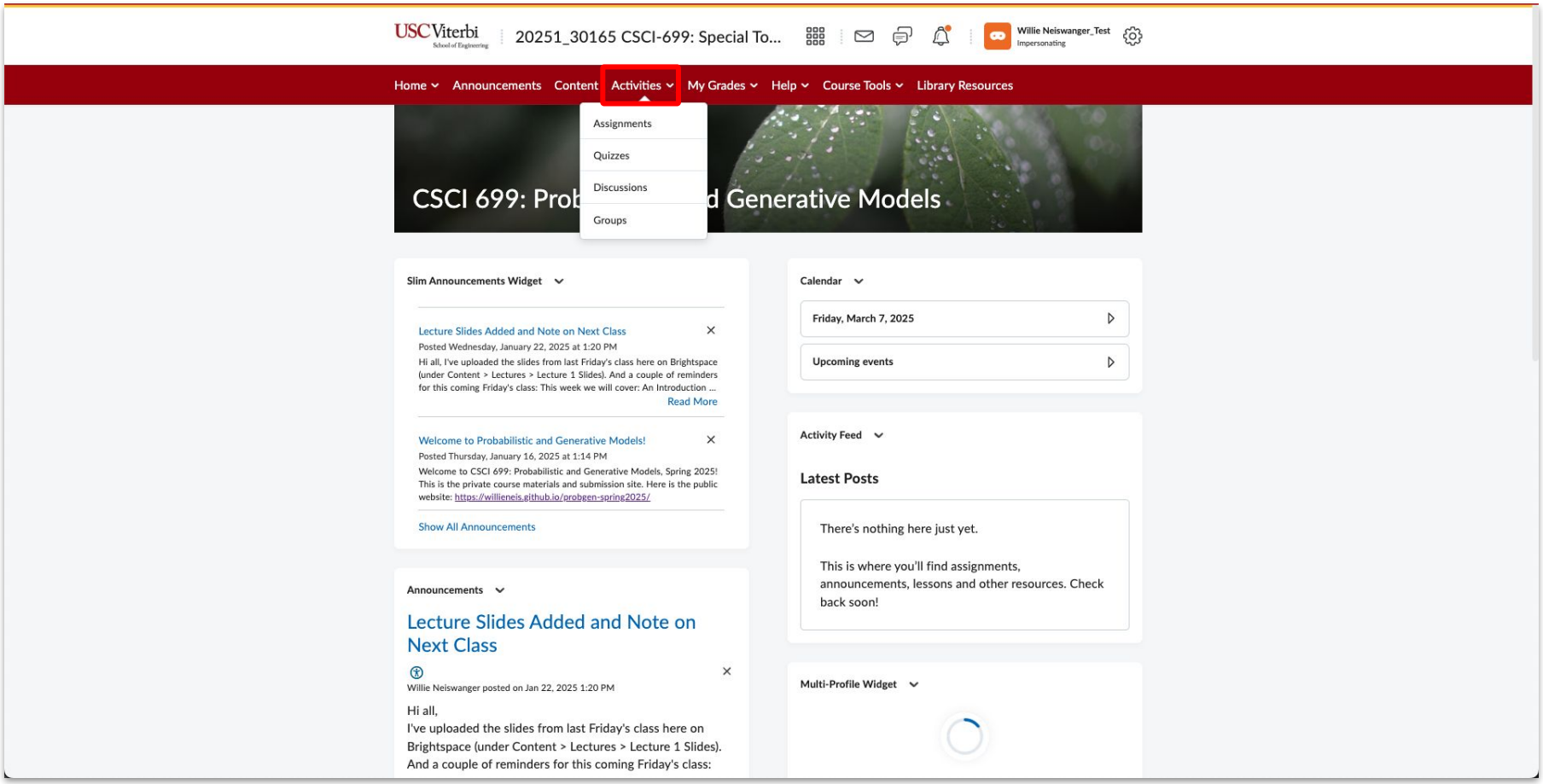
Today

Here's how to upload: (1) Go to main course homepage on Brightspace.



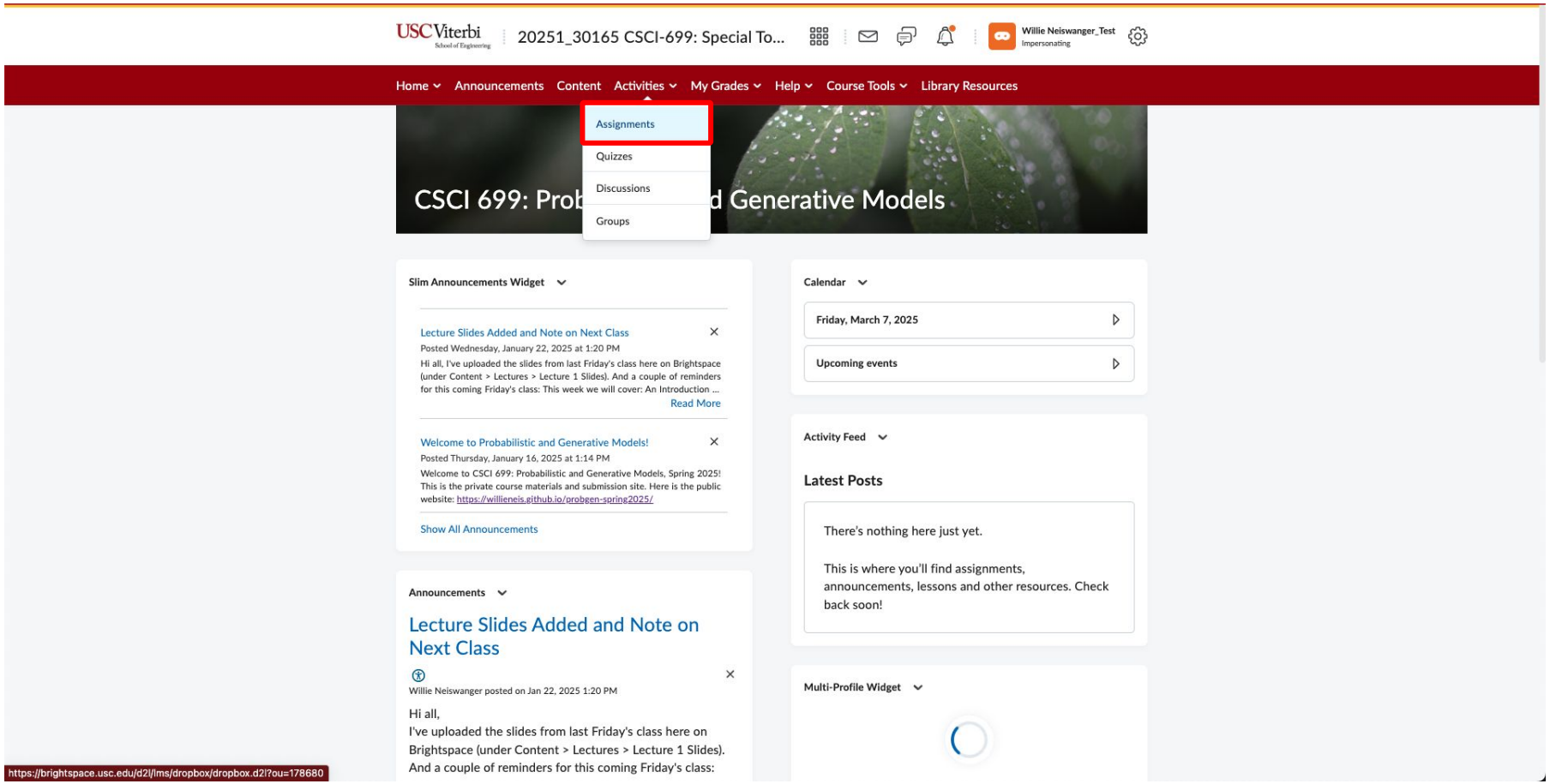
Today

Here's how to upload: (2) Click on *Activities* > *Assignments* in menu bar at top.



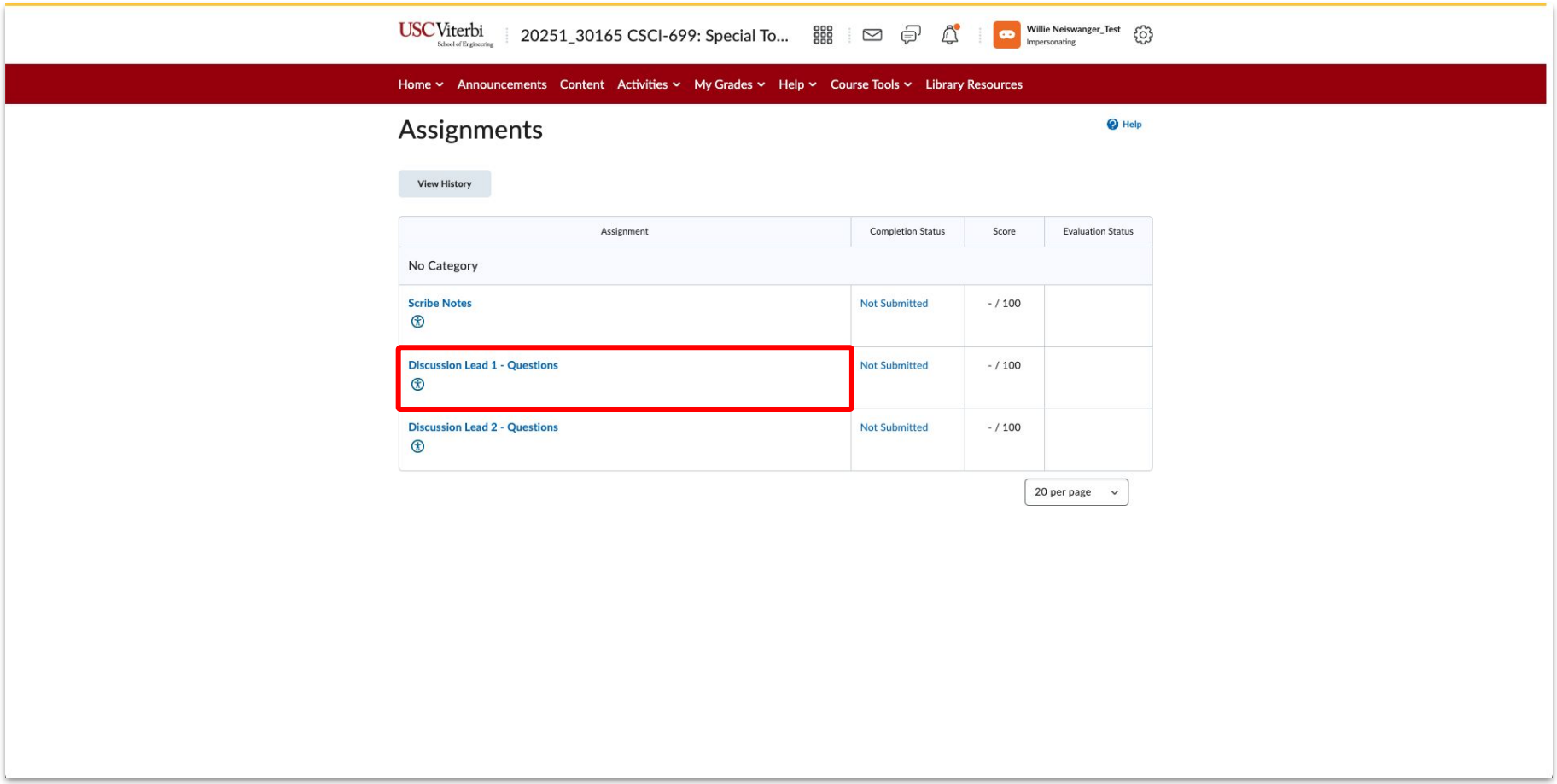
Today

Here's how to upload: (2) Click on *Activities* > *Assignments* in menu bar at top.



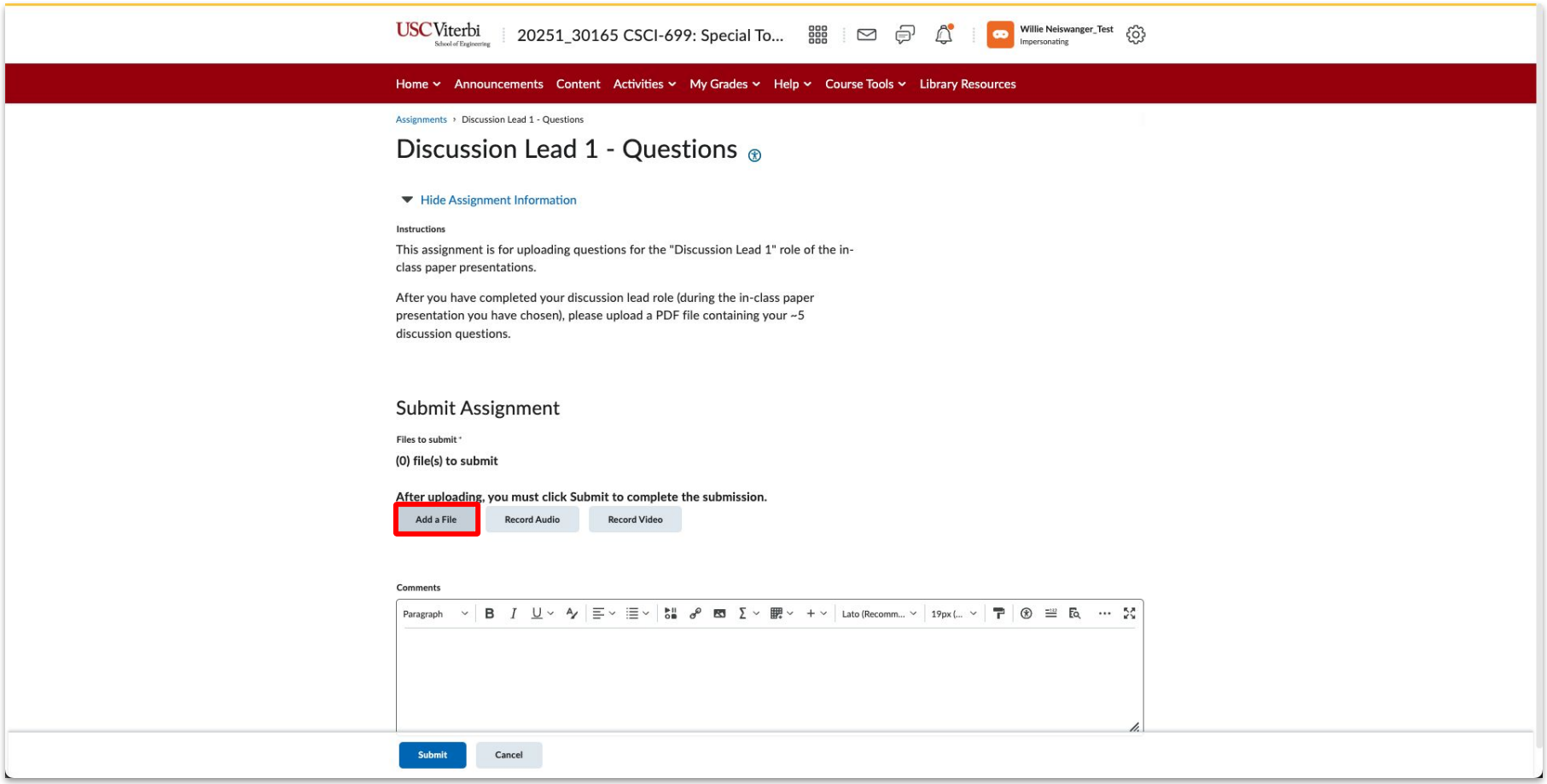
Today

Here's how to upload: (3) Click on any of the listed assignments.



Today

Here's how to upload: (4) Upload PDF (under *Add a File*)



Note — Next Week's Lecture

Note — Next Week's Lecture

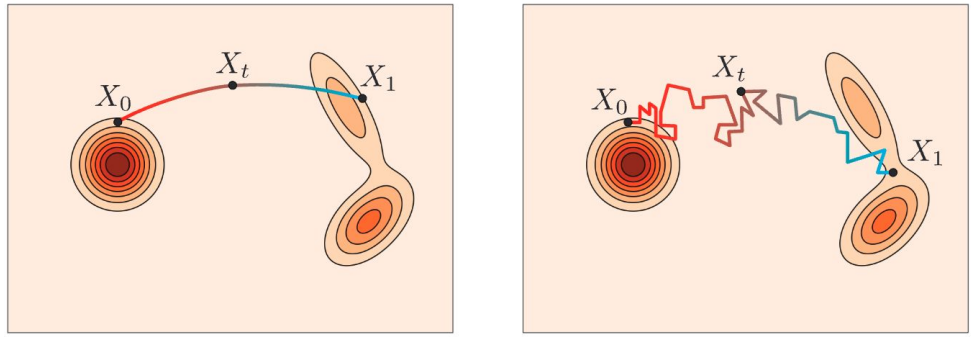
Will be given by our very own Oliver Liu!

Oliver Liu
Teaching Assistant

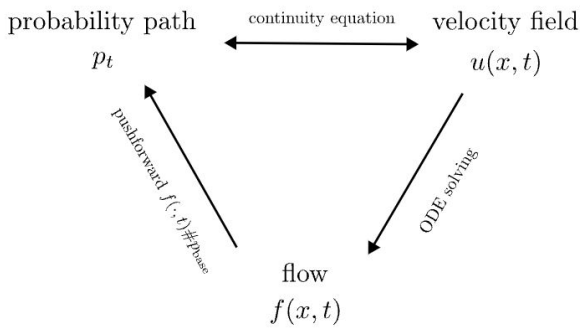
Note — Next Week's Lecture

Will be given by our very own Oliver Liu!

On the topic of **flow matching** ⇒



(a) Flow **(b)** Diffusion



"Flow Matching Guide and Code", Lipman et al., 2024

"A Visual Dive into Conditional Flow Matching", 2024

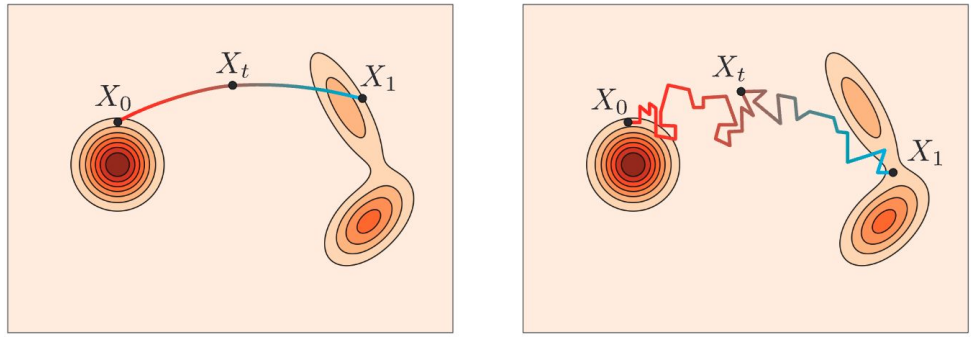
Note — Next Week's Lecture

Will be given by our very own Oliver Liu!

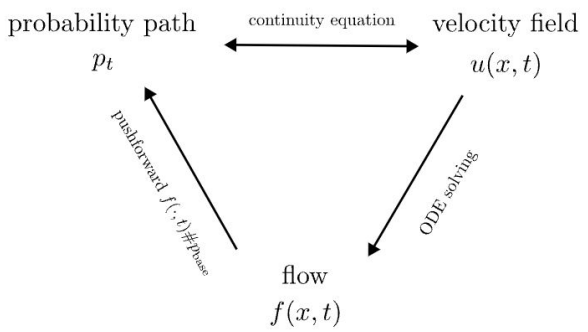
On the topic of **flow matching** ⇒

A good companion to the score-matching + diffusion lecture we are doing today.

And will discuss another key deep generative modeling paradigm: *flow-based models*.



(a) Flow **(b)** Diffusion



"Flow Matching Guide and Code", Lipman et al., 2024

"A Visual Dive into Conditional Flow Matching", 2024

Quick Review: VAE

Review — VAE Original Paper

(From ICLR 2014)

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound with an independent noise variable yields a lower bound estimator that can be jointly optimized w.r.t. variational and generative parameters using standard gradient-based stochastic optimization methods. Second, we show that posterior inference can be made especially efficient by optimizing a probabilistic encoder (also called a recognition model) to approximate the intractable posterior, using the proposed estimator. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we efficiently learn the parameters of directed probabilistic models whose continuous latent variables have intractable posterior distributions? The variational approach to Bayesian inference involves the introduction of an approximation to the intractable posterior, used to maximize the variational lower bound on the marginal likelihood. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a practical differentiable estimator of the lower bound. This SGVB (Stochastic Gradient Variational Bayes) estimator can be straightforwardly used as a stochastic objective function, and that can be jointly optimized w.r.t. both the variational and generative parameters, using standard stochastic gradient ascent techniques.

The SGVB algorithm can be applied to learning almost any generative model with continuous latent variables. When we use a neural network for the posterior approximation, we arrive at a *variational auto-encoder*. The SGVB objective for this case contains a regularization term dictated by the variational bound, and a stochastic data reconstruction term. From the learned generative model it is straightforward to generate samples simply by ancestral sampling. The probabilistic encoder can be used for fast approximate inference of latent variables, i.e. for recognition, representation or visualization purposes. Furthermore, the lower bound estimator can be used for inference tasks such as denoising and inpainting.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference

Review — VAE Original Paper

(From ICLR 2014)

Introduces two distinct things:

(1) *Auto-encoding variational Bayes* (AEVB) algorithm.
(a stochastic-gradient VI procedure)

(2) A special case (involving neural networks): the *variational autoencoder* (VAE).

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound with an independent noise variable yields a lower bound estimator that can be jointly optimized w.r.t. variational and generative parameters using standard gradient-based stochastic optimization methods. Second, we show that posterior inference can be made especially efficient by optimizing a probabilistic encoder (also called a recognition model) to approximate the intractable posterior, using the proposed estimator. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we efficiently learn the parameters of directed probabilistic models whose continuous latent variables have intractable posterior distributions? The variational approach to Bayesian inference involves the introduction of an approximation to the intractable posterior, used to maximize the variational lower bound on the marginal likelihood. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a practical differentiable estimator of the lower bound. This SGVB (Stochastic Gradient Variational Bayes) estimator can be straightforwardly used as a stochastic objective function, and that can be jointly optimized w.r.t. both the variational and generative parameters, using standard stochastic gradient ascent techniques.

The SGVB algorithm can be applied to learning almost any generative model with continuous latent variables. When we use a neural network for the posterior approximation, we arrive at a *variational auto-encoder*. The SGVB objective for this case contains a regularization term dictated by the variational bound, and a stochastic data reconstruction term. From the learned generative model it is straightforward to generate samples simply by ancestral sampling. The probabilistic encoder can be used for fast approximate inference of latent variables, i.e. for recognition, representation or visualization purposes. Furthermore, the lower bound estimator can be used for inference tasks such as denoising and inpainting.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference

1

Review — AEVB Algorithm

The Auto-encoding Variational Bayes (AEVB) algorithm aims to solve the tasks from before, in particular:

(1) Learning the parameters θ of our model p_θ.

The task of generative modeling

(2) Approximate posterior inference over z: $p_\theta(z \mid x)$

We'll start with methods for approximate posterior inference, and then show how this can lead to learning the parameters (and thus to generative modeling!)

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Review — The ELBO

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Review — The ELBO

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Consider an ELBO for this model, written as:

$$\mathcal{L}(p_\theta, q_\phi) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right]$$

Log of joint model (unnormalized PDF)

Log of approximate PDF

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Review — The ELBO

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Consider an ELBO for this model, written as:

$$\mathcal{L}(p_\theta, q_\phi) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right]$$

Log of joint model (unnormalized PDF)

Log of approximate PDF

Note that the ELBO satisfies:

$$\log p_\theta(x) = \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z \mid x)\right] + \mathcal{L}(p_\theta, q_\phi).$$

Evidence equals KL divergence plus ELBO.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Note that an expression for this expectation in closed form is often *not possible*.

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Note that an expression for this expectation in closed form is often *not possible*.

Instead, we could take gradient of a Monte Carlo estimate by sampling z from q_ϕ .

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the **gradient for p_θ:**

$$\nabla_\theta \mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

$$\approx \frac{1}{M}\sum_{m=1}^{M} \nabla_\theta \log p_\theta(x, z^{(m)})$$

where $z^{(m)} \sim q_\phi(z)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the **gradient for p_θ:**

$$\nabla_\theta\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

$$\approx \frac{1}{M}\sum_{m=1}^{M} \nabla_\theta \log p_\theta(x, z^{(m)})$$

where $z^{(m)} \sim q_\phi(z)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the **gradient for p_θ:**

$$\nabla_\theta\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

Just swap the gradient and expectation here.

$$\approx \frac{1}{M}\sum_{m=1}^{M} \nabla_\theta \log p_\theta(x, z^{(m)})$$

where $z^{(m)} \sim q_\phi(z)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the **gradient for p_θ:**

$$\nabla_\theta\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

$$\approx \frac{1}{M}\sum_{m=1}^{M} \nabla_\theta \log p_\theta(x, z^{(m)})$$

where $z^{(m)} \sim q_\phi(z)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

But it's harder to write out the **gradient for** q_ϕ **:**

$$\nabla_{\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \ ??$$

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

But it's harder to write out the **gradient for** q_ϕ **:**

$$\nabla_{\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \ ??$$

Note that we cannot just swap the gradient and expectation, because the expectation is being taken with respect to $q_\phi(z)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Notably: since the gradient is now inside the expectation, we can evaluate this expectation using Monte Carlo methods!

Expectation with respect to q of…

Log joint minus log q (same as in the ELBO)

Multiplied by the gradient of the log of q.

Follows from some basic algebra/calculus, and takes about half a page to derive :D.

⇒ Now can estimate via Monte Carlo methods.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator → SGVB Estimator

One key contribution of the VAE paper is to propose an alternative estimator, which is better behaved (lower variance).

Sometimes referred to as the *stochastic gradient variational Bayes* (SGVB) estimator.

This is done in two steps:

(1) We reformulate the ELBO so that parts of it can be computed in closed form (*i.e.*, without Monte Carlo).

(2) Then we use an alternative gradient estimator (SGVB) based on the so-called **reparameterization trick**.

Reformulating the ELBO

The reformulation of the ELBO is as follows:

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z) \right] - \mathrm{KL} \left[q_\phi(z \mid x) || p_\theta(z) \right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

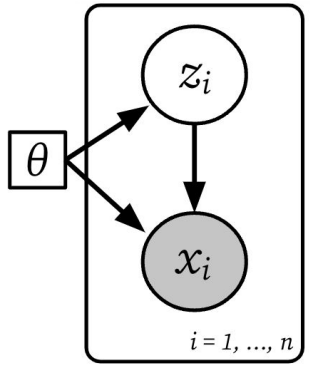
Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*



We know the log-likelihood (typically chosen when we define our probabilistic model).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Reformulating the ELBO

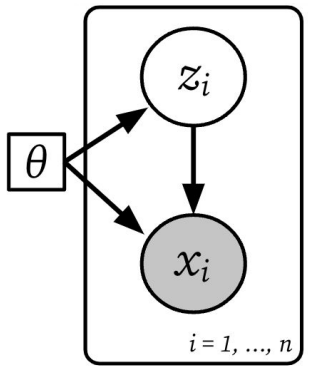
The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*



The prior is also known/chosen when we first define our model.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

The first term:

- ⇒ *i.e.*, It is trying to *reconstruct* x *given the code* z *(drawn from* $q_\phi(z \mid x)$*).*
- ⇒ We will therefore call $p_\theta(x \mid z)$ the **decoder**.
  - And the first term we call the *reconstruction error.*

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

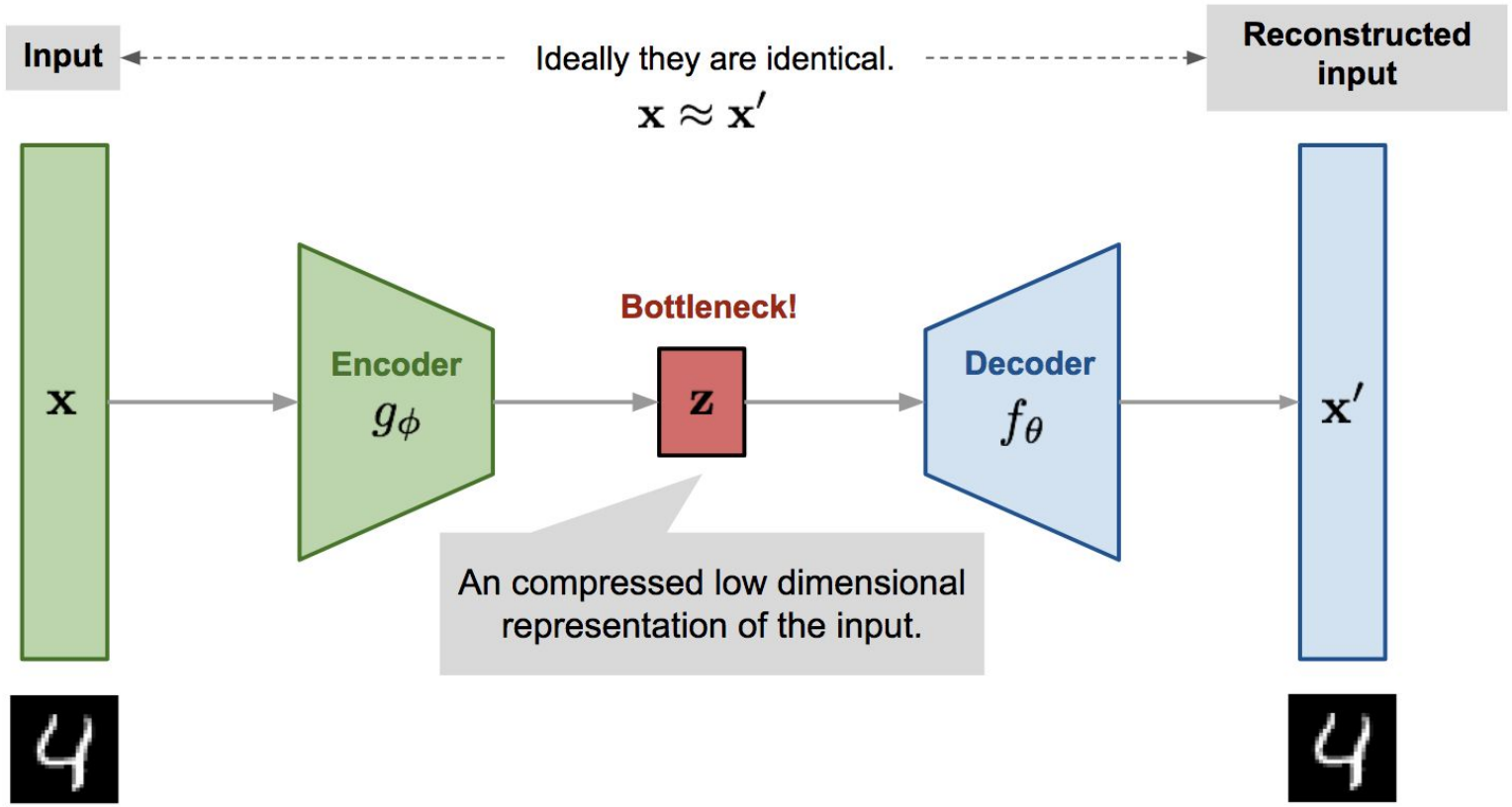
The first term:

- ⇒ *i.e.*, It is trying to *reconstruct* x *given the code* z *(drawn from* $q_\phi(z \mid x)$*).*
- ⇒ We will therefore call $p_\theta(x \mid z)$ the **decoder**.
  - And the first term we call the *reconstruction error.*
- … resembles a classic Autoencoder neural network.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reconstruction Term ⇔ Autoencoder

Aside — illustration of an **autoencoder** (first developed in 1990s, or even late 1980s).



Source: Lil'Log, "From Autoencoder to Beta-VAE"

Reparameterization Trick

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Under certain mild conditions, we can express $q_\phi(z \mid x)$ via the following two-step generative process:

Reparameterization Trick

Under certain mild conditions, we can express $q_\phi(z \mid x)$ via the following two-step generative process:

- First sample a noise variable ϵ from a simple distribution $p(\epsilon)$ (such as a standard Normal $\mathcal{N}(0, I)$):

$$\epsilon \sim p(\epsilon)$$

Note: this does not depend on ϕ.

Reparameterization Trick

Under certain mild conditions, we can express $q_\phi(z \mid x)$ via the following two-step generative process:

- First sample a noise variable ϵ from a simple distribution $p(\epsilon)$ (such as a standard Normal $\mathcal{N}(0, I)$):

$$\epsilon \sim p(\epsilon)$$

Note: this does not depend on ϕ.

- Then we apply a deterministic transformation $g_\phi(\epsilon, x)$ that maps the random noise into a more-complex distribution:

$$z = g_\phi(\epsilon, x)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x, g_\phi(\epsilon, x))\right]$$

$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x, g_\phi(\epsilon, x))\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

Replace q with $p(\epsilon)$, and z with $g(\epsilon, x)$.

$$\nabla_\phi \mathbb{E}_{z \sim q_\phi(z|x)} [f(x, z)] = \nabla_\phi \mathbb{E}_{\epsilon \sim p(\epsilon)} [f(x, g_\phi(\epsilon, x))]$$

$$= \mathbb{E}_{\epsilon \sim p(\epsilon)} [\nabla_\phi f(x, g_\phi(\epsilon, x))]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x, g_\phi(\epsilon,x))\right]$$

$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x, g_\phi(\epsilon,x))\right]$$

Replace q with $p(\epsilon)$, and z with $g(\epsilon, x)$.

Can now swap gradient and expectation!

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x, g_\phi(\epsilon,x))\right]$$

$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x, g_\phi(\epsilon,x))\right]$$

Replace q with $p(\epsilon)$, and z with $g(\epsilon, x)$.

Can now swap gradient and expectation!

Can now take a Monte Carlo estimate of this expectation.

⇒ yields an estimate of the gradient for ϕ, with much lower variance than the score-function estimator.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

So what is the variational autoencoder (VAE), and how does it relate to the AEVB algorithm?

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x)||p_\theta(z)\right]$$

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x)||p_\theta(z)\right]$$

Expectation of the *log-likelihood* — KL between q *posterior* and p *prior*

(2) The black-box (gradient-based) variational inference approach.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* — KL between q *posterior* and p *prior*

(2) The black-box (gradient-based) variational inference approach.

(3) The reparameterization-based SGVB (low-variance) estimator.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* — KL between *q posterior* and *p prior*

(2) The black-box (gradient-based) variational inference approach.

(3) The reparameterization-based SGVB (low-variance) estimator.

⇒ takes gradient steps on θ and ϕ to optimize the alternative ELBO.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

$$p_\theta(z) = \mathcal{N}(z;\ 0, I)$$
$$p_\theta(x \mid z) = \mathcal{N}\left(x;\ \mu_\theta(z), \mathrm{diag}(\sigma_\theta(z))^2\right)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

Prior on latent variable z is a Gaussian.

$$p_\theta(z) = \mathcal{N}(z;\ 0, I)$$

$$p_\theta(x \mid z) = \mathcal{N}\left(x;\ \mu_\theta(z), \mathrm{diag}(\sigma_\theta(z))^2\right)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

Prior on latent variable z is a Gaussian.

$$p_\theta(z) = \mathcal{N}(z;\ 0, I)$$

$$p_\theta(x \mid z) = \mathcal{N}\left(x;\ \mu_\theta(z), \mathrm{diag}(\sigma_\theta(z))^2\right)$$

Decode latent variable z into data point x.
(Also Gaussian).

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The posterior approximation q_ϕ is parameterized as:

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The posterior approximation q_ϕ is parameterized as:

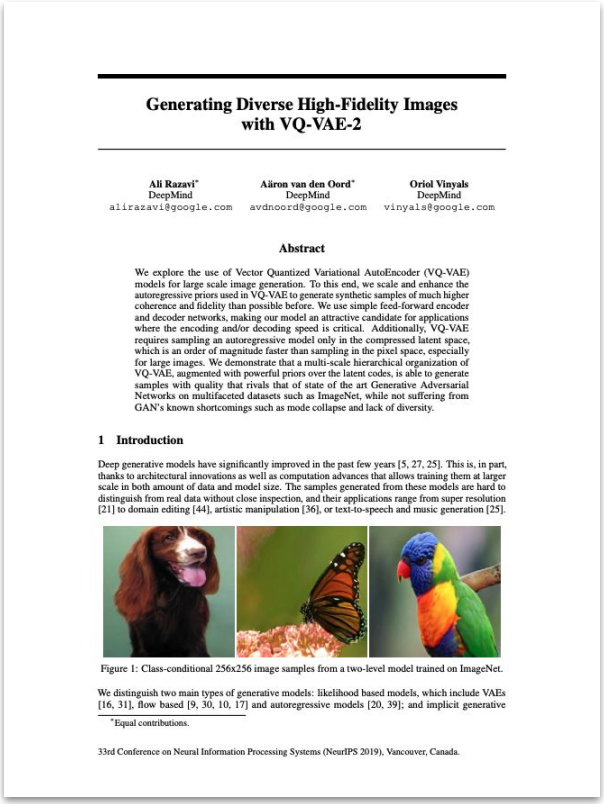
$$q_\phi(z \mid x) = \mathcal{N}\left(z;\ \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))^2\right)$$

Encode data point x as latent variable z.
(Also Gaussian).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE) — Results

So does it work? → A few years later … (*NeurIPS 2019*)



Generating Diverse High-Fidelity Images with VQ-VAE-2

Ali Razavi* DeepMind alirazavi@google.com
Aäron van den Oord* DeepMind avdnoord@google.com
Oriol Vinyals DeepMind vinyals@google.com

Abstract

We explore the use of Vector Quantized Variational AutoEncoder (VQ-VAE) models for large scale image generation. To this end, we scale and enhance the autoregressive priors used in VQ-VAE to generate synthetic samples of much higher coherence and fidelity than possible before. We use simple feed-forward encoder and decoder networks, making our model an attractive candidate for applications where the encoding and/or decoding speed is critical. Additionally, VQ-VAE requires sampling an autoregressive model only in the compressed latent space, which is an order of magnitude faster than sampling in the pixel space, especially for large images. We demonstrate that a multi-scale hierarchical organization of VQ-VAE, augmented with powerful priors over the latent codes, is able to generate samples with quality that rivals that of state of the art Generative Adversarial Networks on multifaceted datasets such as ImageNet, while not suffering from GAN's known shortcomings such as mode collapse and lack of diversity.

1 Introduction

Deep generative models have significantly improved in the past few years [5, 27, 25]. This is, in part, thanks to architectural innovations as well as computation advances that allows training them at larger scale in both amount of data and model size. The samples generated from these models are hard to distinguish from real data without close inspection, and their applications range from super resolution [21] to domain editing [44], artistic manipulation [36], or text-to-speech and music generation [25].

Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

We distinguish two main types of generative models: likelihood based models, which include VAEs [16, 31], flow based [9, 30, 10, 17] and autoregressive models [20, 39]; and implicit generative

*Equal contributions.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

FFQH Dataset

Source: Razavi, Ali, Aaron Van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2", 2019.

Intro to Score-based Generative Models

VAEs $\rightarrow$ Score-based/Diffusion Generative Models

VAEs → Score-based/Diffusion Generative Models

VAEs and GANs were the two most-popular generative models for a few years, until around 2019*.

When a new paradigm of generative modeling came around…

**Actually, the first diffusion paper was in 2015 (or before), but it didn't really gain much attention until around 2019.*

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024 × 1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

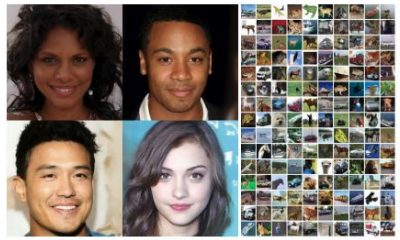


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024 × 1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

First Diffusion Model (ICML 2015)

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

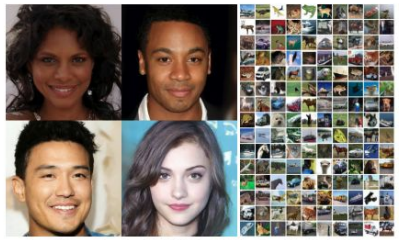


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024 × 1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

Score-based Generative Model (NeurIPS 2019)

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

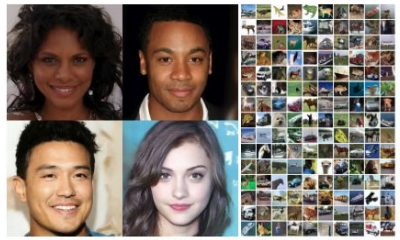


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024 × 1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

Denoising Diffusion Prob Model (NeurIPS 2020)

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song Stanford University yangsong@cs.stanford.edu
Stefano Ermon Stanford University ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho UC Berkeley jonathanho@berkeley.edu
Ajay Jain UC Berkeley ajayj@berkeley.edu
Pieter Abbeel UC Berkeley pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

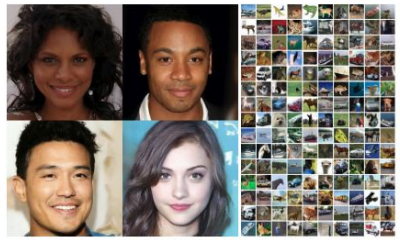


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song* Stanford University yangsong@cs.stanford.edu
Jascha Sohl-Dickstein Google Brain jaschasd@google.com
Diederik P. Kingma Google Brain durk@google.com
Abhishek Kumar Google Brain abhishk@google.com
Stefano Ermon Stanford University ermon@cs.stanford.edu
Ben Poole Google Brain pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024 × 1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

Score-based Models via SDEs (ICLR 2021)

Score-based/Diffusion Generative Models

Score-based and diffusion generative models take advantage of the approximate inference algorithms we have recently learned!

Score-based/Diffusion Generative Models

Score-based and diffusion generative models take advantage of the approximate inference algorithms we have recently learned!

Score-based generative models → [today]

- Involve Langevin Monte Carlo (LMC) with an annealed proposal distribution.

Score-based/Diffusion Generative Models

Score-based and diffusion generative models take advantage of the approximate inference algorithms we have recently learned!

Score-based generative models → [today]

- Involve Langevin Monte Carlo (LMC) with an annealed proposal distribution.

Denoising diffusion generative models → [later/upcoming]

- Involve variational inference (VI) via ELBO maximization.

Score-based Generative Models — Setup

Score-based Generative Models — Setup

Suppose that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$

Where each data point is assumed to be drawn from some unknown distribution, *i.e.*,

$$x_i \sim p(x)$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

Suppose that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$

Where each data point is assumed to be drawn from some unknown distribution, *i.e.*,

$$x_i \sim p(x)$$

Suppose we want to model this distribution using a model $p_\theta(x)$, where we assume this distribution is parameterized by a $\theta \in \mathbb{R}^d$.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

Suppose that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$

Where each data point is assumed to be drawn from some unknown distribution, *i.e.*,

$$x_i \sim p(x)$$

Suppose we want to model this distribution using a model $p_\theta(x)$, where we assume this distribution is parameterized by a $\theta \in \mathbb{R}^d$.

⇒ We want to learn θ such that $p_\theta(x) \approx p(x)$.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x) = \frac{1}{Z(\theta)} e^{-f_\theta(x)}$$

Where, as before,

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x) = \frac{1}{Z(\theta)} e^{-f_\theta(x)}$$

Where, as before, $Z(\theta)$ is the normalization constant.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x) = \frac{1}{Z(\theta)} e^{-f_\theta(x)}$$

Where, as before, $Z(\theta)$ is the normalization constant.

And $\tilde{p}_\theta(x)$ is an unnormalized PDF.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x) = \frac{1}{Z(\theta)} e^{-f_\theta(x)}$$

Where, as before, $Z(\theta)$ is the normalization constant.

And $\tilde{p}_\theta(x)$ is an unnormalized PDF.

And $f_\theta(x)$ is equal to the negative log of the unnormalized PDF, *i.e.*,

$$f_\theta(x) = -\log \tilde{p}_\theta(x)$$

Sometimes called an *energy-based model*.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Difficulties with MLE

To perform generative modeling, we want to learn model parameters θ.

Score-based Generative Models — Difficulties with MLE

To perform generative modeling, we want to learn model parameters θ.

Previous strategies involve trying to learning parameters θ via maximum likelihood:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_\theta(x_i)$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Difficulties with MLE

To perform generative modeling, we want to learn model parameters θ.

Previous strategies involve trying to learning parameters θ via maximum likelihood:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_\theta(x_i)$$

However:

- To optimize (or evaluate) $p_\theta(x)$, we need the normalization constant $Z(\theta)$.
- …which is intractable in general for most $\tilde{p}_\theta(x)$.
- In VAEs, we perform MLE on θ via the use of an approximate posterior q.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Main Strategy

So instead, we will take the following strategy:

Score-based Generative Models — Main Strategy

So instead, we will take the following strategy:

(1) **First**, we will aim to learn what is referred to as the **(Stein) score function** of the data density.

(2) **Second**, Given this score function, we will be able to draw samples from $p_\theta(x)$ using Langevin Monte Carlo.

Score-based Generative Models — Main Strategy

So instead, we will take the following strategy:

(1) **First**, we will aim to learn what is referred to as the **(Stein) score function** of the data density.

(2) **Second**, Given this score function, we will be able to draw samples from $p_\theta(x)$ using Langevin Monte Carlo.

Going through these two steps…

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

Gradient of the PDF with respect to x

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

Gradient of the PDF with respect to x

Note that:

- The term *score* is also sometimes used to refer to the gradient of the log-likelihood function with respect to the parameter θ.
- We've seen this term before! (Recall in LMC/HMC…).

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

$$= \nabla_x \log \tilde{p}_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = \nabla_x \log \tilde{p}(x)$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

Definition of unnormalized PDF

$$= \nabla_x \log \tilde{p}_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = \nabla_x \log \tilde{p}(x)$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

Definition of unnormalized PDF

$$= \nabla_x \log \tilde{p}_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = \nabla_x \log \tilde{p}(x)$$

Second term is 0 (constant wrt x)

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

Definition of unnormalized PDF

$$= \nabla_x \log \tilde{p}_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = \nabla_x \log \tilde{p}(x)$$

Second term is 0 (constant wrt x)

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

$$= -\nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = -\nabla_x f_\theta(x)$$

Also (in terms of energy based models):

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

$$= -\nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = -\nabla_x f_\theta(x)$$

Also (in terms of energy based models):

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

$$= -\nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = -\nabla_x f_\theta(x)$$

Also (in terms of energy based models):

Definition of energy-based model

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the data density $p(x)$, *i.e.*,

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

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

$$= -\nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = -\nabla_x f_\theta(x)$$

Also (in terms of energy based models):

Definition of energy-based model

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

Suppose we have observed a set of data: $D = \{x_1, x_2, \ldots, x_n\} \overset{\text{iid}}{\sim} p(x)$.

And also suppose that we have (somehow) used D to learn an approximation $s_\theta(x)$ to the true score function:

Score-based Generative Models

Suppose we have observed a set of data: $D = \{x_1, x_2, \ldots, x_n\} \overset{\text{iid}}{\sim} p(x)$.

And also suppose that we have (somehow) used D to learn an approximation $s_\theta(x)$ to the true score function:

$$s_\theta(x) \approx \nabla_x \log p(x) = \nabla_x \log \tilde{p}(x)$$

I.e., it approximates the score function of the distribution that generated our data

Score-based Generative Models

Suppose we have observed a set of data: $D = \{x_1, x_2, \ldots, x_n\} \overset{\text{iid}}{\sim} p(x)$.

And also suppose that we have (somehow) used D to learn an approximation $s_\theta(x)$ to the true score function:

$$s_\theta(x) \approx \nabla_x \log p(x) = \nabla_x \log \tilde{p}(x)$$

I.e., it approximates the score function of the distribution that generated our data

Recall that in Langevin Monte Carlo (SMC):

Score-based Generative Models

Suppose we have observed a set of data: $D = \{x_1, x_2, \ldots, x_n\} \overset{\text{iid}}{\sim} p(x)$.

And also suppose that we have (somehow) used D to learn an approximation $s_\theta(x)$ to the true score function:

$$s_\theta(x) \approx \nabla_x \log p(x) = \nabla_x \log \tilde{p}(x)$$

I.e., it approximates the score function of the distribution that generated our data

Recall that in Langevin Monte Carlo (SMC):

- In order to sample from a PDF, you just need access to the gradient of the unnormalized PDF!

Score-based Generative Models

Quick LMC Recap:

Gradient Based MCMC

Suppose we have a given PDF $p(x), \quad x \in \mathbb{R}^n$.

Consider an objective function: $f(x) = -\log p(x)\,.$

The gradient of this objective function is:

$$\nabla_x f(x) = \left[\frac{d}{dx_1} f(x), \ldots, \frac{d}{dx_n} f(x)\right]^\top$$

Score-based Generative Models

Quick LMC Recap:

Gradient Based MCMC — Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

$$x_t = x_{t-1} - \frac{\epsilon_t}{2}\nabla_x f(x_{t-1}) + \sqrt{\epsilon_t}\mathcal{N}(0, I_n)$$

Main difference is an extra "stochastic term"

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

Can view this as "*gradient descent plus noise*"

Compare with gradient descent

Score-based Generative Models

Therefore (after learning the score function):

Score-based Generative Models

Therefore (after learning the score function):

We can simply run LMC with our learned score function $s_\theta(x)$:

$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x s_\theta(x) + \sqrt{2\epsilon}\,\mathcal{N}(0, I), \quad i = 0, 1, \cdots, K,$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

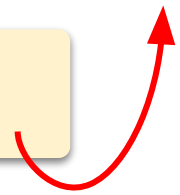
Score-based Generative Models

Therefore (after learning the score function):

We can simply run LMC with our learned score function $s_\theta(x)$:

$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x s_\theta(x) + \sqrt{2\epsilon}\,\mathcal{N}(0, I), \quad i = 0, 1, \cdots, K,$$

Note the learned score function is swapped in.



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

Therefore (after learning the score function):

We can simply run LMC with our learned score function $s_\theta(x)$:

$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x s_\theta(x) + \sqrt{2\epsilon}\,\mathcal{N}(0, I), \quad i = 0, 1, \cdots, K,$$

Note the learned score function is swapped in.

This is equivalent to previous LMC, but step size is written slightly different here.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — An illustration

An illustration:

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — An illustration

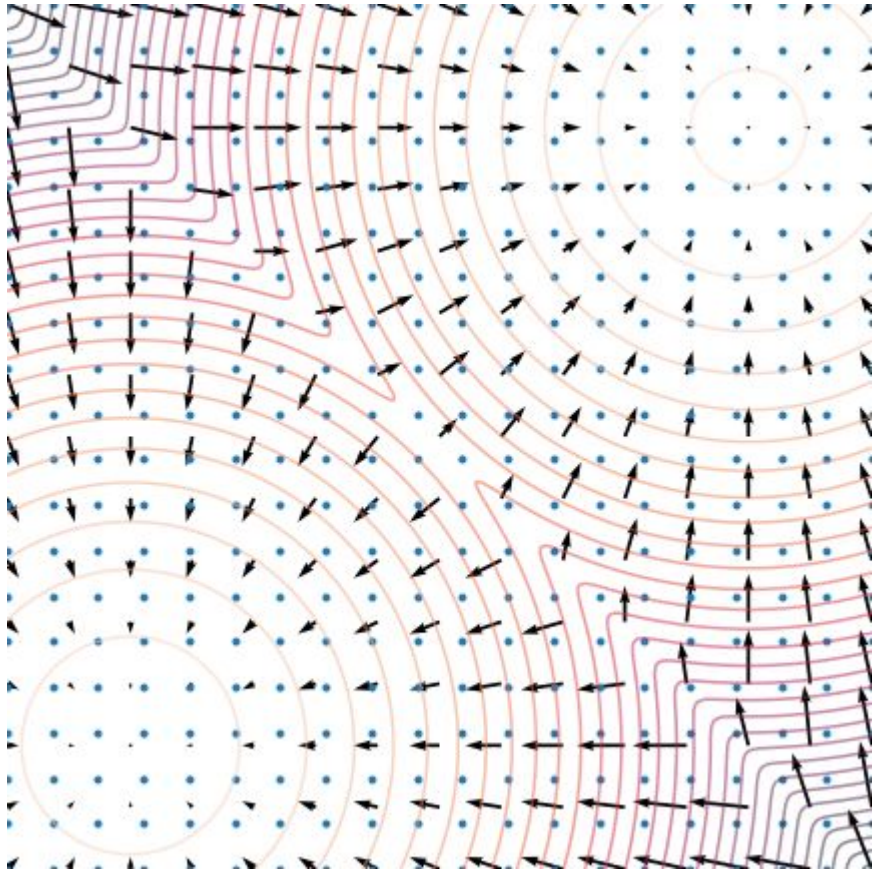
An illustration:

Contour lines denote the PDF $p_\theta(x)$.

Black arrows denote the score function $s_\theta(x)$.

Blue dots denote the samples produced via LMC.

(Many chains, randomly initialized).



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

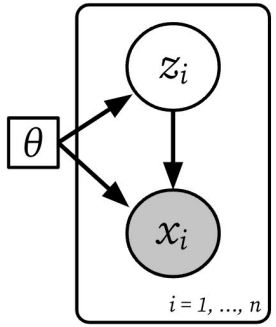
Score-based Generative Models — Intuition on Learning

What are we learning here (*e.g.*, with a neural network) versus in VAEs?

Score-based Generative Models — Intuition on Learning

What are we learning here (*e.g.*, with a neural network) versus in VAEs?

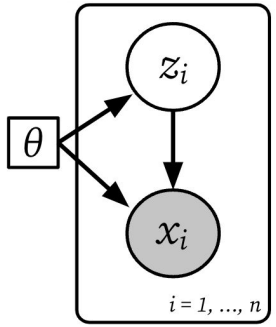
- In VAEs, we aim to learn a **direct map** from noise to data.
  - *I.e.*, directly learn the PDF of interest.
  - ⇒ a neural network that generates a sample from this PDF.



Score-based Generative Models — Intuition on Learning

What are we learning here (*e.g.*, with a neural network) versus in VAEs?

- In VAEs, we aim to learn a **direct map** from noise to data.
  - *I.e.*, directly learn the PDF of interest.
  - ⇒ a neural network that generates a sample from this PDF.



- In score-based models, we aim to learn a **score function**
  - Which we can view as pointing us in the direction of steepest ascent of PDF.

    ⇒ Use neural network *repeatedly* to take gradient steps (along with noise) to sample from this PDF.

Score-based Generative Models

So the main question is: How do we learn an approximation $s_\phi(x)$ to the score function $s_\theta(x)$?

Score-based Generative Models

So the main question is: How do we learn an approximation $s_\phi(x)$ to the score function $s_\theta(x)$?

⇒ Answer: we will minimize the (so called) *Fisher divergence* between the model and data distributions, written as:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

So the main question is: How do we learn an approximation $s_\phi(x)$ to the score function $s_\theta(x)$?

⇒ Answer: we will minimize the (so called) *Fisher divergence* between the model and data distributions, written as:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

Expectation taken over the data density.

Intuitively: this compares the squared L2 distance between the ground-truth data score function and the score-based model.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

So the main question is: How do we learn an approximation $s_\phi(x)$ to the score function $s_\theta(x)$?

⇒ Answer: we will minimize the (so called) *Fisher divergence* between the model and data distributions, written as:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

Expectation taken over the data density.

However, there is an **issue**: we don't know the ground truth PDF $p(x)$ nor ground truth data score $\nabla_x \log p(x)$).

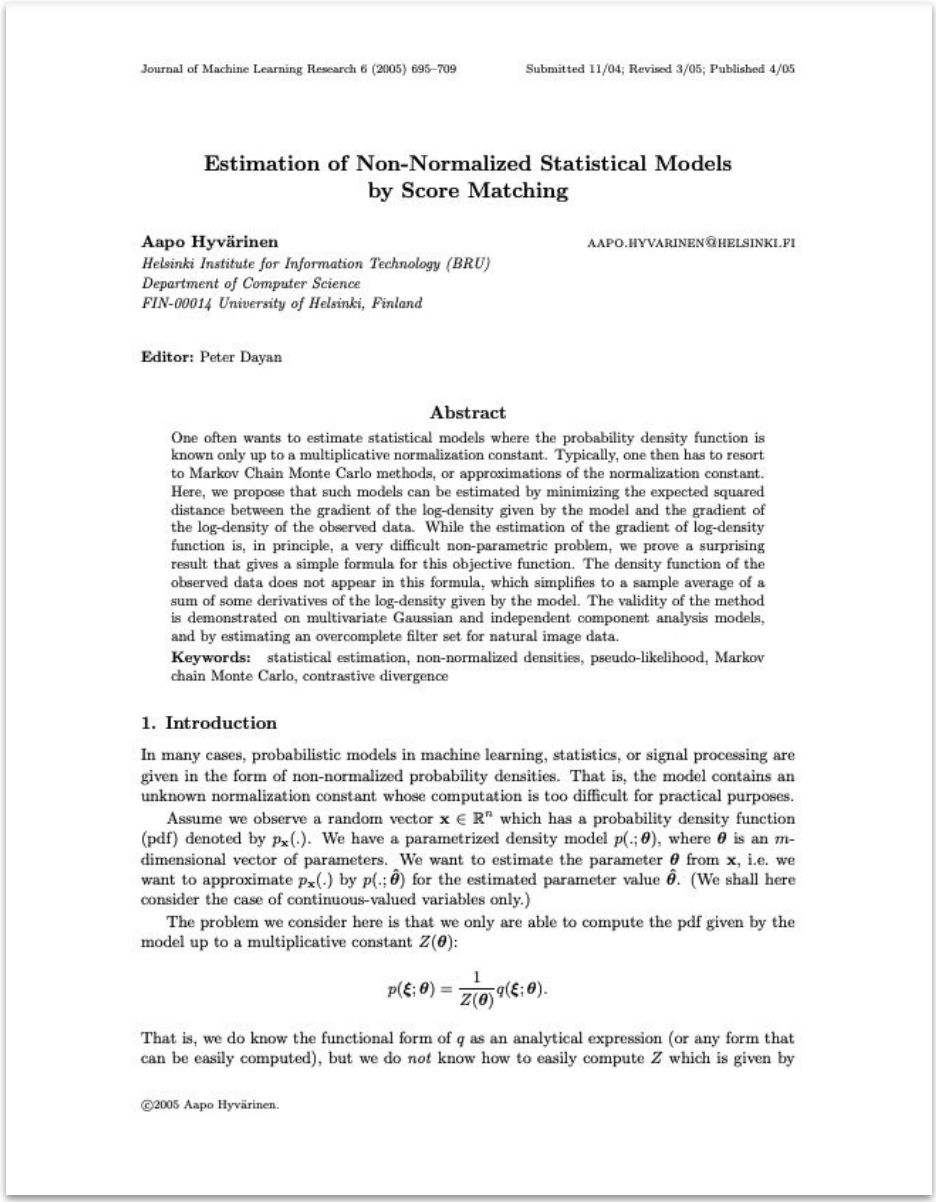
Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

Solution: Use a technique called **Score Matching.**

This will let us minimize the Fisher divergence without knowledge of the ground-truth data score.

Very clever trick!



Journal of Machine Learning Research 6 (2005) 695–709 Submitted 11/04; Revised 3/05; Published 4/05

Estimation of Non-Normalized Statistical Models by Score Matching

Aapo Hyvärinen AAPO.HYVARINEN@HELSINKI.FI
Helsinki Institute for Information Technology (BRU)
Department of Computer Science
FIN-00014 University of Helsinki, Finland

Editor: Peter Dayan

Abstract

One often wants to estimate statistical models where the probability density function is known only up to a multiplicative normalization constant. Typically, one then has to resort to Markov Chain Monte Carlo methods, or approximations of the normalization constant. Here, we propose that such models can be estimated by minimizing the expected squared distance between the gradient of the log-density given by the model and the gradient of the log-density of the observed data. While the estimation of the gradient of log-density function is, in principle, a very difficult non-parametric problem, we prove a surprising result that gives a simple formula for this objective function. The density function of the observed data does not appear in this formula, which simplifies to a sample average of a sum of some derivatives of the log-density given by the model. The validity of the method is demonstrated on multivariate Gaussian and independent component analysis models, and by estimating an overcomplete filter set for natural image data.

Keywords: statistical estimation, non-normalized densities, pseudo-likelihood, Markov chain Monte Carlo, contrastive divergence

1. Introduction

In many cases, probabilistic models in machine learning, statistics, or signal processing are given in the form of non-normalized probability densities. That is, the model contains an unknown normalization constant whose computation is too difficult for practical purposes.

Assume we observe a random vector $\mathbf{x} \in \mathbb{R}^n$ which has a probability density function (pdf) denoted by $p_{\mathbf{x}}(.)$. We have a parametrized density model $p(.;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is an m-dimensional vector of parameters. We want to estimate the parameter $\boldsymbol{\theta}$ from $\mathbf{x}$, i.e. we want to approximate $p_{\mathbf{x}}(.)$ by $p(.;\hat{\boldsymbol{\theta}})$ for the estimated parameter value $\hat{\boldsymbol{\theta}}$. (We shall here consider the case of continuous-valued variables only.)

The problem we consider here is that we only are able to compute the pdf given by the model up to a multiplicative constant $Z(\boldsymbol{\theta})$:

$$p(\boldsymbol{\xi};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} q(\boldsymbol{\xi};\boldsymbol{\theta}).$$

That is, we do know the functional form of q as an analytical expression (or any form that can be easily computed), but we do *not* know how to easily compute Z which is given by

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Get ready for some derivations :D

Start with the Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)} \left[\| \nabla_x \log p(x) - s_\theta(x) \|_2^2 \right]$$

Score Matching

Start with the Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - \nabla_x \log p_\theta(x)\|_2^2\right]$$

Definition of score model

Score Matching

Start with the Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - \nabla_x \log p_\theta(x)\|_2^2\right]$$

Expand norm of the difference

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x)\|^2 - 2(\nabla_x \log p(x))^\top(\nabla_x \log p_\theta(x)) + \|\nabla_x \log p_\theta(x)\|^2\right]$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Start with the Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - \nabla_x \log p_\theta(x)\|_2^2\right]$$

Expand norm of the difference

$$= \mathbb{E}_{p(x)}\left[\cancel{\|\nabla_x \log p(x)\|^2} - 2(\nabla_x \log p(x))^\top(\nabla_x \log p_\theta(x)) + \|\nabla_x \log p_\theta(x)\|^2\right]$$

Can ignore when optimizing for θ (is a constant)

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Start with the Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - \nabla_x \log p_\theta(x)\|_2^2\right]$$

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x)\|^2 - 2(\nabla_x \log p(x))^\top(\nabla_x \log p_\theta(x)) + \|\nabla_x \log p_\theta(x)\|^2\right]$$

$$= -2\,\mathbb{E}_{p(x)}\left[(\nabla_x \log p(x))^\top(\nabla_x \log p_\theta(x))\right] + \mathbb{E}_{p(x)}\left[\|\nabla_x \log p_\theta(x)\|^2\right] + C$$

Rewriting without first term

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Start with the Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - \nabla_x \log p_\theta(x)\|_2^2\right]$$

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x)\|^2 - 2(\nabla_x \log p(x))^\top(\nabla_x \log p_\theta(x)) + \|\nabla_x \log p_\theta(x)\|^2\right]$$

$$= -2\,\mathbb{E}_{p(x)}\left[(\nabla_x \log p(x))^\top(\nabla_x \log p_\theta(x))\right] + \mathbb{E}_{p(x)}\left[\|\nabla_x \log p_\theta(x)\|^2\right] + C$$

We can handle this term!
(via sampling from $p(x)$)

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Start with the Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - \nabla_x \log p_\theta(x)\|_2^2\right]$$

$$= \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x)\|^2 - 2(\nabla_x \log p(x))^\top(\nabla_x \log p_\theta(x)) + \|\nabla_x \log p_\theta(x)\|^2\right]$$

$$= -2\,\mathbb{E}_{p(x)}\left[(\nabla_x \log p(x))^\top(\nabla_x \log p_\theta(x))\right] + \mathbb{E}_{p(x)}\left[\|\nabla_x \log p_\theta(x)\|^2\right] + C$$

Need to focus on how to handle this term!

We can handle this term! (via sampling from $p(x)$)

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Inside the expectation of this first term (consider the i-th component of vector):

$$\mathbb{E}_{p(x)}\left[\nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i\right]$$

Start with this intractable term.

Score Matching

Inside the expectation of this first term (consider the i-th component of vector):

$$\mathbb{E}_{p(x)}\left[\nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i\right]$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i \; p(x) dx$$

Write as an integral

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Inside the expectation of this first term (consider the i-th component of vector):

$$\mathbb{E}_{p(x)}\left[\nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i\right]$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i \; p(x) dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \frac{\nabla_x p(x)_i}{p(x)} p(x) dx$$

Apply chain rule for gradient of a log

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Inside the expectation of this first term (consider the i-th component of vector):

$$\mathbb{E}_{p(x)}\left[\nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i\right]$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i \; p(x) dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \frac{\nabla_x p(x)_i}{\cancel{p(x)}} \cancel{p(x)} dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x p(x)_i dx$$

Cancel out $p(x)$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Inside the expectation of this first term (consider the i-th component of vector):

$$\mathbb{E}_{p(x)}\left[\nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i\right]$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i \; p(x) dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \frac{\nabla_x p(x)_i}{\cancel{p(x)}} \cancel{p(x)} dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x p(x)_i dx$$

Doesn't look tractable or like an expectation... What to do here?

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Inside the expectation of this first term (consider the i-th component of vector):

$$\mathbb{E}_{p(x)}\left[\nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i\right]$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i \; p(x) dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \frac{\nabla_x p(x)_i}{\cancel{p(x)}} \cancel{p(x)} dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x p(x)_i dx$$

Integration by parts ✨

Score Matching

Inside the expectation of this first term (consider the i-th component of vector):

$$\mathbb{E}_{p(x)}\left[\nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i\right]$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i \; p(x) dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \frac{\nabla_x p(x)_i}{\cancel{p(x)}} \cancel{p(x)} dx$$

Integration by parts ✨

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x p(x)_i dx = -\int_{-\infty}^{\infty} (\nabla_x^2 \log p_\theta(x))_{i,i} \; p(x) dx$$

(and some regularity assumptions)

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Inside the expectation of this first term (consider the i-th component of vector):

$$\mathbb{E}_{p(x)}\left[\nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i\right]$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x \log p(x)_i \; p(x) dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \frac{\nabla_x p(x)_i}{\cancel{p(x)}} \cancel{p(x)} dx$$

$$= \int_{-\infty}^{\infty} \nabla_x \log p_\theta(x)_i \nabla_x p(x)_i dx = -\int_{-\infty}^{\infty} (\nabla_x^2 \log p_\theta(x))_{i,i} \; p(x) dx$$

Integration by parts ✨

i-th diagonal entry of the Hessian matrix

(and some regularity assumptions)

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Back to the Fisher divergence objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)} \left[\| \nabla_x \log p(x) - s_\theta(x) \|_2^2 \right]$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Back to the Fisher divergence objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

From initial derivation

$$= -2\mathbb{E}_{p(x)}\left[(\nabla_x \log p(x))^\top (\nabla_x \log p_\theta(x))\right] + \mathbb{E}_{p(x)}\left[\|\nabla_x \log p_\theta(x)\|^2\right]$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Back to the Fisher divergence objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

From initial derivation

$$= -2\mathbb{E}_{p(x)}\left[(\nabla_x \log p(x))^\top (\nabla_x \log p_\theta(x))\right] + \mathbb{E}_{p(x)}\left[\|\nabla_x \log p_\theta(x)\|^2\right]$$

$$= 2\,\mathbb{E}_{p(x)}\left[\mathrm{tr}(\nabla_x^2 \log p_\theta(x))\right] + \mathbb{E}_{p(x)}\left[\|\nabla_x \log p_\theta(x)\|_2^2\right]$$

Sub in first term

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Back to the Fisher divergence objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

From initial derivation

$$= -2\mathbb{E}_{p(x)}\left[(\nabla_x \log p(x))^\top (\nabla_x \log p_\theta(x))\right] + \mathbb{E}_{p(x)}\left[\|\nabla_x \log p_\theta(x)\|^2\right]$$

$$= 2\,\mathbb{E}_{p(x)}\left[\mathrm{tr}(\nabla_x^2 \log p_\theta(x))\right] + \mathbb{E}_{p(x)}\left[\|\nabla_x \log p_\theta(x)\|_2^2\right]$$

Sub in first term

$$\propto \mathbb{E}_{p(x)}\left[\mathrm{tr}\left(\nabla_x^2 \log p_\theta(x)\right) + \frac{1}{2}\|\nabla_x \log p_\theta(x)\|_2^2\right]$$

Can be rewritten like this

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Fisher divergence — Monte Carlo estimate:

(Last line on previous slide)

$$\mathcal{L}(\theta) \propto \mathbb{E}_{p(x)}\left[\operatorname{tr}\left(\nabla_x^2 \log p_\theta(x)\right) + \frac{1}{2}\|\nabla_x \log p_\theta(x)\|_2^2\right]$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Fisher divergence — Monte Carlo estimate:

$$\mathcal{L}(\theta) \propto \mathbb{E}_{p(x)}\left[\operatorname{tr}\left(\nabla_x^2 \log p_\theta(x)\right) + \frac{1}{2}\|\nabla_x \log p_\theta(x)\|_2^2\right]$$

⇒ Monte Carlo estimate!

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Fisher divergence — Monte Carlo estimate:

$$\mathcal{L}(\theta) \propto \mathbb{E}_{p(x)}\left[\operatorname{tr}\left(\nabla_x^2 \log p_\theta(x)\right) + \frac{1}{2}\|\nabla_x \log p_\theta(x)\|_2^2\right]$$

⇒ Monte Carlo estimate!

$$\approx \frac{1}{M}\sum_{m=1}^{M} \operatorname{tr}\left(\nabla_x^2 \log p_\theta(x^{(m)})\right) + \frac{1}{2}\|\nabla_x \log p_\theta(x^{(m)})\|_2^2,$$

where $x^{(m)} \sim p(x)$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

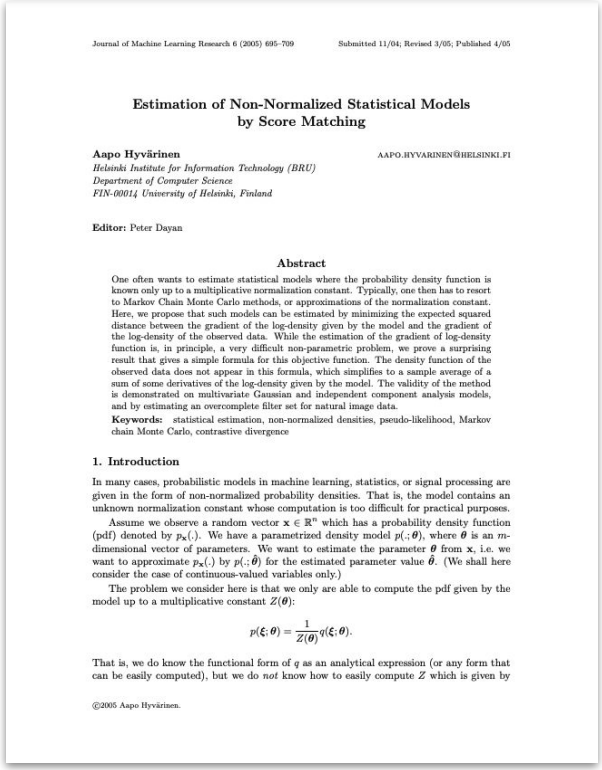
Fisher divergence — Monte Carlo estimate:

$$\mathcal{L}(\theta) \propto \mathbb{E}_{p(x)}\left[\operatorname{tr}\left(\nabla_x^2 \log p_\theta(x)\right) + \frac{1}{2}\|\nabla_x \log p_\theta(x)\|_2^2\right]$$

⇒ Monte Carlo estimate!

$$\approx \frac{1}{M}\sum_{m=1}^{M} \operatorname{tr}\left(\nabla_x^2 \log p_\theta(x^{(m)})\right) + \frac{1}{2}\|\nabla_x \log p_\theta(x^{(m)})\|_2^2,$$

where $x^{(m)} \sim p(x)$



Journal of Machine Learning Research 6 (2005) 695–709 Submitted 11/04; Revised 3/05; Published 4/05

Estimation of Non-Normalized Statistical Models by Score Matching

Aapo Hyvärinen AAPO.HYVARINEN@HELSINKI.FI
Helsinki Institute for Information Technology (BRU)
Department of Computer Science
FIN-00014 University of Helsinki, Finland

Editor: Peter Dayan

Abstract

One often wants to estimate statistical models where the probability density function is known only up to a multiplicative normalization constant. Typically, one then has to resort to Markov Chain Monte Carlo methods, or approximations of the normalization constant. Here, we propose that such models can be estimated by minimizing the expected squared distance between the gradient of the log-density given by the model and the gradient of the log-density of the observed data. While the estimation of the gradient of log-density function is, in principle, a very difficult non-parametric problem, we prove a surprising result that gives a simple formula for this objective function. The density function of the observed data does not appear in this formula, which simplifies to a sample average of a sum of some derivatives of the log-density given by the model. The validity of the method is demonstrated on multivariate Gaussian and independent component analysis models, and by estimating an overcomplete filter set for natural image data.

Keywords: statistical estimation, non-normalized densities, pseudo-likelihood, Markov chain Monte Carlo, contrastive divergence

1. Introduction

In many cases, probabilistic models in machine learning, statistics, or signal processing are given in the form of non-normalized probability densities. That is, the model contains an unknown normalization constant whose computation is too difficult for practical purposes.

Assume we observe a random vector $\mathbf{x} \in \mathbb{R}^n$ which has a probability density function (pdf) denoted by $p_{\mathbf{x}}(.)$. We have a parametrized density model $p(.;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is an m-dimensional vector of parameters. We want to estimate the parameter $\boldsymbol{\theta}$ from $\mathbf{x}$, i.e. we want to approximate $p_{\mathbf{x}}(.)$ by $p(.;\hat{\boldsymbol{\theta}})$ for the estimated parameter value $\hat{\boldsymbol{\theta}}$. (We shall here consider the case of continuous-valued variables only.)

The problem we consider here is that we only are able to compute the pdf given by the model up to a multiplicative constant $Z(\boldsymbol{\theta})$:

$$p(\boldsymbol{\xi};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} q(\boldsymbol{\xi};\boldsymbol{\theta}).$$

That is, we do know the functional form of q as an analytical expression (or any form that can be easily computed), but we do *not* know how to easily compute Z which is given by

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score Matching

Important: scalability of method, and how to actually optimize this loss to learn score network $s_\theta(x)$:

- Hard to scale up equation below to high dimensions.
- More scalable methods: *sliced score matching*, and *denoising score matching*.

Fisher divergence — Monte Carlo estimate:

$$\mathcal{L}(\theta) \propto \mathbb{E}_{p(x)}\left[\operatorname{tr}\left(\nabla_x^2 \log p_\theta(x)\right) + \frac{1}{2}\|\nabla_x \log p_\theta(x)\|_2^2\right]$$

⇒ Monte Carlo estimate!

$$\approx \frac{1}{M}\sum_{m=1}^{M} \operatorname{tr}\left(\nabla_x^2 \log p_\theta(x^{(m)})\right) + \frac{1}{2}\|\nabla_x \log p_\theta(x^{(m)})\|_2^2,$$

where $x^{(m)} \sim p(x)$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Summary

To summarize the (naive/vanilla) procedure:

Summary

To summarize the (naive/vanilla) procedure:

(1) Use Monte Carlo estimate from previous slide to learn a score network $s_\theta(x)$.

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

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Summary

To summarize the (naive/vanilla) procedure:

(1) Use Monte Carlo estimate from previous slide to learn a score network $s_\theta(x)$.

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

(2) Generate samples via Langevin Monte Carlo

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Summary

To summarize the (naive/vanilla) procedure:

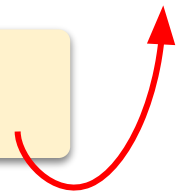
(1) Use Monte Carlo estimate from previous slide to learn a score network $s_\theta(x)$.

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

(2) Generate samples via Langevin Monte Carlo

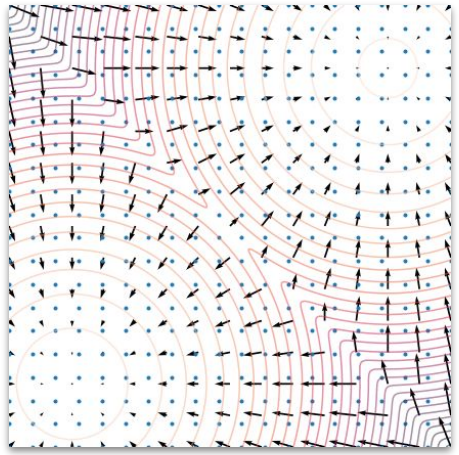
$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x s_\theta(x) + \sqrt{2\epsilon}\,\mathcal{N}(0, I), \quad i = 0, 1, \cdots, K,$$

Note the learned score function is swapped in.



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Summary



To summarize the (naive/vanilla) procedure:

(1) Use Monte Carlo estimate from previous slide to learn a score network $s_\theta(x)$.

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

(2) Generate samples via Langevin Monte Carlo

$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x s_\theta(x) + \sqrt{2\epsilon}\,\mathcal{N}(0, I), \quad i = 0, 1, \cdots, K,$$

Note the learned score function is swapped in.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Unfortunately, this naive/vanilla procedure has some pitfalls in practice!

⇒ Running this without modification does not work great.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Unfortunately, this naive/vanilla procedure has some pitfalls in practice!

⇒ Running this without modification does not work great.

Main issue: estimate score function is inaccurate in low-density regions of $p(x)$.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Unfortunately, this naive/vanilla procedure has some pitfalls in practice!

⇒ Running this without modification does not work great.

Main issue: estimate score function is inaccurate in low-density regions of $p(x)$.

Intuition — to learn the score function, we minimized the Fisher divergence:

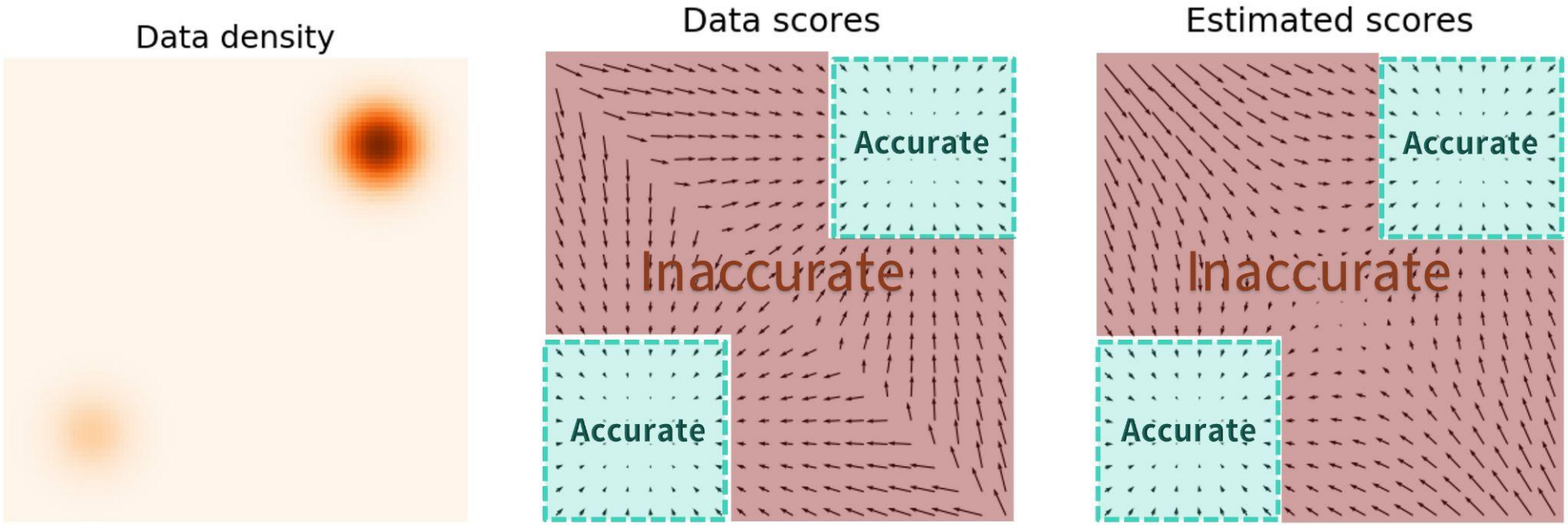
$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right] = \int p(x)\|\nabla_x \log p(x) - s_\theta(x)\|_2^2 dx$$

Note the loss is only large when $p(x)$ has a high value!

⇒ When taking Monte Carlo samples from $p(x)$, we only get samples in high-mass areas

Source: Yan

Difficulties to Solve in Practice



$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right] = \int p(x)\|\nabla_x \log p(x) - s_\theta(x)\|_2^2 dx$$

Note the loss is only large when $p(x)$ has a high value!

⇒ When taking Monte Carlo samples from $p(x)$, we only get samples in high-mass areas

Source: Yan

Difficulties to Solve in Practice

Consequences of this:

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Consequences of this:

- Our initial samples will likely be in low-density regions (especially when our data is high dimensional, *e.g.*, images.)

Difficulties to Solve in Practice

Consequences of this:

- Our initial samples will likely be in low-density regions (especially when our data is high dimensional, *e.g.*, images.)
- ⇒ Langevin Monte Carlo becomes “derailed” at the beginning of the procedure.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Consequences of this:

- Our initial samples will likely be in low-density regions (especially when our data is high dimensional, *e.g.*, images.)
- ⇒ Langevin Monte Carlo becomes “derailed” at the beginning of the procedure.
- ⇒ Unable to generate high quality samples.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

However…

- Suppose that we perturb the data points with noise, and train score-based models on the noisy data points instead.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

However…

- Suppose that we perturb the data points with noise, and train score-based models on the noisy data points instead.

- When the noise is large enough, it can populate low-density regions of $p(x)$.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

However…

- Suppose that we perturb the data points with noise, and train score-based models on the noisy data points instead.
- When the noise is large enough, it can populate low-density regions of $p(x)$.
- ⇒ This improves the accuracy of estimated scores (for a slightly different/biased distribution).

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

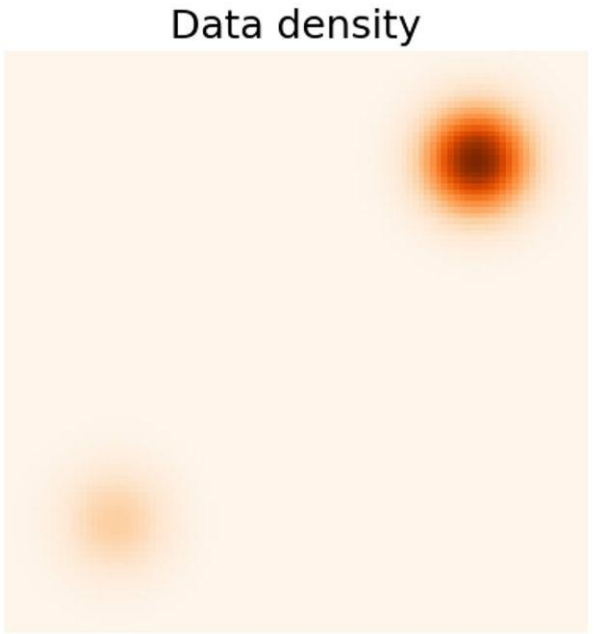
Difficulties to Solve in Practice

Visually, using previous figure:

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Visually, using previous figure:



Before: **No perturbation.**

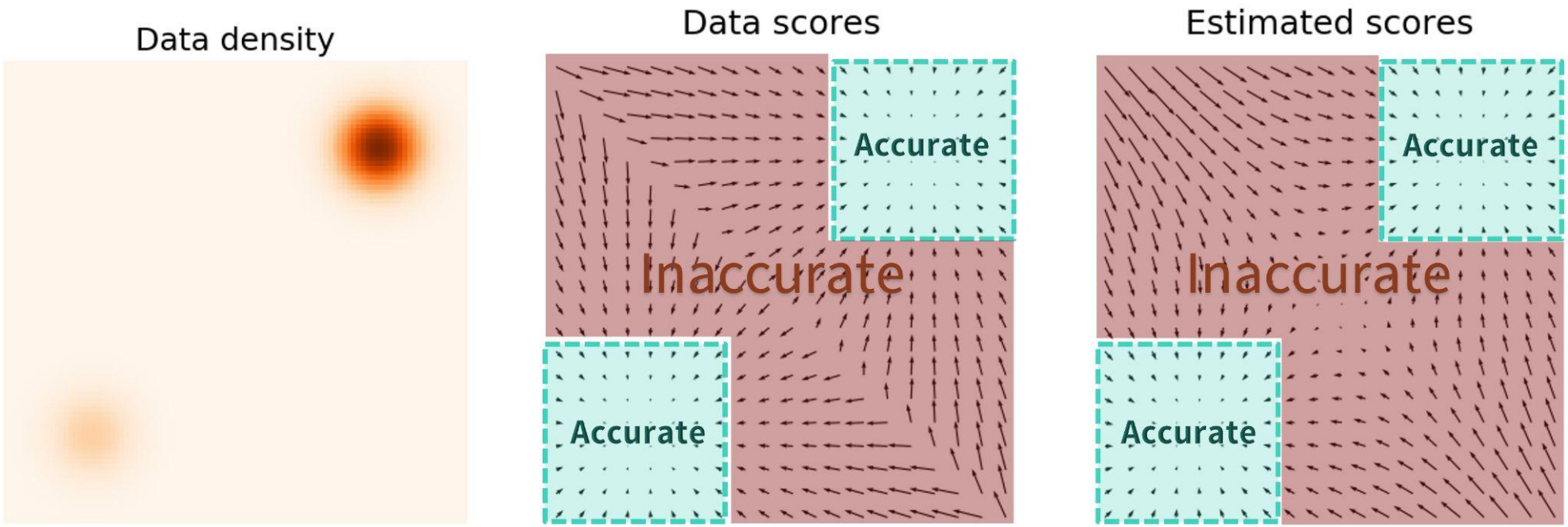
⇒ Estimated score is inaccurate in low-density regions.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Visually, using previous figure:

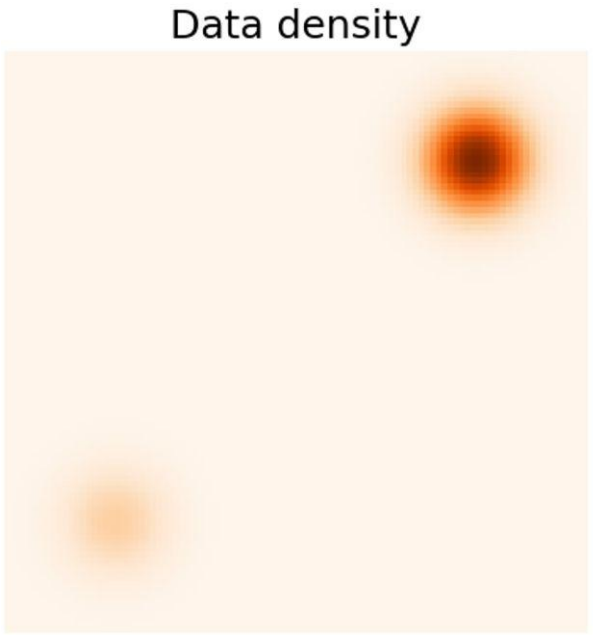
Before: **No perturbation.**
⇒ Estimated score is inaccurate in low-density regions.



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Visually, using previous figure:

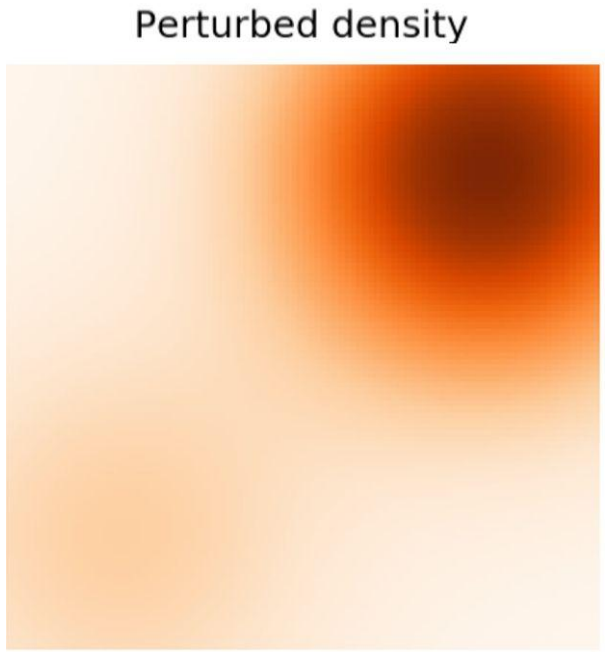


Instead: **Perturb (add noise to)** $p(x)$**.**
⇒ Produces more samples in low-density regions, thus making score more accurate.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Visually, using previous figure:



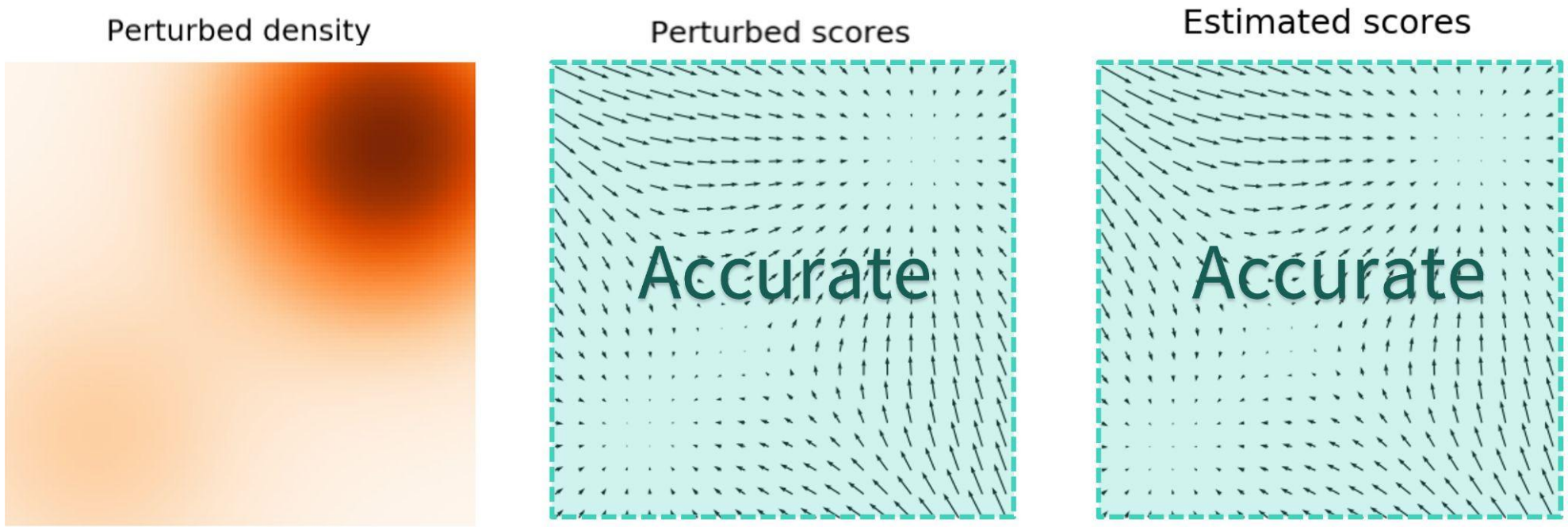
Instead: **Perturb (add noise to)** $p(x)$**.**
⇒ Produces more samples in low-density regions, thus making score more accurate.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Difficulties to Solve in Practice

Visually, using previous figure:

Instead: **Perturb (add noise to)** $p(x)$**.**
⇒ Produces more samples in low-density regions, thus making score more accurate.



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

So we have a tradeoff:

Score-based Generative Models — Multiple Noise Perturbations

So we have a tradeoff:

Decreasing the noise makes score matching **harder** (especially at start of LMC), but yields more-accurate (*unbiased*) scores.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

So we have a tradeoff:

Decreasing the noise makes score matching **harder** (especially at start of LMC), but yields more-accurate (*unbiased*) scores.

Increasing the noise makes score matching **easier** (samples cover low-density regions), but yields less-accurate (*biased*) scores.

Score-based Generative Models — Multiple Noise Perturbations

So we have a tradeoff:

Decreasing the noise makes score matching **harder** (especially at start of LMC), but yields more-accurate (*unbiased*) scores.

Increasing the noise makes score matching **easier** (samples cover low-density regions), but yields less-accurate (*biased*) scores.

Solution:

Perform score matching (learn the score function) for *multiple noise levels simultaneously*!

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

So we have a tradeoff:

Decreasing the noise makes score matching **harder** (especially at start of LMC), but yields more-accurate (*unbiased*) scores.

Increasing the noise makes score matching **easier** (samples cover low-density regions), but yields less-accurate (*biased*) scores.

Solution:

Perform score matching (learn the score function) for *multiple noise levels simultaneously*!

And then, to sample in LMC, we will traverse through the noise levels (from high to low noise).

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

In particular:

Score-based Generative Models — Multiple Noise Perturbations

In particular:

- Suppose we perturb the data with isotropic Gaussian noise.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

In particular:

- Suppose we perturb the data with isotropic Gaussian noise.
- And that we have L increasing noise levels: $i = 1, \ldots, L$.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

In particular:

- Suppose we perturb the data with isotropic Gaussian noise.
- And that we have L increasing noise levels: $i = 1, \ldots, L$.
- For each noise level i, we can add Gaussian noise $\mathcal{N}(0, \sigma_i^2 I)$ to $p(x)$ to form the *noise-perturbed distribution*:

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

In particular:

- Suppose we perturb the data with isotropic Gaussian noise.
- And that we have L increasing noise levels: $i = 1, \ldots, L$.
- For each noise level i, we can add Gaussian noise $\mathcal{N}(0, \sigma_i^2 I)$ to $p(x)$ to form the *noise-perturbed distribution*:

$$p_{\sigma_i}(x) = \int p(x')\mathcal{N}(x \mid x', \sigma_i^2 I)dx' = \mathbb{E}_{p(x')}\left[\mathcal{N}(x \mid x', \sigma_i^2 I)\right]$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

Next, we can estimate the score function for each noise-perturbed distribution *i*, which we can denote $s_\theta(x, i)$, using score matching.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

Next, we can estimate the score function for each noise-perturbed distribution *i*, which we can denote $s_\theta(x, i)$, using score matching.

At the end of this procedure, our score network should approximate:

$$s_\theta(x, i) \approx \nabla_x \log p_{\sigma_i}(x), \quad i = 1, 2, \ldots, L$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

Next, we can estimate the score function for each noise-perturbed distribution *i*, which we can denote $s_\theta(x, i)$, using score matching.

At the end of this procedure, our score network should approximate:

$$s_\theta(x, i) \approx \nabla_x \log p_{\sigma_i}(x), \quad i = 1, 2, \ldots, L$$

We will call this model the **noise conditional score network (NCSN)**.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

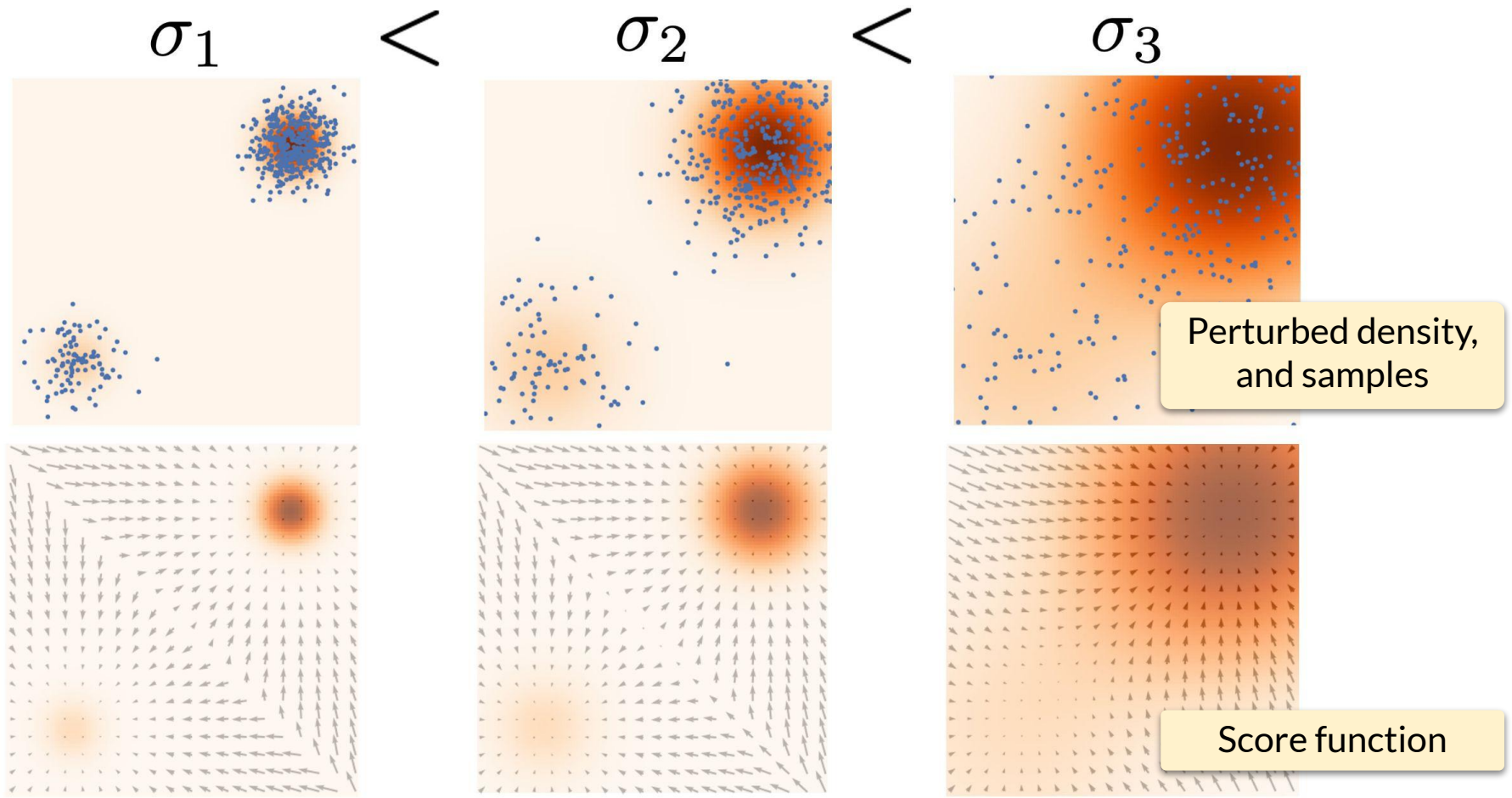
Score-based Generative Models — Multiple Noise Perturbations

Visualizing this:

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

Visualizing this:



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Multiple Noise Perturbations

If we perturb a data with multiple scales of Gaussian noise, we see an increasingly noisy data point.

An image example:

Note: it is easy for us to sample this perturbed data (given our original dataset).

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Learning the NCSN

To learn the noise-conditional score network (NCSN) $s_\theta(x, i)$:

Score-based Generative Models — Learning the NCSN

To learn the noise-conditional score network (NCSN) $s_\theta(x, i)$:

We use a modified training objective — a weighted sum of Fisher divergences over all of the noise scales:

$$\mathcal{L}(\theta) = \sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(x)} \left[\| \nabla_x \log p_{\sigma_i}(x) - s_\theta(x, i) \|_2^2 \right]$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Learning the NCSN

To learn the noise-conditional score network (NCSN) $s_\theta(x, i)$:

We use a modified training objective — a weighted sum of Fisher divergences over all of the noise scales:

$$\mathcal{L}(\theta) = \sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(x)} \left[\| \nabla_x \log p_{\sigma_i}(x) - s_\theta(x, i) \|_2^2 \right]$$

Where the weighting function is often taken to be $\lambda(i) = \sigma_i^2$.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Generation from NCSN

Finally, to sample from this model:

Score-based Generative Models — Generation from NCSN

Finally, to sample from this model:

Similar to before, run Langevin Monte Carlo (LMC) using the learned (noise-conditional) score network $s_\theta(x, i)$...

While sequentially stepping (backwards) through the noise levels: $i = L, L\text{-}1, L\text{-}2, \ldots, 1$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Generation from NCSN

Finally, to sample from this model:

Similar to before, run Langevin Monte Carlo (LMC) using the learned (noise-conditional) score network $s_\theta(x, i)$...

While sequentially stepping (backwards) through the noise levels: $i = L, L\text{-}1, L\text{-}2, \ldots, 1$

⇒ This is LMC with an annealed noise level/step size! *(Mentioned in MCMC lecture).*

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Generation from NCSN

Altogether, how does this look?

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Generation from NCSN

Altogether, how does this look?

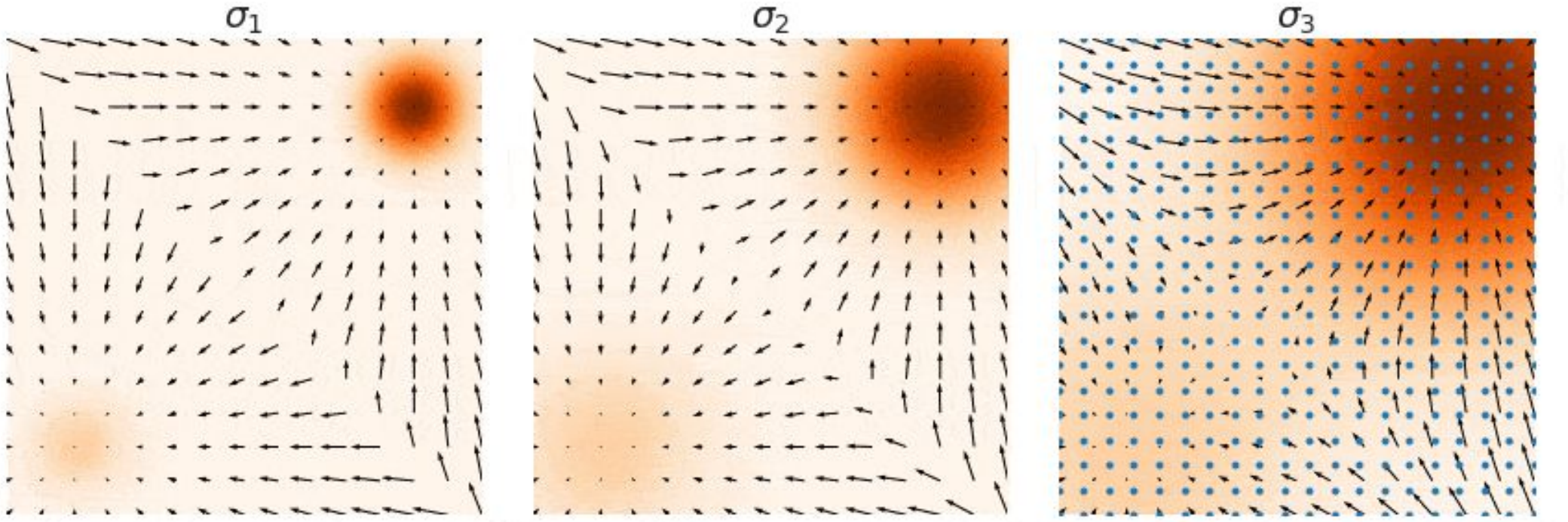
← Proceeding from **right** to **left**.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Generation from NCSN

Altogether, how does this look?

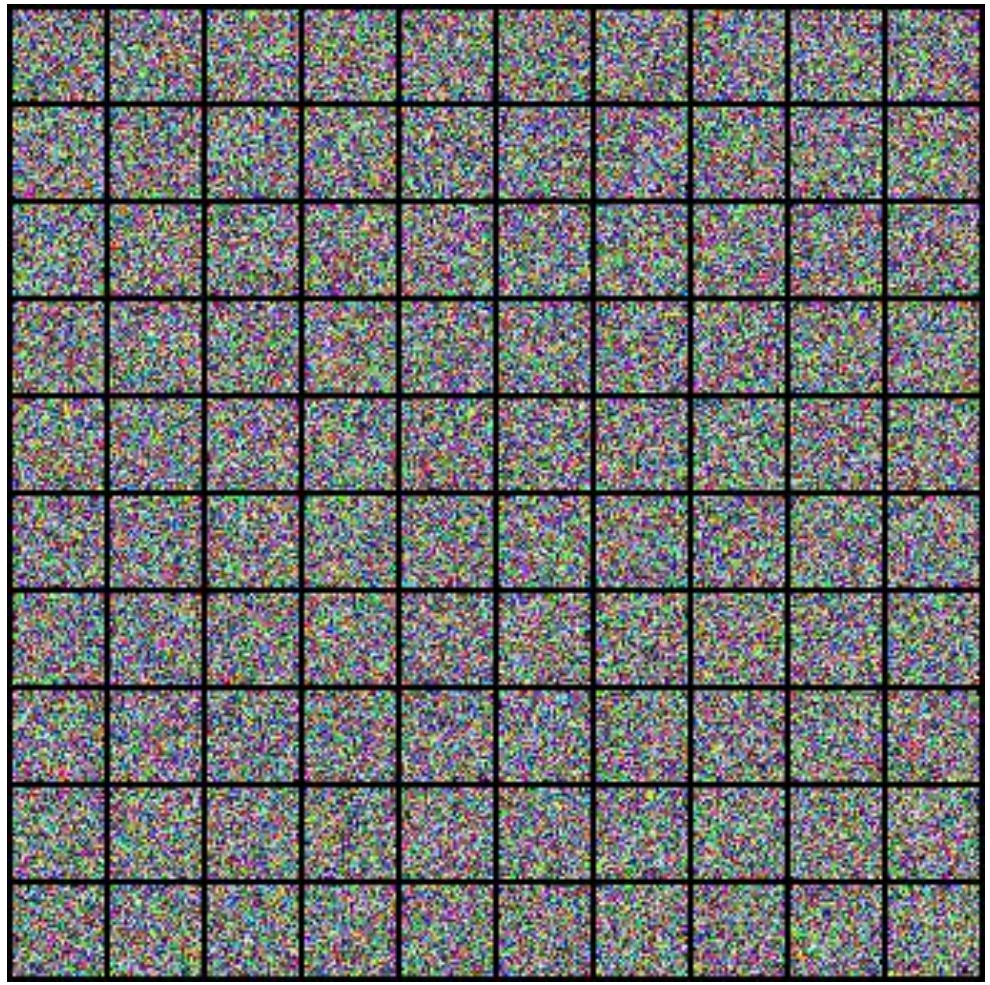
← Proceeding from **right** to **left**.



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Generation from NCSN

And on higher dimensional data (face images):



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Connections to Diffusion Models

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Connections to Diffusion Models

You can see how score-based models start to look like what are typically called **diffusion models**.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Connections to Diffusion Models

You can see how score-based models start to look like what are typically called **diffusion models**.

Diffusion models can be viewed in the following way:

Connections to Diffusion Models

You can see how score-based models start to look like what are typically called **diffusion models**.

Diffusion models can be viewed in the following way:

- Iteratively add noise to data points (via a *forward diffusion process*).

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Connections to Diffusion Models

You can see how score-based models start to look like what are typically called **diffusion models**.

Diffusion models can be viewed in the following way:

- Iteratively add noise to data points (via a *forward diffusion process*).
- Then learn a *reverse diffusion process*, which iteratively denoises data.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Connections to Diffusion Models

You can see how score-based models start to look like what are typically called **diffusion models**.

Diffusion models can be viewed in the following way:

- Iteratively add noise to data points (via a *forward diffusion process*).
- Then learn a *reverse diffusion process*, which iteratively denoises data.
- By sampling from the reverse diffusion process, can generate new data samples.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Connections to Diffusion Models

A popular example is the **denoising diffusion probabilistic model (DDPM)**, *NeurIPS 2019*.

(We will hear a presentation of this paper later today.)

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

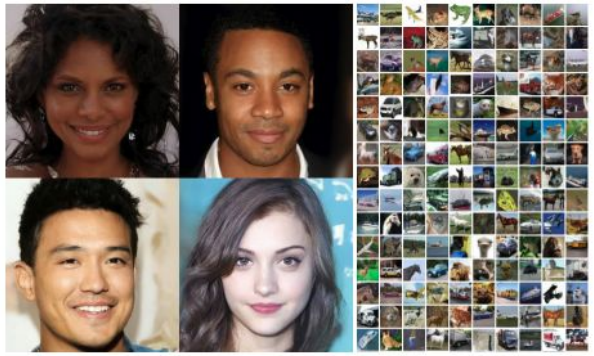


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Connections to Diffusion Models

But for a brief overview:

Connections to Diffusion Models

But for a brief overview:

- Define the DDPM as a probabilistic model.
- Where we view the *forward diffusion process* as a variational approximation to the posterior.

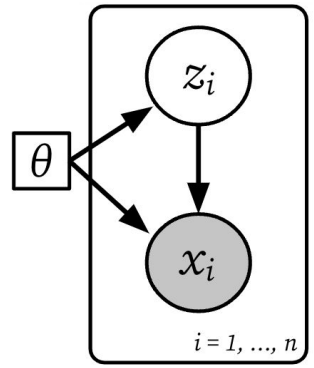
Connections to Diffusion Models

But for a brief overview:

- Define the DDPM as a probabilistic model.
- Where we view the *forward diffusion process* as a variational approximation to the posterior.
- And we view the *reverse diffusion process* as a Markov chain, with generative model parameters θ that we'd like to learn.

Connections to Diffusion Models

Somewhat similar in this way to VAEs

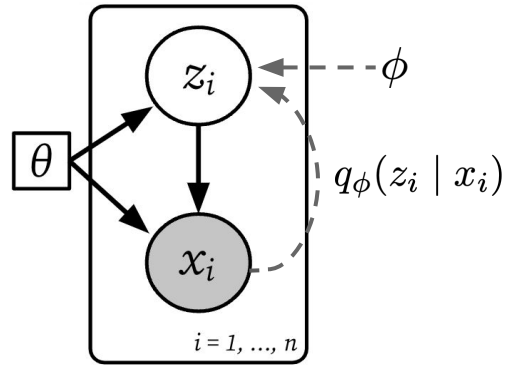


But for a brief overview:

- Define the DDPM as a probabilistic model.
- Where we view the *forward diffusion process* as a variational approximation to the posterior.
- And we view the *reverse diffusion process* as a Markov chain, with generative model parameters θ that we'd like to learn.

Connections to Diffusion Models

Somewhat similar in this way to VAEs

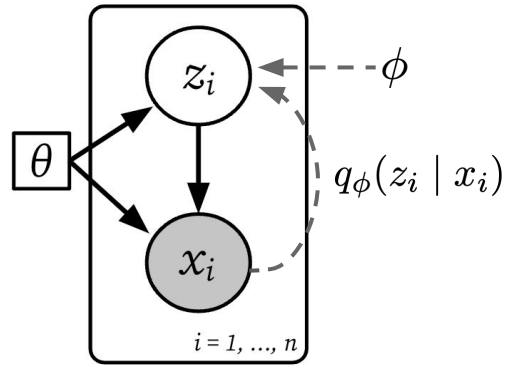


But for a brief overview:

- Define the DDPM as a probabilistic model.
- Where we view the *forward diffusion process* as a variational approximation to the posterior.
- And we view the *reverse diffusion process* as a Markov chain, with generative model parameters θ that we'd like to learn.

Connections to Diffusion Models

Somewhat similar in this way to VAEs



But for a brief overview:

- Define the DDPM as a probabilistic model.
- Where we view the *forward diffusion process* as a variational approximation to the posterior.
- And we view the *reverse diffusion process* as a Markov chain, with generative model parameters θ that we'd like to learn.
- And similar to VAE models, we maximize the ELBO while learning θ.

Connections to Diffusion Models

But for a brief overview:

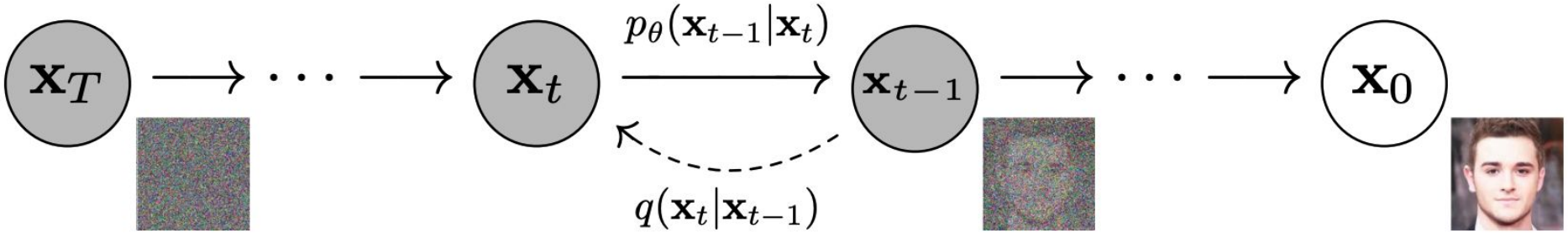
The graphical model for the DDPM is popularly illustrated as:

Source: “Denoising Diffusion Probabilistic Models”, Ho et al., 2019

Connections to Diffusion Models

But for a brief overview:

The graphical model for the DDPM is popularly illustrated as:

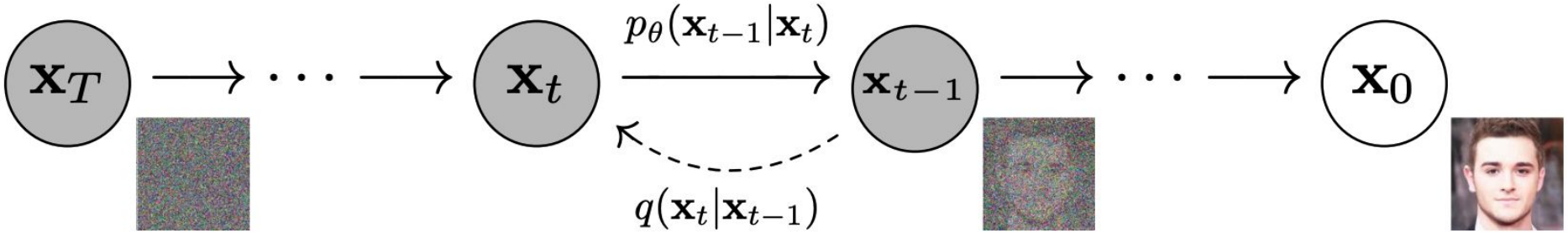


Source: “Denoising Diffusion Probabilistic Models”, Ho et al., 2019

Connections to Diffusion Models

But for a brief overview:

The graphical model for the DDPM is popularly illustrated as:



Notice q is not learned (like in VAEs), but is a fixed process

… except for noise level parameters, in some cases.

Source: "Denoising Diffusion Probabilistic Models", Ho et al., 2019

Connections to Diffusion Models

So, to be explicit, the probabilistic model (Bayesian network) is:

$$p_\theta(x_0, ..., x_T) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

Connections to Diffusion Models

So, to be explicit, the probabilistic model (Bayesian network) is:

$$p_\theta(x_0, \ldots, x_T) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

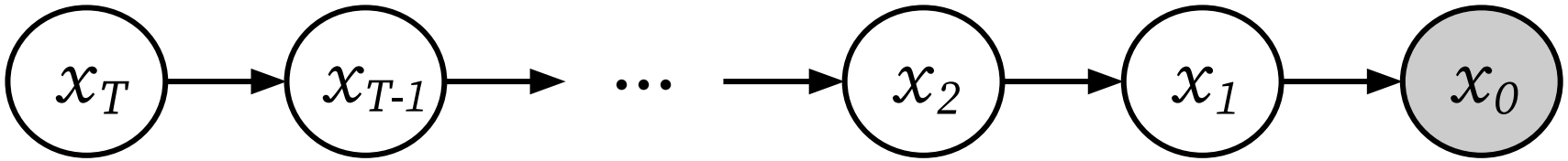
A Markov chain (with one node observed)

Connections to Diffusion Models

So, to be explicit, the probabilistic model (Bayesian network) is:

$$p_\theta(x_0, \ldots, x_T) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

A Markov chain (with one node observed)



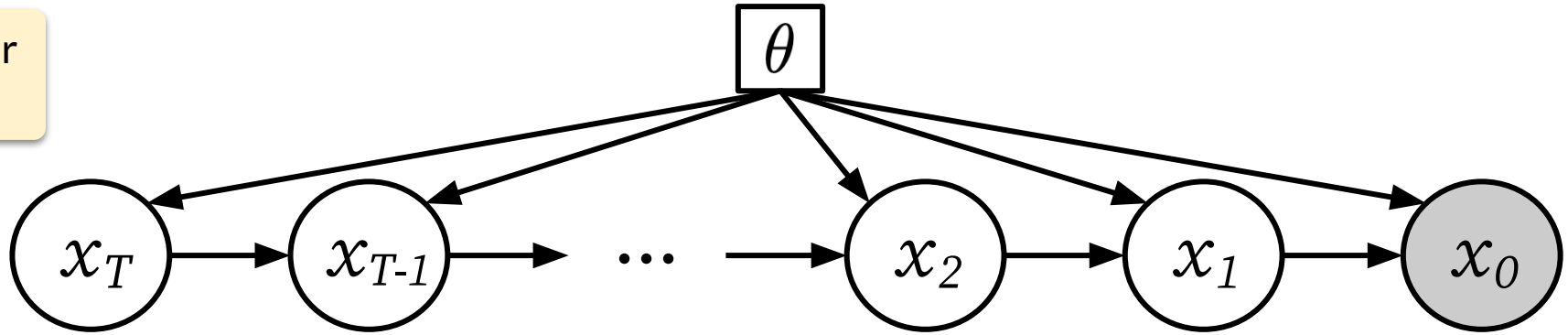
Connections to Diffusion Models

So, to be explicit, the probabilistic model (Bayesian network) is:

$$p_\theta(x_0, \ldots, x_T) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

A Markov chain (with one node observed)

With parameter θ shown



Connections to Diffusion Models

While the variational family of *posterior approximations* can be written:

Connections to Diffusion Models

While the variational family of *posterior approximations* can be written:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

Connections to Diffusion Models

While the variational family of *posterior approximations* can be written:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

(Another Markov chain!)

Connections to Diffusion Models

While the variational family of *posterior approximations* can be written:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

(Another Markov chain!)

The graphical model for this distribution might be drawn as:

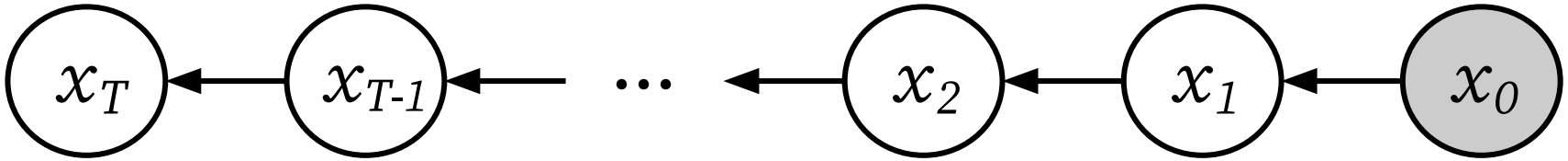
Connections to Diffusion Models

While the variational family of *posterior approximations* can be written:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

(Another Markov chain!)

The graphical model for this distribution might be drawn as:



Connections to Diffusion Models

While the variational family of *posterior approximations* can be written:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

(Another Markov chain!)

And sticking both together we get:

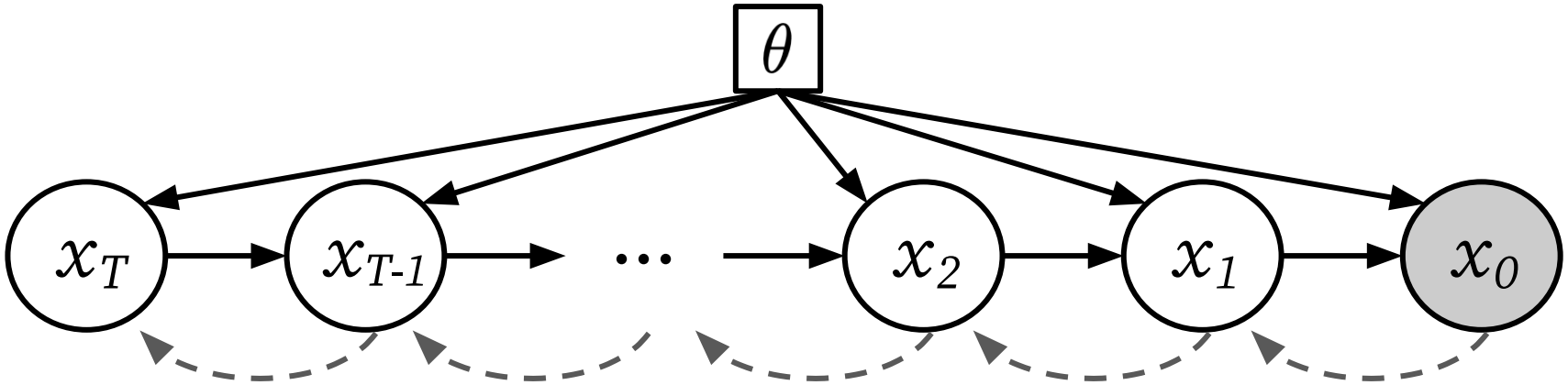
Connections to Diffusion Models

While the variational family of *posterior approximations* can be written:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

(Another Markov chain!)

And sticking both together we get:



Connections to Diffusion Models

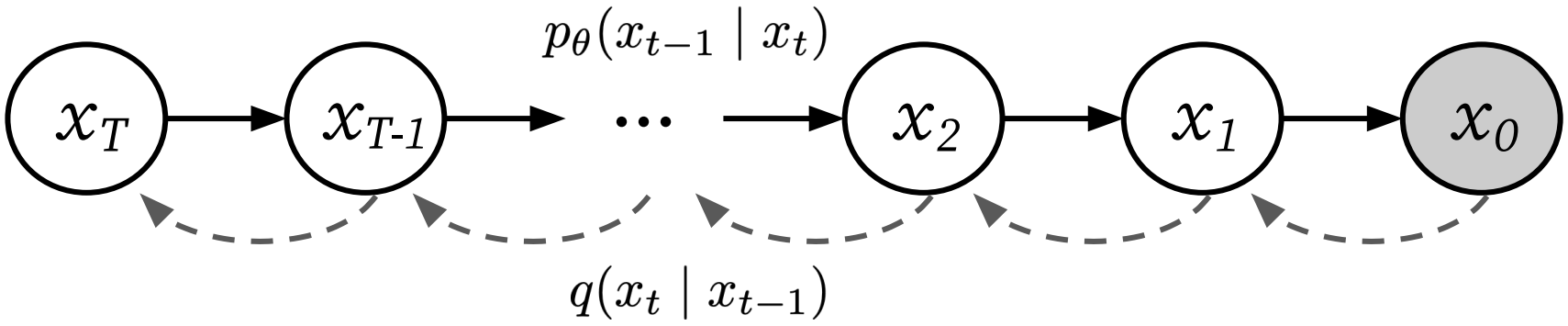
While the variational family of *posterior approximations* can be written:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

(Another Markov chain!)

And sticking both together we get:

Similar to the "usual" PGM diagram



Connections to Diffusion Models

Similar to VAE, you can:

- Write out the ELBO — has a particular form due to the structure of p and q.
- Then try to optimize it with respect to θ (and/or any VI parameters).

Connections to Diffusion Models

Before the DDPM paper, diffusion models as described here (roughly) already existed in prior work.

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

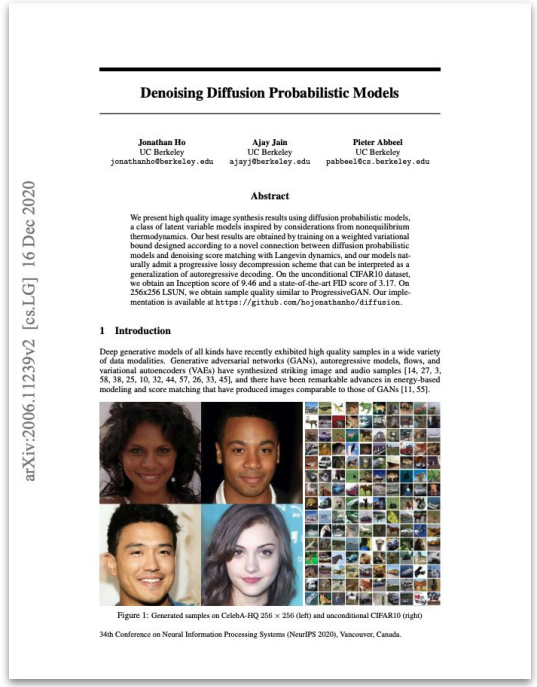
Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Source: “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”, Sohl-Dickstein et al., 2015

Connections to Diffusion Models

Before the DDPM paper, diffusion models as described here (roughly) already existed in prior work.

However, the DDPM paper does a couple of things:



arXiv:2006.11239v2 [cs.LG] 16 Dec 2020

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

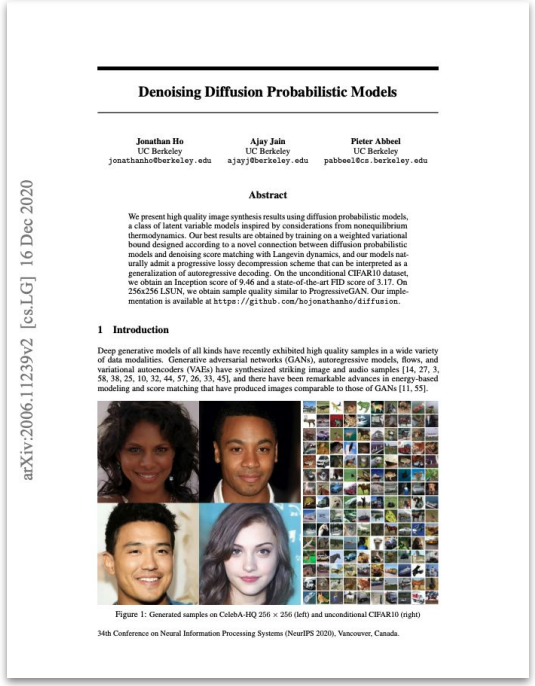
Source: “Denoising Diffusion Probabilistic Models”, Ho et al., 2019

Connections to Diffusion Models

Before the DDPM paper, diffusion models as described here (roughly) already existed in prior work.

However, the DDPM paper does a couple of things:

(1) Shows that under a certain parameterization of the model, their algorithm is *equivalent to* denosing score matching.



Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Source: “Denoising Diffusion Probabilistic Models”, Ho et al., 2019

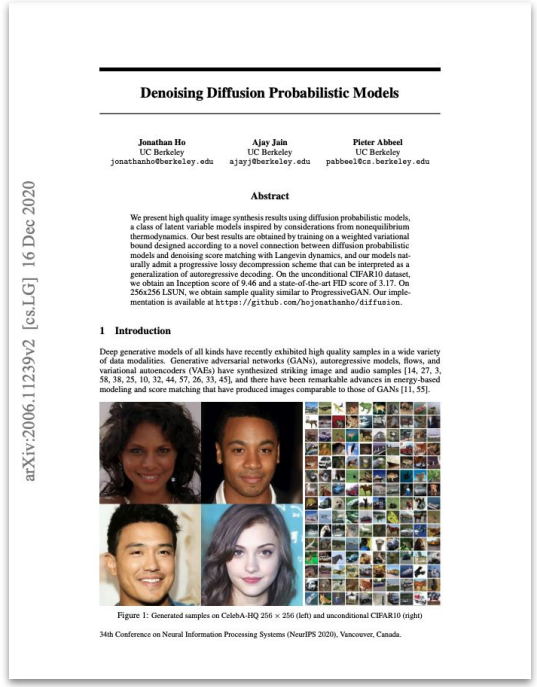
Connections to Diffusion Models

Before the DDPM paper, diffusion models as described here (roughly) already existed in prior work.

However, the DDPM paper does a couple of things:

(1) Shows that under a certain parameterization of the model, their algorithm is *equivalent to* denosing score matching.

- Shows optimizing the ELBO ⇒ equivalent to the "*sum of Fisher divergences*" loss from before!
- And that sampling from the forward model ⇒ equivalent to Langevin dynamics on a learned score function!



arXiv:2006.11239v2 [cs.LG] 16 Dec 2020

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Source: "Denoising Diffusion Probabilistic Models", Ho et al., 2019

Connections to Diffusion Models

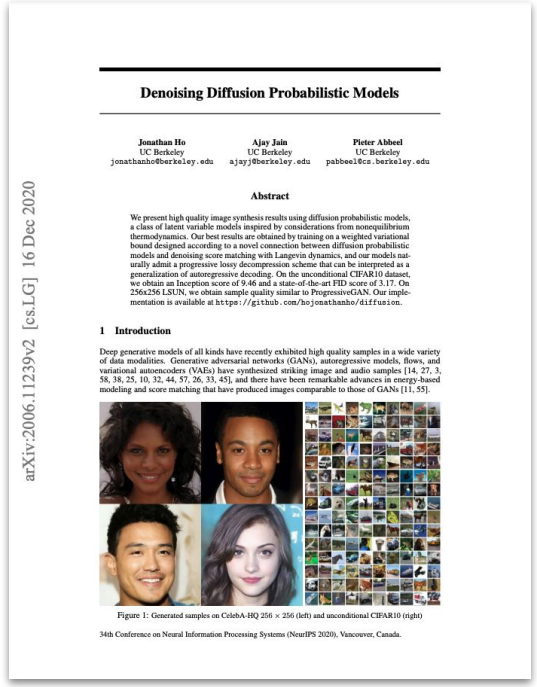
Before the DDPM paper, diffusion models as described here (roughly) already existed in prior work.

However, the DDPM paper does a couple of things:

(1) Shows that under a certain parameterization of the model, their algorithm is *equivalent to* denosing score matching.

(2) Shows that under this parameterization (along with a few additional tricks to better handle image data), sample quality is can be very good.

(Possibly better than most/all previous generative models up until this point).



arXiv:2006.11239v2 [cs.LG] 16 Dec 2020

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Source: “Denoising Diffusion Probabilistic Models”, Ho et al., 2019

Connections to Diffusion Models

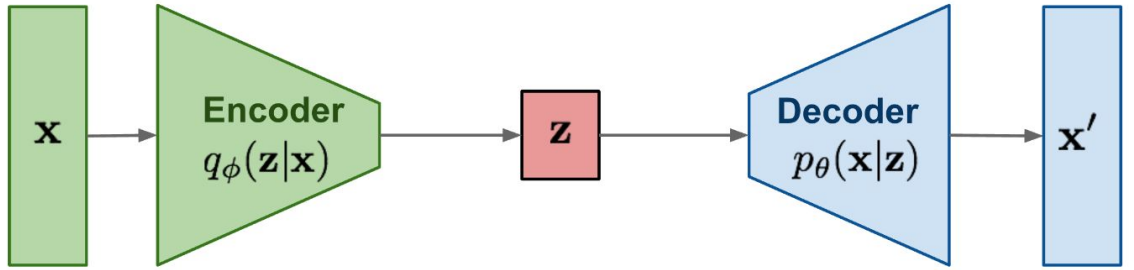
A comparison of generative modeling paradigms (from Lil'Log blog):

Source: "What are Diffusion Models?", Lilian Weng., 2021

Connections to Diffusion Models

A comparison of generative modeling paradigms (from Lil'Log blog):

VAE: maximize variational lower bound

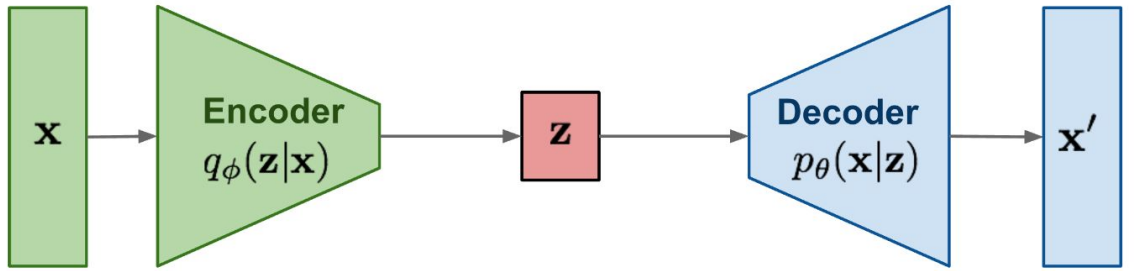


Source: "What are Diffusion Models?", Lilian Weng., 2021

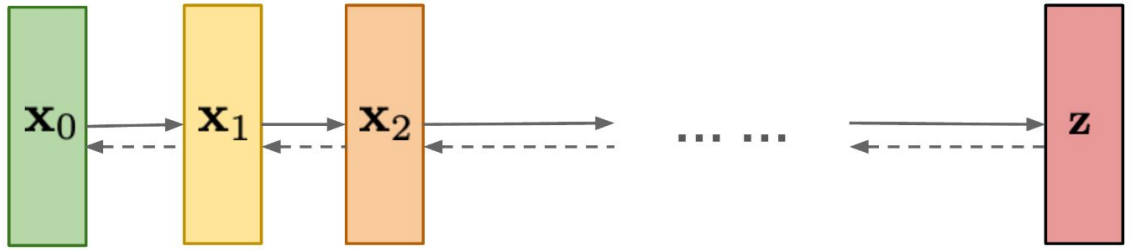
Connections to Diffusion Models

A comparison of generative modeling paradigms (from Lil'Log blog):

VAE: maximize variational lower bound



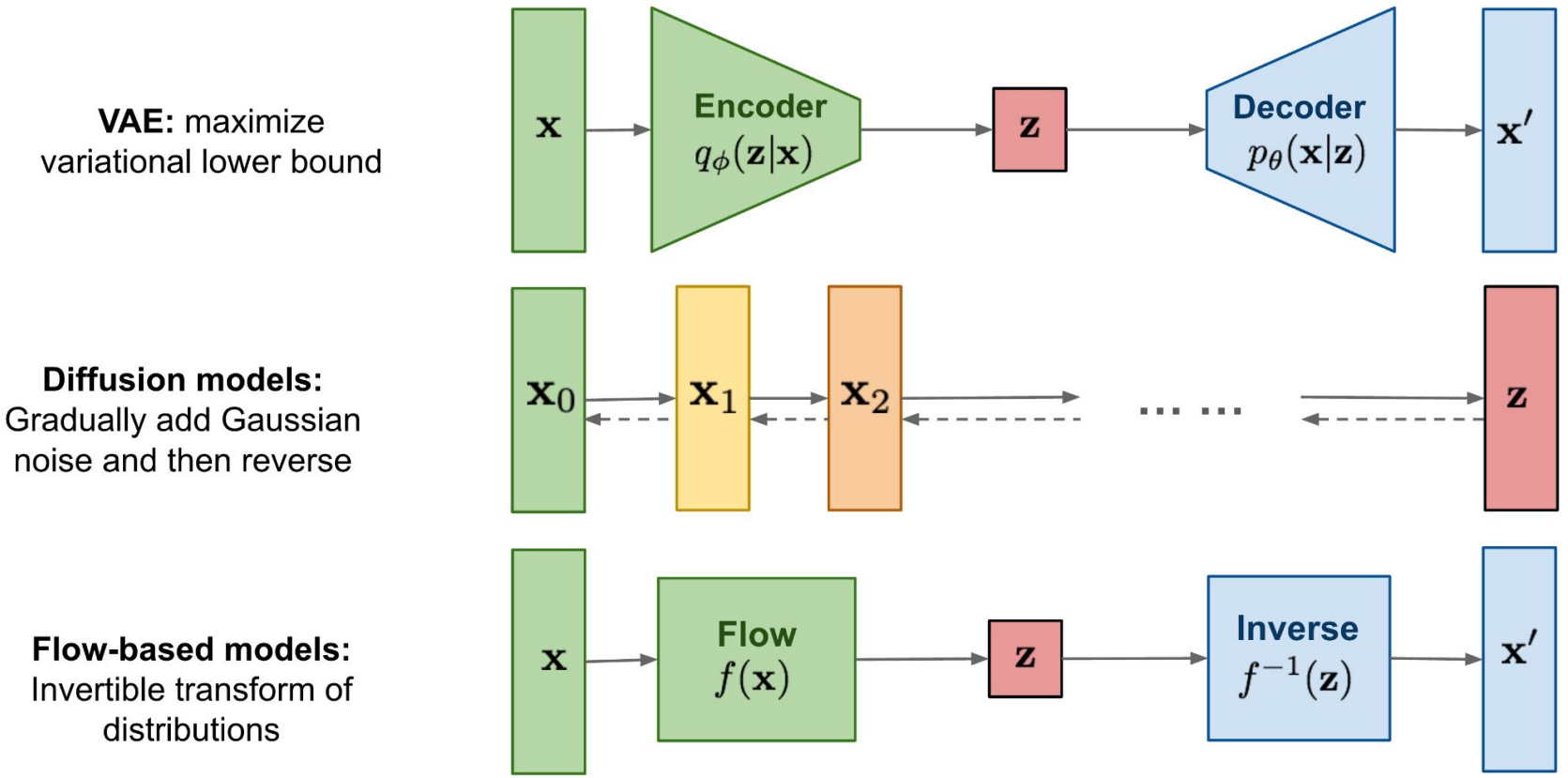
Diffusion models: Gradually add Gaussian noise and then reverse



Source: "What are Diffusion Models?", Lilian Weng., 2021

Connections to Diffusion Models

A comparison of generative modeling paradigms (from Lil'Log blog):



Source: "What are Diffusion Models?", Lilian Weng., 2021