Advanced Course on Deep Generative Models

Lecture 10: Posterior Sampling and Guidance with Diffusion Models

Adapted from Stefano Ermon, Volodymyr Kuleshov and CVPR tutorial

Dan Rosenbaum, CS Haifa

Outline

- Diffusion models – Recap
- Posterior sampling
- Classifier Guidance
- Classifier free Guidance

- Course Summary

Latent variable models with variational training

Sohl-Dickstein, Weiss, Maheswaranathan and Ganguli,
"Deep Unsupervised Learning using Nonequilibrium Thermodynamics", 2015.

Score based models

Song and Ermon,
"Generative Modeling by Estimating Gradients of the Data Distribution", 2019.

Denoising Diffusion Probabilistic Models

Jonathan Ho, Ajay Jain, Pieter Abbeel, 2020

Score-Based Generative Modeling through Stochastic Differential Equations

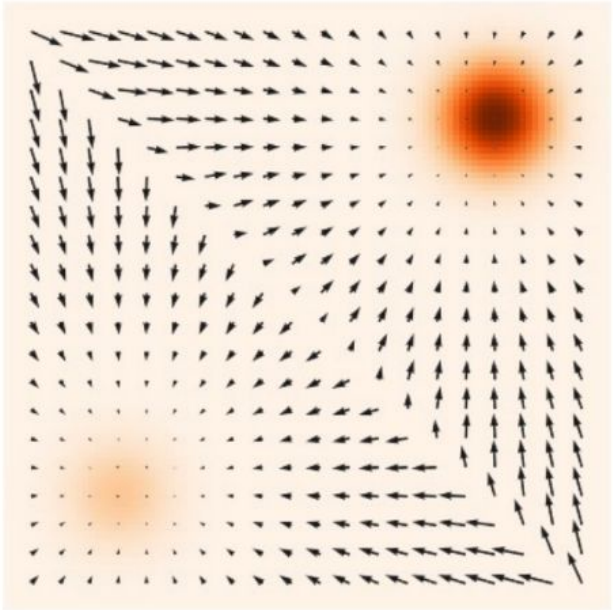
Song, Sohl-Dickstein, Kingma, Kumar, Ermon and Poole 2021.

Score function

Idea: Instead of learning $p_\theta(\mathbf{x})$, learn a model $\nabla_\mathbf{x} \log p_\theta(\mathbf{x})$ of the score function of the data $\nabla_\mathbf{x} \log p_{\text{data}}(\mathbf{x})$!

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp(f_\theta(\mathbf{x})) = \frac{1}{Z(\theta)} \exp(f_\theta(\mathbf{x}))$$

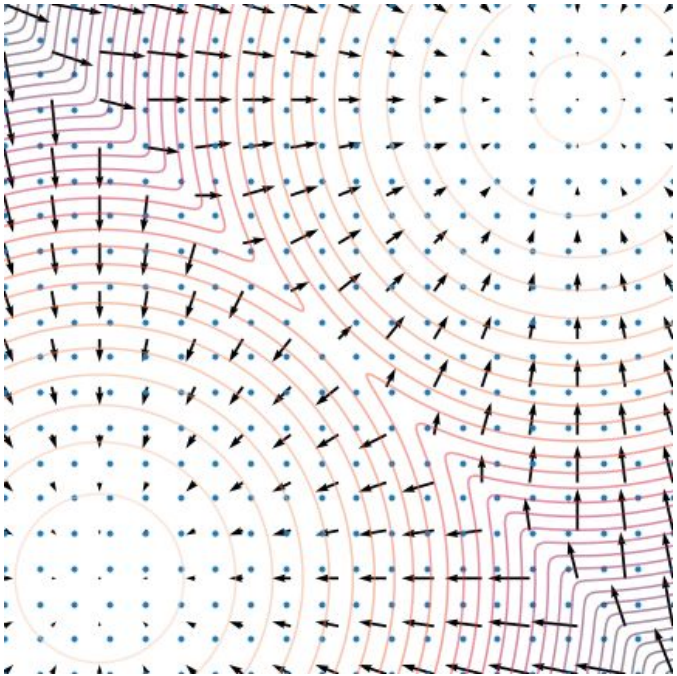
$$\nabla_\mathbf{x} \log p_\theta(\mathbf{x}) = \nabla_\mathbf{x} f_\theta(\mathbf{x}) - \nabla_\mathbf{x} \log Z(\theta) = \nabla_\mathbf{x} f_\theta(\mathbf{x})$$



Sampling with score-based models

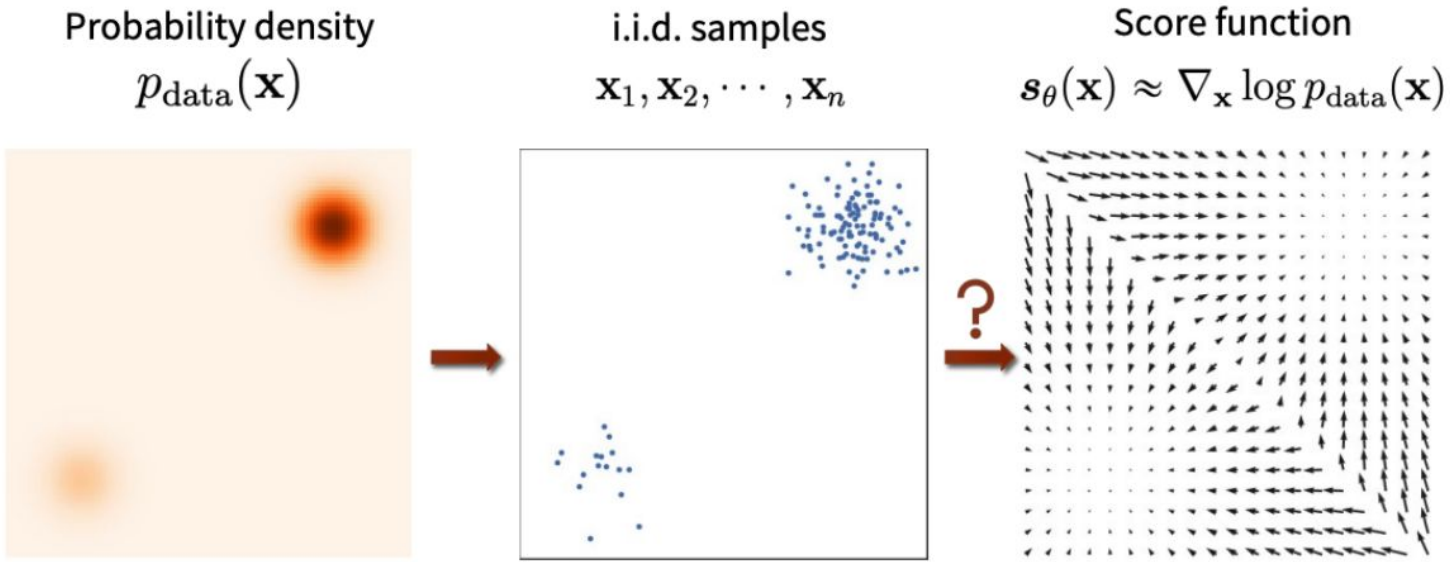
Langevin dynamics

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\, \mathbf{z}_i, \quad i = 0, 1, \cdots, K$$



Score Function Estimation

The idea of score function estimation is to learn a model $s_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$ of the score function from a dataset of sample points.



Score Matching

$$\frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[||\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x}) - s_\theta(\mathbf{x})||_2^2\right]$$

Problem: We don't know the ground truth score.

Solutions:

1. Score Matching: $\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}||s_\theta(\mathbf{x})||_2^2 + \text{tr}\left(\nabla_{\mathbf{x}} s_\theta(\mathbf{x})\right)\right]$

2. Denoising Score Matching: $\mathbb{E}_{x\sim p(x),\epsilon=\mathcal{N}(0,I)}\left[\|\epsilon - s_\theta(x+\sigma\cdot\epsilon)\|_2^2\right]$

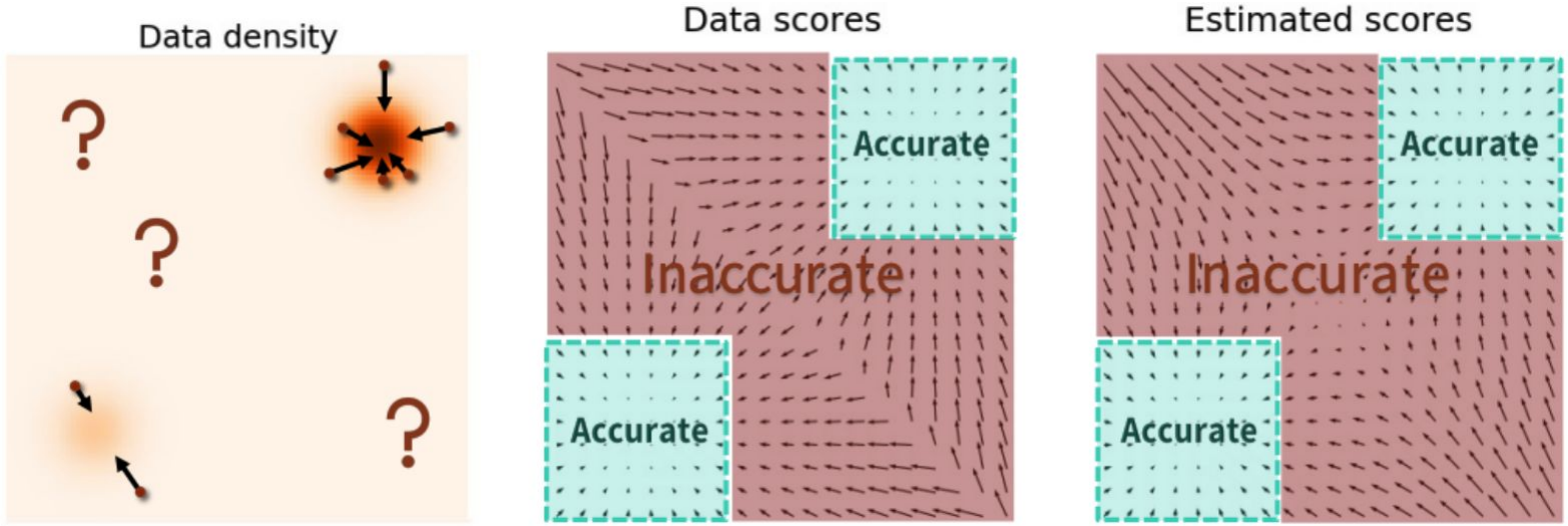
Manifold Hypothesis

- Images lie on a low dimensional manifold in space

- Intuition of denoising score matching:
  - adding noise takes us out of the manifold
  - Going back is the quickest direction to increase probability

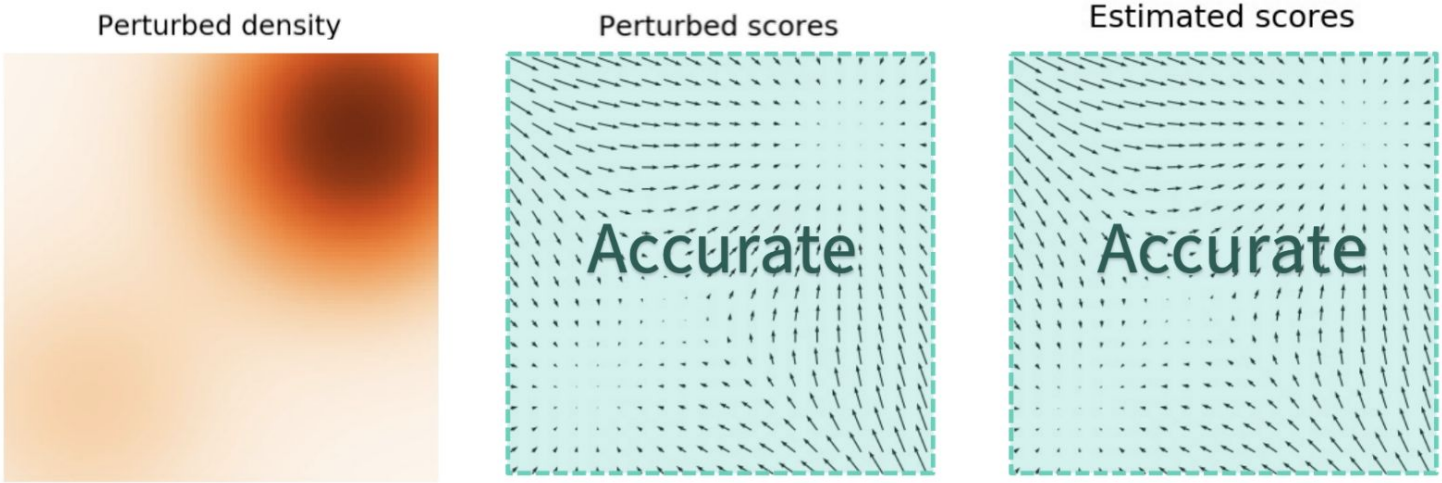
- Problem: how is the score defined outside the manifold?

Learning and Sampling in Low Density Regions



- Not enough samples to learn the score in low density regions
- Score is very small and unstable ⇒ to weak to generate samples

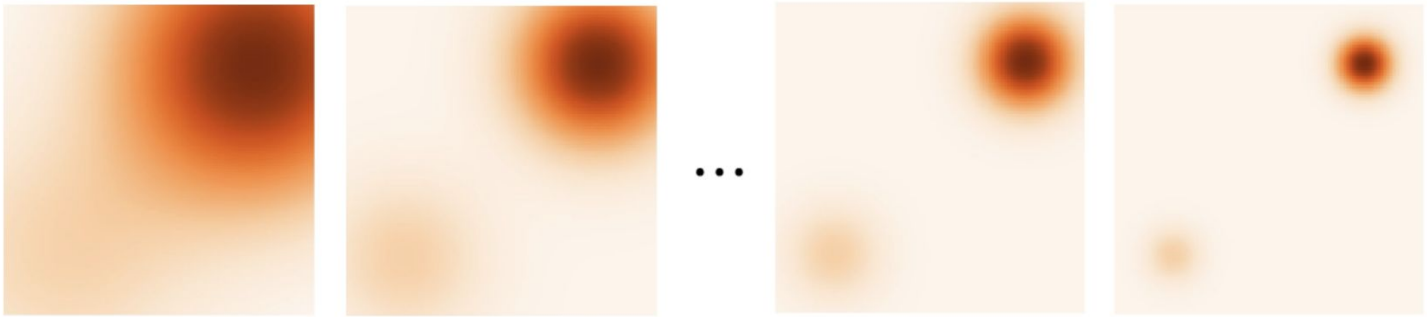
Adding Strong Noise



- Score is well defined everywhere
- Enough samples for learning, smoother distribution for sampling
- The distribution is different than the original - tradeoff

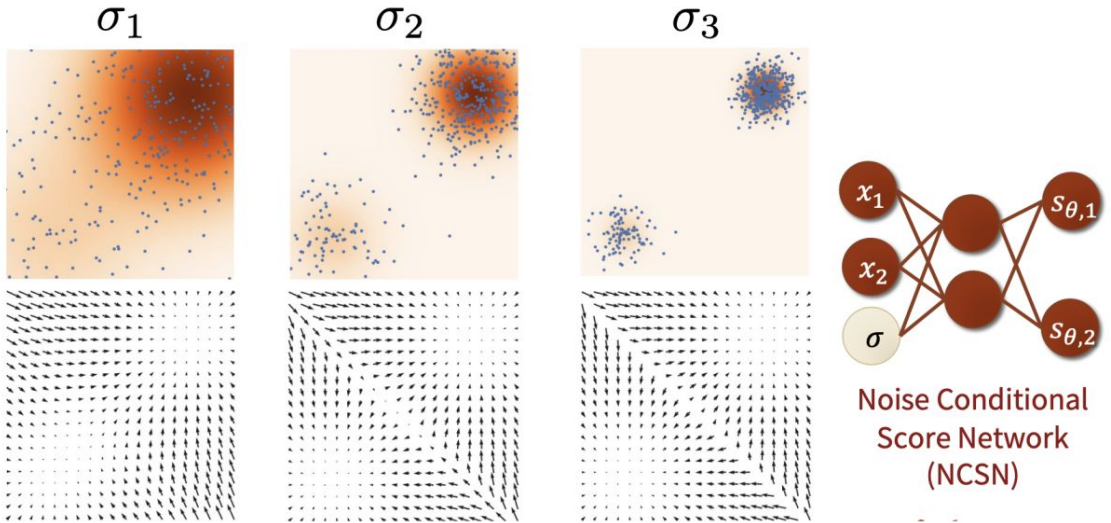
Multi-Scale Noise

$$\sigma_1 > \sigma_2 > \cdots > \sigma_{L-1} > \sigma_L$$



- Train using both small and large amounts of noise
- Start sampling using larger noise, gradually reduce the noise

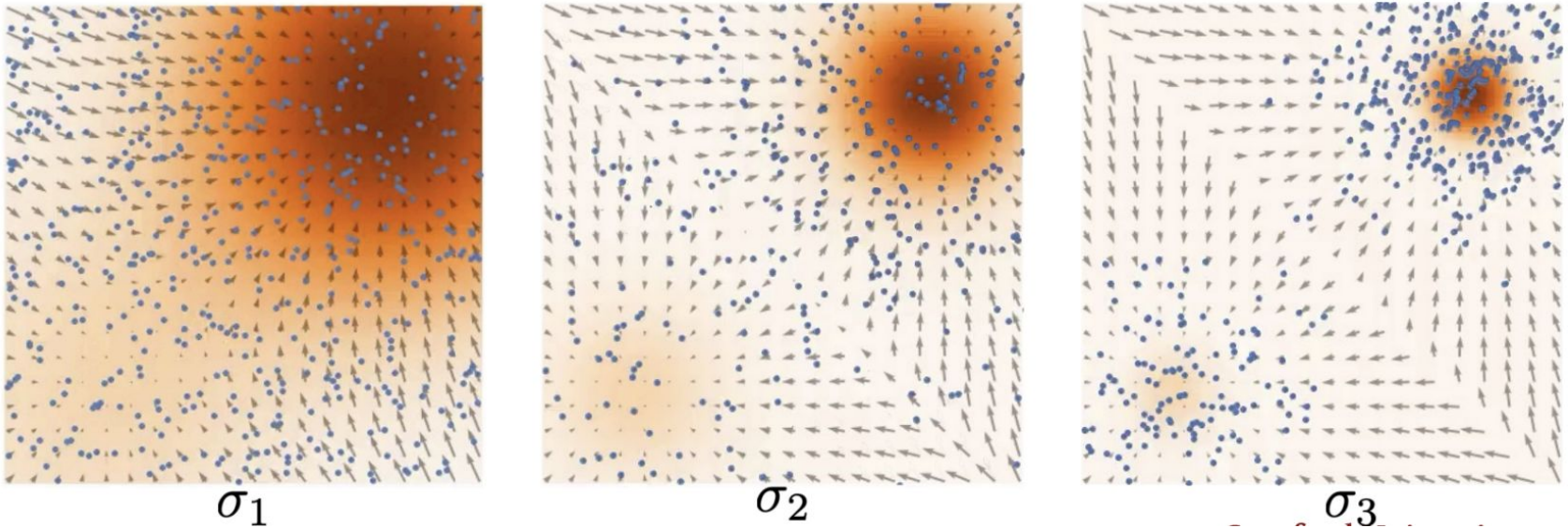
Noise Conditional Score Networks

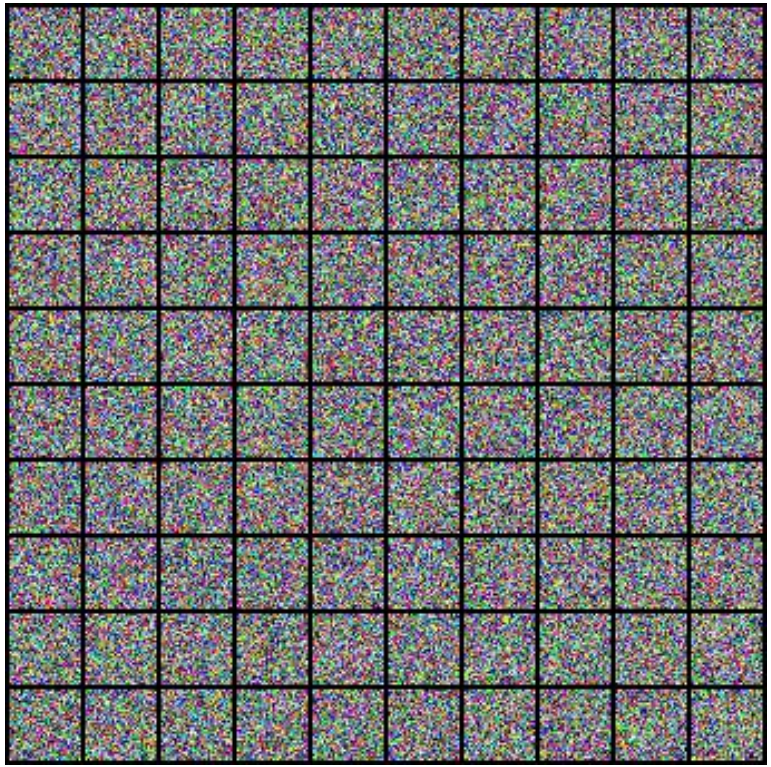


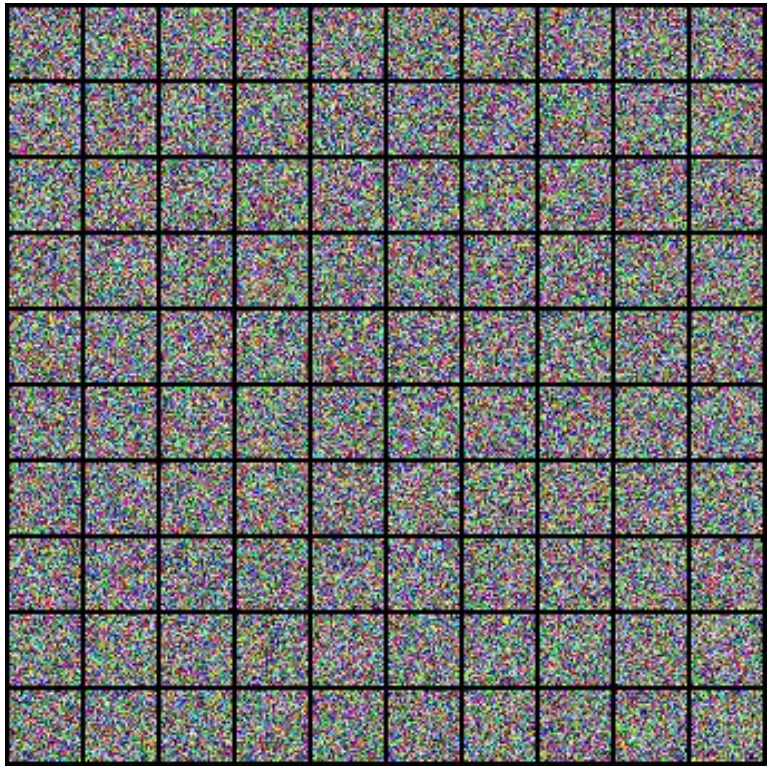
Same model is trained for all σ. Trained using the following objective:

$$\mathbb{E}_{\sigma\sim\{\sigma_1,\sigma_2,\ldots,\sigma_L\}}\mathbb{E}_{x\sim p(x),\epsilon=\mathcal{N}(0,I)}\left[\|\epsilon - s_\theta(x+\sigma\cdot\epsilon,\sigma)\|_2^2\right]$$

Annealed Langevin Dynamics







Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

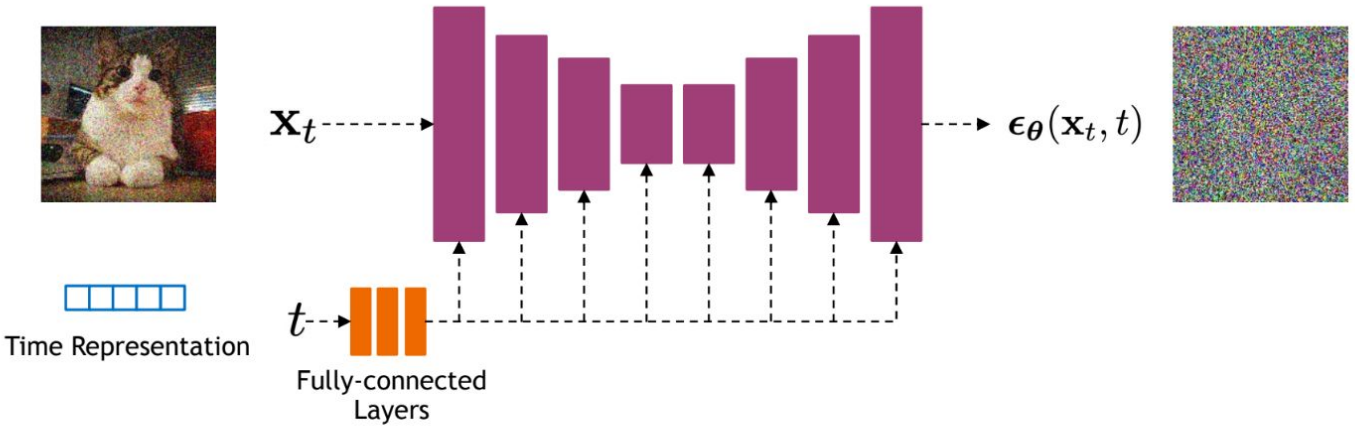
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

- β_t is the variance of noise added to x_{t-1} to form x_t
- $\alpha_t = 1 - \beta_t$
- $\bar{\alpha}_t = \prod_{t'=1}^{t} \alpha_{t'}$

Implementation

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

Outline

- Diffusion models – Recap
- Posterior sampling
- Classifier Guidance
- Classifier free Guidance

- Course Summary

Posterior Sampling

Inverse problem: **x** ⟶ **y**

Generating samples from **p(x | y)** can unlock many applications of approximate probabilistic inference.

Examples:

- Denoising: **y** is the noisy image
- Inpainting: **y** is part of image
- Class conditioning: **y** is a class
- Constraints

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Posterior samples is not the same as MAP

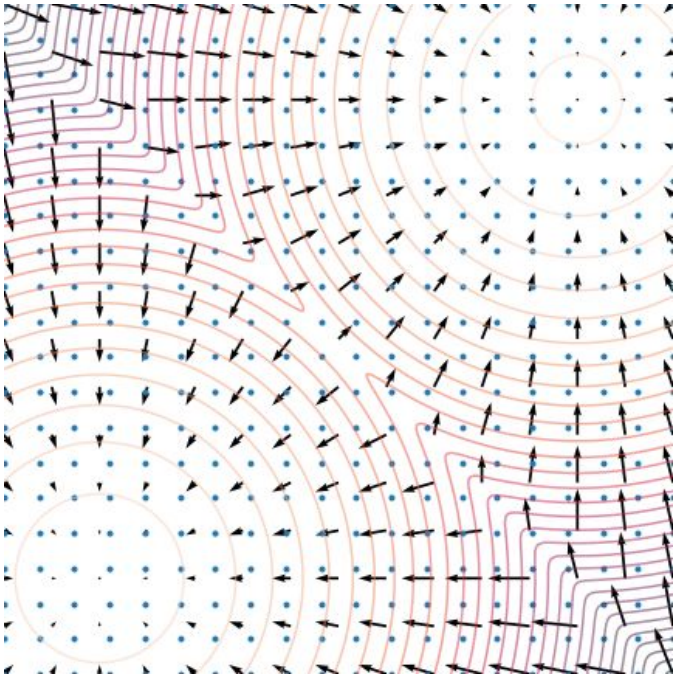
Sampling with score-based models

Langevin dynamics on the prior:

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\, \mathbf{z}_i, \quad i = 0, 1, \cdots, K$$

We can also use Langevin dynamics on a posterior distribution **p(x | y)**

$$\begin{aligned} \nabla_x \log p(x|y) &= \nabla_x \log p(x) + \nabla_x \log p(y|x) - \nabla_x \log p(y) \\ &= \nabla_x \log p(x) + \nabla_x \log p(y|x) \end{aligned}$$



Posterior Score

$$\nabla_{x_t} \log p_t(x_t|y) = \nabla_{x_t} \log p_t(x_t) + \nabla_{x_t} \log p_t(y|x_t)$$

- Use a pre-trained diffusion model as a prior **p(x)**
- Given a task, formulate a corresponding likelihood function **p(y | x)**
- prior score + likelihood score = posterior score

Problem

We usually only have $\mathbf{p(y \mid x_0)}$ and not $\mathbf{p(y \mid x_t)}$

Exact solution requires dealing with the Integral

$$p(y|x_t) = \int p(y, x_0|x_t)\ dx_0$$

$$= \int p(y|x_0)p(x_0|x_t)\ dx_0$$

Solutions

1. Ignore - pointwise integral around $\mathbf{x_t}$

   $\Rightarrow \quad \mathbf{p(y \mid x_0) = p(y \mid x_t)}$

2. Pointwise integral around mean of $\mathbf{x_0}$

   $\Rightarrow \quad \mathbf{p(y \mid x_0) = p(y \mid E[x_0 | x_t])}$

3. Variational inference

Posterior sampling using $E[x_0 \mid x_t]$

$$\hat{\boldsymbol{x}}_0 := \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] = \frac{1}{\sqrt{\bar{\alpha}(t)}}(\boldsymbol{x}_t + (1-\bar{\alpha}(t))\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t))$$

$$\hat{\boldsymbol{x}}_0 \simeq \frac{1}{\sqrt{\bar{\alpha}(t)}}(\boldsymbol{x}_t + (1-\bar{\alpha}(t))\boldsymbol{s}_{\theta^*}(\boldsymbol{x}_t, t))$$

Algorithm 1 Posterior Sampling

Require: $T, \mathbf{y}, \{\zeta_t\}_{t=1}^{T}, \{\sigma_t\}_{t=1}^{T}$

$x_N \sim \mathcal{N}(0, I)$

for $t = T$ to 1 **do**

  $\mathbf{z} \sim \mathcal{N}(0, I)$

  $\mathbf{x}'_{t-1} \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)) + \sigma_t \mathbf{z}$

  $\hat{\mathbf{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - (1-\bar{\alpha}_t)\epsilon_\theta(\mathbf{x}_t, t))$

  $\mathbf{x}_{t-1} \leftarrow \mathbf{x}'_{t-1} - \zeta_t \nabla_{\mathbf{x}_t} \log p(y|\hat{\mathbf{x}}_0)$

end for

return $\hat{\mathbf{x}}_0$

Implementation

- Usually map data values to **[−1, 1]**
- Advantage of passing through $\mathbf{x_0}$**:**

  Can clip values of $\mathbf{x_0}$ that are outside **[−1, 1]** during sampling

  fixes issue with Gaussian approximation of $\mathbf{p(x_{t-1} \mid x_t)}$
- Simplify Posterior sampling: alternate between computing the gradients of prior and likelihood ⇒ no need to backprop through unet.

Algorithm 2 Posterior Sampling - Gaussian

Require: $N, \mathbf{y}, \{\zeta_i\}_{i=1}^{N}, \{\tilde{\sigma}_i\}_{i=1}^{N}$

$x_N \sim \mathcal{N}(0, I)$

for $t = T$ to 1 **do**

  $\hat{\mathbf{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - (1 - \bar{\alpha}_t)\epsilon_\theta(\mathbf{x}_t, t))$

  $\mathbf{z} \sim \mathcal{N}(0, I)$

  $\mathbf{x}'_{t-1} \leftarrow \frac{\sqrt{\alpha_t(1-\bar{\alpha}_{t-1})}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_0 + \tilde{\sigma}_t\mathbf{z}$

  $\mathbf{x}_{t-1} \leftarrow \mathbf{x}'_{t-1} - \zeta_t \nabla_{\mathbf{x}_t} \|\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}}_0)\|_2^2$

end for

return $\hat{\mathbf{x}}_0$

Classifier Guidance

$$\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y \mid x)$$

- **y** can be a class
- **p(y | x)** is a classifier
- Can train a classifier on **x**
- Can solve issue of $\mathbf{x_t}$ vs. $\mathbf{x_0}$ by training also on noisy data

  ⇒ $\mathbf{p(y \mid x_t)}$
- Better results with higher weight to classifier: “guidance”

Guidance Scale

$$\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y \mid x)$$

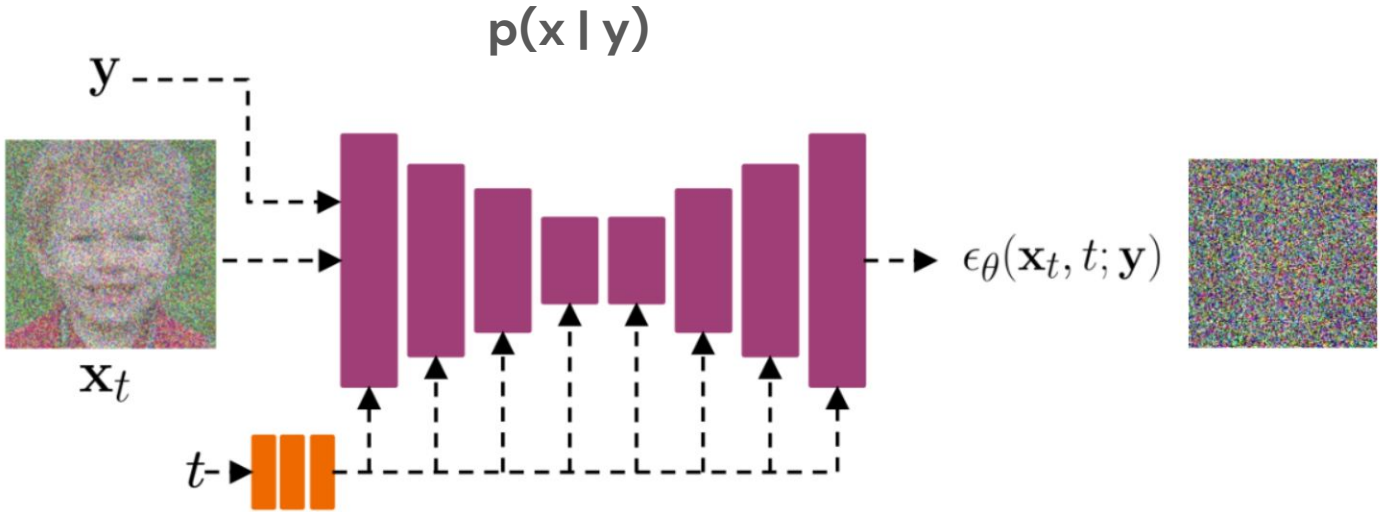
- Weight of likelihood vs. prior
- High values amplify the influence of the conditioned class but result in less diverse samples

scale = 1.0

scale = 10.0

Conditional Training

Directly train a conditional diffusion model instead of classifier



Classifier-Free Guidance

In practice, better results are achieved by:

1. Train a diffusion model that can be used both conditionally and unconditionally. ⇒ using dropout

2. At inference time:

$$(1+w)\nabla_{\mathbf{x}_t}\log p(\mathbf{y}|\mathbf{x}_t) + \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t) = (1+w)\big(\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{y}) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)\big) + \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)$$
$$= (1+w)\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{y}) - w\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t)$$

"Classifier-free guidance": increasing the guidance scale results in better class-conditioned samples (at the expense of less diversity)

Image | “dog”

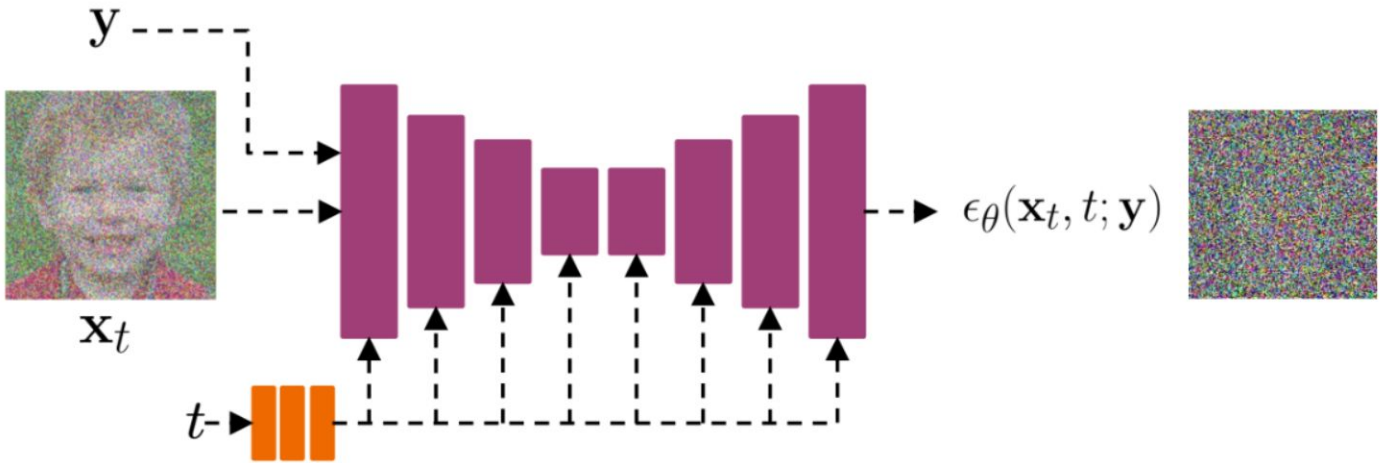
Non-guidance

$\omega = 1$

$\omega = 3$

Image | prompt

- Instead of classes, we can condition the model on text prompts using embeddings of an LLM (or. CLIP embedding)



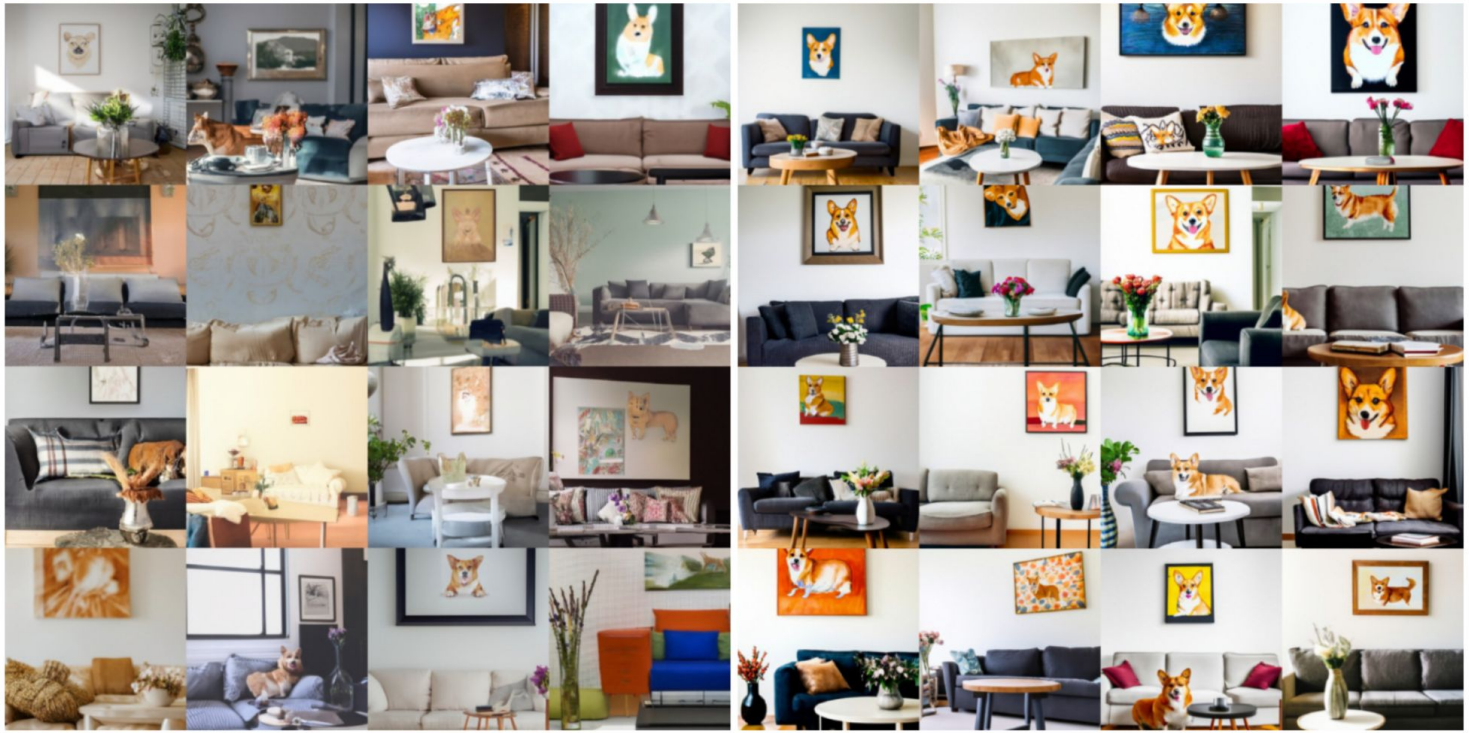
DALL·E 2

“a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese”

IMAGEN

“A photo of a raccoon wearing an astronaut helmet, looking out of the window at night.”

Two sets of samples from OpenAI's GLIDE model, for the prompt *'A stained glass window of a panda eating bamboo.'*, taken from their paper. Guidance scale 1 (no guidance) on the left, guidance scale 3 on the right.



Two sets of samples from OpenAI's GLIDE model, for the prompt *'"A cozy living room with a painting of a corgi on the wall above a couch and a round coffee table in front of a couch and a vase of flowers on a coffee table.'*, taken from their paper. Guidance scale 1 (no guidance) on the left, guidance scale 3 on the right.

Posterior sampling and guidance with diffusion models

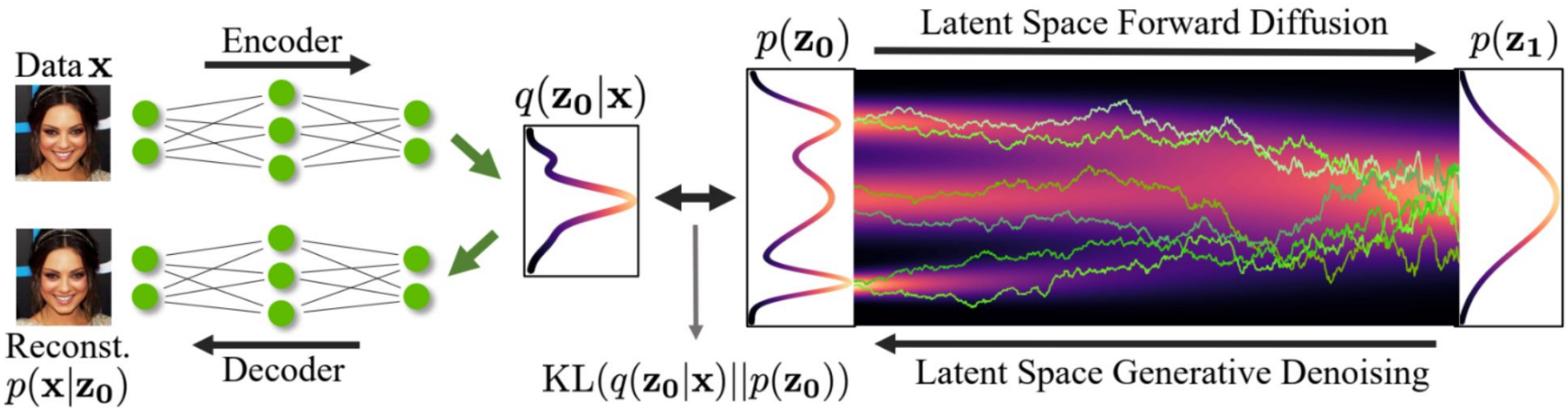
- Easy to use model as a general prior, and plug it in probabilistic inference pipeline

- Combining posterior inference formulation with conditional training achieves very impressive results

- The secret ingredient: huge amounts of data, and efficient training setups:
  E.g . 1. train LLM on large amounts of text
  2. then train conditional image model on smaller amounts of text+image data (still a lot)

Combining Different Approaches

Latent Diffusion



Variational AutoEncoder + GAN loss

Denoising Diffusion Prior

Course Summary

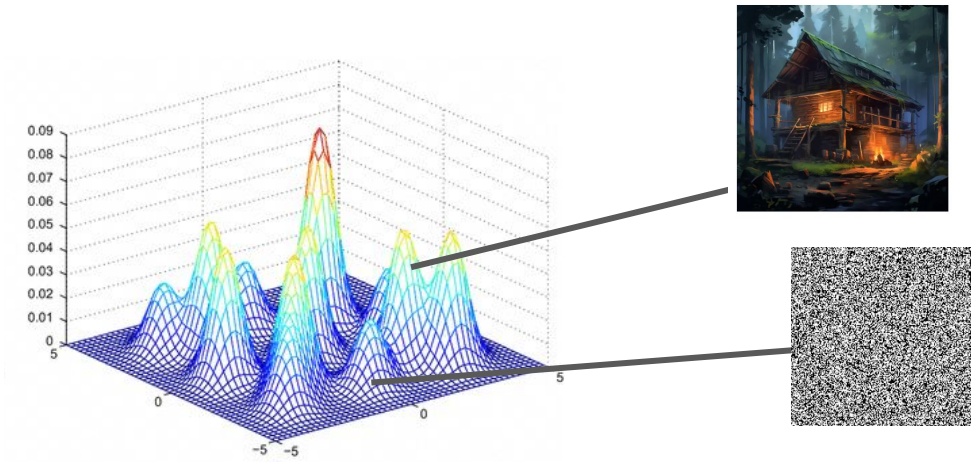
Generative models

Probabilistic model with high dimensional output.

- looking for the parameters θ such that p_θ is close to p_{data}

Usage:

- Generate data
- Representation learning
- Probabilistic inference



Components for training a generative model

1. Data – representative of the space
2. Model and independence assumption / parameterizations

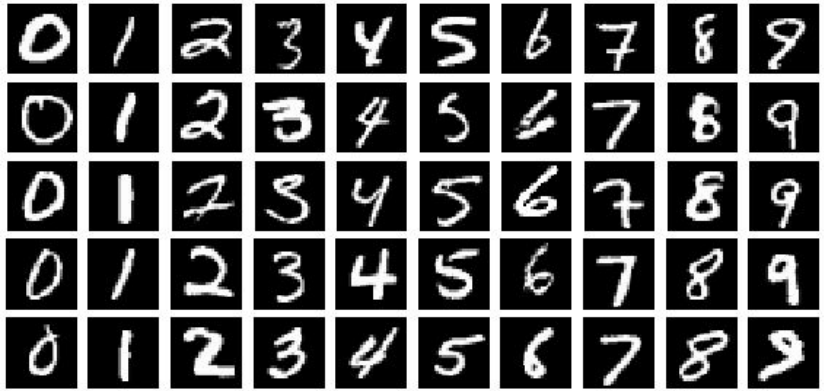
   (e.g. Gaussian, GMM, Latent variable models, autoregressive models)
3. Objective (e.g. maximum likelihood, score matching)
4. Optimization (e.g. Gradient Descent, Variational inference, MCMC)

Modeling p_θ

Bernoulli distribution: biased coin flip

- X domain: **{Heads, Tails}**
- Model: **P(X = Heads) = p, P(X = Tails) = 1 – p**
- Parameter: θ **= p**

- Categorical distribution?
- Multivariate categorical distribution?
- How many parameters?
- Curse of dimensionality



Lifting the curse

Using two important rules in probability

1. **Chain rule** Let $S_1, \ldots S_n$ be events, $p(S_i) > 0$.

$$p(S_1 \cap S_2 \cap \cdots \cap S_n) = p(S_1)p(S_2 \mid S_1) \cdots p(S_n \mid S_1 \cap \ldots \cap S_{n-1})$$

2. **Bayes' rule** Let S_1, S_2 be events, $p(S_1) > 0$ and $p(S_2) > 0$.

$$p(S_1 \mid S_2) = \frac{p(S_1 \cap S_2)}{p(S_2)} = \frac{p(S_2 \mid S_1)p(S_1)}{p(S_2)}$$

Options to reduce number of parameters

Conditional independence / simplifying parameterization given **observed** variables:

$$p(x_1, \ldots, x_n) = \Pi_i \, p(x_i \mid x_{[i-k:i-1]})$$

For Discrete representation:

$d^k \cdot (d-1) \cdot n$ parameters

Conditional independence given **latent** variables:

$$p(x_1, \ldots, x_n) = \Pi_i \, p(x_i \mid z)$$

For Discrete representation:

$|h| \cdot (d-1) \cdot n$ parameters

Constrained parameterization:
continuous rep. function approximation (neural nets)

Options to reduce number of parameters

Conditional independence / simplifying parameterization given **observed** variables:

$$p(x_1, \ldots, x_n) = \Pi_i \, p(x_i \mid x_{[i-k \,:\, i-1]})$$

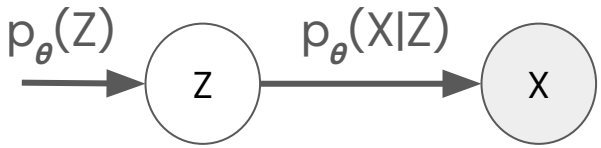
- Easy log-likelihood optimization
- Examples:
  - NADE
  - PixelRNN
  - PixelCNN

Conditional independence given **latent** variables:

$$p(x_1, \ldots, x_n) = \Pi_i \, p(x_i \mid z)$$

- Hard log-likelihood optimization:
  - EM, MCMC, Variational Inference
- Examples:
  - GMM
  - Variational AutoEncoders (VAEs)
  - GANS
  - Normalizing Flows
  - Diffusion models

Latent Variable Models



- Assume there's an additional variable which we don't see (latent, hidden)
- Use it to construct an efficient conditional independence structure
- This can solve the issues we had (efficient non-Gaussian models)
- Has a more natural interpretation
- Semi-Bayesian approach:
  - Unknown latent variable **Z** + unknown parameters $\boldsymbol{\theta}$.
  - Maximum likelihood for $\boldsymbol{\theta}$, Bayesian for **Z**

Bayesian Inference Methods

Two ways to deal with latent variables:

1. Sampling (MCMC, Gibbs, Langevin)
2. Variational Inference (inference via optimization)

Can be used within model, or for probabilistic inference at test time

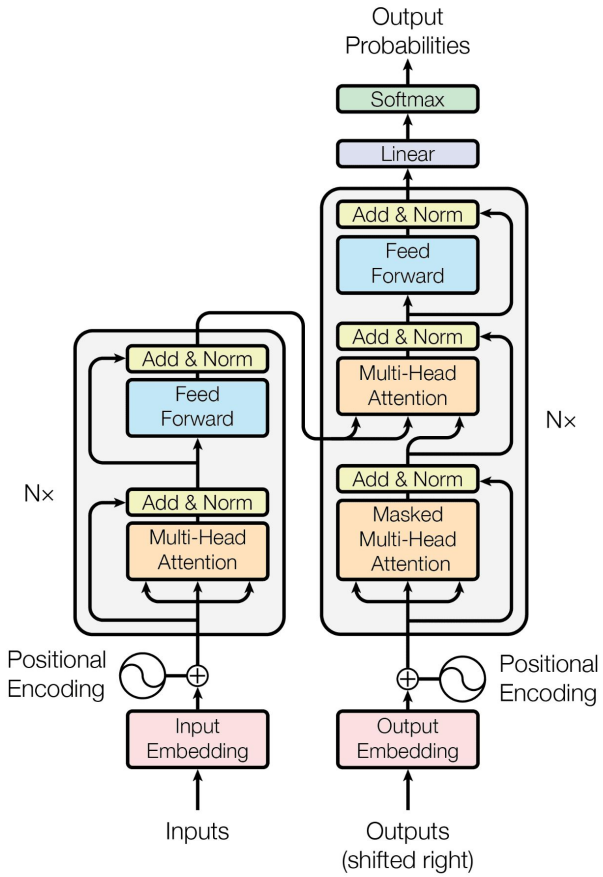
Practical Experience

- Multivariate Gaussians
- MCMC methods (Gibbs, Langevin)
- Expectation Maximization (EM)
- Gaussian Mixture Models
- General probabilistic inference
- Training deep generative models:
  - PixelCNN
  - VAE
  - Normalizing Flows
  - Diffusion Models

Not Covered in the Course

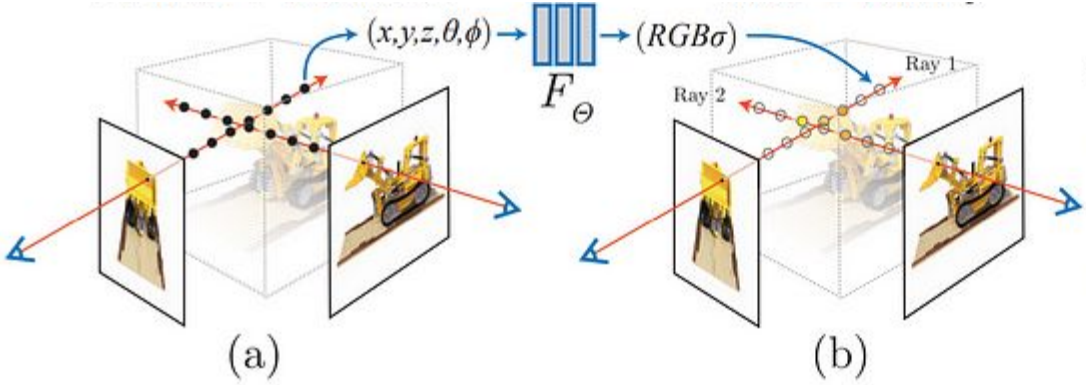
Language Models

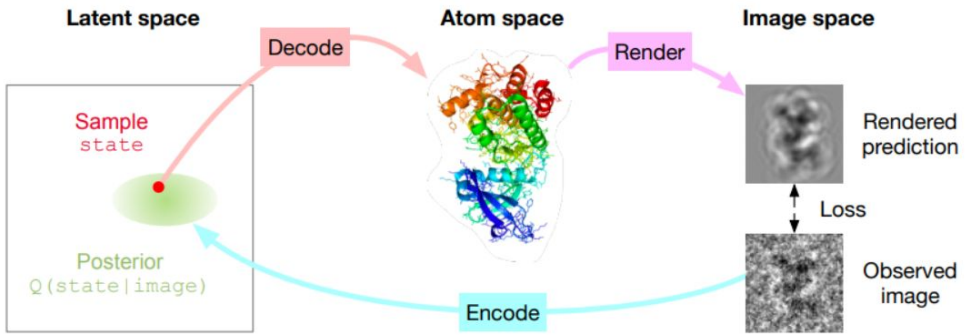
- Think of PixelCNN + Transformer architecture
- Transformer is also used in image models



Exciting Research

- Improving models:
  efficiency, uncertainty
- New domains:
  3D scenes, molecule structure
- Real world applications:
  underwater imagery,
  protein design





Thank you!