Diffusion

EECS 442 Fall 2023

Presenters: Sarah Jabbour, Yiming Dou, Daniel Geng
Recall: Data Synthesis

Let's say we have some training set of data following some distribution $p_{\text{data}}(x)$:

The goal of generative machine learning models is to learn this distribution to the best of their ability.

We generate new data by sampling from the learned distribution.
2 Methods for Learning Data Distribution

(1) We want to train models that maximize the expected log likelihood of $p_\theta(x)$. I.e., If I sample from the distribution and get a
- high likelihood $\rightarrow$ likely the sample came from the training distribution
- low likelihood $\rightarrow$ the sample probably didn’t come from the training distribution

(2) We want to minimize some divergence metrics between the training data distribution, and the distribution that the model learns
Sampling from Noise

Source distribution \( p(z) \) \( \rightarrow \) Target distribution \( p(x) \)
GANs

- Convert latent noise vector to target distribution in one step

- Lots of issues with training:
  - Vanishing gradients: if discriminator is too good, gradients go to zero
  - Mode collapse: if the generator learns to generate an exceptionally plausible output, it will just continue generating it. Then the discriminator will learn to always reject it, and then the generator will produce the same outputs, which the discriminator will then reject... bad loop!
Diffusion

- Idea: Estimating and analyzing small step sizes is more tractable/easier than a single step from random noise to the learned distribution
- Convert a well-known and simple base distribution (like a Gaussian) to the target (data) distribution iteratively, with small step sizes, via a Markov chain:

  Diffusion models:
  Gradually add Gaussian noise and then reverse

- Markov chain: outlines the probability associated with a sequence of events occurring based on the state in the previous event.
Forward Process

- Noise added can be parameterized by:

\[
q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I) \quad q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \quad \{\beta_t \in (0, 1)\}_{t=1}^{T}
\]

Vary the parameters of the Gaussian according to a *noise schedule*

- You can prove with some math that as \( T \) approaches infinity, you eventually end up with an Isotropic Gaussian (i.e. pure random noise)

- Note: forward process is fixed
Reparameterization trick

Do you have to add noise iteratively to get to some timestep $t$? Nope!

Reverse process can be written in one step:

$$q(x_t \mid x_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I\right)$$

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

This will be useful during training!
Implementing Forward Process

\[ q(x_t | x_0) = \mathcal{N}(\sqrt{\alpha_t}x_0, (1 - \alpha_t)I) \]

\[ \alpha_t = 1 - \beta_t \]

\[ \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i \]

1. Sample an image from the dataset: 🐱
2. Sample noise \( \epsilon \sim N(0, I) \) (from a standard normal distribution)
3. Scale the image by \( \sqrt{\alpha_t} \): \( \sqrt{\alpha_t} x_0 \)
   where
   \[ \alpha_t = 1 - \beta_t \]
   \[ \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i \]
4. Add \( \sqrt{1 - \bar{\alpha}_t} \epsilon \): \( \sqrt{\alpha_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon \) 🐱
Reverse Process
Reverse Process

\[ X_T \rightarrow \cdots \rightarrow X_t \rightarrow X_{t-1} \rightarrow \cdots \rightarrow X_0 \]

\[ q(x_t | x_{t-1}) \]
Reverse Process

\[ x_T \rightarrow \cdots \rightarrow x_t \rightarrow x_{t-1} \rightarrow \cdots \rightarrow x_0 \]

\[ q(x_t | x_{t-1}) \]

\[ q(x_{t-1} | x_t) \text{ is unknown} \]
The goal of a diffusion model is to **learn** the reverse *denoising* process to iteratively **undo** the forward process.

In this way, the reverse process appears as if it is generating new data from random noise!
Diffused Data Distribution
Diffused Data Distribution

$q(x_0)$  $q(x_1)$  $q(x_2)$  $q(x_3)$  $q(x_4)$  $q(x_5)$

$x_t$  $x_5 = X$

True Denoising Distribution

$q(x_4|x_5 = X)$
What should the distribution look like?

Turns out that for small enough forward steps, i.e. \( \{\beta_t \in (0, 1)\}_{t=1}^{T} \)

the reverse process step \( q(x_{t-1} \mid x_t) \) can be estimated as a Gaussian distribution too.

Therefore, we can parametrize the learned reverse process as

\[
p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))
\]

In practice, \( \Sigma \) is just the identify matrix, so we only need to learn the mean of the distribution.
Preliminary objective

When we write out the loss function, we get something that looks like this:

\[
L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0
\]

where

\[
L_T = D_{\text{KL}}(q(x_T|x_0) \parallel p_\theta(x_T))
\]

\[
L_t = D_{\text{KL}}(q(x_t|x_{t+1}, x_0) \parallel p_\theta(x_t|x_{t+1})) \text{ for } 1 \leq t \leq T - 1
\]

\[
L_0 = -\log p_\theta(x_0|x_1)
\]
Middle Loss Term - Intuition

\[ L_t = D_{KL}(q(x_t|x_{t+1}, x_0) \parallel p_\theta(x_t|x_{t+1})) \text{ for } 1 \leq t \leq T - 1 \]

KL Divergence: measures distance between two distributions

→ If high, very dissimilar distributions

→ If low, very similar distributions

Goal: drive this very low
Final Loss

Recall, our goal was to learn the following $\mu_\theta$ (network that parameterizes the mean of the data distribution):

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

So we minimize:

$$\text{MSE}(\mu_\theta(x_t, t), x_{t-1})$$
How do we do this in practice?

Step 1: Sample image from the dataset, generate noisy image using forward process

\[ q(x_t \mid x_0) = \mathcal{N}\left(\sqrt{\alpha_t}x_0, (1 - \alpha_t)I\right) \]

Step 2: Given noisy image, generate slightly noisier image
How do we do this in practice?

\[ x_{t+1} \quad \hat{x}_t \]

Loss: \( \text{MSE}(x_t, \hat{x}_t) \)
Neural Network that predicts noise

Input  U-net  Output
Training

Algorithm 1 Training

1: repeat
2: \( x_0 \sim q(x_0) \)
3: \( t \sim \text{Uniform}(\{1, \ldots, T\}) \)
4: \( \epsilon \sim \mathcal{N}(0, I) \)
5: Take gradient descent step on
\[
\nabla_{\theta} \left\| \epsilon - \epsilon_{\theta}(\sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \epsilon, t) \right\|^2
\]
6: until converged
Forward process: converting the image distribution to pure noise

Reverse process: sampling from the image distribution, starting with pure noise
Diffusion Models Beats GANs

BigGAN  Diffusion  Training Set
Diffusion Models Beats GANs

BigGAN

Diffusion

Training Set
Problem: operating in the input space is very computationally expensive!
Option #1: Generate Low-Resolution + Upsample

Input: 1024x1024

U-net: 256x256

Output: 1024x1024

Downsample: 256x256

Upsample: 256x256
Option #2: Generate in Latent Space

U-net

Input

1024x1024

Encoder

Downsample

Output

U-net

1024x1024

Decoder

Upsample
Stable Diffusion

What's going on here?
Guided/Conditioned Diffusion

Lets say we train a diffusion model on images of cats and dogs:

If we start from random noise, and generate a new image, what will the model generate?
Leveraging Diffusion Models for Visual-Tactile Cross Generation

EECS 442 Team
Contents

1. Background

2. Previous Methods

3. Learning to Read Braille (Vision-to-Touch)

4. Generating Visual Scenes from Touch (Touch-to-Vision)
1 Background: Tactile Sensor & Touch Images

Figure 2: Exploded view of a single DIGIT sensor. A) elastomer, B) acrylic window, C) snap-fit holder, D) lighting PCB, E) plastic housing, F) camera PCB, G) back housing.
2 Previous Methods

VisGel [1]: GAN-based exocentric generation

ObjectFolder [2]: GAN-based egocentric generation


3 Learning to Read Braille (Vision-to-Touch)

Data: Simulated local depth map & Real tactile images collected on YCB dataset
3.1 Training Tactile Diffusion

- **Data:**
  
  Simulated local depth map & Real tactile images collected on YCB dataset

- **Diffusion decoder:**
  
  Conditional U-Net backbone that takes depth map as input and renders colorful tactile images

- **Evaluation:**
  
  SSIM (structural similarity) & MSE (mean squared error)
3.1 Training Tactile Diffusion

Simulation, real, tactile diffusion results
(SSIM is generally above 0.80)
3.2 Training Braille Classifier in Simulator

- Sim2Real Transfer:
  
  Train a classifier to detect real-world braille letters with DIGIT sensor

- Comparison:
  
  Compare results from Sim / cGAN / Diffusion / Real data.
3.3 Reading Braille with Real-World Sensor

<table>
<thead>
<tr>
<th>Training data source</th>
<th>% real data fine-tuning</th>
<th>Accuracy %</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sim</td>
<td>-</td>
<td>30.23</td>
<td>0.34</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>64.99</td>
<td>0.71</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>73.11</td>
<td>0.80</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>73.95</td>
<td><strong>0.81</strong></td>
<td>0.74</td>
</tr>
<tr>
<td>Sim + data aug.</td>
<td>-</td>
<td>43.48</td>
<td>0.61</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>73.23</td>
<td>0.76</td>
<td>0.73</td>
</tr>
<tr>
<td>cGAN</td>
<td>-</td>
<td>31.18</td>
<td>0.40</td>
<td>0.31</td>
</tr>
<tr>
<td>Tactile diffusion</td>
<td>-</td>
<td><strong>75.74</strong></td>
<td>0.79</td>
<td><strong>0.76</strong></td>
</tr>
<tr>
<td>Real</td>
<td>-</td>
<td>100.0</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Training cGAN on 100% real, tactile diffusion on YCB-Slide + 20% real
3.3 Reading Braille with Real-World Sensor
4 Generating Visual Scenes from Touch (Touch-to-Vision)

(a) Touch-to-Image Generation

(b) Tactile-driven Image Stylization

(c) Tactile-driven Shading Estimation

4.1 Contrastive Visuo-tactile Pretraining (CVTP)

Given N visual-tactile image pairs, sample K from them and perform contrastive learning (mapping them into the uniform hidden space) using InfoNCE loss:

$$L_{i}^{V_{I}, V_{T}} = -\log \frac{\exp(E_{\phi_{I}}(v_{I}^{i}) \cdot E_{\phi_{T}}(v_{T}^{i})/\tau)}{\sum_{j=1}^{K} \exp(E_{\phi_{I}}(v_{I}^{j}) \cdot E_{\phi_{T}}(v_{T}^{j})/\tau)}$$  (1)
4.2 Touch-conditioned Image Generation

- Touch signal is represented by multi-frames from tactile sensor
- The diffusion process is conditioned on the touch signal.

The loss function is:

$$L(\theta, \phi) = \mathbb{E}_{z_I, c, \epsilon, t} \left[ \| \epsilon_t - \epsilon_\theta(z_I^t, t, E_{\phi_T}(\nu_T)) \|^2_2 \right]$$
4.3 Qualitative Results: Tactile-driven Image Stylization

Touch and Go: An indoor-outdoor dataset with humans holding DIGIT sensor to touch objects
4.4 Qualitative Results: Visual-Tactile Cross Generation

VisGel: A dataset that collects paired touch videos and third-view robot arms.

VisGel [38]
Impressive Results

**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.”

---

How to Control Diffusion Models

- Explicit Conditioning
- Classifier Guidance
- Classifier-Free Guidance

Slide Credit: CVPR 2023 Diffusion Models Tutorial
How to Control Diffusion Models

- Explicit Conditioning
- Classifier Guidance
- Classifier-Free Guidance

Slide Credit: CVPR 2023 Diffusion Models Tutorial
Explicit Conditioning

“a young schoolboy in a red shirt”
Explicit Conditioning

“a young schoolboy in a red shirt”
Explicit Conditioning

How do we train this?
Explicit Conditioning

How do we train this?

Use an Image-Text dataset (for example, LAION 5B)
Explicit Conditioning

How do we train this?

Use an Image-Text dataset (for example, LAION 5B)
How to Control Diffusion Models

Explicit Conditioning

Classifier Guidance

Classifier-Free Guidance
Classifier Guidance

Diffusion goes from noise to real images step-by-step
Classifier Guidance

Diffusion goes from noise to real images step-by-step
Classifier Guidance

Diffusion goes from noise to real images step-by-step

Idea: Perturb the Denoising Trajectory

Image Credit: CVPR 2023 Diffusion Models Tutorial
Classifier Guidance

How do we get this perturbation?
Classifier Guidance

How do we get this perturbation?

Let’s take an image classifier $p(x|y)$
Classifier Guidance

How do we get this perturbation?

Let’s take an image classifier $p(x|y)$

And look at it’s gradients (w.r.t. $x$) $\nabla_x \log p(x|y)$
Classifier Guidance

How do we get this perturbation?

Let’s take an image classifier \( p(x|y) \)

And look at its gradients (w.r.t. \( x \)) \( \nabla_x \log p(x|y) \)

Intuitively: how to change \( x \) so it looks like a “\( y \)”
Classifier Guidance

Perturb using gradients of a classifier:
Classifier Guidance

Perturb using gradients of a classifier:

\[ \mathbf{x}_T \rightarrow \mathbf{x} \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2 \rightarrow \mathbf{x}_3 \rightarrow \mathbf{x}_0 \]
Classifier Guidance

Perturb using gradients of a classifier:

$$\gamma \nabla_{x_t} \log p(x_t | y)$$
Classifier Guidance

Perturb using gradients of a classifier:

$$\gamma \nabla_{x_t} \log p(x_t | y)$$

$$\tilde{\epsilon}_t(x_t, t, y) = \epsilon_\theta(x_t, t) - \gamma \nabla_{x_t} \log p(x_t | y)$$
Classifier Guidance

Perturb using gradients of a classifier:

$$\gamma \nabla_{x_t} \log p(x_t | y)$$

$$\tilde{\epsilon}_t(x_t, t, y) = \epsilon_\theta(x_t, t) - \gamma \nabla_{x_t} \log p(x_t | y)$$

There’s a small problem though…
Classifier Guidance

**Problem:** Classifier isn’t trained on noisy images!
Classifier Guidance

**Problem:** Classifier isn’t trained on noisy images!

**Solution:** Finetune the classifier on noisy images
Classifier Guidance

**Problem:** Classifier isn’t trained on noisy images!

**Solution:** Finetune the classifier on noisy images
Classifier Guidance

Problem: Classifier isn’t trained on noisy images!

Solution: Finetune the classifier on noisy images

( Labrador Retriever )  Augment  ( Labrador Retriever )
Classifier Guidance

Guidance Weight 1.0

Guidance Weight 10.0

Dhariwal and Nichol, “Diffusion Models Beat GANs on Image Synthesis”
Problems with Classifier Guidance
Problems with Classifier Guidance

- Need to fine-tune or re-train a classifier on noisy data
Problems with Classifier Guidance

- Need to fine-tune or re-train a classifier on noisy data
- Need a pre-trained classification model
Problems with Classifier Guidance

- Need to fine-tune or re-train a classifier on *noisy* data
- Need a pre-trained classification model
  - What if we want to use *any* text prompt as input?
Problems with Classifier Guidance

- Need to fine-tune or re-train a classifier on **noisy** data
- Need a pre-trained classification model
  - What if we want to use *any* text prompt as input?
- Classifier gradients are poor. They can suffer from “shortcuts”
How to Control Diffusion Models

Explicit Conditioning

Classifier Guidance

Classifier-Free Guidance

Slide Credit: CVPR 2023 Diffusion Models Tutorial
Classifier Free Guidance

Idea: Use the diffusion model itself to get perturbations for guidance
Classifier Free Guidance

Idea: Use the diffusion model itself to get perturbations for guidance

Train an explicitly conditioned diffusion model: $\varepsilon_\theta(x_t, t, y)$
Classifier Free Guidance

Idea: Use the diffusion model itself to get perturbations for guidance

Train an explicitly conditioned diffusion model: \( \epsilon_\theta(x_t, t, y) \)

But also train it to be unconditional
Classifier Free Guidance

Idea: Use the diffusion model itself to get perturbations for guidance

Train an explicitly conditioned diffusion model: \( \epsilon_{\theta}(x_t, t, y) \)

But also train it to be unconditional

We can do this with *conditioning dropout*: \( \epsilon_{\theta}(x_t, t, \emptyset) \)
Classifier Free Guidance

\[ \epsilon_\theta(x_t, t, y) \]

\[ \epsilon_\theta(x_t, t, \emptyset) \]
Classifier Free Guidance

\[ \epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t, \emptyset) \]
Classifier Free Guidance

\[ \epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t, \emptyset) \]

“Direction” from unconditional sample to conditional sample
Classifier Free Guidance

\[ \epsilon_{\theta}(x_t, t, y) - \epsilon_{\theta}(x_t, t, \emptyset) \]

“Direction” from unconditional sample to conditional sample

Use this as our guidance perturbation
Our new noise estimate will then be:

\[
\tilde{\epsilon}(x_t, t, y) = \epsilon_\theta(x_t, t, \emptyset) + \gamma(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t, \emptyset))
\]

“Direction” from unconditional to conditional
Classifier Free Guidance

“A stained glass window of a panda eating bamboo”

\[ \gamma = 1 \]

Equivalent to explicit conditioning.
No guidance
Classifier Free Guidance

“A stained glass window of a panda eating bamboo”

\[ \gamma = 1 \]

Equivalent to explicit conditioning.
No guidance

\[ \gamma = 3 \]

Nichol et al. “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”
Classifier Free Guidance

“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.”

\[ \gamma = 1 \]

Equivalent to explicit conditioning.
No guidance

Nichol et al. “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”
Classifier Free Guidance

“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.”

\[ \gamma = 1 \]

Equivalent to explicit conditioning. No guidance

\[ \gamma = 3 \]

Nichol et al. “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”
More Resources


- Guidance Tutorial by Sander Dieleman: https://sander.ai/2022/05/26/guidance.html
Image Editing with Diffusion Models
SDEdit

Idea: Add noise to an image, and then remove it with a diffusion model

Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”
Idea: Add noise to an image, and then remove it with a diffusion model

Add noise by running the forward process: $q(x_t | x_0)$
SDEdit

Stroke Painting to Image

Input
SDEdit

Stroke Painting to Image

Input

Output

Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”
SDEdit
Stroke-based Editing

Source

Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”
SDEdit
Stroke-based Editing

Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”
SDEdit
Stroke-based Editing

Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”
Prompt-to-Prompt

“A basket full of apples.”

Source image

apples → cookies

apples → oranges

apples → chocolates

apples → kittens

basket → bowl

basket → box

basket → nest

Hertz et al. “Prompt-to-Prompt Image Editing with Cross-Attention Control”
Prompt-to-Prompt

“A photo of a butterfly on a flower.”

Source image

flower → bread

flower → mug

flower → computer

flower → mirror

butterfly → bird

butterfly → snail

butterfly → drone
(High Level) Idea: Features inside diffusion models encode very high level information such as: style, content, and structure.

Hertz et al. “Prompt-to-Prompt Image Editing with Cross-Attention Control”
(High Level) Idea: Features inside diffusion models encode very high level information such as: style, content, and structure.

Reuse (copy and paste) the features from the previous prompt.

Hertz et al. “Prompt-to-Prompt Image Editing with Cross-Attention Control”
Motion Guidance
Motion Guidance

Earlier: We saw we can do classifier guidance with an ImageNet classifier.
Motion Guidance

Earlier: We saw we can do classifier guidance with an ImageNet classifier.
Motion Guidance

We can use other models besides image classifiers for classifier guidance
Motion Guidance

We can use other models besides image classifiers for classifier guidance

Idea: Let’s do classifier guidance with a “motion estimator” (optical flow network)
Motion Guidance

We can use other models besides image classifiers for classifier guidance

Idea: Let’s do classifier guidance with a “motion estimator” (optical flow network)

Teed and Deng. “RAFT: Recurrent All Pairs Field Transforms for Optical Flow”
Motion Guidance

“a photo of topiary”
Motion Guidance

“a photo of topiary”
Motion Guidance

a) “a teapot floating in water”
Motion Guidance

a) “a teapot floating in water”
Motion Guidance
Motion Guidance