CS5489
Lecture 12.1: Neural Networks and Deep Learning Part VII: Advanced Topics

Kede Ma

City University of Hong Kong (Dongguan)

Slide template by courtesy of Benjamin M. Marlin

Outline

1 Background

2 Generative Adversarial Networks (GANs)

3 Diffusion Models

Generative Models

Generative models reproduce the probabilistic process that generates a set of examples $\mathcal{D}$

- For $\mathcal{D} = \left\{ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{M}$: Supervised generative models learn the joint distribution $p\left(\mathbf{x}, y\right)$, often by estimating $p\left(\mathbf{x} \mid y\right)$

- For $\mathcal{D} = \left\{ \mathbf{x}^{(i)} \right\}_{i=1}^{M}$: Unsupervised generative models learn the distribution $p\left(\mathbf{x}\right)$
  - By direct estimation
  - By introducing a set of latent variables $\mathbf{z}$ and estimating $p\left(\mathbf{x}, \mathbf{z}\right)$, followed by marginalization

Generative Models

- **Direct estimation:** Choose a parameterized family $p(\mathbf{x}; \boldsymbol{\theta})$ and learn $\boldsymbol{\theta}$ by maximizing the log-likelihood

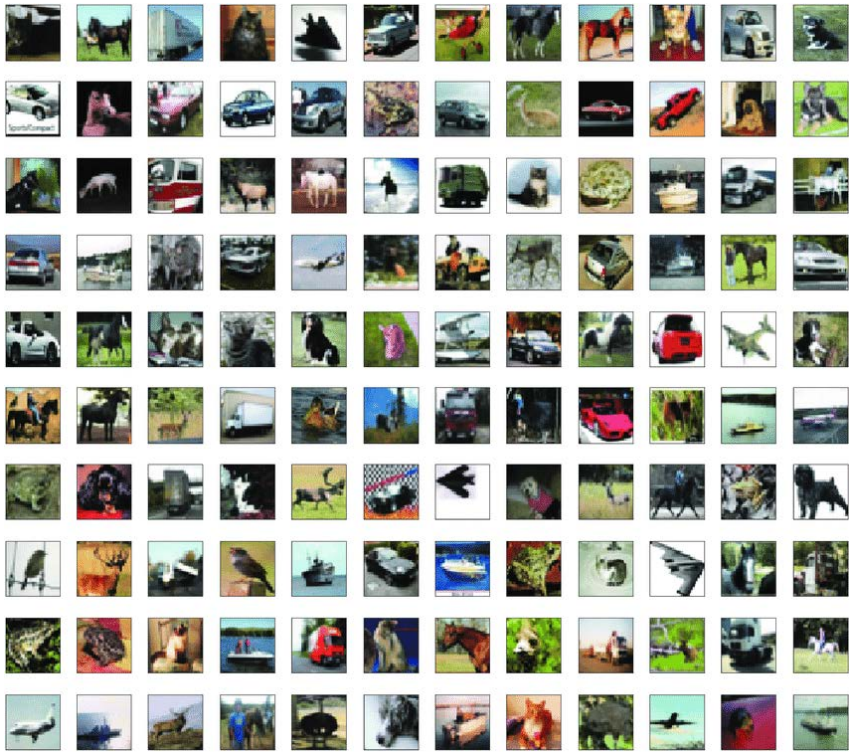
$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \frac{1}{M} \sum_{i=1}^{M} \log p\left(\mathbf{x}^{(i)}; \boldsymbol{\theta}\right)$$

- **Latent variable models:** Define a joint distribution $p\left(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}\right)$ and learn $\boldsymbol{\theta}$ by maximizing the log-marginal likelihood

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \frac{1}{M} \sum_{i=1}^{M} \log \int_{\mathbf{z}^{(i)}} p\left(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}; \boldsymbol{\theta}\right) d\mathbf{z}^{(i)}$$

  - Capture the hidden factors that generate the data
  - Provide more interpretable and meaningful representations of the data

Generative Models



How can we learn such generative models for very complex and highly structured data?

Generative Models

We can design parameterized densities with huge capacity!

- **Normalizing flows:** Transform a simple distribution (e.g., an isotropic Gaussian distribution) into a more complicated one, with a sequence of non-linear and *bijective* transformations $\mathbf{f}_{\boldsymbol{\theta}_{0:K}}$

$$\mathbf{x} = \mathbf{f}_{\boldsymbol{\theta}_{0:K}}(\mathbf{z}) = \mathbf{f}_{\boldsymbol{\theta}_0} \circ \mathbf{f}_{\boldsymbol{\theta}_1} \circ \cdots \circ \mathbf{f}_{\boldsymbol{\theta}_K}(\mathbf{z}),$$

  where $\mathbf{z} = \mathbf{f}_{\boldsymbol{\theta}_K}^{-1} \circ \cdots \circ \mathbf{f}_{\boldsymbol{\theta}_1}^{-1} \circ \mathbf{f}_{\boldsymbol{\theta}_0}^{-1}(\mathbf{x})$

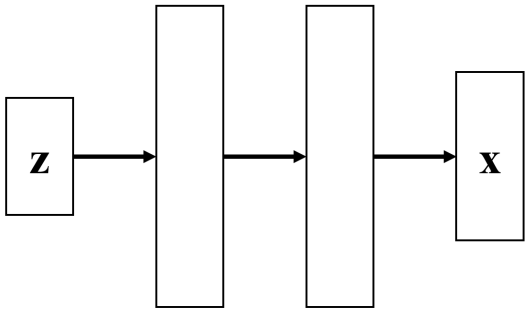
- **Variational autoencoders** (**VAEs**): Latent-variable models where the encoder network specifies the parameters of the distribution of the latent variables

$$\mathbf{z} \sim p(\mathbf{z}; \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})), \quad \hat{\mathbf{x}} = \mathbf{g}_{\boldsymbol{\phi}}(\mathbf{z})$$

  - $\mathbf{f}_{\boldsymbol{\theta}}$ is the encoder network
  - $\mathbf{g}_{\boldsymbol{\phi}}$ is the decoder network

Implicit Generative Models

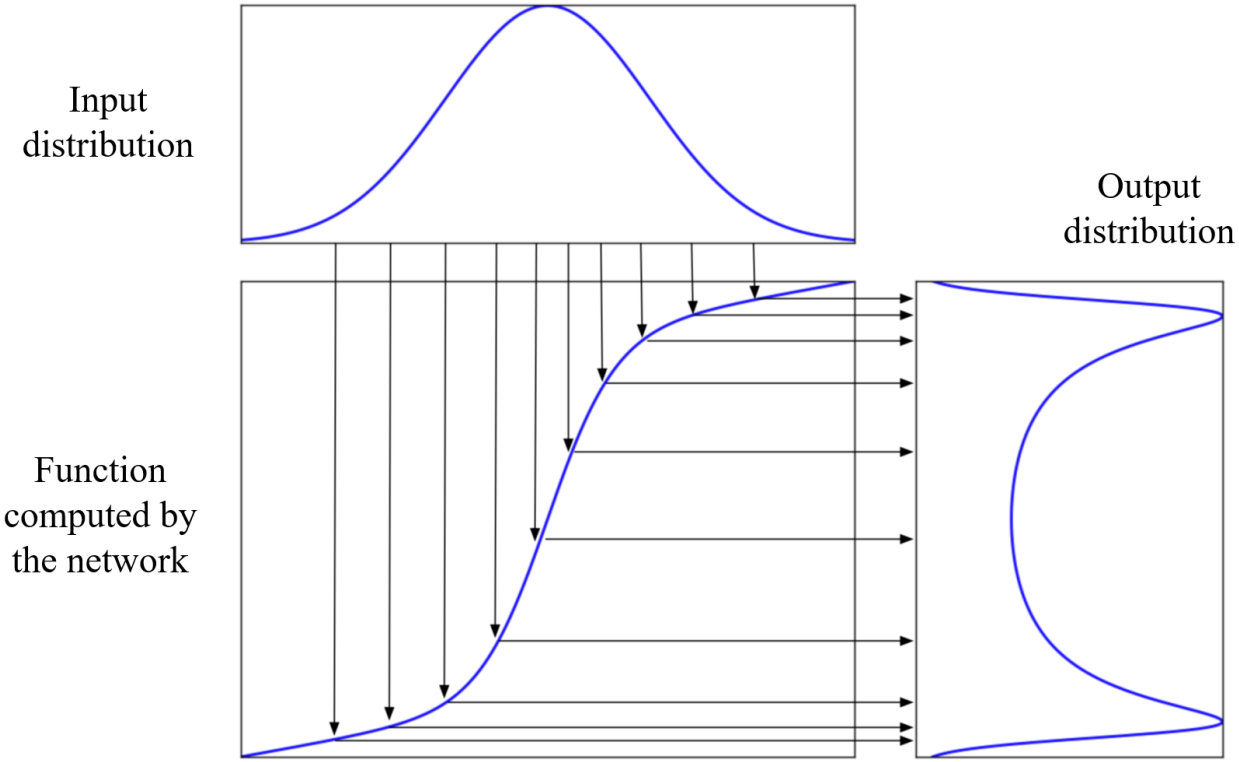
- Implicit generative models implicitly define probability distributions through their sampling procedures
- The generation starts by sampling the latent vector $\mathbf{z}$ from a fixed, simple distribution (e.g., Gaussian)
- The generator network G serves as a differentiable function, mapping $\mathbf{z}$ to $\mathbf{x}$ in input data space



$$\mathbf{x} = G(\mathbf{z})$$

Implicit Generative Models

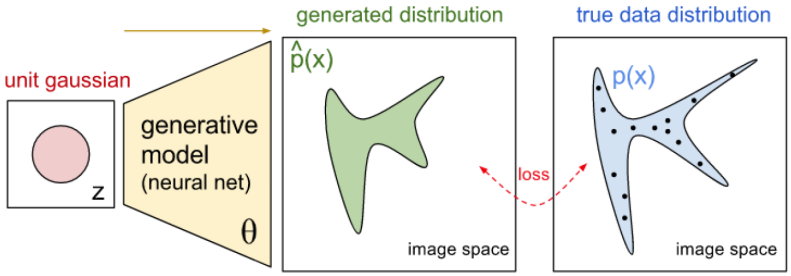
- A one-dimensional example:



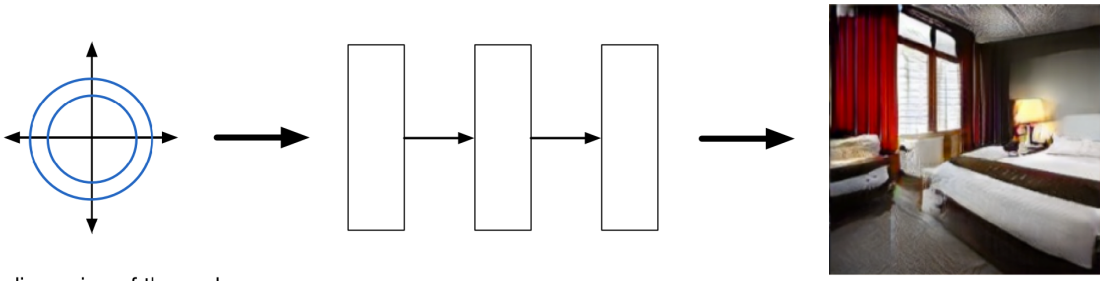
- In one dimension, this can be done elegantly using a technique called **inverse transform sampling**

Implicit Generative Models

- A high-dimensional example:



- Generate images:



Each dimension of the code vector is sampled independently from a simple distribution, e.g. Gaussian or uniform.

This is fed to a (deterministic) generator network.

The network outputs an image.

Outline

1 Background

2 Generative Adversarial Networks (GANs)

3 Diffusion Models

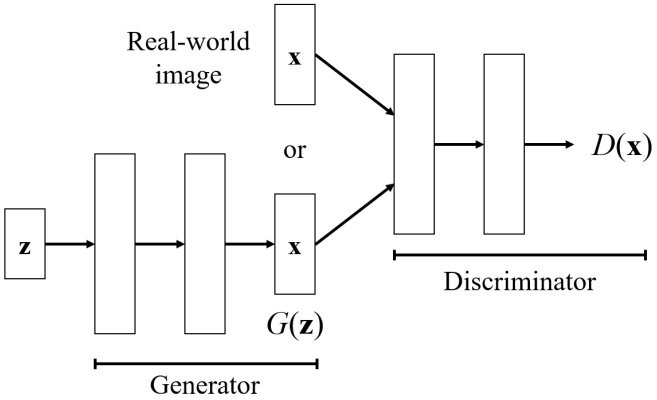
Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are implicit generative models

- Generative
  - Learn a generative model
- Adversarial
  - Learn the generative model in an adversarial setting
- Networks
  - Parameterize the generative model by a deep neural network

Generative Adversarial Networks (GANs)

- The idea of GANs: Train two different networks
  - The *generator network* tries to produce realistic samples
  - The *discriminator network* tries to figure out whether an image comes from the training set or the generator network
- Competition: The generator tries to fool the discriminator, while the discriminator tries to catch the images produced by the generator



Loss Function

- The discriminator D predicts the probability of the input image being realistic
- Loss function for the discriminator: Cross-entropy loss for the task of classifying between training and generated images

$$\ell_D = \mathbb{E}_{\mathbf{x}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

  - A special case of Kullback-Leibler divergence
- Loss function for the generator: The opposite of the discriminator's loss function

$$\ell_G = -\ell_D = \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

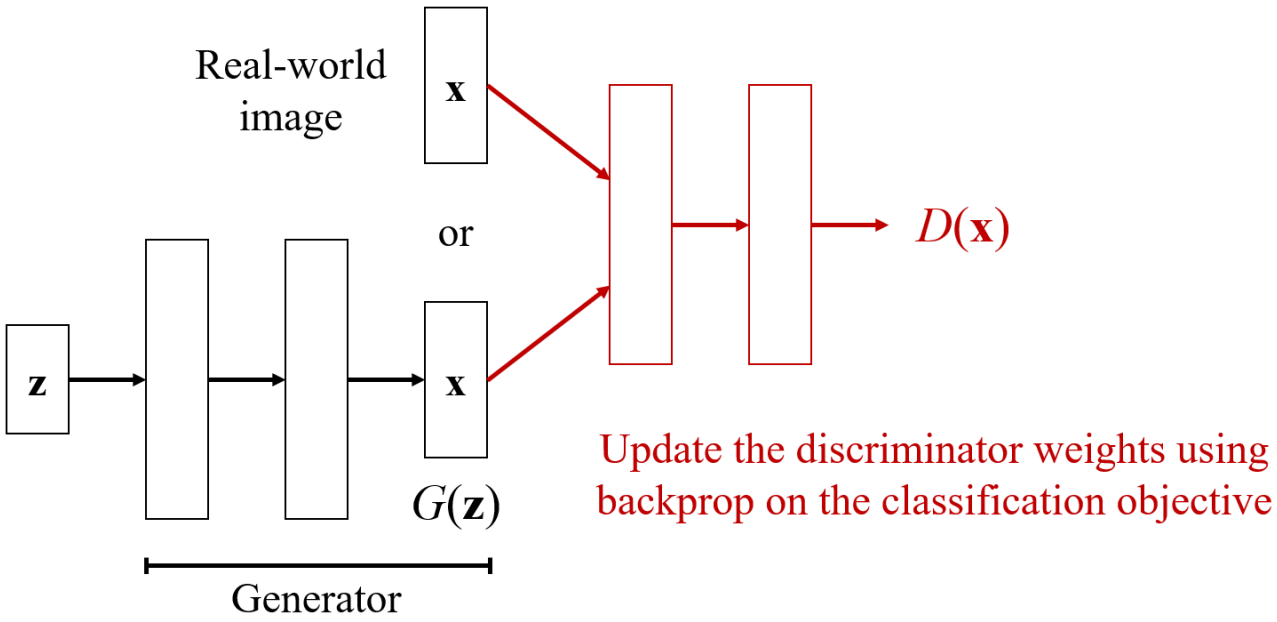
- This is called the *minimax formulation*, since the generator and the discriminator are playing a minimax game:

$$\max_G \min_D \ell_D$$

Training Procedure

Alternate between training the discriminator and the generator:

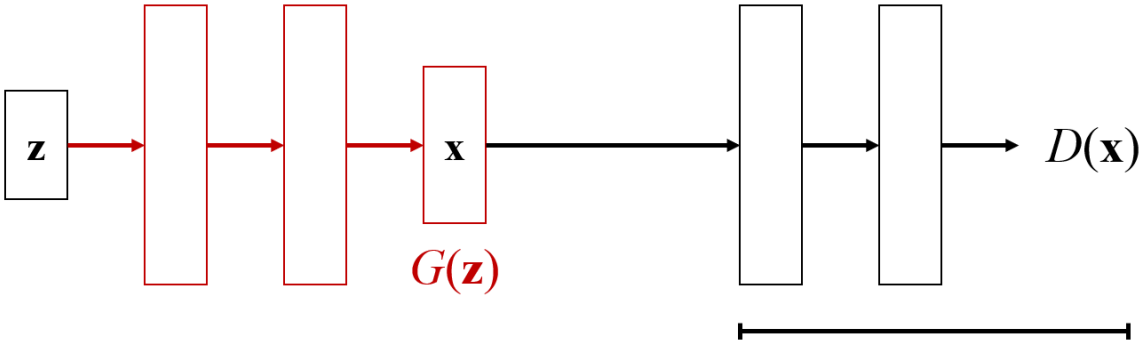
- Update the discriminator



Training Procedure

Alternate between training the discriminator and the generator

- Update the generator



Update the generator weights using backprop

Backprop the derivatives but do not modify the discriminator weights

GAN Samples

- Celebrities

Tero Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation," *International Conference on Learning Representations*, 2018.

GAN Samples

- Bedrooms

Tero Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation," *International Conference on Learning Representations*, 2018.

GAN Samples

- Objects

Tero Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation," *International Conference on Learning Representations*, 2018.

Limitations of GANs

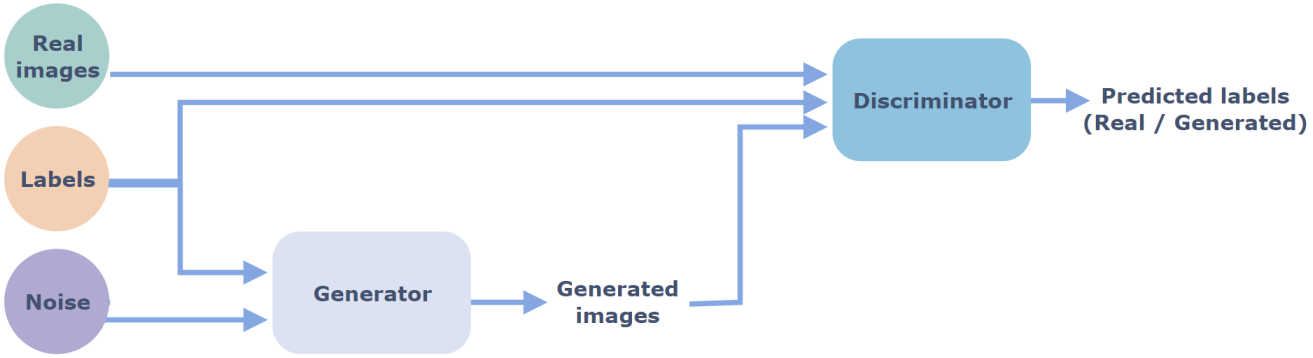
- Mode collapse
  - Generated samples have limited variations, often converging to a small subset of the input distribution
- Training instability
  - Training is sensitive to hyperparameter settings, including the learning rate, network architecture, and loss function
- Lack of objective metrics for evaluation
  - Traditional metrics like log-likelihood are challenging to apply
  - Assessing the realism and diversity of generated samples often relies on subjective human judgment

Improvements of GANs

- Since its inception in 2014, there have been hundreds of papers, improving the architectures, training methods, and loss functions
- GAN Zoo: `https://github.com/hindupuravinash/the-gan-zoo`

Variants of GANs

- Conditional generation
  - We do not want to generate some random stuff
  - We want to generate objects based on conditions
- How to achieve conditional generation?
  - Incorporate the class label



  - Connect (a subset of) latent variables to the outputs (e.g., InfoGAN)

InfoGAN

- InfoGAN is designed to maximize the *mutual information* between a small subset $\mathbf{c}$ of the latent variables $[\mathbf{z}^T, \mathbf{c}^T]^T$ and the output $G(\mathbf{z}, \mathbf{c})$
  - In other words, the information in the latent codes $\mathbf{c}$ should not be lost in the generation process
- The mutual information can be expressed as the difference between two entropy terms

$$I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X} \mid \mathbf{Y}) = H(\mathbf{Y}) - H(\mathbf{Y} \mid \mathbf{X})$$

  - $I(\mathbf{X}; \mathbf{Y})$ is the reduction of uncertainty in $\mathbf{X}$ when $\mathbf{Y}$ is observed

Xi Chen et al., "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," *Advances in Neural Information Processing Systems*, 2016.

InfoGAN

- In the original GAN, the minimax game is

$$\max_G \min_D \ell_D = \mathbb{E}_{\mathbf{x}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

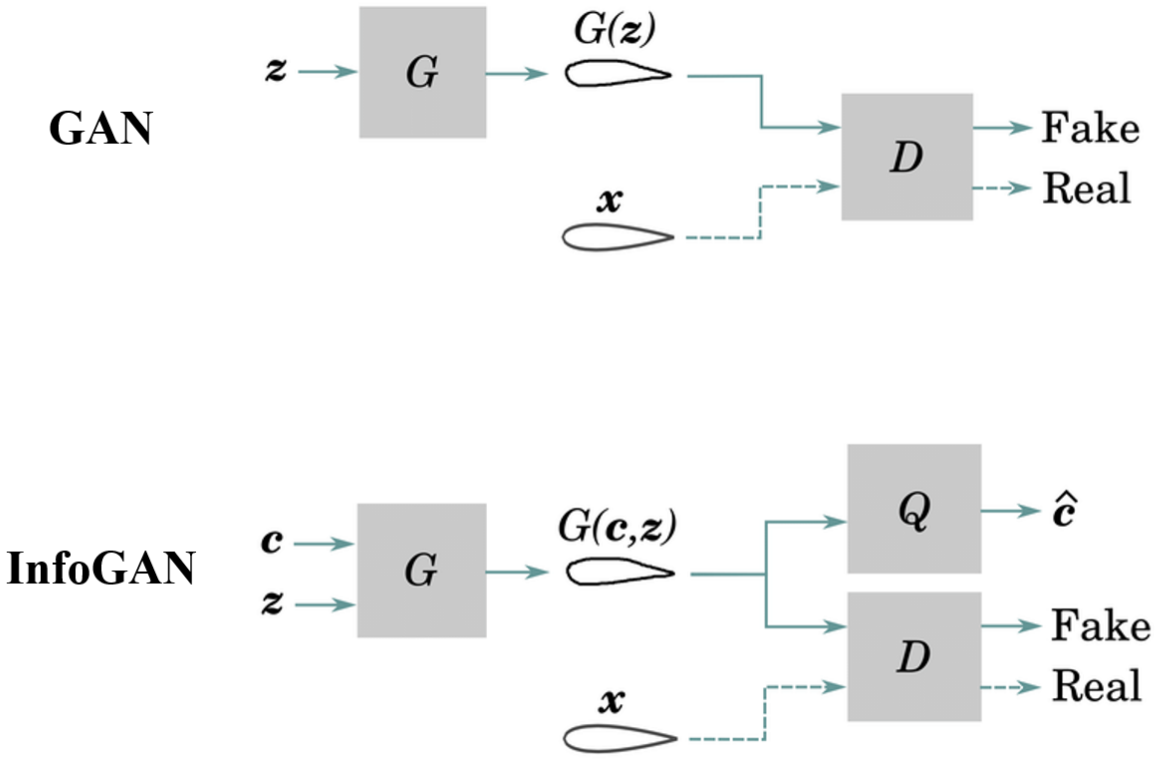
- In InfoGAN, an information-regularized minimax game is solved

$$\max_G \min_D \ell_D + \lambda I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$$

- In practice, the mutual information term $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$ is hard to maximize directly as it requires access to the posterior $P(\mathbf{c}|\hat{\mathbf{x}})$
- But a lower bound can be obtained by defining an auxiliary distribution $Q(\mathbf{c}|\hat{\mathbf{x}})$ to approximate $P(\mathbf{c}|\hat{\mathbf{x}})$
- The auxiliary distribution Q is parameterized as a neural network

InfoGAN

- Comparison between GAN and InfoGAN



InfoGAN

- InfoGAN successfully disentangles writing styles from digit shapes on MNIST

(a) Varying c_1 on InfoGAN (Digit type)

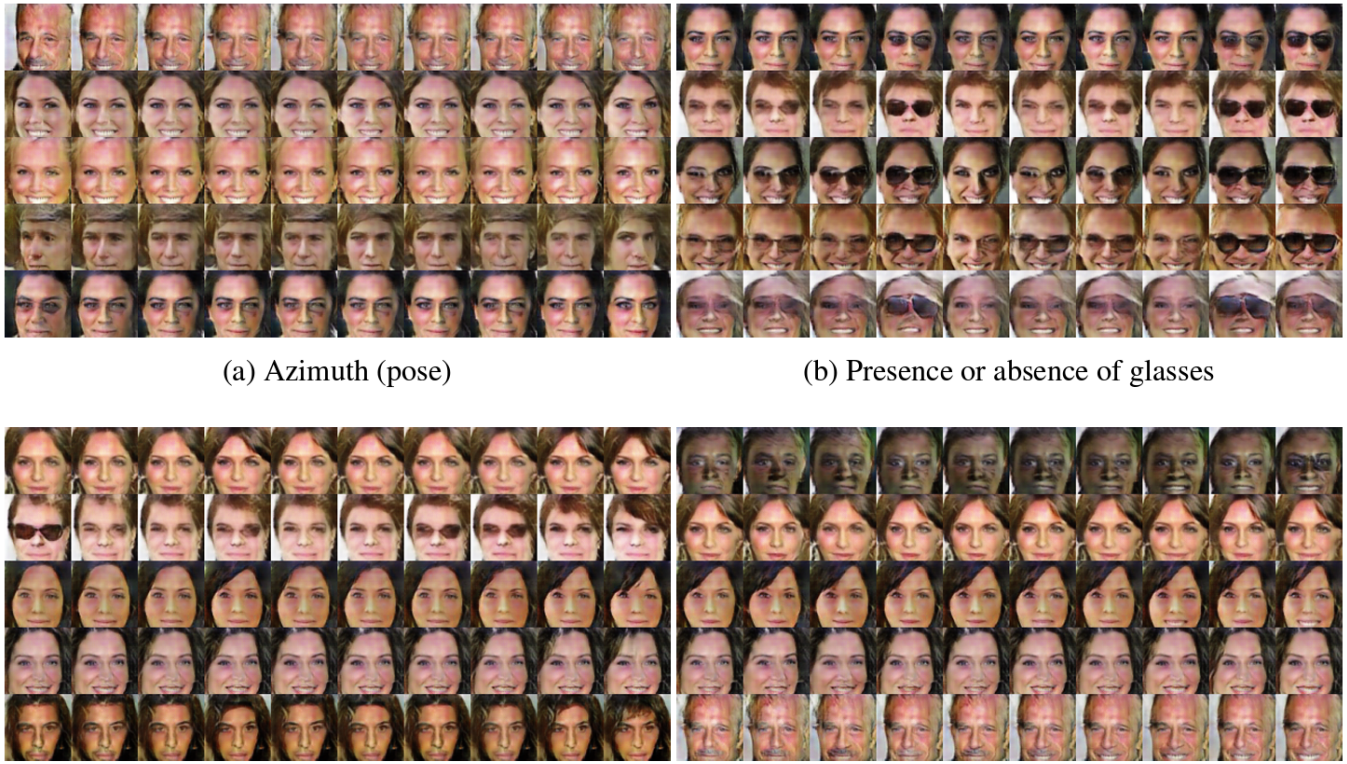
(b) Varying c_1 on regular GAN (No clear meaning)

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

InfoGAN

- InfoGAN successfully disentangles the visual concepts on the CelebA face dataset

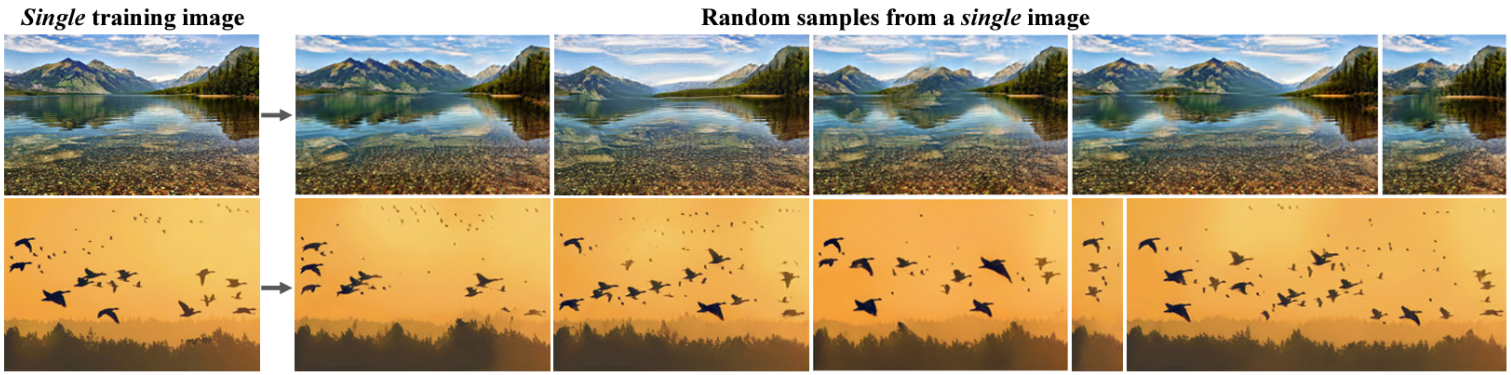


(a) Azimuth (pose) (b) Presence or absence of glasses

(c) Hair style (d) Emotion

SinGAN

- SinGAN: Train a GAN using a single image
  - An unconditional generative model
  - Learn internal statistics of patches within a single image
- Example application: Image retargeting

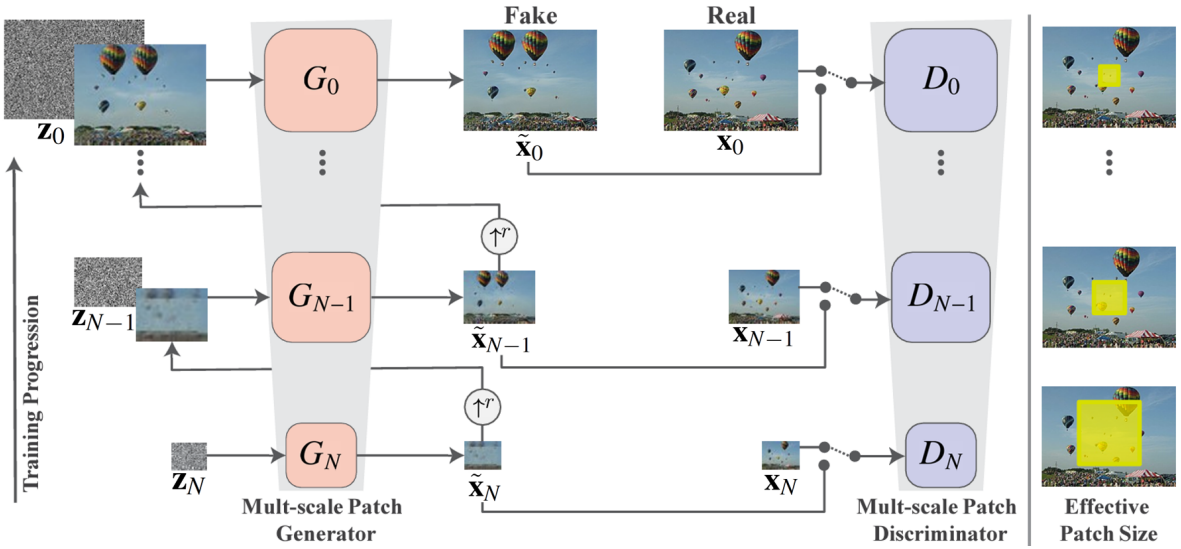


Tamar R. Shaham et al., "SinGAN: Learning a Generative Model from a Single Natural Image," *International Conference on Computer Vision*, 2019.

SinGAN

- SinGAN architecture consists of a multi-scale pipeline
  - A generator and a discriminator are present in each scale
  - Learn representations at different scales
  - Is trained in a coarse-to-fine fashion
    - Why?

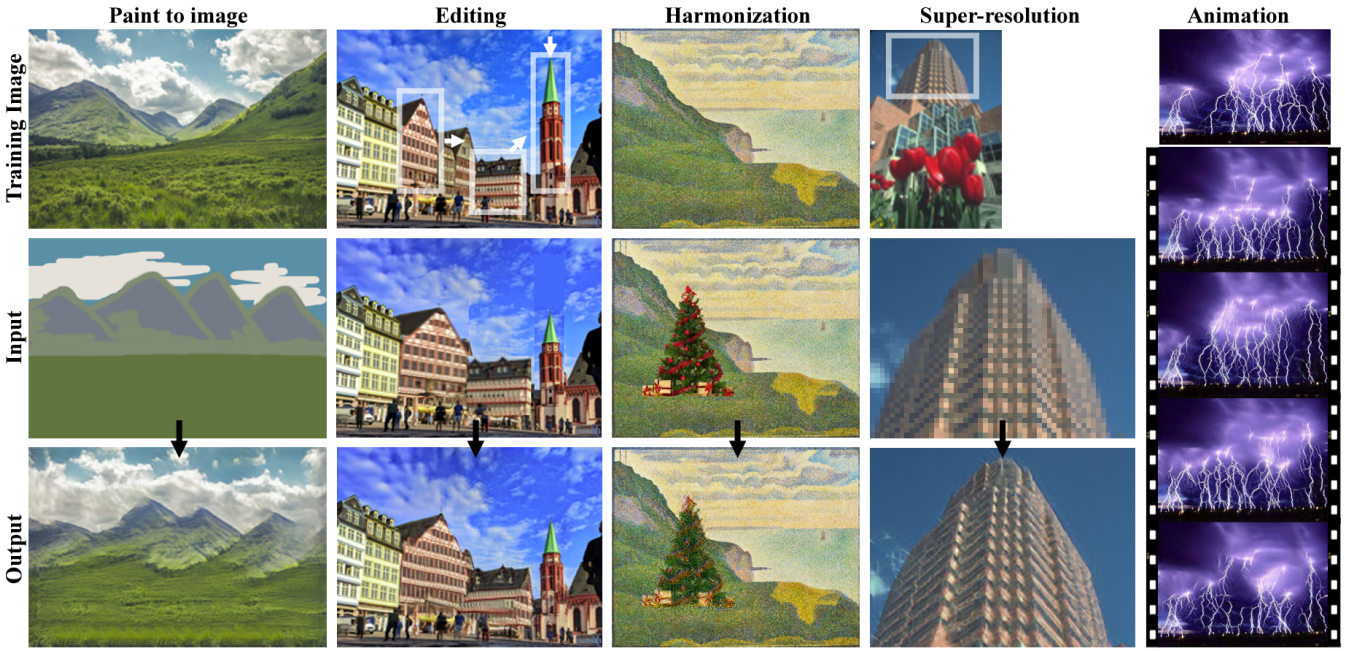
SinGAN



- Procedure
  - The input to the k-th generator is the concatenation of the *upsampled* output of the $k+1$-th generator and the random noise
    - The N-th generator is fed with the random noise only
  - The original image is downsampled to the respective sizes and fed into the discriminator along with the generator's output
    - Patches from (downsampled) original images as a training dataset

SinGAN

- In addition to unconditional generation, SinGAN shows impressive results on various image manipulation tasks



Outline

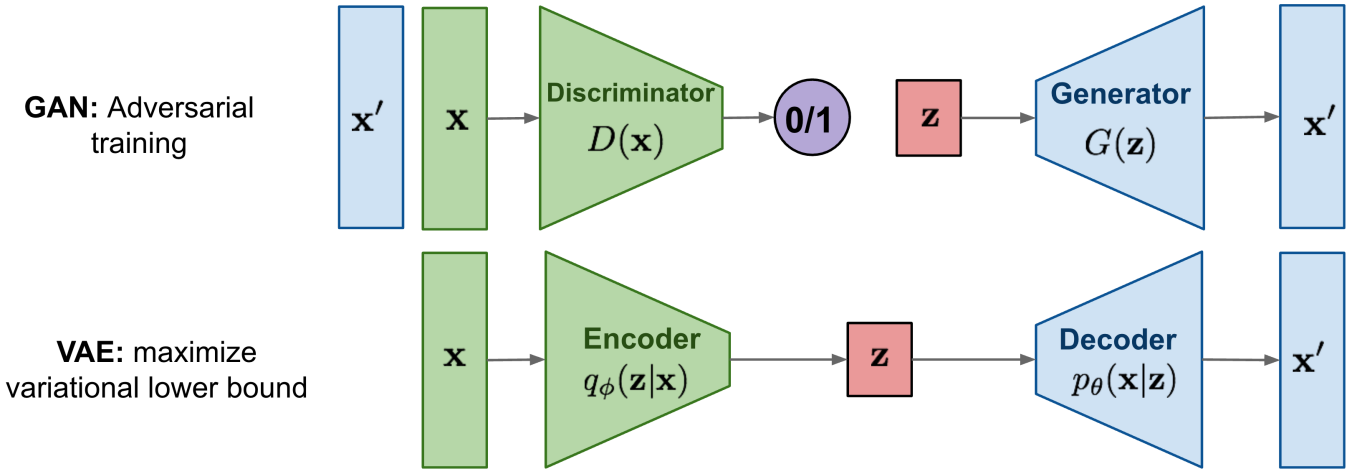
1 Background

2 Generative Adversarial Networks (GANs)

3 Diffusion Models

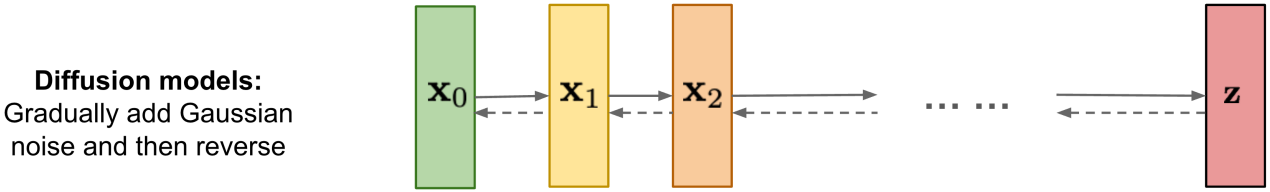
Diffusion Models

What are diffusion models?



- Like GANs or VAEs, diffusion models are generative models
  - Generate better data than VAEs
  - Easier to train than GANs

Diffusion Models

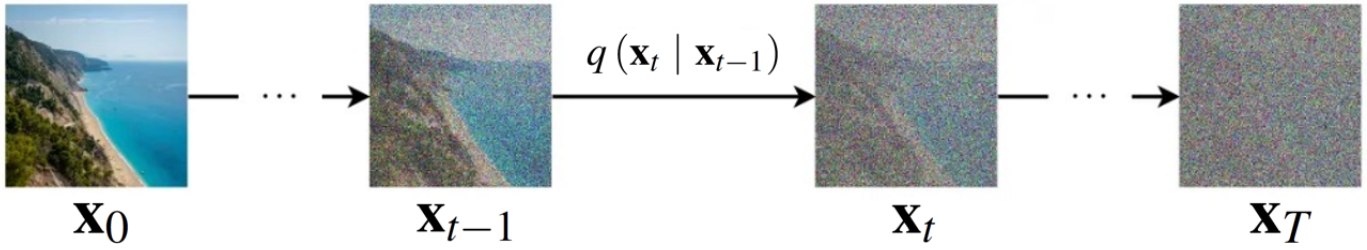


- Diffusion models work by
  - Destroying training data through the successive addition of Gaussian noise
  - Learning to recover the data by reversing this noising process
- After training, data can be generated by passing randomly sampled noise through the learned denoising process

Jonathan Ho et al., "Denoising Diffusion Probabilistic Models," *Advances in Neural Information Processing Systems*, 2020.

Forward Diffusion Process

- Forward diffusion process (gradually adding noise to input)



Distribution of the noised images — Output — Mean — Variance

$$\boxed{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} = \mathcal{N}\left(\boxed{\mathbf{x}_t}; \boxed{\sqrt{1-\beta_t}\mathbf{x}_{t-1}}, \boxed{\beta_t \mathbf{I}}\right)$$

- Notation:

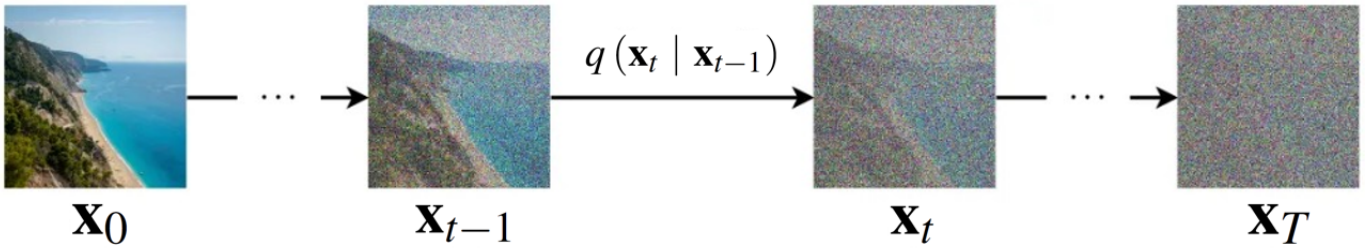
  t: Discrete timestep in $\{1, \ldots, T\}$

  $\mathbf{x}_0$: Data sampled from the training distribution $q(\mathbf{x})$

  $\{\beta_t\}$: Variance scheduling, where $0 \leq \beta_t \leq 1$ and $\beta_{t-1} < \beta_t$

  $\mathbf{I}$: Identity matrix

Stochastic Transformation



- The reparameterization trick:
  - If $\mathbf{x} \sim \mathcal{N}\left(\boldsymbol{\mu}, \sigma^2 \mathbf{I}\right)$
    - $\mathbf{x} = \boldsymbol{\mu} + \sigma \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
- Back to the noising process:
  $q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$
  - $\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Stochastic Transformation

- The noising process can be speeded up by computing $\mathbf{x}_t$ in one step:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$$

  where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, and $\alpha_i = 1 - \beta_i$

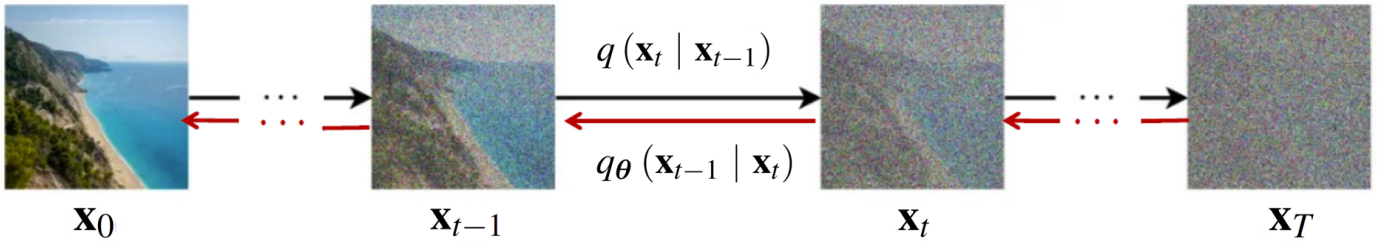
- In the limit, $\mathbf{x}_\infty$ is a random vector from an isotropic Gaussian:

$$\lim_{t \to \infty} \mathbf{x}_t = \sqrt{\bar{\alpha}_\infty}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_\infty}\boldsymbol{\epsilon} = \boldsymbol{\epsilon}$$

  since $\bar{\alpha}_\infty \to 0$

Reverse Denoising Process

- Reverse denoising process (learns to generate data by denoising)



- Forward factorization:
  $q(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T) = q(\mathbf{x}_0) \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$

- Reverse factorization:
  $q_\theta(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T) = \prod_{t=1}^{T} q_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) q(\mathbf{x}_T)$
  - Since the joint distribution is Gaussian, $q_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ can also be modeled by a Gaussian
    - $q_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}\right)$
    - σ_t^2 is a pre-determined time dependent constant (e.g., $\sigma_t^2 = \beta_t$)

Reverse Conditional Gaussian

- The reverse conditional Gaussian $q_{\boldsymbol{\theta}}\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right), \sigma_t^2 \mathbf{I}\right)$ needs to be learned
- Ho, Jain, and Abbeel (2020) derived the following formula for $\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right)$

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right),$$

  - $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is unknown in the reverse process
  - $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t$
- $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right)$ is trained to esitmate $\boldsymbol{\epsilon}_t$ by minimizing

$$\ell(\boldsymbol{\theta}) = \left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right)\right\|_2^2$$

Training Algorithm

For training diffusion models, repeat the following steps until convergence

- Sample data $\mathbf{x}_0$ from the training distribution $q(\mathbf{x})$
- Randomly select a timestep $t \sim \texttt{Uniform}(\{1, \ldots, T\})$
- Sample Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Adjust the amount of noise according to the timestep t, and add noise to image $\mathbf{x}_0$

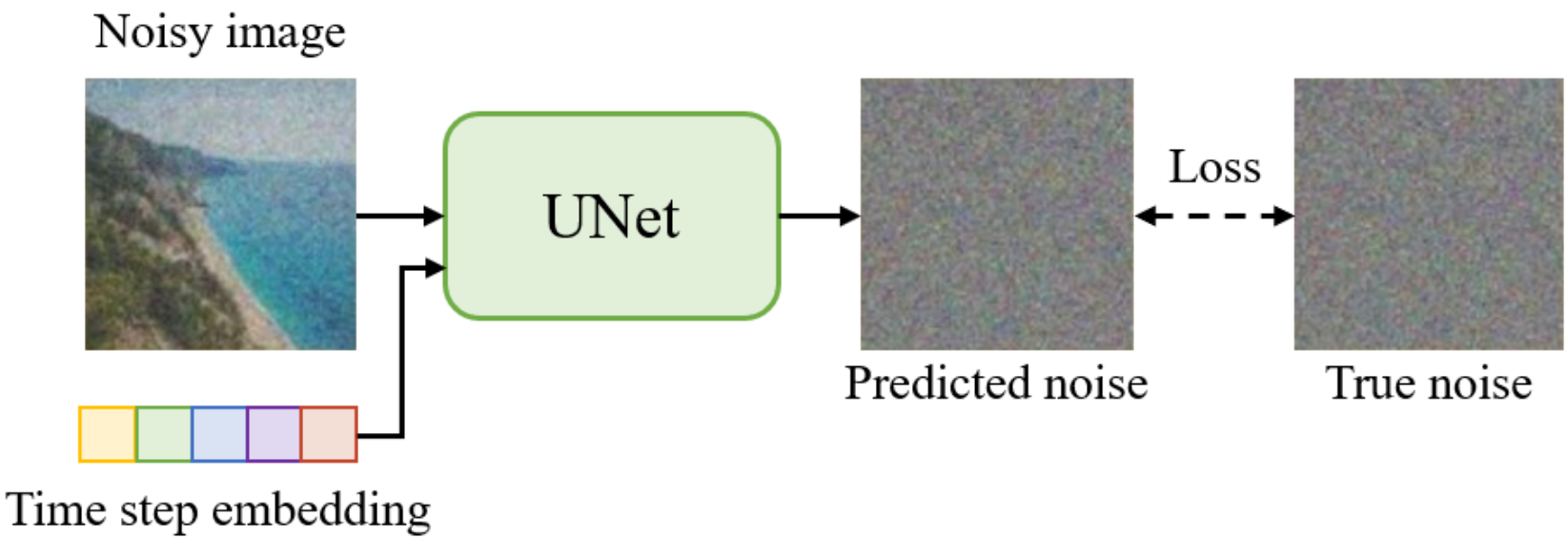
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$$

- Update the parameters $\boldsymbol{\theta}$ of the noise prediction network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \right\|_2^2$$

Training Algorithm

Training pipeline:

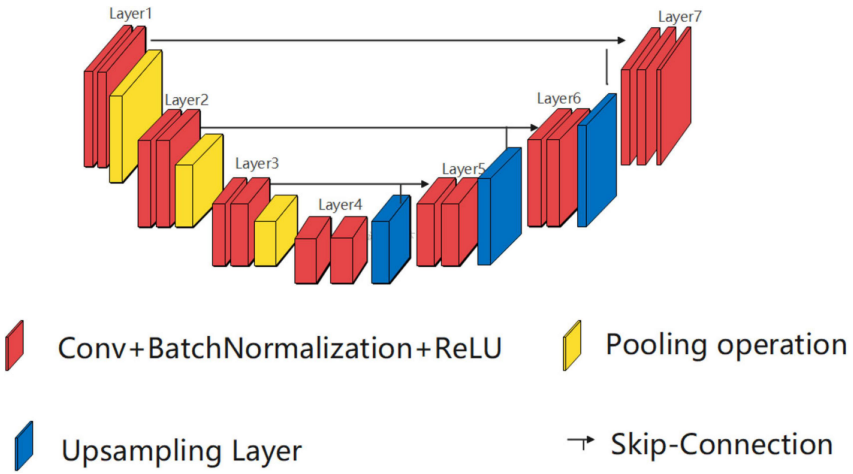


- UNet is a special type of fully convolutional network (FCN)

UNet

A U-shaped architecture consists of a specific encoder-decoder scheme

- The encoder reduces the spatial dimensions in every layer and increases the channels
- The decoder increases the spatial dimensions while reducing the channels



Data Generation Algorithm

For data generation

- Sample a Gaussian noise $\mathbf{x}_T$

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

- Iteratively denoise the image

  For $t = T, \ldots, 1$ do

  - $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\boldsymbol{\epsilon} = \mathbf{0}$
  - $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}} \left(\mathbf{x}_t, t \right) \right) + \sigma_t \boldsymbol{\epsilon}$

- Return $\mathbf{x}_0$

Conditional Diffusion Models

- Include the condition $\mathbf{c}$ as part of the input to the reverse process

$$q_{\boldsymbol{\theta}}\left(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T | \mathbf{c}\right) = q\left(\mathbf{x}_T\right) \prod_{t=1}^{T} q_{\boldsymbol{\theta}}\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}\right)$$

$$q_{\boldsymbol{\theta}}\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t, \mathbf{c}\right), \sigma_{(t,\mathbf{c})}^2 \mathbf{I}\right)$$

- Incorporate conditions into UNet
  - Class label conditioning
    - Encode the class label as a vector embedding
    - Incorporate the timestep and class embedding into each residual block after a group normalization operation
  - Text conditioning
    - Single vector embedding $\rightarrow$ Spatial addition or adaptive group normalization
    - A sequence of vector embeddings $\rightarrow$ Cross-attention

Text to Image Generation

Case Study: Imagen

Text to Image Generation

Imagen: Text-to-image diffusion models (by Google)

- Input: Texts
- Output: Images $(1,024 \times 1,024)$
- An unprecedented degree of photorealism
  - State-of-the art Fréchet Inception Distance (FID) scores & human ratings
- A deep level of language understanding

Chitwan Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," *Advances in Neural Information Processing Systems*, 2022. `https://imagen.research.google/`

Text to Image Generation

Imagen

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

Text to Image Generation

Imagen

A dragon fruit wearing karate belt in the snow.

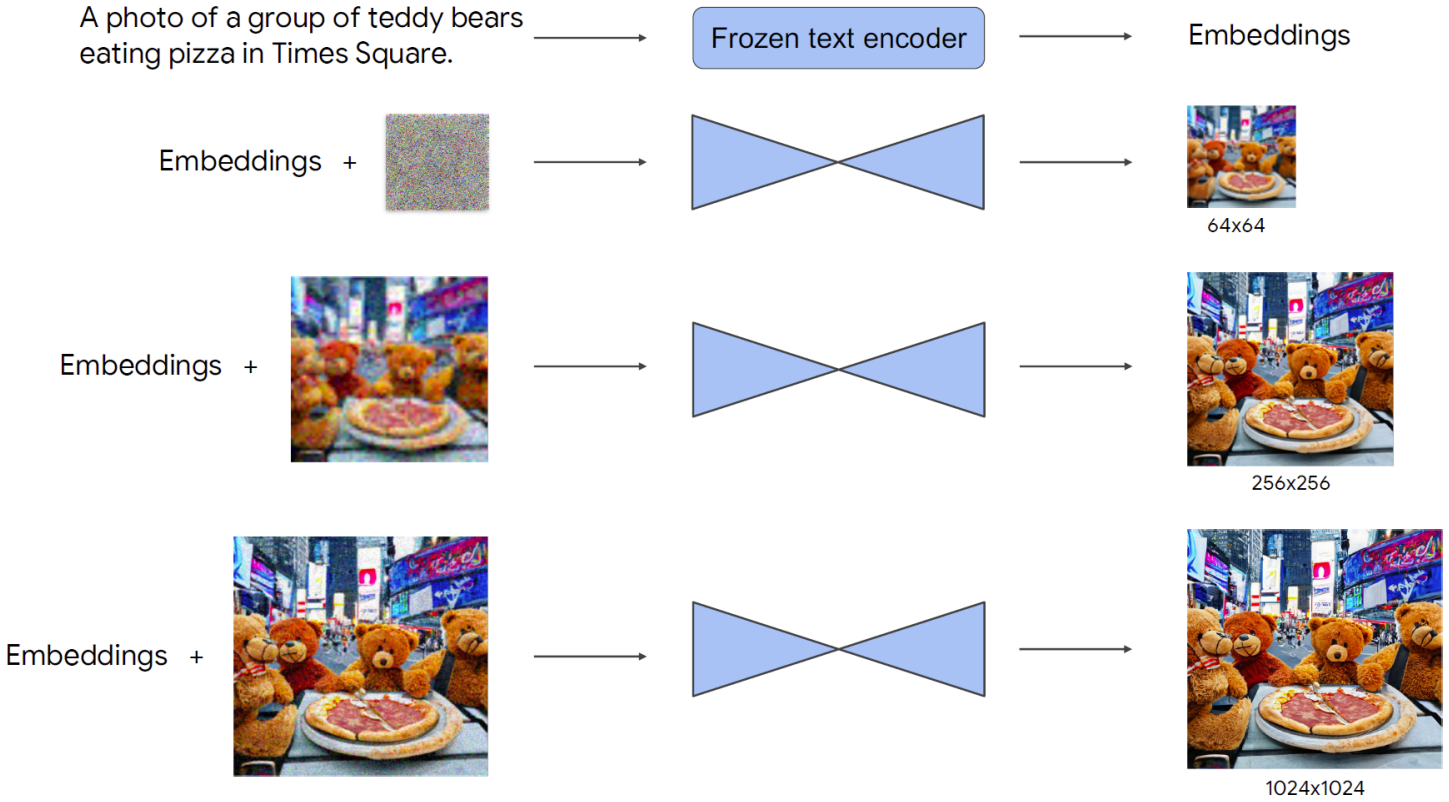
Text to Image Generation

What goes to Imagen?

- Data
  - Image-text pairs
  - LAION-400-Million open dataset
  - Internal dataset (500 Million images)
- Model
  - Diffusion models
  - Cascaded super-resolution
  - Frozen text encoders
- Sampler
  - Classifier guidance
  - Classifier-free guidance

Imagen

Cascaded super-resolution generation pipeline



Imagen

Sampler technique: Classifier guidance

- Assume pairs of data $(\mathbf{x}, \mathbf{c})$
- A classifier guidance diffusion model consists of
  - A trained unconditional diffusion model
  - A trained classifier model on *noisy* data $\mathbf{x}_t$
- During sampling, at each denoising step, modify the objective function to

$$\nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_t \mid \mathbf{c}) \approx \nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x}_t) + \lambda \nabla_{\mathbf{x}} \log q_{\boldsymbol{\phi}}(\mathbf{c} \mid \mathbf{x}_t)$$

  - $\nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x}_t)$ is from the unconditional diffusion model
  - $\nabla_{\mathbf{x}} \log q_{\boldsymbol{\phi}}(\mathbf{c} \mid \mathbf{x}_t)$ is from the classifier model
- Upweight samples that the classifier assigns high probabilities with $\rightarrow$ Better alignment with $\mathbf{c}$

Imagen

Sampler technique: Classifier-free guidance

- Classifier guidance
  - Need to train an additional classifier; increase model complexity
- Classifier-free guidance
  - Assume there are two diffusion models, one conditional model and one unconditional model
- By the Bayes' rule, we can define an implicit classifier
$$q_{\boldsymbol{\theta}}(\mathbf{c} \mid \mathbf{x}_t) \propto q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{c}) / q_{\boldsymbol{\theta}}(\mathbf{x}_t)$$
- The modified objective function during sampling then becomes
$$\nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{c}) \approx \lambda \nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{c}) - (\lambda - 1) \nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x}_t)$$
- The two models can share weights, with the unconditional model taking a null class label $\mathbf{c}$