AI61201: Visual Computing With AI/ML

Module 9: Generative Models for Visual Data

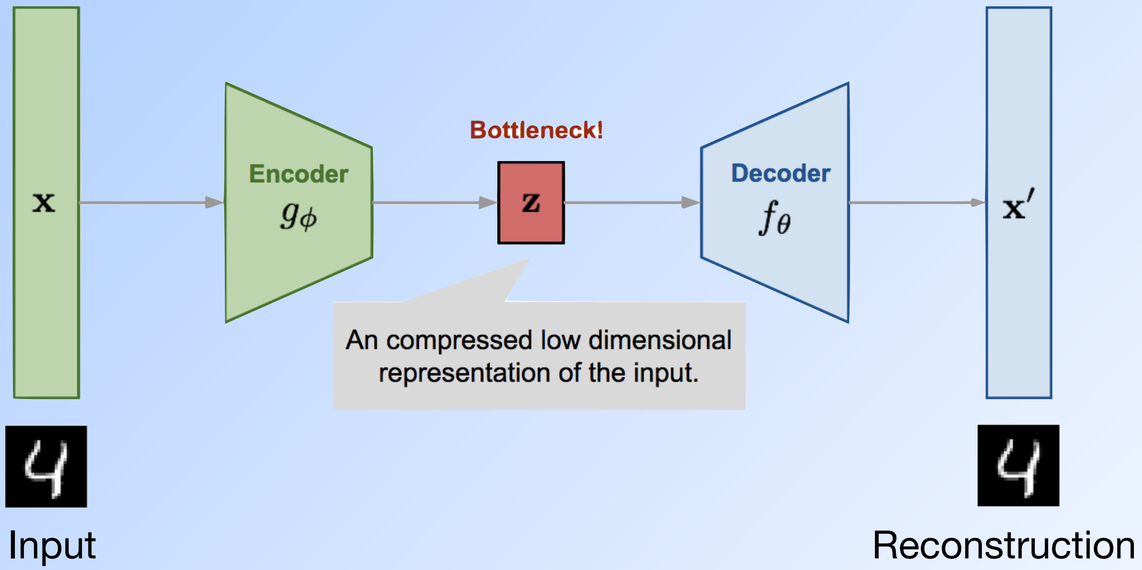
Dr. Somdyuti Paul

Discriminative Models Vs. Generative Models

- The goal of discriminative models is to extract knowledge or information such as class labels, object locations, segmentation maps, etc. from the given visual data.
- The underlying task of discriminative models is to learn the conditional distribution $p(y \mid \boldsymbol{x})$, where y is the prediction, and $\boldsymbol{x}$ is the given data (images in this context).
- Generative models seek to generate new data that resembles the training data by learning the conditional distribution $p(\boldsymbol{x} \mid y)$.
- Various classes of Generative models have been developed for visual data, such as:
  - Autoencoders
  - Generative Adversarial Networks
  - Diffusion Models

Autoencoders

- Autoencoders are neural networks that are trained to reconstruct a given input from its compressed representation in a self-supervised manner.
- The components of a vanilla autoencoder are the following
  - Encoder: the encoder is a set of layers that reduce the dimension of its input, into a lower dimensional space called the latent space or bottleneck.
  - Decoder: the goal of the decoder is to reconstruct the original input from the latent representation.
- Autoencoders are trained using a reconstruction loss function (such as MSE) that measures how well the decoder can reconstruct the original input from the latent representation.
- Autoencoders achieve compression by discarding redundant information from the input while retaining its most important features.



Autoencoder Model

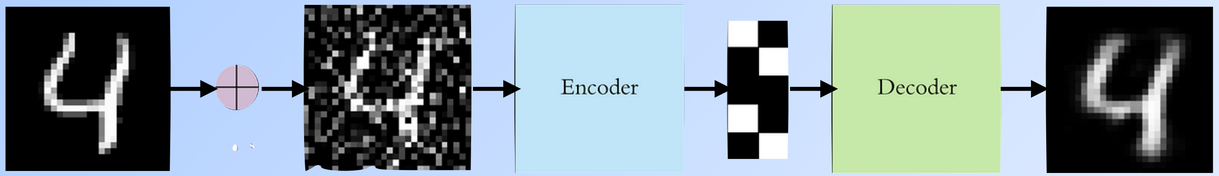
Encoder function: $z = g_\phi(\boldsymbol{x})$

Decoder function) : $\boldsymbol{x}' = f_\theta(z)$

Objective: $\min_{\phi,\theta} \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2$

Autoencoders

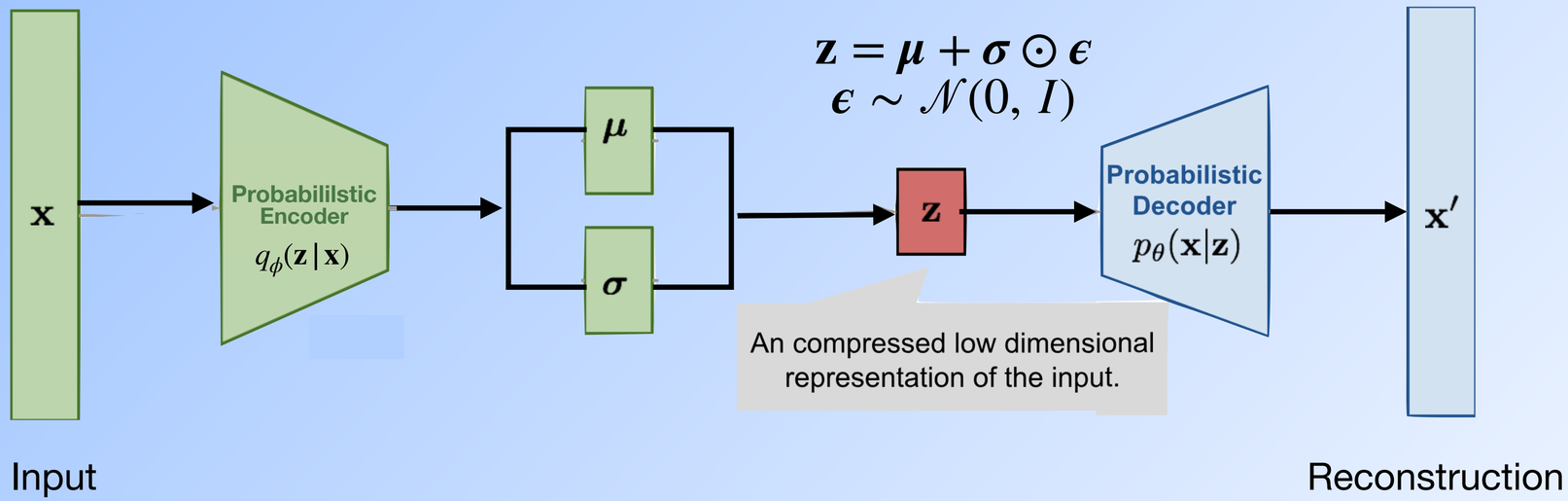
- Autoencoders can also be trained for effectively denoising input images by training on corrupted or noisy versions of the training images.
- The reconstruction loss in case of denoising auto encoders is computed between the original clean inputs and the reconstructed outputs.
- Other applications include unsupervised feature extraction, inpainting, and anomaly detection.



Denoising Using Autoencoders

Variational Autoencoders

- We could generate new image samples (i.e. those that are not present) in the training set if we could effectively sample from the distribution of the latent space.
- Variational autoencoders (VAEs) incorporate a probabilistic framework to model the latent space distribution of training images (Kingma and Welling, 2013).
- Instead of mapping individual inputs to a low dimensional as in the case of vanilla autoencoders, VAEs learn to map input data to a distribution over the latent variable.



Variational Autoencoders

- In VAE, instead of predicting a single latent variable z for each input $\boldsymbol{x}$, a distribution $q_\phi(z \mid \boldsymbol{x})$ is predicted.
- The encoder of VAE outputs the parameters of a distribution, which is typically taken to be a Gaussian with mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})$ and standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})$, where $\boldsymbol{\phi}$ are the parameters of the encoder.
- From the predicted distribution, the latent variable z is sampled as $z = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. This reparameterization trick allows the gradients to be computed for the sampling operation, which is inherently non-differentiable.
- The decoder takes the latent variable z sampled from the predicted distribution, and generates a distribution over the input space given the latent, i.e. $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid z)$, where $\boldsymbol{\theta}$ are the parameters of the decoder.

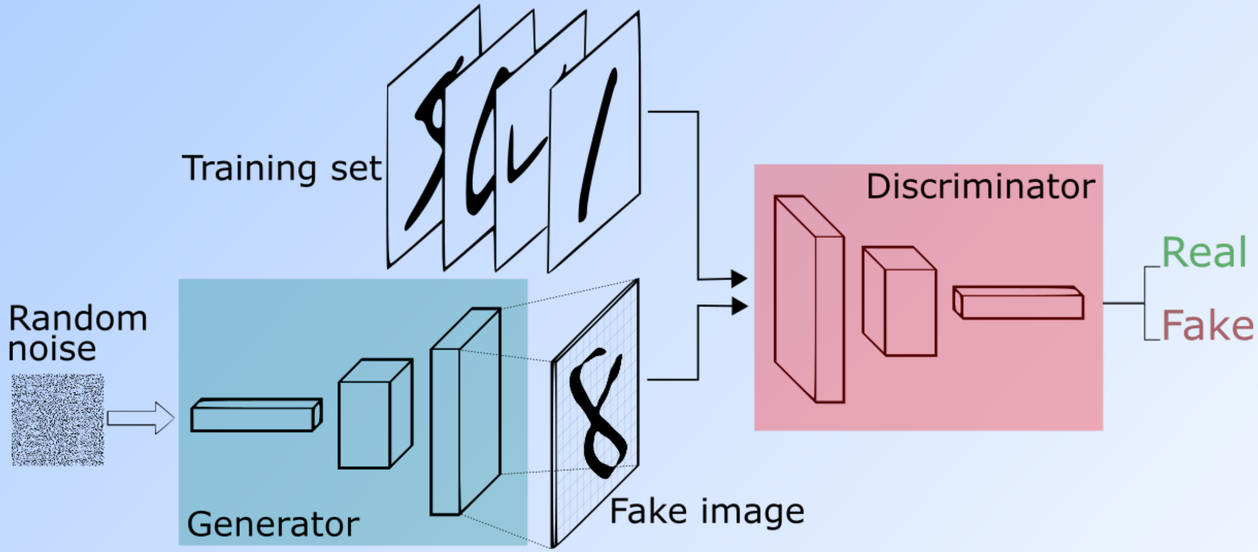
Variational Autoencoders

- The VAE is trained with a loss that has two components:
  - Reconstruction loss: the expected negative log likelihood of the actual data $\boldsymbol{x}$, given the latent variable z, i.e.
    $$\mathscr{L}_{recon} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log p_\theta(\mathbf{x}\,|\,\mathbf{z})\right]$$
  - KL Divergence loss
    $$\mathscr{L}_{\text{KL}} = D_{\text{KL}}\left(q_\phi(z\,|\,\mathbf{x})\,||\,p(z)\right) = \mathbb{E}_{q(z|\boldsymbol{x})}\left[\log q(z\,|\,\boldsymbol{x}) - \log p(z)\right]$$
- The overall loss of VAE is given by
  $$\mathscr{L}(\theta,\phi) = -\,\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}\,|\,\mathbf{z})\right] + D_{\text{KL}}\left(q_\phi(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z})\right)$$

Images Synthesized by VAE using the CelebA dataset for training

Generative Adversarial Networks

- Generative Adversarial Networks (GANs) consist of a class of models that generate data through a competitive interaction between two networks in a game like setting (Goodfellow et al., 2014).
  - Generator - seeks to create visual data that is as realistic as possible from random noise inputs.
  - Discriminator - seeks to distinguish visual data created by the generator from real visual data.
- This competitive learning causes the generator to gradually get better in generating realistic data, and the discriminator to get better in generating fake data.
- At equilibrium, the generator produces data that matches the distribution of real data (i.e. highly realistic), and the discriminator is unable to distinguish between real and fake data.



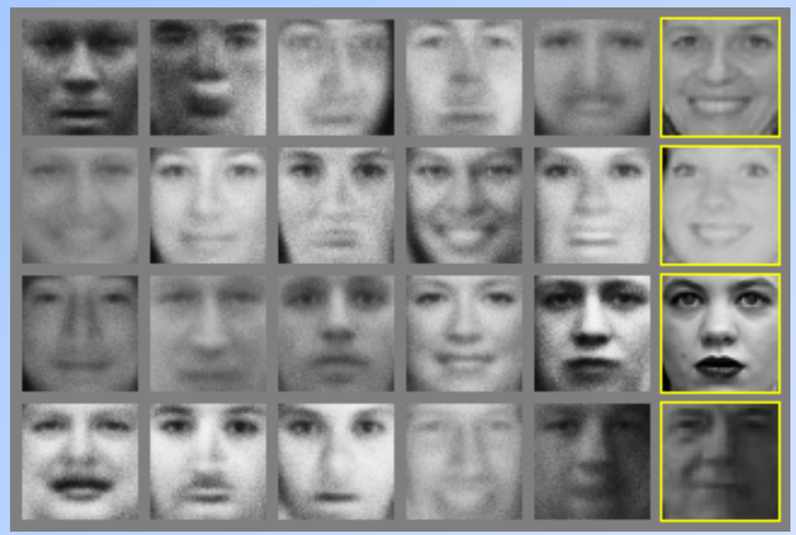
GAN components

Generative Adversarial Networks

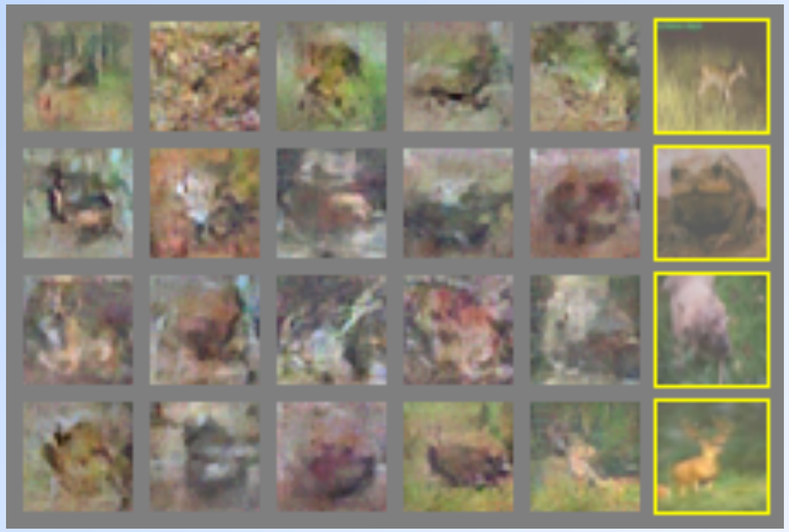
- The objective of GANs is as follows:
  $V(D, G) = \mathbb{E}_{x \sim p_X(x)} \left[\log D(x)\right] + \mathbb{E}_{z \sim p_Z(z)} \left[\log \left(1 - D(G(z))\right)\right]$, where $D(x)$ is the discriminator's prediction that the input sample x is real, and $G(z)$ is the generator's output for a random noise input z.
- The discriminator tries to maximize this objective,
- The generator tries to minimize $\mathbb{E}_{z \sim p_Z(z)} \left[\log \left(1 - D(G(z))\right)\right]$.
- So, the optimization problem for GANs could be stated as $\min_G \max_D V(D, G)$, which is called the minimax loss, and works as a zero-sum game.
- The discriminator and the generator are alternately updated during training.

Generative Adversarial Networks

Generations using MNIST as training data



Generations using TFD as training data



Generations using CIFAR-10 training data

Images Generated by GAN with rightmost column showing closest training image of the neighboring sample

Deep Convolutional GANs

- The quality of GAN outputs were significantly improved by using deep convolutional networks as both the generator and the discriminator (DCGAN), where the discriminator consisted of strided convolutional layers, and the generator consisted of transposed convolutional layers (Radford et al., 2015).
- DCGAN also explored combining input space representations to generate novel visual concepts.

Smiling Woman − Neutral Woman + Neutral Man = Smiling Man

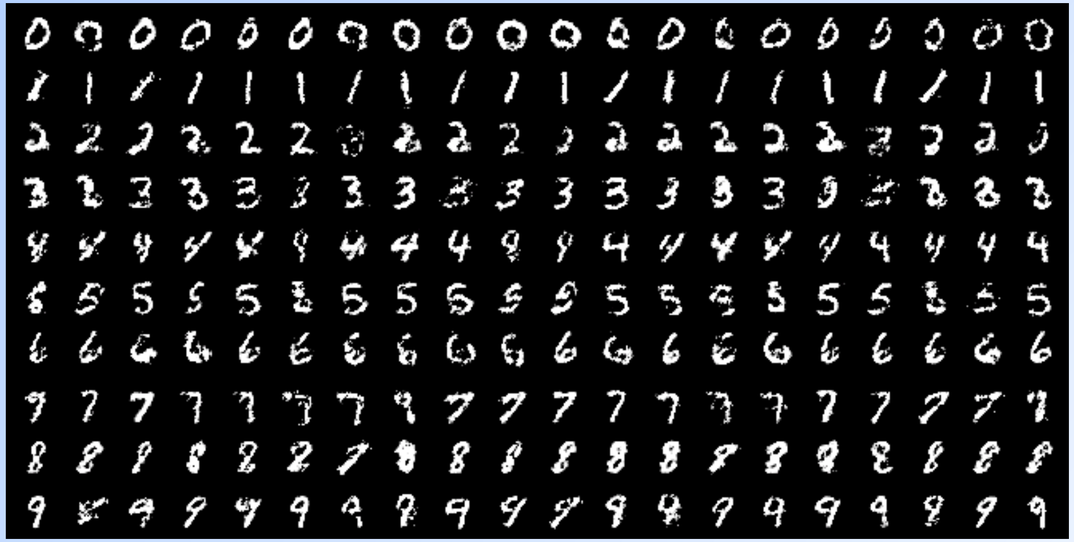
Changing Facial Pose in the input space

Conditional GANs

- Conditional GANs developed a way of controlling or guiding the generation process to generate specific images (such as images belonging to a particular class), by conditioning both the generator and the discriminator on some auxiliary information, such as class labels, text, another image etc. (Mirza & Osindero, 2014)
- In CGANs, the discriminator takes the conditional information y and data sample $\boldsymbol{x}$ as input, while the generator takes random noise $\boldsymbol{z}$ and conditional information y as input
- For cGANs, the objective function of the minimax game was modified as follows:

$$V(D, G) = \mathbb{E}_{\boldsymbol{x}}\left[\log D(\boldsymbol{x} \mid y)\right] + \mathbb{E}_{z}\left[\log\left(1 - D(G(\boldsymbol{z} \mid y))\right)\right]$$

- Class labels were encoded as one-hot vectors for training the generator and discriminator models.

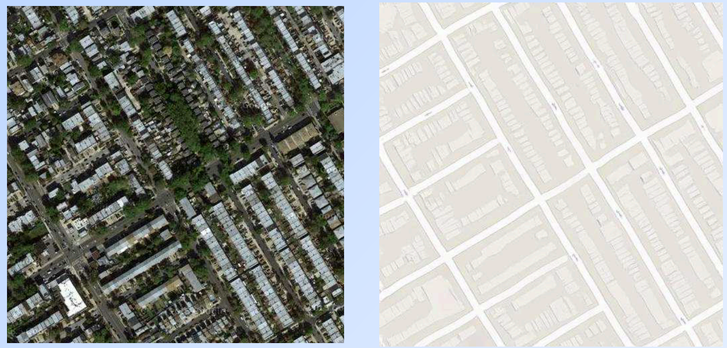


MNIST images generated by CGAN, where each row is conditioned on one label.

Conditional GANs

- Conditional GANs were also shown to be effective for many image-to-image translation tasks by using different renderings of the same underlying structure as conditioning information while training in Pix2Pix (Isola et al., 2017).

Labels to street scene



Aerial to map

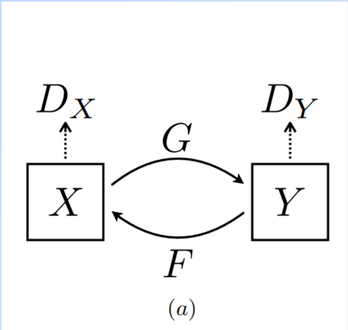
Grayscale to Color

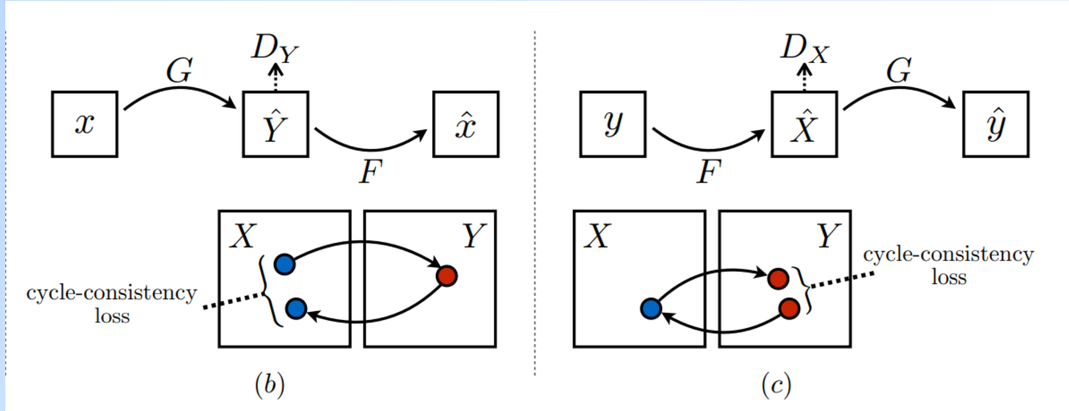
Edges to Photo

Day to Night

Cycle GANs

- Cycle Consistent Generative Adversarial Networks (CycleGAN) was designed for image to image translation for tasks where paired training data is unavailable (Zhu et al., 2017)
- The core idea behind Cycle GANs is to enforce cycle consistency, i.e. after mapping an image from domain X to domain Y, it should be possible to map it back to the domain X to recover the original image.
- CycleGANs accomplish this using two generators and two discriminators.
- A cycle consistency loss is used for training, in addition to the regular adversarial loss.





$$G : X \rightarrow Y$$
$$F : Y \rightarrow X$$

Forward cycle consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$

Backward cycle consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

Cycle GANs

- Adversarial loss for mapping $X \to Y$:
  $L_{GAN}(G, D_A) = \mathbb{E}_{x \sim p_X(x)} \left[\log D_A(x)\right] + \mathbb{E}_{y \sim p_Y(y)} \left[\log(1 - D_A(G(x)))\right]$
- Adversarial loss for mapping $Y \to X$:
  $L_{GAN}(F, D_B) = \mathbb{E}_{y \sim p_Y(y)} \left[\log D_B(y)\right] + \mathbb{E}_{x \sim p_X(x)} \left[\log(1 - D_B(F(y)))\right]$
- Cycle consistency loss:
  $L_{cyc}(G, F) = \mathbb{E}_{x \sim p_X(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_Y(y)} \left[\|G(F(y)) - y\|_1\right]$
- Overall objective: $L_{GAN}(G, D_A) + L_{GAN}(F, D_B) + \lambda L_{cyc}(G, F)$

Cycle GANs

Horses ↔ Zebra

Summer ↔ Winter

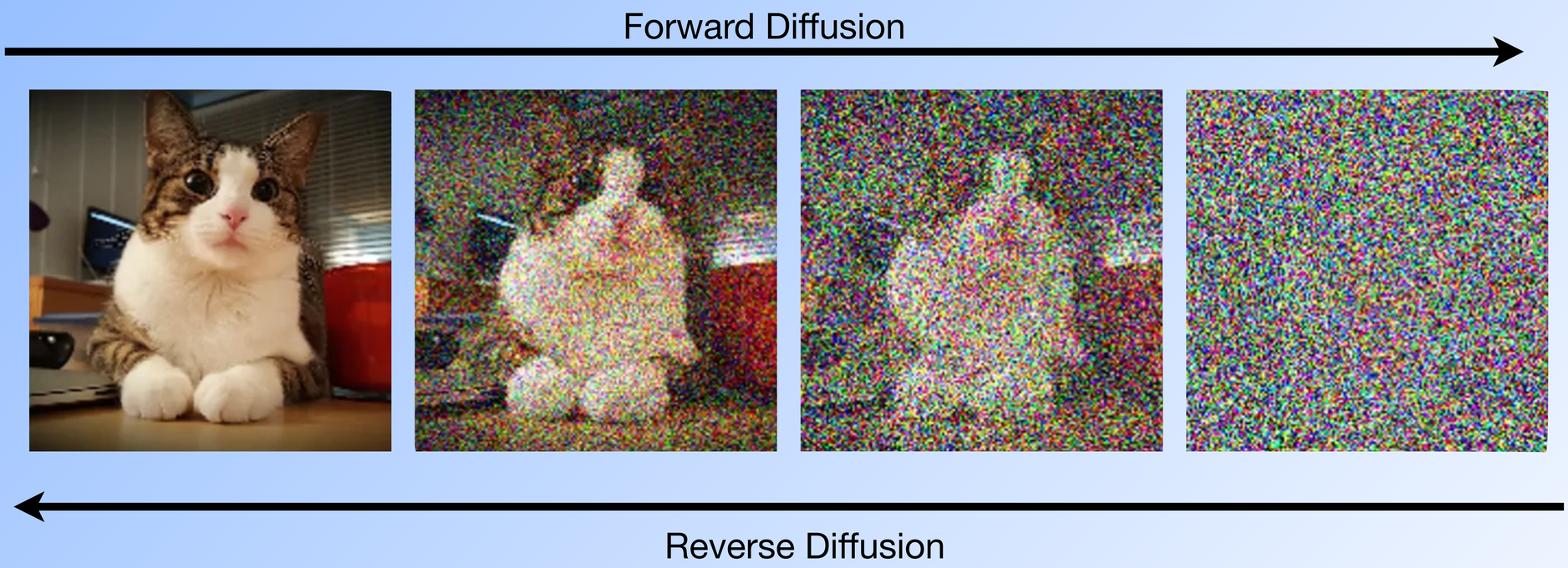
Apples ↔ Oranges

Challenges of Training GANs

- Mode collapse: the generator learns to generate only a subset of the possible outputs.
- Unstable and difficult to train: fail to converge to a stable solution or oscillate between different solutions.
- Vanishing Gradients: when the discriminator becomes too strong, the generator may receive very small gradients during training, causing the updates to be very slow or ineffective.
- Difficult to evaluate model performance.
- Hyperparameter sensitivity.
- High computational cost

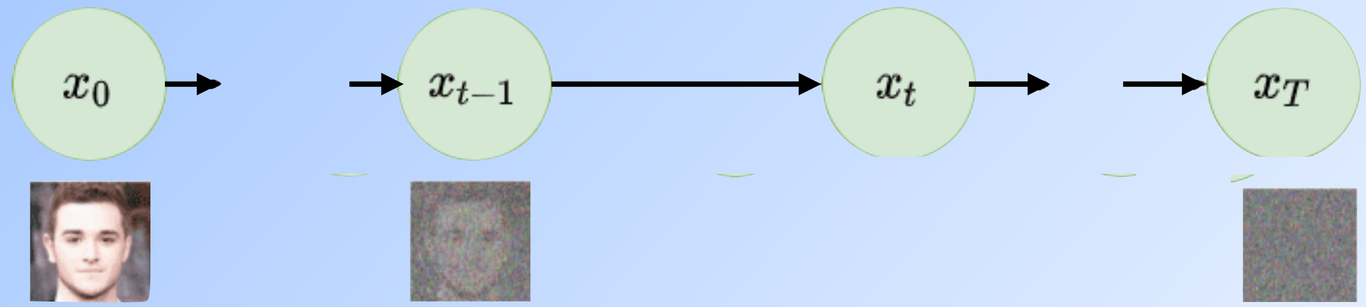
Diffusion Models for Visual Data Generation

- Diffusion models are a class of generative models that produce visual data through progressive denoising (Ho et al., 2020):
  - Forward diffusion: addition of noise to a clean image in successive steps
  - Reverse diffusion: removal of noise in successive steps to recover the clean image.



Diffusion Models for Visual Data Generation

- The forward diffusion process is modeled as a Markov chain with T steps:
  - At step 0, $\mathbf{x}_0 \sim q(\mathbf{x})$ (clean image sampled from the distribution of the training data).
  - At each step of the Markov chain, Gaussian noise with variance β_t is added to the image from the previous step.
  - At step t, $q(\mathbf{x}_t | \mathbf{x_{t-1}}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t I)$.
- The posterior for the forward diffusion process is $q(\mathbf{x}_{1:T} | \mathbf{x}_0) = \prod_{i=1}^{T} q(\mathbf{x}_i | \mathbf{x}_{i-1})$



Diffusion Models for Visual Data Generation

- However, iterating through several steps to generate training samples corresponding to each time step would be prohibitively slow.
- The reparameterization trick is useful to fix this issue as follows:

  Let $\epsilon_t \sim \mathcal{N}(0, I) \forall t$

$$
\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \text{ (Let } \alpha_t = 1-\beta_t) \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon_{t-2} \\
&= \vdots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0
\end{aligned}
$$

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

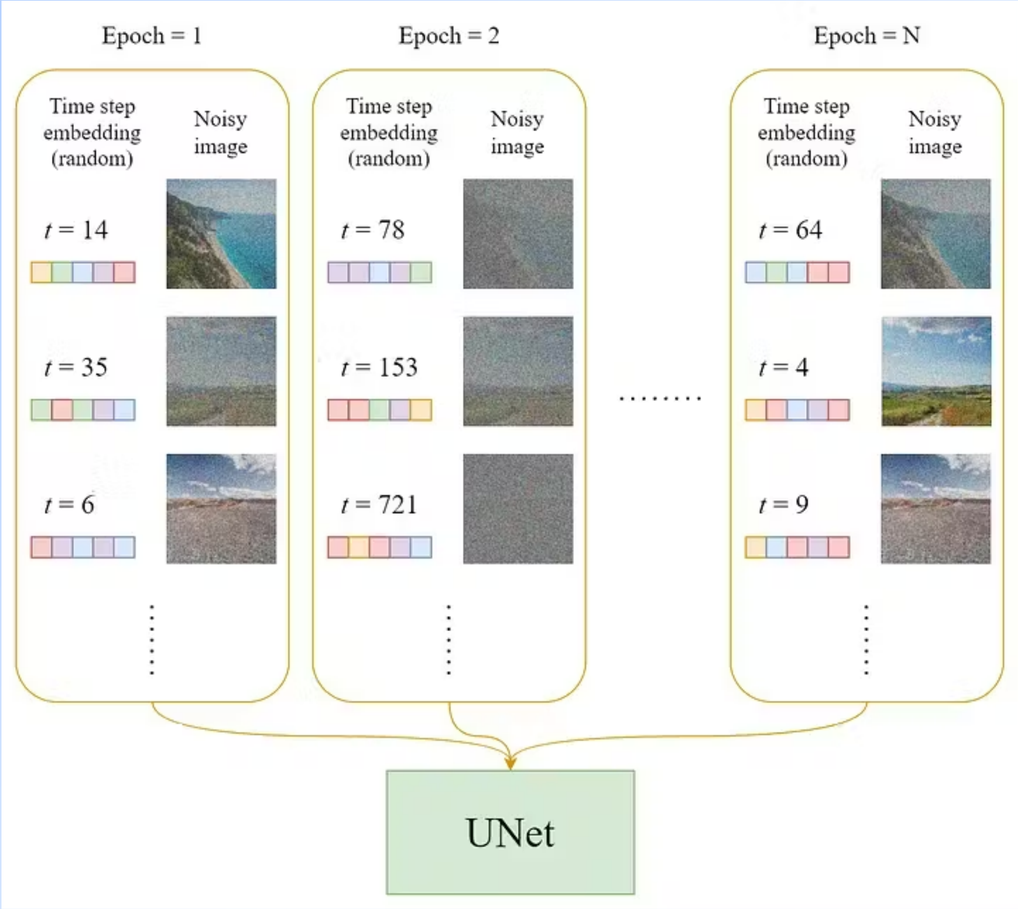
Diffusion Models for Visual Data Generation

- The noise variance $\beta_t \in (0,1)$ progressively increases with each time step as per a fixed schedule that could be specified using different strategies such as linear or cosine scheduling.
- At $T \to \infty$, $\mathbf{x}_T$ is just random noise. So, it should be feasible to recover the image $\mathbf{x}_0$ by learning the reverse distribution. $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$.
- The reverse distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is approximated with a neural network $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$. At each time step, the neural network predicts the Gaussian noise parameters $\mu_\theta(\mathbf{x}_t, t)$, and $\Sigma_\theta(\mathbf{x}_t, t)$, i.e. $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \sim \mathcal{N}\left(\mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)\right)$
- In practice, a U-Net architecture was used as the neural network.

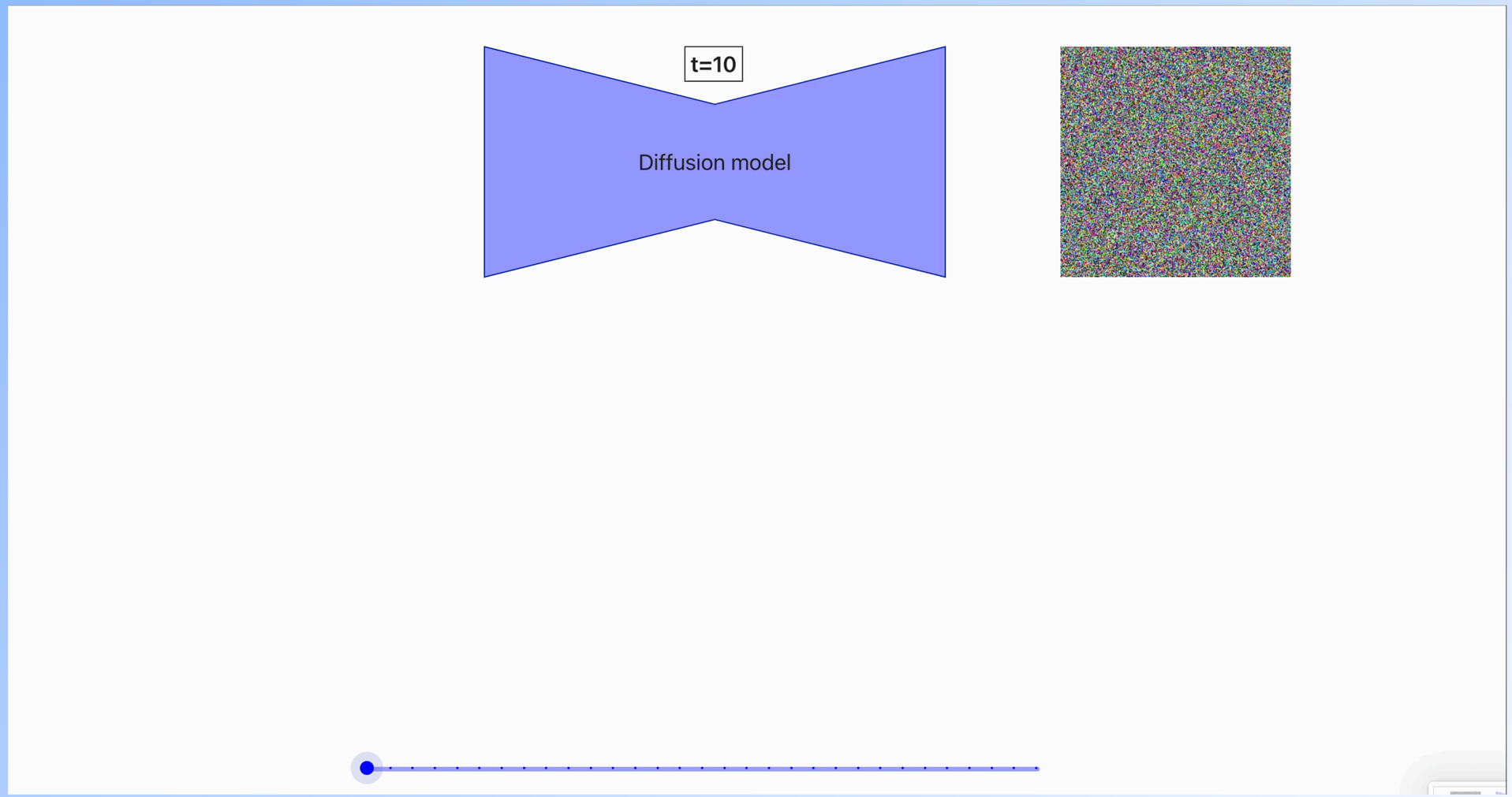
Diffusion Models for Visual Data Generation

- During training, a random time step t is selected for each training sample, and the Gaussian noise corresponding to t is added.
- The noisy image and the time step embedding are used as the input to the neural network (U-Net).
- The loss is computed using the predicted noise $\epsilon_\theta(\mathbf{x}_t, t)$ and the true noise ϵ as follows:
  $L_t = \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\|\epsilon_\theta(\mathbf{x}_t, t) - \epsilon\|^2\right]$.
- The trained diffusion model is used for image generation by successive denoising steps starting from a random noise image, where the the model predicts the entire noise to be removed at a given time step.
- When $\Sigma_\theta(\mathbf{x}_t, t) = \beta_t I$, the noise removal at time step t is performed as follows:

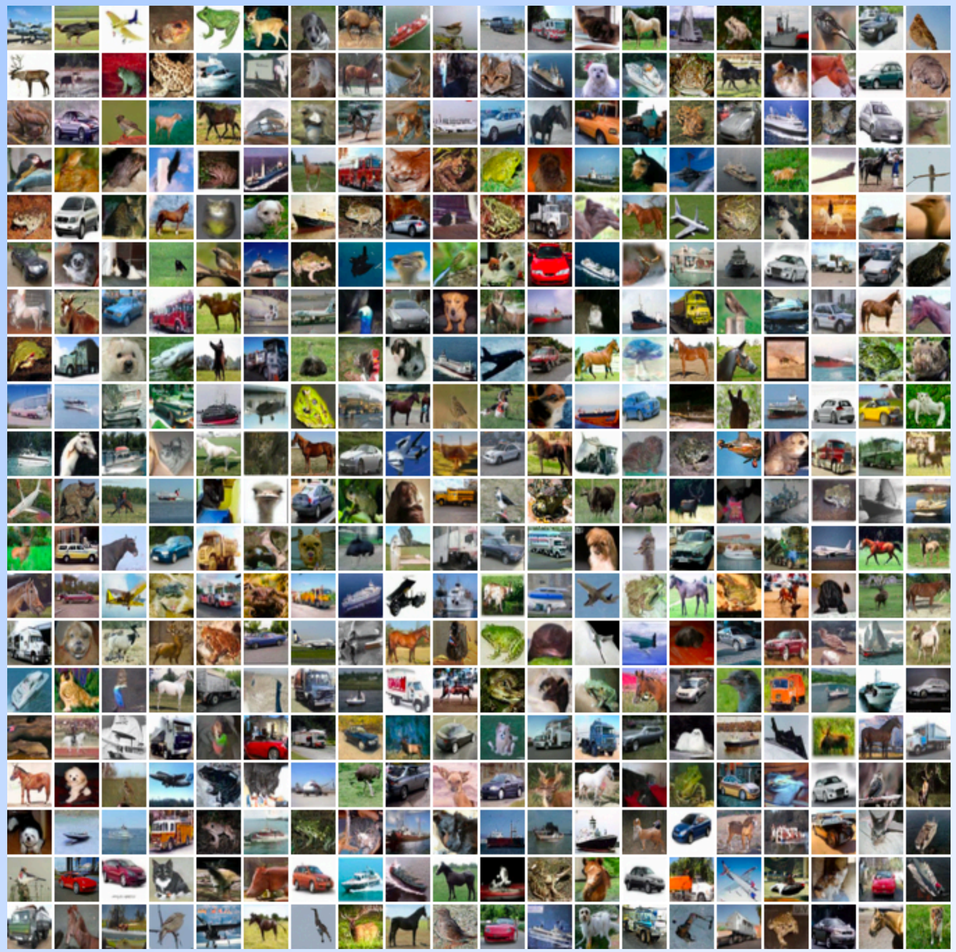
$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sqrt{\beta_t}\epsilon = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sqrt{\beta_t}\epsilon$$



Diffusion Models for Visual Data Generation



Diffusion Models for Visual Data Generation



Diffusion Models for Visual Data Generation

- Further developments in diffusion models involved the introduction of latent diffusion models, that performed the forward and reverse diffusion processes in a lower dimensional latent space, rather than in the high dimensional input space (Rombach et al., 2022).
- Operation in the latent space significantly reduced computational costs and improved generative quality.
- Diffusion models offer several compelling advantages over GANs:
  - Lack of adversarial training makes the training more stable
  - Capable of producing highly photorealistic, high resolution images.
- However, the disadvantage is that multiple denoising steps need to be conducted to produce an acceptable visual output, which necessitates multiple forward passes through the trained neural network model, making the generation process much slower than single step GAN models.
- Popular diffusion models:
  - Stable Diffusion
  - DALL-E
  - Midjourney