ETTI

BIOSINF

Generative Models

Lect. PhD. Ana Nicolae &
T.A. PhD(s) Andrei Radu

Multimedia Technologies in Biometrics and Information Security Applications
National University of Science and Technology ''Politehnica" of Bucharest

Octombrie 2025

Content

1. Intro

Let's have some fun with generative models!

Try these websites

- `https://thispersondoesnotexist.com`
- `https://ai-vs-real.com`
- `https://britannicaeducation.com/blog/quiz-real-or-ai/`

Discriminative vs Generative

Discriminative Models

- **Goal:** Learns a decision boundary.
- **Models:** The conditional probability $p(y|x)$.
- **Tasks:** Classification, Segmentation, Information Retrieval, *etc.*
- **Examples:** Logistic Regression, SVMs, FFC.

Discriminative vs Generative

Generative Models

- **Goal:** Learns the underlying data distribution.
- **Models:** The joint probability $p(x, y)$ or the marginal $p(x)$.
- **Tasks:** Generation, translation, editing, *etc.*
- **Examples:** VAEs, GANs, Diffusion Models.

Study Case: ChatGPT – where generative meets discriminative

What does GPT stand for?

- GPT stands for **G**enerative **P**re-trained **T**ransformer.
- **Generative:** Its primary purpose is to *generate* new text that is statistically similar to the data it was trained on. It creates content, rather than just classifying or analyzing it.
- **Pre-trained:** The model is first trained on a massive, unlabeled dataset. This *pre-training* phase teaches it grammar, facts, reasoning abilities, and stylistic patterns. It's later *fine-tuned* or *down-streamed* for specific tasks like instruction following.
- **Transformer:** This is the specific neural network architecture it uses. The Transformer is exceptionally good at handling long sequences of data.

Study Case: ChatGPT – where generative meets discriminative

How does ChatGPT work?

- At its heart, a GPT model is a sophisticated next-token predictor $\rightsquigarrow$ **discriminative**.
- Given a sequence of text (*prompt*), its fundamental job is to predict the most probable next word (or, more accurately, *token*).
- For example, given the input *The cat sat on the...*, the model's goal is to output a probability distribution over its entire vocabulary, with a high probability assigned to tokens like *mat*, *floor*, or *couch*.
- By repeatedly predicting the next token and appending it to the input it can generate long, coherent sequences of text one token at a time $\rightsquigarrow$ **generative**.

2. AutoEncoders

AutoEncoders (AEs)

What is an AE?

- An **autoencoder** is an **unsupervised** neural network trained to perform a simple task: **reconstruct its own input**.
- It consists of two (or three) main parts:
  - An **encoder**, $\mathcal{E}$, which maps the input x to a compressed latent representation $z = \mathcal{E}(x)$.
  - A **bottleneck**, which contains the compressed latent representation, sometimes further processing or guiding/conditioning it.
  - A **decoder**, $\mathcal{D}$, which reconstructs the input from the latent code, $\hat{x} = \mathcal{D}(z) = \mathcal{D}(\mathcal{E}(x))$.
- The network is trained to minimize a **reconstruction loss**, $\mathcal{L}(x, \hat{x})$.

The Importance of the Bottleneck

All-shaped bottleneck

- **Complete bottleneck:** the dimension of the latent is equal to that of the input: $\dim(x) = \dim(z)$
  Does this cause any problems?

- **Overcomplete bottleneck:** $\dim(z) \gg \dim(x)$
  Is this useful for anything?

- **Undercomplete bottleneck:** $\dim(z) \ll \dim(x)$
  Now we're talking!

Where AE shines...

What are AE used for?

- **Dimensionality Reduction:** The latent code z is a low-dimensional version of x, useful for compression or visualization.
- **Feature Extraction:** The encoder can be used as a pre-trained feature extractor for a down-stream supervised task.
- **Anomaly Detection:** A model trained on normal data will have a high reconstruction error for anomalous inputs.
- **Image Denoising:** The AE is a natural fit for denoising, and the encoder/decoder pair can be used for restoration.

...and its limitations.

Underlying problems

- **Irregular latent:** Some parts might be unused, other overly-used.
- **GIGO:** Sampling a random point in the latent might result in *garbage* results.
- **Not generational:** Can only encode/decode existing data $\rightsquigarrow$ unable to produce new samples!

The VAE: A Generative Approach

No more fixed points!

- A VAE is a **probabilistic generative model** built with an autoencoder-like architecture.
- Instead of mapping an input to a single point z, the encoder maps it to a **probability distribution** over the latent space.
- Specifically, the encoder outputs the parameters (mean μ and variance σ^2) of a Gaussian distribution.
- The latent code z is then *sampled* from this distribution, $z \sim \mathcal{N}(\mu, \sigma^2)$.

The Evidence Lower Bound (ELBO)

From data to probabilities

- The goal is to maximize the log-likelihood of the data, $\log p(x)$.
- Instead, we maximize a lower bound on it, known as the **Evidence Lower Bound (ELBO)**.

$$\log p_\theta(x) \geq \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{Reconstruction Term}} - \underbrace{D_{KL}(q_\phi(z|x)||p(z))}_{\text{Regularization Term}}$$

- **Reconstruction Term:** How well does the decoder reconstruct the input, given a sample from the latent distribution?
- **Regularization Term (KL Divergence):** How much does our learned latent distribution $q_\phi(z|x)$ diverge from a simple prior (usually $\mathcal{N}(0, I)$)? This forces the latent space to be well-structured.

Still not generative enough

Stochastic sampling step

- **Problem:** How to do backprop on stochastic sampling?
- **Answer:** We include it in the model input!

What we replace, exactly?

- Instead of directly sampling the latent

$$z \sim \mathcal{N}(\mu, \sigma^2)$$

- We sample some noise ϵ and add it to our z:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$$

VQ-VAE: Discrete Latent Spaces

The most modern approach

- The encoder outputs a vector, which is then replaced by the nearest vector from a learnable **codebook** (dictionary) of embeddings.
- This quantization step leads to higher-fidelity reconstructions.
- A more powerful model can then be trained over the discrete latent codes to create a strong conditional generative model.

Summary: AE vs. VAE

Autoencoder (AE)

- Deterministic
- Learns a point-estimate latent code
- Simple reconstruction loss
- Good for compression, anomaly detection
- Not generative

Variational Autoencoder (VAE)

- Probabilistic / Generative
- Learns a distribution over latent space
- Complex ELBO loss (Reconstruction + KL)
- Good for data generation, learning smooth representations
- More complex to train

3. Generative Adversarial Networks

What is a GAN?

Definition

A Generative Adversarial Network is an **implicit generative model**. It learns to generate data without defining a probability density function $p(x)$.

But... how?

It learns through a **two-player**, **zero-sum** game between:

- A **Generator (G)**: An *art forger* that takes random noise z (from a simple prior distribution like a Gaussian) and tries to create realistic data samples $G(z)$.
- A **Discriminator (D)**: An *art critic* or *detective* that is a binary classifier. It tries to distinguish between real data samples from the training set and fake samples from the generator.

The two networks are trained in opposition: G gets better by fooling D, and D gets better by seeing more convincing fakes from G.

Let's visualize them

The Minimax Objective Function

GAN loss function

The training process is a minimax game described by the following objective:

$$\min_G \max_D V(D,G) = \underbrace{\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]}_{\text{D wants to maximize this}} + \underbrace{\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]}_{\text{D wants to maximize this}}$$

Taking turns

- **The Discriminator's Turn ($\max_D$)**: This is equivalent to a standard binary cross-entropy loss where the label for real data is 1 and for fake data is 0.
- **The Generator's Turn ($\min_G$)**: G Minimization happens when $D(G(z))$ is high (i.e., when D is fooled into thinking fake samples are real).

Convergence and Divergence

Optimal Discriminator

For a fixed G, we want:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)},$$

where:

- If the sample is real: $D^*(x) \approx 1$
- If the sample is fake: $D^*(x) \approx 0$

Optimal Generator

For a fixed D, we want:

$$p_{data}(x) = p_g(x) \rightsquigarrow D(x) = \frac{1}{2},$$

meaning that the discriminator is maximally confused.

The Unstable Equilibrium

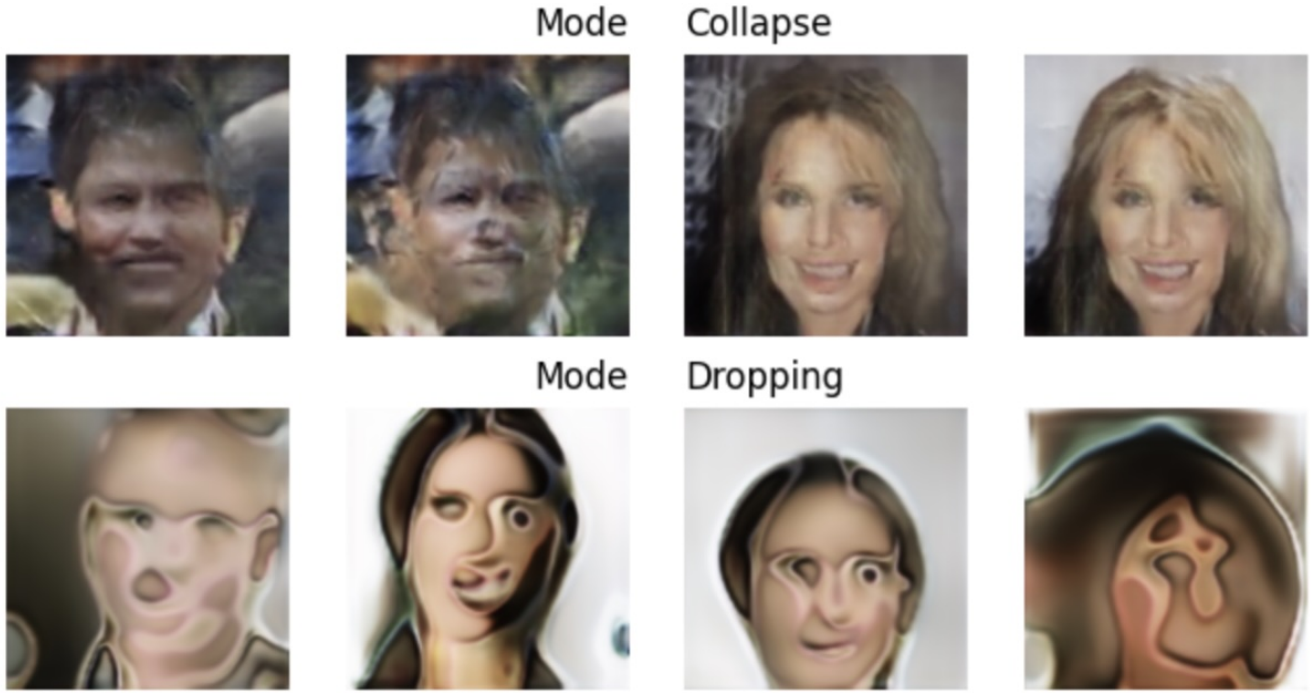
Why we cant achieve Nash equilibrium

- We do not have a straight-forward loss function, optimizing two networks at the same time $\rightsquigarrow$ unstable training
- One network (discriminator) learns much quicker than the other (generator)

What is stoping us?

- **Vanishing gradients:** The generator gradients are too low when the discriminator becomes too powerful.
- **Oscillation:** The loss function fluctuates without finding a local minima $\rightsquigarrow$ generator and discriminator do not improve
- **Generator starts cheating:**
  - Reproduces only a small set of examples $\rightsquigarrow$ **mode collapse**
  - Reproduces similar features in all its data $\rightsquigarrow$ **mode dropping**

Visualizing Failure



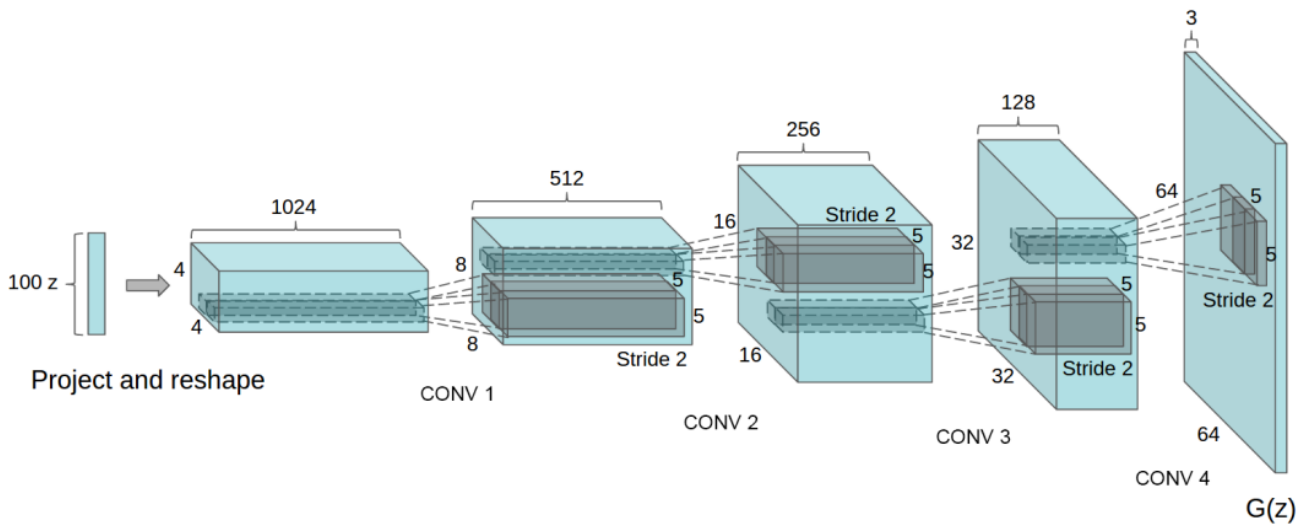
First implementation: DCGAN

Key Innovations

- **All convolutional Net:** Both the generator and discriminator are fully-convolutional, learning specific down samplings/up samplings.
- **Batch Normalization:** Stabilizes training by normalizing distributions after each layer, allowing proper gradient flow.

Impact

- First high quality image generation model
- Latent spaces has a structure, allowing for latent interpolation.



Making Training Stable: WGAN

Key Innovations

- **Critic, not Discriminator:** Replace the final prediction with a *realness* score, not a real/fake prediction. This new score is used to tell the Generator *how far* it is from the actual sample.
- **Better loss:** Replaced the standard GAN loss with Wasserstein distance (*i.e. minimum energy to "move" from a distribution to another)*. This new loss is used jointly with the *realness* score to optimize the networks.
- **Weights clipping:** Proposes a method for stabilizing the training, especially in the Discriminator, slowing it down and making it more robust to mode collapse.

Impact

- Showcases that GANs can be trained in a more stable manner.

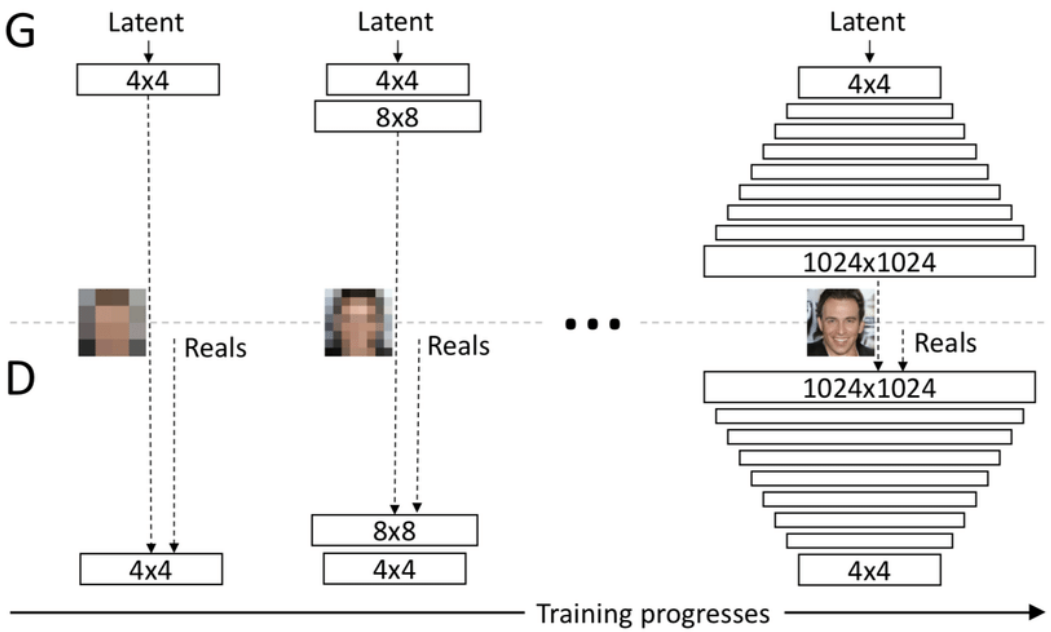
Iterations Makes Perfections: PGAN

Key Innovations

- **Start small:** Generate HQ samples iteratively, by first producing low-res ones, refine and then upscale!
- **Adding layers:** During training, once a plateau is reached, more layers are added to increase the resolution.

Impact

- First method to achieve 1024×1024 photorealistic images.



Adding disentanglement: StyleGAN

Key Innovations

- **Mapping Network:** Add a new module to process the z latents, creating a more meaningful (and disentangled) W space.
- **AdaIN:** Adaptive Instance Normalization uses the style codes from W to control the μ and σ of the features at each layer, resulting in different layers controlling features at different scales: first layers create general appearance while higher ones focus on finer details.

Impact

- Allowed image editing by combining different styles in the W space, with explicit control over coarse-to-fine details
- Achieves better quality and a more disentangled latent space, being a de-facto model in image generation.

Combining styles

Make them conditional: cGAN

Key Innovations

- **Conditional:** Adds information in the form of one-hot vectors to both the Generator and Discriminator.
- **Affects both models;** The Generator learns features specific to a class in the provided condition. The Discriminator is forced to predict the sample class (including a *fake* one).

Impact

- Was a foundation for conditioning generative models.
- Allowed finer control over the generation, not just a generic image.
- Almost free-cost, which is why later added to other GANs (such as StyleGAN-2)

Image-to-Image Translation: CycleGAN

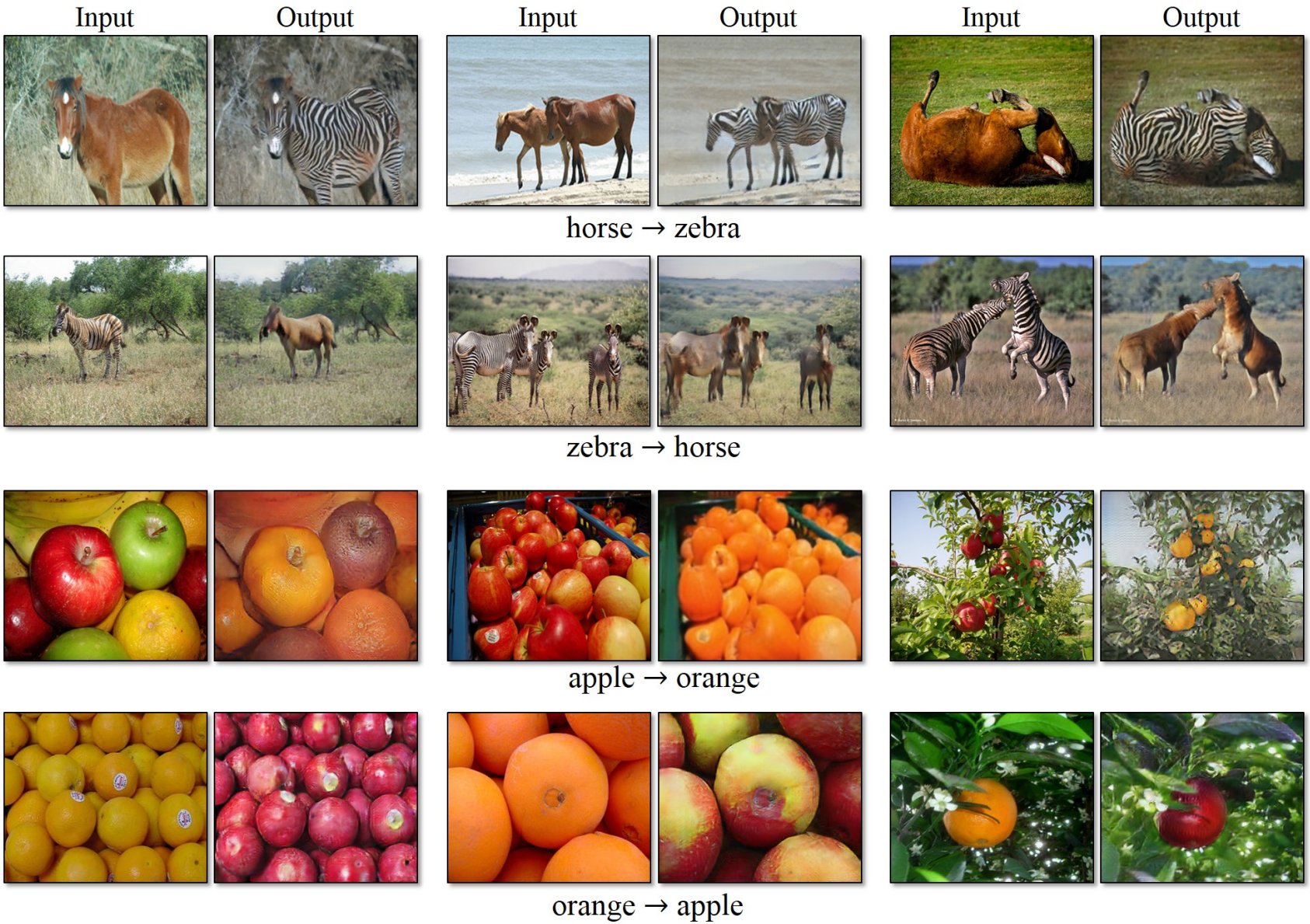
Key Innovations

- **Cycles:** Learns to translate between two sets of images: A and B, to morph images from one set to the other.
- **No paired data:** Does not require exact pairs of images, only two distinct but similar sets (*e.g.* zebra and horse, summer and winter, photos and paintings).
- **We have 2 GANS:** G_{AB} and G_{BA}, which are trained on *cyclic generation:* $\hat{x}_A = G_{BA}(G_{AB}(x_A))$

Impact

- Extens GANs to image-translation tasks.
- Allows for explicit changes without latent manipulation tricks or conditioning.

CycleGAN in action



Applications: Where are GANs Used?

Usage in multiple domains:

- **Creative Arts & Design:** Generating novel artistic images, textures, and designs.
- **Data Augmentation & Simulation:** Creating synthetic data to train other machine learning models, especially in low-data domains like medical imaging. Generating realistic scenarios for training autonomous systems (e.g., self-driving cars).
- **Image Editing & Super-Resolution:** "Inpainting" missing parts of images, increasing image resolution.
- **Fashion & E-commerce:** Creating virtual try-ons and wrapping product images.

How about misuses?

- We better do not talk about this...

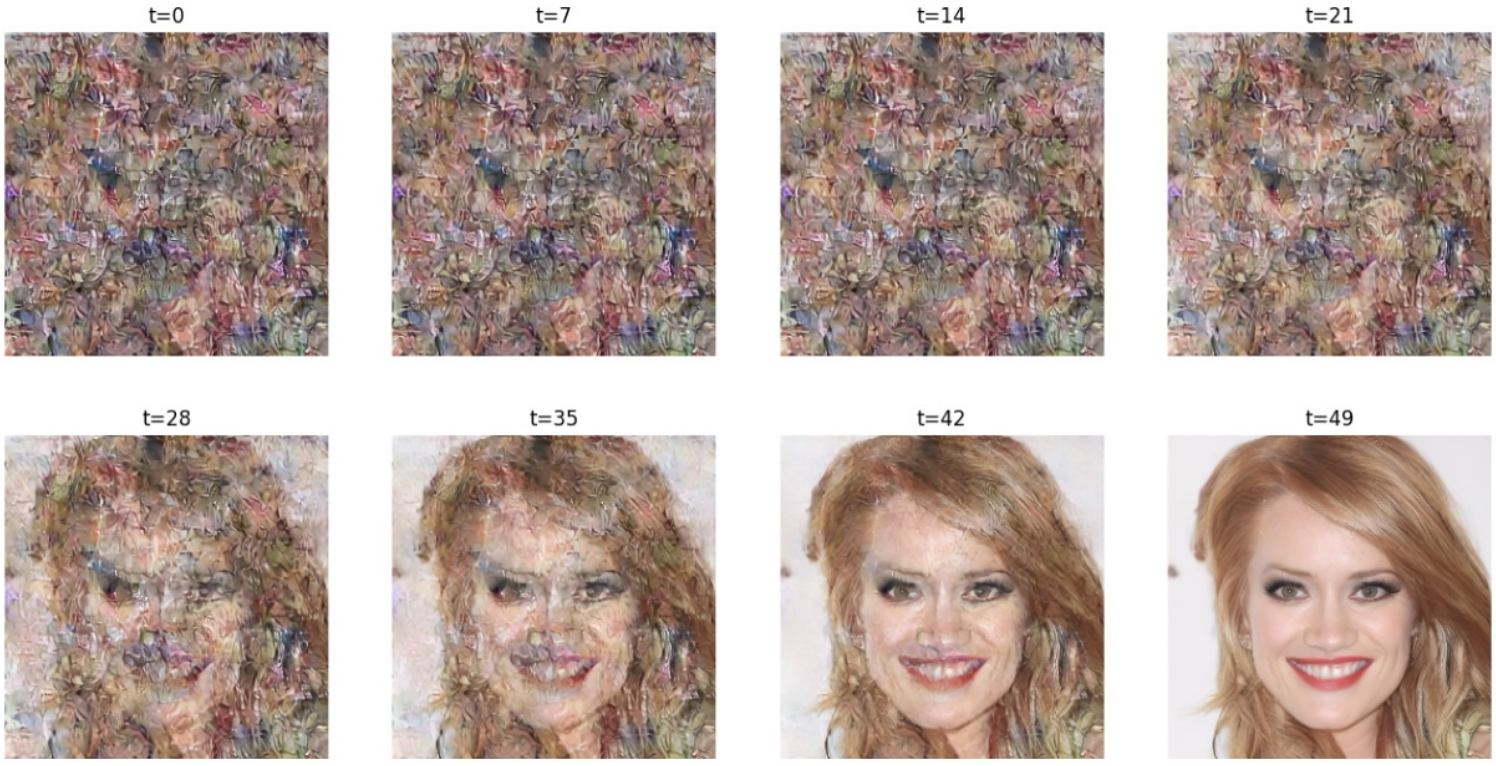
4. Diffusion Models

Diffusion Models

Intuition and Formulation

- Diffusion models are a class of generative models inspired by thermodynamics.
- They learn to generate data by reversing a gradual **noising process**.
- **The Two-Step Process:**
  1. **Forward Process (Fixed):** Systematically destroy structure in the data by gradually adding Gaussian noise over a series of steps. This is a fixed, non-learned process.
  2. **Reverse Process (Learned):** Train a neural network to reverse this process. The network learns to gradually denoise an image, starting from pure noise, to generate a clean sample.

Visualizing this process



Mathematical approach

Forward

- We start with a clean data sample $x_0 \sim p_{data}(x)$.
- We define a sequence of T noising steps (e.g., $T = 1000$). At each step t, we add a small amount of Gaussian noise according to a variance schedule β_t.
- We can sample x_t at any arbitrary timestep t directly from x_0, without iterating:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(0, I) \text{ and } \bar{\alpha}_t = \prod_{i=1}^{t}(1 - \beta_i)$$

Note: *The β_t values are not random! We chose them such that if we apply this process T times, x_T is approximately pure Gaussian noise ($x_T \approx \mathcal{N}(0, I)$).*

Mathematical approach II

Reverse

- The goal is to learn the reverse transition $p_\theta(x_{t-1}|x_t)$ which will undo the noising process at timestep t.
- This reverse transition is also a Gaussian, but with learned mean and fixed variance:

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

- The network's job is to predict the mean $\mu_\theta(x_t, t)$ of the slightly less noisy image x_{t-1}.

Note: *Since the reverse is also a Gaussian, the forward process is "locked". Choosing other distribution as a starting point (x_T) will not yield satisfactory results!*

DDPM: Let us simplify training

Objective function

- In theory, we can use the ELBO, similar to VAEs. However, it will be slower and inaccurate because of the high number of steps.
- Since the σ is fixed, predicting the μ_θ is similar to predicting the noise ϵ.
- This leads to an even simpler loss: $\mathcal{L}_2$:

$$\mathcal{L}_2 = \mathbb{E}_{t,x_0,\epsilon}\|\epsilon - \epsilon_{\text{pred}}\|^2 = \mathbb{E}_{t,x_0,\epsilon}\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2$$

Quite easy, actually!

1. Pick a random training image x_0, a timestep t and generate some noise ϵ
2. **Forward:** Create the noisy image x_t using the forward formula.
3. **Reverse:** Ask the neural network to predict ϵ from x_t and t.
4. The loss is the difference between the network's prediction and the actual noise.

The Denoising Network Architecture

UNet

- An encoder-decoder architecture with **skip connections** between corresponding layers in the encoder and decoder.
- Preserving low-level details (from skip connections) while capturing high-level semantics (from the bottleneck).
- **Timestep Information (t)** is typically encoded into a vector using sinusoidal positional embeddings.

Further Reading

It all depends what you want!

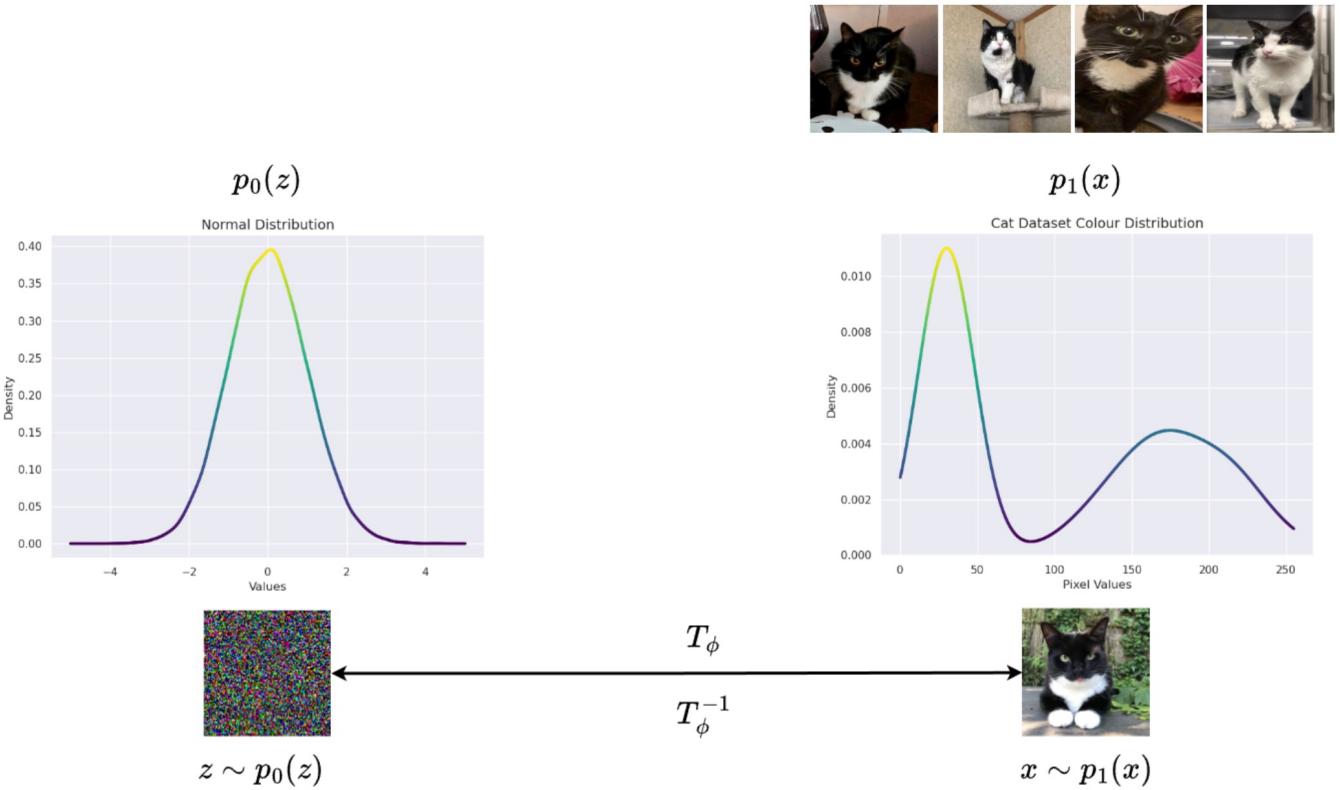
- Conditional Generation:
  - Contrastive Learning (CLIP, CLAP)
  - Latent Diffusion Models (LDMs)
  - Classifier-free guidance (CFG)
- Speeding up the process:
  - Denoising Diffusion Implicit Models (DDIM)
  - Latent Diffusion Models (LDMs)
  - Normalizing Flows (NF), Continuos NF, Flow Matching
- Applications:
  - Text-to-Image Generation: DALL-E, Stable Diffusion, Midjourney
  - Text/Image-to-Video: Sora, Lumiere
  - Text-to-Audio: AudioLDM, a-WaSP, Tango

5. Flow-based Models

Intuition

What we are interested in

- Apply a random transformation to an image until it no longer makes sense.
- Apply the inverse transformation to noise until it starts making sense!



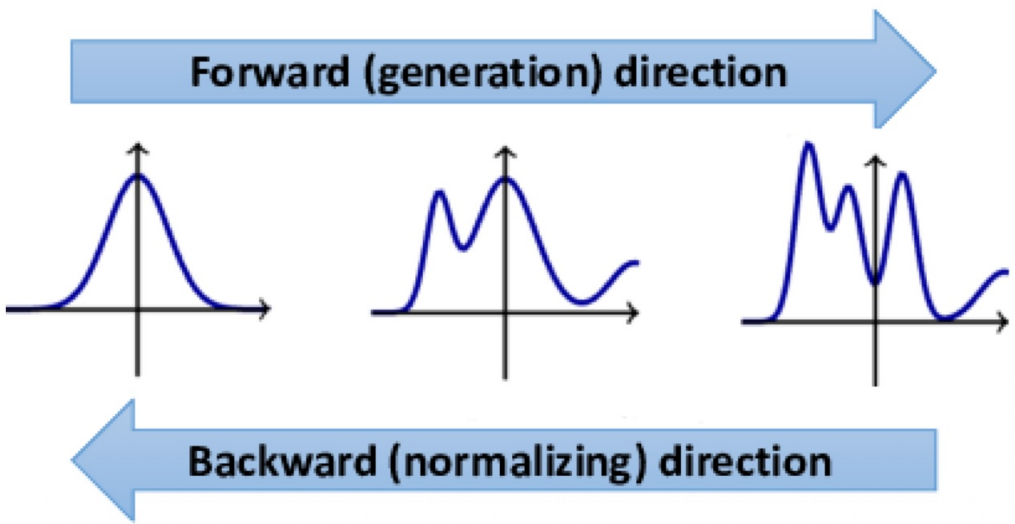
Is a single transformation enough?

Normalizing Flows

- Would you train a Neural Networks with a single layer?
- We can write T_ϕ as a composition of functions:

$$z_0 \xrightarrow{T_{\phi,1}} z_1 \xrightarrow{T_{\phi,2}} \cdots \xrightarrow{T_{\phi,i}} z_i \xrightarrow{T_{\phi,i+1}} \cdots \xrightarrow{T_{\phi,k}} z_k = x$$

with each $T_{\phi,i}$, $i \in [1, k]$ representing a different transformation.



Flow Matching

Linked with Continuous Normalizing Flows

- We can treat each transformation as a *change of state* of the previous one:

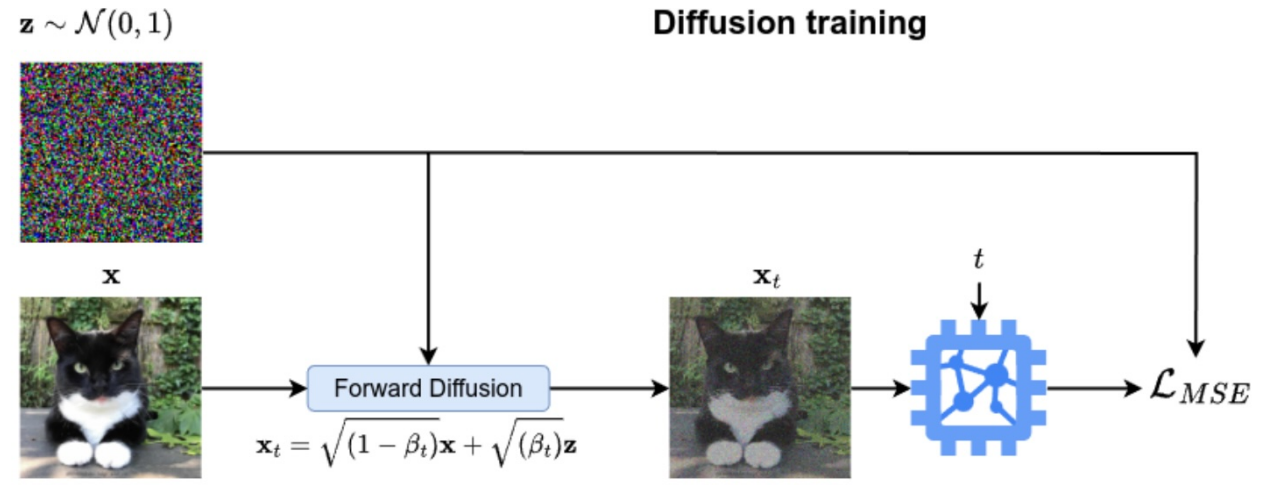
$$z_{k+1} = z_k + r_k(z_k)$$

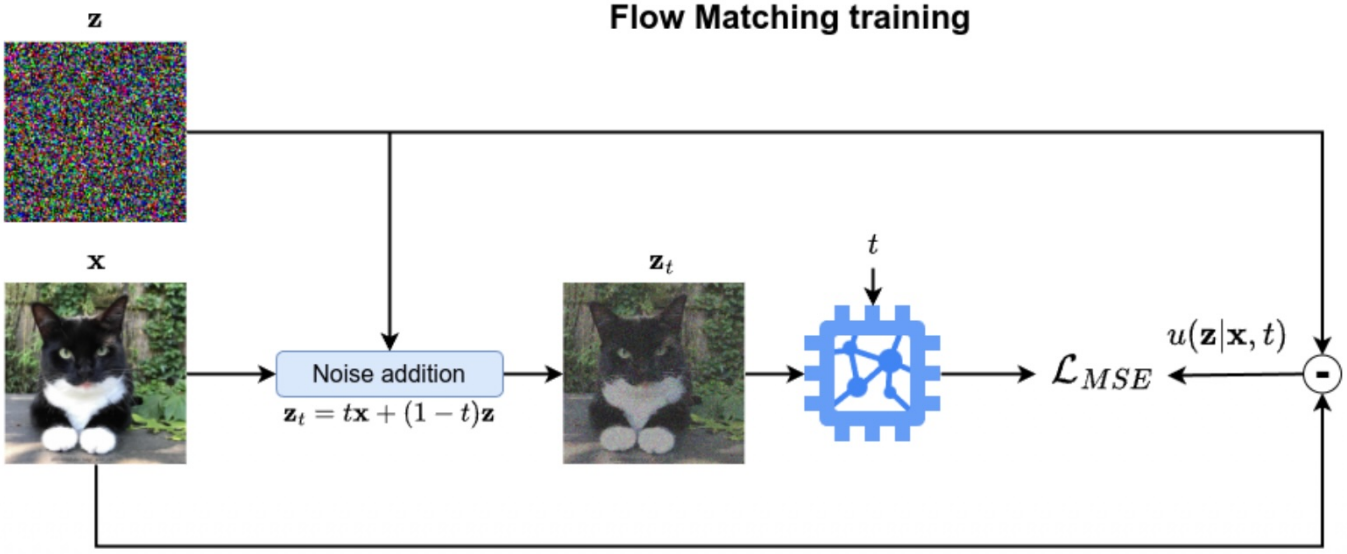
- If we consider $k \to \infty$:

$$\frac{\delta z_t}{\delta t} = r_t(z_t) = u(z_t, \phi, t)$$

- This is very similar to diffusion models...

Let's compare Diffusion and Flow Matching



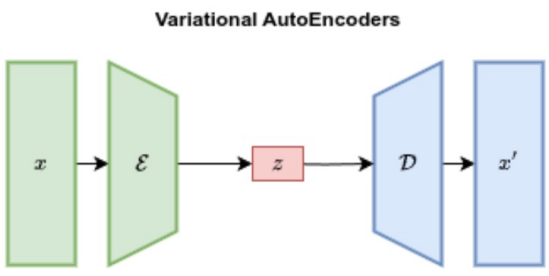


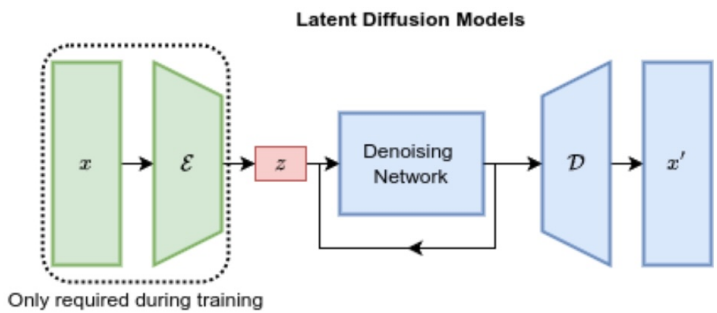
6. Conclusions

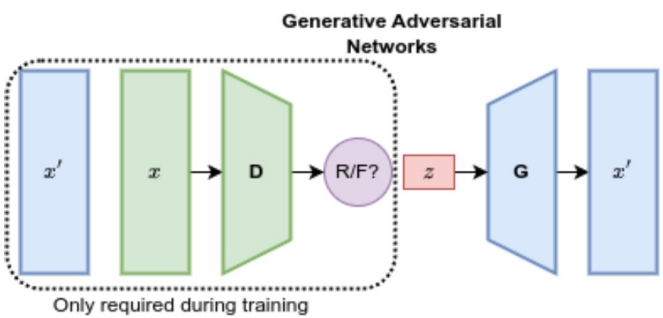
Summary

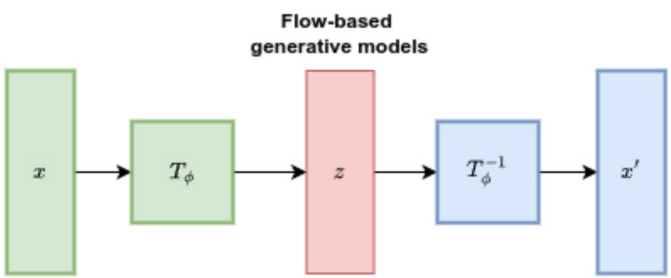
What we did today?

- We explored the theory behind **VAEs**, the adversarial dynamics of **GANs**, and the iterative refinement of **Diffusion Models**.
- We explored the **Generative Trilemma**.









Q&A

End of Course III.