

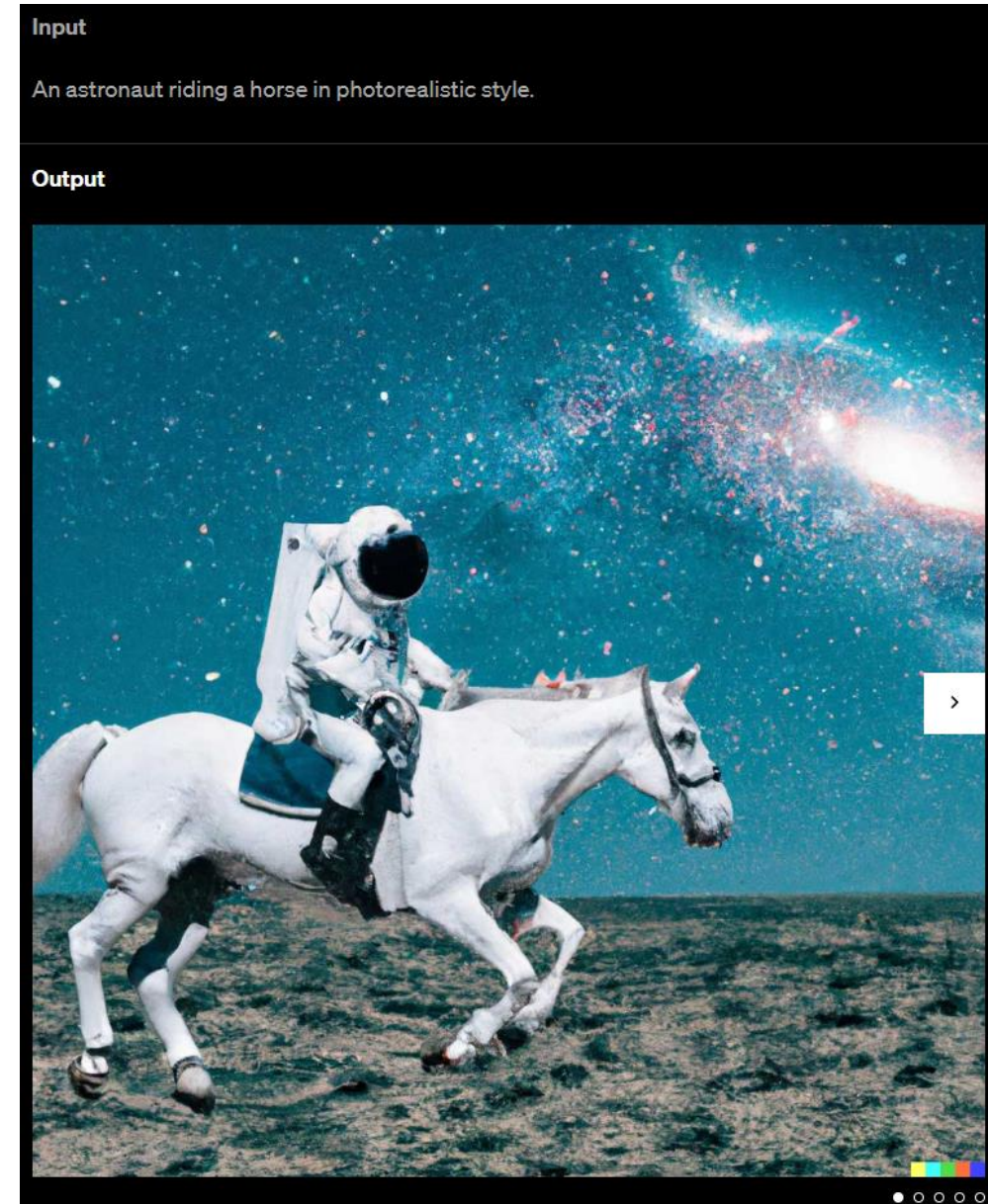
Image Synthesis

Deep Learning and Image Processing

idea: generate new images as variations of training data (same distribution)

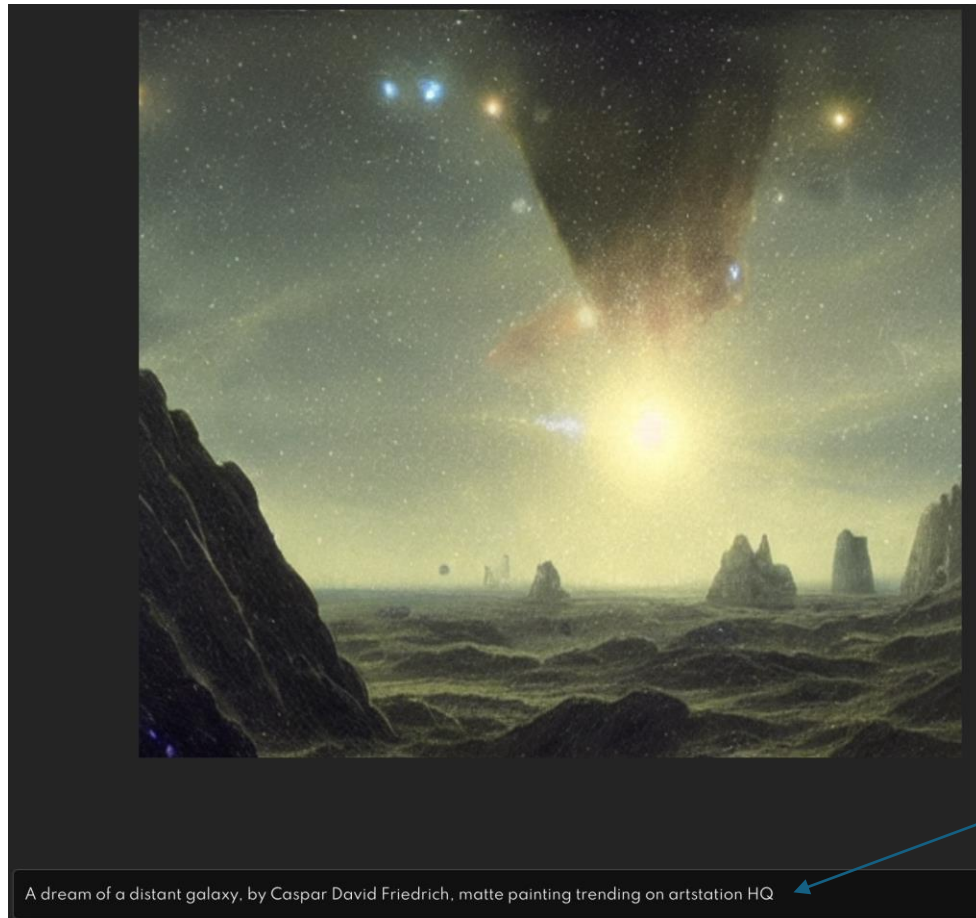
usually conditioned on text (prompt)

compared to text generation, additional mechanism needed (e.g., diffusion) due to more complex image structures

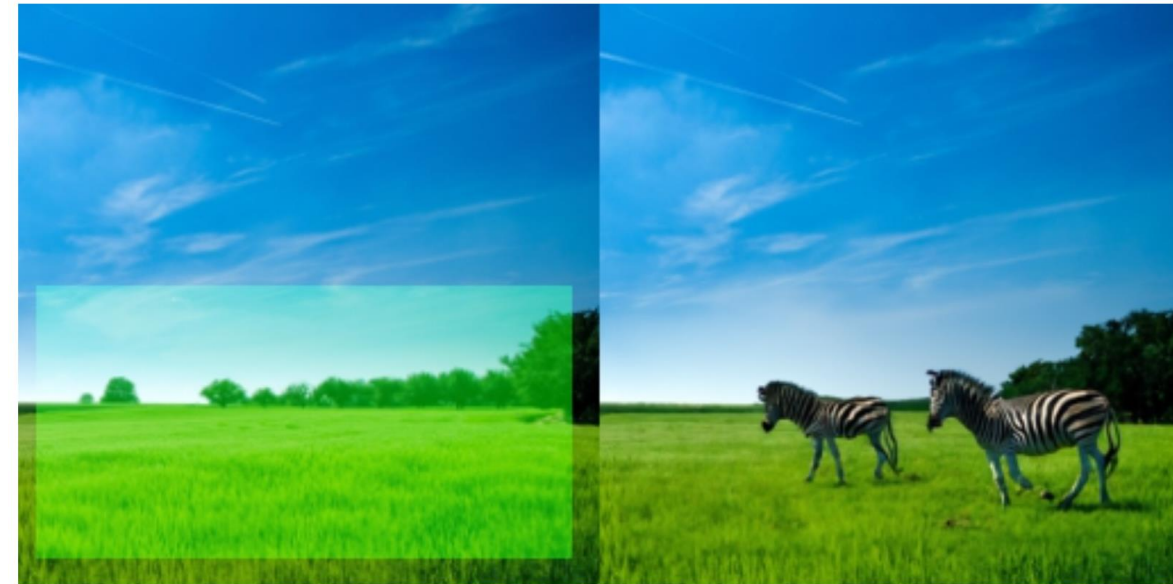


plenty of products: [DALL-E](#), [Stable Diffusion](#), [ImageGen](#), [Midjourney](#), ...

web app for Stable Diffusion: [DreamStudio](#)



inpainting example ([GLIDE](#)):



prompt → “zebras roaming in the field”

[source](#)

Generative vs Predictive/Discriminative Models

discriminative models:

predict conditional probability $P(Y|\mathbf{X})$

generative models:

predict joint probability $P(Y, \mathbf{X})$

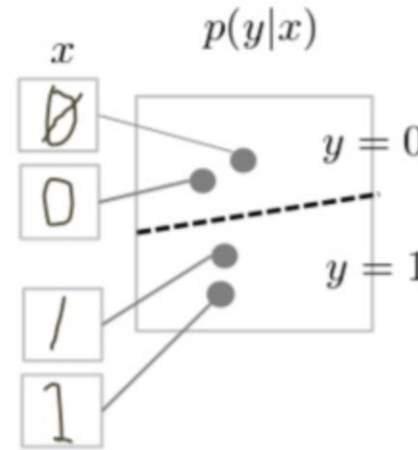
(or just $P(\mathbf{X}) \rightarrow$ unsupervised learning)

\rightarrow allow to generate new data samples

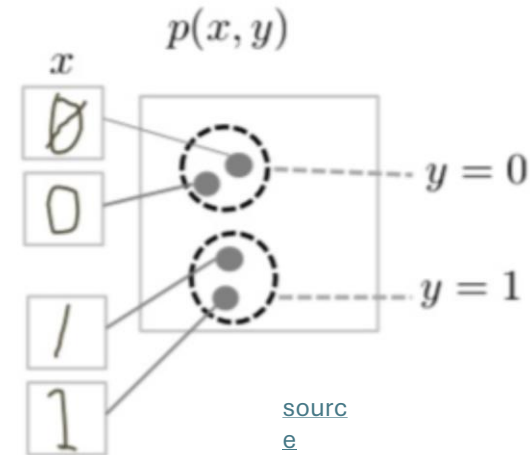
Generative models can be used for predictive tasks (Bayes theorem).

But predictive models are usually better at it.

discriminative model



generative model



task of generative models more difficult: need to model full data distribution rather than merely find patterns in inputs to distinguish outputs

Classic ML vs Deep Learning

Generative vs Predictive Models

text generation

image synthesis

text-to-video

ChatGPT

```
done = False
total_reward = 0
while not done:
    state = torch.tensor(state, dtype=torch.float32)
    logits = policy_network(state)
    action = torch.argmax(logits).item()
    next_state, reward, done, _ = env.step(action)
    total_reward += reward
    state = next_state
    total_rewards.append(total_reward)

# Print average reward
print("Average Reward:", np.mean(total_rewards))
```

This code uses PyTorch to implement the policy gradient method (REINFORCE algorithm) to solve the CartPole problem. The policy network is defined as a simple feedforward neural network, and the training loop updates the policy network parameters to maximize the expected reward. Finally, it evaluates the learned policy by running it for 100 episodes and prints the average reward. Adjust hyperparameters and network architecture as needed for better performance.

Can you extend this to an actor-critic method?

Certainly! Here's an extension of the previous code using the actor-critic method to solve the CartPole problem:

Message ChatGPT

ChatGPT can make mistakes. Check important info.

ChatGPT



Prompt: Epic anime artwork of a wizard atop a mountain at night casting a cosmic spell into the dark sky that says "Stable Diffusion 3" made out of colorful energy

Stable Diffusion 3 — Stability AI



Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She...

Sora | OpenAI

BERT family

tabular data

computer vision



dmlc XGBoost

Classify

Detect

Segment

Track



YOLO



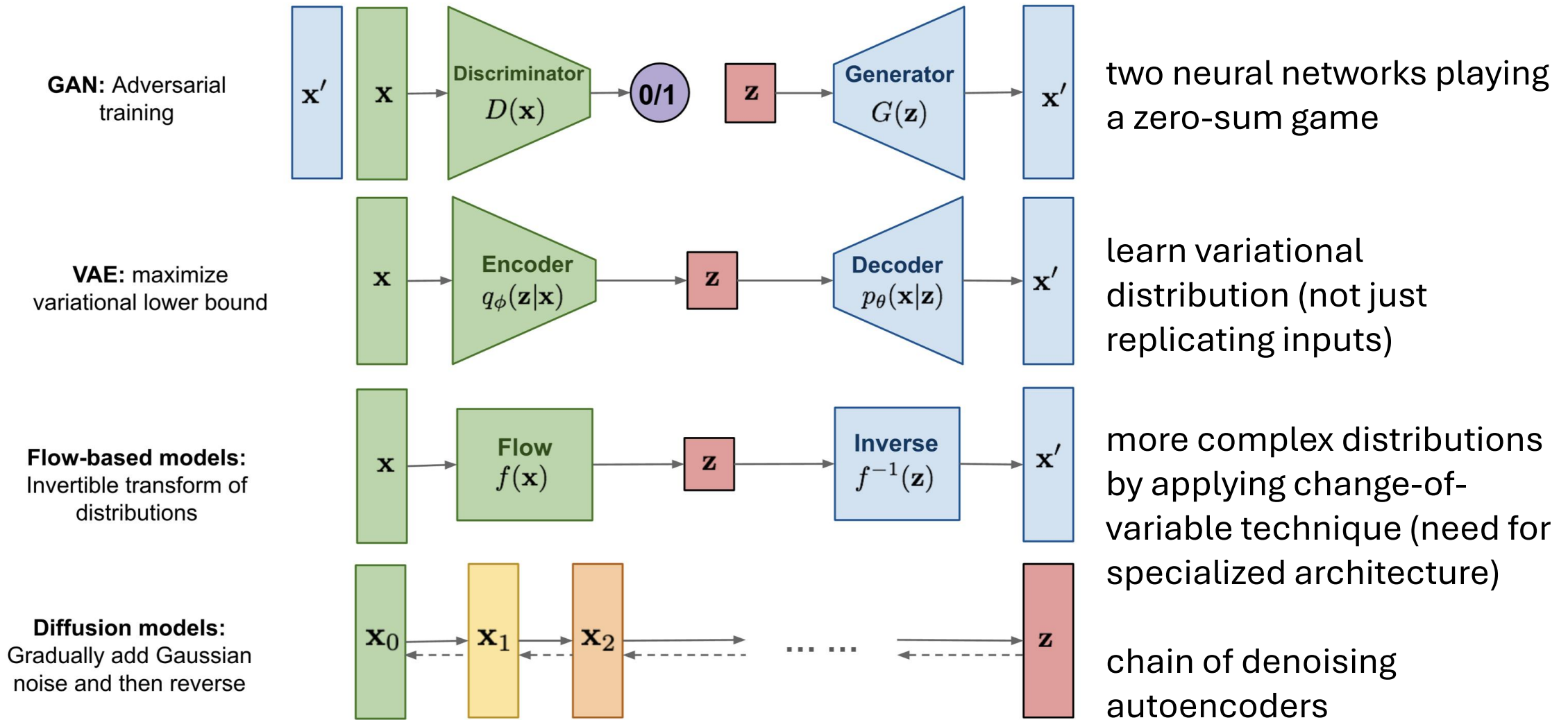
Deep Learning for Generative AI

Depending on the application, there are currently two dominant approaches for generative AI:

- text generation: decoder LLMs
- image synthesis: diffusion models

note the difference between image synthesis and multimodal understanding in LLMs
(images as additional input sequences to transformer, tokenized by splitting into patches)

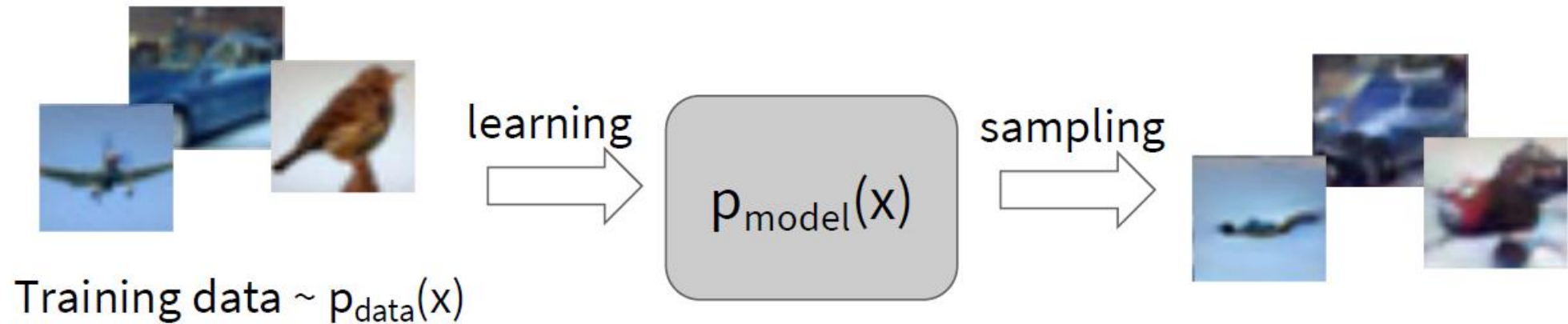
Different Model Types for Image Synthesis



→ generalization: [flow matching](#)

source

Generative Modeling



Objectives:

1. Learn $p_{\text{model}}(x)$ that approximates $p_{\text{data}}(x)$
2. Sampling new x from $p_{\text{model}}(x)$

Explicit density estimation: explicitly define and solve for $p_{\text{model}}(x)$

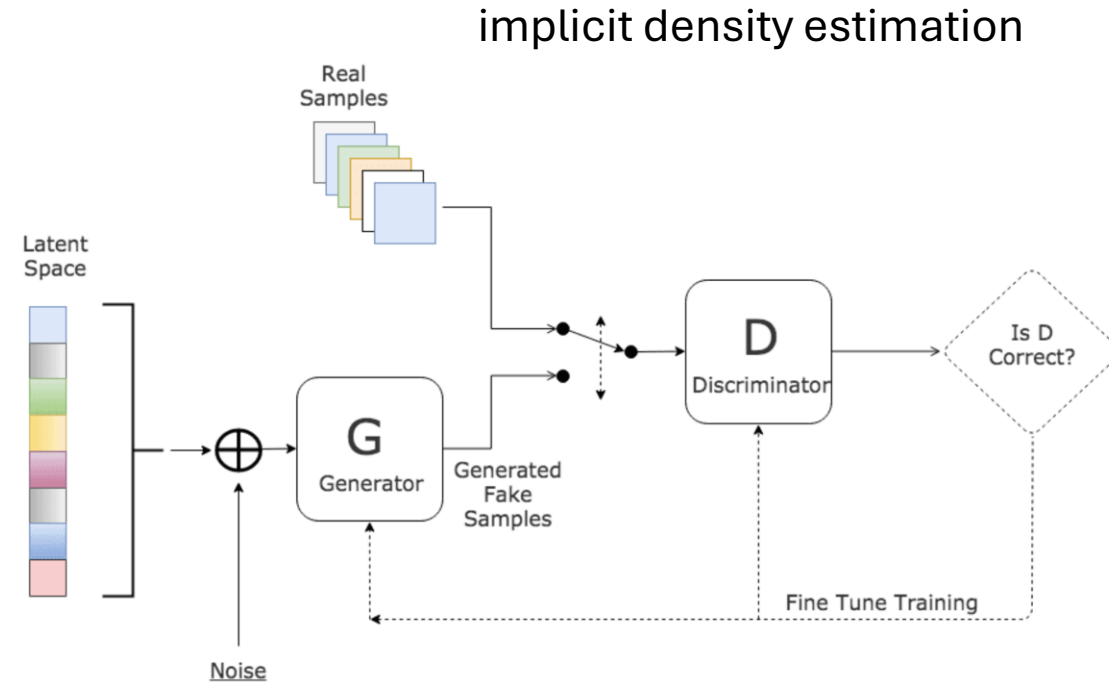
Implicit density estimation: learn model that can sample from $p_{\text{model}}(x)$ without explicitly defining it.

Generative Adversarial Networks (GAN)

two neural networks playing a zero-sum game:

- the generator network G generating new (fake) samples
- the discriminator network D trying to distinguish between real and fake samples

indirect training via D: G not trained directly to minimize reconstruction error of real samples, but to fool D → self-supervised approach



[source](#)

common loss for generator and discriminator:

$$L(x_i) = E_{x \sim p_r(x)} [\ln D(x_i)] + E_{x \sim p_g(x)} [\ln(1 - D(x_i))]$$

G trying to minimize

D trying to maximize

Vector Arithmetic in GAN Latent Space



[source](#)



Variational Autoencoder (VAE)

goal: generation of variations of input data rather than compressed representation

→ learn variational distribution instead of identity function

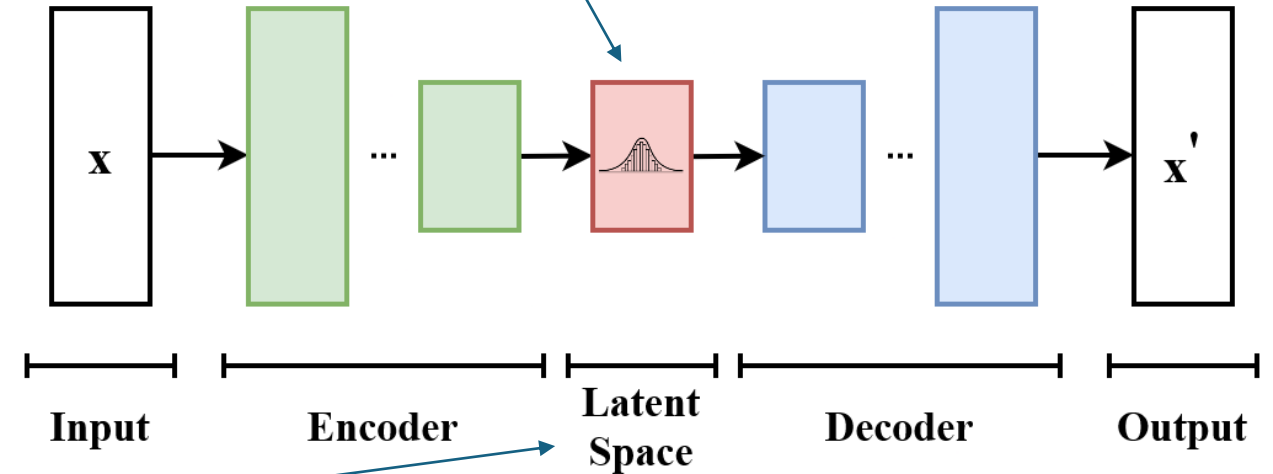
to be precise: parametrized variational distribution of latent encoding variables \mathbf{z}

prior (simple distribution, in usual VAE: Gaussian): $p_{\theta}(\mathbf{z})$

posterior: $p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{\int p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})d\mathbf{z}}$

$p_{\theta}(\mathbf{x})$: mixture of Gaussians

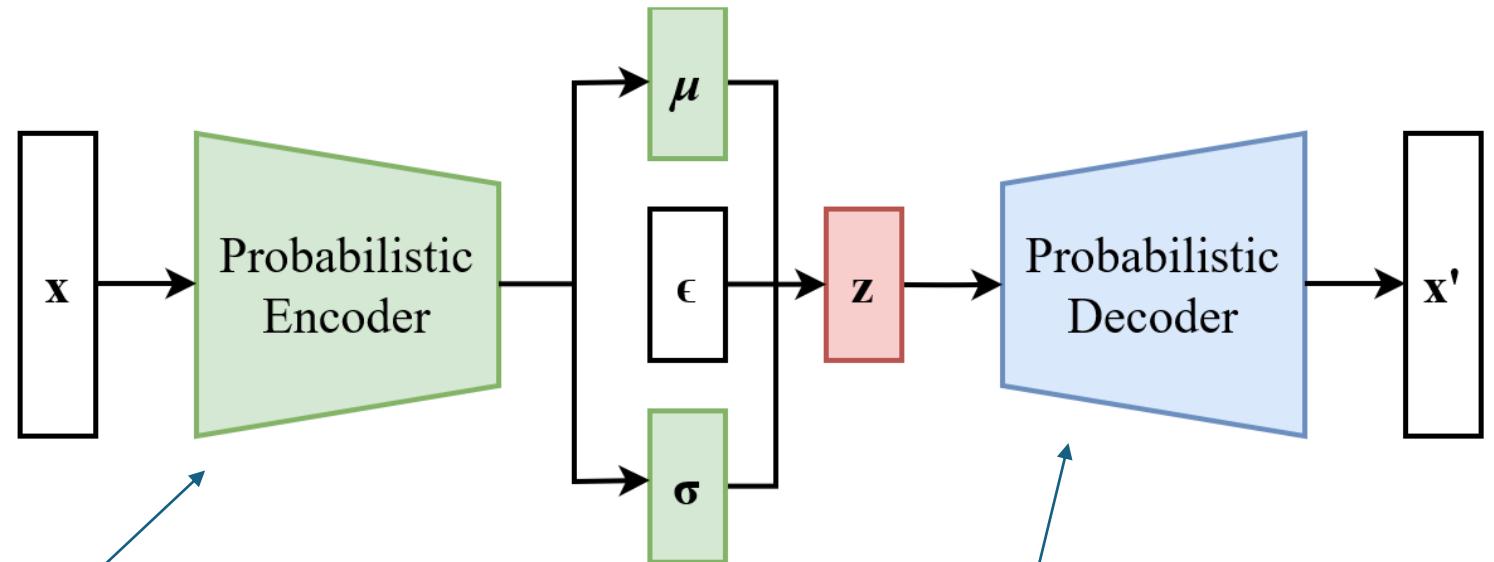
from which to sample



Variational Bayesian Method

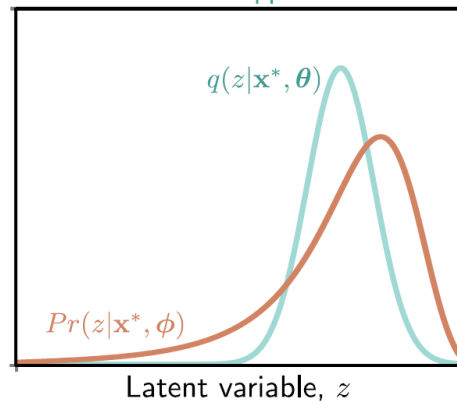
Gaussian Approximation

learn mean and variance of multivariate Gaussian with diagonal covariance structure



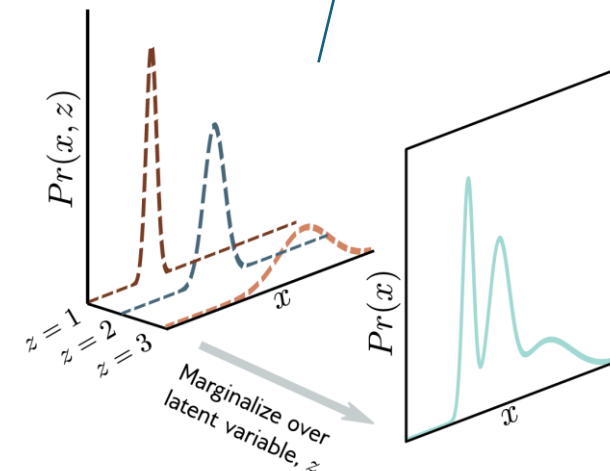
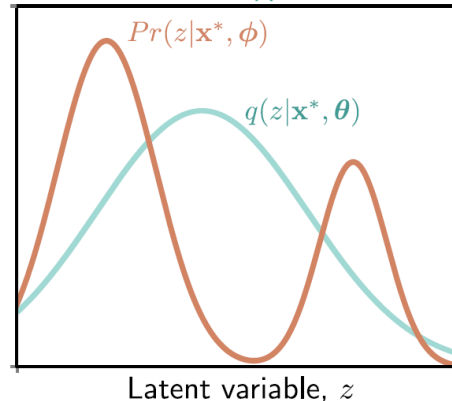
good approximation:

Posterior and approximation



poor approximation:

Posterior and approximation

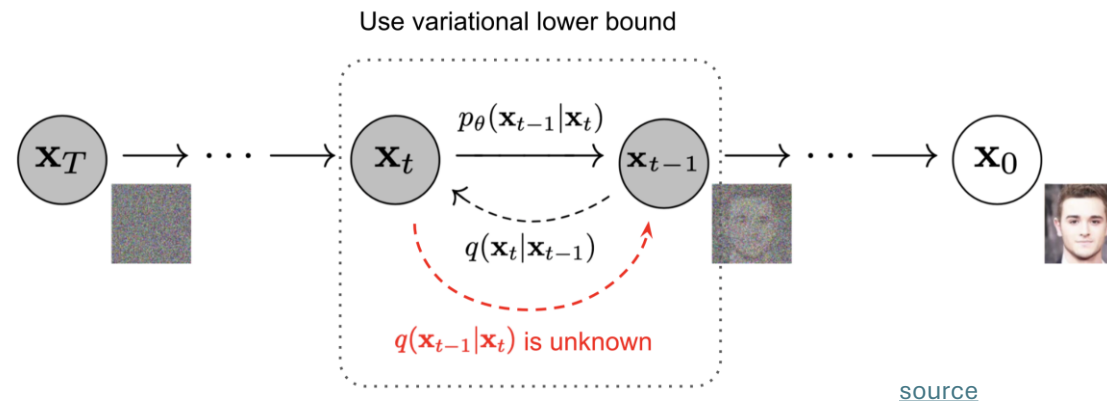


[source](#)

Diffusion

training: distort training data by successively adding random noise, then learn to reverse this process (denoising)

generation: sample random noise and run through the learned denoising process



advantages: easy to train, produce high-quality/realistic samples

can be interpreted as special case of hierarchical VAE (one latent variable generates another) with fixed encoder and latent space of same size as the data

→ more sophisticated latent space than just Gaussian mixture in VAE

Diffusion as Chain of Denoising Autoencoders

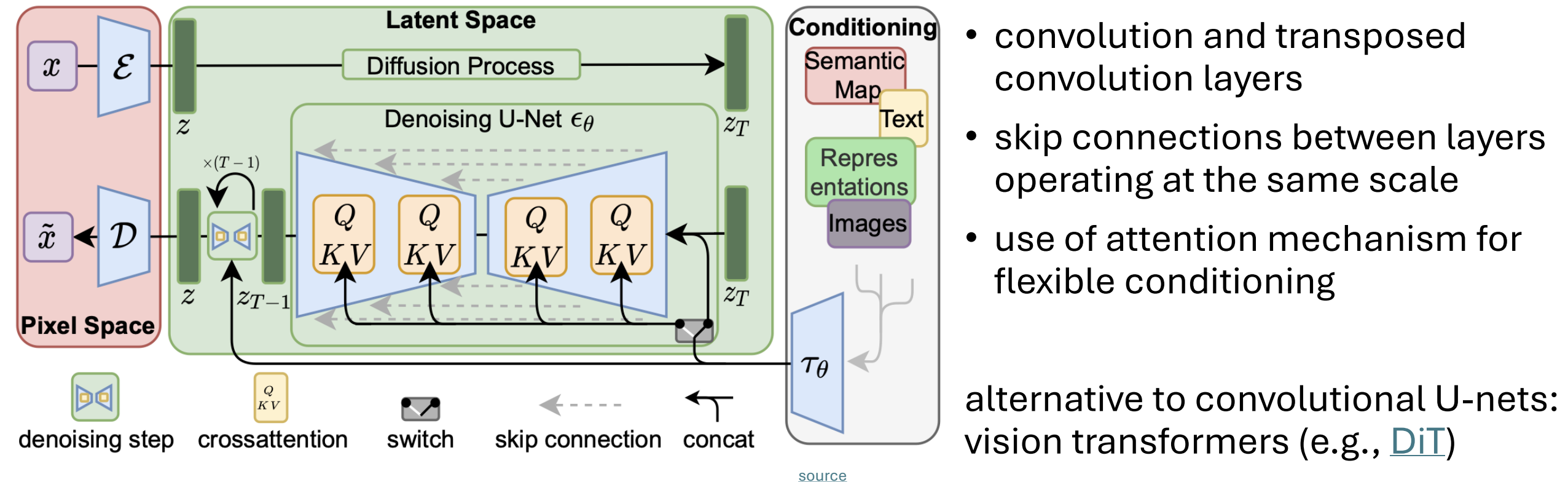
differences of diffusion models to typical denoising autoencoders:

- no bottleneck (care about output here, not internal representation): latent space with high dimensionality (same as original data)
- handle many different noise levels with single set of shared parameters

important application: [AlphaFold 3](#) uses diffusion-based architecture for protein structure prediction

Latent Diffusion

add noise to latent representation rather than raw data
→ significant speedup



- convolution and transposed convolution layers
- skip connections between layers operating at the same scale
- use of attention mechanism for flexible conditioning

alternative to convolutional U-nets:
vision transformers (e.g., [DiT](#))

Outlook: Text2Anything

next step: text-to-video ([Make-A-Video](#), [Lumiere](#), [Sora](#), ...)

→ rudimentary physics understanding

at some point maybe also generation of proteins, materials, ...