Image Synthesis

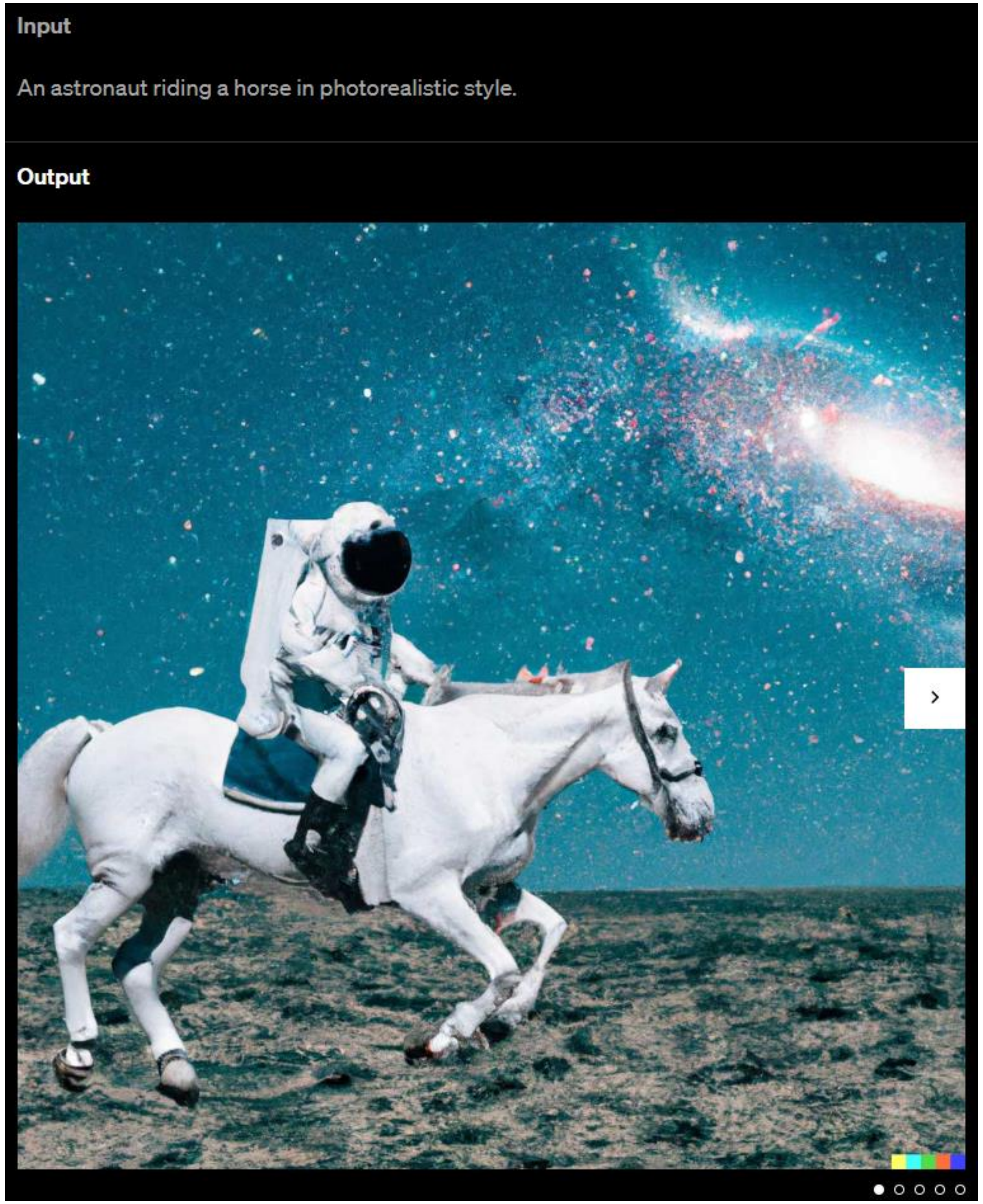
Computer Vision

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

example: DALL-E 2



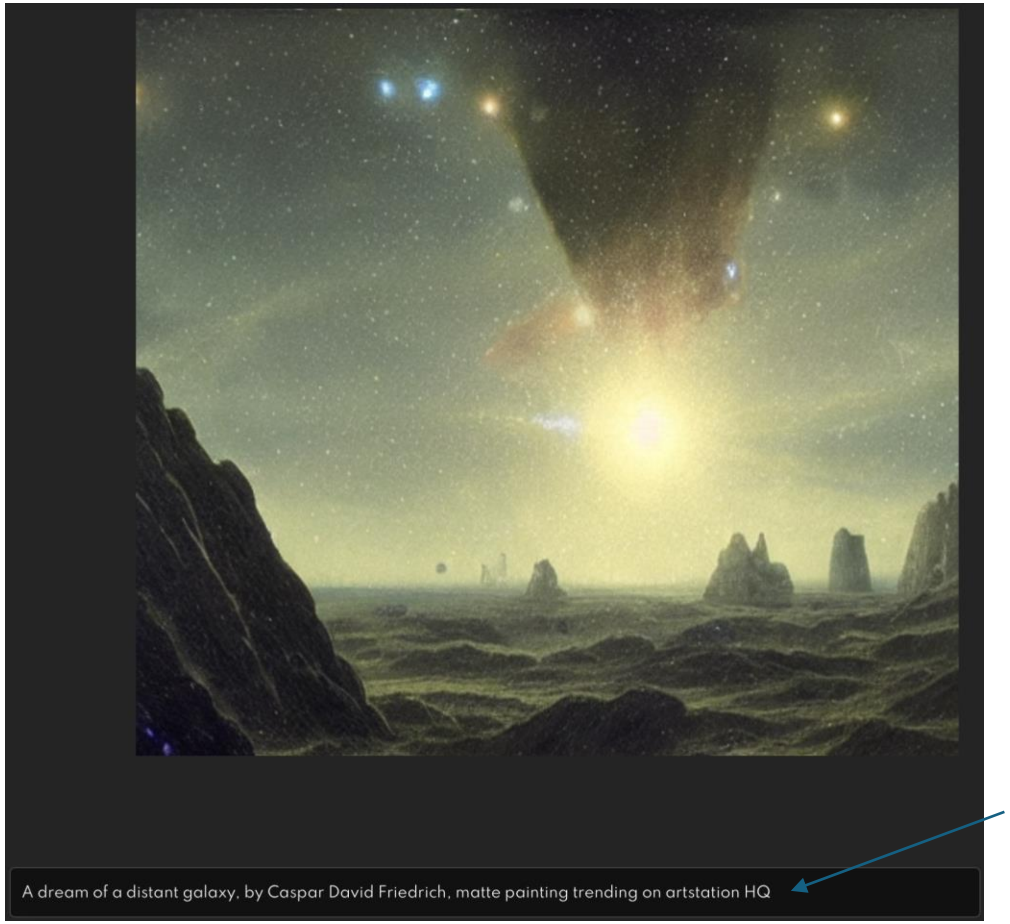
Input

An astronaut riding a horse in photorealistic style.

Output

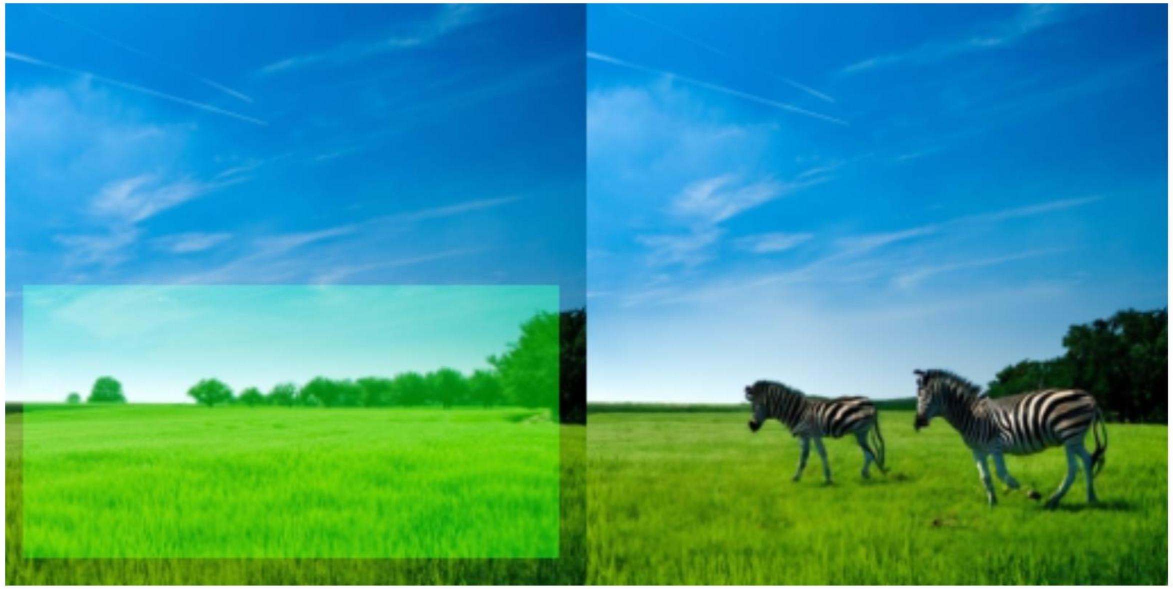
plenty of products: DALL-E, Stable Diffusion, ImageGen, Midjourney, ...

web app for Stable Diffusion: DreamStudio



A dream of a distant galaxy, by Caspar David Friedrich, matte painting trending on artstation HQ

inpainting example (GLIDE):



prompt

“zebras roaming in the field”

source

Generative vs Predictive/Discriminative Models

discriminative models:

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

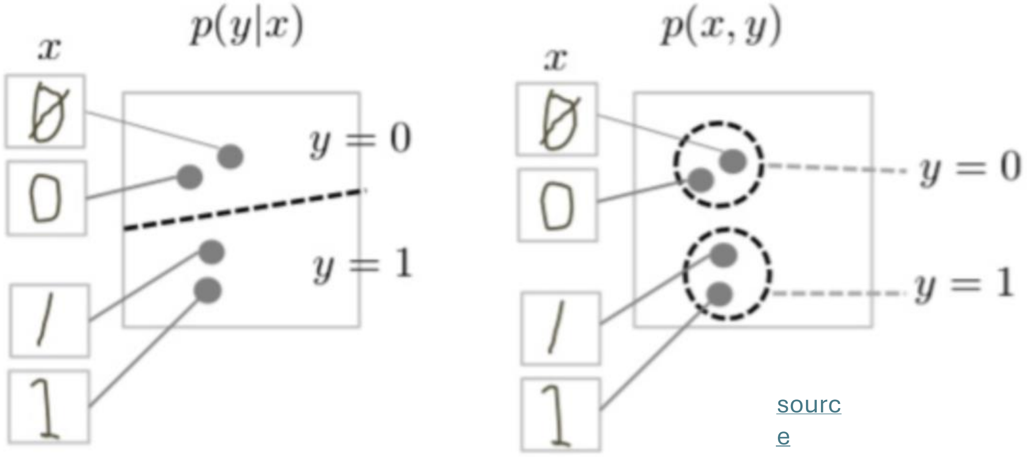
generative models:

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

(or just $P(\boldsymbol{X})$ → unsupervised learning)

→ allow to generate new data samples

discriminative model generative model



source

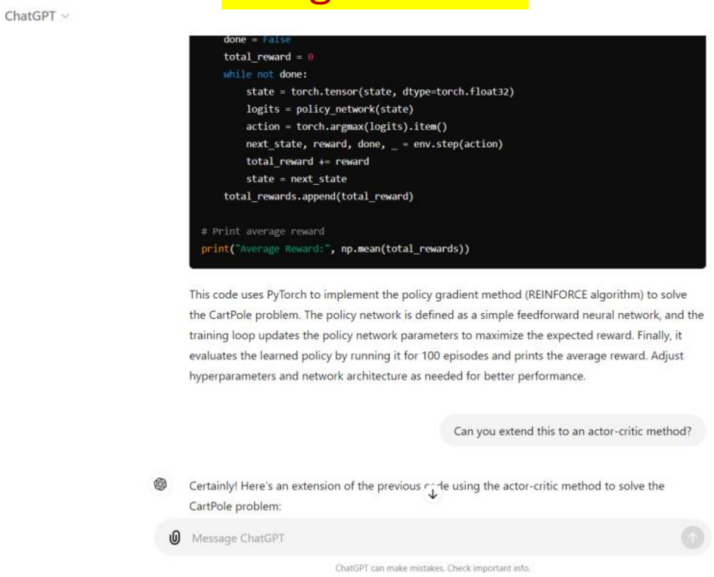
task of generative models more difficult: need to model full data distribution rather than merely find patterns in inputs to distinguish outputs

Generative models can be used for predictive tasks (Bayes theorem).
But predictive models are usually better at it.

Classic ML vs Deep Learning

Generative vs Predictive Models

text generation



```
    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:

ChatGPT can make mistakes. Check important info.

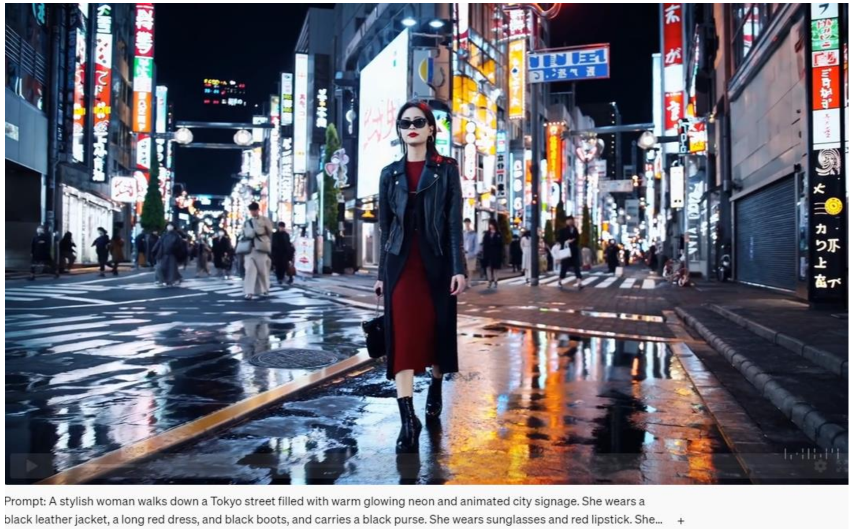
ChatGPT

image synthesis

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

text-to-video

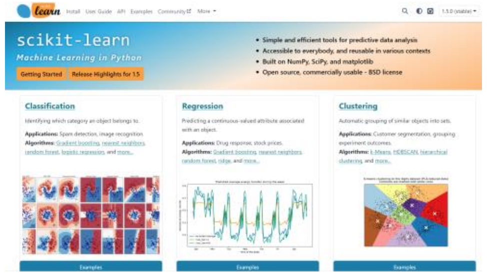


Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She...

Sora | OpenAI

BERT family

tabular data



scikit-learn

Machine Learning in Python

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

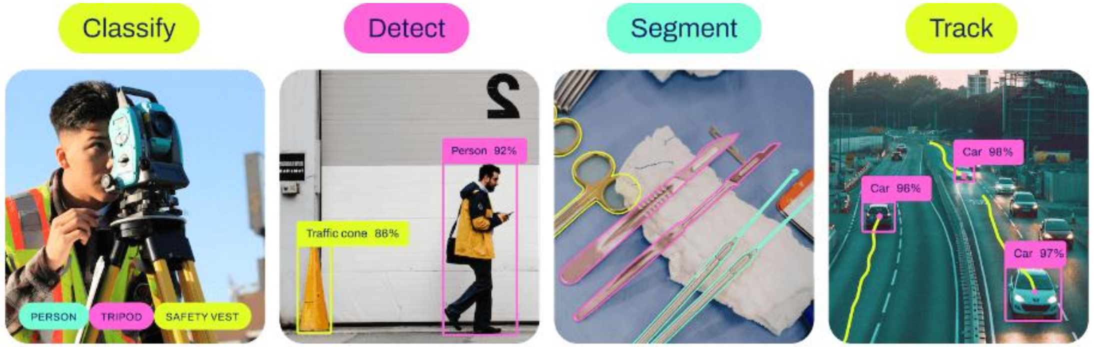
Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, grouping experiment outcomes.
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

dmlc XGBoost

computer vision



YOLO

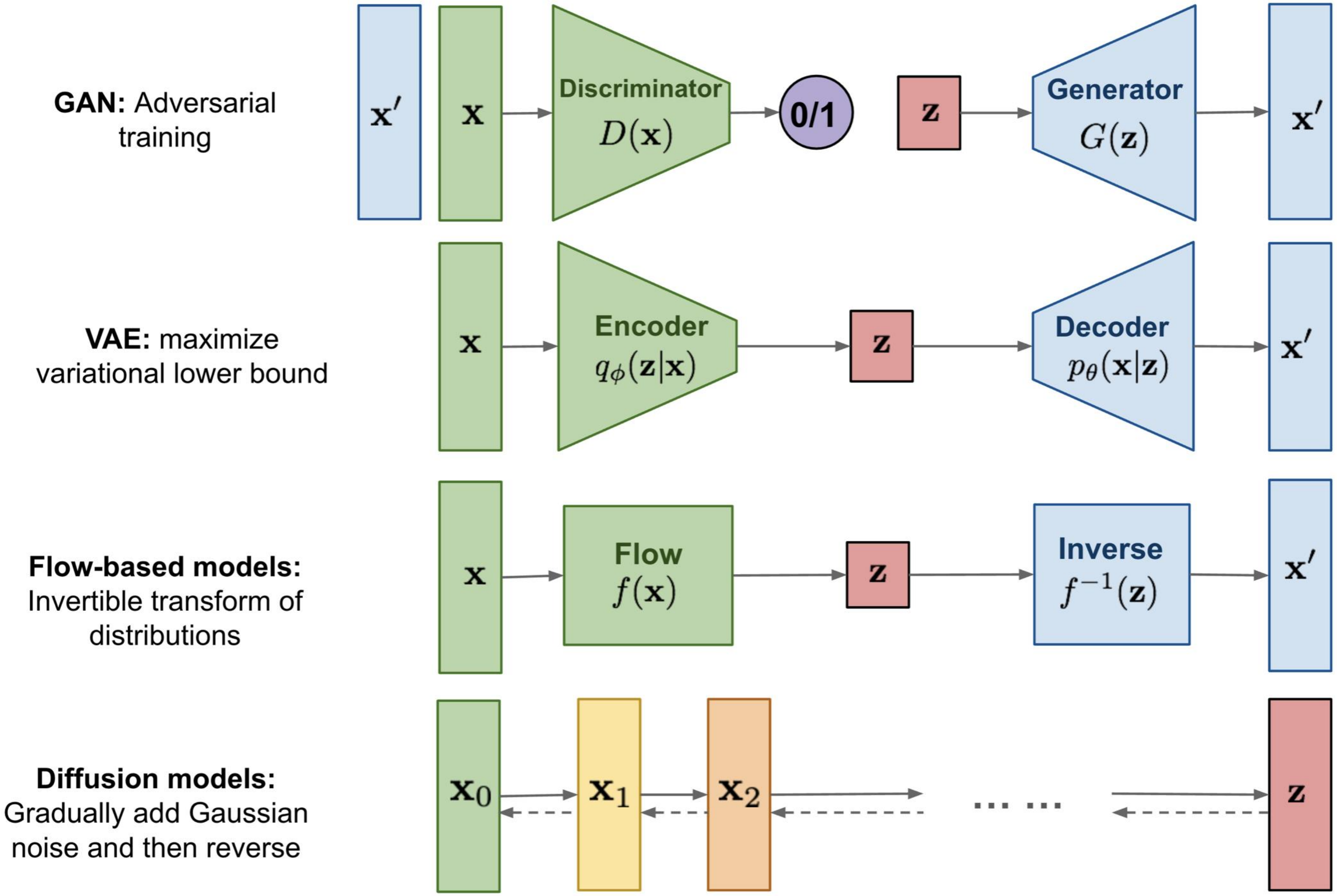
Deep Learning for Generative AI

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



GAN: Adversarial training

two neural networks playing a zero-sum game

VAE: maximize variational lower bound

learn variational distribution (not just replicating inputs)

Flow-based models: Invertible transform of distributions

more complex distributions by applying change-of-variable technique (need for specialized architecture)

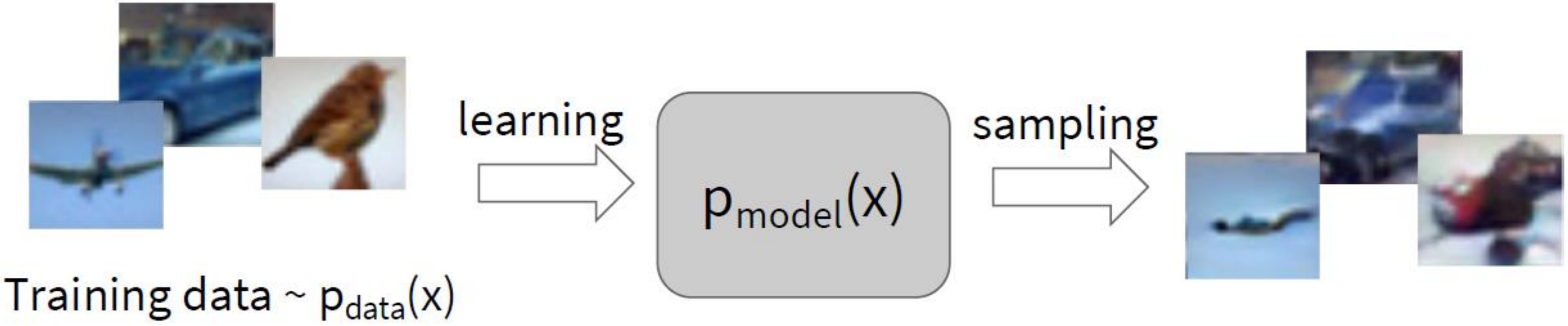
Diffusion models: Gradually add Gaussian noise and then reverse

chain of denoising autoencoders

source

→ generalization: flow matching

Generative Modeling



learning

$p_{model}(x)$

sampling

Training data ~ $p_{data}(x)$

Objectives:

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

Explicit density estimation: explicitly define and solve for $p_{model}(x)$
Implicit density estimation: learn model that can sample from $p_{model}(x)$ without explicitly defining it.

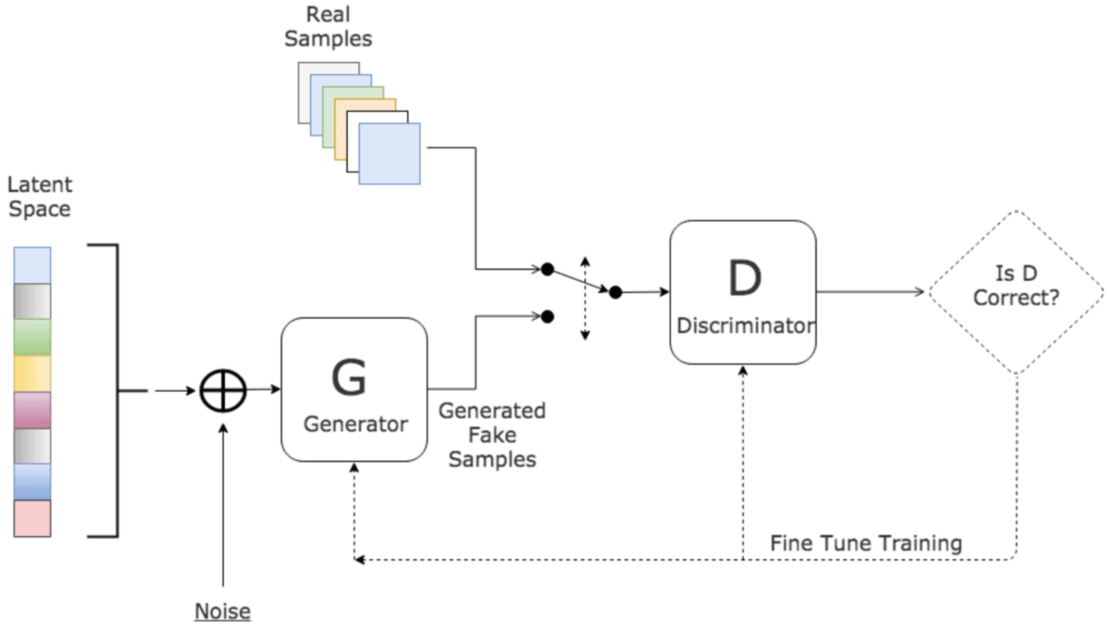
Generative Adversarial Networks (GAN)

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

implicit density estimation



source

common loss for generator and discriminator:

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

G trying to minimize

D trying to maximize

Conditional GANs

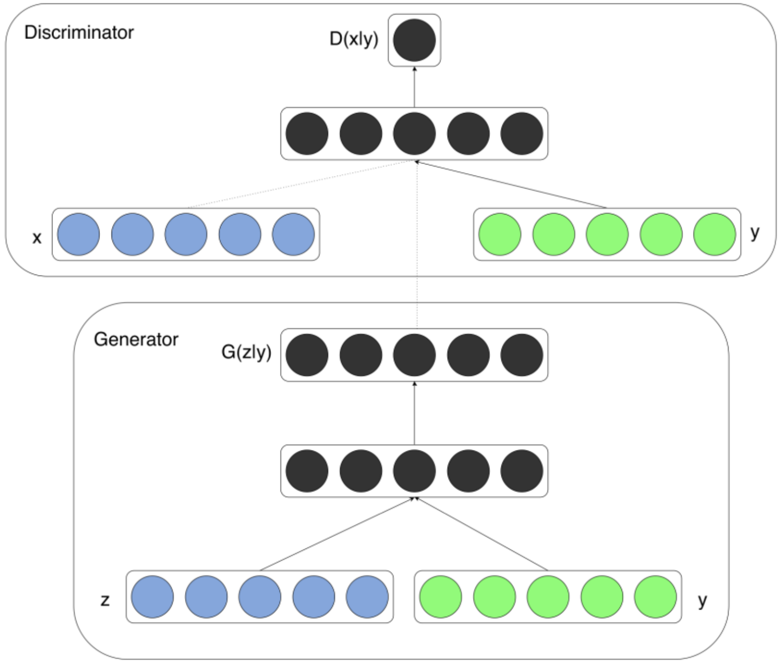
as discussed so far, generative methods give no control over what kind of data is generated (limited usability)

→ need for conditional approach (e.g., conditioning on describing text)

example GANs:

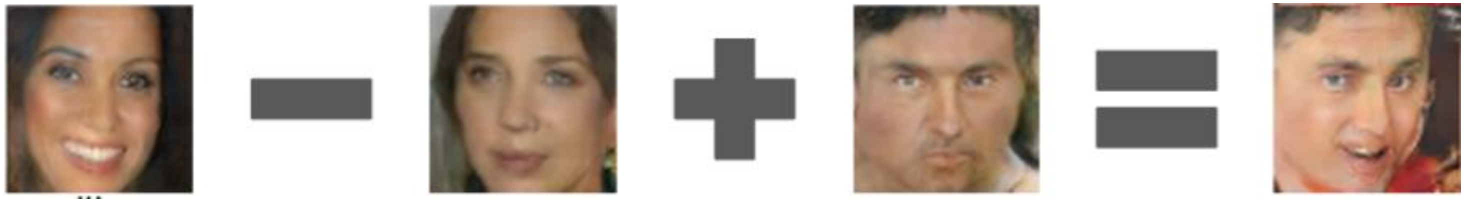
transform usual GAN to conditional model by feeding extra information y (e.g., class labels) as additional input layer into both generator and discriminator

$$L(\boldsymbol{x}_i) = E_{\boldsymbol{x}\sim p_r(\boldsymbol{x})}[\ln D(\boldsymbol{x}_i|y_i)] + E_{\boldsymbol{x}\sim p_g(\boldsymbol{x})}\left[\ln\left(1 - D(\boldsymbol{x}_i|y_i)\right)\right]$$



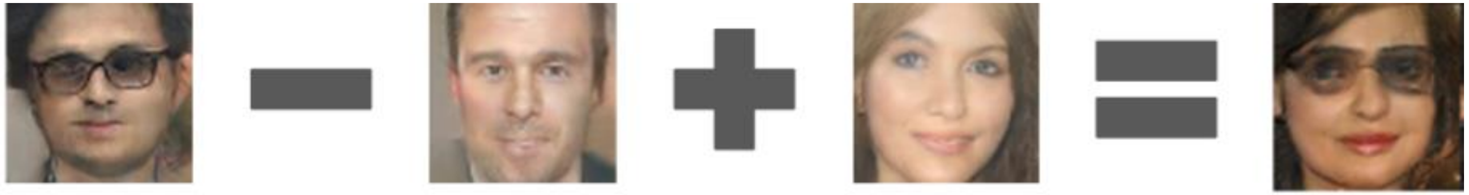
source

Vector Arithmetic in GAN Latent Space



smiling woman neutral woman neutral man smiling man

source



man with glasses man without glasses woman without glasses woman with glasses

Variational Autoencoder (VAE)

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

→ learn variational distribution instead of identity function

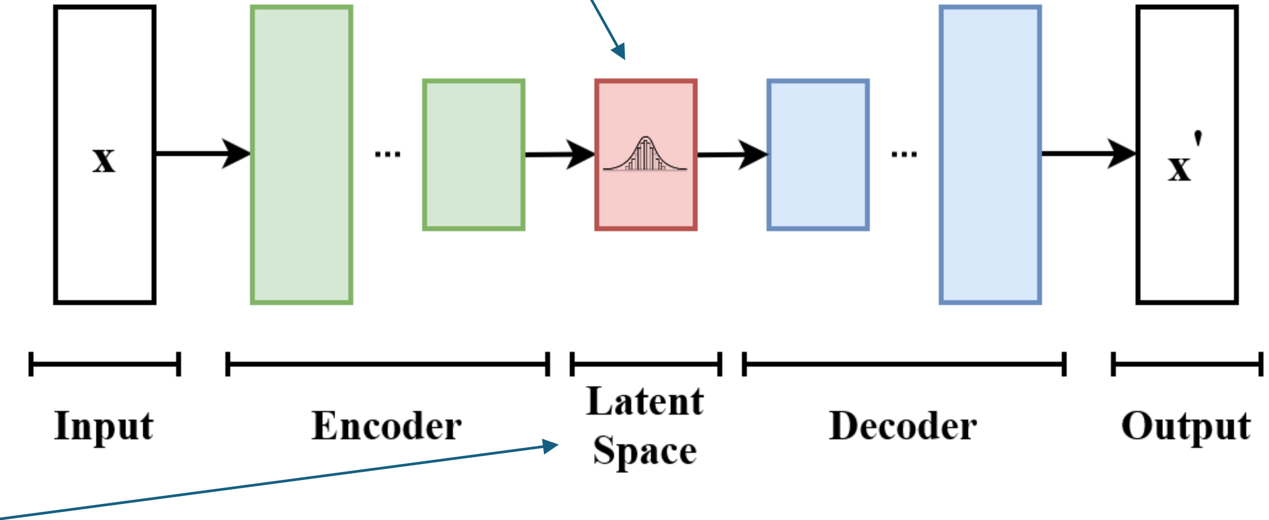
to be precise: parametrized variational distribution of latent encoding variables $\boldsymbol{z}$

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

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

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

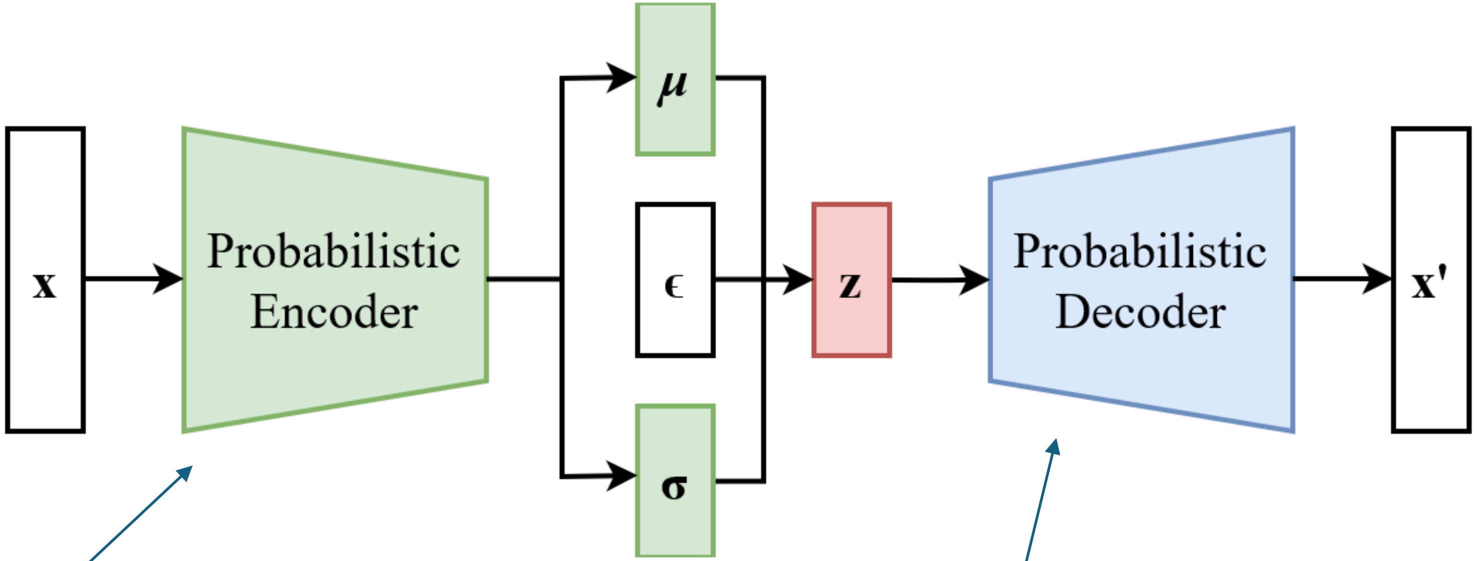
from which to sample



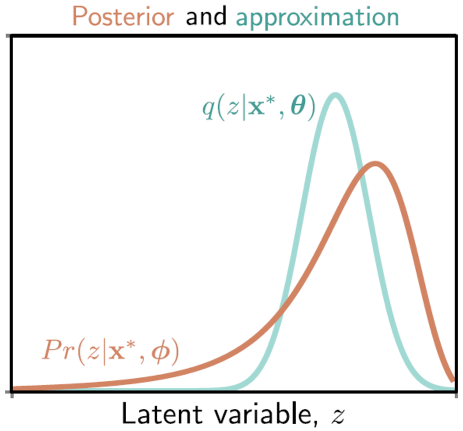
Variational Bayesian Method

Gaussian Approximation

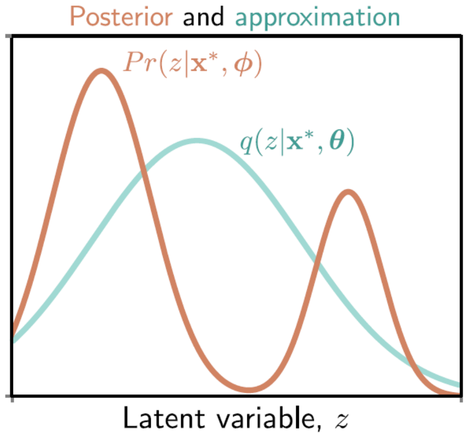
learn mean and variance of multivariate Gaussian with diagonal covariance structure

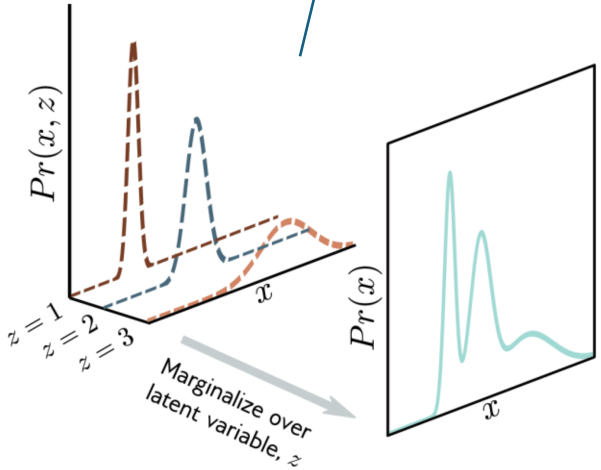


good approximation:



poor approximation:



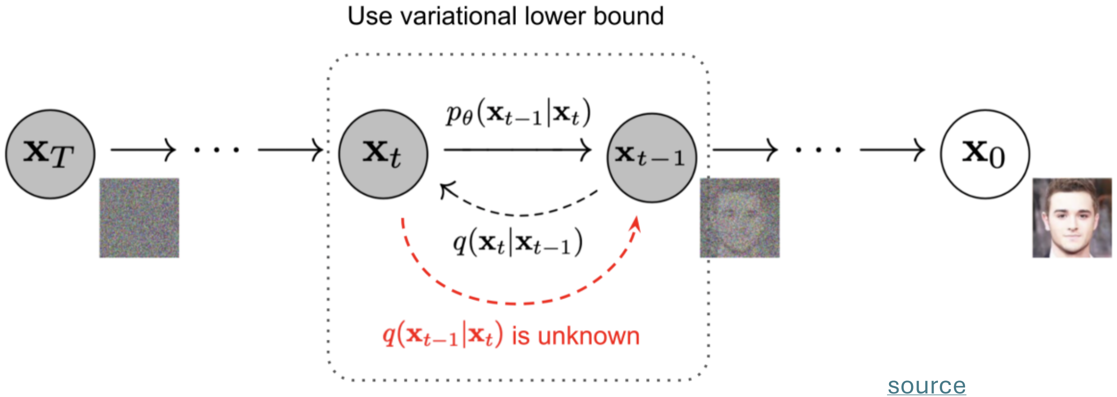


source

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



Use variational lower bound
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is unknown

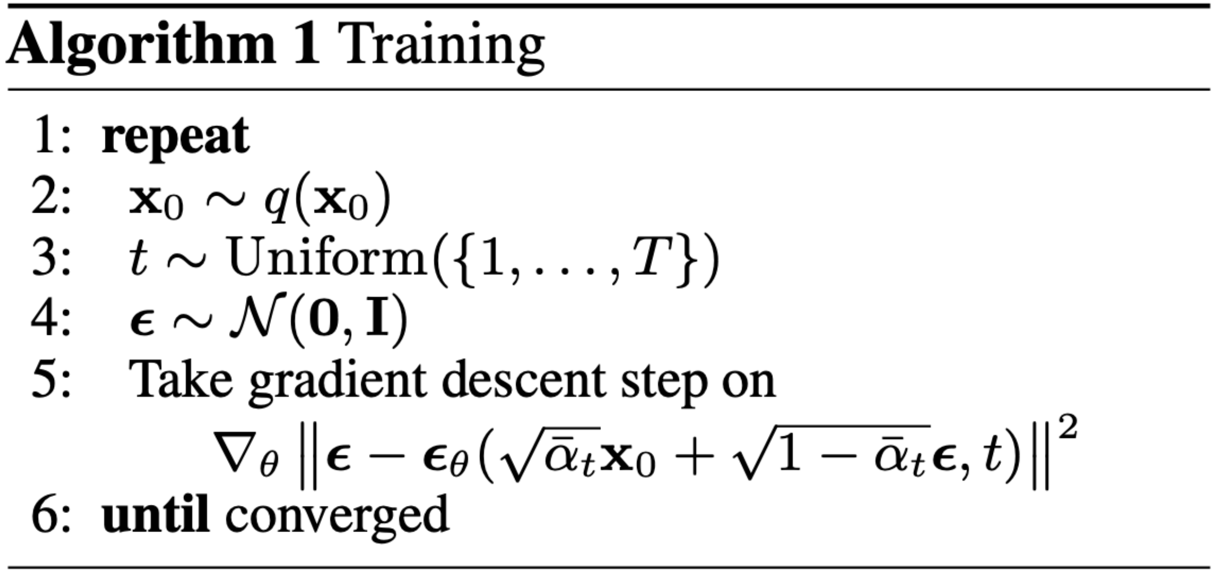
source

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

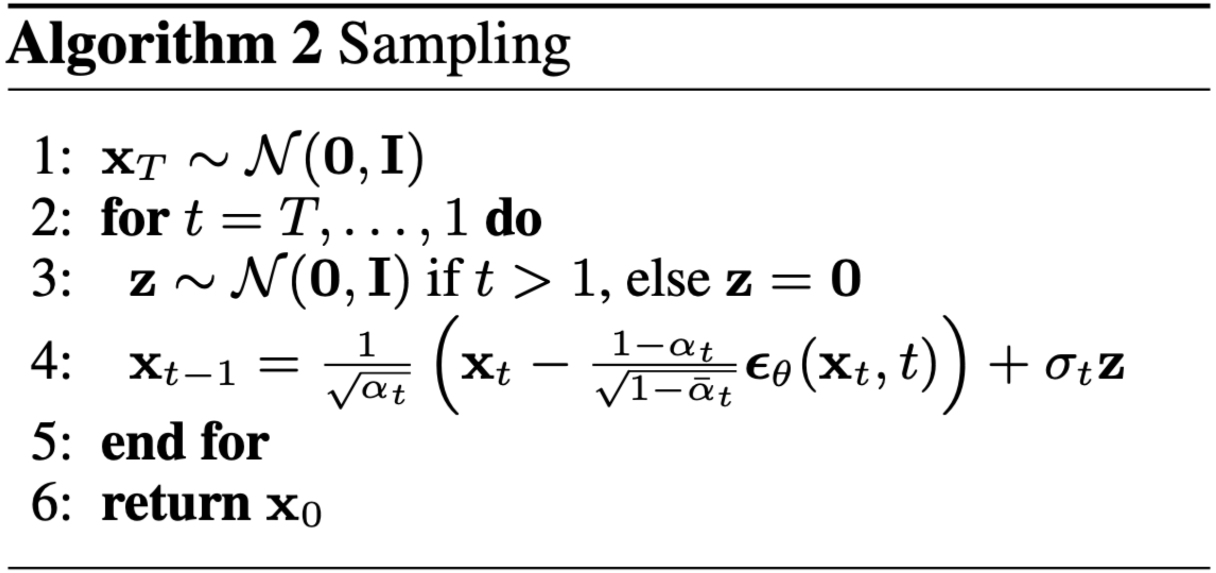
→ more sophisticated latent space than just Gaussian mixture in VAE

Noise Prediction



Algorithm 1 Training

```
repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
      $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
until converged
```



Algorithm 2 Sampling

```
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ do
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$
```

source

Diffusion as Chain of Denoising Autoencoders

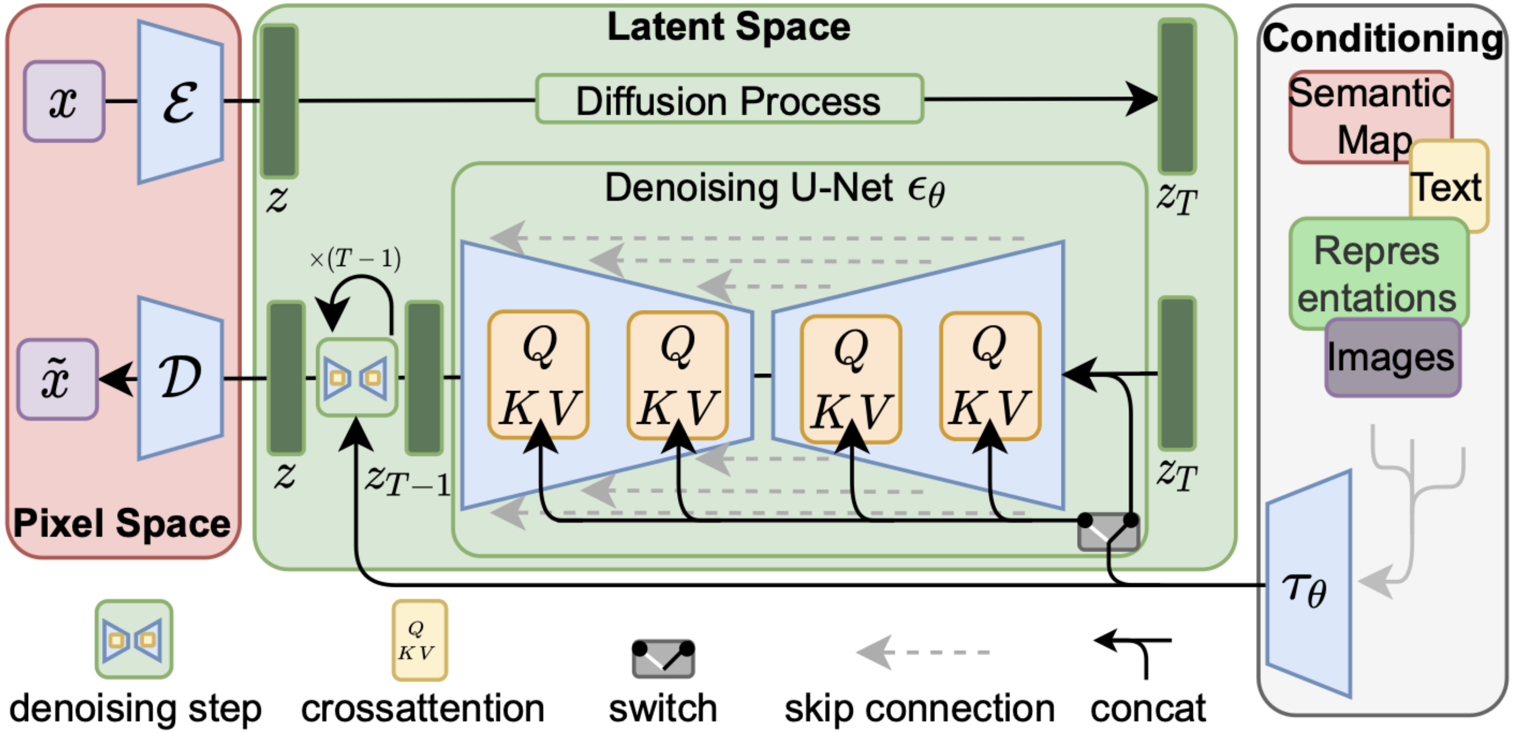
differences of diffusion models to typical denoising autoencoders:

- 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



source

- 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)

Guided Diffusion

condition diffusion process on class information (label or just text)

$$\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t|\emptyset) + s \cdot \left(\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t|y) - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t|\emptyset)\right)$$

s: hyperparameter to control tradeoff between diversity (unconditioned) and fidelity (guidance)

similar idea as softmax temperature in auto-regressive LLMs

“Pembroke Welsh corgi”

source

Outlook: Text2Anything

next step: text-to-video (Make-A-Video, Lumiere, Sora, ...)
→ rudimentary physics understanding

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