Strojové učení II

Generative Models

Supervised vs Unsupervised Learning

Supervised

- Data: (x,y)

  x…data, y…label

- Goal: learn function to map

$$x \to y$$

- Examples: classification, regression, object detection, semantic segmentation, ...

Unsupervised

- Data: x

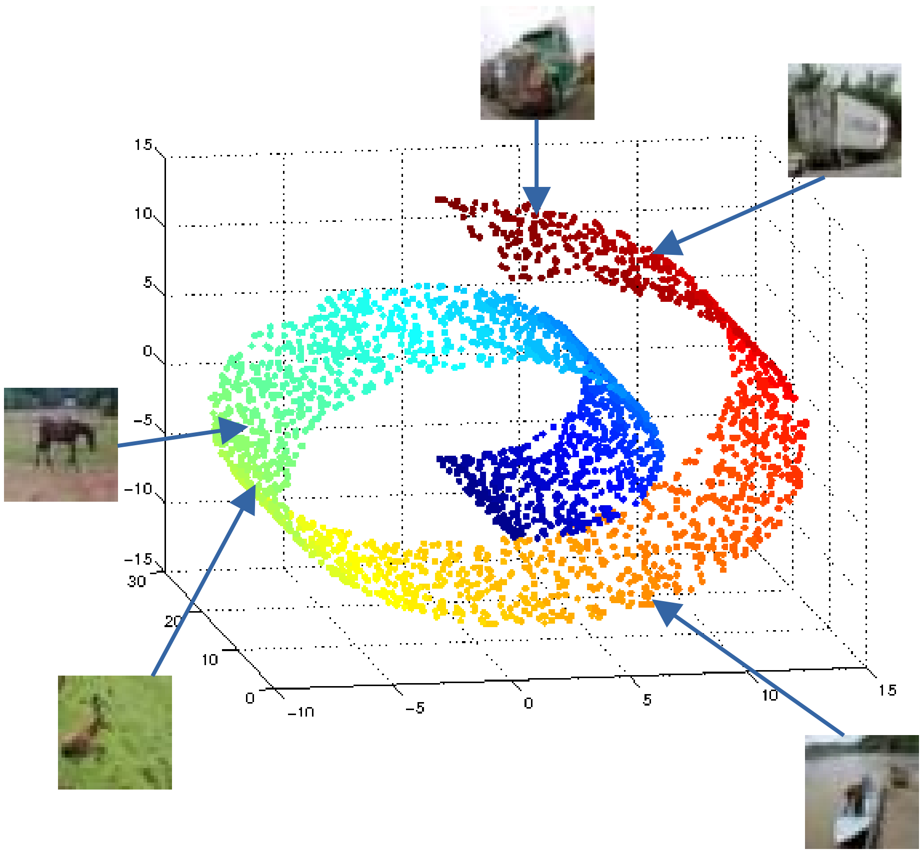
  x...data, no labels!

- Goal: learn hidden (underlying) structure of the data

- Examples: clustering, feature or dimensionality reduction, …

Generative Modeling

- Natural images lie on a manifold



Input samples

$$p_{\text{data}}(x)$$

- How to sample from this distribution?

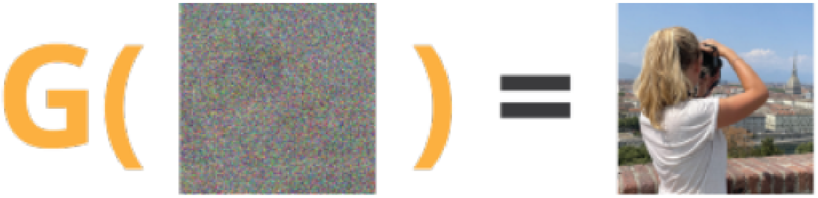
Generative Modeling

- Take training samples from the data distribution and learn a mapping from a simple distribution (e.g. normal) to the data distribution such that

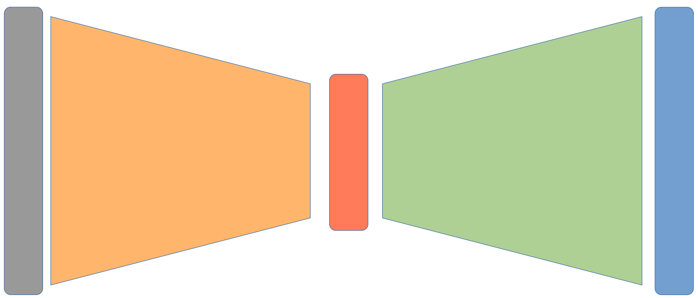
$\simeq$

Input samples
$p_{\text{data}}(x)$

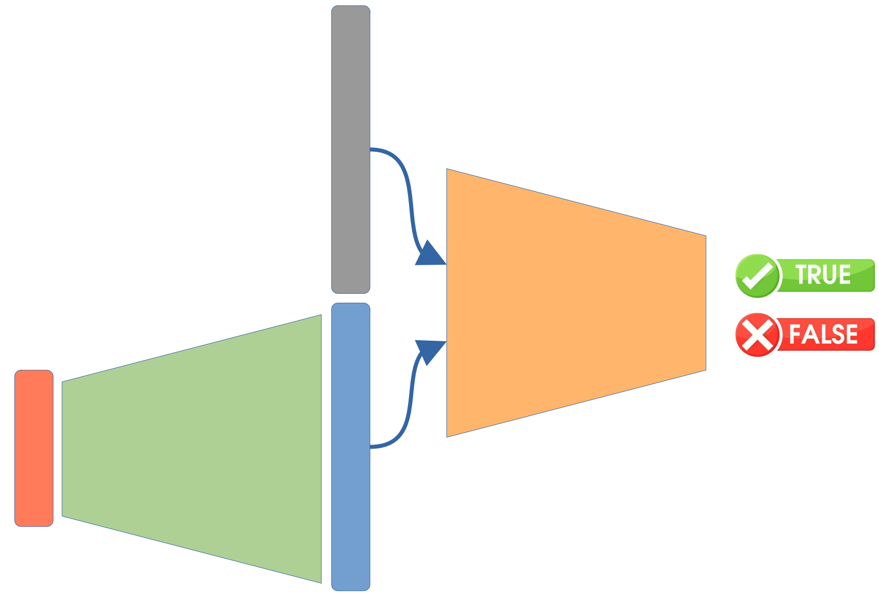
Generated samples
$p_{\text{model}}(x)$



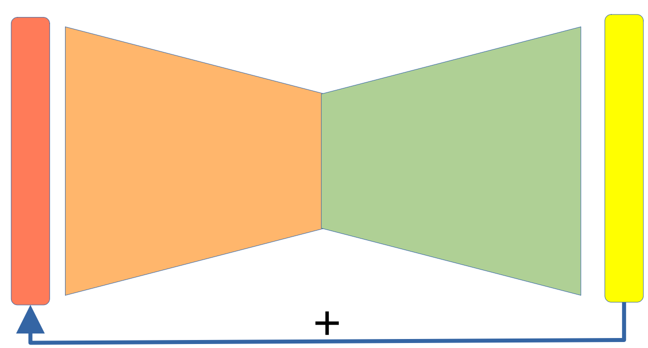
Generative Models



Autoencoders, Variational
Autoencoders (VAEs)



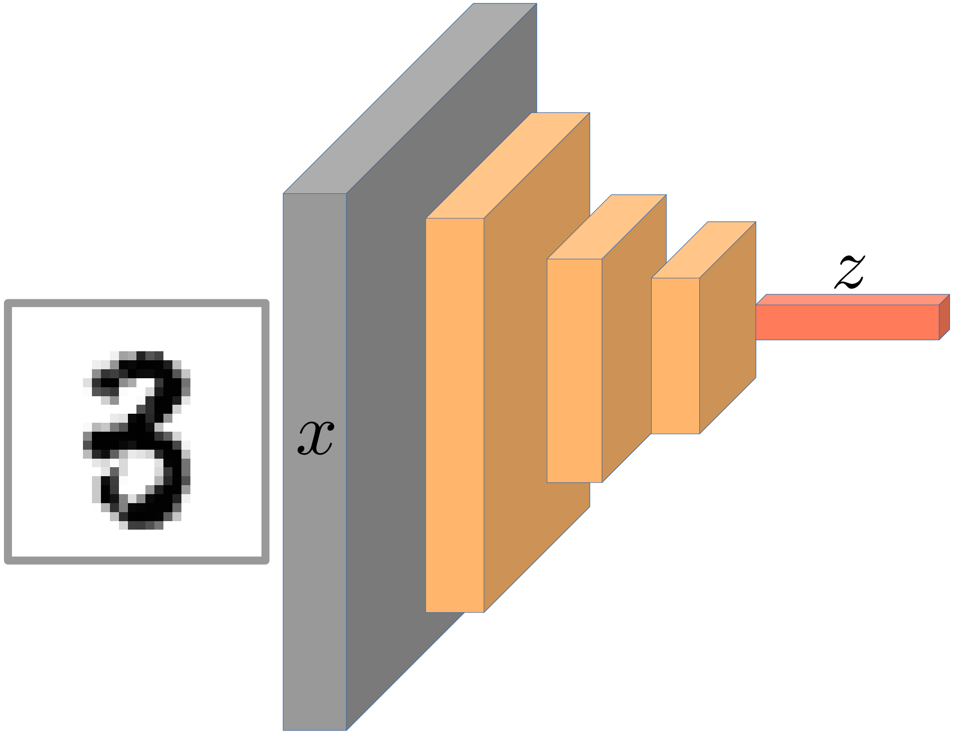
Generative Adversarial
Networks (GANs)



Diffusion Models

Autoencoders: background

- Unsupervised learning of a lower-dimensional feature representation from unlabeled data



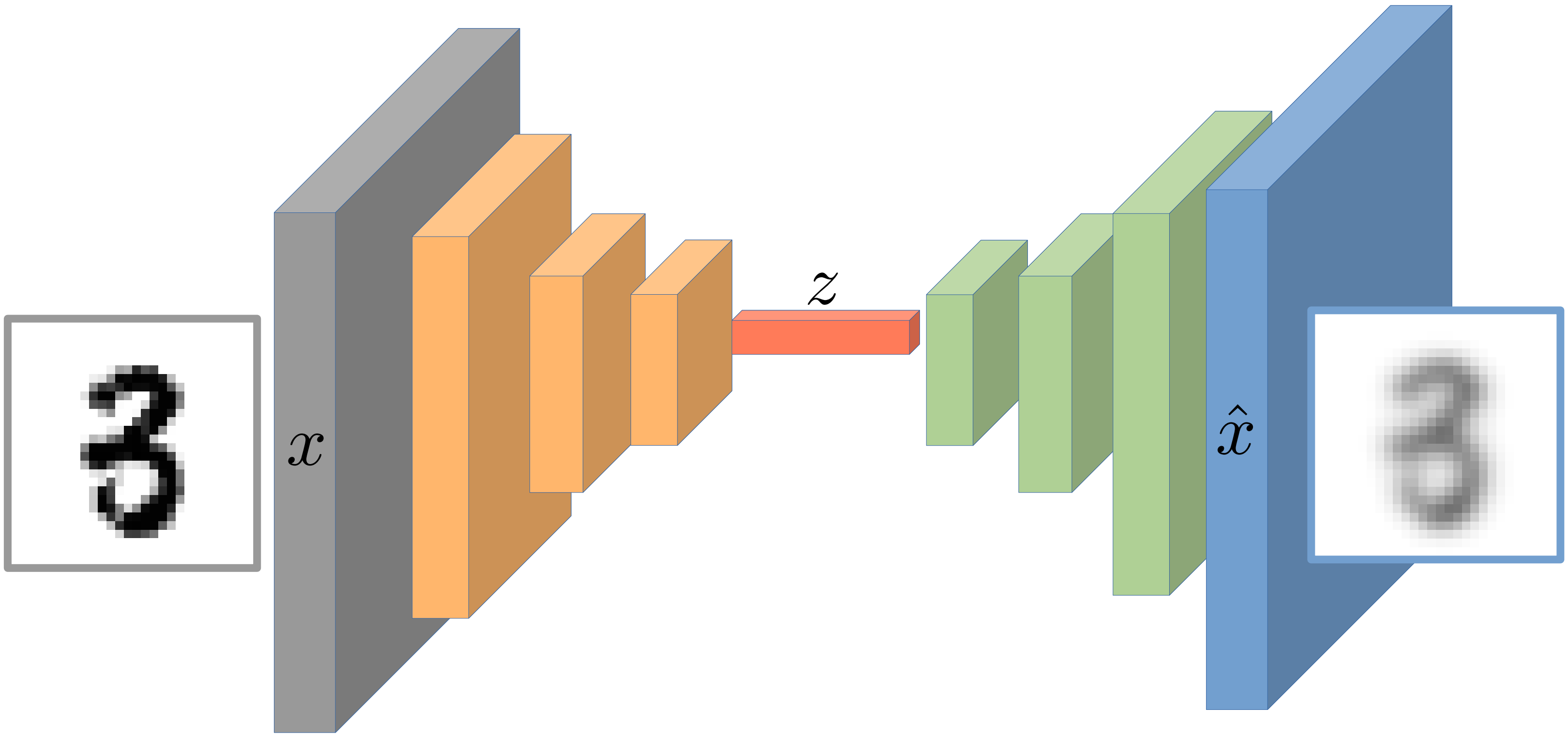
Why we want a low-dimensional latent space?

- **Encoder** learns mapping $f(x, \phi) : x \rightarrow z$

  where z is low-dim. latent space

Autoencoders: background

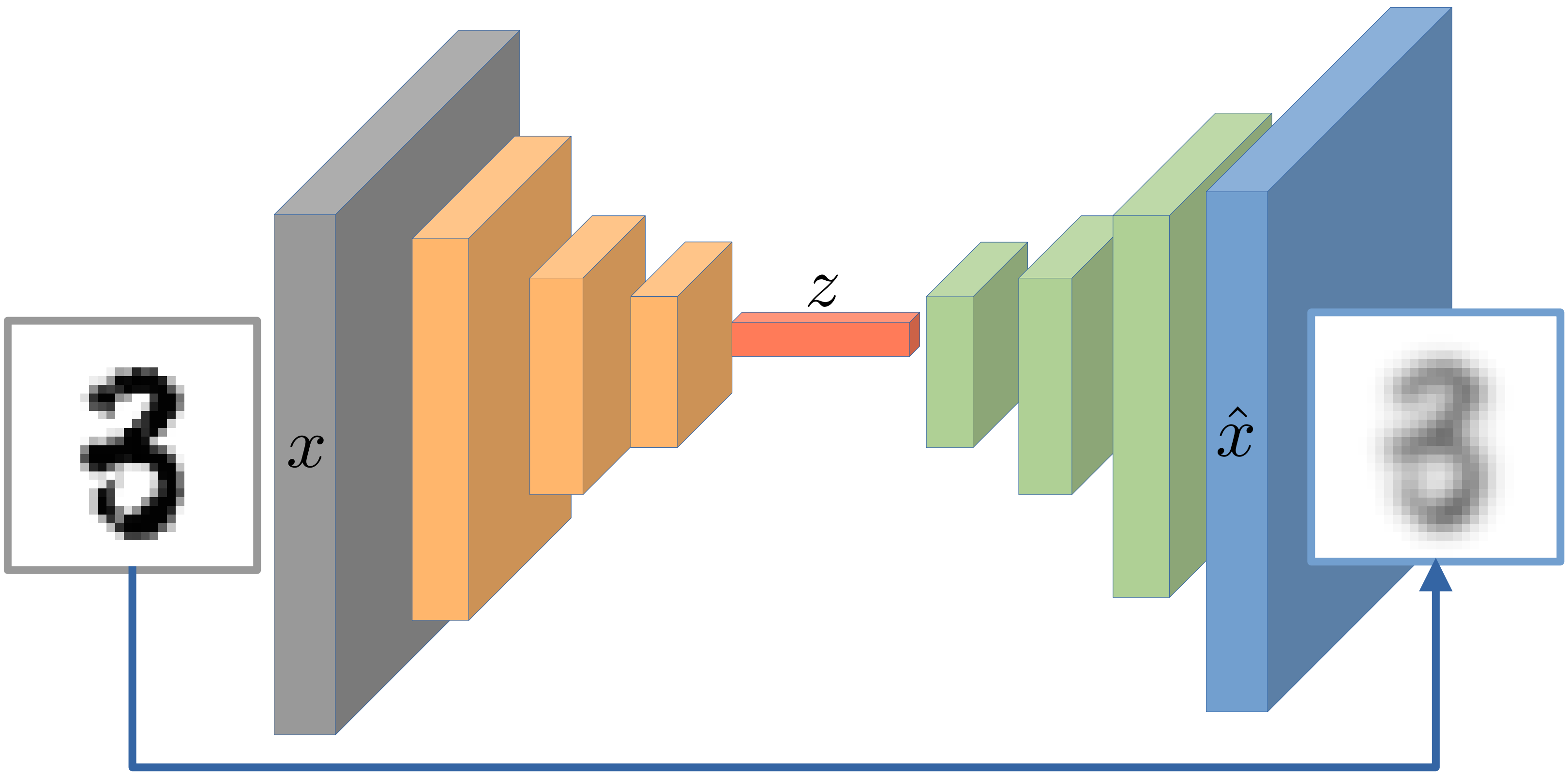
- How can we learn the latent space?



- **Decoder** learns mapping $g(z, \theta) : z \to \hat{x}$
  from the latent space *z* to a reconstructed observation $\hat{x}$

Autoencoders: background

- Train the model to use *z* to reconstruct the original *x*



$$L(x) = \|x - \hat{x}\|^2 = \|x - g(f(x))\|^2$$

- Loss is without labels

$$\min_{\phi,\psi} \sum_i L(x^{(i)})$$

Dimensionality of latent space

- Autoencoding is a form of compression

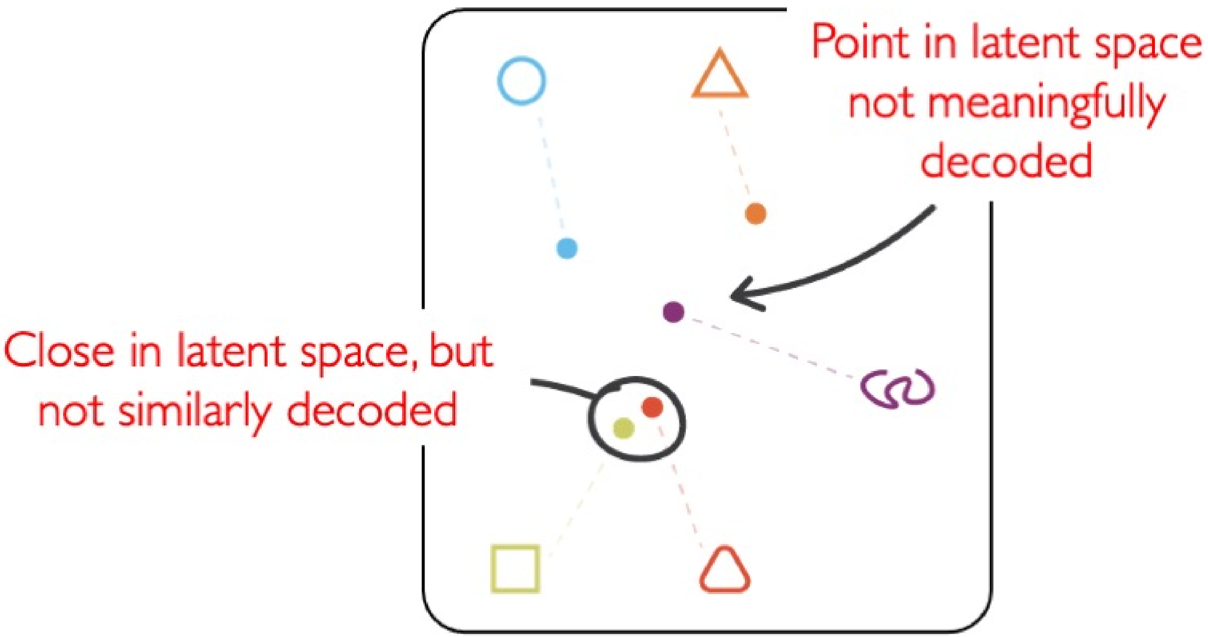
2D latent space

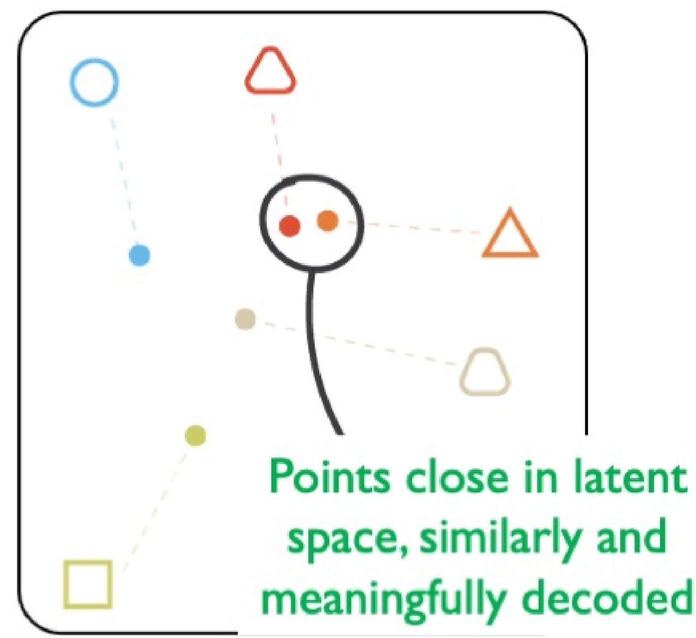
5D latent space

GT

Obstacles

- AE easily overfits and encoding in the latent space is meaningless.
- Encoder and Decoder with sufficient capacity can learn even for the 1D latent space z.





Regularization on z

MIT course, Introduction to deep learning

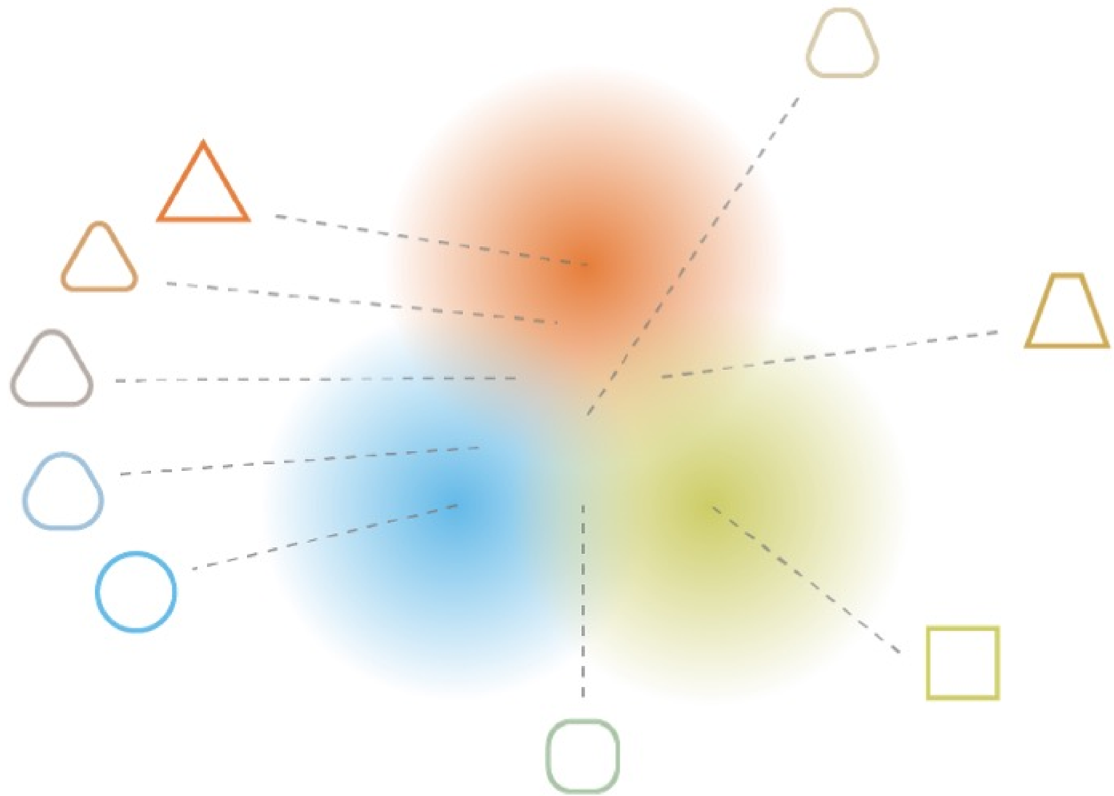
Regularization

- Continuity:

  points close in latent space - > similar content after decoding

- Completeness:

  sampling from latent space - > “meaningful” content after decoding



MIT course, Introduction to deep learning

Regularization

- Contractive autoencoder (CAE)

$$L(x) = \|x - g(f(x))\|^2 + \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2$$

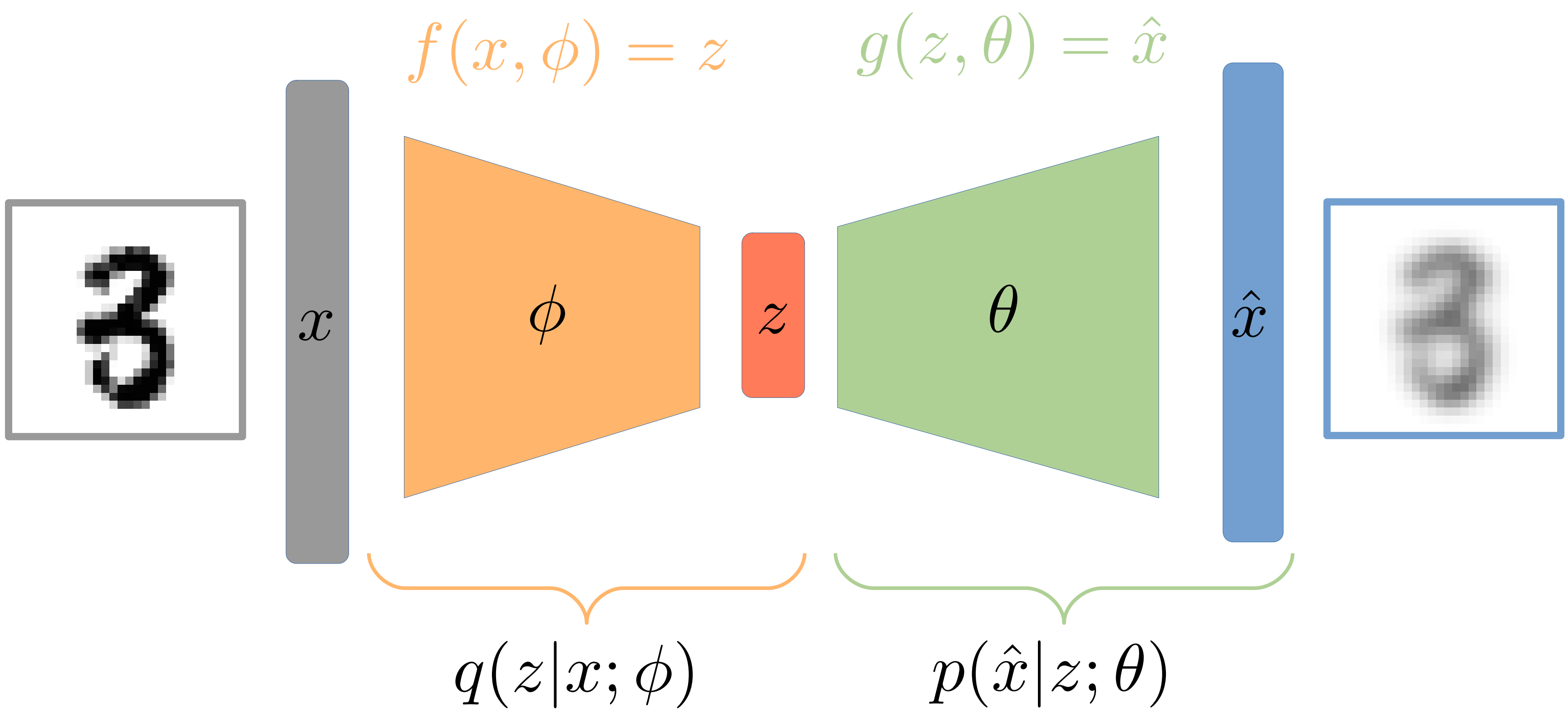
- Denoising autoencoder (DAE)

$$L(x) = \|x - g(f(x + \epsilon))\|^2$$

- Variational autoencoder (VAE)

Variational Autoencoder

- Traditional autoencoder

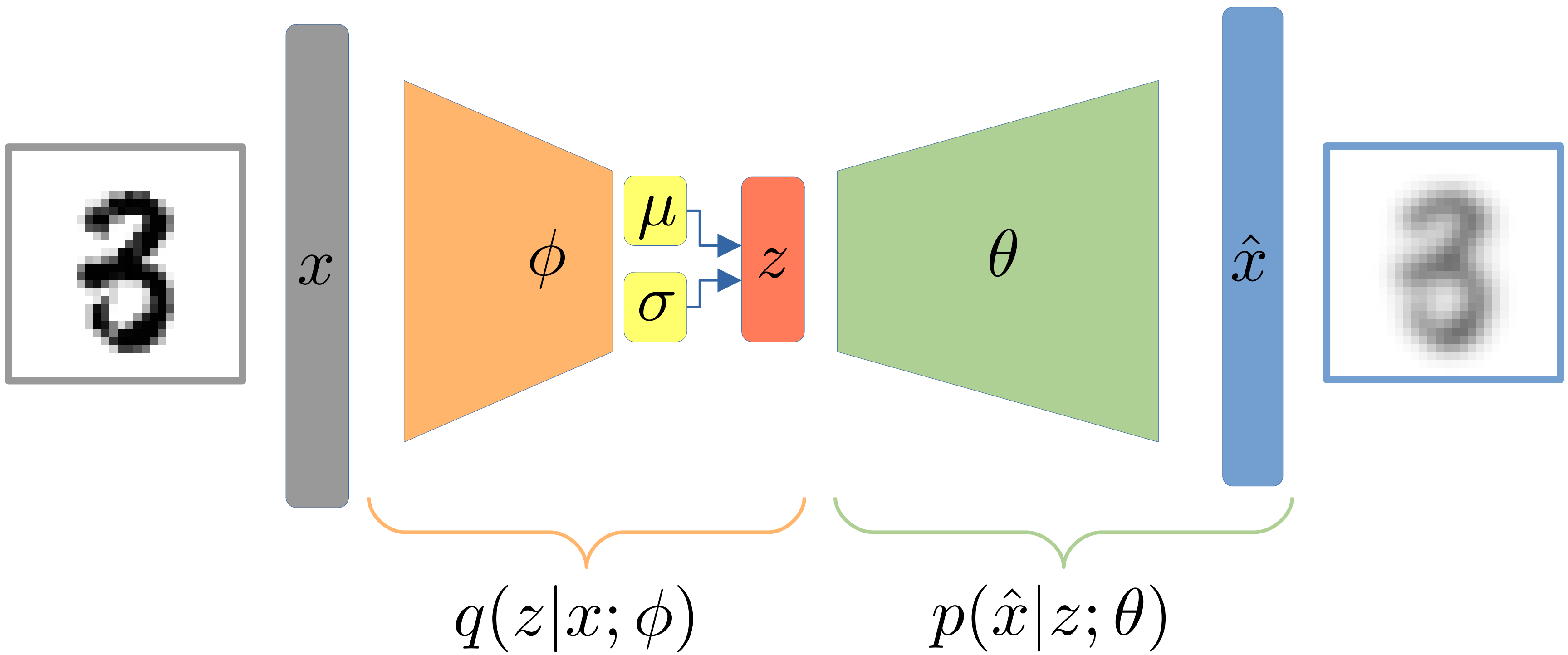


- Better think of probabilities –> AE learns the means:

$$f(x) = \mathbb{E}_{q(z|x)}[z] \qquad g(z) = \mathbb{E}_{p(\hat{x}|z)}[\hat{x}]$$

Variational Autoencoder

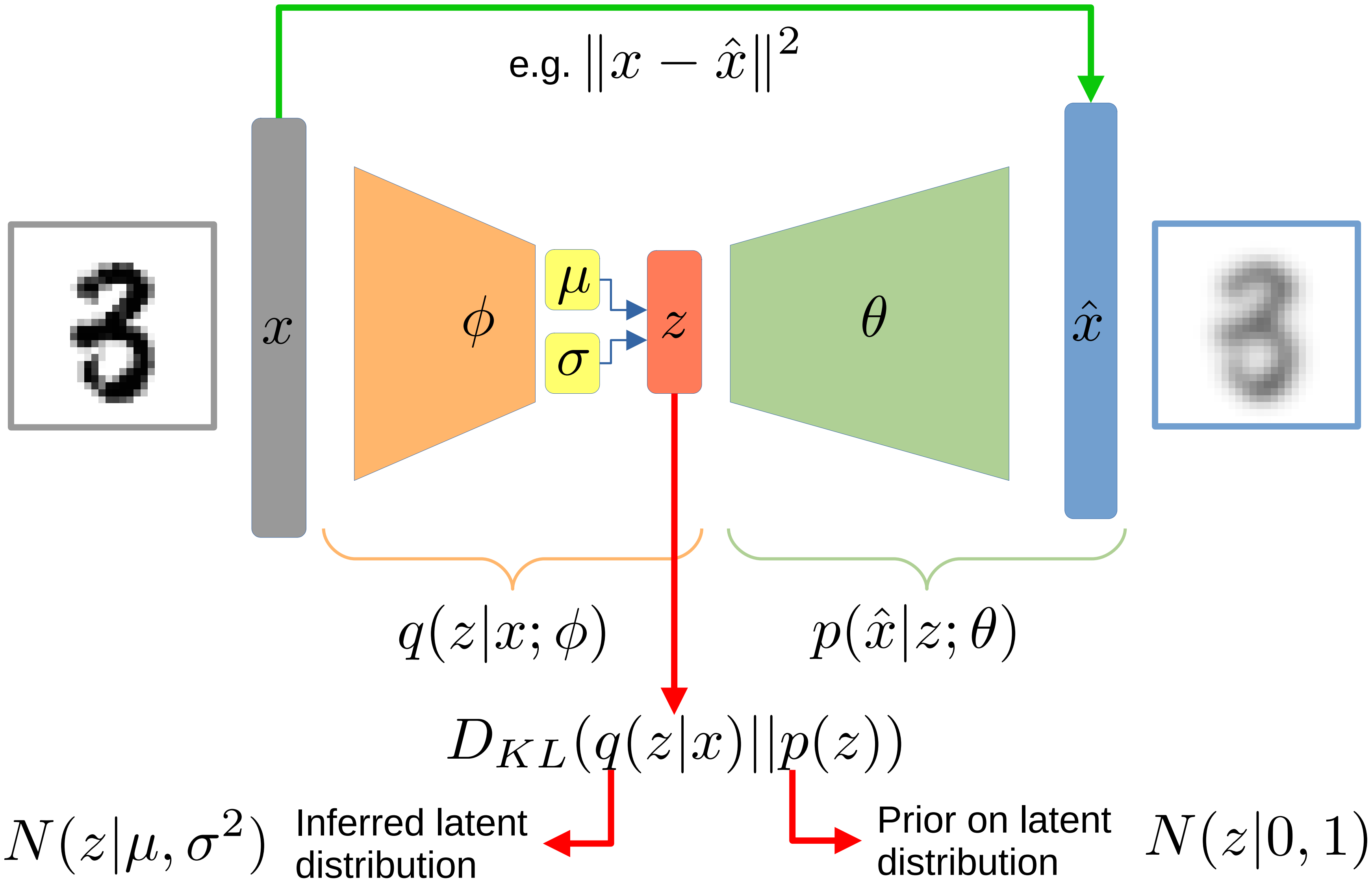
- VAE approximates $q(z) \approx N(z|\mu, \sigma^2)$ and learns both mean and standard deviation vectors



- Latent variable *z* is sampled from estimated $q(z|x;\phi)$

Variational Autoencoder

- Loss: $L(x, \phi, \theta) = \text{reconstruction loss} + \text{regularization term}$



Priors on the latent distribution

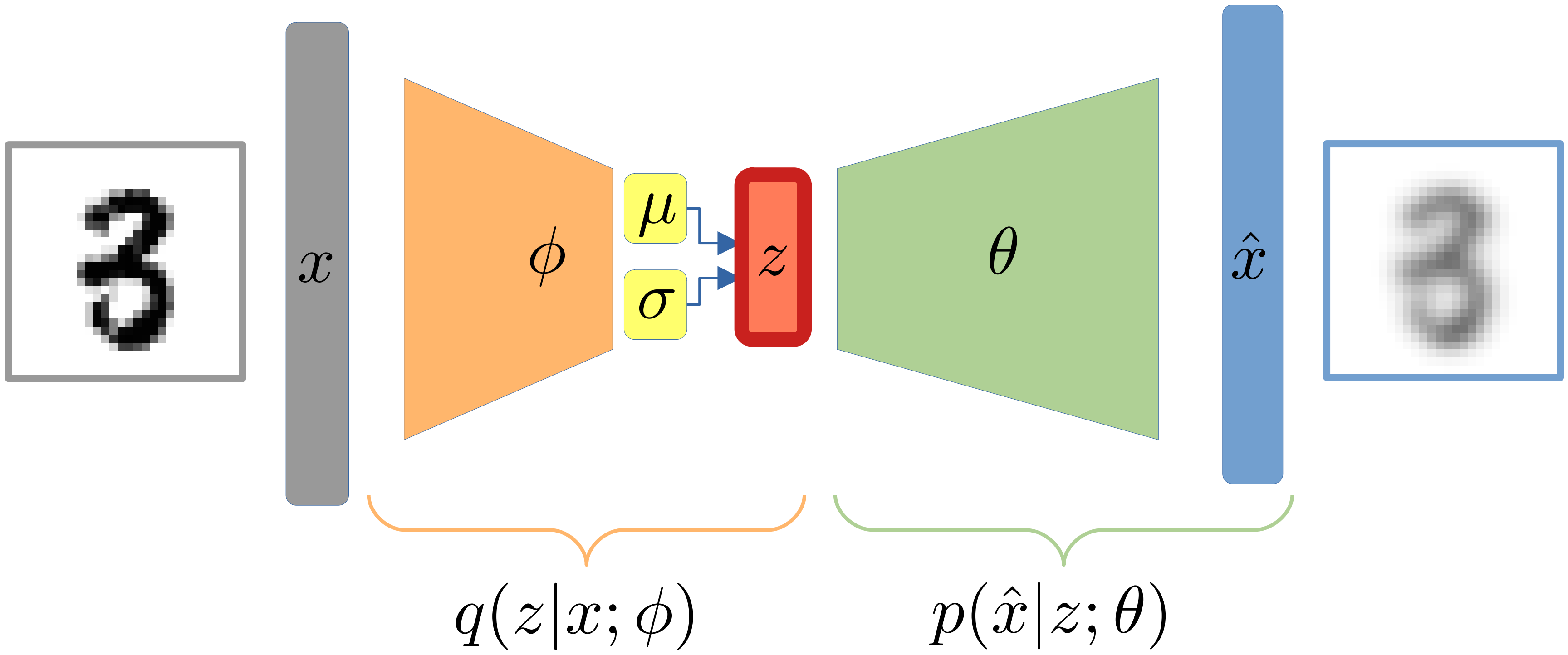
$$D_{KL}(q(z|x)||p(z))$$

- q and p are normal distributions --> KL divergence has closed-form expression

$$D_{KL}(N(z|\mu,\sigma^2)||N(z|0,1)) = \frac{1}{2}(\mu^2 + \sigma^2 - 1 - \log\sigma^2)$$

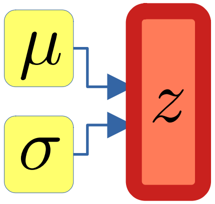
VAE Computation Graph

- We cannot use backpropagation if *z* is sampled.

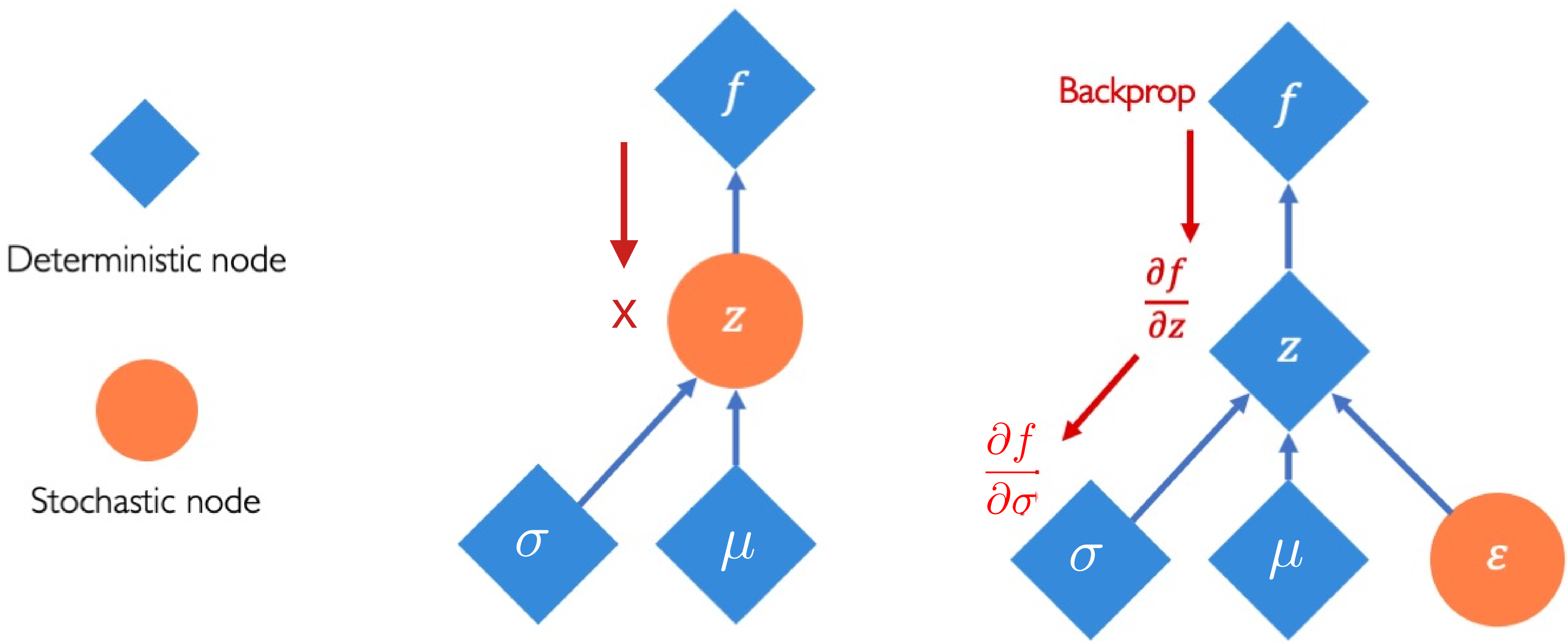


Reparametrization Trick

- Sample an auxiliary variable $\epsilon \sim N(0, 1)$
- Calculate: $z = \mu + \sigma\epsilon$
- Then $z \sim N(\mu, \sigma)$



Reparametrization Trick



Original form

Reparametrized form

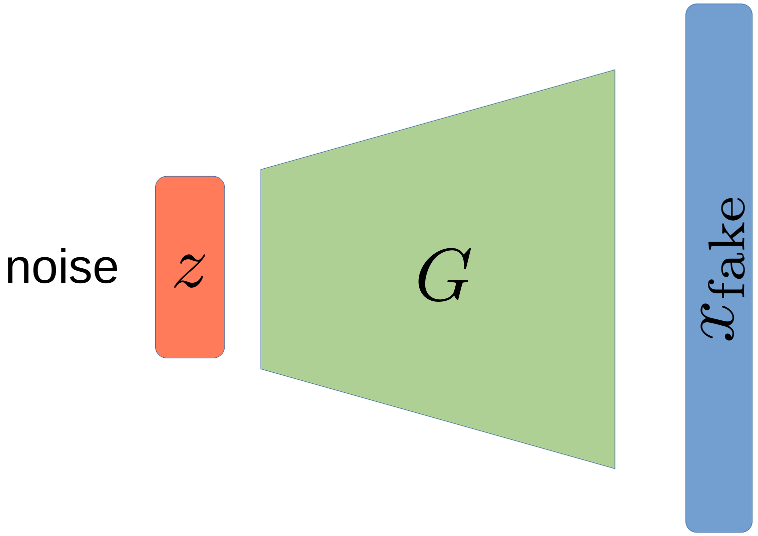
MIT course, Introduction to deep learning

VAE Latent Perturbation

Kingma et al., VAE, 2014

Generative Adversarial Networks - GANs

- What if we just want to sample?
- Idea: Don't explicitly model density, just sample
- Solution: Sample from something simple (white noise) and learn a transformation to the data distribution.



$$x_{\text{fake}} \sim p_{\text{data}}(x)$$

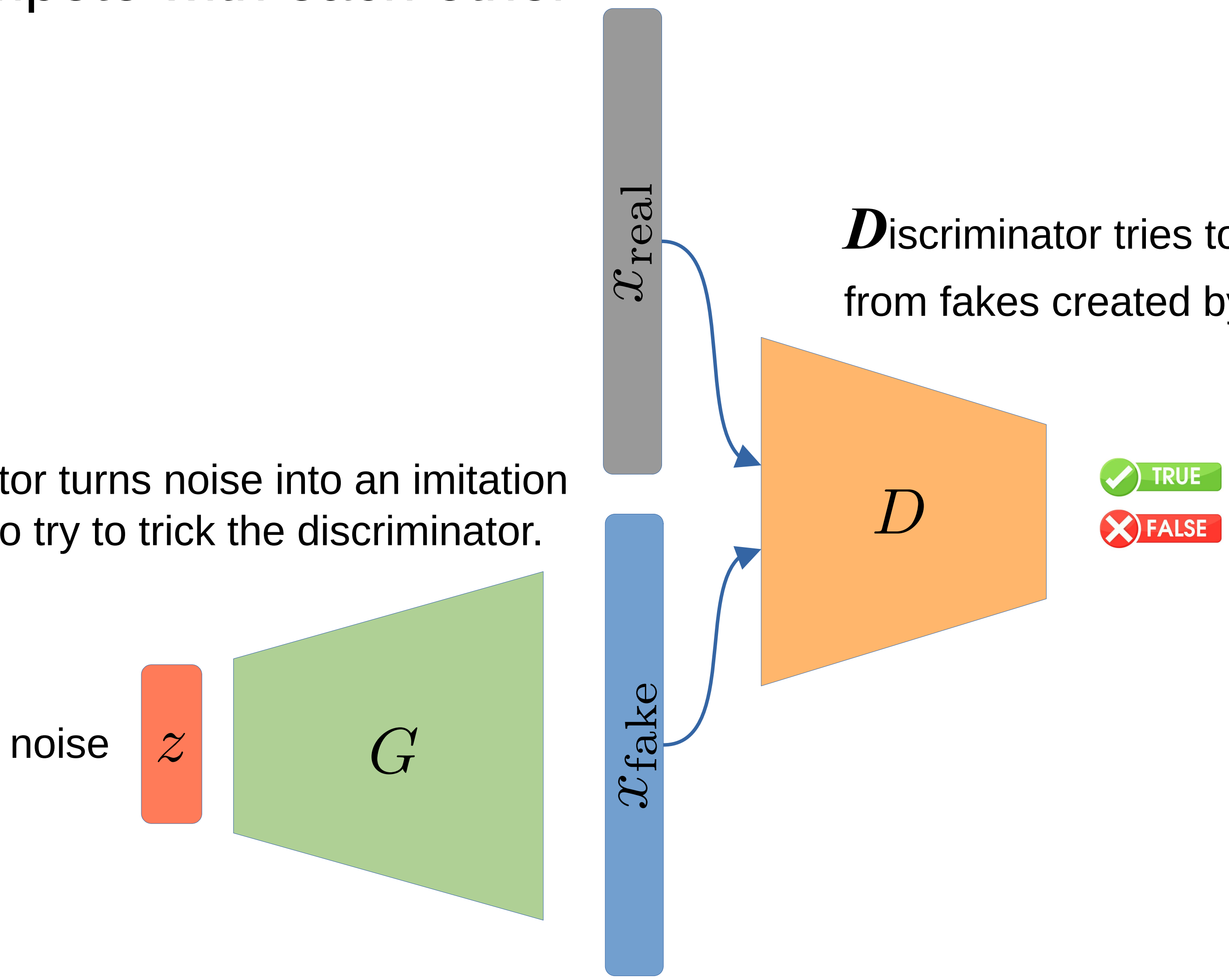
Generator Network

Generative Adversarial Networks - GANs

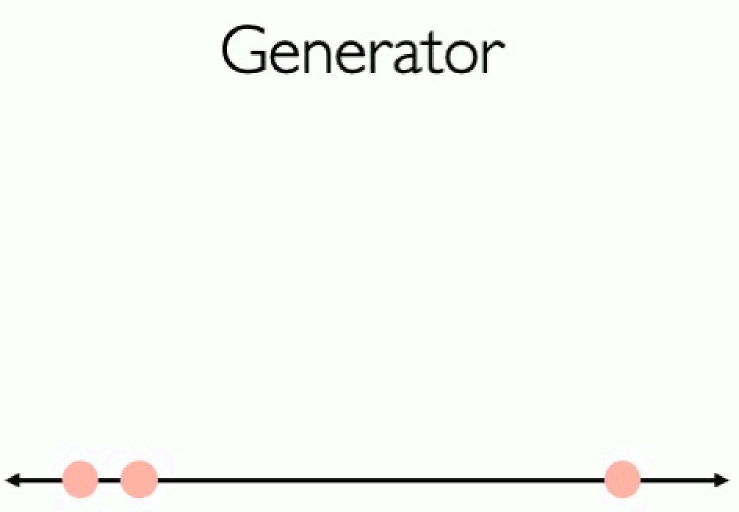
- GAN: create a generative model by having two networks compete with each other.

Discriminator tries to identify real data from fakes created by ***G***enerator.

Generator turns noise into an imitation of data to try to trick the discriminator.



Intuition behind GANs



Real data   Fake data

MIT course, Introduction to deep learning

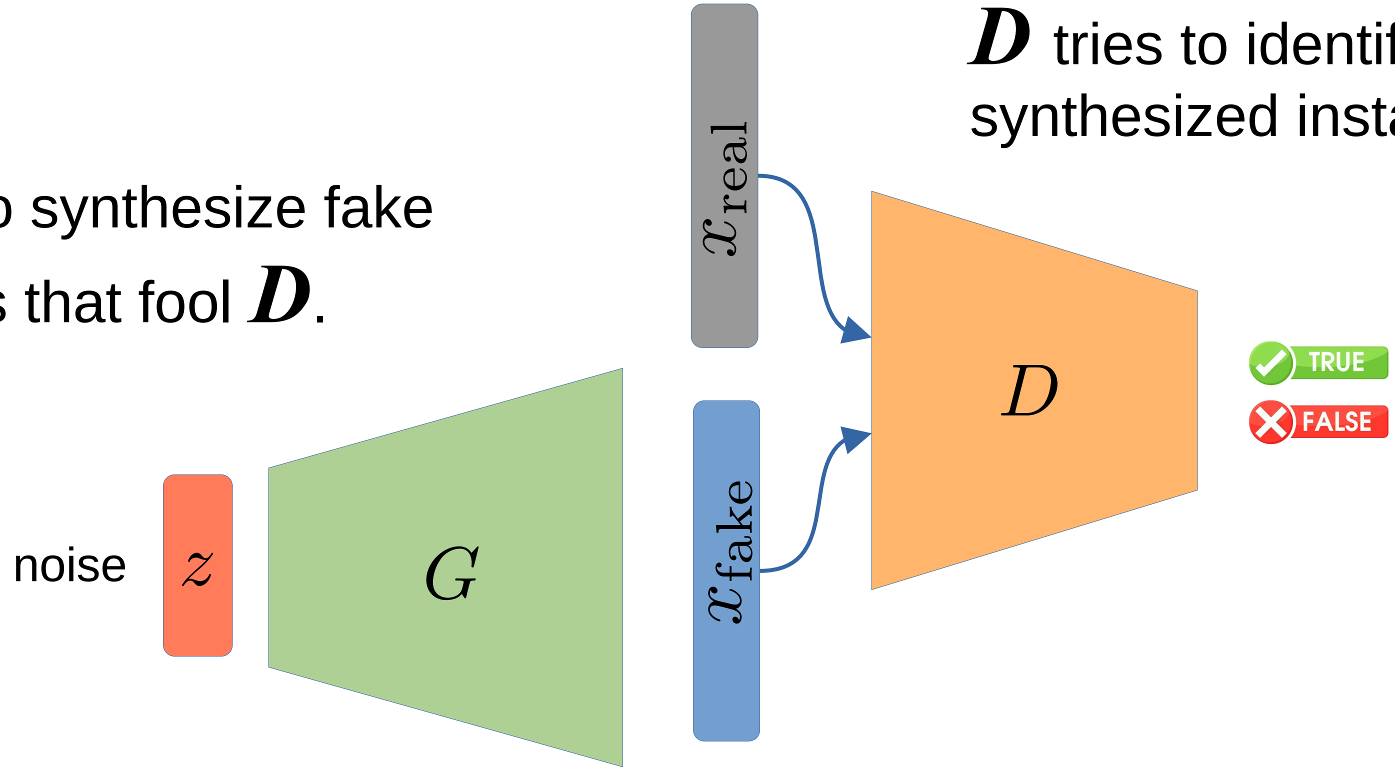
Training GANs

- Adversarial objectives for *D* and *G*

$$\min_G \max_D \mathbb{E}_{z,x}[\log(1 - D(G(z))) + \log D(x)]$$

$\boldsymbol{D}$ tries to identify the synthesized instances.

$\boldsymbol{G}$ tries to synthesize fake instances that fool $\boldsymbol{D}$.

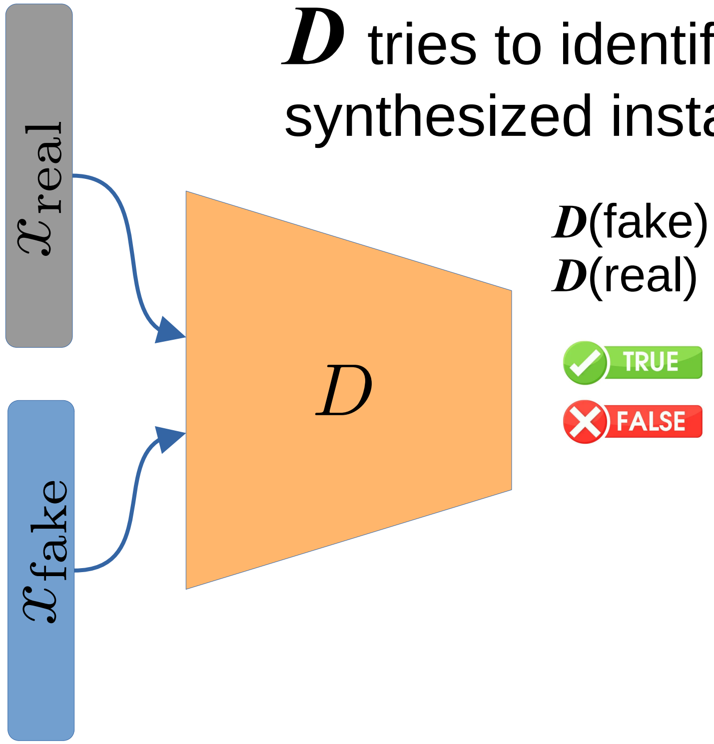


Training GANs

- Loss:

$$\max_{D} \mathbb{E}_{z,x}[\log(1 - D(G(z))) + \log D(x)]$$

fake real



D tries to identify the synthesized instances.

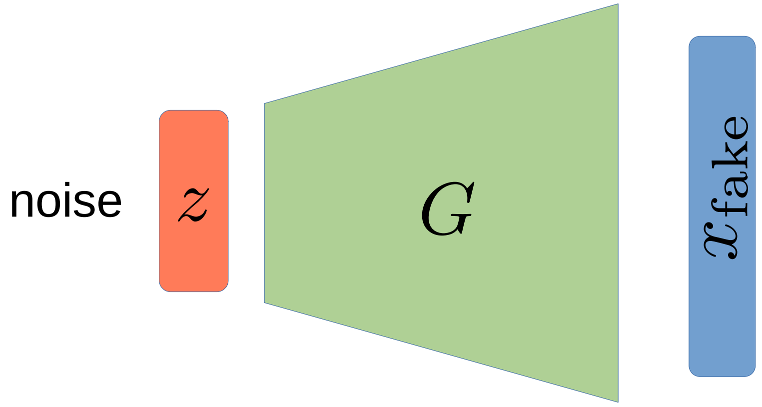
D(fake) = 0
D(real) = 1

Training GANs

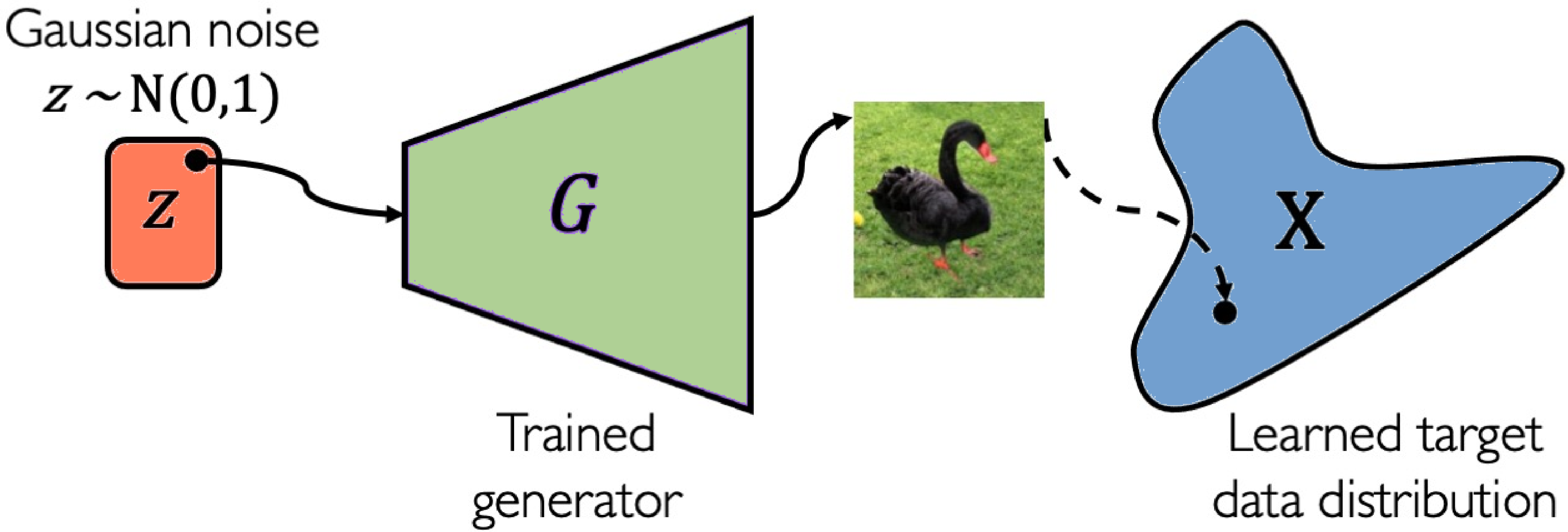
- Loss:

$$\min_{G} \mathbb{E}_{z,x}[\log(1 - D(G(z))) + \log D(x)]$$

$\boldsymbol{G}$ tries to synthesize fake instances that fool $\boldsymbol{D}$.

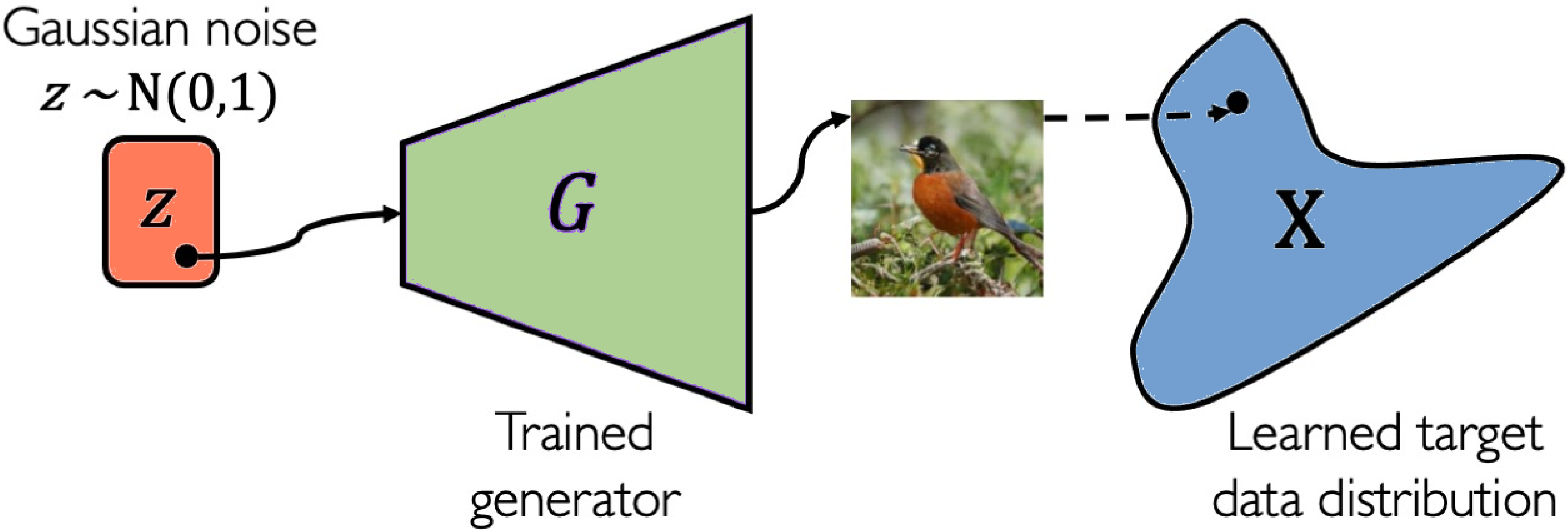


GANs are distribution transformers



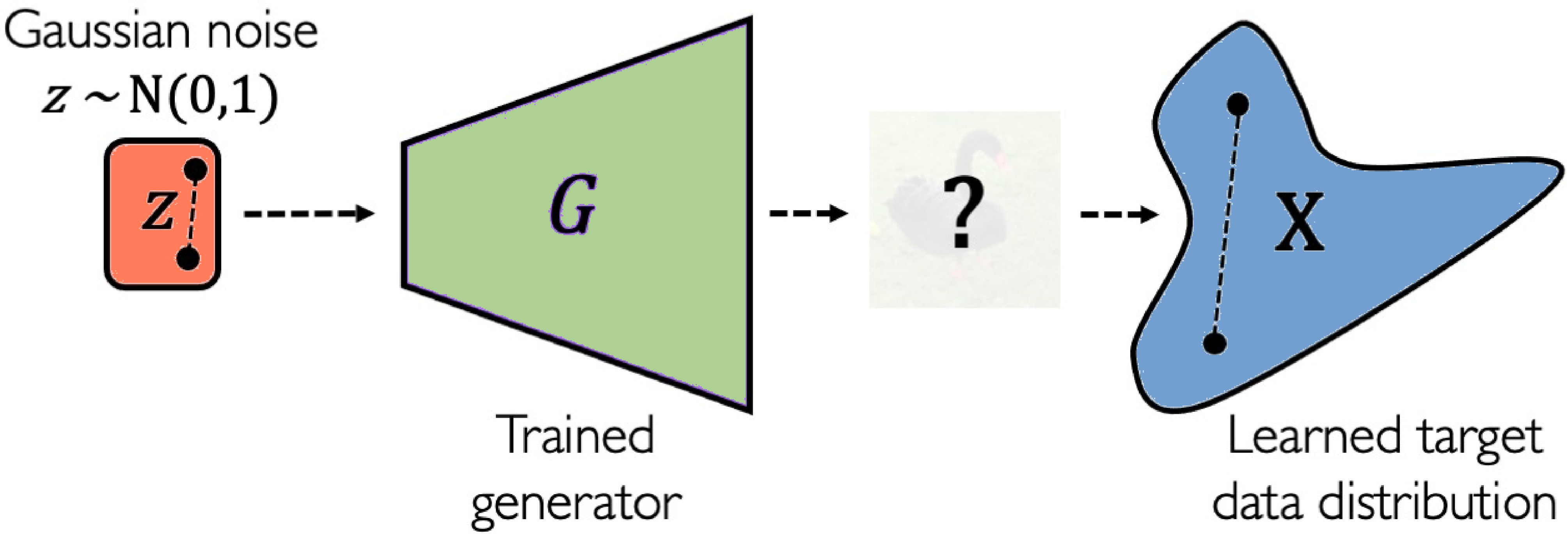
MIT course, Introduction to deep learning

GANs are distribution transformers



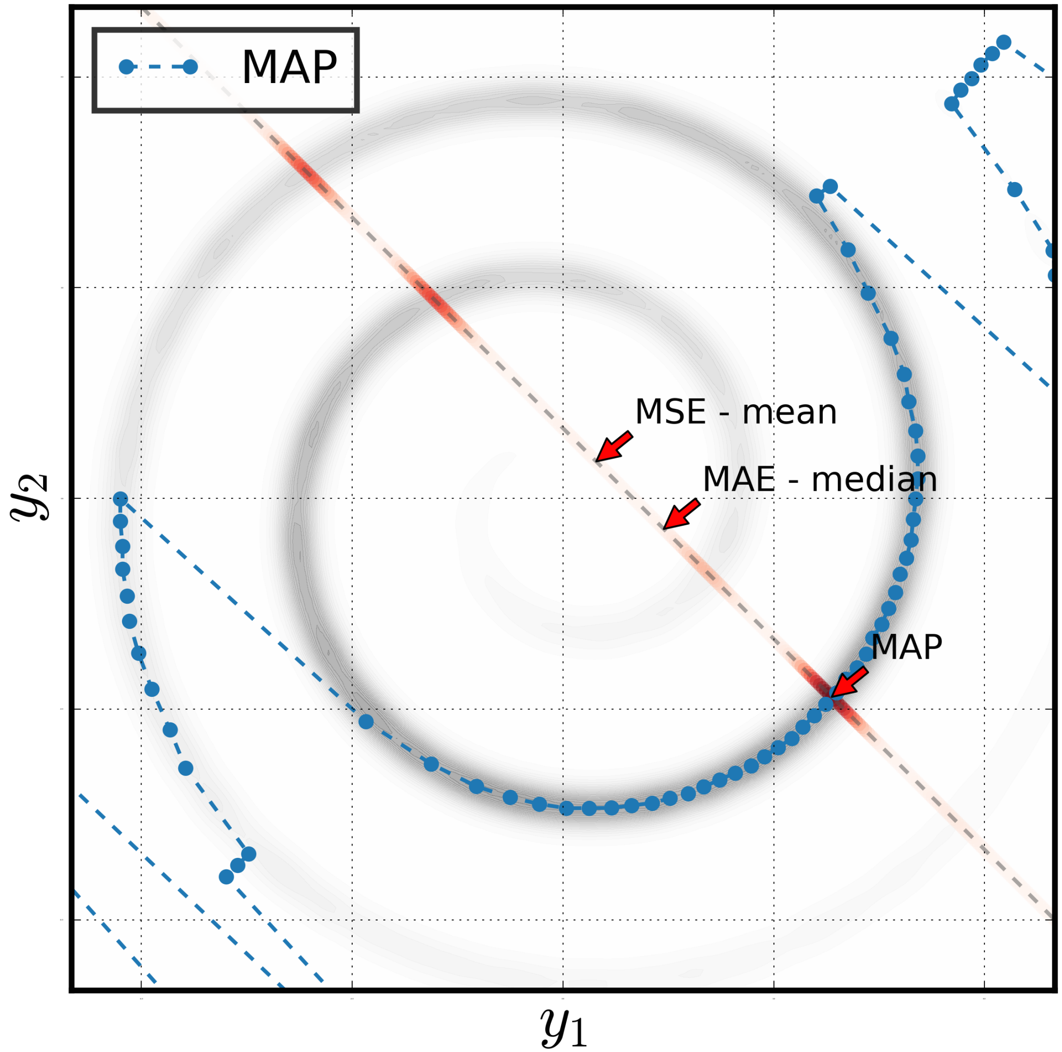
MIT course, Introduction to deep learning

GANs are distribution transformers

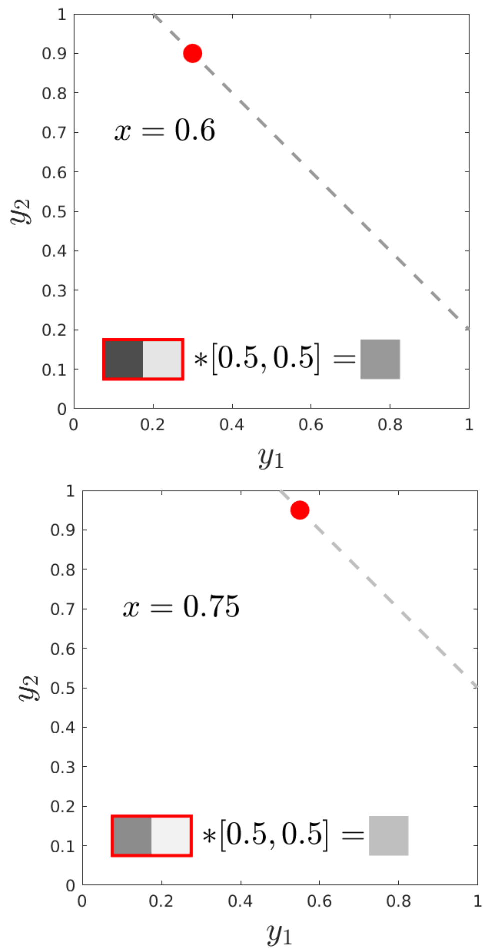


MIT course, Introduction to deep learning

GANs vs AEs



$$\frac{1}{2}(y_1 + y_2) = x$$



MSE – $(h * y - x)^2$

MAE – $|h * y - x|$

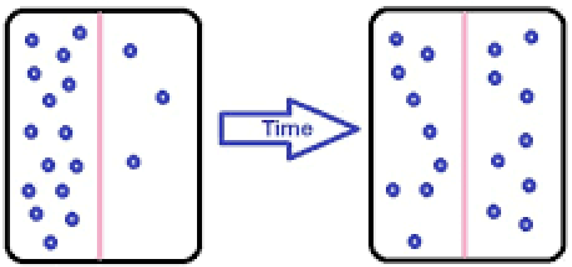
MAP – Adversarial loss

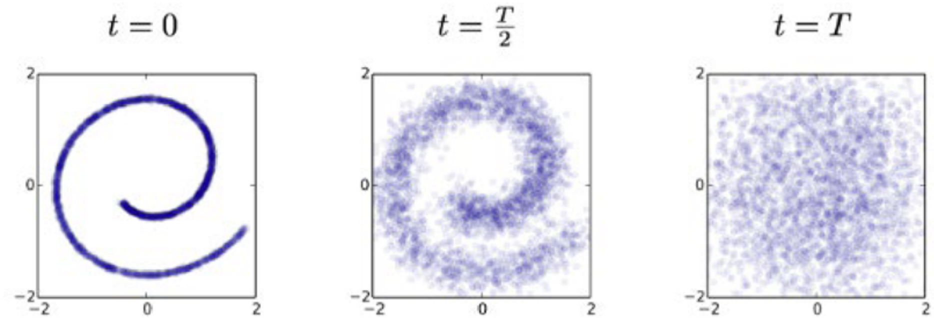
Sonderby et al., ICLR, 2017

Diffusion Models

Dickstein et al., 2015
Ho et al., 2020

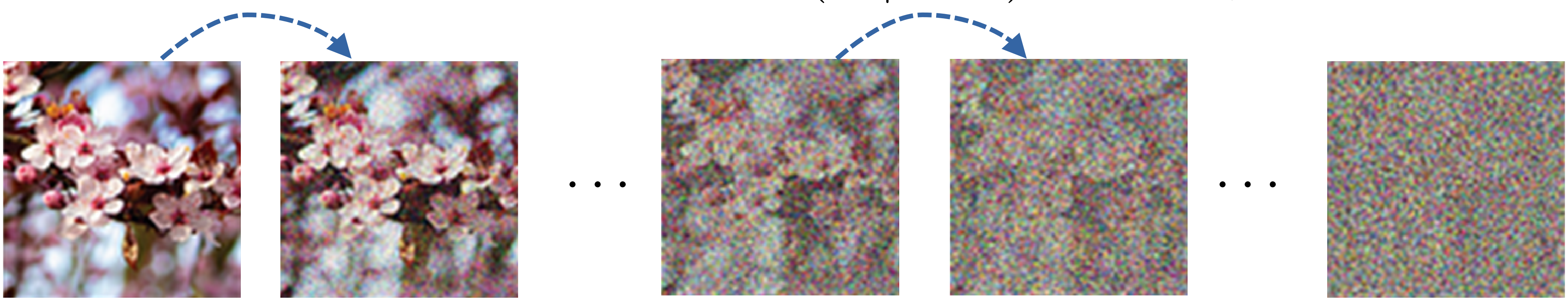
- Diffusion





- Forward process

$$q(x_t|x_{t-1}) \equiv N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$



$\mathbf{x}_0$ $\mathbf{x}_1$ $\cdots$ $\mathbf{x}_{t-1}$ $\mathbf{x}_t$ $\cdots$ $\mathbf{x}_T$

$\sim N(x_T; 0, I)$

$$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

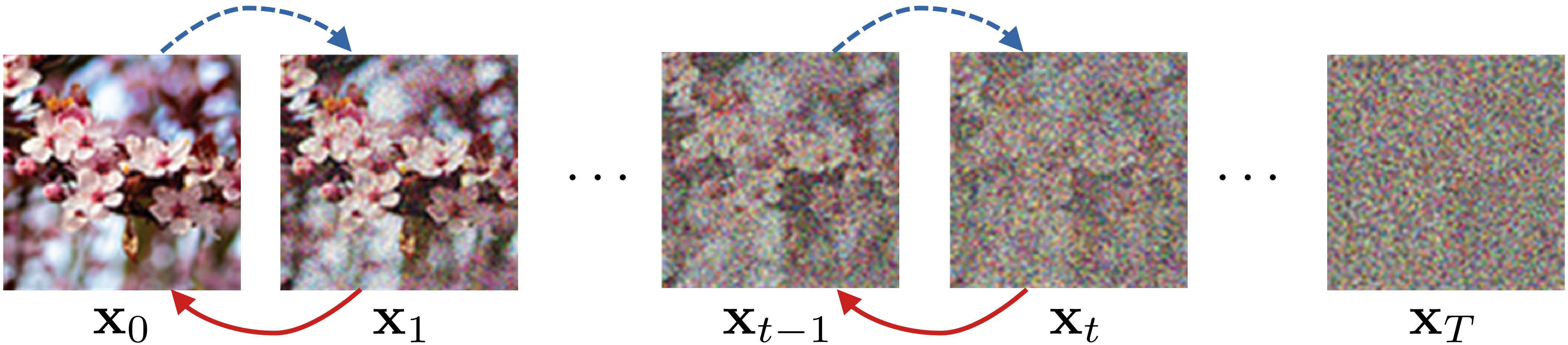
$$\bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)$$

Diffusion Model

- Forward process

$$\equiv N(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I)$$

$q(x_1|x_0)$ $\quad$ $q(x_t|x_{t-1})$



$\mathbf{x}_0$ $\quad$ $\mathbf{x}_1$ $\quad$ $\mathbf{x}_{t-1}$ $\quad$ $\mathbf{x}_t$ $\quad$ $\mathbf{x}_T$

$\sim N(x_T; 0, I)$

$p(x_0|x_1)$ $\quad$ $p(x_{t-1}|x_t)$

$$= N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

- Reverse process
  - Training (ML): $\min \mathbb{E}_{q(x_0)}[-\log p(x_0)]$

$$p(x_0) = \int p(x_0, \ldots, x_T) dx_1 \ldots dx_T = \int p(x_T) \prod_{i=1}^{T} p(x_{i-1}|x_i) dx_1 \ldots dx_T$$

- Forward

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I) \qquad \bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)$$

reparametrization: $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \qquad \epsilon \sim N(0, I)$

- Reverse

$$p(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

- Training

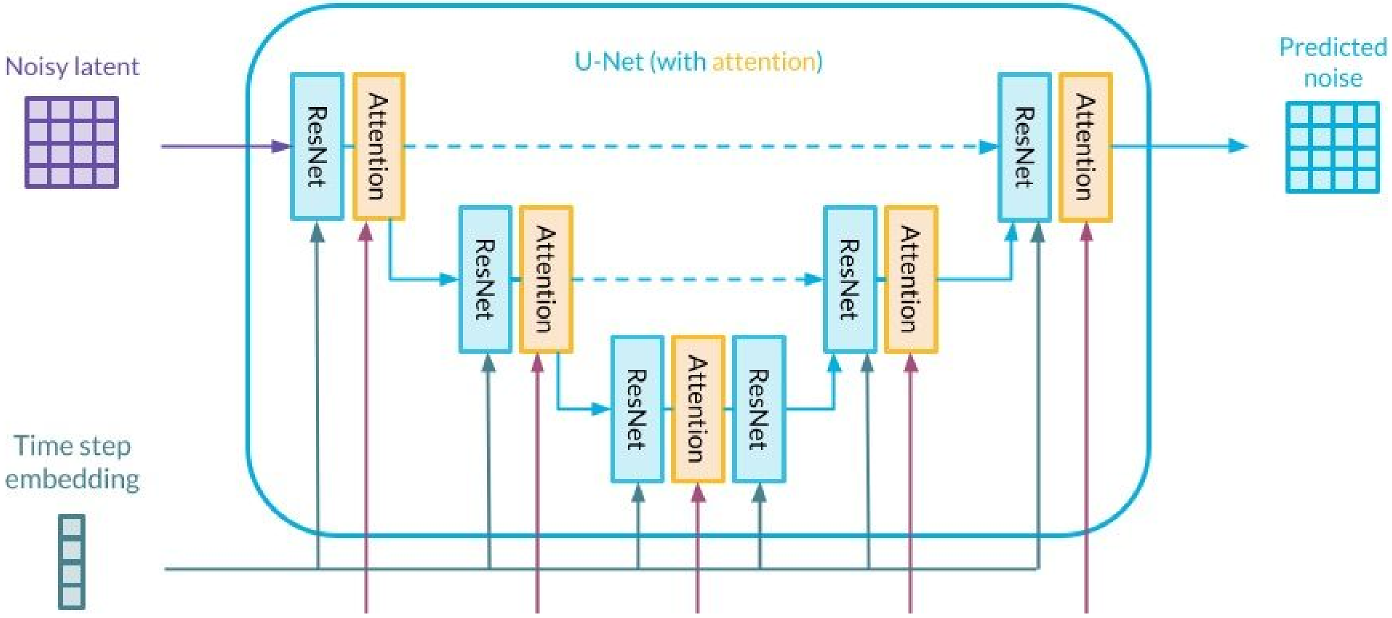
$$\min_\theta \mathbb{E}_{q(x_0)}[-\log p(x_0)] = \dots$$

$$\min_\theta \mathbb{E}_{x_0,\epsilon,t}\left[\frac{1}{2\|\Sigma_\theta\|^2}\left\|\frac{1}{\sqrt{1-\beta_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \mu_\theta(x_t, t)\right\|^2\right]$$

or predict noise $\displaystyle\min_\theta \mathbb{E}_{x_0,\epsilon,t}\left[\frac{1}{2\|\Sigma_\theta\|^2}\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$

- Prediction of $\mu_\theta(x_t, t)$ or $\epsilon_\theta(x_t, t)$ is done by

  U-Net(θ) with residual and attention blocks
  and t implemented as sinusoid positional encoding.

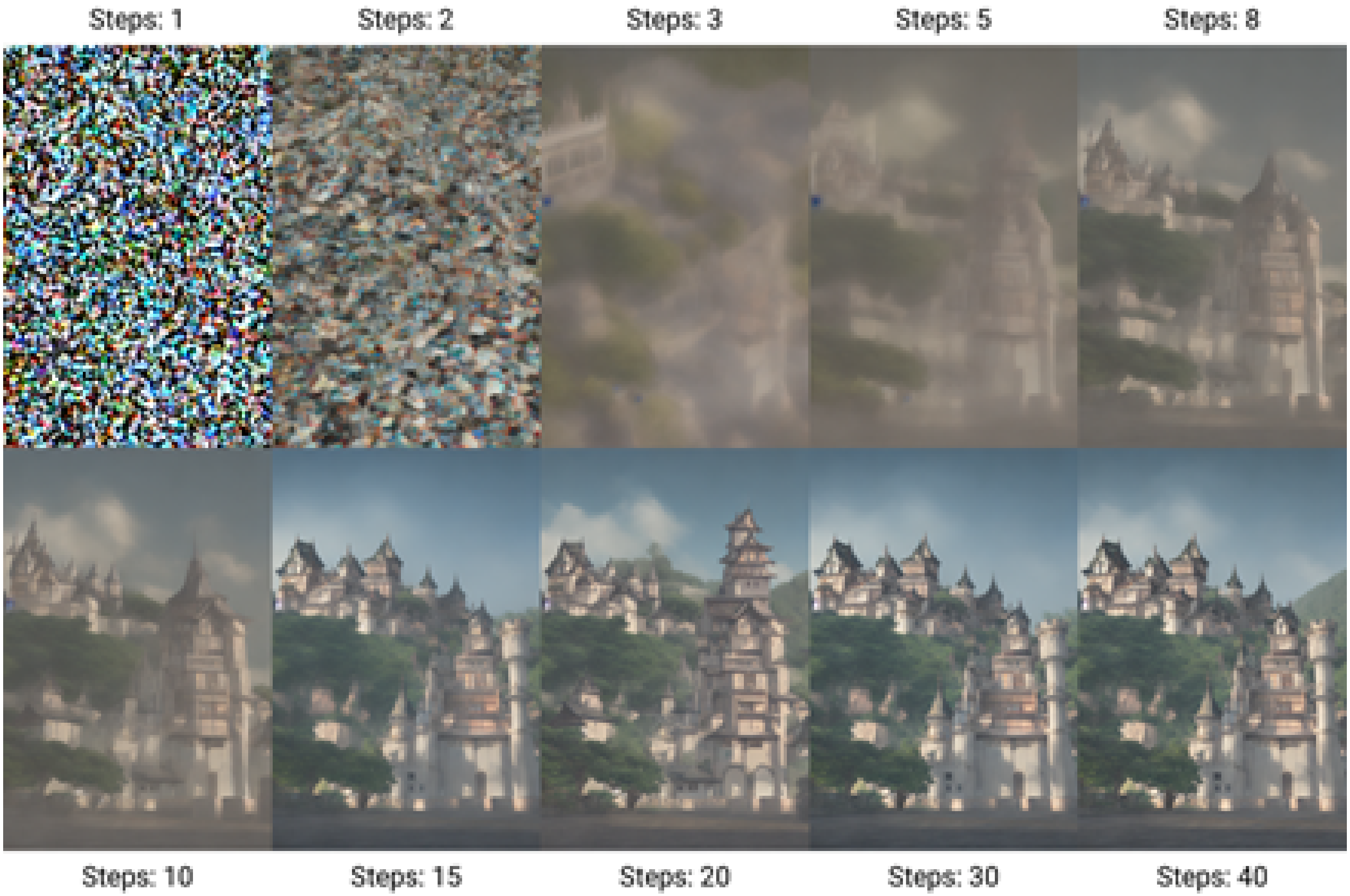


Algorithm 1 Training

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

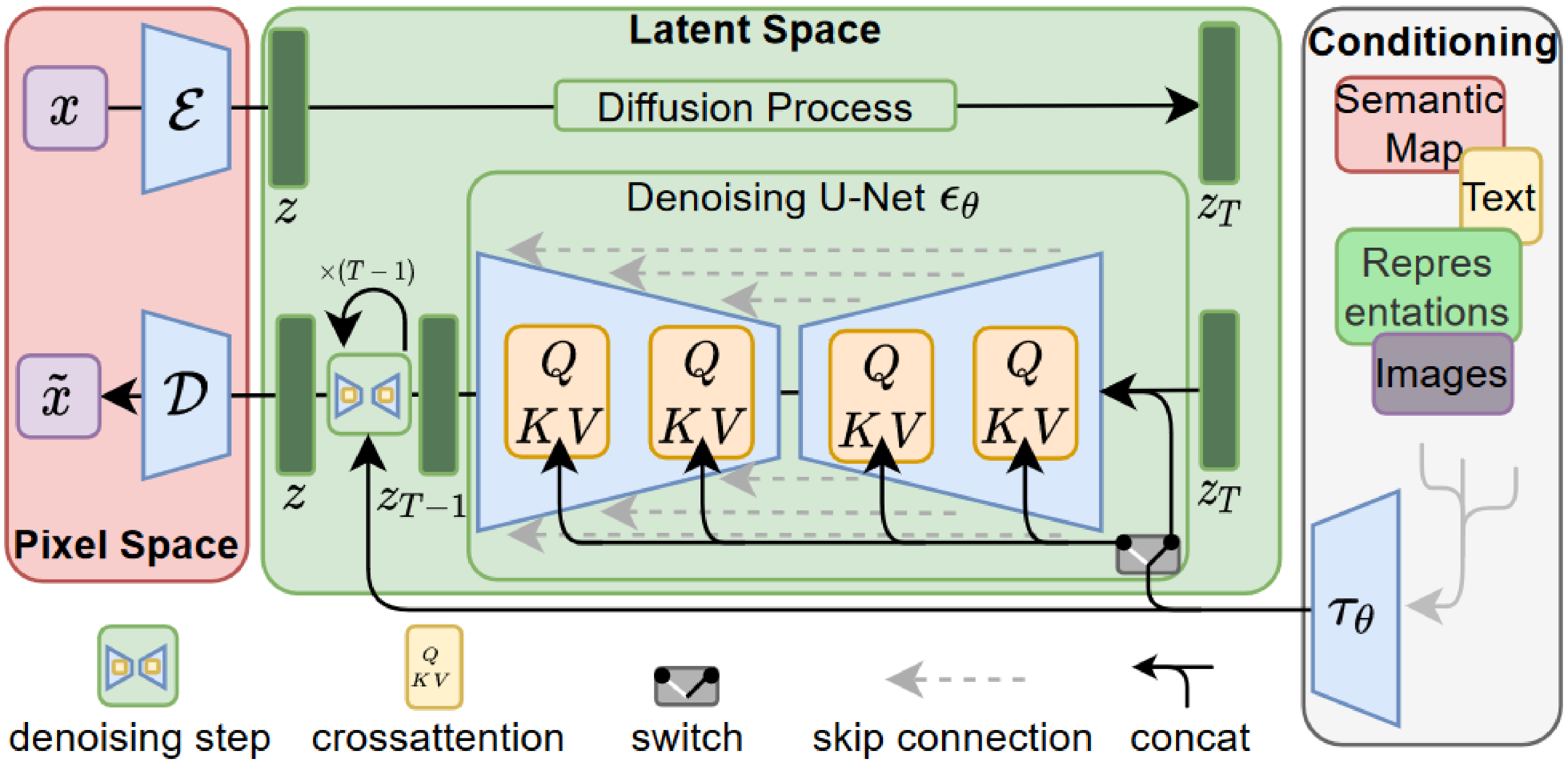
Algorithm 2 Sampling

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


Steps: 1
Steps: 2
Steps: 3
Steps: 5
Steps: 8
Steps: 10
Steps: 15
Steps: 20
Steps: 30
Steps: 40

Stable Diffusion Model

- Latent space
- Conditional diffusion (cross-attention)

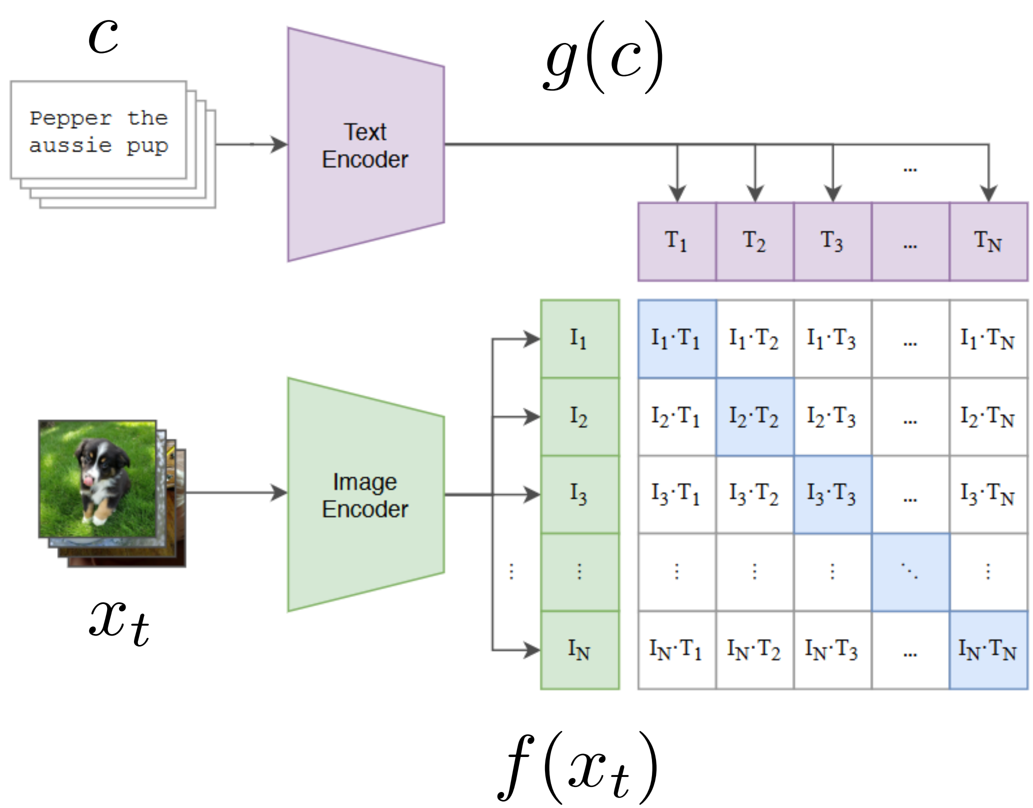


Rombach et al. LDM, 2022

DALL-E

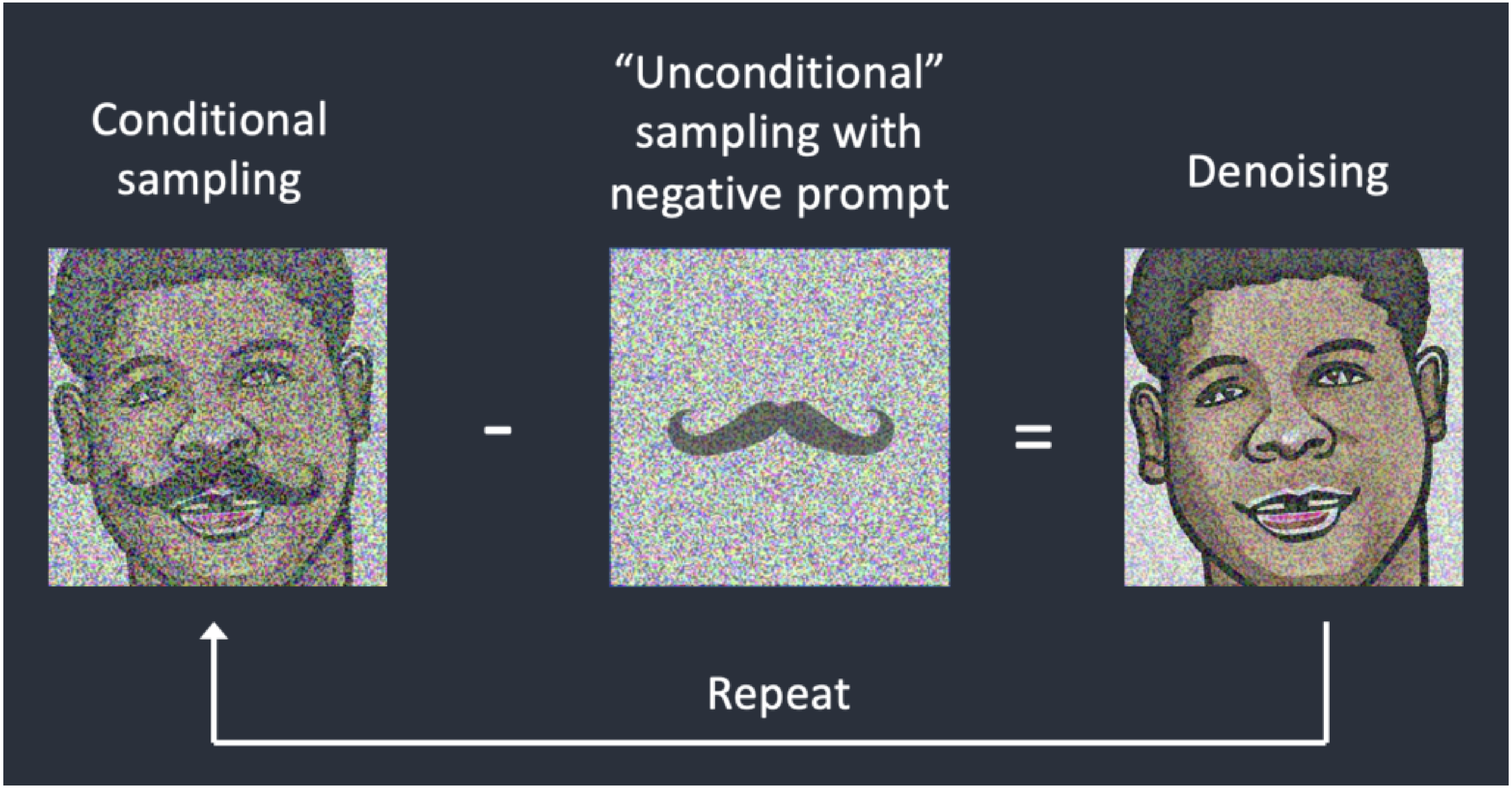
gradient scale

$$\hat{\mu}_\theta(x_t, t) = \mu_\theta(x_t, t) + s\Sigma_\theta(x_t, t)\nabla_{x_t}\underbrace{\langle f(x_t), g(c)\rangle}_{\approx \log p(c|x_t)}$$



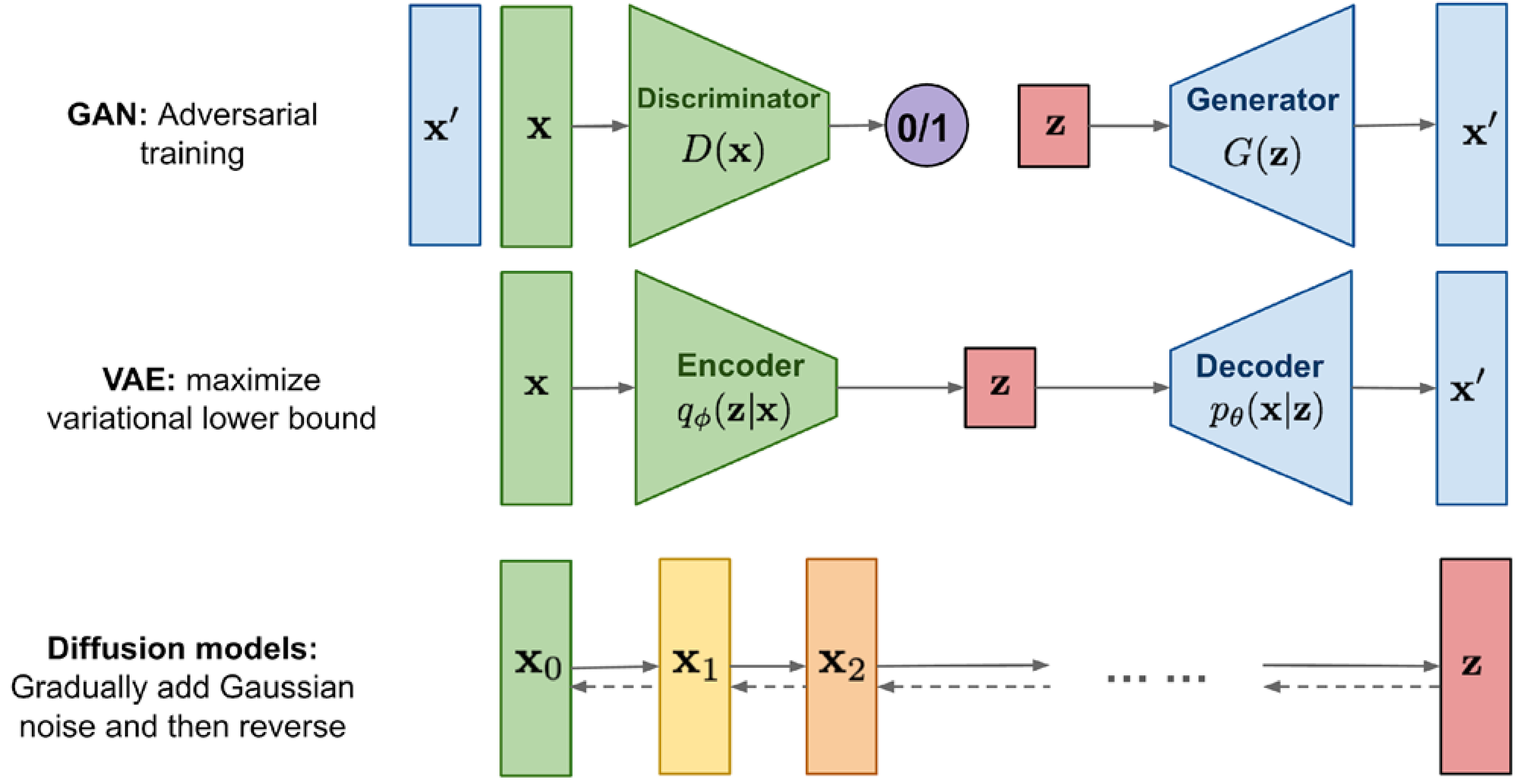
Nichol et al. GLIDE, 2022

Negative Prompt



detailed meadow with colorful flowers

detailed meadow with colorful flowers
–no blue



GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator
$D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator
$G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{z}$