Generative Neural Networks

CS 6384 Computer Vision
Professor Yu Xiang
The University of Texas at Dallas

Supervised Learning

$\mathbf{x}$ ⇨  ⇨ $\mathbf{y}$

Image | Object class

$$f(\mathbf{x})$$

Training Data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$

Input | Output

Unsupervised Learning

- Training data $\{\mathbf{x}_i\}_{i=1}^{N}$ No label
- Goal: discover some underlying hidden structure of the data
- Examples
  - Dimension reduction
  - Clustering
  - Probability density estimation

Dimension Reduction

- Map data from a high-dimension space to a low-dimension space

$$\mathbf{x} \in \mathcal{R}^n \rightarrow \mathbf{y} \in \mathcal{R}^m \qquad m < n$$

- The low-dimensional representation maintains meaningful properties of the original data
  - E.g., can be used to reconstruct the original data

- Applications
  - Data compression, data visualization, data representation learning

Principal Component Analysis (PCA)

- Linear mapping $\mathbf{y} = P\mathbf{x}$

$$\underset{m \times 1}{\mathbf{y}} = \underset{m \times m}{P} \underset{m \times 1}{\mathbf{x}}$$

Rows of P, principal components

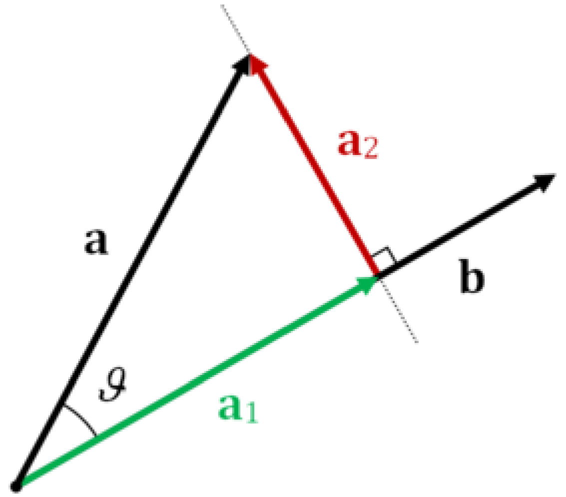
$$\mathbf{y} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_m \end{bmatrix} \mathbf{x}$$

Principal Component Analysis (PCA)

- Change of basis

$$\mathbf{y} = \begin{bmatrix} \mathbf{p}_1 \cdot \mathbf{x} \\ \mathbf{p}_2 \cdot \mathbf{x} \\ \vdots \\ \mathbf{p}_m \cdot \mathbf{x} \end{bmatrix}$$

Projection



$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

$$\mathbf{a}_1 = \|\mathbf{a}\| \cos\theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|}$$

If $\|\mathbf{b}\| = 1$ $\quad \mathbf{a}_1 = \mathbf{a} \cdot \mathbf{b}$

Principal Component Analysis (PCA)

- Given a set of data points

$$X \in \mathcal{R}^{\underset{\text{dimension}}{m} \times \underset{\text{\# data points}}{n}}$$

$$Y = PX$$

- Covariance matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} \\ \vdots \\ \mathbf{x_m} \end{bmatrix}$$

Rows of X

$$\mathbf{C_X} \equiv \frac{1}{n}\mathbf{X}\mathbf{X}^T$$

$$\mathbf{C_Y}$$

Principal Component Analysis (PCA)

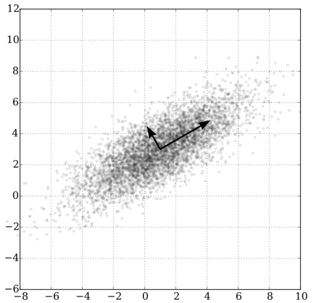
- The goal of PCA
  - All off-diagonal terms in $\mathbf{C_Y}$ should be zero (Y is decorrelated)
  - Each successive dimension of Y should be rank-ordered according to variance
- Solution

$$
\begin{aligned}
\mathbf{C_Y} &= \frac{1}{n}\mathbf{Y}\mathbf{Y}^T \\
&= \frac{1}{n}(\mathbf{PX})(\mathbf{PX})^T \\
&= \frac{1}{n}\mathbf{P}\mathbf{X}\mathbf{X}^T\mathbf{P}^T \\
&= \mathbf{P}(\frac{1}{n}\mathbf{X}\mathbf{X}^T)\mathbf{P}^T \\
\mathbf{C_Y} &= \mathbf{P}\mathbf{C_X}\mathbf{P}^T
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{C_Y} &= \mathbf{P}\mathbf{C_X}\mathbf{P}^T \\
&= \mathbf{P}(\mathbf{E}^T\mathbf{D}\mathbf{E})\mathbf{P}^T \\
&= \mathbf{P}(\mathbf{P}^T\mathbf{D}\mathbf{P})\mathbf{P}^T \\
&= (\mathbf{P}\mathbf{P}^T)\mathbf{D}(\mathbf{P}\mathbf{P}^T) \\
&= (\mathbf{P}\mathbf{P}^{-1})\mathbf{D}(\mathbf{P}\mathbf{P}^{-1}) \\
\mathbf{C_Y} &= \mathbf{D}
\end{aligned}
$$

The principal components P is the eigenvectors of

$$\mathbf{C_X} \equiv \frac{1}{n}\mathbf{X}\mathbf{X}^T$$



A Tutorial on Principal Component Analysis. Jonathon Shlens, 2014

Principal Component Analysis (PCA)

- Dimension reduction

$$\mathbf{y} = P\mathbf{x}$$

$$\underset{m \times 1}{\mathbf{y}} = \underset{m \times m}{P}\ \underset{m \times 1}{\mathbf{x}}$$

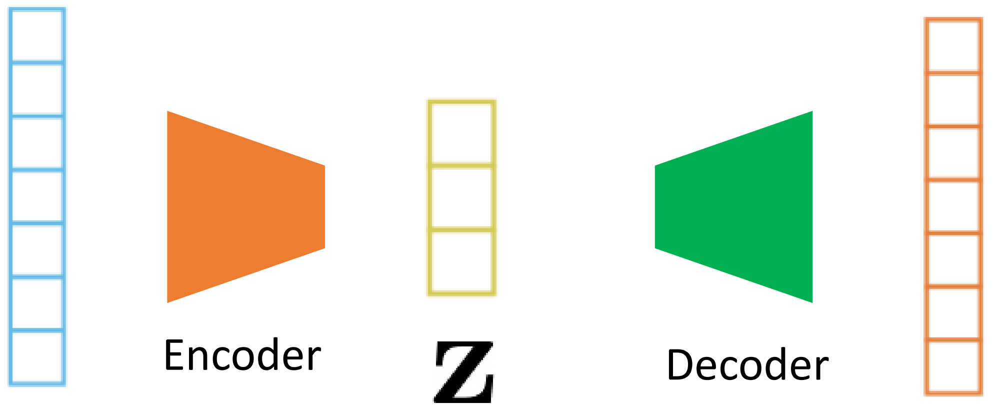
$$\mathbf{y} = P_L \mathbf{x}$$

$$\underset{L \times 1}{\mathbf{y}} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_L \end{bmatrix} \mathbf{x}$$

Use L < m principal components

Autoencoder

- Use a neural network for dimension reduction



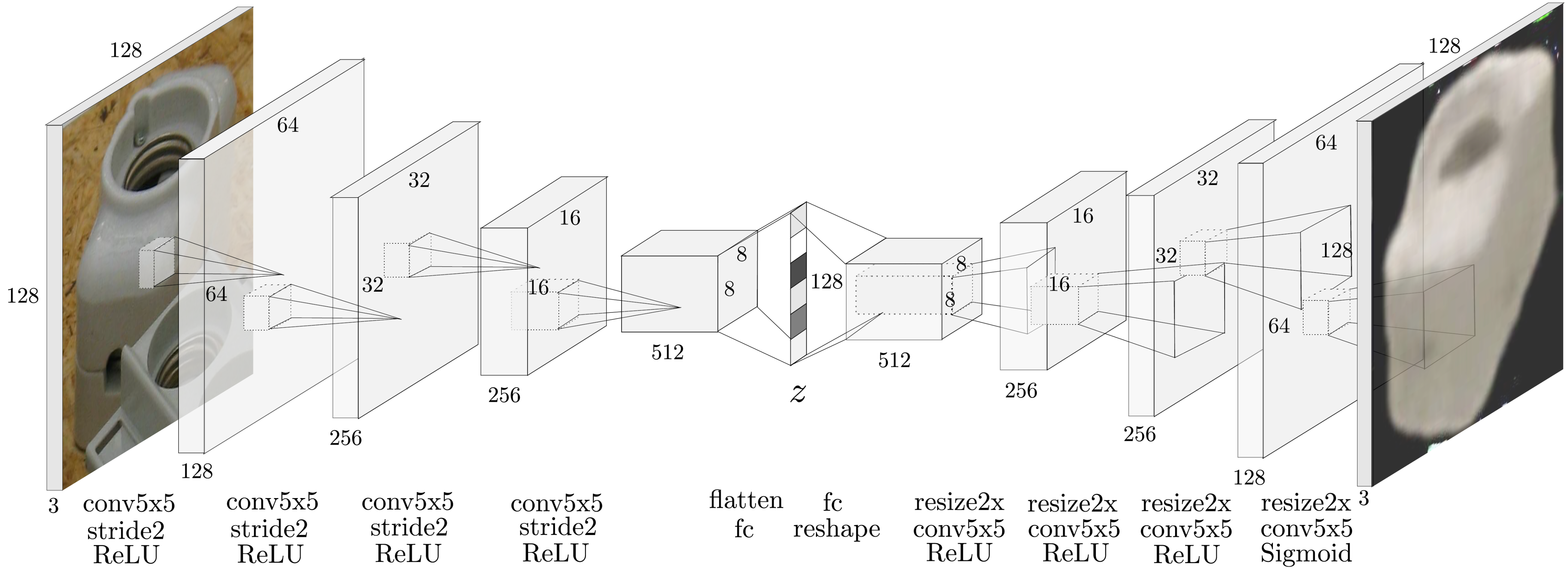
$\mathbf{x}$ $\mathbf{z}$ $\hat{\mathbf{x}}$

Reconstruction loss function

$$L_2 = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

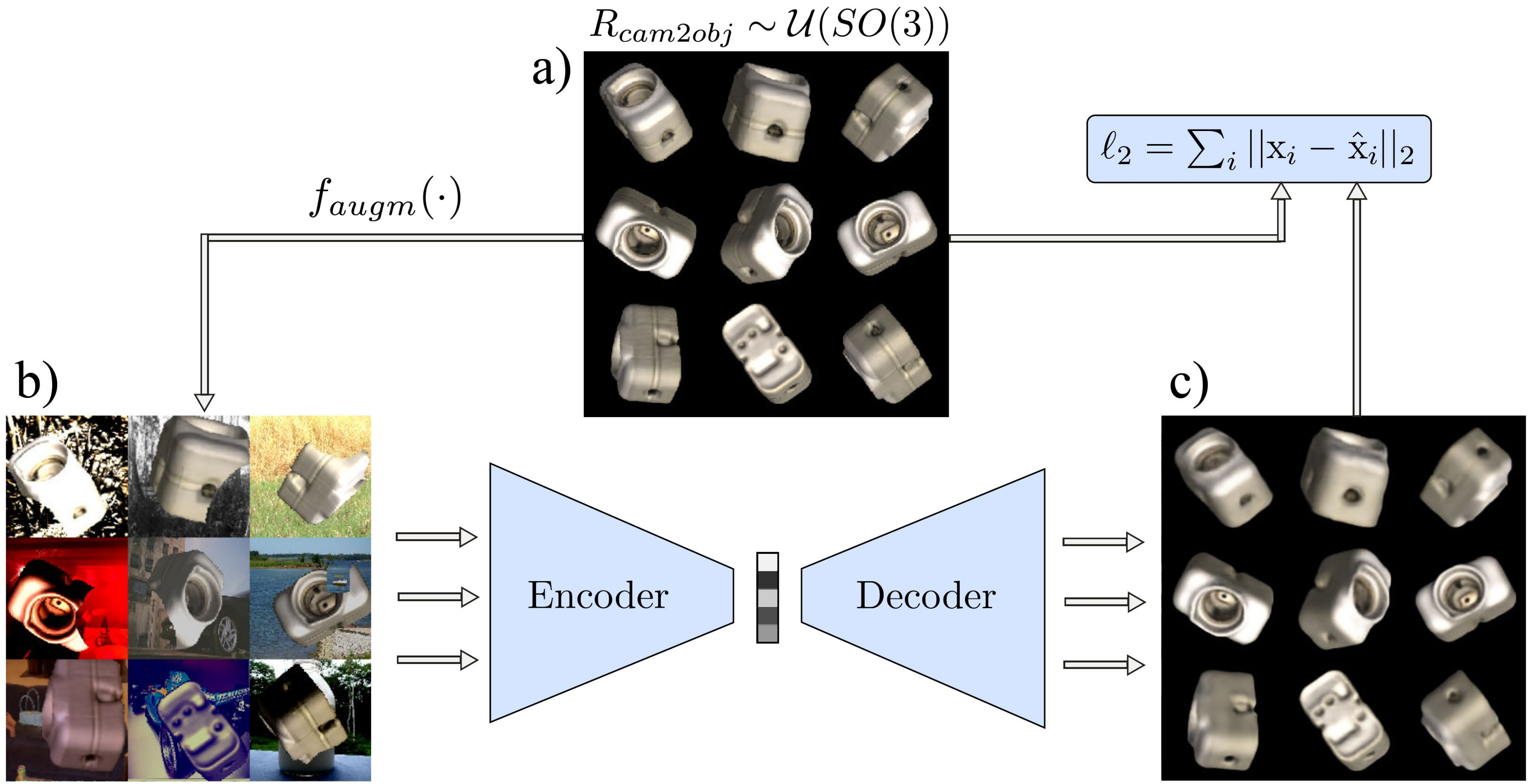
$$\mathbf{z} = f(\mathbf{x}) \qquad \hat{\mathbf{x}} = g(\mathbf{z})$$

Case Study: Augmented Autoencoder



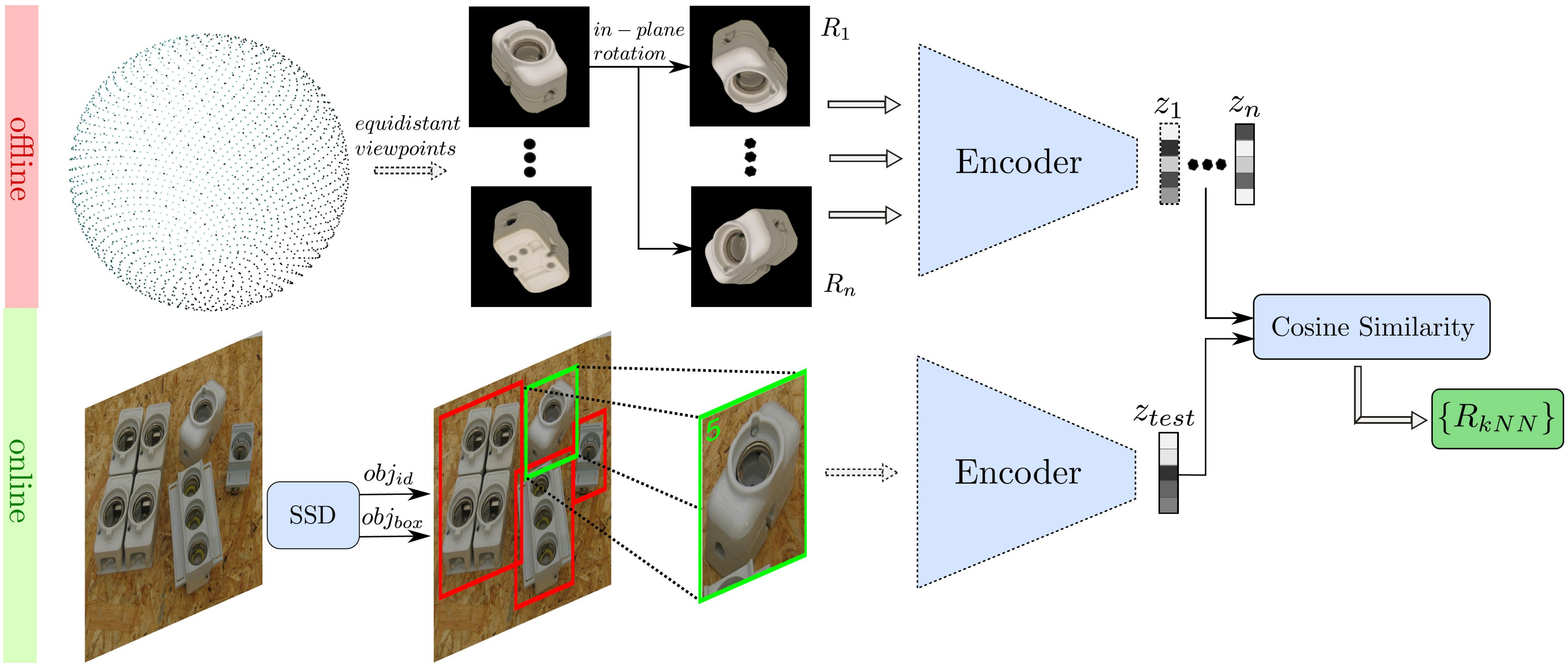
Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection. Sundermeyer et al., IJCV’20

Case Study: Augmented Autoencoder



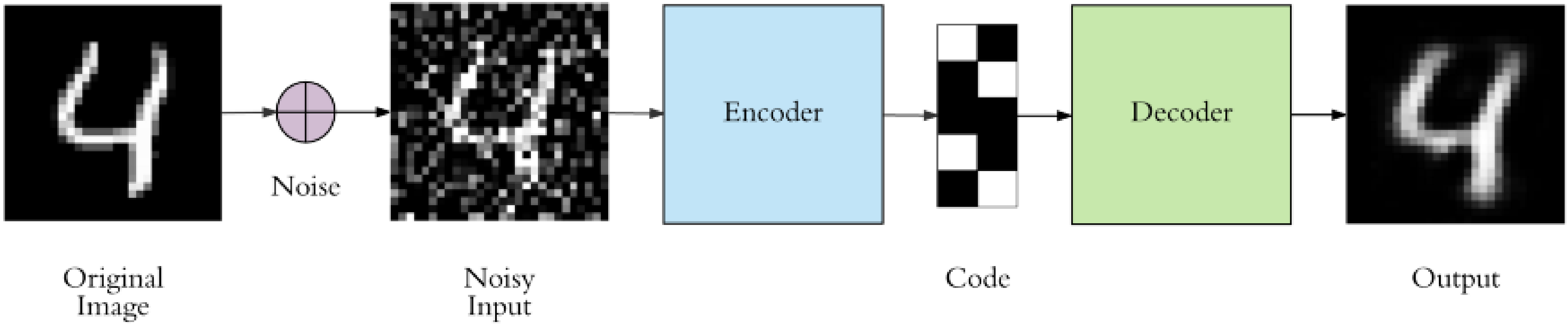
Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection. Sundermeyer et al., IJCV'20

Case Study: Augmented Autoencoder



Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection. Sundermeyer et al., IJCV’20

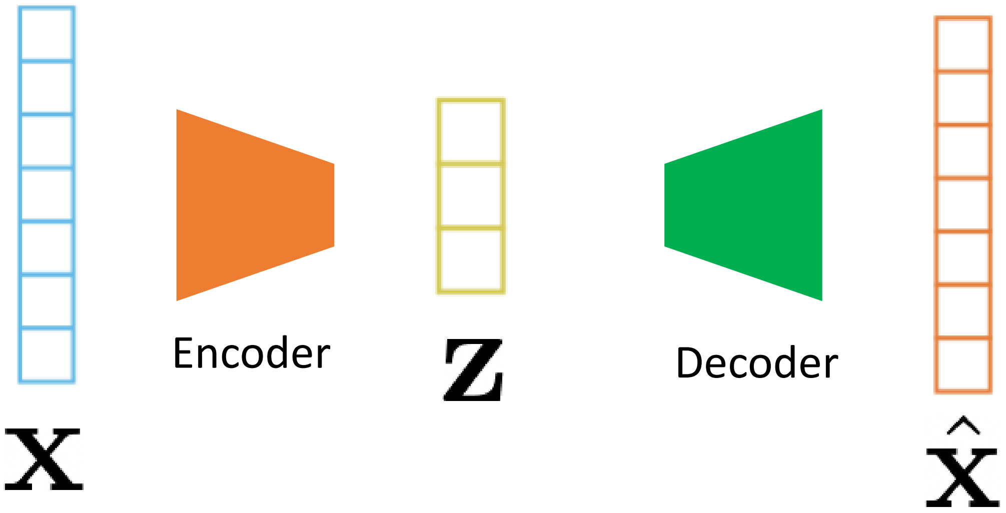
Case Study: Denoising Autoencoder

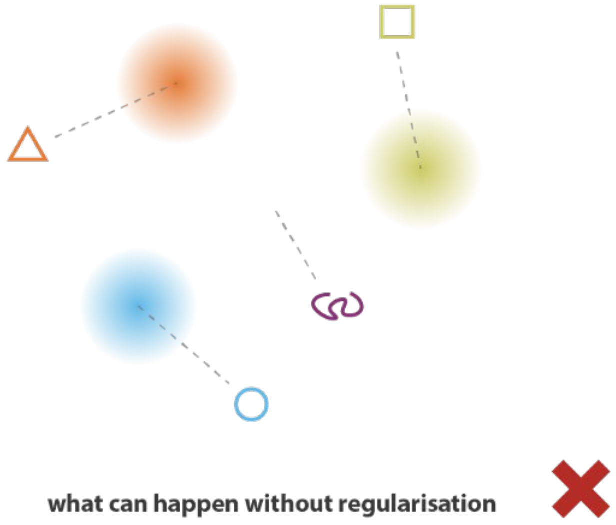


https://www.analyticsvidhya.com/blog/2021/07/image-denoising-using-autoencoders-a-beginners-guide-to-deep-learning-project/

Content Generation

- Given a dataset $\{\mathbf{x}_i\}_{i=1}^N$
- How to generate new content from the underlying distribution P(x)?
- Autoencoder is not suitable for content generation

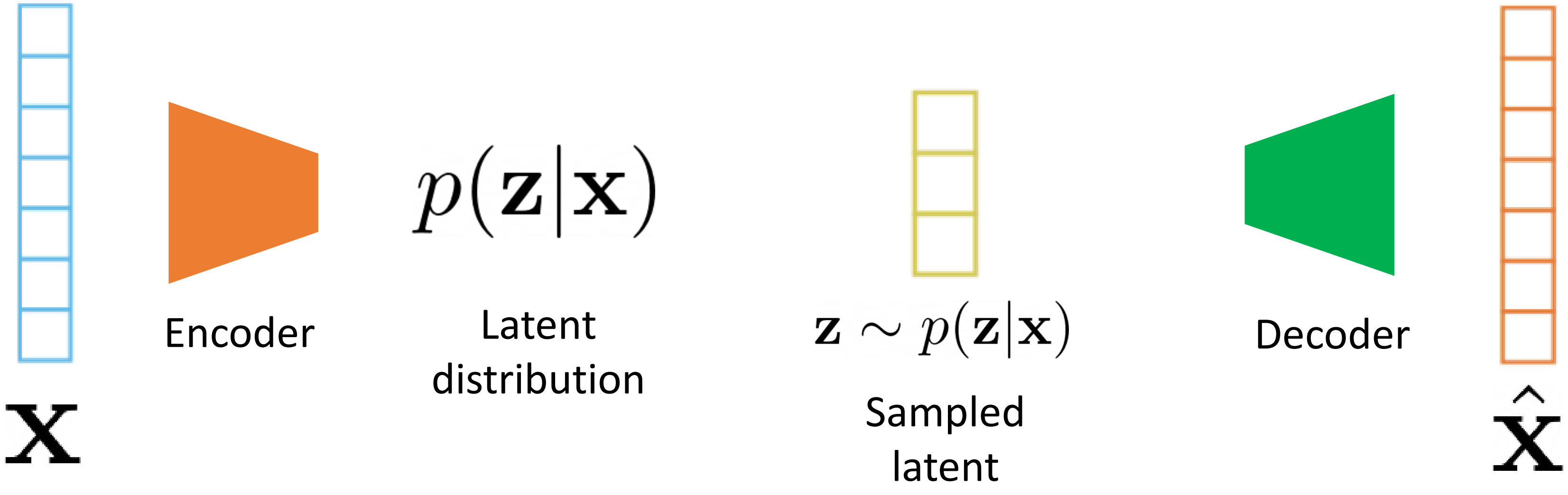




The latent space is not regularized. Some latent vectors may generate meaningless content.

Variational Autoencoder

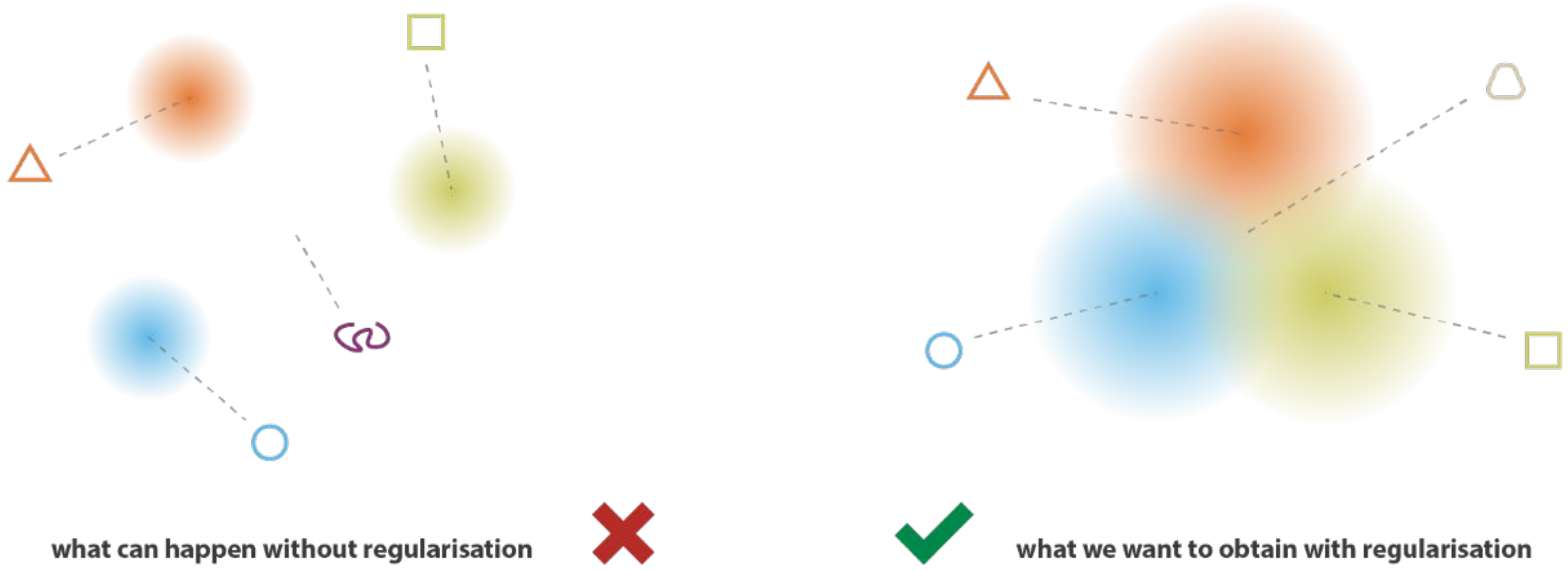
- Introduce regularization to the latent space
- Probabilistic formulation



$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_x, \sigma_x) \Longleftrightarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$$ Prior distribution

Variational Autoencoder

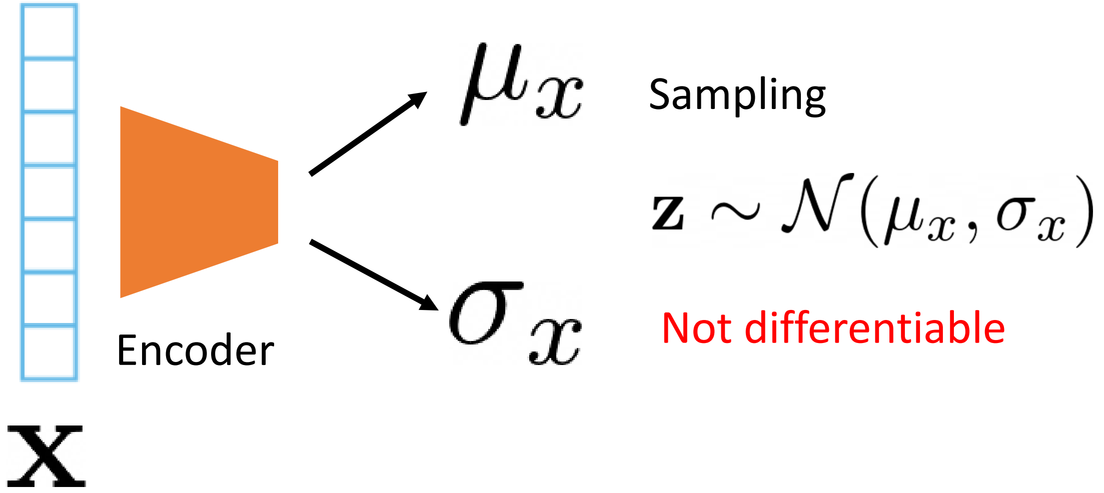
- Latent space
  - Continuity (close points in latent space decode similar outputs)
  - Completeness (a sampled latent should generate meaningful output)



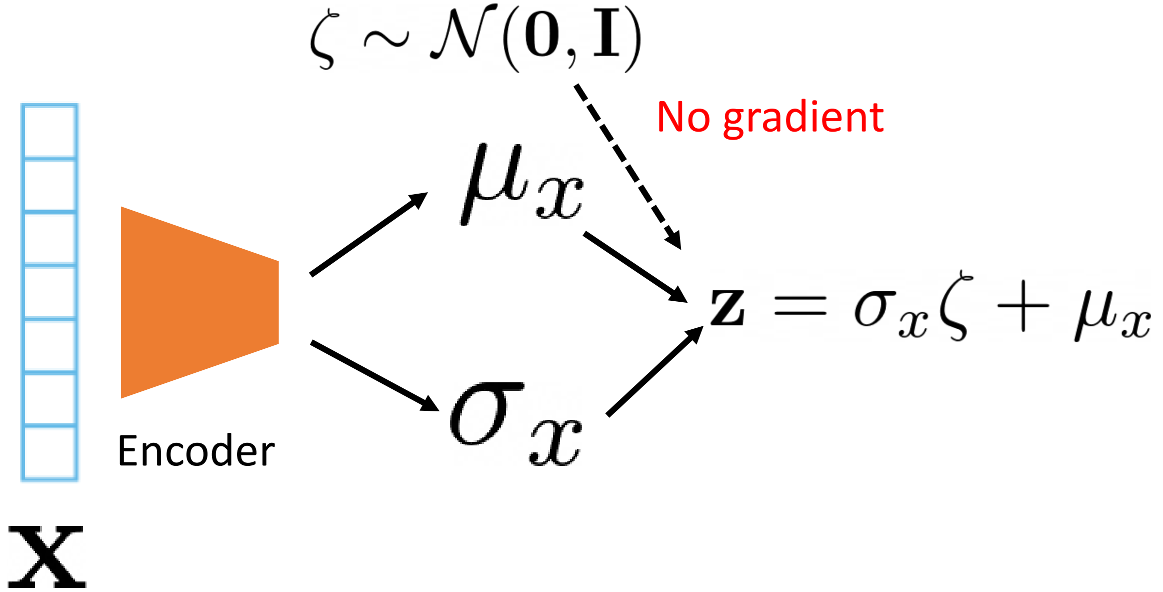
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder

- Encoder

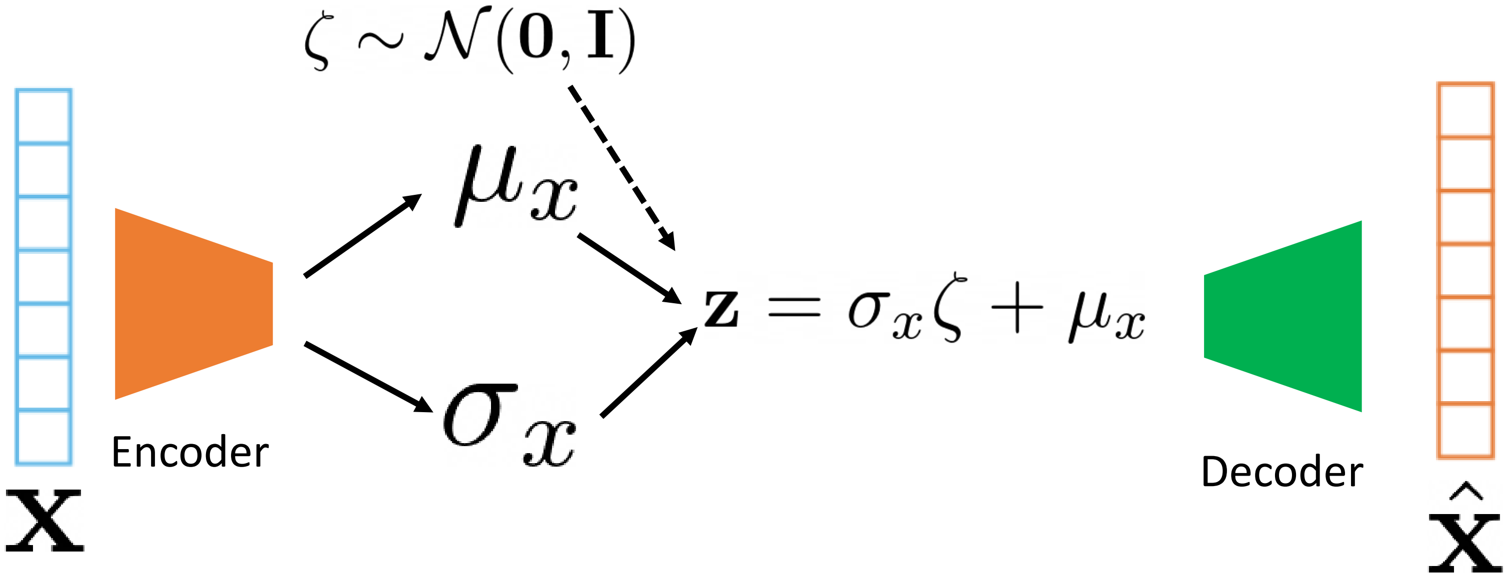


Reparameterization



Variational Autoencoder

- Encoder-Decoder



- Loss function

$$L = C\|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \mathrm{KL}(\mathcal{N}(\mu_x, \sigma_x), \mathcal{N}(\mathbf{0}, \mathbf{I}))$$

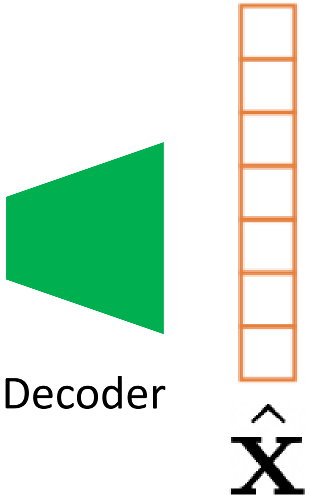
Reconstruction loss

Prior loss $D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$

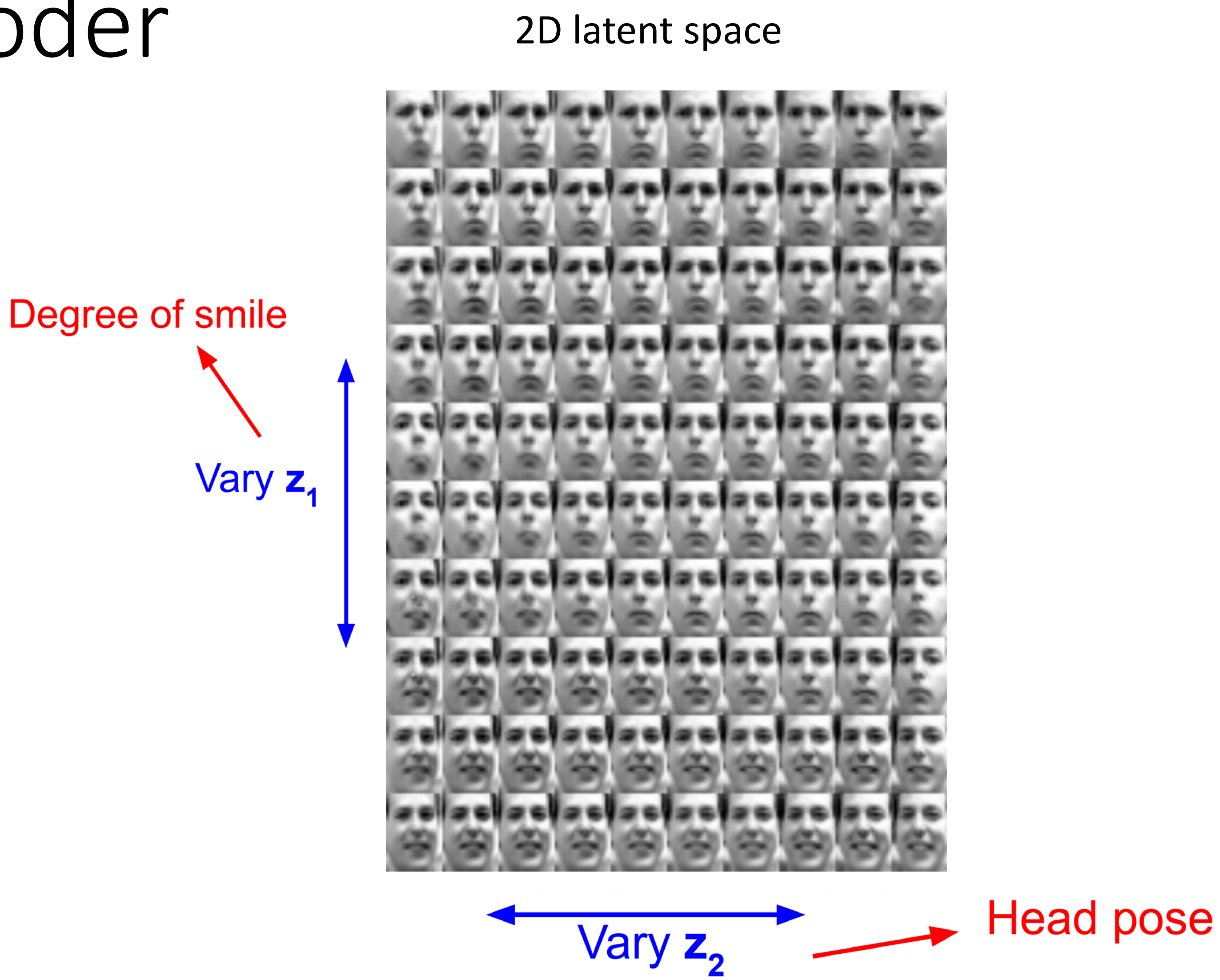
Variational Autoencoder

- Generating data

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$



- Diagonal prior on z -> independent latent variables
- Different dimensions of z encode interpretable factors of variation



Auto-Encoding Variational Bayes. Kingma & Welling, ICLR'14.

Direct Content Generation

- VAE models the density as

$$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$$

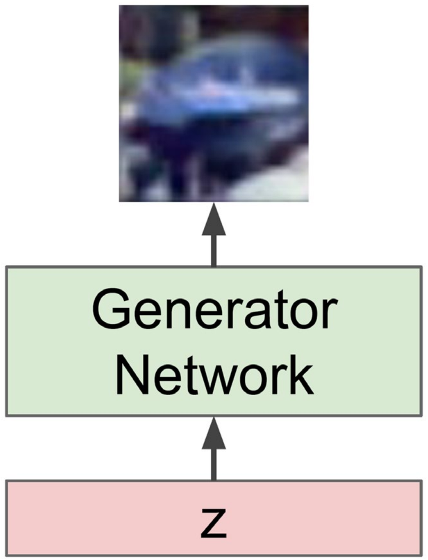
- Directly sample from the training distribution without modeling the probability density

- Generative Adversarial Networks (GANs) can generate better samples compared to VAEs

Generative Adversarial Network (GAN)

- Goal: sample examples from training distribution $P(\mathbf{x})$
- Solution
  - First sample from a simple distribution (e.g., uniform distribution)
  - Learn transformation to the training distribution

Output: sample from the training distribution



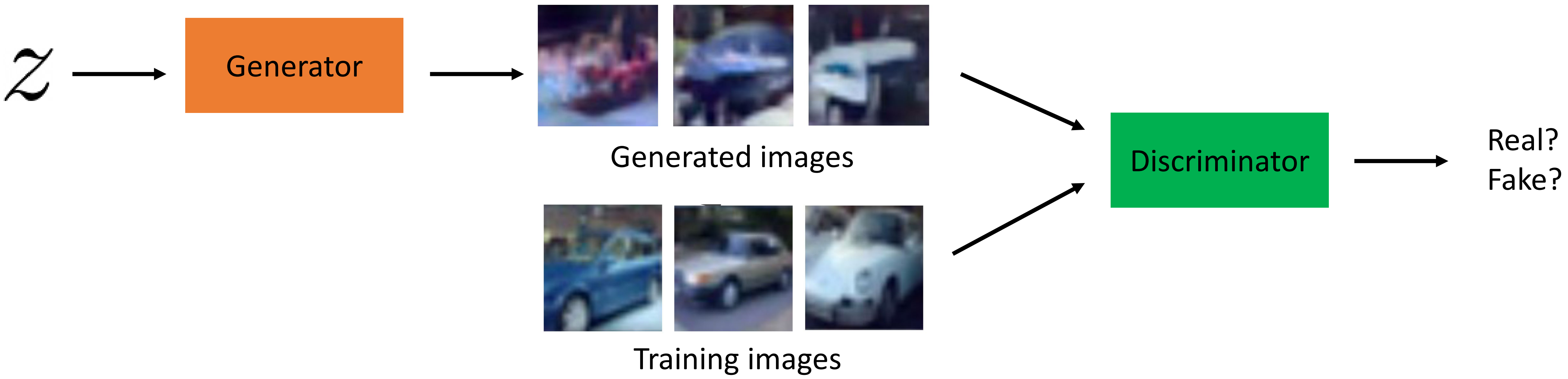
Input: random noise

How to train the generator?
- We do not know the mapping from z to training data

Generative Adversarial Network (GAN)

- Generator-Discriminator

Training GAN: Two-player Game



- Discriminator: try to distinguish between real image and fake images (generated images from the generator)
- Generator: try to fool the discriminator by generating real-look images

Training GAN: Two-player Game

- Minmax objective function

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

Discriminator output
for real data x
- Likelihood in (0, 1)

Discriminator output
for generated fake data

Generator output

- Discriminator: **maximize** the objective such that D(x) is close to 1 and D(G(z)) is close to 0
- Generator: **minimize** the objective such that D(G(z)) is close too 1 (fool the discriminator)

Generative Adversarial Nets. Goodfellow et al. NeurIPS'14

Training GAN: Two-player Game

- Minmax objective function

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

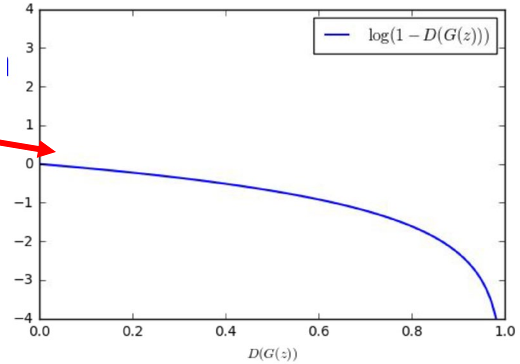
- Alternate between
  - Gradient ascent on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

  - Gradient descent on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Gradient is relative flat



Generative Adversarial Nets. Goodfellow et al. NeurIPS'14

Training GAN: Two-player Game

- Minmax objective function

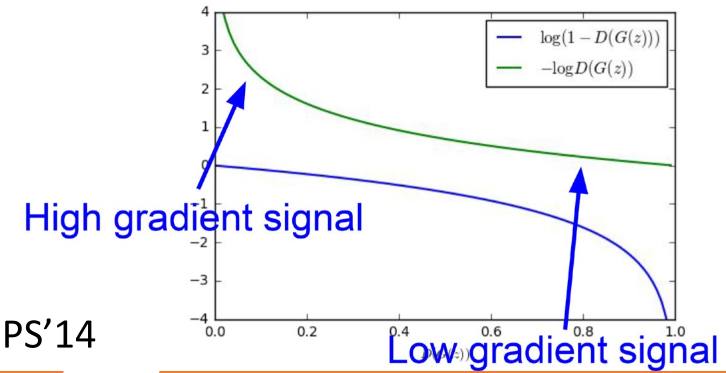
$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- Alternate between
    - Gradient ascent on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

    - Gradient ascent on generator

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



Generative Adversarial Nets. Goodfellow et al. NeurIPS'14

Training GAN: Two-player Game

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

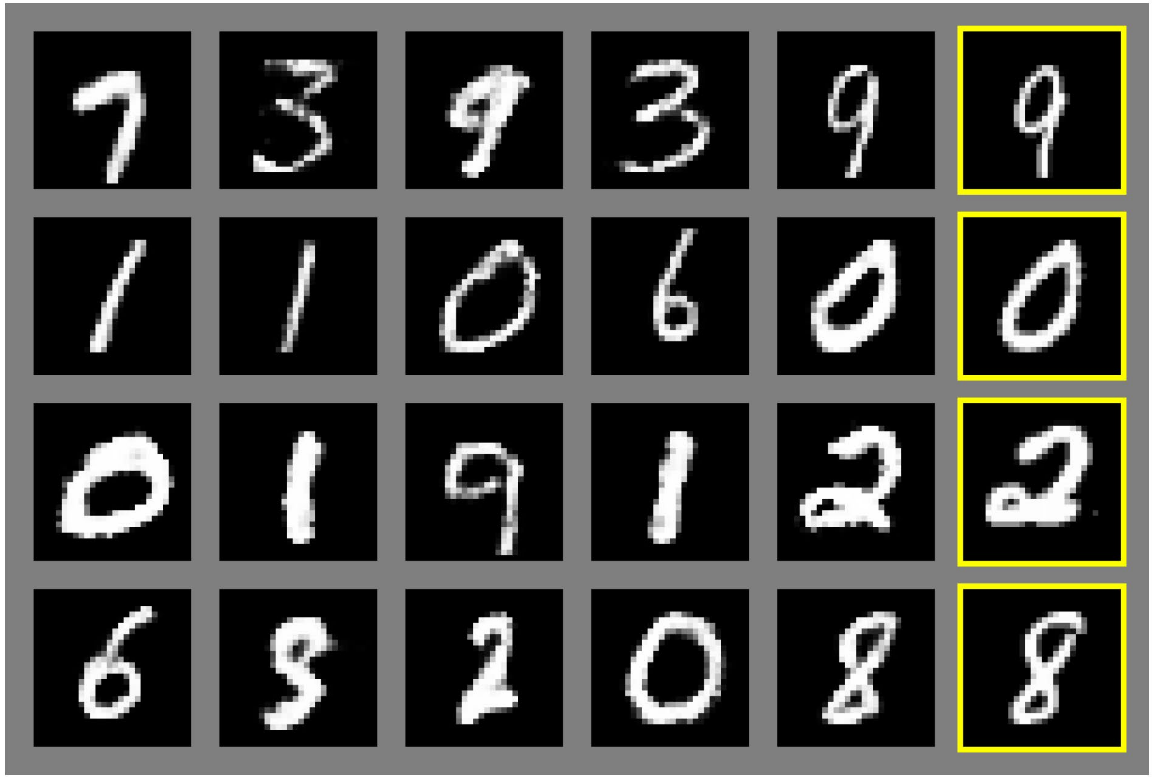
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

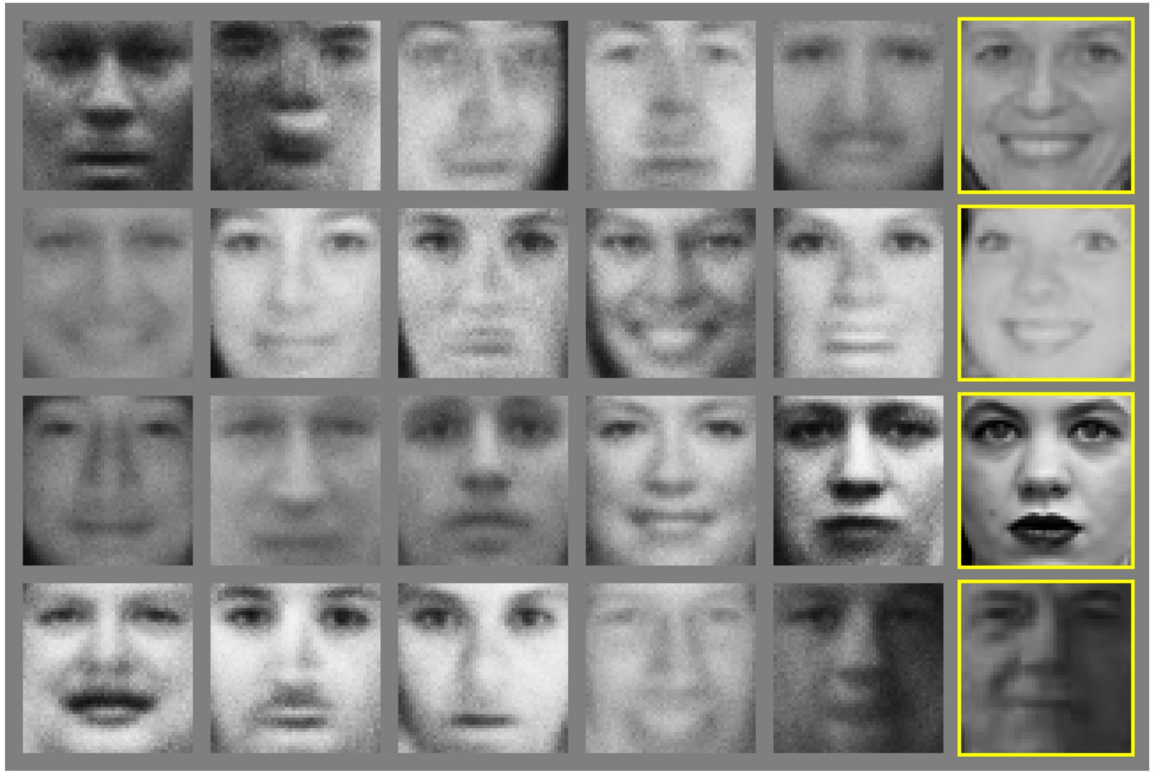
end for

Generative Adversarial Nets. Goodfellow et al. NeurIPS'14

Generative Adversarial Network (GAN)

Visualization of samples from the model



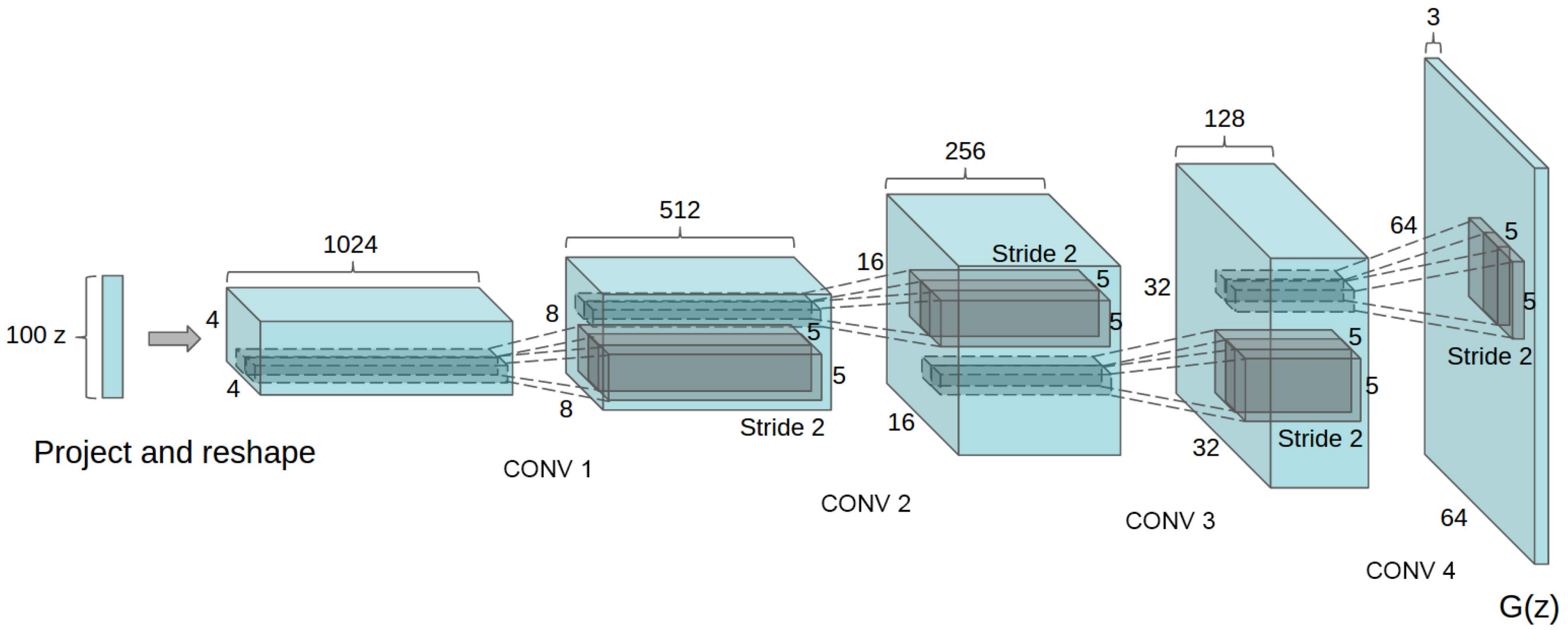


Nearest neighbor from training set

Generative Adversarial Nets. Goodfellow et al. NeurIPS'14

Deep Convolutional GANs (DCGANs)

- Use CNNs for generator and discriminator



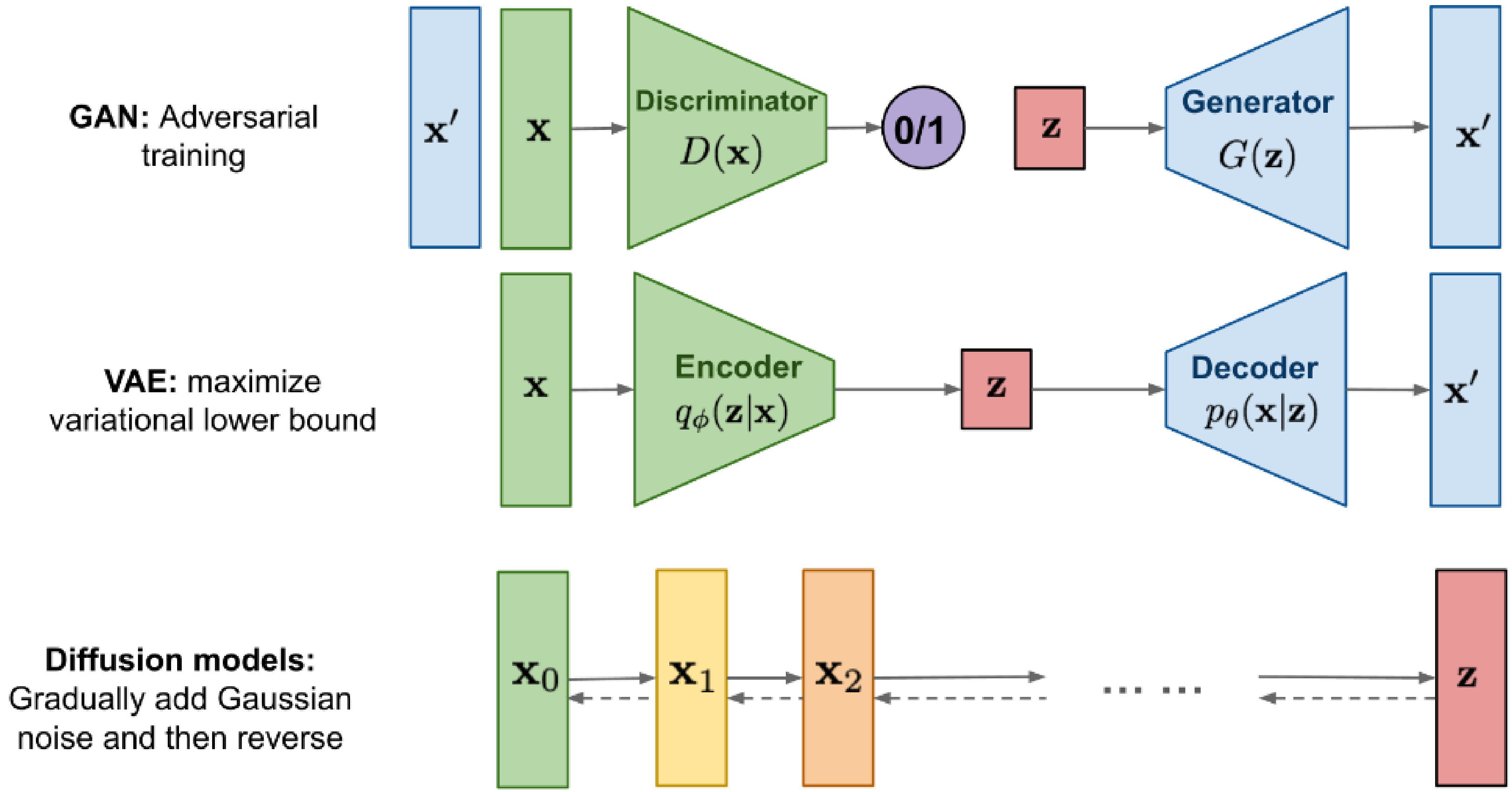
UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS. Radford et al., ICLR'16

Deep Convolutional GANs (DCGANs)

Generated samples

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS. Radford et al., ICLR'16

Diffusion Model



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Summary

- Autoencoder
  - Good for dimension reduction, cannot generate new data
- Variational autoencoder
  - Probabilistic formulation
  - Regularized latent space, can be used to generate new data
- Generative Adversarial Network
  - Directly sample training distribution to generate data
  - Better samples compared VAEs

Further Reading

- A Tutorial on Principal Component Analysis. Jonathon Shlens, 2014. https://arxiv.org/abs/1404.1100
- Auto-Encoding Variational Bayes. Kingma & Welling, ICLR, 2004. https://arxiv.org/abs/1312.6114
- Autoencoders. Dor Bank, Noam Koenigstein, Raja Giryes, 2021. https://arxiv.org/abs/2003.05991
- Generative Adversarial Nets. Goodfellow et al. NeurIPS'14. https://arxiv.org/abs/1406.2661
- UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS. Radford et al., ICLR'16. https://arxiv.org/abs/1511.06434