In The Name Of God

Subject: Generative Models

Presenter: Reza Karimzadeh

اولین کنگره بین المللی

فناوریهای پیشرفته در حوزه سلامت و کاربردهای هوش مصنوعی در پزشکی

ICAHT 2021

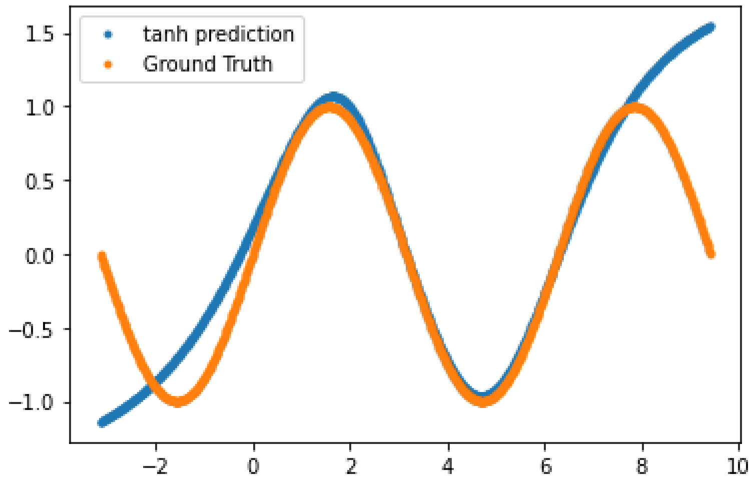
The First International Congress on

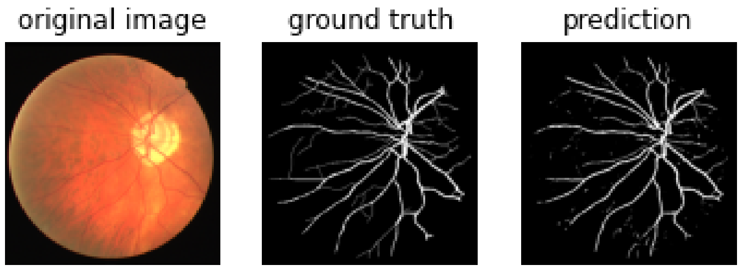
Advanced Health Technologies-Artificial Intelligence in Medicine

Supervised Learning

- ❖ Available pairs of data (X, Y)
  - X: input
  - Y: label

- ❖ Learn a function for mapping X to Y
- ❖ Examples:
  - regression
  - classification
  - segmentation
  - image captioning
  - Etc.

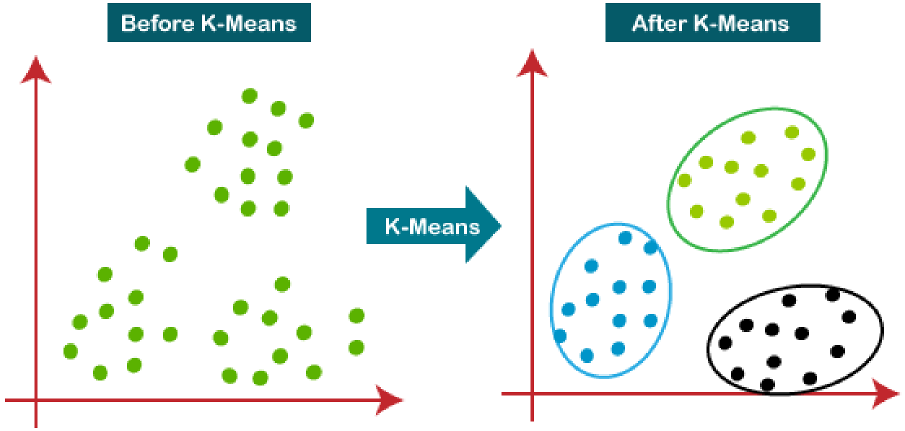


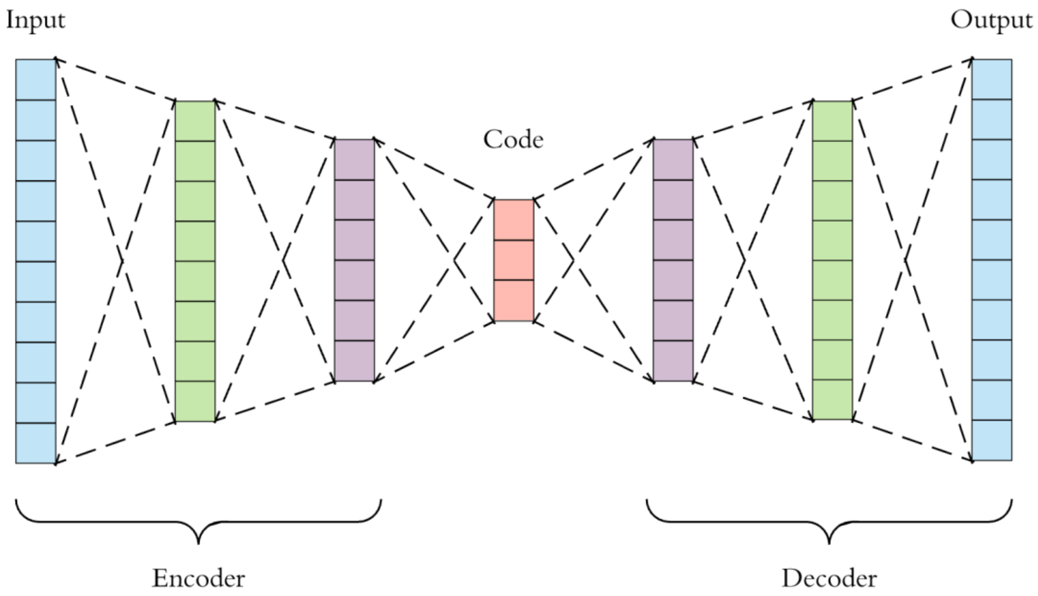


Unsupervised Learning

- ❖ Available data just X
  - No label

- ❖ Learn hidden structure of data
- ❖ Examples:
  - clustering
  - dimensionality reduction
  - feature learning
  - etc.

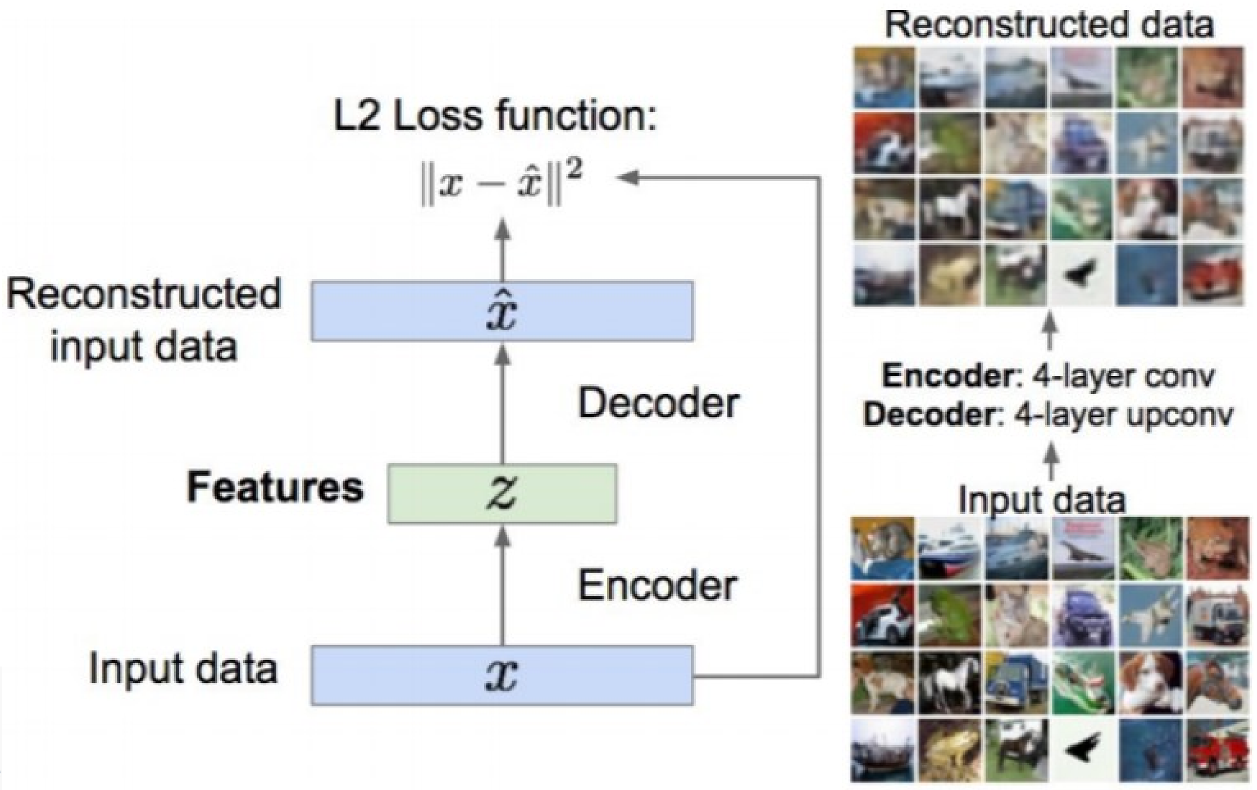




Autoencoders

- ❖ map input data (X) to a lower diminution (Z)
- ❖ reconstruct data ($\hat{X}$) from extracted features (Z)

- ❖ L2-norm for cost function



Autoencoders

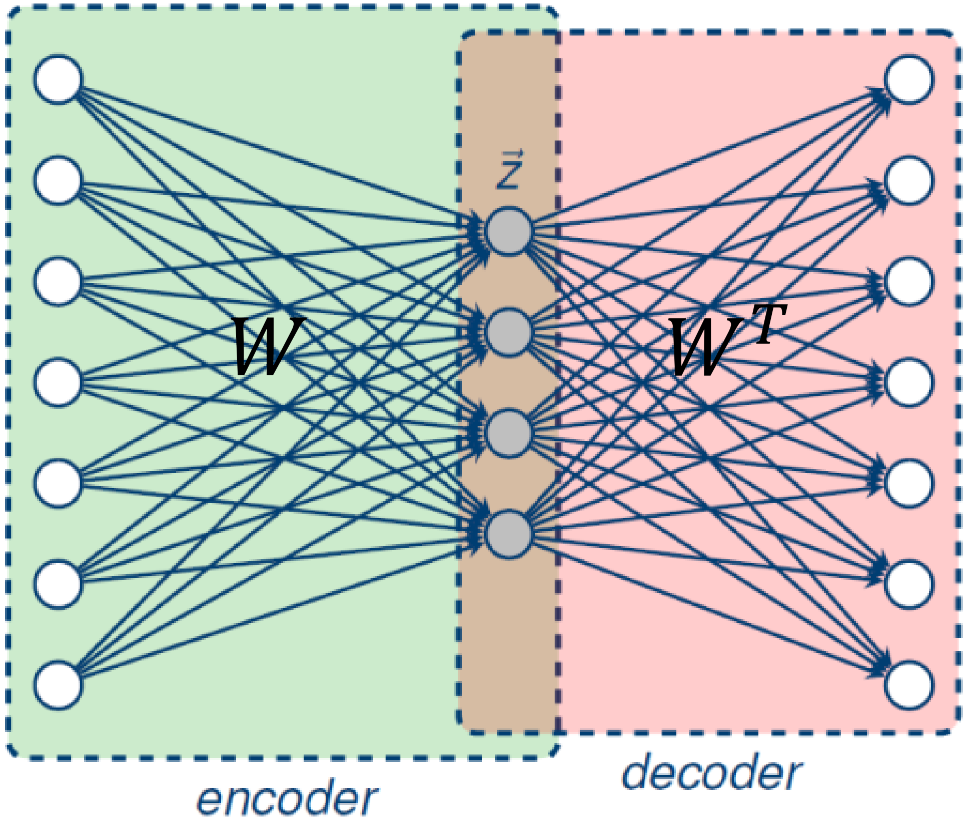
- consider a simple AE

$$\hat{X} = W^T W X$$

$$Loss = \left\|\hat{X} - X\right\|^2 = \|W^T W X - X\|^2$$

$$W^* = \min_{W} \|W^T W X - X\|^2$$

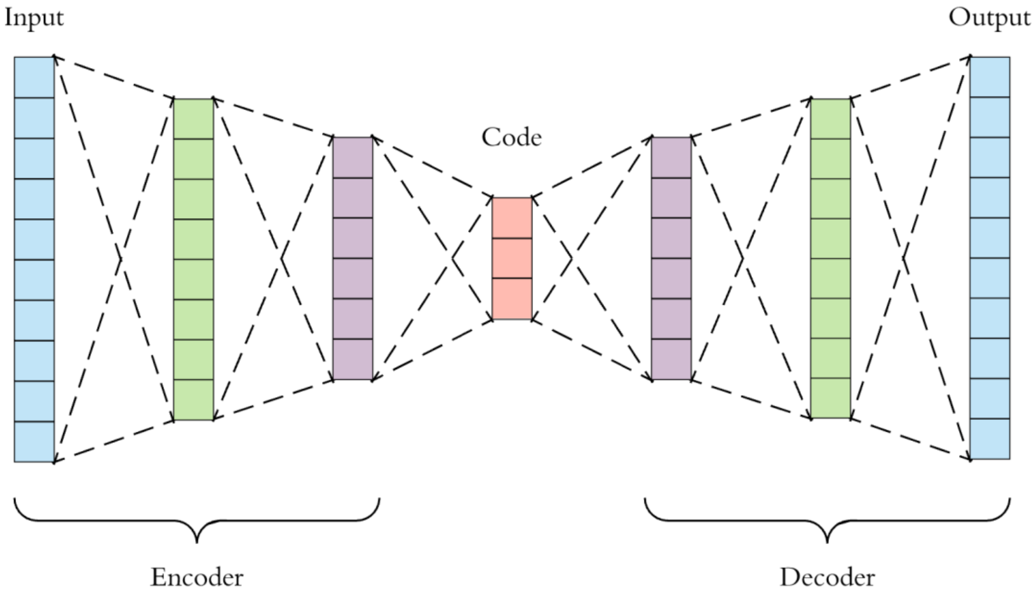
- It is PCA !!

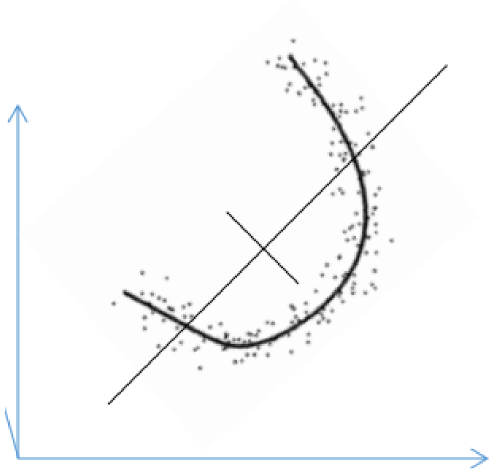


Autoencoders

❖ We add some nonlinearity to Autoencoders i.e. activation function
  - Nonlinear PCA

❖ Deeper networks can capture more complicated manifolds → deep autoencoder



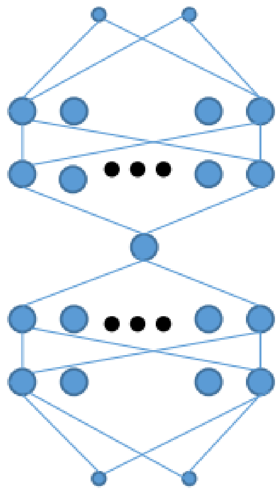


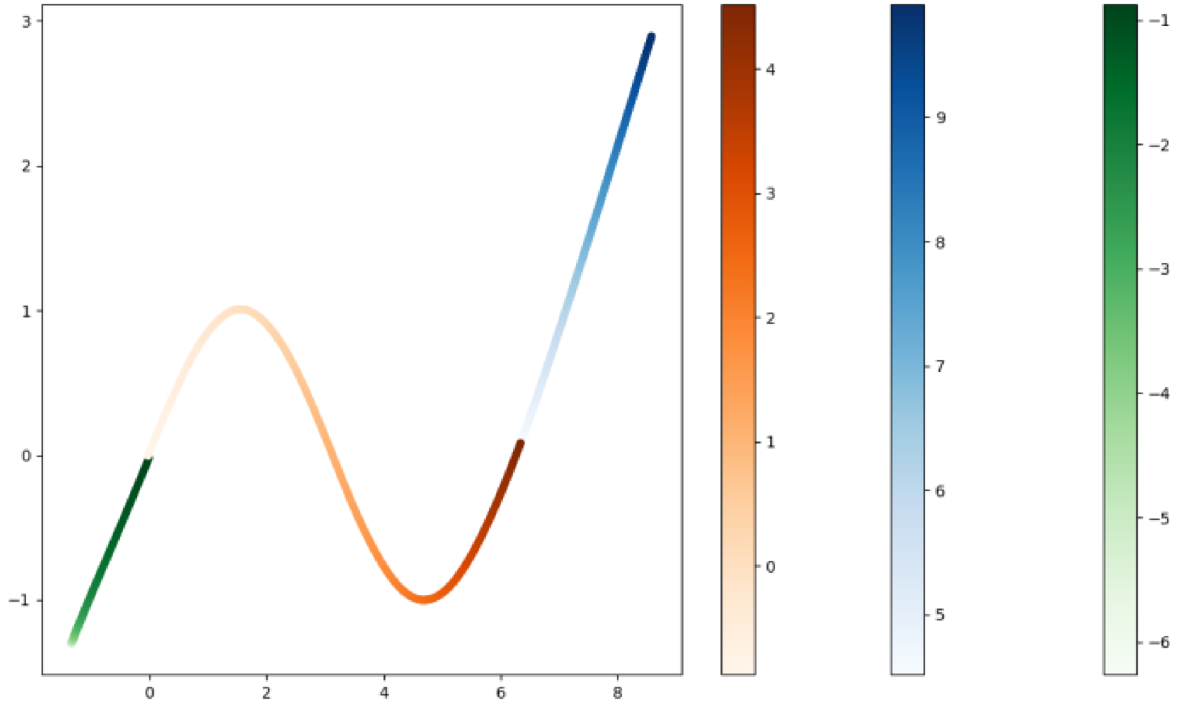
Autoencoders

❖ AE learn to reconstruct sin function in [0, 2*pi]

❖ if we keep decoder part and vary the latent space (Z) the output will be a sin function in [0, 2*pi] but in other parts is non-sin

❖ Is it a generative model?

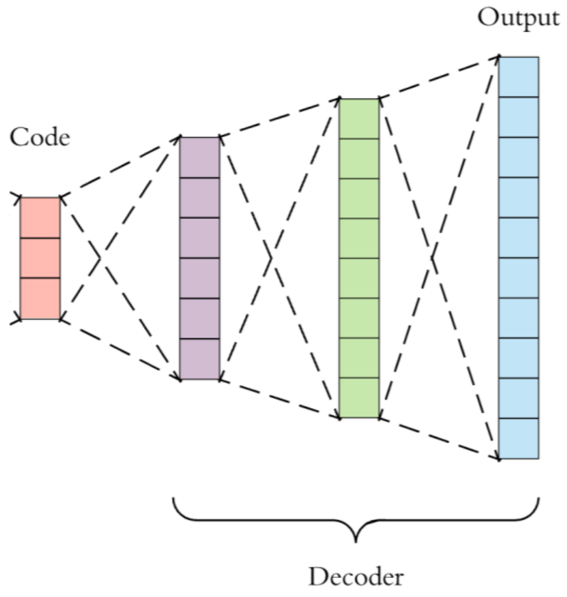




Autoencoders

❖ Is it a generative model?

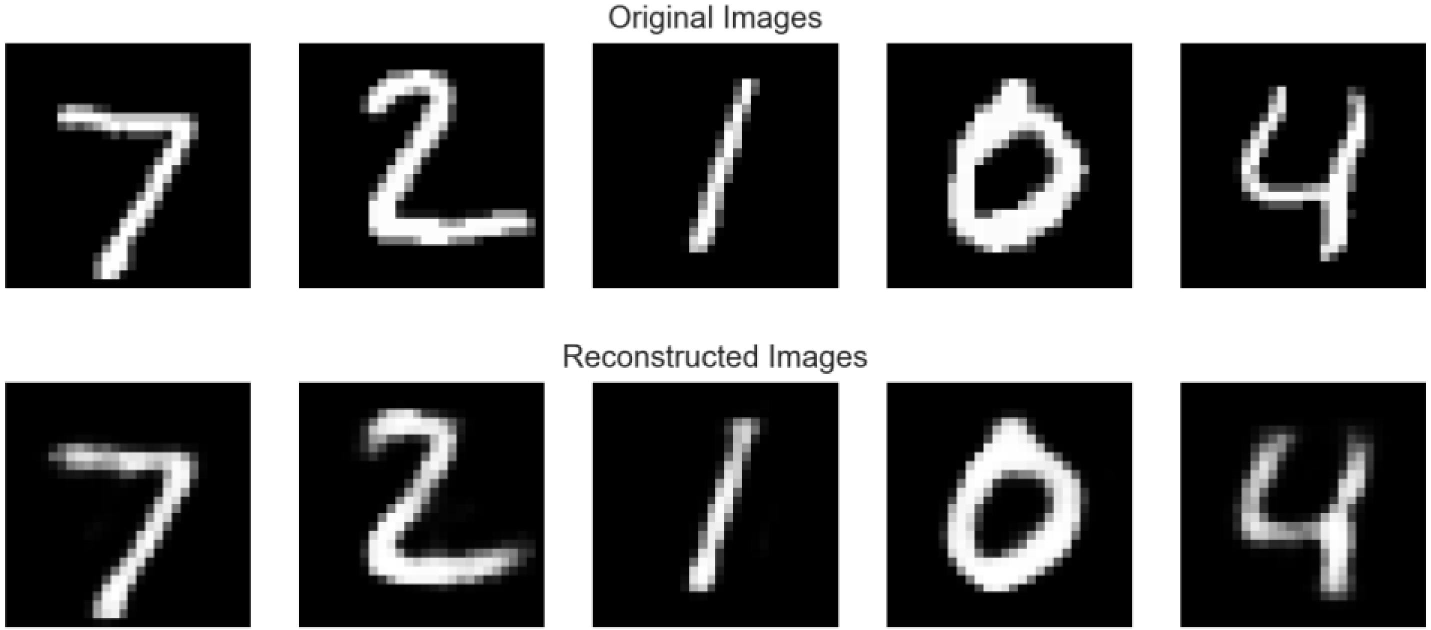
➢ Decoder only can generate data that lie on training data manifold
➢ Samples generated from distribution of training data
➢ Samples are similar to training data



Autoencoders (applications)

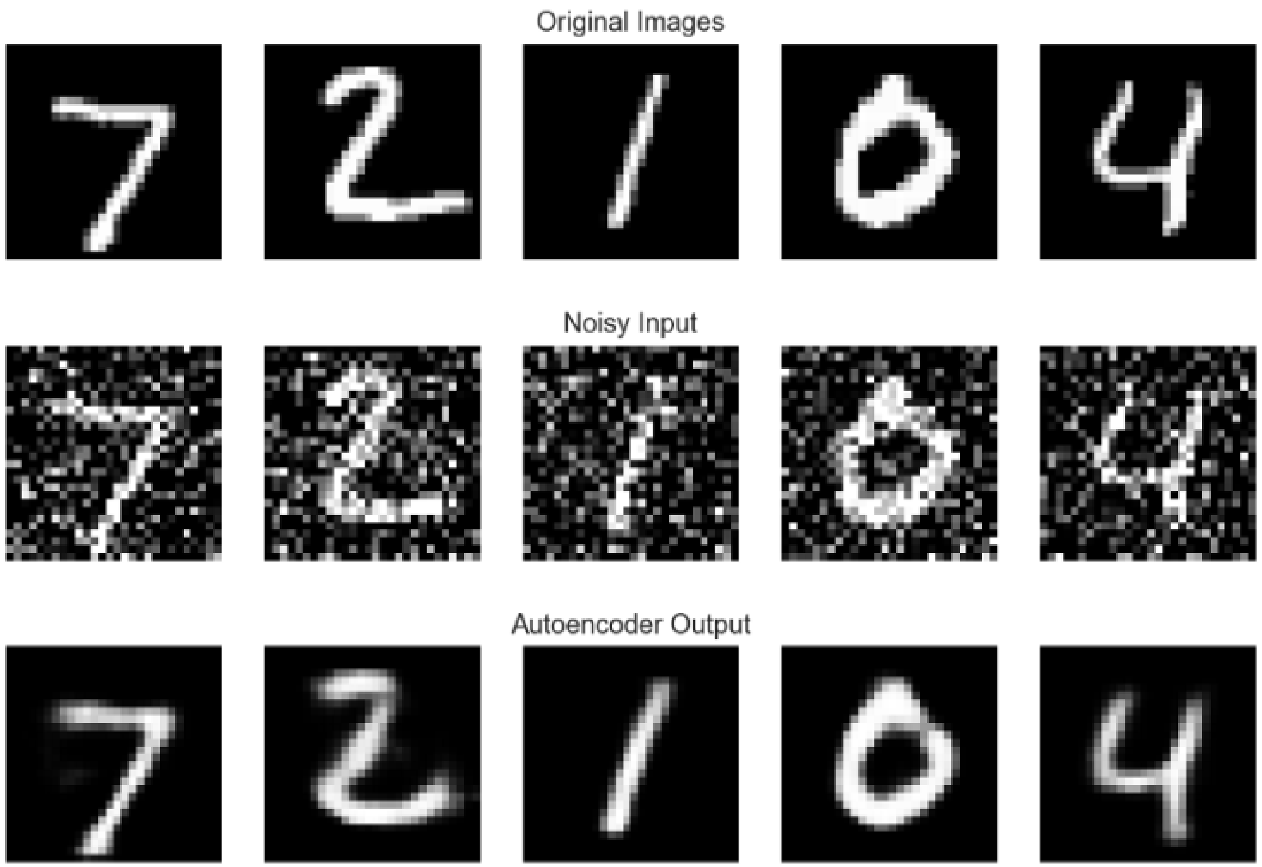
❖ Low dimensional feature extraction

784D → 32D → 784D



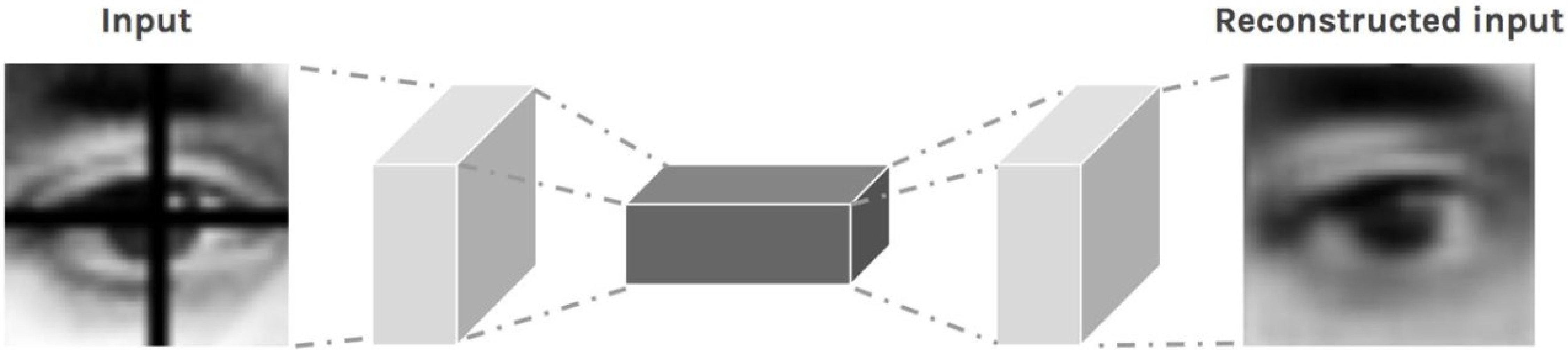
Autoencoders (applications)

- ❖ Denoising



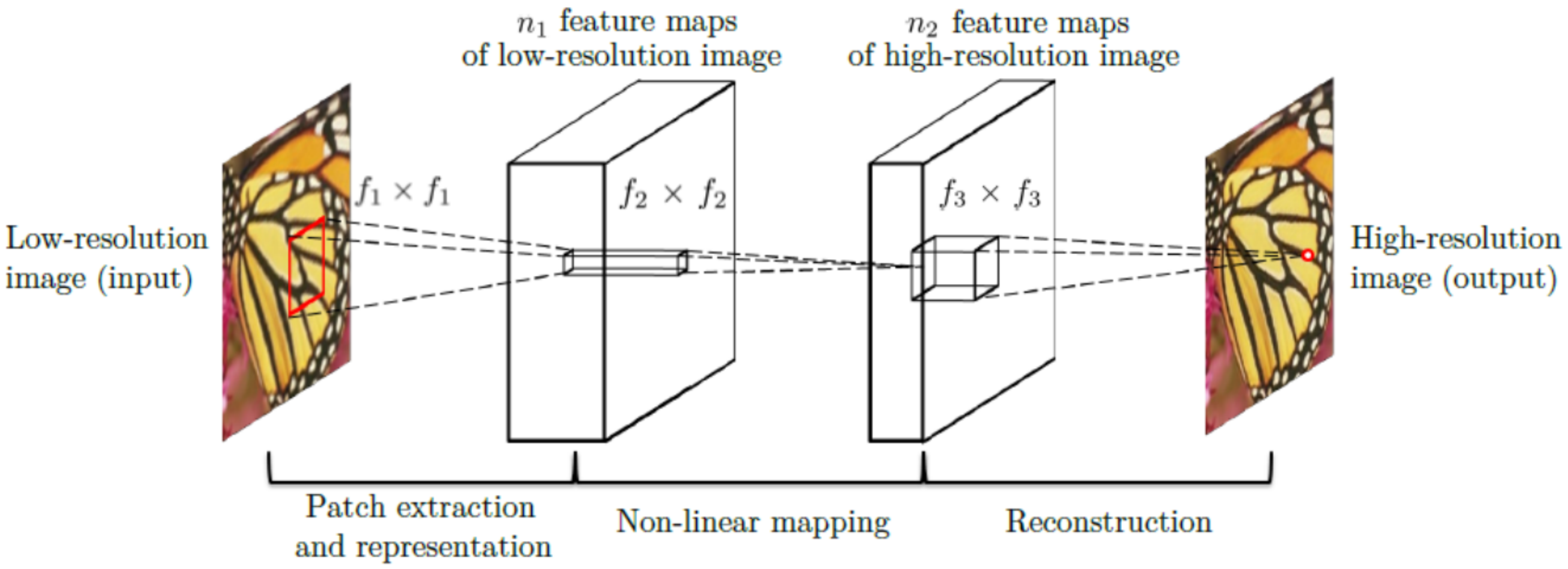
Autoencoders (applications)

❖ Image reconstruction



Autoencoders (applications)

❖ Super Resolution



Coupled Deep Autoencoder for Single Image Super-Resolution, IEEE, ITC, 2015

Variational Autoencoder

Generative Models

- ❖ Explicitly working with the distribution
  - Variational autoencoders

- ❖ Implicit generative model (Generative Adversarial Networks)
  - trained without even needing to explicitly define a density functions

Variational Autoencoder

Let's define some notions:

1. X: data that we want to model a.k.a the animal
2. z: latent variable a.k.a our imagination
3. $P(X)$: probability distribution of the data, i.e. that animal kingdom
4. $P(z)$: probability distribution of latent variable, i.e. our brain, the source of our imagination
5. $P(X|z)$: distribution of generating data given latent variable, e.g. turning imagination into real animal

Variational Autoencoder

❖ Our objective here is to model the data, hence we want to find P(X). Using the law of probability, we could find it in relation with z as follows:

$$P(X) = \int P(X|z)P(z)dz$$

❖ VAE idea: Infer P(z) from P(z|X)

- estimate latent variable likely using our data

❖ How to infer P(z|X) ?

- infer P(z|X) using a method called Variational Inference (VI)
- The main idea of VI is to pose the inference by approach it as an optimization problem By modeling the true distribution P(z|X) using simpler distribution that is easy to evaluate, e.g. Gaussian,

Variational Autoencoder

❖ Let infer $P(z|X)$ using Q(z|X) via KL distance:

$$D_{KL}[Q(z|X)\|P(z|X)] = \sum_z Q(z|X)\,\log\frac{Q(z|X)}{P(z|X)}$$

$$= E\left[\log\frac{Q(z|X)}{P(z|X)}\right]$$

$$= E[\log Q(z|X) - \log P(z|X)]$$

Variational Autoencoder

- Let use P(X), P(X|z), and P(z), via Bayes Rule

$$D_{KL}[Q(z|X)\|P(z|X)] = E\left[\log Q(z|X) - \log \frac{P(X|z)P(z)}{P(X)}\right]$$

$$= E[\log Q(z|X) - (\log P(X|z) + \log P(z) - \log P(X))]$$

$$= E[\log Q(z|X) - \log P(X|z) - \log P(z) + \log P(X)]$$

Expectation is over z, this moves P(X) outside

Variational Autoencoder

$$D_{KL}[Q(z|X)\|P(z|X)] = E[\log Q(z|X) - \log P(X|z) - \log P(z)] + \log P(X)$$

$$D_{KL}[Q(z|X)\|P(z|X)] - \log P(X) = E[\log Q(z|X) - \log P(X|z) - \log P(z)]$$

- If we look carefully at the right hand side of the equation, we would notice that it could be rewritten as another KL divergence.

Variational Autoencoder

❖ rewritten another KL divergence on the right hand side by rearranging sign.

$$D_{KL}[Q(z|X)\|P(z|X)] - \log P(X) = E[\log Q(z|X) - \log P(X|z) - \log P(z)]$$

$$\begin{aligned}\log P(X) - D_{KL}[Q(z|X)\|P(z|X)] &= E[\log P(X|z) - (\log Q(z|X) - \log P(z))] \\ &= E[\log P(X|z)] - E[\log Q(z|X) - \log P(z)] \\ &= E[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)]\end{aligned}$$

Variational Autoencoder

❖ VAE objective function:

$$\log P(X) - D_{KL}[Q(z|X) \| P(z|X)] = E[\log P(X|z)] - D_{KL}[Q(z|X) \| P(z)]$$

❖ So we have got:

1. Q(z|X) that project our data X into latent variable space → Encoder
2. z, the latent variable → Generated code
3. P(X|z) that generate data given latent variable → Decoder network

❖ VAE tries to find the lower bound of logP(X) (ELBO (Evidence Lower Bound))

Objective: **Maximizing** *logP(X/z)* and **minimizing** KL distance between our simple distribution *Q(z/X)* and the true latent distribution *P(z)*.

Variational Autoencoder

- How to solve?

$$\log P(X) - D_{KL}[Q(z|X)\|P(z|X)] = E[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)]$$

- **Maximizing** *logP(X/z)*: Any discriminative supervised-parametric model (SVM/MLP/…)

- Any classifier with z as input and X as output, and a proper loss

Variational Autoencoder

❖ How to solve?

$$\log P(X) - D_{KL}[Q(z|X)\|P(z|X)] = E[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)]$$

❖ minimizing $D_{KL}[Q(z|X)\|P(z)]$:

- $P(z)$: Latent variable pdf, we will sample it to generate X. → $P(z) \sim N(0,1)$
- Easy to make $Q(z|X)$ as-close-as possible to it
- $Q(z|X)$: Another simplification: $N(\mu(X), \Sigma(X))$, and diagonal, $\Sigma(X)$!
- Easy to compute KL between two Gaussian!

Variational Autoencoder

- How to solve?

$$\log P(X) - D_{KL}[Q(z|X)\|P(z|X)] = E[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)]$$

- minimizing $D_{KL}[Q(z|X)\|P(z)]$:
  - $P(z) \sim N(0,1)$, $Q(z|X) \sim N(\mu(X), \Sigma(X))$

$$D_{KL}[N(\mu(X),\Sigma(X))\|N(0,1)] = \frac{1}{2}\left(\mathrm{tr}(\Sigma(X)) + \mu(X)^T\mu(X) - k - \log\det(\Sigma(X))\right)$$

Variational Autoencoder

- minimizing $D_{KL}[Q(z|X) \| P(z)]$:
  - $P(z) \sim N(0,1)$, $Q(z|X) \sim N(\mu(X), \Sigma(X))$

$$D_{KL}[N(\mu(X),\Sigma(X))\|N(0,1)] = \frac{1}{2}\left(\mathrm{tr}(\Sigma(X)) + \mu(X)^T\mu(X) - k - \log\det(\Sigma(X))\right)$$

$$D_{KL}[N(\mu(X),\Sigma(X))\|N(0,1)] = \frac{1}{2}\left(\sum_k \Sigma(X) + \sum_k \mu^2(X) - \sum_k 1 - \log\prod_k \Sigma(X)\right)$$

$$= \frac{1}{2}\left(\sum_k \Sigma(X) + \sum_k \mu^2(X) - \sum_k 1 - \sum_k \log\Sigma(X)\right)$$

$$= \frac{1}{2}\sum_k \left(\Sigma(X) + \mu^2(X) - 1 - \log\Sigma(X)\right)$$

Variational Autoencoder

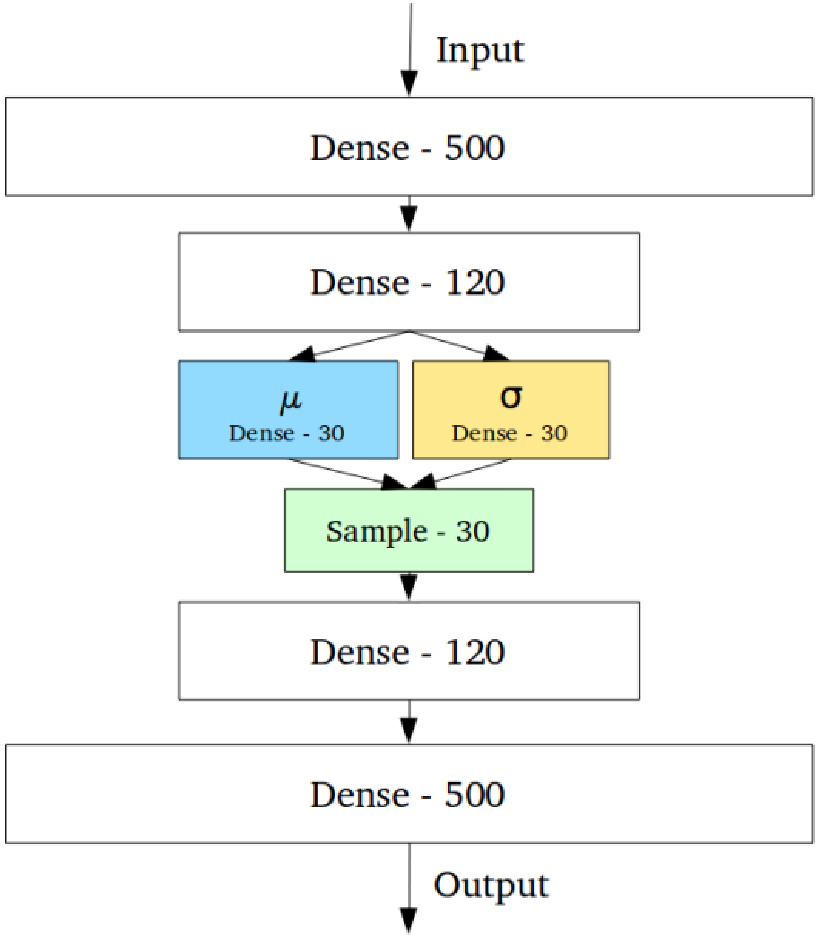
- minimizing $D_{KL}[Q(z|X) \| P(z)]$:
  - $P(z) \sim N(0,1)$, $Q(z|X) \sim N(\mu(X), \Sigma(X))$

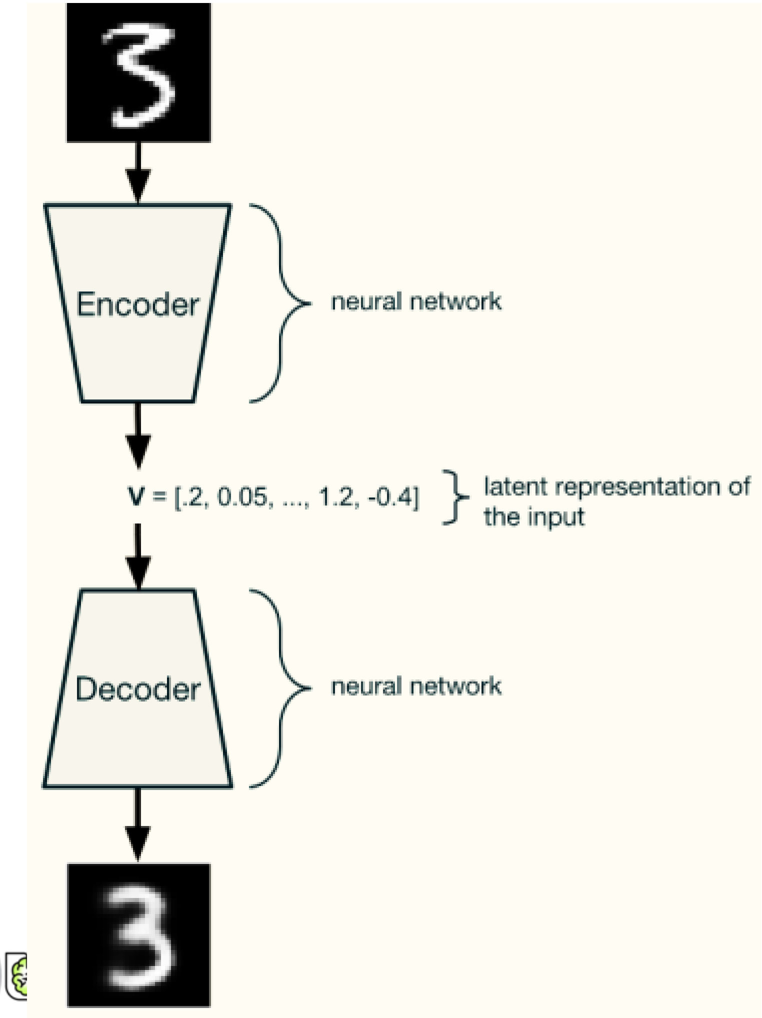
- In practice, however, it's better to model $\Sigma(X)$ as $\log\Sigma(X)$, as it is more numerically stable to take exponent compared to computing log.

$$D_{KL}[N(\mu(X), \Sigma(X)) \| N(0,1)] = \frac{1}{2} \sum_k \left(\exp(\Sigma(X)) + \mu^2(X) - 1 - \Sigma(X)\right)$$

Variational Autoencoder

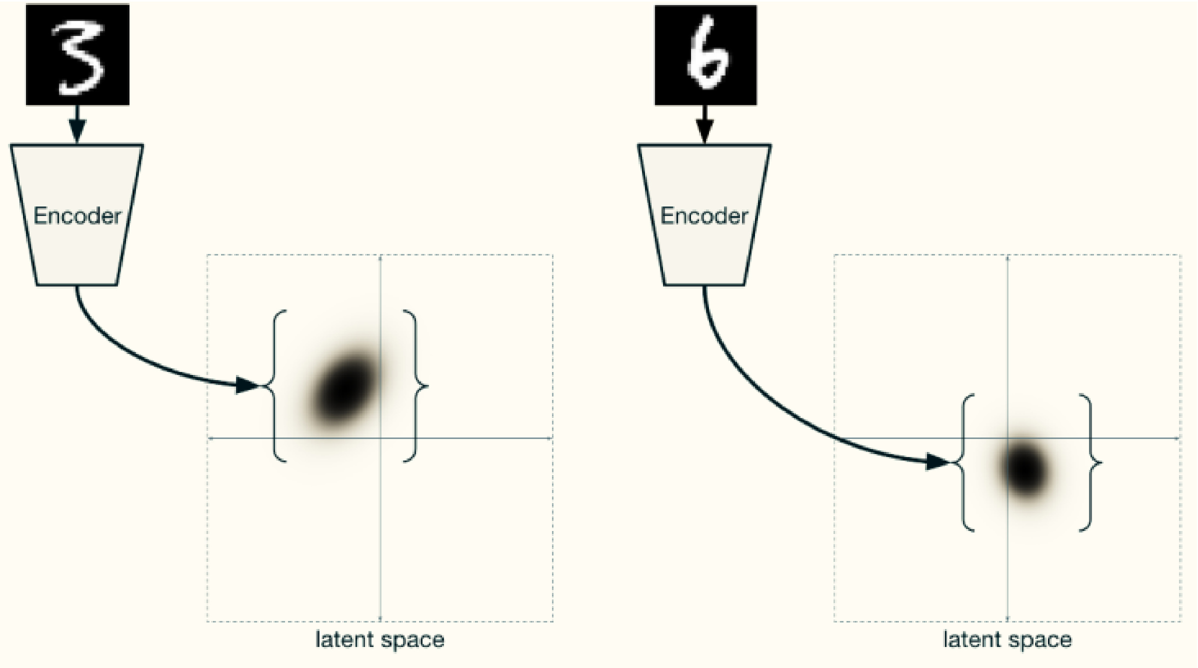
❖ Overall description

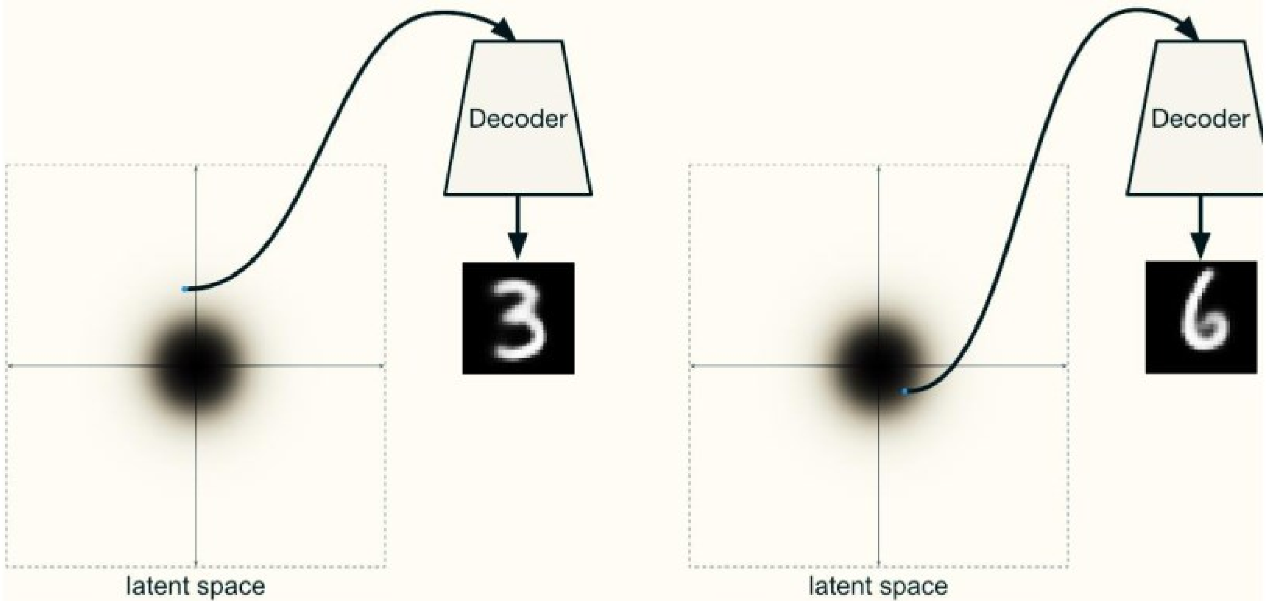




Variational Autoencoder

❖ Overall description





Variational Autoencoder

❖ References

- Auto-encoding variational bayes (https://arxiv.org/pdf/1312.6114.pdf?source=post_page---------------------------)
- https://agustinus.kristia.de/techblog/2016/12/10/variational-autoencoder/
- Attribute2image: Conditional image generation from visual attributes

Generative adversarial Networks (GANs)

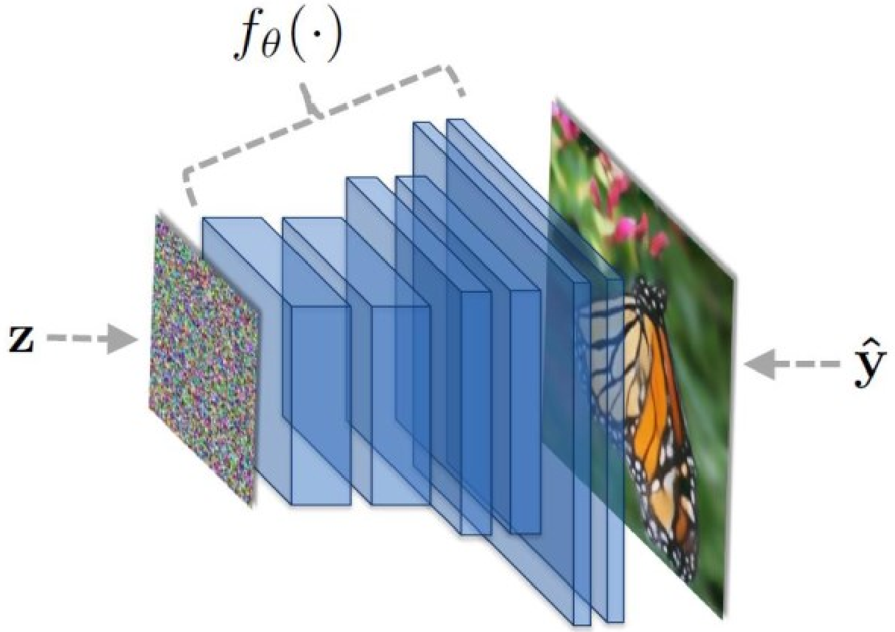
Generative adversarial Networks

- ❖ Explicitly working with the distribution
  - Variational autoencoders

- ❖ Implicit generative model (Generative Adversarial Networks)
  - trained without even needing to explicitly define a density functions

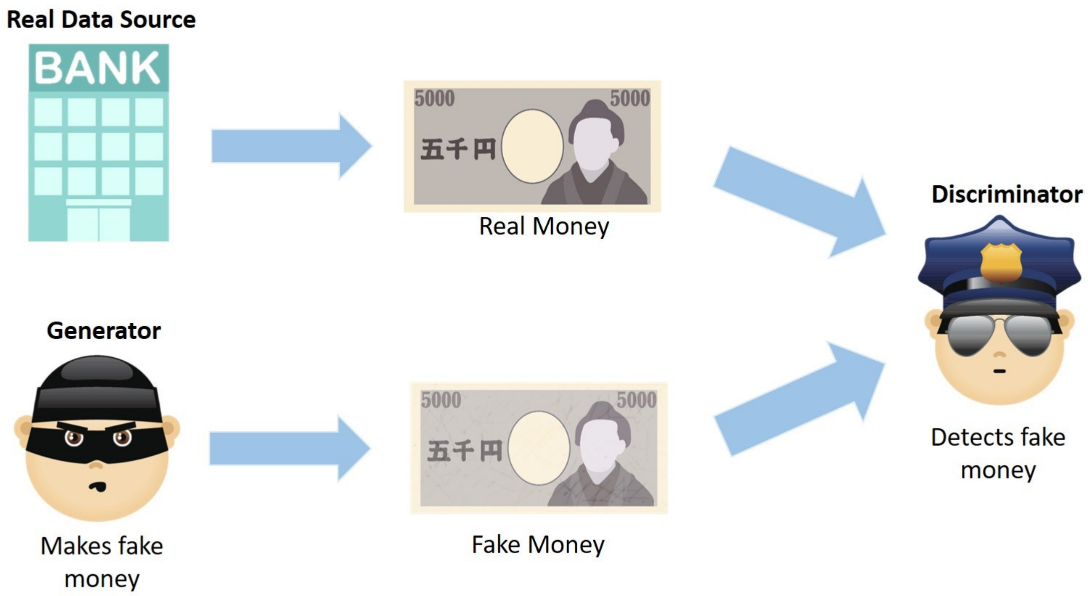
Generative adversarial Networks

- Sample from a simple distribution, e.g. random noise
- Then, learn transformation to training distribution



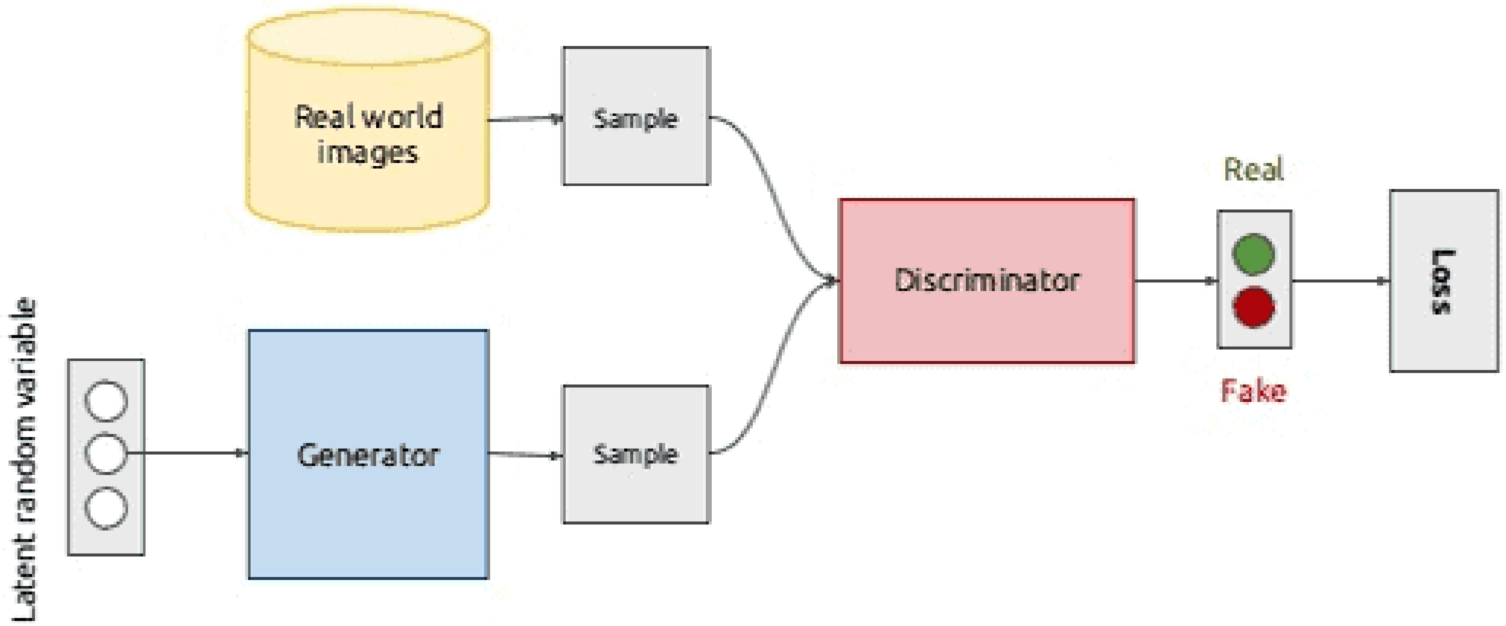
Generative adversarial Networks

- Generator network: try to fool the discriminator by generating real-looking images
- Discriminator network: try to distinguish between real and fake images



Generative adversarial Networks

- ❖ Generator network: try to fool the discriminator by generating real-looking images
- ❖ Discriminator network: try to distinguish between real and fake images



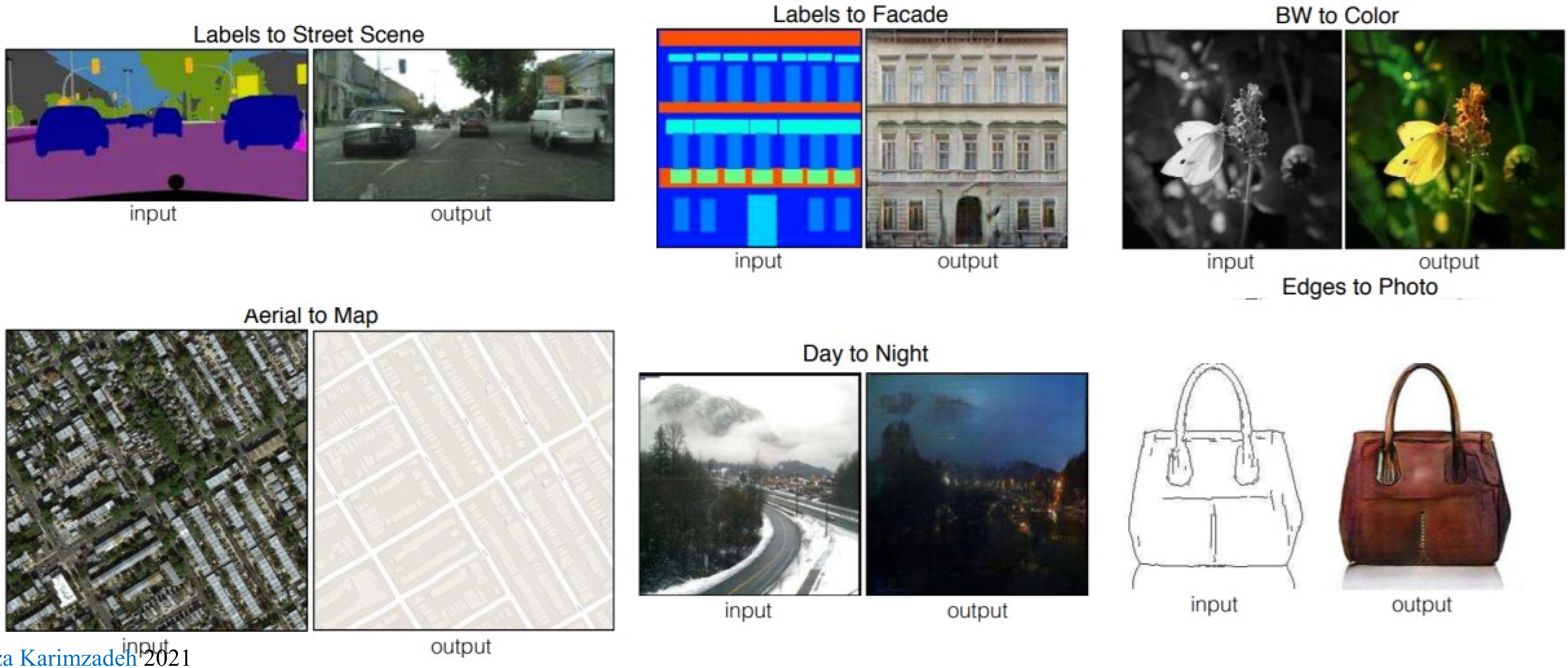
Generative adversarial Networks

- GAN:
  - Generative: Learn a Generative Model
  - Adversarial: Trained in an Adversarial Setting
  - Network: Use Deep Neural Networks

- GAN final goal:
  - Generation of samples from some distribution
  - Create Art
  - Image-to-Image translation (Aerial images to Maps)

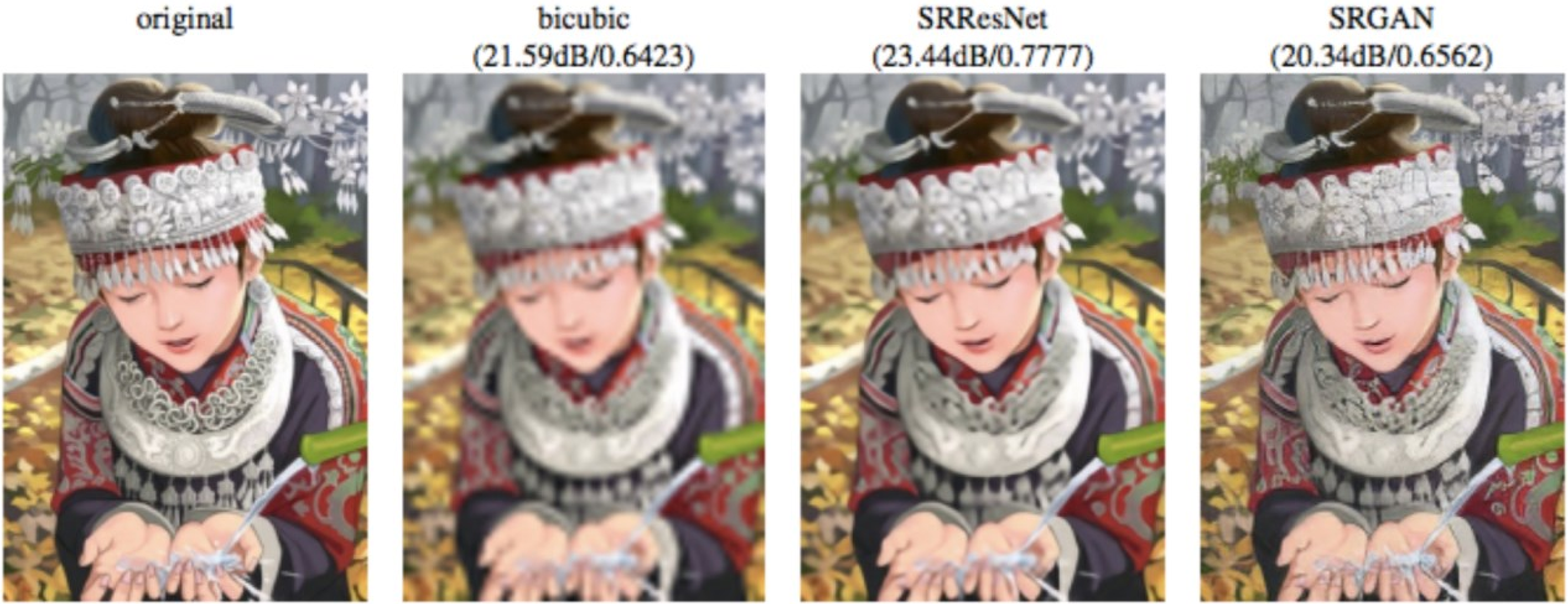
GANs output examples

Phillip Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017.



GANs output examples

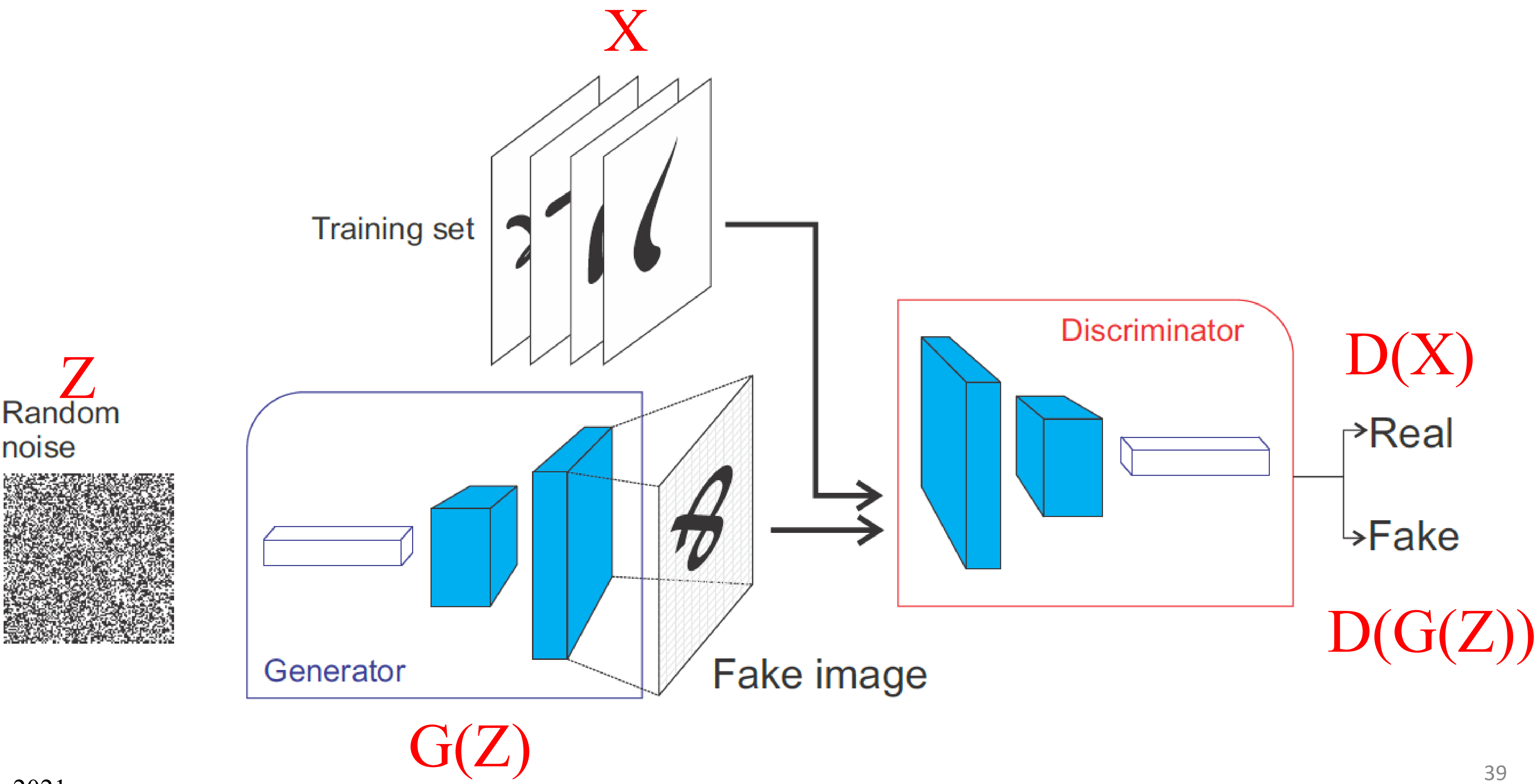
C. Ledig et al, Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network, CVPR 2017.



GANs output examples

https://thispersondoesnotexist.com/

Generative adversarial Networks



Generative adversarial Networks

- ❖ Generator network: intend to generate real-looking samples
- ❖ Discriminator network: intend to distinguish between real and fake samples

Discriminator output for real samples

Discriminator output for generated samples

$$J^{(D)} = \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log \boxed{D(\boldsymbol{x})}] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - \boxed{D(G(\mathbf{z}))})]$$

$$J^{(G)} = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log D(G(\mathbf{z}))]$$

$$\theta_D^* = \max_{\theta_D} \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_{\theta_D}(\boldsymbol{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(1 - D_{\theta_D}(G_{\theta_G}(\mathbf{z}))\right)\right]$$

$$\theta_G^* = \max_{\theta_G} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log\left(D_{\theta_D}(G_{\theta_G}(\mathbf{z}))\right)\right]$$

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014

Generative adversarial Networks

❖ A minimax game between generator and discriminator:

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

❖ Solving problem:

- ❖ 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

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

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014

Generative adversarial Networks

- A minimax game between generator and discriminator:

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

- Solving problem:
  - 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

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

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014

Generative adversarial Networks

❖ Training GANs:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014

Generative adversarial Networks

❖ Generated samples

(Rightmost column shows the nearest training example of the neighboring sample)

a) b) c) d)

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014

Any Question?

❖ Contact me!!
  Rezakarimzadeh1996@gmail.com

اولین کنگره بین المللی

فناوریهای پیشرفته در حوزه سلامت و کاربردهای هوش مصنوعی در پزشکی

ICAHT 2021

The First International Congress on

Advanced Health Technologies-Artificial Intelligence in Medicine