भारतीय प्रौद्योगिकी संस्थान दिल्ली
Indian Institute of Technology Delhi

COV877

Special Module on Visual Computing

Generative AI for Visual Content Creation: Image, Video, and 3D

Generative Adversarial Network (GAN)

Instructor:

Dr. Lokender Tiwari

Research Scientist

Logistics

Lecture timing
- Wednesday 3:30 PM - 5:00 PM
- Friday 5:00 PM - 6:30 PM

Venue: LH 521

Courese webpage : https://lokender.github.io/teaching/COV877.html

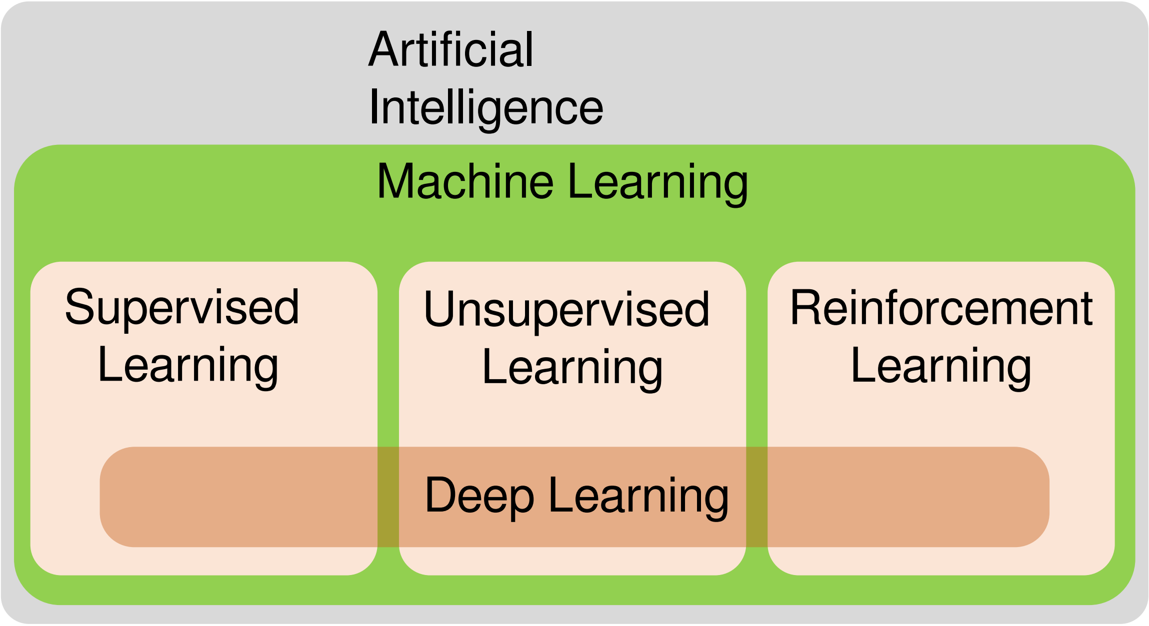
Reference Text Book (for first 2-3 lectures only)

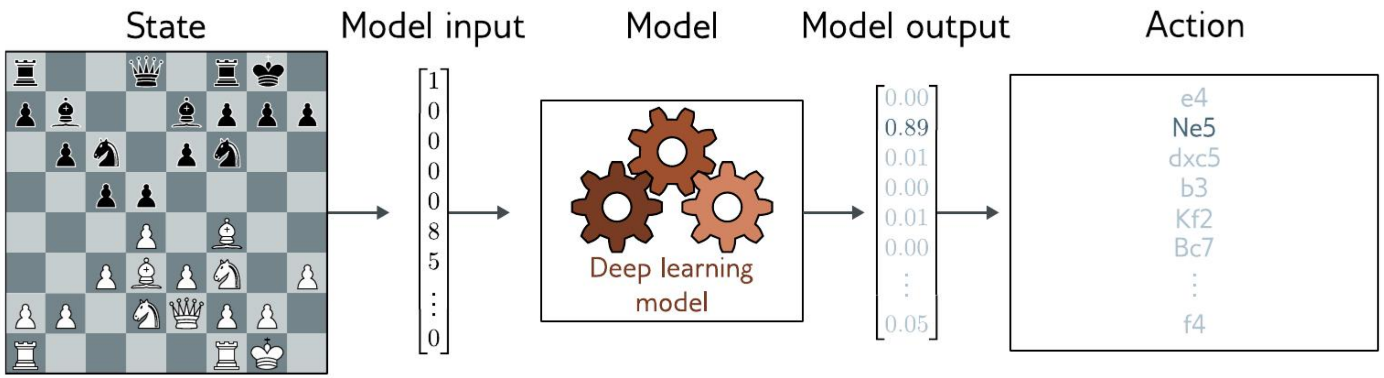
Understanding Deep Learning by Simon J.D Prince

https://udlbook.github.io/udlbook/

Taxonomy of AI

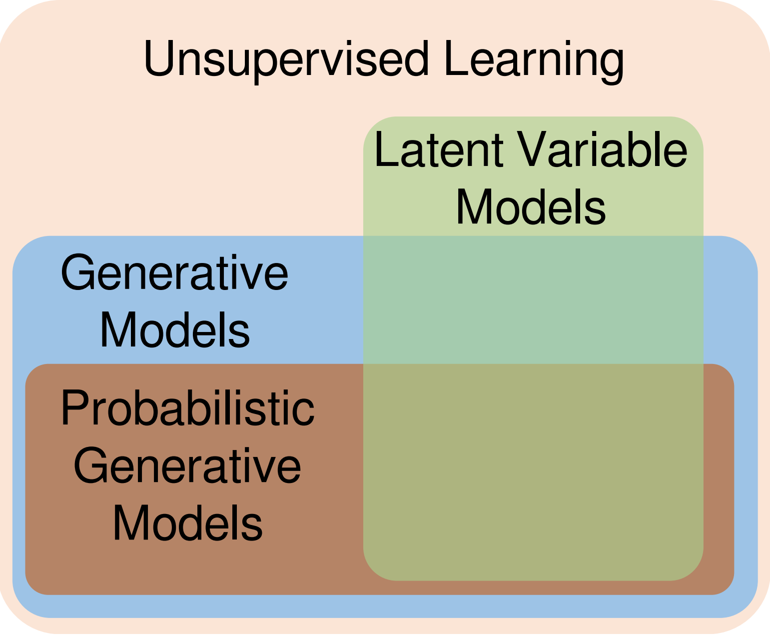
- AI : Everything that focus on simulating intelligent behavior
- ML : Learns to make decisions by fitting mathematical models to the observed data
  - Supervised Learning : *known inputs and output labels*
  - Unsupervised Learning : *output labels unknown*
  - Reinforcement Learning





Learn mapping (policy) from states to actions

Taxonomy of Unsupervised Learning



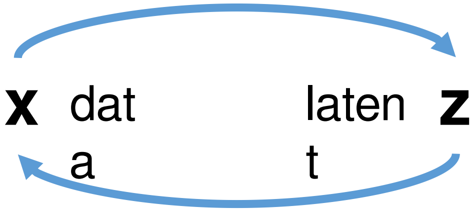
- **Unsupervised Learning**

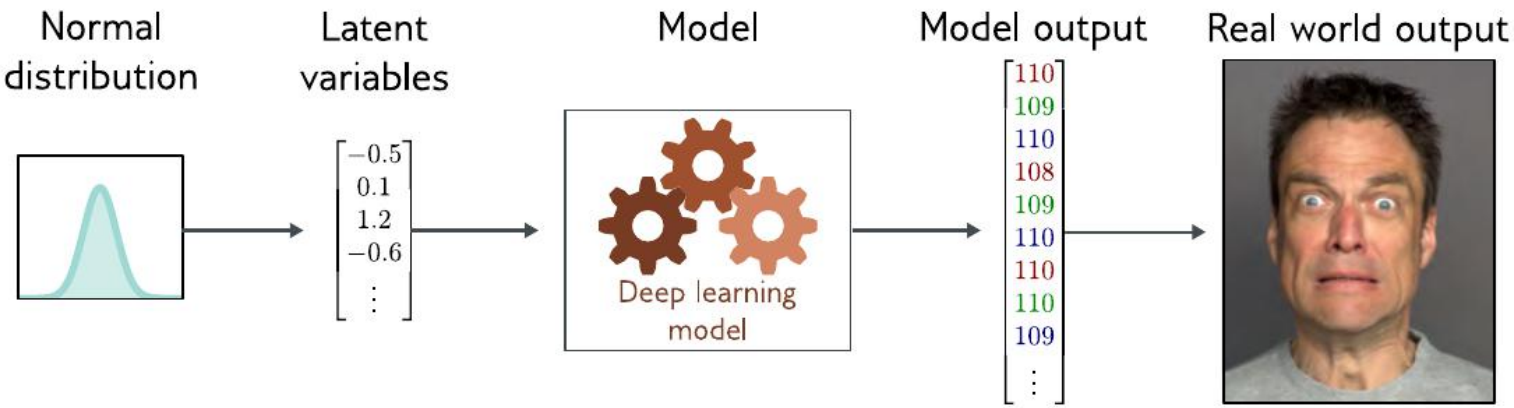
Common approach is to find a mapping between data **x** and unseen *latent* variable **z**

- compressed version of data **x**
- low dimensional than data
- captures essential qualities of **x**

e.g. K-means algorithm,

map **x** to cluster $z \in \{1, 2, \ldots, K\}$





Define a distribution *Pr*(**z**) over latent variable
New samples can be generates
1. Draw from the *Pr*(**z**)
2. Map it to the data space **x**

(**Generative Models**)

- Generative Adversarial Network (GAN) : learn to generate data examples **x** from latent variables **z**
- Normalizing flows, Variational Autoencoders, and Diffusion models (**Probabilistic Generative Models**) : In addition to generating new samples they also assign a probability *Pr*(**x**|**ϵ**) to each data point **x**

Properties of a good generative model

Desired Properties (Non-exhaustive)

- *Efficient sampling* : Sample generation should be computationally inexpensive
- *High-quality samples* : Indistinguishable from the real data with which the model was trained.
- *Coverage* : Generated samples should represent the entire training distribution
- *Well-behaved latent space* : All latent variables should corresponds to a plausible data samples. Smooth changes in $\mathbf{z}$ correspond to smooth changes in $\mathbf{x}$.
- *Disentangled latent space* : Varying each dimension of z should correspond to changing an interpretable property of the data.
- *Efficient likelihood computation* : If the model is probabilistic, we would like to be able to calculate the probability of new examples efficiently and accurately.

Generative Models Covered in This Course

- Generative Adversarial Network (GAN)
- Normalizing Flows
- Variational Autoencoder (VAE)
- Diffusion

Maths refresher

Matrix Calculus

- $y = f[\mathbf{x}]$ where $y \in R$ and $\mathbf{x} \in R^D$
  - derivative $\partial y/\partial \mathbf{x}$ is a D-dimensional vector, where the i^{th} element is computed as $\partial y/\partial x_i$

- $\mathbf{y} = \mathbf{f}[\mathbf{x}]$ where $\mathbf{y} \in R^{Dy}$ and $\mathbf{x} \in R^{Dx}$
  - derivative $\partial \mathbf{y}/\partial \mathbf{x}$ is a Dx × Dy matrix where element (i, j) contains the derivative $\partial y_j/\partial x_i$.
  - also known as a Jacobian and is sometimes written as $\nabla_{\mathbf{x}}\mathbf{y}$ in other documents.

- $\mathbf{y} = \mathbf{f}[\mathbf{X}]$ where $\mathbf{y} \in R^{Dy}$ and $\mathbf{X} \in R^{D1 \times D2}$
  - derivative $\partial \mathbf{y}/\partial \mathbf{X}$ is a 3D tensor containing the derivatives $\partial y_i/\partial x_{jk}$.

matrix and vector derivatives have similar forms

$$y = ax \quad \longrightarrow \quad \frac{\partial y}{\partial x} = a,$$

$$\mathbf{y} = \mathbf{A}\mathbf{x} \quad \longrightarrow \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}^T$$

Norms

The ℓ_p norm of a vector $\mathbf{z}$ is calculated as

$$||\mathbf{z}||_p = \left(\sum_{d=1}^{D} |z_d|^p\right)^{1/p}$$

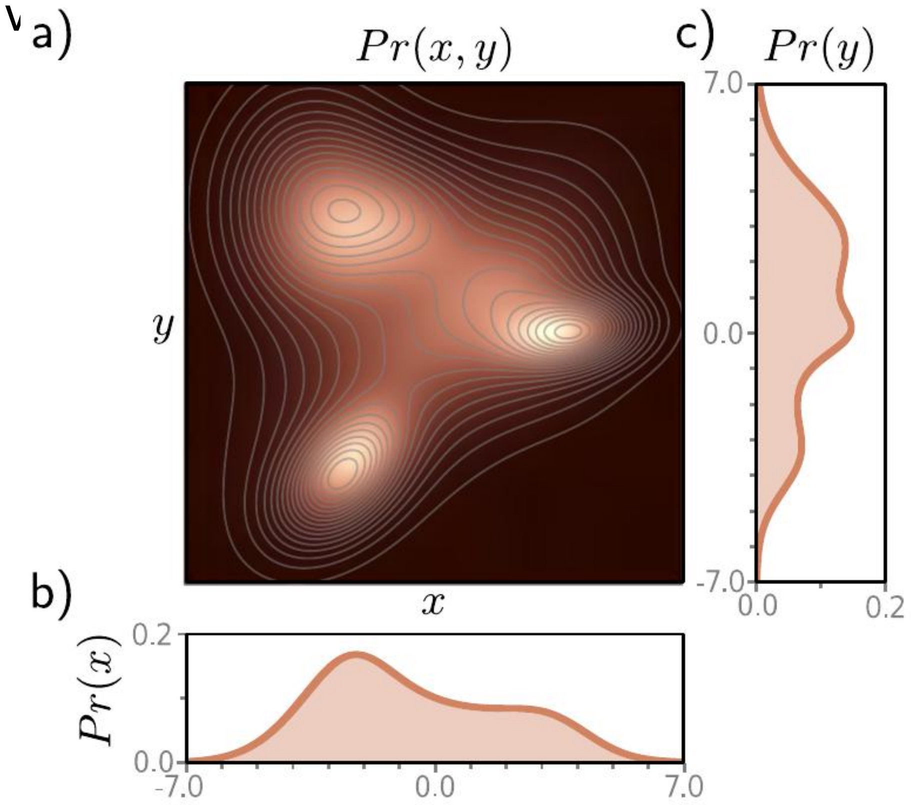
Similarly the ℓ_2 norm of a matrix $\mathbf{Z}$ (known as the Frobenius norm) is calculated as

$$||\mathbf{Z}||_F = \left(\sum_{i=1}^{I}\sum_{j=1}^{J} |z_{ij}|^2\right)^{1/2}$$

Maths refresher

Joint Probability

The joint distribution *Pr*(x, y) tells us about the propensity that x and y take particular combinations of v



$$\iint Pr(x,y) \cdot dxdy = 1$$

Marginalization

The marginal distributions *Pr*(x) and *Pr*(y) can be computed by integrating over the other variable

$$\int Pr(x,y) \cdot dx = Pr(y)$$

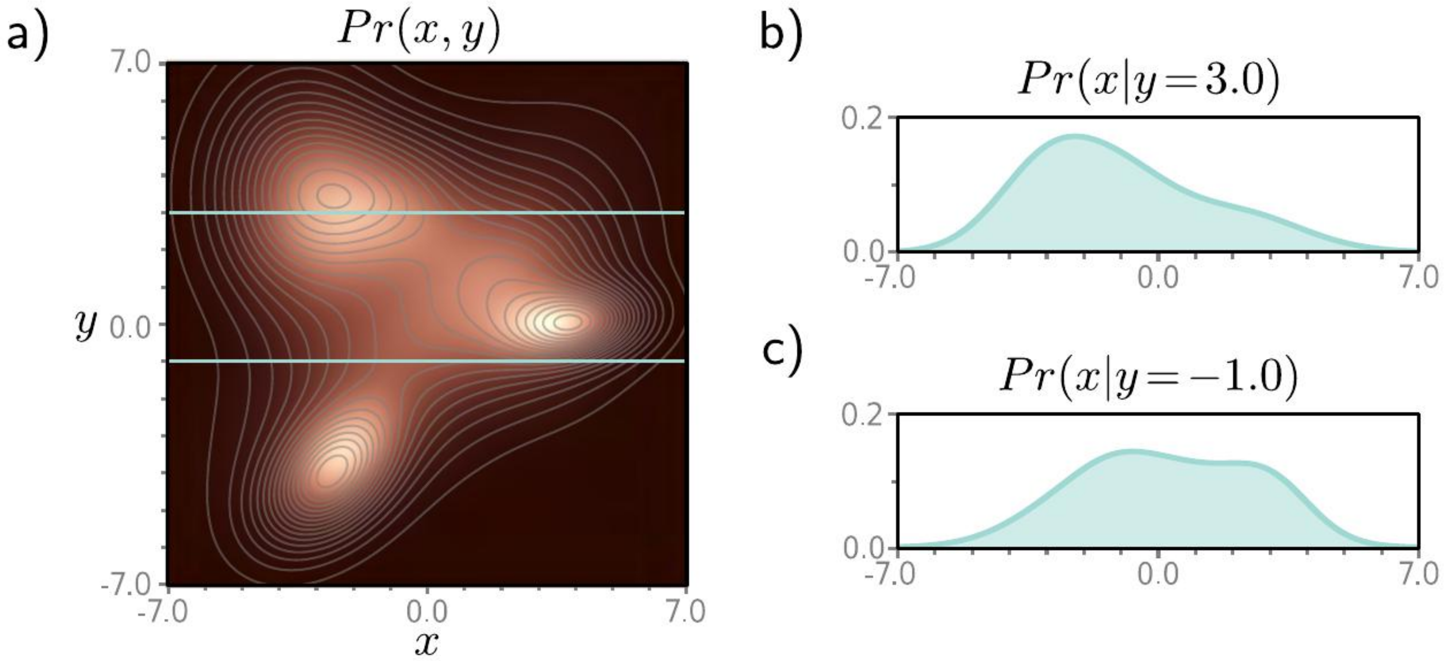
$$\int Pr(x,y) \cdot dy = Pr(x)$$

Interpretation : we are interested in computing the distribution of one variable regardless of the value the other can took

Maths refresher

Conditional Probability

The conditional probability *Pr*(x|y) can be computed by taking a slice through the joint distribution *Pr*(x, y) for a fixed y



$$Pr(x|y) = \frac{Pr(x, y)}{Pr(y)}$$

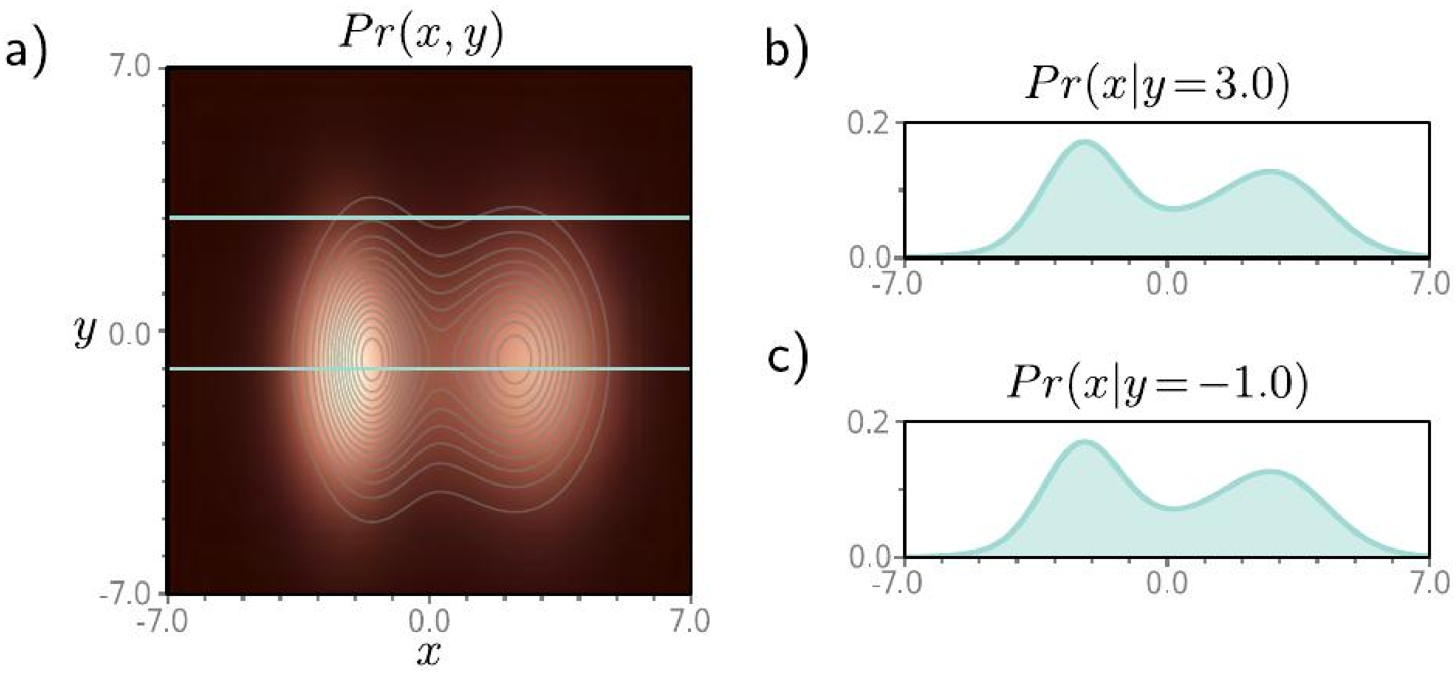
$$Pr(y|x) = \frac{Pr(x, y)}{Pr(x)}$$

$$Pr(x, y) = Pr(x|y)Pr(y) = Pr(y|x)Pr(x)$$

Maths refresher

Independence

- If two random variables x and y are independent,
  - the joint distribution factors into the product of marginal distributions, so *Pr*(x, y) = *Pr*(x) Pr(y).



$$Pr(x, y) = Pr(x|y)Pr(y) = Pr(x)Pr(y)$$

Maths refresher

Univariate Normal Distrubution

$$Pr(x) = \text{Norm}_x[\mu, \sigma^2] = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

mean (μ), variance (σ^2)

Standard Normal Distrubution

Distribution with mean is **zero** and the variance **one**

Standardization

The process of setting the mean of a random variable to zero and the variance to one

$$z = \frac{x-\mu}{\sigma}$$

Multivariate Normal Distrubution

$$\text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}] = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left[-\frac{(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}{2}\right]$$

covariance ($\boldsymbol{\Sigma}$)

$$\mathbf{z} = \boldsymbol{\Sigma}^{-1/2}(\mathbf{x}-\boldsymbol{\mu})$$

(Standardization)

Maths refresher

Sample from a **univariate** distribution

1. First compute the cumulative distribution F[*x*]
2. Draw a sample z* from a uniform distribution over the range [0, 1]
3. Evaluate above against the inverse of the cumulative distribution, the sample *x** is created as: $x^* = \mathrm{F}^{-1}[z^*]$

Sample from **normal** distribution

A sample *x* from a normal distribution with mean *μ* and variance σ^2 can be obtained as

$$x = \mu + \sigma z$$

z is a standard variable with zero mean and unit variance

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\mathbf{z}$$

multivariate case

Ancestral Sampling

The process to generate a sample from the root variable(s) and then sample from the subsequent conditional distributions is known as ancestral sampling.

Consider a joint distribution *Pr*(*x, y, z*) over three variables, *x, y,* and *z*, which can be factored as :

Pr(x, y, z) = *Pr(x) Pr(y|x) Pr(z|y)*

To get a sample from this joint distribution,

- first draw a sample *x** from *Pr(x)*
- then draw a sample *y** from *Pr(y|x*)*
- finally, draw a sample *z** from *Pr(z|y*)*

Maths refresher

Distance between two probability distributions

KL Divergence

$$D_{KL}[p(x)||q(x)] = \int p(x) \log \left[\frac{p(x)}{q(x)}\right] dx$$

KL Divergence is not symmetric

$$D_{KL}[p(x)||q(x)] \neq D_{KL}[q(x)||p(x)]$$

Jensen-Shannon Divergence

$$D_{JS}\Big[p(x)||q(x)\Big] = \frac{1}{2} D_{KL}\left[p(x)\Big|\Big|\frac{p(x)+q(x)}{2}\right] + \frac{1}{2} D_{KL}\left[q(x)\Big|\Big|\frac{p(x)+q(x)}{2}\right]$$

Generative Adversarial Network (GAN)

Generative Adversarial Network (GAN)

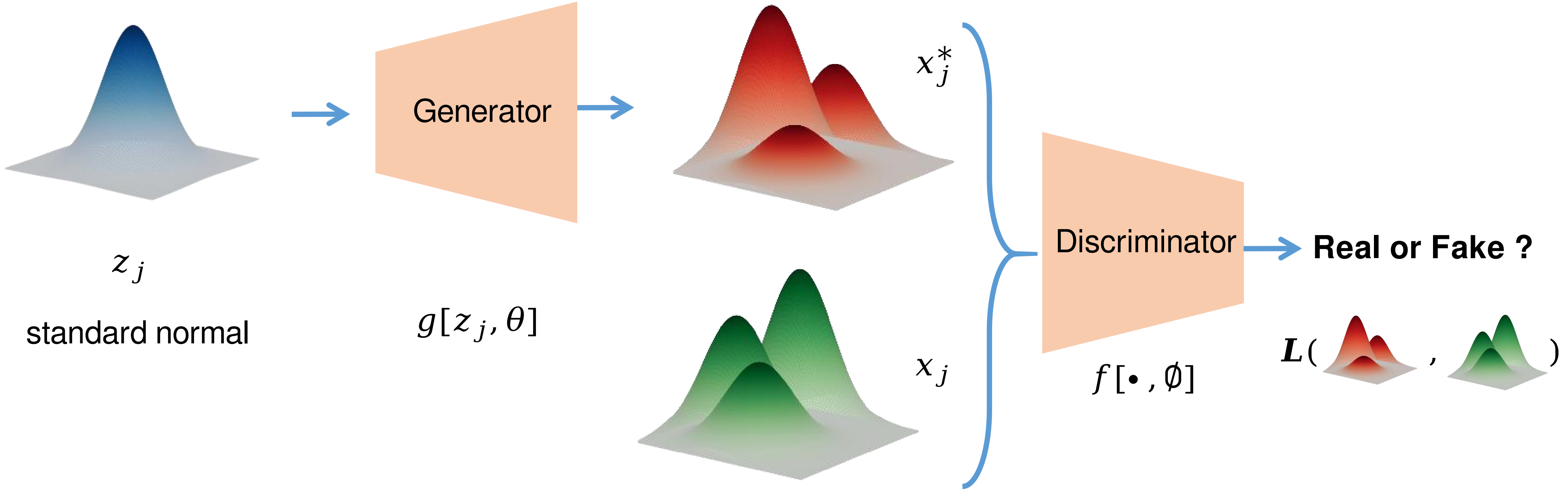
Unsupervised model with an objective to **generate new samples** that are **indistinguishable** from training examples

- GANs focus is to generate new samples only
- Doesm’t build a probability distribution over the training data
    - cannot evaluate the probability that a new data point belongs to the same distribution

The GAN framework:

- *main generator network* : generates new samples by mapping **random noise** to the output data space
- *discriminator network* : distinguish between the generated samples and the real examples

Generative Adversarial Network (GAN)

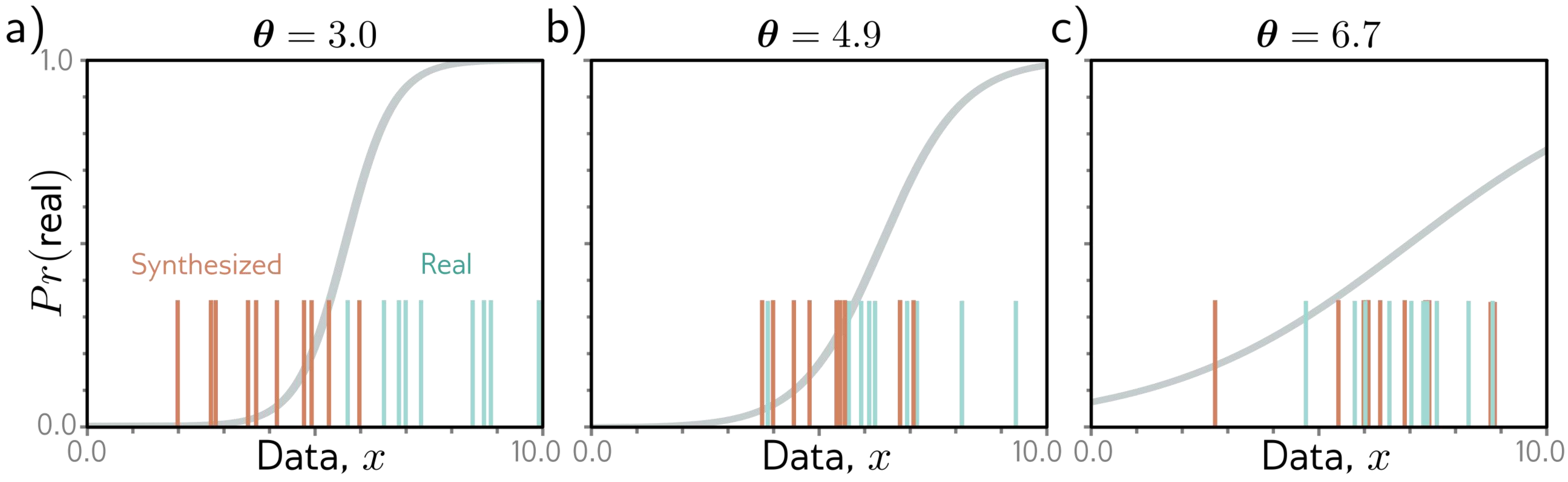


discrimination as a signal

how to define & measure

Training Goal : Find θ such that samples $\{x_j^*\}$ looks similar to $\{x_j\}$

Generative Adversarial Network (GAN) - 1D Toy Example



Training data $\{x_j\}$

A simple generator: $x_j^* = g[z_j, \theta] = z_j + \theta$ Increase θ move samples rightwards

z_j: **standard normal distribution**

Discriminator : slope of the $sigmoid\ function$ sigmoid become flatter

Generative Adversarial Network (GAN) - Loss functions

Binary cross-entropy loss:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_i -(1-y_i) \log\left[1 - \operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\right] - y_i \log\left[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\right] \right]$$

- logistic signmoid function

$y = 1$ for real example, $y = 0$ for fake example

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_j -\log\left[1 - \operatorname{sig}[\mathrm{f}[\mathbf{x}_j^*, \boldsymbol{\phi}]]\right] - \sum_i \log\left[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\right] \right]$$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\min_{\boldsymbol{\phi}} \left[\sum_j -\log\left[1 - \operatorname{sig}[\mathrm{f}[\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}], \boldsymbol{\phi}]]\right] - \sum_i \log\left[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\right] \right] \right]$$

minmax game

Optimal solution : generated examples will be indistinguishable from real ones, discriminator will be at chance (0.5)

Generative Adversarial Network (GAN) - Loss functions

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\min_{\boldsymbol{\phi}} \left[\sum_j -\log\left[1 - \operatorname{sig}[\operatorname{f}[\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}], \boldsymbol{\phi}]]\right] - \sum_i \log\left[\operatorname{sig}[\operatorname{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\right] \right] \right]$$

$$L[\boldsymbol{\phi}] = \sum_j -\log\left[1 - \operatorname{sig}[\operatorname{f}[\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}], \boldsymbol{\phi}]]\right] - \sum_i \log\left[\operatorname{sig}[\operatorname{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\right]$$

Train discriminator

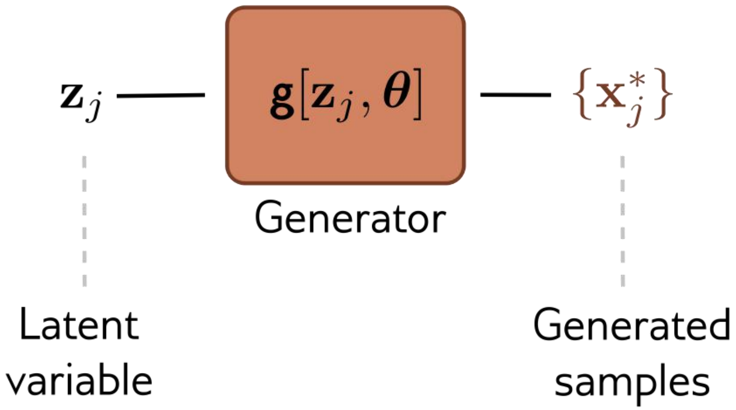
$$L[\boldsymbol{\theta}] = \sum_j \log\left[1 - \operatorname{sig}[\operatorname{f}[\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}], \boldsymbol{\phi}]]\right],$$

Train generator

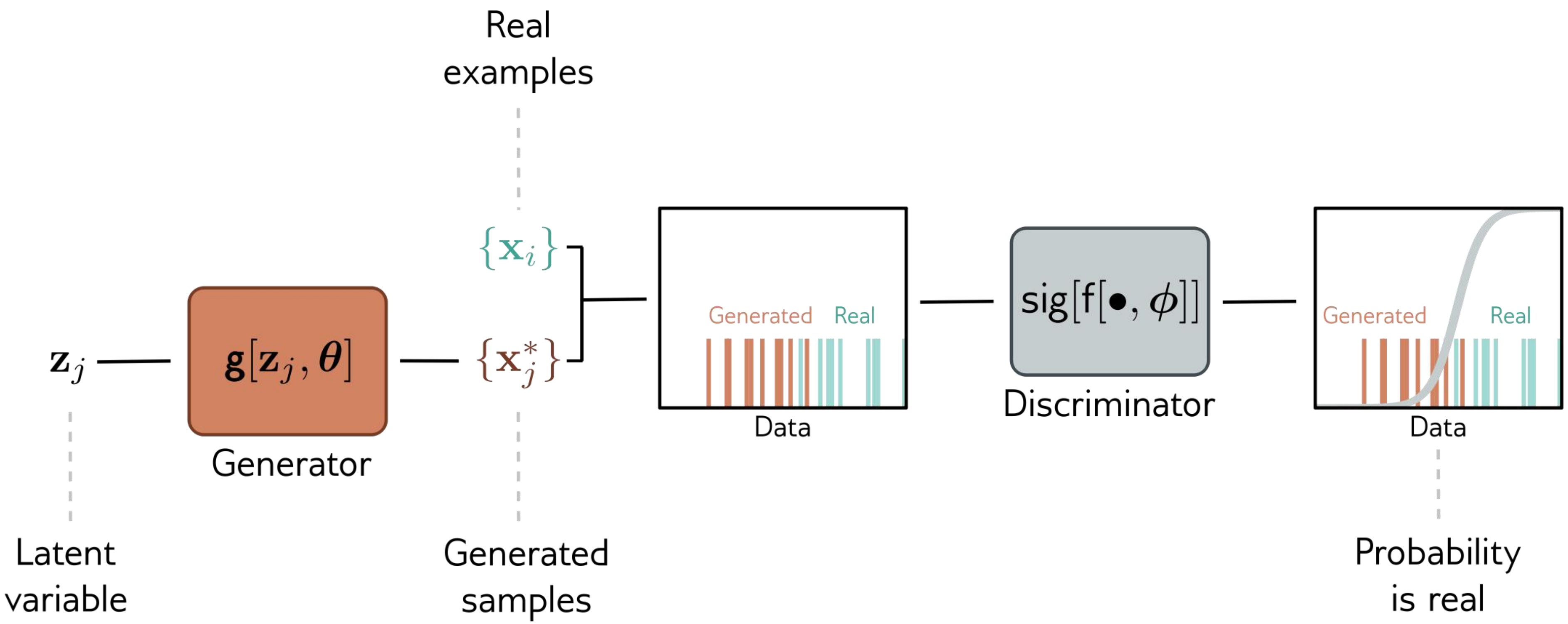
Overall training strategy

- draw a batch of latent variables z_j from the base distribution
- generator create samples $x_j = g[z_j , \theta]$
- choose a batch of real training examples .
- Given the two batches, we can now perform one or more gradient descent steps on each loss function

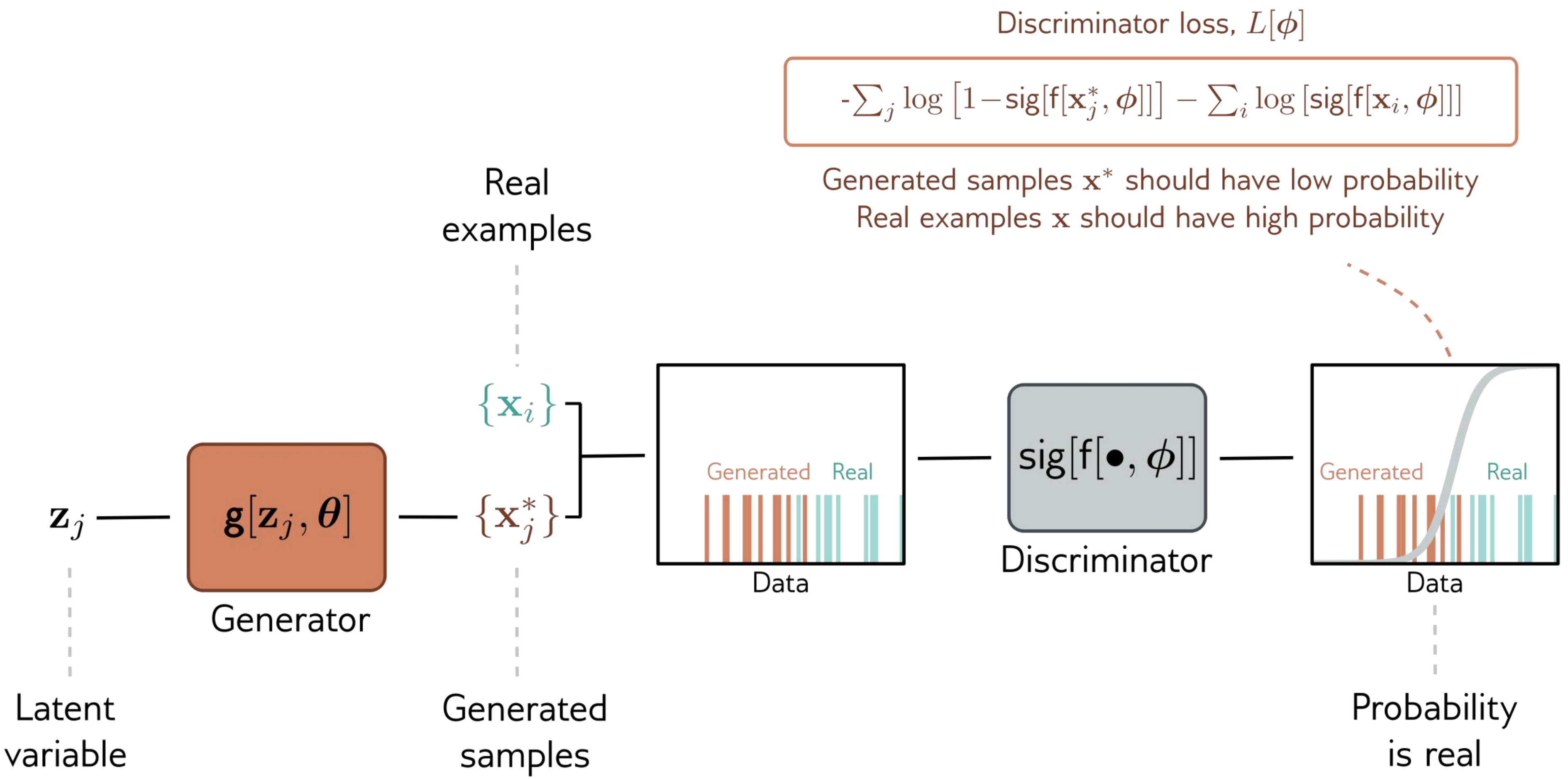
Generative Adversarial Network (GAN)



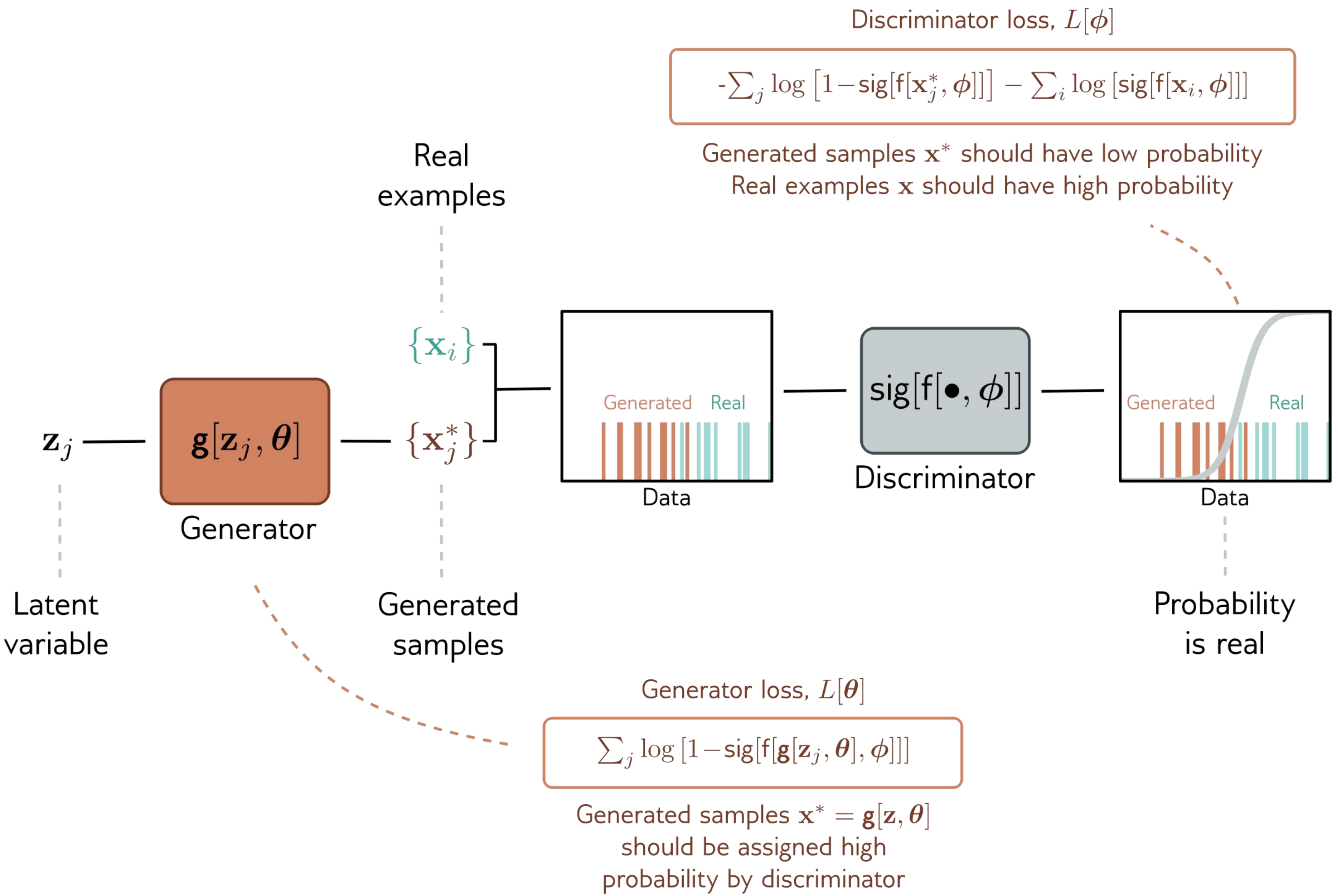
Generative Adversarial Network (GAN)



Generative Adversarial Network (GAN)

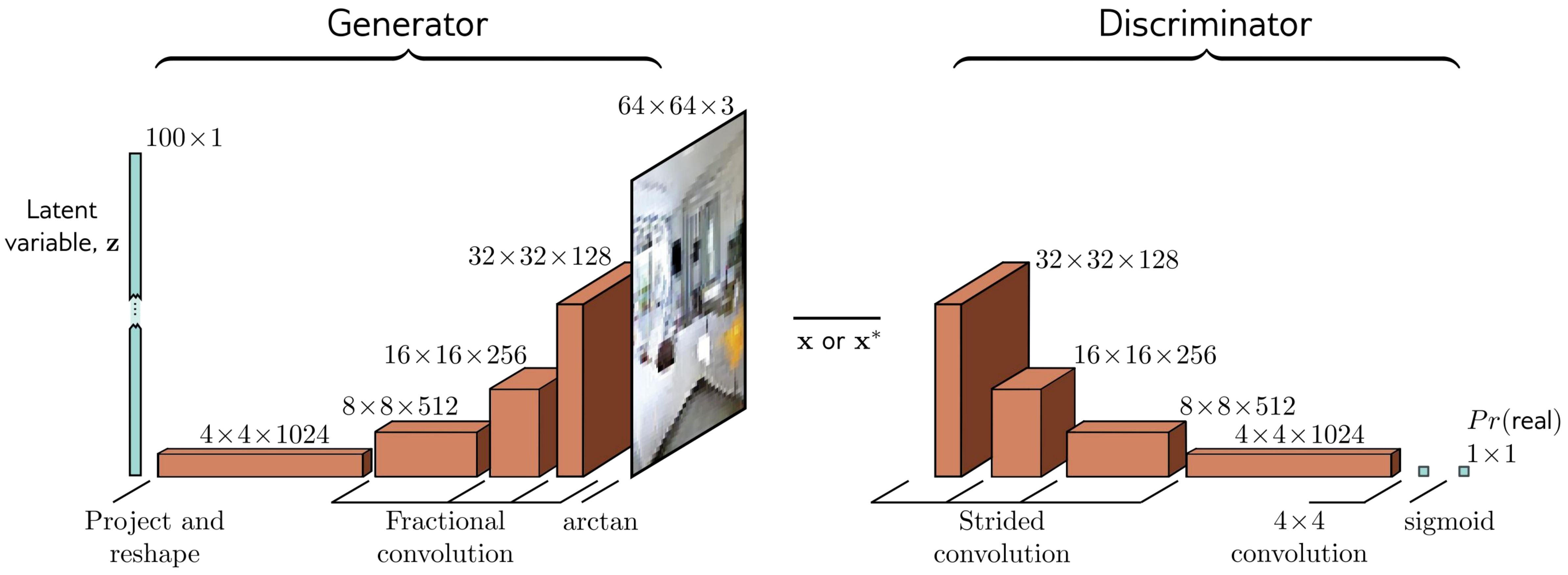


Generative Adversarial Network (GAN)



Generative Adversarial Network (GAN)

Deep convolutional GAN (DCGAN)



Issues and Instability in GAN training

- Mode Dropping
- Mode Collapse
- Vanishing Gradient

Issues and Instability in GAN training

- **Mode Dropping**
- Mode Collapse
- Vanishing Gradient

Generator is able to generate plausible samples, but only represent a subset of the data

For example for faces, it might never generate faces with beards

Issues and Instability in GAN training

- Mode Dropping
- **Mode Collapse**
- Vanishing Gradient

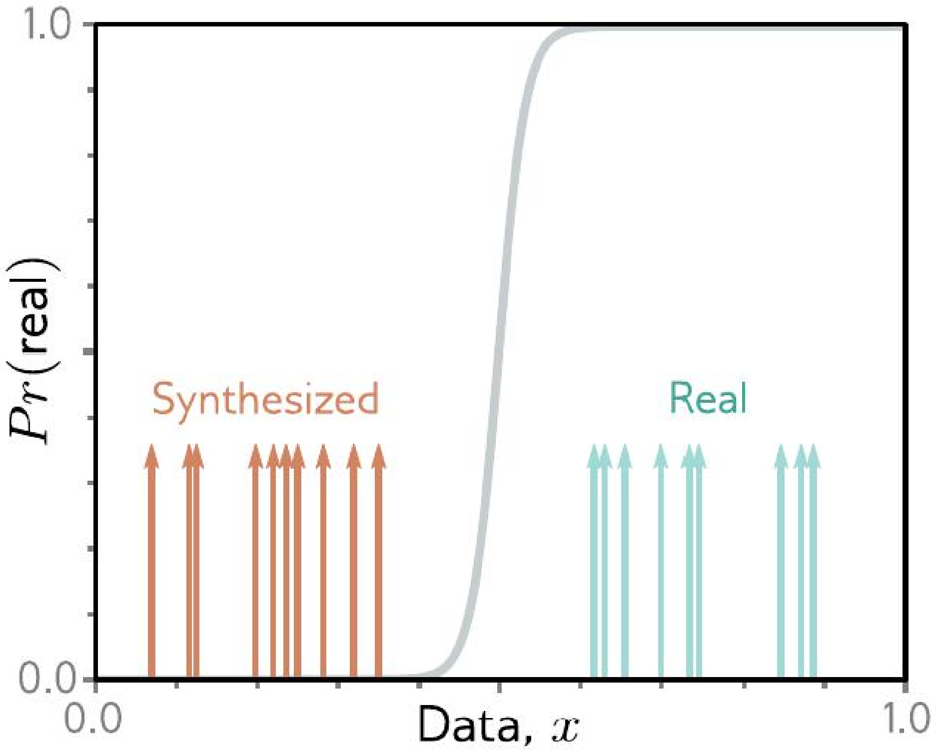
Images are generated from a GAN trained on the LSUN scene understanding dataset using an MLP generator with a similar number of
parameters and layers to the DCGAN

Generator entirely or mostly ignores the latent variables ***z*** and collapses all samples to one or a few points

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." International conference on machine learning. PMLR, 2017.

Issues and Instability in GAN training

- Mode Dropping
- Mode Collapse
- **Vanishing Gradient**



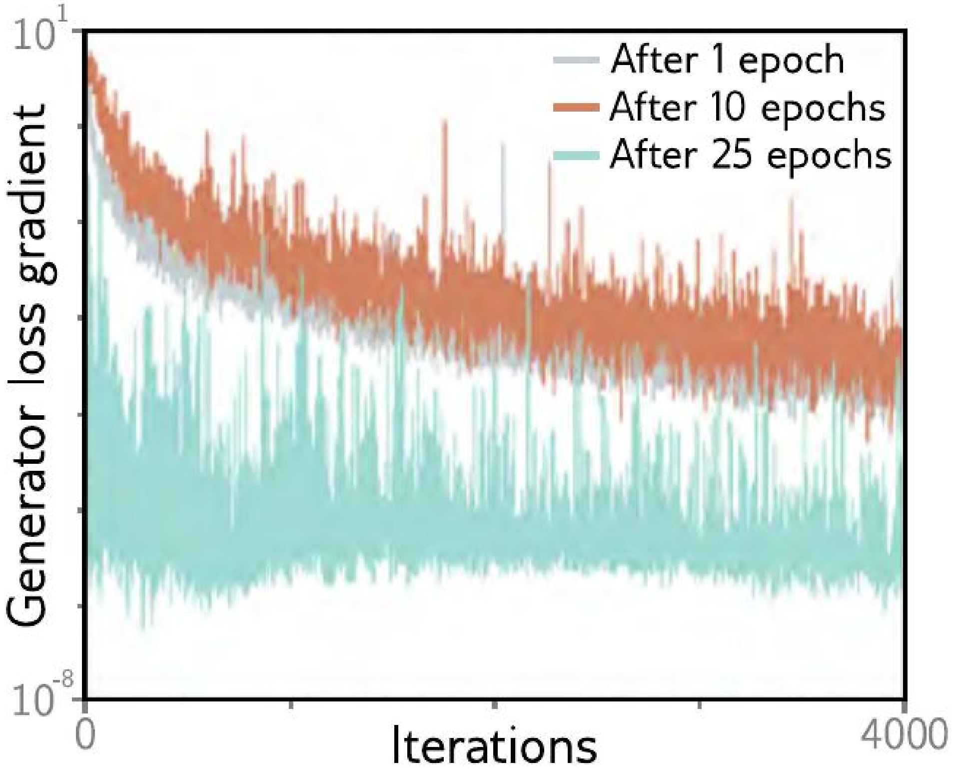
gradient to update the parameter of the generator may be tiny, when it is easy for discriminator to distinguish between real and generated examples,

the discriminator may have a very shallow slope **at the positions of the samples**;

Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. International Conference on Learning Representations.

Issues and Instability in GAN training

- Mode Dropping
- Mode Collapse
- **Vanishing Gradient**



- The generator is frozen after 1, 10, and 25 epochs, and the discriminator is trained further
- The gradient of the generator decreases rapidly
- If the discriminator becomes too accurate, the gradients for the generator vanish.

Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. International Conference on Learning Representations.

Issues and Instability in GAN training

- does not depend on the generator
- happy to generate a subset of possible examples accurately.
- potential reason for mode dropping

- **Potential reason ?**

Issue with the original loss formulation:

the gradient of the distance becomes zero when the generated samples are too easy to distinguish from the real examples

$$
\begin{aligned}
L[\phi] &= -\frac{1}{J}\sum_{j=1}^{J}\left(\log\left[1-\mathrm{sig}[\mathrm{f}[\mathbf{x}_j^*,\phi]]\right]\right) - \frac{1}{I}\sum_{i=1}^{I}\left(\log\left[\mathrm{sig}[\mathrm{f}[\mathbf{x}_i,\phi]]\right]\right) \\
&\approx -\mathbb{E}_{\mathbf{x}^*}\left[\log\left[1-\mathrm{sig}[\mathrm{f}[\mathbf{x}^*,\phi]]\right]\right] - \mathbb{E}_{\mathbf{x}}\left[\log\left[\mathrm{sig}[\mathrm{f}[\mathbf{x},\phi]]\right]\right] \\
&= -\int Pr(\mathbf{x}^*)\log\left[1-\mathrm{sig}[\mathrm{f}[\mathbf{x}^*,\phi]]\right]d\mathbf{x}^* - \int Pr(\mathbf{x})\log\left[\mathrm{sig}[\mathrm{f}[\mathbf{x},\phi]]\right]d\mathbf{x} \\
&= -\int Pr(\mathbf{x}^*)\log\left[1-\frac{Pr(\mathbf{x})}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x}^* - \int Pr(\mathbf{x})\log\left[\frac{Pr(\mathbf{x})}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x} \\
&= -\int Pr(\mathbf{x}^*)\log\left[\frac{Pr(\mathbf{x}^*)}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x}^* - \int Pr(\mathbf{x})\log\left[\frac{Pr(\mathbf{x})}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x}.
\end{aligned}
$$

KL Divergence

it penalizes regions with samples *x** but no real examples *x* *(Quality)*

it penalizes regions with real examples but no samples *(Coverage)*

This is the Jensen-Shannon divergence between the synthesized distribution *Pr(x*)* and the true distribution *Pr(x)*

Wasserstein Formulation of GAN

- Original loss function

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \left[\sum_j -\log\left[1 - \operatorname{sig}[\mathrm{f}[\mathbf{x}_j^*, \phi]]\right] - \sum_i \log\left[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \phi]]\right] \right]$$

- Wasserstein GAN loss function

$$L[\phi] = \sum_j \mathrm{f}[\mathbf{x}_j^*, \phi] - \sum_i \mathrm{f}[\mathbf{x}_i, \phi]$$

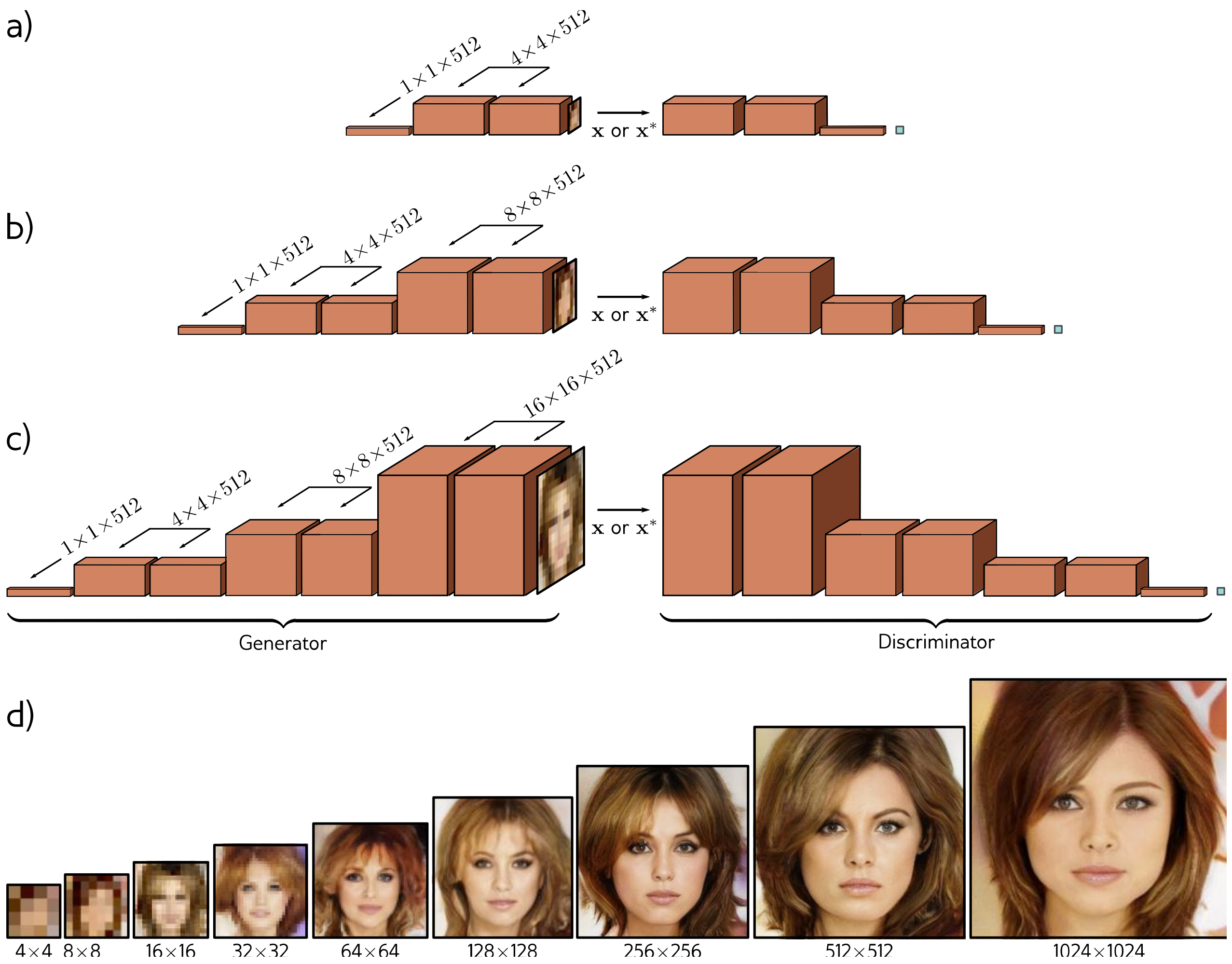
Subject to $\left| \frac{\partial \mathrm{f}[\mathbf{x}, \phi]}{\partial \mathbf{x}} \right| < 1$ Constrain the discriminator to have an absolute gradient norm at every position *x*

How ?

- Clip the discriminator weights to a small range (*e.g., ±0.01*)
- Use gradient penalty i.e., add a regularization term that increases as the gradient norm deviates from unity. (WGAN-GP)

Issues and Instability in GAN training

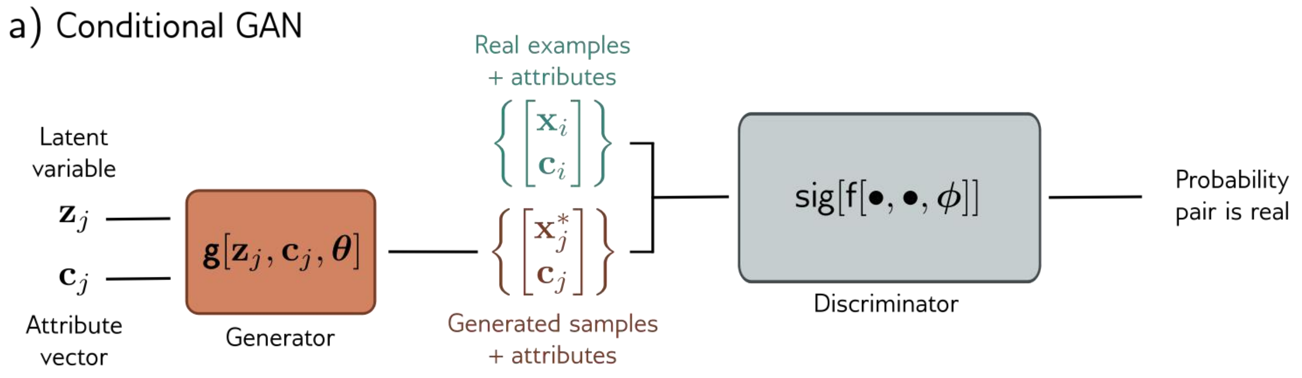
- Wassertein formulation makes the training stable
- Other tricks that improve quality are *progressive growing*, *minibatch discrimination*, and *truncation* etc..



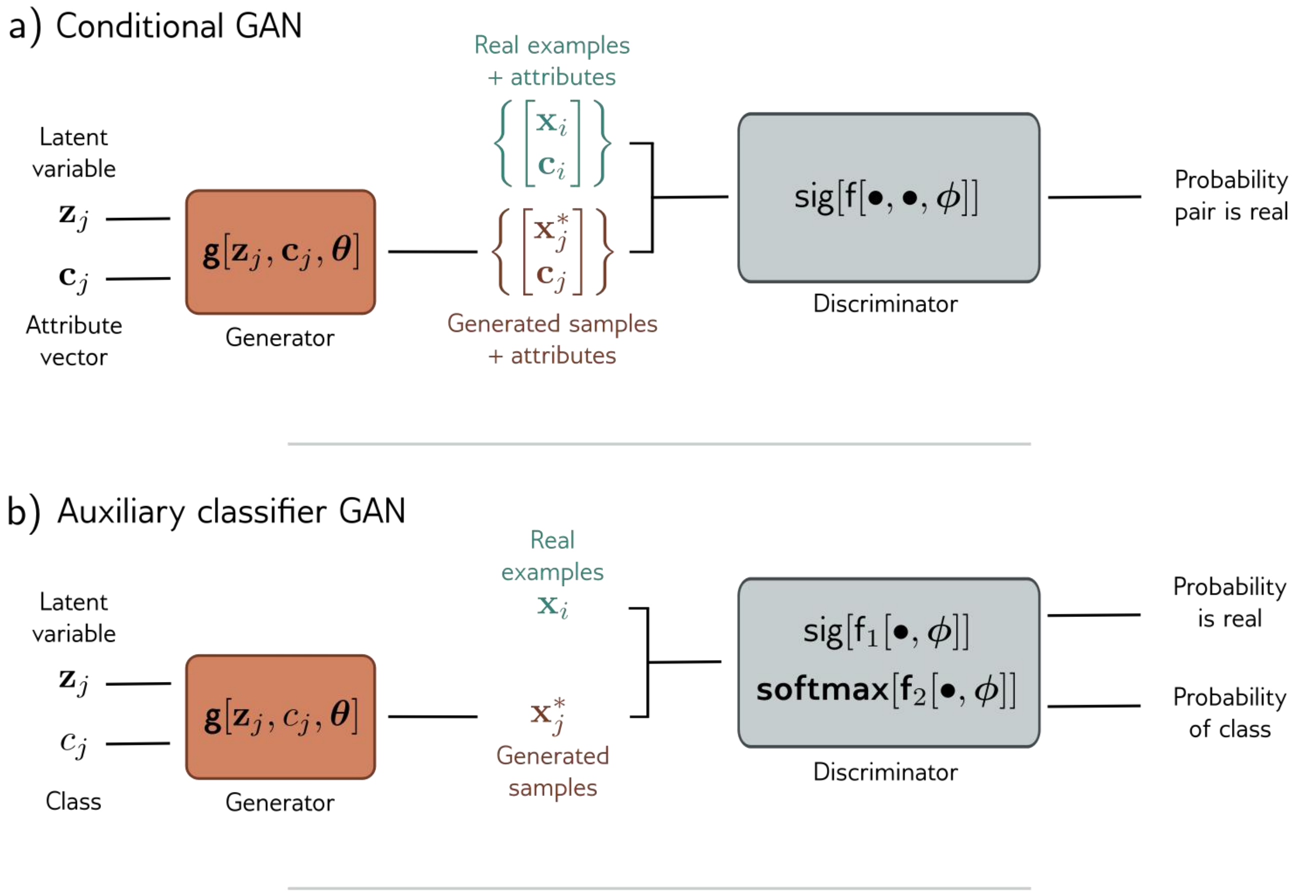
Issues and Instability in GAN training

- Combination of all tricks allow GAN's to generate varied and realistic images

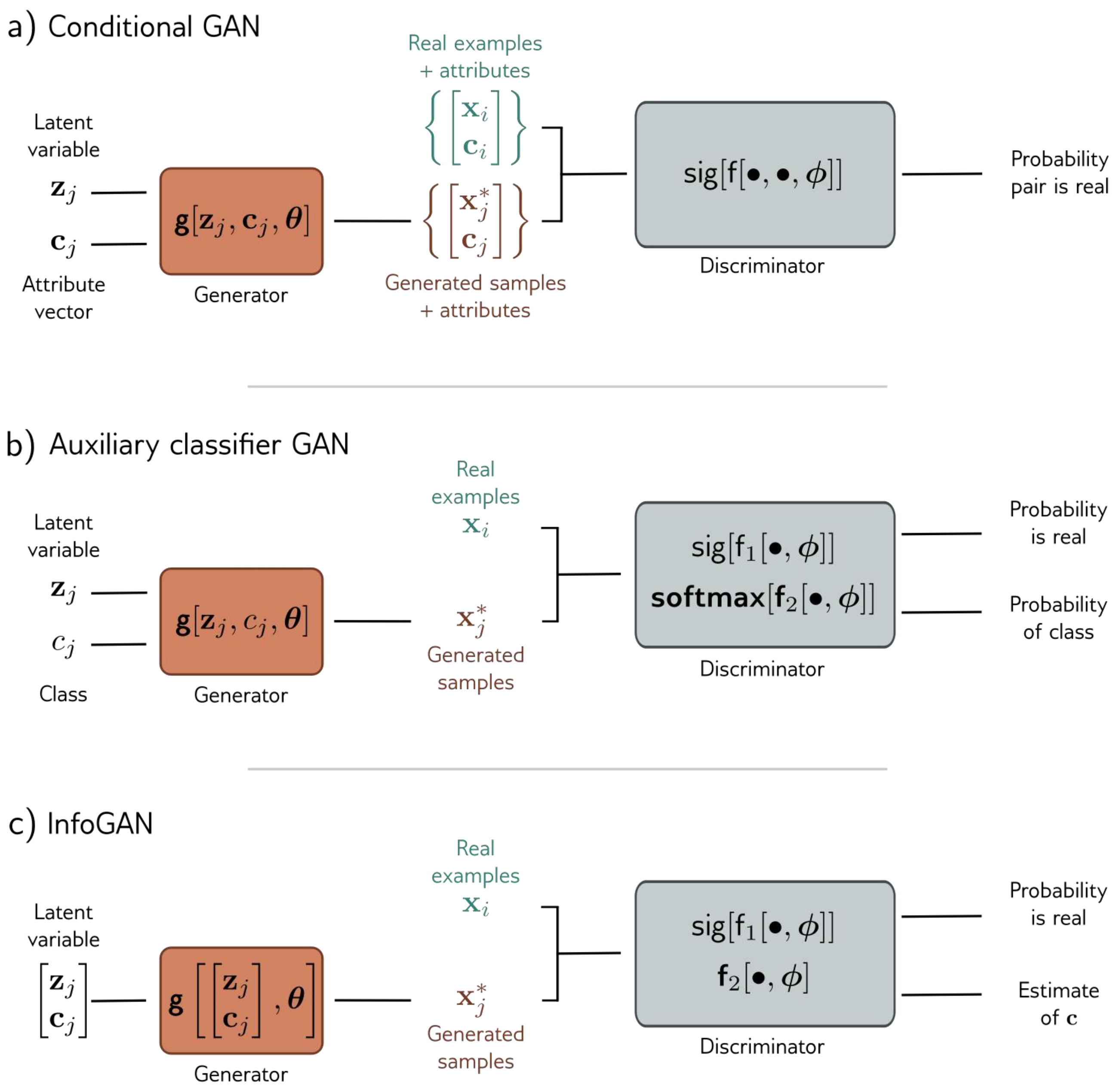
Conditional GAN



Conditional GAN



Conditional GAN



ACGAN Results

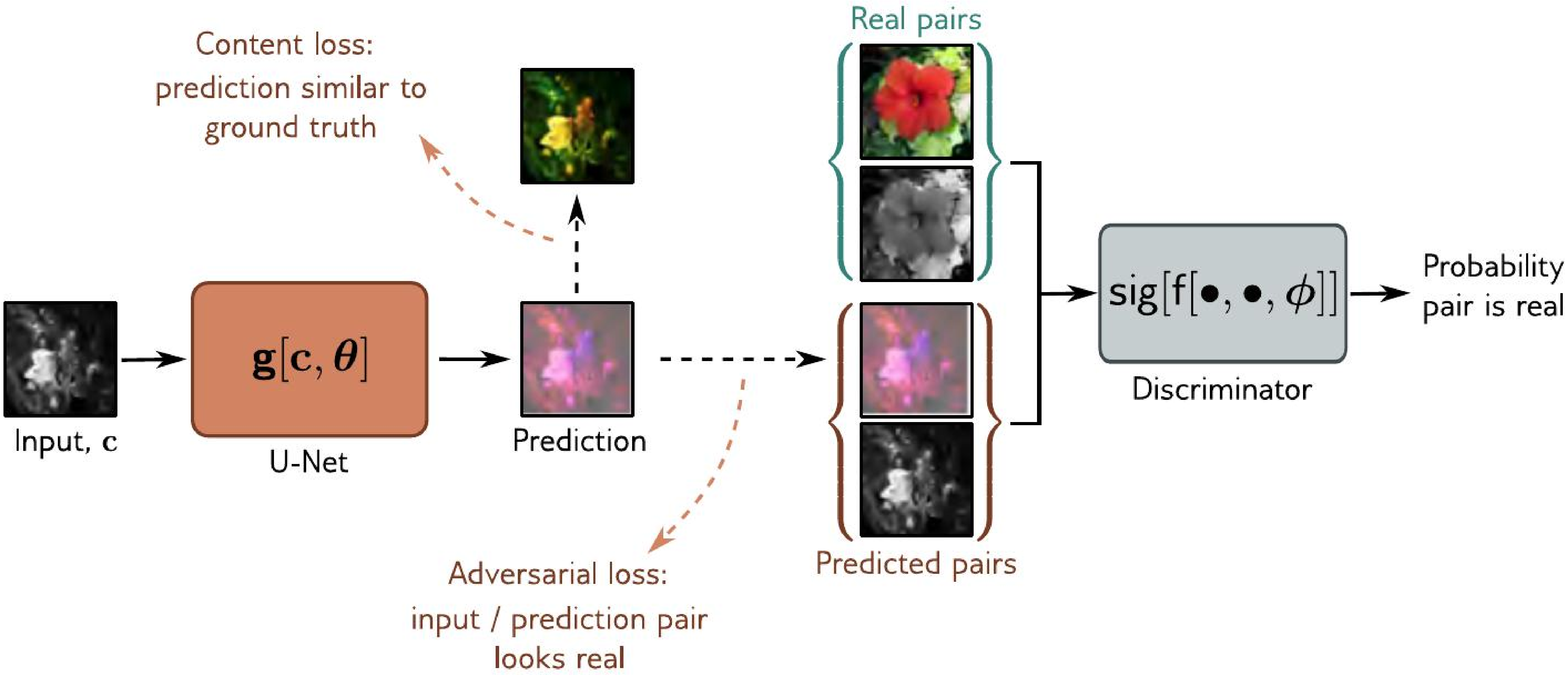
Condition : Class label

Image Translation

GANs used in image translation tasks like

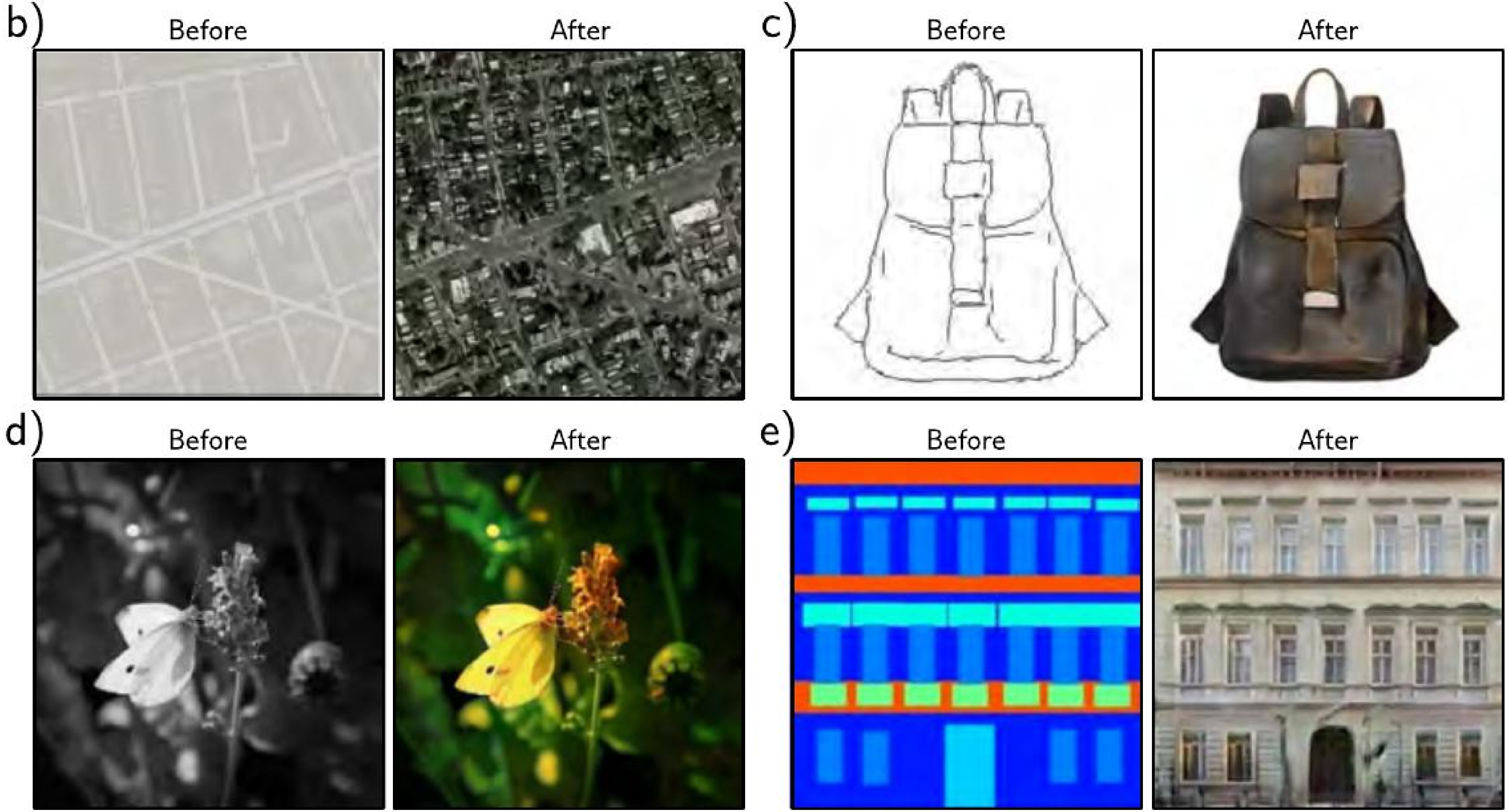
- Pix2Pix
- CycleGAN
- StyleGAN

Image Translation - Pix2Pix



Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. Image-to-image translation with condi tional adversarial networks, CVPR 2017

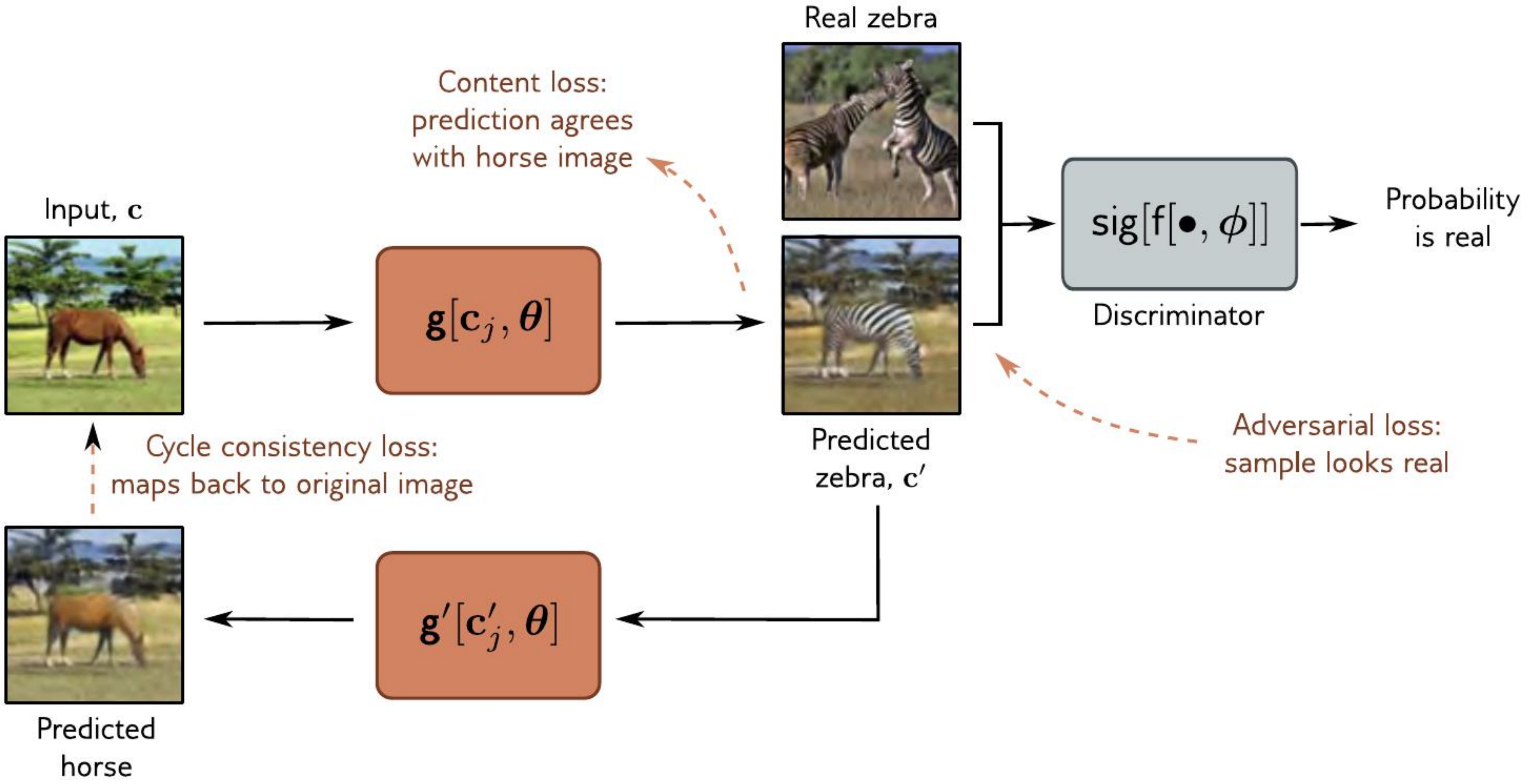
Image Translation - Pix2Pix (Results)



Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. Image-to-image translation with condi tional adversarial networks, CVPR 2017

Image Translation - CycleGAN

- What if we do not have the labeled before/after images for adversarial loss.
- The CycleGAN addresses the situation where two sets of data with distinct styles are available but no matching pairs



Zhu, J.-Y et.al., Unpaired image-to-image translation using cycle-consistent adversarial networks, ICCV 2017

Image Translation - CycleGAN (Results)

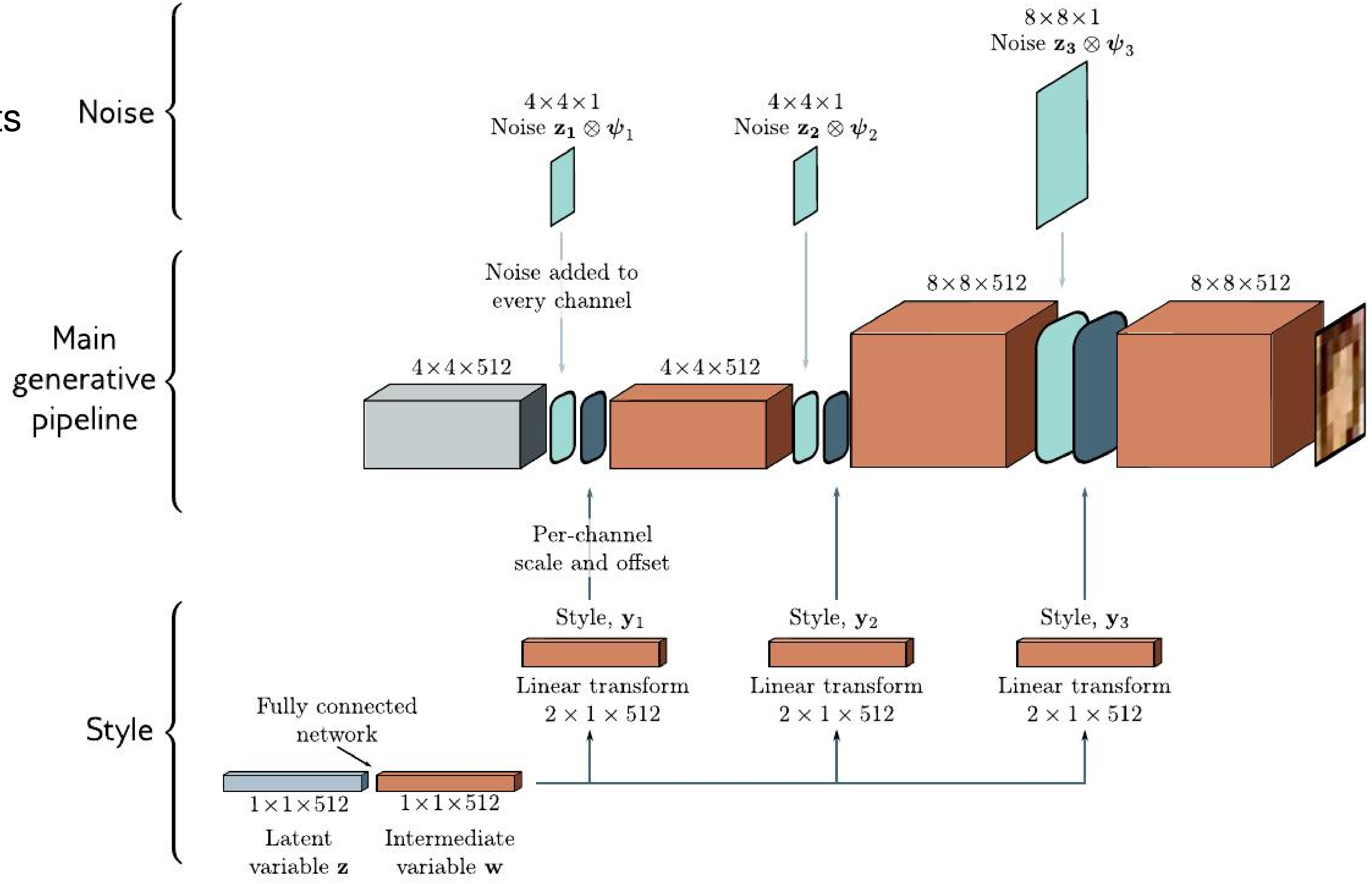
Zhu, J.-Y et.al., Unpaired image-to-image translation using cycle-consistent adversarial networks, ICCV 2017

Image Translation - StyleGAN

- StyleGAN controls the output image at different scales and separates style from noise

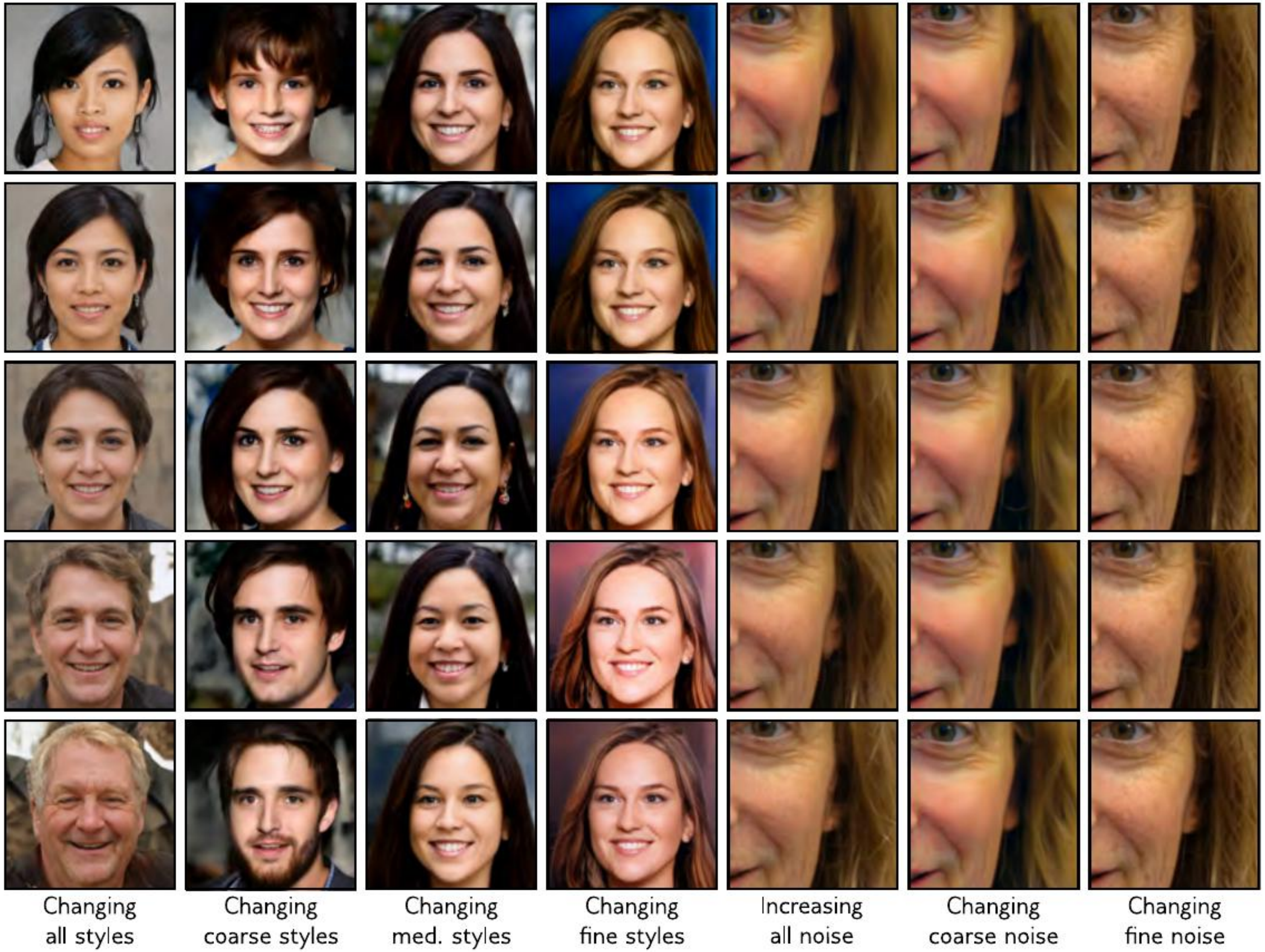
- latent variable injected to the inputs of the generator at various points

- to modify the current representation at these points in different ways.



Karras, T., Laine, S., & Aila, T. (2019). A style based generator architecture for generative adversarial networks, CVPR 2019

Image Translation - StyleGAN (Results)



Karras, T., Laine, S., & Aila, T. (2019). A style based generator architecture for generative adversarial networks, CVPR 2019