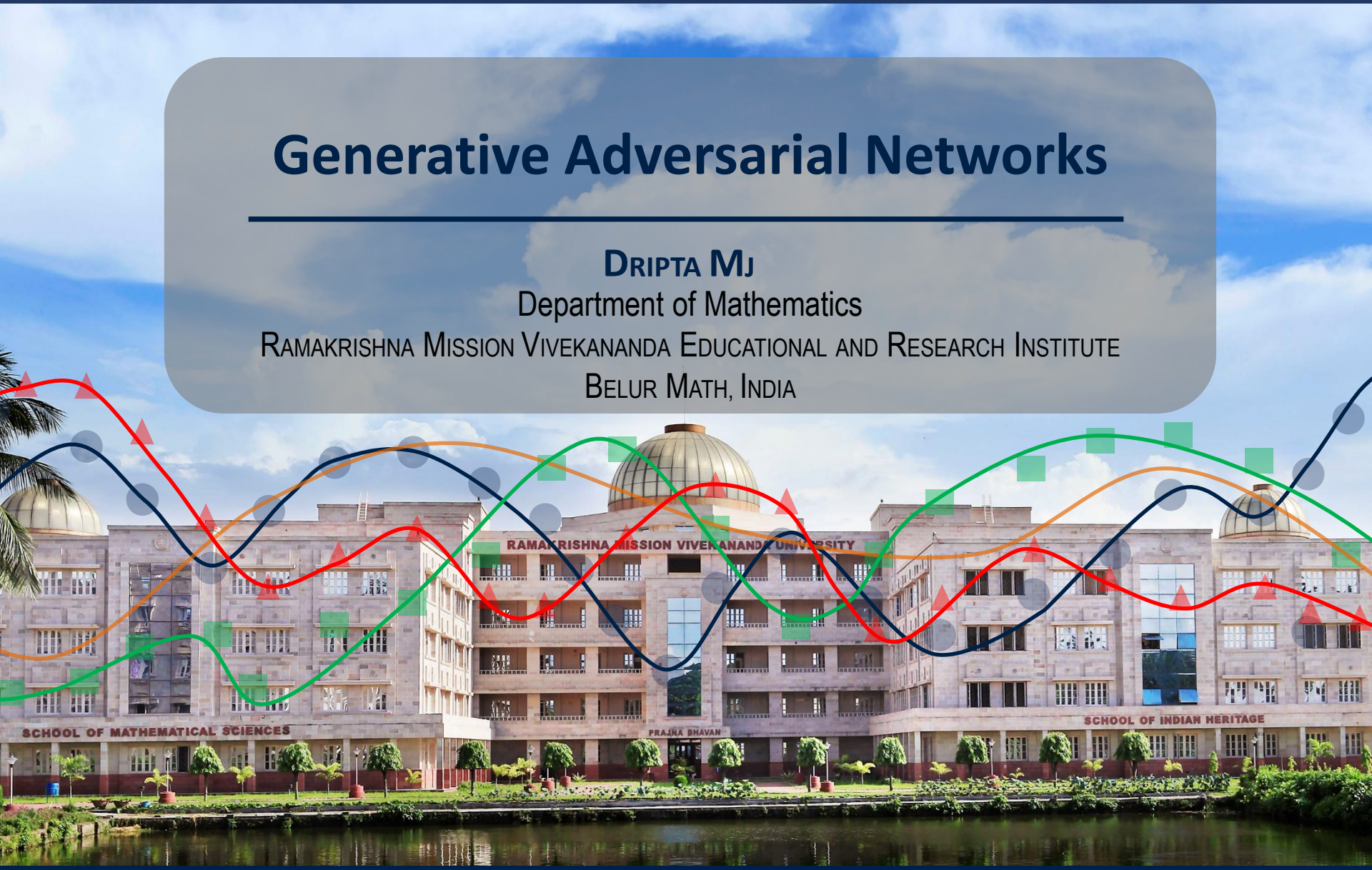
Generative Adversarial Networks

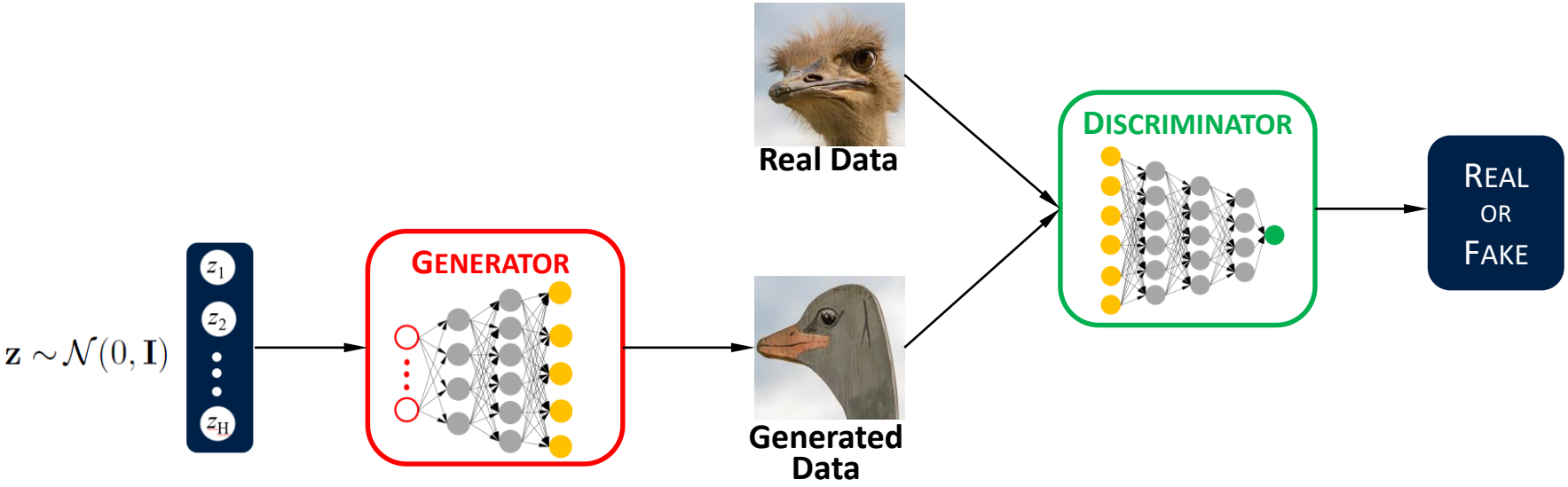
Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India



GAN

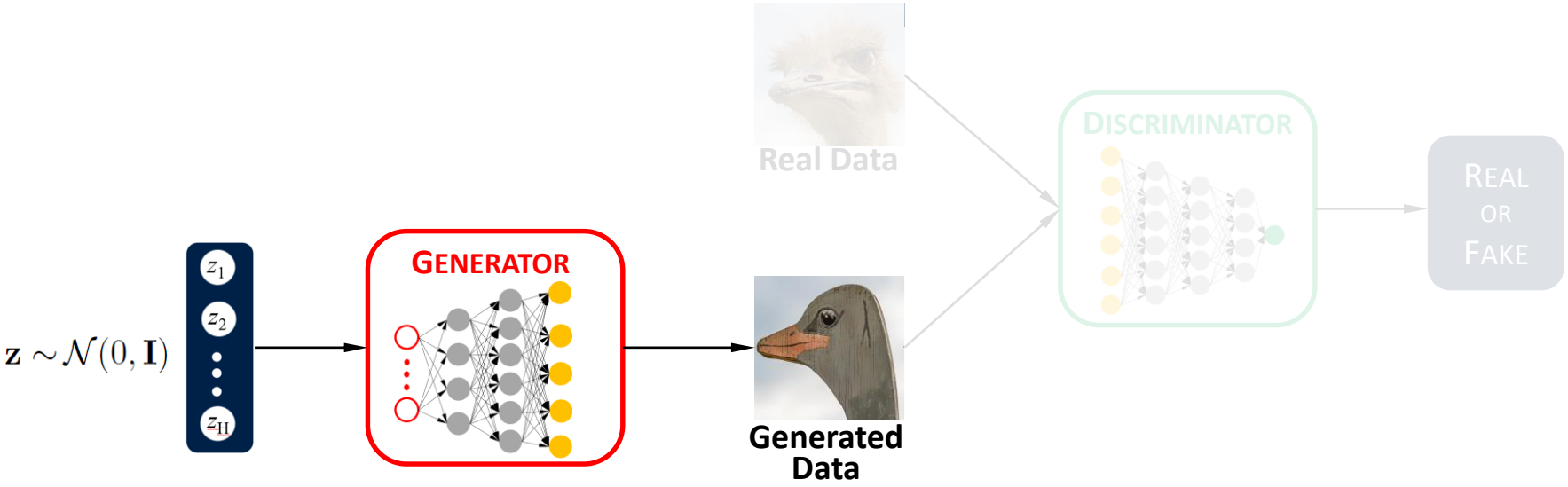
- The training data comes from some underlying complex high-dimensional distribution.
- New data can be generated by sampling from this distribution.
- GANs overcome this problem by sampling from a simple distribution, and then learn a complex distribution to generate training data.
- The complex transformation is a neural network.
- Training is done using a two player game which comprise a generator and a discriminator.
- The generator produces images that appear to be real.
- The discriminator tries to detect if an image is real or fake.

GAN components



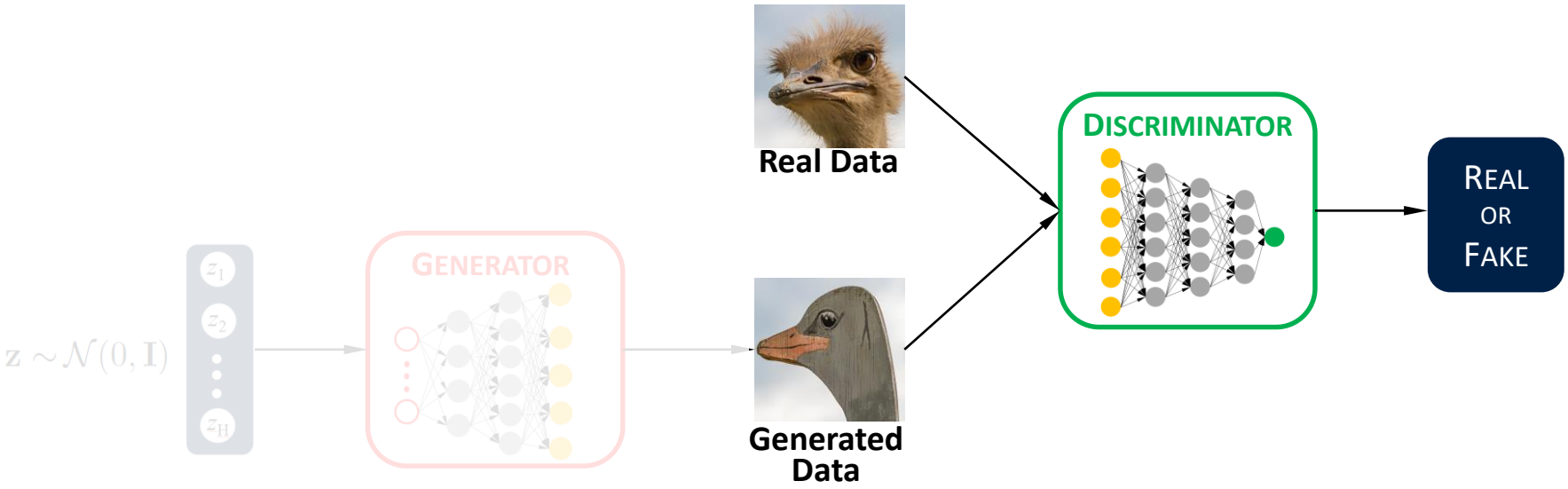
- Training is done using a two player game which comprise a generator and a discriminator.
- The generator produces images that appear to be real.
- The discriminator tries to detect if an image is real or fake.

Generator



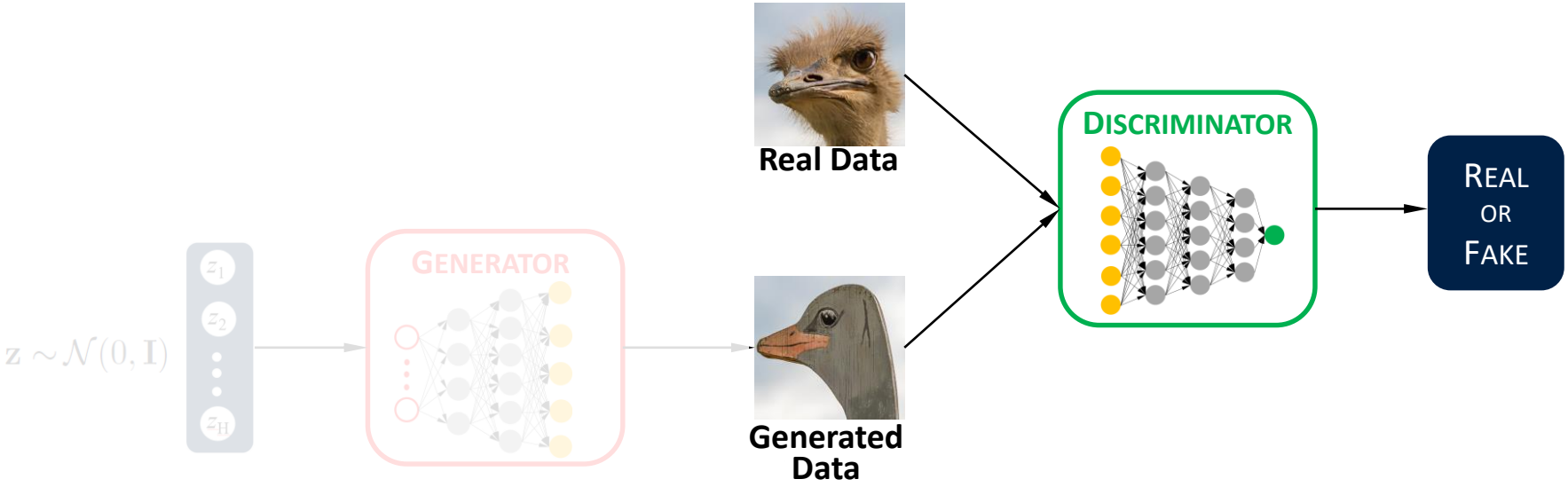
- Generator model:
  - $G_{\boldsymbol{\theta}}(\mathbf{z})$, where $\mathbf{z}$ is the input and $\boldsymbol{\theta}$ are the parameters of the model.
  - Input: Noise vector $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ which can be $\mathcal{N}(0, \mathbf{I})$.
  - $G_{\boldsymbol{\theta}}$ is a neural network model.

Discriminator



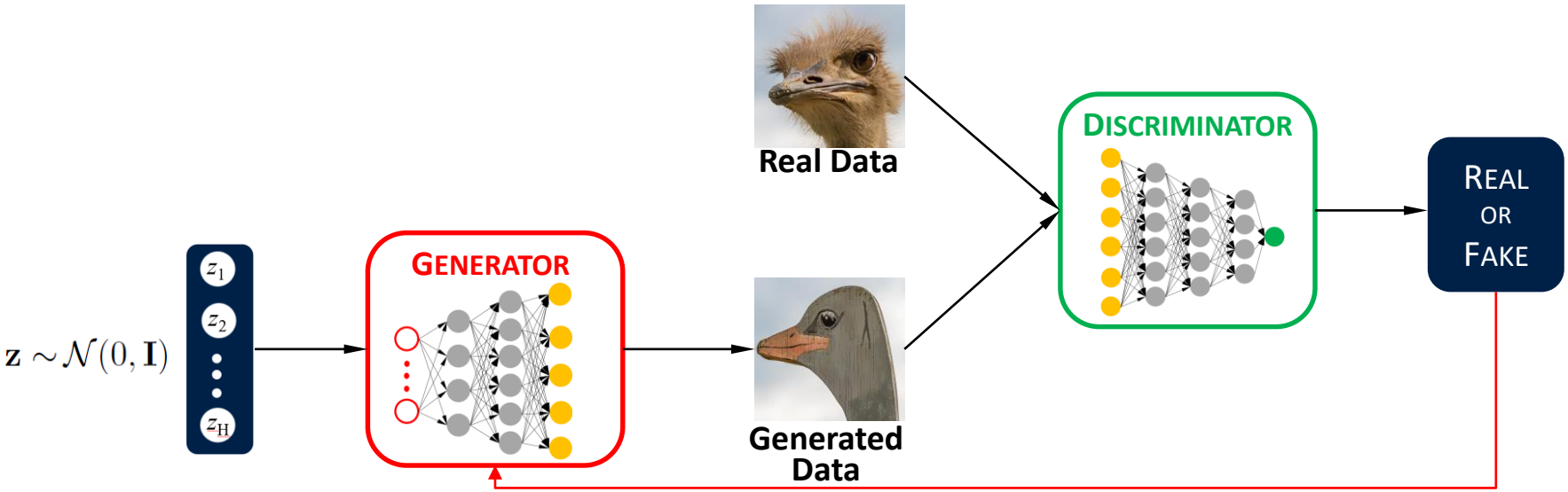
- Discriminator model:
  - $D_{\boldsymbol{\varphi}}(\mathbf{x})$, where $\mathbf{x}$ is the input to the discriminator and $\boldsymbol{\varphi}$ are the parameters.
  - Input can come from data or generator:
    - $\mathbf{x}$ if coming from the data.
    - $G_{\boldsymbol{\theta}}(\mathbf{z})$ if coming from the generator.
  - $D_{\boldsymbol{\varphi}}$ is a neural network model.

Discriminator



- The discriminator $D_{\varphi}(\mathbf{x})$ outputs a score between 0 and 1.
  - This is the probability of the image being real or fake.
  - The output is 0 if the image is fake and 1 if it is real.

Generator objective



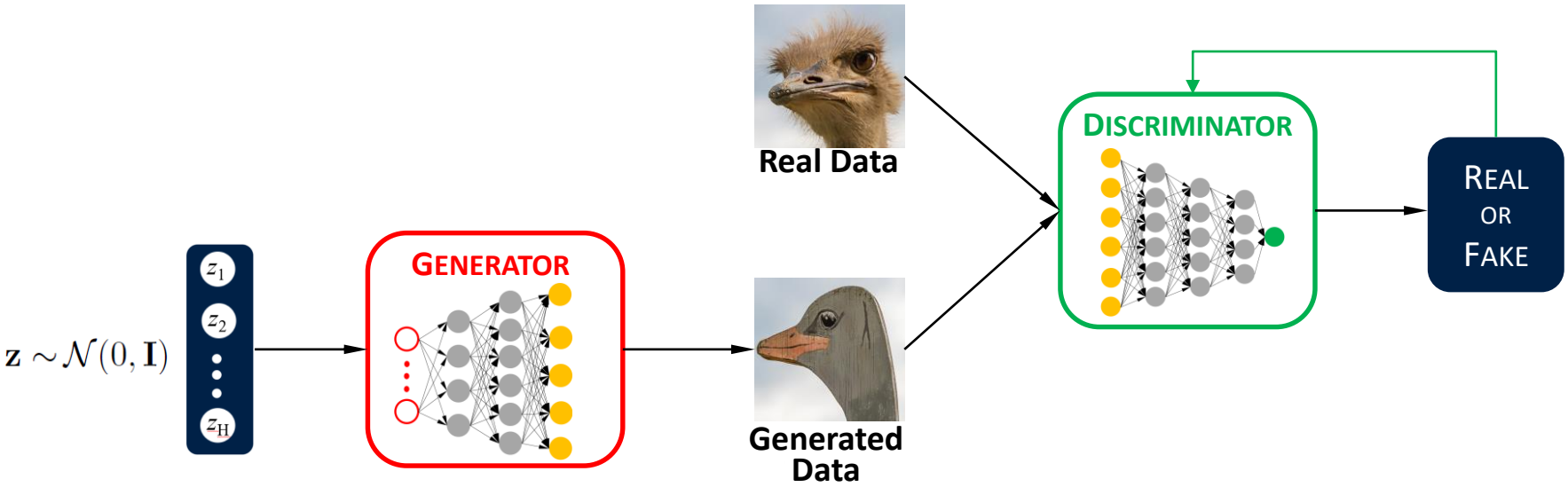
- The generator wants its output $G_{\boldsymbol{\theta}}(\mathbf{z})$ to be classified as real. Therefore it wants to

$$\max_{\boldsymbol{\theta}} \log D_{\boldsymbol{\varphi}}\big(G_{\boldsymbol{\theta}}(\mathbf{z})\big) \quad \text{or} \quad \min_{\boldsymbol{\theta}} \log\big(1 - D_{\boldsymbol{\varphi}}\big(G_{\boldsymbol{\theta}}(\mathbf{z})\big)\big)$$

- We want the generator to do this task for all possible values of $\mathbf{z}$ sampled from the distribution $p_{\mathbf{z}}(\mathbf{z})$. Therefore the objective becomes

$$\min_{\boldsymbol{\theta}} \int p_{\mathbf{z}}(\mathbf{z}) \log\big(1 - D_{\boldsymbol{\varphi}}\big(G_{\boldsymbol{\theta}}(\mathbf{z})\big)\big) \mathrm{d}\mathbf{z} = \min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}\big[\log\big(1 - D_{\boldsymbol{\varphi}}\big(G_{\boldsymbol{\theta}}(\mathbf{z})\big)\big)\big]$$

Discriminator objective



- The discriminator should assign high score to real images and low score to generated images.
  - maximize the score assigned to real images i.e.

$$\max_{\boldsymbol{\varphi}} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log D_{\boldsymbol{\varphi}}(\mathbf{x}) \right]$$

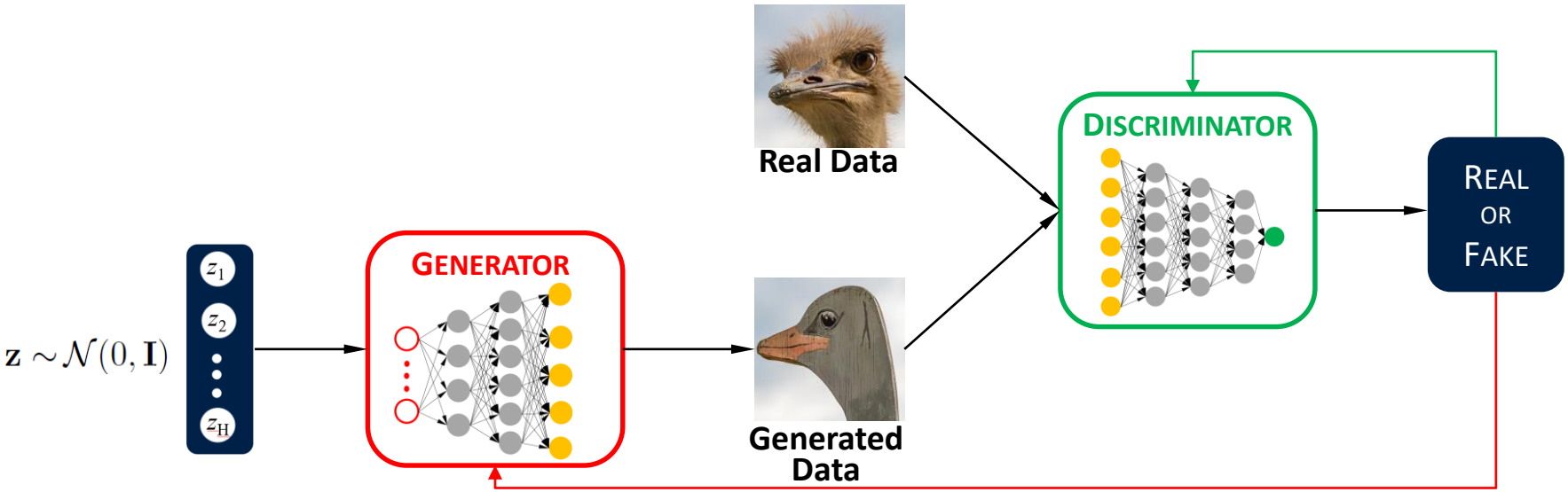
  - minimize the score assigned to generated images i.e.

$$\min_{\boldsymbol{\varphi}} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[\log D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z})) \right] \quad \text{or} \quad \max_{\boldsymbol{\varphi}} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[\log(1 - D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z}))) \right]$$

- So the combined discriminator objective function can be written as

$$\max_{\boldsymbol{\varphi}} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log D_{\boldsymbol{\varphi}}(\mathbf{x}) \right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[\log(1 - D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z}))) \right]$$

Combined objective function

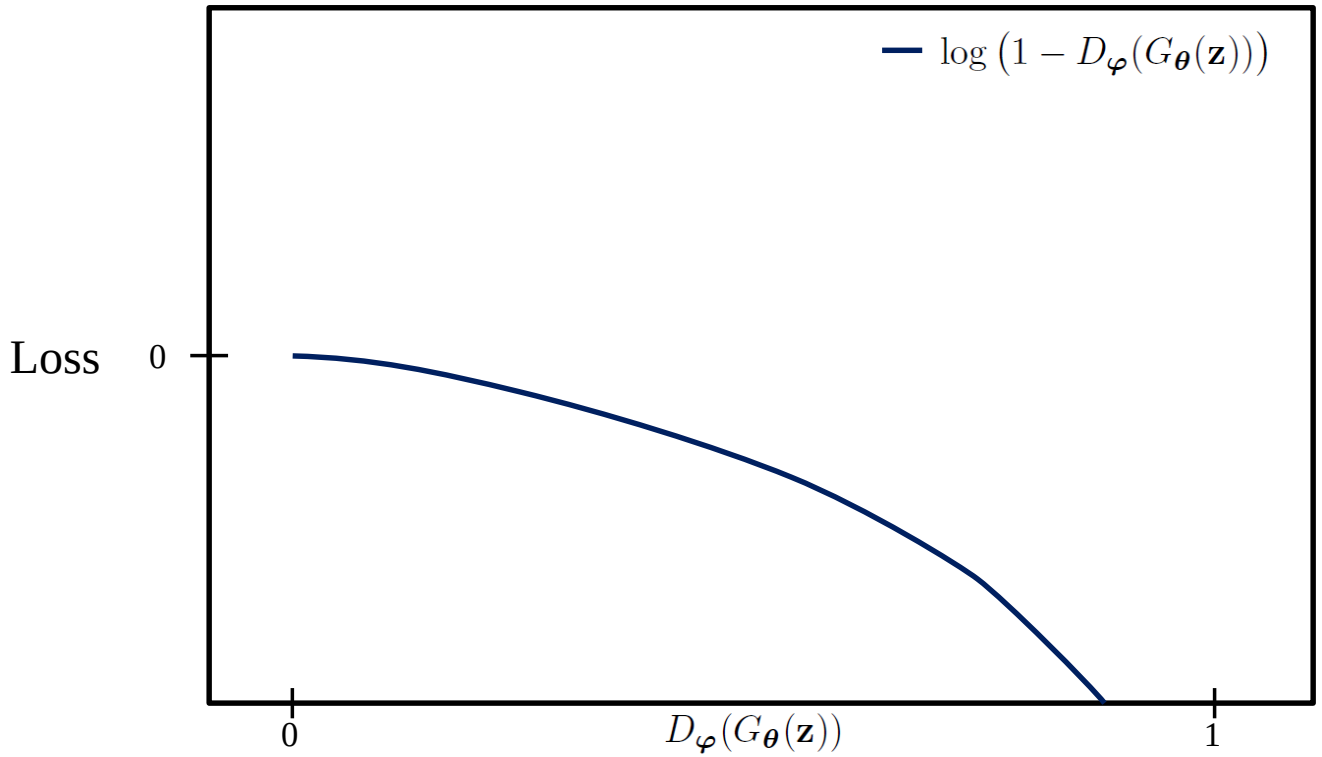


- The generator objective and the discriminator objective combined yields a minimax problem:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\varphi}} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log D_{\boldsymbol{\varphi}}(\mathbf{x}) \right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[\log(1 - D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z}))) \right]$$

- The first expectation is independent of $\boldsymbol{\theta}$.
- The second expectation is minimized w.r.t. $\boldsymbol{\theta}$ and maximized w.r.t. $\boldsymbol{\varphi}$.
  - Therefore the generator wants to minimize the second term while the discriminator wants to maximize it.

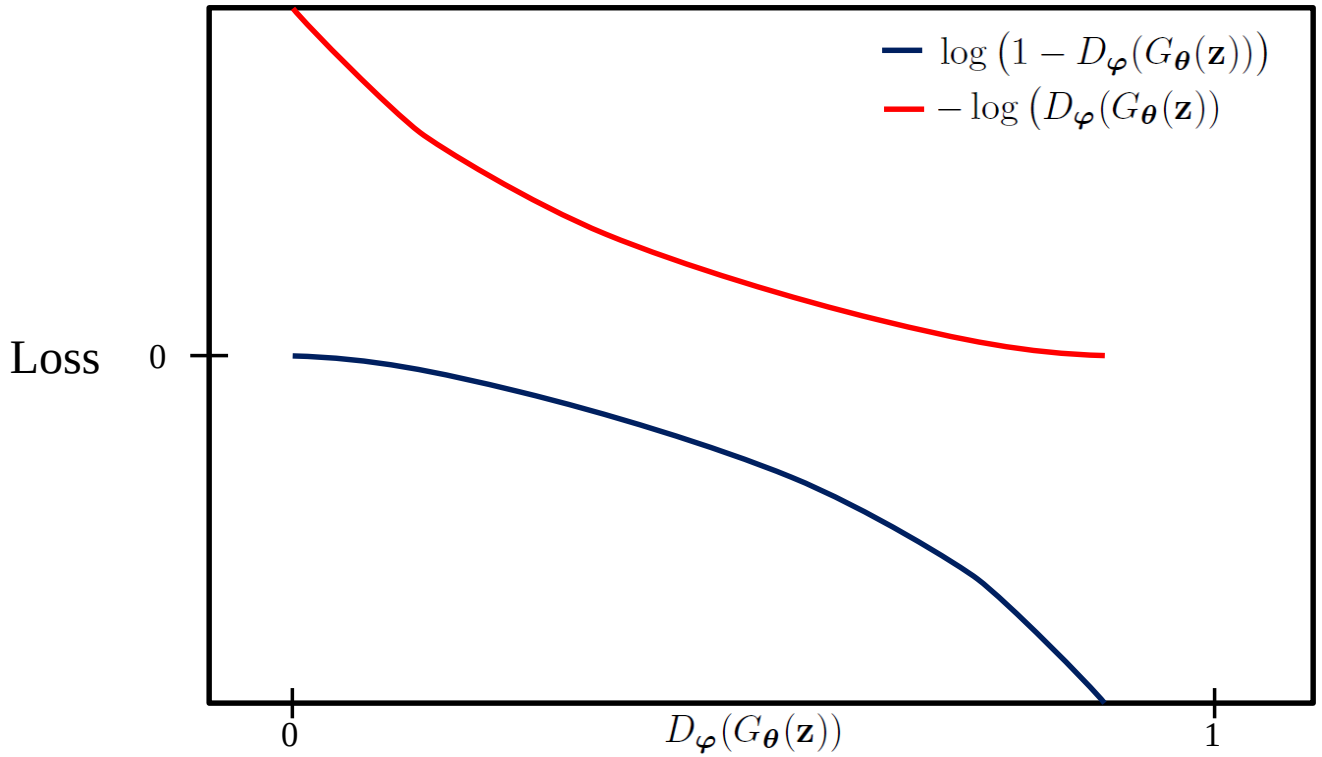
Problem in gradient



- Early on in training it is more likely that $D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z})) \approx 0$ as the generator has not learnt much and it is easy for the discriminator to identify the difference between real and fake examples.
- In such a situation, the gradient of $\log\left(1 - D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z}))\right)$ is close to zero.
  - The generator does not learn much in the beginning, i.e. there is little change in $\boldsymbol{\theta}$.

Figure for illustration only

Modified objective



- Originally we were performing minimization w.r.t. generator parameters $\boldsymbol{\theta}$

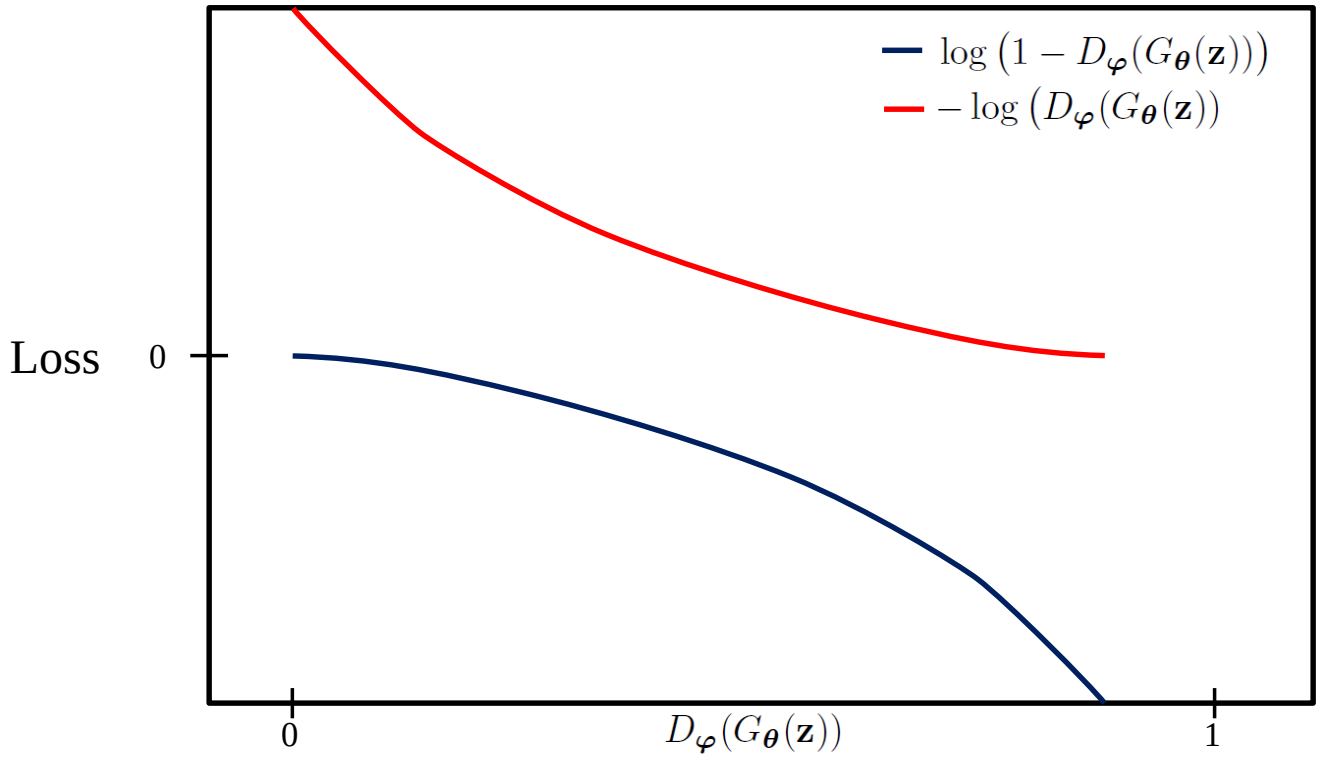
$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\log(1 - D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z})))\right]$$

- Now we use a modified version of the above (minimization) objective:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[-\log(D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z})))\right] \equiv \max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\log(D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z})))\right]$$

Figure for illustration only

Modified objective



- We modify the objective function to $-\log\left(D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z}))\right)$
  - This has large gradient when $D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z})) \approx 0$.
  - This enables the generator to learn more in the early training period.

Figure for illustration only

Algorithm

for number of iterations **do**

for j steps **do**

- Sample minibatch of M noisy samples $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(M)}\}$.
- Sample minibatch of M examples $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(M)}\}$ from $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\boldsymbol{\varphi}} \frac{1}{M} \sum_{m=1}^{M} \left[\log D_{\boldsymbol{\varphi}}(\mathbf{x}^{(m)}) \right] + \log(1 - D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z}^{(m)})))]$$

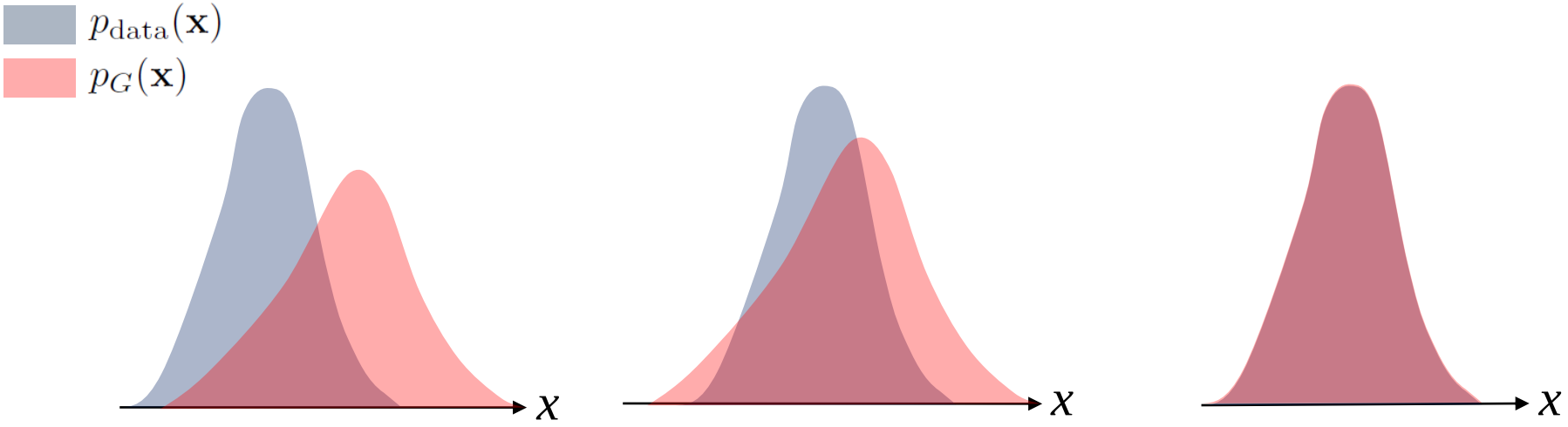
end for

- Sample minibatch of M noisy samples $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(M)}\}$.
- Update the generator by ascending its stochastic gradient:

$$\nabla_{\boldsymbol{\theta}} \frac{1}{M} \sum_{m=1}^{M} \left[\log(D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z}^{(m)}))) \right]$$

end for

$p_G = p_{data}$



- Distribution of true data: $p_{data}(\mathbf{x})$.
- Distribution of data produced by the generator $p_G(\mathbf{x})$.
- Eventually we want that

$$p_G(\mathbf{x}) = p_{data}(\mathbf{x})$$

- Can the described model achieve this?

Theoretical analysis

- Objective:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\big[\log D_{\boldsymbol{\varphi}}(\mathbf{x})\big] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\big[\log(1 - D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z})))\big]$$

- Can write the objective as

$$\min_G \max_D \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log D_{\boldsymbol{\varphi}}(\mathbf{x})\mathrm{d}\mathbf{x} + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z}) \log(1 - D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z})))\mathrm{d}\mathbf{z}$$

$$\Rightarrow \quad \min_G \max_D \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log D_{\boldsymbol{\varphi}}(\mathbf{x})\mathrm{d}\mathbf{x} + \int_{\mathbf{x}} p_G(\mathbf{x}) \log(1 - D_{\boldsymbol{\varphi}}(\mathbf{x}))\mathrm{d}\mathbf{x}$$

- Revised objective:

$$\min_G \max_D \mathcal{M}(G_{\boldsymbol{\theta}}, D_{\boldsymbol{\varphi}})$$

where

$$\mathcal{M}(G_{\boldsymbol{\theta}}, D_{\boldsymbol{\varphi}}) = \int_{\mathbf{x}} \big(p_{\text{data}}(\mathbf{x}) \log D_{\boldsymbol{\varphi}}(\mathbf{x}) + p_G(\mathbf{x}) \log(1 - D_{\boldsymbol{\varphi}}(\mathbf{x}))\big)\mathrm{d}\mathbf{x}$$

Theoretical analysis

- For a given generator $G_{\boldsymbol{\theta}}$, want to know the discriminator $D_{\boldsymbol{\varphi}}$ which maximizes the objective.
- The objective is maximized when the integrand is maximized. Differentiating the integrand w.r.t. $D_{\boldsymbol{\varphi}}$ yields

$$\frac{\mathrm{d}}{\mathrm{d}(D_{\boldsymbol{\varphi}}(\mathbf{x}))}\left(p_{\text{data}}(\mathbf{x})\log D_{\boldsymbol{\varphi}}(\mathbf{x}) + p_G(\mathbf{x})\log(1 - D_{\boldsymbol{\varphi}}(\mathbf{x}))\right) = 0$$

$$\left(p_{\text{data}}(\mathbf{x})\frac{1}{D_{\boldsymbol{\varphi}}(\mathbf{x})} - p_G(\mathbf{x})\frac{1}{1 - D_{\boldsymbol{\varphi}}(\mathbf{x})}\right)\frac{\mathrm{d}}{\mathrm{d}(D_{\boldsymbol{\varphi}}(\mathbf{x}))}D_{\boldsymbol{\varphi}}(\mathbf{x}) = 0$$

$$p_{\text{data}}(\mathbf{x})\frac{1}{D_{\boldsymbol{\varphi}}(\mathbf{x})} = p_G(\mathbf{x})\frac{1}{1 - D_{\boldsymbol{\varphi}}(\mathbf{x})}$$

$$D_{\boldsymbol{\varphi}}(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}$$

- Therefore for optimal discriminator we have

$$D^*_{\boldsymbol{\varphi}}(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}$$

Theoretical analysis

- Let $\mathcal{C}(G_{\boldsymbol{\theta}}) = \max \mathcal{M}(G_{\boldsymbol{\theta}}, D_{\boldsymbol{\varphi}}) = \mathcal{M}(G_{\boldsymbol{\theta}}, D^*_{\boldsymbol{\varphi}})$. Therefore we have

$$
\begin{aligned}
\mathcal{C}(G_{\boldsymbol{\theta}}) &= \int \left(p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})} + p_G(\mathbf{x}) \log \left(1 - \frac{p_{\text{data}}(\mathbf{x})}{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})} \right) \right) \mathrm{d}\mathbf{x} \\
&= \int \left(p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})} + p_G(\mathbf{x}) \log \left(\frac{p_G(\mathbf{x})}{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})} \right) \right) \mathrm{d}\mathbf{x} \\
&= \int \left(p_{\text{data}}(\mathbf{x}) \big(\log 2 + \log \frac{p_{\text{data}}(\mathbf{x})}{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})} \big) + p_G(\mathbf{x}) \Big(\log 2 + \right. \\
&\qquad \left. \log \left(\frac{p_G(\mathbf{x})}{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})} \right) \Big) - (p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})) \log 2 \right) \mathrm{d}\mathbf{x} \\
&= \int \left(p_{\text{data}}(\mathbf{x}) \Big(\log \frac{p_{\text{data}}(\mathbf{x})}{\frac{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}{2}} \Big) + p_G(\mathbf{x}) \Big(\log \left(\frac{p_G(\mathbf{x})}{\frac{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}{2}} \right) \Big) \right) \mathrm{d}\mathbf{x} \\
&\qquad - \log 2 \int \big(p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x}) \big) \mathrm{d}\mathbf{x} \\
&= \mathrm{KL}\left(p_{\text{data}}(\mathbf{x}) \middle|\middle| \frac{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}{2} \right) + \mathrm{KL}\left(p_G(\mathbf{x}) \middle|\middle| \frac{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}{2} \right) - 2 \log 2
\end{aligned}
$$

Theoretical analysis

Theorem: The global minimum of $\mathcal{C}(G_{\boldsymbol{\theta}})$ is achieved if and only if $p_G = p_{\text{data}}$.

$$\mathcal{C}(G_{\boldsymbol{\theta}}) = \text{KL}\left(p_{\text{data}}(\mathbf{x}) \middle|\middle| \frac{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}{2}\right) + \text{KL}\left(p_G(\mathbf{x}) \middle|\middle| \frac{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}{2}\right) - 2\log 2$$

- If $p_G = p_{\text{data}}$, then minimum of $\mathcal{C}(G_{\boldsymbol{\theta}})$ is attained
  - For $p_G = p_{\text{data}}$, we have $\min_G \mathcal{C}(G_{\boldsymbol{\theta}}) = -\log 4$ as $\text{KL}(p_G || p_G) = 0$
  - For $p_G \neq p_{\text{data}}$, we have $\min_G \mathcal{C}(G_{\boldsymbol{\theta}}) \geq -\log 4$ as $\text{KL}(.||.) \geq 0$
- If the minimum of $\mathcal{C}(G_{\boldsymbol{\theta}})$ is achieved, then $p_G = p_{\text{data}}$

$$\begin{aligned}\mathcal{C}(G_{\boldsymbol{\theta}}) &= \text{KL}\left(p_{\text{data}}(\mathbf{x}) \middle|\middle| \frac{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}{2}\right) + \text{KL}\left(p_G(\mathbf{x}) \middle|\middle| \frac{p_G(\mathbf{x}) + p_{\text{data}}(\mathbf{x})}{2}\right) - \log 4 \\ &= 2\text{JS}\left(p_{\text{data}}(\mathbf{x}) \middle|\middle| p_G(\mathbf{x})\right) - \log 4\end{aligned}$$

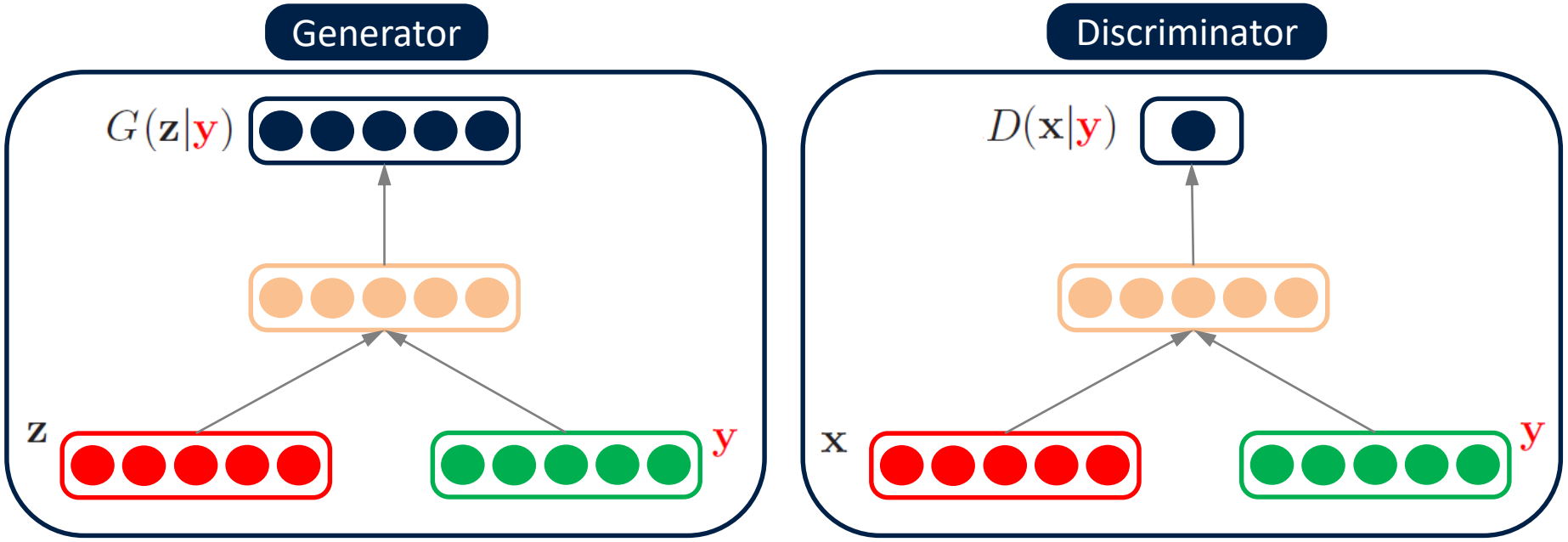
  - $\text{JS}\left(p_{\text{data}}(\mathbf{x}) \middle|\middle| p_G(\mathbf{x})\right) = 0$ only when $p_G = p_{\text{data}}$

GAN Architectures

Conditional GAN

- The generator and the discriminator are conditioned on some extra information e.g. class labels.
- Conditioning is done by feeding the extra information (say $\mathbf{y}$) as extra input layer into $D_{\boldsymbol{\varphi}}$ and $G_{\boldsymbol{\theta}}$.
- Training objective:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\varphi}} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \left[\log D_{\boldsymbol{\varphi}}(\mathbf{x}|\mathbf{y}) \right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} \left[\log(1 - D_{\boldsymbol{\varphi}}(G_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{y}))) \right]$$

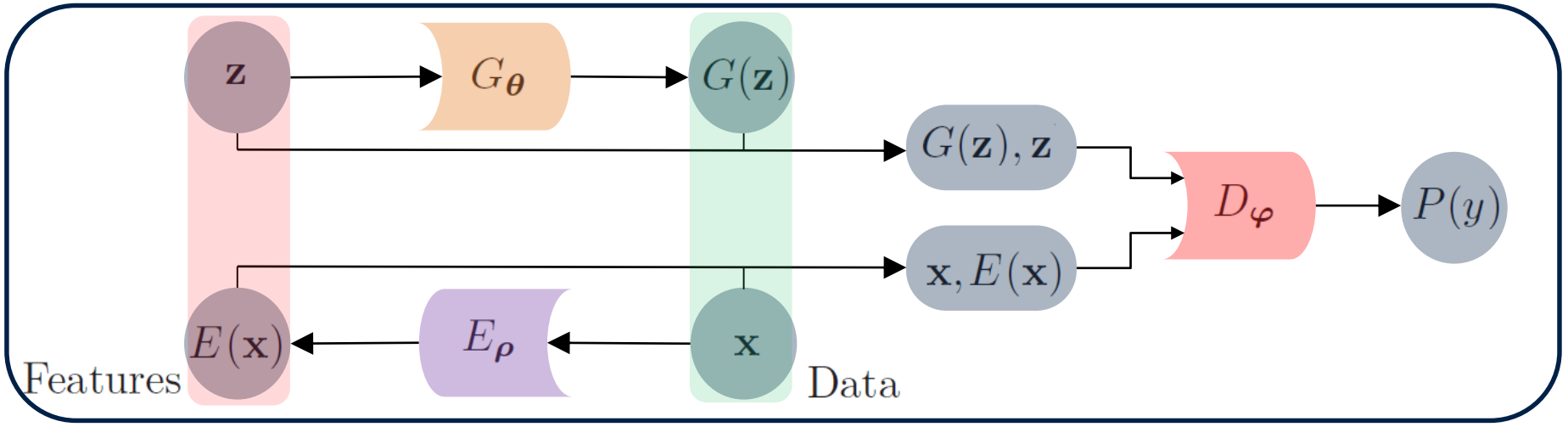


Source: Mirza and Osindero, *Conditional generative adversarial nets*, NIPS 2014.

Bidirectional GAN (BiGAN)

- Predicting the semantic latent representations given data may be useful for problems where semantics are relevant.
  - Standard GANs do not have an inverse mapping from data to latent representations.
- BiGANs comprise a generator (as in standard GANs) and an encoder that maps data $\mathbf{x}$ to latent representations $\mathbf{z}$.
- BiGAN training objective:

$$\min_{\boldsymbol{\theta},\boldsymbol{\rho}} \max_{\boldsymbol{\varphi}} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\Big[\mathbb{E}_{\mathbf{z}\sim p_E(.|\mathbf{x})}\big[\log D_{\boldsymbol{\varphi}}(\mathbf{x},\mathbf{z})\big]\Big] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\Big[\mathbb{E}_{\mathbf{x}\sim p_G(.|\mathbf{z})}\big[\log(1-D_{\boldsymbol{\varphi}}(\mathbf{x},\mathbf{z}))\big]\Big]$$



Source: Donahue et. al., *Adversarial feature learning*, ICLR 2017.

Generating videos

- Architecture enforces a static background and moving foreground.
- A two-stream architecture is used where the generator is governed by the combination:

$$G(z) = m(z) \odot f(z) + \big(1 - m(z)\big) \odot b(z)$$

  - $b(z)$ produces a spatial image that is replicated over time, it tries to enforce a background model in the generations.
  - $f(z)$ produces a spatio-temporal cuboid, it acts as foreground model.
  - $m(z)$ is the spatio-temporal mask for each pixel location and timestep.

Source: Vondrick et. al. '*Generating videos with scene dynamics*", NIPS 2016

Story GAN

- New task: Story visualization
  - Model generates a new story-to-image-sequence
- Given a multi-sentence paragraph, the story is visualized by generating a sequence of images, one for each sentence.

Source: Li et. al. "StoryGAN: A sequential conditional GAN for story visualization", CVPR 2019

Story GAN

- StoryGAN creates a sequence of images to describe an input story $S = [s_1, s_2, .., s_T]$, where s_t is the tth sentence.
- Key components:
  - Context Encoder to track the flow of the story.
  - Story discriminator for consistency in generated sequences.
  - Image discriminator to enhance the quality of the image.

Source: Li et. al. "StoryGAN: A sequential conditional GAN for story visualization", CVPR 2019

Cycle GAN

- Image-to-Image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using using a training set of aligned image pairs.
- But for many tasks, paired training data may not be available
- Here an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples.

Paper: Zhu et.al. "Unpaired Image-to-Image translation using Cycle-consistent adversarial network"

Cycle GAN

- Goal: Learn a mapping $G : X \rightarrow Y$ such that the distribution of images from $G(X)$ is indistinguishable from the distribution Y using an adversarial loss.
- Since the mapping G is highly under-constrained, we couple G with an inverse mapping $F : Y \rightarrow X$.
  - Introduce a cycle consistency loss to enforce $F(G(X)) \approx X$
- Learn mapping functions between two domains X and Y given training samples $\{x^{(n)}\}_{n=1}^{N}$ where $x^{(n)} \in X$ and $\{y^{(m)}\}_{m=1}^{M}$ where $y^{(m)} \in Y$.
- There are two mappings in the model $G : X \rightarrow Y$ and $F : Y \rightarrow X$.

Paper: Zhu et.al. "Unpaired Image-to-Image translation using Cycle-consistent adversarial network"

Cycle GAN

- For the mapping $G : X \rightarrow Y$ and its discriminator D_Y, the objective can be written as

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}\big[\log D_Y(y)\big] + \mathbb{E}_{x \sim p_{\text{data}}(x)}\big[(1 - \log D_Y(G(x)))\big]$$

- For the mapping $F : Y \rightarrow X$ and its discriminator D_X, the objective can be written as

$$\mathcal{L}_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{\text{data}}(x)}\big[\log D_X(x)\big] + \mathbb{E}_{y \sim p_{\text{data}}(y)}\big[(1 - \log D_X(F(y)))\big]$$

- Cycle consistency loss is defined as

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}\big[\big|\big|F(G(x)) - x\big|\big|_1\big] + \mathbb{E}_{y \sim p_{\text{data}}(y)}\big[\big|\big|G(F(y)) - y\big|\big|_1\big]$$

- Full objective:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F)$$

Paper: Zhu et.al. "Unpaired Image-to-Image translation using Cycle-consistent adversarial network"

Stack GAN

- Goal: Synthesizing high quality images from text descriptions.
- Stage-I GAN sketches the primitive shape and colors based on text description.
- Stage-II GAN takes Stage-I results and given text as inputs, and generates high-resolution images with photo-realistic details.
- Training: D_0 and G_0 of Stage-I GAN are first iteratively trained by fixing Stage-II GAN. Then D and G of Stage-II GAN are iteratively trained by fixing Stage-I GAN.

Source: Zhang et. al. "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks", ICCV 2017

LapGAN

- LapGAN: Laplacian pyramid of generative adversarial networks.
- Generation high-resolution images, which are complex and high-dimensional, is challenging.
- LapGAN builds a series of generative models, each of which captures image structure at a particular scale.

Source: Denton et. al., *Deep generative image models using a Laplacian pyramid of advesarial networks*

LapGAN

- Sampling procedure:
  - At each scale a convolutional network based generative model is trained.
  - The output (image) at each stage is upsampled and used as a conditioning variable for the next level generative model.
  - The generator takes as inputs a noise sample and the upsampled image (from the previous level) to generate a difference image.
  - The difference image is added to the previous upsampled image to create a new output.
  - The output is again upsampled and taken as an input to the next level.

Source: Denton et. al., *Deep generative image models using a Laplacian pyramid of advesarial networks*

Cycle^{2}GAN

- Task: Keypoint-guided image generation
- Applications: Entertainment, virtual reality, data augmentation
- The task requires a high-level semantic understanding of the mapping between the input images and the output images since the objects in the inputs may have arbitrary poses, sizes, locations.

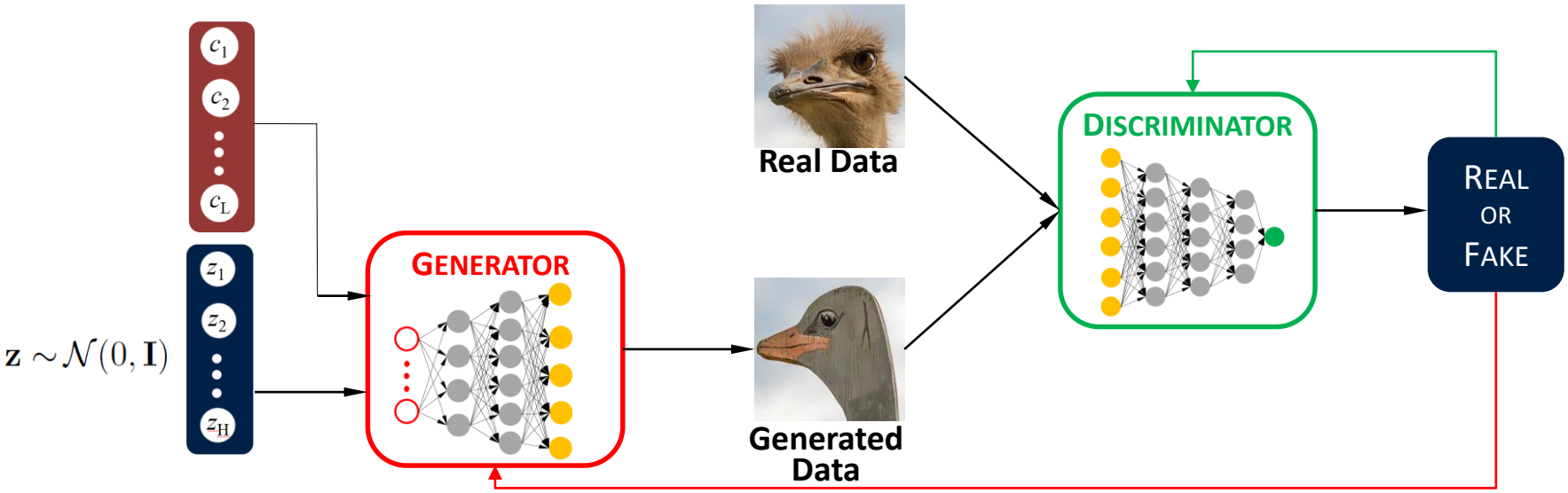
Source: Tang et. al. "Cycle In Cycle Generative Adversarial Networks for Keypoint-Guided Image Generation", ACM MM, 2019

Cycle^{2}GAN

- Model comprise three cycled sub-networks:
  - Image generation cycle – I2I2I: $\left[x, L_y\right] \rightarrow \left[y^*, L_x\right] \rightarrow x^*$
  - Keypoint generation cycles – K2G2K: $\left[x, L_y\right] \rightarrow y^* \rightarrow L_y^*$
  - Keypoint generation cycles – K2R2K: $\left[y^*, L_x\right] \rightarrow x^* \rightarrow L_x^*$

Source: Tang et. al. "Cycle In Cycle Generative Adversarial Networks for Keypoint-Guided Image Generation", ACM MM, 2019

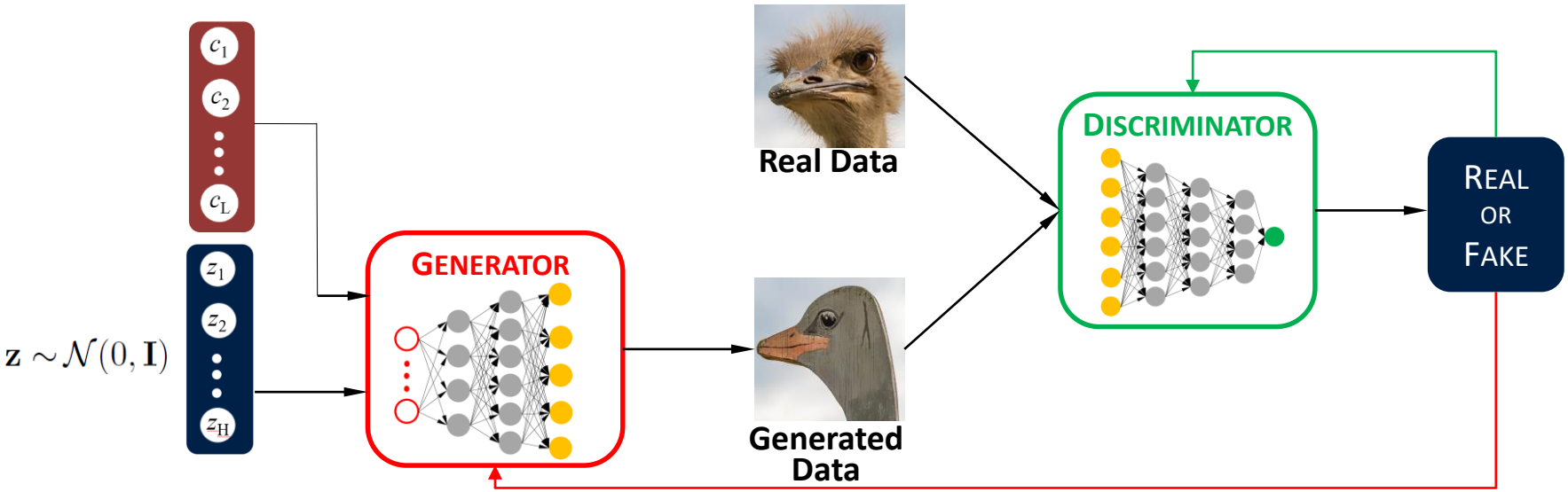
InfoGAN



- In the standard GAN formulation, the generator does not impose any restriction on the manner in which the noise vector $\mathbf{z}$ is used.
- So $\mathbf{z}$ could be used by the generator in a highly entangled way.
  - In that case the individual dimensions of $\mathbf{z}$ will not correspond to semantic features of the data.
- InfoGAN decomposes the input noise vector into two components:
  1. $\mathbf{z}$, which is treated as a source of incompressible noise
  2. $\mathbf{c}$, which will target the salient semantic features of the data

Source: Chen et. al. "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets", NIPS 2016

InfoGAN



- Standard GANs may ignore $\mathbf{c}$ to find a solution such that $p_G(\mathbf{x}|\mathbf{c}) = p_G(\mathbf{x})$
- In information theory, the mutual information $I(x; y)$ measures the reduction in uncertainty in x when y is observed.
- Generator output: $G(\mathbf{z}, \mathbf{c})$
- InfoGAN uses an information-theoretic regularization: high mutual information between $\mathbf{c}$ and $G(\mathbf{z}, \mathbf{c})$.
- The information-regularized minimax game becomes:

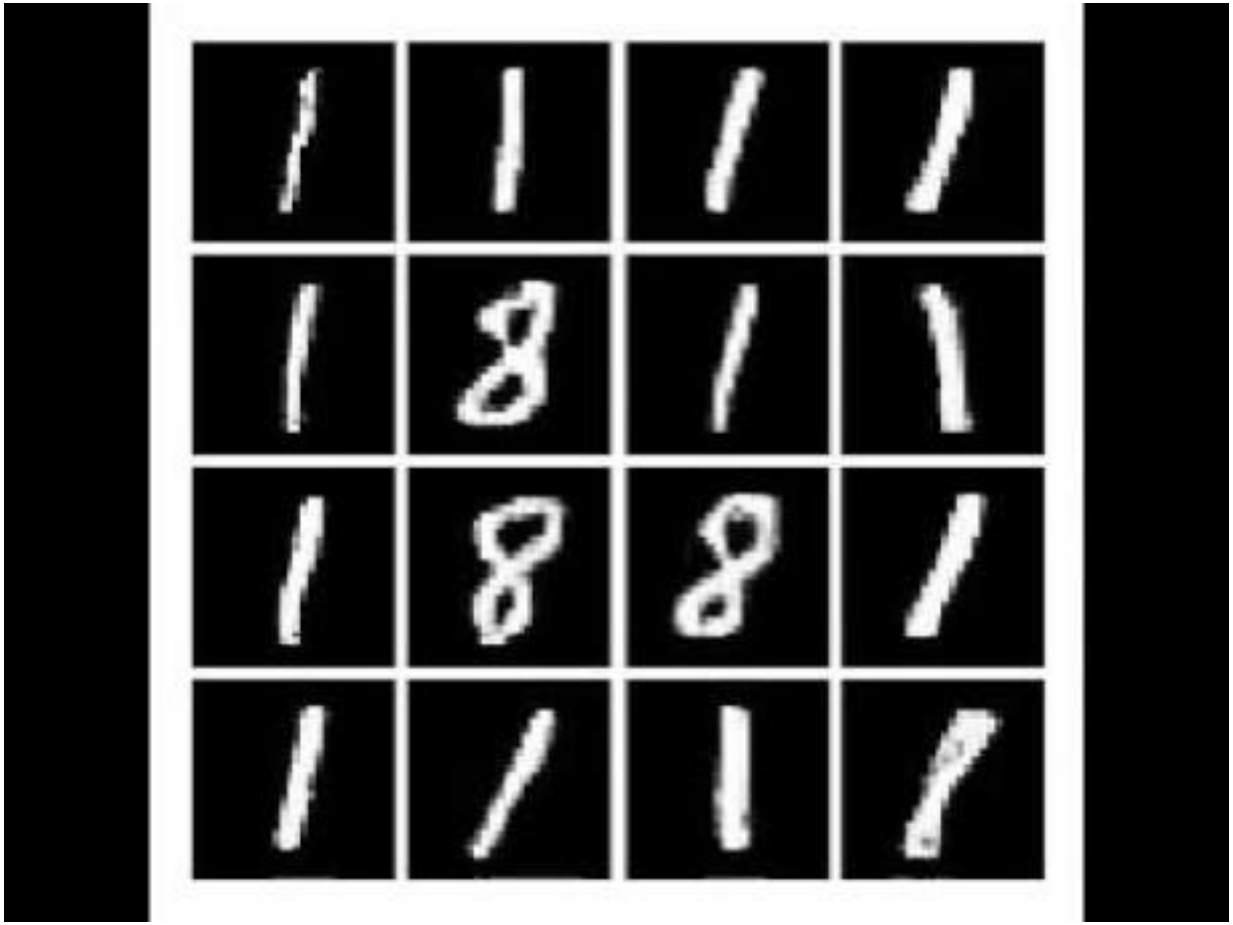
$$\min_G \max_D V(D, G) - \lambda I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$$

Source: Chen et. al. "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets", NIPS 2016

GAN
Training Issues

Mode collapse

- Generator produces outputs that corresponds to a small subspace and not the entire distribution.

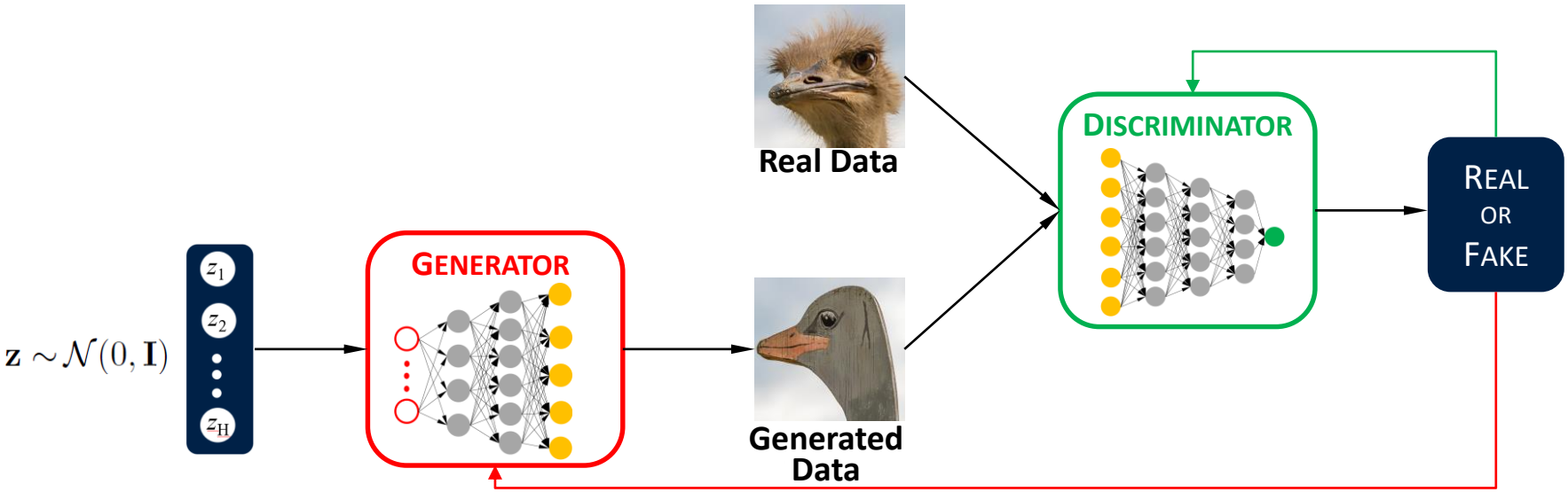


Link: https://www.youtube.com/watch?v=ktxhiKhWoEE

Mode collapse

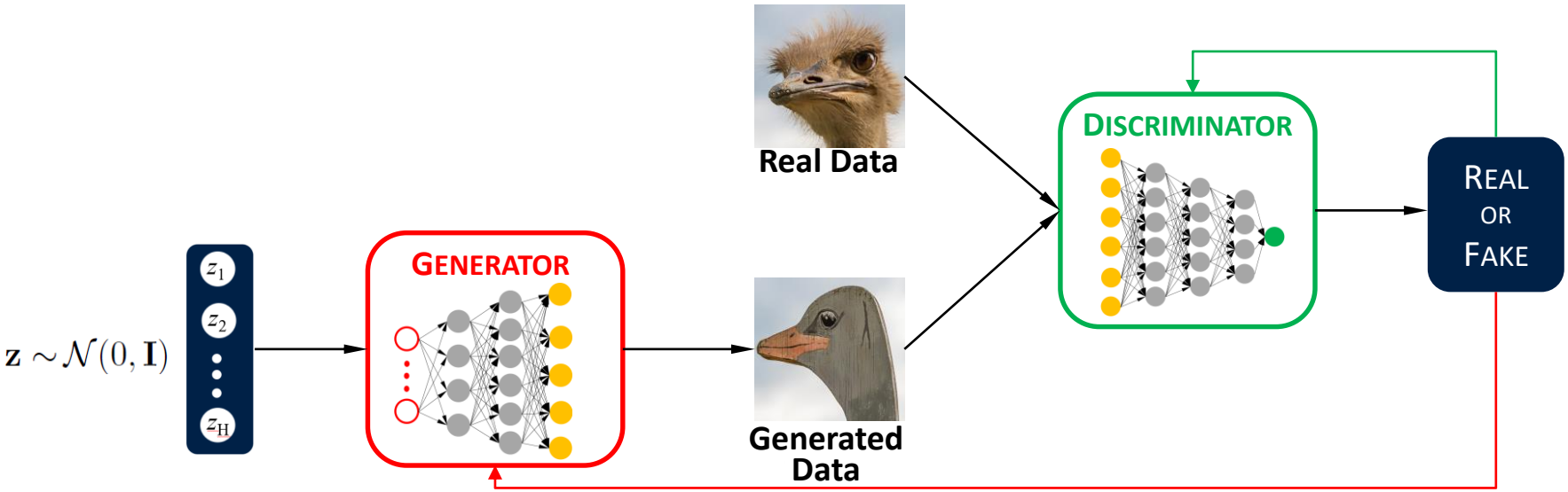
- Generator produces outputs that corresponds to a small subspace and not the entire distribution.
- Suppose if the generator is trained extensively without any updates to the discriminator, then it may converge to a particular output that fools the discriminator the most.
  - From the discriminator's perspective, this output could be the most realistic.
  - So the generator output may collapse to a few modes, and there is less sample variety.
- When the discriminator is retrained, the easiest way for it to detect the generated data is to detect the modes.
  - This deteriorates its capability to detect others.
- The generator and discriminator are both overfitting to exploit the weakness of the opponent in short-term.
- Thus the generator rotates through a small set of output types.

Feature matching



- Address instability of GANs
  - Specifies a new objective for the generator which prevents it from overfitting on the current discriminator.
- Main idea: Statistics of the generated image must match statistics of the real images.
- Instead of maximizing the output of the discriminator, the generator is trained to match the expected value of the features on an intermediate layer of the discriminator.
- Suppose $\mathbf{h}(\mathbf{x})$ denotes the activations on an intermediate layer of the discriminator.

Feature matching



- Then the new generator objective can be defined as:

$$\left|\left|\mathbb{E}_{\mathbf{x} \sim p_{data}}[\mathbf{h}(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}}[\mathbf{h}(G_{\boldsymbol{\theta}}(\mathbf{z}))]\right|\right|_2^2$$

- The discriminator is trained in the usual way.

Minibatch discriminator

- An intermediate layer of the discriminator network is augmented to measure some form of distance between the current input and all other inputs in that minibatch (in the real/fake category).
- The discriminator while classifying a particular example is able to use information from other examples in the minibatch.

Source: Salimans et. al. Improved Techniques for Training GANs, NIPS 2016.

References

- Goodfellow et. al., “Generative adversarial nets”, NIPS 2014
- Mirza and Osindero, “Conditional generative adversarial nets”, NIPS 2014
- Denton et. al., “Deep generative image models using a Laplacian pyramid of advesarial networks”, NIPS 2015
- Donahue et. al., “Adversarial feature learning”, ICLR 2017
- Zhu et.al., ”Unpaired Image-to-Image translation using Cycle-consistent adversarial networks”, ICCV 2017
- Vondrick et. al., “Generating videos with scene dynamics”, NIPS 2016
- Li et. al., “StoryGAN: A sequential conditional GAN for story visualization”, CVPR 2019
- Zhang et. al., “StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks”, ICCV 2017
- Tang et. al., “Cycle In Cycle Generative Adversarial Networks for Keypoint-Guided Image Generation”, ACM MM, 2019
- Chen et. al., “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets”, NIPS 2016
- Salimans et. al., “Improved Techniques for Training GANs”, NIPS 2016