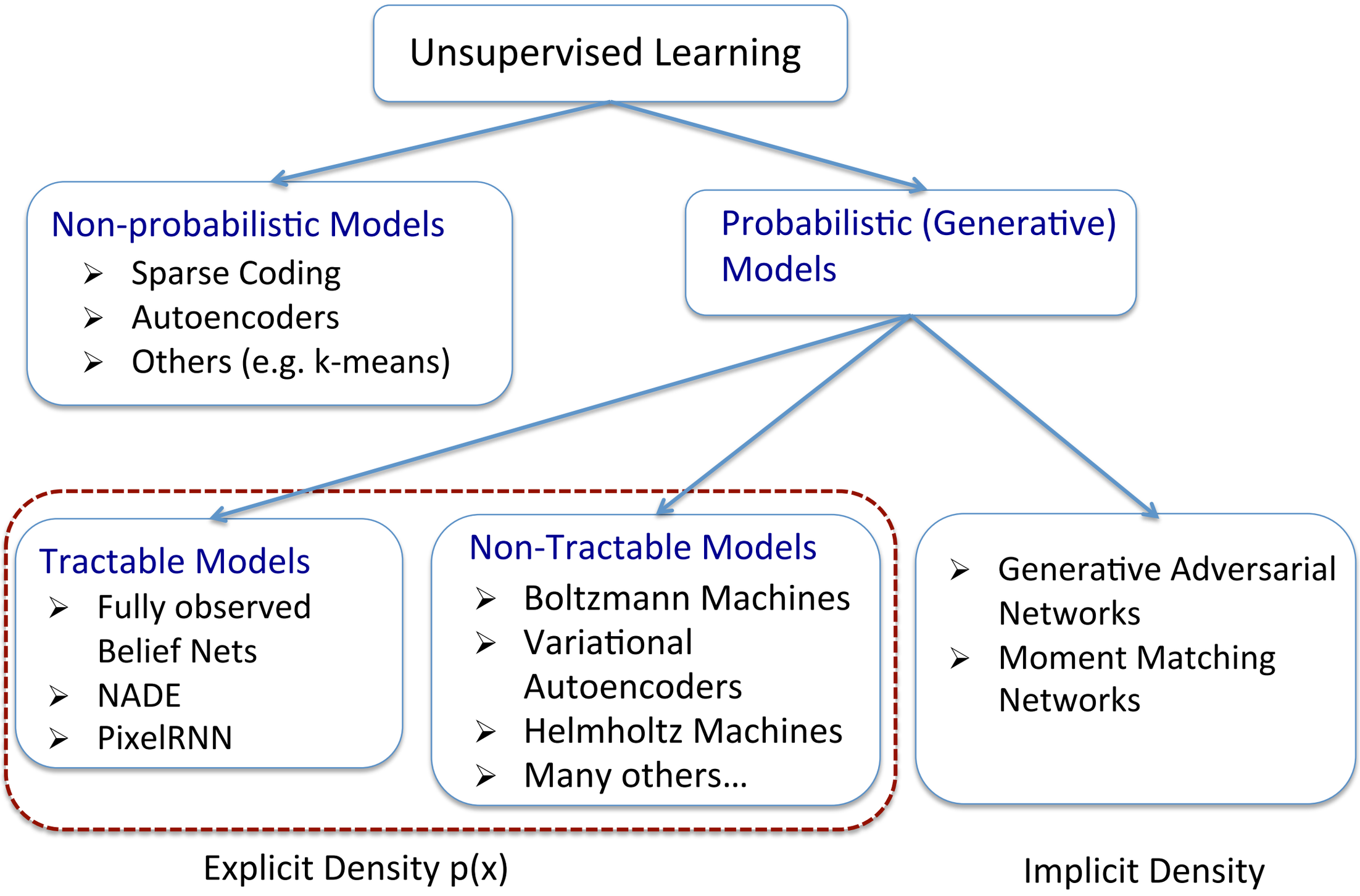
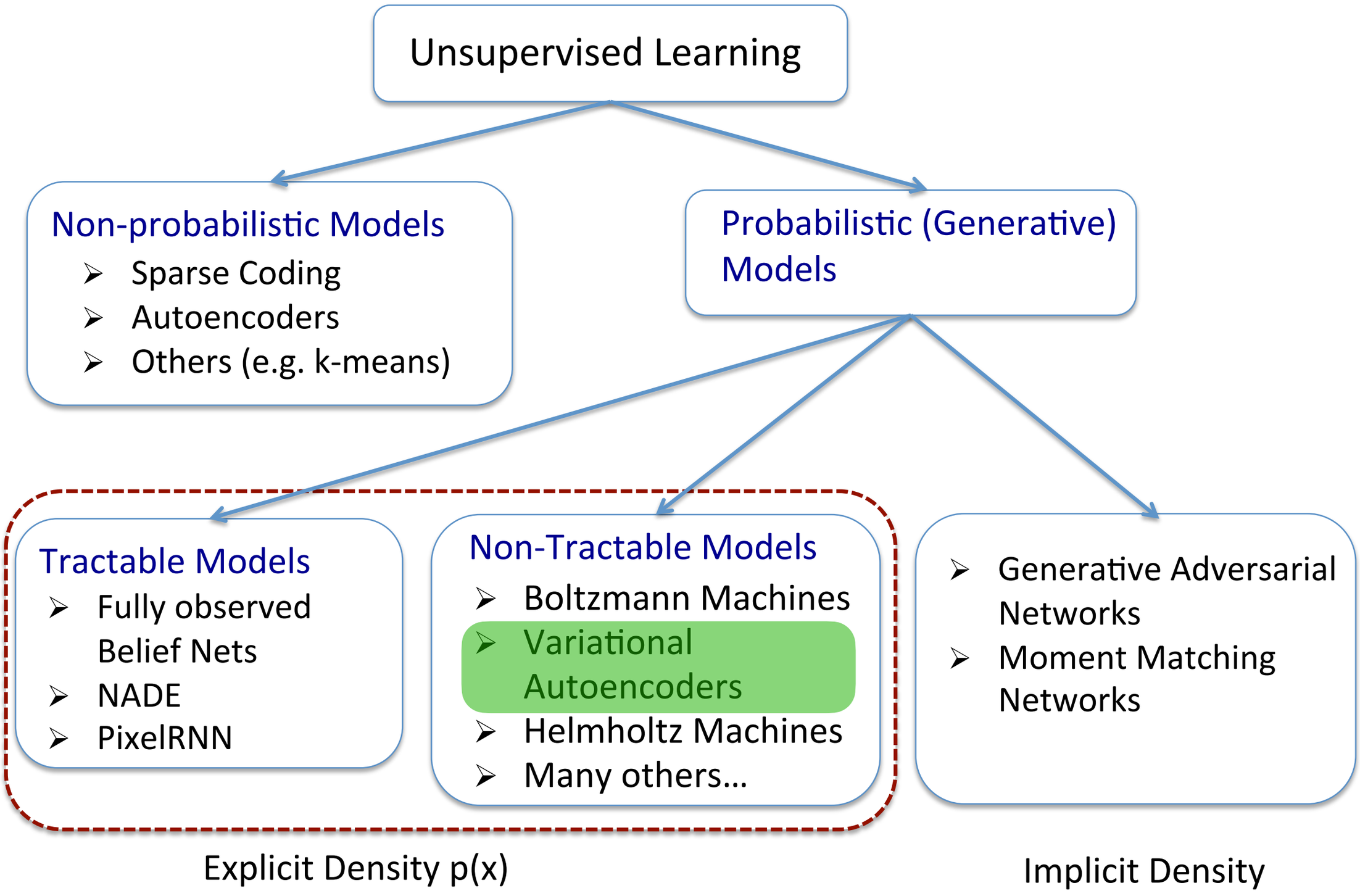
Deep Learning with Neural Networks

Introduction to Generative Adversarial Networks

Alejandro Lancho Serrano, alancho@ing.uc3m.es

Material original por: Pablo Martínez Olmos


Unsupervised Learning
Non-probabilistic Models
Sparse Coding
Autoencoders
Others (e.g. k-means)
Probabilistic (Generative) Models
Tractable Models
Fully observed Belief Nets
NADE
PixelRNN
Non-Tractable Models
Boltzmann Machines
Variational Autoencoders
Helmholtz Machines
Many others…
Generative Adversarial Networks
Moment Matching Networks
Explicit Density p(x)
Implicit Density


Unsupervised Learning
Non-probabilistic Models
Sparse Coding
Autoencoders
Others (e.g. k-means)
Probabilistic (Generative) Models
Tractable Models
Fully observed Belief Nets
NADE
PixelRNN
Non-Tractable Models
Boltzmann Machines
Variational Autoencoders
Helmholtz Machines
Many others…
Generative Adversarial Networks
Moment Matching Networks
Explicit Density p(x)
Implicit Density

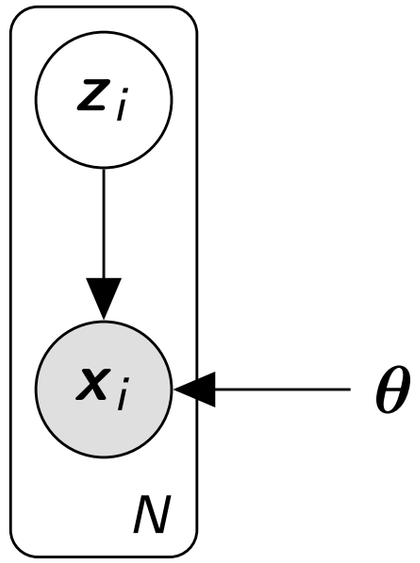
uc3m

Variational Autoencoder (Kigma & Welling 2014)

Consider a set of i.i.d observations $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$, $i = 1, \ldots, N$. Let $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N$.

Generative Model using a Latent Space

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z}$$



- $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_k)$ $(k \leq d)$ (low-dimensional embedding)
- $\boldsymbol{x}|\boldsymbol{z} \sim \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{z}), \text{diag}(\sigma_{\boldsymbol{\theta}}(\boldsymbol{z})))$
- $\mu_{\boldsymbol{\theta}}(\boldsymbol{z})$ and $\log(\sigma_{\boldsymbol{\theta}}(\boldsymbol{z}))$ are the outputs of a NN $\mathbb{R}^k \to \mathbb{R}^d$ with parameter vector $\boldsymbol{\theta}$ (Decoding network).

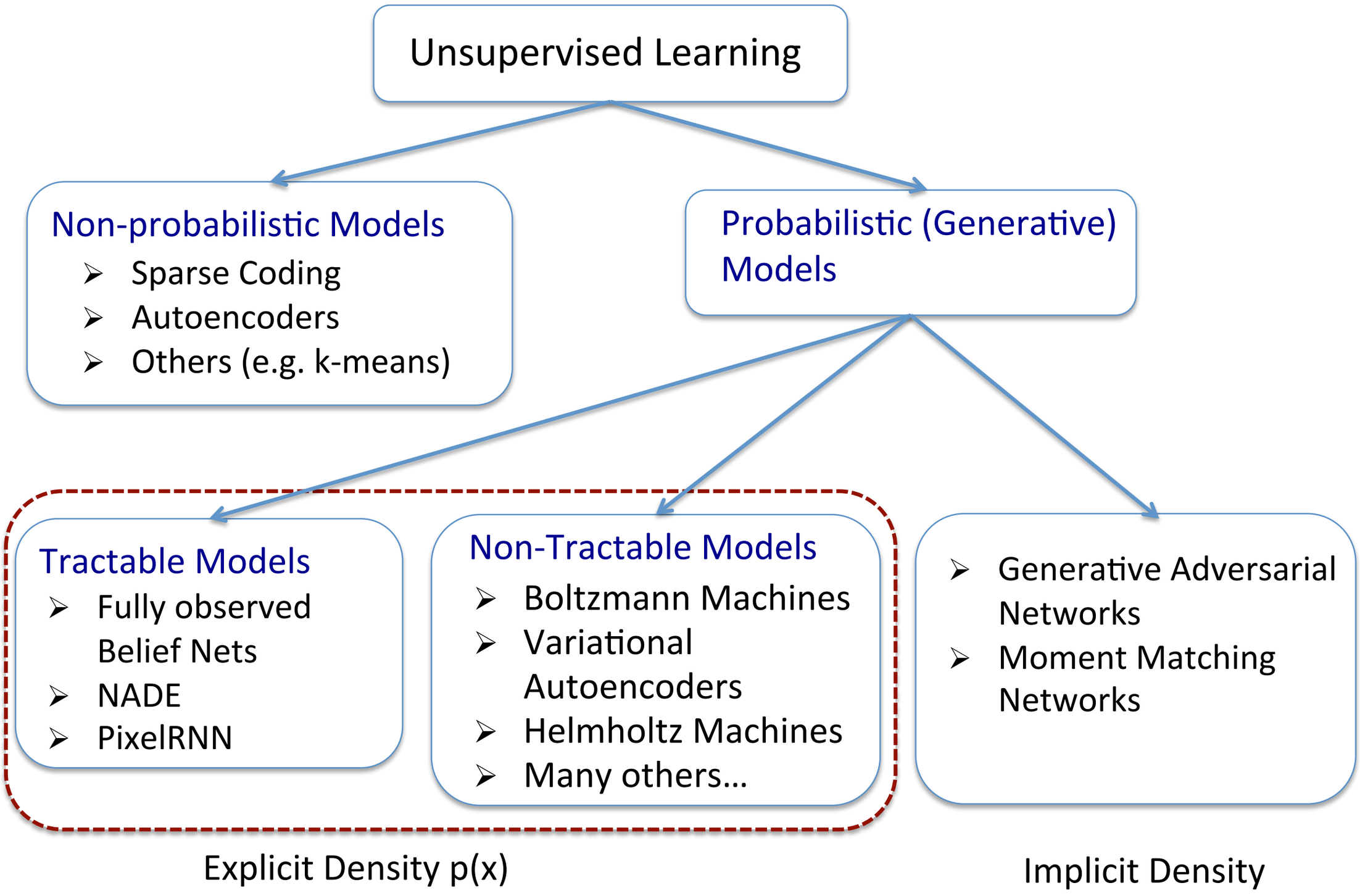
Variational Autoencoder Pros and Cons summary

Pros:

- **Stability** and **ease of training**
- **Probabilistic interpretation** through the probabilistic latent space
- Easy to get **diverse samples** and avoid mode collapse
- Good for **semi-supervised** and structured learning tasks
  - How? conditioning latent space
- Suitable for applications where **uncertainty quantification** is needed
  - Anomaly detection, decision making, etc.

Cons:

- Prone to give **blurry samples** due to reconstruction error terms based on MSE
- **Limited** in **capturing complex distributions**
  - KL term pulls the latent space towards a Gaussian distribtions
    - Limits generation quality in some cases


Unsupervised Learning
Non-probabilistic Models
Sparse Coding
Autoencoders
Others (e.g. k-means)
Probabilistic (Generative) Models
Tractable Models
Fully observed Belief Nets
NADE
PixelRNN
Non-Tractable Models
Boltzmann Machines
Variational Autoencoders
Helmholtz Machines
Many others…
Generative Adversarial Networks
Moment Matching Networks
Explicit Density p(x)
Implicit Density

Unsupervised Learning

Non-probabilistic Models

- Sparse Coding
- Autoencoders
- Others (e.g. k-means)

Probabilistic (Generative) Models

Tractable Models

- Fully observed Belief Nets
- NADE
- PixelRNN

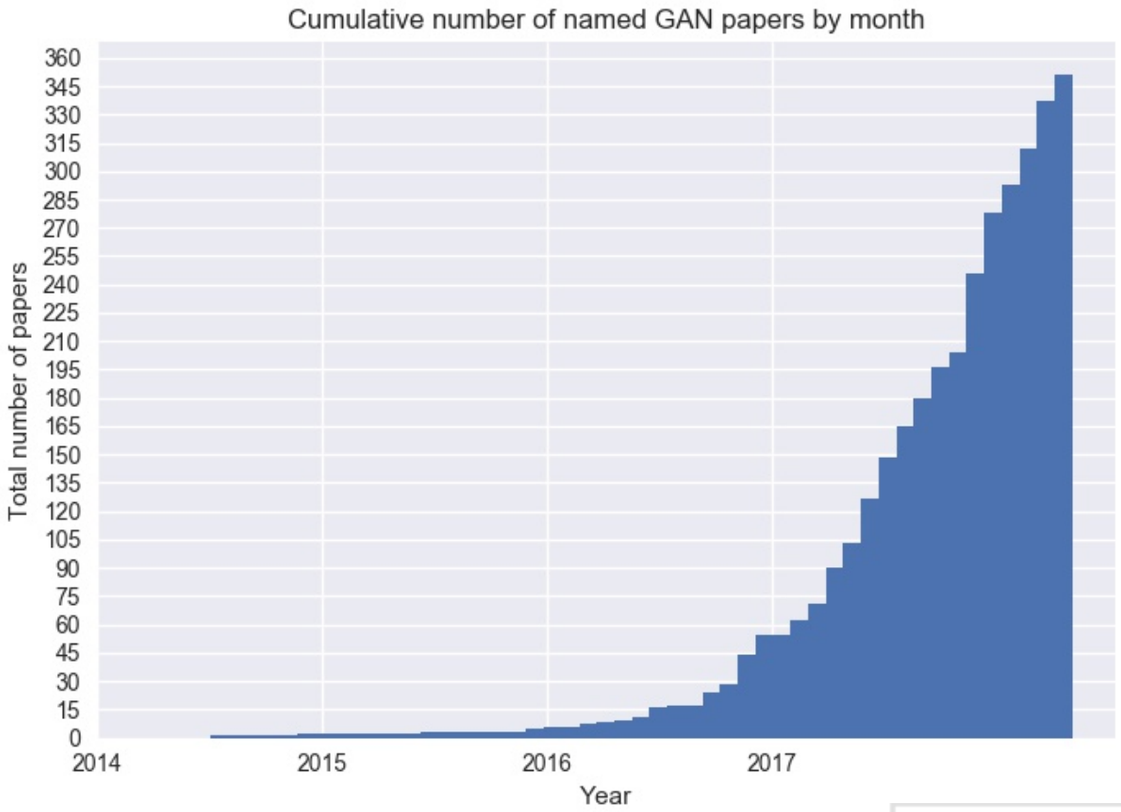
Non-Tractable Models

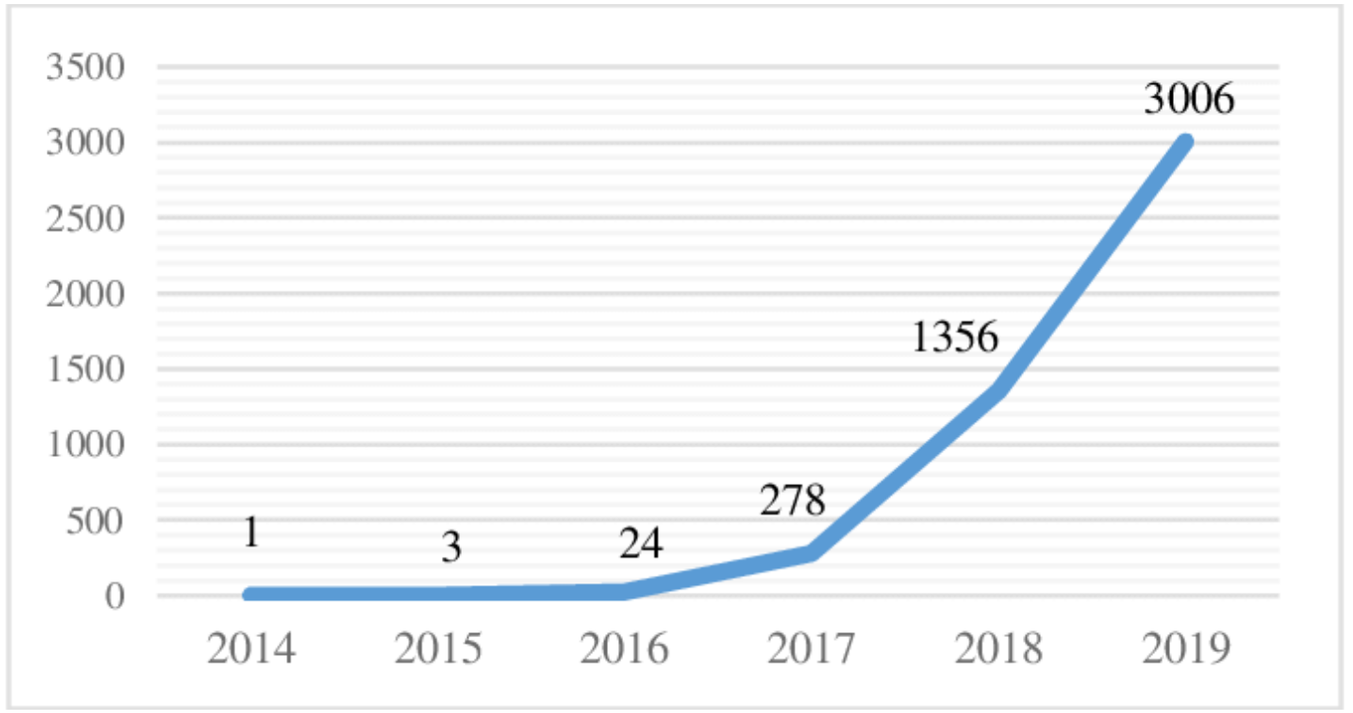
- Boltzmann Machines
- Variational Autoencoders
- Helmholtz Machines
- Many others…

Explicit Density p(x)

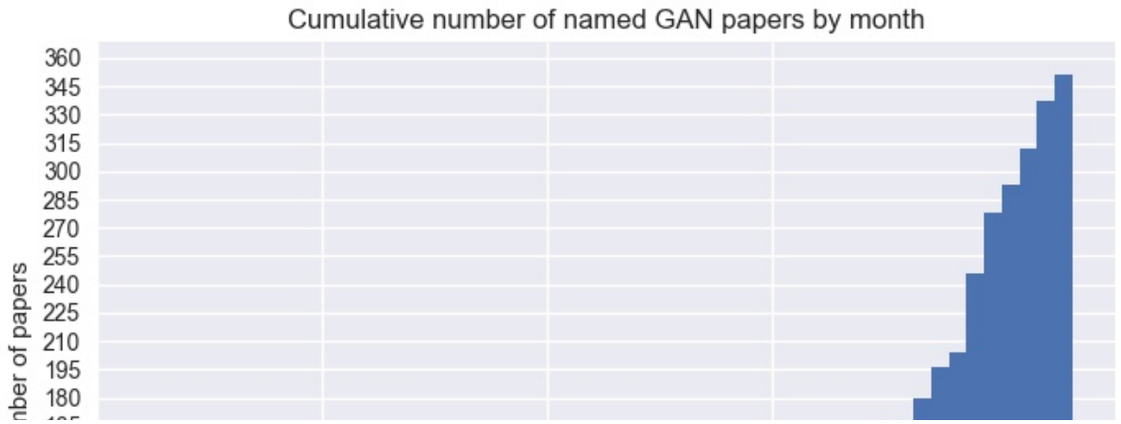
- Generative Adversarial Networks
- Moment Matching Networks

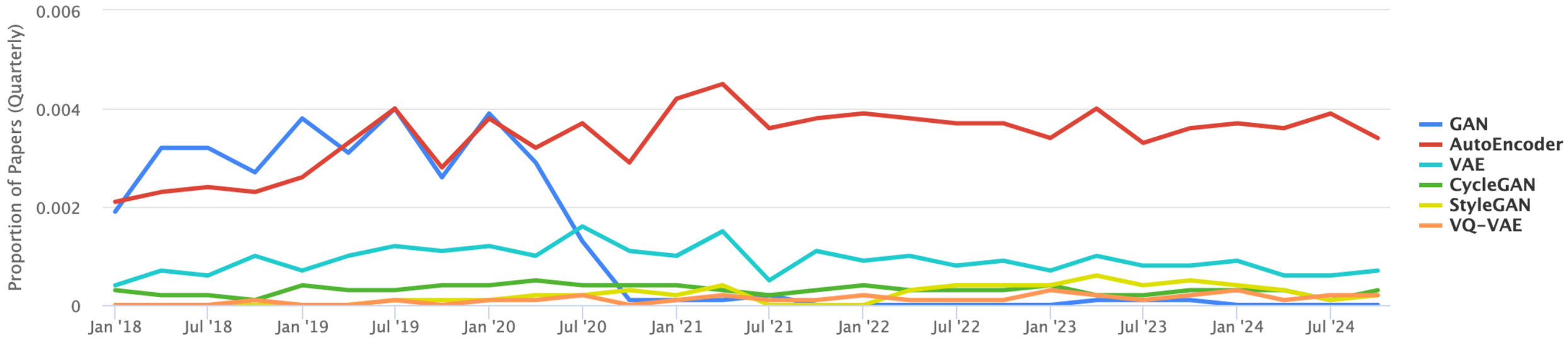
Implicit Density

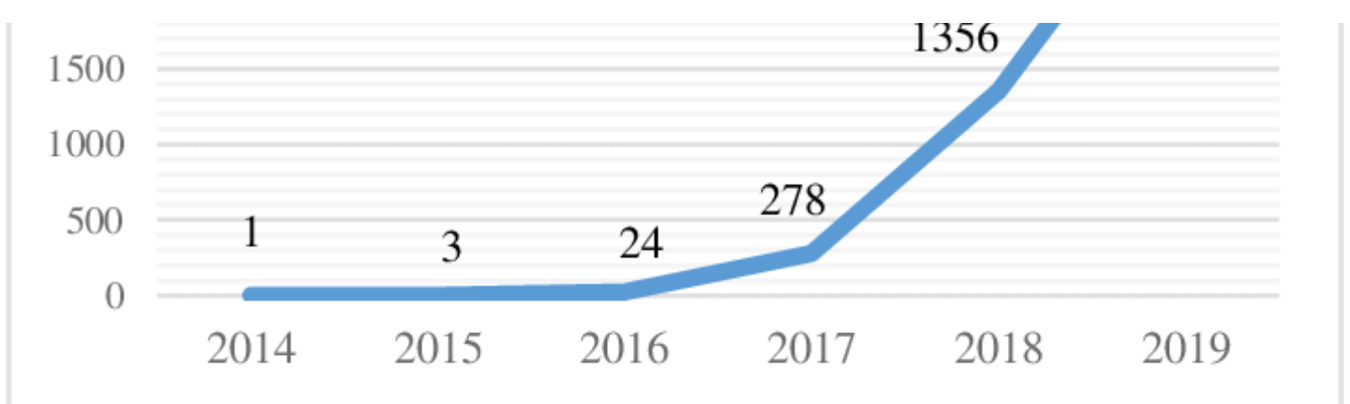

Cumulative number of named GAN papers by month
Total number of papers
360
345
330
315
300
285
270
255
240
225
210
195
180
165
150
135
120
105
90
75
60
45
30
15
0
2014
2015
2016
2017
Year


3500
3000
2500
2000
1500
1000
500
0
1
3
24
278
1356
3006
2014
2015
2016
2017
2018
2019

uc3m

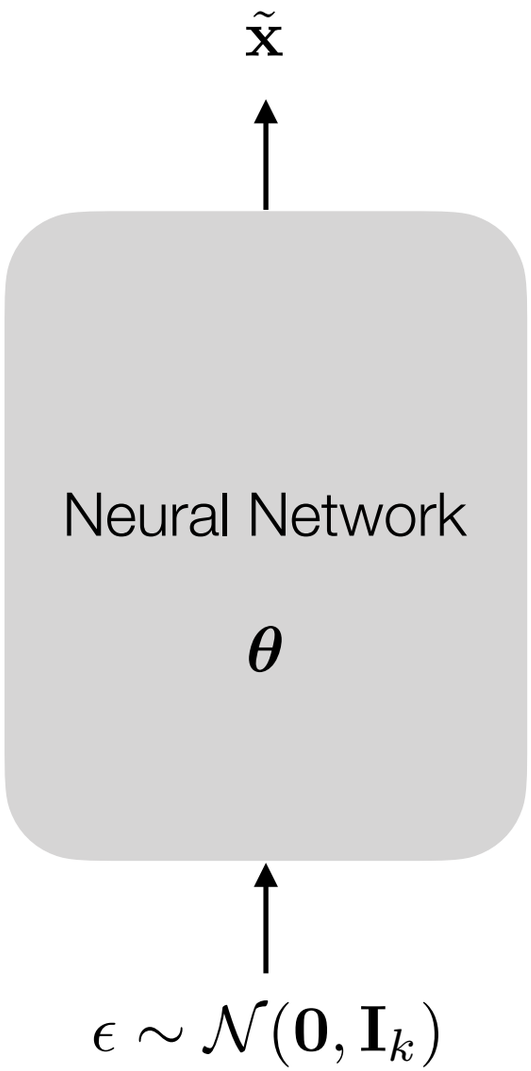

Cumulative number of named GAN papers by month
360
345
330
315
300
285
270
255
240
225
210
195
180


Proportion of Papers (Quarterly)
0.006
0.004
0.002
0
Jan '18
Jul '18
Jan '19
Jul '19
Jan '20
Jul '20
Jan '21
Jul '21
Jan '22
Jul '22
Jan '23
Jul '23
Jan '24
Jul '24
GAN
AutoEncoder
VAE
CycleGAN
StyleGAN
VQ-VAE


1500
1000
500
0
1
3
24
278
1356
2014
2015
2016
2017
2018
2019

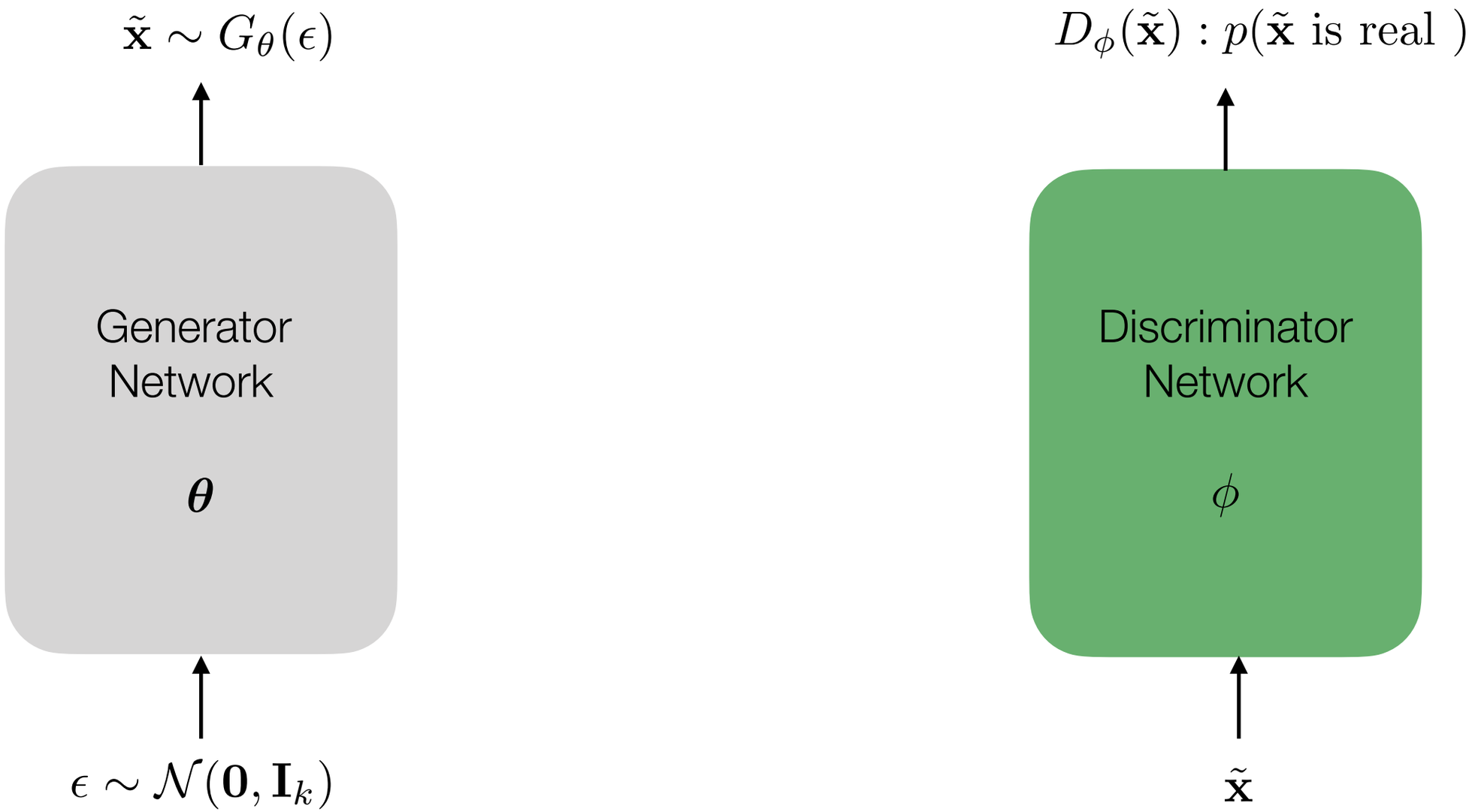
uc3m

Implicit density estimation

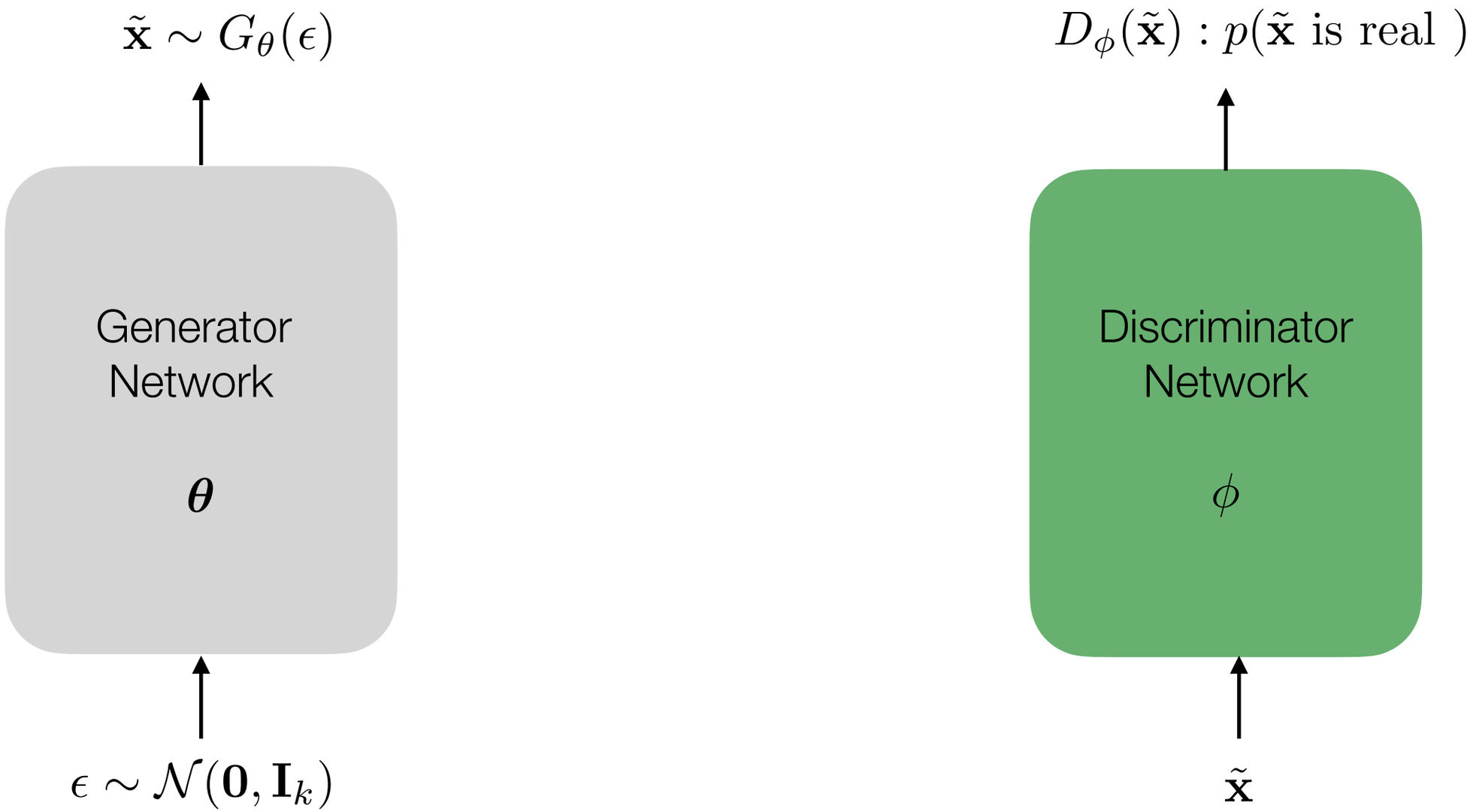


- **Unknown density** at the output
- **Sampling** is easy!

Generative Adversarial Network (Goodfellow 2014)



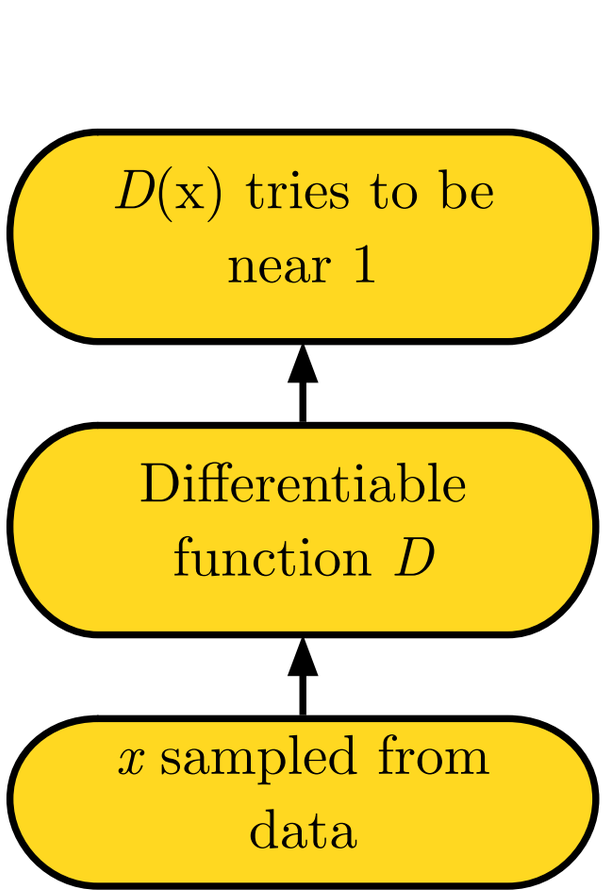
Generative Adversarial Network (Goodfellow 2014)

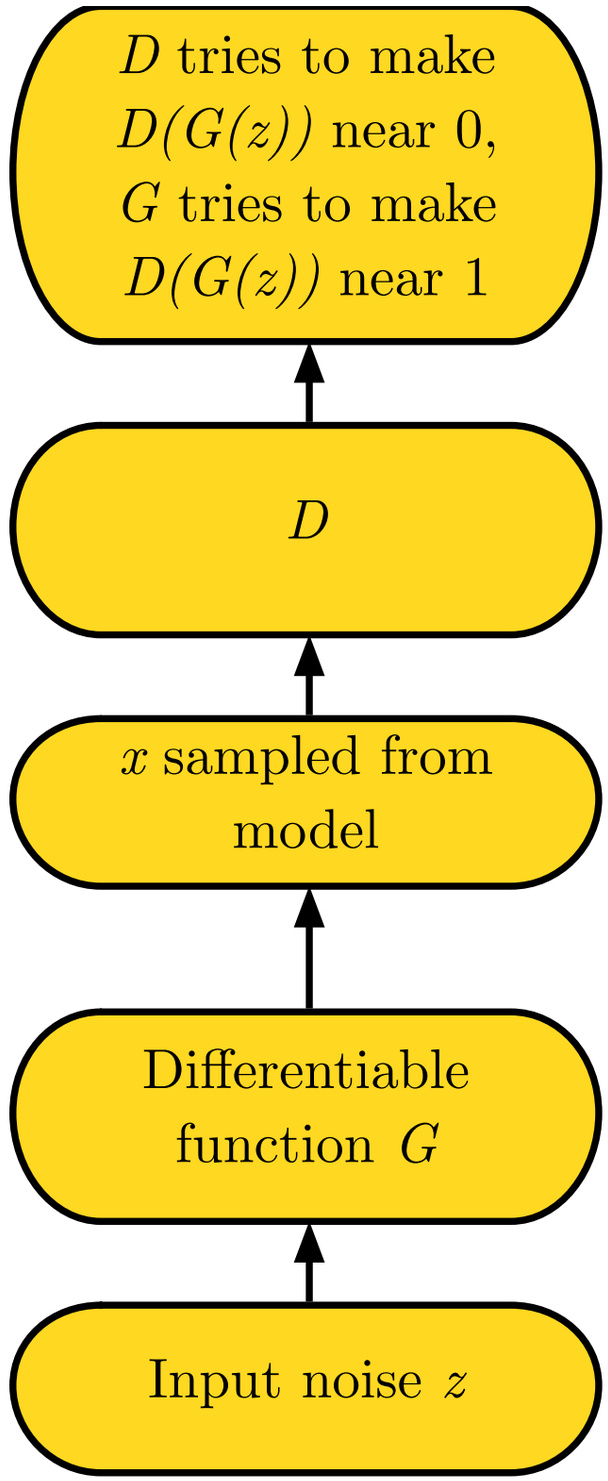


Binary Classifier with a binary cross entropy loss

$$D_\phi(\tilde{\mathbf{x}}) \in (0, 1)$$

Generative Adversarial Network (Goodfellow 2014)





Generative Adversarial Network (Goodfellow 2014)

- Use SGD-like algorithm of choice (Adam) on two mini batches simultaneously:
    - A minibatch of training examples
    - A minibatch of generated samples
- Minimax Game:

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

$$J^{(G)} = -J^{(D)}$$

- Equilibrium is a saddle point of the discriminator loss
- Generator minimizes the log-probability of the discriminator being correct

Additional reading: LINK

Generative Adversarial Network (Goodfellow 2014)

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.

Generative Adversarial Network (Goodfellow 2014)

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

obtained by $\dfrac{\delta}{\delta D(\boldsymbol{x})} J^{(D)} = 0$

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))]$$

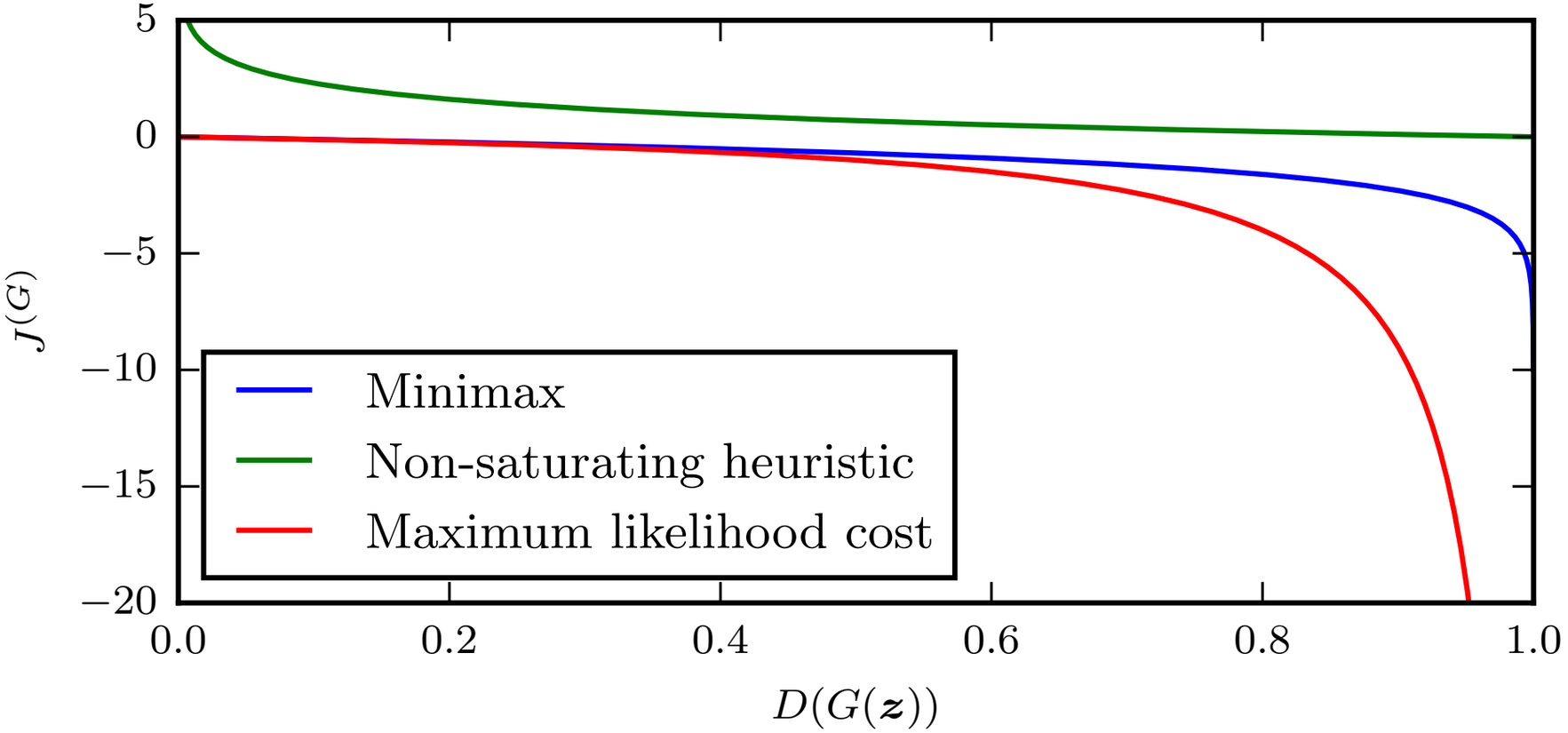
then p_g converges to p_{data}

Generative Adversarial Network (Goodfellow 2014)

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right) \qquad \textbf{Minimax}$$

$$J^{(G)} = -J^{(D)}$$

- When the discriminator successfully rejects generator samples with high confidence, **the generator's gradient vanishes**

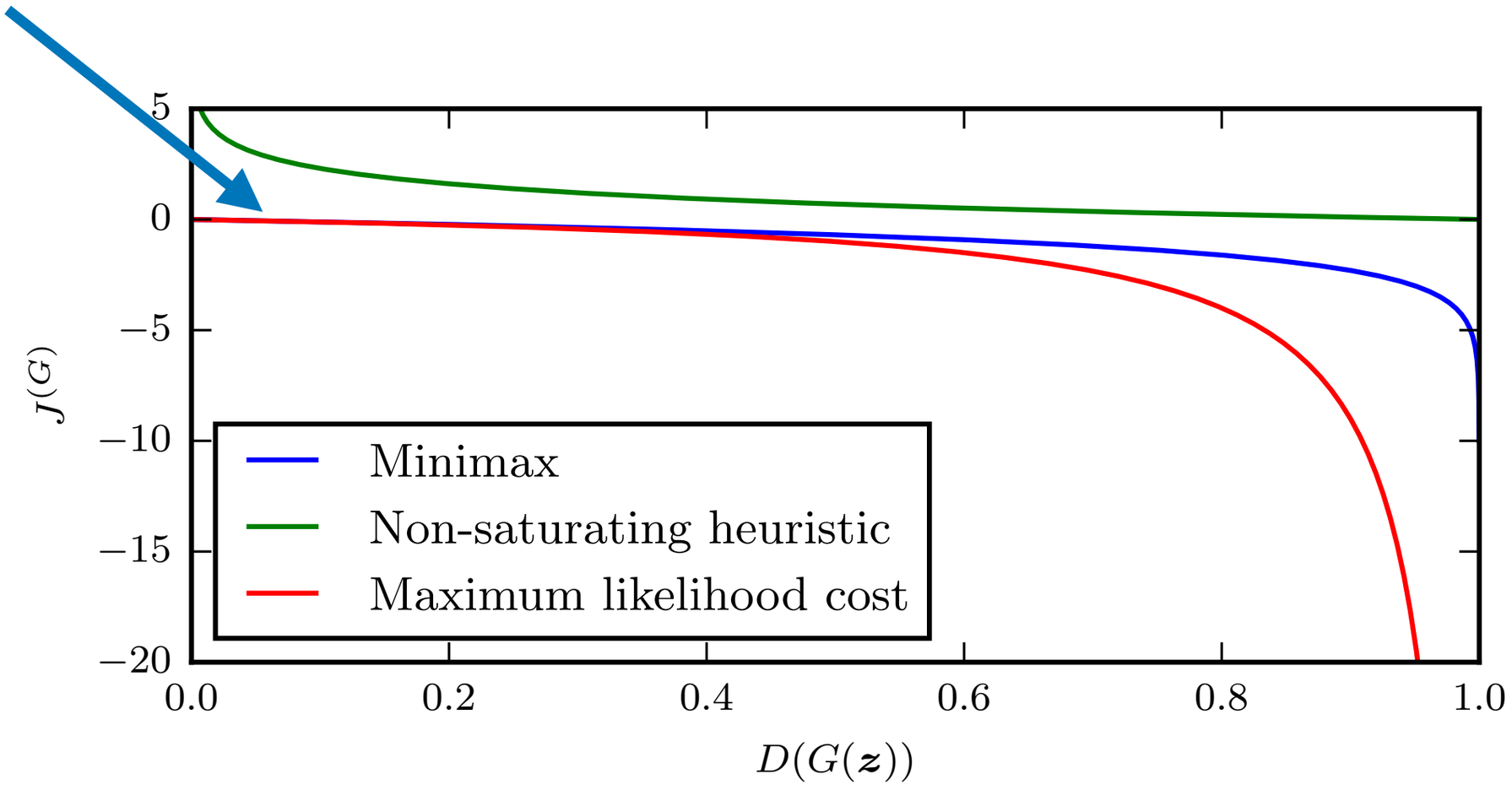


Generative Adversarial Network (Goodfellow 2014)

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right) \qquad \textbf{Minimax}$$

$$J^{(G)} = -J^{(D)}$$

- When the discriminator successfully rejects generator samples with high confidence, **the generator's gradient vanishes**



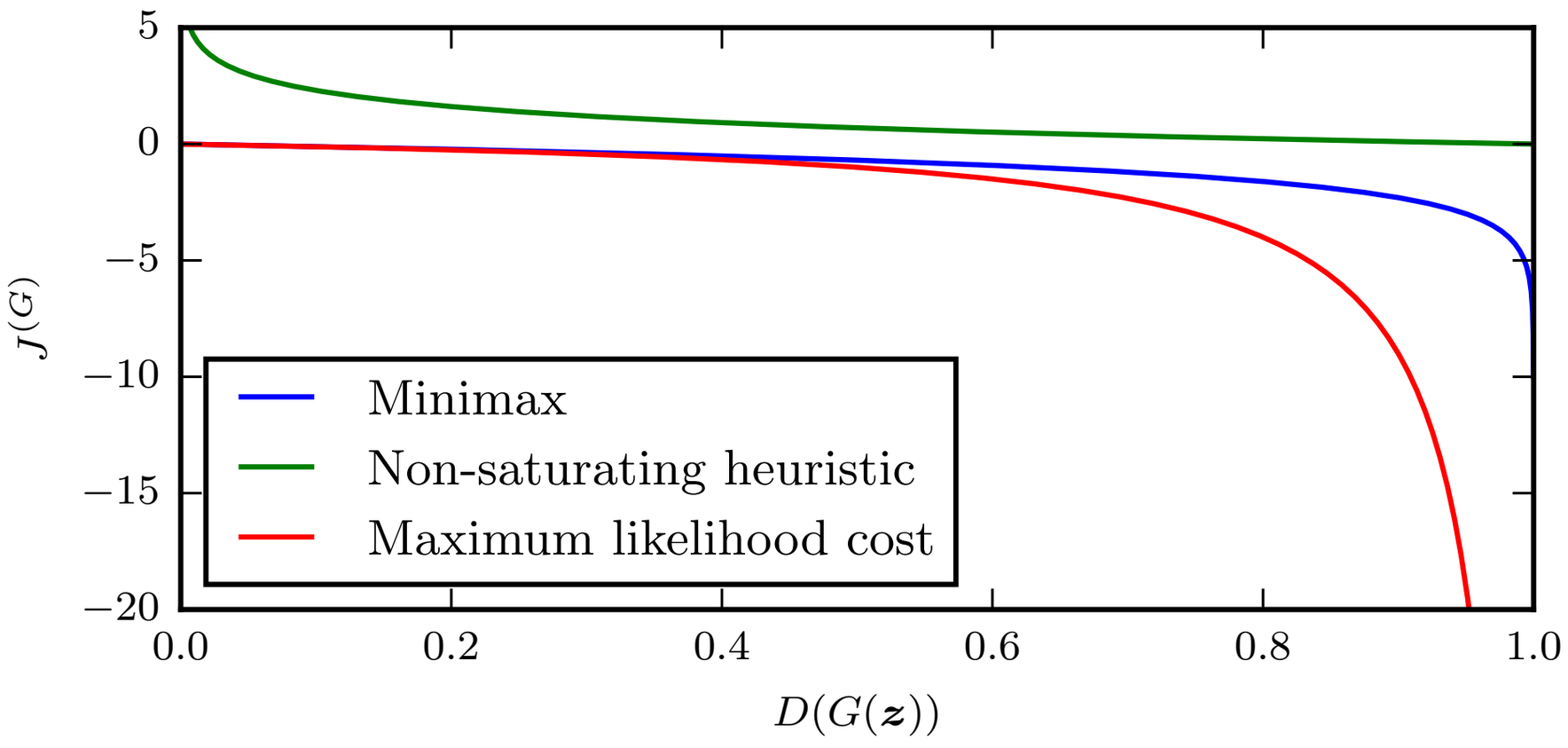
Generative Adversarial Network (Goodfellow 2014)

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

Non-saturating heuristic

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z}))$$

- The generator maximizes the log probability of the discriminator **being mistaken**
- The sole motivation for this version of the game is to **ensure that each player has a strong gradient when that player is "losing" the game**.



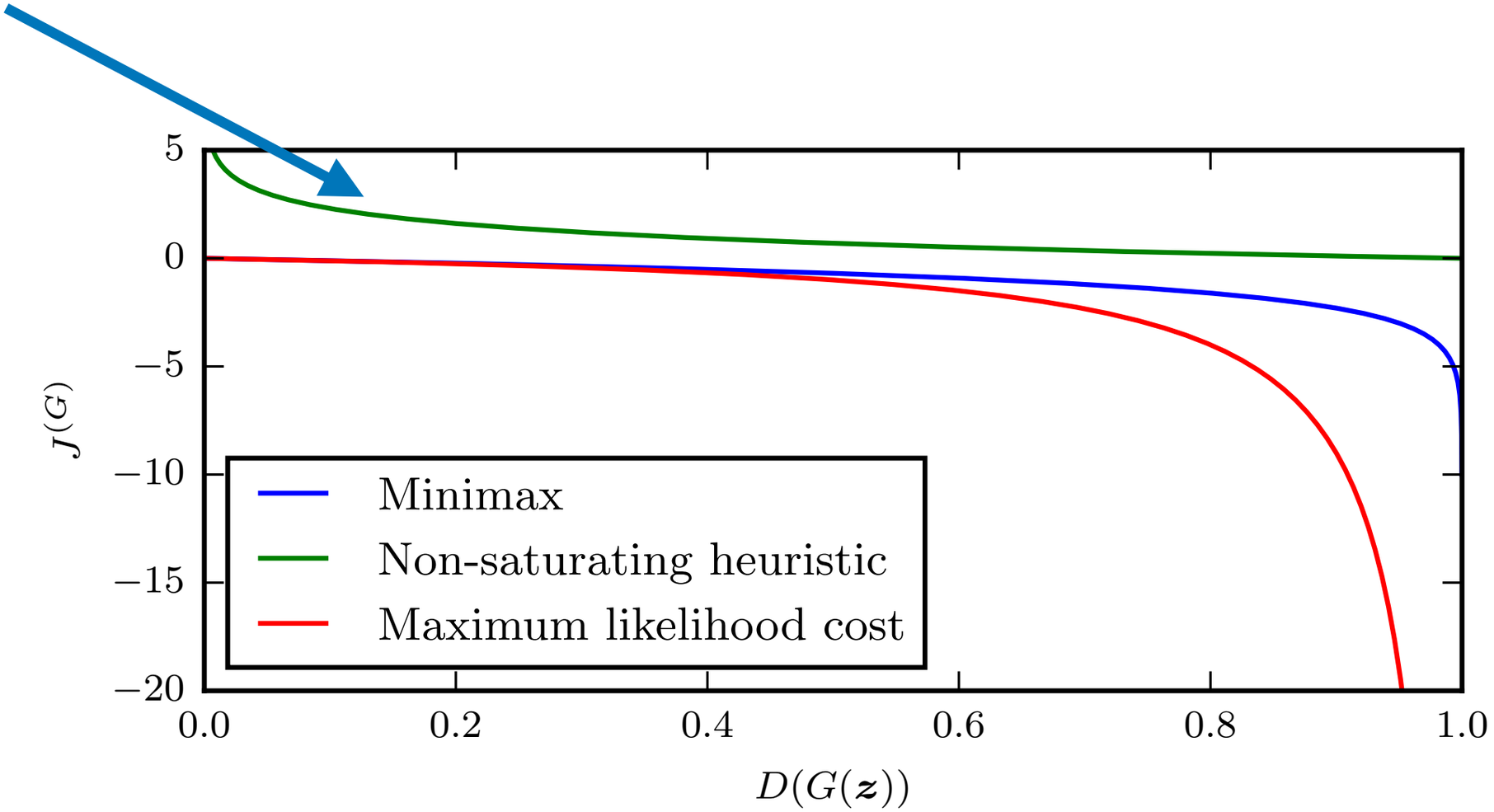
Generative Adversarial Network (Goodfellow 2014)

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

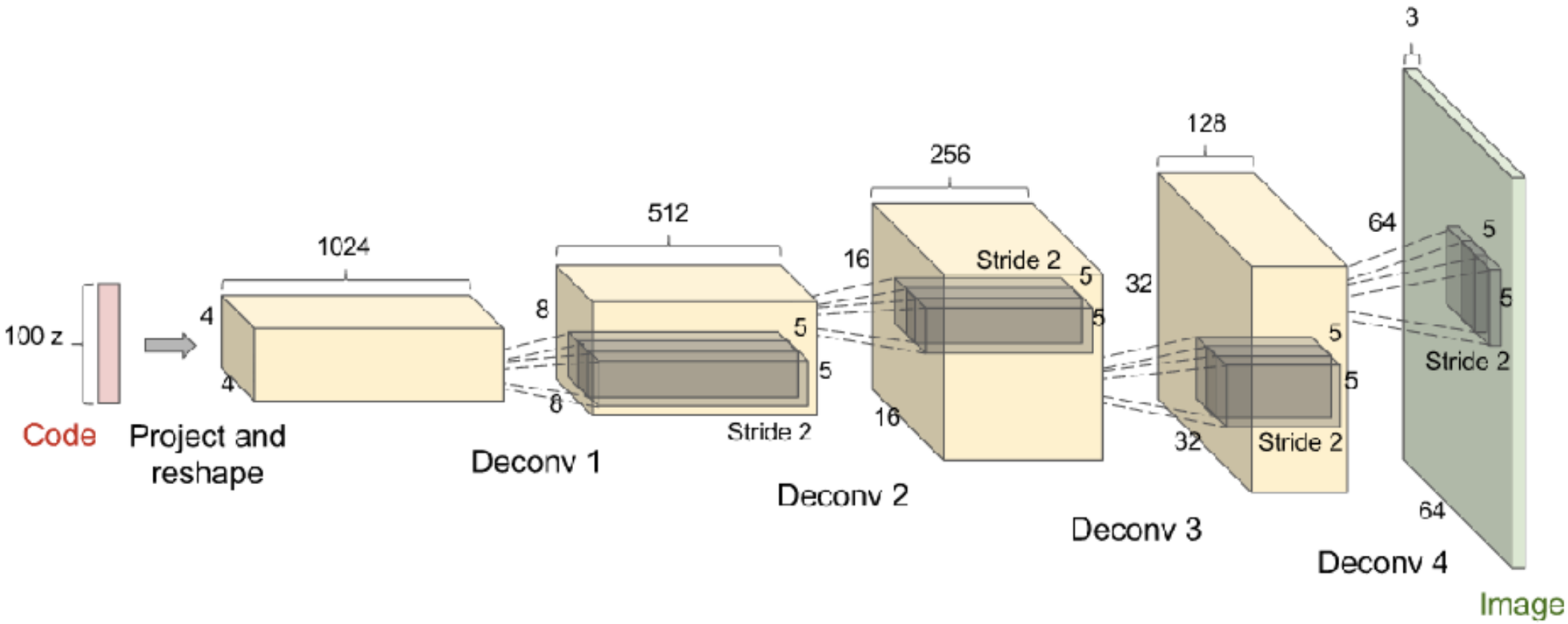
Non-saturating heuristic

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z}))$$

- The generator maximizes the log probability of the discriminator **being mistaken**
- The sole motivation for this version of the game is to **ensure that each player has a strong gradient when that player is "losing" the game**.

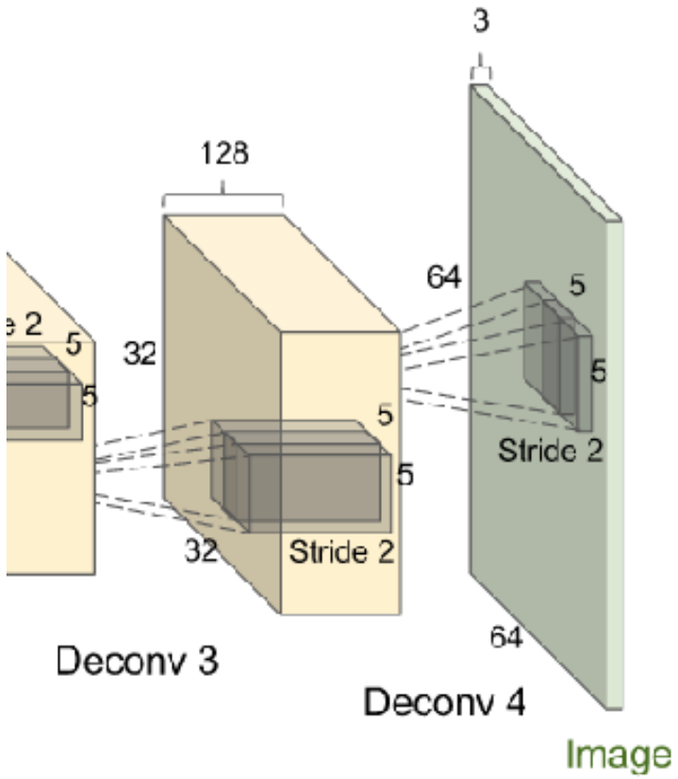


DCGAN Architecture (Radford 2015)



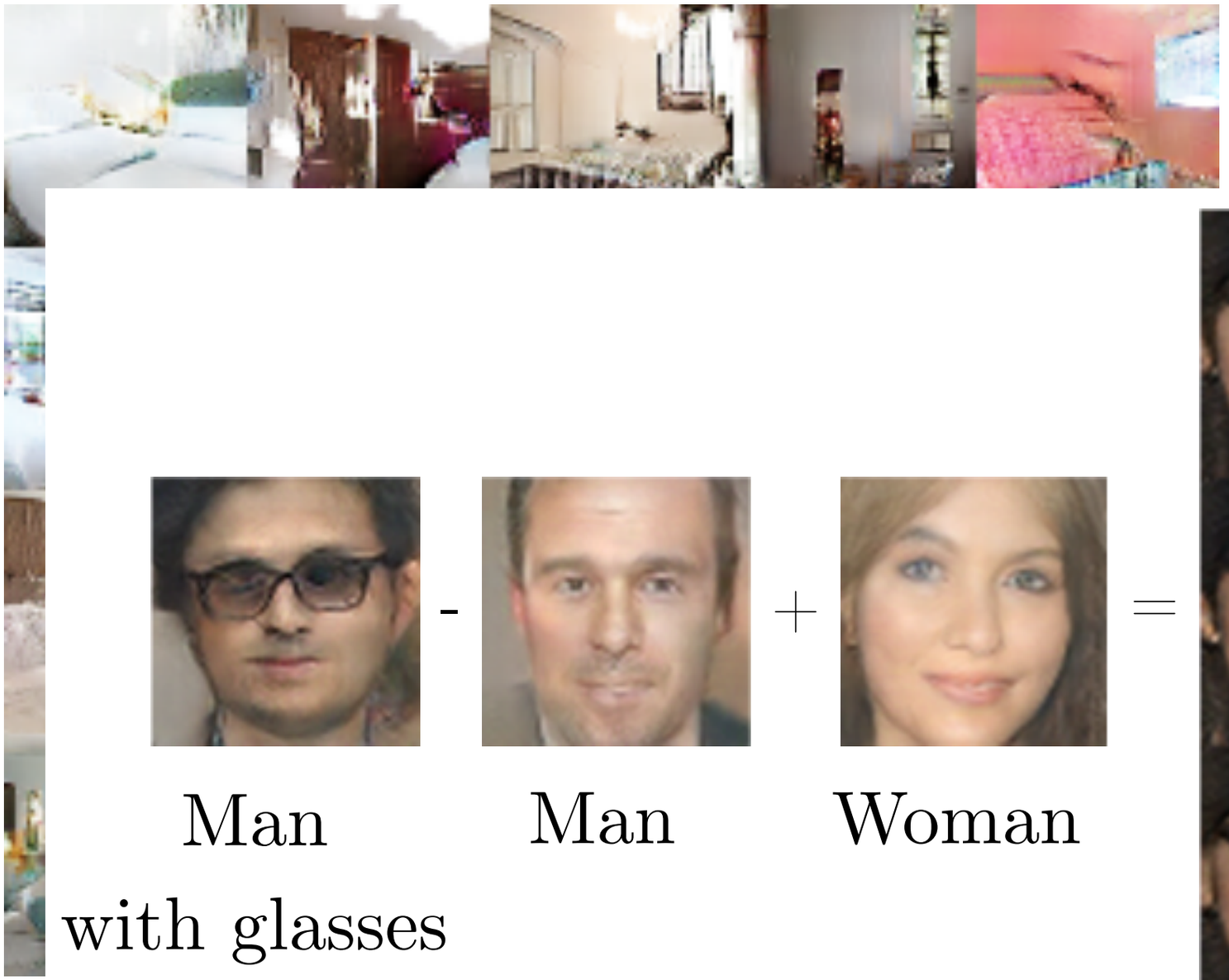
(Radford et al 2015)

DCGAN Architecture (Radford 2015)



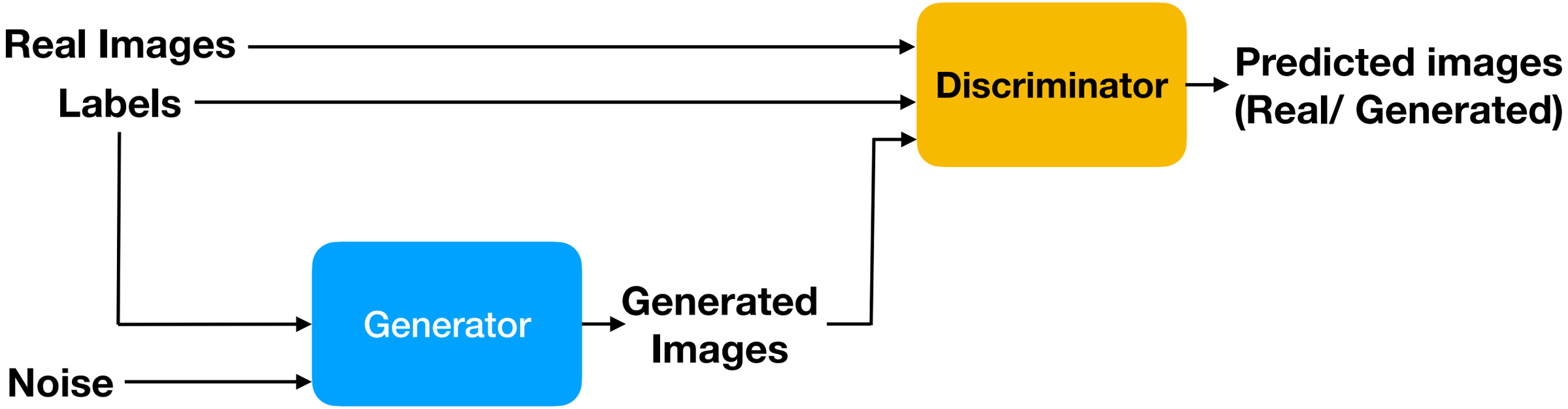
(Radford et al 2015)

DCGAN Architecture (Radford 2015)

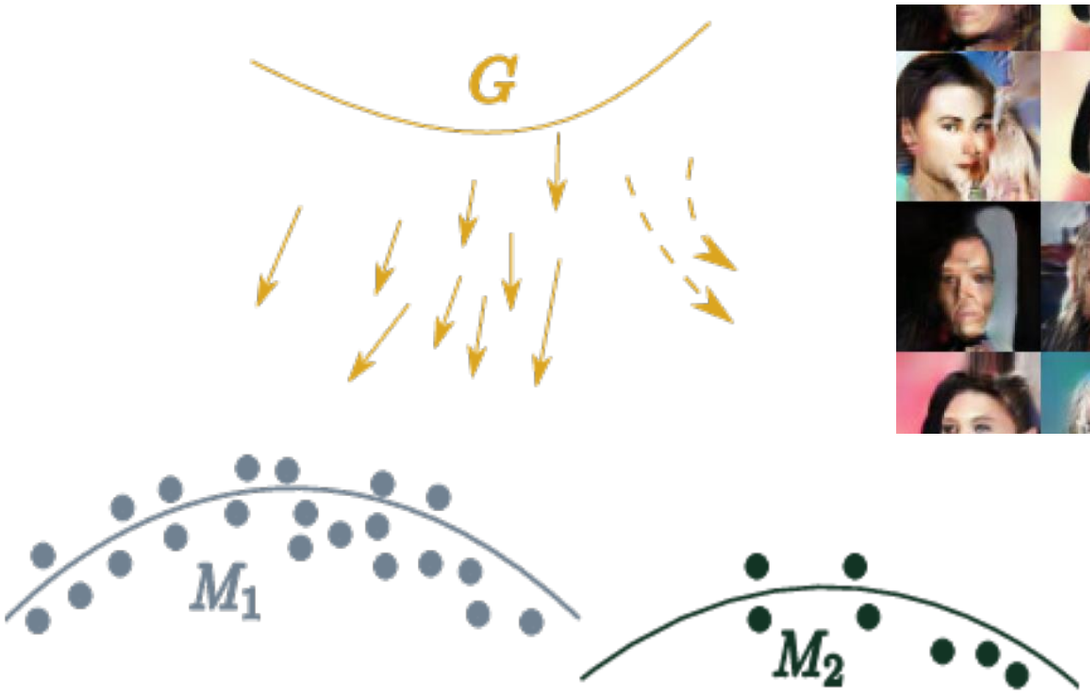


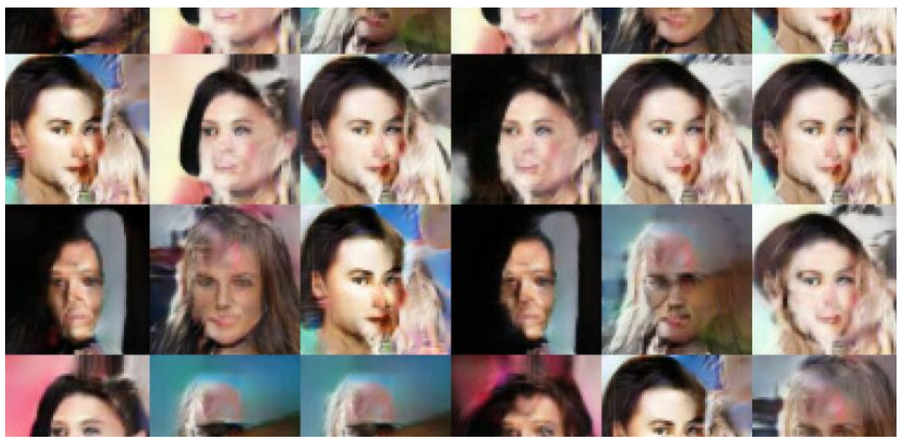
Contitional GAN (CGAN)

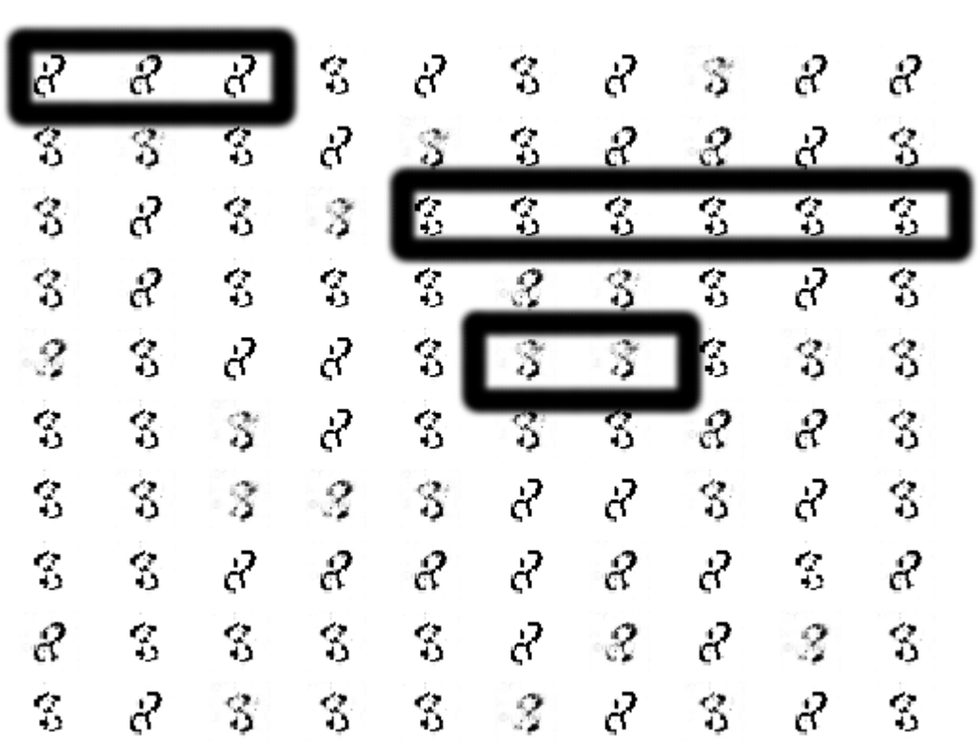
- In a **standard GAN** we have **no control on the class of output** we are producing
- **Conditional GAN: Add a y-label** to generator and discriminator
- Generator and discriminator **trained for the specific y-label**
- In **generation**, we can **pass the y-label** to generator to produce a class of ouput

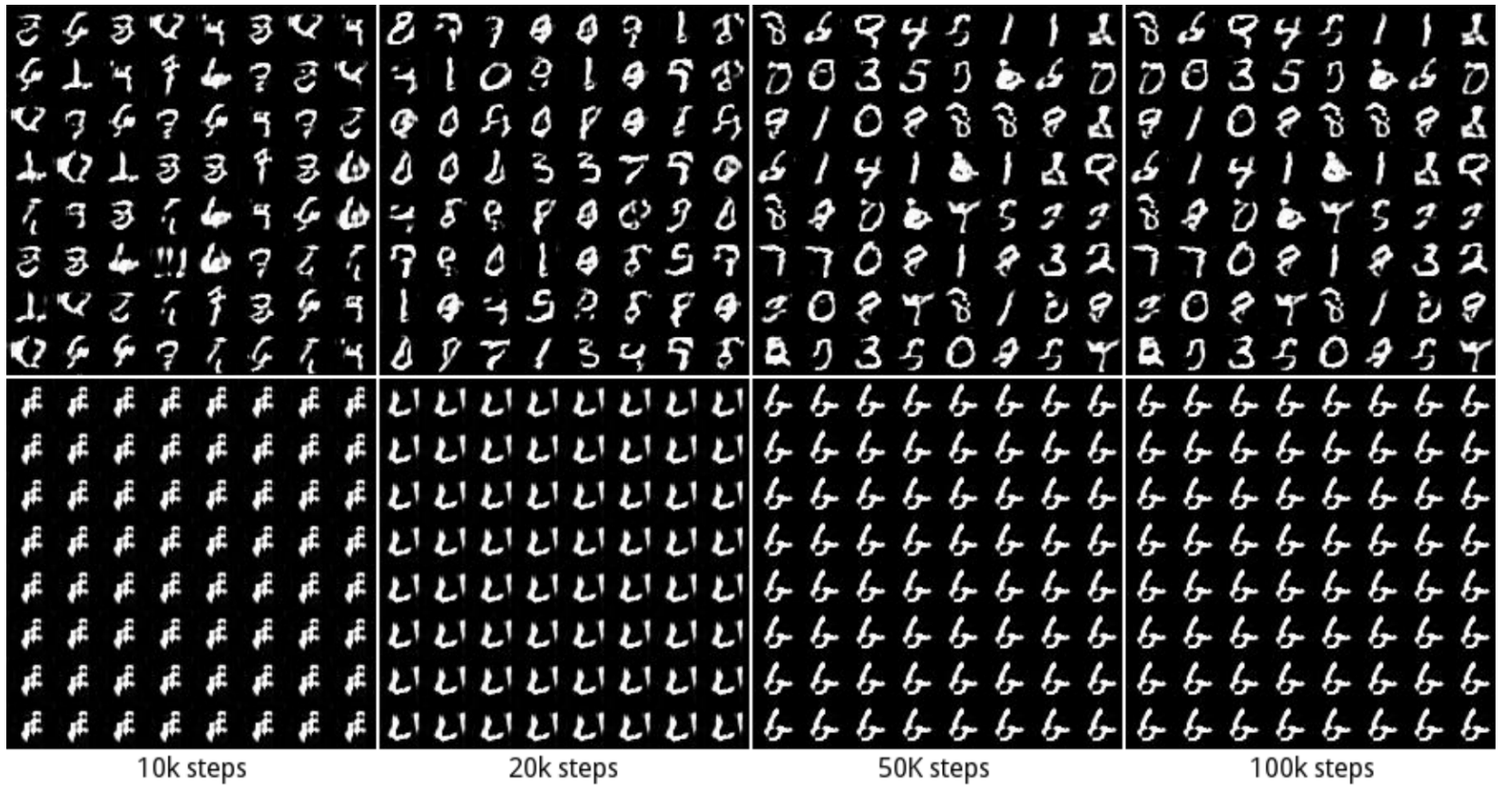


A problem of GANs: Mode Collapse

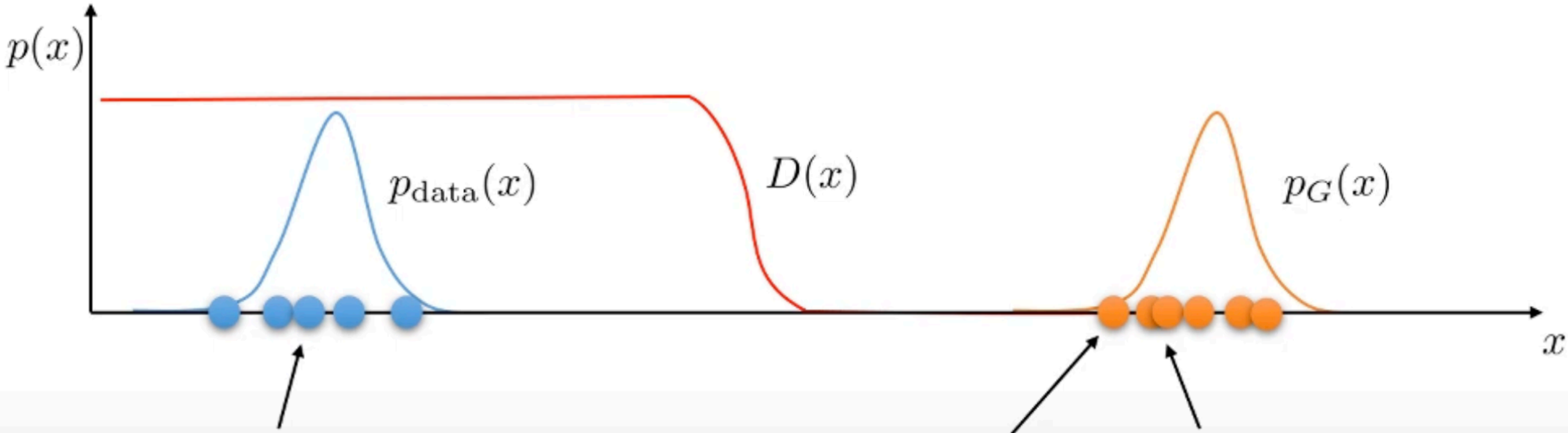








Why is training GANs hard?

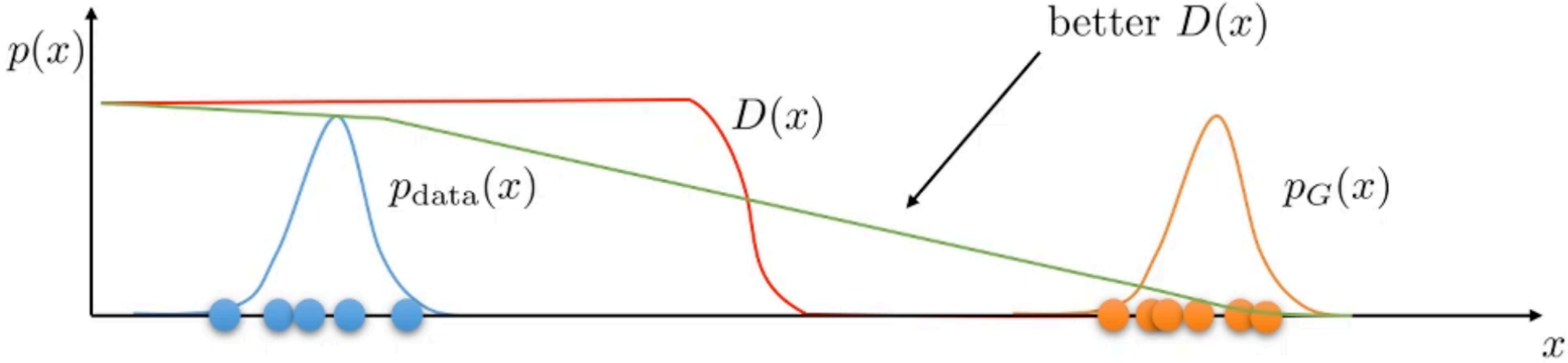


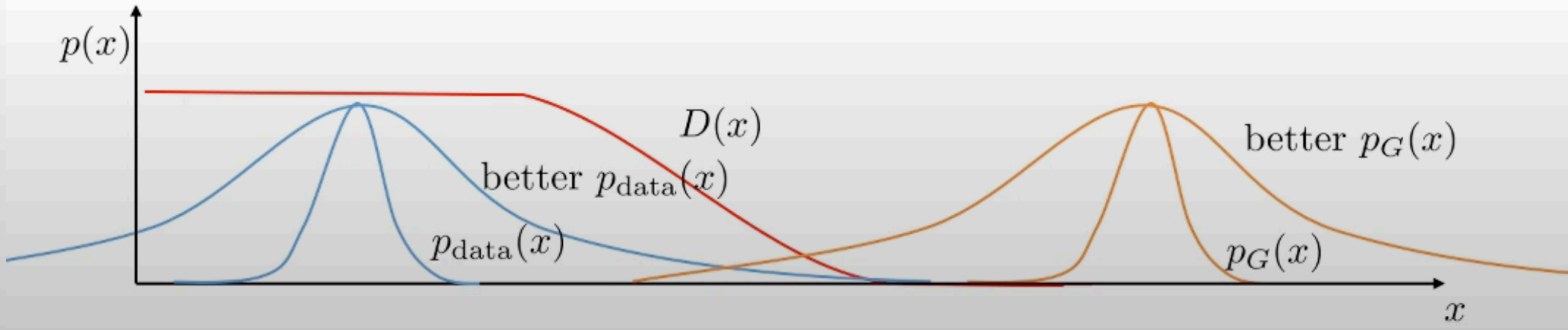
what is the generator gradient here?

$$\min_{\theta} \max_{\phi} V(\theta, \phi) = E_{x \sim p_{\text{data}}(x)}[\log D_\phi(x)] + E_{z \sim p(z)}[\log(1 - D_\phi(G_\theta(z)))]$$

all of these values are basically the same

Why is training GANs hard?



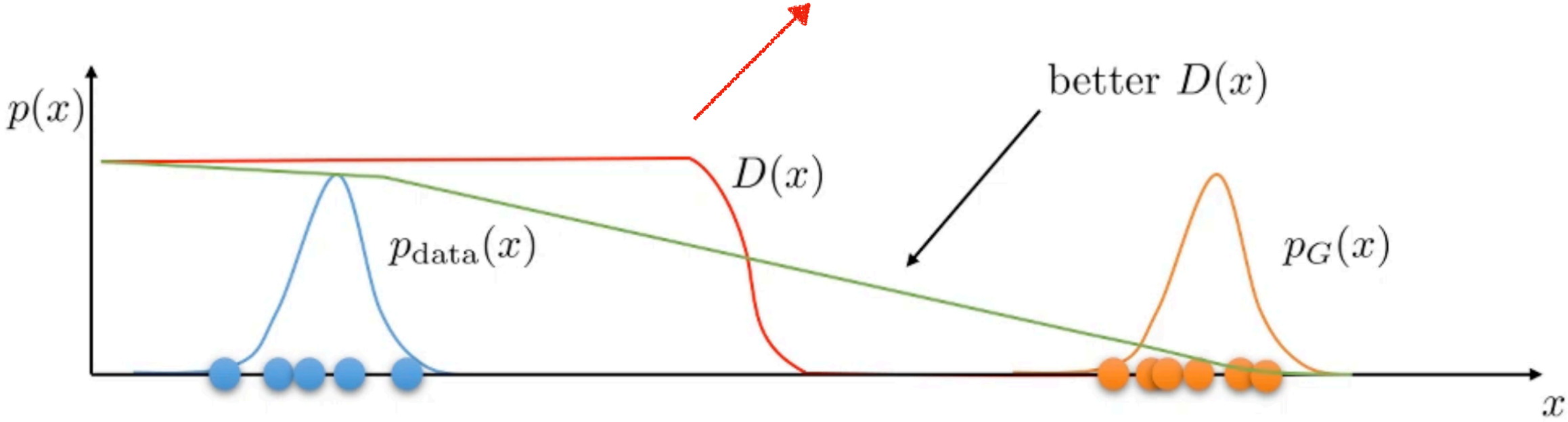


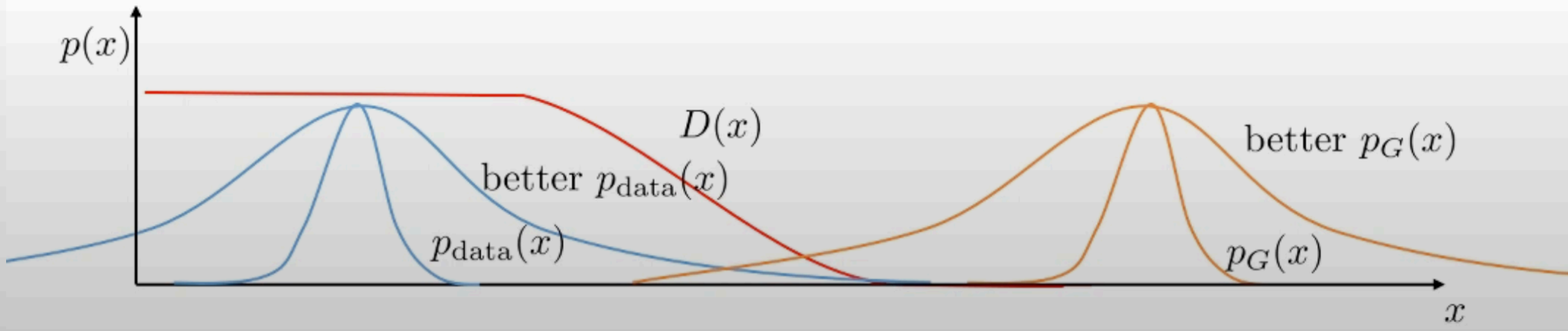
Why is training GANs hard?

Least-squares GAN (LSGAN): discriminator outputs real-valued number

Wasserstein GAN (WGAN): discriminator is constrained to be Lipschitz-continuous

Gradient Penalty, spectral norm, etc



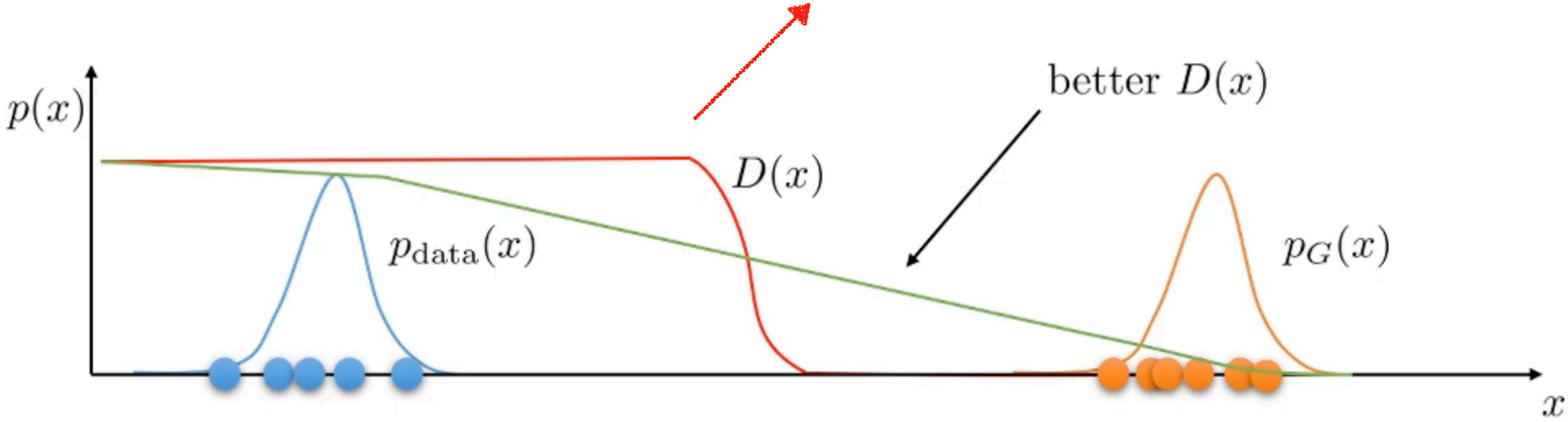


Why is training GANs hard?

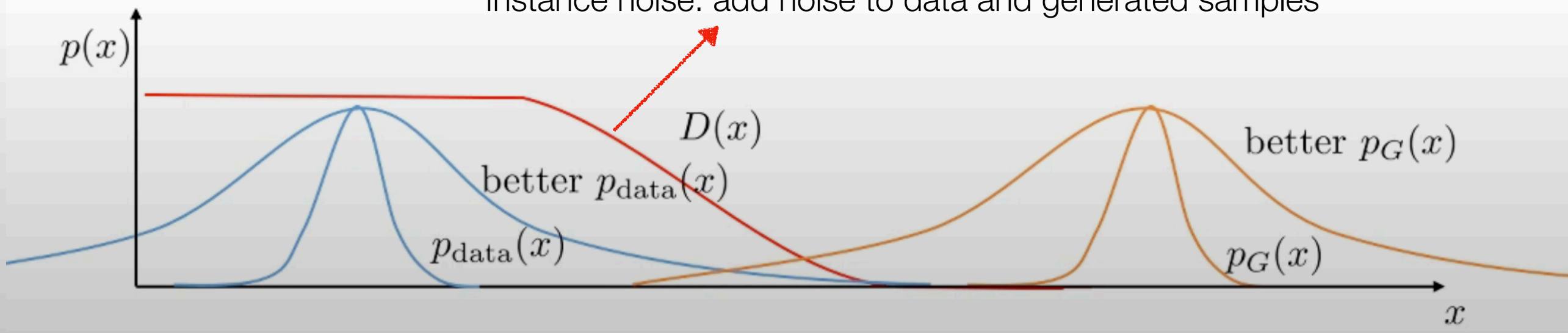
Least-squares GAN (LSGAN): discriminator outputs real-valued number

Wasserstein GAN (WGAN): discriminator is constrained to be Lipschitz-continuous

Gradient Penalty, spectral norm, etc



Instance noise: add noise to data and generated samples



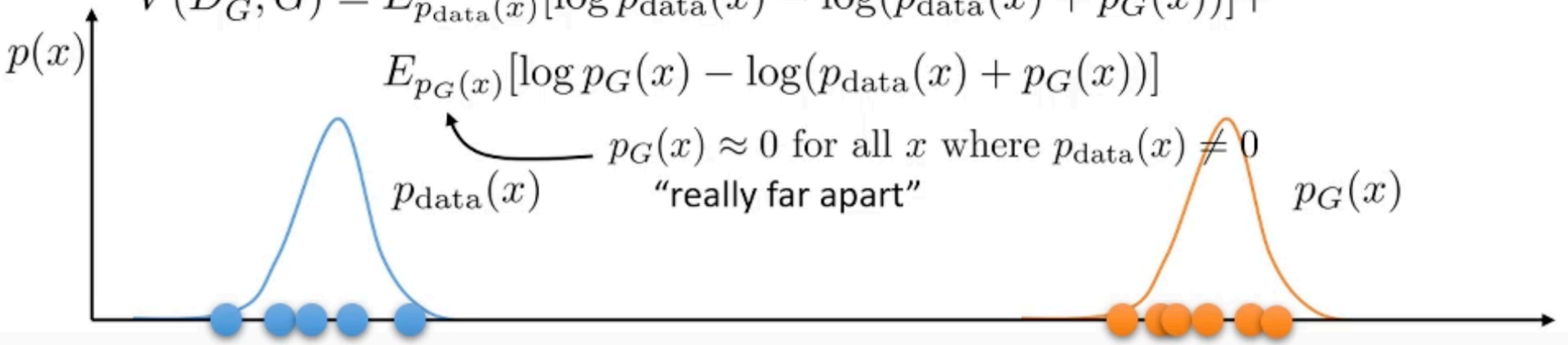
Wasserstein GANs (Arjovsky 2017)

High level intuition: the JS divergence used in classic GAN doesn't account for "distance"

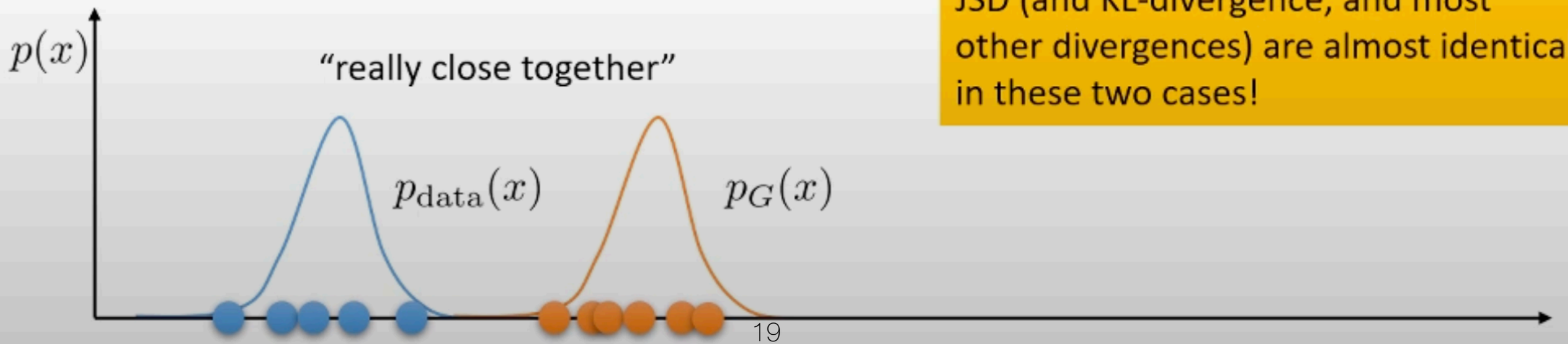
$$V(D_G^\star, G) = E_{p_{\text{data}}(x)}[\log p_{\text{data}}(x) - \log(p_{\text{data}}(x) + p_G(x))] +$$
$$E_{p_G(x)}[\log p_G(x) - \log(p_{\text{data}}(x) + p_G(x))]$$

$p_{\text{data}}(x) \approx 0$ for all x where $p_G(x) \neq 0$

$p_G(x) \approx 0$ for all x where $p_{\text{data}}(x) \neq 0$



JSD (and KL-divergence, and most other divergences) are almost identical in these two cases!



Wasserstein GANs (Arjovsky 2017)

- **Intuition**: Distance used to measure the cost of having to transport mass in Euclidean space
  - In other words, how far do I have to go to “transport” one distribution to another

- Improves the stability when training the model and provides a loss function that correlates with the quality of generated images.

- Wasserstein distance between the real data distribution P_r and the generator’s one P_θ

$$W(P_r, P_\theta) = \sup_{\|f\| \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_f}[f(x)]$$

where the supremum is taken over all **1-Lipschitz functions.**

1-Lipschitz function: $|f(x) - f(y)| \leq |x - y|$

- functions can’t be too sharp -> guarantees smoothness of f

Wasserstein GANs (Arjovsky 2017)

- **Intuition**: Distance used to measure the cost of having to transport mass in Euclidean space
  - In other words, how far do I have to go to “transport” one distribution to another

- Improves the stability when training the model and provides a loss function that correlates with the quality of generated images.

- Wasserstein distance between the real data distribution P_r and the generator’s one P_θ

$$W(P_r, P_\theta) = \sup_{\|f\| \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_f}[f(x)]$$

where the supremum is taken over all **1-Lipschitz functions.**

1-Lipschitz function: $|f(x) - f(y)| \leq |x - y|$

- functions can’t be too sharp -> guarantees smoothness of f

- In practice, smoothness is achieved by **weight clipping** ($w \in [-c, c]$)

Wasserstein GANs (Arjovsky 2017)

- **Intuition**: Distance used to measure the cost of having to transport mass in Euclidean space
  - In other words, how far do I have to go to “transport” one distribution to another

- Improves the stability when training the model and provides a loss function that correlates with the quality of generated images.

- Wasserstein distance between the real data distribution P_r and the generator’s one P_θ

$$W(P_r, P_\theta) = \sup_{\|f\| \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_f}[f(x)]$$

where the supremum is taken over all **1-Lipschitz functions.**

1-Lipschitz function: $|f(x) - f(y)| \leq |x - y|$

- functions can’t be too sharp -> guarantees smoothness of f

- In practice, smoothness is achieved by **weight clipping** ($w \in [-c, c]$)

Wasserstein GANs (Arjovsky 2017)

- **Intuition**: Distance used to measure the cost of having to transport mass in Euclidean space
  - In other words, how far do I have to go to “transport” one distribution to another

- Improves the stability when training the model and provides a loss function that correlates with the quality of generated images.

- Wasserstein distance between the real data distribution P_r and the generator’s one P_θ

$$W(P_r, P_\theta) = \sup_{\|f\| \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_f}[f(x)]$$

where the supremum is taken over all **1-Lipschitz functions.**

Not a probability!!!

1-Lipschitz function: $|f(x) - f(y)| \leq |x - y|$

- functions can’t be too sharp -> guarantees smoothness of f

- In practice, smoothness is achieved by **weight clipping** ($w \in [-c, c]$)

Wasserstein GANs (Arjovsky 2017)

$$W(p_{\text{data}}, p_G) = \sup_{||f||_L \leq 1} E_{p_{\text{data}}}[f(x)] - E_{p_G(x)}[f(x)]$$

1. update f_θ using gradient of $E_{x \sim p_{\text{data}}}[f_\theta(x)] - E_{z \sim p(z)}[f_\theta(G(x))]$
2. clip all weight matrices in θ to $[-c, c]$
3. update generator to *maximize* $E_{z \sim p(z)}[f_\theta(G(z))]$

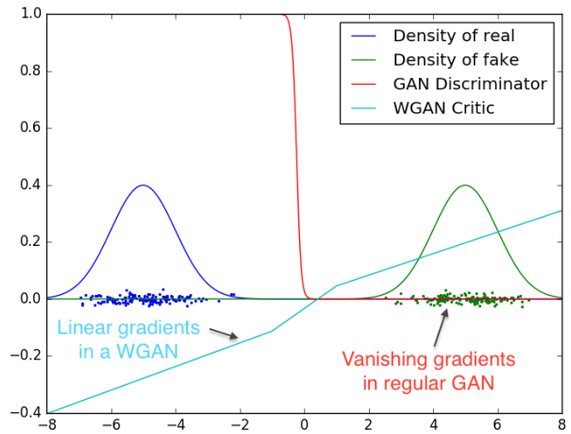
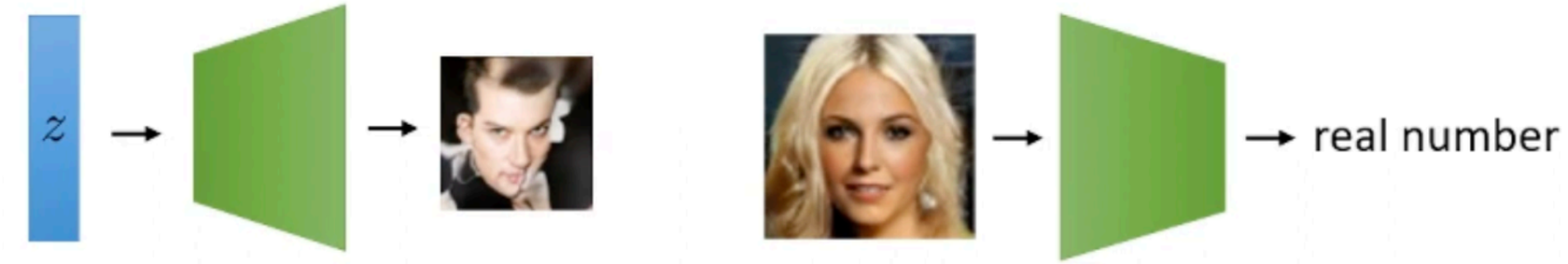


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.



$p(z)$ $G(z)$

"generator"

"discriminator"

discriminator outputs real numbers (not true/false probability)

e.g., $f_\theta(x) = W_2 \text{ReLU}(W_1 x + b_1) + b_2$

discriminator uses *weight clipping*

Arjovsky et al. Wasserstein GAN. 2017.

Wasserstein GANs (Arjovsky 2017)

Discriminator = Critic (just a change of name in that work)

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Improved Wasserstein GANs (Arjovsky 2017)

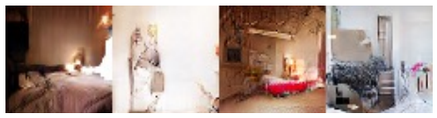
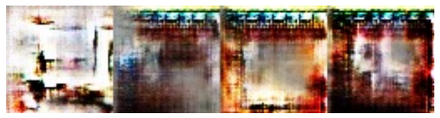
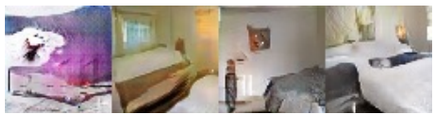
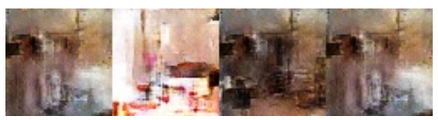
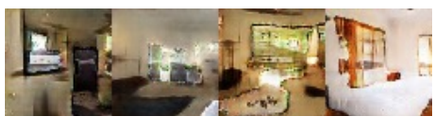
Improved Training of Wasserstein GANs

Ishaan Gulrajani[1]*, **Faruk Ahmed**[1], **Martin Arjovsky**[2], **Vincent Dumoulin**[1], **Aaron Courville**[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow
igul222@gmail.com
{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca
ma4371@nyu.edu

- Discriminator weight clipping can lead to undesired behavior
- A differentiable function is 1-Lipschtiz if and only if it has **gradients with norm at most 1 everywhere**.
- Directly **constrain the gradient norm of the critic's output with respect to its input**.

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}}D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}$$

Improved Wasserstein GANs (Arjovsky 2017)

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)				
G: No BN and a constant number of filters, D: DCGAN				
G: 4-layer 512-dim ReLU MLP, D: DCGAN				
No normalization in either G or D				
Gated multiplicative nonlinearities everywhere in G and D				
$\tanh$ nonlinearities everywhere in G and D				
101-layer ResNet G and D				

Salimans 2016

Improved Techniques for Training GANs

Tim Salimans tim@openai.com
Ian Goodfellow ian@openai.com
Wojciech Zaremba woj@openai.com
Vicki Cheung vicki@openai.com
Alec Radford alec.radford@gmail.com
Xi Chen peter@openai.com

Feature matching addresses the instability of GANs

- The new objective requires the generator to **generate data that matches the statistics of the real data**
- We use the discriminator only to **specify the statistics that we think are worth matching**
- We train the generator to match the expected value of the **features on an intermediate layer of the discriminator**

$$\|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\,\mathbf{f}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\mathbf{f}(G(\boldsymbol{z}))\|_2^2$$

Salimans 2016

Improved Techniques for Training GANs

Tim Salimans tim@openai.com
Ian Goodfellow ian@openai.com
Wojciech Zaremba woj@openai.com
Vicki Cheung vicki@openai.com
Alec Radford alec.radford@gmail.com
Xi Chen peter@openai.com

Minibatch discrimination alleviates mode collapse

- The concept of minibatch discrimination is quite general: any discriminator model that **looks at multiple examples in combination**, rather than in isolation, could potentially help avoid collapse of the generator.

- Basic idea: the feature representation $\mathbf{f}(\boldsymbol{x}_i)$ in a certain layer of the discriminator for any point $\boldsymbol{x}_i$ is augmented with a kernel-based distance w.r.t. other points in the mini-batch

$$\tilde{\mathbf{f}}(\boldsymbol{x}_1) = \left[\mathbf{f}(\boldsymbol{x}_1), \quad k(\mathbf{f}(\boldsymbol{x}_1), \mathbf{f}(\boldsymbol{x}_2)), \quad k(\mathbf{f}(\boldsymbol{x}_1), \mathbf{f}(\boldsymbol{x}_3)), \ldots, k(\mathbf{f}(\boldsymbol{x}_1), \mathbf{f}(\boldsymbol{x}_M))\right]$$

Other GAN variants

Style GAN (2018)

One problem with GANs: Latent space is entangled

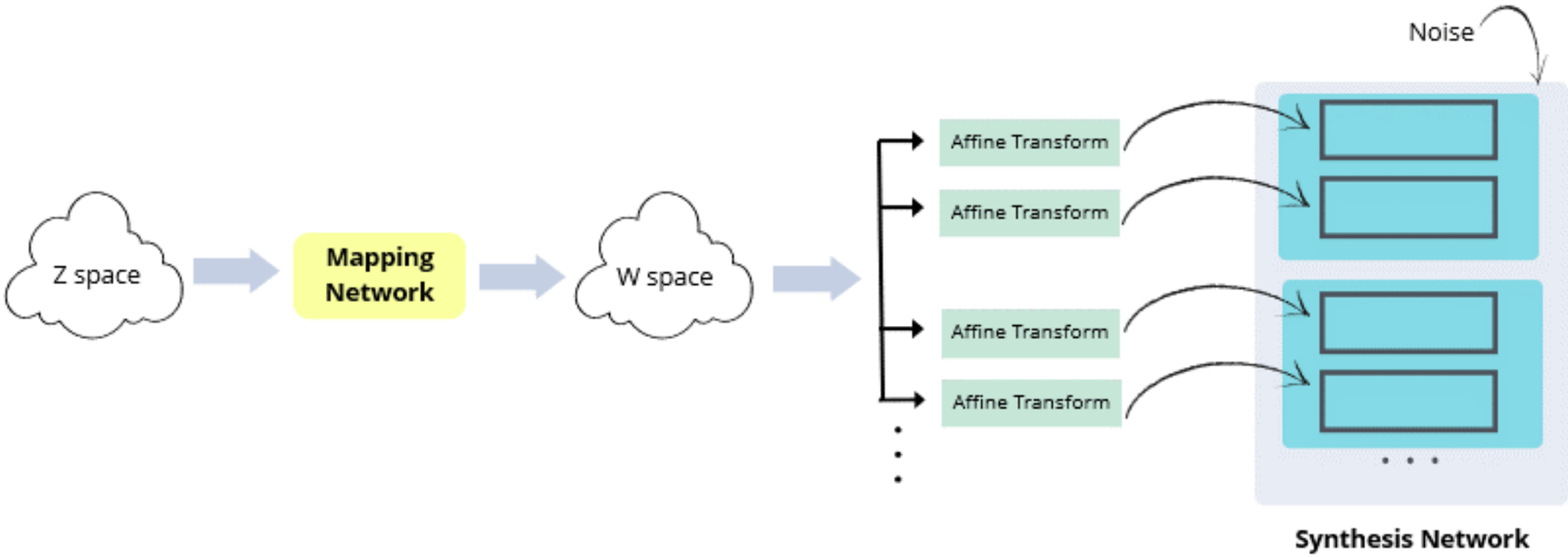
- This means that the N latent dimensions representing an image contains partly several features

For example: What if I ask how would my cat look like with short hair?

Undesired changes due to entanglement: The short hair dimension could contain dog features as well

Style GAN (2018): key contribution

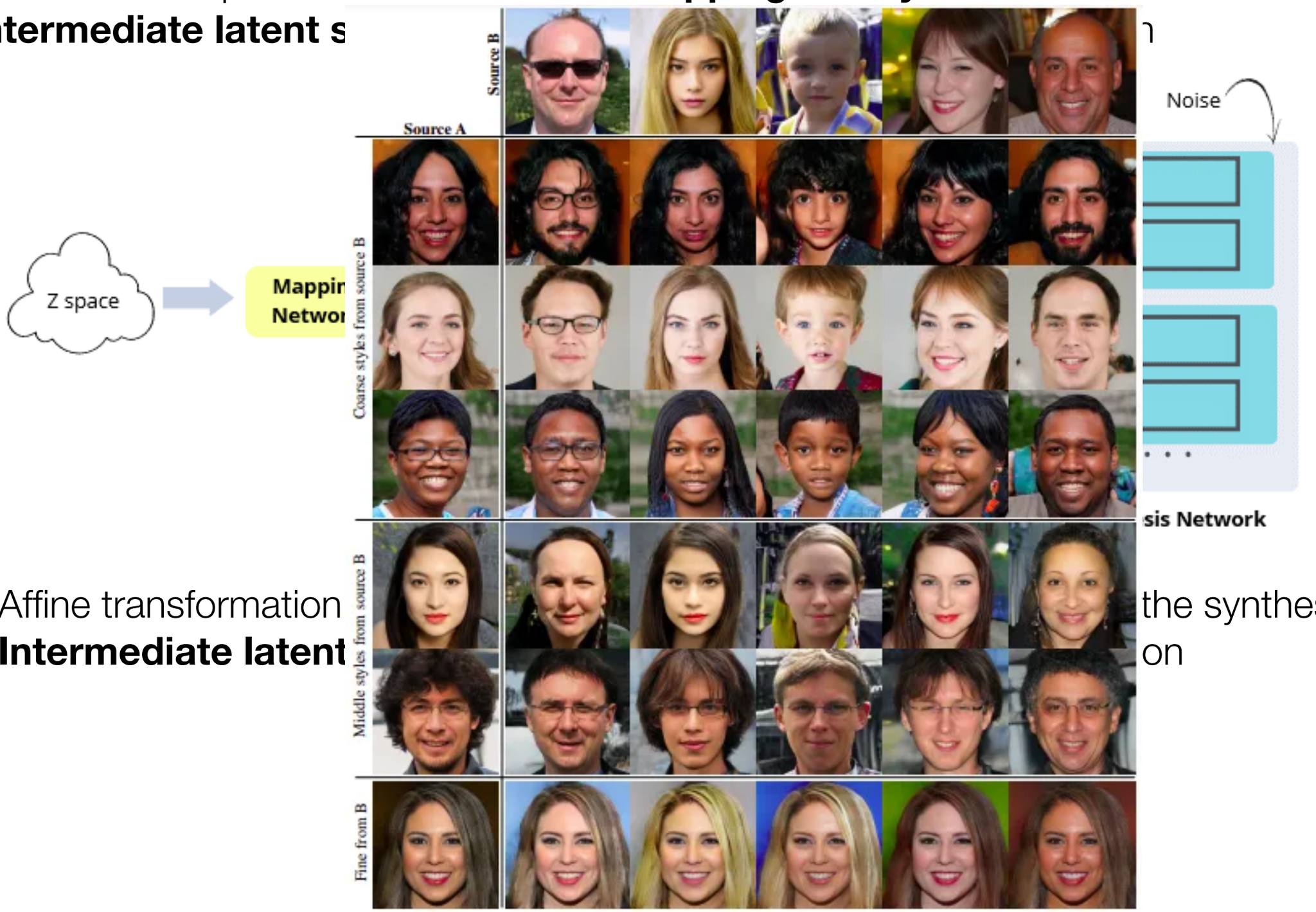
- Generator composed of two networks: **mapping and synthesis**
- **Intermediate latent space (W space)** alongside affine transformation



- Affine transformation: Turns **w** vectors into a “style” that is the fed to the synthesis net
- **Intermediate latent space (W space)** alongside affine transformation

Style GAN (2018): key contribution

- Generator composed of two networks: **mapping and synthesis**
- **Intermediate latent s** ɳ



- Affine transformation the synthesis net
- **Intermediate latent** on

Cycle GAN (2017)

two (conditional) generators:

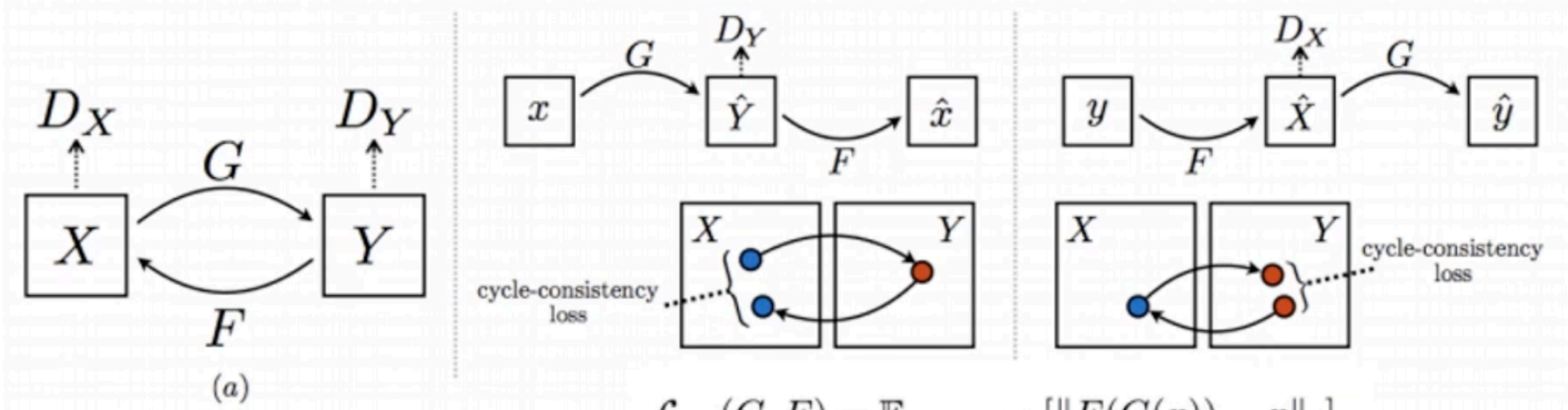
G: turn X into Y (e.g., horse into zebra)

F: turn Y into X (e.g., zebra into horse)

two discriminators:

D_X: is it a realistic horse?

D_Y: is it a realistic zebra?



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

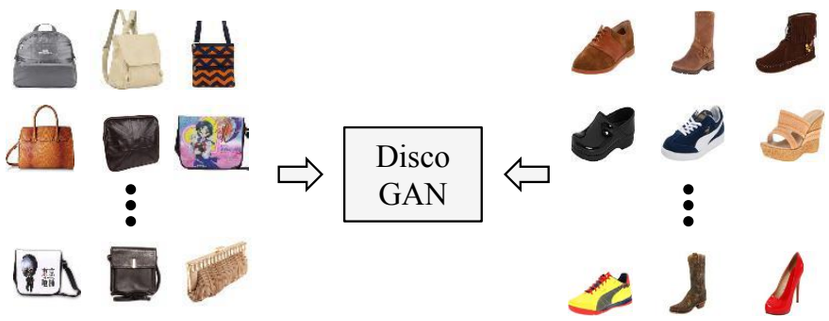
Problem: why should the "translated" zebra look anything like the original horse?

Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). **Unpaired image-to-image translation using cycle-consistent adversarial networks**.

DiscoGAN (2018)

Learning to Discover Cross-Domain Relations with Generative Adversarial Networks

Taeksoo Kim [1] Moonsu Cha [1] Hyunsoo Kim [1] Jung Kwon Lee [1] Jiwon Kim [1]



(a) Learning cross-domain relations **without any extra label**

(b) Handbag images (input) & **Generated** shoe images (output)

(c) Shoe images (input) & **Generated** handbag images (output)

Bidirectional GANs (2017)

ADVERSARIAL FEATURE LEARNING

Jeff Donahue
jdonahue@cs.berkeley.edu
Computer Science Division
University of California, Berkeley

Philipp Krähenbühl
philkr@utexas.edu
Department of Computer Science
University of Texas, Austin

Trevor Darrell
trevor@eecs.berkeley.edu
Computer Science Division
University of California, Berkeley

- Learn both the generator and the inverse mapping
- Discrimination not only in data space, but jointly in data and latent space

Bidirectional GANs (2017)

ADVERSARIAL FEATURE LEARNING

Jeff Donahue
jdonahue@cs.berkeley.edu
Computer Science Division
University of California, Berkeley

Philipp Krähenbühl
philkr@utexas.edu
Department of Computer Science
University of Texas, Austin

Trevor Darrell
trevor@eecs.berkeley.edu
Computer Science Division
University of California, Berkeley

- Learn both the generator and the inverse mapping
- Discrimination not only in data space, but jointly in data and latent space

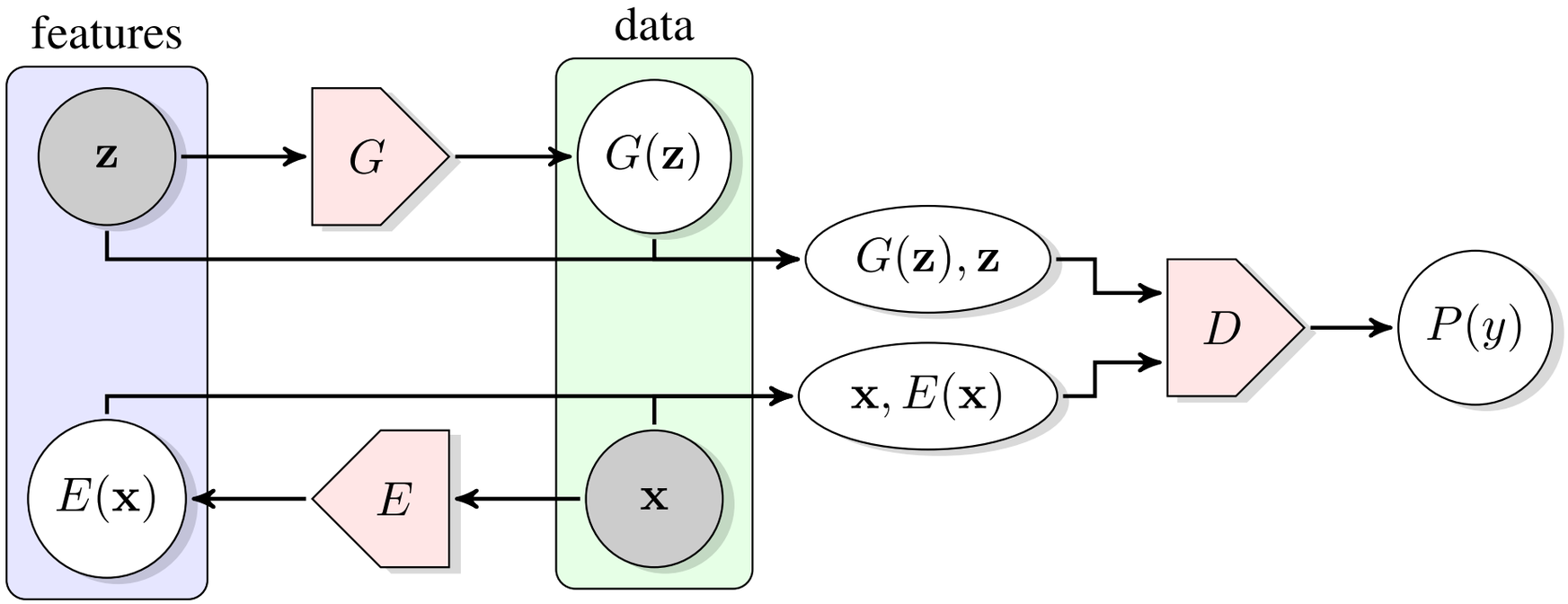


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

Bidirectional GANs (2017)

- Learn both the generator and the inverse mapping
- Discrimination not only in data space, but jointly in data and latent space

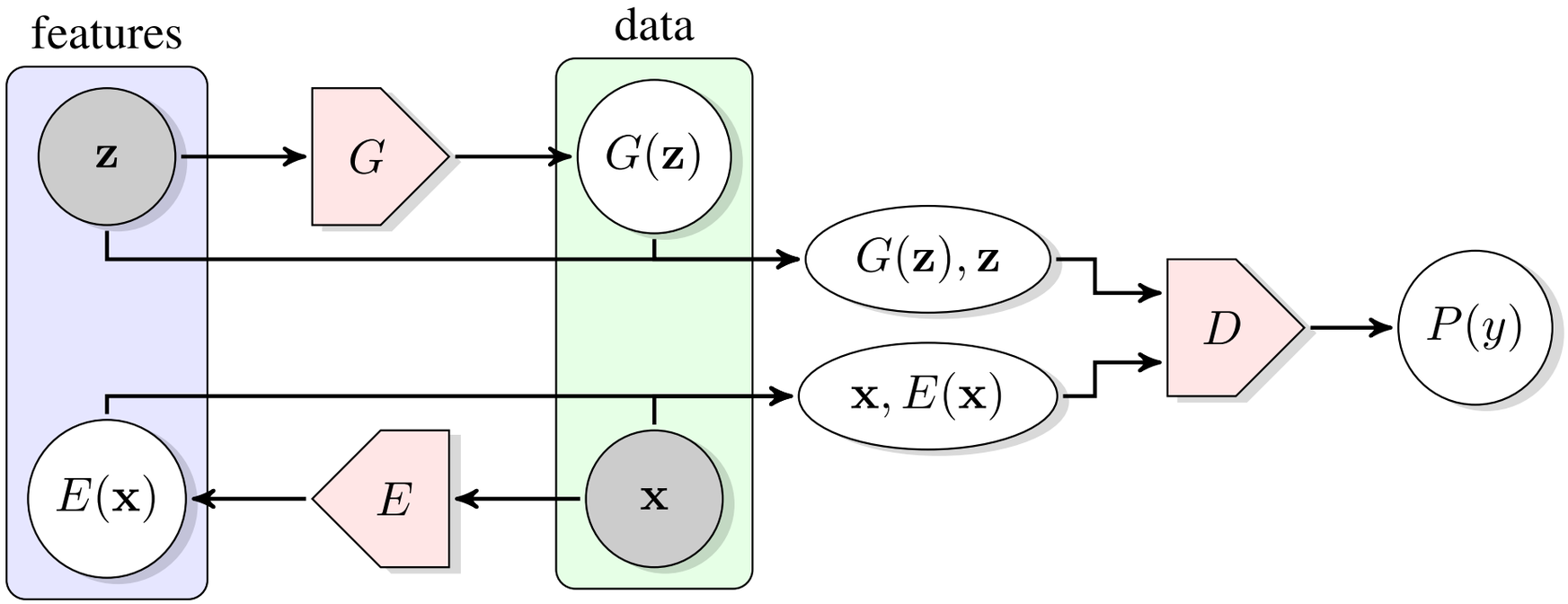


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

Bidirectional GANs (2017)

$$\min_{G,E} \max_D V(D,E,G)$$

where

$$V(D,E,G) := \mathbb{E}_{\mathbf{x}\sim p_\mathbf{X}}[\underbrace{\mathbb{E}_{\mathbf{z}\sim p_E(\cdot|\mathbf{x})}[\log D(\mathbf{x},\mathbf{z})]}_{\log D(\mathbf{x},E(\mathbf{x}))}] + \mathbb{E}_{\mathbf{z}\sim p_\mathbf{Z}}[\underbrace{\mathbb{E}_{\mathbf{x}\sim p_G(\cdot|\mathbf{z})}[\log(1-D(\mathbf{x},\mathbf{z}))]}_{\log(1-D(G(\mathbf{z}),\mathbf{z}))}]$$

- Learn both the generator and the inverse mapping
- Discrimination not only in data space, but jointly in data and latent space

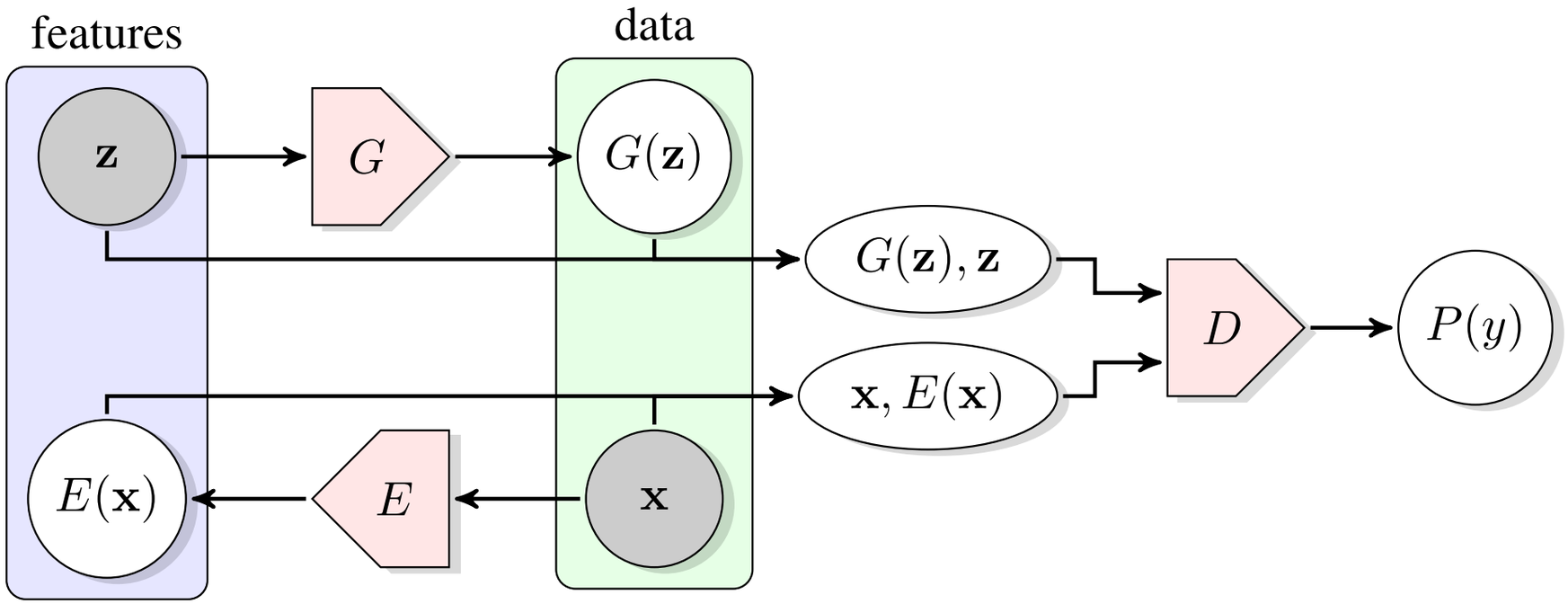


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

Domain Transfer Networks (2017)

UNSUPERVISED CROSS-DOMAIN IMAGE GENERATION

Yaniv Taigman, Adam Polyak & Lior Wolf
Facebook AI Research
Tel-Aviv, Israel
{yaniv,adampolyak,wolf}@fb.com

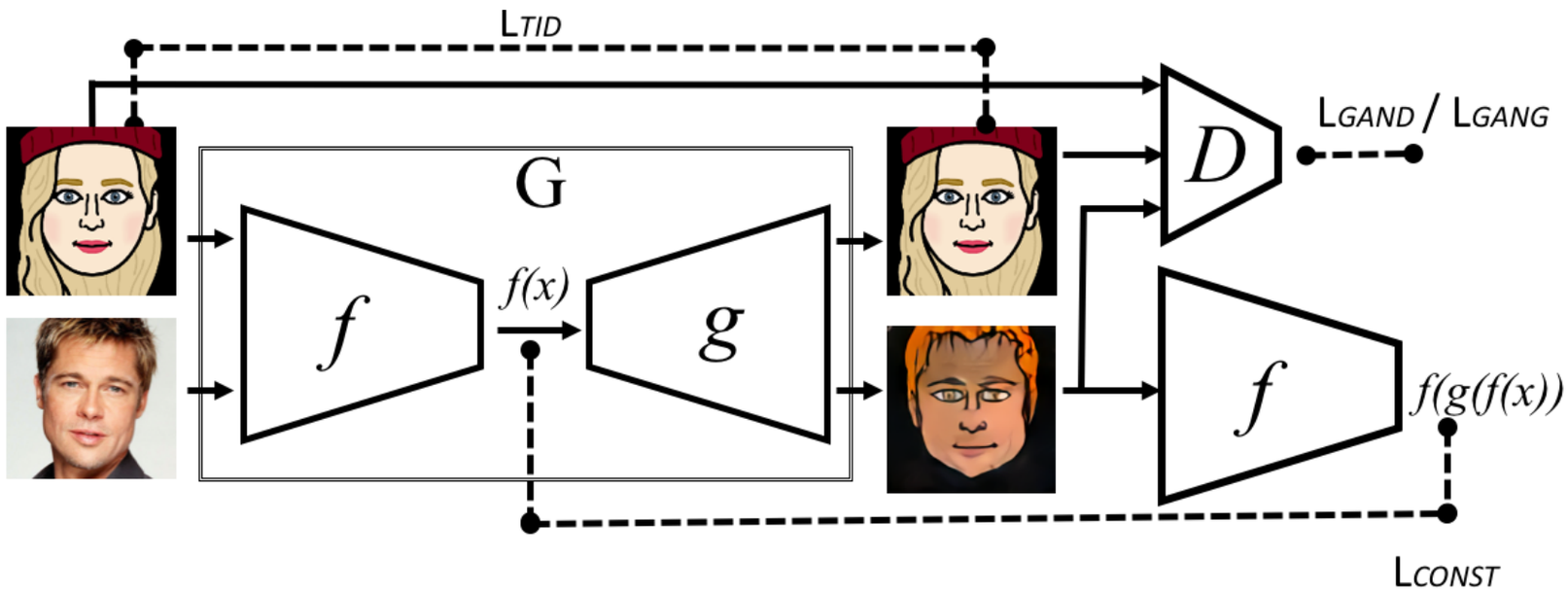


Figure 1: The Domain Transfer Network. Losses are drawn with dashed lines, input/output with solid lines. After training, the forward model G is used for the sample transfer.

Domain Transfer Networks (2017)

UNSUPERVISED CROSS-DOMAIN IMAGE GENERATION

Yaniv Taigman, Adam Polyak & Lior Wolf
Facebook AI Research
Tel-Aviv, Israel
{yaniv,adampolyak,wolf}@fb.com