Generative Adversarial Networks

SIBGRAPI 2017 Tutorial
Everything you wanted to know about Deep Learning for Computer Vision
but were afraid to ask

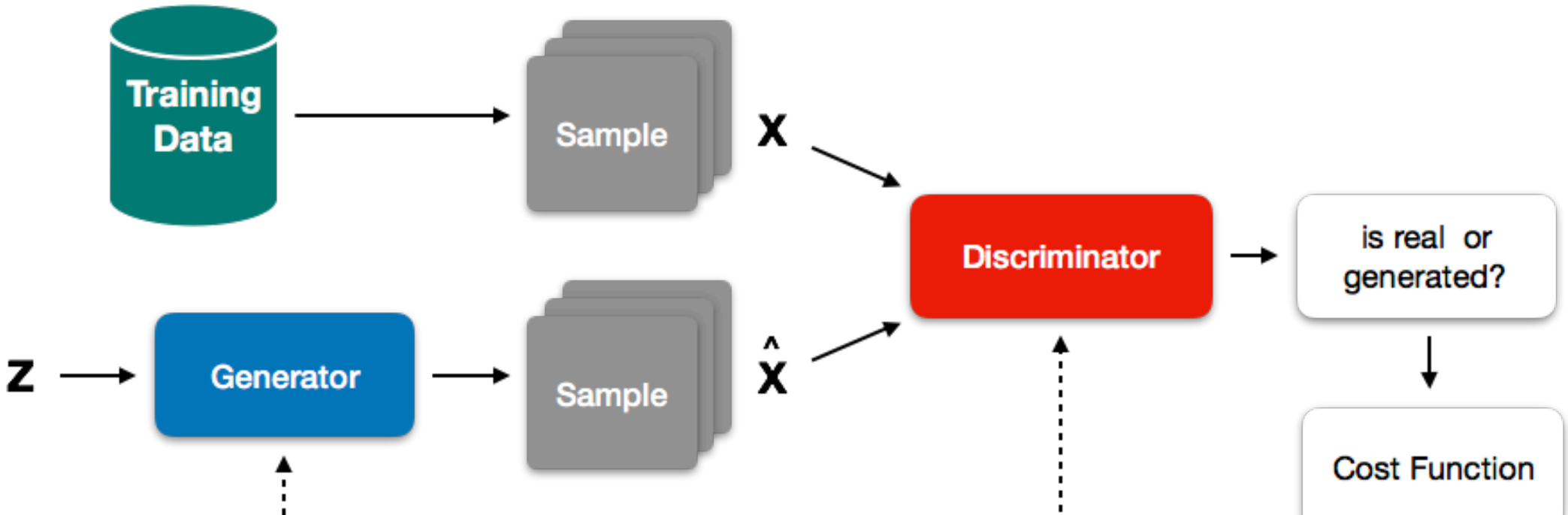
Presentation content inspired by Ian Goodfellow's tutorial "Generative Adversarial Networks" on NIPS 2016

Moacir A. Ponti, Leonardo S. F. Ribeiro, Tiago S. Nazare, Tu Bui, John Collomosse

Generative Adversarial Networks

SIBGRAPI 2017 Tutorial
Everything you wanted to know about Deep Learning for Computer Vision
but were afraid to ask

Presentation content inspired by Ian Goodfellow's tutorial "Generative Adversarial Networks" on NIPS 2016



Generative Adversarial Networks

Generative Models

Generator vs. Discriminator

Applications

Future Research on GANs

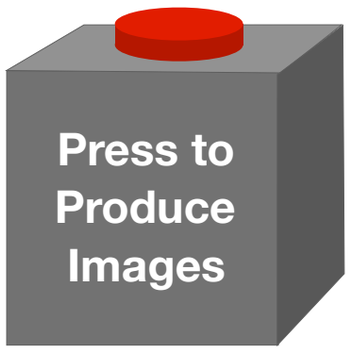
Generative Adversarial Networks

What are Generative Models?

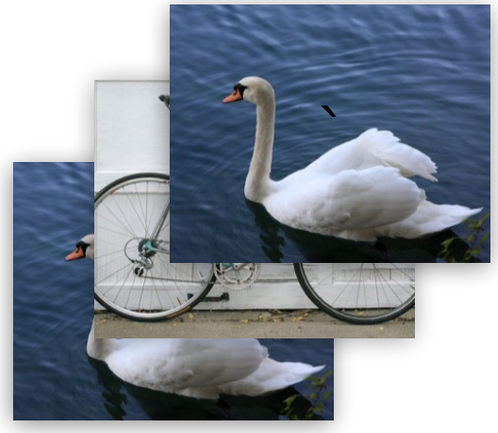
Why Generative Models?

Generative Adversarial Networks

What are Generative Models?



Black Box



Samples

Generative Adversarial Networks

Why Generative Models?

Manipulate High Dim Probability Distributions

Labels are hard to come by for many applications

Create art or manipulate natural images

Generative Adversarial Networks

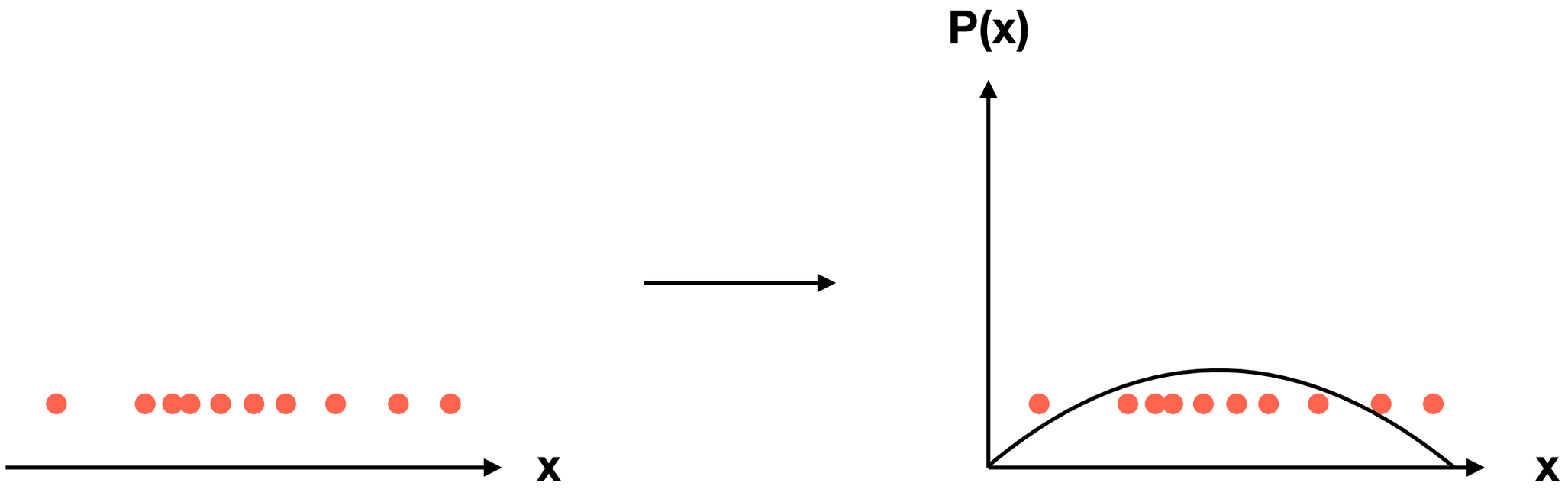
Generative Models

Generator vs. Discriminator

Applications

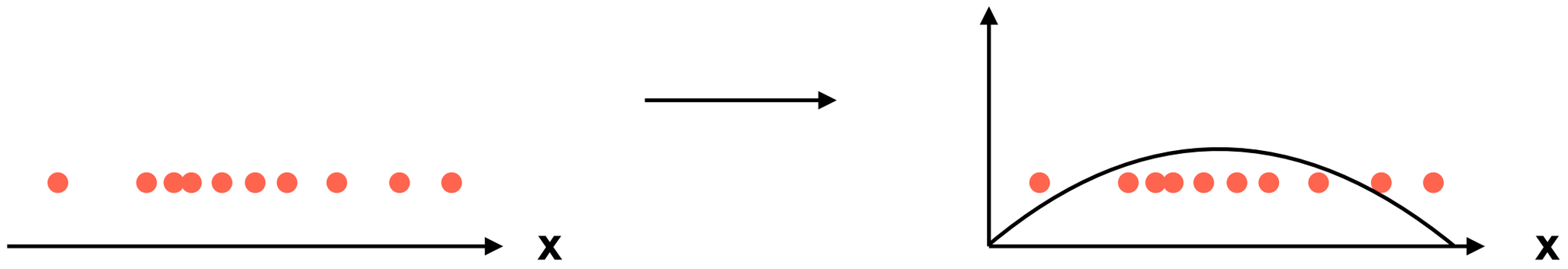
Future Research on GANs

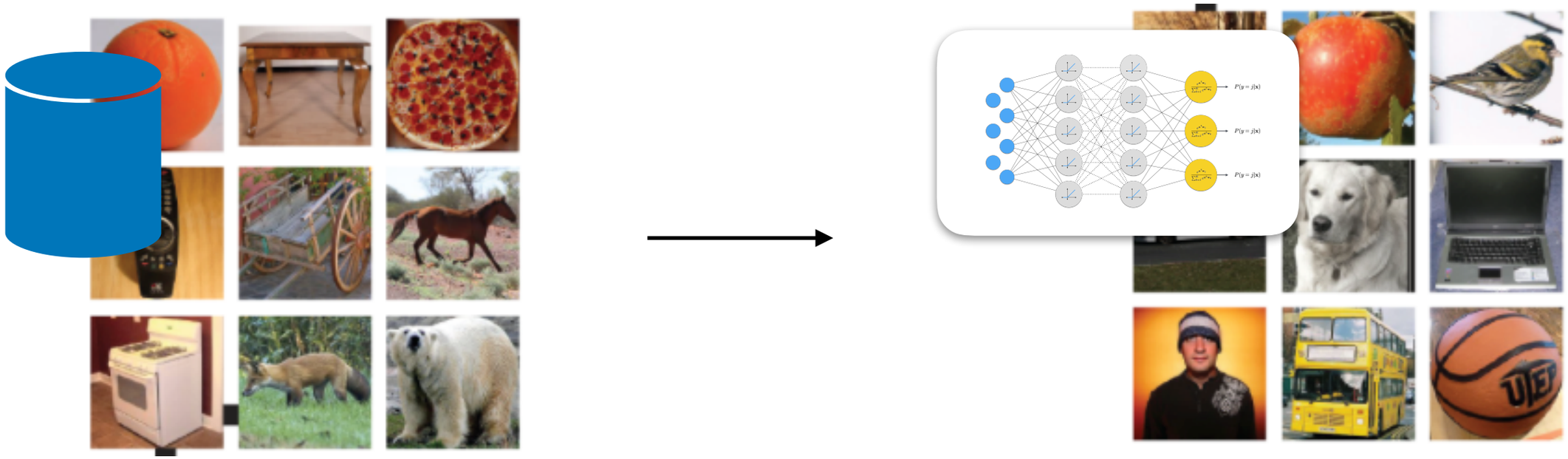
Generative Models



Some generative models learn to produce a probability density function or sample from one such density that represents an estimation of the original density where the data was drawn from

Generative Models





But sometimes we are more interested in the data produced by such densities and not on the densities themselves

Maximum Likelihood Estimation

$$\mathscr{L}(x, \theta) = \prod_{i=1}^{m} p_{\text{model}}(x_i; \theta)$$

One way a generative model can work is by using the principle of maximum likelihood (ML)

The likelihood is the joint probability of all samples from the training data “happening” on the estimated distribution

Maximum Likelihood Estimation

$$\mathscr{L}(x, \theta) = \prod_{i=1}^{m} p_{\text{model}}(x_i; \theta)$$

One way a generative model can work is by using the principle of maximum likelihood (ML)

The likelihood is the joint probability of all samples from the training data “happening” on the estimated distribution

GANs by default are not based on the ML principle but can, for the sake of comparison, be altered to do so (Goodfellow, 2014)

Categorization

Generative Models

Explicit Density Functions

Implicit Density Functions

We can categorize the methods that are based on the ML principle into two main buckets

Categorization

Generative Models

Explicit Density Functions

- Fully Visible Belief Nets (Frey *et al, 1996, 1998* WaveNet, 2016)
- Boltzmann Machines (Fahlman, 1983)
- Variational Autoencoders

Implicit Density Functions

- Methods based on Markov Chains
- Generative Adversarial Networks

Generative Adversarial Networks

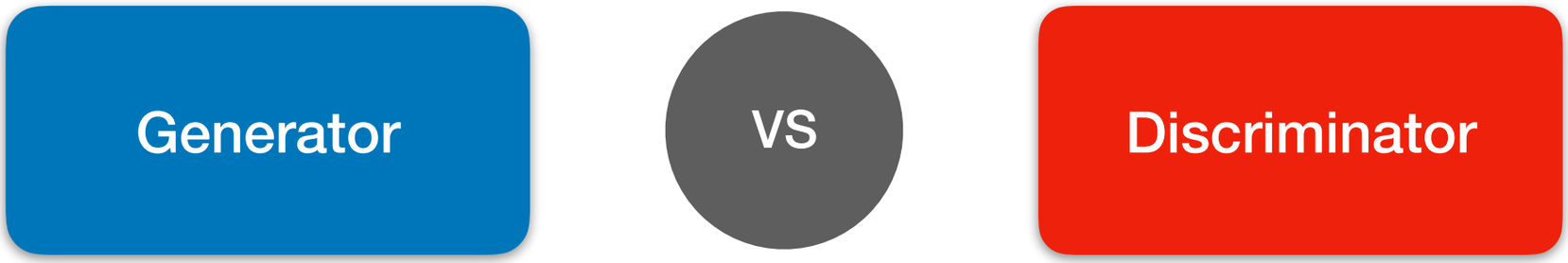
Generative Models

Generator vs. Discriminator

Applications

Future Research on GANs

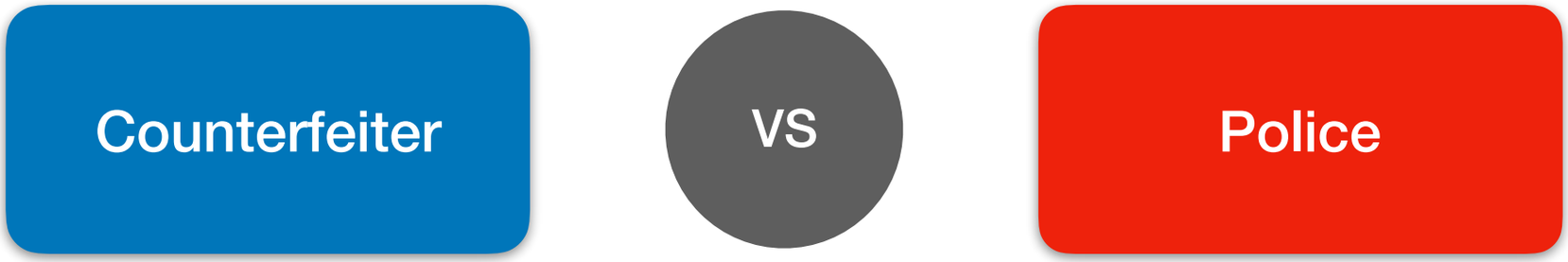
Generator vs. Discriminator



Two models are trained simultaneously, one is a **Generator** model that given an input z produces a sample x from an implicit probability distribution; The other is the **Discriminator**, a classifier that should identify if a sample was produced by the original distribution or by the generator approximation.

Intuition

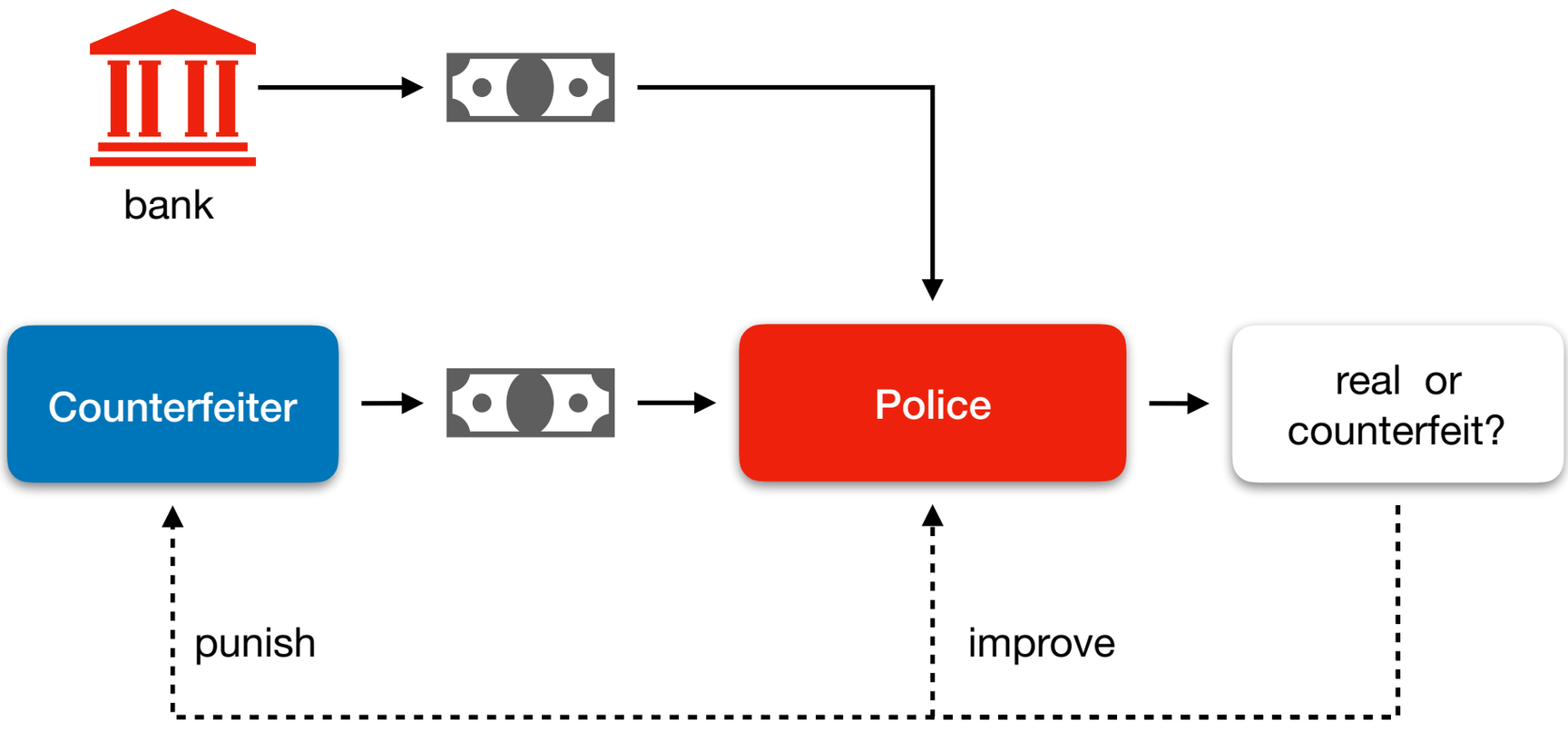
Generator vs. Discriminator



It is possible to think of the generator as a money counterfeiter, trained to fool the discriminator which can be thought as the police force

Intuition

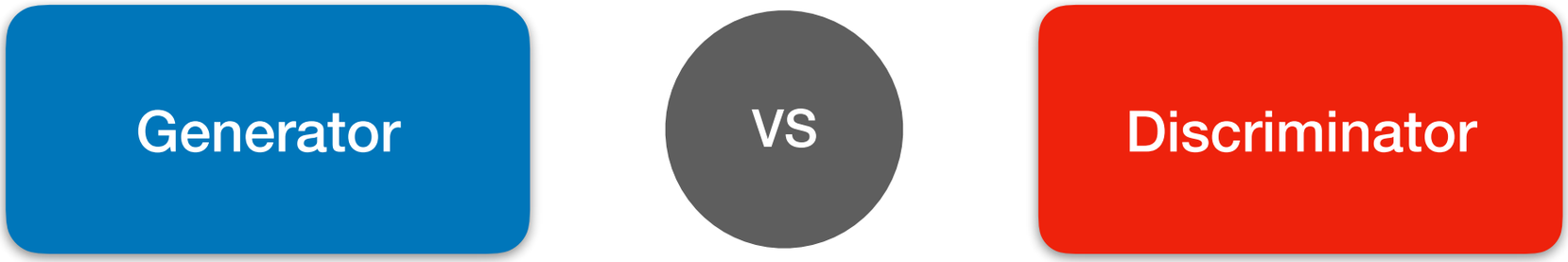
Generator vs. Discriminator



The police are then always improving its counterfeiting detection techniques at the same time as counterfeiters are improving their ability of producing better counterfeits.

GANs as a probabilistic model

Generator vs. Discriminator



Formally, GANs are a structured probabilistic model with latent variables z and observed variables x. The two models are represented generally by functions with the only requirement being that both must be **differentiable**.

GANs as a probabilistic model

Generator vs. Discriminator

Formally, GANs are a structured probabilistic model with ***latent variables z*** and ***observed variables x***. The two models are represented generally by functions with the only requirement being that both must be **differentiable.**

GANs as a probabilistic model

Generator vs. Discriminator

Latent Variables Space

Observed Variables Space

Formally, GANs are a structured probabilistic model with ***latent variables z*** and ***observed variables x***. The two models are represented generally by functions with the only requirement being that both must be **differentiable.**

Formally

Generator vs. Discriminator

Generator

Discriminator

Formally

Generator vs. Discriminator

Generator	The Generator can be any **differentiable** function G that takes z as an input and uses predefined parameters to output the observed variables x
Discriminator	The Discriminator can be any **differentiable** function D that takes x and uses predefined parameters to output a single scalar: a label defining where x was sampled from

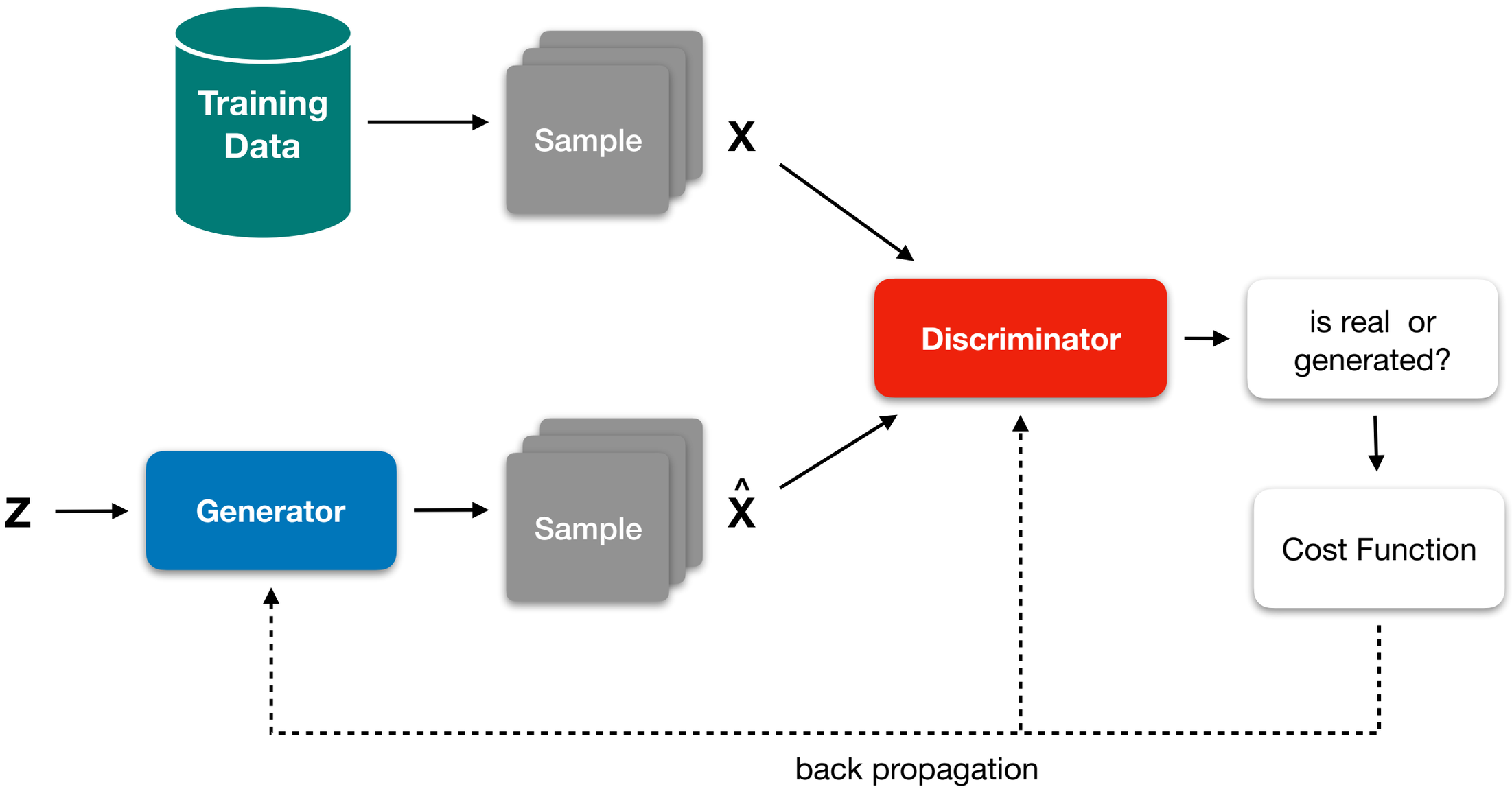
Formally

Generator vs. Discriminator

Generator	G is trained to minimize correct assignments of D
Discriminator	The function D is optimized to assign the correct labels to both training data and data produced by G

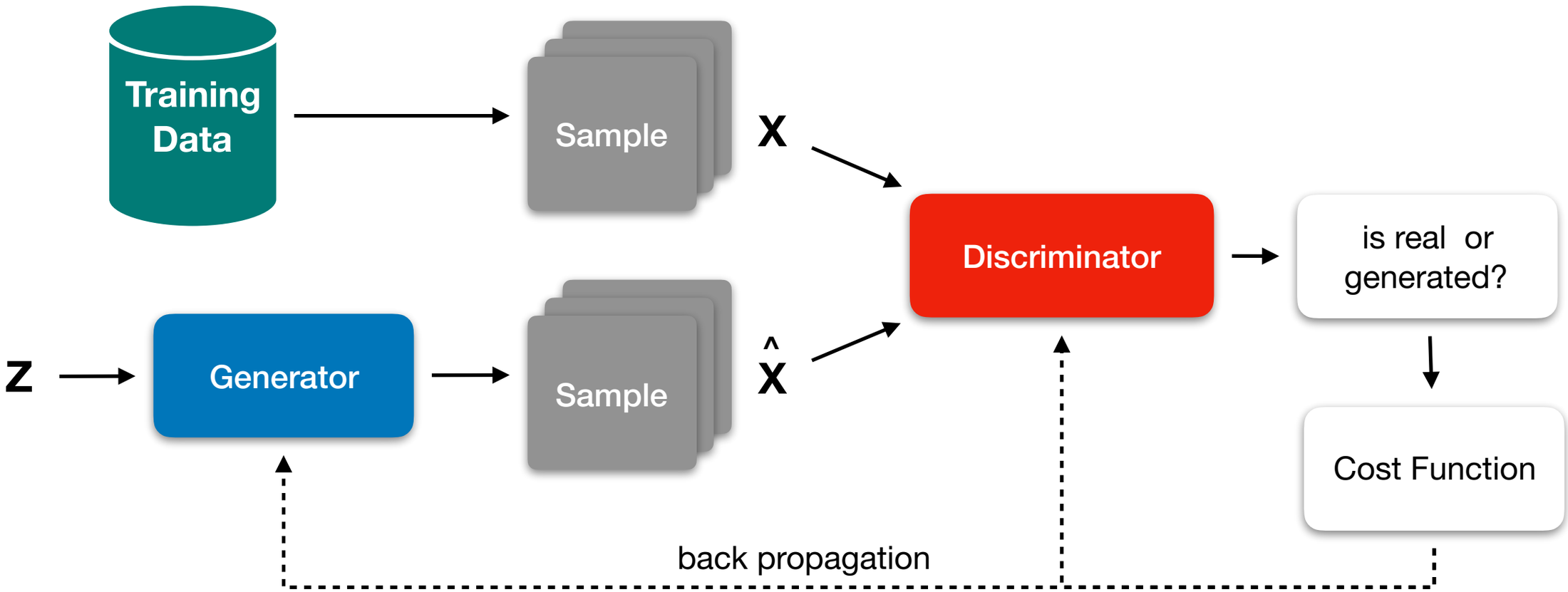
Diagram of training procedure

Generator vs. Discriminator



Training procedure

Generator vs. Discriminator



At each minibatch, values are sampled from both the training set and the set of random latent variables z and after one step through the networks, one gradient-based optimization method of choice is applied to update G and D's parameters based on loss functions designed to optimize each agent for the task

Training Procedure

Generator vs. Discriminator

Cost Function

Training Procedure

Generator vs. Discriminator

Cost Function	Designed to represent a zero-sum game. This way both players G and D, if implemented as compact convex functions, are guaranteed by the minimax theorem to achieve an **equilibrium point.**

Training Procedure

Generator vs. Discriminator

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_g(z)}[\log(1 - D(G(z)))]$$

Cost Function

In the original formulation, D and G play a **minimax game** with value function V(G, D)

Training Procedure

Generator vs. Discriminator

$$\mathscr{L}^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] - \frac{1}{2}\mathbb{E}_{z}[\log(1 - D(G(x)))]$$

$$\mathscr{L}^{(G)}(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}$$

Cost Function

The **minimax game** value function can be used to design the cost functions for the training procedure

Training Procedure

Generator vs. Discriminator

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

$$p_g = p_{data} \rightarrow D_G^*(x) = \frac{1}{2}$$

If we fix the Generator, this is the optimum discriminator value because we know functions of the kind $y \rightarrow a \log(y) + b \log(1 - y)$ achieve their maximum at $\frac{a}{a+b}$

Training Procedure

Generator vs. Discriminator

$$C(G) = \mathbb{E}_{x \sim p_{data}}[\log D^*_G(x)] + \mathbb{E}_{x \sim p_g}[1 - \log D^*_G(x)]$$

$$C(G) = \mathbb{E}_{x \sim p_{data}}[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}] + \mathbb{E}_{x \sim p_g}[1 - \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}]$$

Fixing the discriminator at its optimum point, we can redefine the value function

Training Procedure

Generator vs. Discriminator

$$C(G) = -\log 4 + 2.JSD(p_{data} || p_g)$$

It is then possible to manipulate this value function and rewrite it with a Jensen-Shannon divergence, which is always non-negative or zero, proving that there is an equilibrium at $C(G) = \log 4$ when $p_g = p_{data}$

Training Procedure

Generator vs. Discriminator

$$\mathscr{L}^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] - \frac{1}{2}\mathbb{E}_{z}[\log(1 - D(G(x)))]$$

$$\mathscr{L}^{(G)}(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}$$

Guaranteed to converge if both G and D are **convex functions updated in function space.**

Training Procedure

Generator vs. Discriminator

$$\mathscr{L}^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] - \frac{1}{2}\mathbb{E}_{z}[\log(1 - D(G(x)))]$$

$$\mathscr{L}^{(G)}(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}$$

There is at least one **problem** without these guarantees, when discriminator achieves the point where it is performing its task successfully, the gradient of the **discriminator's cost approaches zero.**

Training Procedure

Generator vs. Discriminator

$$\mathscr{L}^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] - \frac{1}{2}\mathbb{E}_z[\log(1 - D(G(x)))]$$

$$\mathscr{L}^{(G)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_z \log D(G(x))$$

Cost Function

The **heuristic non-saturating game** suggested by (Goodfellow, 2016) is a variation of the minimax game.

Training Procedure

Generator vs. Discriminator

$$\mathscr{L}^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] - \frac{1}{2}\mathbb{E}_z[\log(1 - D(G(x)))]$$

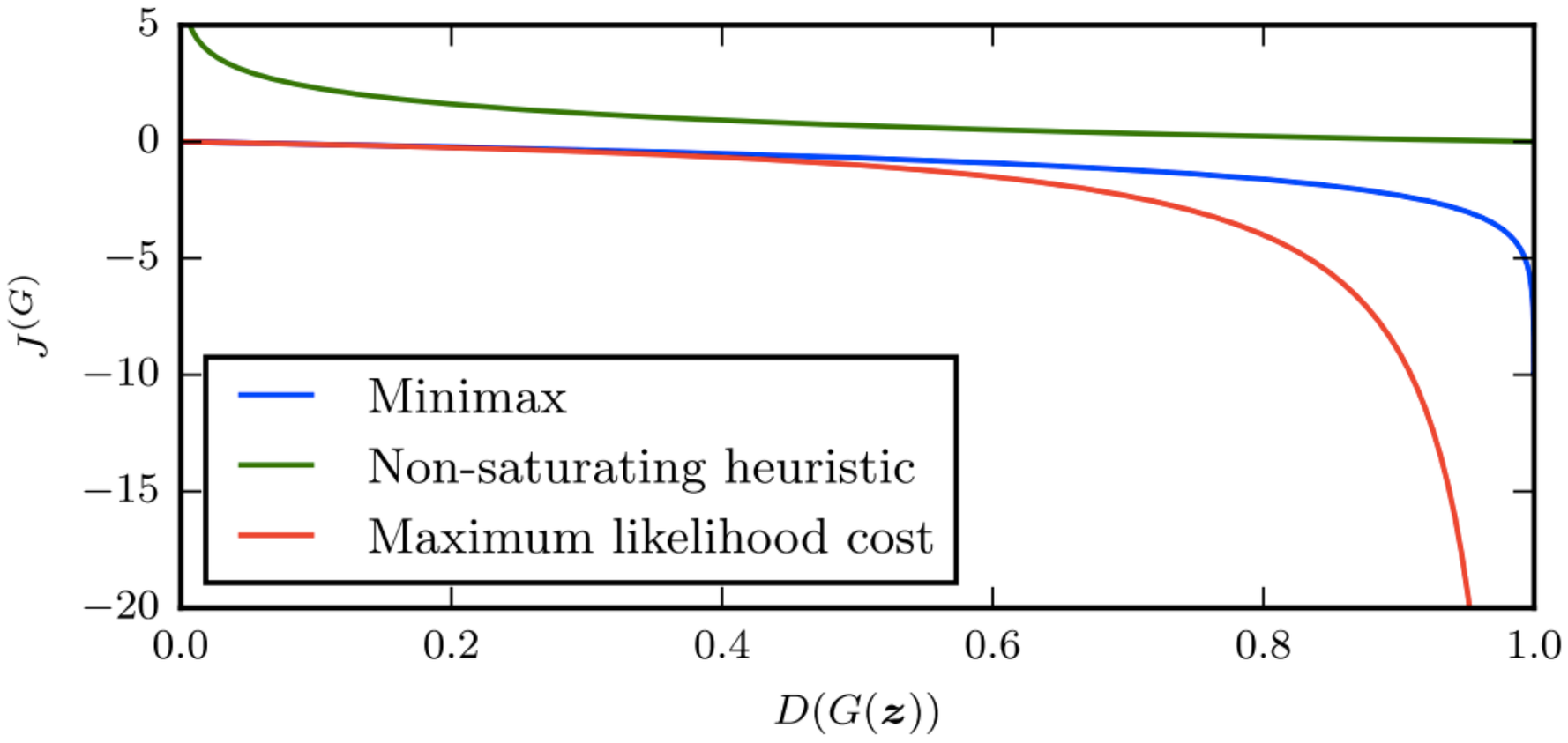
$$\mathscr{L}^{(G)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_z \log D(G(x))$$

Cost Function

In the **heuristic non-saturating game**, rather than minimizing log-probability of the discriminator being correct, the generator maximises the log-probability of the discriminator being mistaken

Training Procedure

Generator vs. Discriminator



Cost Function

Comparison of the obtained gradients for the generator with different cost functions

Generative Adversarial Networks

Generative Models

Generator vs. Discriminator

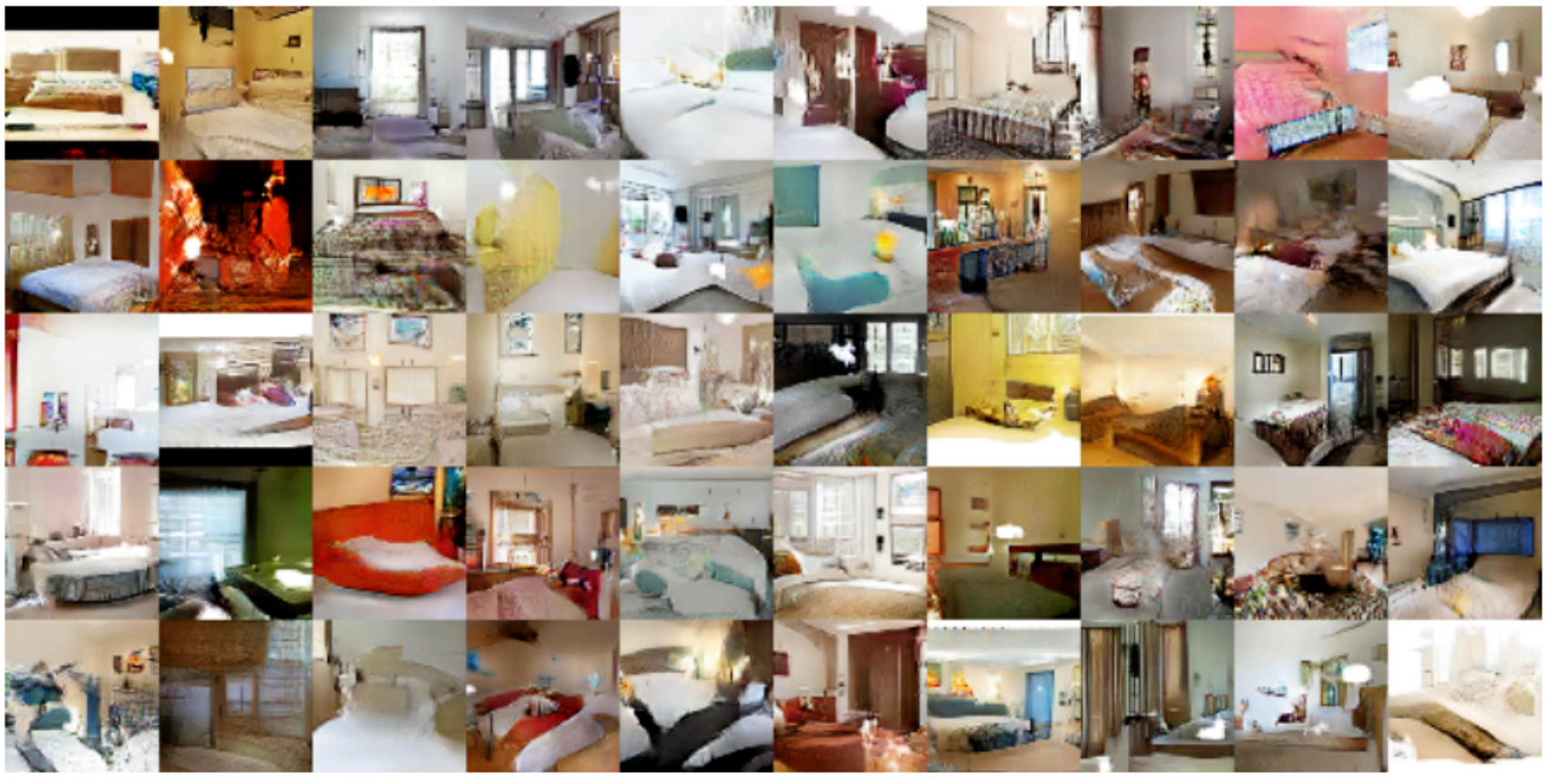
Applications

Future Research on GANs

DCGAN for Image Generation

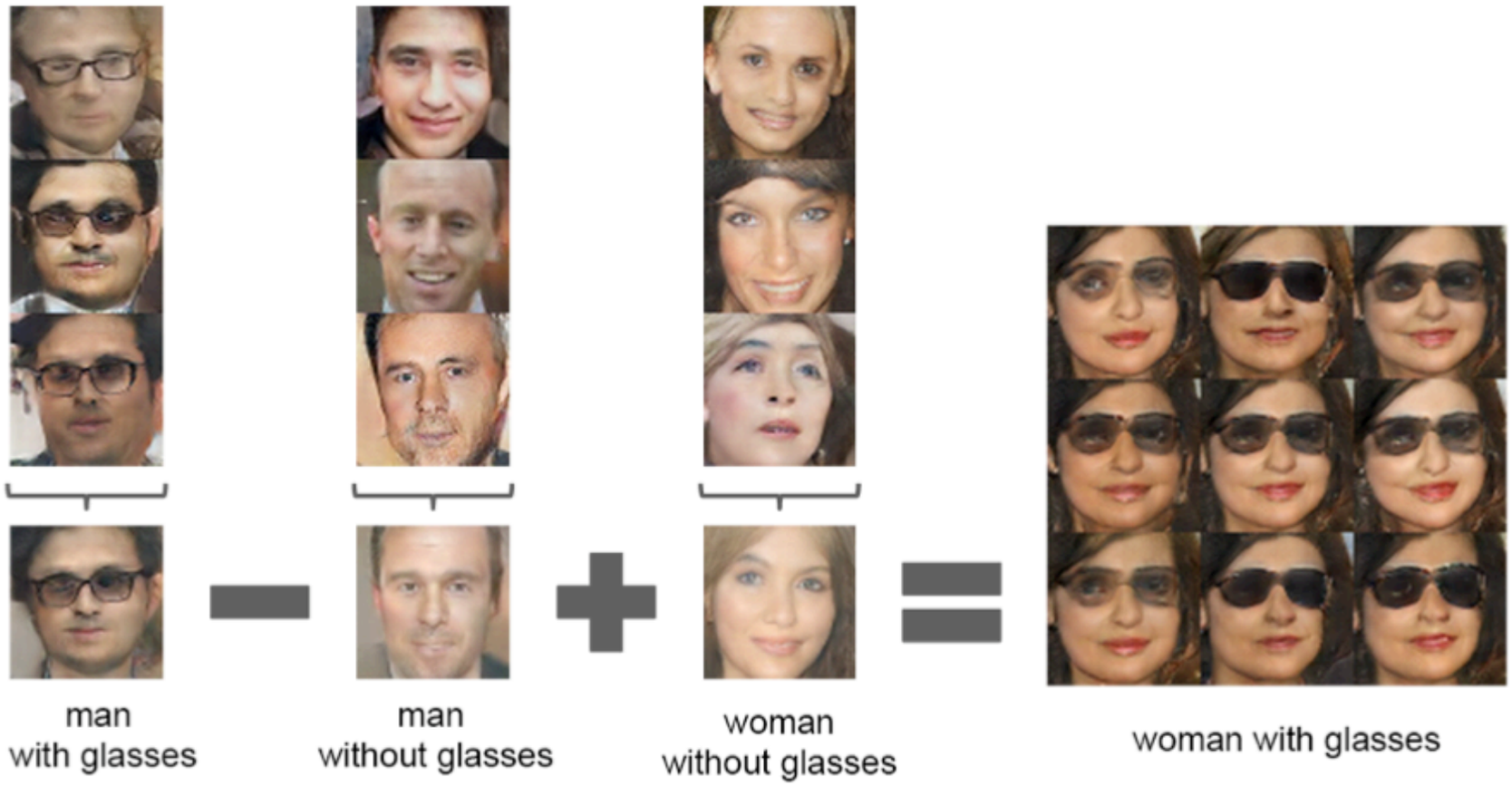
Applications

DCGAN trained for 5 epochs on LSUN Bedrooms



Alec Radford, Luke Metz and Soumith Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”

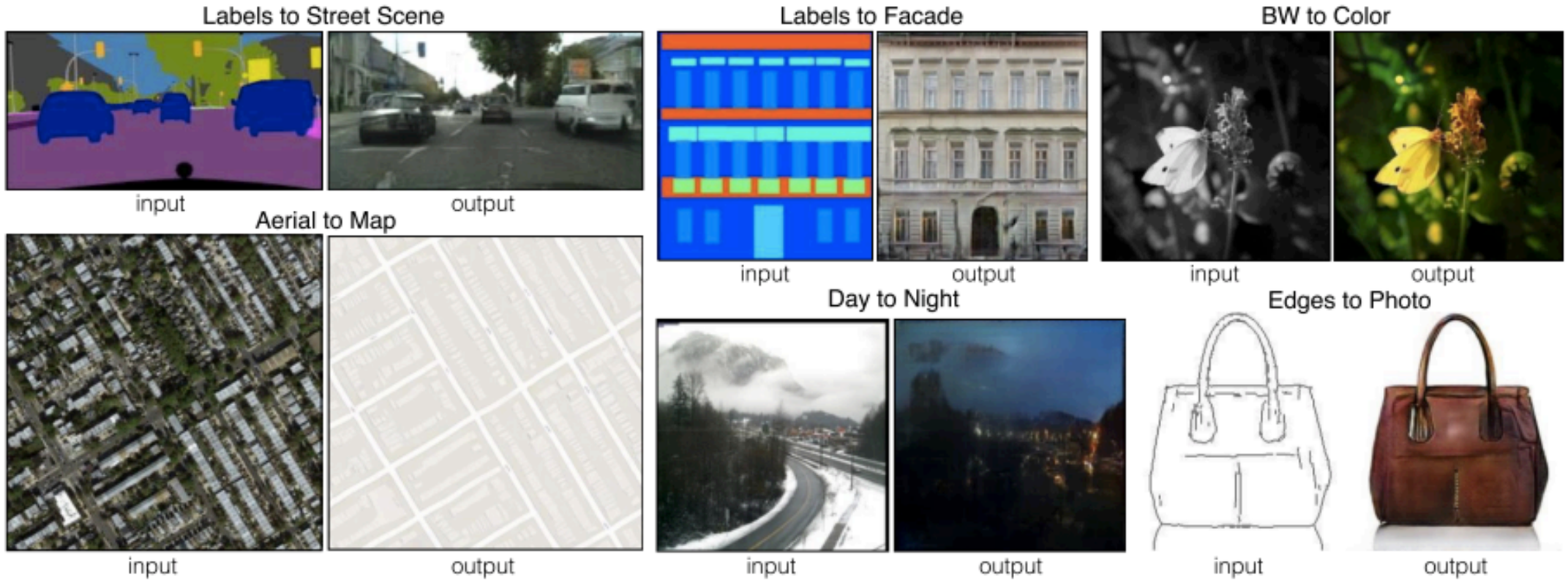
Vector Space Arithmetic



Alec Radford, Luke Metz and Soumith Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”

Image to Image Translation

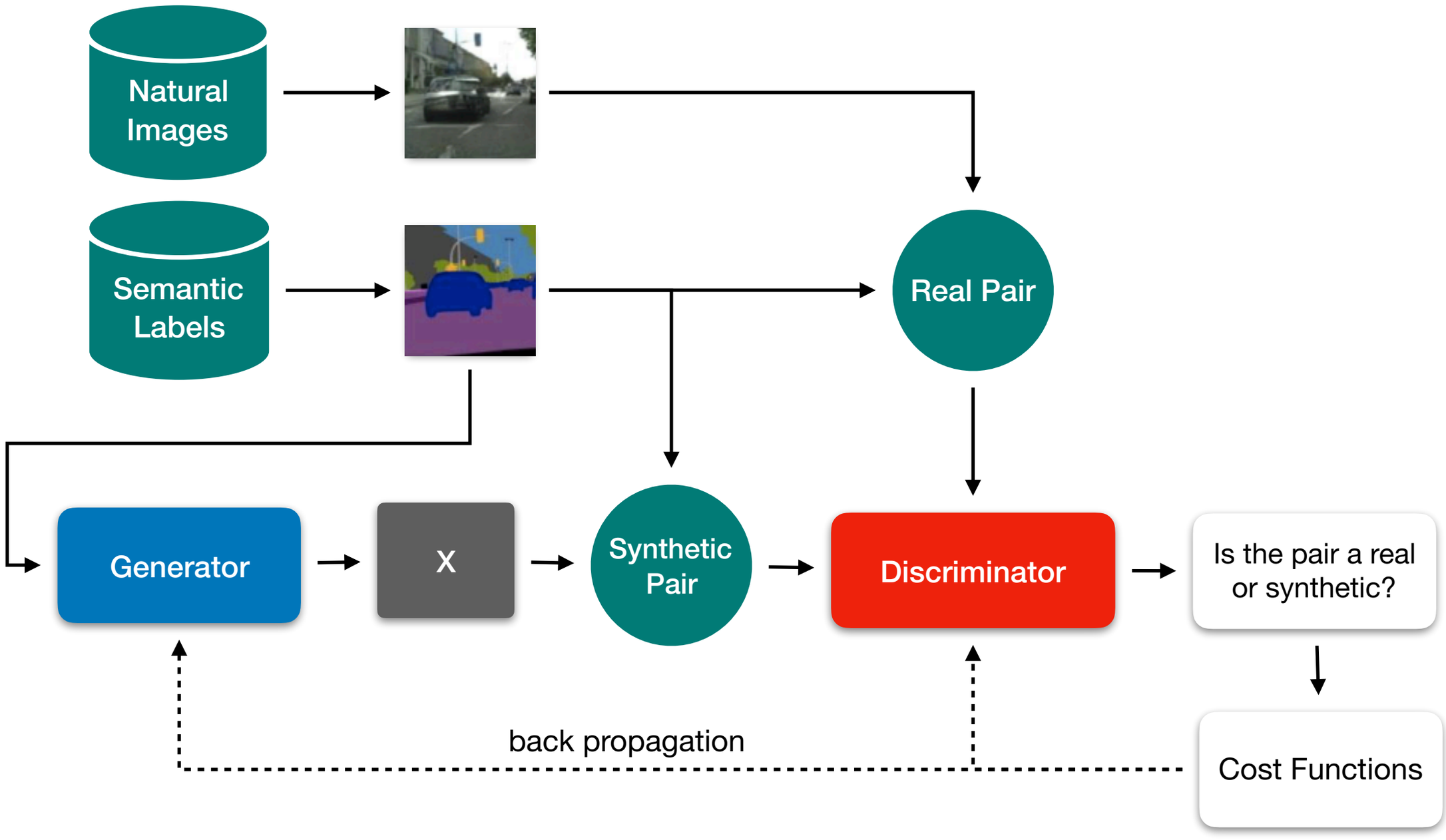
Applications



Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks”

Image to Image Translation

Applications



Single Image Super-Resolution

Applications

bicubic
(21.59dB/0.6423)

SRResNet
(23.53dB/0.7832)

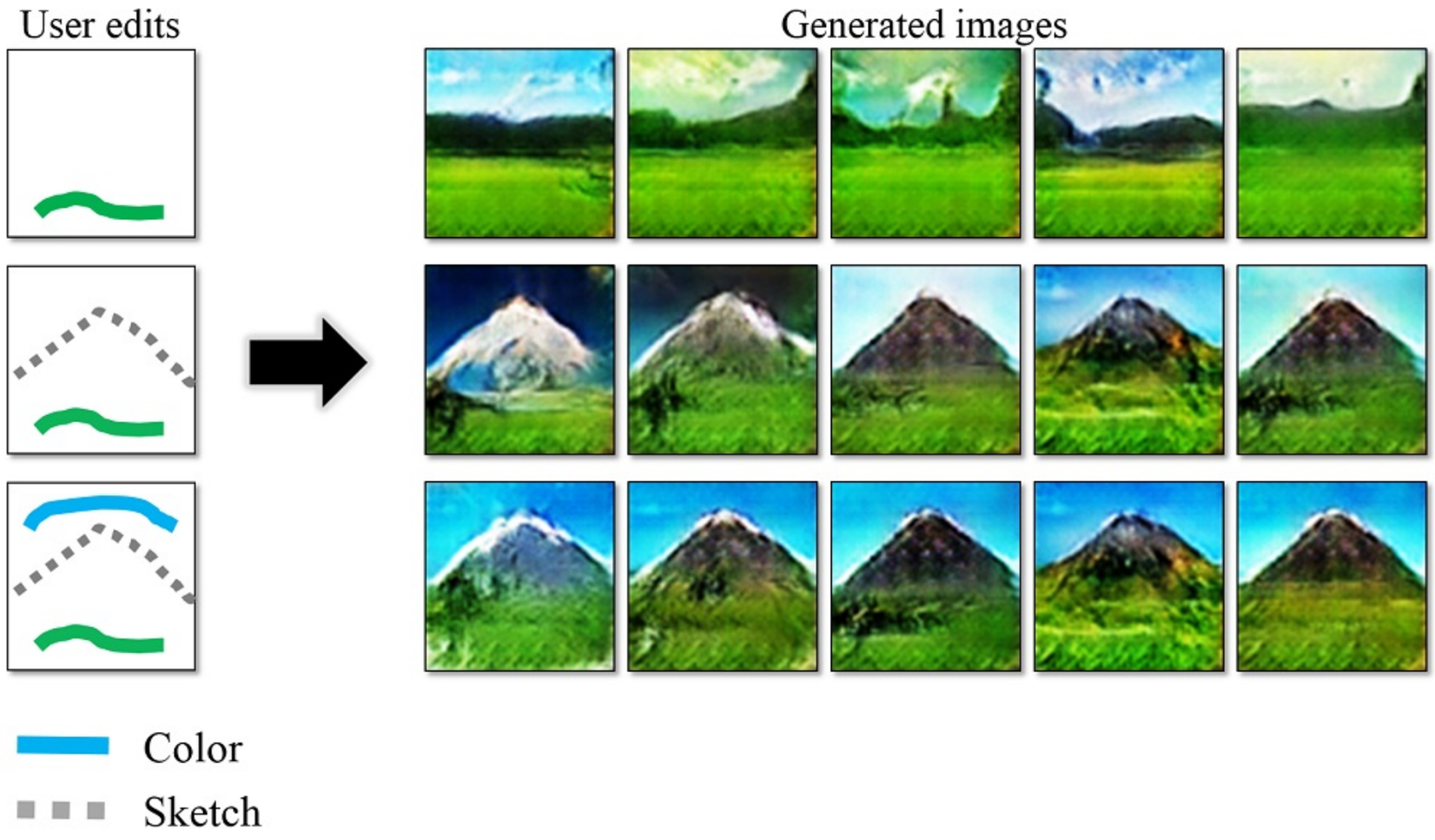
SRGAN
(21.15dB/0.6868)

original

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang and Wenzhe Shi, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”

Visual Manipulation

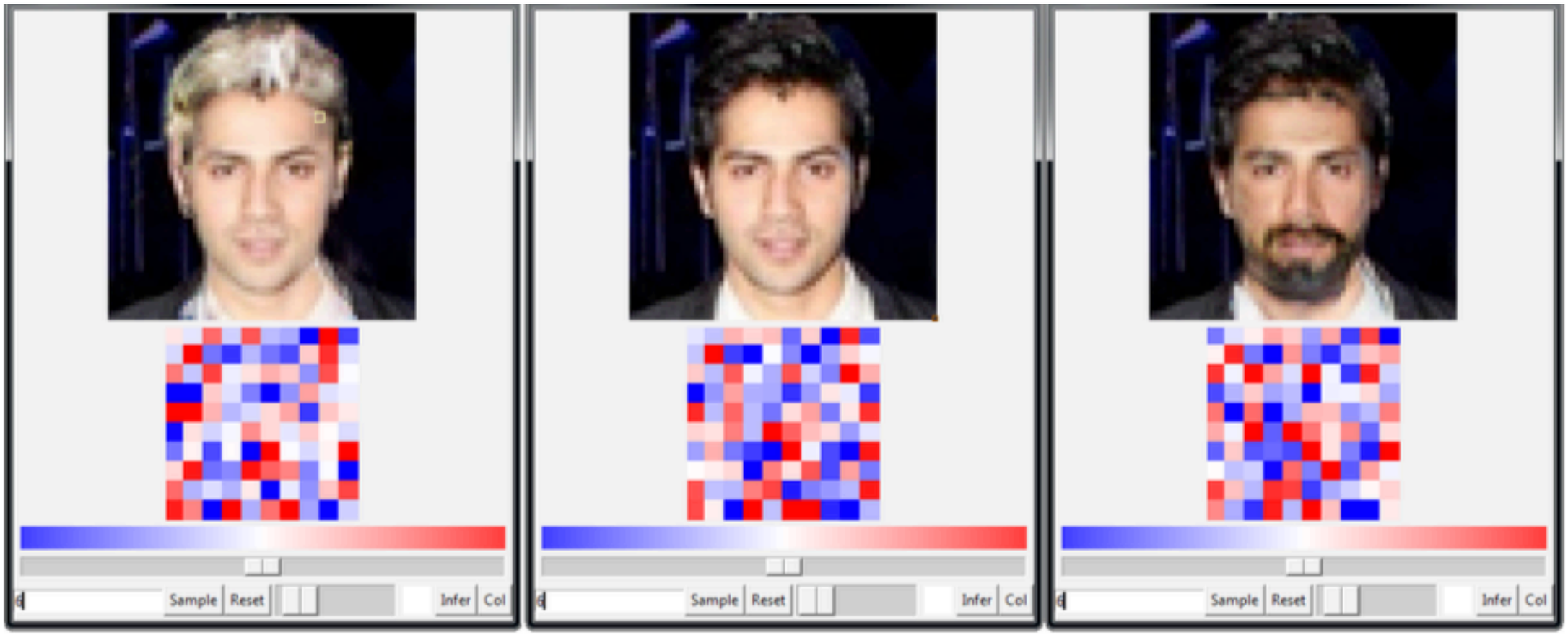
Applications



Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman and Alexei A. Efros, “Generative Visual Manipulation on the Natural Image Manifold”

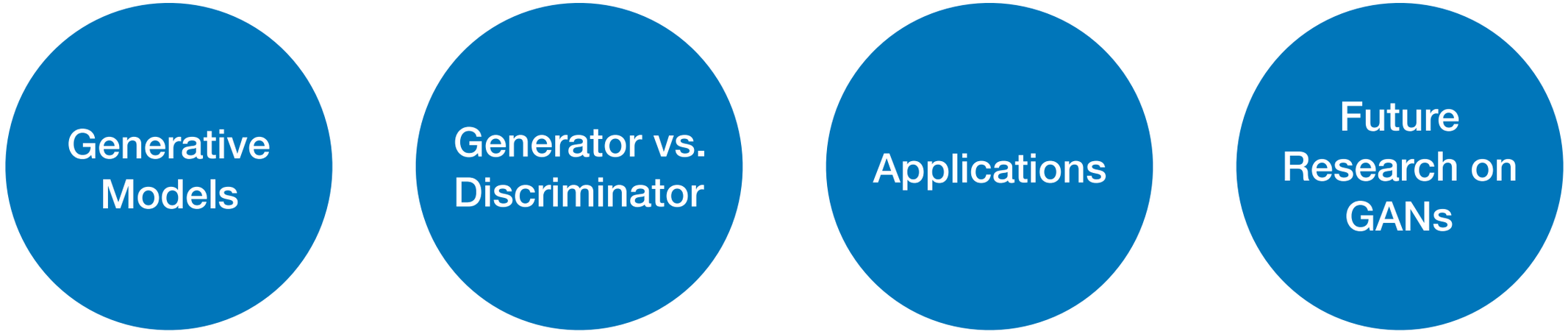
Visual Manipulation

Applications



Andrew Brock, Theodore Lim, J.M. Ritchie and Nick Weston, “Neural Photo Editing with Introspective Adversarial Networks”

Generative Adversarial Networks



Future Research on GANs

Non-convergence

Mode collapse

Evaluation of GANs

Discrete Output

Latent code usage

Semi-supervised learning

Generative Adversarial Networks

Generative Models

Generator vs. Discriminator

Applications

Future Research on GANs

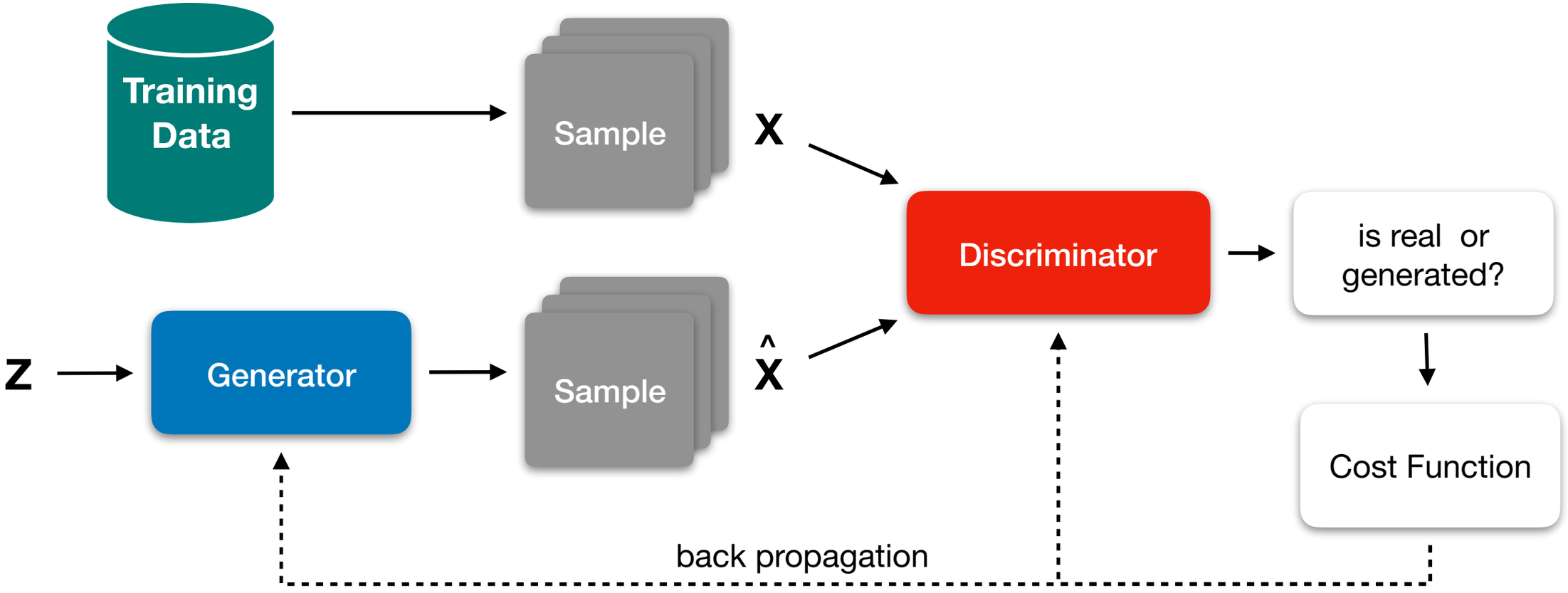
Generative Adversarial Networks

Generative Models

Generator vs. Discriminator

Applications

Future Research on GANs



Bibliography

G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “WaveNet: A Generative Model for Raw Audio,” sep 2016. [Online]. Available: http: //arxiv.org/abs/1609.03499

I. Goodfellow, “NIPS 2016 Tutorial: Generative Adversarial Networks,” dec 2016. [Online]. Available: http:// arxiv.org/ abs/1701.00160

C. Doersch, “Tutorial on variational autoencoders,” *arXiv preprint arXiv:1606.05908*, 2016.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, 2014

A. Fischer and C. Igel, “Training restricted boltzmann ma- chines: An introduction,” *Pattern Recognition*, vol. 47, no. 1, pp. 25–39, 2014.

R. Salakhutdinov and G. Hinton, “Deep boltzmann ma- chines,” in *Artificial Intelligence and Statistics*, 2009, pp. 448–455.