Generative Adversarial Networks

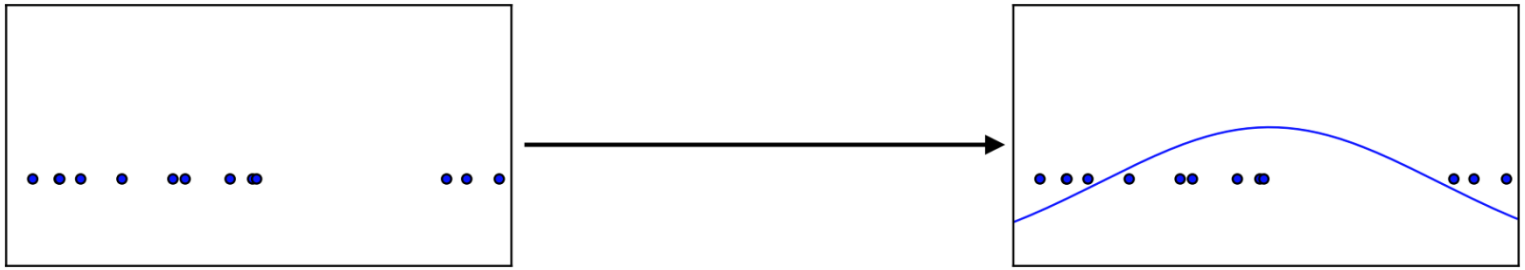
Miloš Jordanski

Generative models

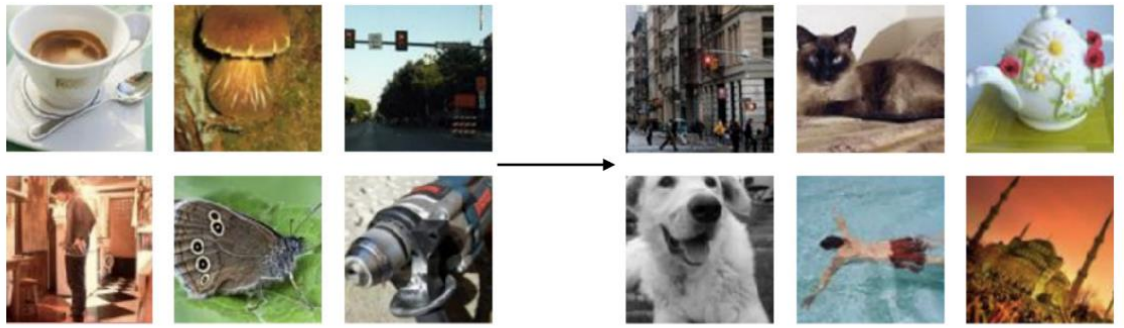
- Training data $D = \{x_1, \ldots, x_N\} \sim p_{data}(x)$
- Result: distribution $p_{model}(x) \cong p_{data}(x)$

Generative models

- $p_{model}(x)$ directly



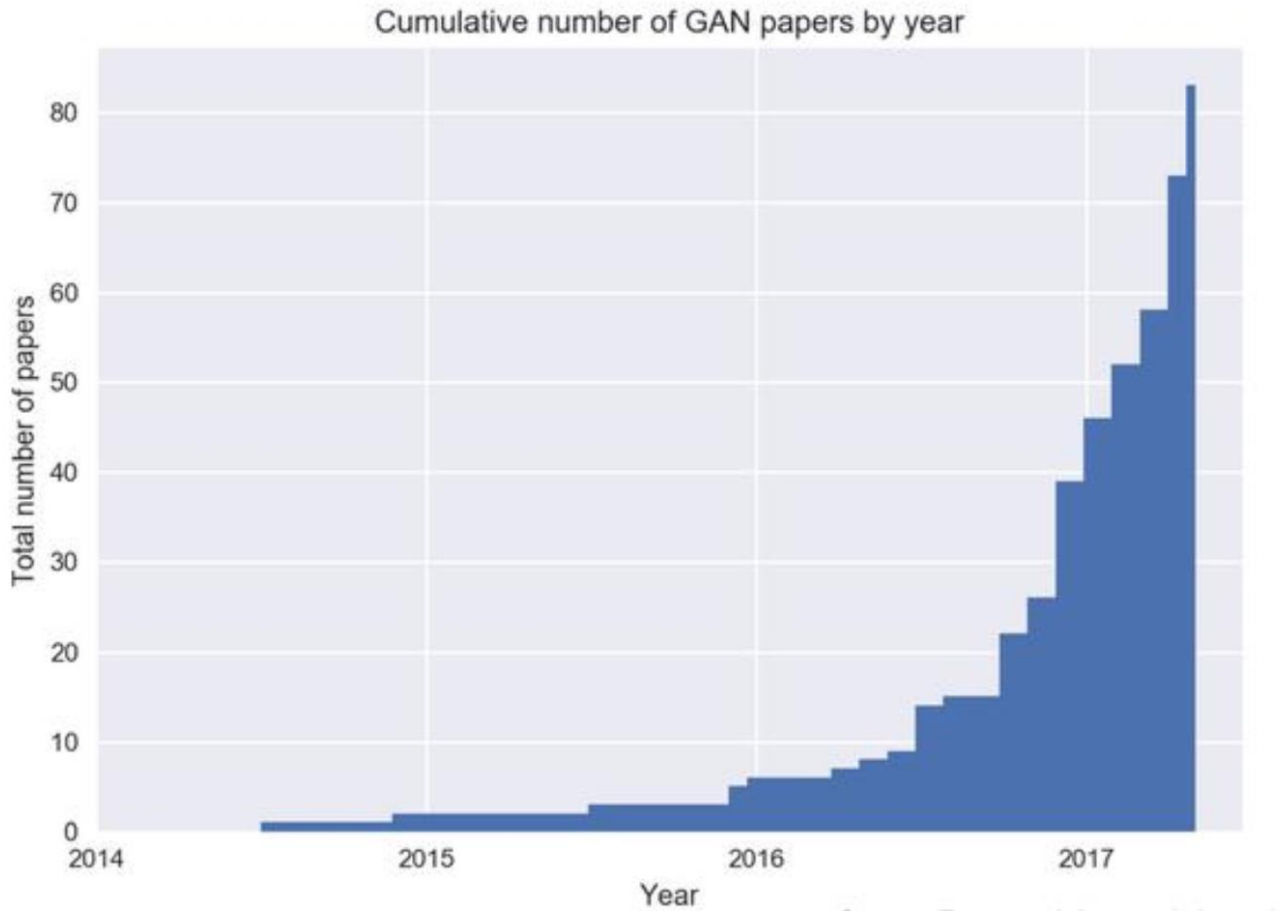
- Generate samples from $p_{model}(x)$



Applications

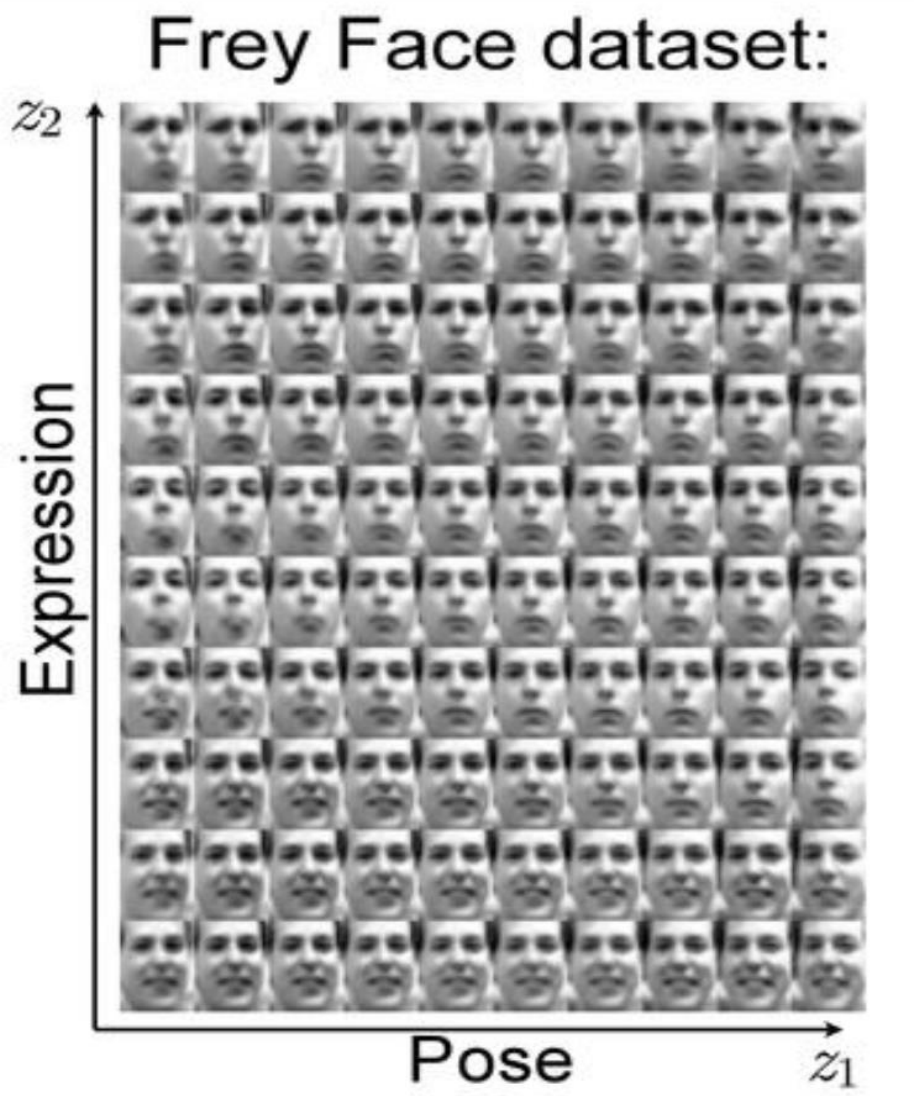
- Additional training data
- Missing values
- Semi-supervised learning
- Reinforcement learning
- Multiple correct answers
- Text-to-Image Synthesis
- Learn useful embeddings
- …

Generative Adversarial Networks

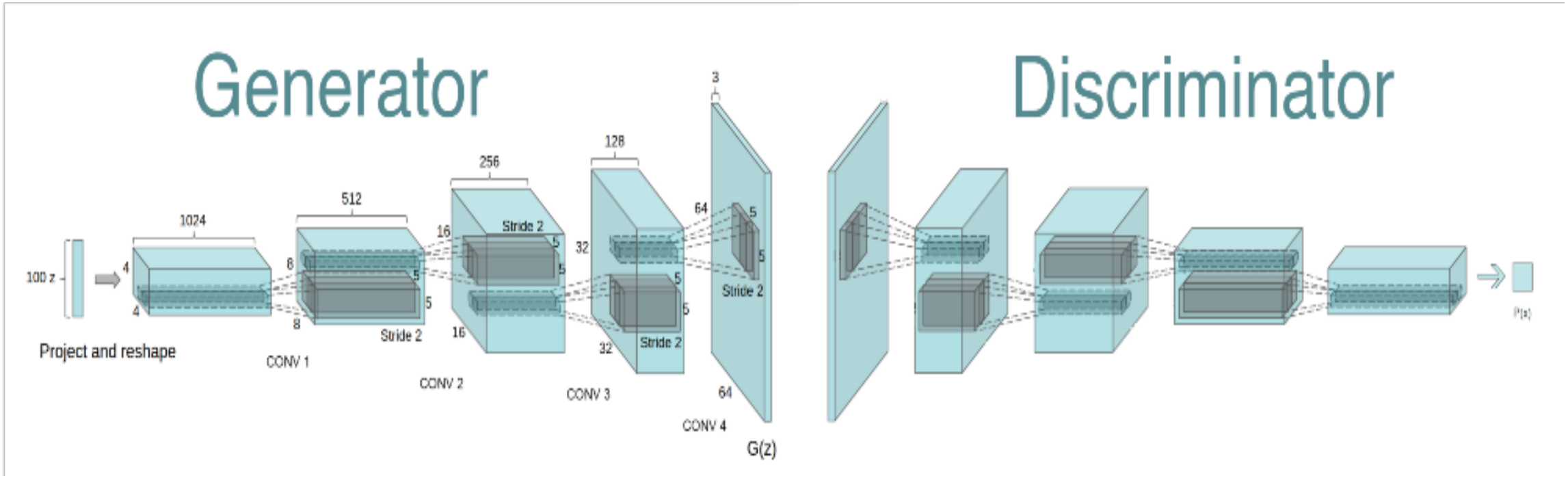


Models with latent space

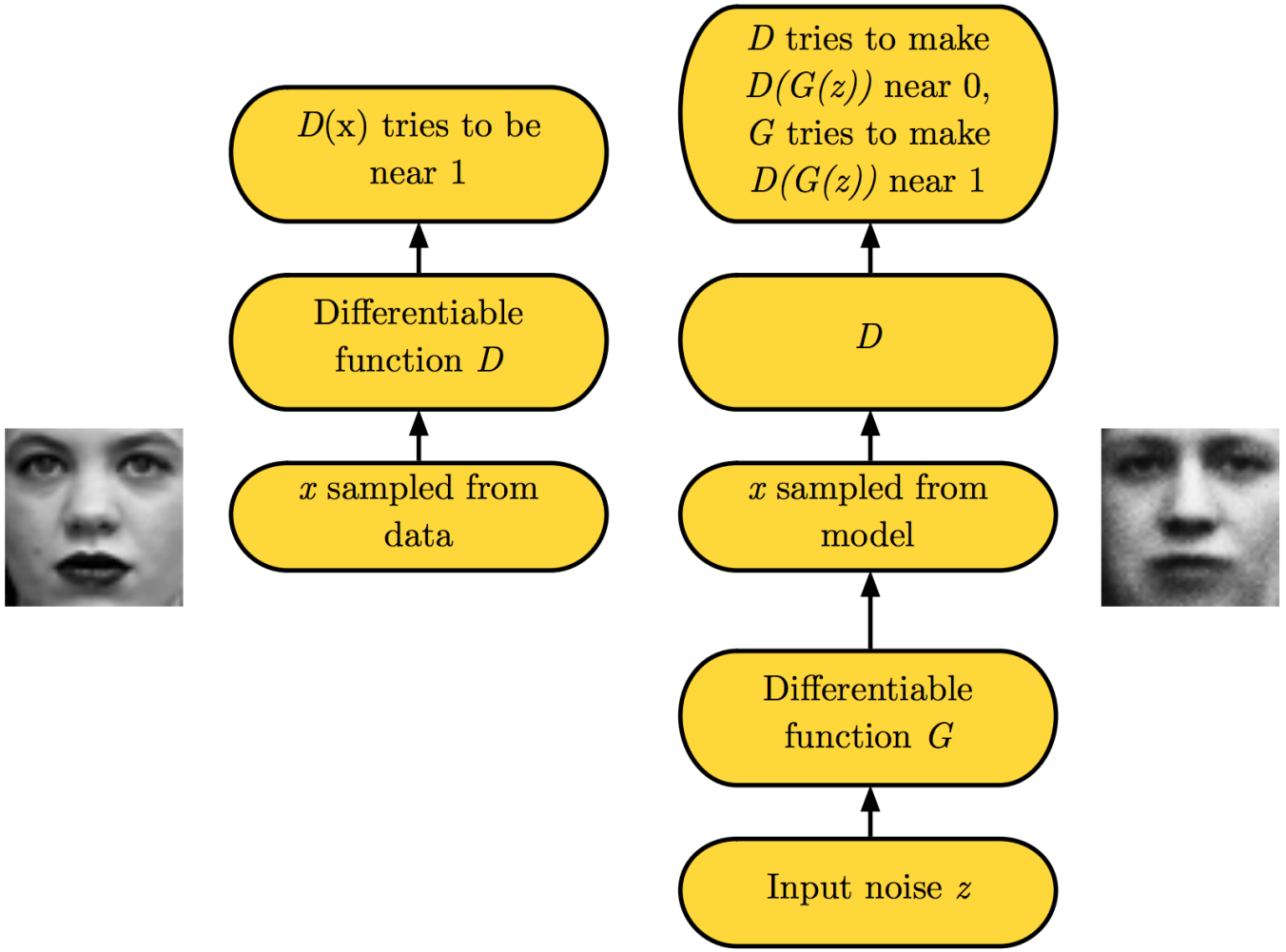
- x - observed variables
- z – latent variables



DCGAN


Generator
Discriminator
100 z
Project and reshape
1024
512
256
128
3
4
8
16
32
64
5
Stride 2
CONV 1
CONV 2
CONV 3
CONV 4
G(z)

GAN - architecture

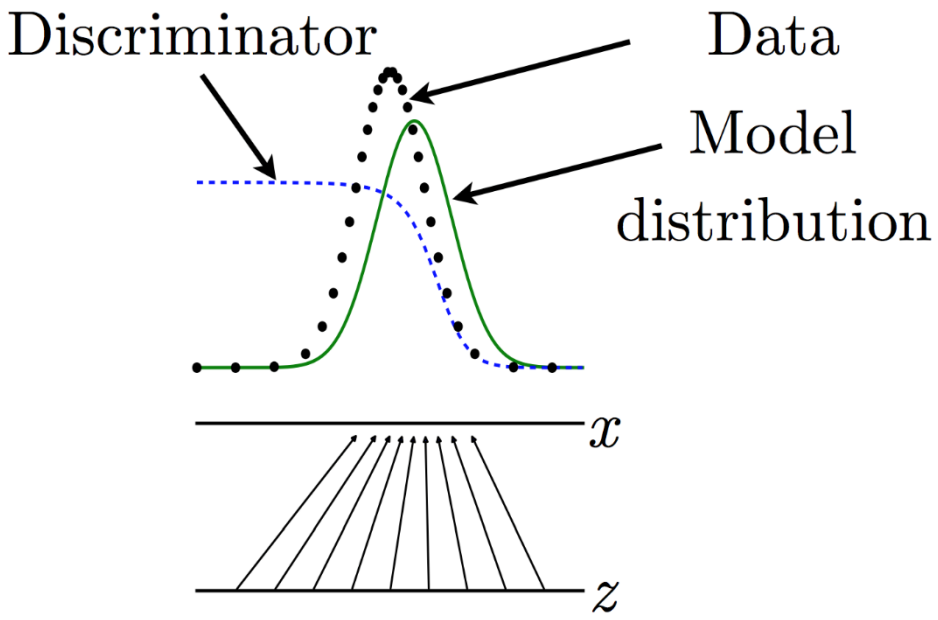


GAN - discriminator

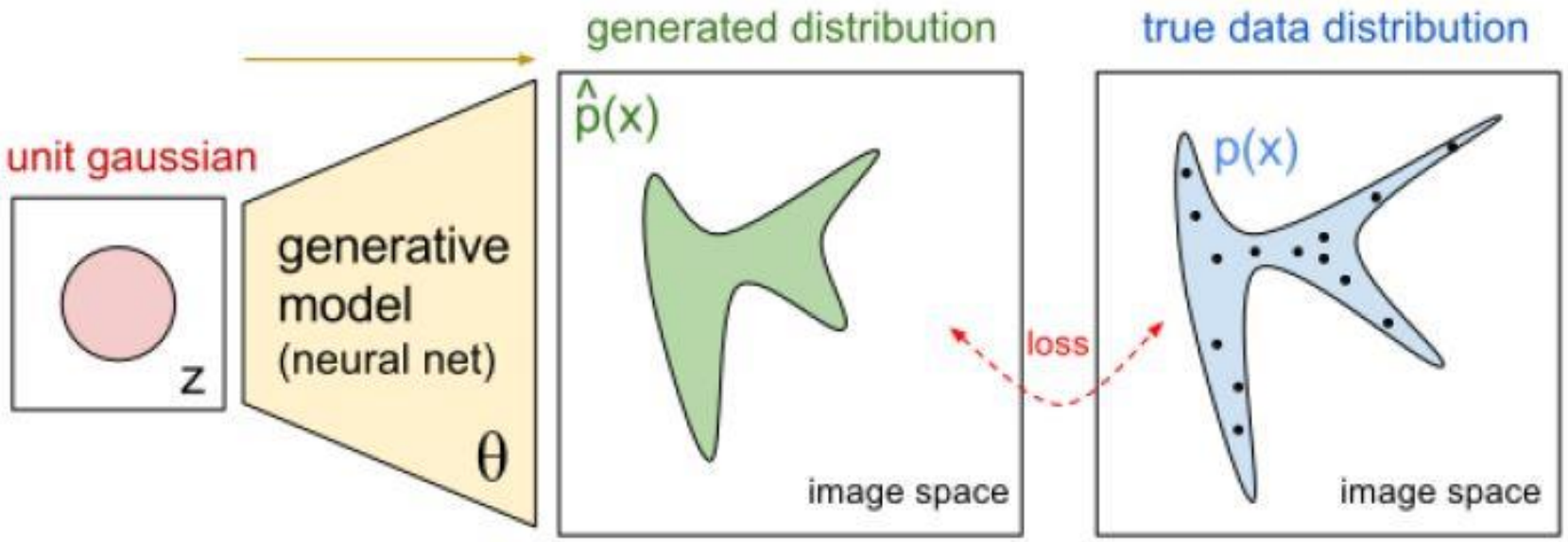
$$J^{(D)}\left(\theta^{(D)}, \theta^{(G)}\right) = -\frac{1}{2}E_{x\sim p_{data}} logD(x) - \frac{1}{2}E_z \log(1 - D(G(z)))$$

- Optimal discriminator:

$$D^*(x) = \frac{p_{data}(x)}{p_{model}(x) + p_{data}(x)}$$



GAN – Choice of divergence



- By varying θ we can make P_θ distribution arbitrarily "close" to P_{data}
- How to define "close"?

GAN – Choice of divergence

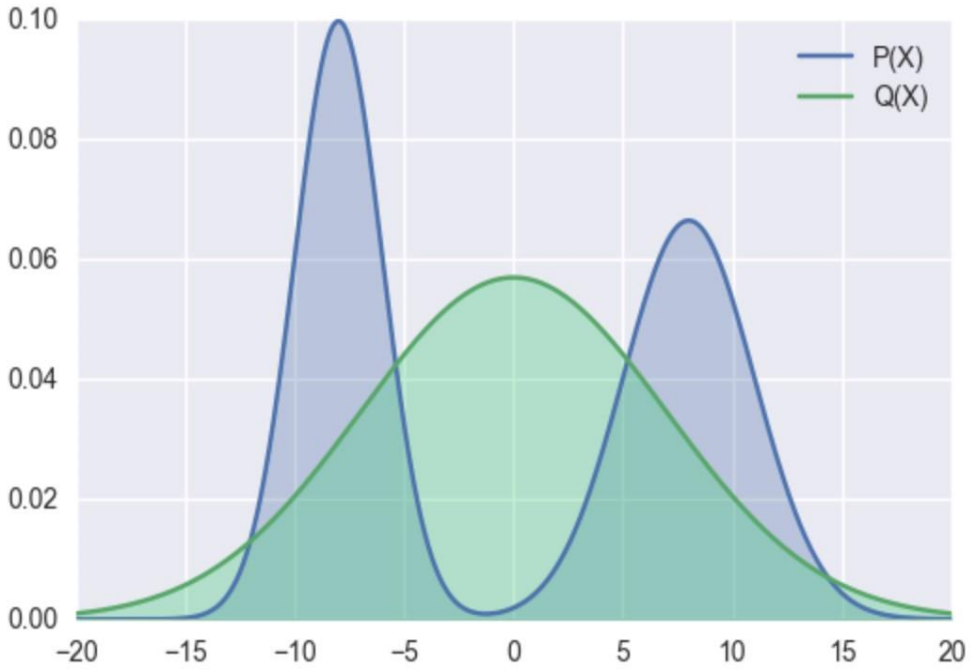
- Kullback-Leibler divergence:

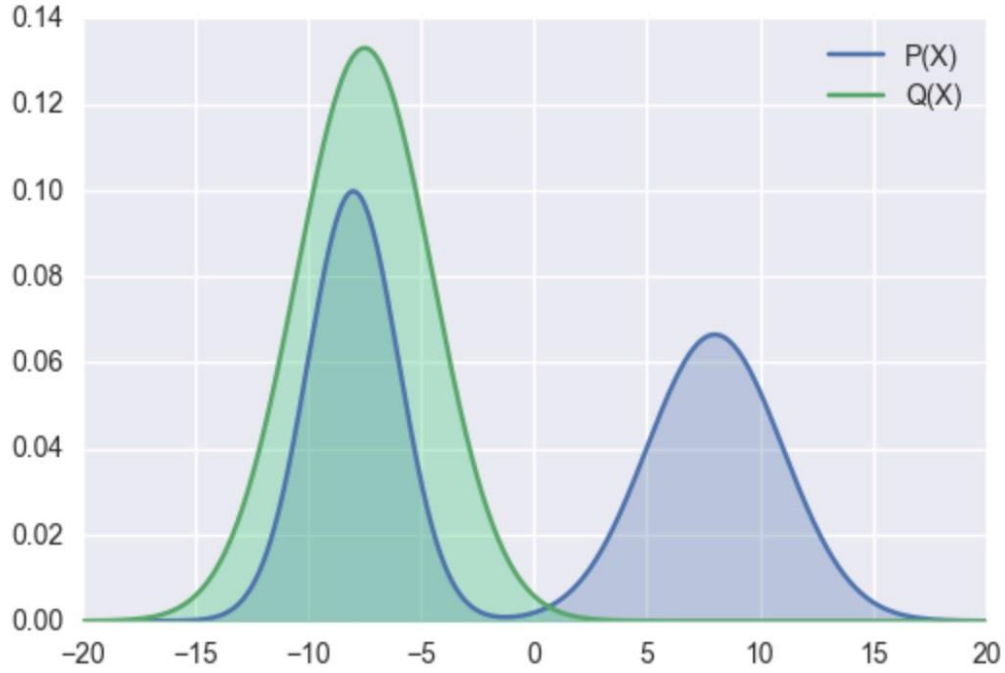
$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}[log\frac{\mathbb{P}(x)}{\mathbb{Q}(x)}]$$

- Jensen Shannon divergence:

$$JS(\mathbb{P}||\mathbb{Q}) = \frac{1}{2}KL(\mathbb{P}\,||\frac{1}{2}(\mathbb{P}+\mathbb{Q})) + \frac{1}{2}KL(\mathbb{Q}||\frac{1}{2}(\mathbb{P}+\mathbb{Q}))$$

GAN – Choice of divergence





$$KL(\mathbb{P}||\mathbb{Q}) = \int \mathbb{P}(x) log \frac{\mathbb{P}(x)}{\mathbb{Q}(x)}$$

$$KL(\mathbb{Q}||\mathbb{P}) = \int \mathbb{Q}(x) log \frac{\mathbb{Q}(x)}{\mathbb{P}(x)}$$

Zero-sum game

- N players
- Payoff matrix

Red \ Blue	A	B	C
1	-30 30	10 -10	-20 20
2	10 -10	-20 20	20 -20

A zero-sum game

- Nash equilibrium of a game

Generator cost function

$$J^{(G)}\left(\theta^{(D)}, \theta^{(G)}\right) = -J^{(D)}\left(\theta^{(D)}, \theta^{(G)}\right)$$

- Minimax game:

$$V(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}\left(\theta^{(D)}, \theta^{(G)}\right)$$

- Solution:

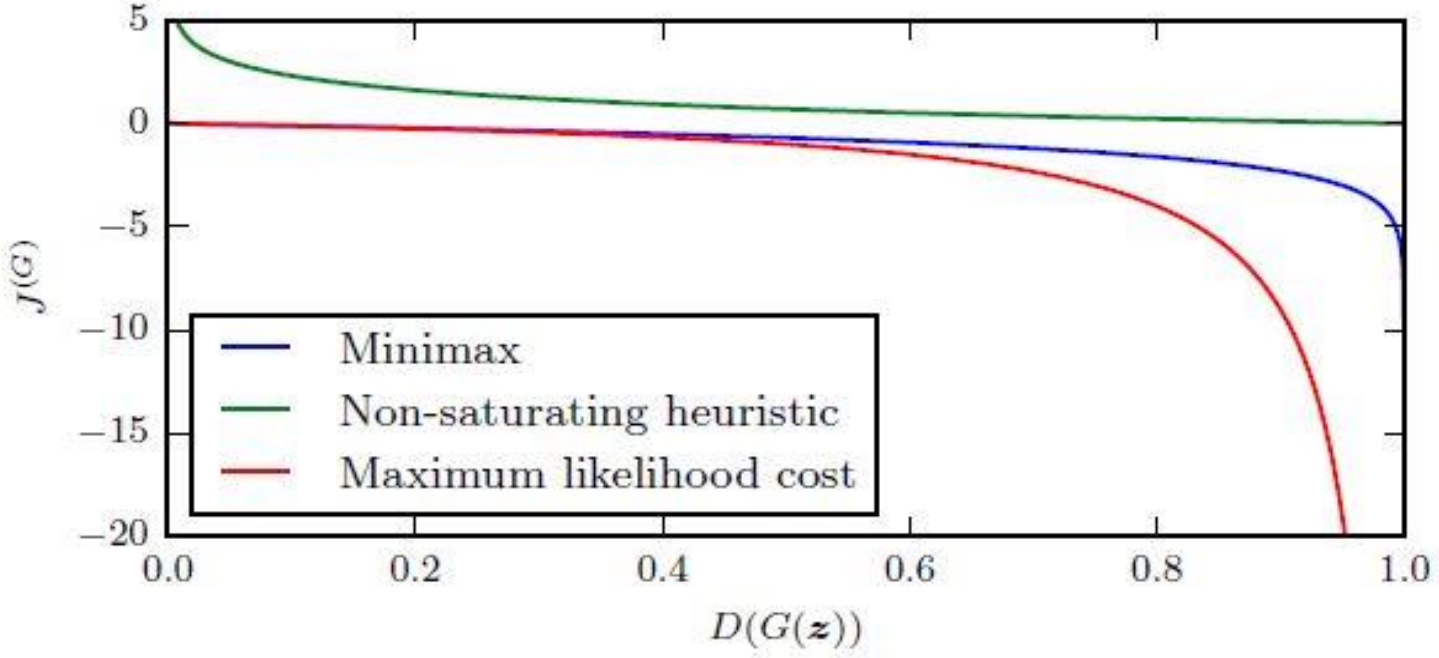
$$\theta^{(D)*} = \underset{\theta^{(G)}}{argmin} \max_{\theta^{(D)}} V(\theta^{(D)}, \theta^{(G)})$$

In practice:

$$J^{(G)}\left(\theta^{(D)}, \theta^{(G)}\right) = -\frac{1}{2} E_z \log(D(G(z)))$$

Generator Cost function

- Minimax: $J^{(G)} = \frac{1}{2} E_{x \sim p_{data}} logD(x) + \frac{1}{2} E_z \log(1 - D(G(z)))$
- Heuristic: $J^{(G)} = -\frac{1}{2} E_z \log(D(G(z)))$
- MLE: $J^{(G)} = -\frac{1}{2} E_z e^{(\sigma^{-1}(D(G(z))))}$



GAN - Training

- Simultaneous stochastic gradient descent
- Discriminator updates $\theta^{(D)}$ to reduce $J^{(D)}$
- Generator updates $\theta^{(G)}$ to reduce $J^{(G)}$
- $J^{(G)}$ does not make reference to the training data directly

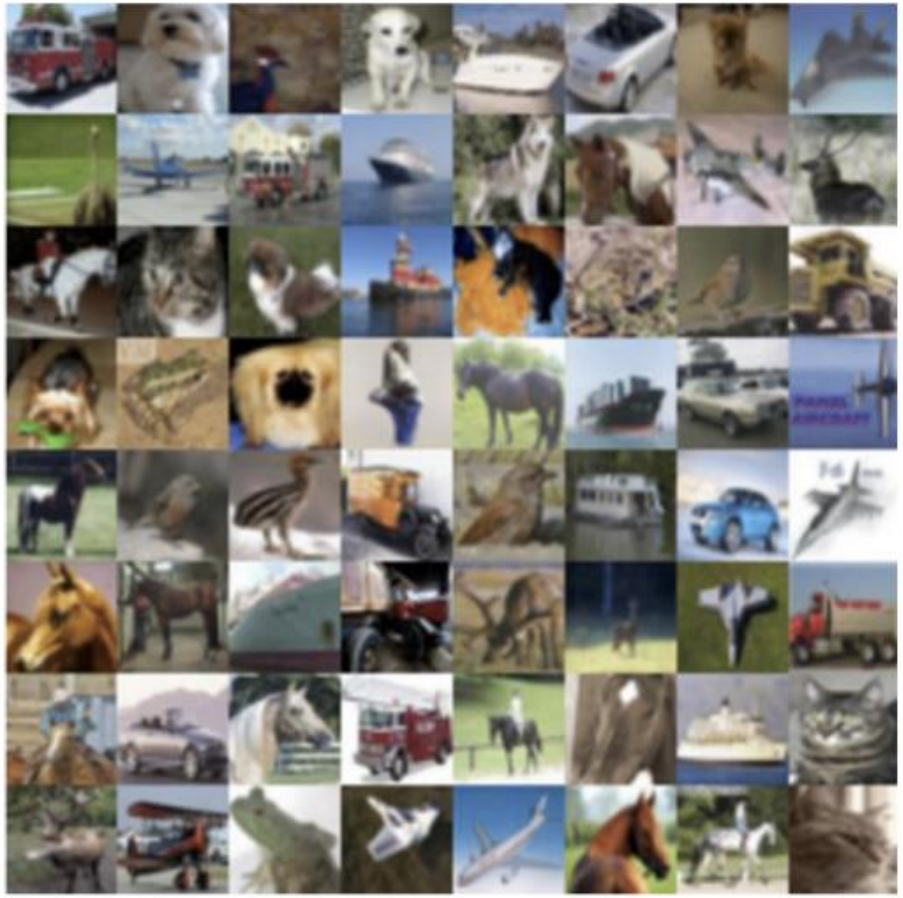
GAN - Tricks

- $x \in [-1, 1]$
- Using labels
- One-side label smoothing
- Batch normalization
- Running more steps of one player

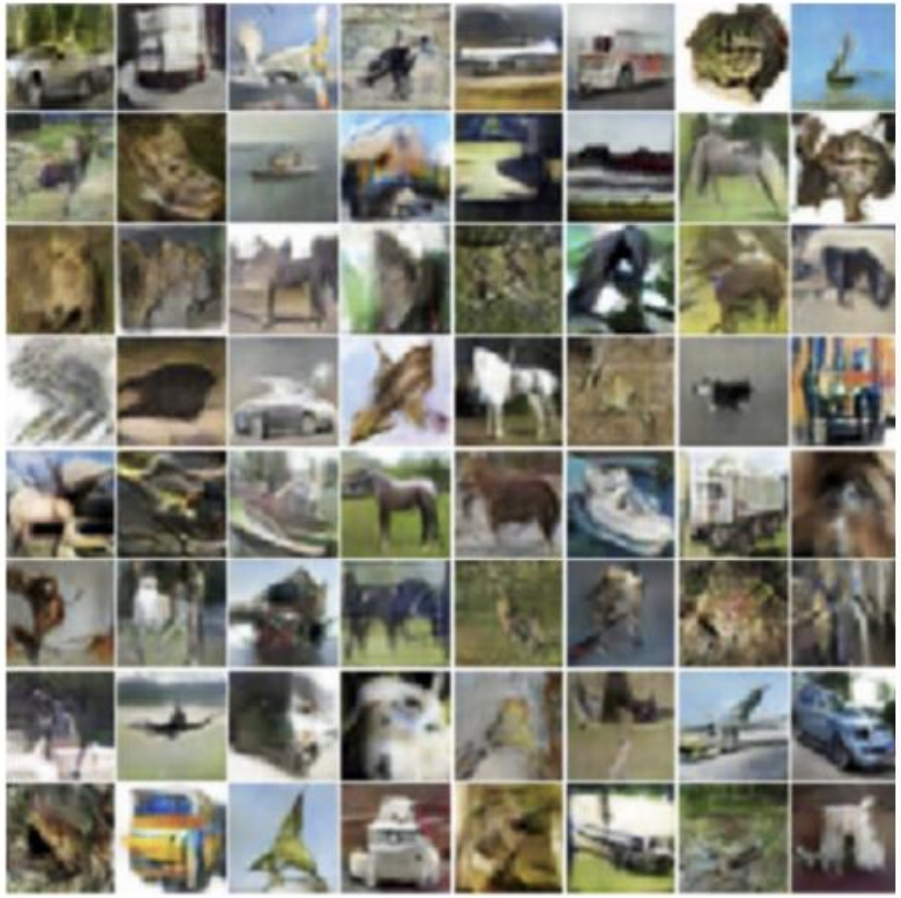
Generative Adversarial Networks

- Advantages:
  - Sample generation in a single step (does not depend on dimensionality of X)
  - Does not use Markov chain
  - Does not use variational inference
  - Generation function has very few restrictions
- Disadvantages:
  - Hard to train

GAN - results

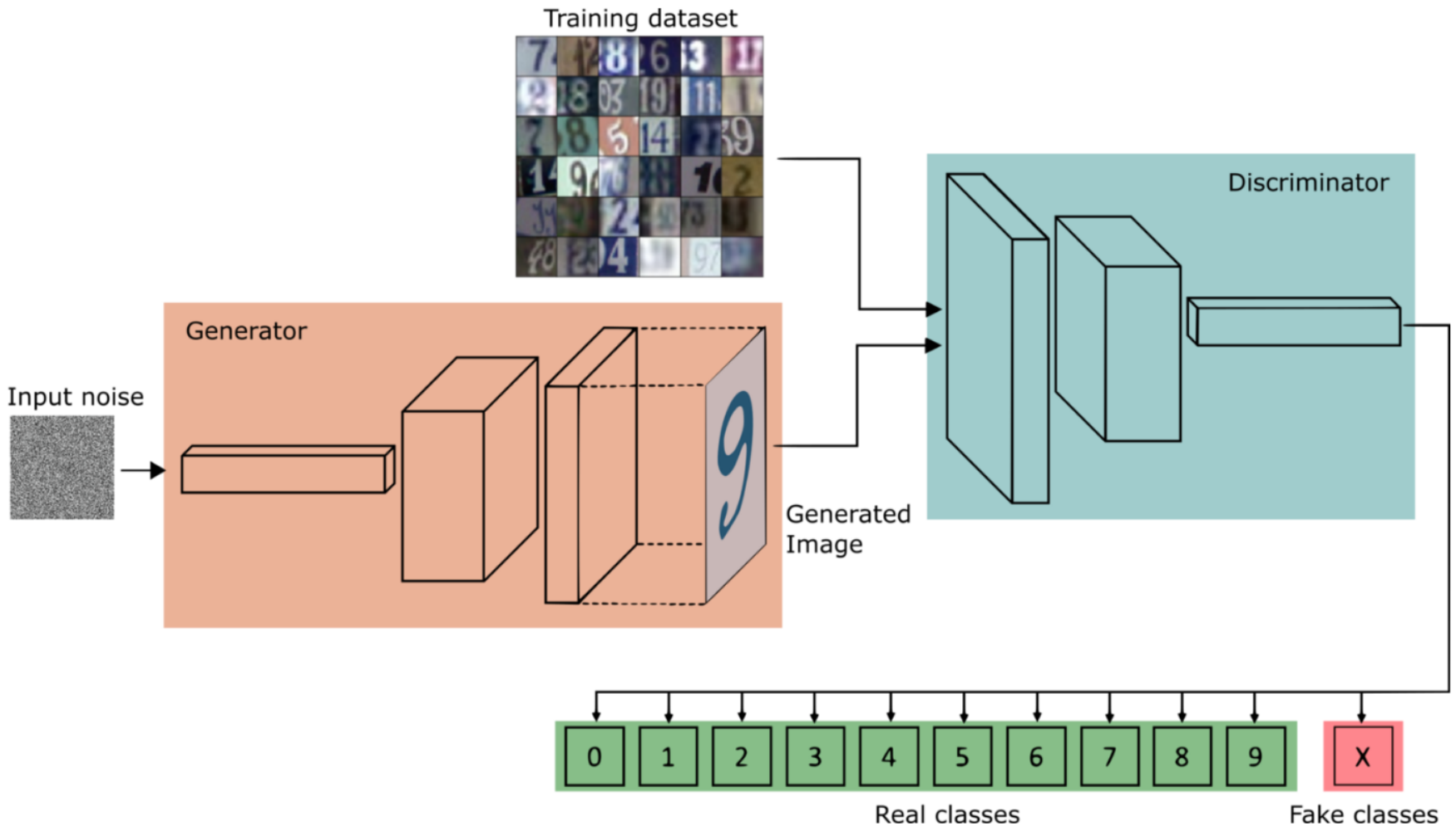


Training Data

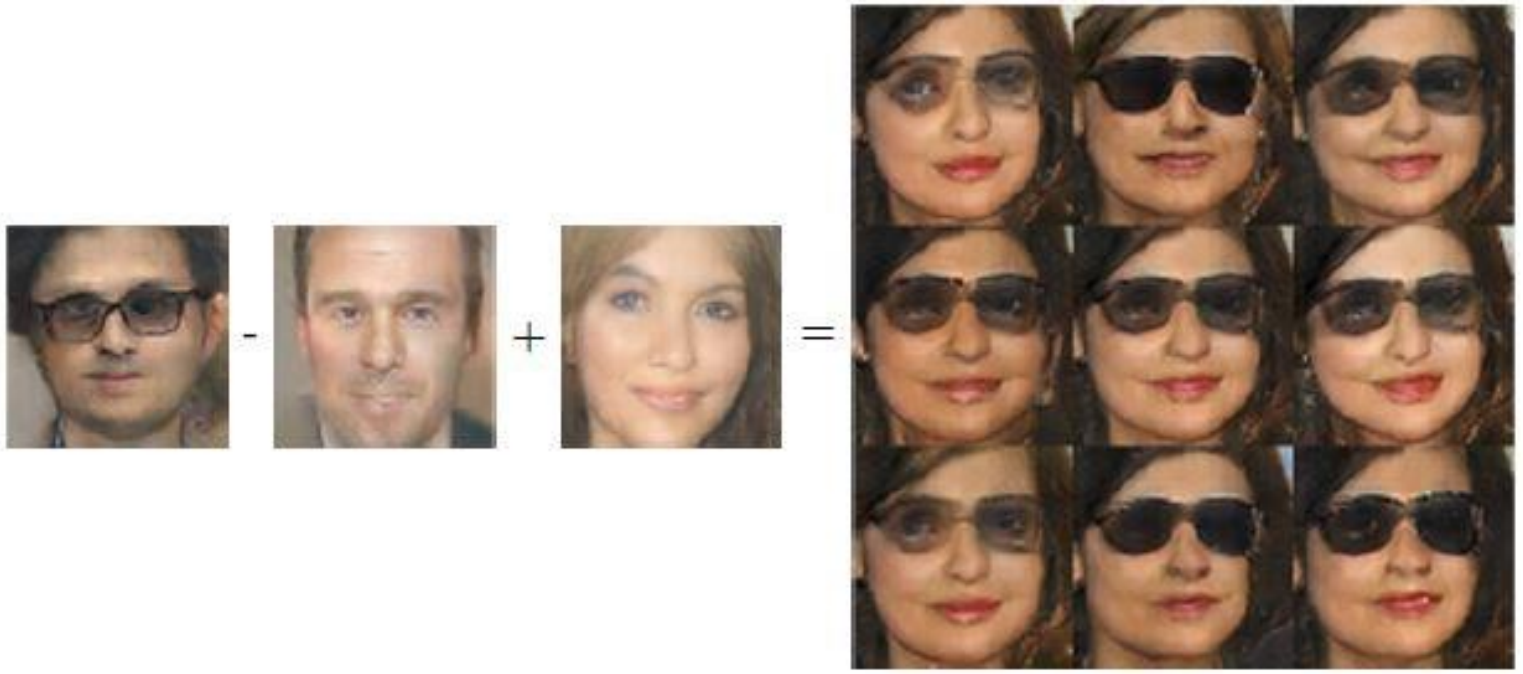


Samples

Semi-supervised learning with GAN



GAN – Latent Space Arithmetic



GAN – another perspective

- Cooperative instead of adversarial
- Learn cost function
- Reinforcement Learning
- Actor-Critic
- …

THANKS!