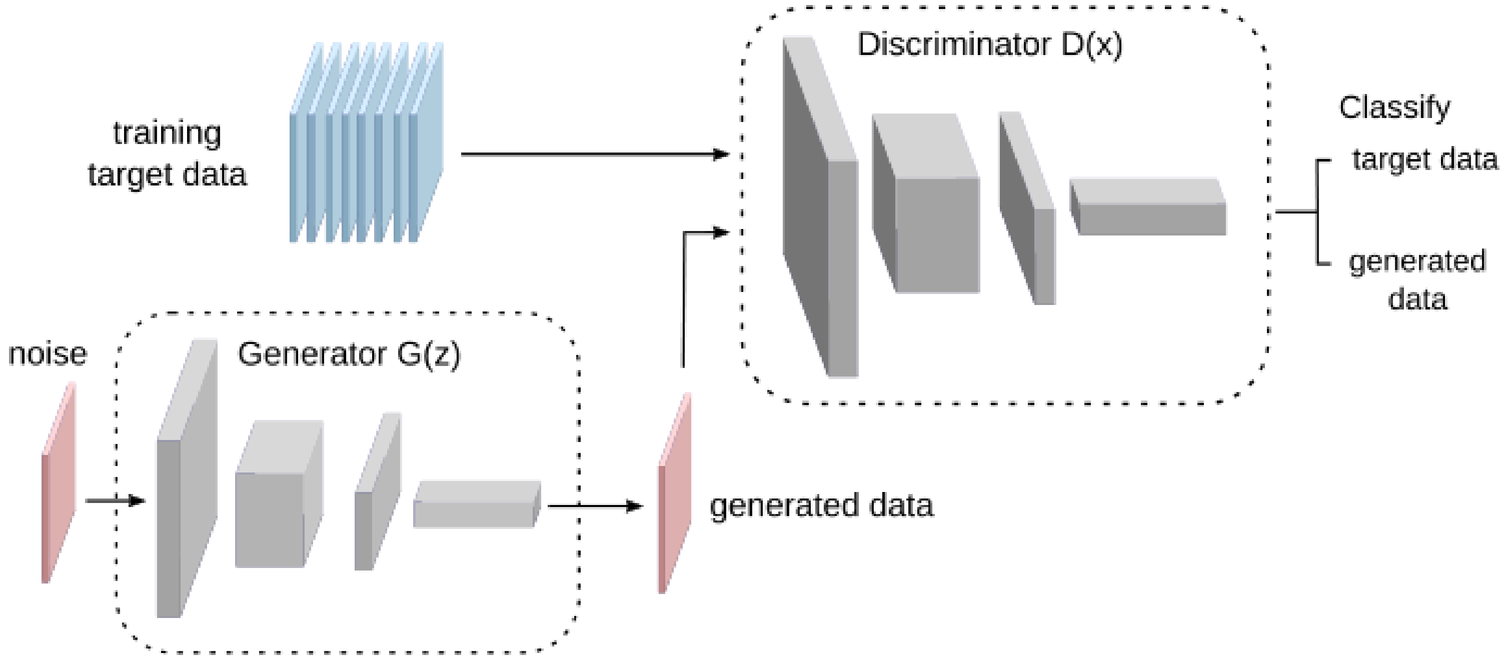
Introduction to Machine Learning Foundations and Applications

Paul J. Atzberger
University of California Santa Barbara

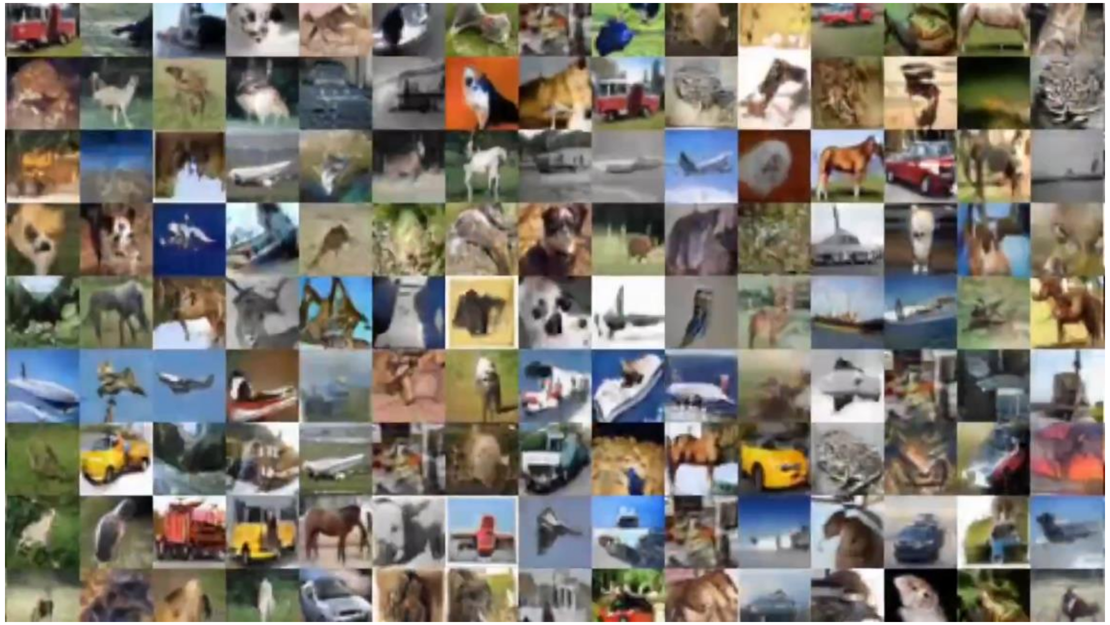
Generative Adversarial Networks (GANs)

Paul J. Atzberger



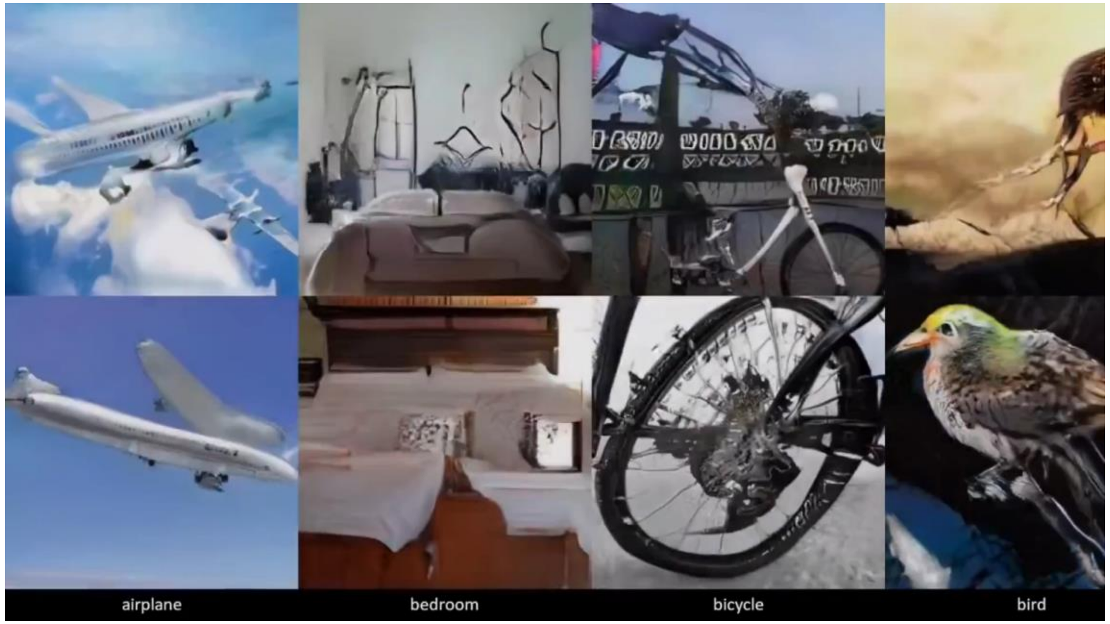
Motivations: Image Generation

GANs: CIFAR-10, 32x32



Karras 2018

GANs: LSUN, 256x256



Karras 2018

CycleGANs

Zhu 2018

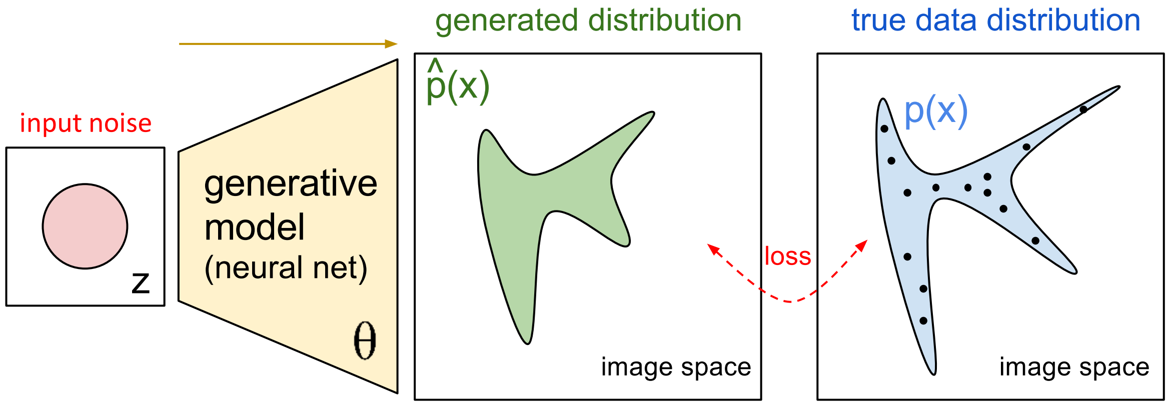
GANs Celeb-HQ

Many other applications…

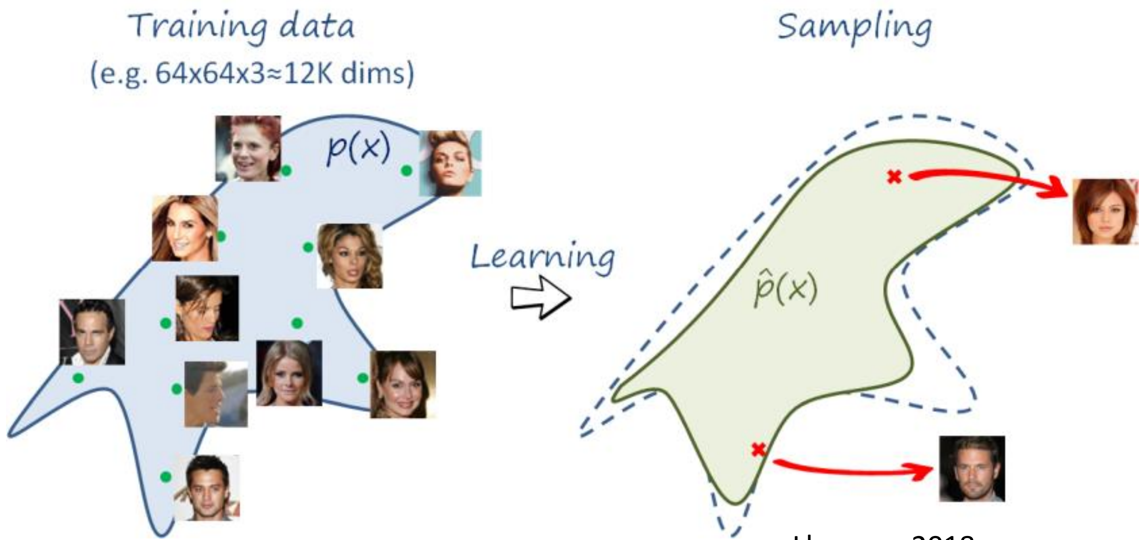
Motivations

Manifold-like structures in high dimensional spaces (natural images, audio, physical fields, PDE solutions).

Challenge: How to learn high dimensional probability distributions, generators G(z) for sampling?



open.ai

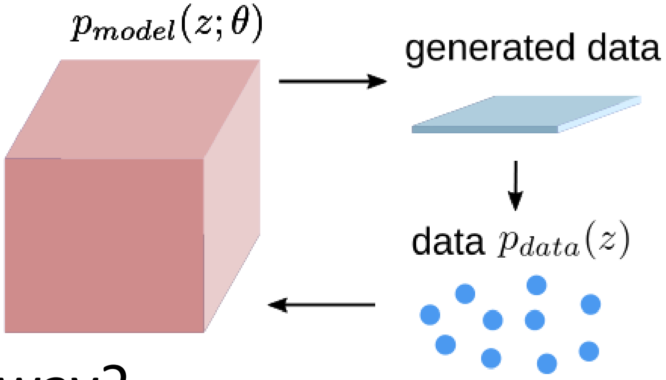


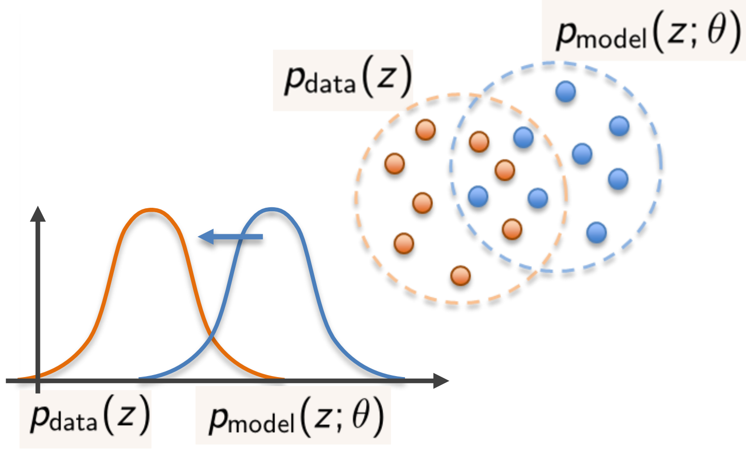
Lherranz 2018

Generative Modeling

Approaches for learning models:

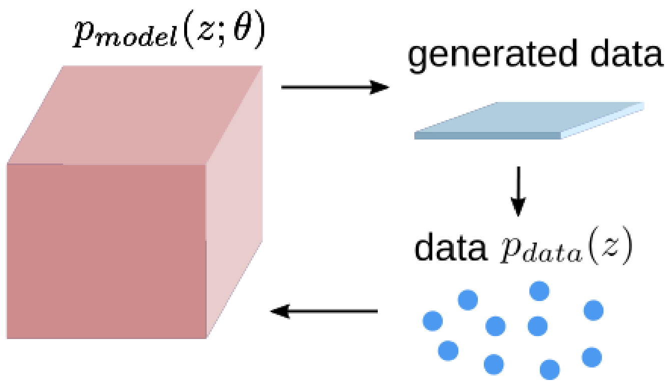
- Bayesian Methods
- Maximum Likelihood Estimation (MLE)
- and many more…

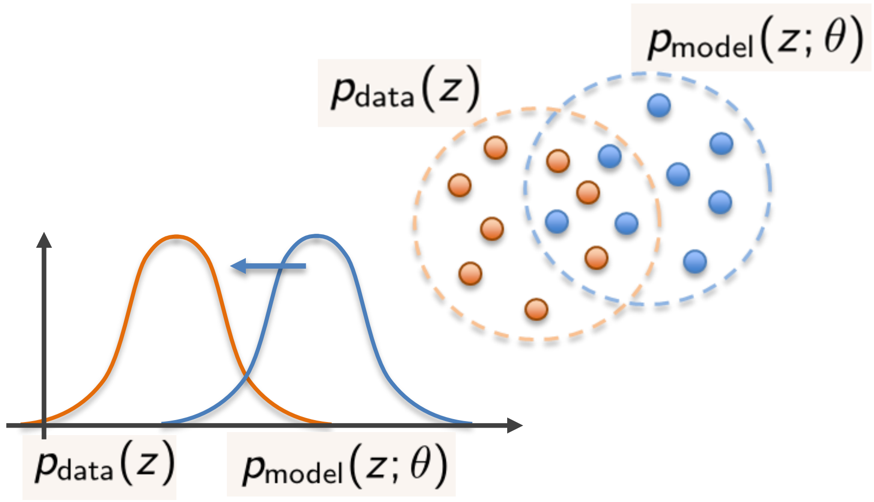




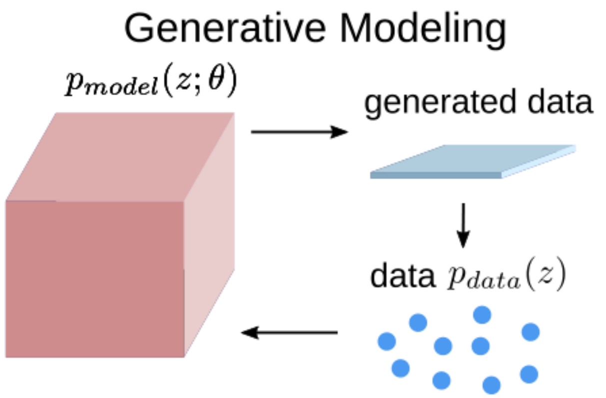
Challenge: How to do this in a tractable way?

Generative Modeling





Generative Models



Goal

Learn a model distribution $p_{\text{model}}(z;\theta)$ approximating the data distribution $p_{\text{data}}(z)$.

Classification: For input x assign the class $y^* = \text{arg-max}_y\, p_{\text{model}}(y|x;\theta)$ (approximates the Bayes classifier).

For $z = (x, y)$ this is typically broken down using $p(x, y) = p(y|x)p(x)$. The model distribution with parameter θ is then $p_{\text{model}}(x, y;\theta) = p(y|x;\theta)p_{\text{data}}(x)$, where $p_{\text{data}}(x) = \int p_{\text{data}}(x, y)d\mu_y$.

Optimization Formulation

For an objective function $J[p_{\text{model},\theta}, p_{\text{data}}]$, find

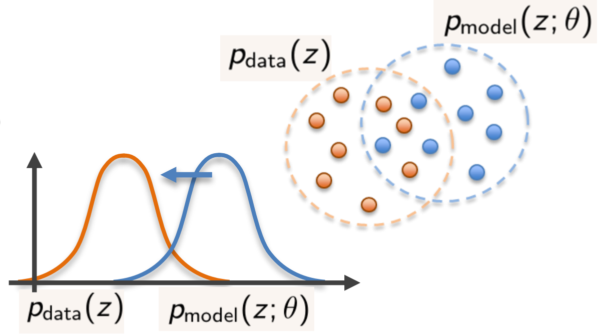
$$\theta^* = \text{arg-min}_\theta J[p_{\text{model},\theta}, p_{\text{data}}].$$

Maximum Likelihood is a widely used approach, corresponds to the objective

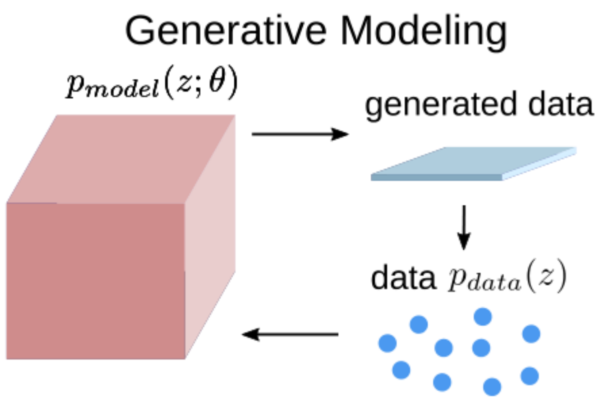
$$J[\theta] = J[p_{\text{model},\theta}, p_{\text{data}}] = -\mathbb{E}_{(x,y)\sim p_{\text{data}_-}}[log\left(p_{\text{model}}(x, y;\theta)\right)].$$

This is equivalent to minimizing the **Kullback-Leibler Divergence** D_{KL} with

$$J[\theta] = D_{KL}\left(p_{\text{data}} \| p_{\text{model},\theta}\right).$$



Generative Models



In practice: We do not have data distribution but only training samples $\{z_i\}_{i=1}^m$.
We construct the **empirical data distribution**

$$\tilde{p}_{\text{data}}(z) = \frac{1}{m}\sum_{i=1}^{m}\delta(z - z_i).$$

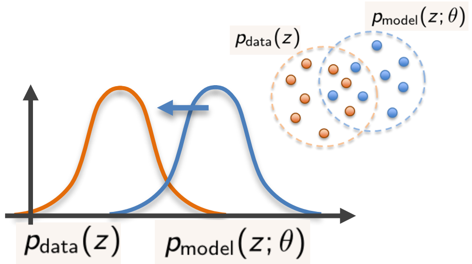
Goal (empirical distribution)

Learn a model distribution $p_{\text{model}}(z;\theta)$ approximating the data distribution $\tilde{p}_{\text{data}}(z)$.
Find

$$\theta^* = \text{arg-min}_\theta J[p_{\text{model},\theta}, \tilde{p}_{\text{data}}].$$

Maximum Likelihood (empirical data distribution): For $\tilde{p}_{\text{data}}$ becomes

$$J[\theta] = -\mathbb{E}_{(x,y)\sim\tilde{p}_{\text{data}}}\left[log\left(p_{\text{model}}(x,y;\theta)\right)\right] = -\frac{1}{m}\sum_{i=1}^{m} log\left(p_{\text{model}}(x_i,y_i;\theta)\right).$$



In practice: p_{data} often high dimensional requiring rich class of $p_{\text{model},\theta}$. Above requires some way to compute the log-likelihood function. To get good gradient need overlap of distributions (absolute continuity). Often difficult to compute functional form of p_{model}. Need for alternatives.

Generative Adversarial Networks (GANs)


Ian Goodfellow

Goodfellow 2014: Generative Adversarial Networks (GANs).

GANs: Utilizes deep learning with DNNs for generators $G(z;\theta)$.

Key idea: Use properties of supervised learning and generalization behaviors of classifiers D to train generators $G(z;\theta)$.

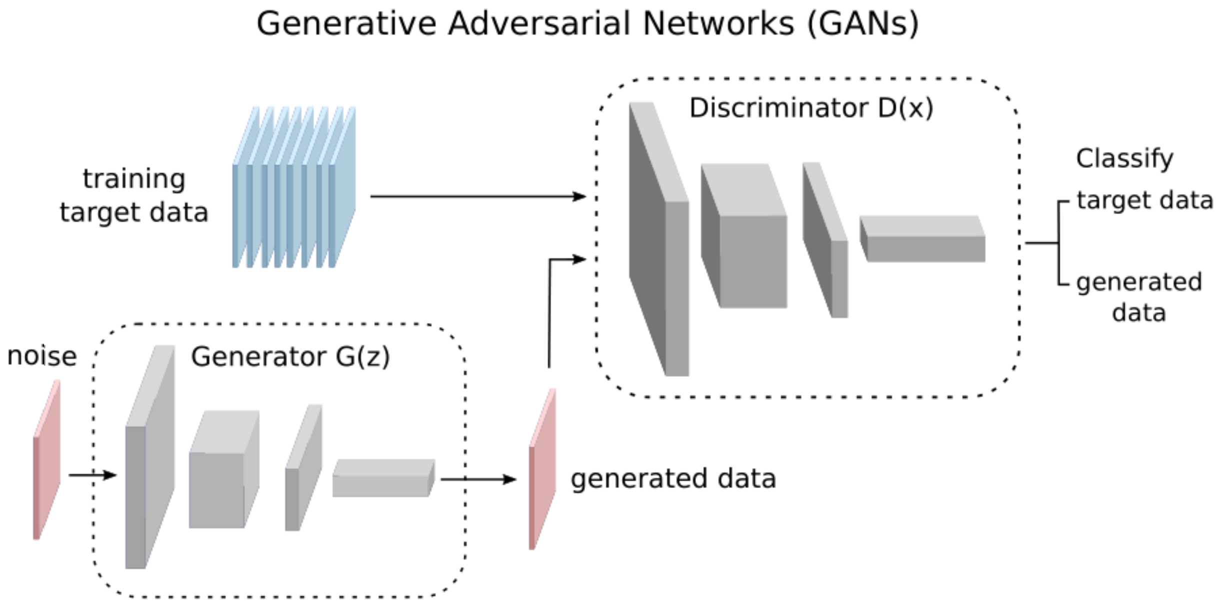
Synthetic data distribution mixture of "real" and "fake" samples.

Two player-game:
- (i) D aims to classify x as "real" or "fake."
- (ii) G aims to generate "fake" samples so well D can not tell difference.

Successes: image generation, video augmentation, and other applications. Challenges (counting, spatial alignment,…)


early work on GANs

Generative Adversarial Networks (GANs)



Learn generative models using:

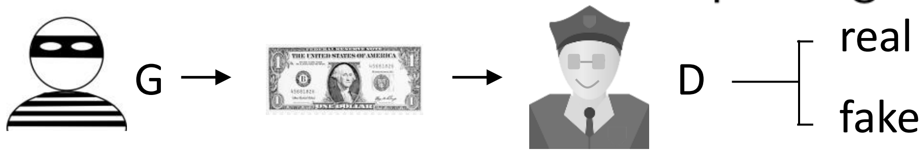
GANs

Generator G: samples $x \sim p_{\text{model}}(x;\theta^G)$.

Discriminator $D(x)$: binary classifier for if
  (i) input x is sampled from $p_{\text{data}}(x)$, or
  (ii) generated from $p_{\text{model}}(x;\theta^G)$.

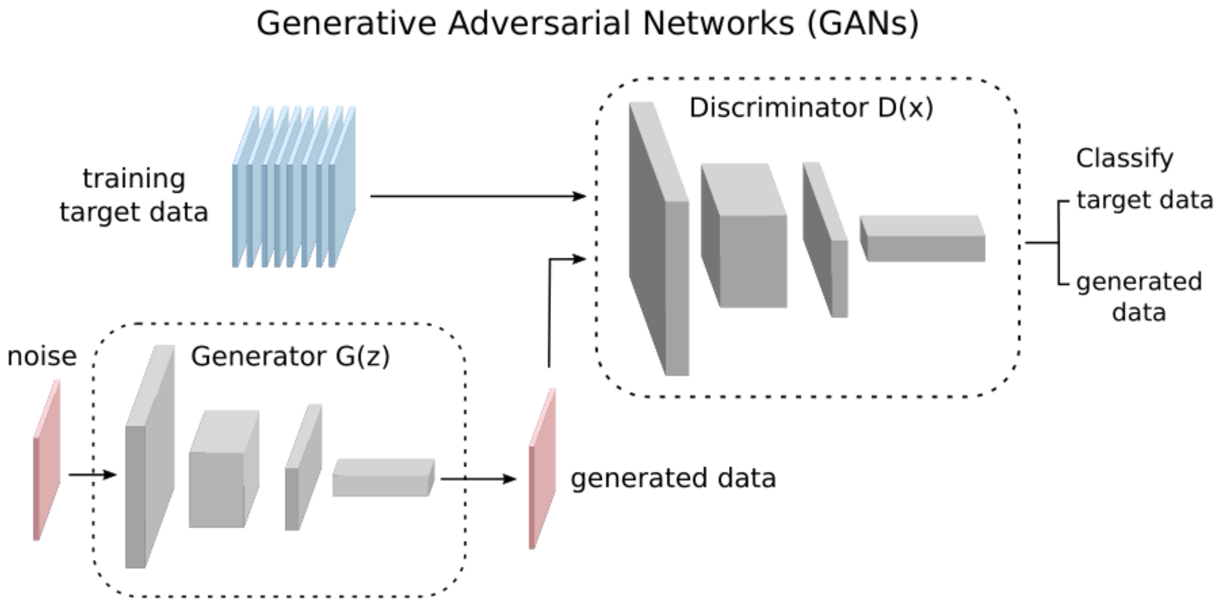
Remark: Two-player game with G generating samples so well that the discriminator D can not distinguish from samples of the data distribution.

Remark: The objective is similar to a counterfeiter G printing money so that the police D can not tell if the bills are real or fake.



Key Idea: Replaces the problematic calculation using D_{KL}-objective by instead using the discriminator D to serve to drive the model distribution p_{model} toward p_{data}. Leverages capabilities of supervised learning methods.

Generative Adversarial Networks (GANs)



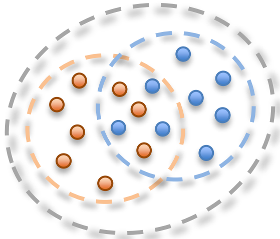
Learn generative models using:

GANs

Generator G: samples $\mathrm{x} \sim p_{\text{model}}(x;\theta^G)$.

Discriminator $D(\mathrm{x})$: binary classifier for if
  (i) input x is sampled from $p_{\text{data}}(\mathrm{x})$, or
  (ii) generated from $p_{\text{model}}(x;\theta^G)$.

Synthetic Labeled Data: Create a synthetic labeled set of data as follows:
  (i) with probability $1/2$ sample x from the data distribution $p_{\text{data}}(\mathrm{x})$ and assign the label 1,
  (ii) with probability $1/2$ sample x from the model distribution $p_{\text{model}}(x;\theta^G)$ and assign the label 0.

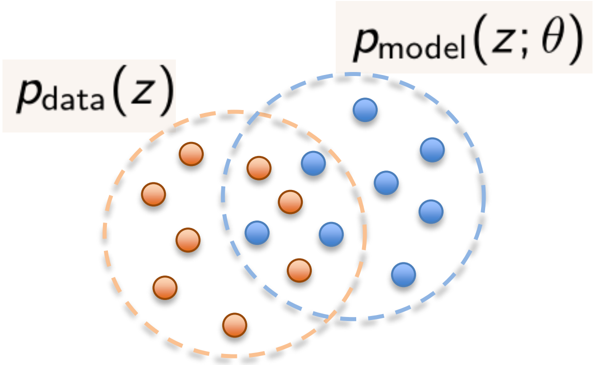


Binary Classifier: Consider generative classifier that assigns the probability $D(x)$ that x was sampled from the data distribution. Then $1 - D(x)$ is the assigned probability that x was generated from the model distribution.

$$D(\mathrm{x}) = p_D(y=1|\mathrm{x}) \approx \Pr\{Y=1|X=x\}, \qquad 1 - D(\mathrm{x}) = p_D(y=0|\mathrm{x}) \approx \Pr\{Y=0|X=x\}.$$

Classification: For input x assign the class $y^* = \text{arg-max}_y \Pr\{Y=y|X=x\}$ (approximates Bayes classifier).

Generative Adversarial Networks (GANs)

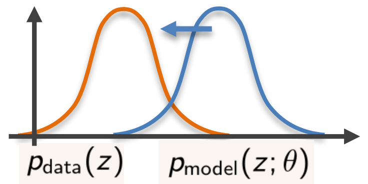


Synthetic Labeled Data: This has the data distribution $p_{\text{synth-l}}$ given by

$$p_{\text{synth-l}}(\mathrm{x}, y) = 1_{y=1}\frac{1}{2}p_{\text{data}}(\mathrm{x}, y) + 1_{y=0}\frac{1}{2}p_{\text{model}}(\mathrm{x}, y; \theta^G).$$

For this distribution we have

$$\Pr\{Y = 1|X = x\} = \frac{p_{\text{data}}(\mathrm{x})}{p_{\text{data}}(\mathrm{x}) + p_{\text{model}}(\mathrm{x})}.$$



Thus, $D(\mathrm{x}) = p_{\text{data}}(\mathrm{x})/\left(p_{\text{data}}(\mathrm{x}) + p_{\text{model}}(\mathrm{x})\right)$ would give us the best possible discriminator (Bayes classifier).

Remark: If we were successful in getting our model distribution to exactly match the data distribution then $p_{\text{model}} = p_{\text{data}}$ and $D(\mathrm{x}) = 1/2$.

Remark: When $D(\mathrm{x}) = 1/2$ the discriminator can not tell if the sample was more likely to come from the dat distribution or from the generator. For generative discriminator, let $p_D(\mathrm{x}, y; \theta^D) := p_D(y|\mathrm{x}; \theta^D)p_{\text{synth-l}}(\mathrm{x})$.

We aim to achieve this outcome by learning simultaneously θ^D for the optimal discriminator D and learning θ^G for an optimal generator G. Let $C(\theta^G)$ term be entropy of the synthetic distribution.

We formulate the classification problem for D using cross-entropy loss with objective function

$$\hat{J}^D(\theta^D, \theta^G) = -\mathbb{E}_{\mathrm{x},y\sim p_{\text{synth-l},\,\theta^G}}\left[\log p_D(\mathrm{x}, y; \theta^D)\right] = -\frac{1}{2}\mathbb{E}_{\mathrm{x}\sim p_{\text{data}}}\left[\log\left(D(\mathrm{x})\right)\right] - \frac{1}{2}\mathbb{E}_{\mathrm{x}\sim p_{\text{model},\theta^G}}\left[\log\left(1 - D(\mathrm{x})\right)\right] + C(\theta^G).$$

Generative Adversarial Networks (GANs)

Discriminator D

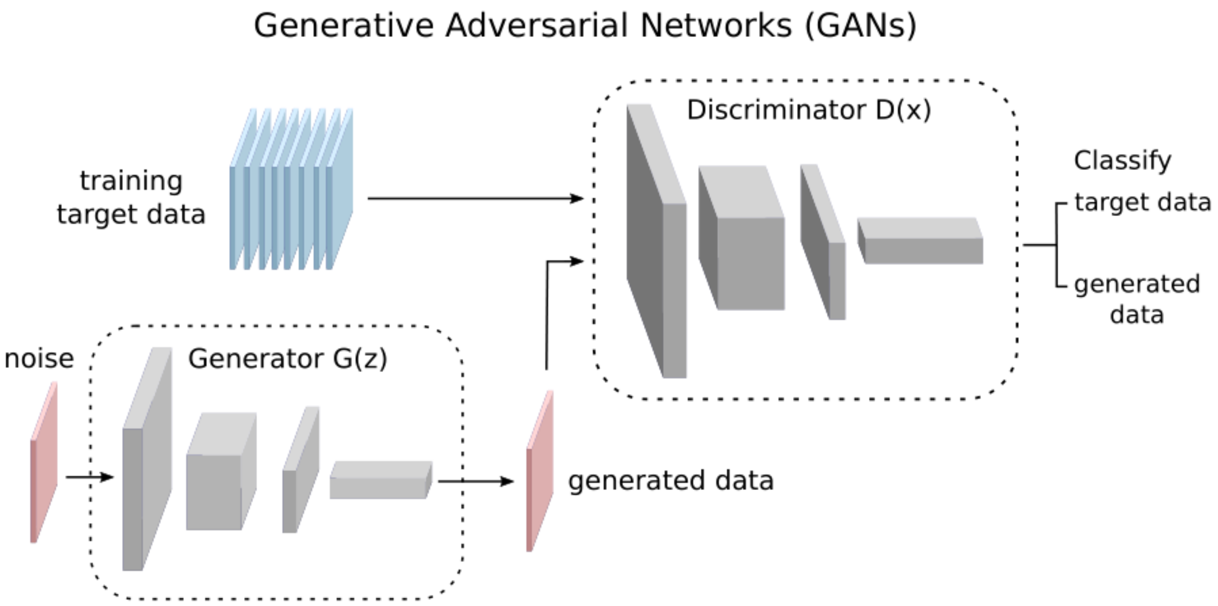
Find $\theta^{D*} = \text{arg-min}\, J^D(\theta^D, \theta^G)$ with

$$J^D(\theta^D,\theta^G) = -\mathbb{E}_{\mathrm{x},y\sim p_{\text{synth-I},\,\theta^G}}\left[\log p_D(y|\mathrm{x};\theta^D)\right] = -\frac{1}{2}\mathbb{E}_{\mathrm{x}\sim p_{\text{data}}}\left[\log\left(D(\mathrm{x})\right)\right] - \frac{1}{2}\mathbb{E}_{\mathrm{x}\sim p_{\text{model},\theta^G}}\left[\log\left(1-D(\mathrm{x})\right)\right].$$

Entropy term $C(\theta^G)$ not used. Generator G aims for distribution close to data distribution.

Generator G: Approach I

Find $\theta^{G*} = \text{arg-max}\, J^G(\theta^D, \theta^G)$ with $J^G = J^D$.



Generative Adversarial Networks (GANs)

Discriminator D

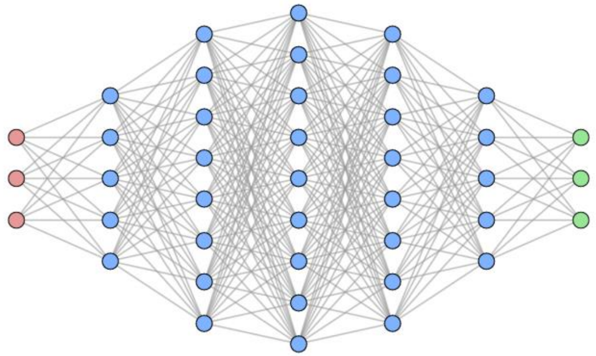
Find $\theta^{D*} = \text{arg-min}\, J^D(\theta^D, \theta^G)$ with

$$J^D(\theta^D, \theta^G) = -\mathbb{E}_{\mathrm{x},y\sim p_{\text{synth-I},\,\theta^G}}\left[\log p_D(y|\mathrm{x};\theta^D)\right] = -\frac{1}{2}\mathbb{E}_{\mathrm{x}\sim p_{\text{data}}}\left[\log\left(D(\mathrm{x})\right)\right] - \frac{1}{2}\mathbb{E}_{\mathrm{x}\sim p_{\text{model},\theta^G}}\left[\log\left(1 - D(\mathrm{x})\right)\right].$$

Entropy term $C(\theta^G)$ not used. Generator G aims for distribution close to data distribution.

Generator G: Approach I

Find $\theta^{G*} = \text{arg-max}\, J^G(\theta^D, \theta^G)$ with $J^G = J^D$.



Deep Neural Network

This gives a zero-sum game, so has valuation function $V(\theta^D, \theta^G) = J^D = J^G$.

Remark: Deep Neural Networks will be used to learn $D(\mathrm{x};\theta^D)$ and $G(\mathrm{z};\theta^G)$.

Remark: Notice the objective functions now no longer require evaluating the expression of the model probability distribution. They only require expectations, which can be approximated from sampling $\mathrm{x} \sim p_{\text{model}}$.

We use the **reparameterization technique** to generate $\mathrm{x} \sim p_{\text{model}}$ using $\mathrm{x} = G(\mathrm{z};\theta^G)$, where $\mathrm{z} \sim \hat{p}_{\text{model}}$ with $\hat{p}_{\text{model}}$ an easy to generate distribution. The challenge is shifted to learning the function $G(\mathrm{z};\theta^G)$.

Generative Adversarial Networks (GANs)

Discriminator D

Find $\theta^{D*} = \text{arg-min}\, J^D(\theta^D, \theta^G)$ with

$$J^D(\theta^D, \theta^G) = -\mathbb{E}_{x,y\sim p_{\text{synth-I},\ \theta^G}}\left[\log p_D(y|x;\theta^D)\right] = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}}\left[\log\left(D(x)\right)\right] - \frac{1}{2}\mathbb{E}_{x\sim p_{\text{model},\theta^G}}\left[\log\left(1 - D(x)\right)\right].$$

Entropy term $C(\theta^G)$ not used. Generator G aims for distribution close to data distribution.

Generator G: Approach I

Find $\theta^{G*} = \text{arg-max}\, J^G(\theta^D, \theta^G)$ with $J^G = J^D$.

Vanishing Gradient Issue: For bad generators the discriminator can become very good at just rejecting samples from the model distribution resulting in vanishing gradient in θ^G and no learning.

Alternative Formulation: We aim for generator to make the discriminator probability $D(x)$ as large as possible (hence fooling it). We use

Generator G: Approach II

Find $\theta^{G*} = \text{arg-max}\, J^G(\theta^D, \theta^G)$ with $J^G = \mathbb{E}_{z\sim p_{\text{model},\ \theta^G}}\left[\log\left(D(x;\theta^D)\right)\right]$.

Generative Adversarial Networks (GANs)

Discriminator D

Find $\theta^{D*} = \text{arg-min}\, J^D(\theta^D, \theta^G)$ with

$$J^D(\theta^D, \theta^G) = -\mathbb{E}_{x \sim p_{\text{synth-l},\, \theta^G}} \left[\log p_D(y|x;\theta^D)\right] = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}}\left[\log\left(D(x)\right)\right] - \frac{1}{2}\mathbb{E}_{x\sim p_{\text{model},\theta^G}}\left[\log\left(1 - D(x)\right)\right].$$

Generator G: Approach II

Find $\theta^{G*} = \text{arg-max}\, J^G(\theta^D, \theta^G)$ with $J^G = \mathbb{E}_{z\sim p_{\text{model},\, \theta^G}}\left[\log\left(D(x;\theta^D)\right)\right]$.

No longer a zero-sum game, the solution $(\theta^{D*}, \theta^{G*})$ now characterized as a Nash Equilibrium.

Training Protocol: Alternate minimizing discriminator objective with maximizing the generator objective.

Remark: This can result in oscillatory learning dynamics. Current area of research on best ways to address (likely this is application dependent).

JS-GANs: Jensen-Shannon Distance

Jensen-Shannon Distance

$$JS(p_{data}, p_{model}) = \frac{1}{2} KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_{\text{model}}}{2}\right) + \frac{1}{2} KL\left(p_{\text{model}} \middle\| \frac{p_{\text{data}} + p_{\text{model}}}{2}\right)$$

$JS(p,q) \geq 0$ and $JS(p,q) = 0 \;\Rightarrow p = q \;\; (a.s)$. $KL(p\|q) = \mathbb{E}_{x\sim p}\left[\log\left(\frac{p}{q}\right)\right]$.

The optimal discriminator is $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_{\text{model}}(x)}$. Substituting, we have

$$J^D(\theta^{D,*}, \theta^G) = -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}}\left[\log\left(\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}\right)\right] - \frac{1}{2}\mathbb{E}_{x\sim p_{\text{model},\theta^G}}\left[\log\left(\frac{p_{\text{model}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}\right)\right].$$

This gives

$$J^D(\theta^{D,*}, \theta^G) = -JS(p_{data}, p_{model,\theta^G}) + \log(2).$$

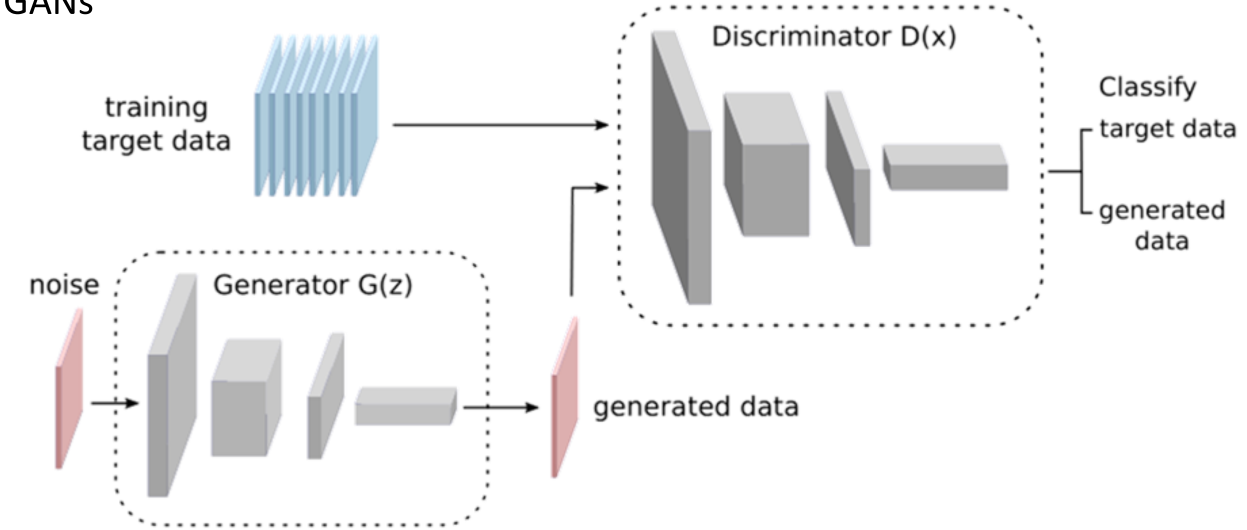
As a result, when $J^G = J^D$, we have $\theta^{G*} = \text{arg-max}_{\theta^G} J^D(\theta^{D,*}, \theta^G) = \text{arg-min}_{\theta^G} JS(p_{data}, p_{model,\theta^G})$.

Shows that original GANs with optimal discriminator $D^*(x)$ is equivalent to following gradients to minimize the JS-Distance between the model distribution p_{model} and p_{data}.

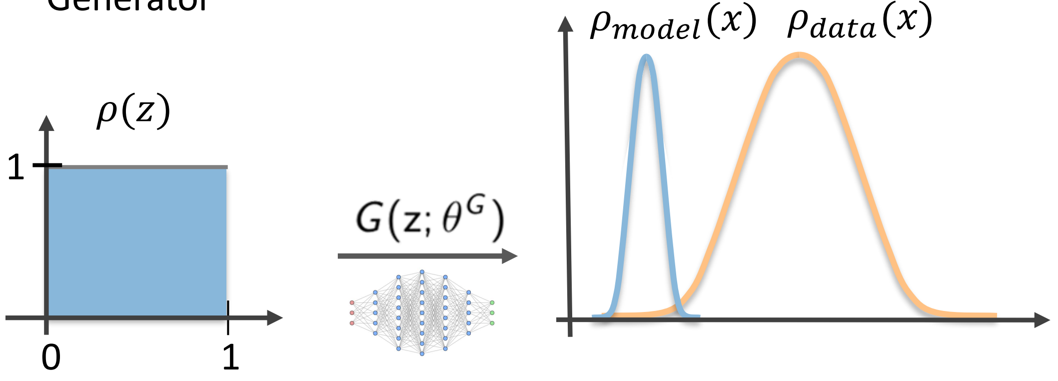
GANs have been successfully applied in many practical applications: Image Synthesis, Super-Resolution Imaging, Generative Art, Face and Video Synthesis. Other formulations of GANs (Wasserstein WGANs, E-GANs, etc...)

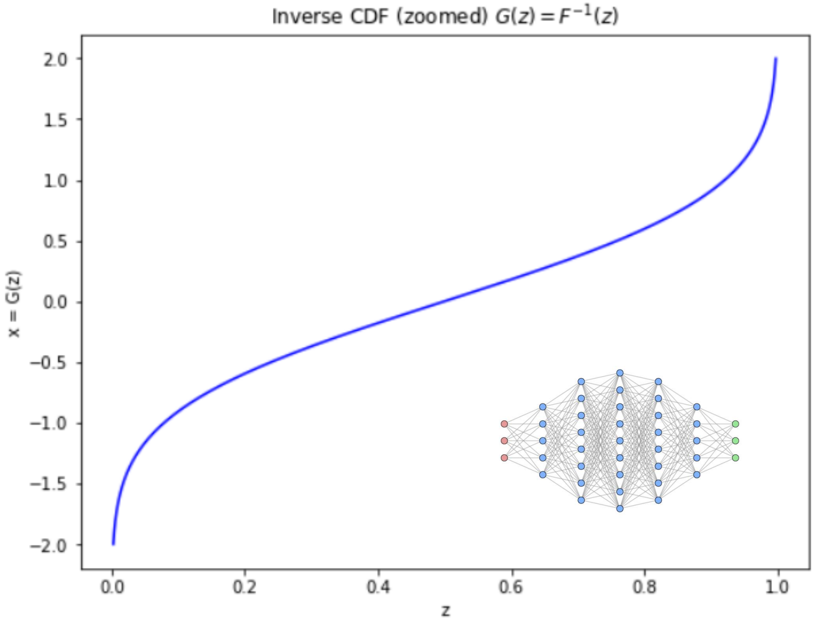
Example: Gaussian Target Distribution

GANs



Generator





<u>Task:</u> Use GANs to learn Gaussian target data distribution $\rho_{data}(x)$.

Generator → Approximated by Deep Neural Network (DNN) and SGD.

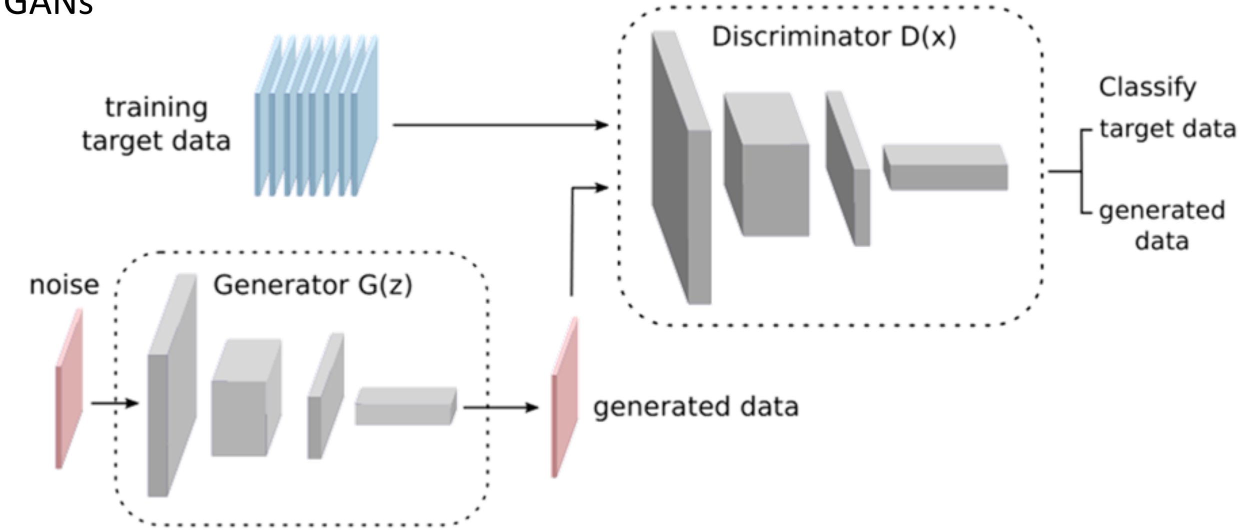
Training: Alternate between minimization for D(x) and maximization for G(z).

Remark: Cumulative Distribution Function (CDF) → Inverse gives a generator.

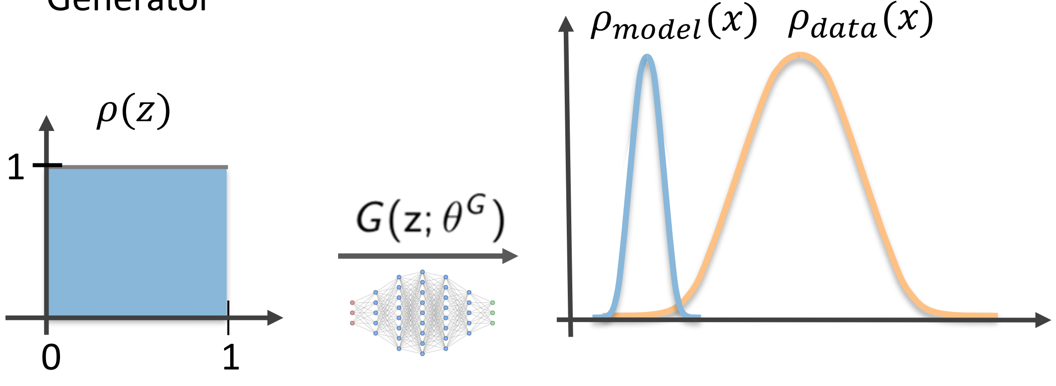
Remark: Gaussians this diverges to give small probability for tails. Noise sources type important consideration in practice.

Example: Gaussian Target Distribution

GANs

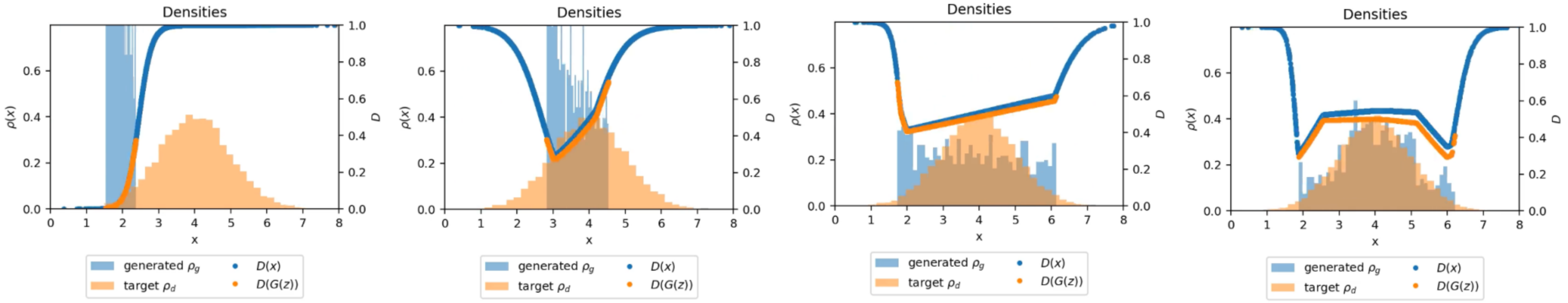


Generator

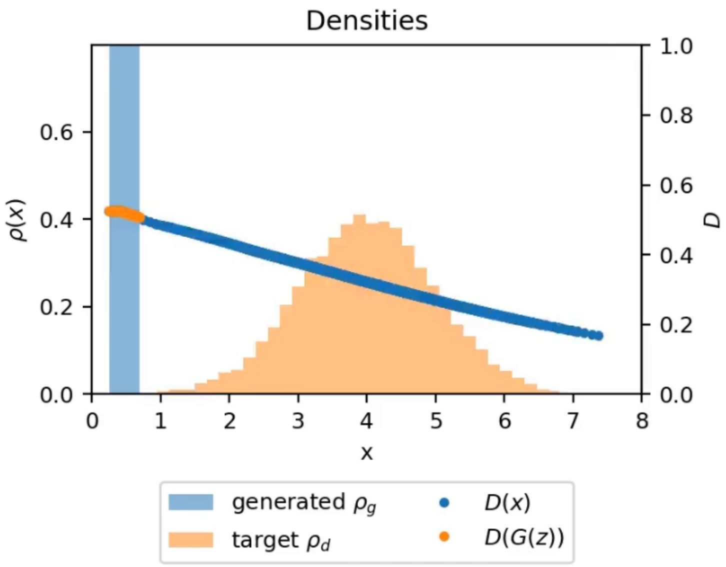


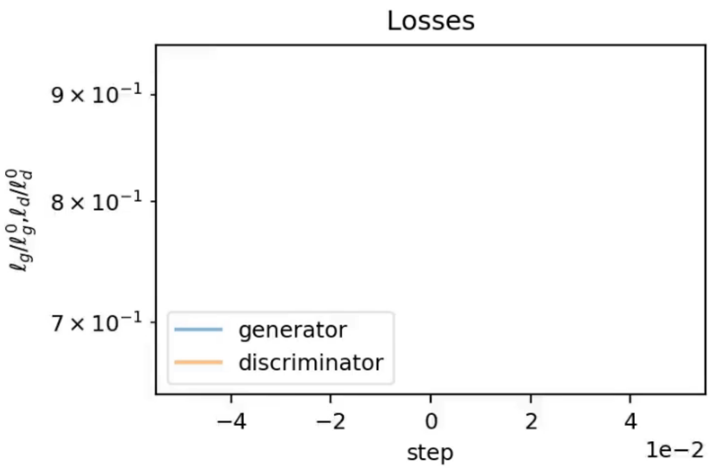
Results:

Atzberger 2020

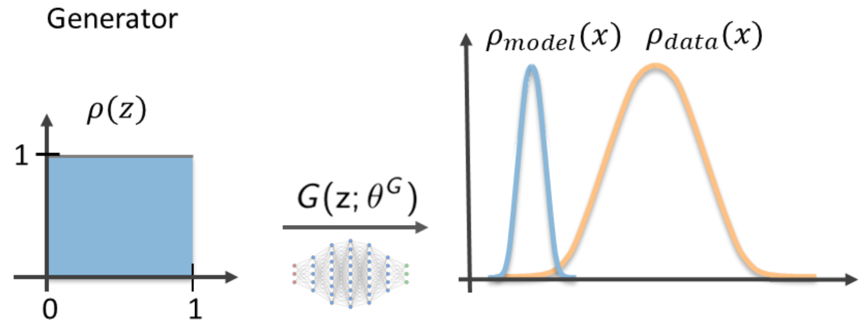


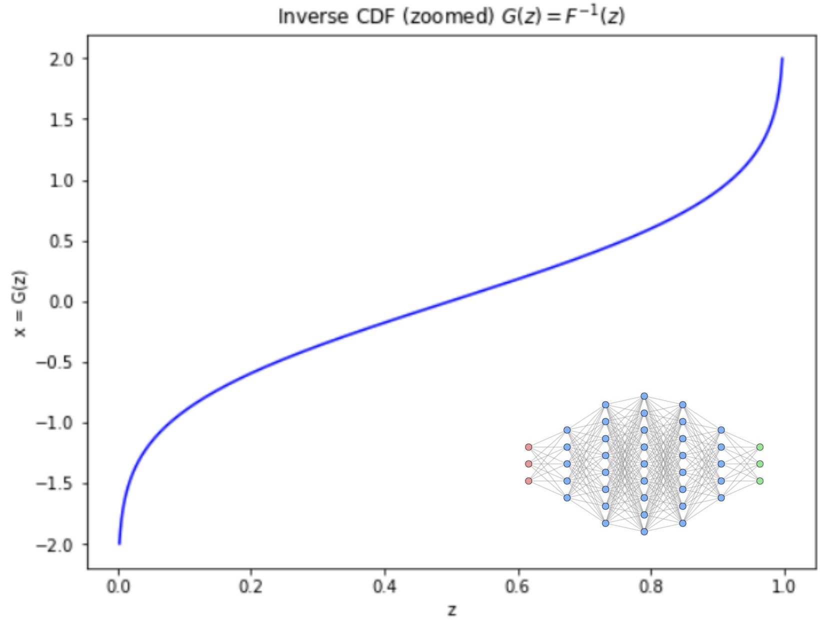
Example: Gaussian Target Distribution





Atzberger 2020





<u>Task:</u> Use GANs to learn Gaussian target data distribution $\rho_{data}(x)$.

Generator → Approximated by Deep Neural Network (DNN) and SGD.

Training: Alternate between minimization for D(x) and maximization for G(z).

Remark: Cumulative Distribution Function (CDF) → Inverse gives a generator.

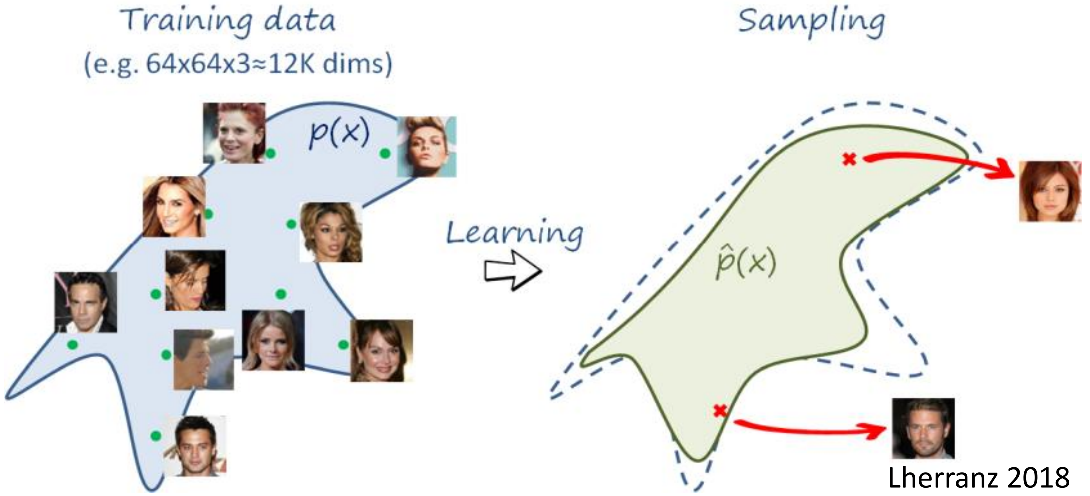
Remark: Gaussians this diverges to give small probability for tails. Noise sources type important consideration in practice.

GANs Celeb-HQ

Karras 2018

Image Generation

Karras 2018



Lherranz 2018

<u>Task:</u> Use GANs to generate images similar $\rho_{data}(x)$.

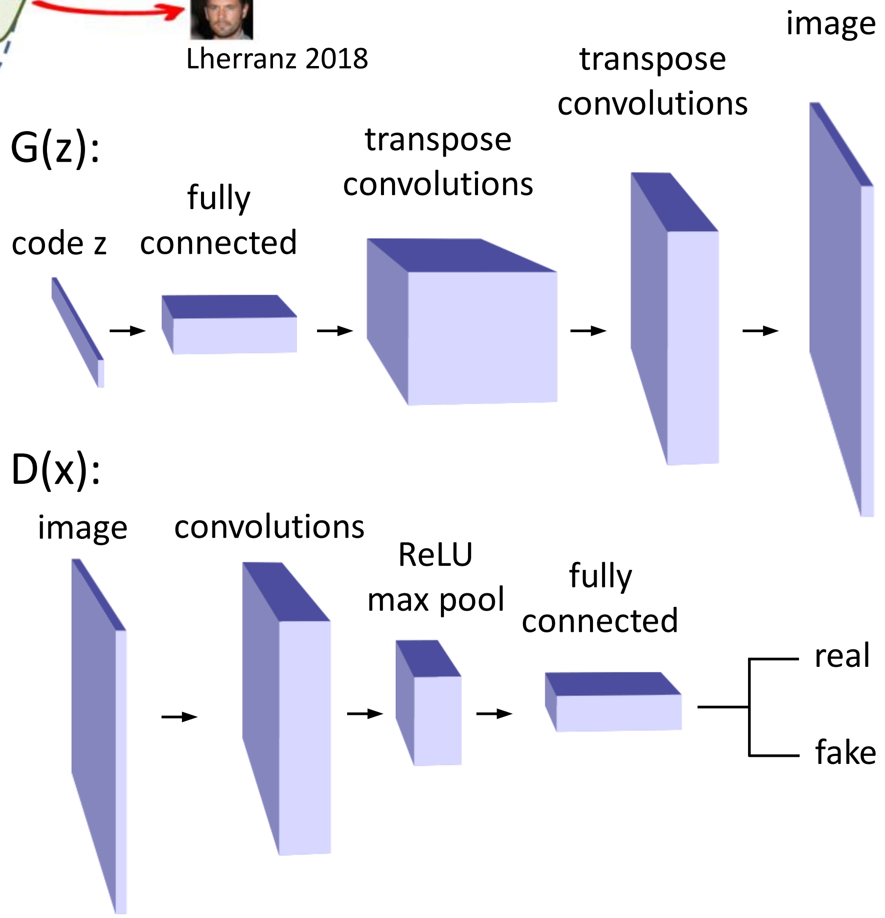
Generator G(z): maps noise from latent space Z → images X.

DNN Generator: Generate images using deep Transpose Convolutional Neural Networks (T-CNNs).

Discriminator D(x): Image classifier based on Convolutional Neural Networks (CNNs).

GANs: Use SGD to learn both classifier and generator at the same time.

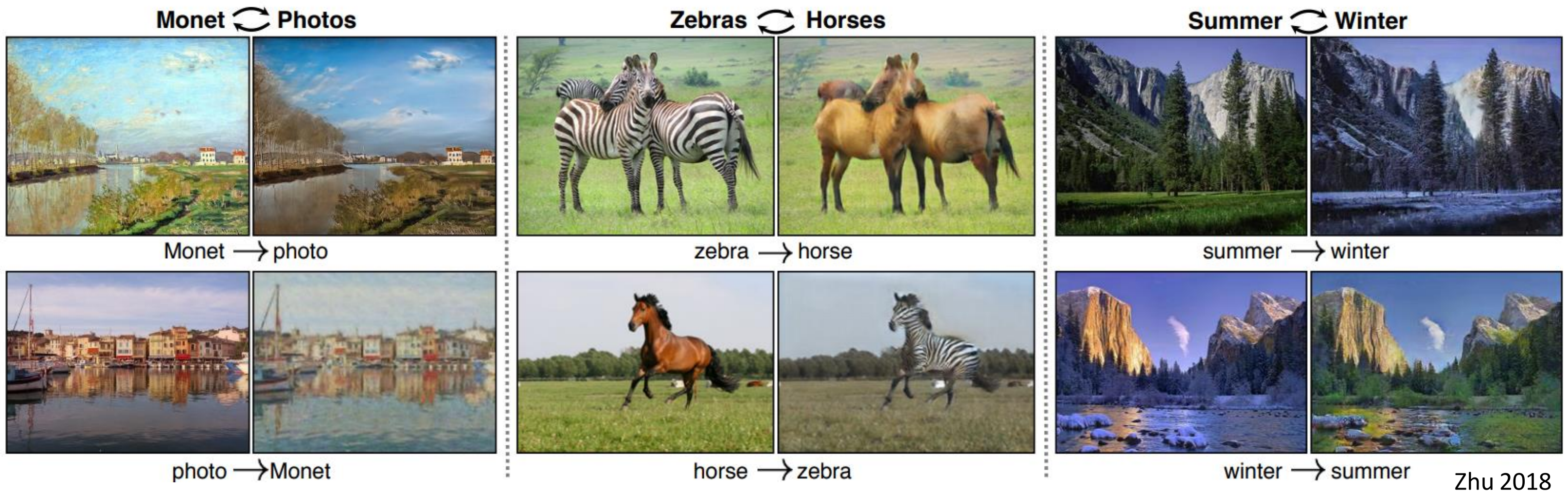
Important Considerations: architecture, regularizations (batch normalization), data quality, training protocols (balancing D and G),…



CycleGANs

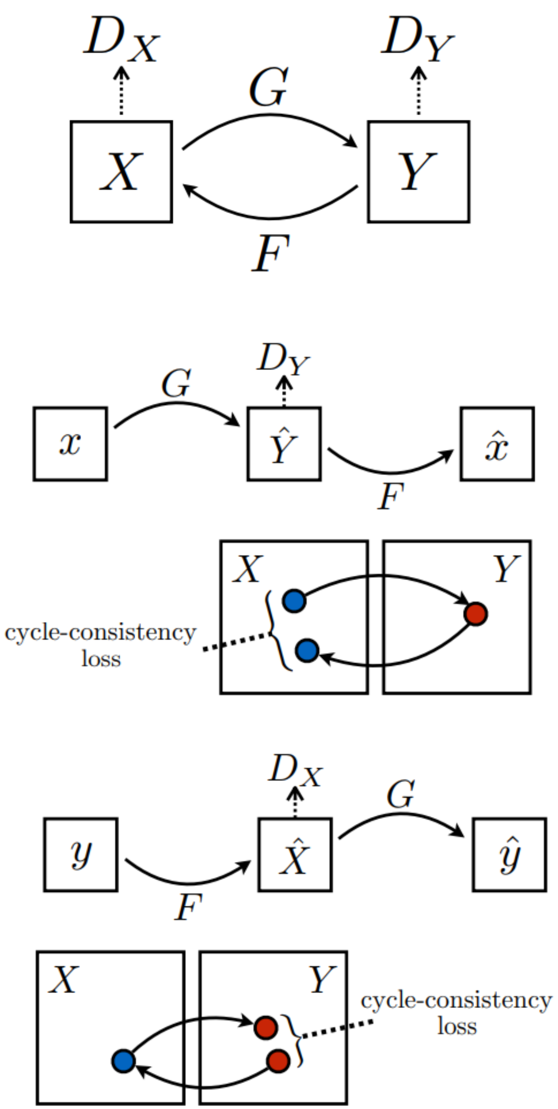
Zhu 2018

CycleGANs



Zhu 2018

Training Protocol



<u>Task:</u> Use input image to generate image of another class.

GANs trains **two generator maps** G(X) and F(Y).

Two discriminators: D_X and D_Y try to keep in space of natural images.

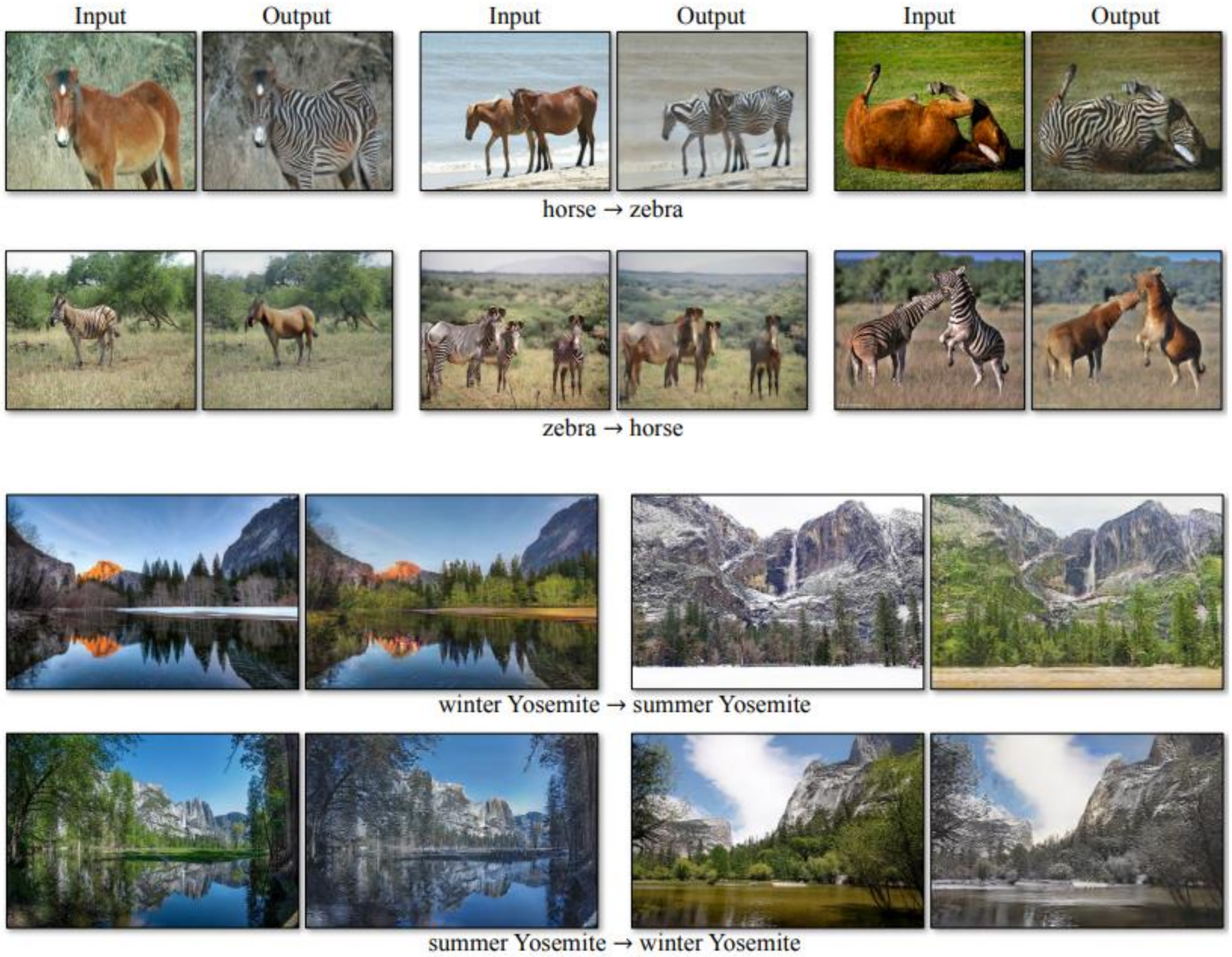
Reconstruction condition: $X \to Y \to \tilde{X}$ for information preservation.

Training: SGD over a large corpus of images or videos.

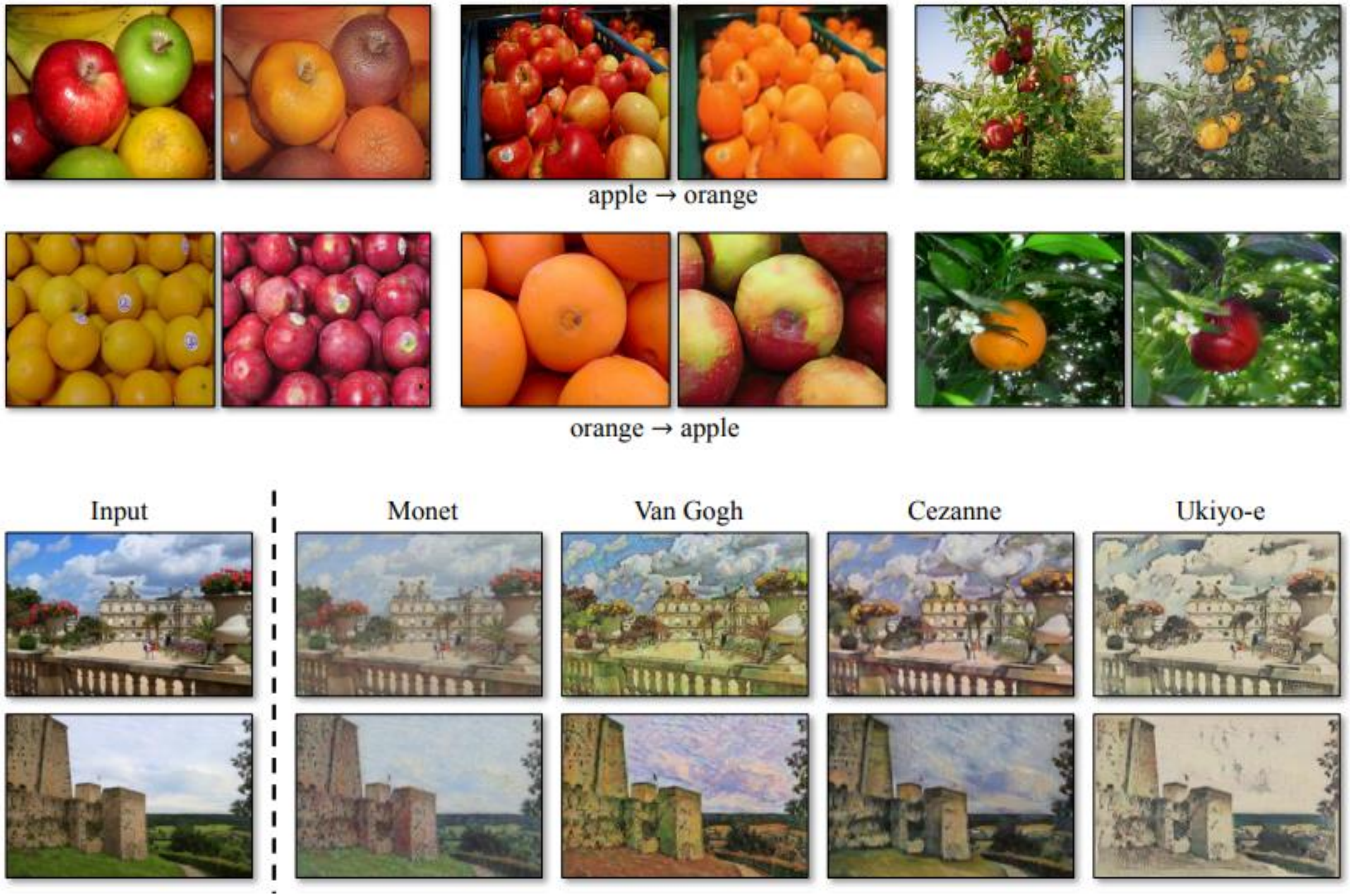
Results:

- image-to-image conversions (style, time-of-year, object class).
- video-to-video conversions (style, time-of-year, object class).

CycleGANs

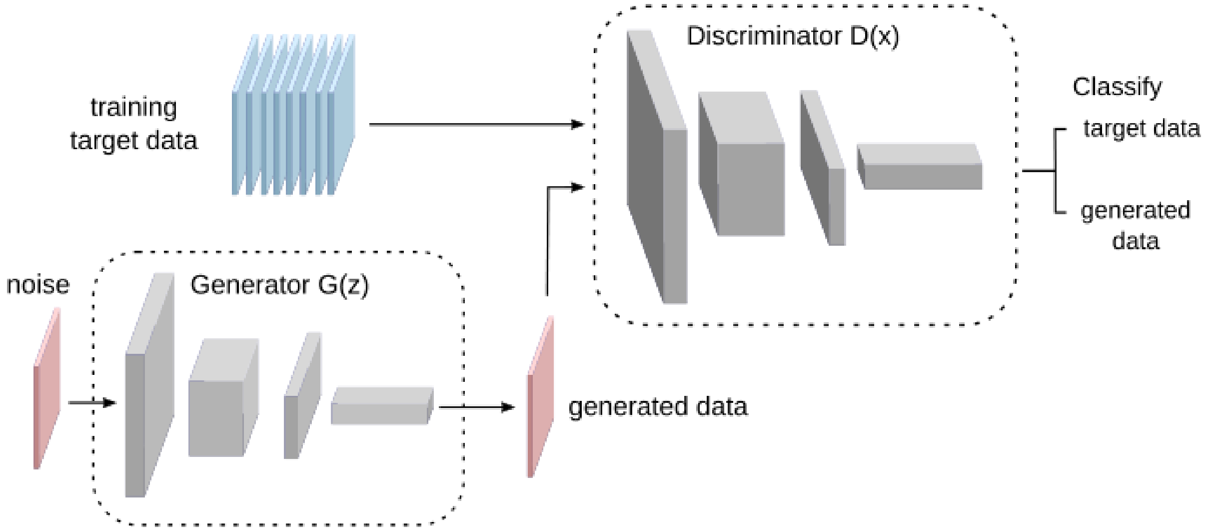

Input
Output
Input
Output
Input
Output
horse → zebra
zebra → horse
winter Yosemite → summer Yosemite
summer Yosemite → winter Yosemite

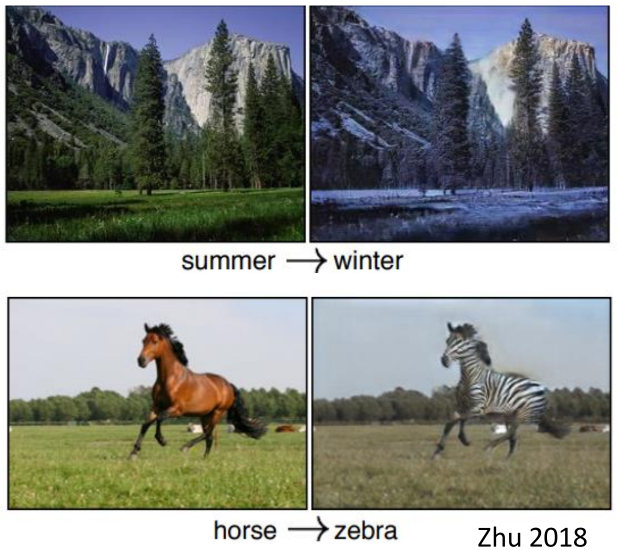
CycleGANs



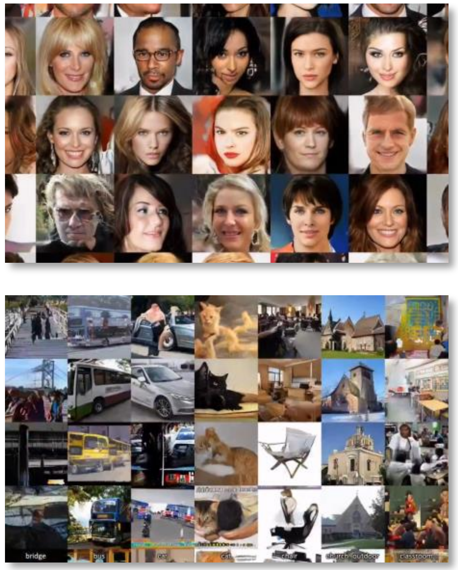
apple → orange
orange → apple
Input
Monet
Van Gogh
Cezanne
Ukiyo-e

Summary





Zhu 2018



Karras 2018

GANs provides approach for **training Generative Models.**

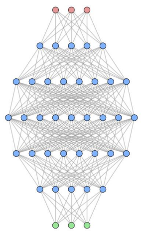
JS-GANs uses properties of supervised learning for discriminator D to obtain loss functions related to classifier behaviors.

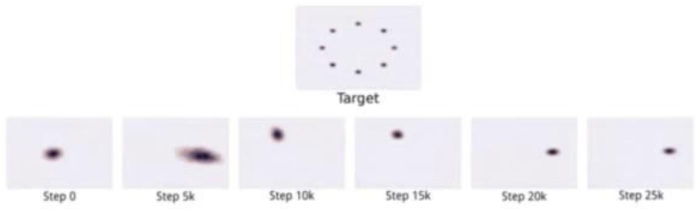
Many variants of GANs: Wasserstein (WGANs), Gradient Penalty (GP-GANs), Energy-based (E-GANs), …

Provides representations and parameterizations for subsets of manifold-like structures.

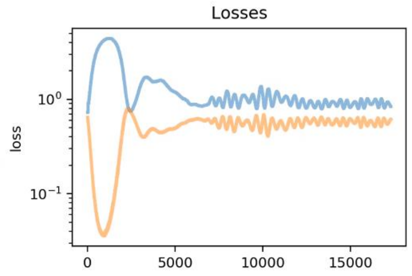
Challenges remain:

- computationally expensive (involves training DNNs).
- learning full probability distribution (mode collapse).
- reliable training (oscillations, lack of convergence).





Goodfellow 2016



Successes in image processing / video (interpolation, super-resolution, reconstruction, augmentation).

Emerging applications in the sciences and engineering (surrogate models, subgrid models, model reductions).