Introduction to Deep Learning

Lecture 7
Generative Adversarial Networks

Maria Vakalopoulou & Stergios Christodoulidis

MICS Laboratory

CentraleSupélec
Université Paris-Saclay

Wednesday, December 08, 2021

Last Lecture

ResNet block

- $H(x) = F(x) + x$
- If dimensions don't match
  - Either zero padding
  - Or a projection layer to match dimensions

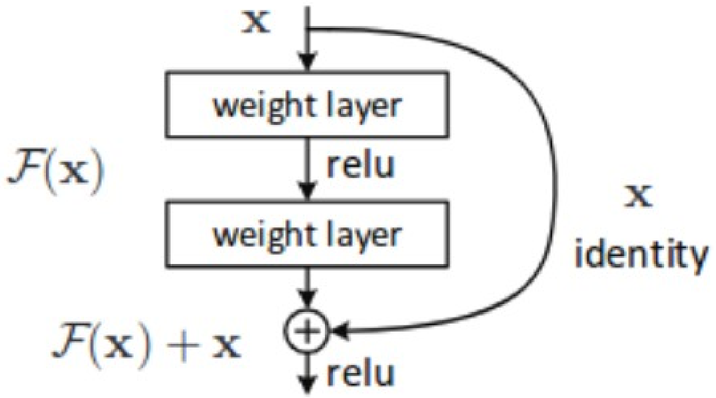
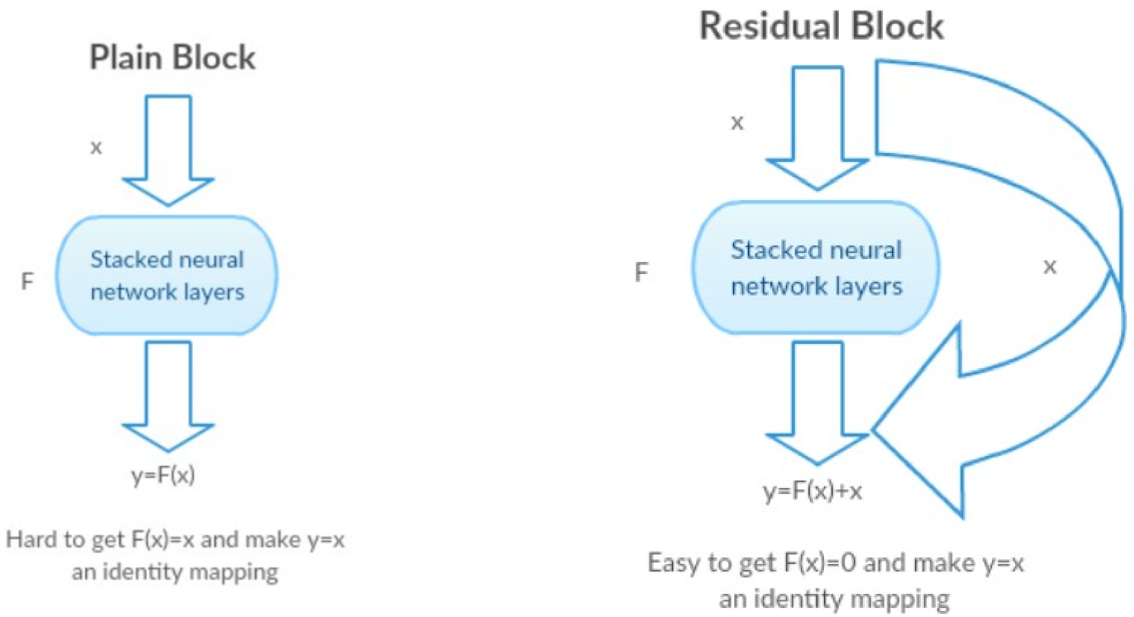
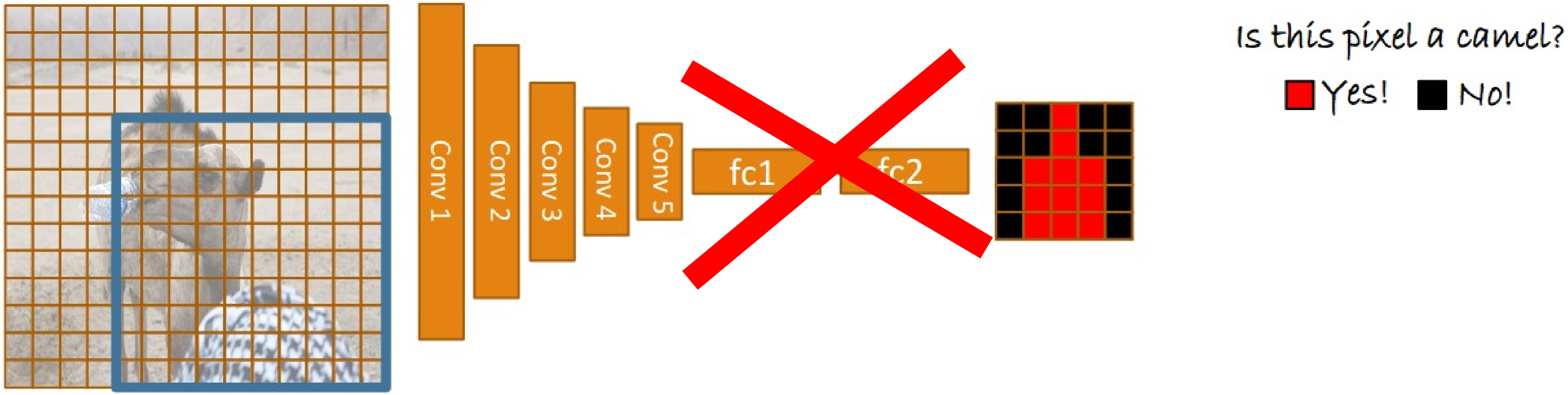


Figure 2. Residual learning: a building block.



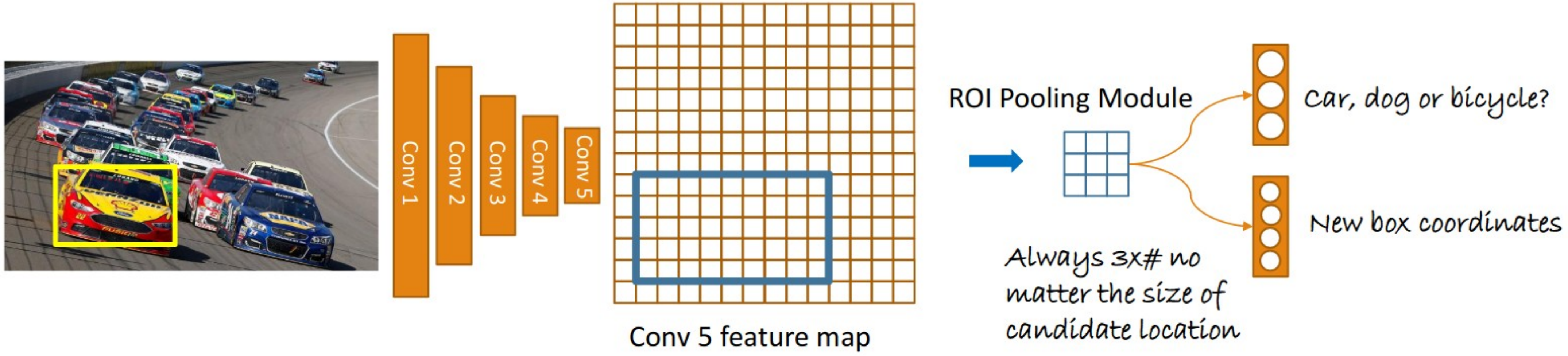
Going Fully Convolutional [LongCVPR2014]

- Image larger than network input → slide the network



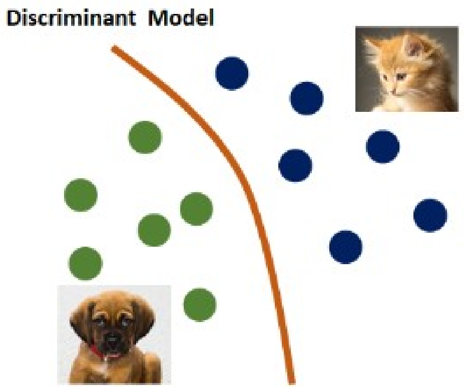
Fast R-CNN: Steps

- Process the whole image up to conv5
- Compute possible locations for objects (some correct, most wrong)
- Given single location → ROI pooling module extracts fixed length feature
- Connect to two final layers, 1 for classification, 1 for

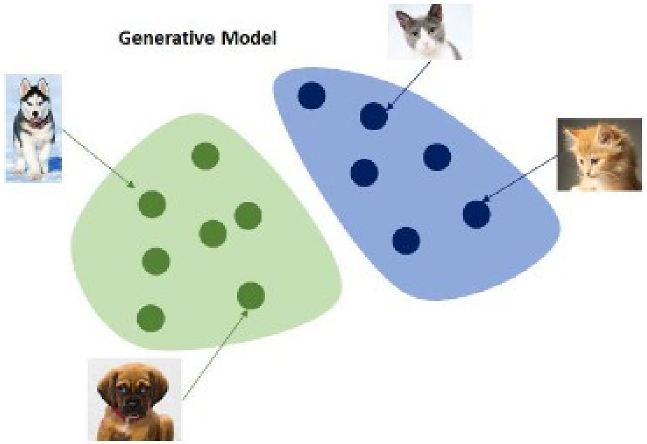


Discriminative vs Generative Learning

- Discriminative Models:
  - Discriminate between different kinds of data instances.
  - Captures the conditional PDF: $p(y|x)$
  - Task-oriented
  - E.g., Logistic Regression, SVM

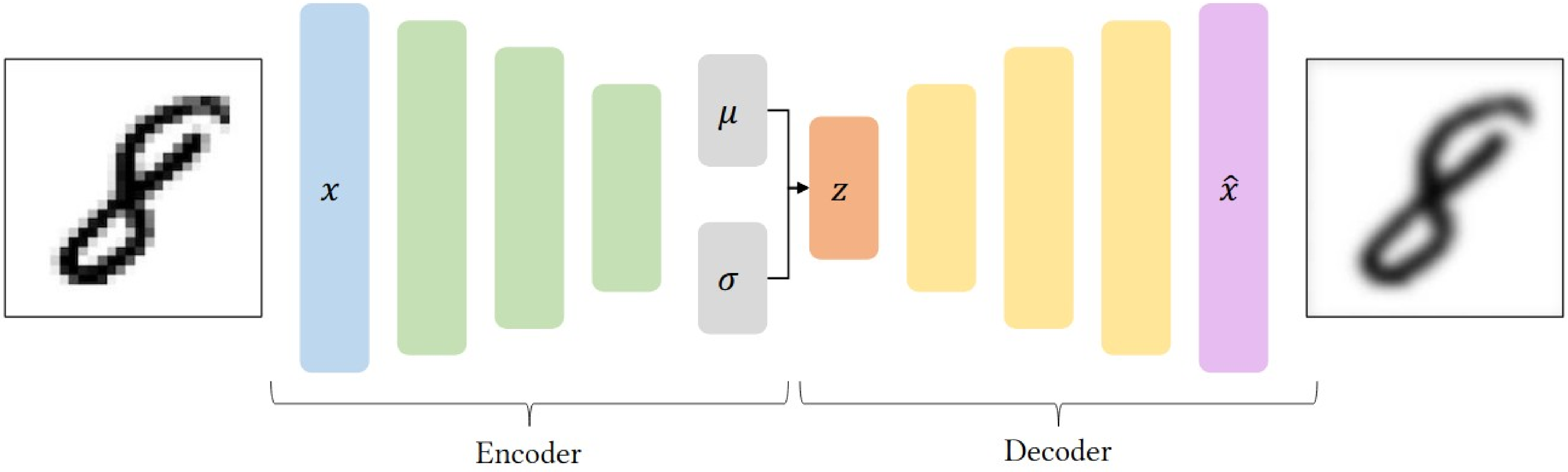


- Generative Models:
  - Can generate new data instances.
  - Captures the joint PDF: $p(x, y)$ or just $p(x)$ is no labels
  - Model the world $\longrightarrow$ Perform tasks, e.g. use Bayes rule to classify, i.e $p(y|x)$
  - Naive Bayes, Variational Autoencoders, GANs



Variational Autoencoders (VAE)

- Instead of straight learning the latent representation z
- We learn the parameters of a multivariate gaussian from which we sample z
- Not deterministic any more $\longrightarrow$ Stochastic sampling operation



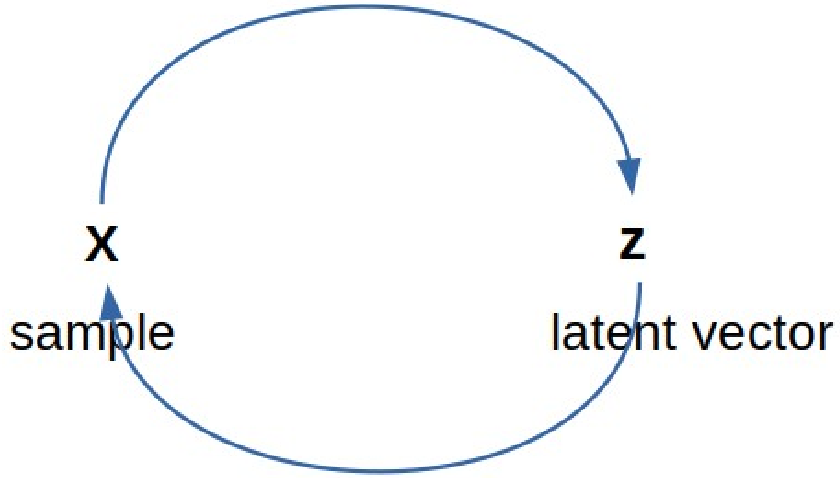
Today’s Lecture

Today’s Lecture

- Gentle intro to generative models
- Generative Adversarial Networks
- Variants of Generative Adversarial Networks

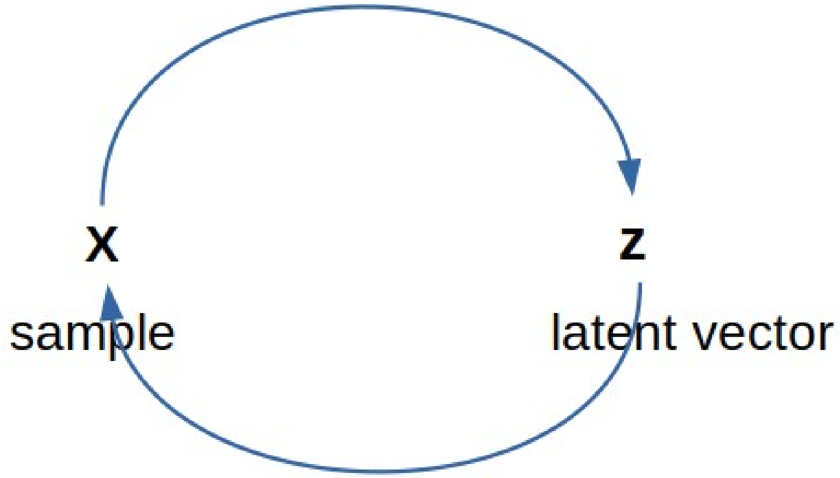
Types of Learning

- Generative modelling
  - Learn the joint pdf: $p(x, y)$
  - Model the world → Perform tasks, e.g. use Bayes rule to classify: $p(y|x)$
  - Naive Bayes, Variational Autoencoders, GANs



Types of Learning

- Generative modelling
  - Learn the joint pdf: $p(x, y)$
  - Model the world → Perform tasks, e.g. use Bayes rule to classify: $p(y|x)$
  - Naive Bayes, Variational Autoencoders, GANs
- Discriminative modelling
  - Learn the conditional pdf: $p(y|x)$
  - Task-oriented
  - E.g., Logistic Regression, SVM



Types of Learning

- What to pick?
  - V. Vapnik: “One should solve the *[classification]* problem directly and never solve a more general *[and harder]* problem as an intermediate step”
- Typically, discriminative models are selected to do **the** job
- Generative models give us more theoretical guarantees that the model is going to work as intended
  - Better generalization
  - Less overfitting
  - Better modelling of causal relationships

Applications of generative modeling?

Applications of generative modeling?

- Act as a regularizer in discriminative learning
  - Discriminative learning often too goal-oriented
  - Overfitting to the observations
- Semi-supervised learning
- Simulating “possible futures” fro Reinforcement Learning
- Data-driven generation/sampling/ simulation

Applications of generative modeling?

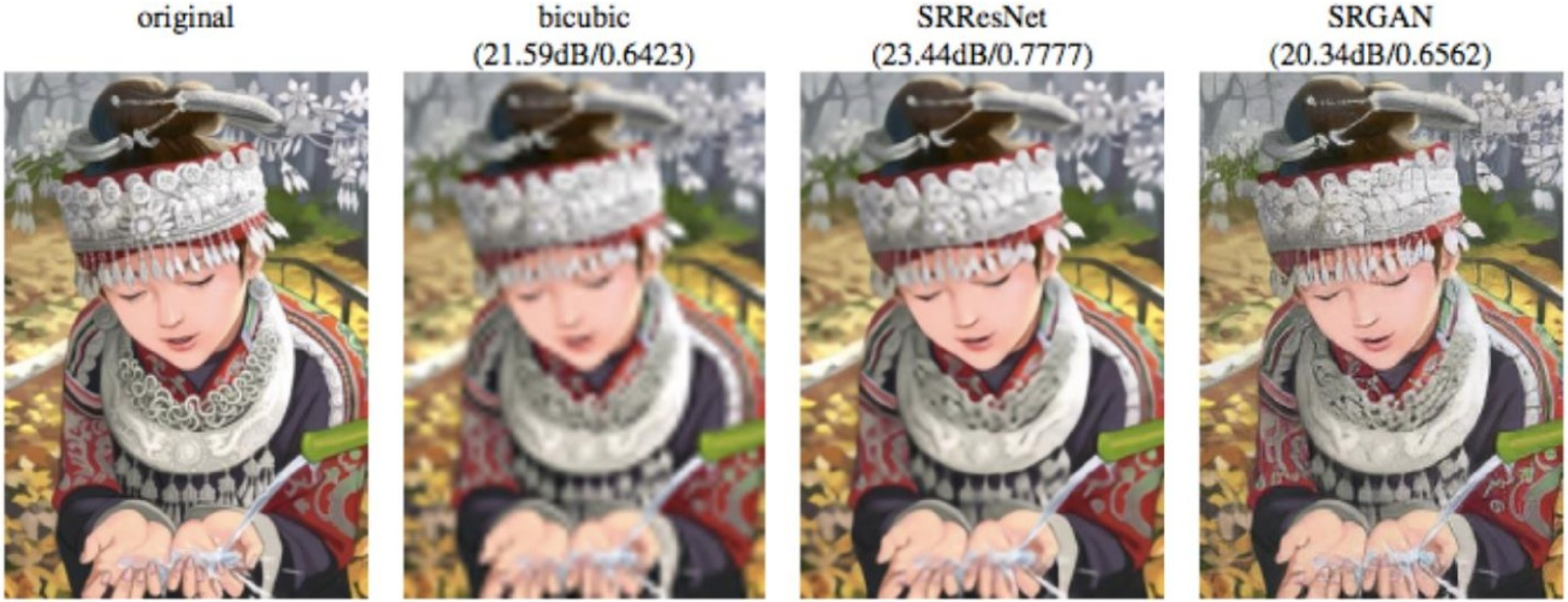
- Image Generation

(a) Generated by LSGANs.

(b) Generated by DCGANs (Reported in [13]).

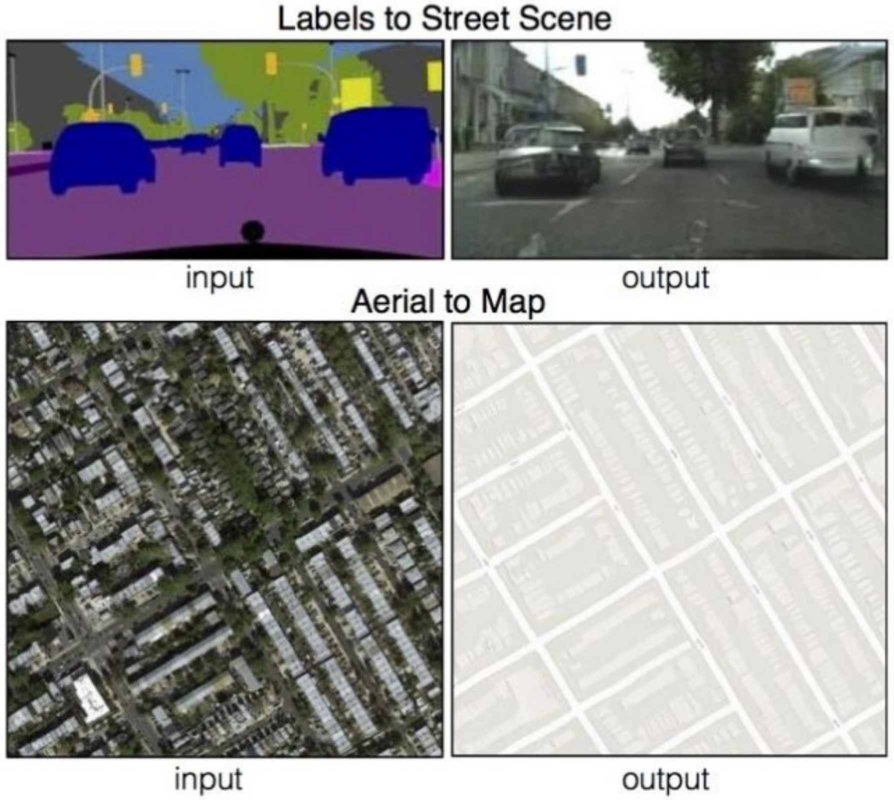
Applications of generative modeling?

- Super-resolution



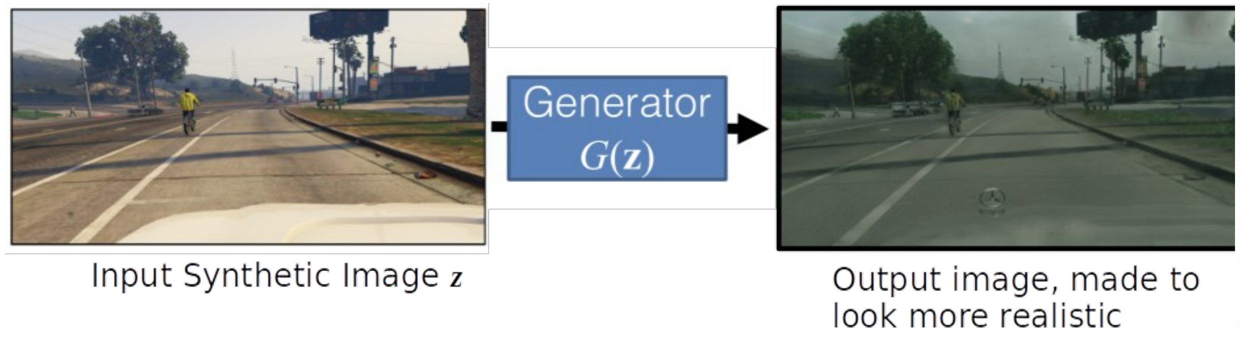
Applications of generative modeling?

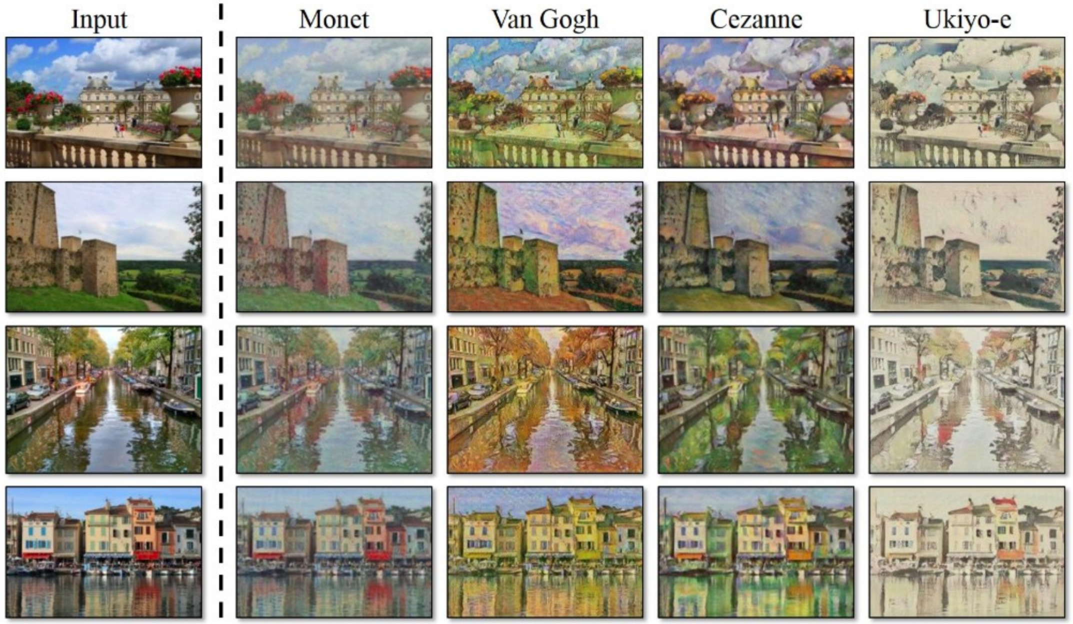
- Cross-model translation



Applications of generative modeling?

- Domain adaptation/ Style Transfer

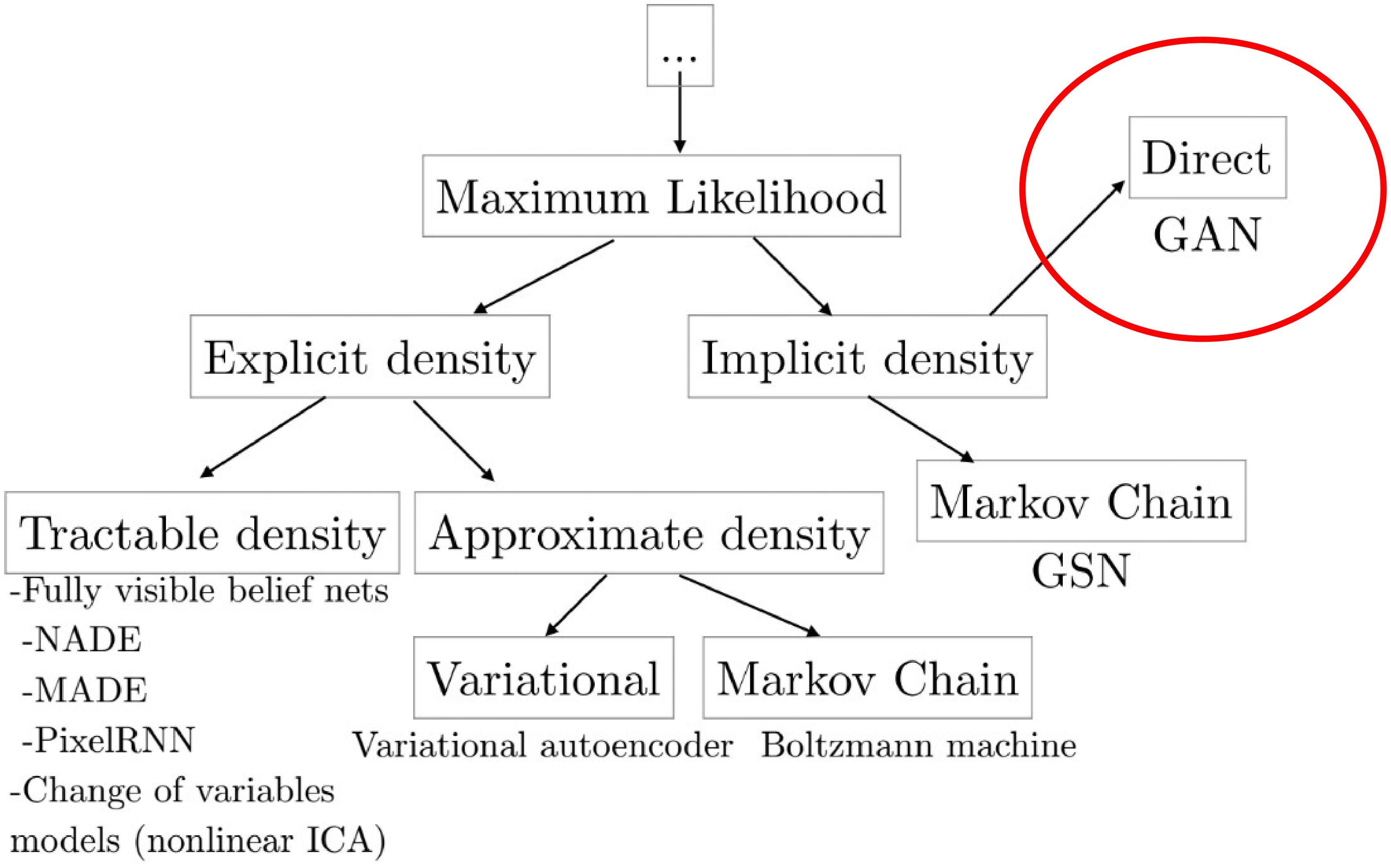




Other Applications

- M. Mustafa et al. “Cosmogan: Creating High-Fidelity Weak Lensing Covergence Maps Using Generative Adversarial Networks” in Arxiv 2017
- S. Collaboration. “Fast Simulation of Muons Produced at the ShiP Experiment Using Generative Adversarial Networks”. In Arxiv 2019
- Z. E. et al. “Deep Learning Enables Rapid Identification of Potent DDR1Kinase Inhibitors” In. Nature Biotechnology 2019
- Deep Fakes.

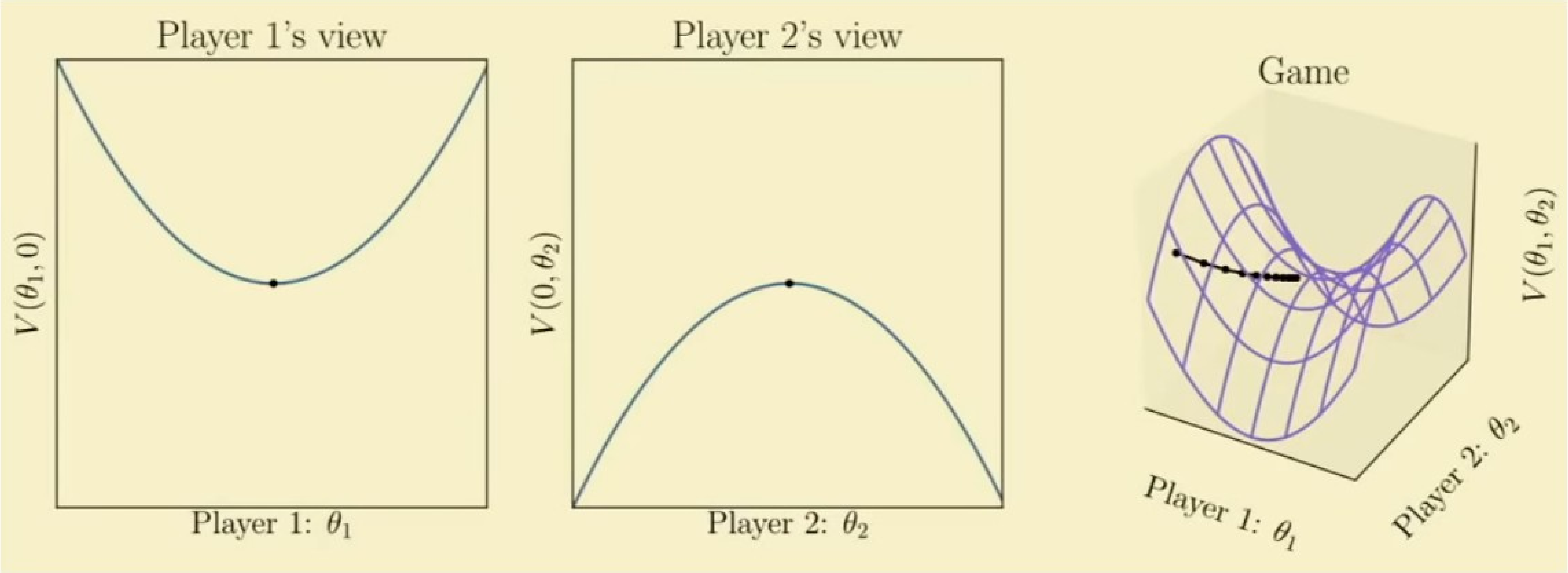
A map of generative models



What is a GAN?

- **G**enerative
  - You can sample novel input samples
  - E.g., you can literally “create” images that never existed
- **A**dversarial
  - Our generative model **G** learns adversarially, by fooling an discriminative oracle model **D**
- **N**etwork
  - Implemented typically as a (deep) neural network
  - Easy to incorporate new modules
  - Easy to learn via backpropagation

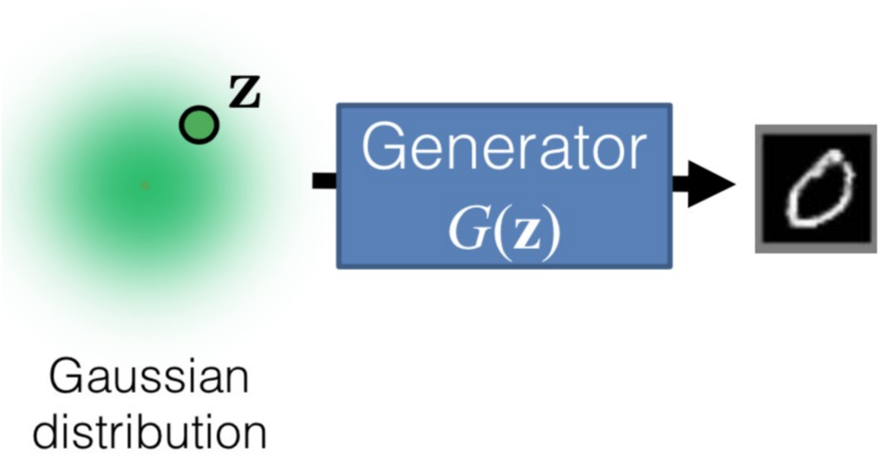
Adversarial Learning



Goodfellow, 2019

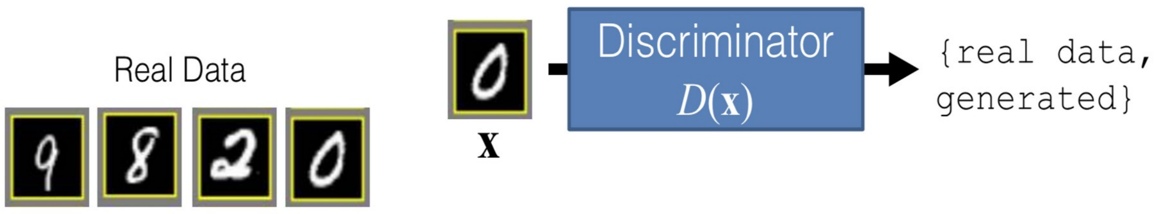
Generative Adversarial Networks

- We would like to train a network G to generate images from some domain from random vectors z:



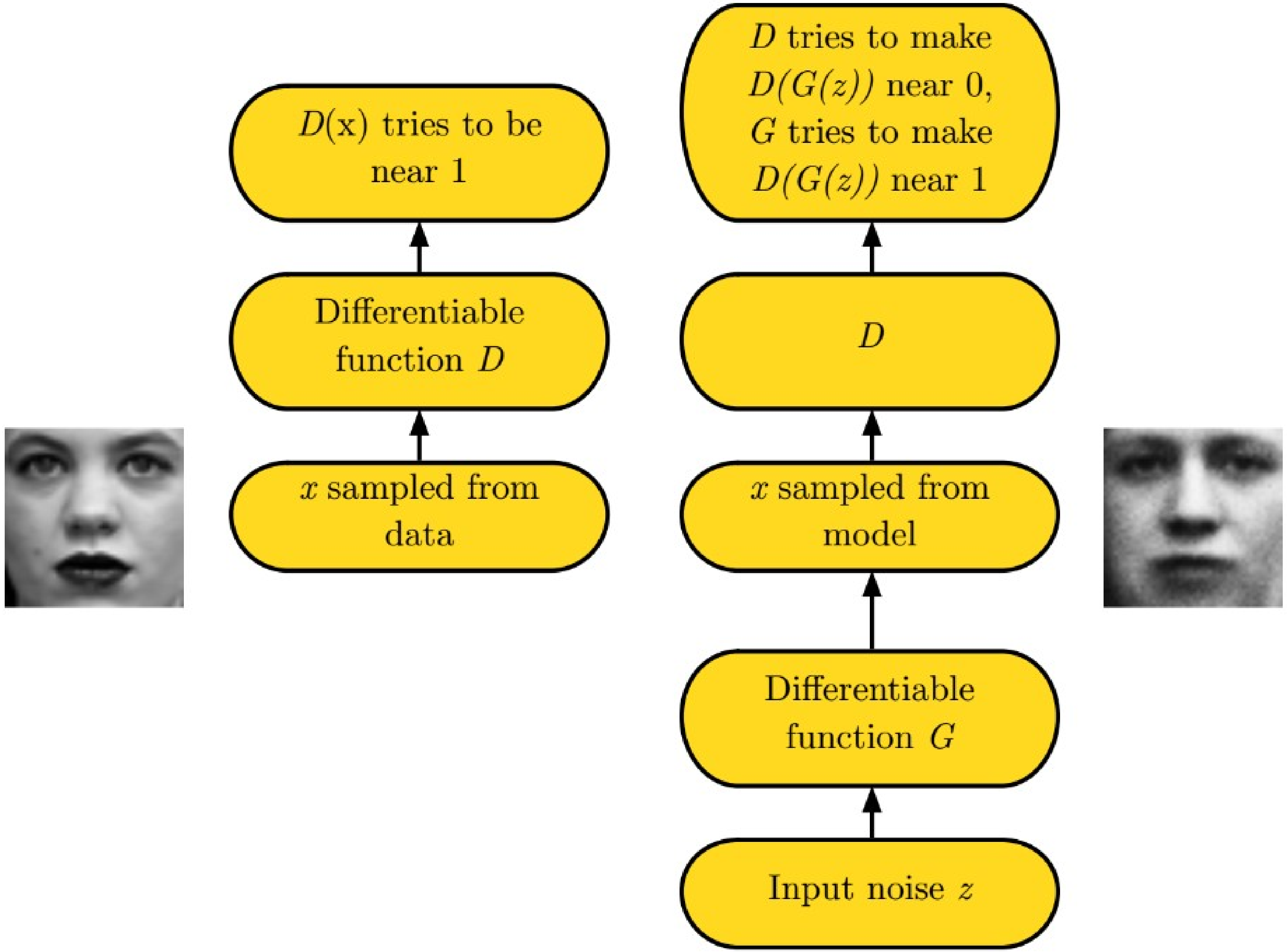
Generative Adversarial Networks

- Idea: Add a second network (Discriminator D) jointly trained with the Generator G to recognize if an input is a real sample from the domain of interest or if it was created by the Generator.



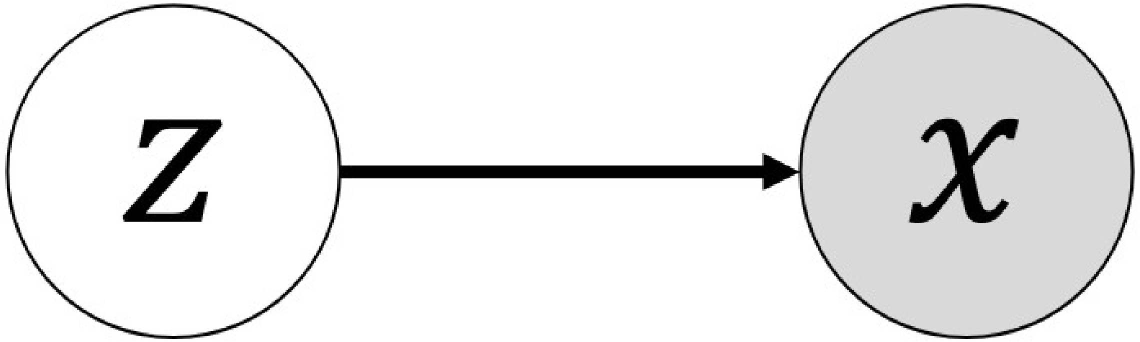
- When the Discriminator cannot distinguish the generated images from the real ones, the Generator generates realistic images

GAN: Pipeline



Generator network $x = G(z; \theta^{(G)})$

- Must be differentiable
- No invertibility requirement
- Trainable for any size of z
- Can make conditionally Gaussian given z, but no strict requirement



Generator & Discriminator: Implementation

- The discriminator is just a standard neural network
- The generator looks like an inverse discriminator

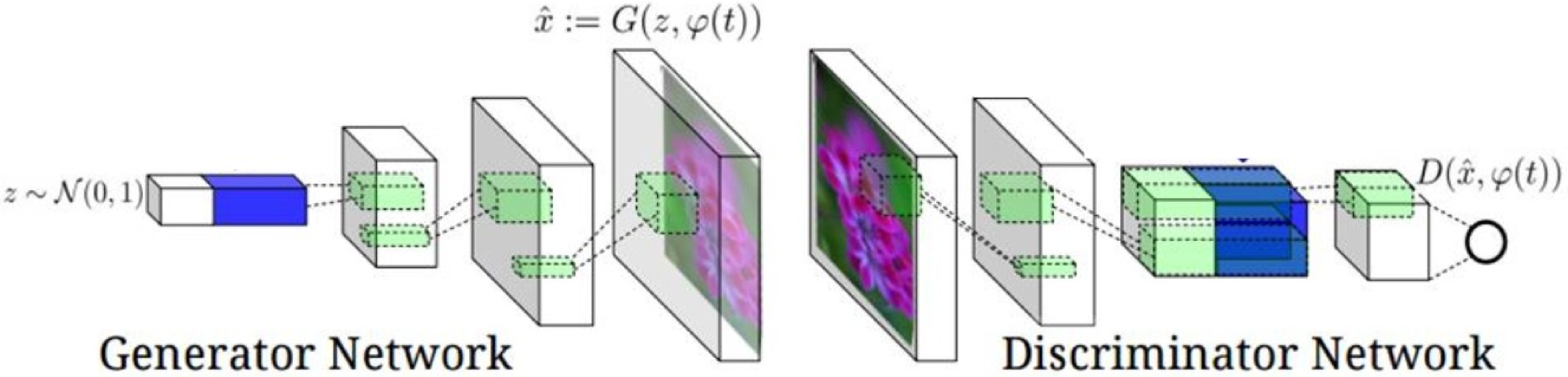


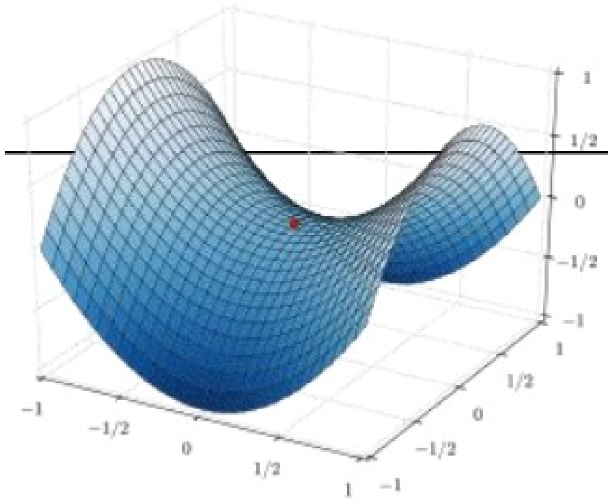
Figure 2. Our text-conditional convolutional GAN architecture. Text encoding $\varphi(t)$ is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

Network Architecture

Training definitions

- Minimax
- Maximin
- Heuristic, non-saturating game
- Max likelihood game

Minimax Game



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

- $D(x) = 1$ → The discriminator believes that x is a true image
- $D(G(z)) = 1$ → The discriminator believes that G(z) is a true image

- Equilibrium is a saddle point of the discriminator loss
- Final loss resembles Jenssen-Shannon divergence

https://arxiv.org/pdf/1701.00160.pdf

Minimax Game

- For the simple case of zero-sum game

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

- So, we can summarize game by

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

- Easier theoretical analysis

Minimax Game

- For the simple case of zero-sum game

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

- So, we can summarize game by

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

- Easier theoretical analysis

- In practice not used → when the discriminator starts to recognize fake samples, then generator gradients vanish

Heuristic non-saturating game

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

- Equilibrium not any more describable by a single loss
- Generator maximizes the log-probability of the discriminator being mistaken
  - Good G(z) → D(G(z)) =1 → $J^{(G)}$ is maximized
- Heuristically motivated; generator can still learn even when discriminator successfully rejects all generator samples

Original Algorithm

for number of training iterations **do**

for k steps **do**

Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$

Sample minibatch of m real samples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$

Update the discriminator D by stochastic gradient ascend.

Gradient:

$$\frac{\partial}{\partial D} \left(\frac{1}{m} \sum_{i=1}^{m} \log D(\mathbf{x}^{(i)}) + \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}^{(i)}))) \right).$$

end for

Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$

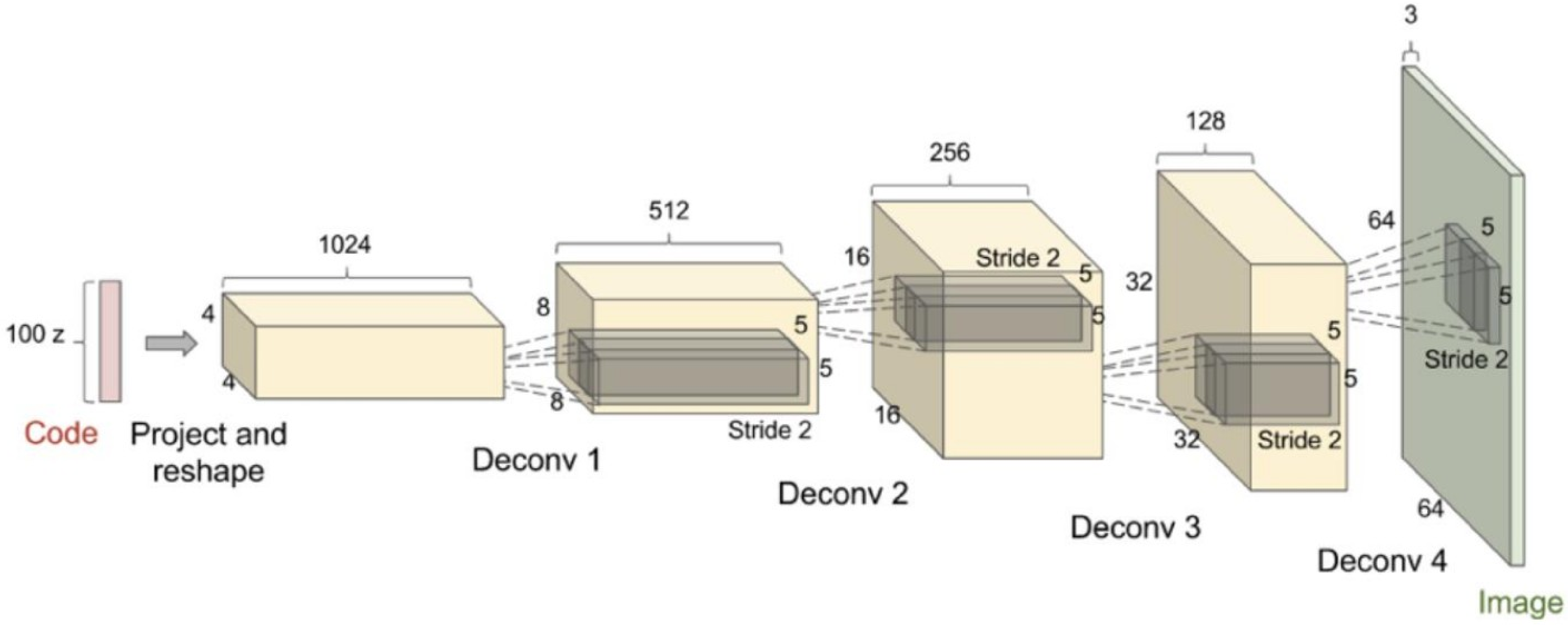
Update the generator G by stochastic gradient ascend.

Gradient:

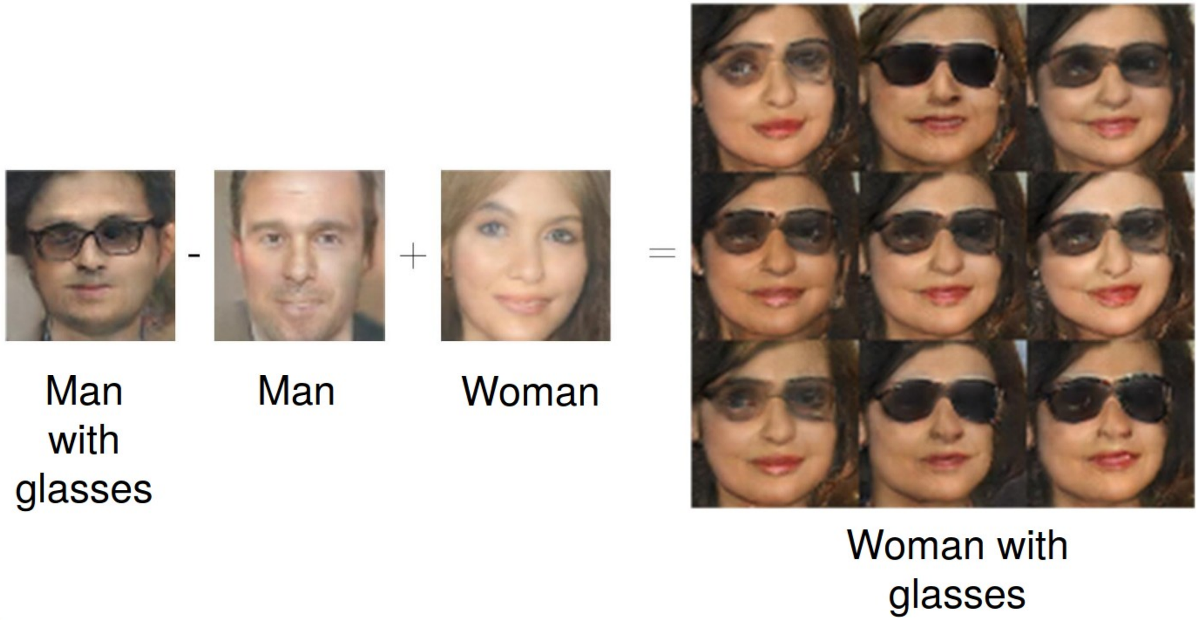
$$\frac{\partial}{\partial G} \left(\frac{1}{m} \sum_{i=1}^{m} \log(D(G(\mathbf{z}^{(i)}))) \right).$$

end for

DCGAN Architecture



Examples *[up to 2015]*



Modifying GANs for Max-Likelihood

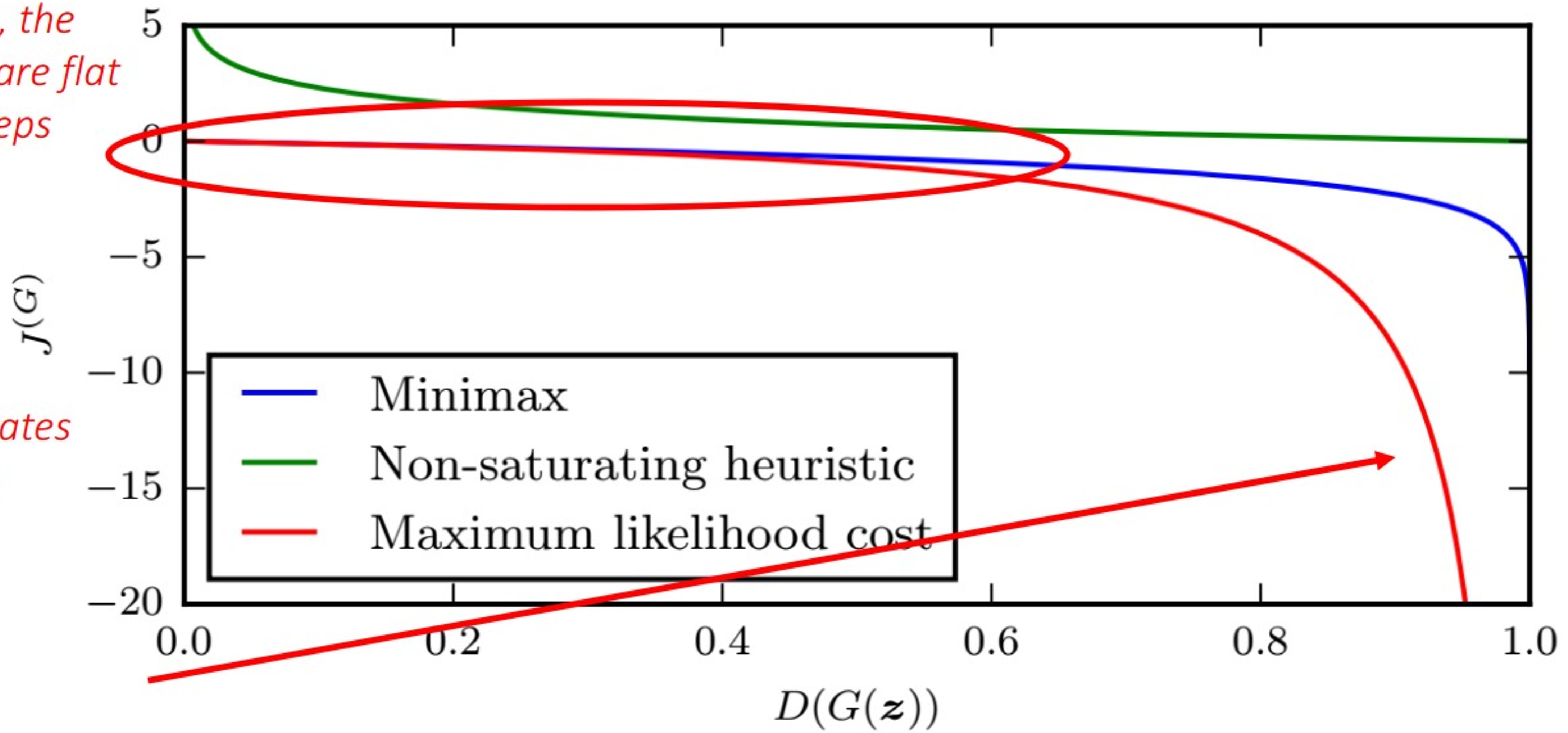
- $J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) \; - \frac{1}{2}\mathbb{E}_{z\sim p_z} \log(1 - D(G(z)))$
- $J^{(G)} = -\frac{1}{2}\mathbb{E}_z \log(\sigma^{-1}(D\big(G(z)\big))$

When discriminator is optimal, the generator gradient matches that of maximum likelihood

https://arxiv.org/abs/1412.6515

Comparison of Generator Losses

When sample is likely fake, the minimax and the ML cost are flat
→ no gradients in early steps



The ML cost variant generates gradients mostly from the "good generations"
→ all gradients from few samples
→ high variance
→ Variance reduction?

GAN Problems: Vanishing Gradients

- $J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) \; -\frac{1}{2}\mathbb{E}_z \log(1 - D(G(z))$
- $J^{(G)} = -\frac{1}{2}\mathbb{E}_z \log(D(G(z))$
- If the discriminator is quite bad
  - No accurate feedback for generator
  - No reasonable generator gradients
- But, if the discriminator is perfect, $D(x) = D^*(x)$
  - Gradients go to 0
  - No learning anymore
- Bad when this happens early in the training
  - Easier to train the discriminator than the generator

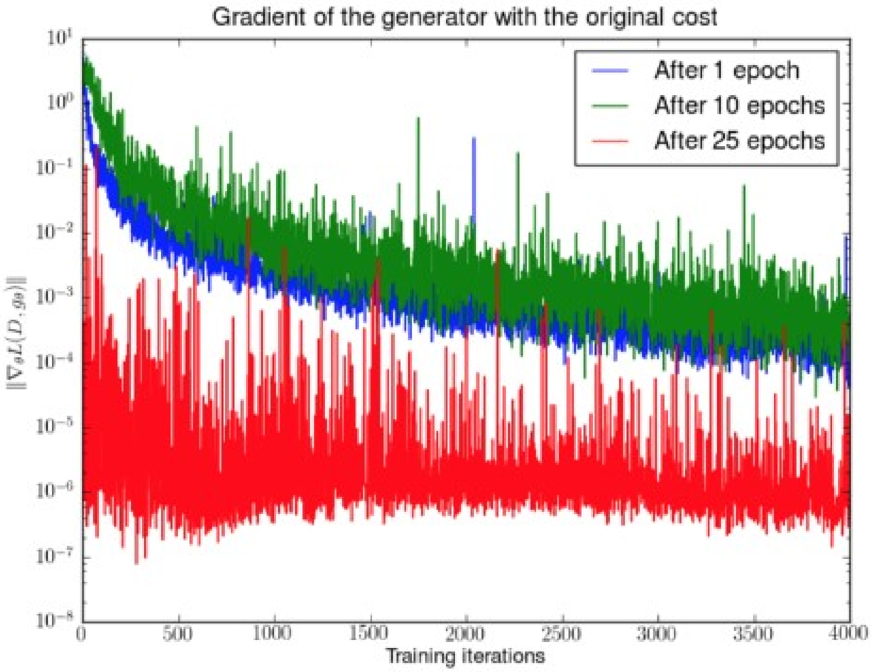


Figure from Arjovsky and Bottou 2016. DCGAN after training 1, 10 and 25 epochs

GAN Problems: Batch Normalization does not work right way

- Batch-normalization causes strong intra-batch correlation
  - Activations depend on other inputs
  - Generations depend on other inputs
- Generation looks smooth but awkward, strong intra batch correlation

Reference Batch Normalization

- Fix a reference batch $R = \{r_1, r_2, \ldots, r_m\}$
- Given new inputs $X = \{x_1, x_2, \ldots, x_m\}$
- Compute mean and standard deviation of feature of R
- Normalize the features of X using the mean and standard deviation from R
- Every x_1 is always treated the same, regardless of which other examples appear in the minibatch

Visual Batch Normalization

- Reference batch norm can overfit to the reference batch. A partial solution is virtual batch norm
- Fix a reference batch $R = \{r_1, r_2, \ldots, r_m\}$
- Given new inputs $X = \{x_1, x_2, \ldots, x_m\}$
- For each x_i in X:
  - Construct a virtual batch V containing both x_i and all of R
  - Compute mean and standard deviation of features of V
  - Normalize the features of x_i using the mean and standard deviation from V

Balancing generator and discriminator

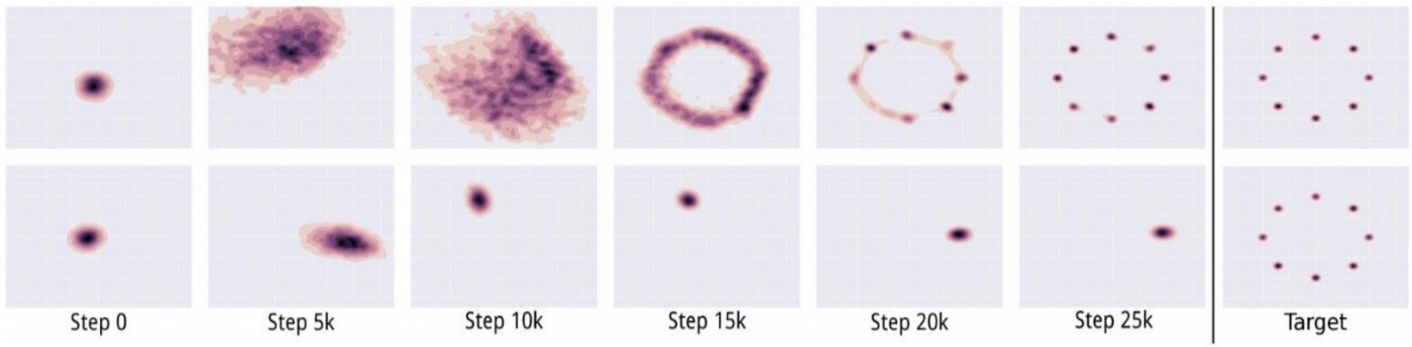
- Usually the discriminator wins
  - Good, as the theoretical justification assumes a perfect discriminator
- Usually the discriminator network is bigger and deeper than the generator
- Sometimes running discriminator more often than generator works better
  - However, no real consensus
- Do not limit the discriminator to avoid making it too smart
  - Making learning “easier” will nor necessarily make generation better
  - Better use non-saturating cost
  - Better use label smoothing

Challenge: Convergence

- Optimization is tricky and unstable
  - Finding a saddle point does not imply a global minimum
  - A saddle point is also sensitive to disturbances
- An equilibrium might not even be reached
- Mode-collapse is the most severe form of non-convergence

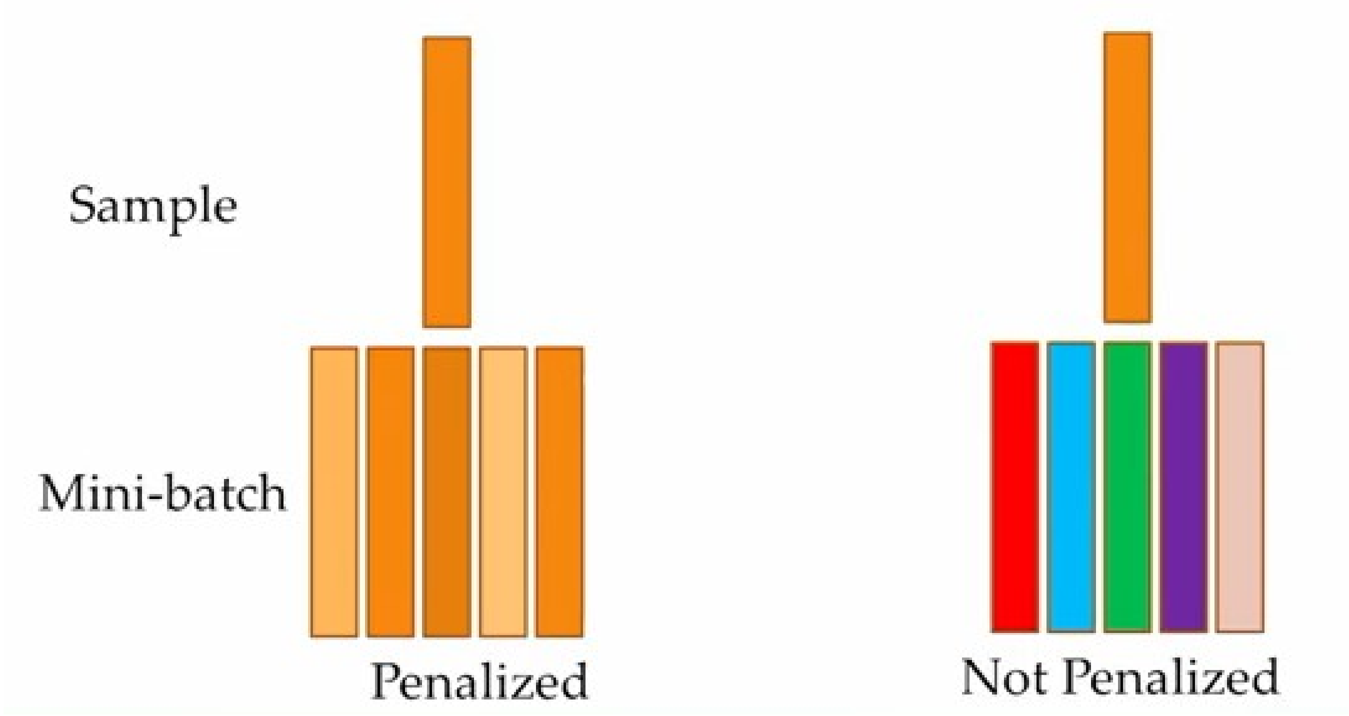
GAN Problems: Mode collapse

- Discriminator converges to the correct distribution
- Generator however places all mass in the most likely point
- All other modes are ignored
  - Underestimating variance
- Low diversity in generating samples



Minibatch features

- Classify each sample by comparing to other examples in the mini-batch
- If samples are too similar, the model is penalized



Challenge: how to evaluate?

- Despite the nice images, who cares?
- It would be nice to quantitatively evaluate the model
- For GANs it is hard to even estimate the likelihood
- In the absence of a precise evaluation metric, do GANs do truly good generations or generations that appeal/ fool to the human eye?
  - Can we trust the generations for critical applications, like medical tasks?
  - *Are humans a good discriminator for the converges generator?*

Training procedure

- Use SGD-like algorithm of choice
  - Adam Optimizer is a good choice
- Use two mini-batches simultaneously
  - The first mini-batch contains real examples from the training set
  - The second mini-batch contains fake generated examples from the generator
- Optional: run k-steps of one player (e.g. discriminator) for every step of the other player (e.g. generator)

Feature matching

- Instead of matching image statistics, match feature statistics

$$J_D = \left\| \mathbb{E}_{\boldsymbol{x} \sim p_{data}} f(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} f\big(G(\boldsymbol{z})\big) \right\|_2^2$$

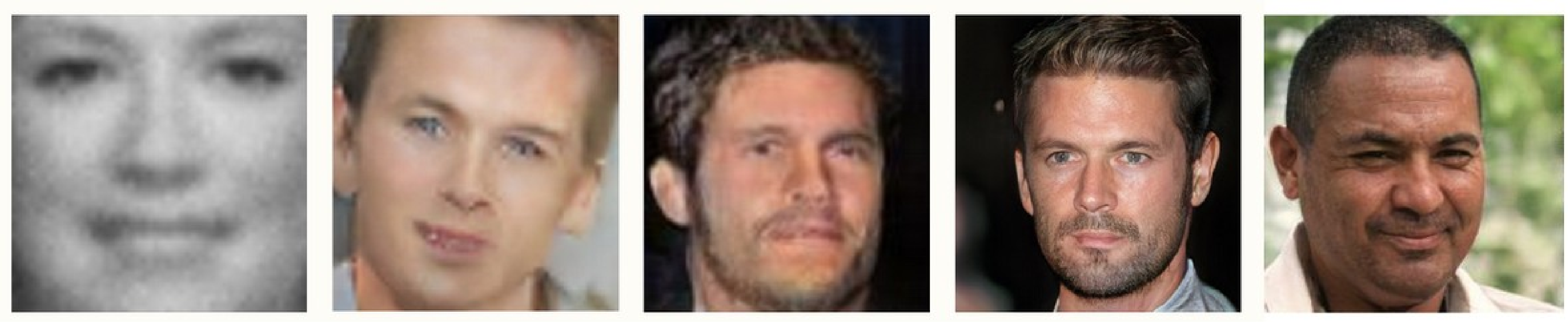
- If can be any statistic of the data, like the mean or the median

Use labels if possible

- Learning a conditional model $p(y|x)$ is often generates better samples
  - Denton et al., 2015
- Even learning $p(x,y)$ makes samples look more realistic
  - Salimans et al., 2016
- Conditional GANs are a great addition for learning with labels

Summary

- GANs are generative models using supervised learning to approximate and intractable cost function
- GANs can simulate many cost functions, including max likelihood
- Finding Nash equilibria in high-dimensional, continuous, non-convex games is an important open research problem
- GAN research is in its infancy, most works published only in 2016. Not mature enough yet, but very compelling results



2014 2015 2016 2017 2018

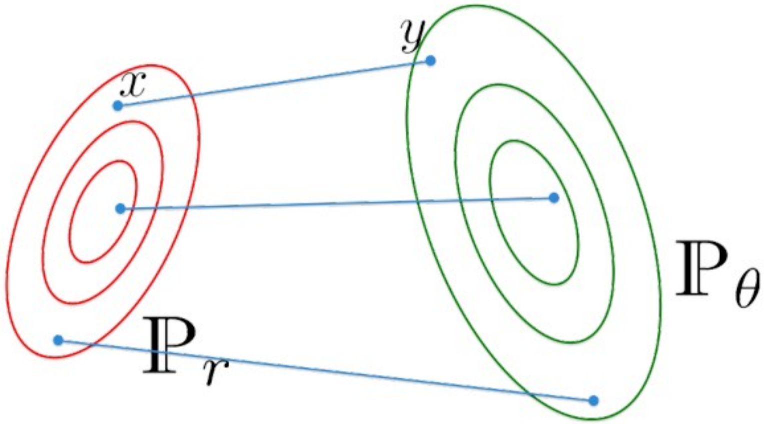
Models

- Wasserstein GAN
- Progressive GANs
- InfoGAN
- Conditional GAN
- StyleGAN
- CycleGAN

Wasserstein GAN [Intuition]

- The distribution of the generated data should be as close as possible to the distribution of the real data [M. Arjovsky et al. 2017]

- The Wasserstein metric is the cost of optimal transport between the two distributions.



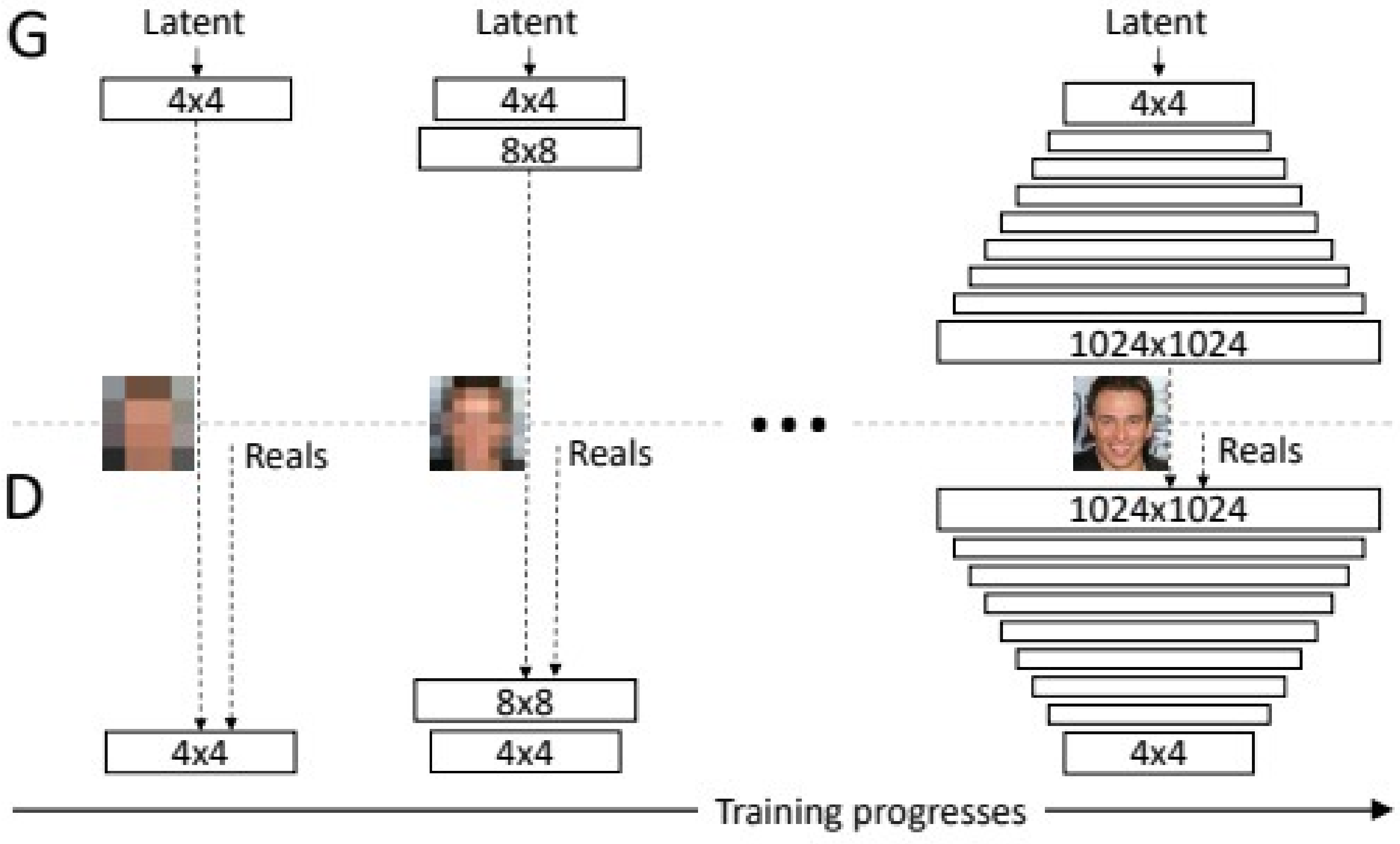
Intuitive (but imperfect) view:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \min_{\gamma \in \Gamma} \sum_{(x,y) \in \gamma} [\|x - y\|],$$

where Γ is the set of all possible sets of correspondences between x and y.

Progressive GANs

- T. Karras et al. 2018



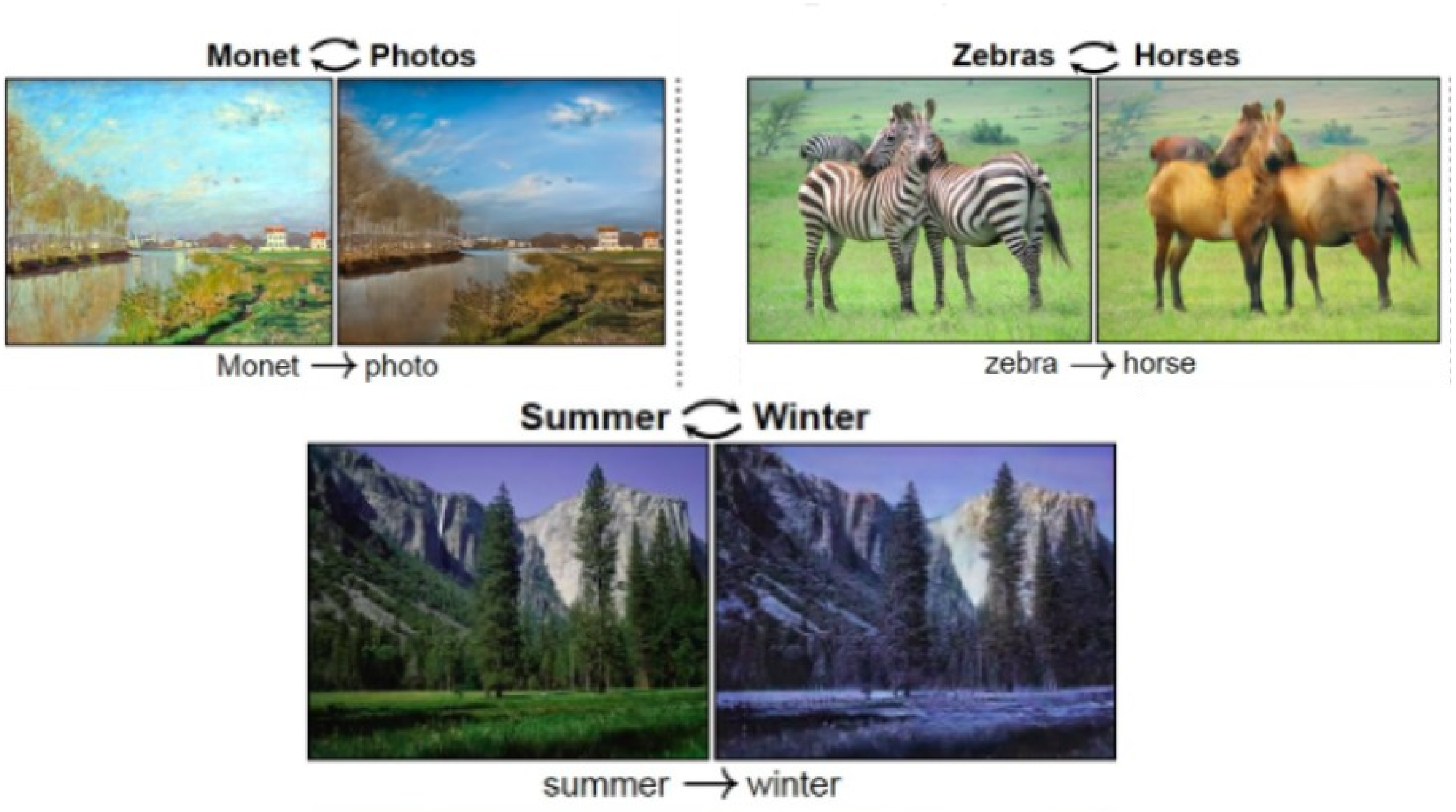
Progressive GANs: Results

Generated Image

Nearest Neighbor in the training set

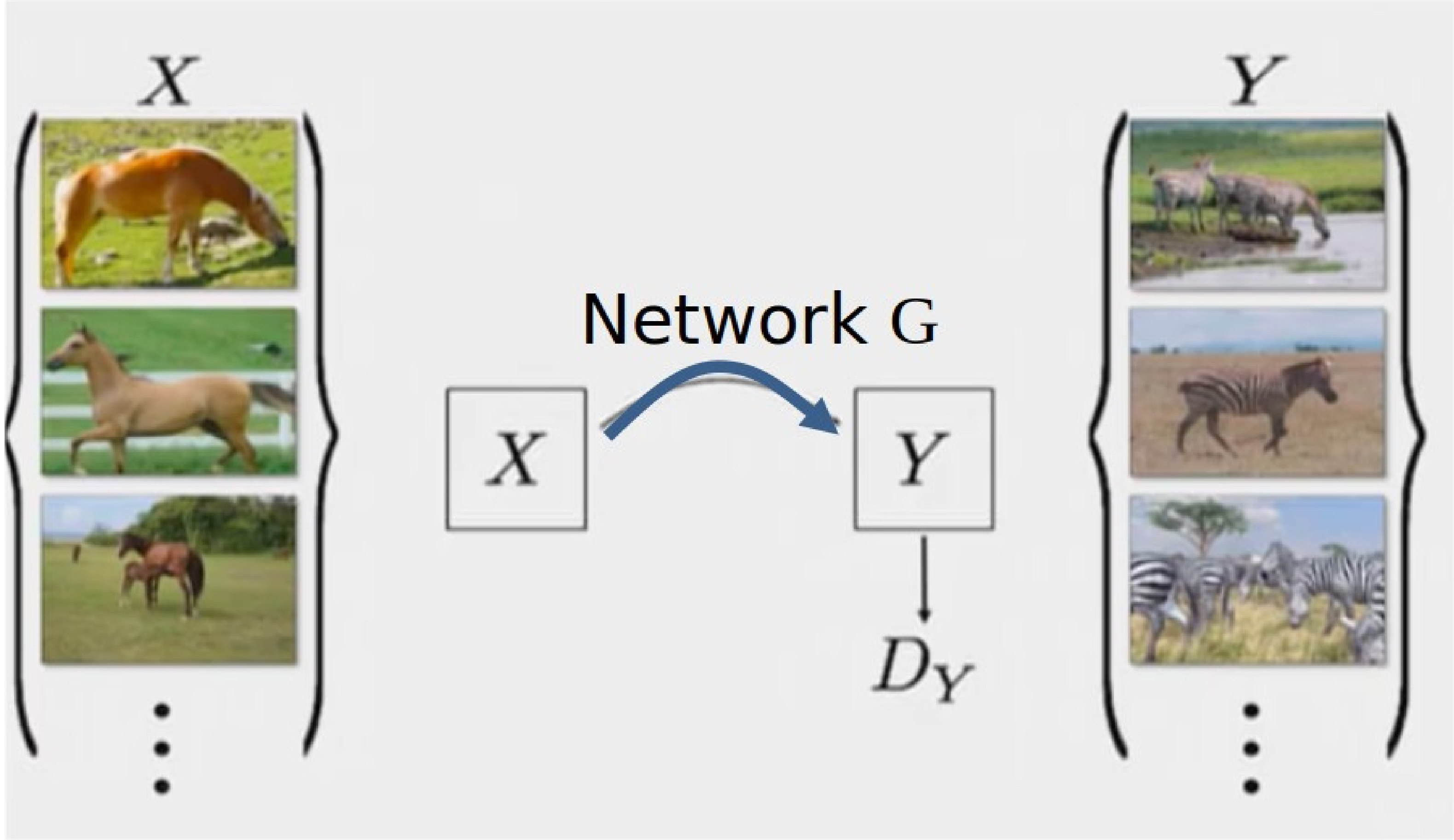
CycleGAN

- How can we make sure we preserve the content of an input image?



CycleGAN

- J.-Y. Zhu et al. 2017



CycleGAN

- J.-Y. Zhu et al. 2017