Generative Models

Slides adapted from: Toronto csc421, Stanford CS231 and Illinois cs498

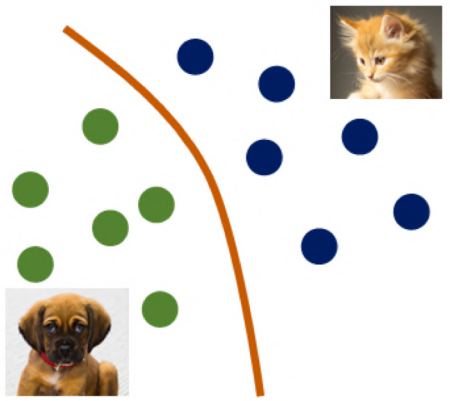
Good explanation of VAE and generative models exists in: https://youtu.be/3G5hWM6jqPk

Why Generative Models?

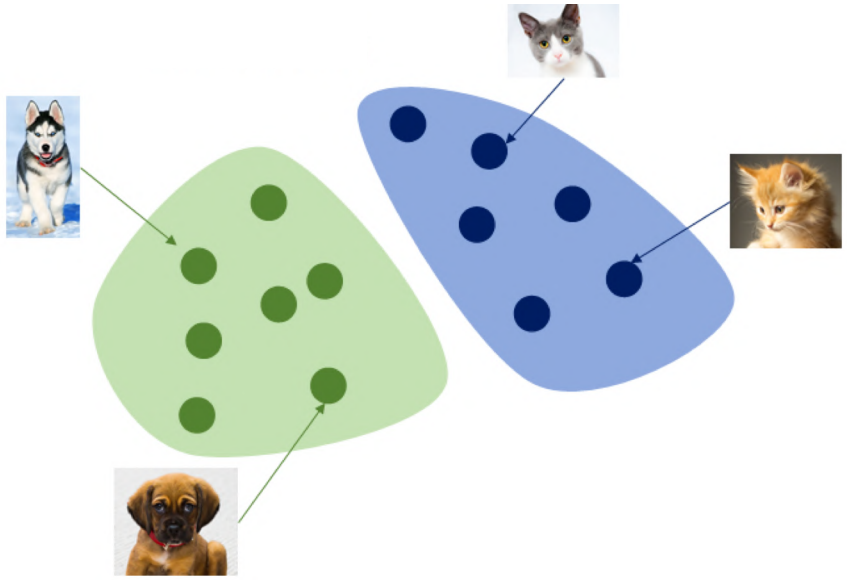
- So far, we've seen only discriminative models
  - Given an image $\boldsymbol{X}$, predict a label $\boldsymbol{Y}$
  - Estimates $\boldsymbol{P}(\boldsymbol{Y}|\boldsymbol{X})$
- Discriminative models have several key limitations
  - We don't know $\boldsymbol{P}(\boldsymbol{X})$, i.e. the probability of seeing a certain signal/image
  - Knowing $\boldsymbol{P}(\boldsymbol{X})$ is needed for non-classification tasks, such as: anomaly detection, signal reconstruction, signal (image) synthesis, Signal compression and representation.
- Generative models (in general) cope with all of above
  - Can model $\boldsymbol{P}(\boldsymbol{X})$
  - Can generate new signals/images

Source: Roger Grosse

Generative vs. Discriminative

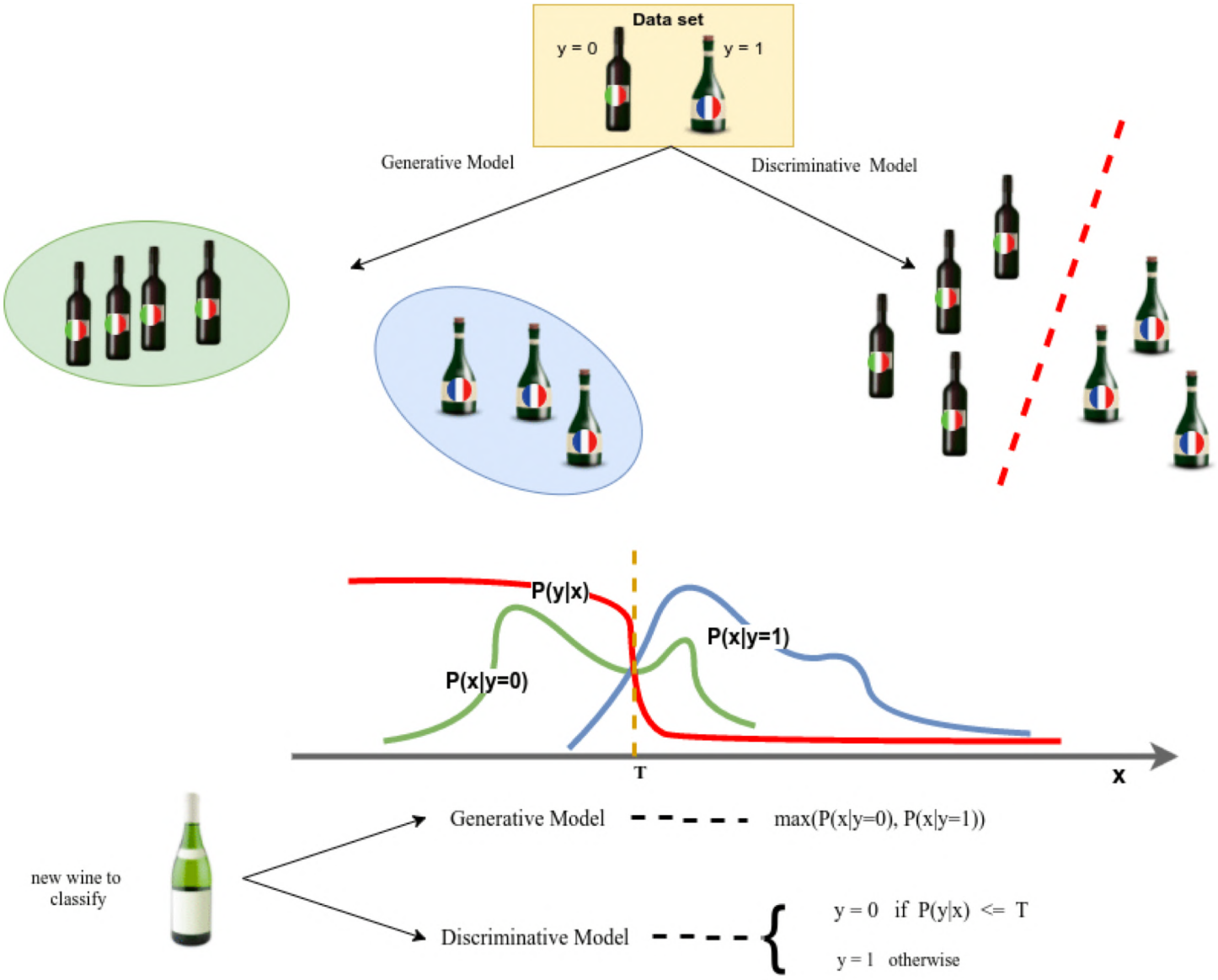


Discriminative Model



Generative Model

Generative vs. Discriminative

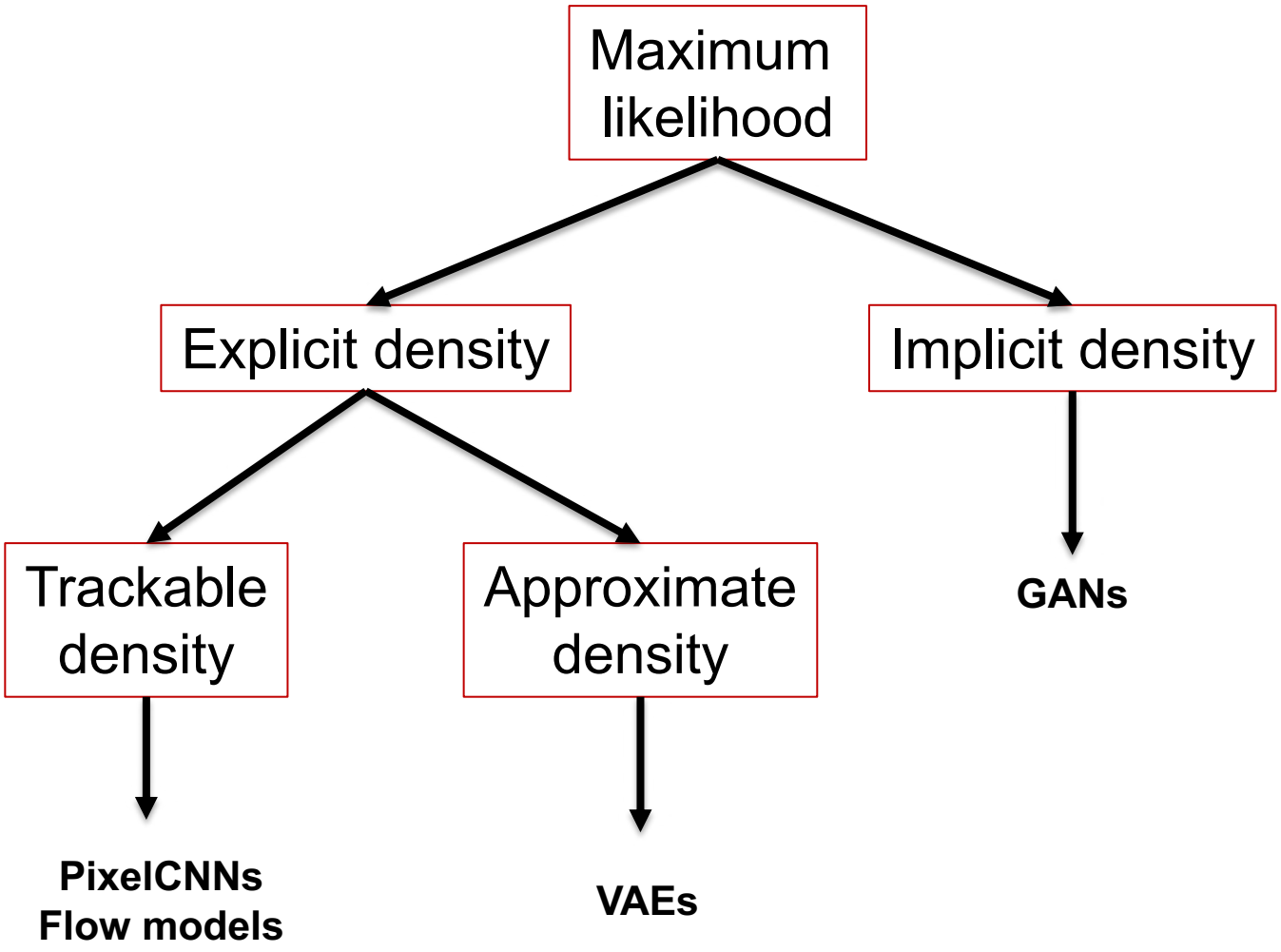


Overview

- In generative modeling, we'd like to train a network that models a distribution, such as a distribution over images.
- For example, face images can be seen as a particular distribution over the "image space".
- One way to judge the quality of the model is to sample from it.
- This field has seen rapid progress in the last few years:

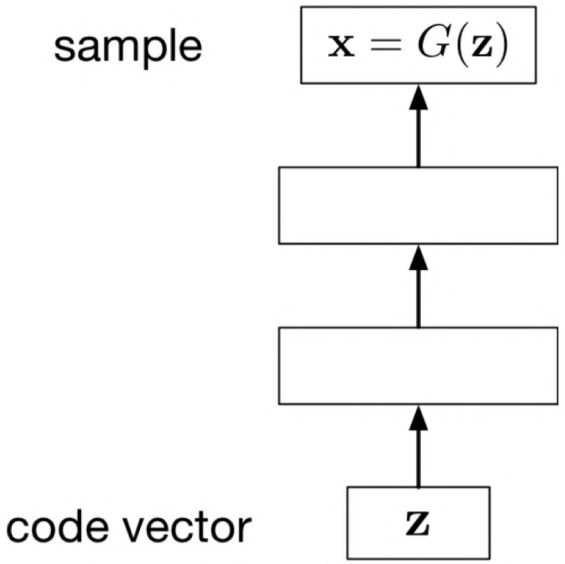
Source: Roger Grosse

Taxonomy of Generative Models



Implicit Generative Models

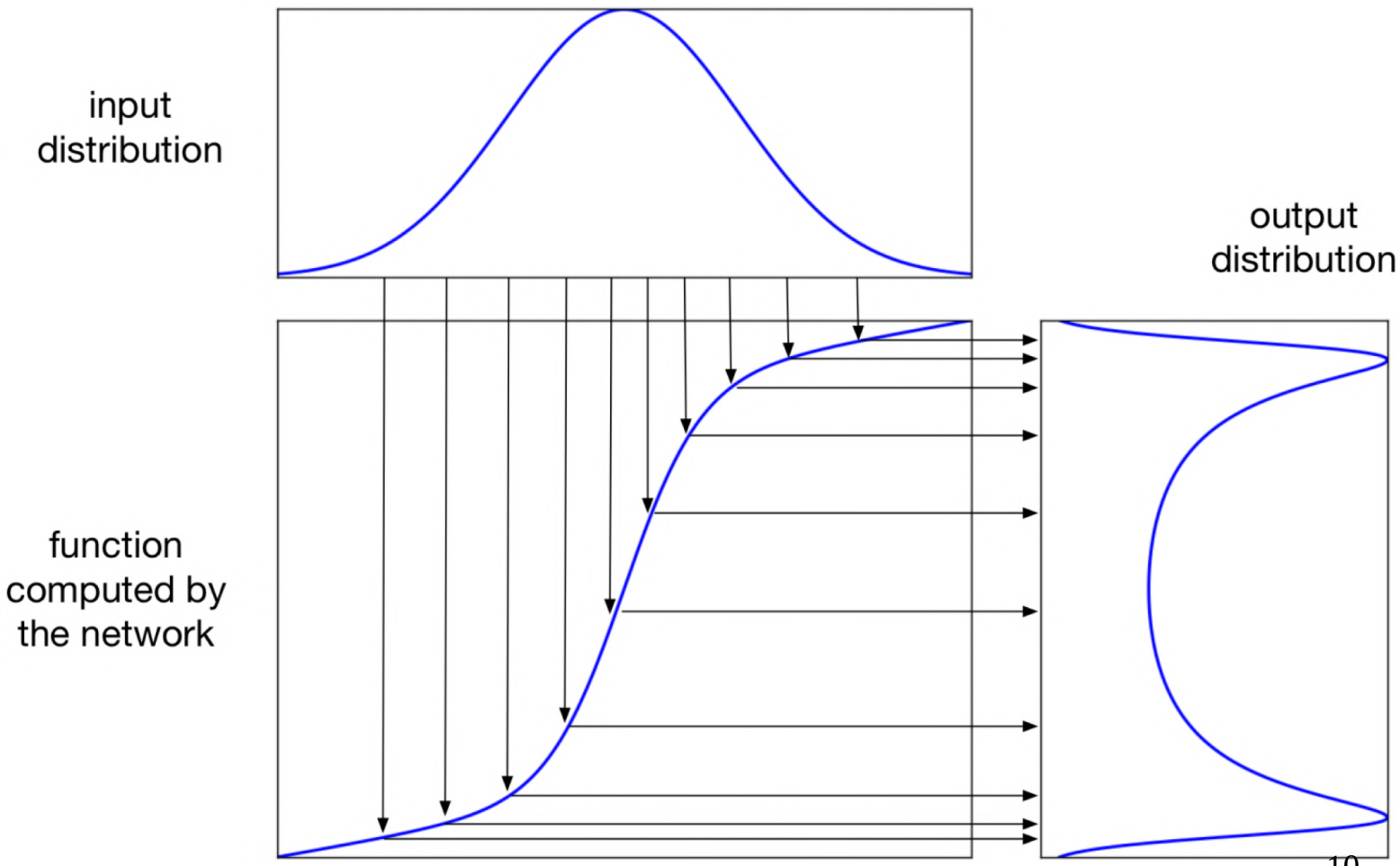
- Implicit generative models implicitly define a probability distribution. We don't know the exact distribution but can sample from it.
- Start by sampling the code/latent vector ***z*** from a fixed, simple distribution (e.g. spherical Gaussian)
- The generator network computes a differentiable function G, mapping ***z*** to an ***x*** in data space:



Source: Roger Grosse

Implicit Generative Models

A 1D example:



Source: Roger Grosse

Implicit Generative Models

A 2D example:

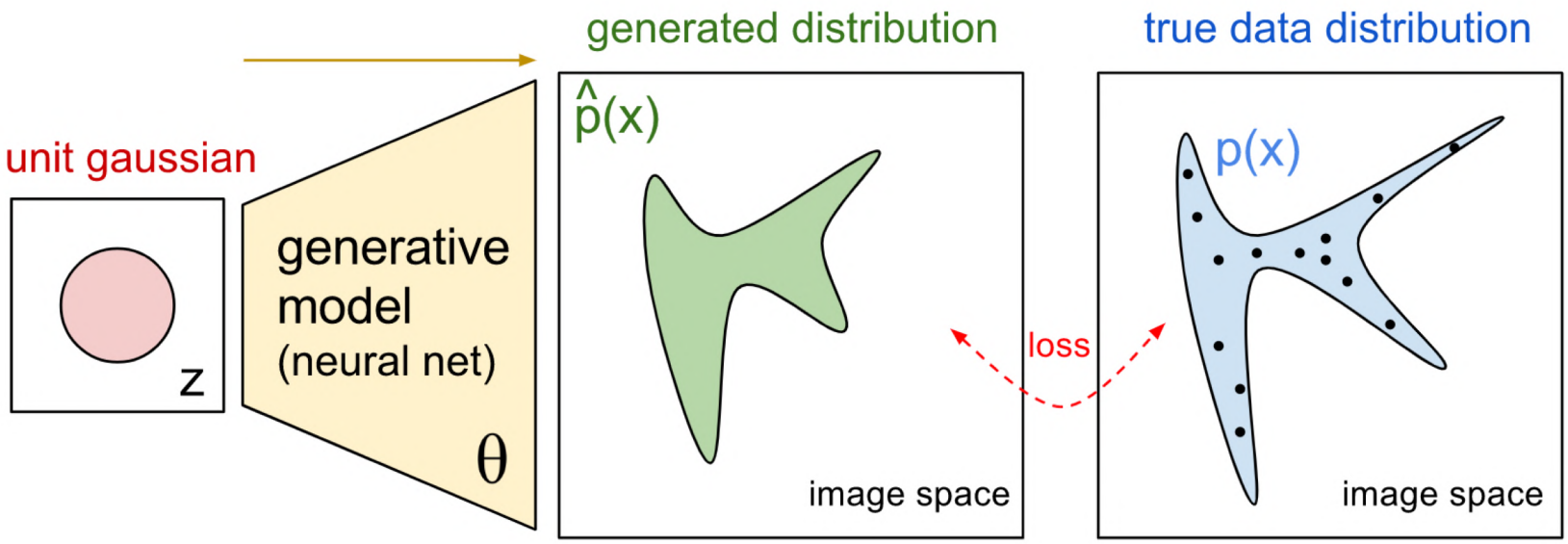
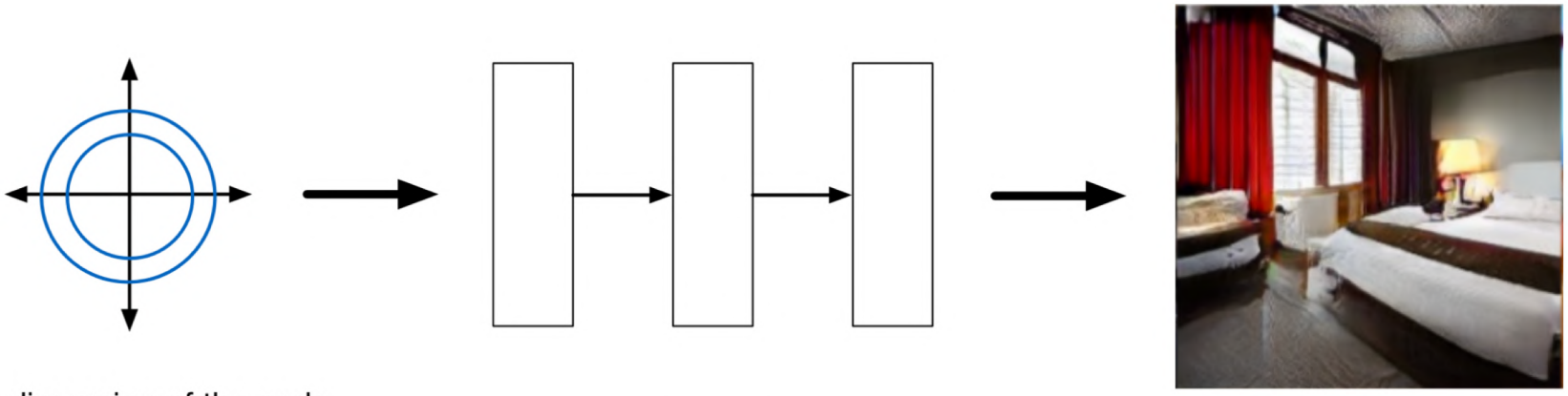


Image source: https://openai.com/blog/generative-models/

Source: Roger Grosse

Implicit Generative Models for images



Each dimension of the code vector is sampled independently from a simple distribution, e.g. Gaussian or uniform.

This is fed to a (deterministic) generator network.

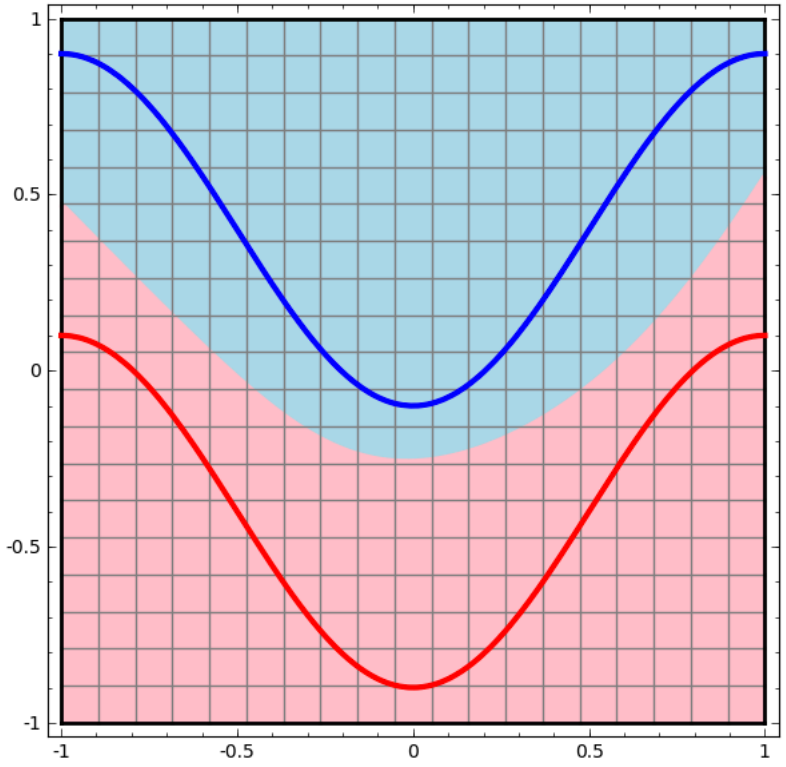
The network outputs an image.

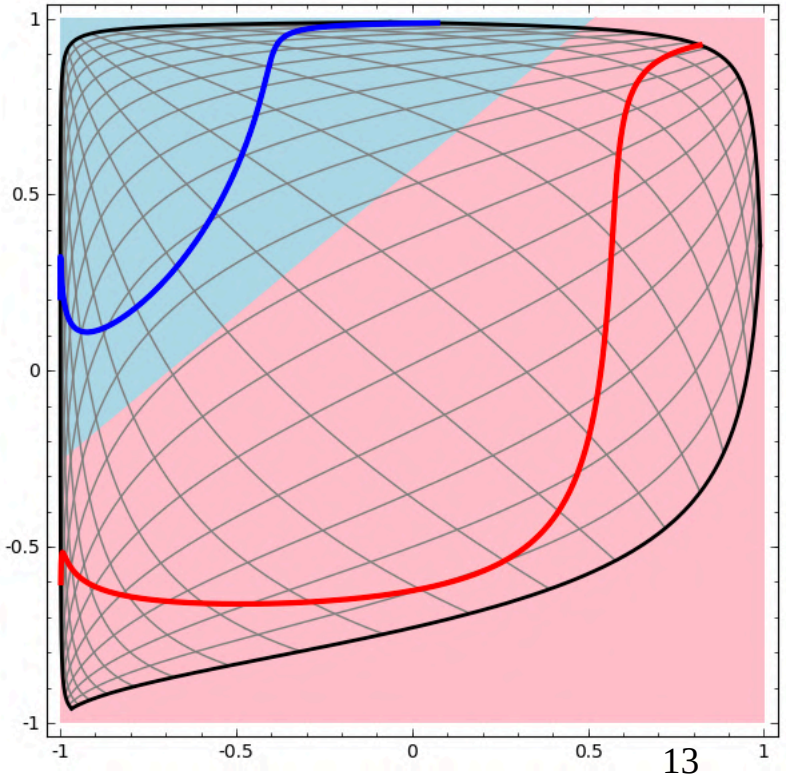
- This sort of architecture sounds impossible, but amazingly, it works!

Source: Roger Grosse

Multilayer: Space Transformation

- Recall, the space transformation for linear classification.
- Similarly, DNN can apply space transformations for imlicit generative models.





Source: http://colah.github.io

Implicit Generative Models

- If we had some functional criterion for evaluating the quality of a sample, then we could compute the gradient of the quality with respect to the network parameters, and update them to make the samples a little better.
- Generative Adversarial Networks (GAN):
  - Train two different networks: the **generator** and a **discriminator**
  - The generator tries to produce probable samples
  - The discriminator network tries to figure out whether an image came from the training set or from the generator
  - The generator tries to fool the discriminator network
  - They improve over time until convergence

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, NIPS 2014

Source: Roger Grosse

The Original GAN Paper

Published in 2014
Cited by 39215 papers

Generative Adversarial Nets

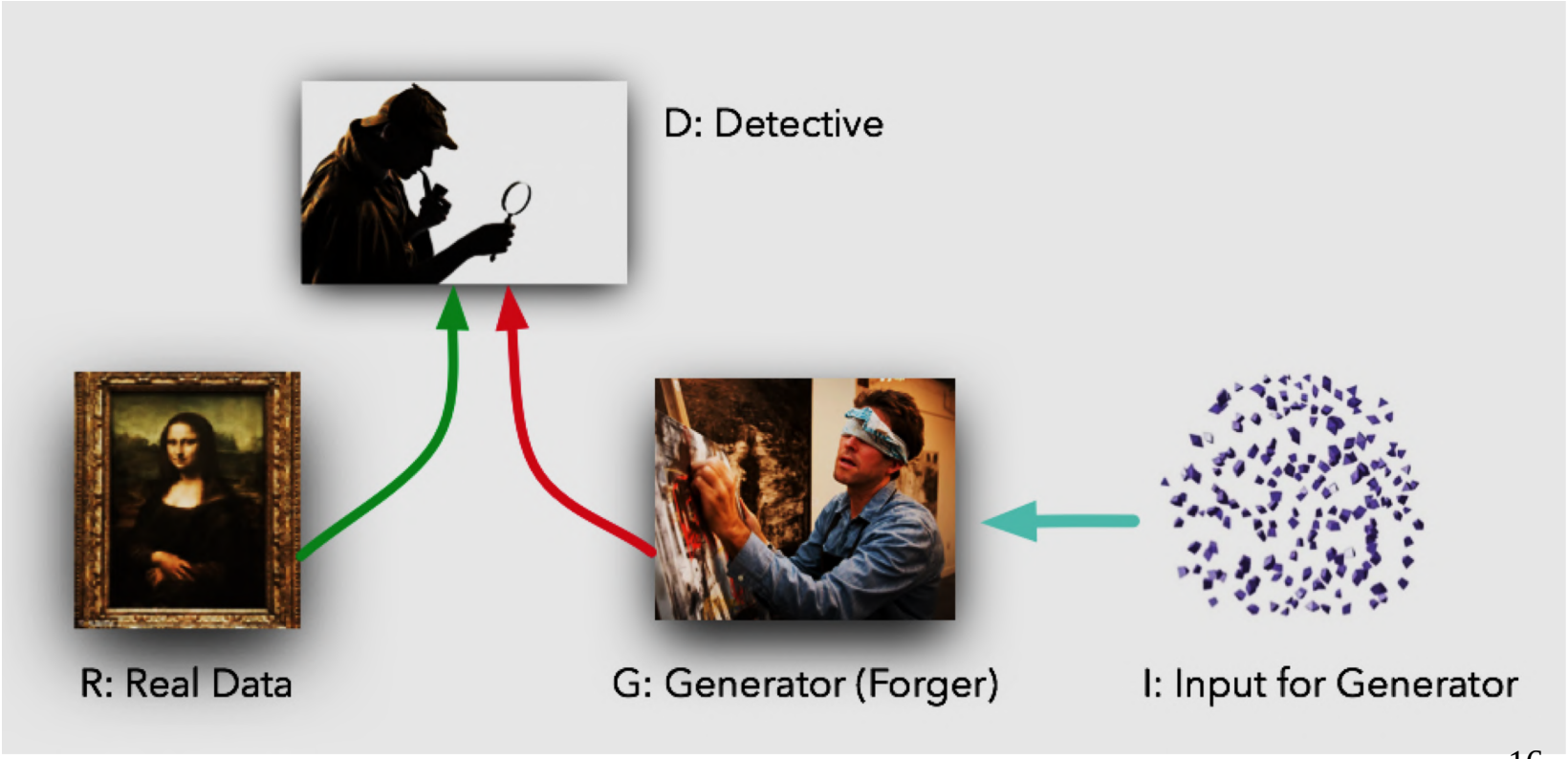
Ian J. Goodfellow,* **Jean Pouget-Abadie**,† **Mehdi Mirza**, **Bing Xu**, **David Warde-Farley**,
Sherjil Ozair,‡ **Aaron Courville**, **Yoshua Bengio**§
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

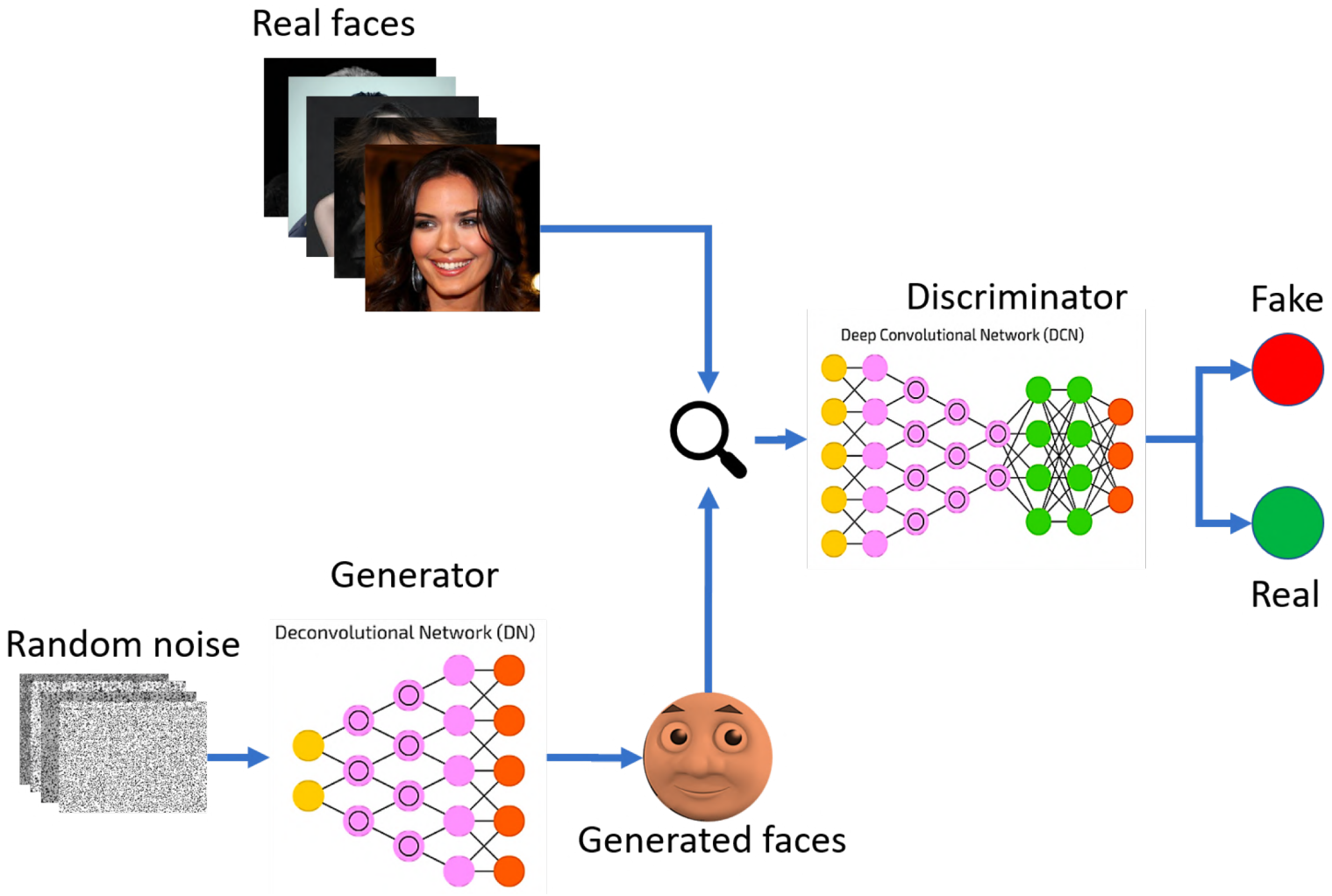
We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, NIPS 2014.

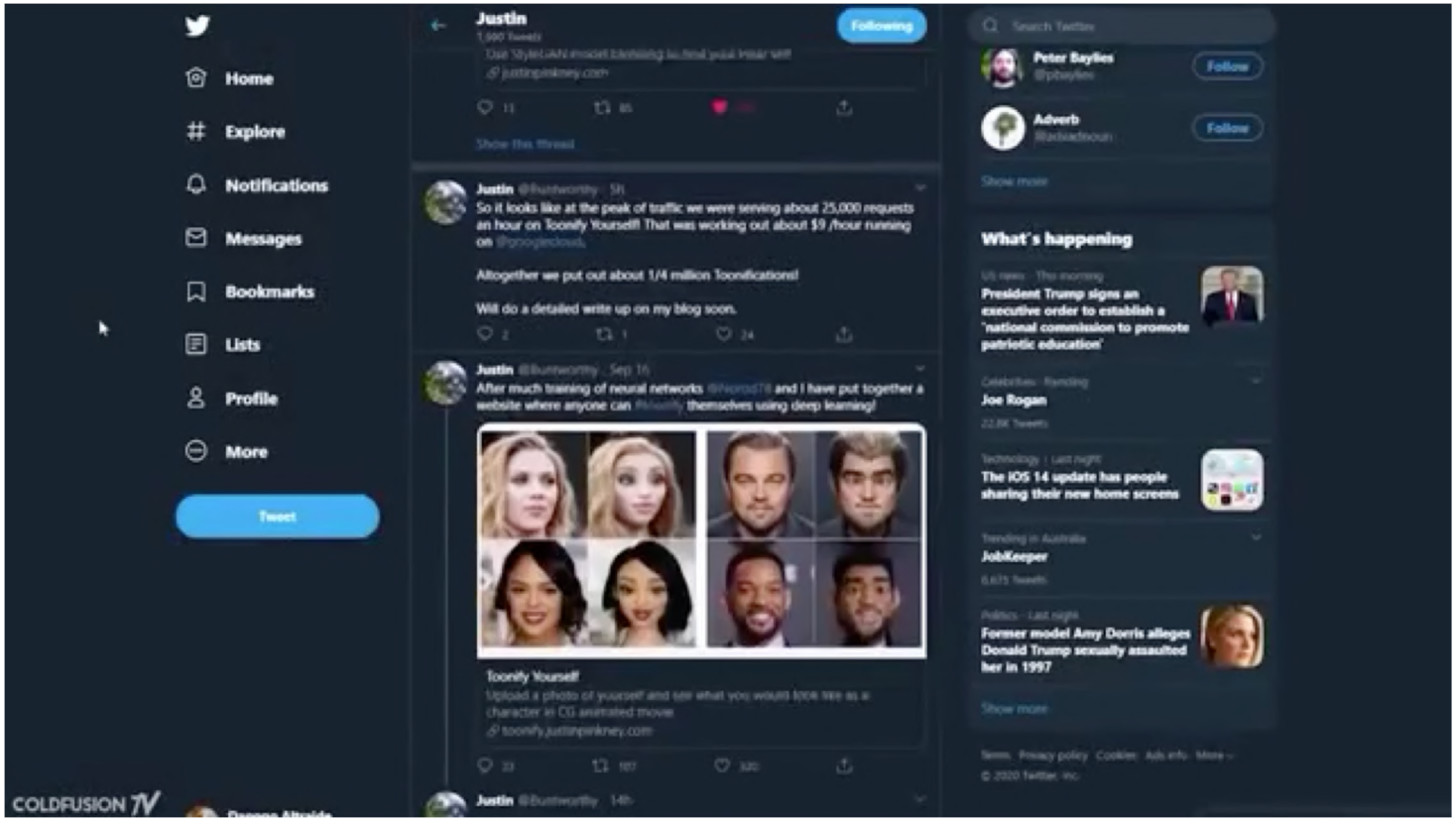
GAN Analogy


D: Detective
R: Real Data
G: Generator (Forger)
I: Input for Generator

מכונות שלומדות ממכונות:

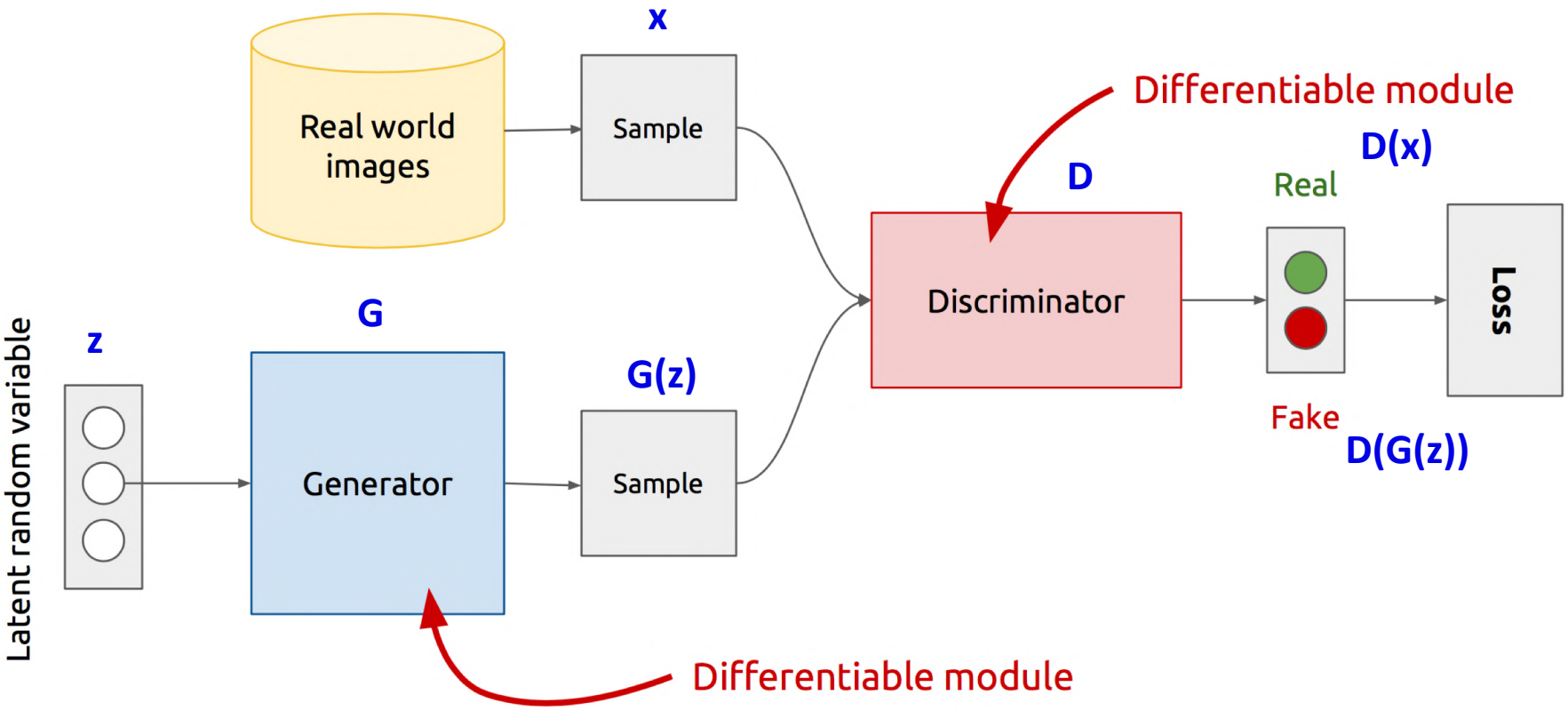


The Power of GAN - a Teaser…

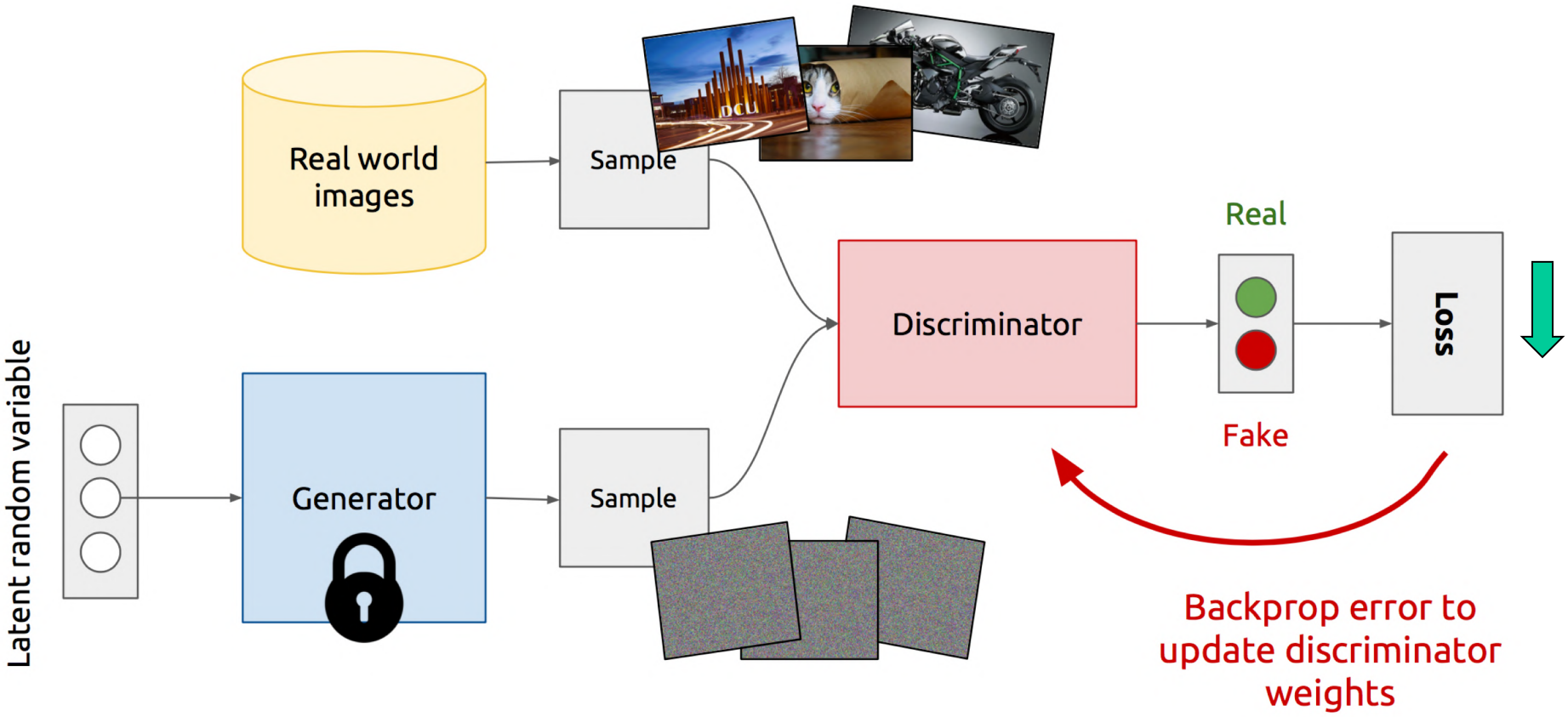


https://www.youtube.com/watch?v=KZ7BnJb30Cc&ab_channel=ColdFusion

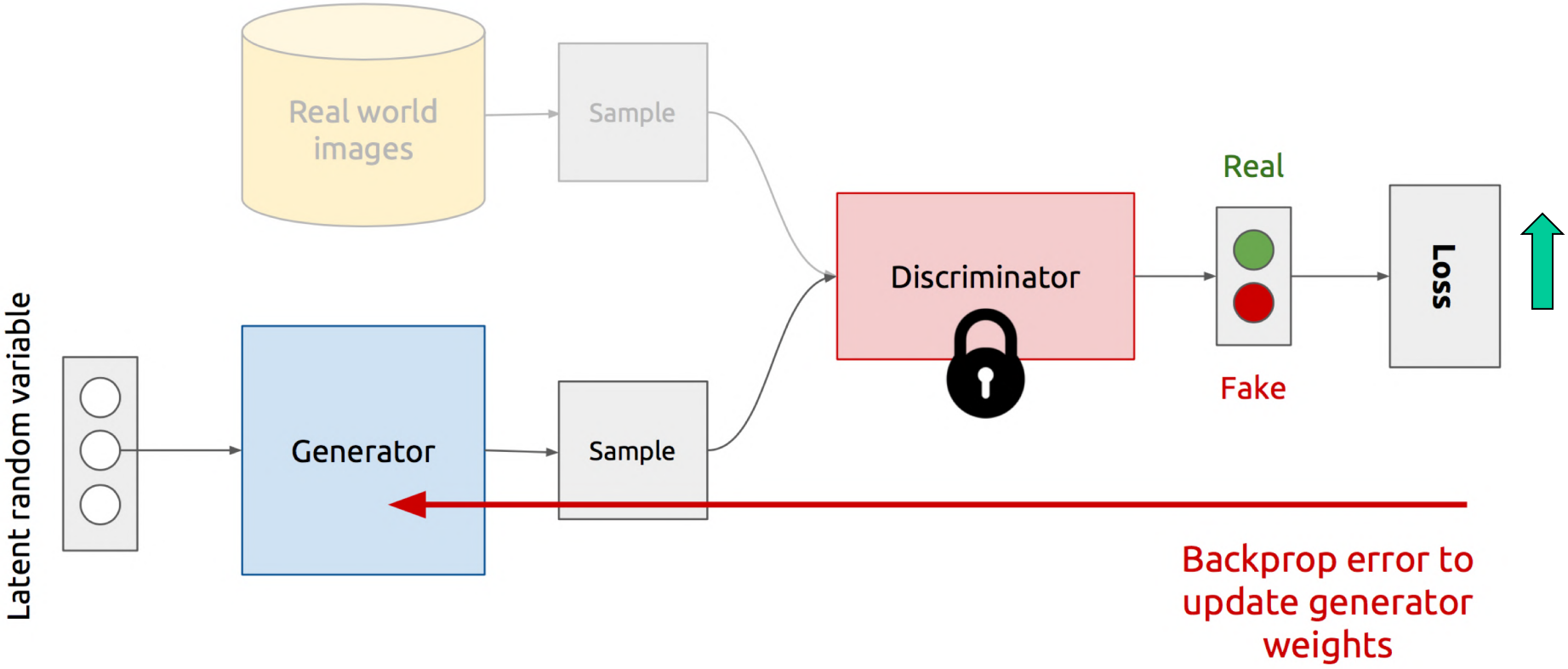
The GAN's Architecture



Training the Discriminator

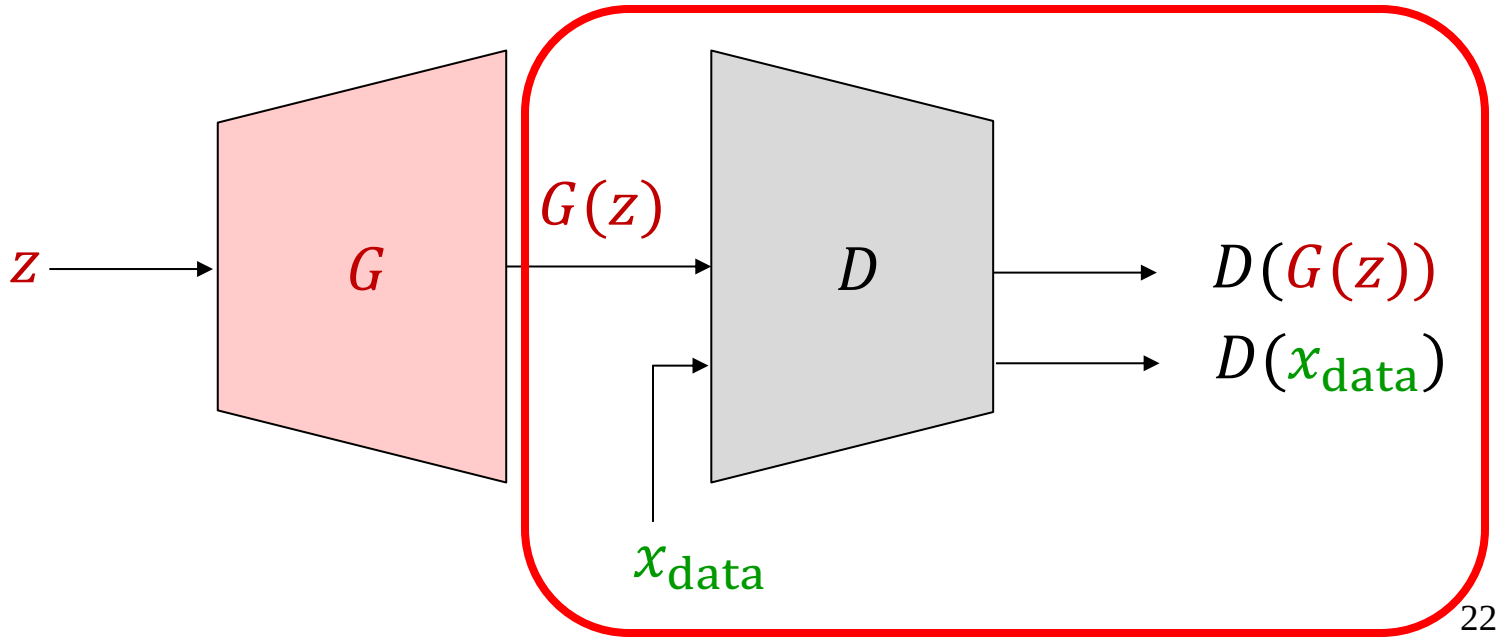


Training the Generator



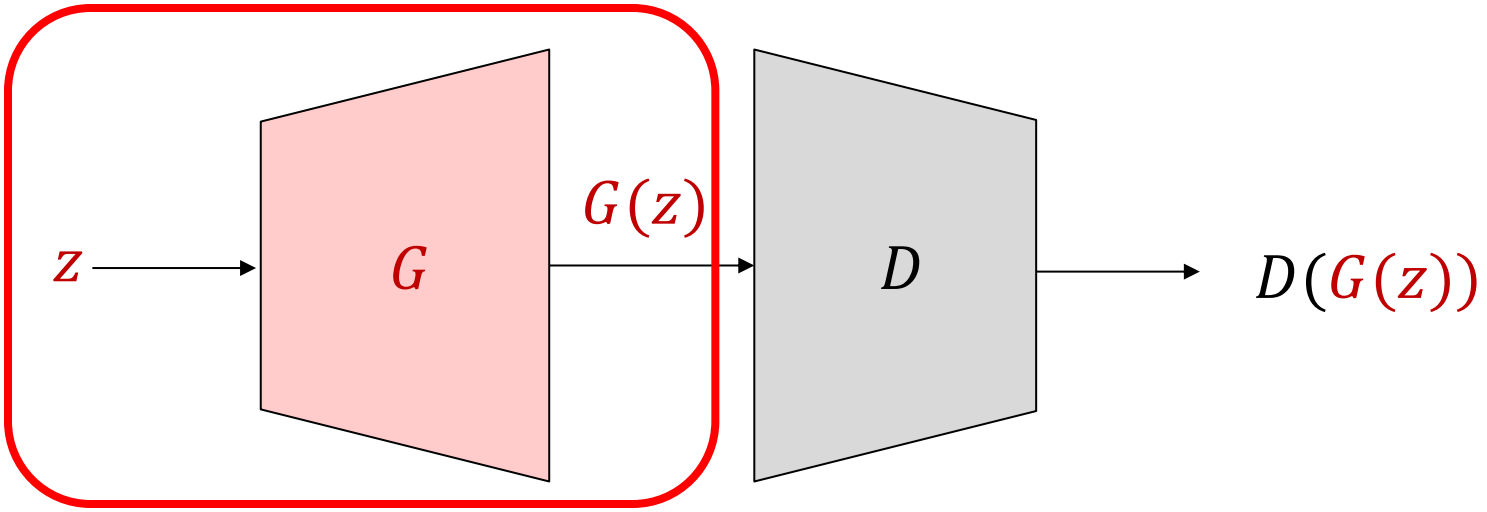
GAN: Conceptual picture

- **Update discriminator**: push $D(x_{\text{data}})$ close to 1 and $D(G(z))$ close to 0
- The generator is a "black box" to the discriminator



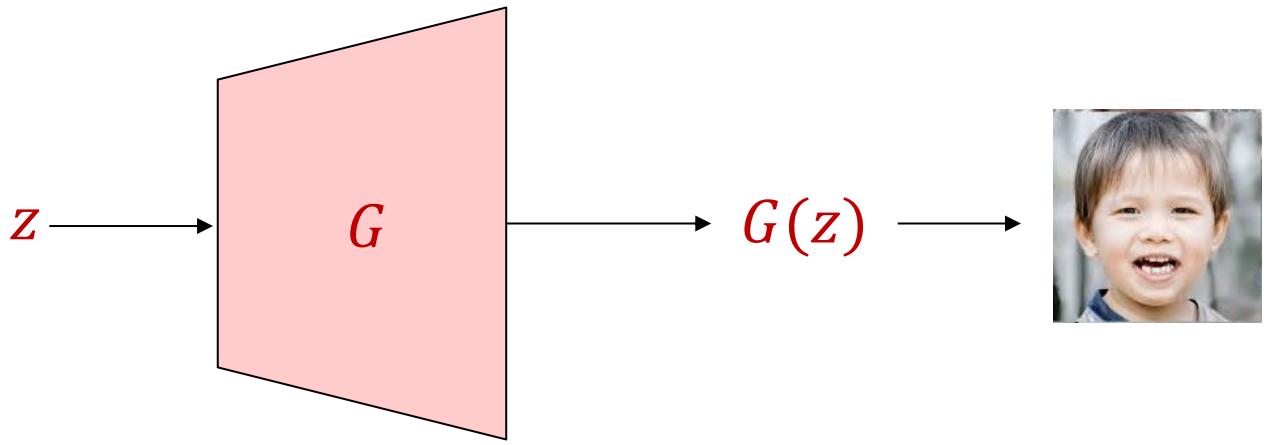
GAN: Conceptual picture

- **Update generator**: push $D(G(z))$ close to 1
  - Requires back-propagating through the composed generator-discriminator network
  - The generator is exposed to real data only via the output of the discriminator (and its gradients)



GAN: Conceptual picture

- **Test time**:



GAN's Loss functions

- Let *D* denotes the discriminator's predicted probability of being a real data: $D(x) \in [0,1]$. Let $p(x)$ be the real distr. and $q(z)$ the latent distr.
- **Discriminator's** cost function:

$$\mathcal{L}_D = \mathbb{E}_{x\sim p(x)}[-\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[-\log\left(1 - D(G(z))\right)\right]$$

For real data For generated data

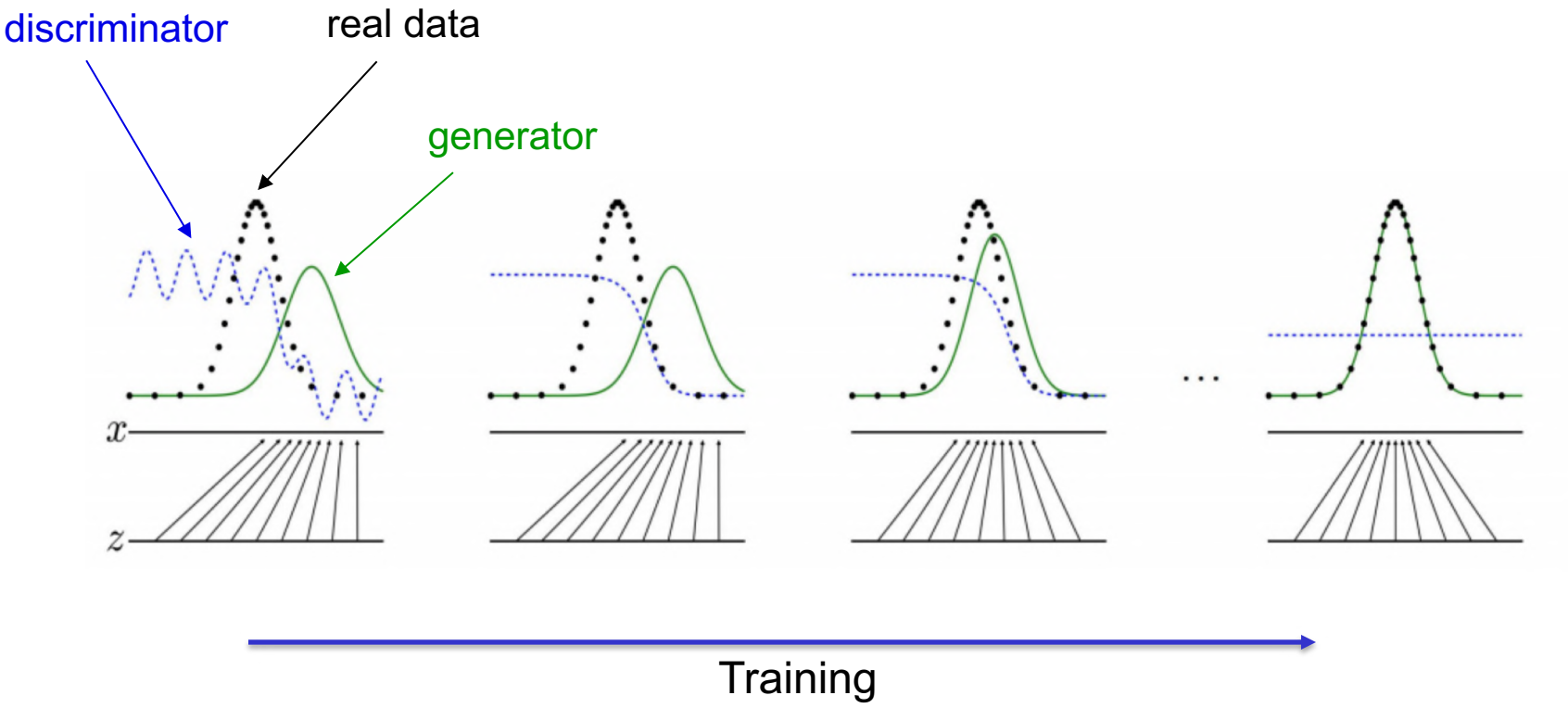
- **Generator's** cost function, the opposite of $\mathcal{L}_D$:

$$\mathcal{L}_G = \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

- This is called the minimax formulation, since the generator and discriminator are playing a zero-sum game against each other:

$$\mathcal{L} = \max_G \min_D \mathcal{L}_D$$

Training GAN



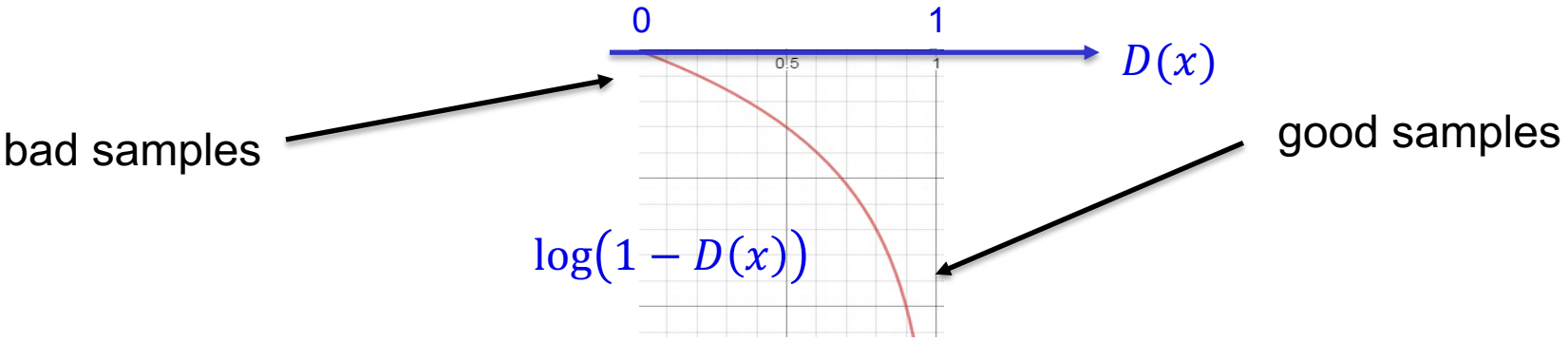
Alternating minimization

Generator ⟷ Discriminator

A Better Loss Function

- We introduced the minimax cost function for the generator:

$$\mathcal{L}_G = \mathbb{E}_{z \sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$



Problem:

- Gradient goes to 0 if D is confident, i.e. where $D\left(G(z)\right) \to 0$
- I.e., If the generated sample is really bad, the discriminator's prediction is close to 0, but the generator's cost is flat.

A Better Loss Function

Solution:

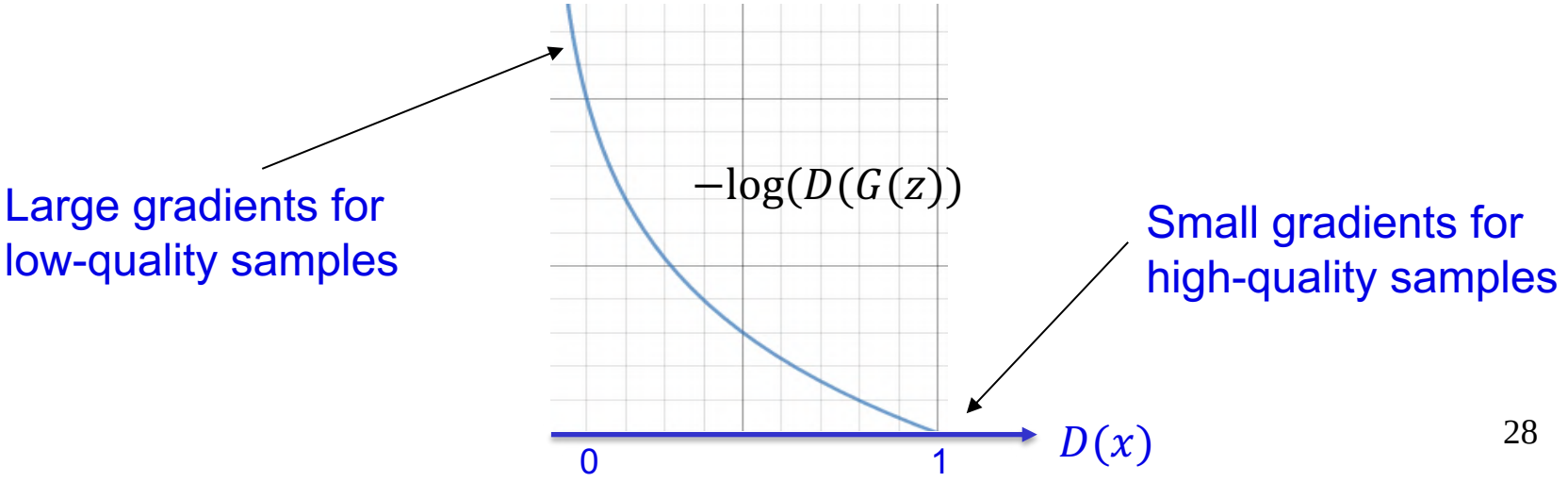
- Original minimax cost:

$$\mathcal{L}_G = \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

- Modified generator cost:

$$\mathcal{L}_G = \mathbb{E}_{z\sim q(z)}\left[-\log D(G(z))\right]$$

- This fixes the saturation problem.



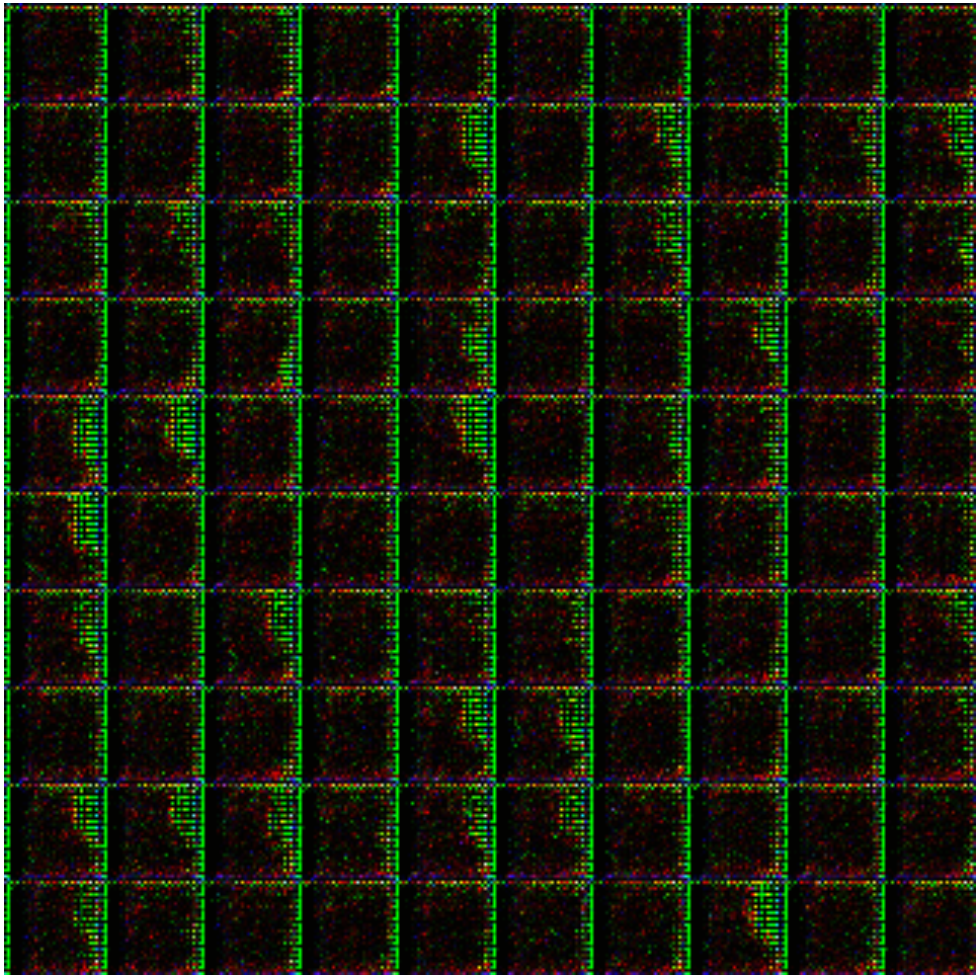
Original GAN Results

- Originally paper: I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, NIPS 2014

Yann LeCun: "GAN is the most interesting idea in the last 10 years in ML"

Nearest real image for sample to the left

Generator in Action



Game Theory

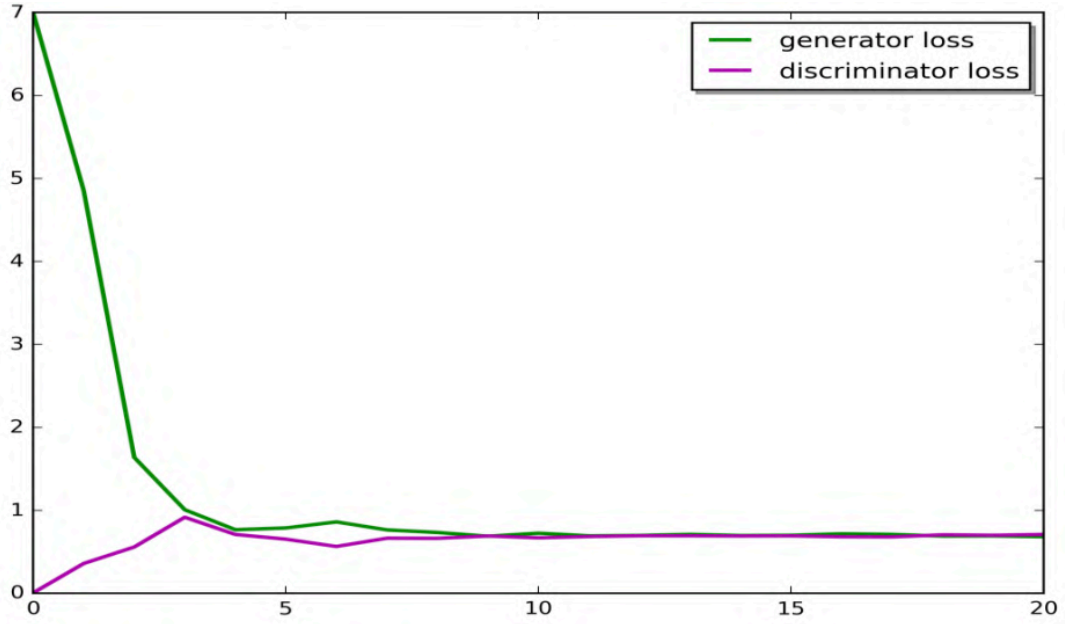
The basic idea of GANs is originated from game theory.

Zero-sum game:

- Players compete for a fixed and limited pool of resources.
- In zero-sum games, each player tries to set things up so that the other player's best move is of as little advantage as possible. This is called a **minimax** technique.
- When players have reached their peak ability given the other player's abilities is called a **Nash equilibrium**.
- In such a case each player is at its best configuration with respect to the other.

Game Theory

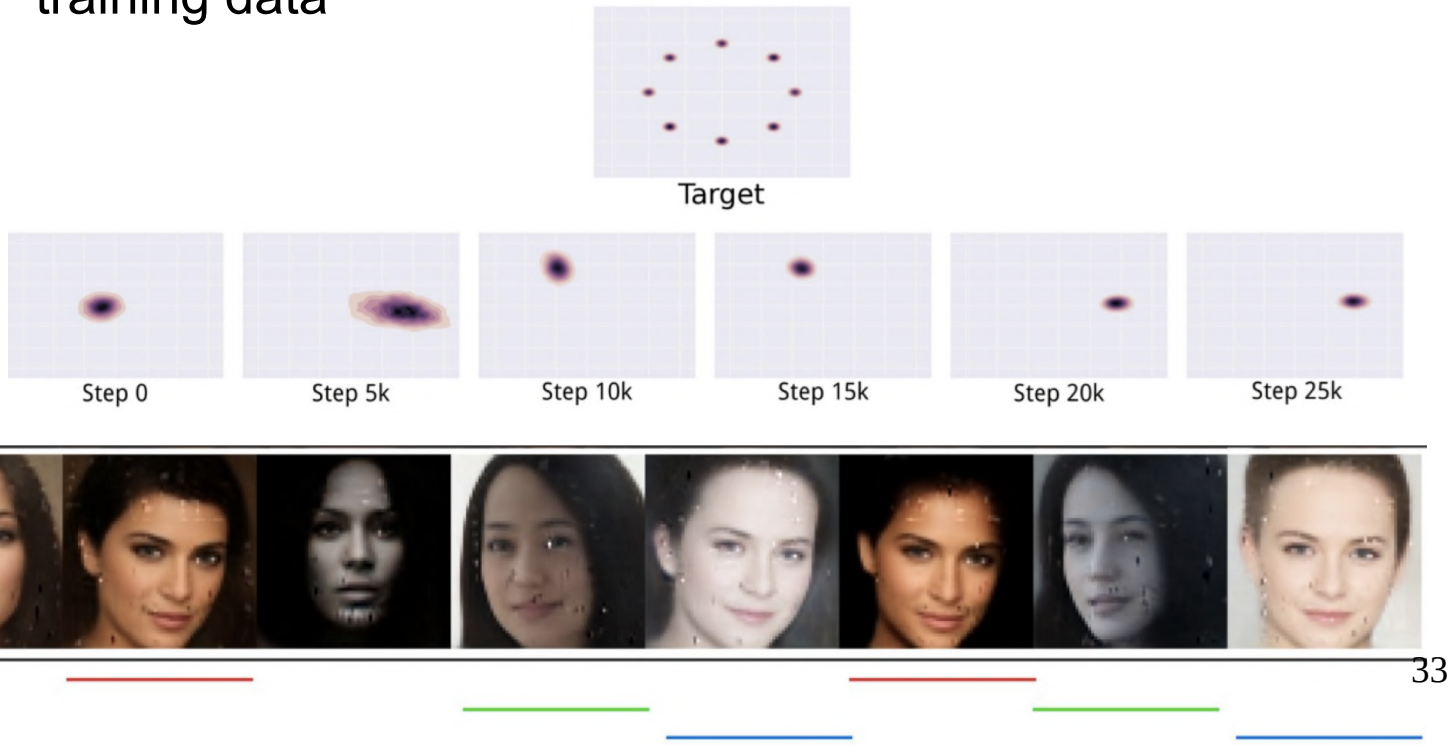
- In GAN framework, the Generator and the Discriminator play a Zero-sum Game (what is the resource?).
- Our goal in training GAN is to produce two networks that are each as good as they can be. In other words, we don't end up with a "winner."



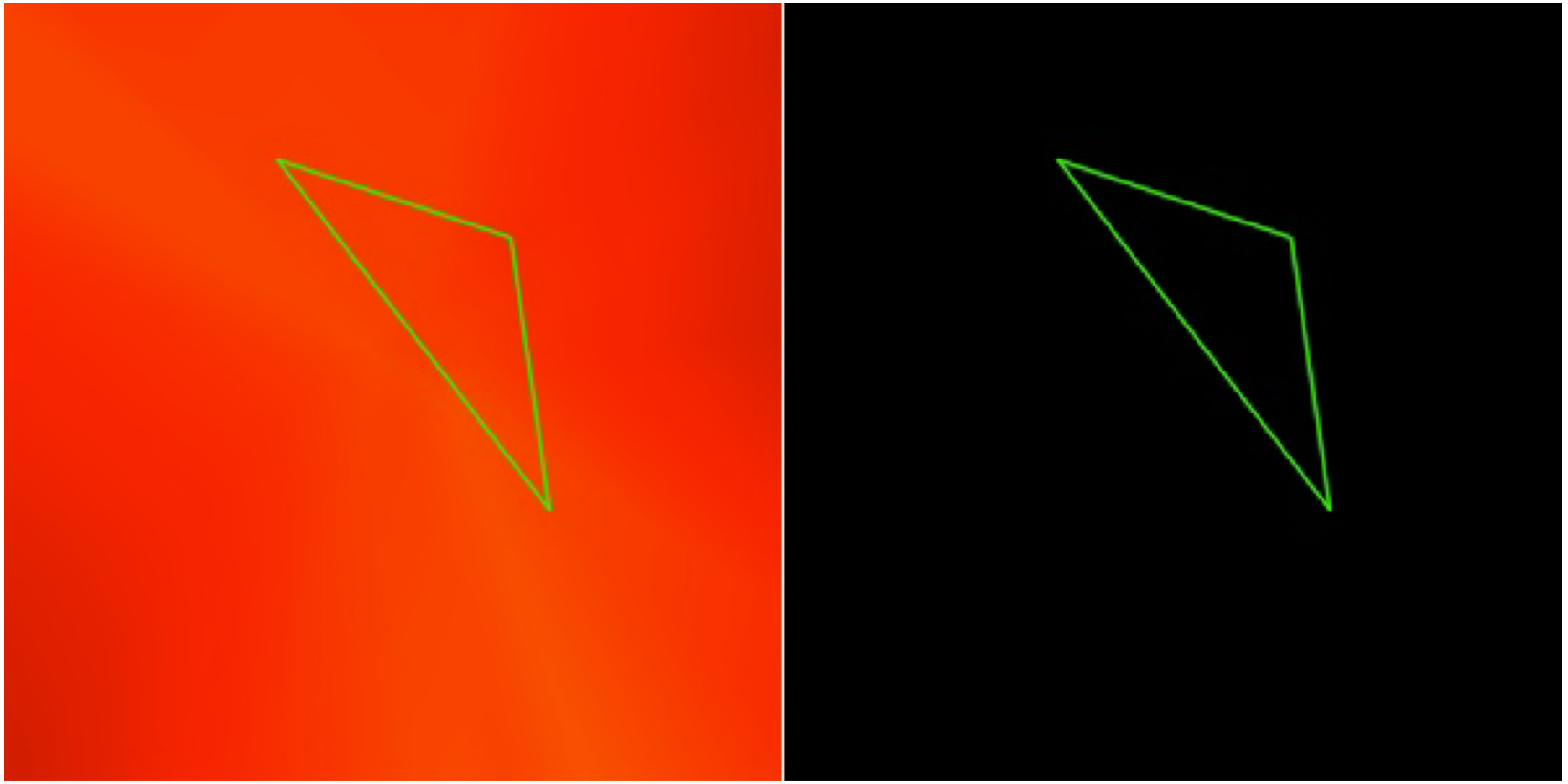
From Matthew Stewart: Introduction to Turing Learning and GANs

Problems with GAN training

- Stability
  - Parameters can oscillate or diverge
  - Behavior very sensitive to hyper-parameter
- Mode collapse
  - Generator ends up modeling only a small subset of the training data

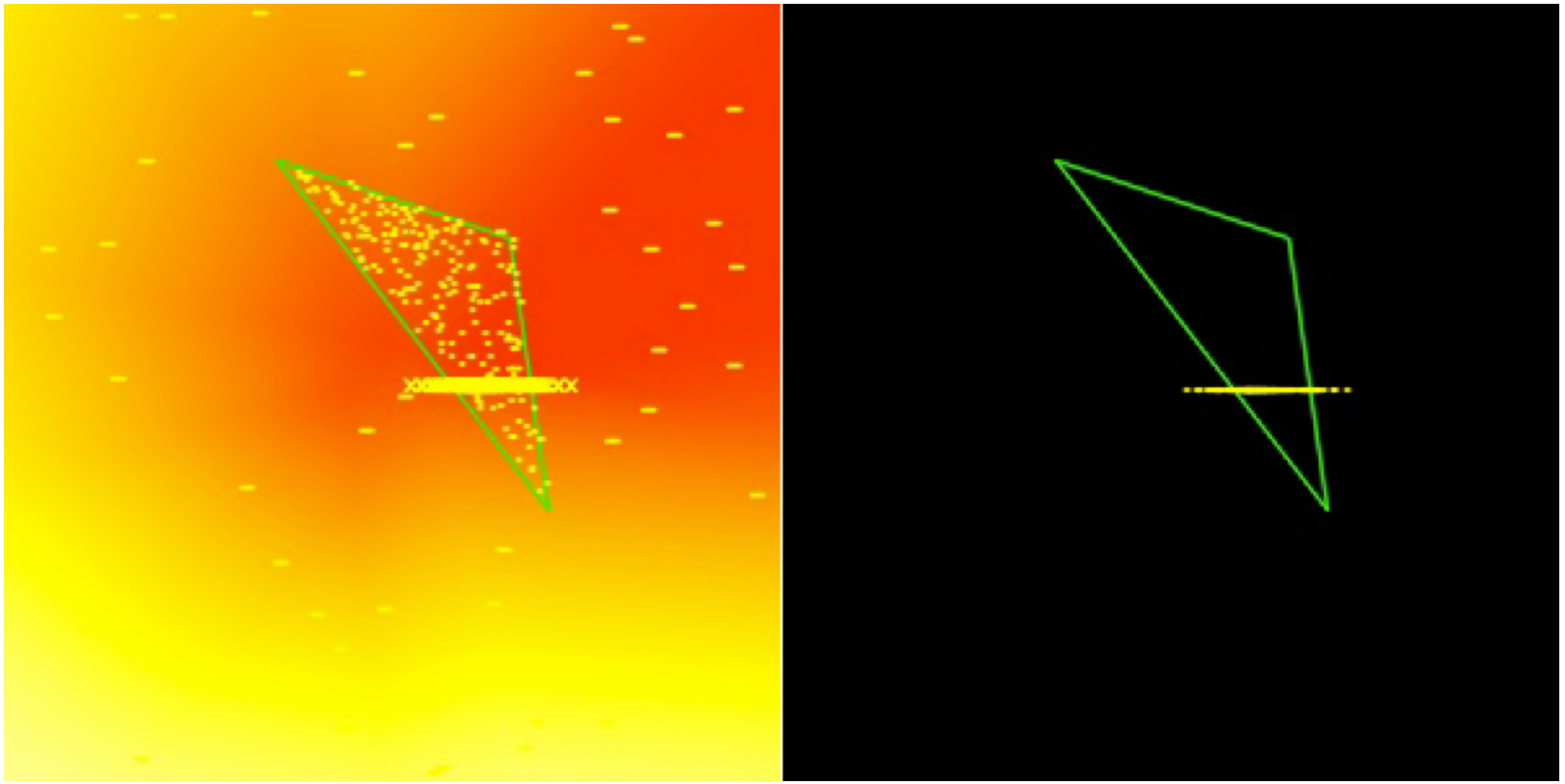


Problems with GAN training



Training D then G (mode collapse)

Problems with GAN training



Alternating minimization (bad convergence)

From: https://towardsdatascience.com/this-will-change-the-way-you-look-at-gans-9992af250454

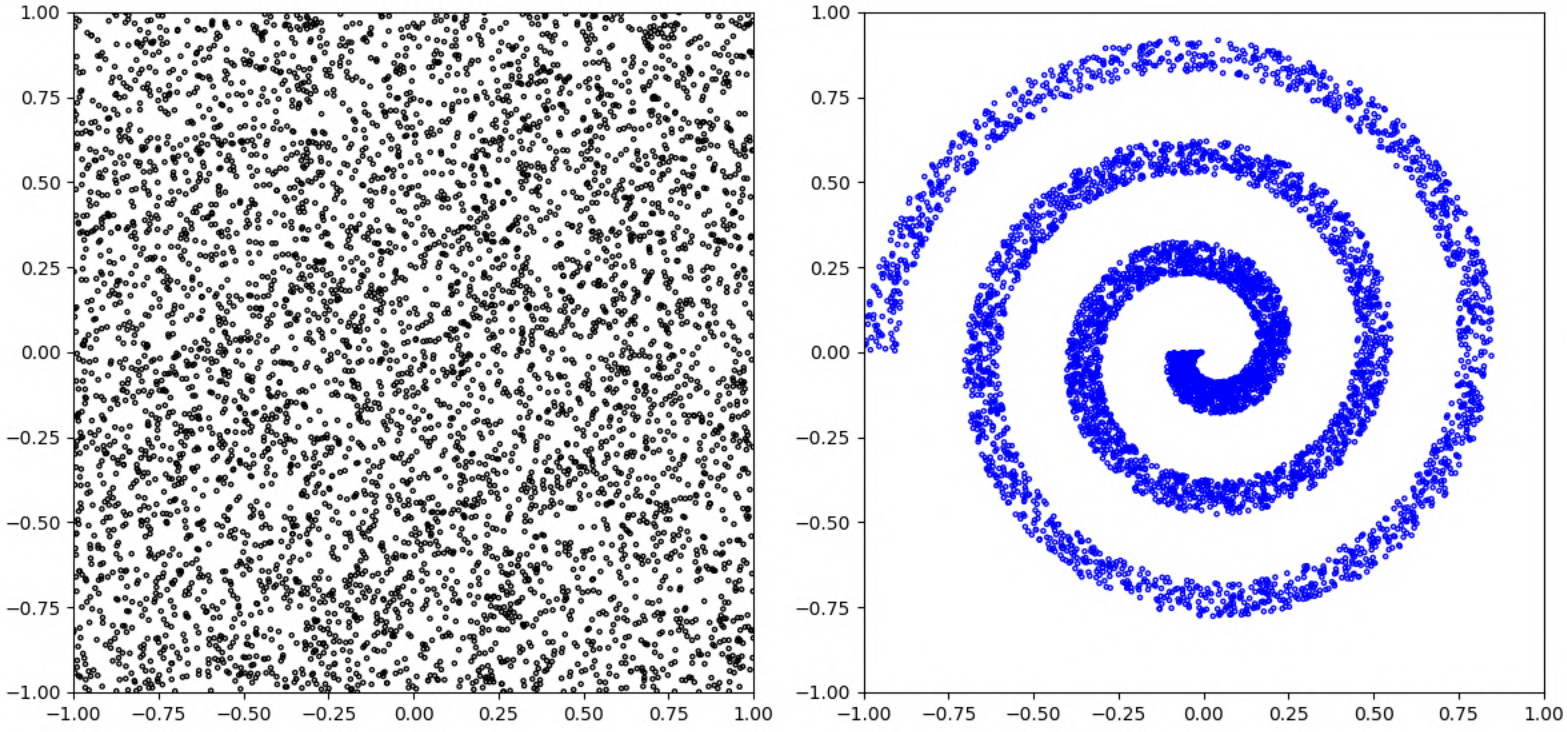


Figure 1: The input space (left) and corresponding output (right) of the target function. The input space will commonly be referred to as the latent space, and the output space is referred to as the sample space.

From: https://towardsdatascience.com/this-will-change-the-way-you-look-at-gans-9992af250454

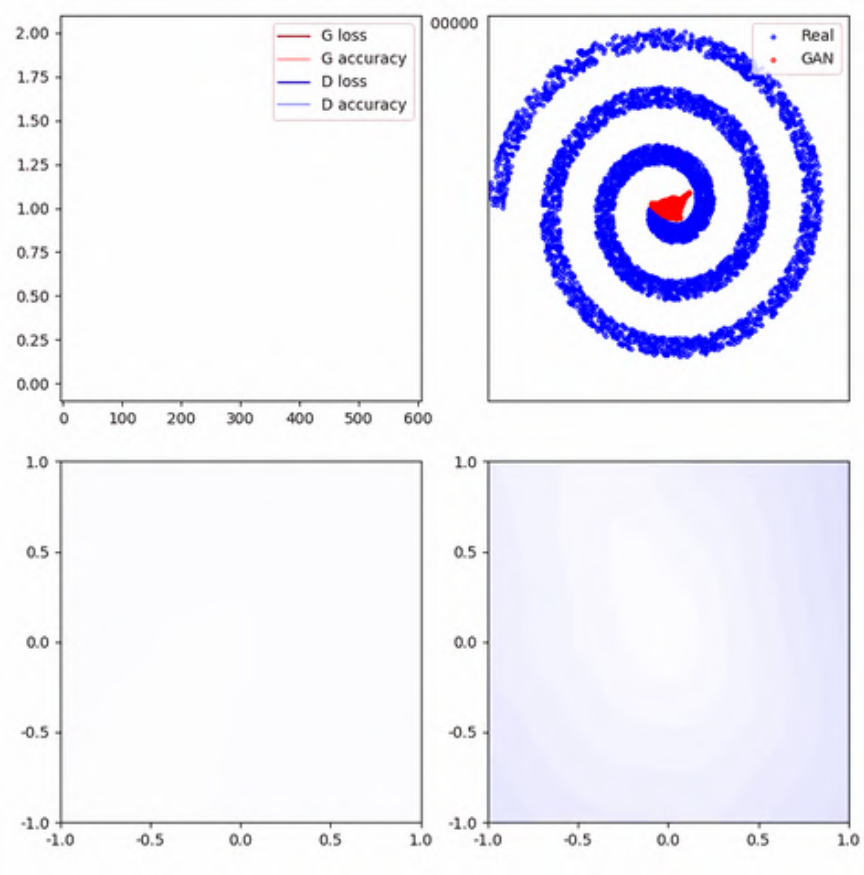


Figure 8: An animation of the GAN training process. As figure 3, but using better optimizer parameters

From: https://towardsdatascience.com/this-will-change-the-way-you-look-at-gans-9992af250454

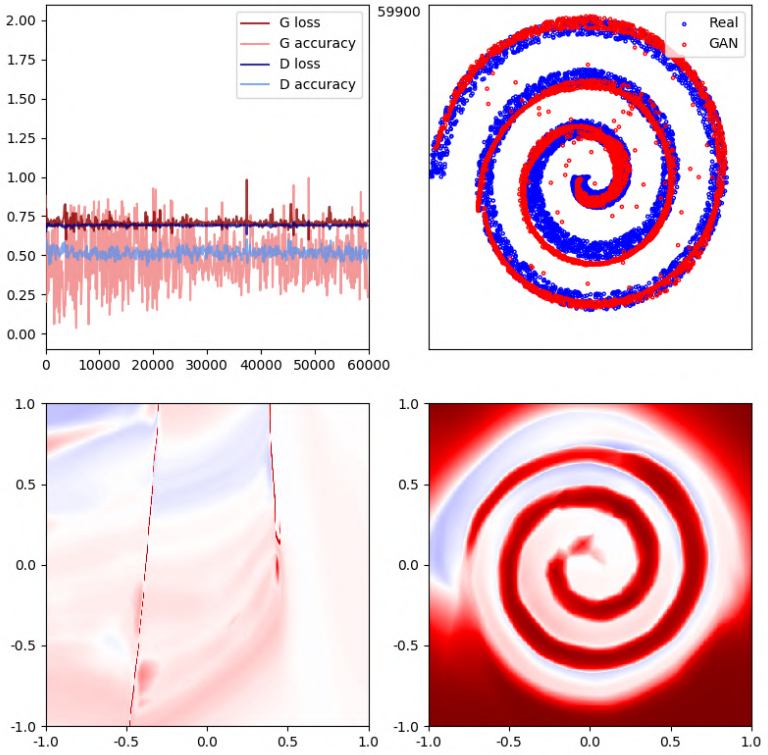


Figure 9: The result of training the GAN with a reduced learning rate for 600000 steps

From: https://towardsdatascience.com/this-will-change-the-way-you-look-at-gans-9992af250454

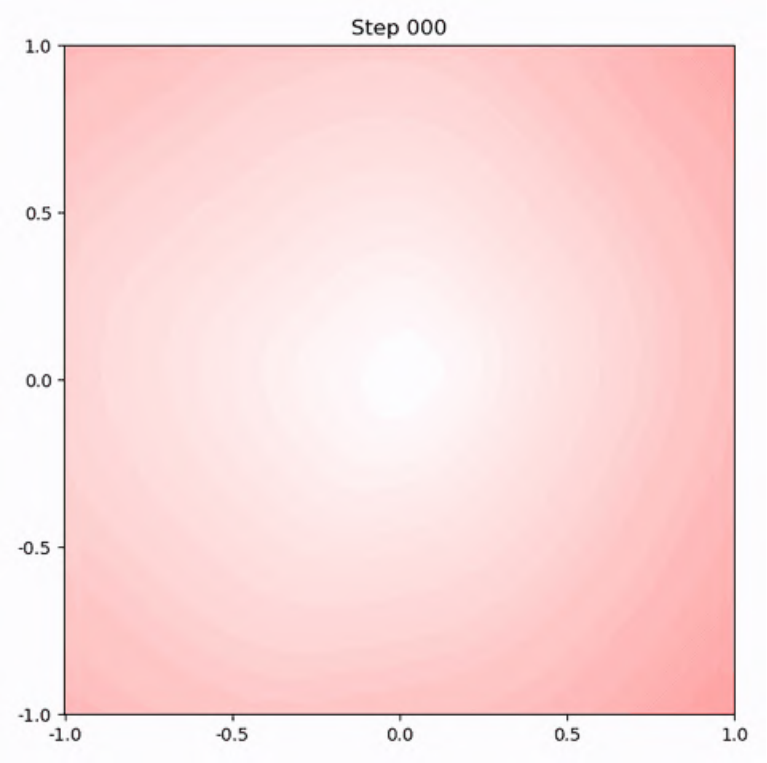


Figure 5: An animation of the discriminator learning the relative probabilities of a point being produced by the target function (blue) or merely uniform noise (red) through supervised classification.

From: https://towardsdatascience.com/this-will-change-the-way-you-look-at-gans-9992af250454

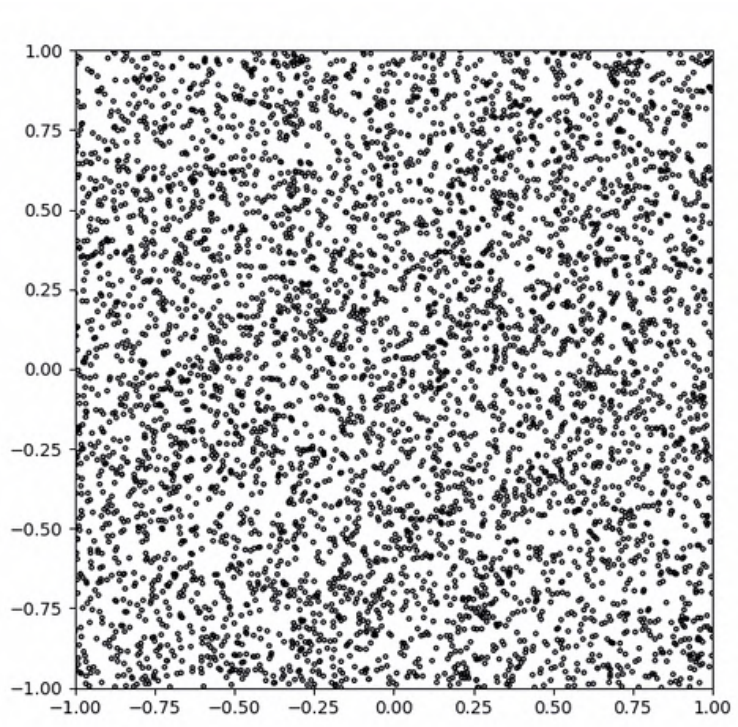
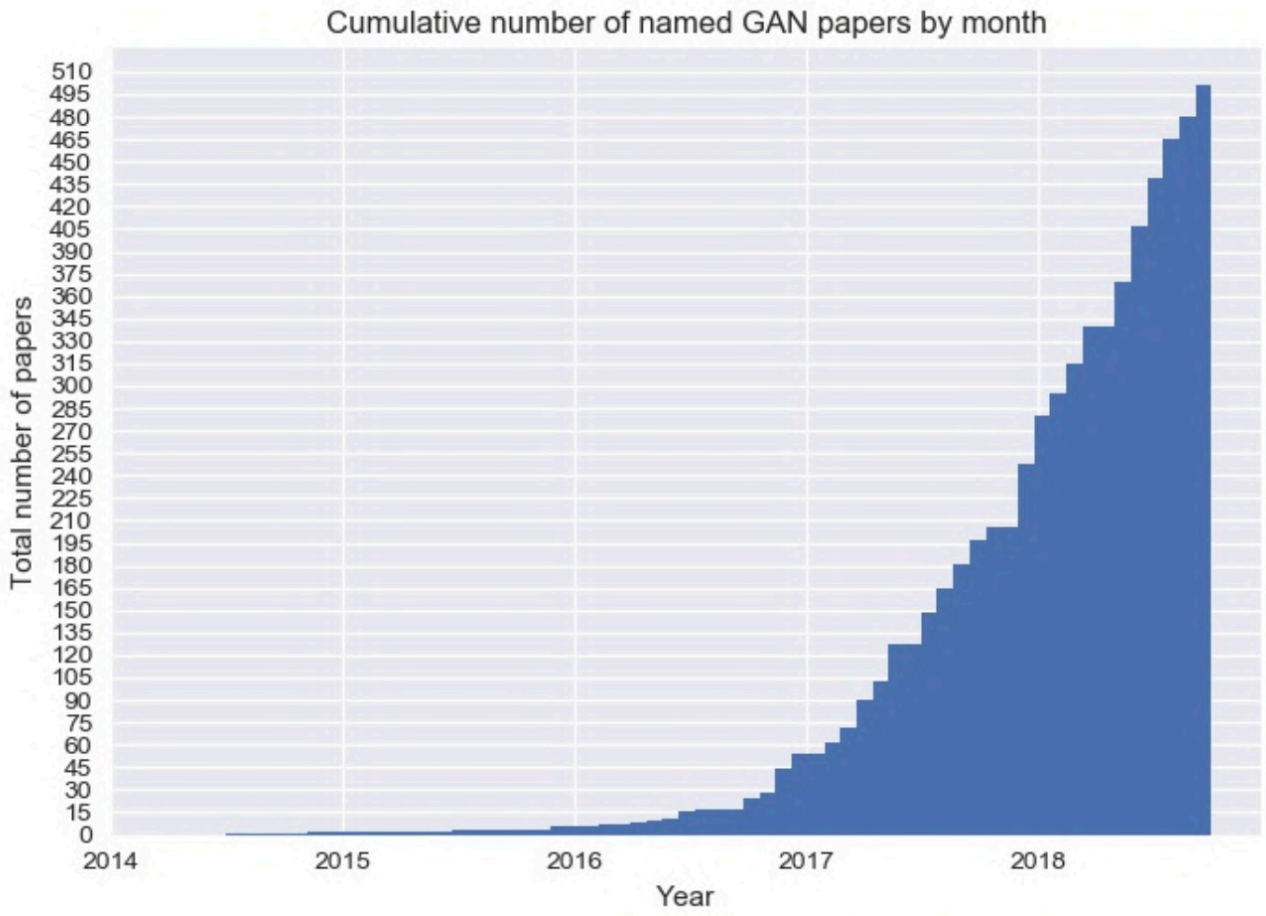


Figure 2: An animation of the target function applied to random input

The GAN BUZZ

- Since 2014, GAN is everywhere
- If GAN is good to generate images – why not for other problems?

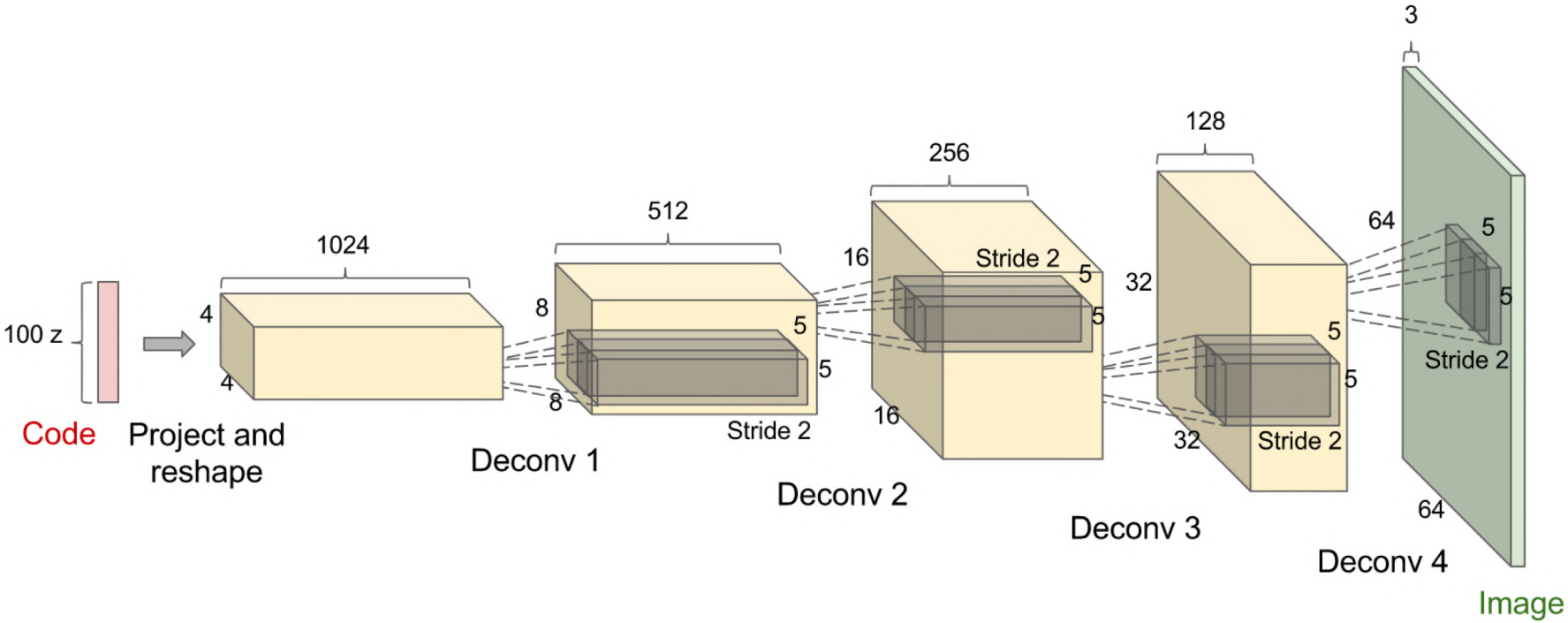
The GAN BUZZ



https://github.com/hindupuravinash/the-gan-zoo/blob/master/gans.tsv

Deep Network: DCGAN (deep convolution GAN)

- This network takes as input 100D random vector drawn from a uniform distribution (latent variable)
- It generates an 64x64 color image



A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, ICLR 2016

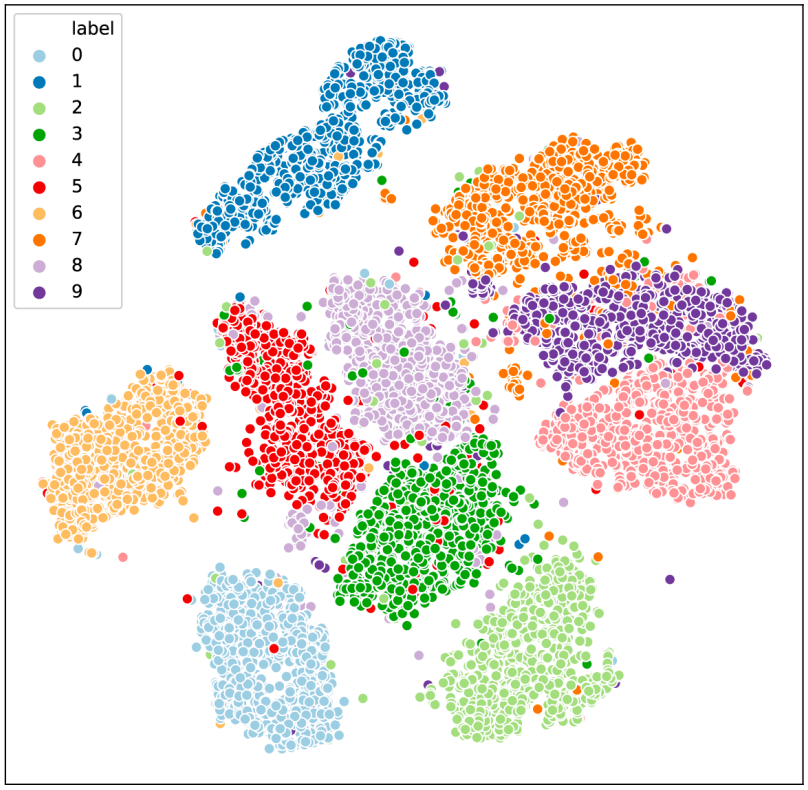
DCGAN Results

Trained on LSUN bedrooms dataset

A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, ICLR 2016

Editing the Latent Space

- The initial code z from which the generator generates an image $x = G(z)$ can be sought as an image representation in **Latent Space**.
- Does this space have any structure/meaning?



DCGAN Results

Smooth semantic transition in latent space

Interpolation between random points in the latent space

A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, ICLR 2016

Latent Space Arithmetic

Vector arithmetic in latent space

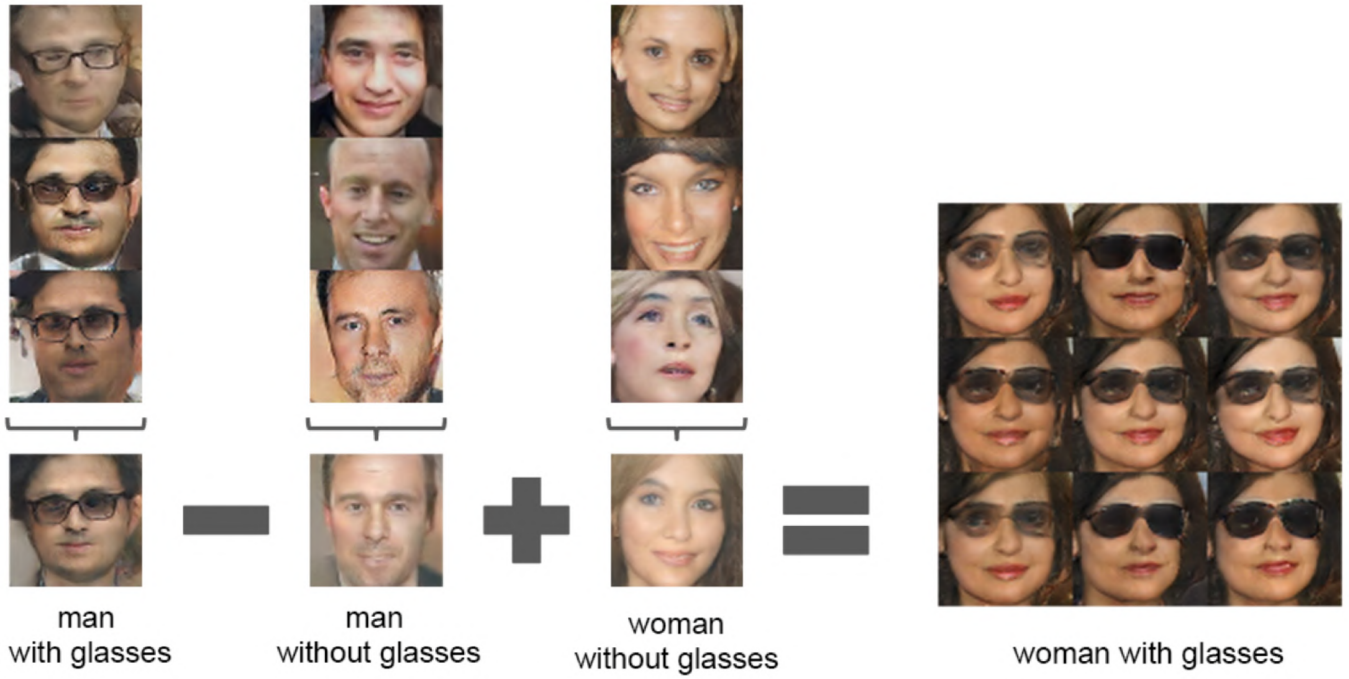
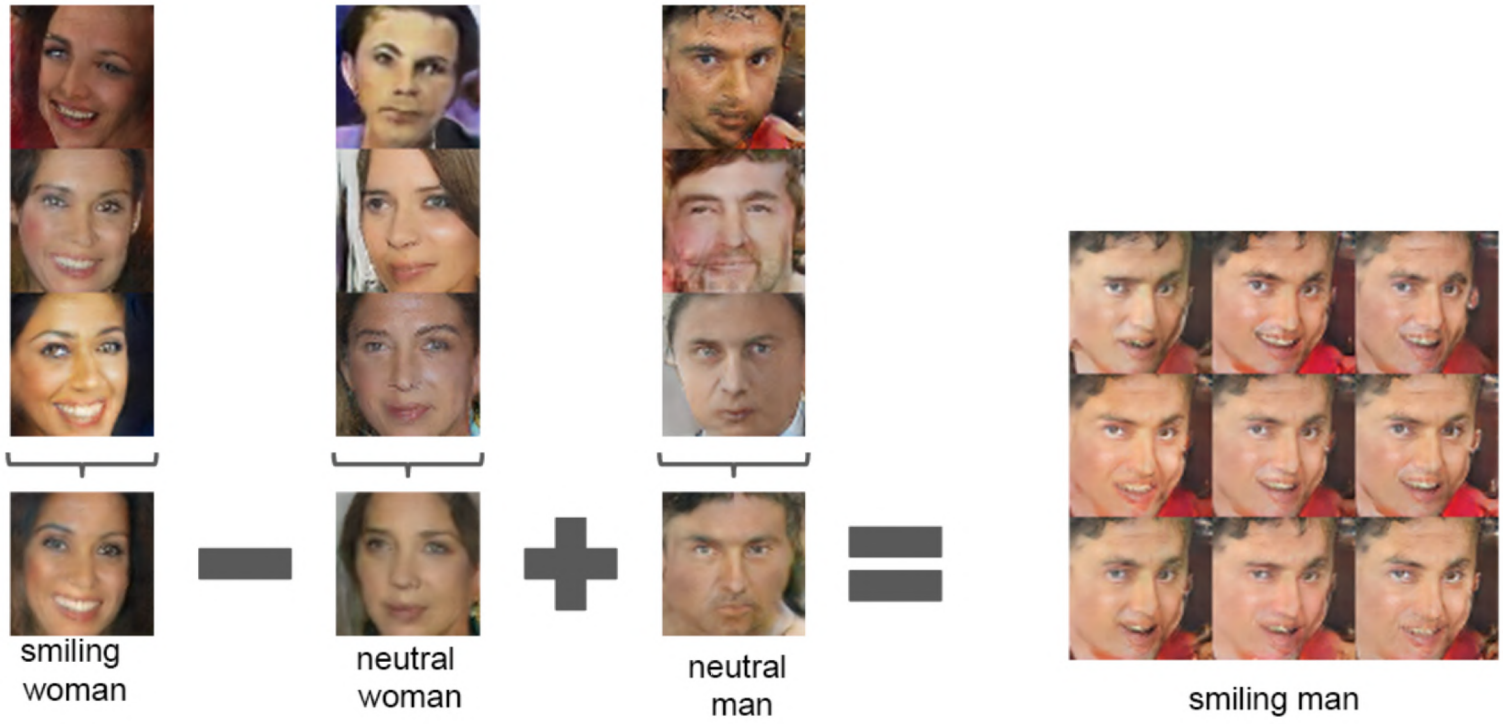


Figure 7: Vector arithmetic for visual concepts. For each column, the Z vectors of samples are averaged. Arithmetic was then performed on the mean vectors creating a new vector Y. The center sample on the right hand side is produce by feeding Y as input to the generator. To demonstrate the interpolation capabilities of the generator, uniform noise sampled with scale +-0.25 was added to Y to produce the 8 other samples.

A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, ICLR 2016

Latent Space Arithmetic

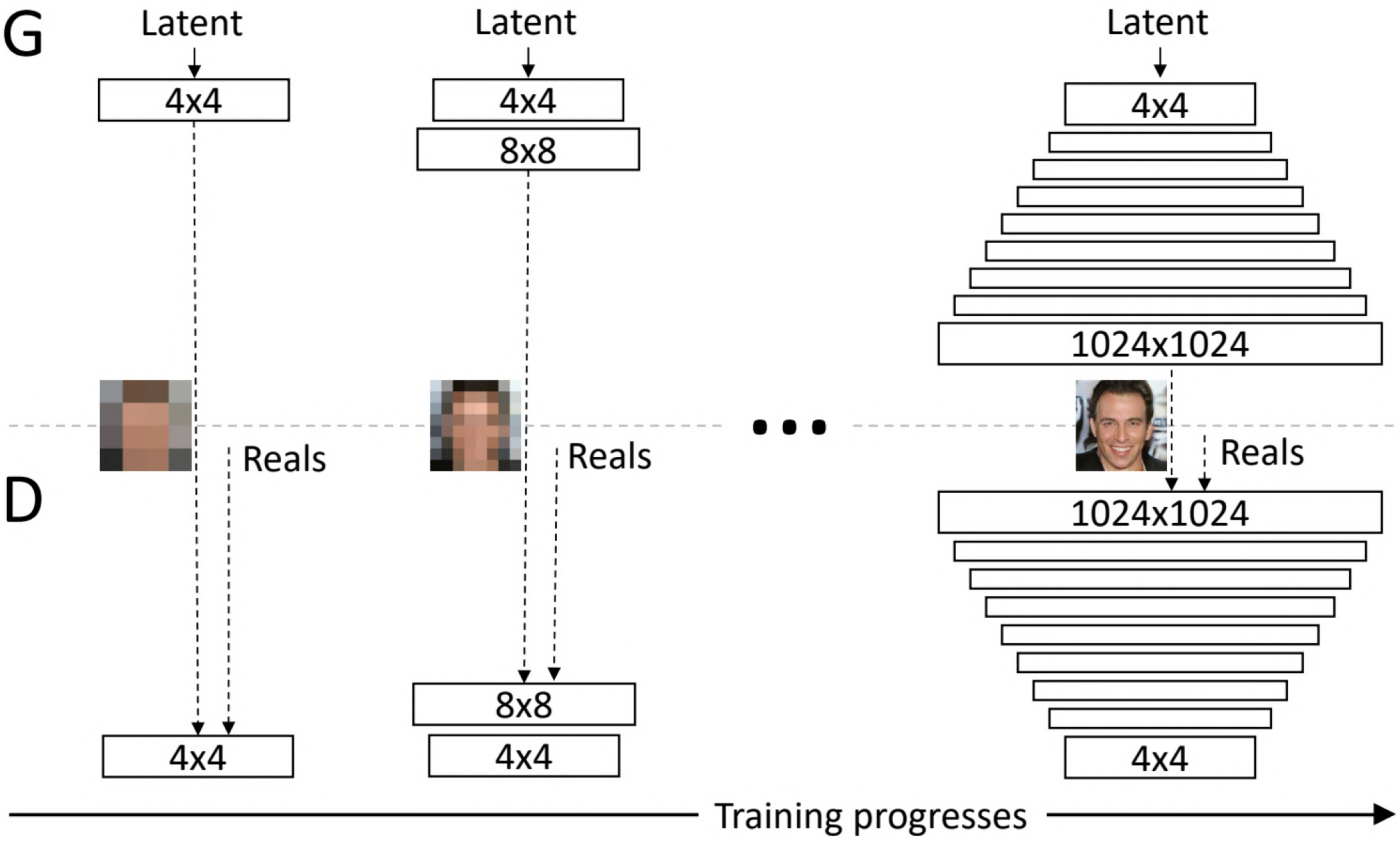
Vector arithmetic in latent space



A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, ICLR 2016

Progressive growing GAN

- The progressive growing of GANs trains network in multiple phases, adding additional convolution layer after the previous phase is stable.



Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

Progressive GAN Results

Generated from Celebrities Dataset

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

Progressive GAN Results

Generated from rom Bedrooms Dataset

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

Progressive GAN Results

Generated from different LSUN categories

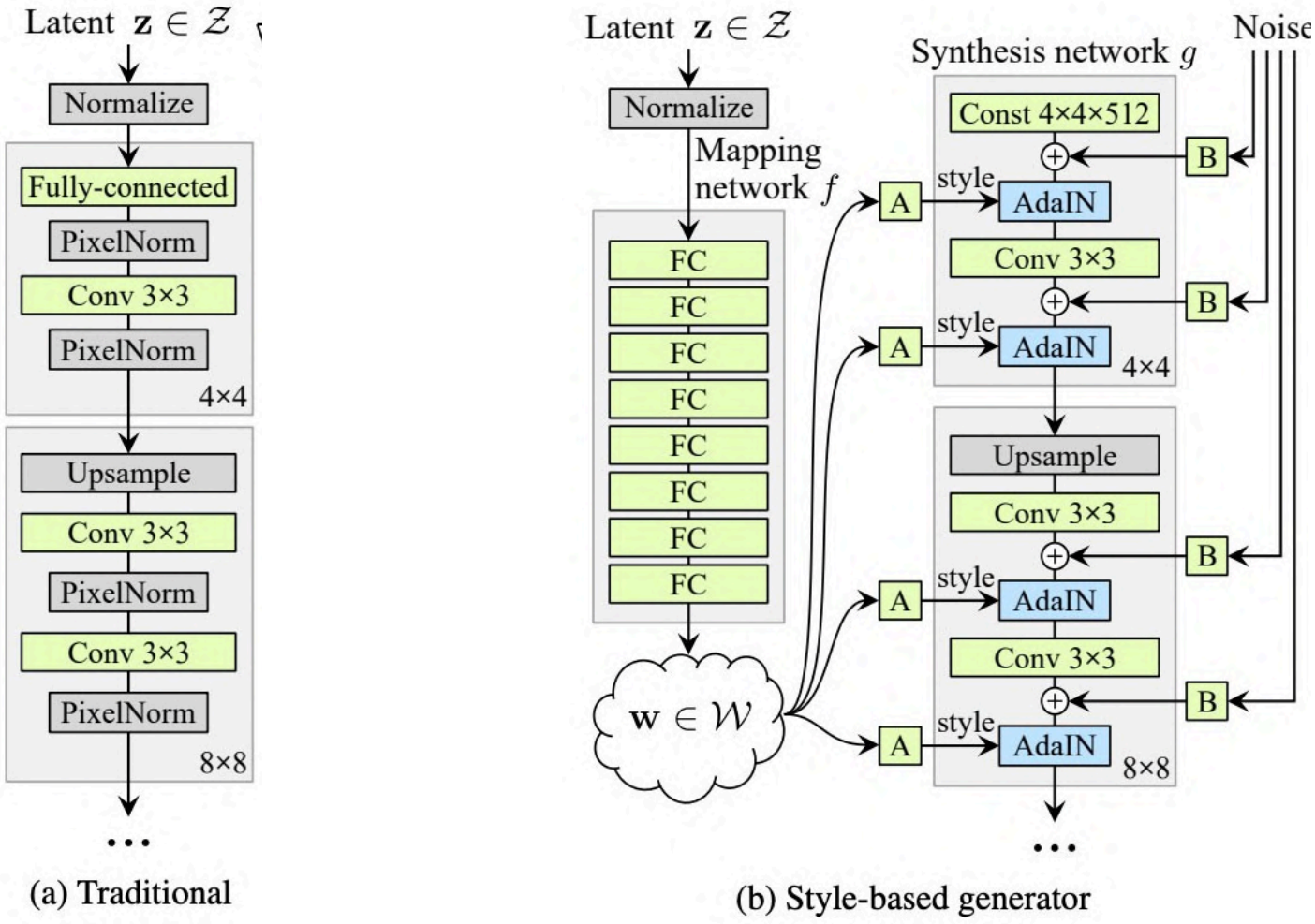
Style GAN

- StyleGAN was proposed by Karras et al. (from NVIDIA) in 2019, generates astonishing results.
- Can control the “styles” of the generated images.
- Built on top of the Progressive GAN, thus can generate high res images (1024x1024).

A Style-Based Generator Architecture for Generative Adversarial Networks

Style GAN

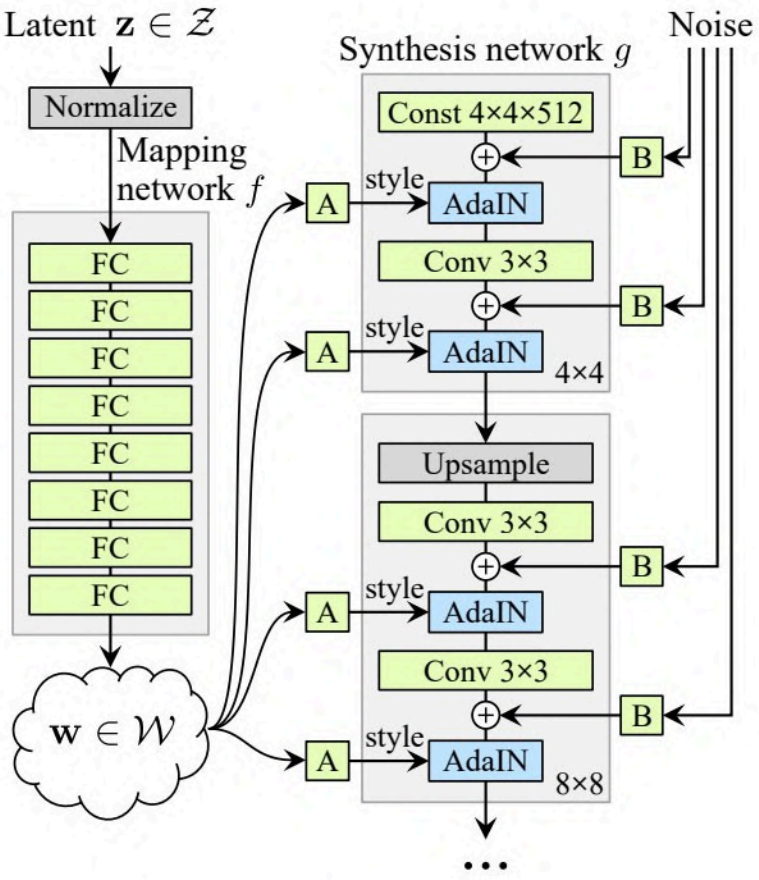
Style GAN vs. Traditional GAN



Style GAN

New ideas:

1. A mapping network converts the latent *z* into an intermediate latent space *w*
2. For each feature layer we learn an affine operation *A* that transforms *w* into the style information.
3. AdaIN (adaptive instance normalization) controls the styling in the feature space.
4. Injecting noise into the feature space for creating stochastic variations.

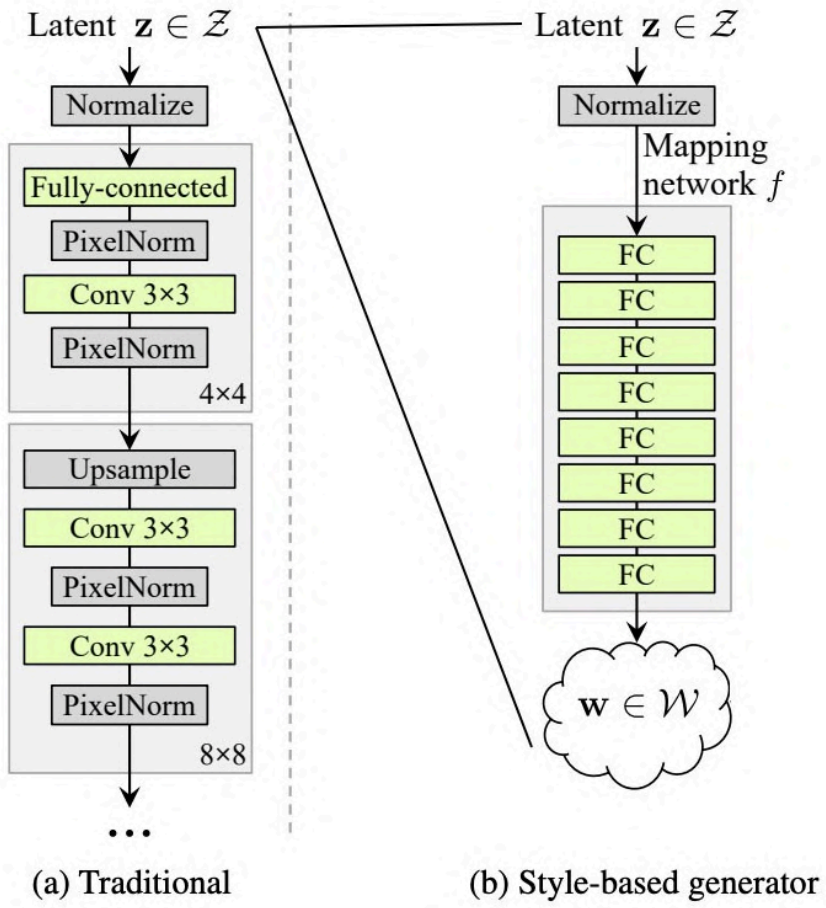


(b) Style-based generator

From: Jonathan Hui

Style GAN: Latent Mapping

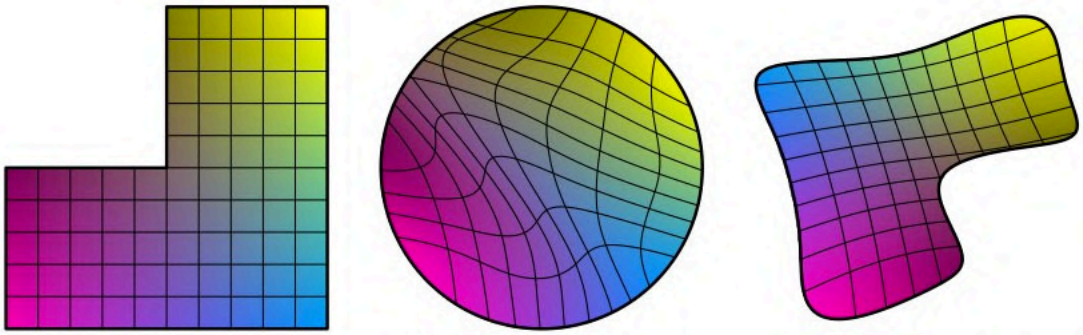
- In vanilla GAN, the distribution of *z* should resemble the latent factor distribution of the real images.
- Normal or uniform distributions, do not represent the real distribution. Thus, the latent space must include entangled factors.
- **Latent space transformation** converts *z* into the *w* space while rectifying the original latent space into linear latent factor space.



From: Jonathan Hui

Style GAN: Latent Mapping

- For example, let's generate portraits for military personnel and visualize the data distribution of the training dataset with two latent factors: *masculinity* and *hair length*.
- The missing upper left corner (left) indicates that male soldiers are not allowed to have long hair.
- StyleGAN warps a space that can be sampled with a uniform or normal distribution (middle) into the latent feature space (left).



(a) Distribution of features in training set

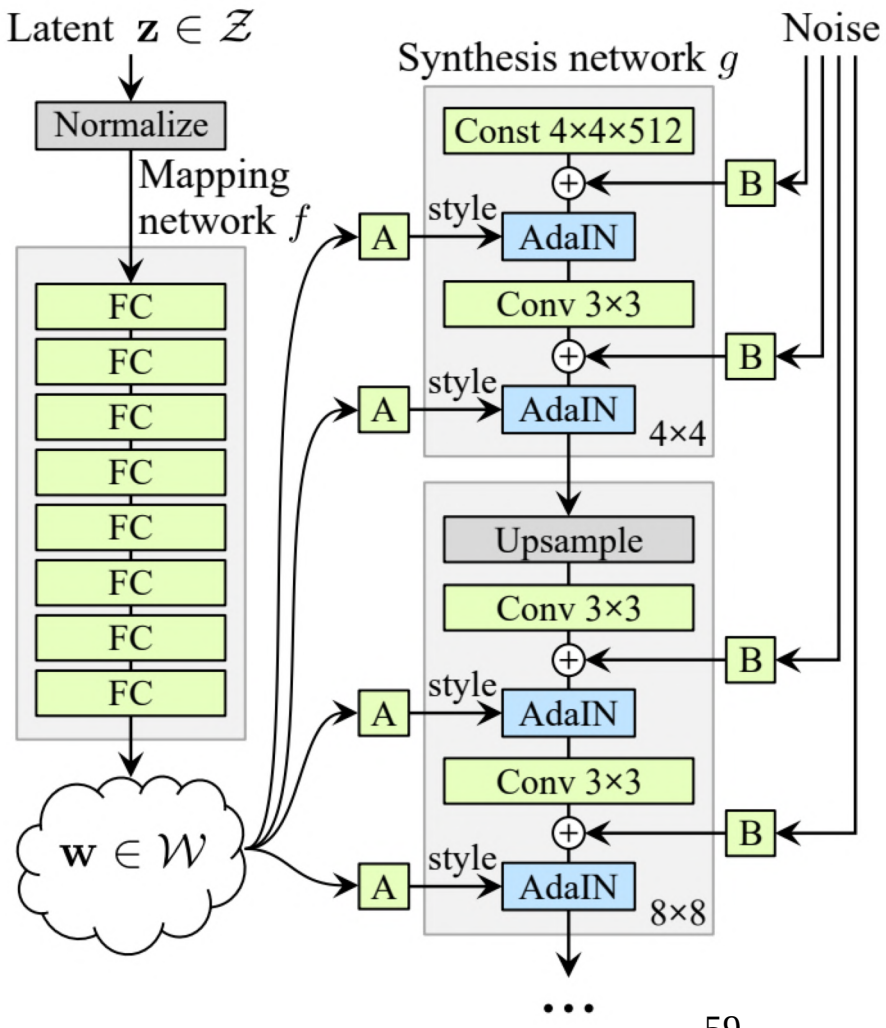
(b) Mapping from $\mathcal{Z}$ to features

(c) Mapping from $\mathcal{W}$ to features

From: Jonathan Hui Example from the original paper

Style GAN: Style Generation

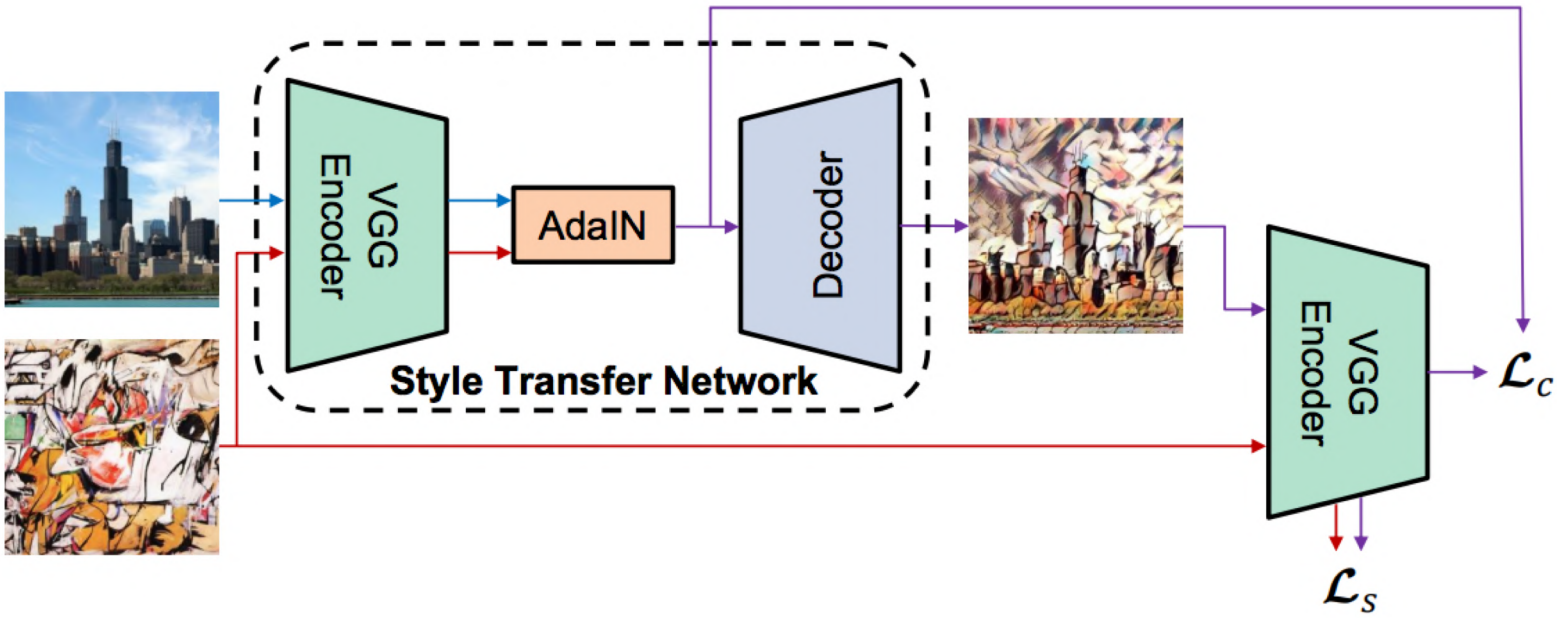
- In the vanilla GAN, the latent factor z is used as the input to the first deep network layer only.
- The role of *z* is diminished as we go deeper into the network.
- In the style-GAN, we apply a learned affine operation *A* to transform *w* for each layer.
- This transformed *w* acts as the style information at each layer.



From: Jonathan Hui

Style GAN: AdaIN

- The style information is controlled by the AdaIN (Adaptive Instance Normalization) framework.
- This method was suggested for image style-transfer by Huang and Belongie 2017.



Huang and Belongie: Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization

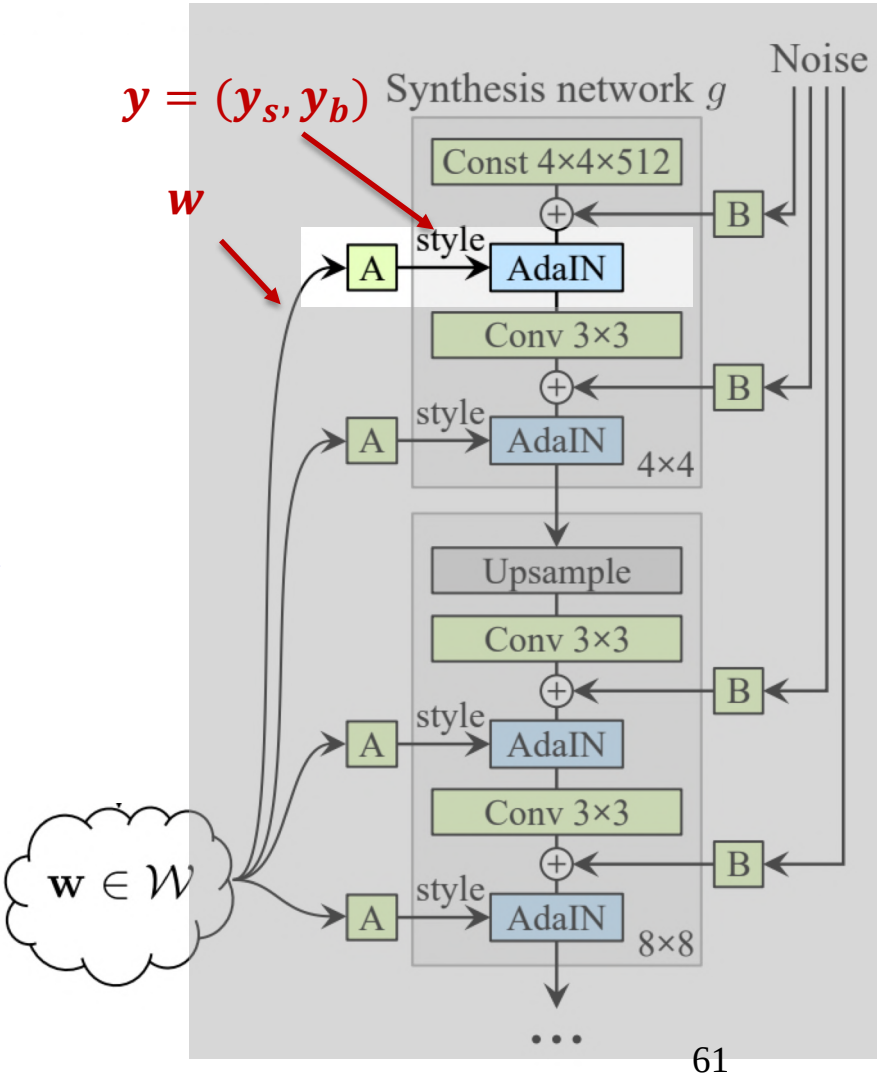
Style GAN: AdaIN

- The AdaIN scales and biases each spatial feature by the style values:

feature map i

$$\text{AdaIN}(\boldsymbol{x}_i, y) = y_{s,i} \frac{\boldsymbol{x}_i - \mu(\boldsymbol{x}_i)}{\sigma(\boldsymbol{x}_i)} + y_{b,i}$$

Normalize the feature map i (instance normalization)

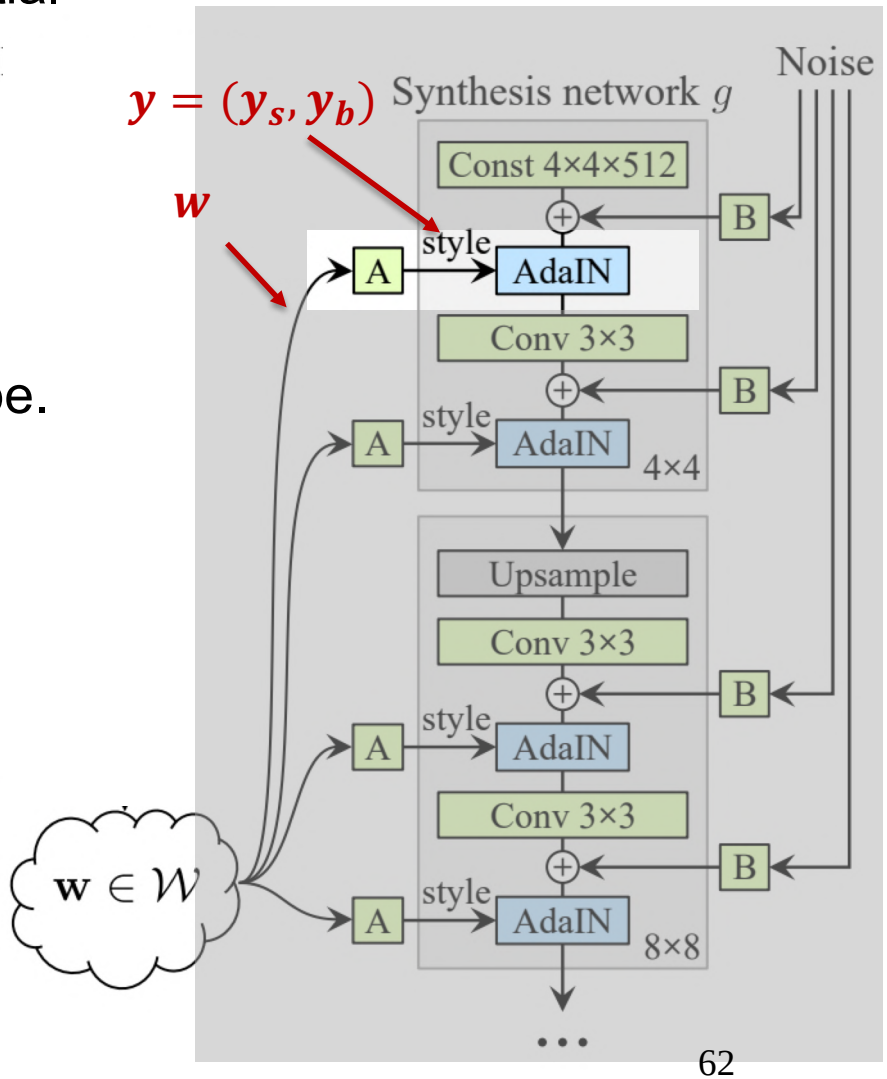


From: Jonathan Hui

Style GAN: AdaIN

- The AdaIN in applied at multiple spatial resolution.
- Each resolution controls a different type of style:

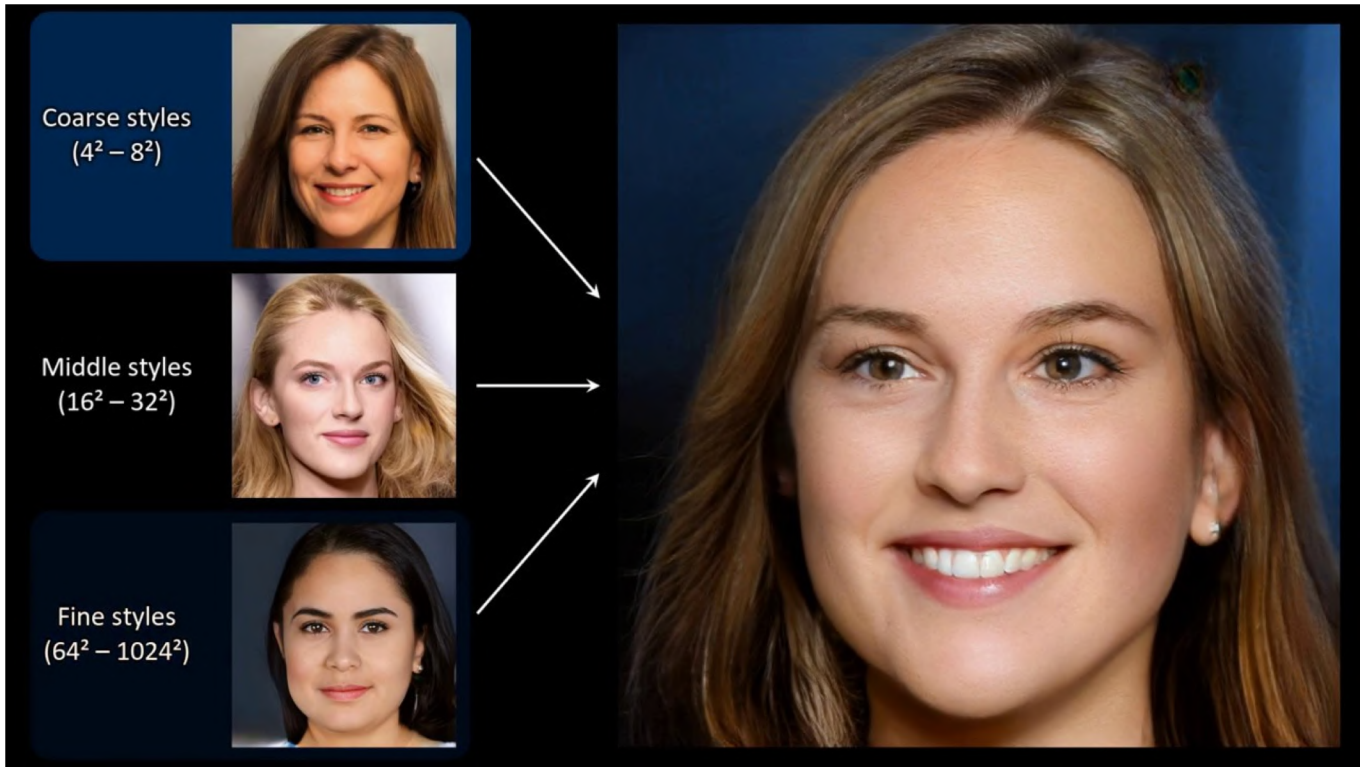
- **Coarse styles**: pose, hair, face shape.
- **Middle styles**: facial features, eyes.
- **Fine styles**: color scheme.



From: Jonathan Hui

Style GAN: AdaIN

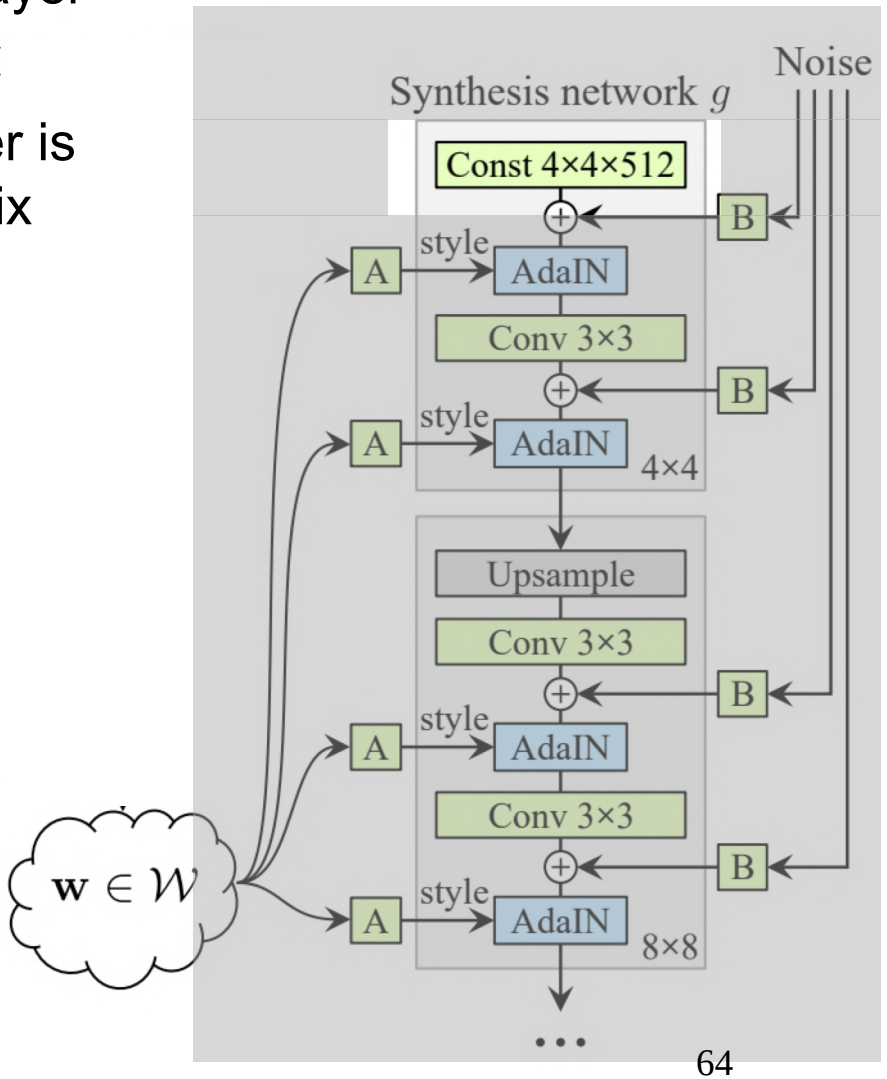
- **Coarse styles**: pose, hair, face shape.
- **Middle styles**: facial features, eyes.
- **Fine styles**: color scheme.



From: Jonathan Hui

Style GAN: AdaIN

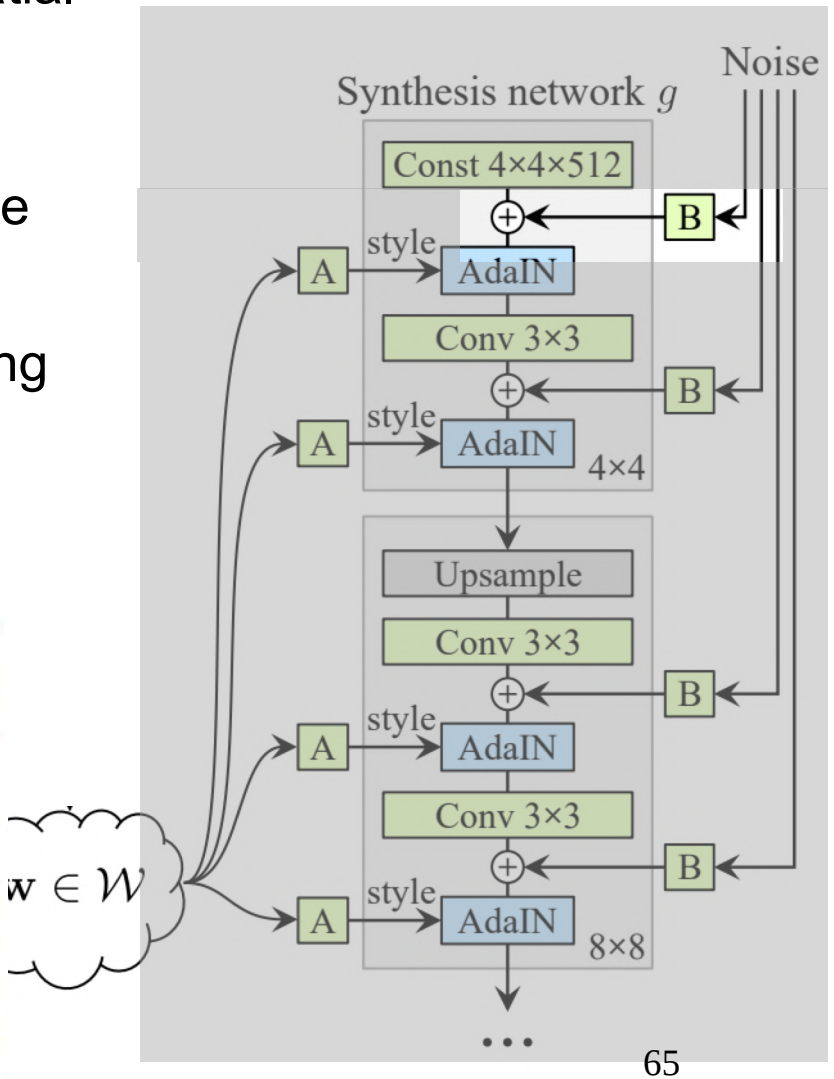
- In vanilla GAN, the input to the first layer is the latent factor z.
- In StyleGAN the input to the first layer is replaced by a **learned** constant matrix with dimension 4×4×512.

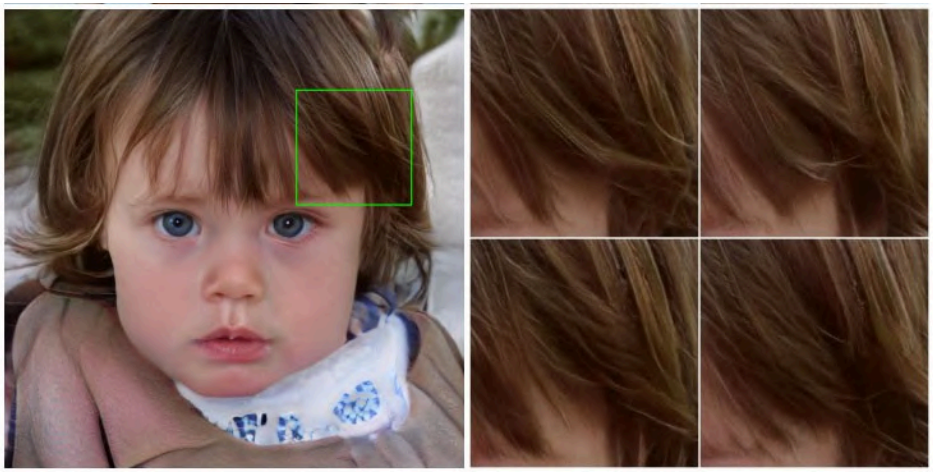


From: Jonathan Hui

Style GAN: Stochastic Noise

- SytleGAN introduces noise into the spatial data for creating stochastic style variations.
- For each spatial layer, a Gaussian noise matrix is added to the feature maps.
- The mapping *B* learns a separate scaling factor for each feature map.





(a) Generated image (b) Stochastic variation

Style GAN: Stochastic Noise

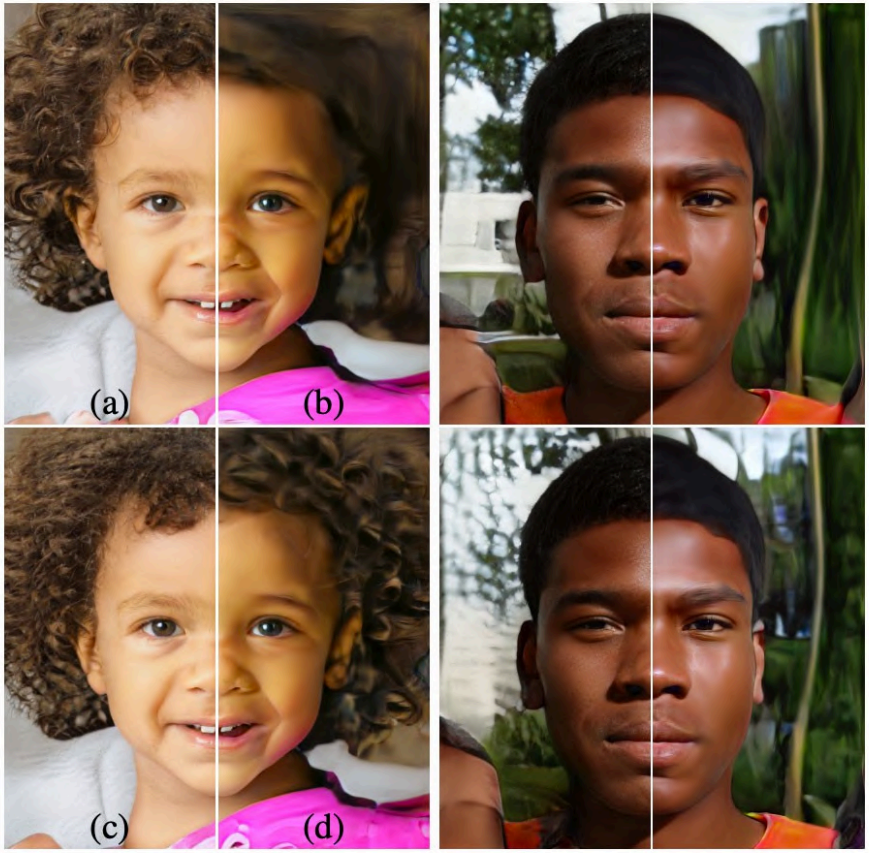
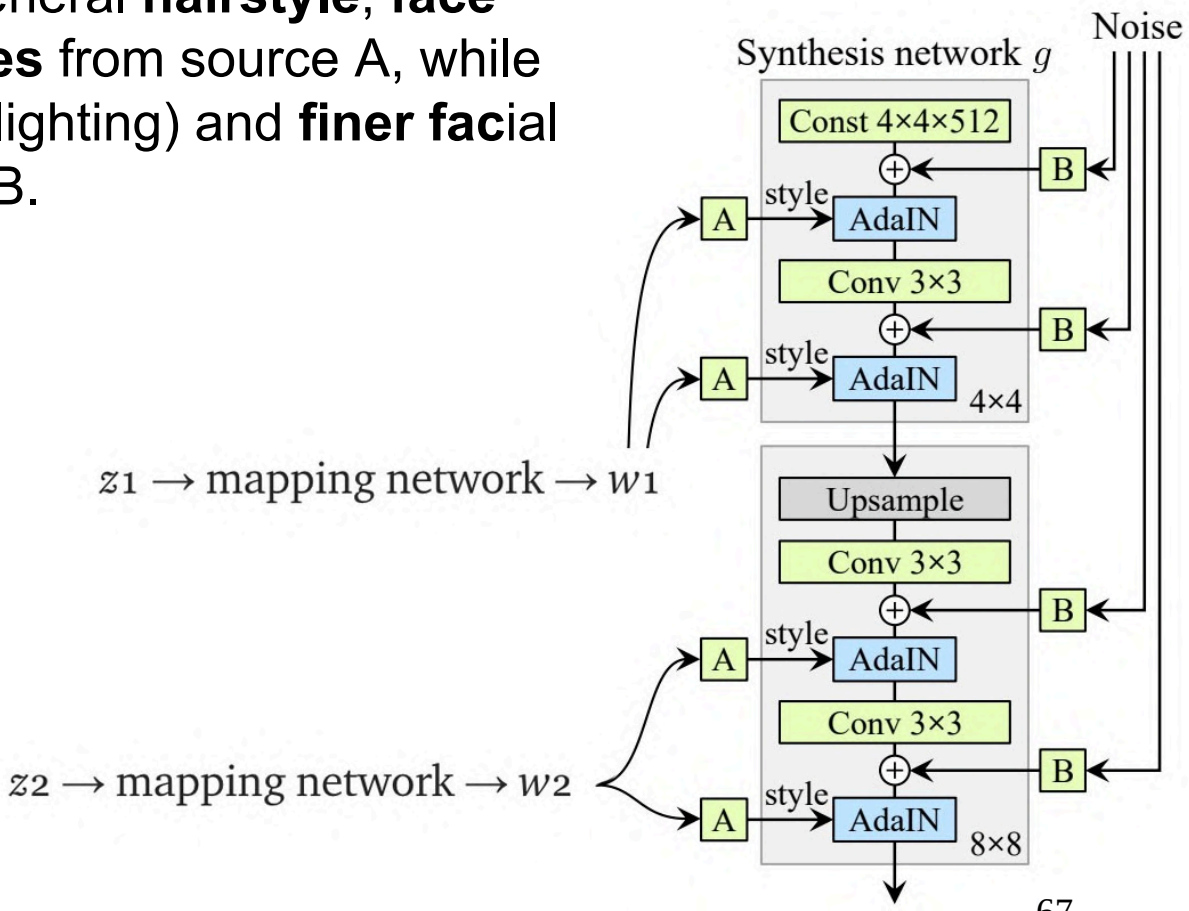


Figure 5. Effect of noise inputs at different layers of our generator. (a) Noise is applied to all layers. (b) No noise. (c) Noise in fine layers only (64^2 – 1024^2). (d) Noise in coarse layers only (4^2 – 32^2). We can see that the artificial omission of noise leads to featureless "painterly" look. Coarse noise causes large-scale curling of hair and appearance of larger background features, while the fine noise brings out the finer curls of hair, finer background detail, and skin pores.

Style GAN: Style Mixing

- We can mix styles from two sources.
- The generated image will have high-level style such as **pose**, general **hairstyle**, **face shape**, and **eyeglasses** from source A, while all **colors** (eyes, hair, lighting) and **finer facial features** from source B.



Style GAN: Style Mixing

Coarse styles: pose, hair, face shape.
Middle styles: facial features, eyes.
Fine styles: color scheme.

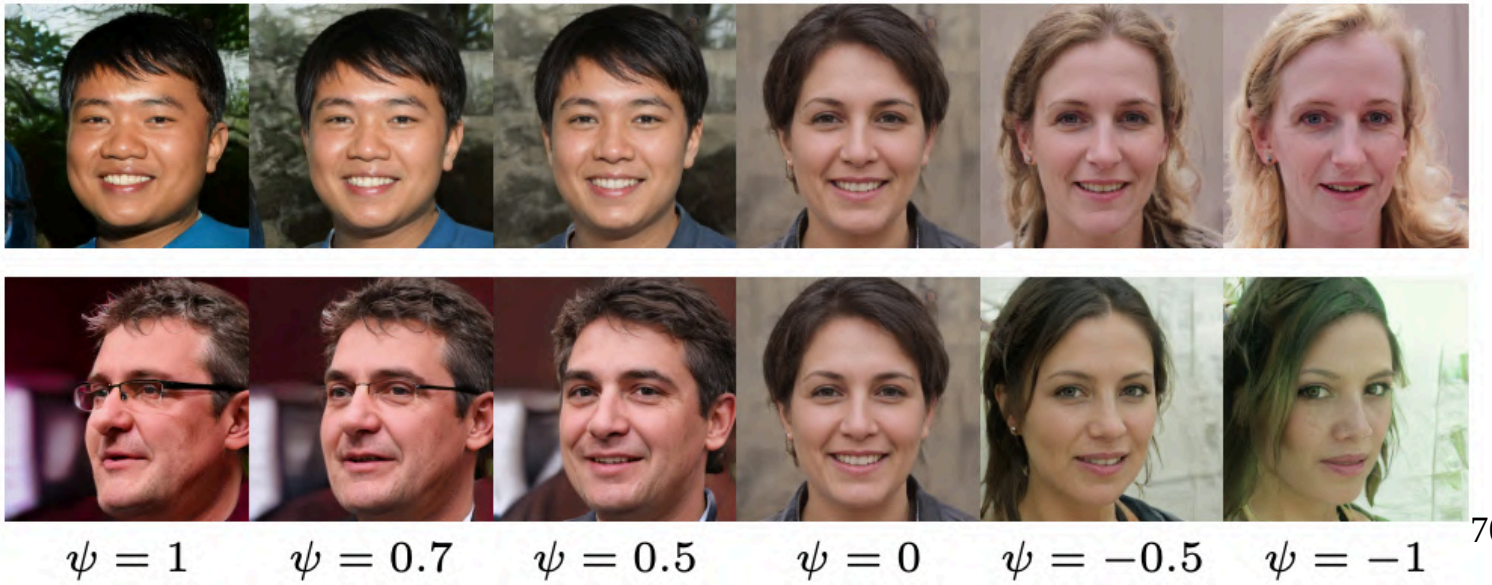
Style GAN: Style Mixing

Coarse styles: pose, hair, face shape.
Middle styles: facial features, eyes.
Fine styles: color scheme.

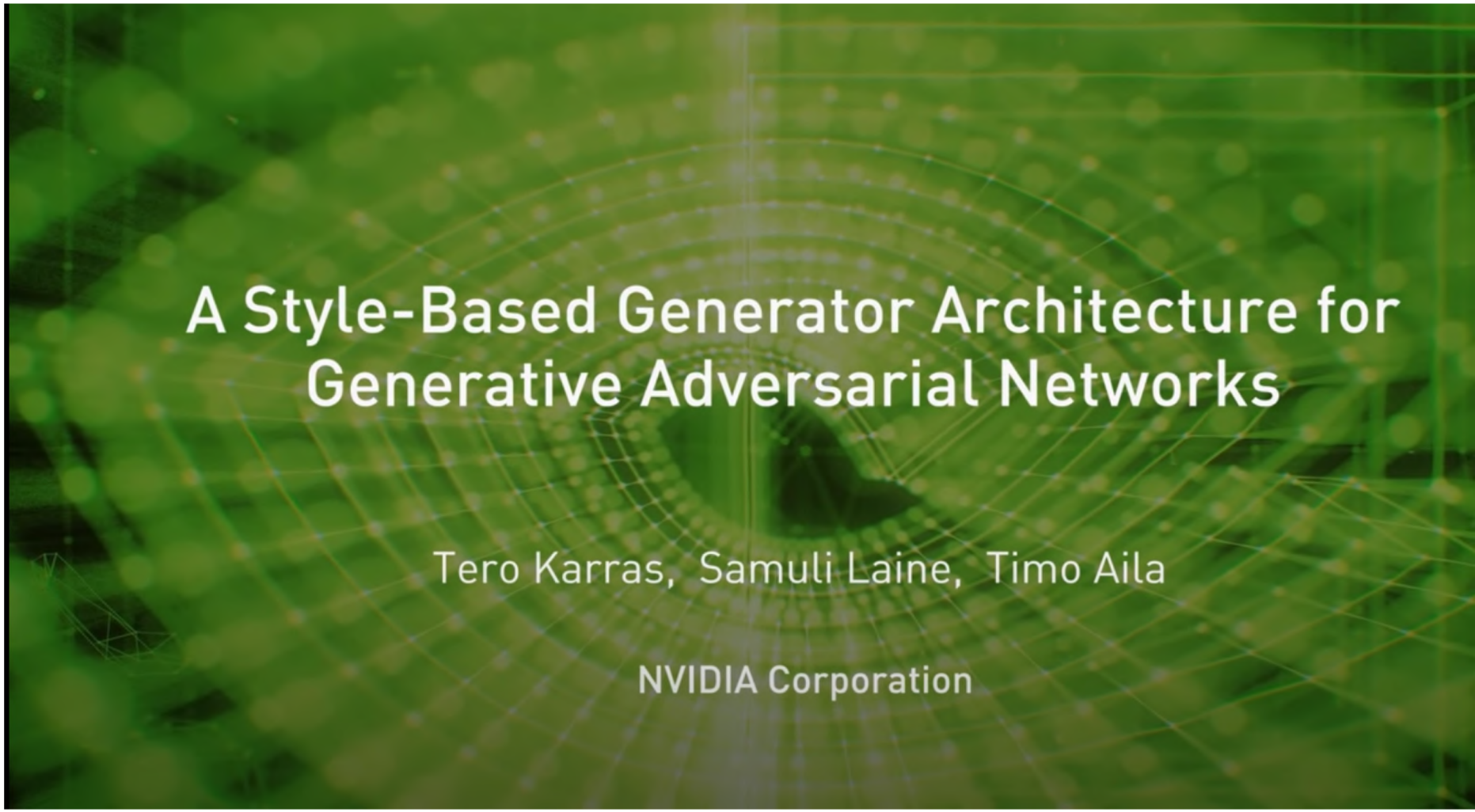
Style GAN: Style Scaling

- For all sampled images we can calculate the mean style $\bar{\boldsymbol{w}}$ which can be interpreted as the “mean face”.
- **Style-scaling** scales the style deviation about the mean.
- This can be implemented to each scale separately.

$$\boldsymbol{w}' = \bar{\boldsymbol{w}} + \psi(\boldsymbol{w} - \bar{\boldsymbol{w}}) \quad \text{where} \quad \bar{\boldsymbol{w}} = \mathbb{E}_{\boldsymbol{z} \sim P(\boldsymbol{z})}[f(\boldsymbol{z})]$$

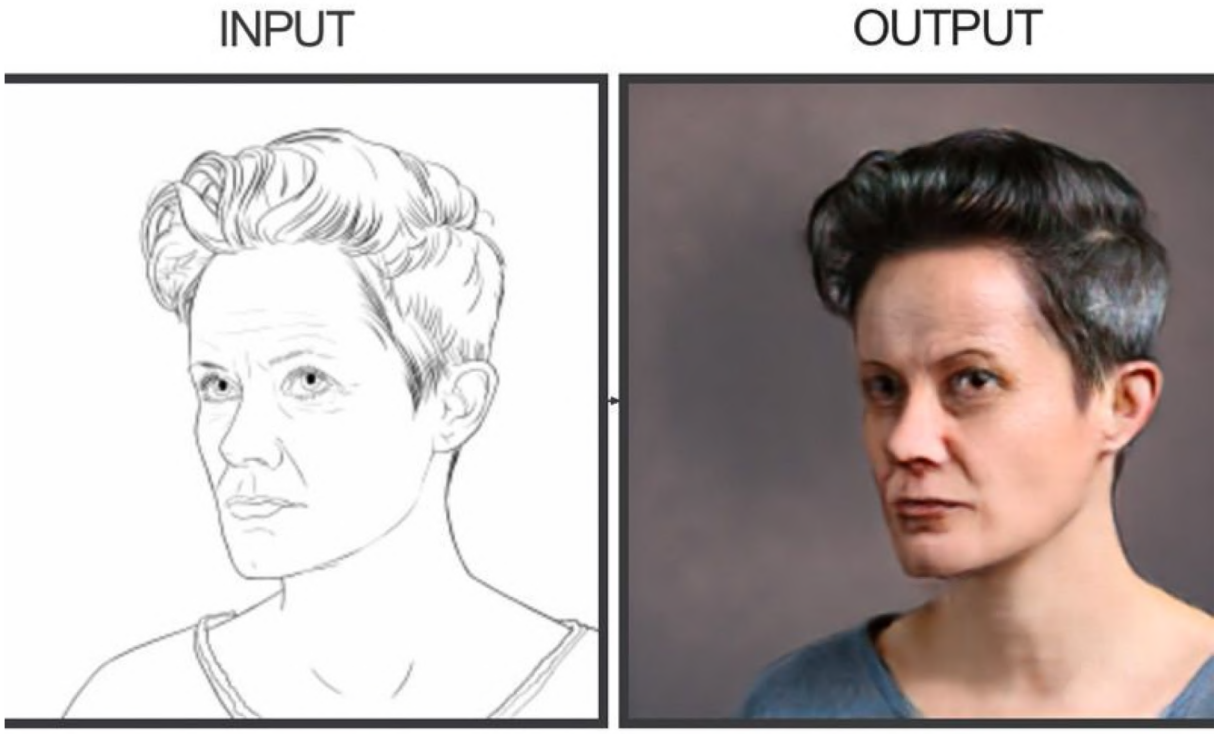


Style GAN: Video



https://www.youtube.com/watch?time_continue=6&v=kSLJriaOumA&feature=emb_logo

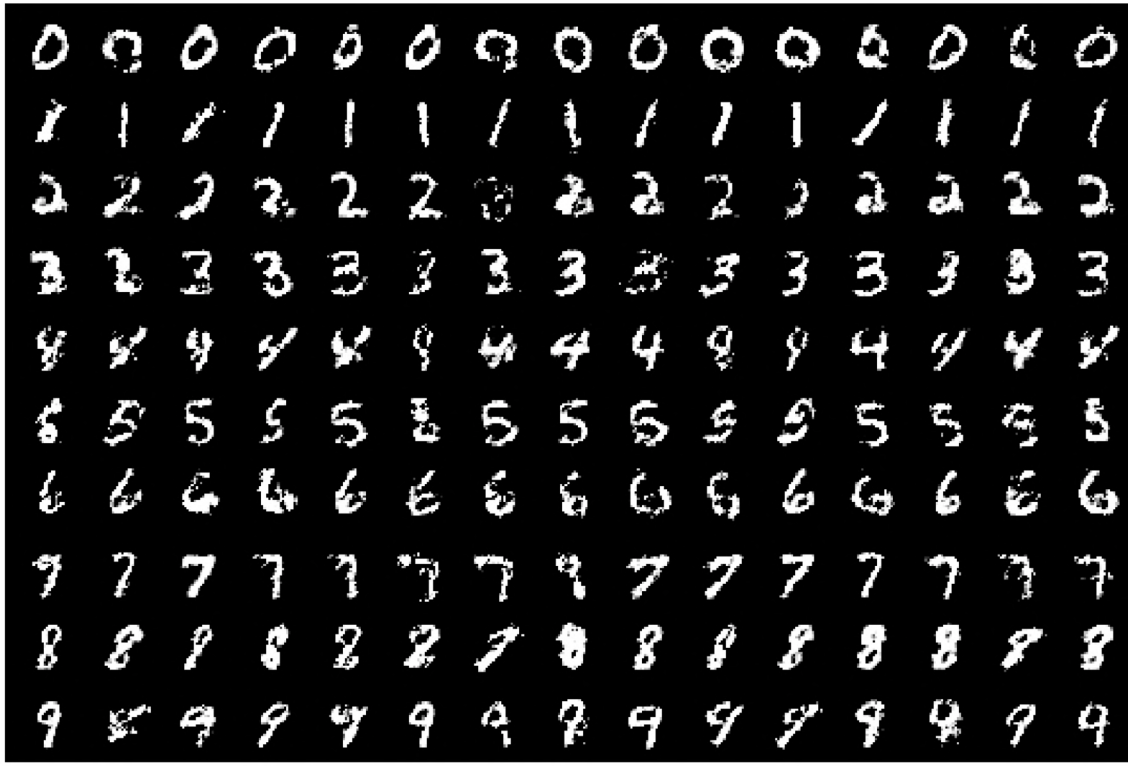
Advanced Generative Models



Conditional GAN

- MNIST digits conditioned on their class label
- Proposed by Mirza & Osindero in 2014.

$[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ →
$[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$ →
$[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]$ →
$[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$ →
$[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$ →
$[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$ →
$[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$ →
$[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]$ →
$[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]$ →
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$ →

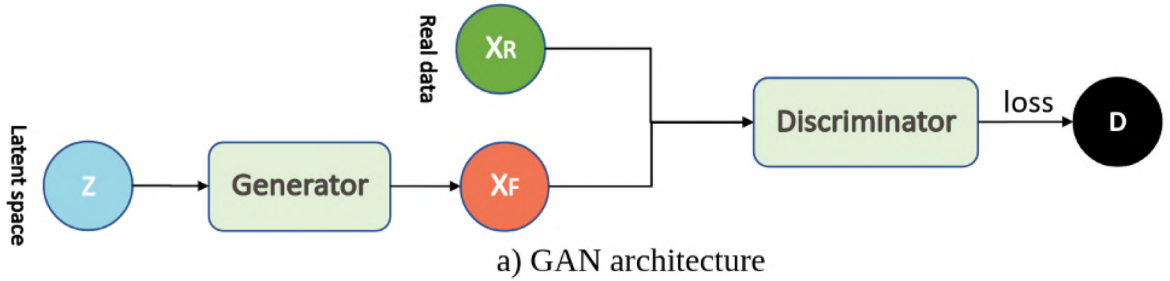


MNIST digits

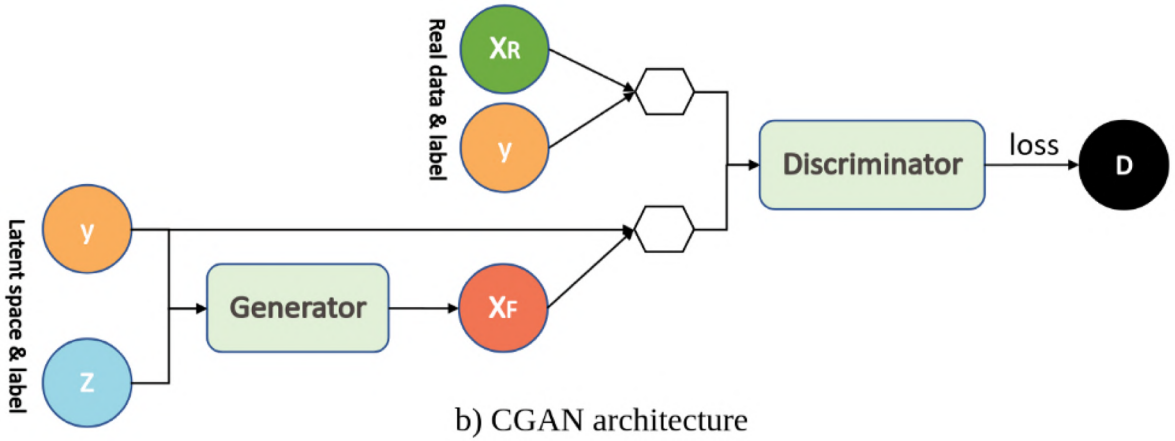
[Paper: Conditional Generative Adversarial Nets]

Conditional GAN

- Simple modification to the original GAN framework that conditions the model on *additional information*
- Leads to many practical applications of GANs when we have explicit *supervision* available.



a) GAN architecture



b) CGAN architecture

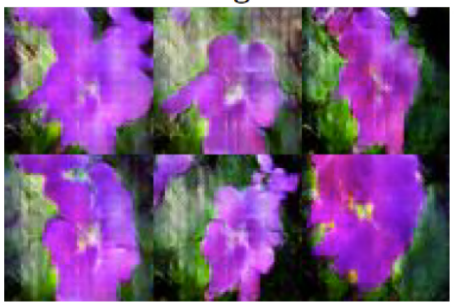
Conditional GAN

- Given a text description, generate images closely associated.
- Uses a conditional GAN with the generator and discriminator being condition on “dense” text embedding.

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma

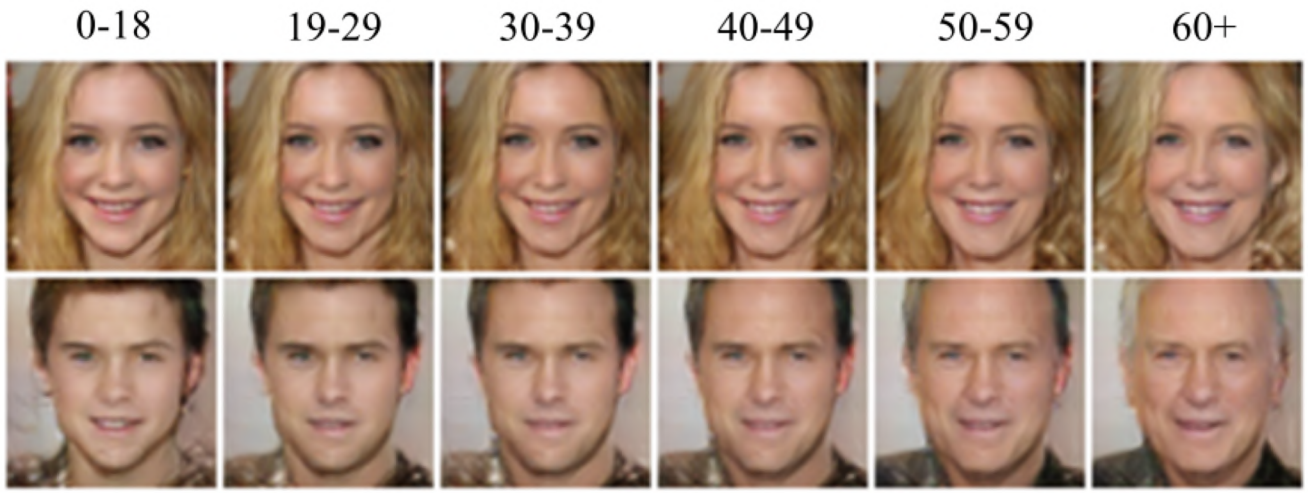


this white and yellow flower have thin white petals and a round yellow stamen

[Text to Image Synthesis Using Generative Adversarial Networks]

Conditional GAN

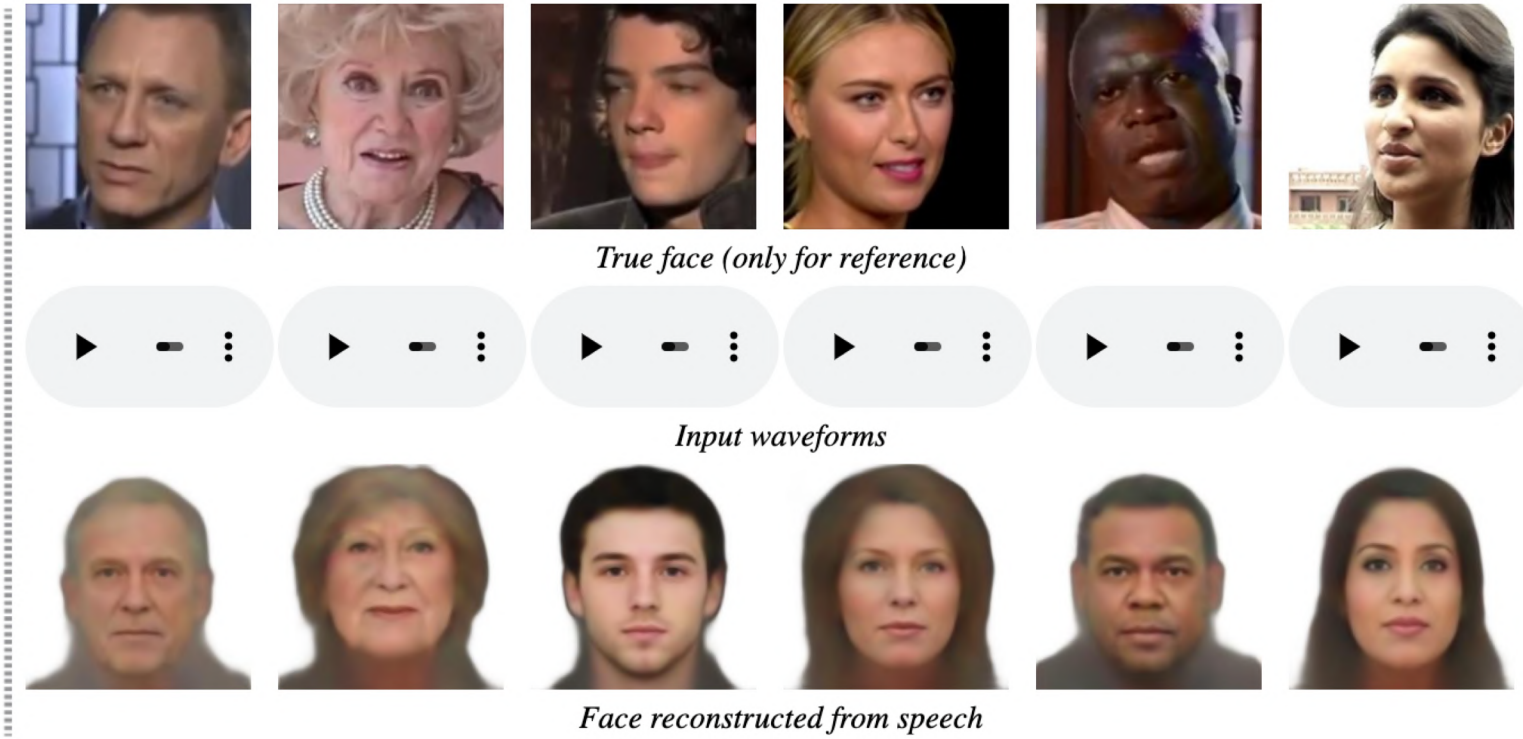
- Latent code is conditioned on age categories.



Face Aging With Conditional Generative Adversarial Networks

Conditional GAN

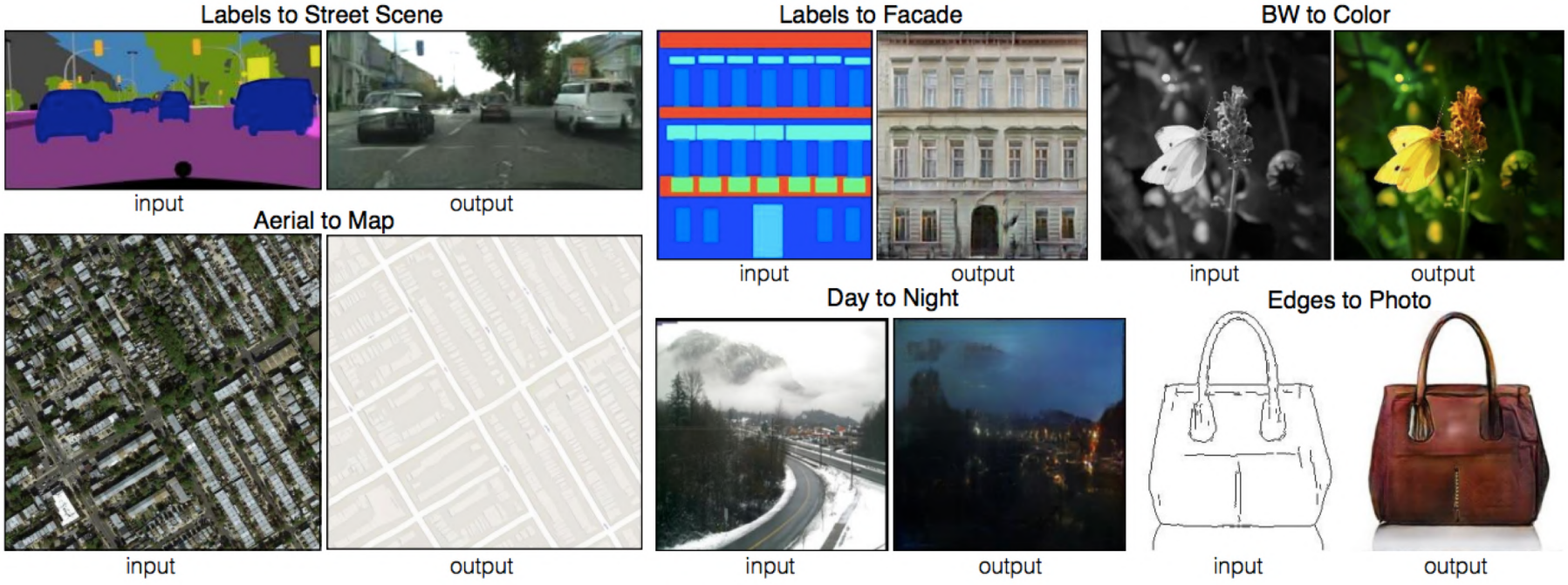
- Latent code is conditioned on speech



Speech2Face: Learning the Face Behind a Voice

Image-to-Image Translation

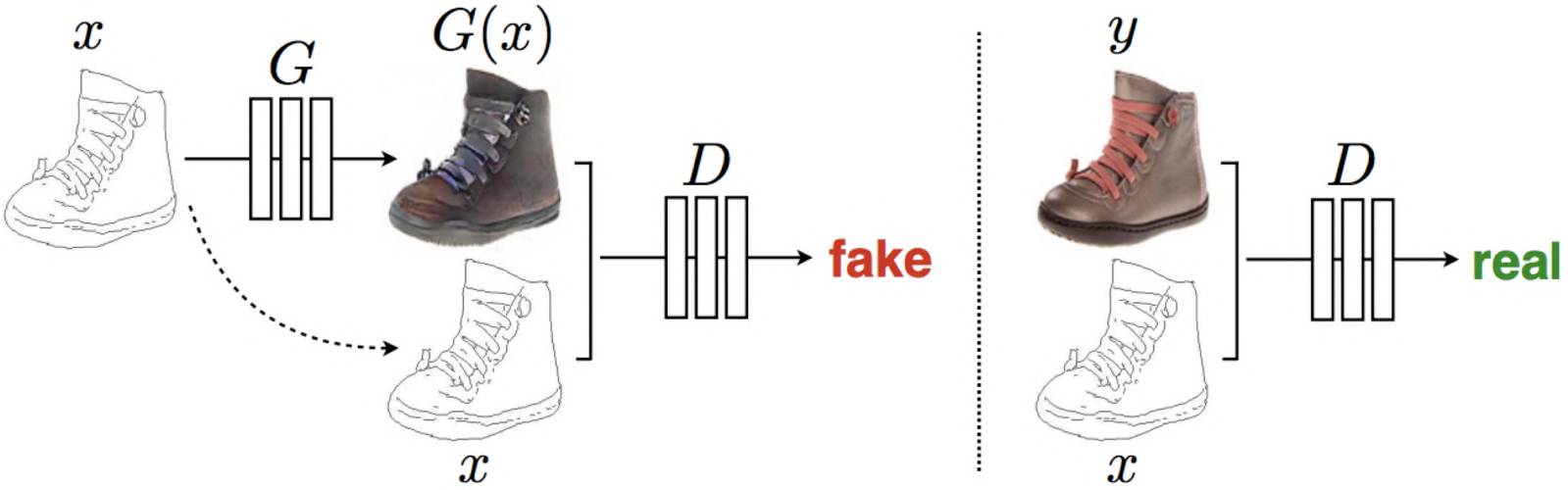
- The Generator/Discriminator scheme is used to translate an image from domain ***A*** into domain ***B***



[Image-to-Image Translation, Isola, Zhu, Zhou, & Efros (Nov 2017)]

Image-to-Image Translation

- Produce modified image y conditioned on input image x
- Generator receives $x \in \mathcal{A}$ as input and generates $y \in \mathcal{B}$
- Discriminator receives an x, y pair and has to decide whether it is real or fake



[Image-to-Image Translation, Isola, Zhu, Zhou, & Efros (Nov 2017)]

Image-to-Image Translation

- Generator architecture: U-Net
- Why should we pool layers? Why earlier layers are needed?
- Note: no z is used as an input, transformation is basically deterministic

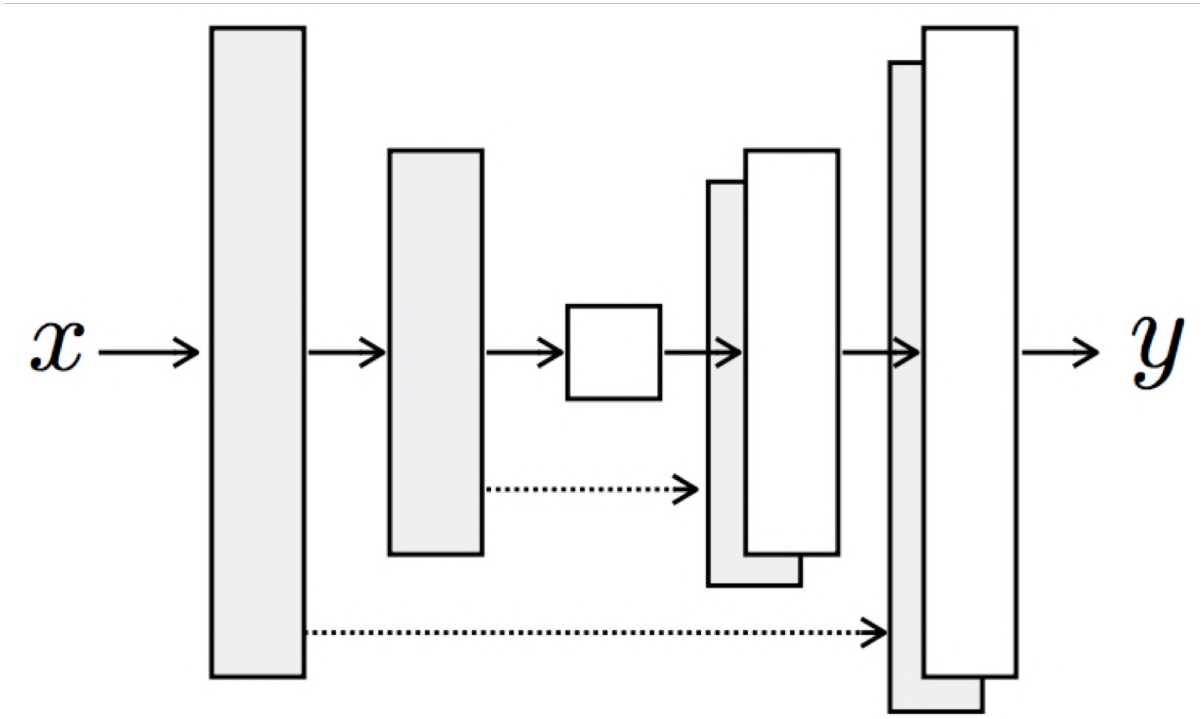


Image-to-Image Translation

- Semantic labels to images

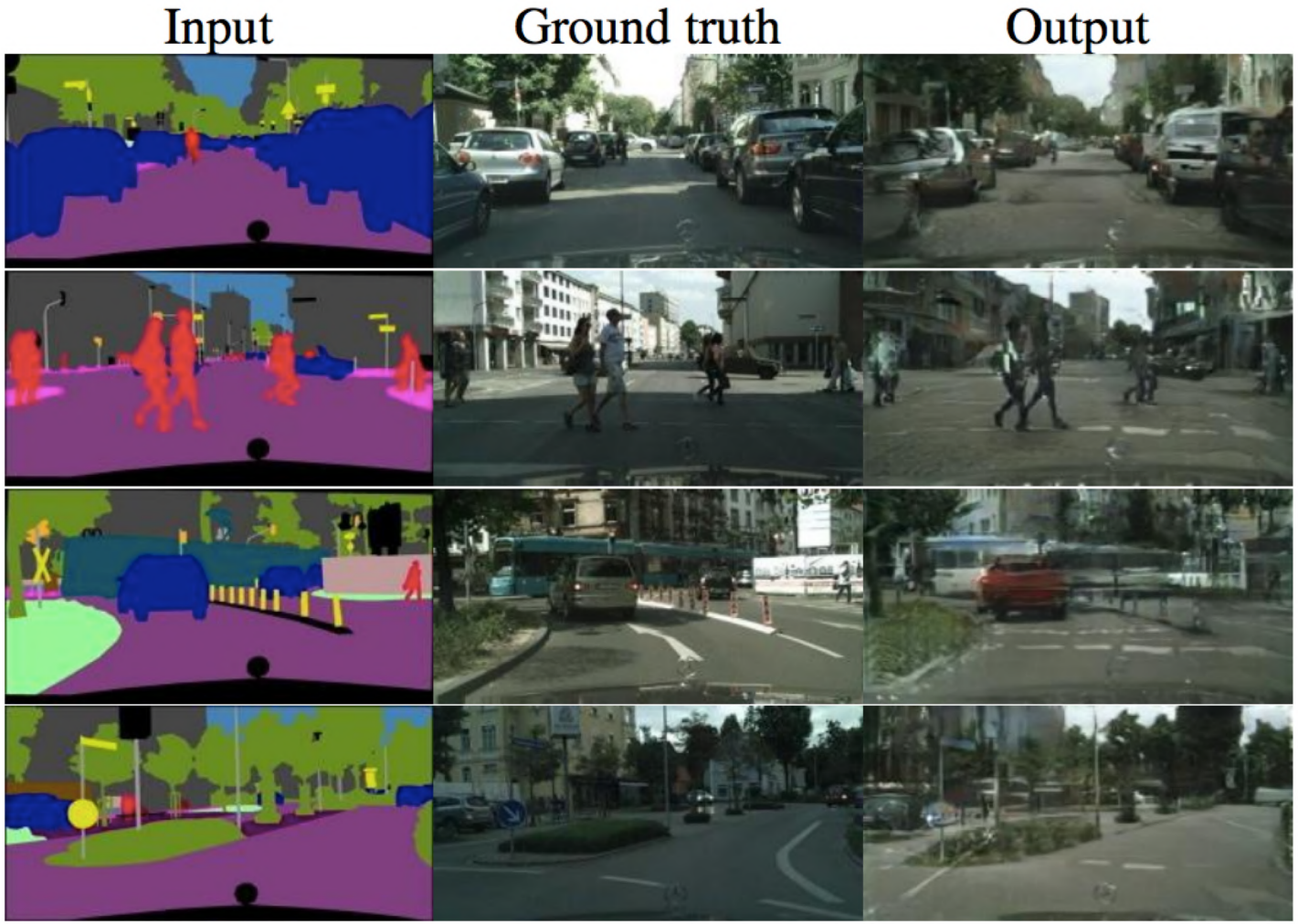


Image-to-Image Translation

- Semantic labels to facades

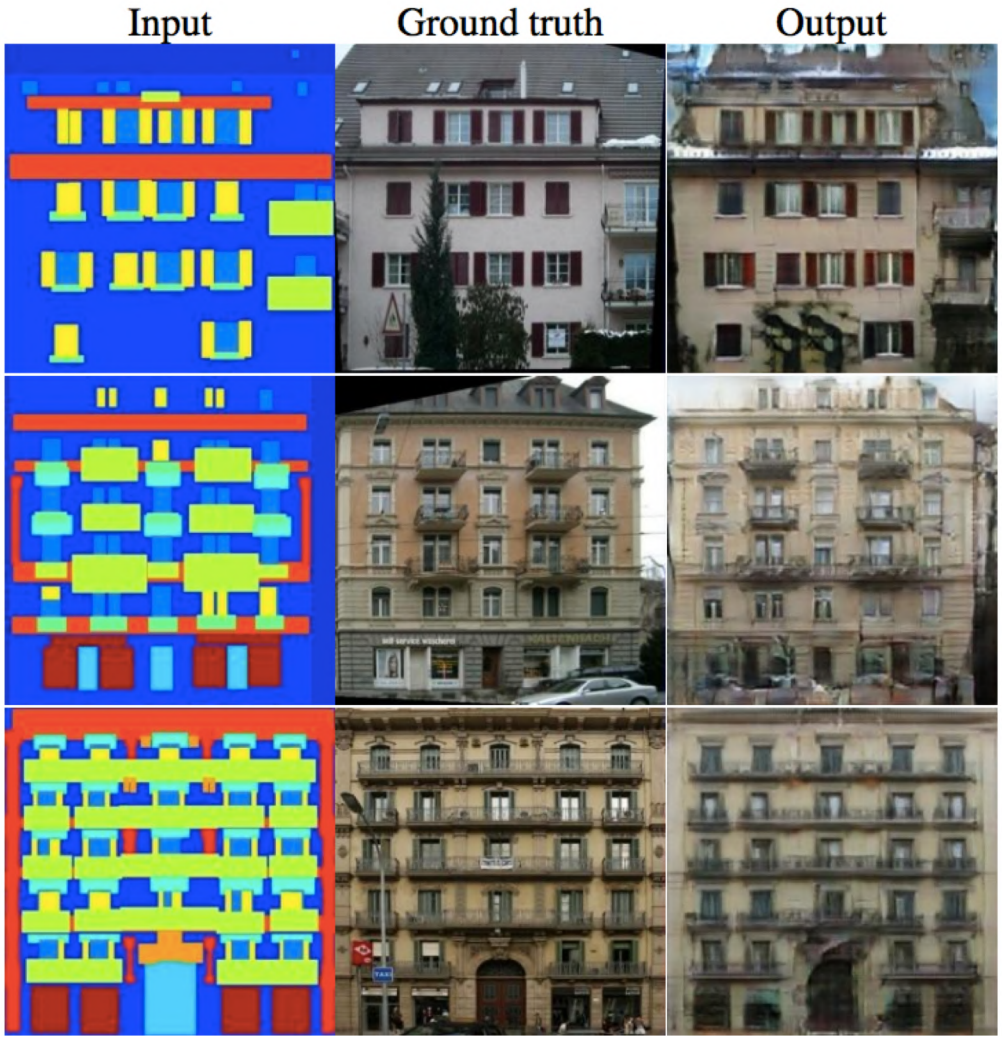


Image-to-Image Translation

- Day to night

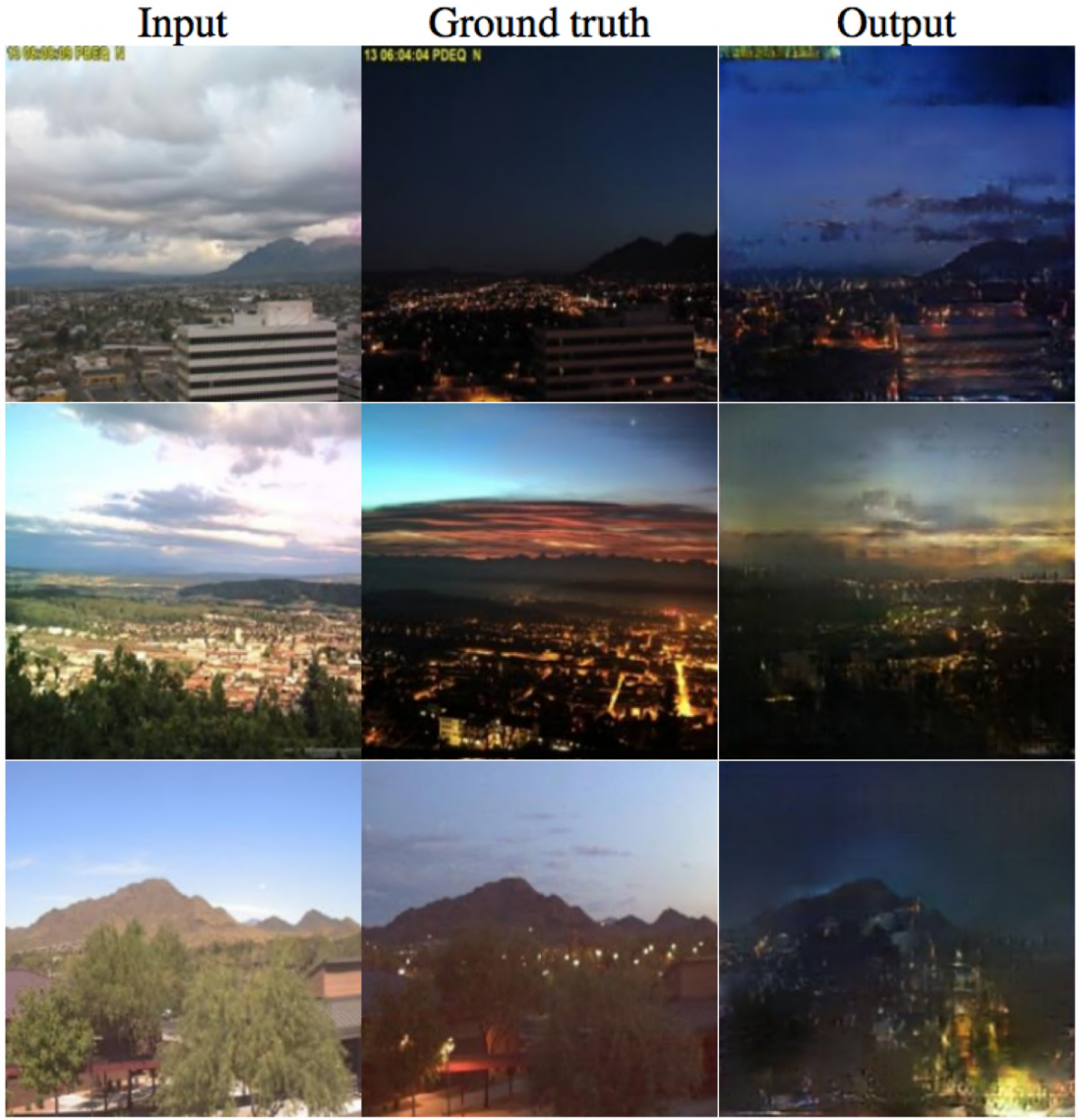
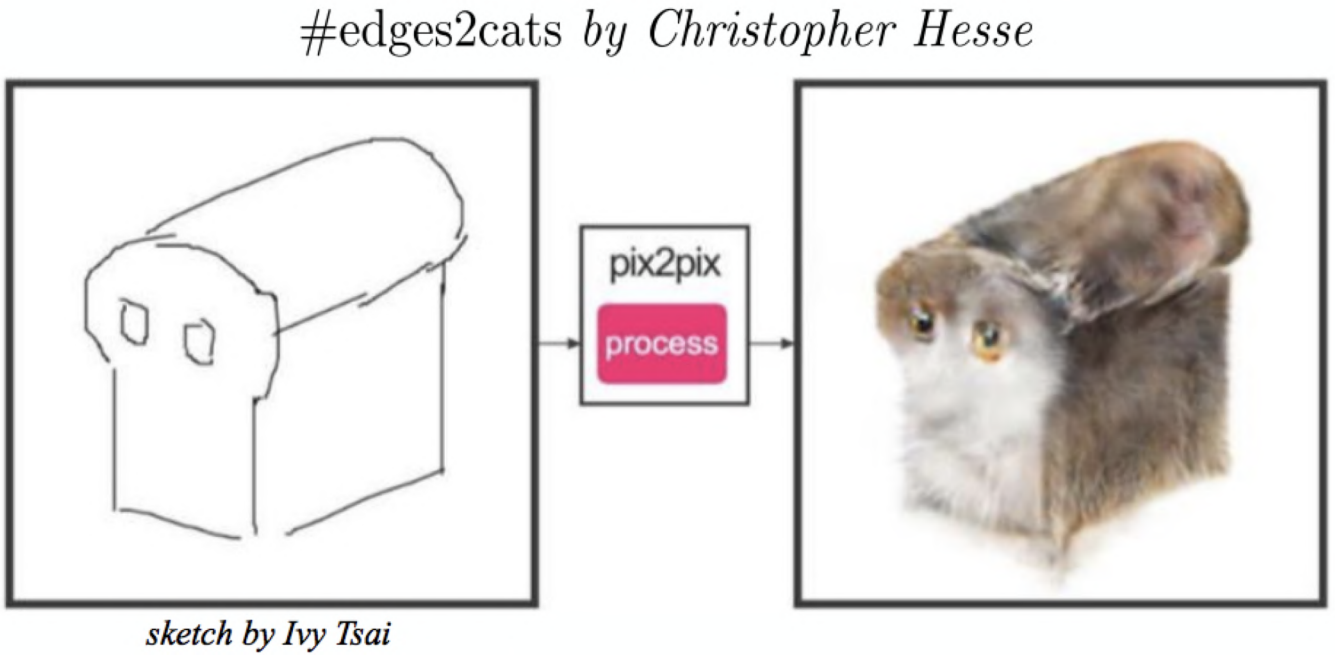


Image-to-Image Translation

- Edges to photos

pix2pix

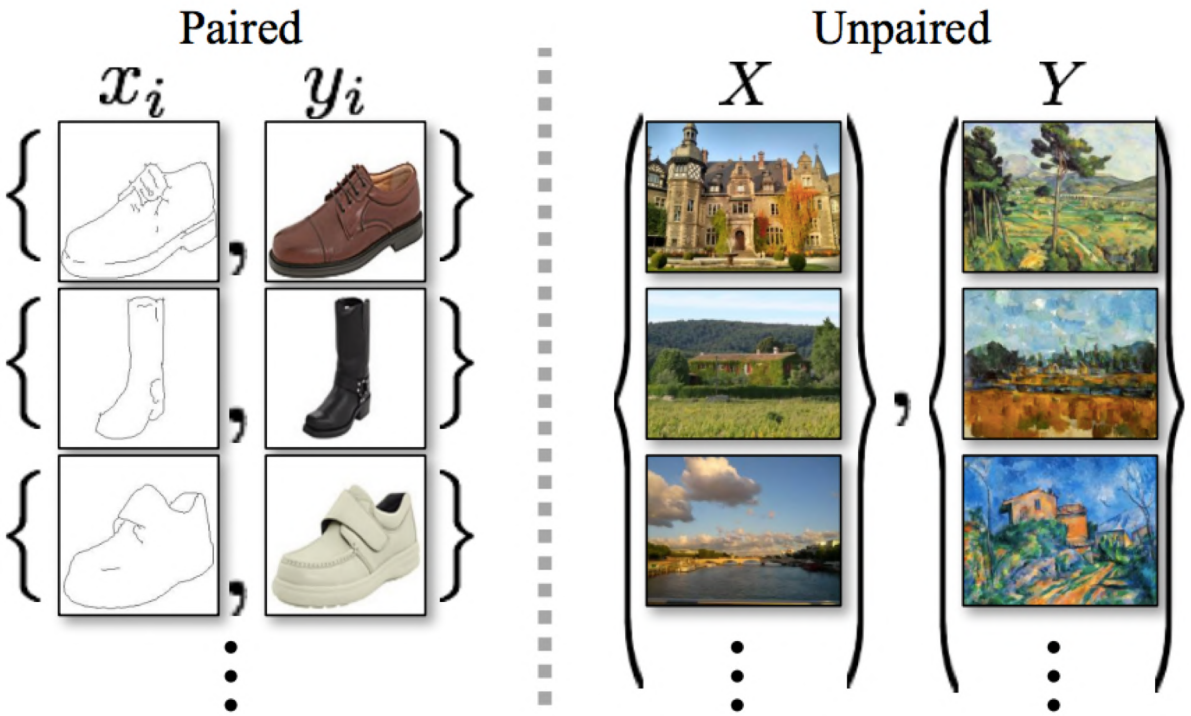


Twitter #edges2cats

pix2pix demo

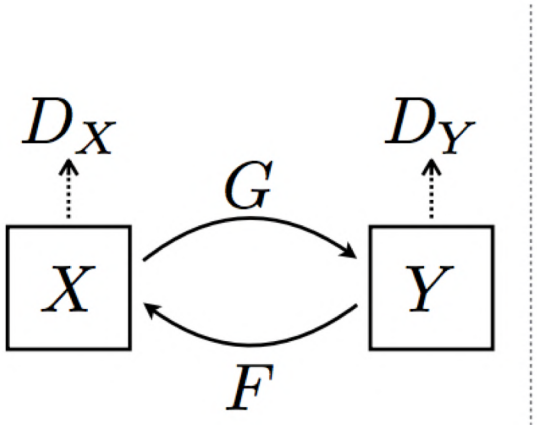
CycleGAN: Unpaired image-to-image translation

- Given two unordered image collections X and Y, learn to "translate" an image from one collection into the other and vice versa



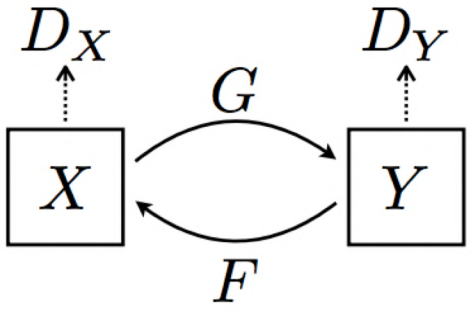
CycleGAN

- Given: domains X and Y
- Train two generators F and G and two discriminators D_X and D_Y
  - G translates from X to Y, F translates from Y to X
  - D_X recognizes images from X, and D_Y recognizes images from Y
  - We want $F(G(x)) \approx x$ and $G(F(y)) \approx y$



J.-Y. Zhu, T. Park, P. Isola, A. Efros, Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks, ICCV 2017

CycleGAN: Loss



CycleGAN **discriminator** loss:

$$\mathcal{L}_{\text{GAN}}(D_Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[(D_Y(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}\left[D_Y\left(G(x)\right)^2\right]$$

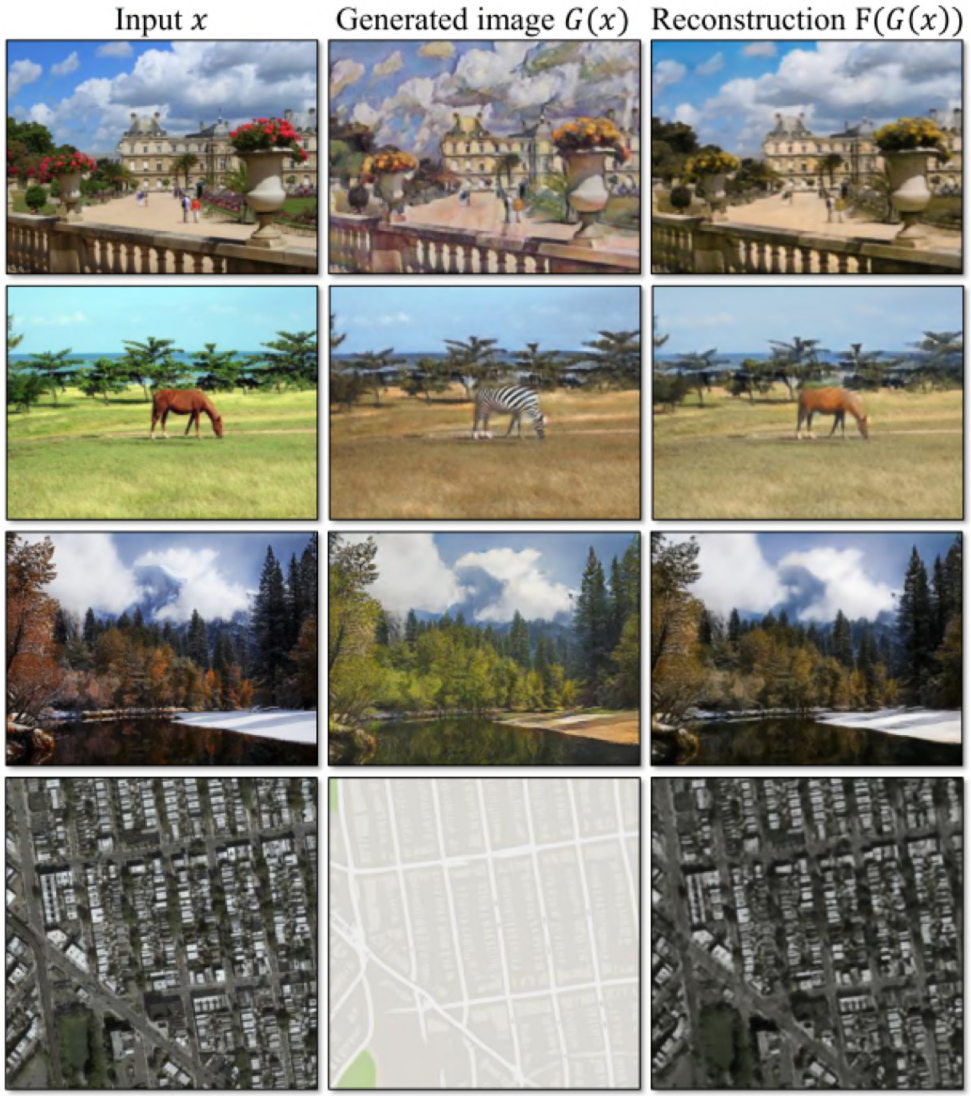
$$\mathcal{L}_{\text{GAN}}(D_X) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[(D_X(x) - 1)^2] + \mathbb{E}_{y \sim p_{\text{data}}(y)}\left[D_X\left(F(y)\right)^2\right]$$

CycleGAN **generator** loss:

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[(D_Y(G(x)) - 1)^2] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[(D_X(F(y)) - 1)^2]$$

$$+ \mathbb{E}_{x \sim p_{\text{data}}(x)}\left[\left\|F\left(G(x)\right) - x\right\|_1\right] + \mathbb{E}_{y \sim p_{\text{data}}(y)}\left[\left\|G\left(F(y)\right) - y\right\|_1\right]$$

CycleGAN

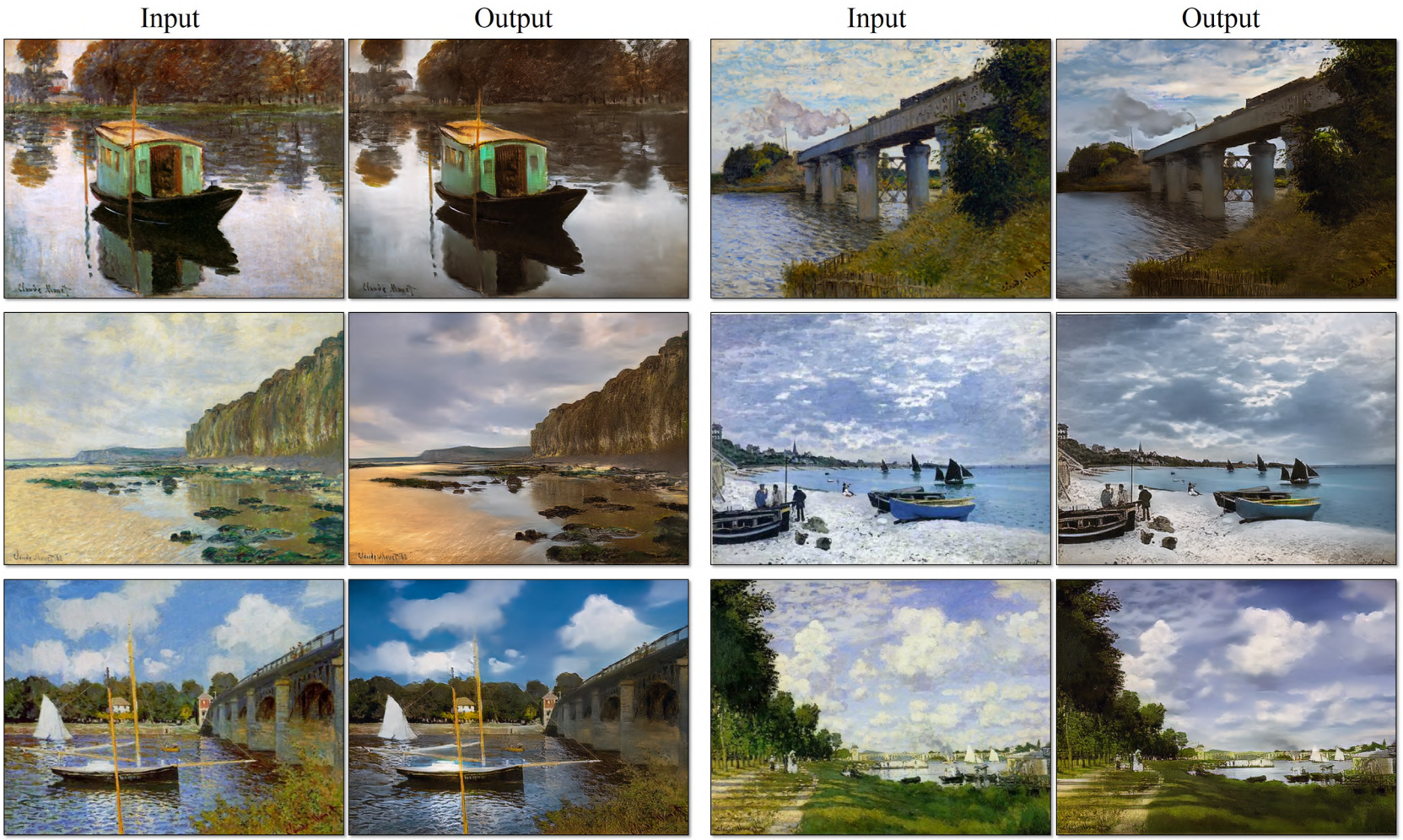


CycleGAN

winter Yosemite → summer Yosemite

summer Yosemite → winter Yosemite

CycleGAN



Monet Paintings to Photos

CycleGAN

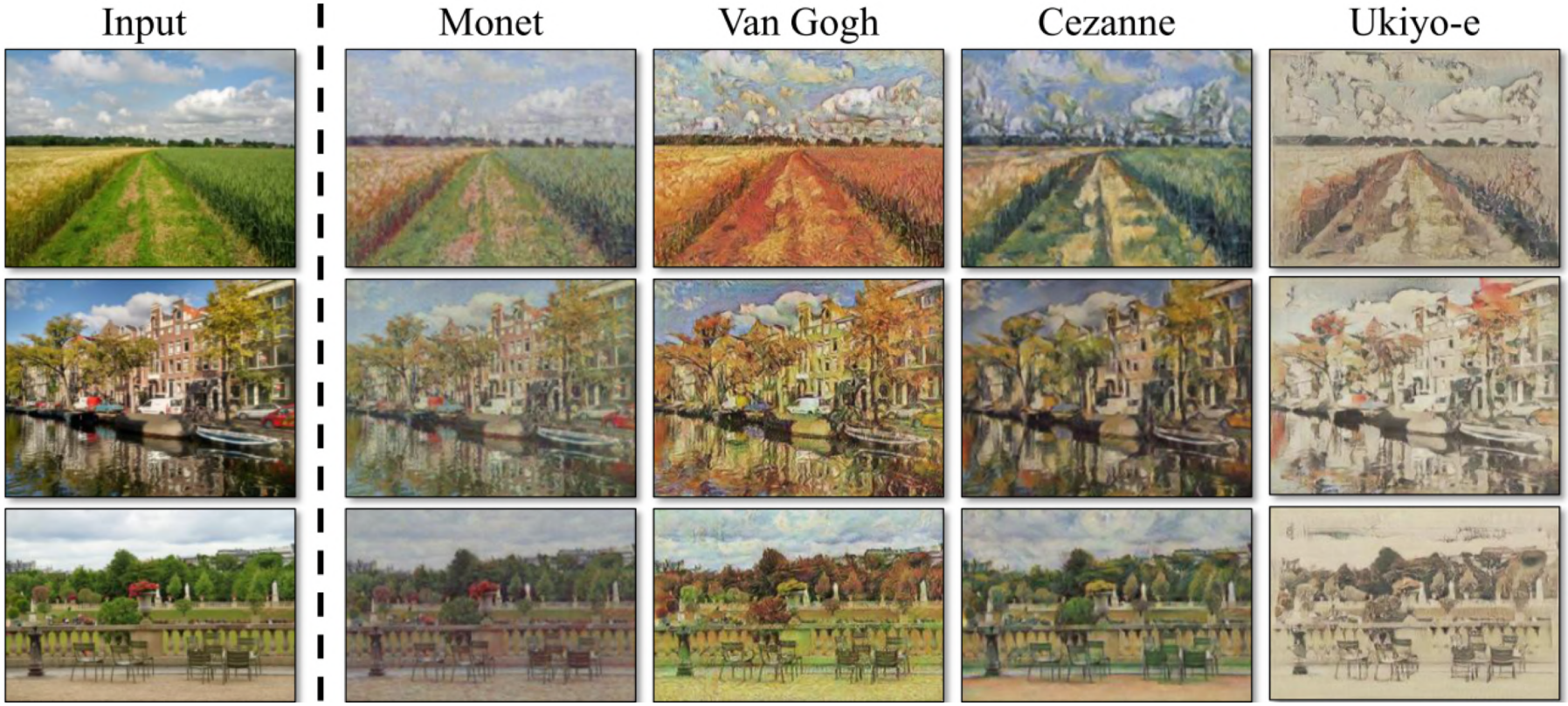
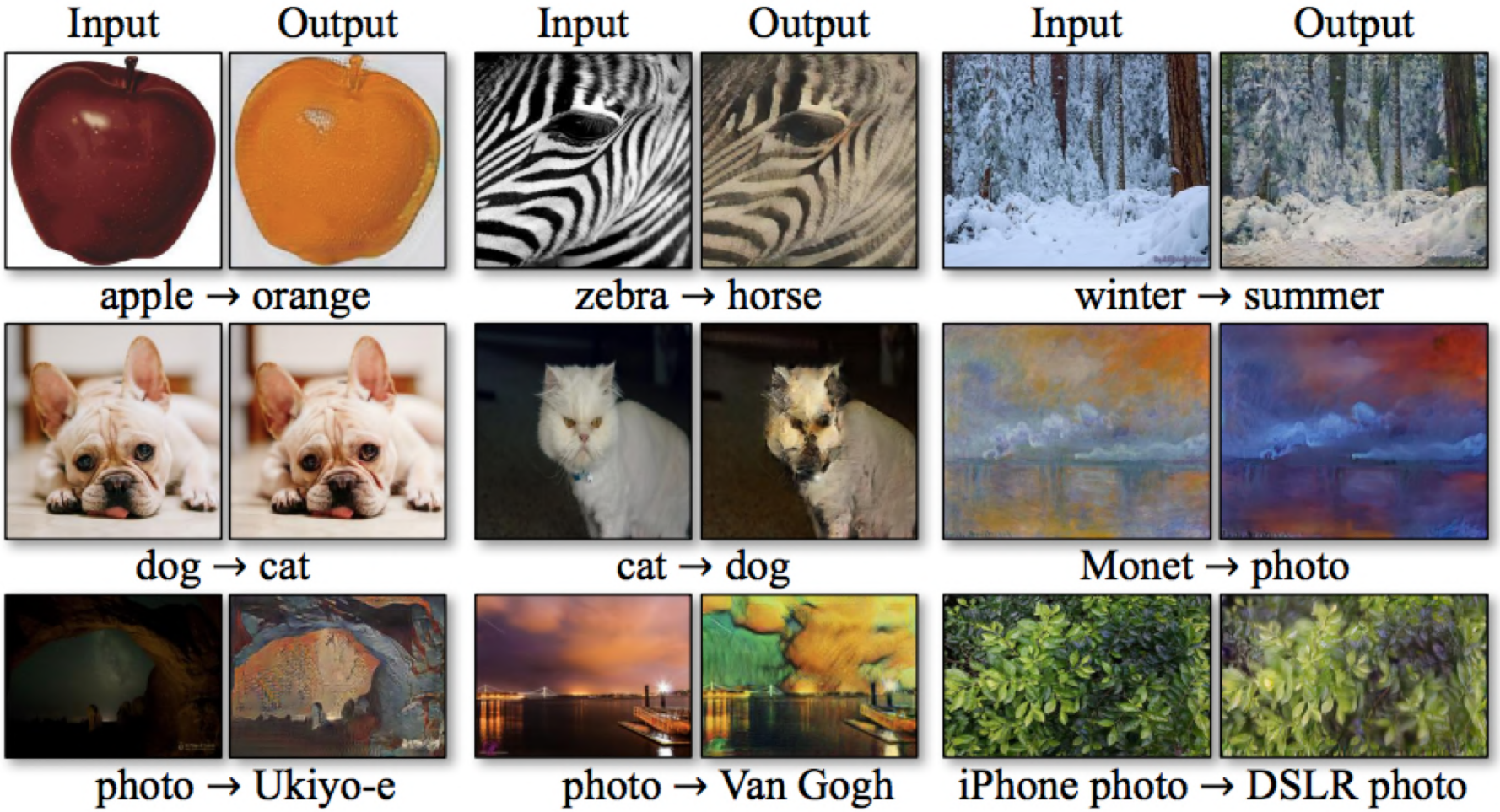
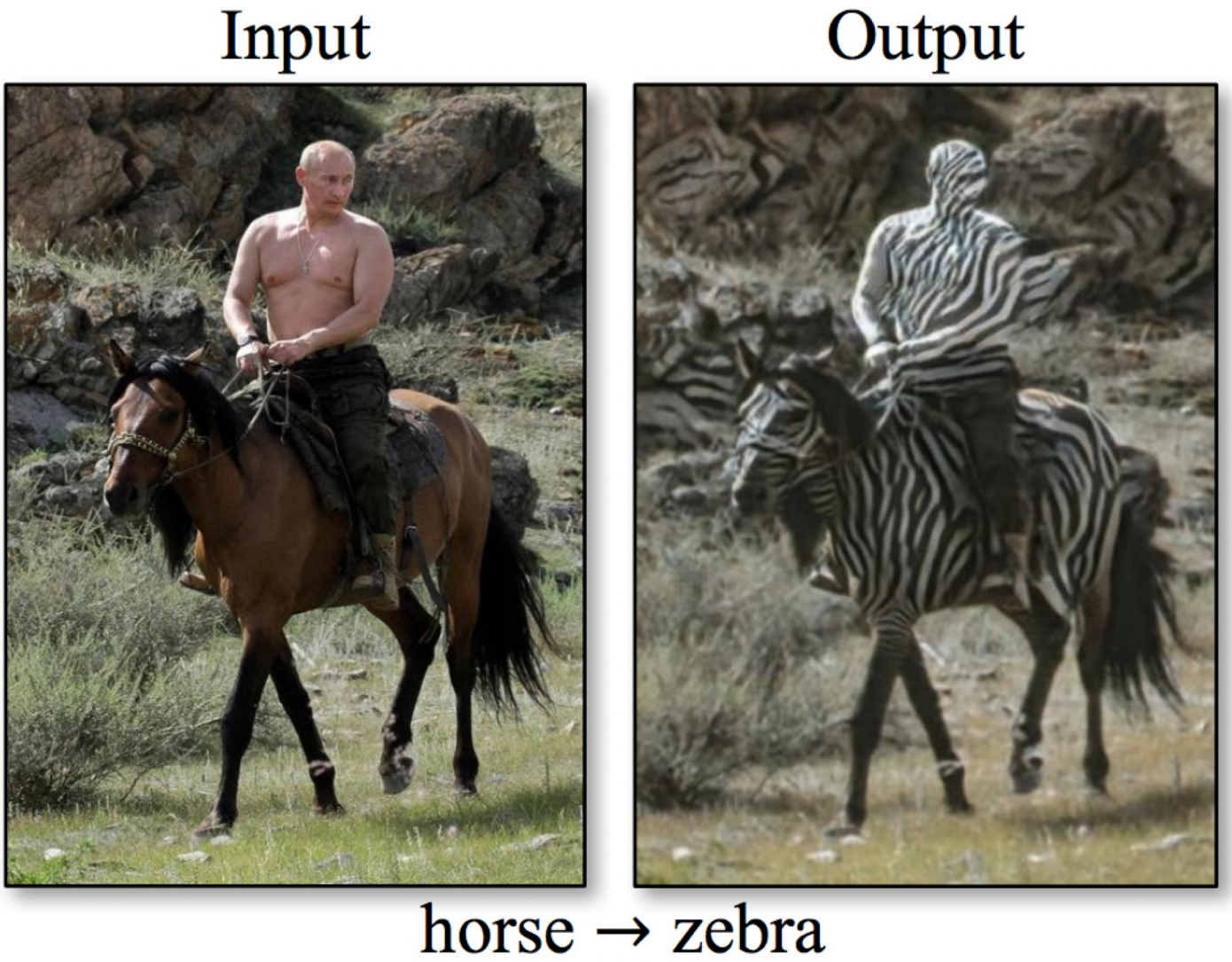


Photo to Paintings

CycleGAN: Failure cases



CycleGAN: Failure cases



CycleGAN: Limitations

- Cannot handle shape changes (e.g., dog to cat)
- Can get confused on images outside of the training domains (e.g., horse with rider)
- Cannot close the gap with paired translation methods
- One transformation direction may be more challenging than the other

Impressive applications of GANs

- Generate Examples for Image Datasets
- Generate Photographs of Human Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation
- Face Frontal View Generation
- Generate New Human Poses
- Photos to Emojis
- Photograph Editing
- Face Aging
- Photo Blending
- Super Resolution
- Photo Inpainting
- Clothing Translation
- Video Prediction
- 3D Object Generation

See https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

Interesting New Yorker article

DEPT. OF TECHNOLOGY NOVEMBER 12, 2018 ISSUE

THE NEW YORKER

IN THE AGE OF A.I., IS SEEING STILL BELIEVING?

Advances in digital imagery could deepen the fake-news crisis—or help us get out of it.

By Joshua Rothman

The age of deepFakes:
https://www.newyorker.com/magazine/2018/11/12/in-the-age-of-ai-is-seeing-still-believing

Reading List

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. Generative adversarial nets, NIPS (2014).

Goodfellow, Ian NIPS 2016 Tutorial: Generative Adversarial Networks, NIPS (2016).

Radford, A., Metz, L. and Chintala, S., Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434. (2015).

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. Improved techniques for training gans. NIPS (2016).

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets, NIPS (2016).

Zhao, Junbo, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126 (2016).

Mirza, Mehdi, and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014).

Liu, Ming-Yu, and Oncel Tuzel. Coupled generative adversarial networks. NIPS (2016).

Denton, E.L., Chintala, S. and Fergus, R., 2015. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. NIPS (2015)

Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., & Courville, A. Adversarially learned inference. arXiv preprint arXiv:1606.00704 (2016).

Applications:

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004. (2016).

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. Generative adversarial text to image synthesis. JMLR (2016).

Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). Face Aging With Conditional Generative Adversarial Networks. arXiv preprint arXiv:1702.01983.

Influential Papers

- DCGAN https://arxiv.org/pdf/1511.06434v2.pdf
- Wasserstein GAN (WGAN) https://arxiv.org/pdf/1701.07875.pdf
- Conditional Generative Adversarial Nets (CGAN) https://arxiv.org/pdf/1411.1784v1.pdf
- Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks (LAPGAN) https://arxiv.org/pdf/1506.05751.pdf
- Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (SRGAN) https://arxiv.org/pdf/1609.04802.pdf
- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (CycleGAN) https://arxiv.org/pdf/1703.10593.pdf
- InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets https://arxiv.org/pdf/1606.03657
- DCGAN https://arxiv.org/pdf/1704.00028.pdf
- Improved Training of Wasserstein GANs (WGAN-GP) https://arxiv.org/pdf/1701.07875.pdf
- Energy-based Generative Adversarial Network (EBGAN) https://arxiv.org/pdf/1609.03126.pdf
- Autoencoding beyond pixels using a learned similarity metric (VAE-GAN) https://arxiv.org/pdf/1512.09300.pdf
- Adversarial Feature Learning (BiGAN) https://arxiv.org/pdf/1605.09782v6.pdf
- Stacked Generative Adversarial Networks (SGAN) https://arxiv.org/pdf/1612.04357.pdf
- StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks https://arxiv.org/pdf/1710.10916.pdf
- Learning from Simulated and Unsupervised Images through Adversarial Training (SimGAN) https://arxiv.org/pdf/1612.07828v1.pdf
-

Course Recap

Course Goals:

- Students will be introduced the **basic principles** and **dev tools** of Deep Learning in the context of Image Processing and Visual Understanding:
  - Linear Regression
  - Linear Classifiers
  - Multi-Layer, Back-propagation
  - Training and Optimization
  - Convolutional Neural Network
  - Basic architectures
  - Image Detection, Localization, ~~Segmentation~~
  - Recurrent Neural networks, Attention models, transformers
  - Generative models, Generative Adversarial Networks

Further Learning:

- Dozens of articles every day in https://arxiv.org/
- More courses: Deep Learning Specialization by Andrew Ng, excellent serious of courses
- SeedBank: Collection of Interactive Machine Learning Examples and tutorials using *Colab*
- Join the MDLI (Machine & Deep Learning Israel) Facebook group.
- More Resources:
  - Two Minute papers
  - Yannic Klicher's videos
- Explore and learn! Blogs, papers, videos,…
- Read everything but don't trust anything.

Hope you have enjoyed the course and have had a good learning experience

THE END