Aprendizado Profundo (Deep Learning)

Generative Adversarial Networks (GANs)

Dario Oliveira
(dario.oliveira@fgv.br)

Overview

Motivation

Introduction to GANs

Conditional GANs

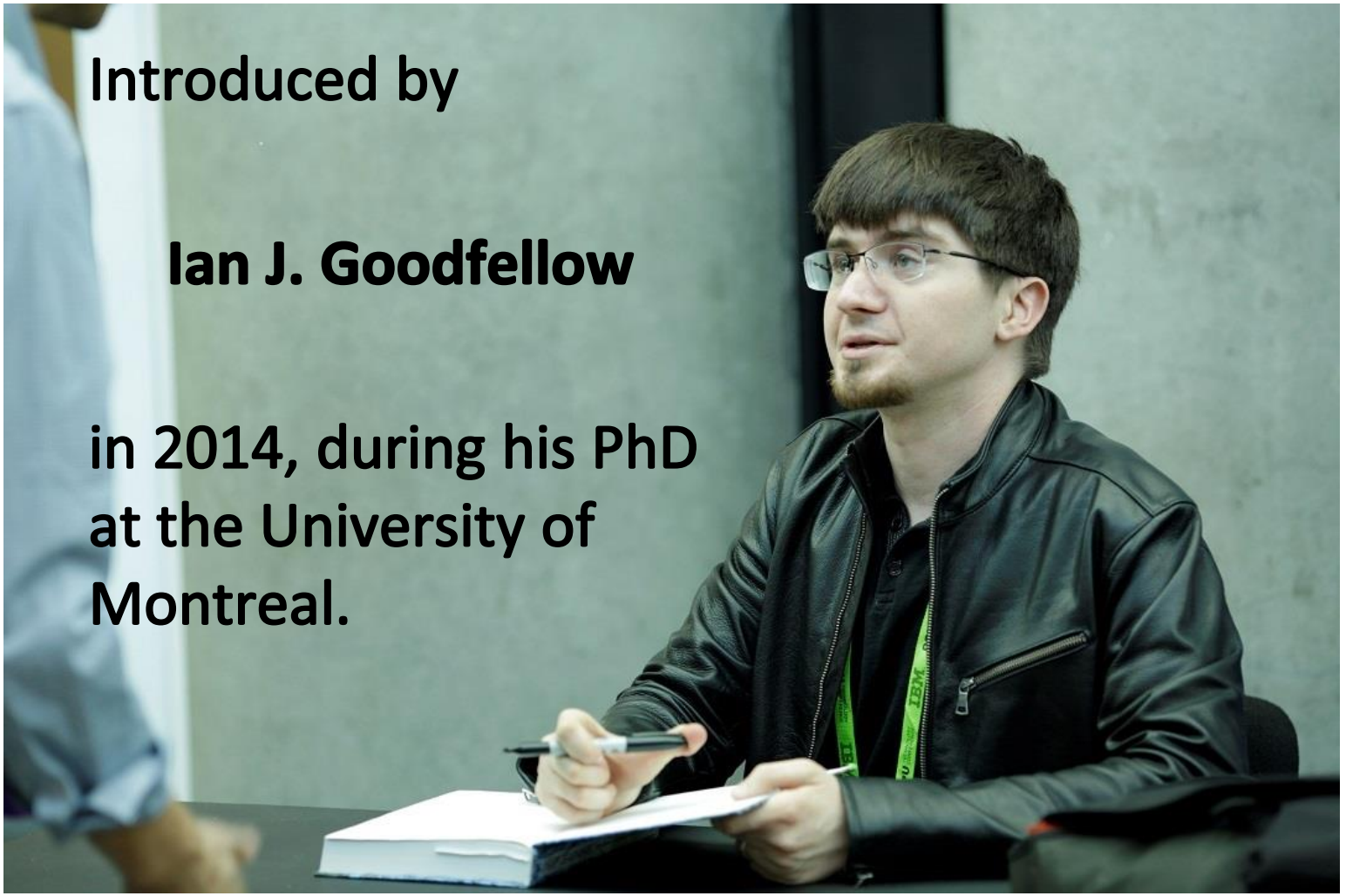
Cycle GANs

Other Applications

GANs

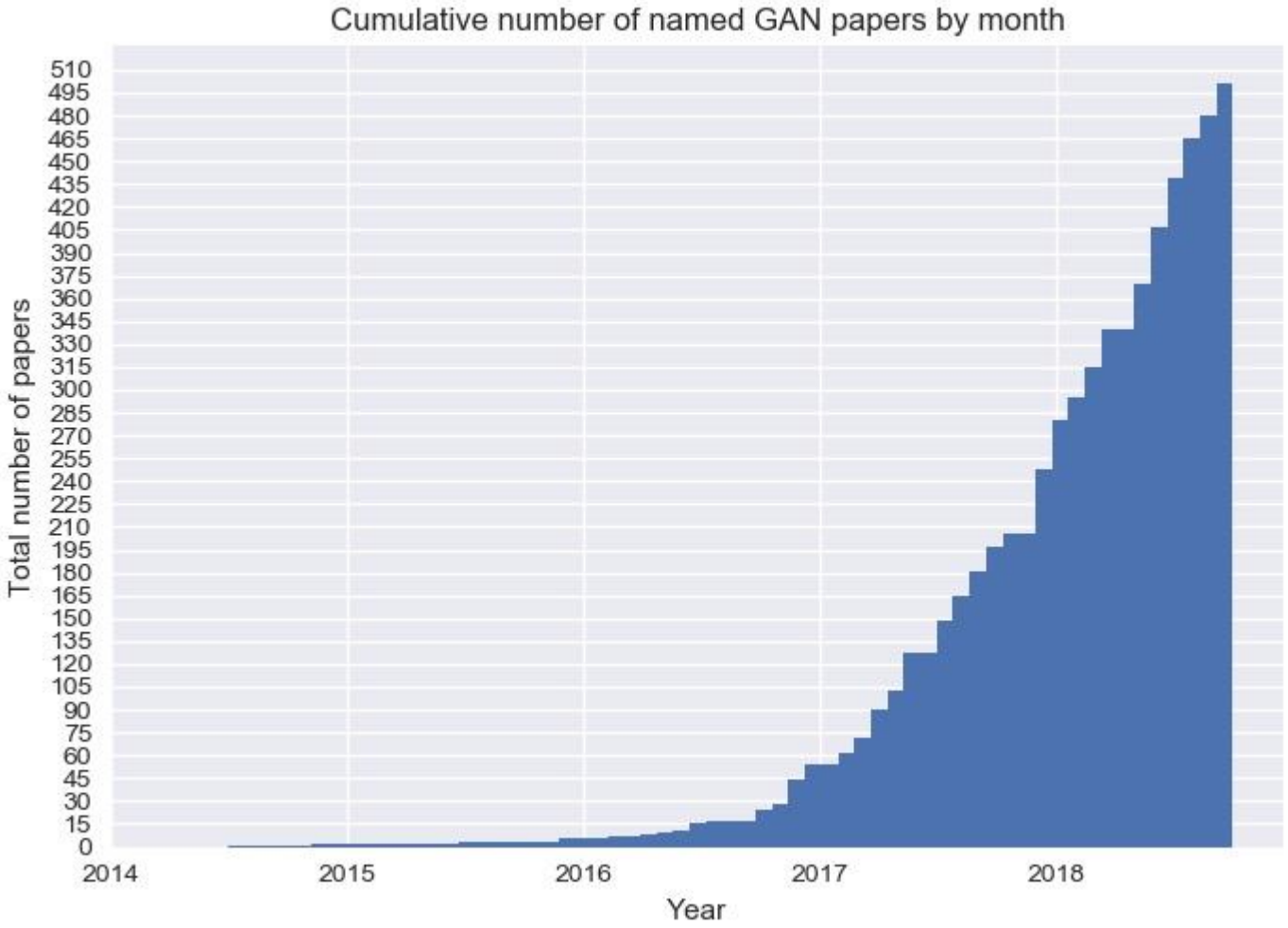
"Generative Adversarial Networks is the most interesting idea in the last 10 years in Machine Learning." Yann LeCun, Director of AI Research at Facebook AI, 2016

GANs



Goodfellow et al., 2014, Generative Adversarial Networks, available at https://arxiv.org/abs/1406.2661

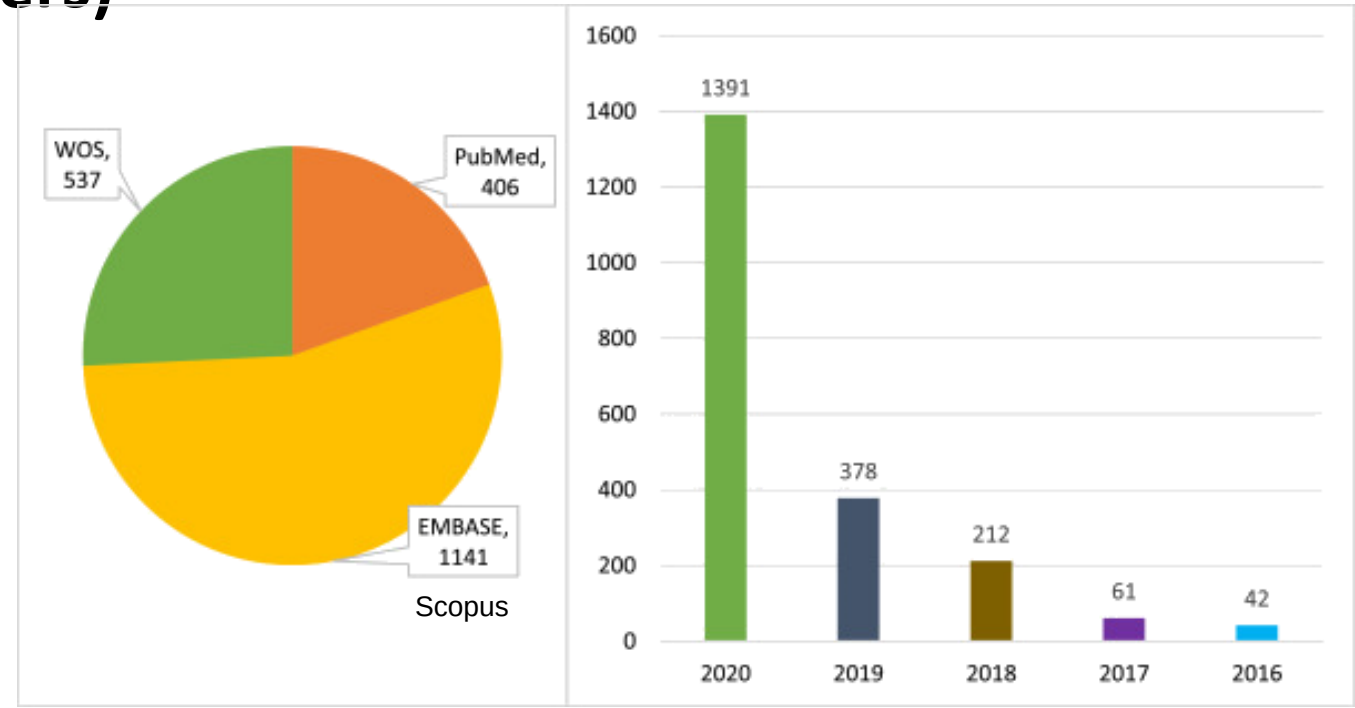
The GAN Boom



Source: GAN Zoo - https://github.com/hindupuravinash/the-gan-zoo , last access May 20th, 2020

Recent Publications on GANs

Distribution of recent scientific publications focusing on GANs over the recent years (out of 2084 papers)



Aggarwal et al., 2021, Generative adversarial network: An overview of theory and applications

Overview

Motivation

Introduction to GANs

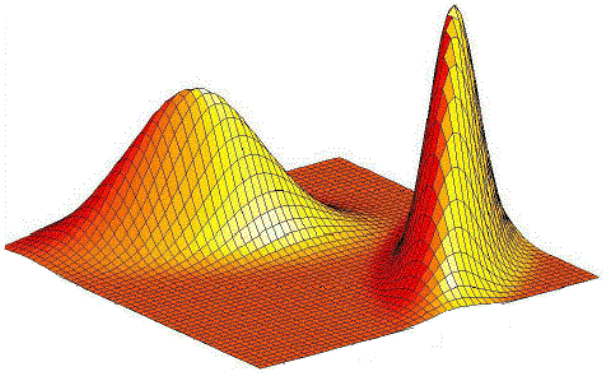
Conditional GANs

Cycle GANs

Other Applications

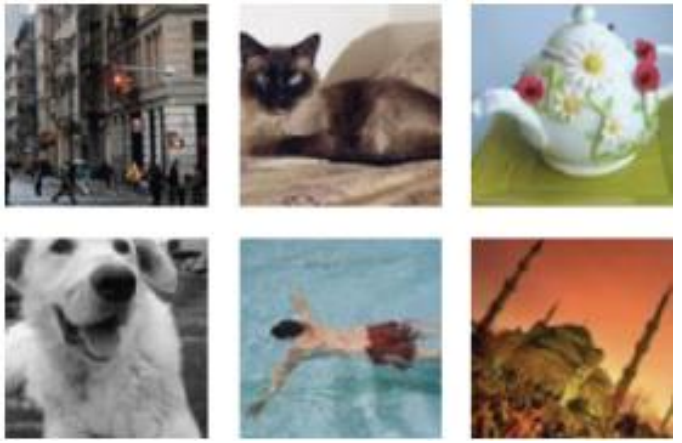
Generative Models

learn the distribution underlying the training samples



generate new samples from the learned distribution

training samples



generated samples

images credited by Goodfellow. 2016

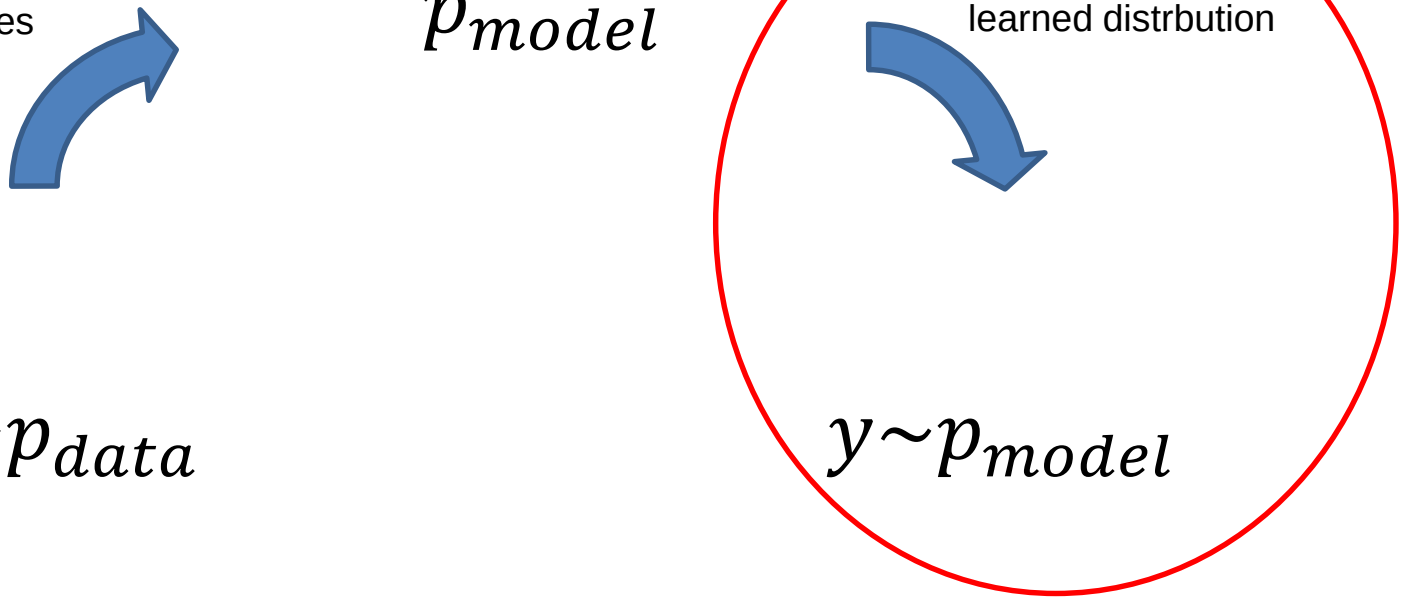
Generative Models

if p_{model} is a high dimensional complex distribution, there is no direct way to do this

learn the distribution underlying the training samples

p_{model}

generate new samples from the learned distrbution



$x \sim p_{data}$

$y \sim p_{model}$

training samples

generated samples

Generative Adversarial Network

learn the distribution underlying the training samples

p_{model}

generate new samples from the learned distribution

$x \sim p_{data}$ **Transformation** $y \sim p_{model}$

training samples

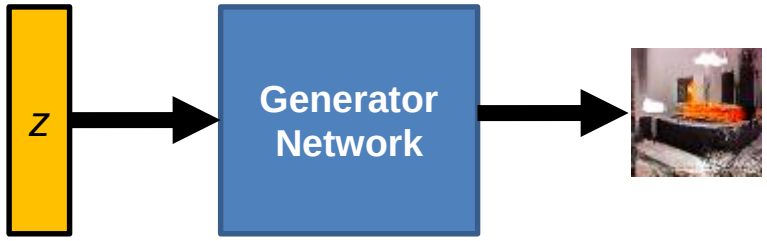
generated samples

Generative Adversarial Network

GANs adopt the following alternative approach:

Sample from a simple distribution, e.g., random noise, and learn a transformation to a sample of the target distribution

Question: How can we design such a complex function that maps samples from a simple distribution to a complex target distribution?



input:

random noise

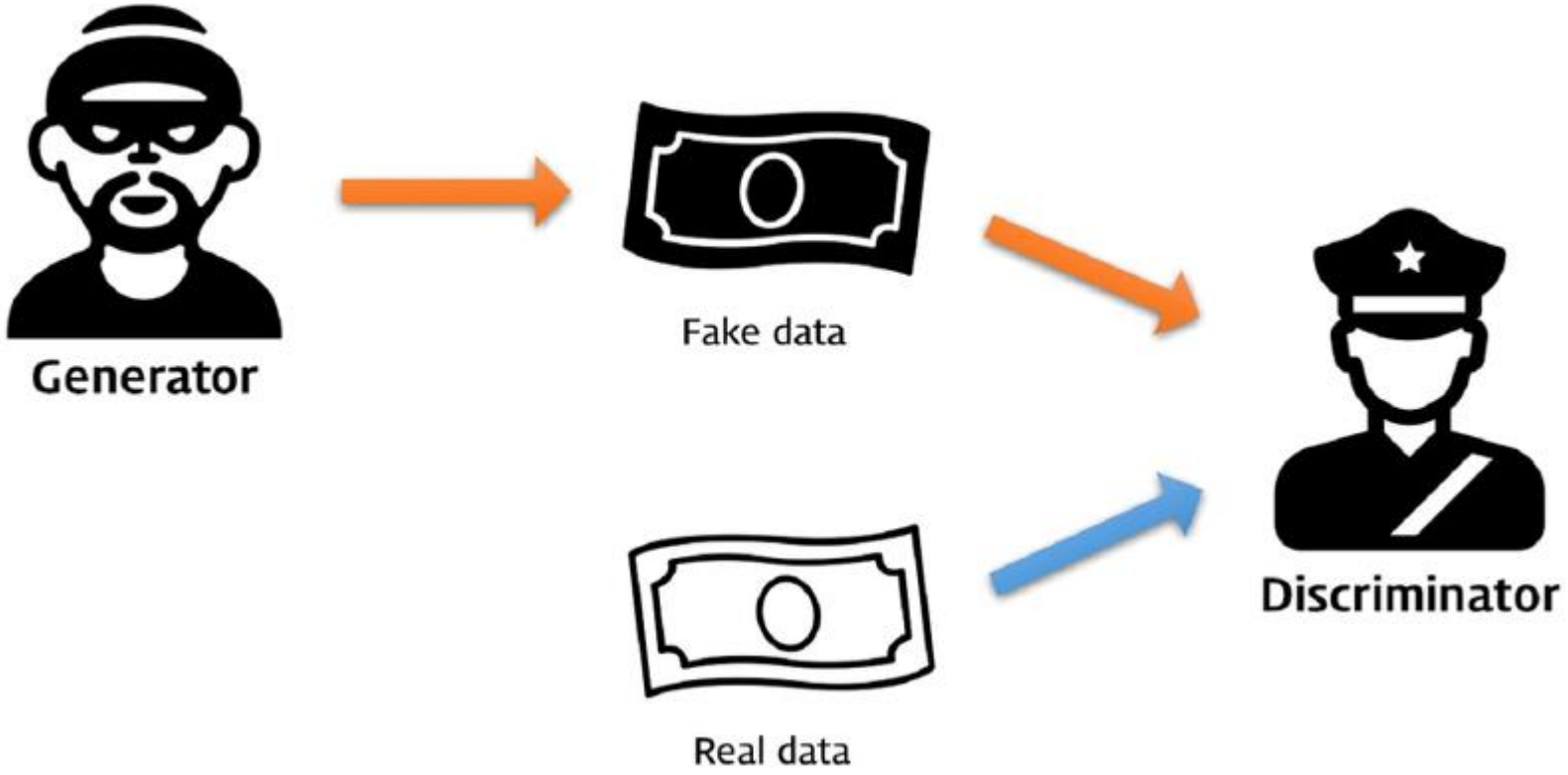
output:

sample from training distribution

Answer: a neural network

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Training GANs: two-player game

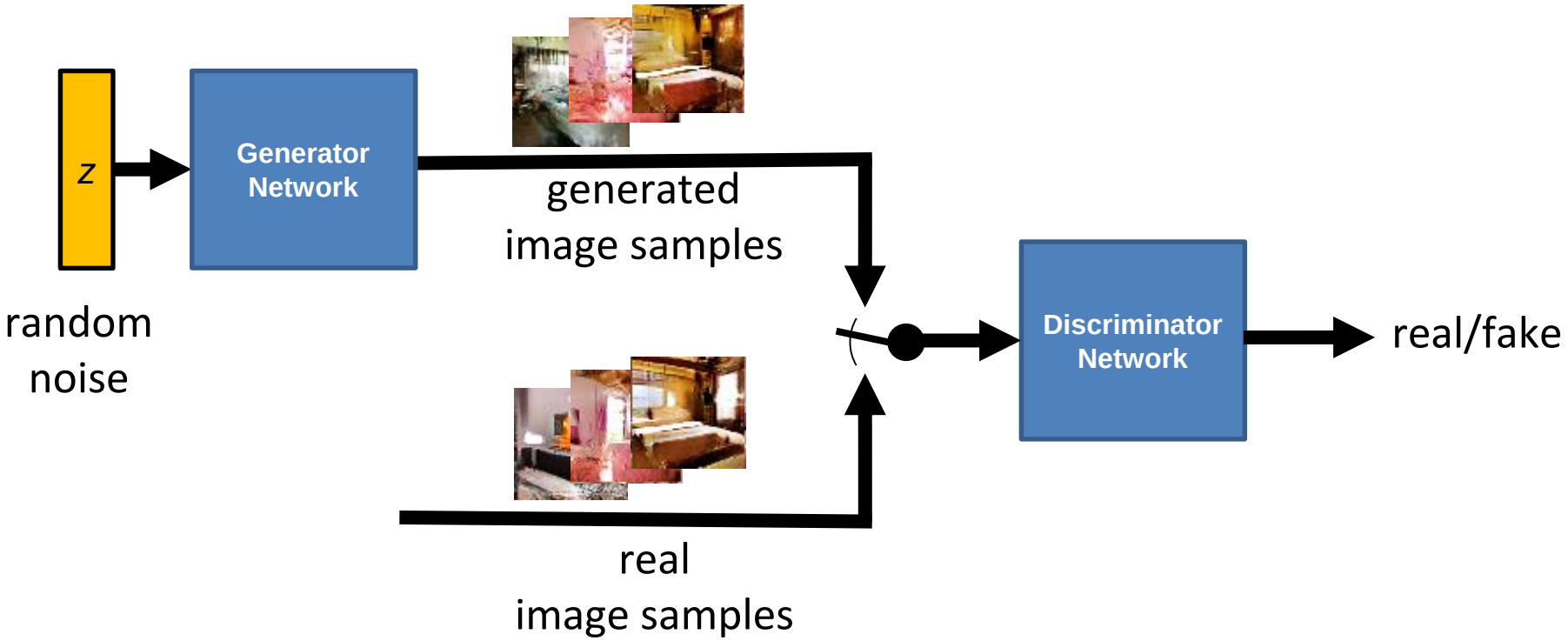


Source https://sites.google.com/site/aidysft/generativeadversialnetwork

Training GANs: two-player game

Generator network: try to fool the discriminator by generating realistic images

Discriminator network: try to distinguish between real and fake images



Training GANs: two-player game

Generator network: try to fool the discriminator by generating realistic images

Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minmax objective function:

Discriminator outputs likelihood in (0,1) real interval

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} \left[\mathbb{E}_{\boldsymbol{x} \sim p_{data}} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} \log \left(1 - D_{\boldsymbol{\theta}_d}\left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right) \right) \right]$$

Discriminator output for real data $\boldsymbol{x}$

Discriminator output for fake data $G_{\boldsymbol{\theta}_g}(\boldsymbol{z})$

Discriminator parameters

Generator parameters

$\mathbb{E}$ is the **expected value operator**. In practice, we take the average over all samples.

Training GANs: two-player game

Generator network: try to fool the discriminator by generation realistic images

Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minmax objective function:

Discriminator outputs likelihood in (0,1) real interval

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} \left[\mathbb{E}_{\boldsymbol{x} \sim p_{data}} \log \underbrace{D_{\boldsymbol{\theta}_d}(\boldsymbol{x})}_{\text{Discriminator output for real data } \boldsymbol{x}} + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} \log \left(1 - \underbrace{D_{\boldsymbol{\theta}_d}\left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right)}_{\text{Discriminator output for fake data } G_{\boldsymbol{\theta}_g}(\boldsymbol{z})} \right) \right]$$

- Discriminator ($\boldsymbol{\theta}_d$) wants to **maximize objective** such $D_{\boldsymbol{\theta}_d}(\boldsymbol{x})$ is close to 1 for real and $D_{\boldsymbol{\theta}_d}\left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right)$ is close to 0 for fake.
- Generator ($\boldsymbol{\theta}_g$) wants to **minimize objective** such that $D_{\boldsymbol{\theta}_d}\left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right)$ is close to 1 (discriminator is fooled into thinking generated $G_{\boldsymbol{\theta}_g}(\boldsymbol{z})$ is real)

Training GANs: two-player game

Minmax objective function:

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} \left[\mathbb{E}_{\boldsymbol{x} \sim p_{data}} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} \log \left(1 - D_{\boldsymbol{\theta}_d} \left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right) \right) \right]$$

Alternate between:

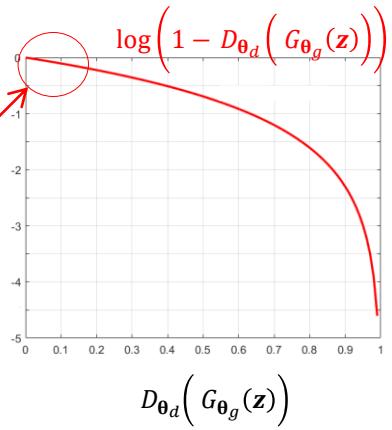
1. **Gradient ascent** on discriminator (keep $\boldsymbol{\theta}_g$ constant)

$$\max_{\boldsymbol{\theta}_d} \left[\mathbb{E}_{\boldsymbol{x} \sim p_{data}} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} \log \left(1 - D_{\boldsymbol{\theta}_d} \left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right) \right) \right]$$

2. **Gradient descent** on generator (keep $\boldsymbol{\theta}_d$ constant)

$$\min_{\boldsymbol{\theta}_g} \left[\mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} \log \left(1 - D_{\boldsymbol{\theta}_d} \left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right) \right) \right]$$

At the beginning fake samples will be easily recognized by the discriminator, but the gradient in this region is flat → slow learning!



Training GANs: two-player game

Minmax objective function:

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} \left[\mathbb{E}_{\boldsymbol{x} \sim p_{data}} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} \log \left(1 - D_{\boldsymbol{\theta}_d} \left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right) \right) \right]$$

Alternate between:

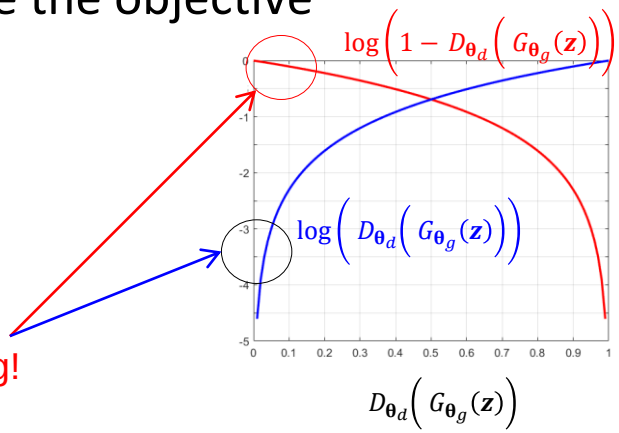
1. **Gradient ascent** on discriminator (keep $\boldsymbol{\theta}_g$ constant)

$$\max_{\boldsymbol{\theta}_d} \left[\mathbb{E}_{\boldsymbol{x} \sim p_{data}} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} \log \left(1 - D_{\boldsymbol{\theta}_d} \left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right) \right) \right]$$

2. **Instead: Gradient ascent** on generator – change the objective

$$\max_{\boldsymbol{\theta}_g} \left[\mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} \log \left(D_{\boldsymbol{\theta}_d} \left(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \right) \right) \right]$$

At the beginning fake samples will be easily recognized by the discriminator, but the gradient in this region is flat → slow learning!



Training GANs: summary

for number of training iterations **do**

for *k* steps **do**

- Sample minibatch of *m* noise samples $\{\boldsymbol{z}^{(1)} \dots \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$
- Sample minibatch of *m* examples $\{\boldsymbol{x}^{(1)} \dots \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{data}(\boldsymbol{x})$
- Update the discriminator

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D_{\theta_d}\left(G_{\theta_g}\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$

end for

- Sample minibatch of *m* noise samples $\{\boldsymbol{z}^{(1)} \dots \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$
- Update the generator by ascending its stochastic gradient (improved objective)

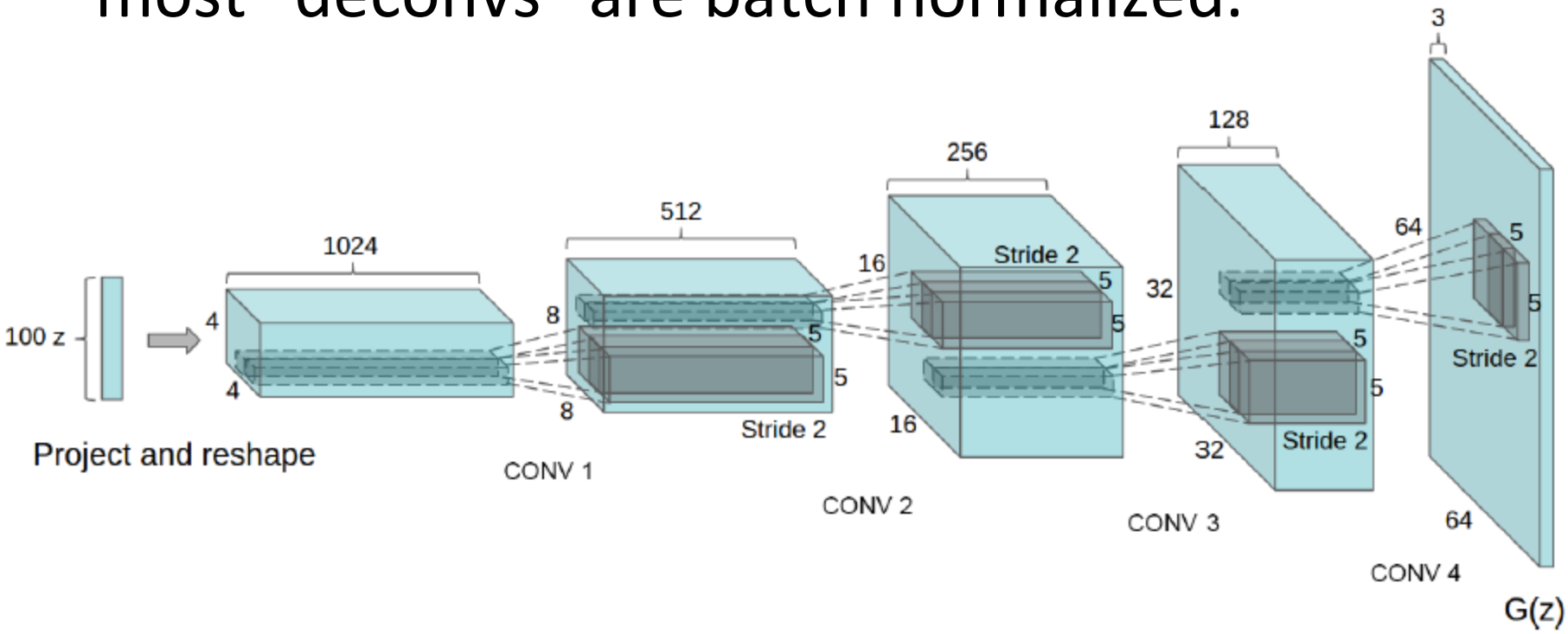
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D_{\theta_d}\left(G_{\theta_g}\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

end for

Generator Implementation

DCGANS:

most “deconvs” are batch normalized.



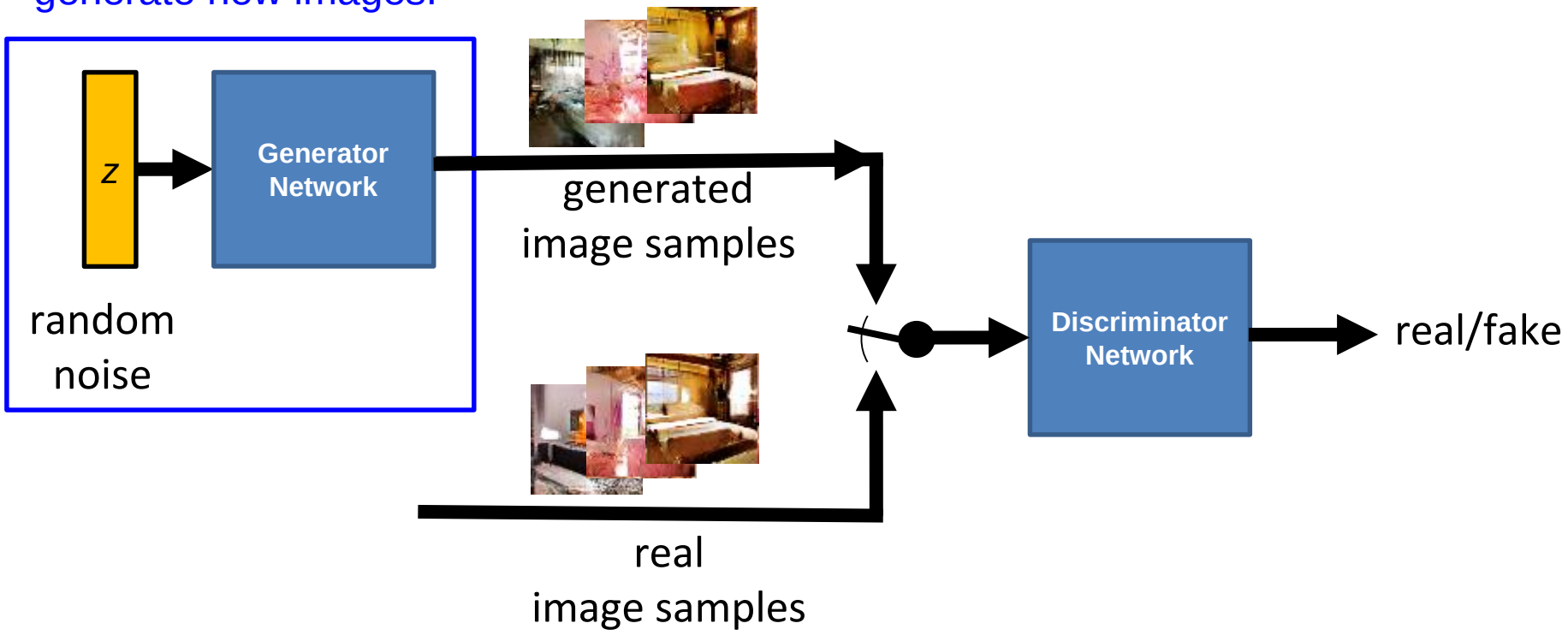
Radfort et al. 2016, UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Once trained…

Generator network: try to fool the discriminator by generating realistic images

Discriminator network: try to distinguish between real and fake images

After training, use generator network to generate new images.



Examples of DCGAN images

Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

Radfort et al. 2016, UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

BigGAN for High Fidelity Image Synthesis

Sample results

[1]Brock et al., CVPR 2019, available at https://arxiv.org/abs/1809.11096

BigGAN for High Fidelity Image Synthesis

Sample results

Available in Youtube at https://www.youtube.com/watch?v=sW6D34mckkk

Other examples at https://www.youtube.com/watch?v=YY6LrQSxIbc

Overview

Motivation

Introduction to GANs

Conditional GANs

Cycle GANs

Other Applications

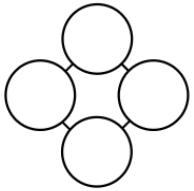
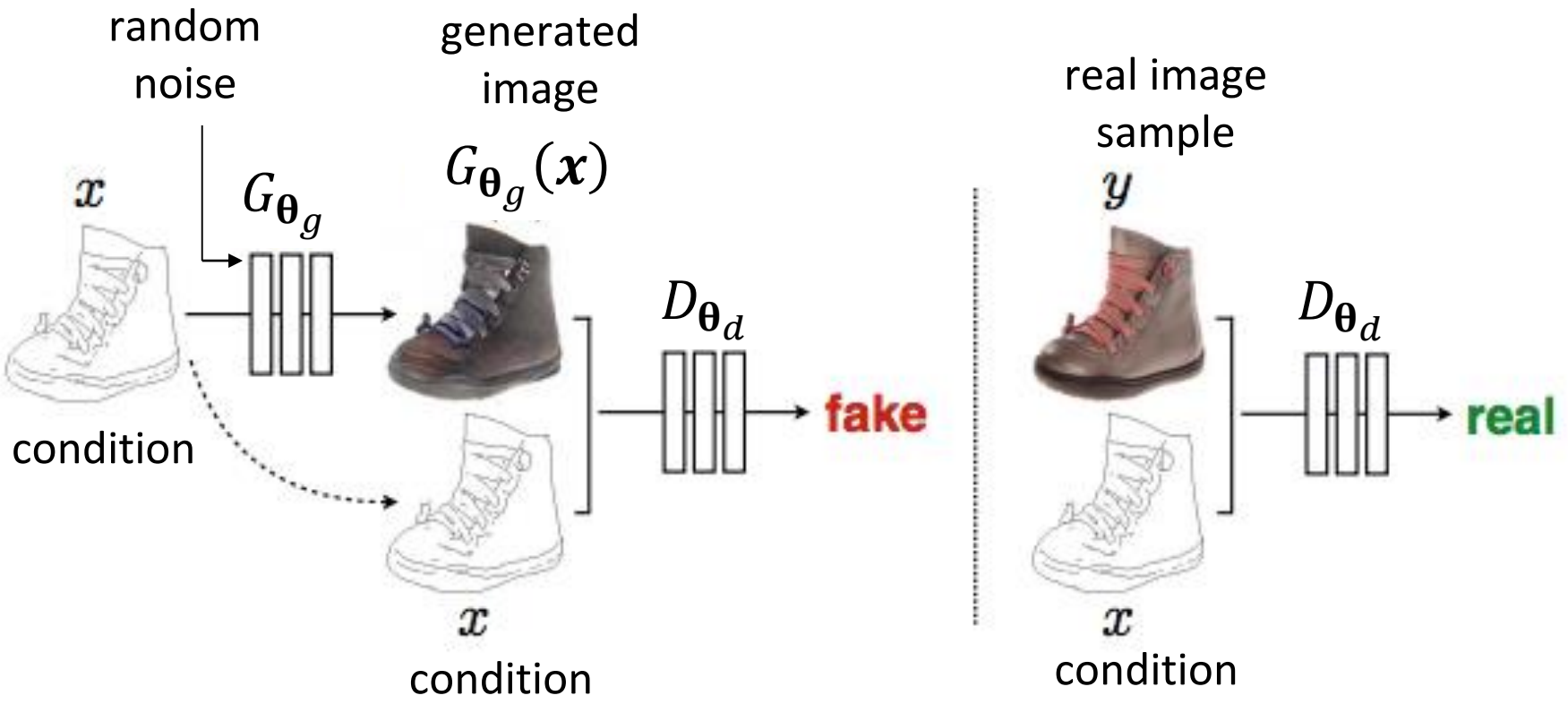


Image-to-Image Demo

Interactive Image Translation with pix2pix-tensorflow

Written by Christopher Hesse — February 19th, 2017

Conditional GANs



Conditional GANs Losses

GAN

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} \Big[\mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} \log \Big(1 - D_{\boldsymbol{\theta}_d} \Big(G_{\boldsymbol{\theta}_g}(\boldsymbol{z}) \Big) \Big) \Big]$$

Conditional GAN

condition

real image

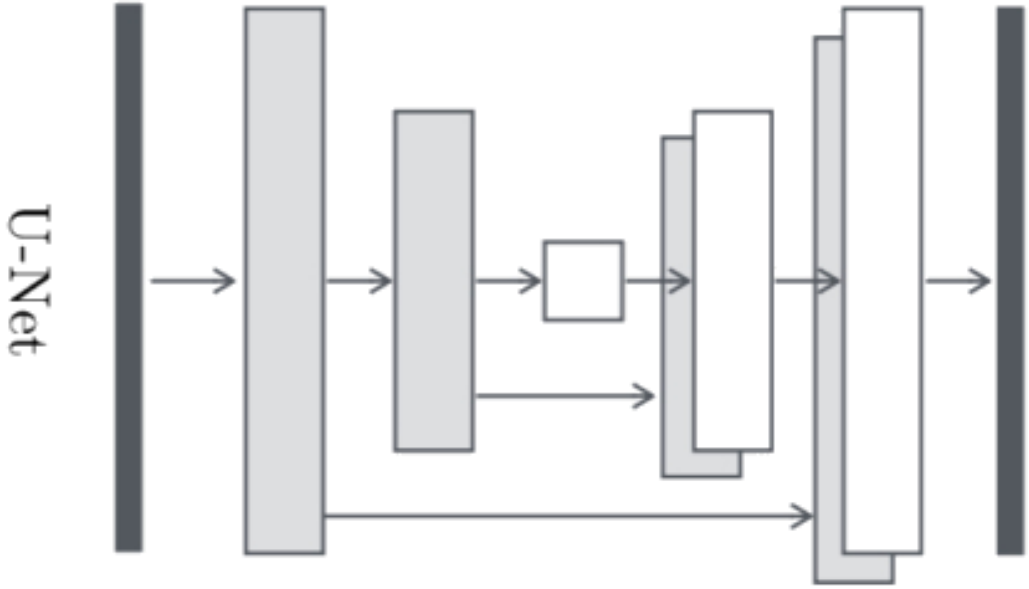
$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} \Big[\mathbb{E}_{\boldsymbol{x},\boldsymbol{y} \sim p_{data}(\boldsymbol{x},\boldsymbol{y})} \log D_{\boldsymbol{\theta}_d}(\boldsymbol{x}, \boldsymbol{y}) + \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x}), \boldsymbol{z} \sim p(\boldsymbol{z})} \log \Big(1 - D_{\boldsymbol{\theta}_d} \Big(\boldsymbol{x}, G_{\boldsymbol{\theta}_g}(\boldsymbol{x}, \boldsymbol{z}) \Big) \Big) \Big]$$

generated image

Conditional GANs Architecture

Example: pix2pix*

Generator



Notice that the output in this case is an **image** and not a label image!

Discriminator

(any) image classification net

* Isola et al., 2018 available at https://arxiv.org/pdf/1611.07004.pdf

Synthesis of Multispectral Optical Images From SAR/Optical Multitemporal Data Using Conditional Generative Adversarial Networks

Jose D. Bermudez, *Student Member, IEEE*, Patrick N. Happ, *Member, IEEE*, Raul Q. Feitosa, *Senior Member, IEEE*, and Dario A. B. Oliveira

***Abstract*— The synthesis of realistic data using deep learning techniques has greatly improved the performance of classifiers in handling incomplete data. Remote sensing applications that have profited from those techniques include translating images of different sensors, improving the image resolution and completing missing temporal or spatial data such as in cloudy optical images. In this context, this letter proposes a new deep-learning-based framework to synthesize missing or corrupted multispectral optical images using multimodal/multitemporal data. Specifically, we use conditional generative adversarial networks (cGANs) to generate the missing optical image by exploiting the correspondent synthetic aperture radar (SAR) data with a SAR-optical data from the same area at a different acquisition date. The proposed framework was evaluated in two land-cover applications over tropical regions, where cloud coverage is a major problem: crop recognition and wildfire detection. In both applications, our proposal was superior to alternative approaches tested in our experiments. In particular, our approach outperformed recent cGAN-based proposals for cloud removal, on average, by 7.7% and 8.6% in terms of overall accuracy and F1-score, respectively.**

***Index Terms*— Conditional generative adversarial networks (cGANs), crop recognition, deep learning, remote sensing, wildfire detection.**

I. Introduction

FULLY exploiting the current wealth of valuable and readily available earth observation data involves dealing with different temporal and spatial resolutions, different methods in the literature to handle this problem rely on different sensors [1] since they allow the observation of a given phenomenon in different wavelengths, which highlight different and complementary features in Earth surface observations.

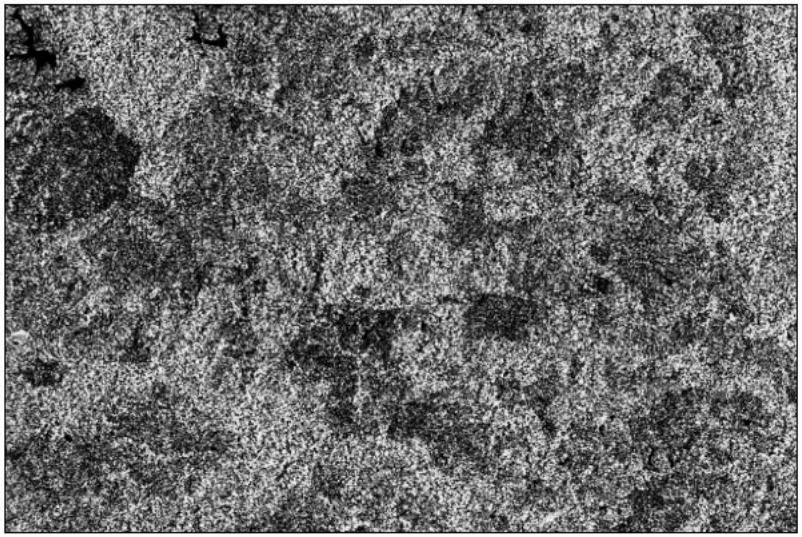
A typical approach consists of using synthetic aperture radar (SAR) images to gather information about the areas affected by clouds in optical images [2]. However, SAR data are usually more difficult to interpret than optical data. Thus, approaches have been recently proposed to exploit SAR images to recover optical data only in areas affected by clouds (see [3]), so that the classification can be performed in the optical domain.

In recent years, generative adversarial networks (GANs) [4] have been used to synthesize missing data or translating data among different domains. More recently, this idea has been exploited in remote sensing. For instance, the method proposed in [5] relies on cyclic-consistent GANs to clean cloudy images. However, the solution is limited to thin clouds. Enomoto *et al.* [6] propose a conditional GAN (cGAN)-based algorithm to recover visible RGB image components to exploit the near-infrared (NIR) data. Again, this method is limited to thin clouds and relies on the NIR, which can only partially penetrate clouds. A later work [7] overcomes part of these shortcomings by exploiting SAR instead of NIR data, but this method is also restricted to thin clouds.

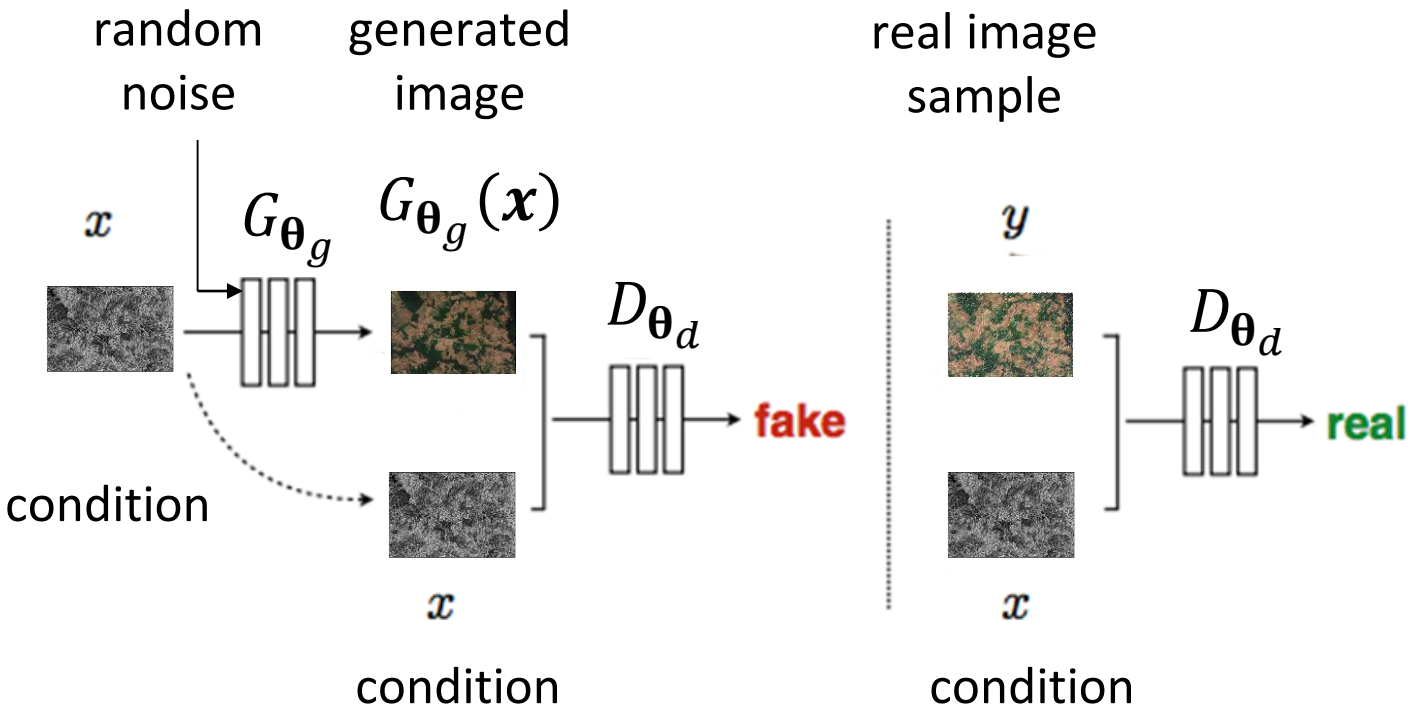
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8637007

cGANs Application: SAR to Optical

SAR data are less descriptive and difficult to interpret, but are nearly independent on weather conditions.



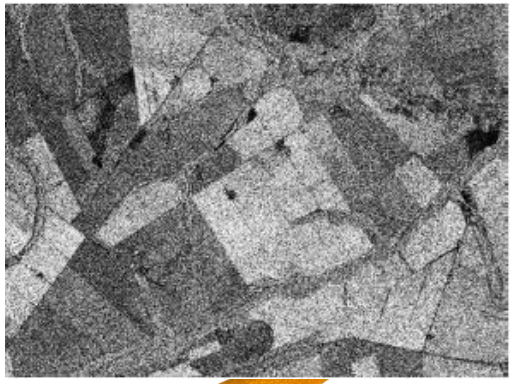
cGANs Application: SAR to Optical



cGANs Application: SAR to Optical

SAR data are less descriptive and difficult to interpret, but are nearly independent on weather conditions.

SAR original



GAN

optical original

optical synthetic

cGANs Application: Skin Lesion Synthesis

2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)
April 3-7, 2020, Iowa City, Iowa, USA

CONTROLLABLE SKIN LESION SYNTHESIS USING TEXTURE PATCHES, BÉZIER CURVES AND CONDITIONAL GANS

Dario Augusto Borges Oliveira

IBM Research
Rua Tutoia 1157, Vila Mariana, Sao Paulo, Brazil

ABSTRACT

Data synthesis is an important tool for improving data availability in cases where data is hard to capture or annotate. In the context of skin lesions data, data synthesis has been used for data augmentation in automated classification methods or for supporting training of dermoscopic images visual inspection. In this paper, we propose a simple yet effective approach for diverse skin lesion image synthesis using conditional generative adversarial networks. Our pipeline takes as input a random Bézier curve representing the lesion mask, and two texture patches: one for skin, and one for lesion; and synthesizes a new dermoscopic image. Our method generates images where lesions and skin reproduce the corresponding provided texture patches, and the lesion conforms to the provided Bézier mask. Our results report realistic controllable synthesis and improved performance for skin lesion segmentation task considering different semantic segmentation networks in a public challenge in comparison to classic data augmentation.

and clinically-meaningful synthetic dermoscopic images. None of these works, however, create means for controlling the synthesis process and increase the variability of shapes and textures of skin lesions in datasets.

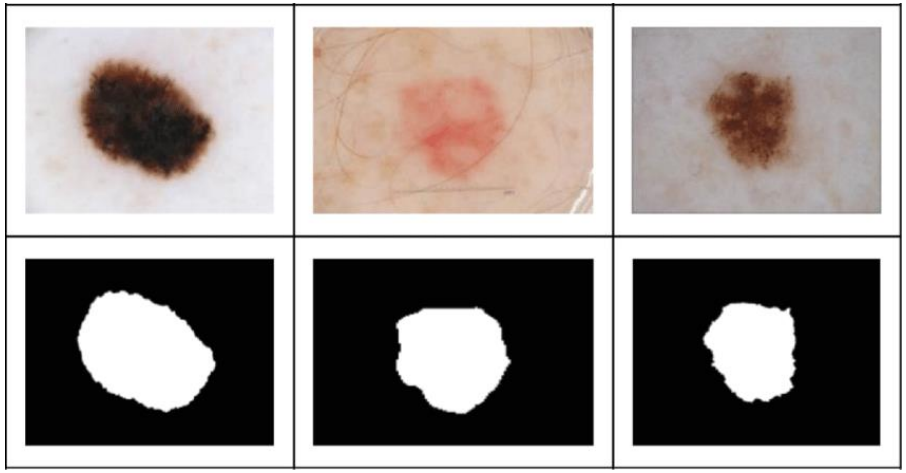
In this paper, we propose to use conditional GANs for diverse skin lesion image synthesis using controllable Bézier curves, and two texture patches - one for skin, and one for lesion. Our results show that the model is able to generate realistic images from new defined curves respecting the input patches textures. We also provide an ablation study that reports consistent improved results using our proposal for augmenting training data of three different semantic segmentation networks in the ISIC challenge segmentation task. To the best of our knowledge, this is the first paper to combine controllable curves and texture patches to synthesize realistic skin lesion images.

2. METHOD

cGANs Application: Skin Lesion Synthesis

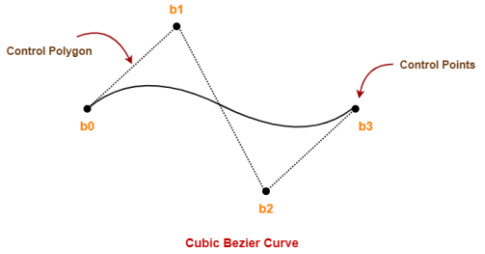
Data synthesis for augmenting training set for skin cancer lesion segmentation

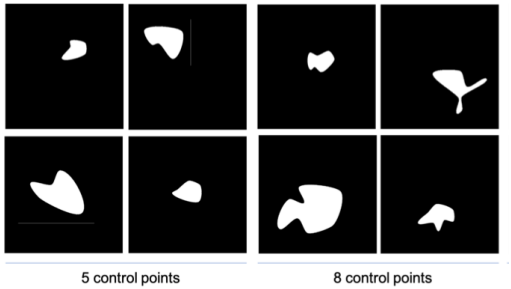
Skin lesion segmentation examples



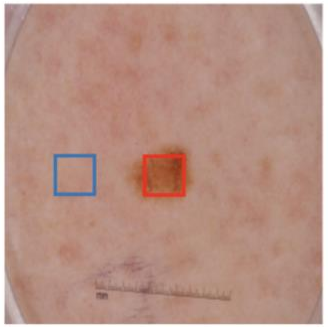
How to increase diversity in shape and appearance?

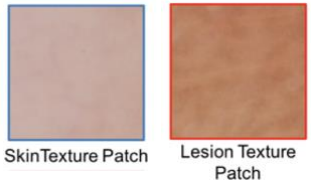
Bézie Curve Generator





Texture Patches Dataset

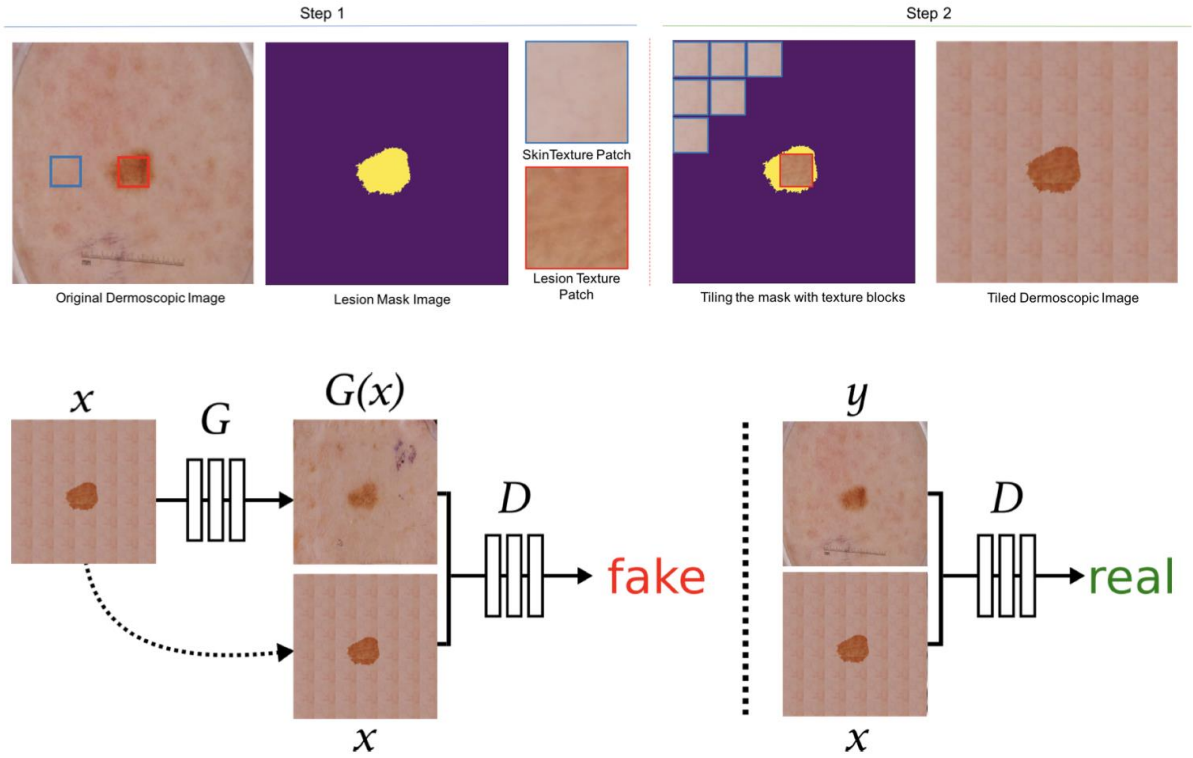




"CONTROLLABLE SKIN LESION SYNTHESIS USING TEXTURE PATCHES, BEZIER CURVES AND CONDITIONAL GANS", Oliveira, IEEE ISBI 2020

cGANs Application: Skin Lesion Synthesis

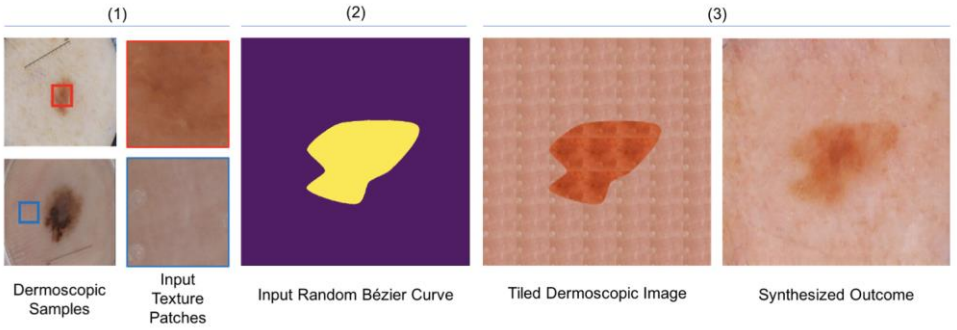
1) Train data synthesis given a closed shape an texture patches inside and outside

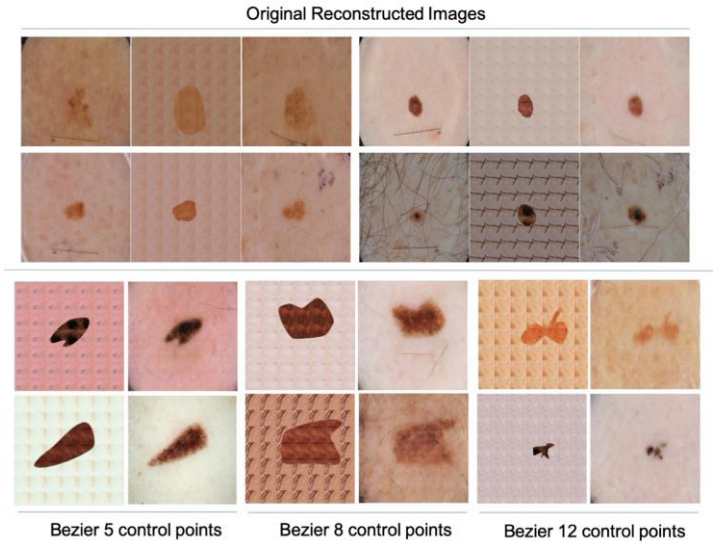


"CONTROLLABLE SKIN LESION SYNTHESIS USING TEXTURE PATCHES, BEZIER CURVES AND CONDITIONAL GANS", Oliveira, IEEE ISBI 2020

cGANs Application: Skin Lesion Synthesis

2) Create (augment) training data given a random shape and random texture patches





"CONTROLLABLE SKIN LESION SYNTHESIS USING TEXTURE PATCHES, BEZIER CURVES AND CONDITIONAL GANS", Oliveira, IEEE ISBI 2020

cGANs Application: Skin Lesion Synthesis

3) Evaluate segmentation networks using different augmentation strategies

Table 1. Results for segmentation of lesions based on semantic segmentation networks trained with different training set configurations.

Experiment	Unet	PSPNet	Deeplab
No Augmentation	84.54%	85.93%	85.54%
Rot.-Flip-Scale	84.63%	85.99%	85.43%
5-Bezier	84.55%	86.89%	86.31%
8-Bezier	84.57%	86.51%	87.65%
12-Bezier	84.77%	86.86%	87.37%
5-8-12 Bezier	**85.55%**	**87.16%**	**88.33%**

"CONTROLLABLE SKIN LESION SYNTHESIS USING TEXTURE PATCHES, BEZIER CURVES AND CONDITIONAL GANS", Oliveira, IEEE ISBI 2020

Further cGAN Application examples

Image-to-image translation

Zhu et al., 2018 , Towards Multimodal Image-to-Image Translation

Pixel2Style2Pixel

Encoding in Style: A StyleGAN Encoder for Image-to-Image Translation

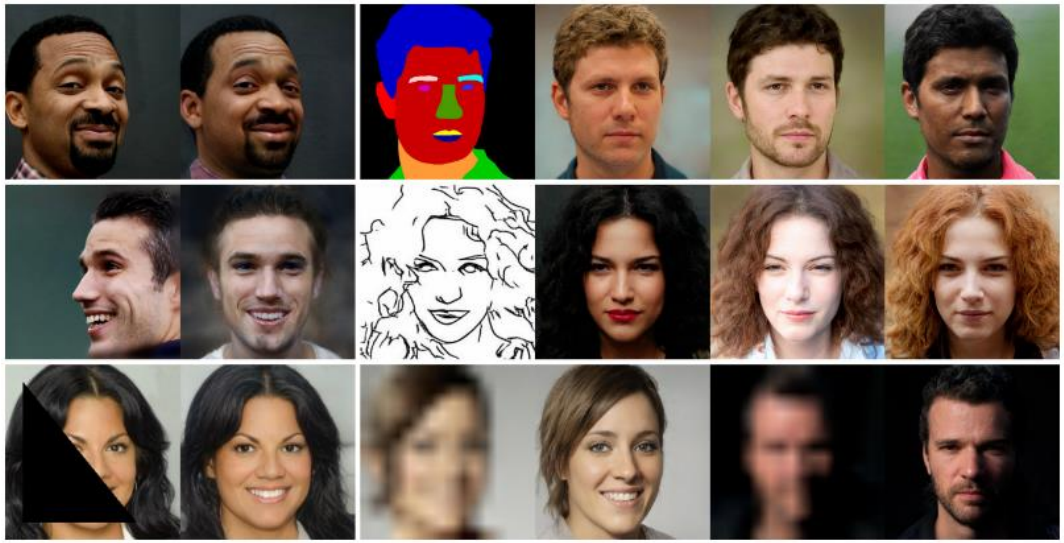
Elad Richardson[1] Yuval Alaluf[1,2] Or Patashnik[1,2] Yotam Nitzan[2]

Yaniv Azar[1] Stav Shapiro[1] Daniel Cohen-Or[2]

Penta-AI[1] Tel-Aviv University[2]

Conference on Computer Vision and Pattern Recognition, 2021

Paper Code Colab Poster



Richardson et. al., 2021, Encoding in Style: A StyleGAN Encoder for Image-to-Image Translation

Overview

Motivation

Introduction to GANs

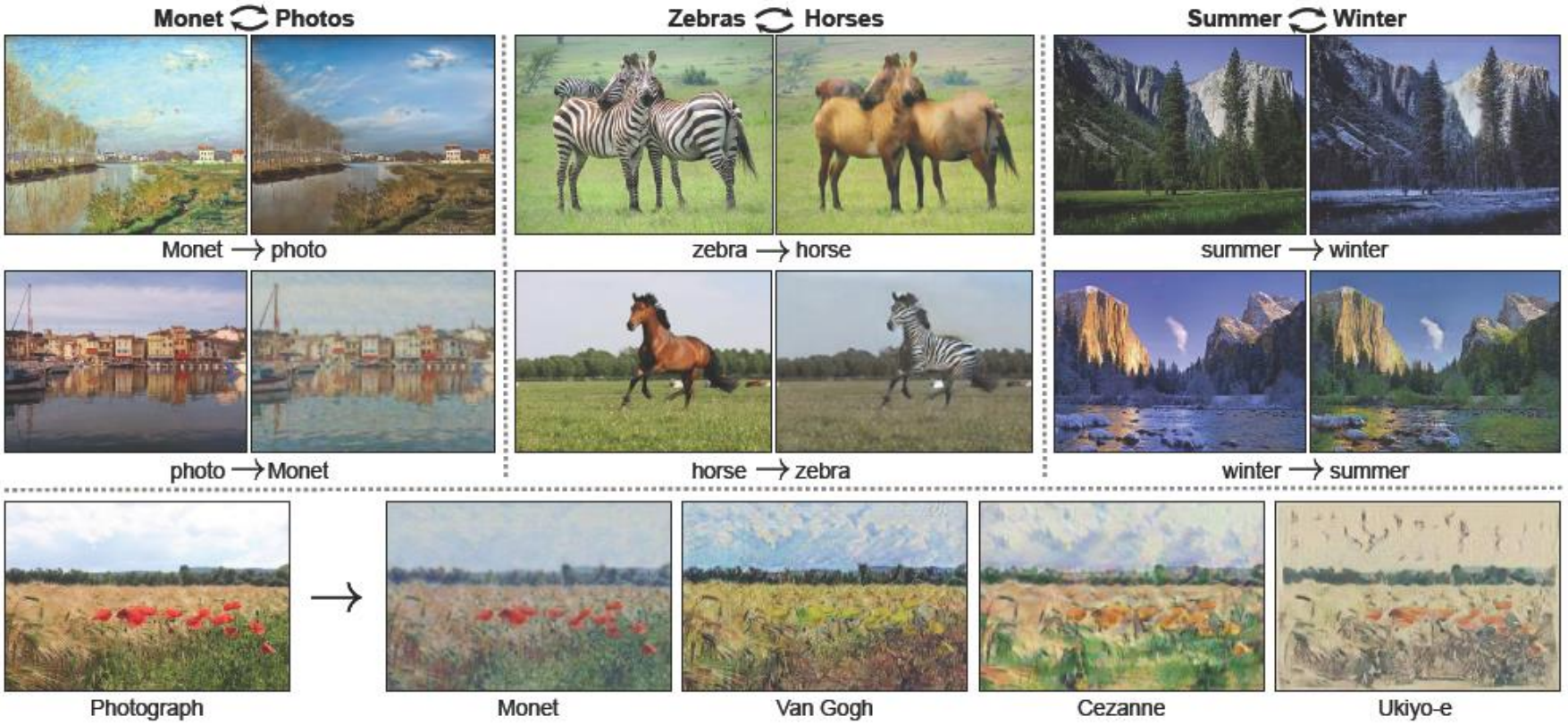
Conditional GANs

Cycle GANs

Other Applications

Cycle GANs

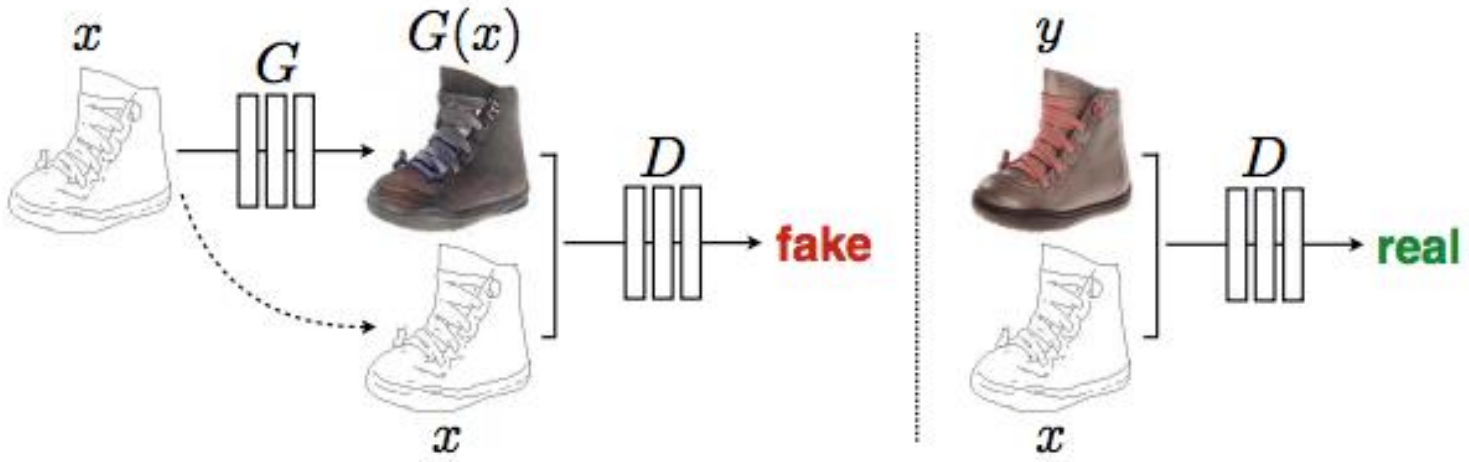
Learn how to translate image from a source domain X to a target domain Y.



Source Zhu et al., 2018 available at https://arxiv.org/pdf/1703.10593.pdf

Conditional GAN vs Cycle GAN

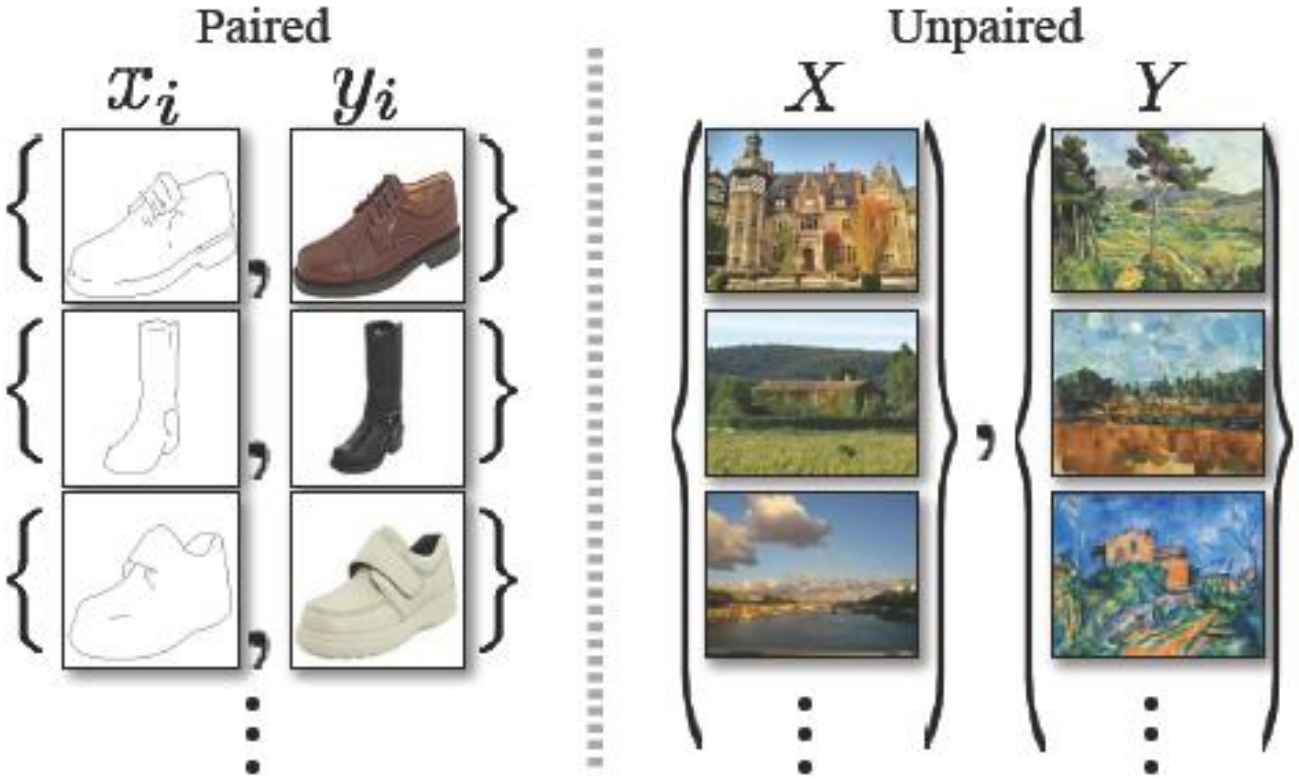
Conditional GANs are trained with **paired samples**:



Source Zhu et al., 2018 available at https://arxiv.org/pdf/1703.10593.pdf

Conditional GAN vs Cycle GAN

Often, there are not enough paired samples



Source Zhu et al., 2018 available at https://arxiv.org/pdf/1703.10593.pdf

Intuition behind Cycle GANs

- No paired sample $\rightarrow$ CGAN does not apply.
- What about regular DCGANs?

We can train a DCGAN to generate horses

$$G: NOISE \rightarrow HORSES$$

and another DCGAN to generate zebras

$$F: NOISE \rightarrow ZEBRAS$$

but correspondence is not guaranteed

$$horse \leftrightarrow zebras$$

Adversarial Loss

For the mapping function $G: X \rightarrow Y$:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{\boldsymbol{y} \sim p_{datay}(\boldsymbol{y})} \log D_Y(\boldsymbol{y}) + \mathbb{E}_{\boldsymbol{x} \sim p_{datax}(\boldsymbol{x})} \log\left(1 - D_Y\left(G(\boldsymbol{x})\right)\right)$$

For the mapping function $F: Y \rightarrow X$:

$$\mathcal{L}_{GAN}(F, D_X, Y, X) = \mathbb{E}_{\boldsymbol{x} \sim p_{datax}(\boldsymbol{x})} \log D_X(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{y} \sim p_{datay}(\boldsymbol{y})} \log\left(1 - D_X\left(F(\boldsymbol{y})\right)\right)$$

"Adversarial training can, in theory, learn mappings G and F that produce outputs identically distributed as target domains Y and X respectively".

But the adversarial loss alone cannot guarantee that the learned functions map "corresponding" samples in both domains.

Source Zhu et al., 2018 available at https://arxiv.org/pdf/1703.10593.pdf

Intuition behind Cycle GANs

- Train $G: HORSE \rightarrow ZEBRA$ such that

  $G(horse)$ is indistinguishable from $zebra$

- Train $F: ZEBRA \rightarrow HORSE$ such that

  $F(zebra)$ is indistinguishable from $horse$

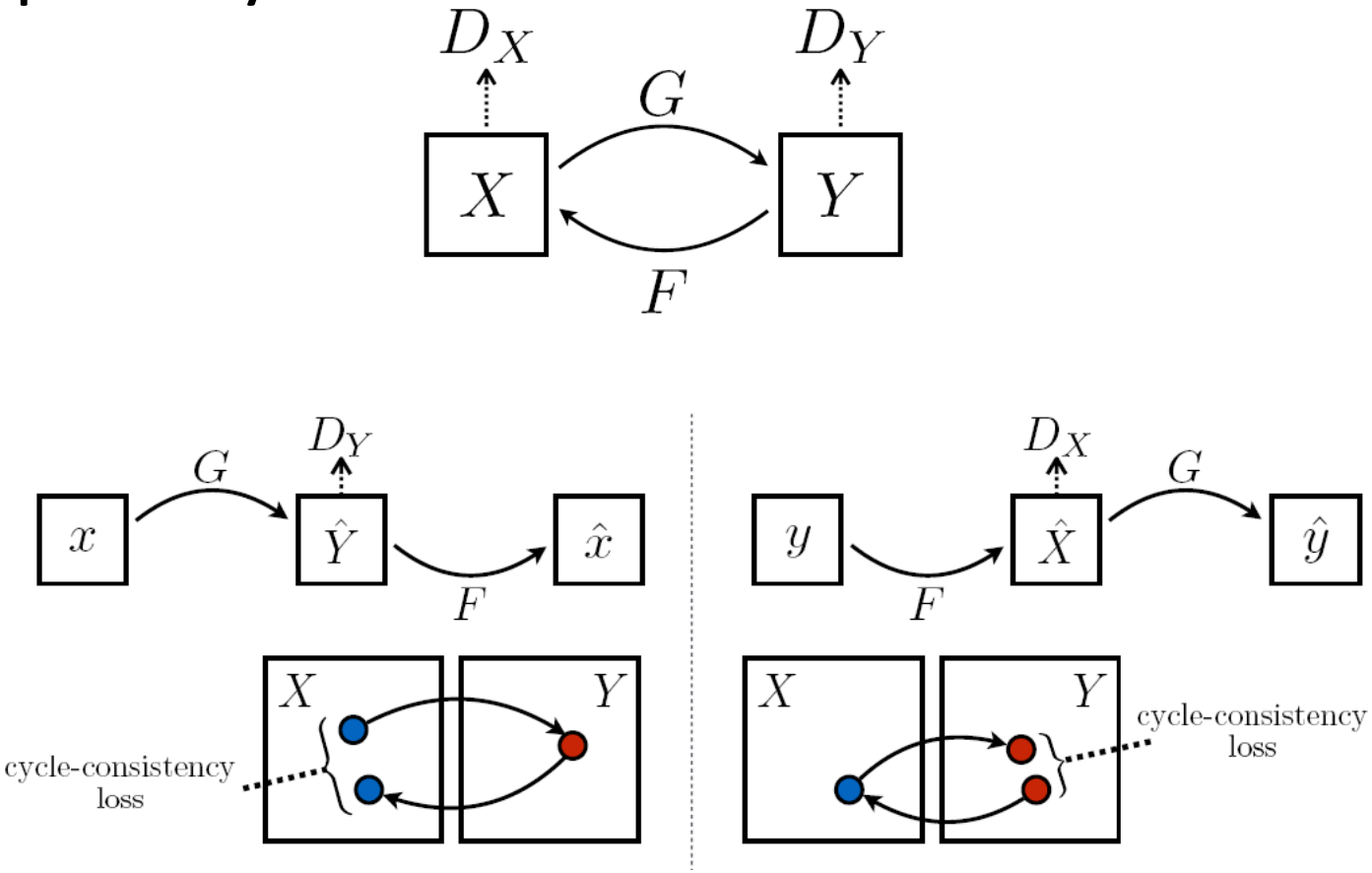
- Enforce consistency
  - Given $G: HORSE \rightarrow ZEBRA$ and $F: ZEBRA \rightarrow HORSE$
  - Then $F\big(G(horse)\big) \approx horse$ and

    $G\big(F(zebra)\big) \approx zebra$

cycle consistency

Cycle Loss Formulation

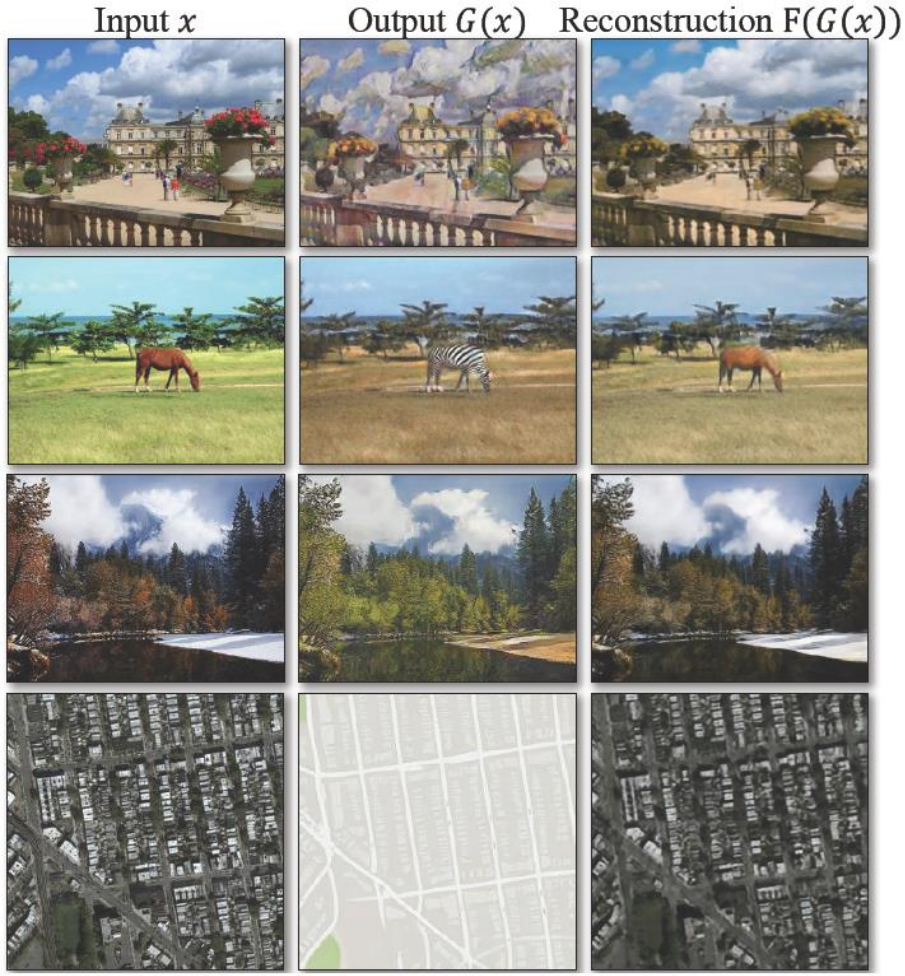
Graphically…



Source Zhu et al., 2018 available at https://arxiv.org/pdf/1703.10593.pdf

Consistency Loss

We have to enforce that for each image $\boldsymbol{x}$ of domain X, the function F is able to bring the image $G(\boldsymbol{x})$ back to the original image, i.e., $F\big(G(\boldsymbol{x})\big) \approx \boldsymbol{x}$ and vice-versa.



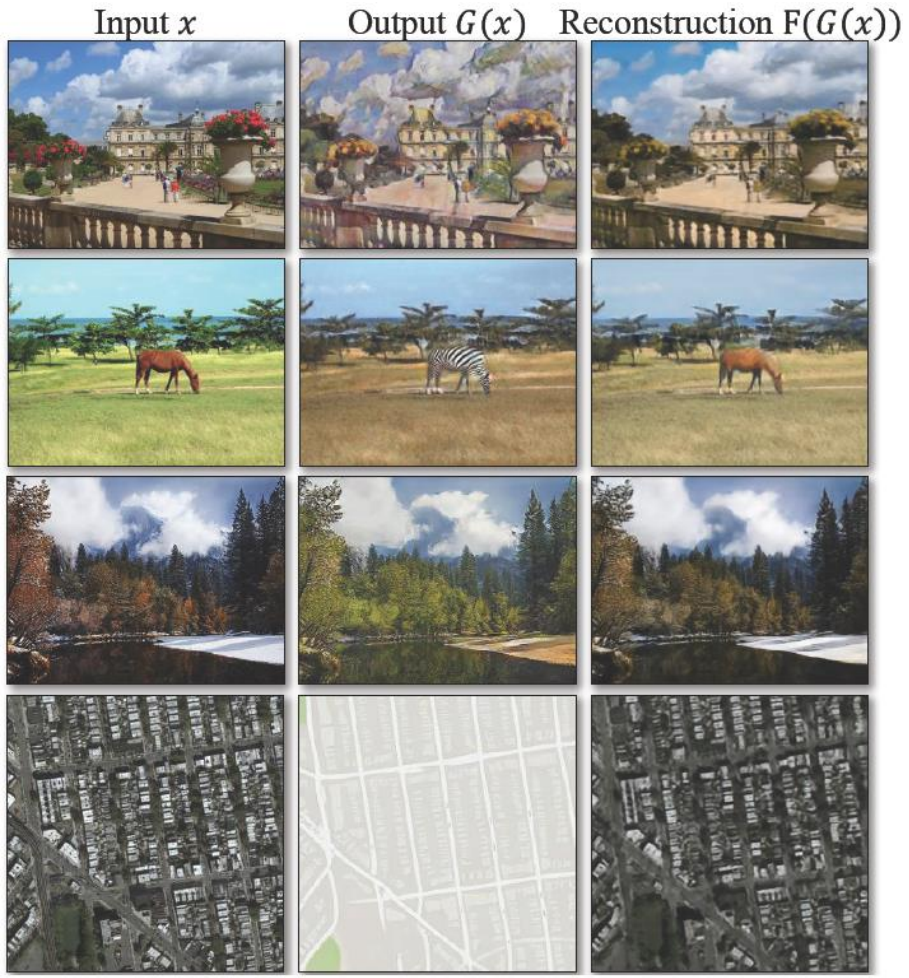
$$\mathcal{L}_{cyc}(G,F) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}x(\boldsymbol{x})}\left[\left\|F\big(G(\boldsymbol{x})\big) - \boldsymbol{x}\right\|_1\right] + \mathbb{E}_{\boldsymbol{y}\sim p_{data}y(\boldsymbol{y})}\left[\left\|G\big(F(\boldsymbol{y})\big) - \boldsymbol{y}\right\|_1\right]$$

Source Zhu et al., 2018 available at https://arxiv.org/pdf/1703.10593.pdf

Identity Loss

To encourage preservation of source domain's structures in the transformed image an identity-mapping loss is added

Identity loss says that, if you feed image $\boldsymbol{y}$ to generator G, it should yield something close to image $\boldsymbol{y}$



$$\mathcal{L}_{iden}(G,F) = \mathbb{E}_{\boldsymbol{x}\sim p_{datax}(\boldsymbol{x})}[\|G(\boldsymbol{y}) - \boldsymbol{y}\|_1] + \mathbb{E}_{\boldsymbol{y}\sim p_{datay}(\boldsymbol{y})}[\|F(\boldsymbol{x}) - \boldsymbol{x}\|_1]$$

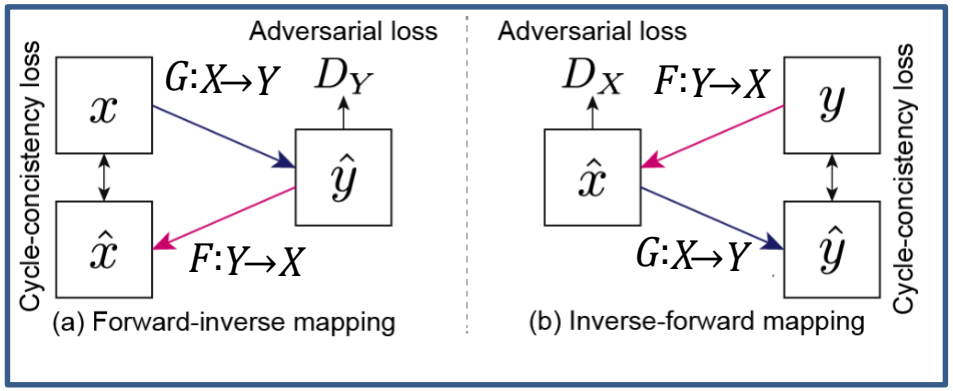
Source Zhu et al., 2018 available at https://arxiv.org/pdf/1703.10593.pdf

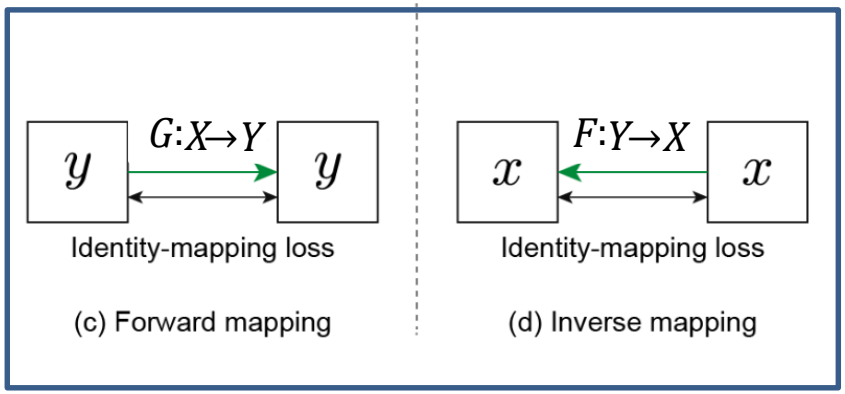
Full Loss

The full loss function takes the form:

$$\mathcal{L}(G, F, D_X, D_Y) = \lambda_{GAN}[\mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X)]$$

$$+\lambda_{cyc}\mathcal{L}_{cyc}(G, F) \qquad +\lambda_{iden}(G, F)$$





where λ_{GAN} , λ_{cyc} and λ_{iden} controls the relative importance of the loss terms.

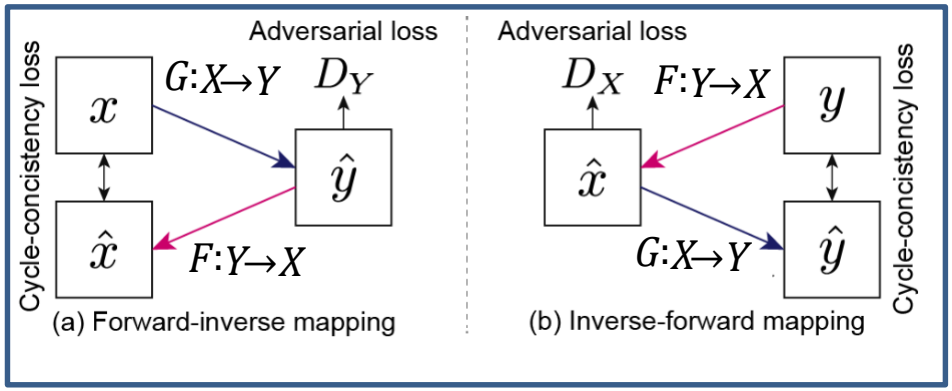
Source Zhu et al., 2018 available at https://arxiv.org/pdf/1703.10593.pdf

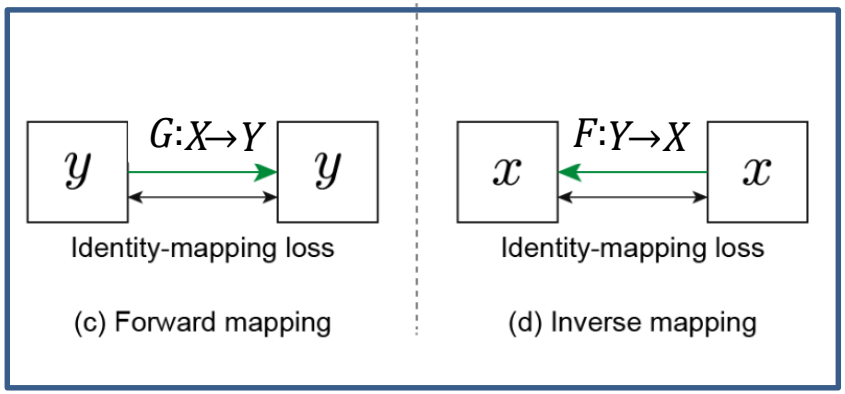
Full Loss

The full loss function takes the form:

$$\mathcal{L}(G, F, D_X, D_Y) = \lambda_{GAN}[\mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X)]$$

$$+\lambda_{cyc}\mathcal{L}_{cyc}(G, F) \qquad +\lambda_{iden}(G, F)$$

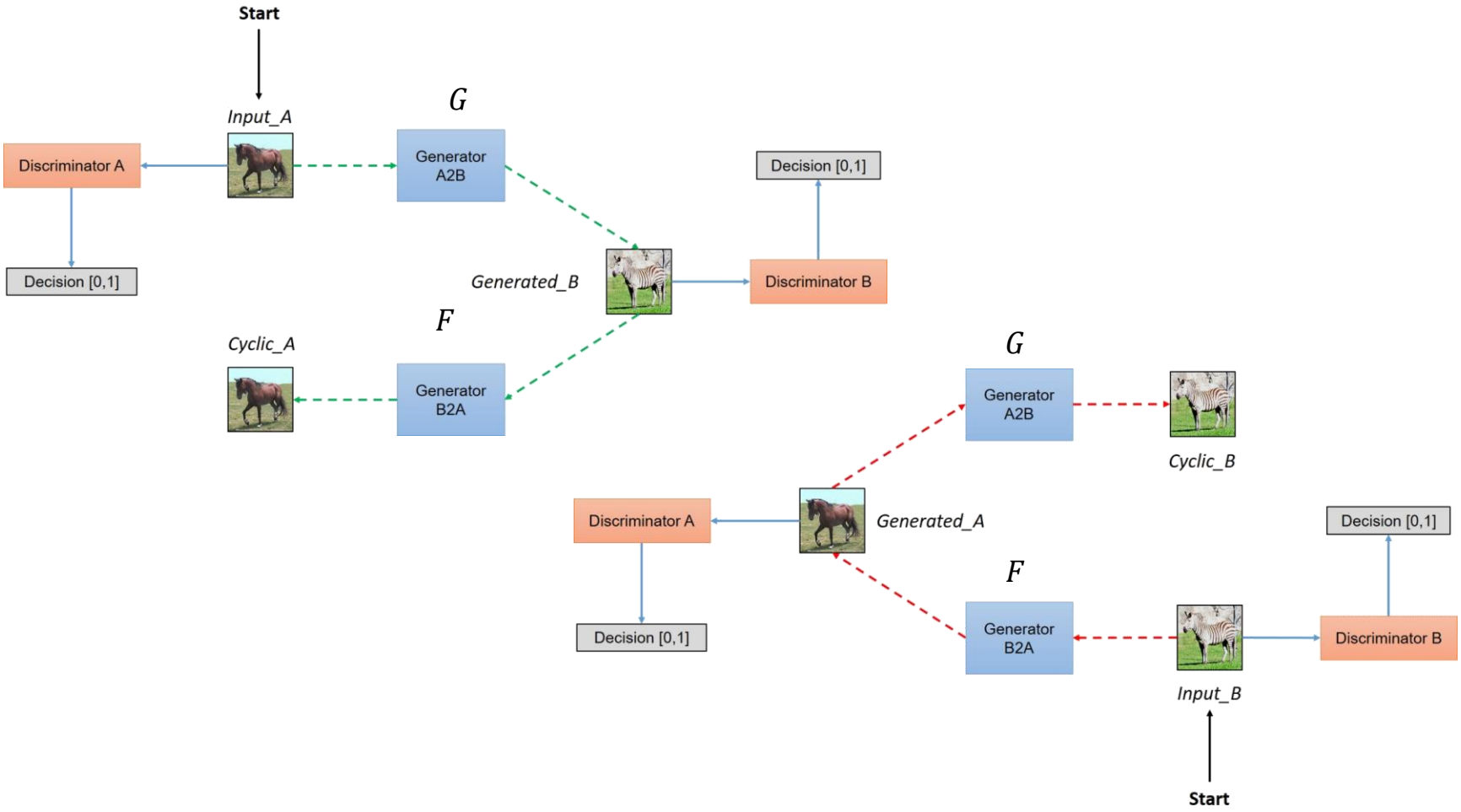




$$G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y)$$

Source Zhu et al., 2018 available at https://arxiv.org/pdf/1703.10593.pdf

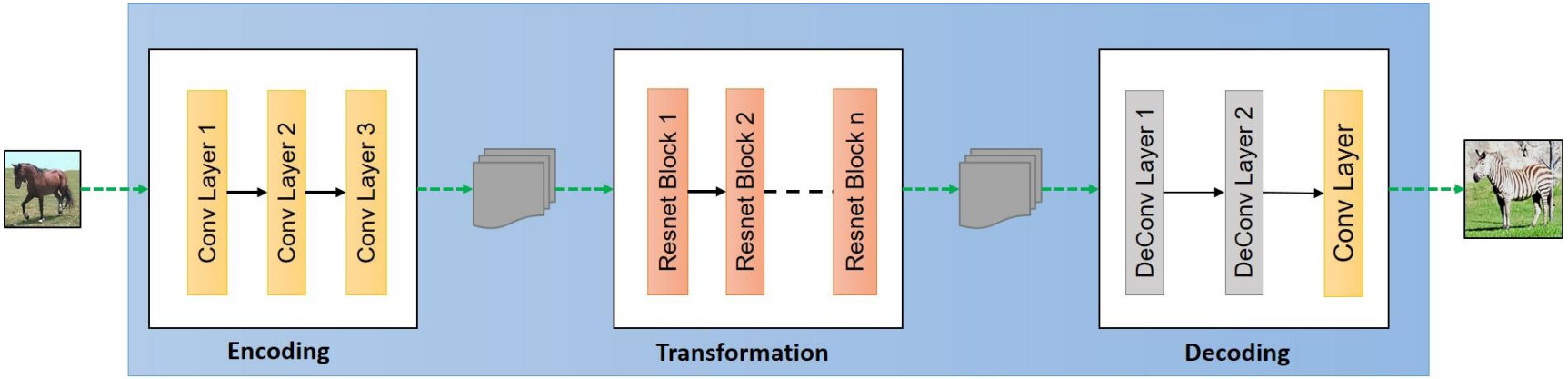
Cycle GAN Network Architecture



Source Bansal & Tathore available at https://hardikbansal.github.io/CycleGANBlog/

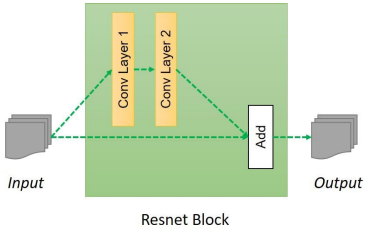
Generator Architecture

High level structure



ins norm ReLu conv2

1) 7×7×32 stride 1
2) 3×3×64 stride 2
3) 3×3×128 stride 2



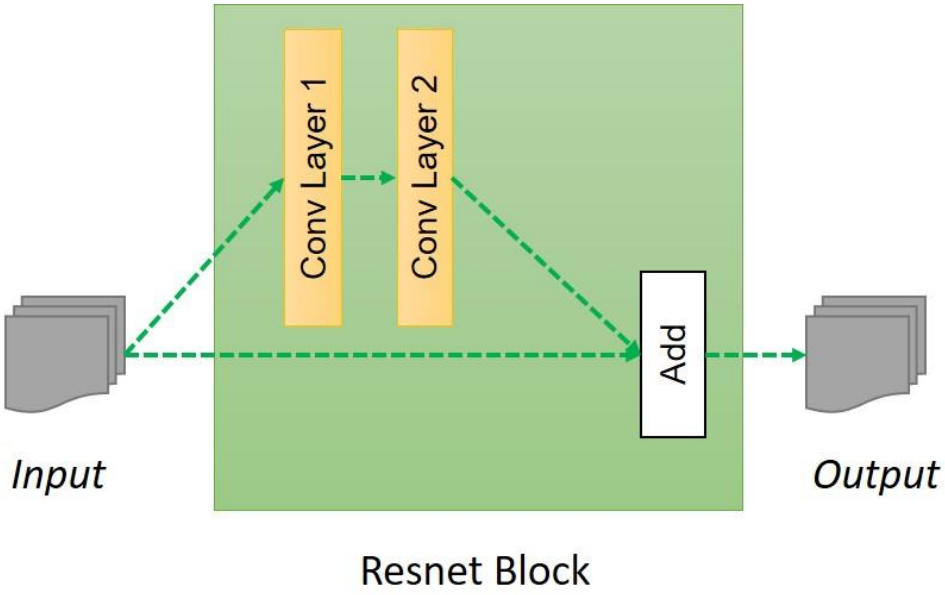
ins norm ReLu conv2

1) 7×7×64 stride 1/2
2) 3×3×32 stride 1/2
3) 7×7×3 stride 2

Source Bansal & Tathore available at https://hardikbansal.github.io/CycleGANBlog/

Generator Architecture

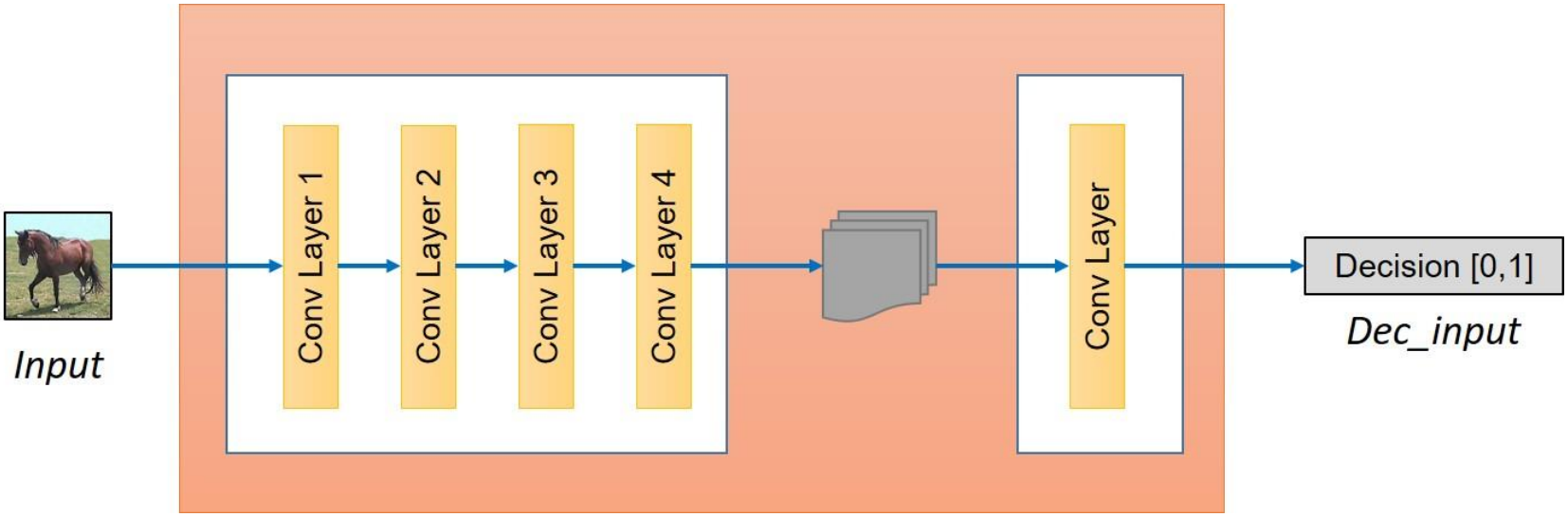
Transformation



Source Bansal & Tathore available at https://hardikbansal.github.io/CycleGANBlog/

Generator Architecture

Discriminator



ins norm ReLu conv2

1) 4×4×64 stride 2
2) 4×4×128 stride 2
3) 4×4×256 stride 2
4) 4×4×512 stride 2

Source Bansal & Tathore available at https://hardikbansal.github.io/CycleGANBlog/

Cycle GAN Application Examples

Animal Transfiguration

Source: Zhu et al., 2017 available at https://junyanz.github.io/CycleGAN/

Cycle GAN Application Examples

Cats ↔ Dogs

Source: Zhu et al., 2017 available at https://junyanz.github.io/CycleGAN/

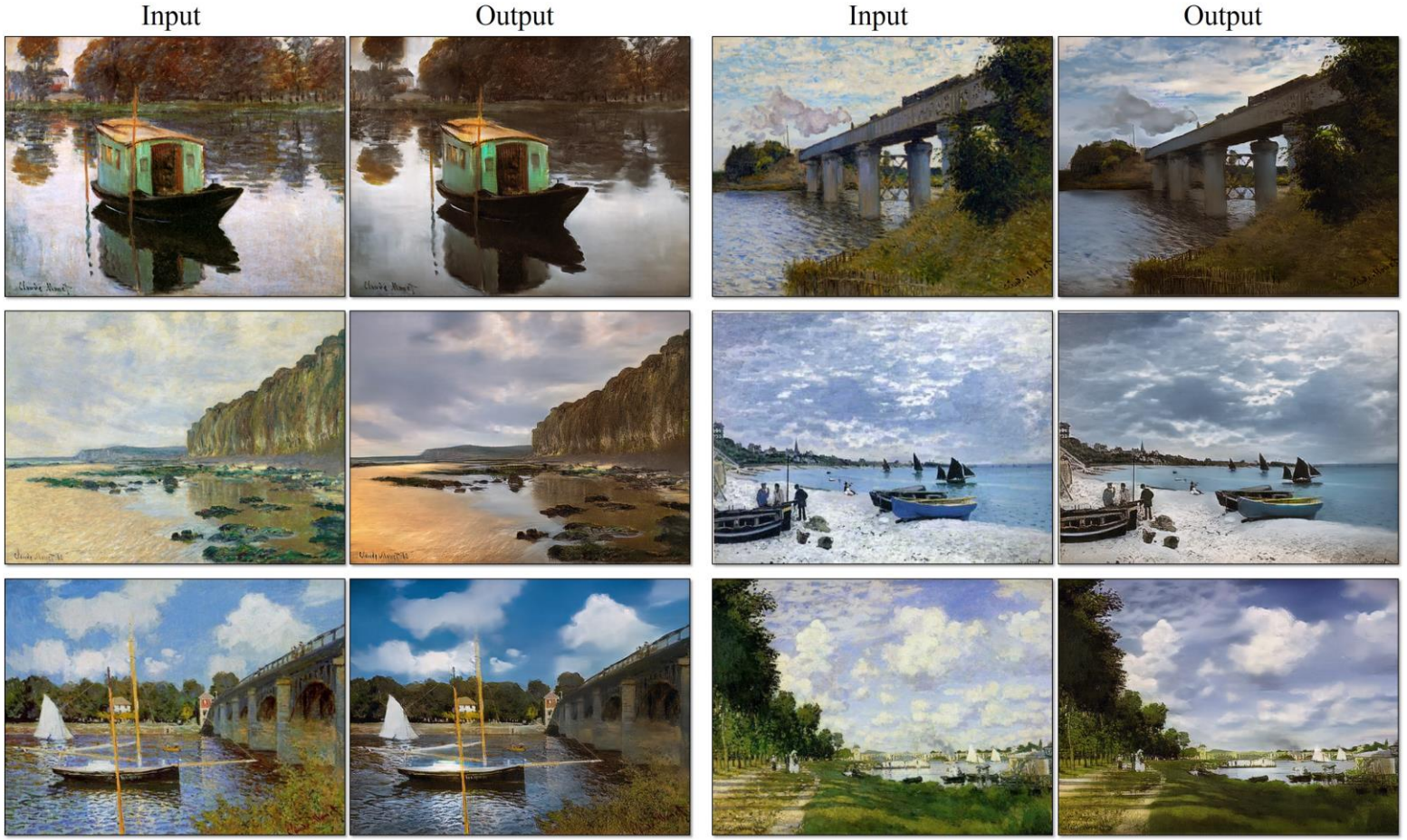
Cycle GAN Application Examples

Horse ↔ Zebra

Source: https://www.youtube.com/watch?v=9reHvktowLY

Cycle GAN Application Examples

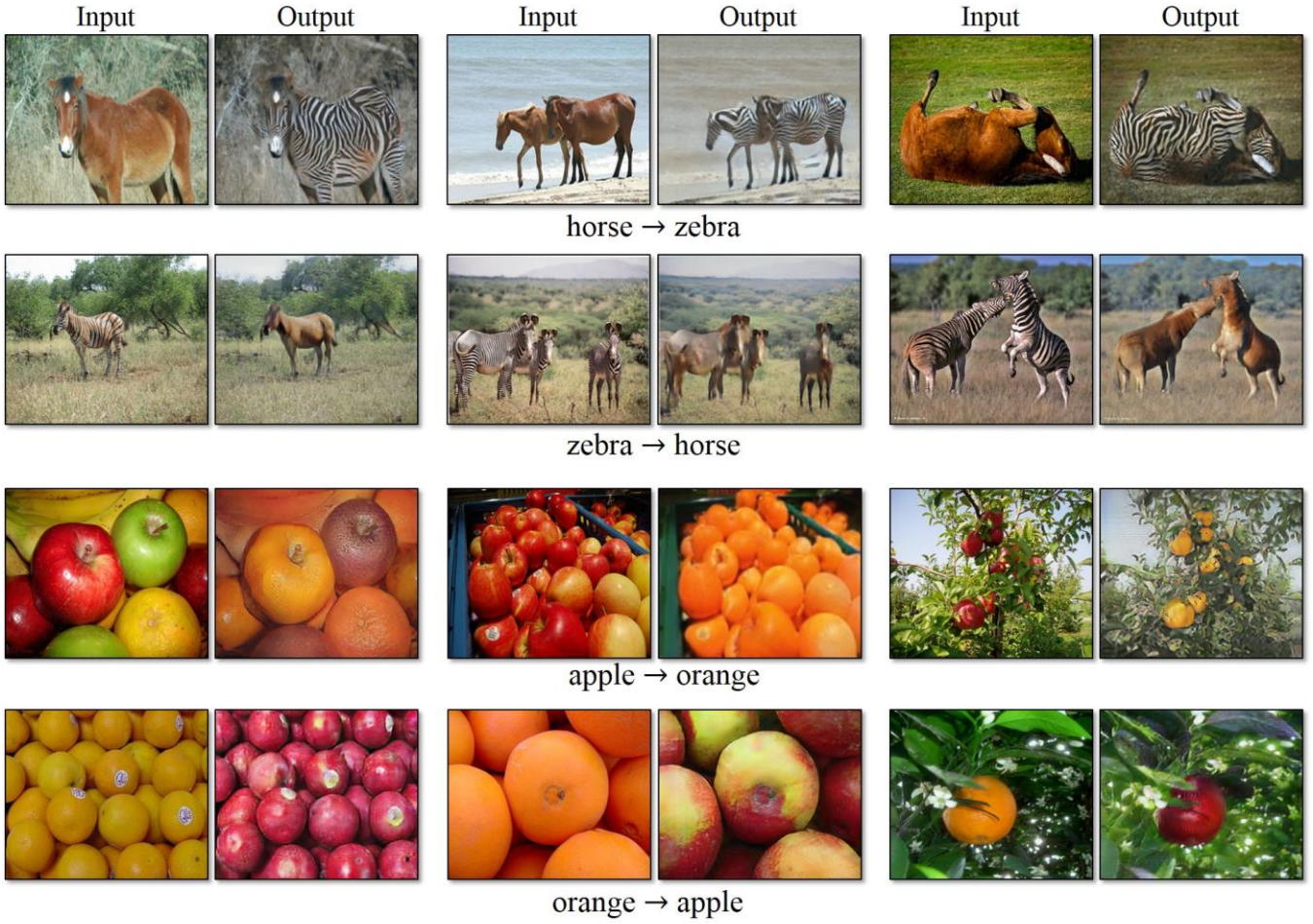
Monet paintings → photographs



Source: Zhu et al., 2017 available at https://junyanz.github.io/CycleGAN/

Cycle GAN Application Examples

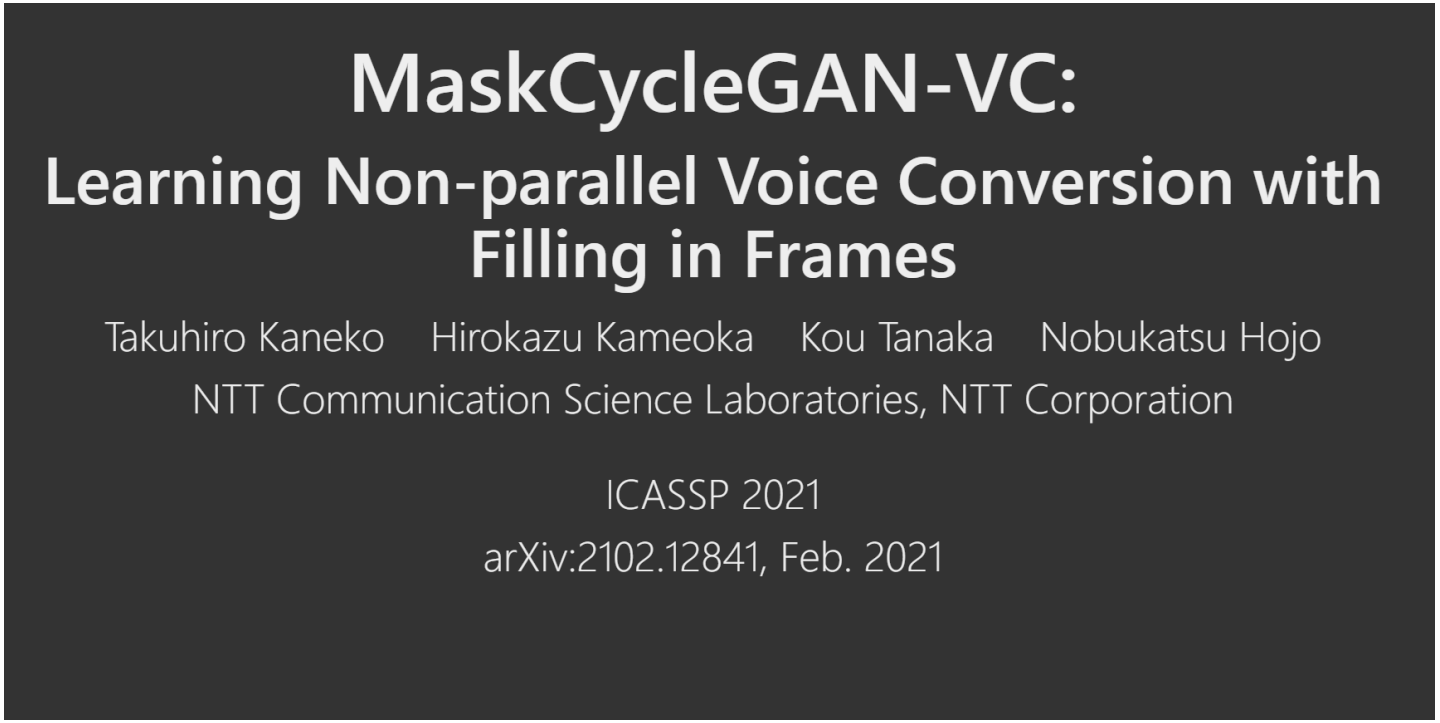
Object Transfiguration



Source: Zhu et al., 2017 available at https://junyanz.github.io/CycleGAN/

Cycle GAN Application Examples

Voice Conversion



http://www.kecl.ntt.co.jp/people/kaneko.takuhiro/projects/maskcyclegan-vc/index.html#F2M

Source: Kaneko & Kameoka, 2018 available at http://www.kecl.ntt.co.jp/people/kaneko.takuhiro/projects/cyclegan-vc/

Overview

Motivation

Introduction to GANs

Conditional GANs

Cycle GANs

Other Applications

Other Applications

Image captioning

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

Source: Radhakrishnan, 2017 available at https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2

Other Applications

Text to image synthesis

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.

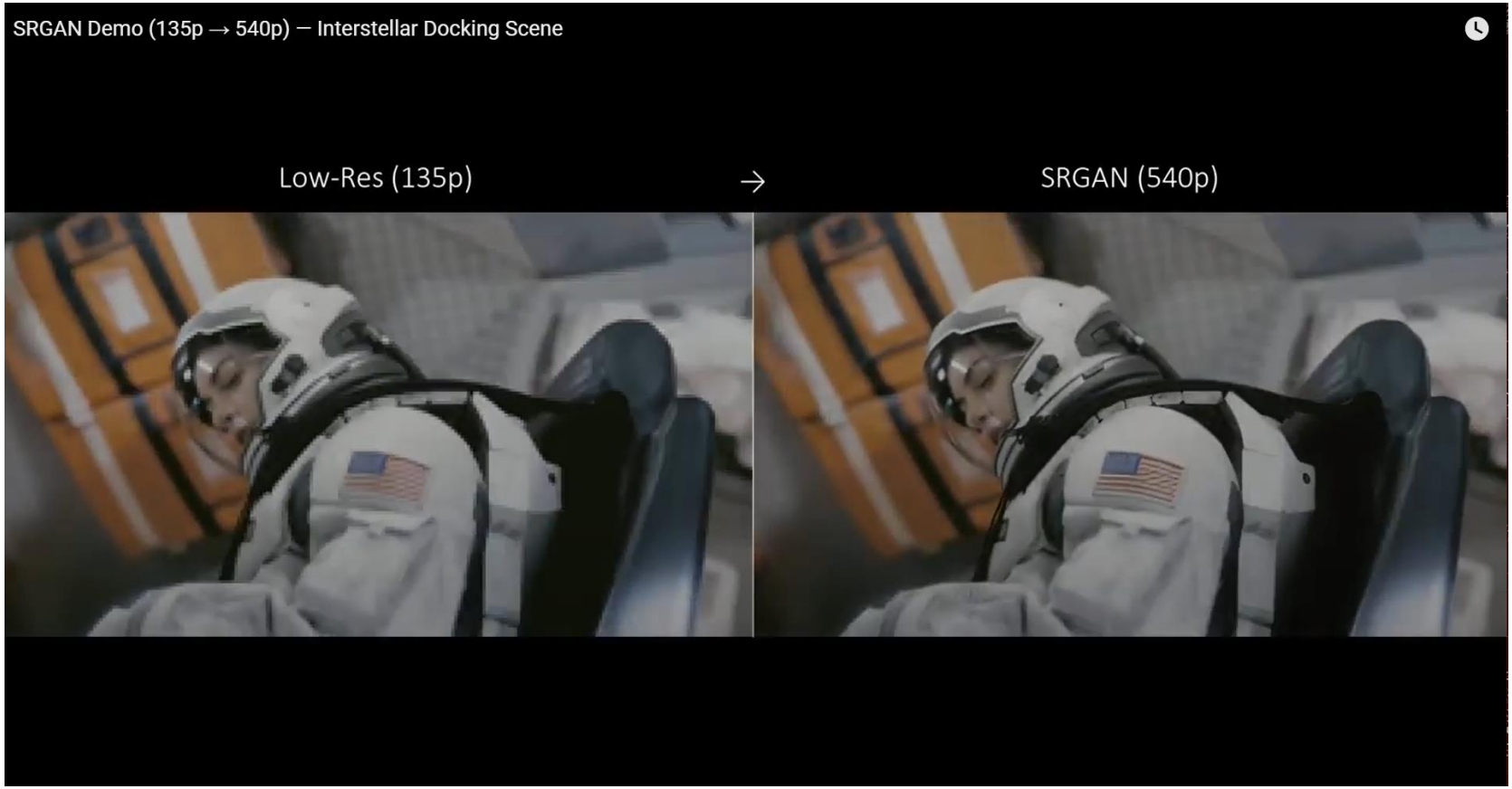
the flower has petals that are bright pinkish purple with white stigma

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

[Reed et al., 2015, Generative Adversarial Text to Image Synthesis]

Other Applications

Super resolution GAN (SRGAN)[1]



Click here to watch the video demo

[1] Ledig et al., CVPR 2017, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8099502

Other Applications

The GAN Zoo: lots of GAN applications

Every week, new GAN papers are coming out and it's hard to keep track of them all, not to mention the incredibly creative ways in which researchers are naming these GANs! So, here's a list of what started as a fun activity compiling all named GANs!

Click here to access The Gan Zoo

Recent Publication on GANs

Aggarwal et al., 2021, Generative adversarial network: An overview of theory and applications

Next Lecture

Lab

Generative Adversarial Networks

See you next class!