Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV), 2017.

Presented by Shyamal H. Anadkat

AIPI 540, CV Module

[Link to paper]

Outline

1. What is **Image-to-Image Translation**?
2. What are **GANs***?
3. What are **CycleGANs**?
4. How does it work?
5. How did they evaluate it?
6. Do they work all the time?
7. Live demo
8. Q/A

Assumption: you know a bit about CNNs.*

Image-to-Image Translation

- Class of computer vision problems where goal is to “learn” the mapping between an input image (X) and an output image (Y)

- Wide range of **applications**, such as collection style transfer, object transfiguration, season transfer and photo enhancement.

What are GANs?

General Adversarial Networks

- NN architecture introduced by Ian Goodfellow et al., in June '14, in their paper Generative Adversarial Nets
- Generator Network: Produces fake data from random inputs
- Discriminator Network: Separates true data from fake data

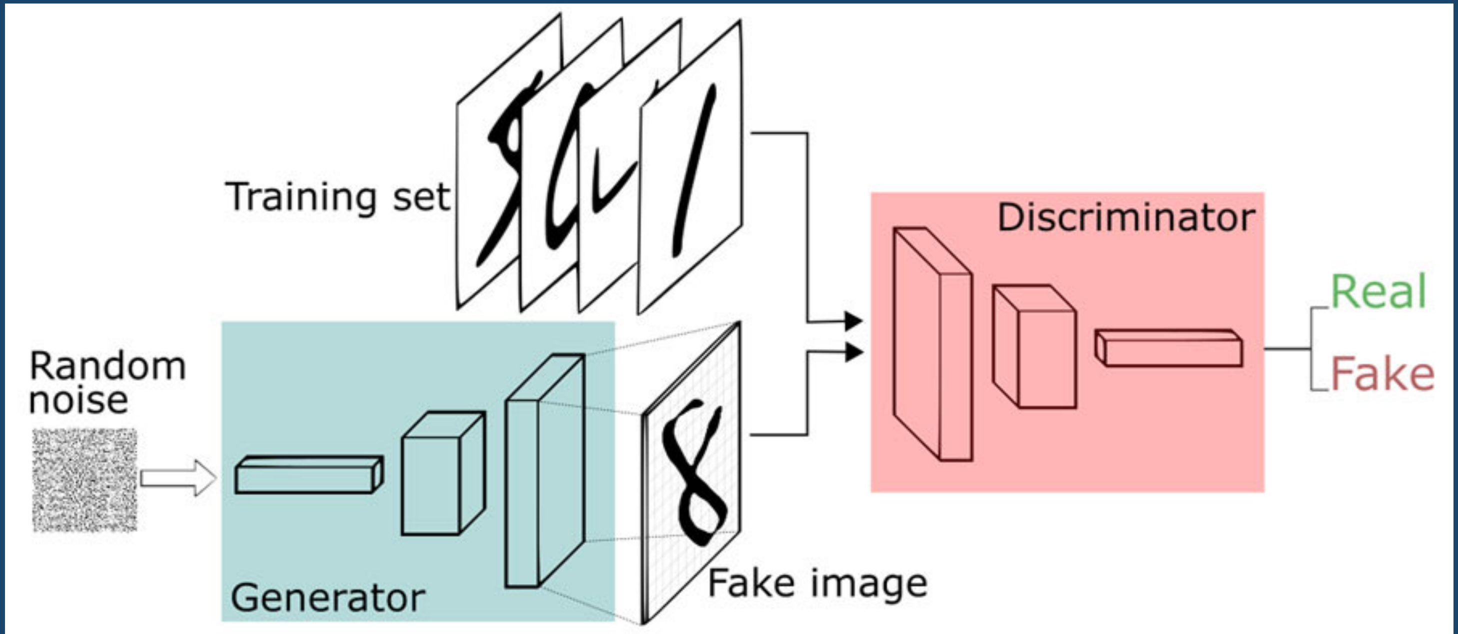
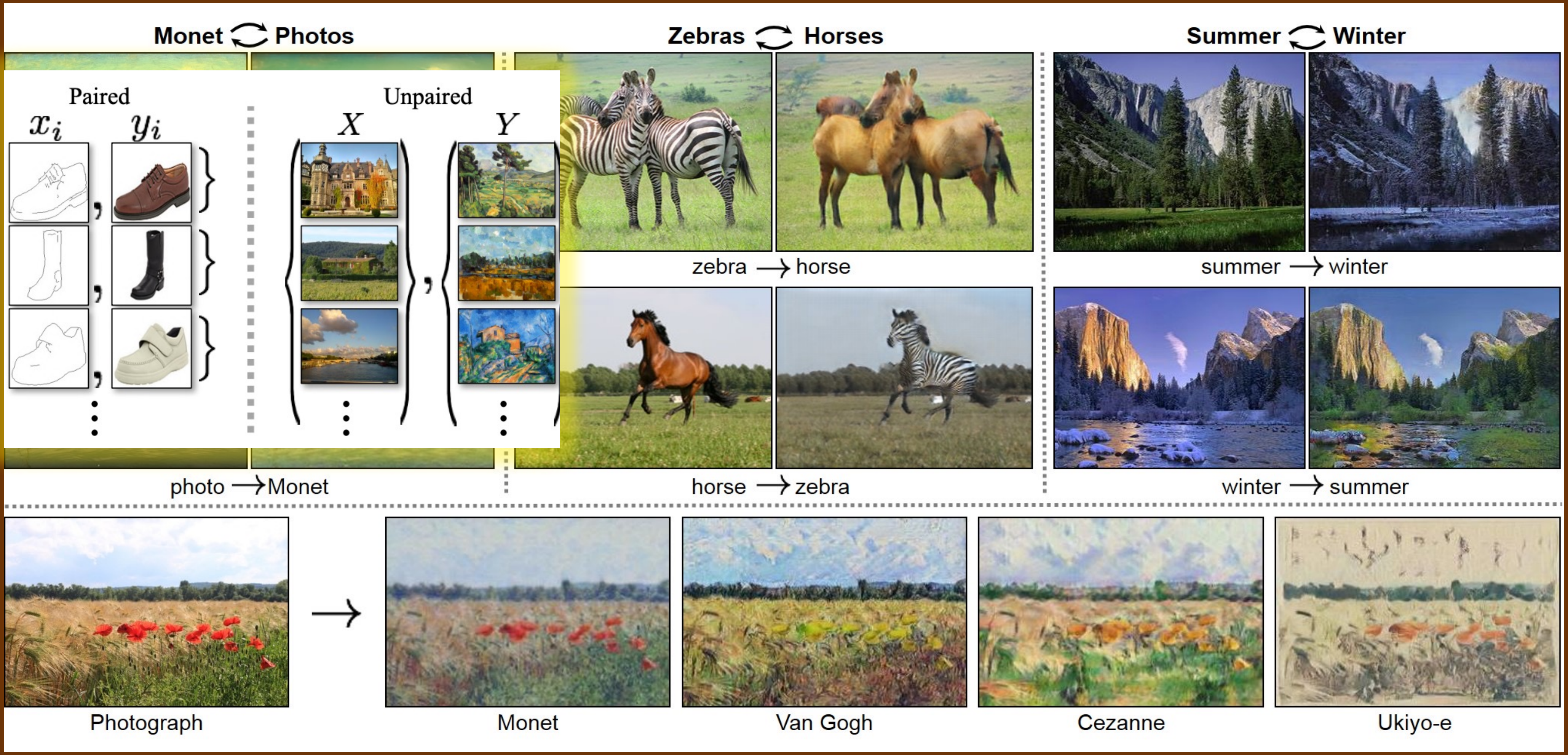


Image source: https://sthalles.github.io/intro-to-gans/

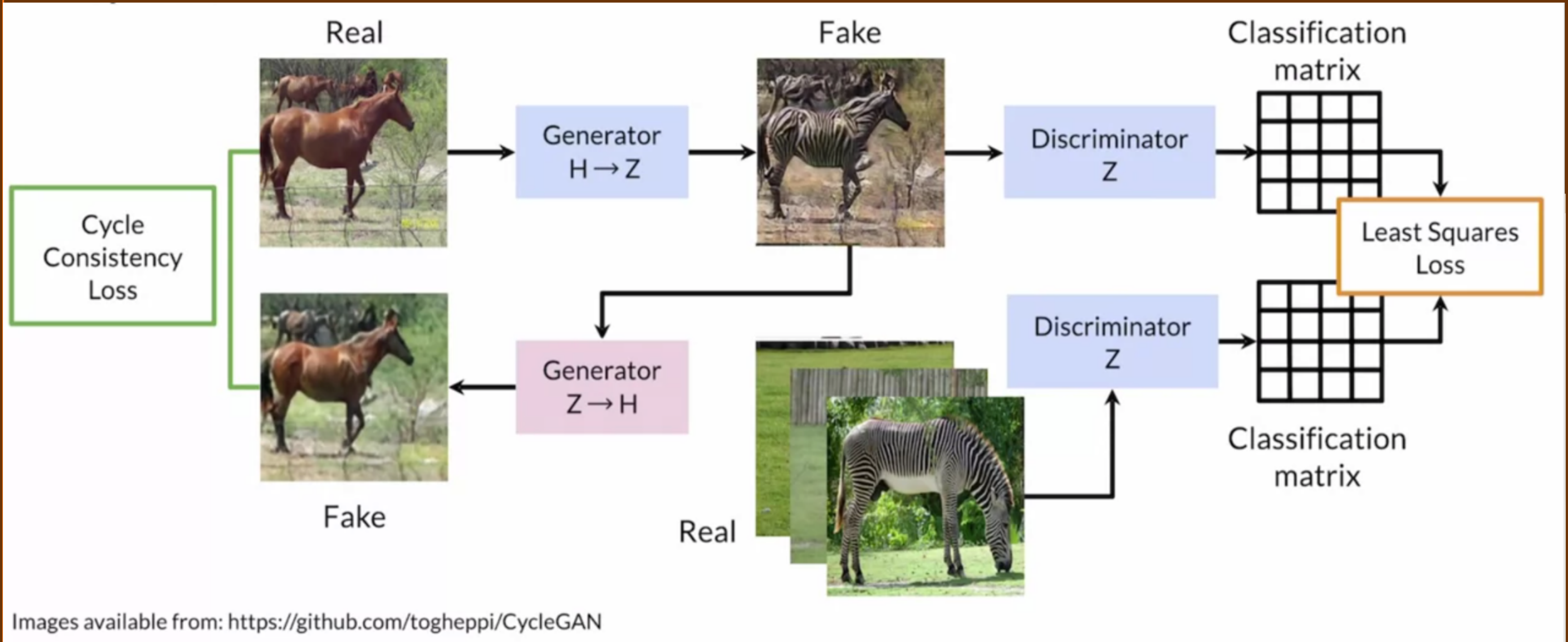
What are CycleGANs?



For two **unpaired** image collections X and Y, CycleGAN learns to "translate" an image from one domain into other and vice versa

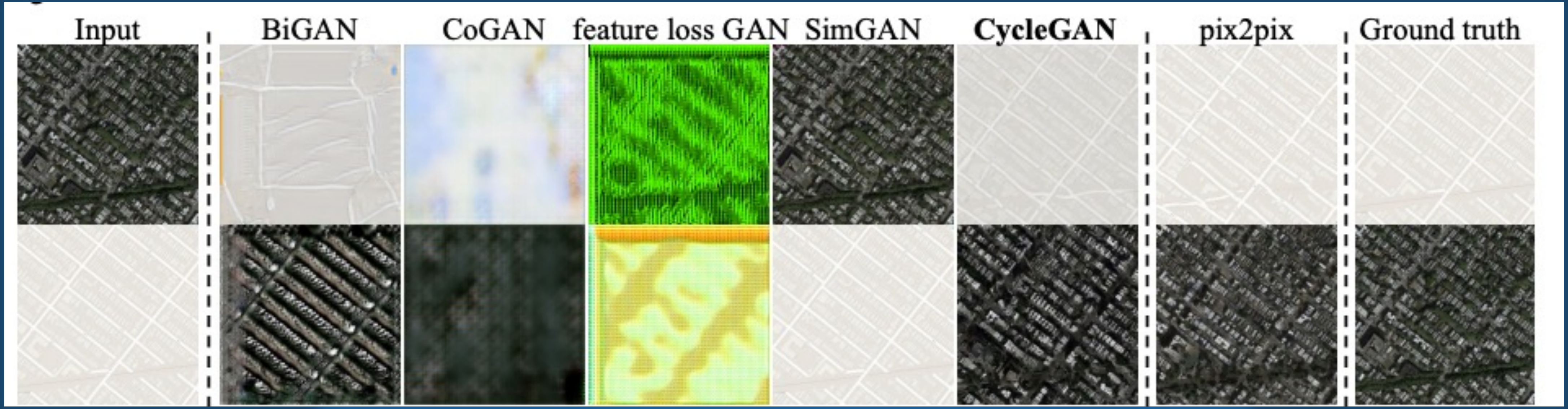
Source: https://junyanz.github.io/CycleGAN/

The intuition behind CycleGANs (2 GANs!)...



Source: https://junyanz.github.io/CycleGAN/

How did the authors evaluate their results?

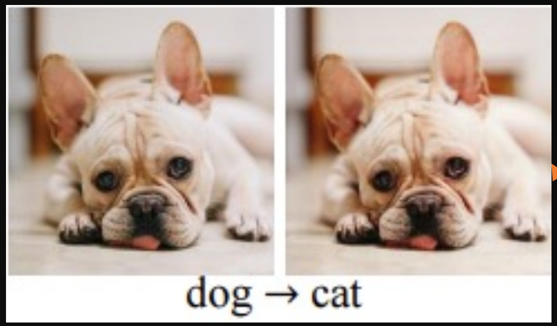


Loss	Map → Photo % Turkers labeled *real*	Photo → Map % Turkers labeled *real*
CoGAN [32]	$0.6\% \pm 0.5\%$	$0.9\% \pm 0.5\%$
BiGAN/ALI [9, 7]	$2.1\% \pm 1.0\%$	$1.9\% \pm 0.9\%$
SimGAN [46]	$0.7\% \pm 0.5\%$	$2.6\% \pm 1.1\%$
Feature loss + GAN	$1.2\% \pm 0.6\%$	$0.3\% \pm 0.2\%$
CycleGAN (ours)	$\mathbf{26.8\% \pm 2.8\%}$	$\mathbf{23.2\% \pm 3.4\%}$

Table 1: AMT "real vs fake" test on maps↔aerial photos at 256×256 resolution.

Image source: https://junyanz.github.io/CycleGAN/

CycleGAN Limitations



More varied transformations like geometric changes

Model does not work well when a test image looks unusual compared to training images

Due to changes in distribution characteristics (test set vs. training set)

Source: https://junyanz.github.io/CycleGAN/

Live Demo

- Link to Notebook

- Demo notebook: https://github.com/shyamal-anadkat/cycle-gan-demo/
- W&B: https://wandb.ai/shyamal/CycleGAN-and-pix2pix/runs/29q8w844?workspace=user-shyamal
- Check out their Github Repo: https://github.com/junyanz/CycleGAN

Summary

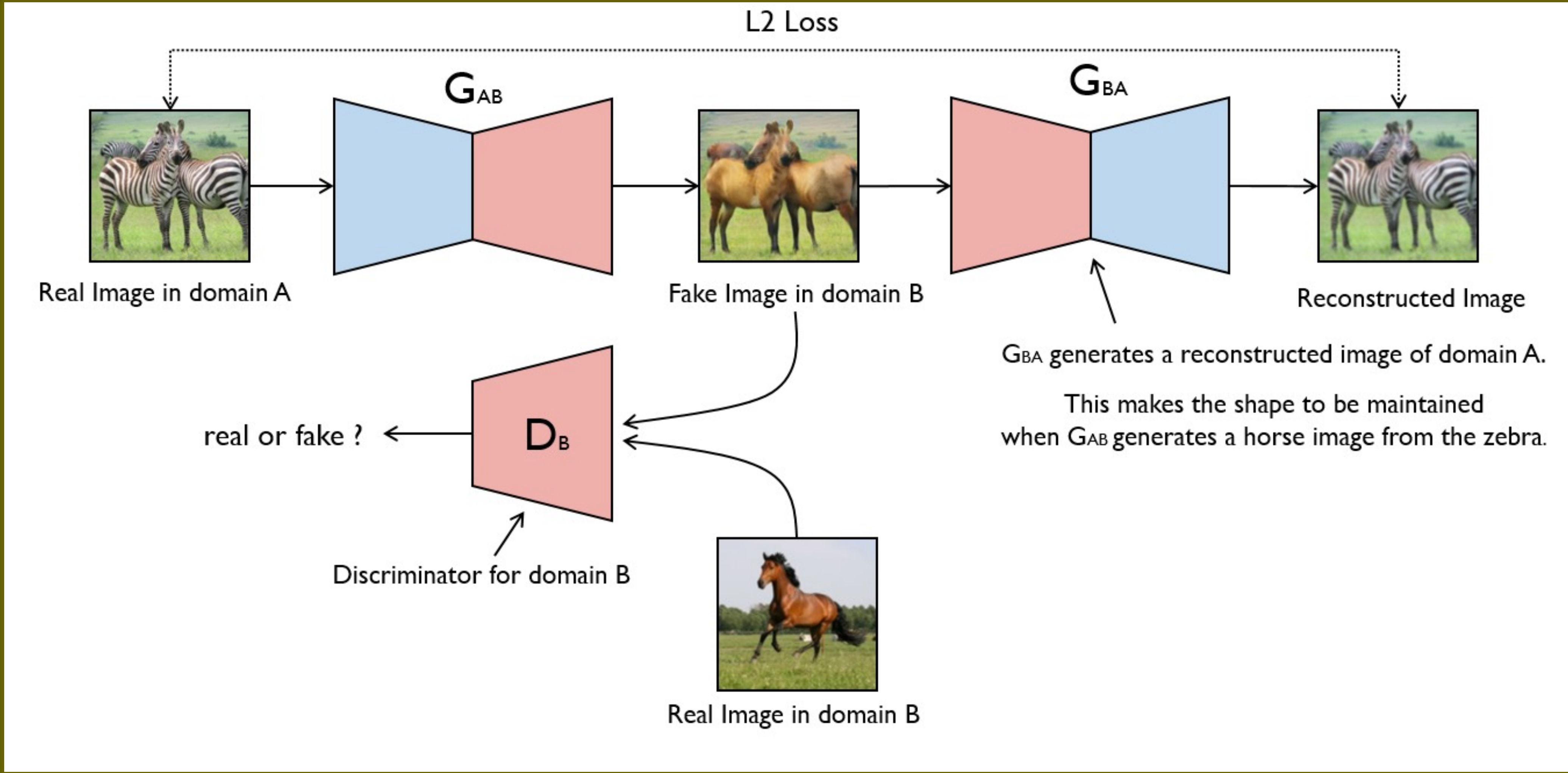
- CycleGAN uses two GANs for unpaired image-to-image translation
- Cycle consistency loss ensures "realistic" translations and content to be preserved while only changing the styles
- Applications in style transfer, object transfiguration etc. (**with some limitations)
- Implementations readily available in PyTorch & TensorFlow

Thank You!

Thanking them:

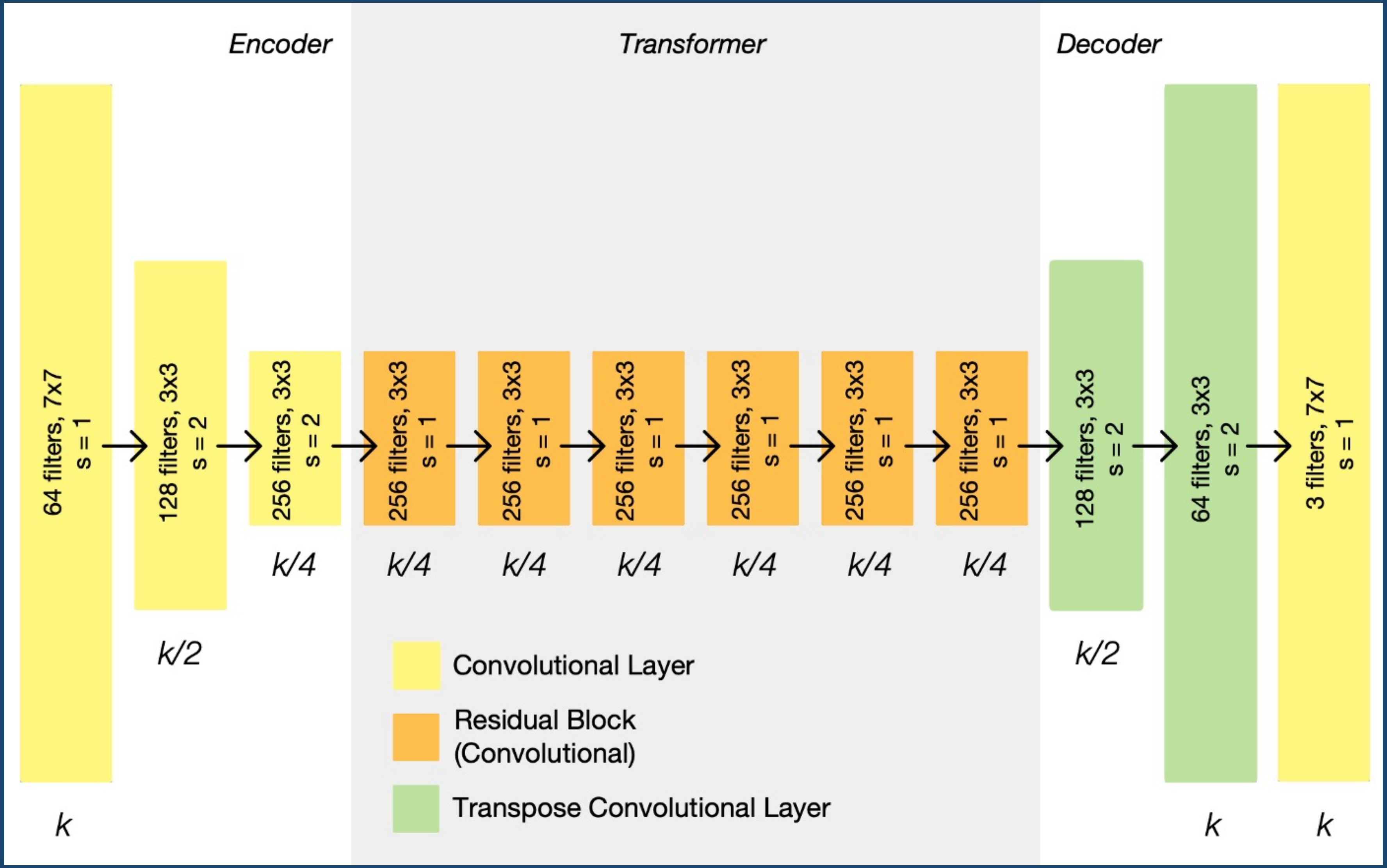
- The paper! https://arxiv.org/pdf/1703.10593v7.pdf
- https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
- https://junyanz.github.io/CycleGAN/
- https://medium.com/analytics-vidhya/
- https://github.com/yunjey/mnist-svhn-transfer

Addendum: Simplified CycleGAN



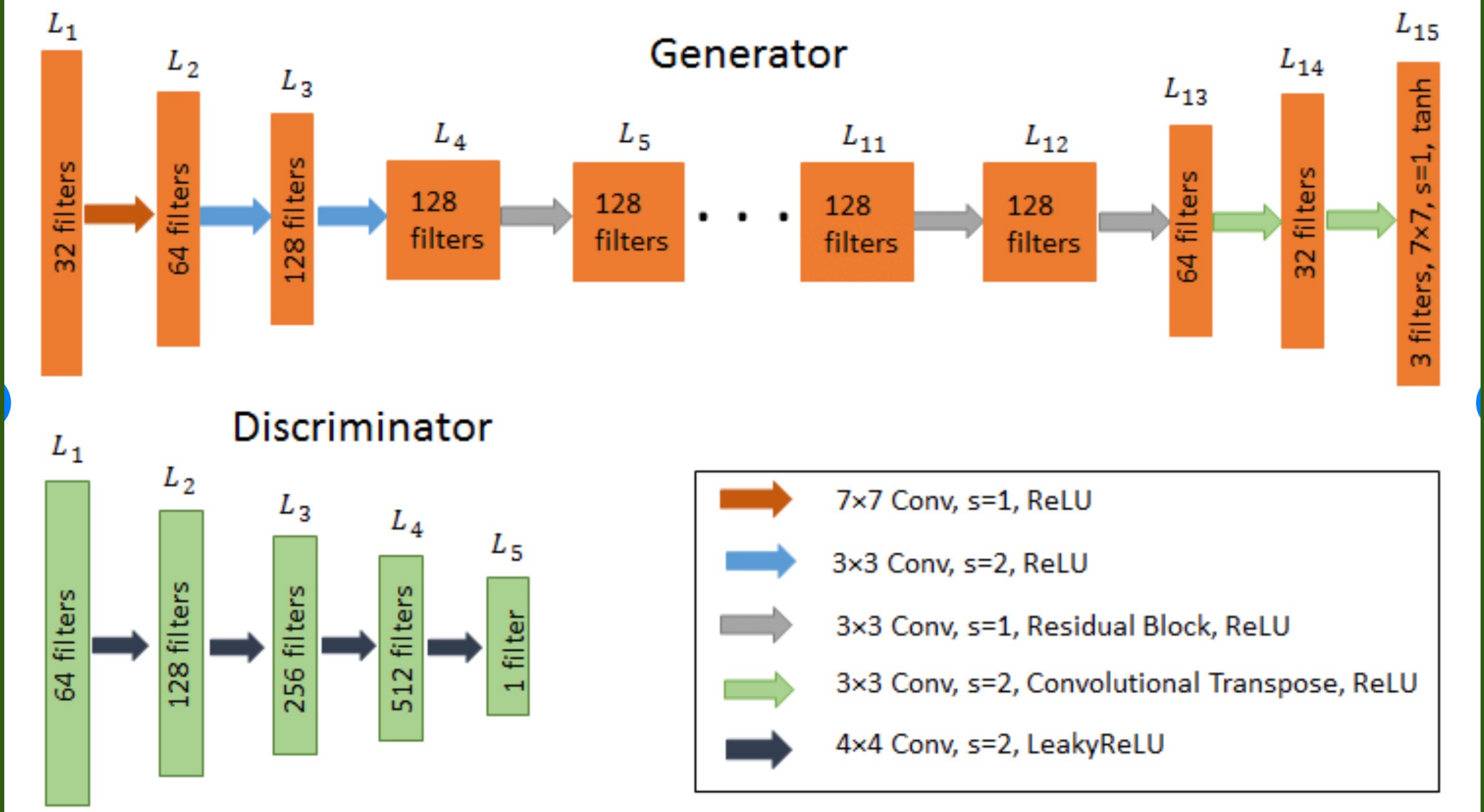
Source: https://github.com/yunjey/mnist-svhn-transfer

Addendum: Architecture



Source: https://github.com/yunjey/mnist-svhn-transfer

Addendum: Architecture



Architecture of the generator and discriminator of unpaired CycleGAN. Conv: 2D Convolutional filter, s: Stride, ReLU: Rectifier linear unit.

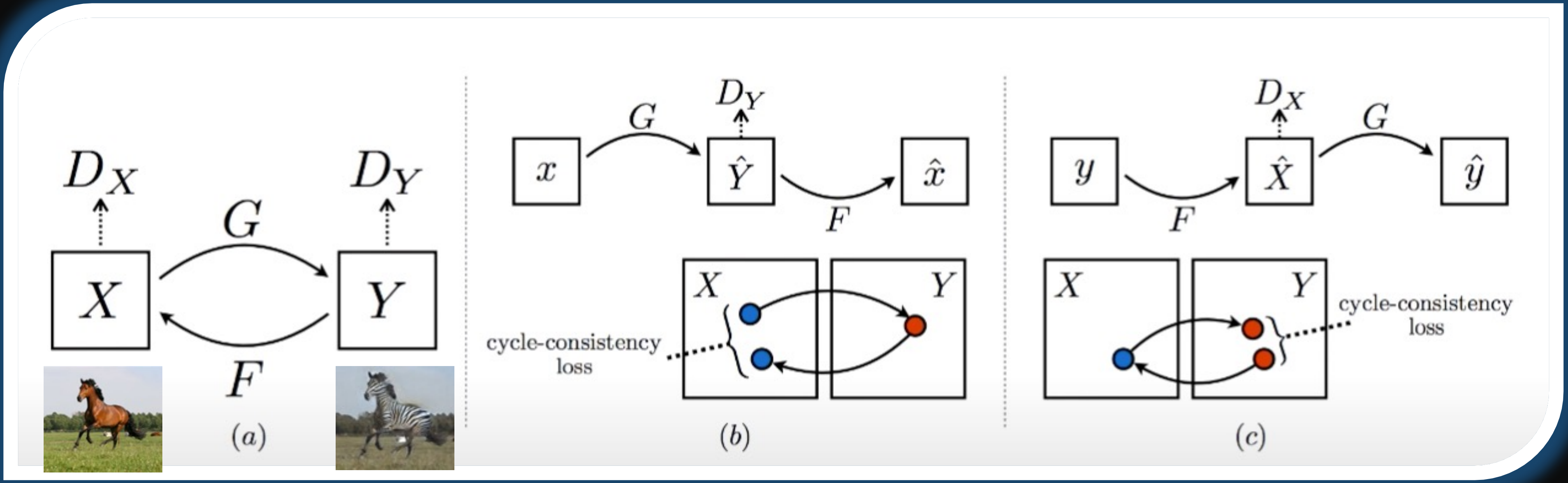
Source: https://github.com/yunjey/mnist-svhn-transfer

Addendum: Mathematical Intuition

a) Two mapping functions $G: X \rightarrow Y$ and $F: Y \rightarrow X$ and discriminators. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F.

b) Forward cycle consistency loss: $x \rightarrow G(x) \rightarrow F\big(G(x)\big) \approx x$

c) Backward cycle consistency loss: $y \rightarrow F(y) \rightarrow G\big(F(y)\big) \approx y$



Source: https://junyanz.github.io/CycleGAN/