IMAGE-TO-IMAGE TRANSLATION

APPLICATIONS OF CYCLEGAN

Prepared for:

CSE 728 Presentation

University at Buffalo The State University of New York

Introduction

- CycleGAN
- Applications

Methods

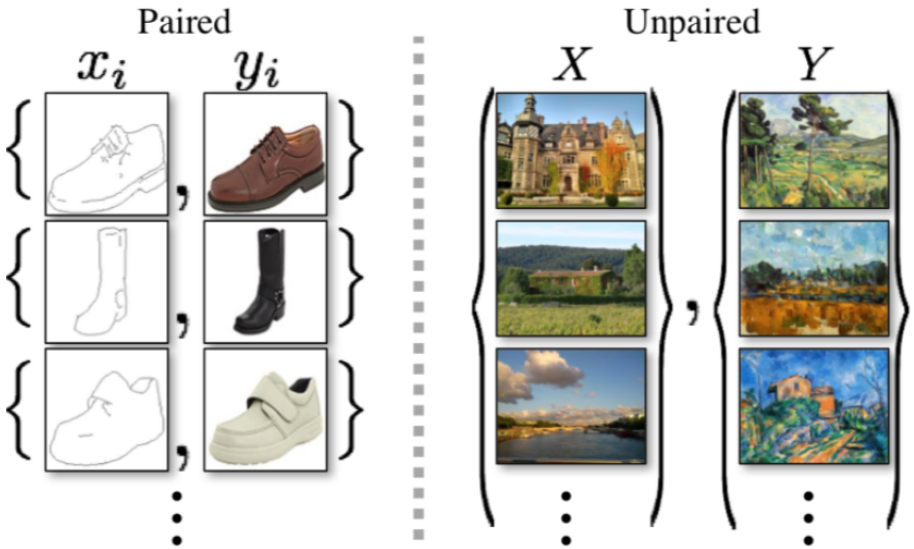
- Unpaired Data Mapping
- Model Architecture
- Cycle Consistency Loss
- Generator and Discriminator
- Loss functions
- Optimizer
- Model Training

Discussion

- Broader Application
- Limitation
- Code Demo

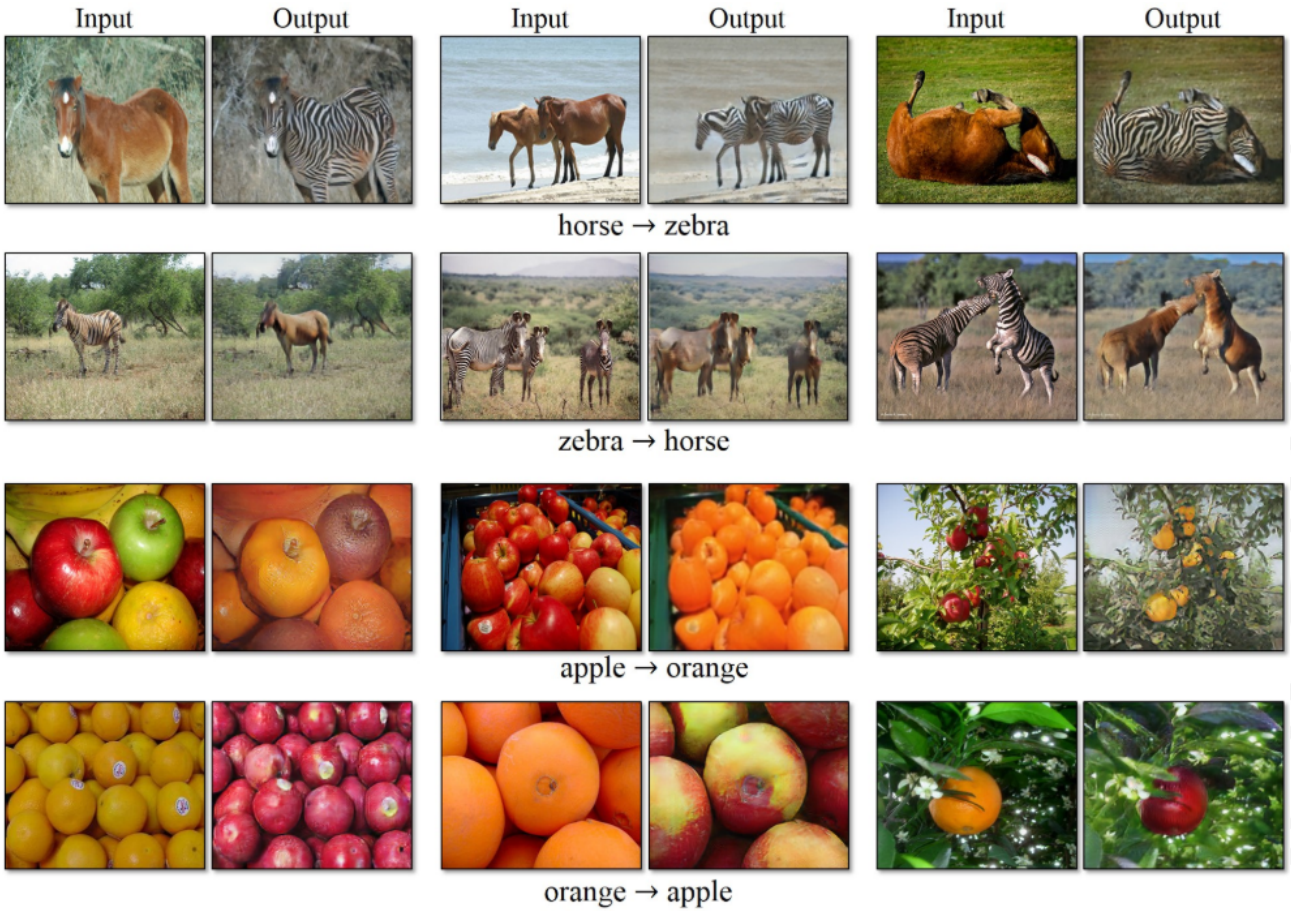
What is CycleGAN

- Instead of generating images from random noise, CycleGAN outputs different versions of input image.
- It is the first models to allow for *unpaired* image-to-image training.



Zhu et al. 2017

CycleGAN Application

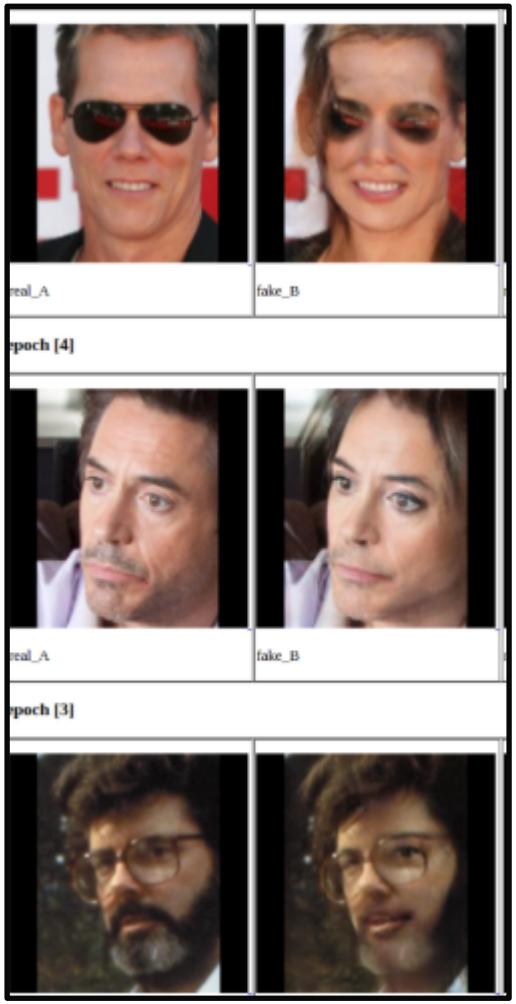


Zhu et al. 2017

CycleGAN Application

http://shikib.com/CycleGan.html

Male → Female



Female → Male

White → Asian

Asian → White

Introduction

- CycleGAN
- Applications

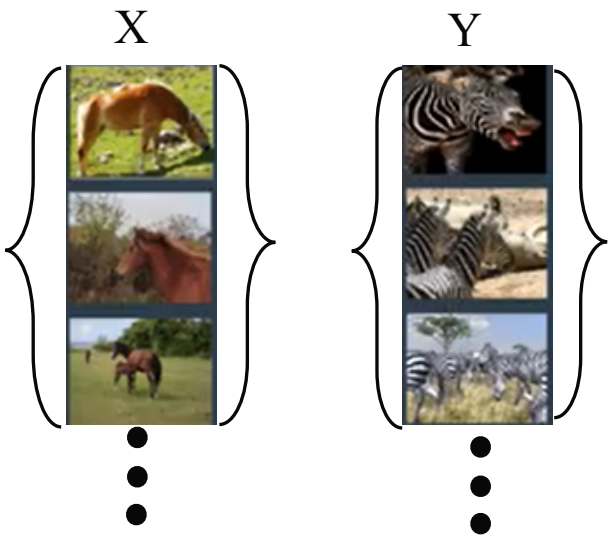
Methods

- Unpaired Data Mapping
- Model Architecture
- Cycle Consistency Loss
- Generator and Discriminator
- Loss functions
- Optimizer
- Model Training

Discussion

- Broader Application
- Limitation
- Code Demo

CycleGAN Unpaired Data Mapping



Mapping

Generator ($G_{X \text{ to } Y}$)

X → Y

Inverse Generator ($G_{Y \text{ to } X}$)

Discriminator D_X Discriminator D_Y

$$G_{Y \text{ to } X}(G_{X \text{ to } Y}(x)) \sim x$$

Cycle Consistency

Model Architecture

Photos, X

Cezanne Paintings, Y

x

$G_{X \text{ to } Y}$

Generated, $G_{X \text{ to } Y}(x)$

y

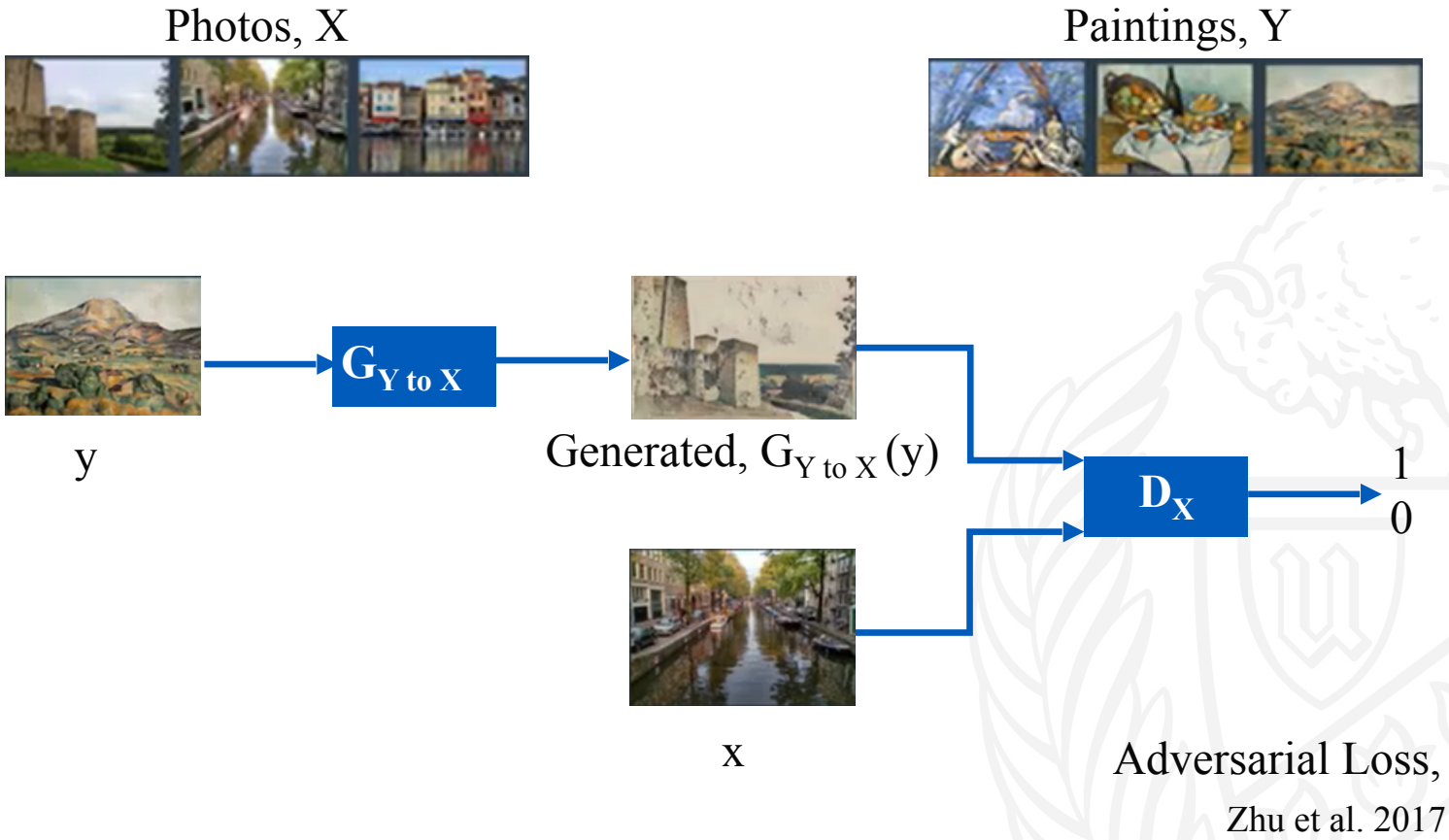
D_Y

1 Real
0 Fake

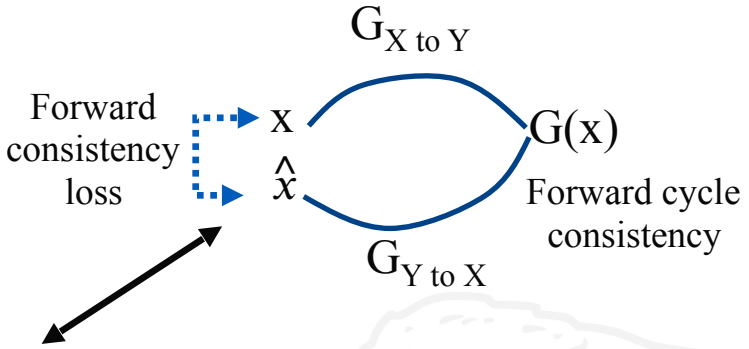
Adversarial Loss, L_Y

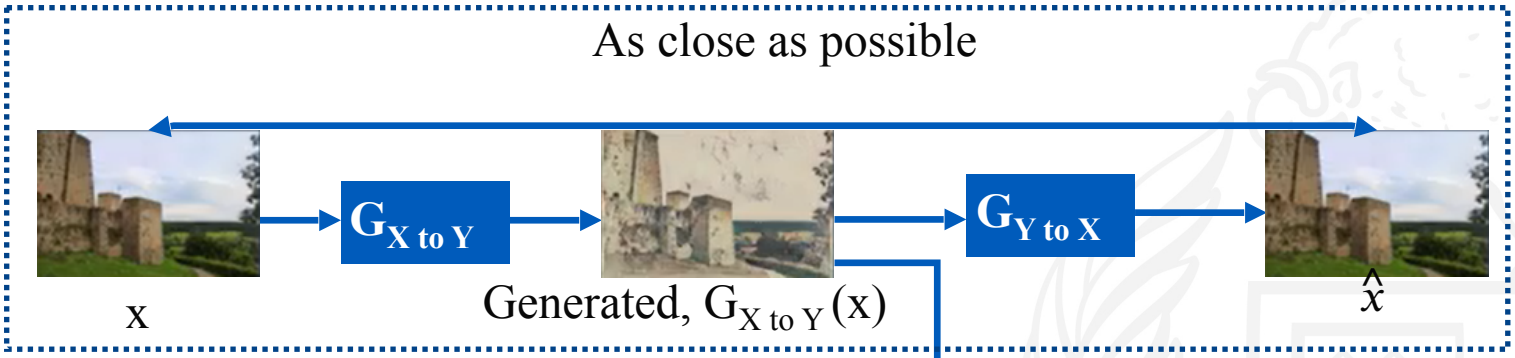
Zhu et al. 2017

Model Architecture



Cycle Consistency Loss

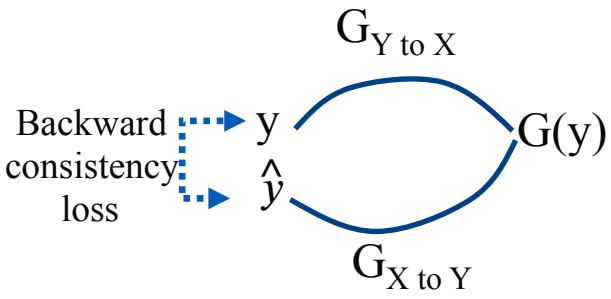




y

D_Y

0 Real
1 Fake



Zhu et al. 2017

Loss

- The full loss is adversarial loss + forward and backward cycle losses
- Adversarial loss is for style control
- Cycle consistency loss is for content control

$$L = L_y + L_x + \lambda L_{cycle}$$

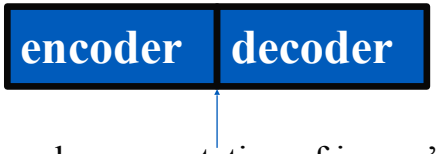
Controls the relative importance of L_{cycle} .
Zhu et al. (2017) suggests 10.

Why Does CycleGAN work

The assumption: the style and content of an image can be separated.

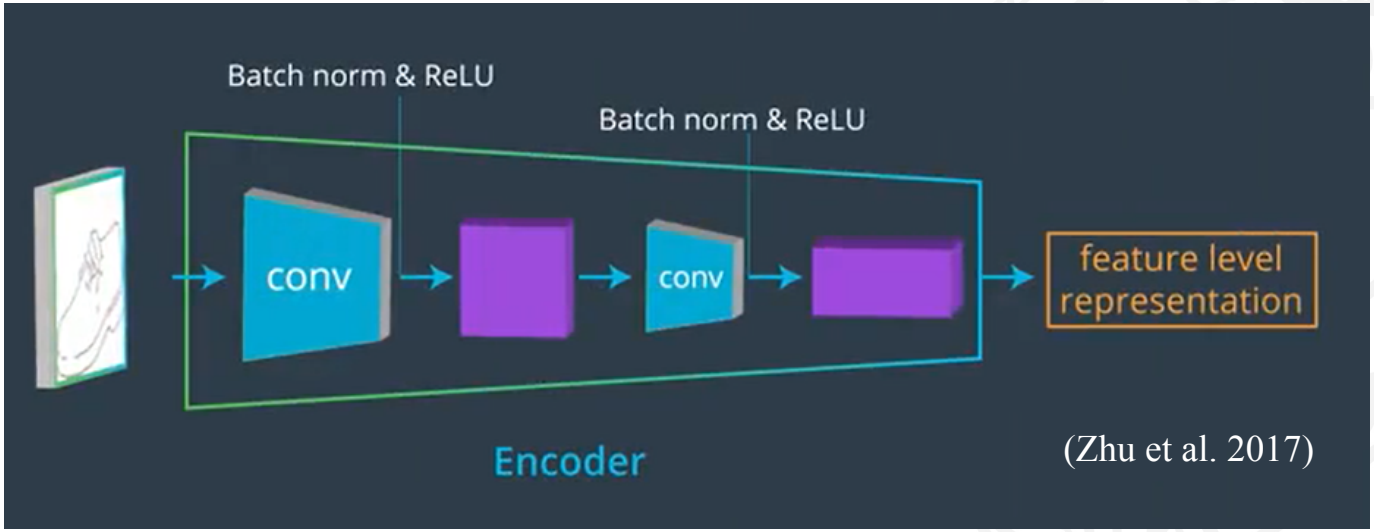
This is what the convolutional layers in the generators should learn

Generator ($G_{X \text{ to } Y}$ and $G_{Y \text{ to } X}$)



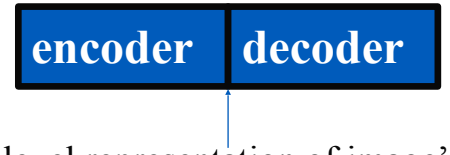
Feature level representation of image's content

Encoder: convolutional neural network that extract features from the image with content and discard other information such as texture and colors.

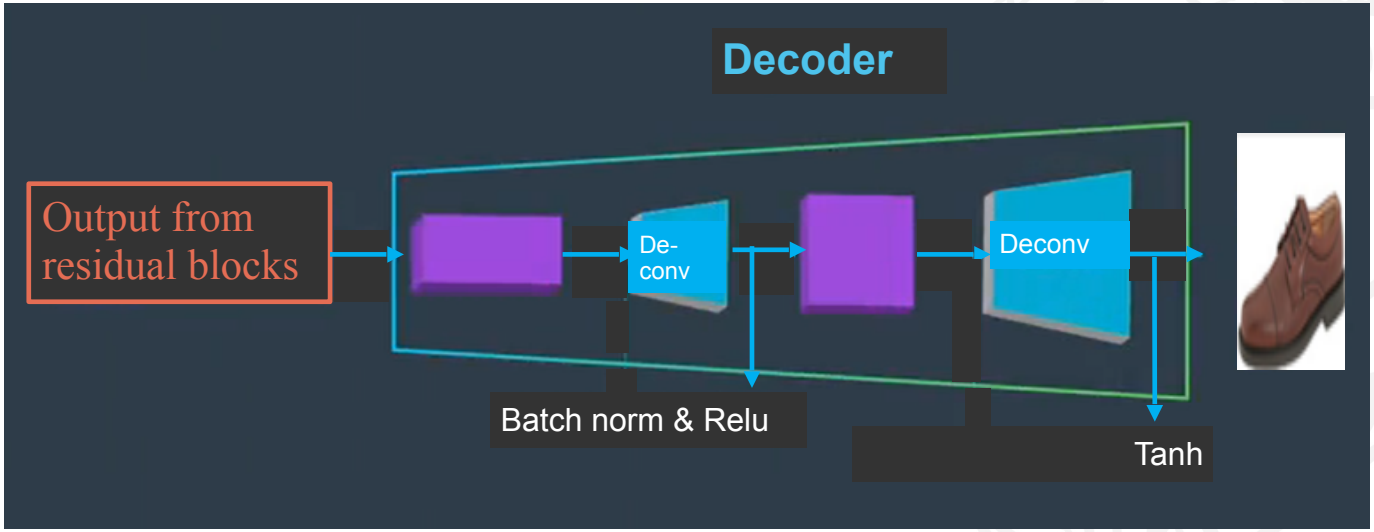


Layers increasing represent the content of the image

Generator ($G_{X \text{ to } Y}$ and $G_{Y \text{ to } X}$)



Decoder: transpose convolutional neural network that is responsible for turning the feature level representation into an transformed image



(Zhu et al. 2017)

Generator ($G_{X \text{ to } Y}$ and $G_{Y \text{ to } X}$)

Residual Blocks: connect the encoder and decoder

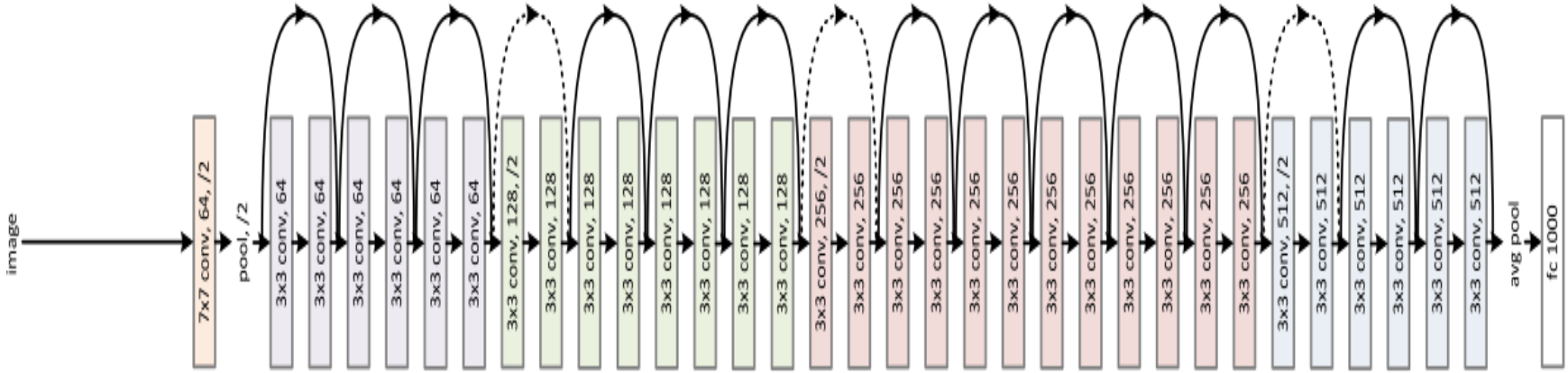
Why?

- Deeper networks are more likely to have vanishing or exploding gradients
- Have trouble reaching convergence.
- Batch normalization helps.
- Training accuracy stops improving at some point.
- In the worst cases, deep models would see their training accuracy worsen over time.

Generator ($G_{X \text{ to } Y}$ and $G_{Y \text{ to } X}$)

encoder | **Residual Blocks** | **decoder**

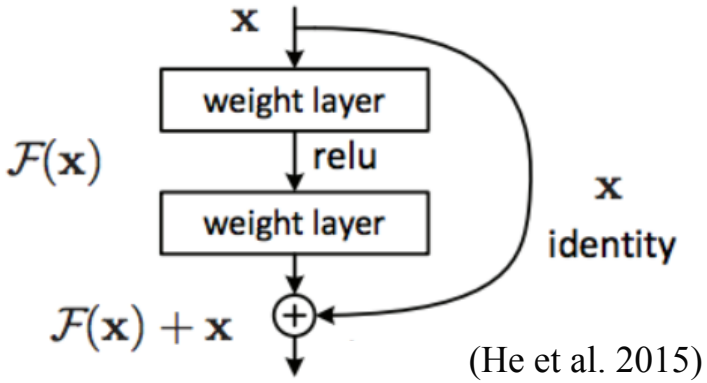
ResNet50 for image classification: connecting the output of one layer with the input of an earlier layer



Generator ($G_{X \text{ to } Y}$ and $G_{Y \text{ to } X}$)

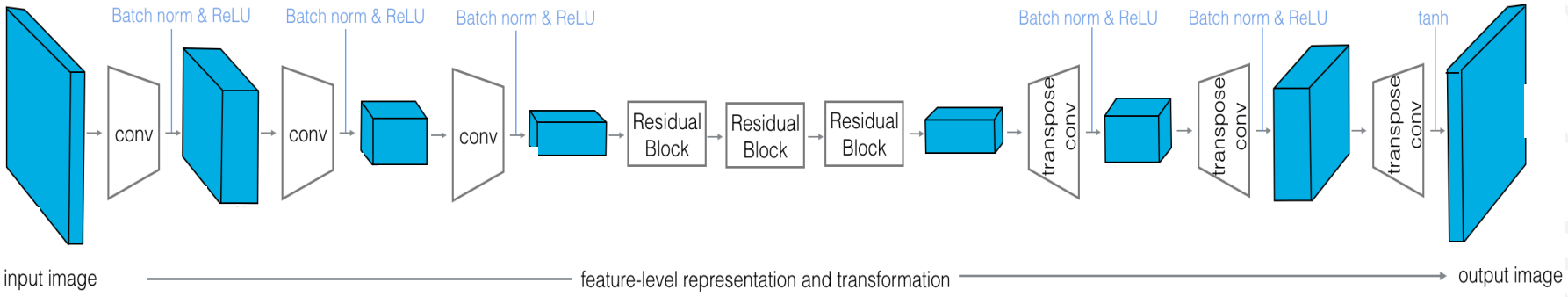
Residual blocks

- allow us to learn so-called *residual functions F(x)*
- typically, two convolutional layers + normalization layer and a ReLu in between.
- convolutional layers should have the same number of inputs as outputs.
- typically 6 or more blocks



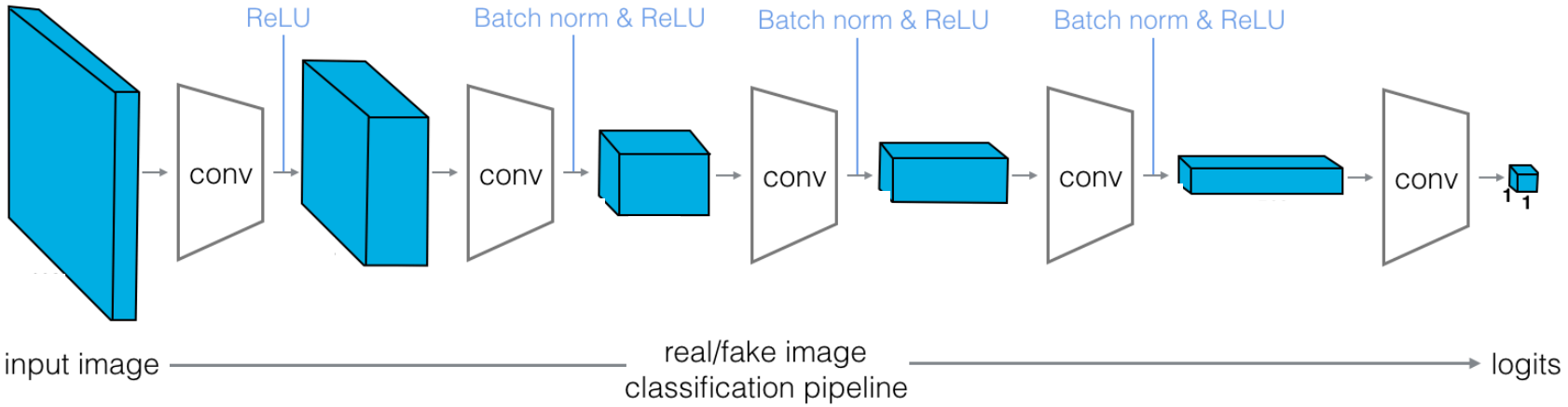
(He et al. 2015)

Generator ($G_{X \text{ to } Y}$ and $G_{Y \text{ to } X}$)



(Zhu et al. 2017)

Discriminator (D_X and D_Y)



(Zhu et al. 2017)

More About Loss Functions

The full loss is adversarial loss + forward and backward cycle losses

Adversarial loss:

Discriminator

$$L_{DY} = E_{y\sim pdata(y)}[\,(D_Y\,(y)-1)^2] + E_{x\sim pdata(x)}[\,D_Y\,(G(x))^2]$$

real loss — fake loss

$$L_{DX} = E_{x\sim pdata(x)}[\,(D_X\,(x)-1)^2] + E_{y\sim pdata(y)}[D_X\,(F(y))^2]$$

real loss — fake loss

Least-square loss is more stable/gradient vanish

Generator

$$L_G = E_{x\sim pdata(x)}[\,D_Y\,(G(x)-1)^2]$$

$$L_F = E_{y\sim pdata(y)}[\,D_X\,(F(y)-1)^2]$$

(Zhu et al. 2017)

More About Loss Functions

The full loss is adversarial loss + forward and backward cycle losses

Cycle consistency loss:

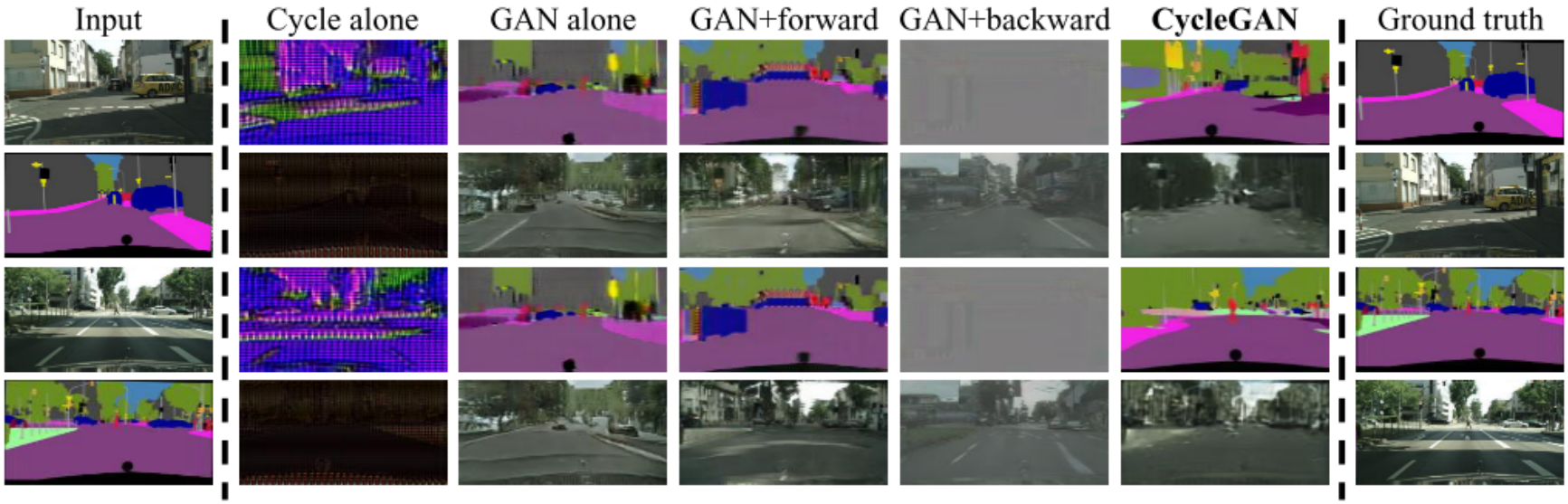
$$L_{cycle} = E_{x\sim pdata(x)}[||F(G(x))-x||_1] + E_{y\sim pdata(y)}[||G(F(y))-y||_1]$$

L1 norm

Forward consistency loss Backward consistency loss

(Zhu et al. 2017)

More About Loss Functions



(Zhu et al. 2017)

Optimizer

Adam with adapted learning rate:

- all networks are trained from scratch with a learning rate of 0.0002 for the first 100 epochs
- linearly decay the rate to zero over the next 100 epochs

(Zhu et al. 2017)

General Training Steps

Training the Discriminators

1. Compute the discriminator D_X loss on real images
2. Generate fake images that look like domain X based on real images in domain Y
3. Compute the fake loss for D_X
4. Compute the total loss and perform backpropagation and D_X optimization
5. Repeat steps 1-4 only with D_Y and your domains switched!

Training the Generators

1. Generate fake images that look like domain X based on real images in domain Y
2. Compute the generator loss based on how D_X responds to fake X
3. Generate *reconstructed* $\hat{Y}$ images based on the fake X images generated in step 1
4. Compute the cycle consistency loss by comparing the reconstructions with real Y images
5. Repeat steps 1-4 only swapping domains
6. Add up all the generator and reconstruction losses and perform backpropagation + optimization

(Zhu et al. 2017)

Introduction

- CycleGAN
- Applications

Methods

- Unpaired Data Mapping
- Model Architecture
- Cycle Consistency Loss
- Generator and Discriminator
- Loss functions
- Optimizer
- Model Training

Discussion

- Broader Application
- Limitation
- Code Demo

Boarder Applications

- Video-to-Video (Zhu et al. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks)
- Voice-to-Voice (Kaneko and Kameoka 2017. Parallel-Data-Free Voice Conversion Using Cycle-Consistent Adversarial Networks)
- Faces-off (Jin et al. 2018. CycleGAN Face-off)
- Many-to-Many Mappings (Almahairi et al. 2018. Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data)

Shortcomings

- It will only show one version of a transformed output even if there are multiple possible outputs.
- A simple CycleGAN produces low-resolution images, though there is some research around high-resolution GANs (Wang et al. 2018. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs)
- Good at color and texture changes, while bad at geometric changes

(Zhu et al. 2017)

Some Improvements

- Cycle consistency on discriminator CNN feature level

  Modified cycle consistency loss as a linear combination of discriminator feature extractor (last layer) and pixel level consistency losses

$$\tilde{\mathcal{L}}_{\text{cyc}}(G, F, D_X, X, \gamma) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\gamma \| f_{D_X}(F(G(x))) - f_{D_X}(x) \|_1 + (1-\gamma) \| F(G(x)) - x \|_1]$$

Wang and Lin 2018

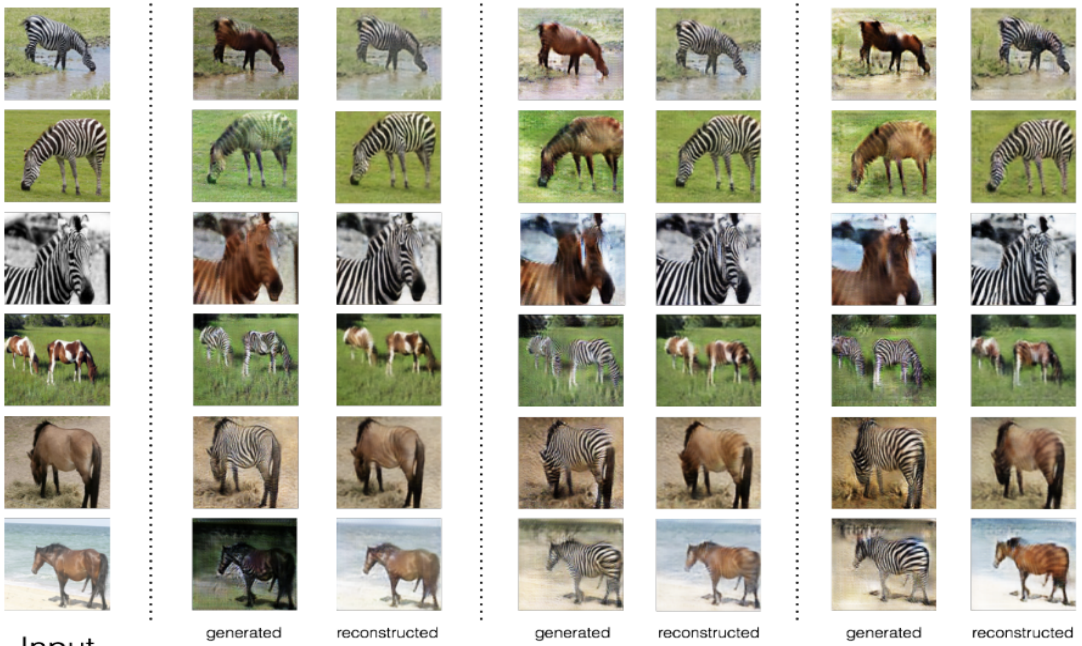
- Cycle consistency weight decay

  Gradually decay the weight of cycle consistency loss λ as training progress but keep it bigger than 0.

Some Improvements

- Weight cycle consistency by quality of generated image

$$\tilde{\mathcal{L}}_{\text{cyc}}(G, F, D_X, X, \gamma) =$$
$$\mathbb{E}_{x \sim p_{\text{data}}(x)} \left[D_X(x) \left(\gamma \| f_{D_X}(F(G(x))) - f_{D_X}(x) \|_1 + (1-\gamma) \| F(G(x)) - x \|_1 \right) \right]$$



Wang and Lin 2018

Code Demo

THANK YOU FOR YOUR ATTENTION!

Q & A!