APS360 Fundamentals of AI

Lisa Zhang

Lecture 14; March 21, 2019

Agenda

- Generative Adversarial Networks
- Adversarial Examples

Generative Models

Generative Models

- A **generative model** learns the *structure* of a set of input data
- In doing so, the model learns to *generate* new data
- Examples:
  - Autoencoders
  - RNN for text generation

Drawbacks of Autoencoders

These faces are generated using a variant of autoencoders

Drawbacks of Autoencoders

- Blurry images, blurry backgrounds
- Loss function:
  - MSE loss
  - Predict the "average" value to minimize MSE loss

Generative Adversarial Network

Idea: train two networks

- **Generator network**: try to fool the discriminator by generating real-looking images
- **Discriminator network**: try to distinguish between real and fake images

Training GANs: two-player game

Play a minmax game:

- the discriminator will try to do the best job it can
- the generator is set to make the discriminator as wrong as possible

Loss function

- **Minimize**, with respect to *generator* parameters, and
- **Maximize**, with respect to *discriminator* parameters:
  - (log) probability that the **discriminator** correctly identifies a *real* image, plus
  - (log) probability that the **discriminator** correctly identifies an image *generated* by the **generator**

Training

Alternate between:

- Training the discriminator
- Training the generator

GANs in practice

- Can work very well, but very difficult to train!
- Difficult to numerically see whether there is progress
  - Plotting the "training curve" (the minmax objective) doesn't help much
- Difficult to generate globablly consistent structure
- Poor training if the discriminator is too good
- Mode collapse (next slide)

Mode Collapse

- Mode = "average"
- GAN model learns to generate one type of input data (e.g. only digit 1)
- Generating anything else leads to detection by discriminator
- Generator gets stuck in that local optima

Leaky Relu activation

Like a relu, but "leaks" data:

- $f(x) = x$ if $x >= 0$
- $f(x) = \alpha x$ if $x < 0$

You've implemented this in assignment 1.

Batch Normalization

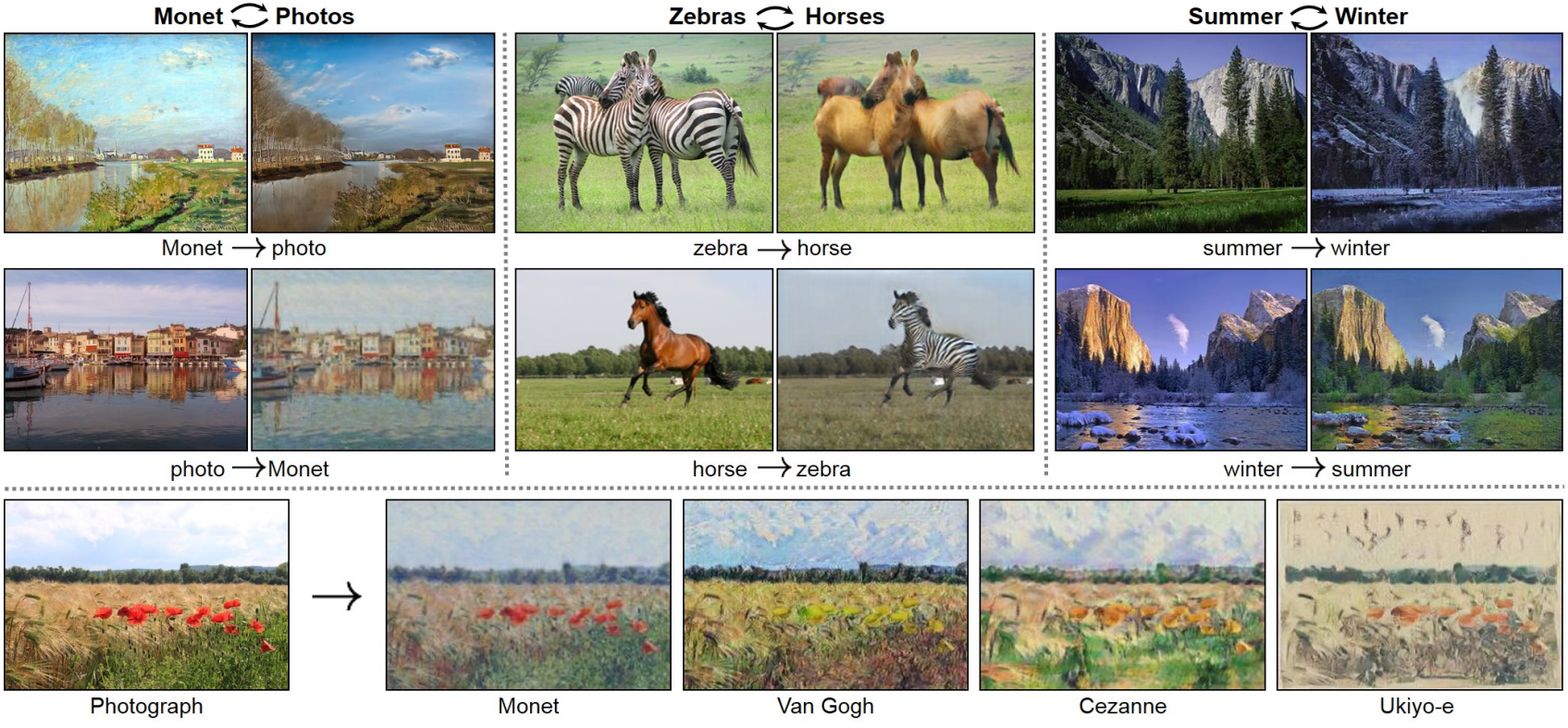
Normalization on input data helps training. But what about the hidden activations?

- **Training time**: normalize activations based on mini-batch statistics, and keep track of those statistics
- **Test time**: normalize activations based on saved statistics

GAN now

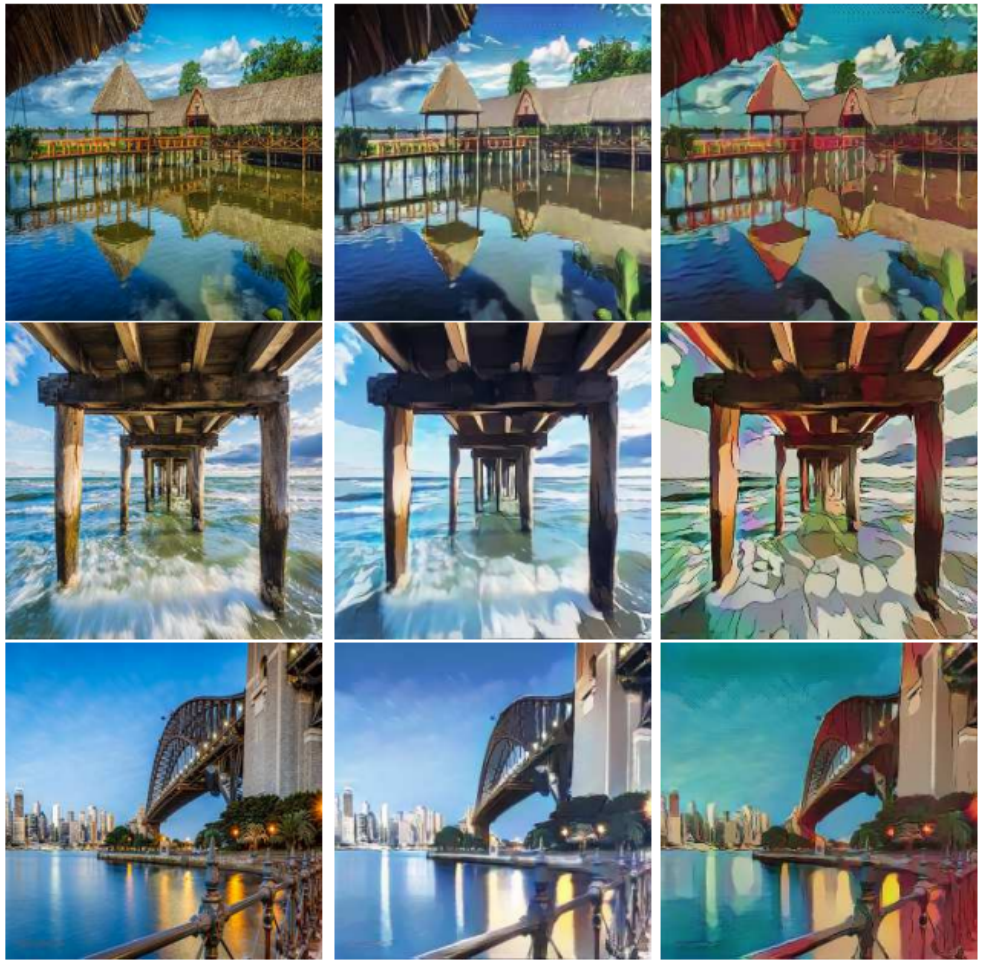
https://arxiv.org/pdf/1710.10196.pdf (2018)

CycleGAN



https://junyanz.github.io/CycleGAN/ (2017)

CartoonGAN



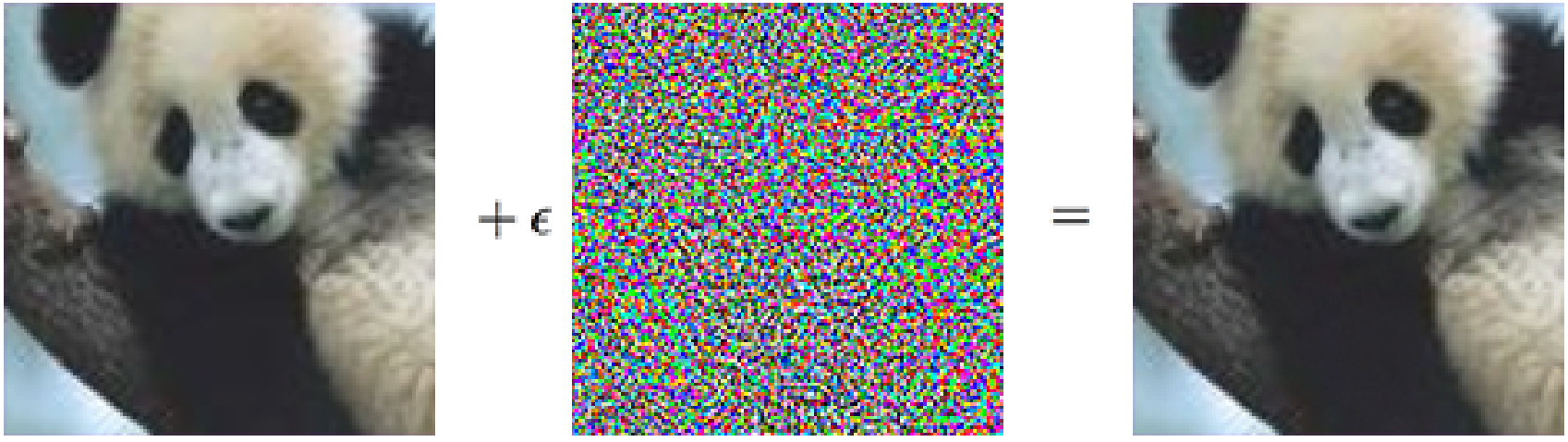
(a) input photo (b) Shinkai style (c) Hayao style

Figure 5. Some results of different artistic styles generated by CartoonGAN. (a) Input real-world photos. (b) Makoto Shinkai style. (c) Miyazaki Hayao style.

http://openaccess.thecvf.com/content_cvpr_2018/papers/Chen_CartoonG

Adversarial Examples

What is this a picture of?



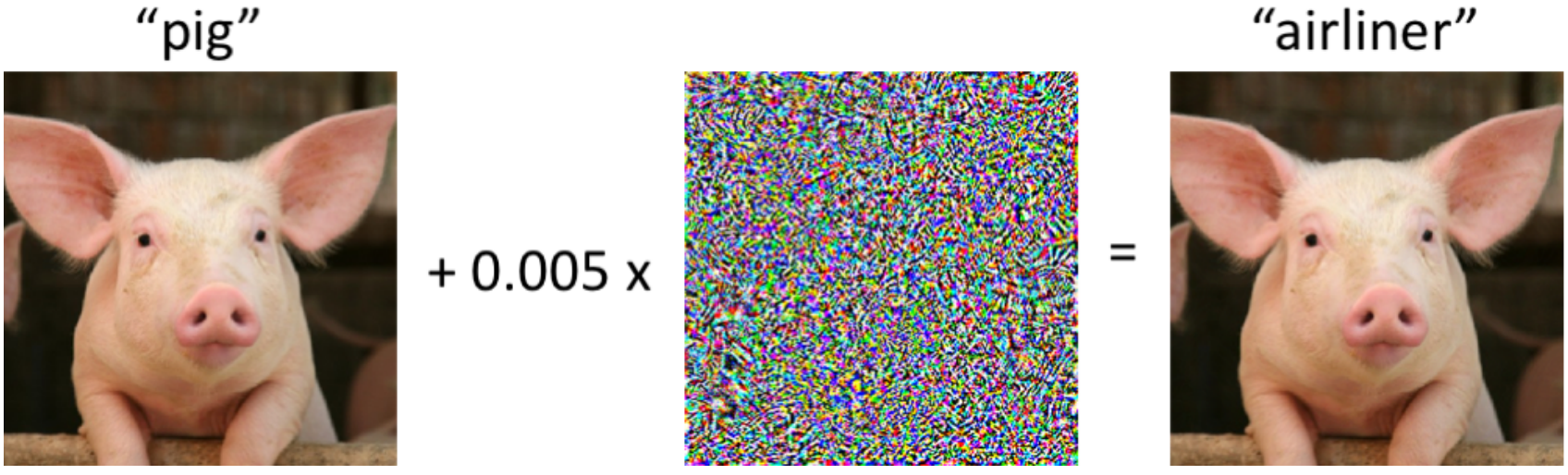
"panda"

57.7% confidence

"gibbon"

99.3% confidence

What is this a picture of?



Adversarial attack

Goal: Choose a small perturbation ϵ on an image x so that a neural network f misclassifies $x + \epsilon$.

Approach:

Use the same optimization process to choose ϵ to minimize the probability that

$$f(x + \epsilon) = \mathit{correctclass}$$

We are treating ϵ as the **parameters**.

Targeted vs Non-Targeted Adversarial Attack

Non-targetted attack

Minimize the probability that $f(x + \epsilon) = \textit{correctclass}$

Targetted attack

Maximize the probability that $f(x + \epsilon) = \textit{targetclass}$

White-box Adversarial Attack

- Assumes that the model is known
- We need to know the architectures and weights of f to optimize ϵ

Black-box Adversarial Attack

- Don't know the architectures and weights of f to optimize ϵ
- Substitute model mimicking target model with known, differentiable function
  - adversarial attacks often transfer across models!

Printed Objects

https://openai-public.s3-us-west-2.amazonaws.com/blog/2017-07/robust-adversarial-examples/iphone.mp4

3D Objects

https://www.youtube.com/watch?v=piYnd_wYlT8

Defenses Against Adversarial Attack

- Active area of research

Failed Defenses

- Generative pre-training
- Adding noise at test time
- Averaging many models
- Weight decay
- Adding noise at training time
- Adding adversarial noise at training time
- Dropout