Economics 2355: Generative Adversarial Networks

Melissa Dell

Harvard University

February 2021

Outline

Two Types of Models

- **Supervised**: provide data (x) and labels (y); estimate a function to map the data to the labels, y=f(x)
  - Classification
  - Localization
  - Segmentation
  - Object detection
- **Unsupervised**: only provide data, no labels. Learn some underlying structure of the data
  - Dimensionality reduction (i.e. tSNE, UMAP)
  - k-means clustering
  - Density estimation
  - Autoencoders

Generative Models

- Generative models are a type of unsupervised model that generate new samples from the same distribution as the training data
- In other words, given the distribution of training data, we need to learn the model distribution
- If we do a good job, the data sampled from the generated distribution will be indistinguishable from real data sampled from that domain

Density Estimation for Generative Models

- **Explicit density estimation**: not our focus
- **Implicit density estimation**: learn a way to sample from the data generating process without explicitly defining it

Applications of Generative Models

Well-known applications are in art, such as style transfer or colorization. A few that seem potentially relevant to us:

- cleaning document backgrounds
- super-resolution
- data augmentation

For More on Generative Models

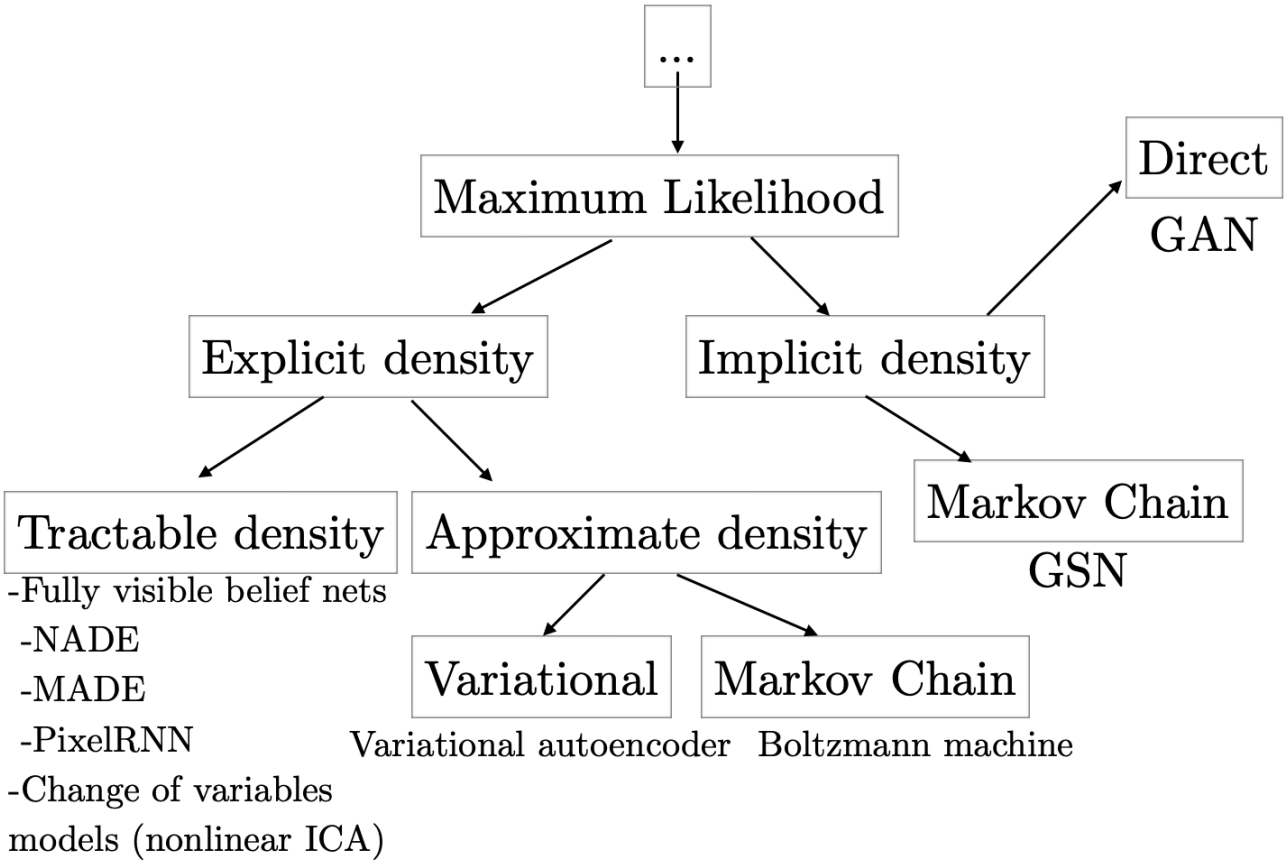

Figure: Goodfellow, Ian. "Nips 2016 tutorial: Generative adversarial networks." arXiv preprint arXiv:1701.00160 (2016). https://arxiv.org/pdf/1701.00160.pdf]

Outline

GAN Motivation

- Recall, our motivation is to be able to sample from the training data distribution, which is a complex, high dimensional object
- One approach: try to estimate or approximate this density (can be slow, difficult)
- GANs use a different approach, that turns out to be much more tractable. Sample from a simple distribution that we know how to sample from (i.e. random noise, Gaussian, uniform). Then learn a model that can transform a sample from the simple distribution to a sample from the training distribution
- We can model this complex transformation with a neural network

How Do We Train this Network

Insight from Goodfellow et al. ”Generative Adversarial Nets.” In NIPS 2014: we can use a game theoretic approach between two adversarial networks!

- **Generator Network:** tries to fool the discriminator by generating images that look like the real ones
- **Discriminator Network:** tries to distinguish real and fake images

GANs

- The discriminator is just a classification network (is the image real or fake?)
- When the discriminator successfully identifies real and fake images, no change is needed to the model parameters, whereas the generator is penalized with large updates to model parameters. Alternately, when the generator fools the discriminator, no change is needed to its parameters, but the discriminator's parameters are updated
- At the limit, the generator would produce perfect replicas and the discriminator would always predict a class score of 0.5 for real and 0.5 for fake

GANs as a Two-Player Game

$$\min_{\theta_g} \max_{\theta_d} [\mathop{\mathbb{E}}_{z \sim p_{data}} log D_{\theta_d}(x) + \mathop{\mathbb{E}}_{z \sim p(z)} log(1 - D_{\theta_d}(G_{\theta_g}(z)))] \quad (1)$$

- θ_g are the parameters of the generator network and θ_d are the parameters of the discriminator network
- D is the likelihood of that the image is real. $D_{\theta_d}(x)$ is the discriminator output for real data x and $D_{\theta_d}(G_{\theta_g}(z))$ is the discriminator output for fake data $G(z)$. To maximize the objective, the discriminator wants D(x) close to 1 and D(G(z)) close to 0
- To minimize the objective, the generator wants $D(G(z))$ to be close to one, fooling the discriminator

Training

Alternate between:

1. **Gradient ascent on discriminator**:
   $\max_{\theta_d}[\mathbb{E}_{z\sim p_{data}} log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$
2. **Gradient descent on generator**
   $\min_{\theta_g} \mathbb{E}_{z\sim p(z)} log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$

In practice, doesn't work well, because gradient of $D_{\theta_d}(G_{\theta_g}(z))$ is flat when we have bad samples, and only becomes steep when the generator is doing a pretty good job. Instead, max likelihood of discriminator being wrong

Training GANs

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

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

end for

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

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

end for

Figure: Goodfellow et al., 2014

GAN Architectures

- Generator is an upsampling network with fractionally strided convolutions
- Discriminator is a conv net (real/fake classifier)

GAN Architectures

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." (ICLR, 2016) gave some practical tips that made a big difference to the performance of GANs:

- Replace pooling layers with strided convolutions (discriminator) and fractionally strided convolutions (generator)
- Use batchnorm both for the generator and discriminator
- Remove fully connected hidden layers for deep architectures
- Use ReLU activations for all layers of the generator except for the output, which uses Tanh
- Use LeakyReLU in the discriminator for all layers

Example of Generator Architecture

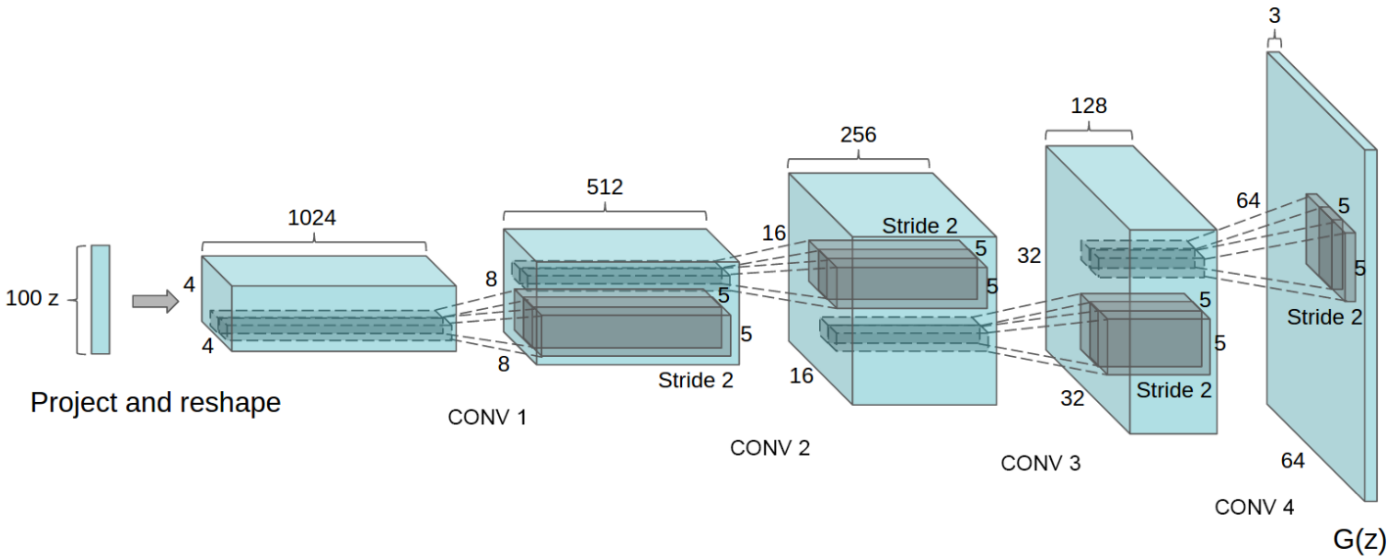

Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

Figure: Redford et al., 2016

Original Goodfellow et al. paper

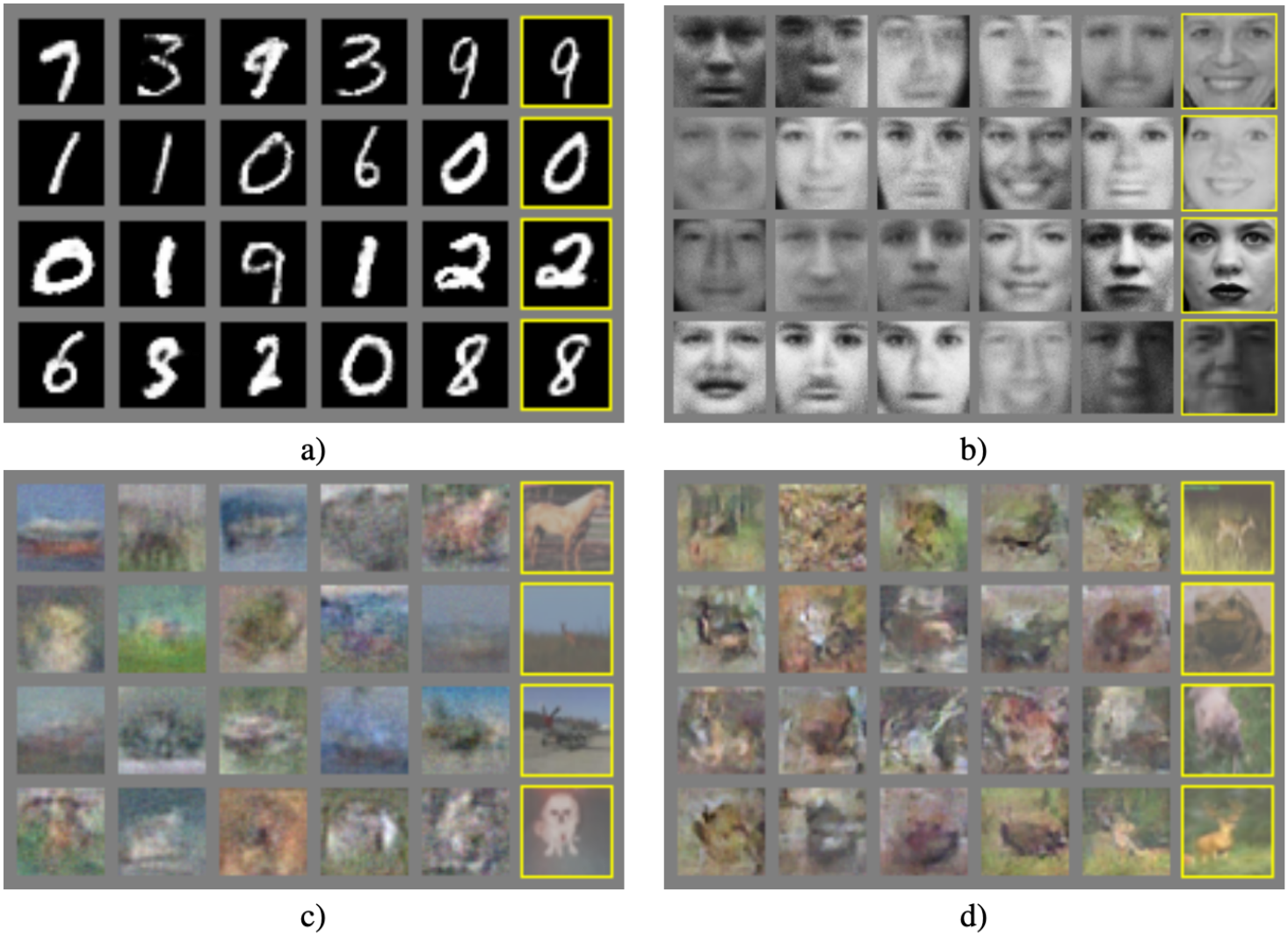

Figure: Goodfellow et al., 2014

Redford et al. paper

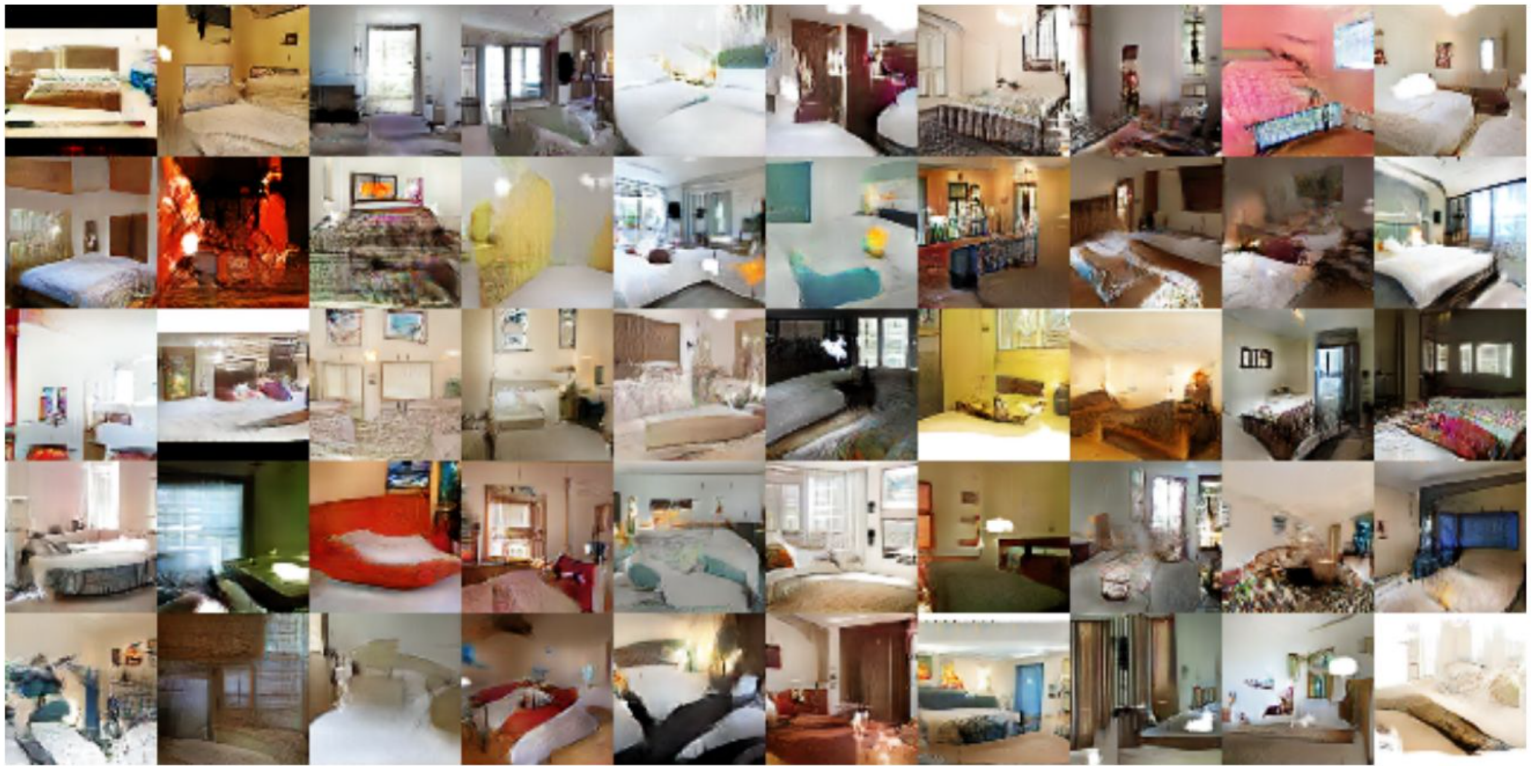

Figure: Redford et al., 2016

The Success of GANs

Figure: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

Outline

CycleGAN Overview

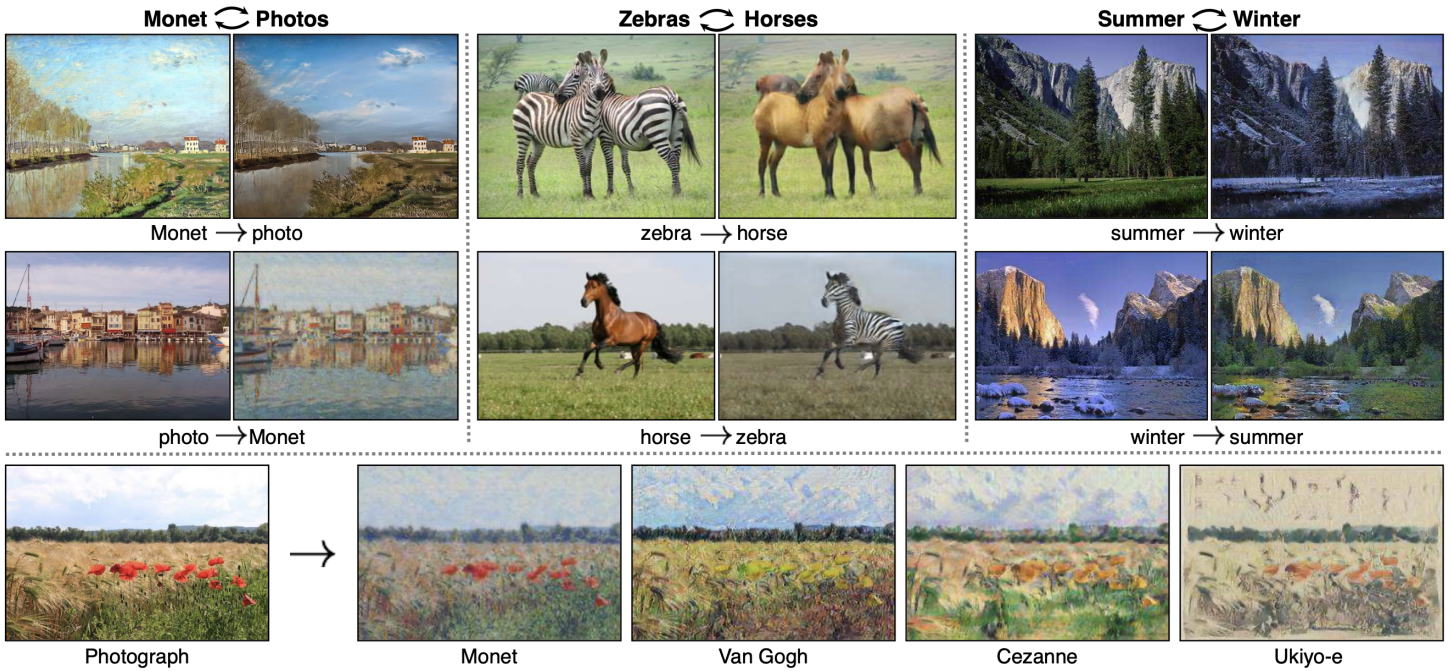

Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa. Example application (*bottom*): using a collection of paintings of a famous artist, learn to render a user's photograph into their style.

Figure: Zhu et al., 2017. CycleGAN is a model that can take an image from one domain and generate a synthetic version of the image with a specific modification (i.e. converting a horse to a zebra, changing the season, converting a painting into a photograph

CycleGAN Allows Unpaired Data

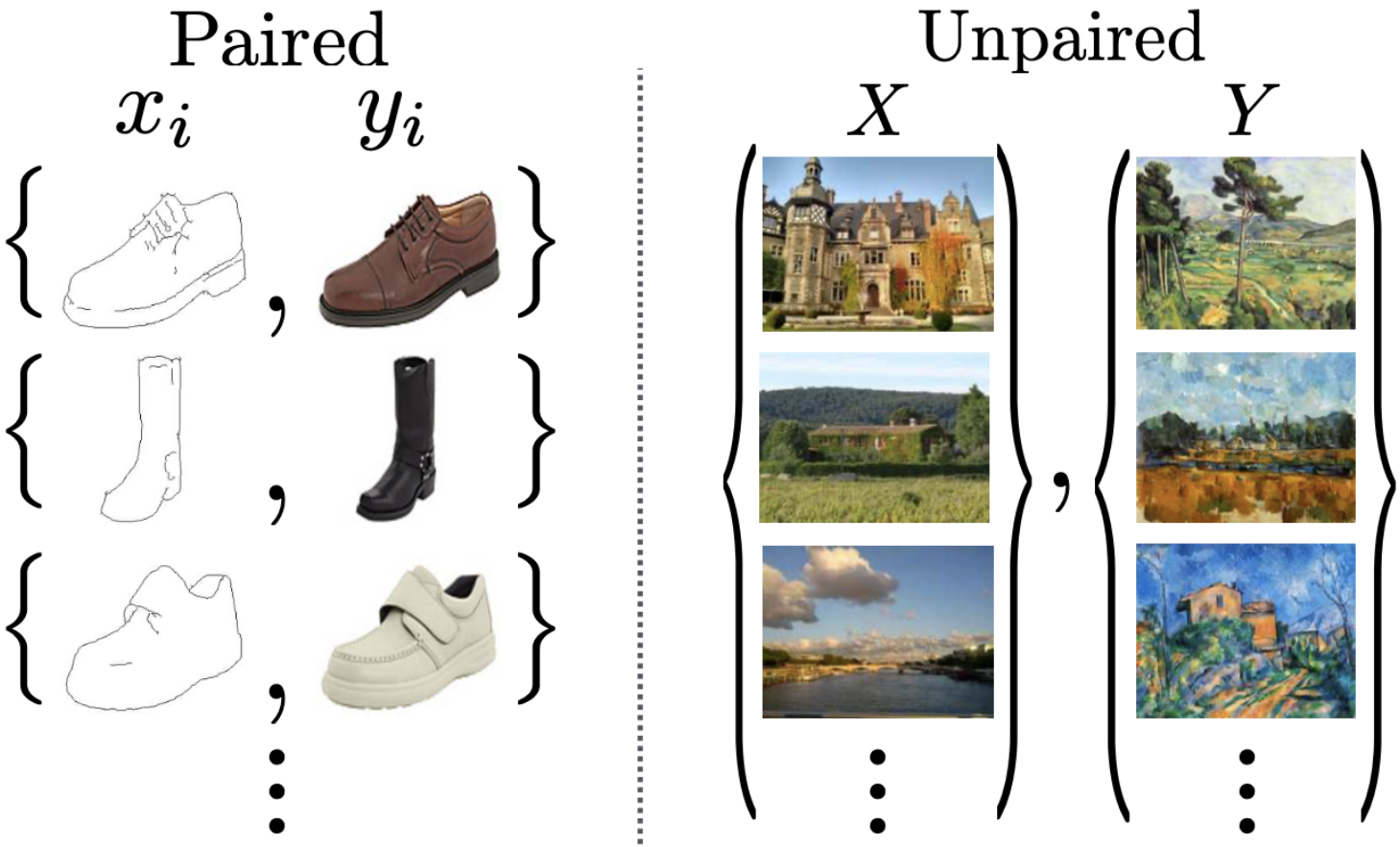

Figure: Zhu et al., 2017. Traditionally, image-to-image translation required paired datasets. These can be very costly, if not impossible, to collect. CycleGAN does not require images in the source and target domains to be paired.

CycleGAN Architecture

- CycleGAN simultaneously trains two generator models and two discriminator models.
- One generator takes images from the first domain and outputs images for the second domain, and the other generator takes images from the second domain as input and generates images for the first domain.

Cycle Consistency

- If you put an image output by the first generator into the second generator, it should return the original image. i.e. if you convert a horse into a zebra and then convert it back into a horse, the converted image should look similar to the original
- CycleGAN adds a loss term that measures this discrepancy, going in both directions
- Analogous to a method used for text translation between languages; can think of CycleGAN as a translator for images

CycleGAN Architecture

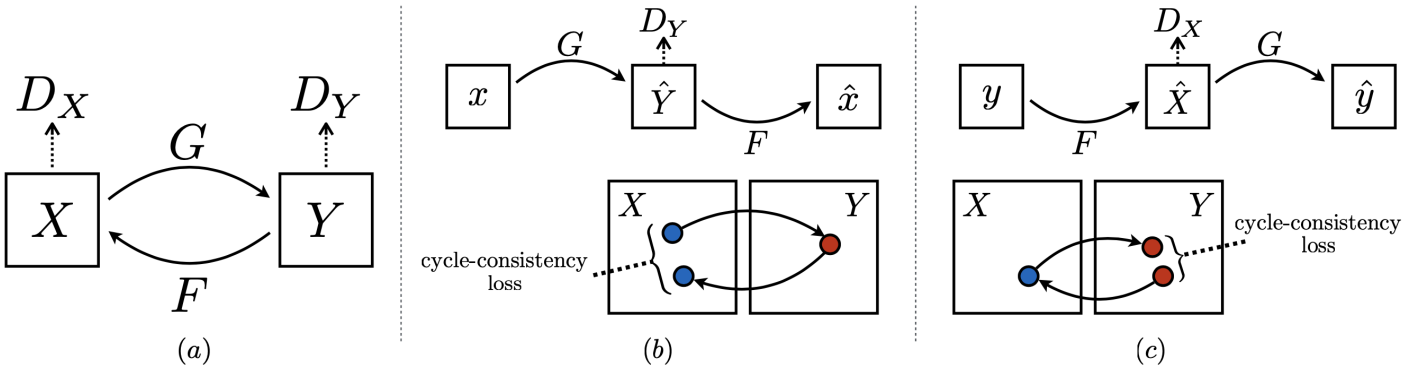

Figure: Zhu et al., 2017

CycleGAN Architecture

- The generators are deep convolutional GANs, implemented using multiple residual blocks
- The discriminators use PatchGAN (Isola et al., 2016). It tries to classify whether patches of the image are real or fake. It is run convolutionally across the image and then all responses are averaged
- The paper uses Adam and a low learning rate for 100 epochs, then an additional 100 epochs with learning rate decay
- The batch size is 1

CycleGAN Examples

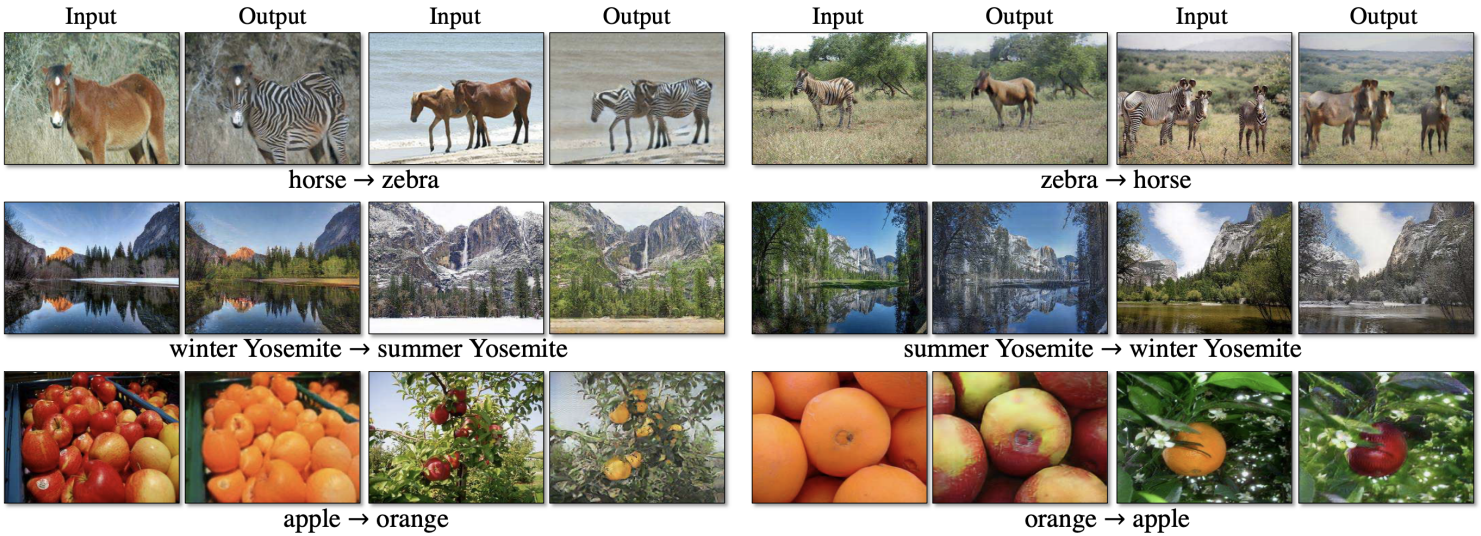

Figure: Zhu et al., 2017

CycleGAN and Style Transfer

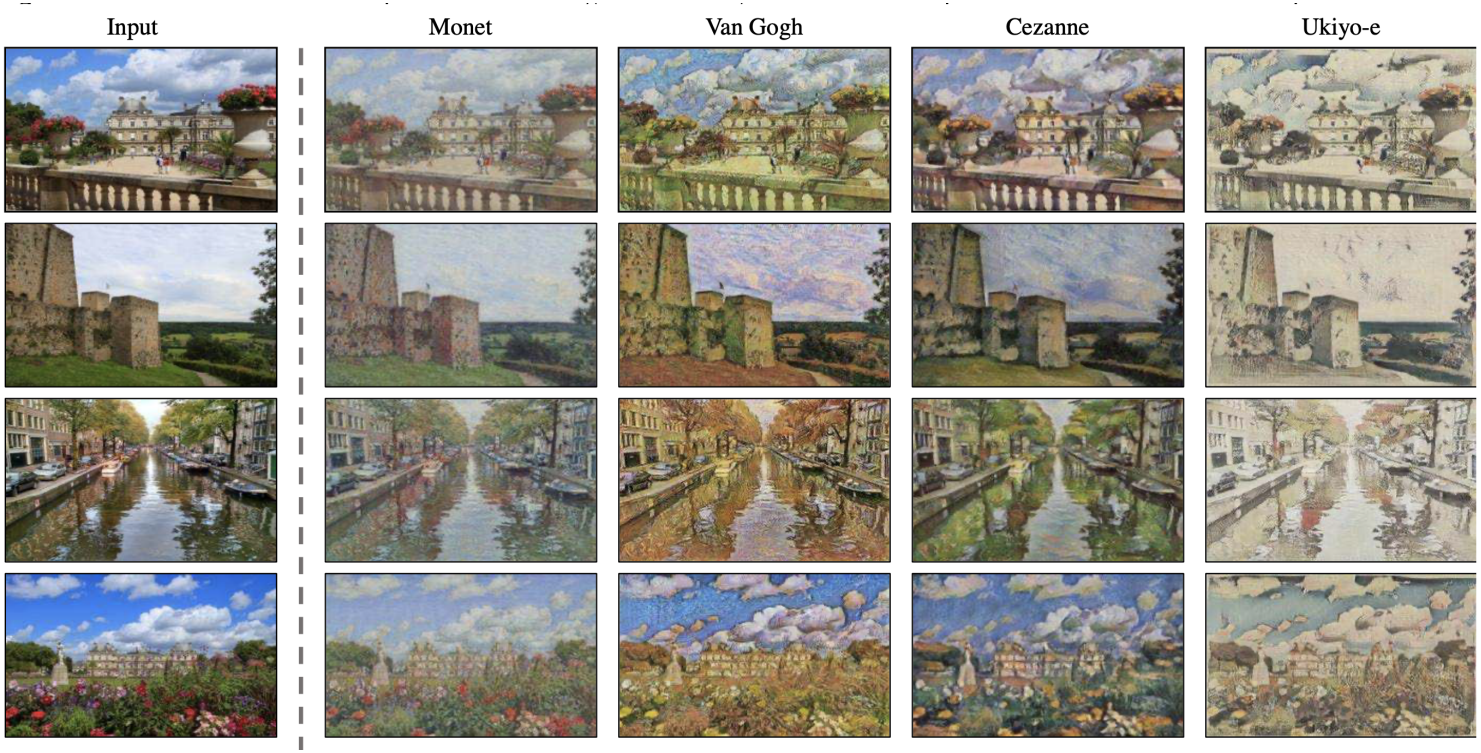

Figure: Zhu et al., 2017

Paintings to Photos

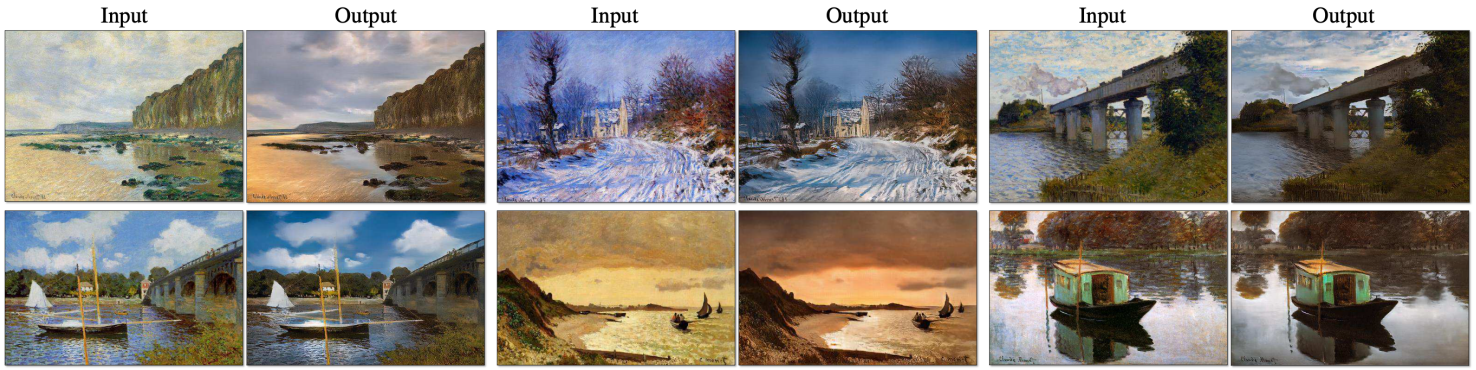

Figure: Zhu et al., 2017

Photo Editing

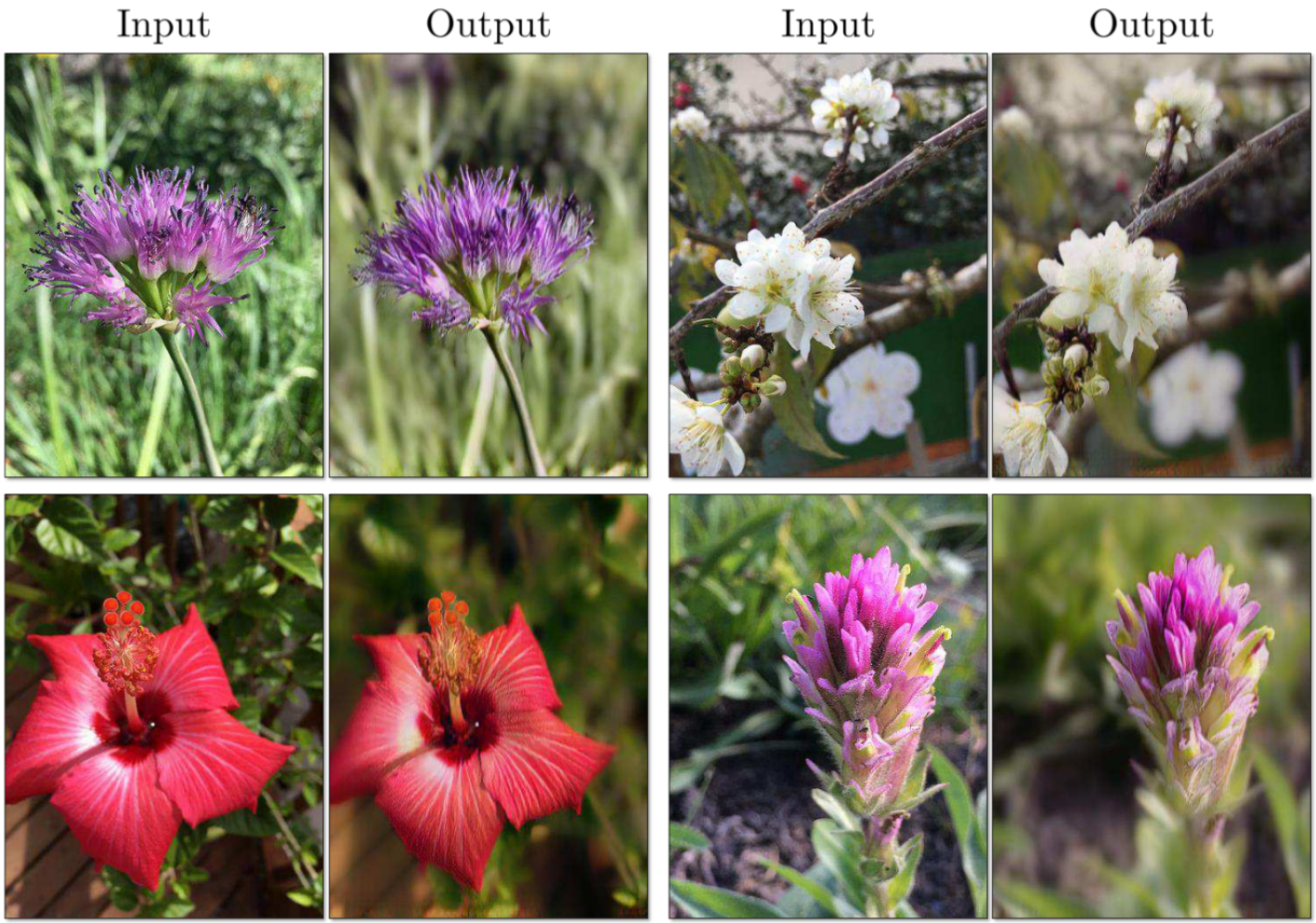

Figure: Zhu et al., 2017

Our Application: Removing Text Bleed

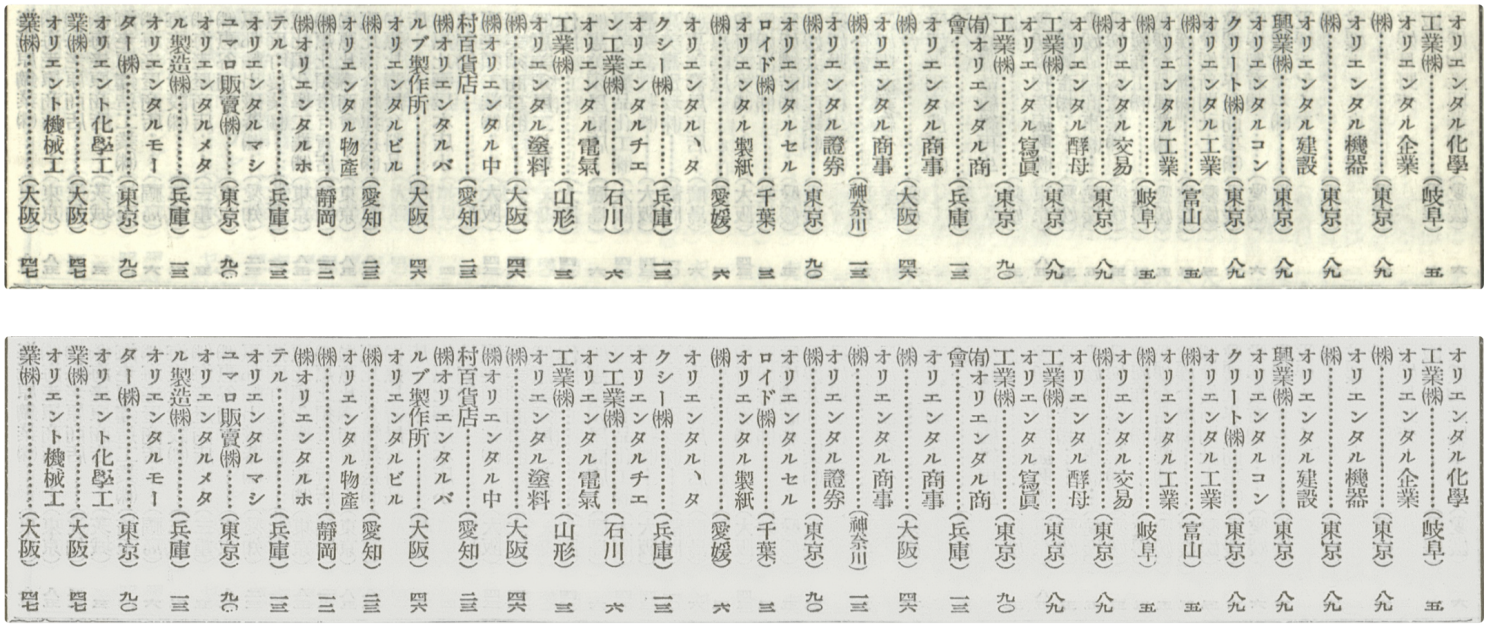

Figure: Two sets of images from the same publication (different years). One with a very clean scan and one with lots of text bleed. CycleGAN converts the top image into the bottom image. Rule-based thresholding methods will not achieve this.

Failure Due to Distributional Features of Training Data

horse → zebra

Figure: Zhu et al., 2017. The model was trained on horses and zebras in the wild.

Fails at Geometric Transformations

Figure: Zhu et al., 2017. CycleGAN often succeeds with color and texture changes, but tends to fail when geometric transformations are required. In the document context, we suspect it would have a hard time converting fonts but do pretty well changing their textures (useful for historical documents)