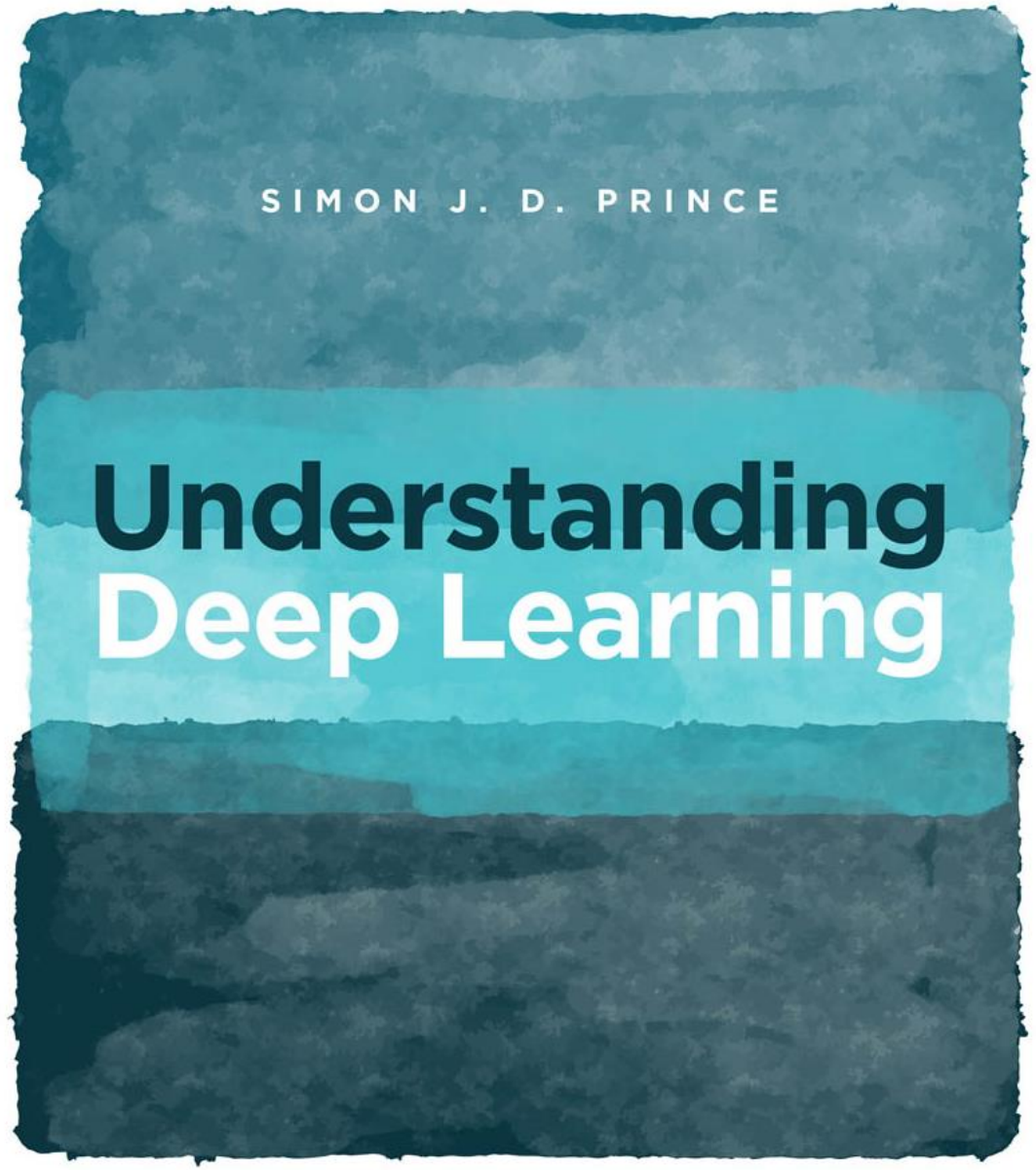

SIMON J. D. PRINCE
Understanding
Deep Learning

Understanding Deep Learning Generative Adversarial Networks (GANs)

April 27, 2024

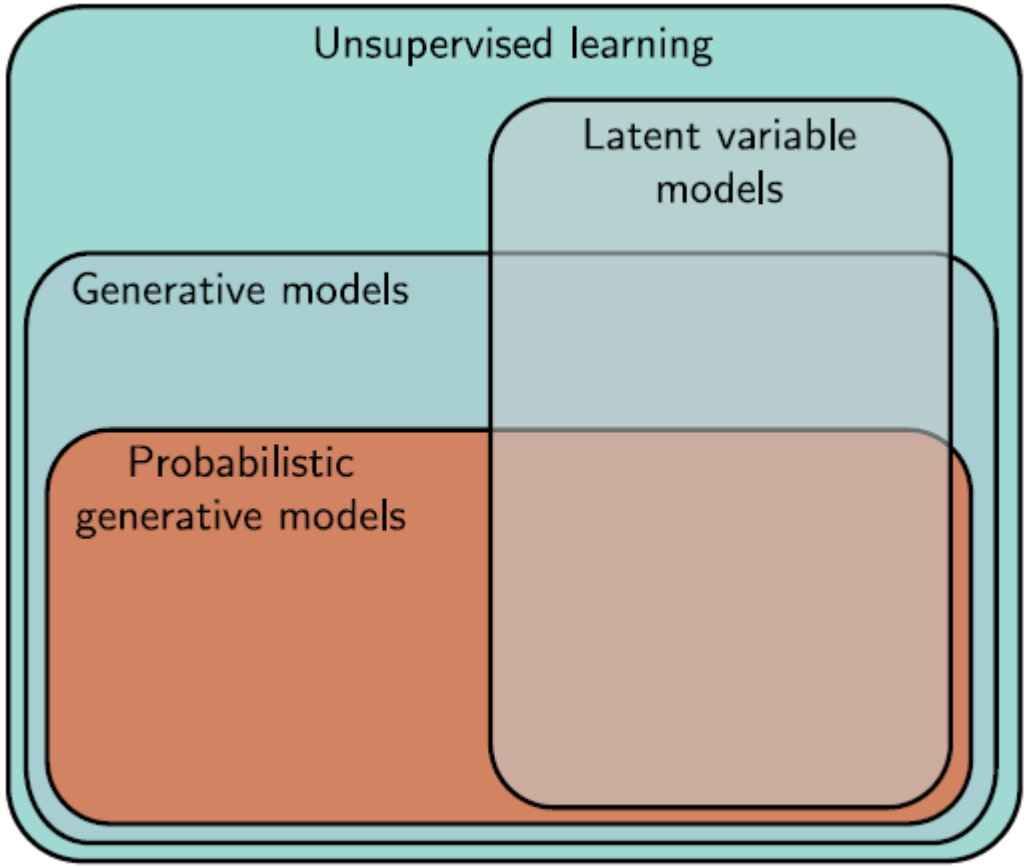
Chapter 14
Unsupervised Learning

Unsupervised learning

- *Unsupervised learning* models do not have labels
  - I differ from the author in that I consider *self-supervised* models to be a subset of supervised learning, because even though the labels weren't provided up front, the model was trained with labels which were crafted
- Unsupervised learning tasks include:
  - Reveal internal structure of the dataset, such as clustering
  - Dimensionality reduction
  - Denoising
  - Compression
  - Identifying outliers

Taxonomy of unsupervised learning models

- Some latent variable models map between data *x* and a latent *z*
- K-means clustering is an example that maps from *x* to *z*
- We will look at generative models that map from *z* to *x*
  - Four chapters, starting with GANs
- Some generative models are probabilistic



What makes a generative model good?

Model	Efficient	Sample quality	Coverage	Well-behaved latent space	Disentangled latent space	Efficient likelihood
GANs	✓	✓	✗	✓	?	n/a
VAEs	✓	✗	?	✓	?	✗
Flows	✓	✗	?	✓	?	✓
Diffusion	✗	✓	?	✗	✗	✗

Measuring generative model performance

- Test set likelihood
  - Intuitive, but only for probabilistic models (not GANs), and only if the likelihood calculation is inexpensive (only normalizing flows)
  - FYI, this is what perplexity measures for LLMs
- Inception score – compares distributions of same vs. different class (used for models trained on ImageNet)
- Fréchet inception distance (FID) – compares (Gaussian approx. of) distributions of real to generated using Inception bottleneck
- Manifold precision/recall – uses hyperspheres to approximate manifold of real and generated distributions

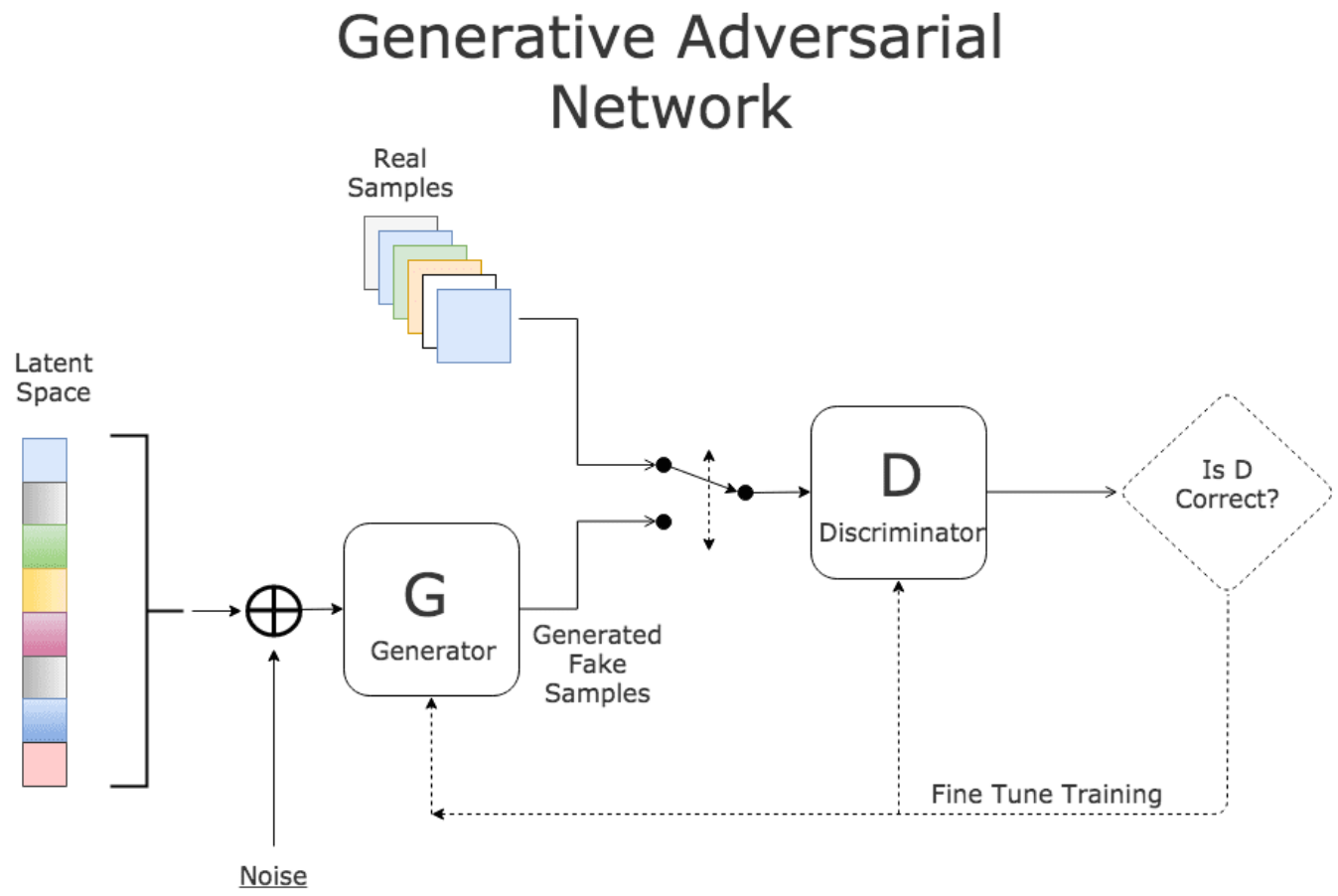
Chapter 15
Generative Adversarial Networks

GAN chapter contents

- GAN architecture and training
- GAN improvements
- Conditional generation
- Image translation and adversarial loss
- StyleGAN

GANs

- GANs have two models:
- Generator creates samples
- Discriminator tries to distinguish generated from real
- Training alternates
- Generator never gets feedback from real data!



https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html

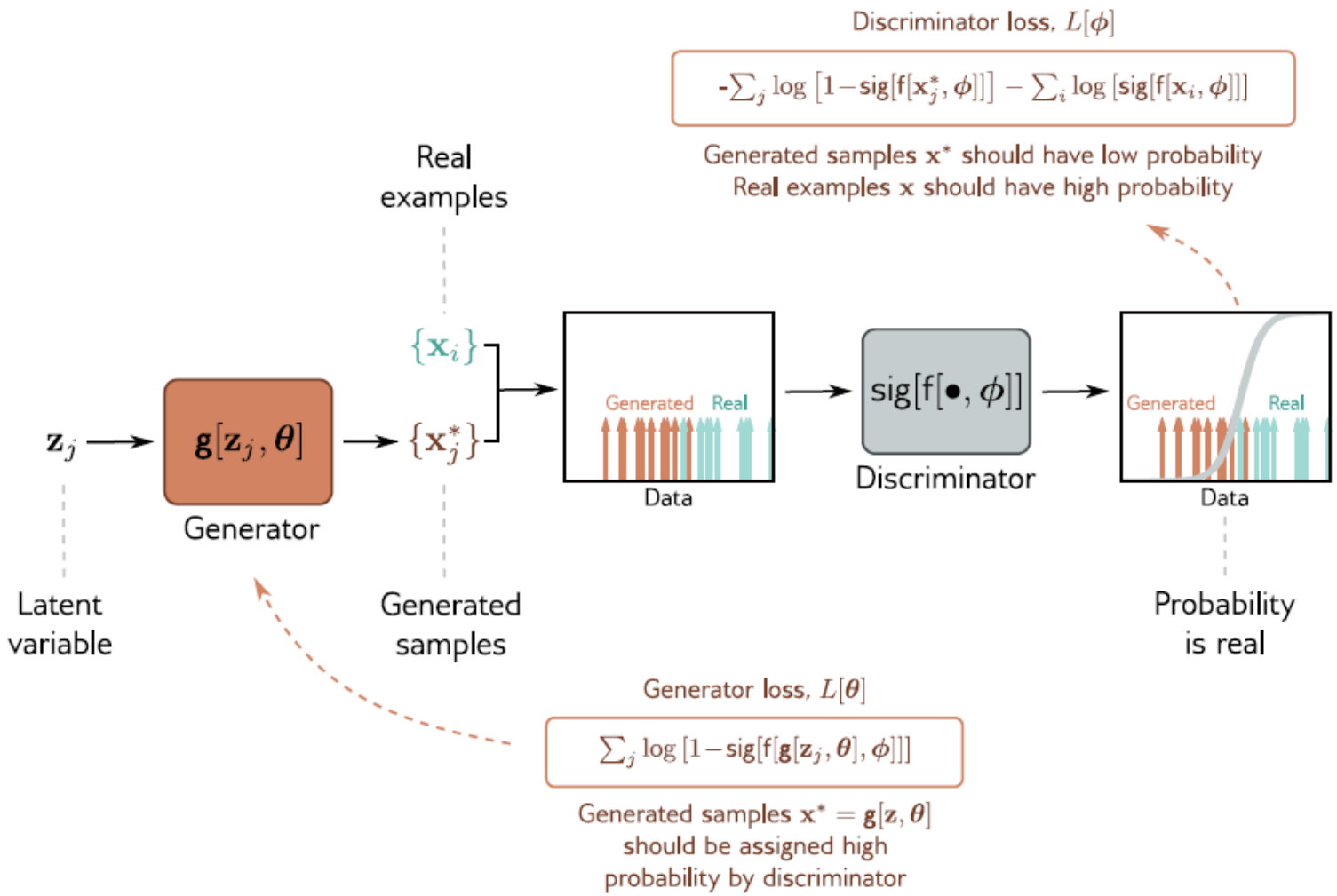
GAN loss function

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left[\min_{\boldsymbol{\phi}}\left[\sum_{j} -\log\left[1-\operatorname{sig}[\mathrm{f}[\mathbf{g}[\mathbf{z}_j,\boldsymbol{\theta}],\boldsymbol{\phi}]]\right] - \sum_{i}\log\left[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i,\boldsymbol{\phi}]]\right]\right]\right]. \quad (15.4)$$

- There are two parts to the loss function, from binary cross-entropy:
  - Left side represents loss for identifying generated samples as fake
  - Right side represents loss for identifying real samples as real
- The discriminator, f(), updates parameters ϕ to minimize both losses
- The generator, g(), updates parameters θ to maximize the left side
- If generator is so good that discriminator can do no better than guessing, then generated distribution will match real distribution

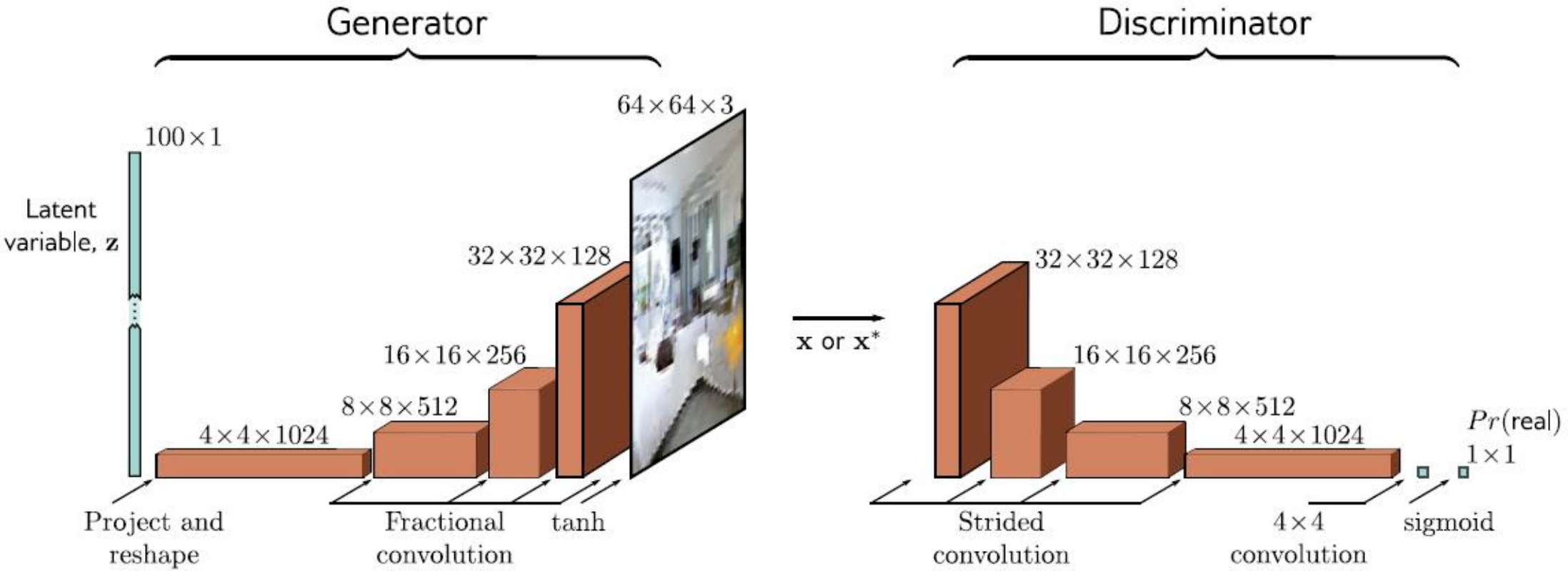
Training GANs

- We can separate out the loss functions for the generator and the discriminator
- Training usually alternates
- Typically, give the discriminator a head start



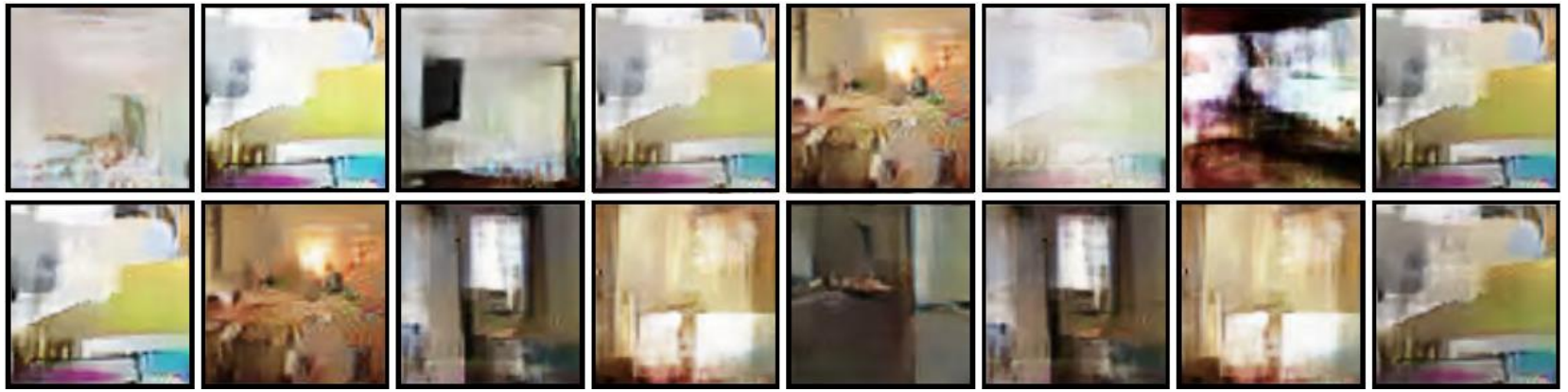
DCGAN

- DCGAN used CNN architecture for both generator and discriminator



Well known difficulty training GANs [1]

- Finicky architectural and hyperparameter choices in GAN recipes are not normally that sensitive for other NN models
- Sometimes a trained model will only generate a subset of possible samples, known as *mode dropping*
- Sometimes the generator and discriminator will fall into a poor Nash equilibrium where all generations are one or a few samples, known as *mode collapse*



Well known difficulty training GANs [2]

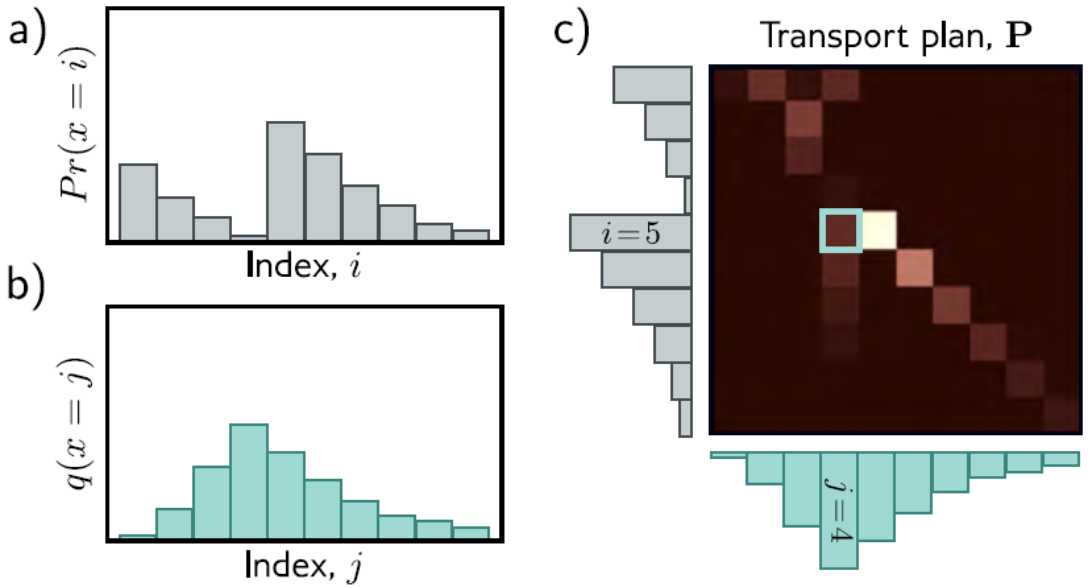
- Part of the sensitivity of GAN training is due to the nature of the generator getting its training signal via its pixels, based on the discriminator's parameters
- The generator's loss depends upon backpropagation using the gradient of the loss w.r.t. the discriminator's parameters
- If the discriminator is too poor at discriminating fake from real, the direction of this gradient will be poor, and the generator is not going to get a good training signal
- However, if the discriminator is too good, the direction of the gradient may be accurate, but the magnitude may be too small – the vanishing gradient problem

Well known difficulty training GANs [3]

- Analyzing the loss function, we can see that it is (disregarding some constants) the *Jensen-Shannon divergence* between the generated distribution and the real data distribution
  - The Jensen-Shannon divergence is average of the *Kullback-Leibler divergence* taken in both directions (KL divergence is not symmetric)
  - These are measures of the difference between two probability distributions
  - One direction measures the quality of match, penalizing low real probability where there are generated samples
  - The other direction measures the coverage of generation, penalizing low generated probability where there are real samples
  - Unfortunately, this second term does not depend on the generator, so there is nothing pushing the generator to improve coverage

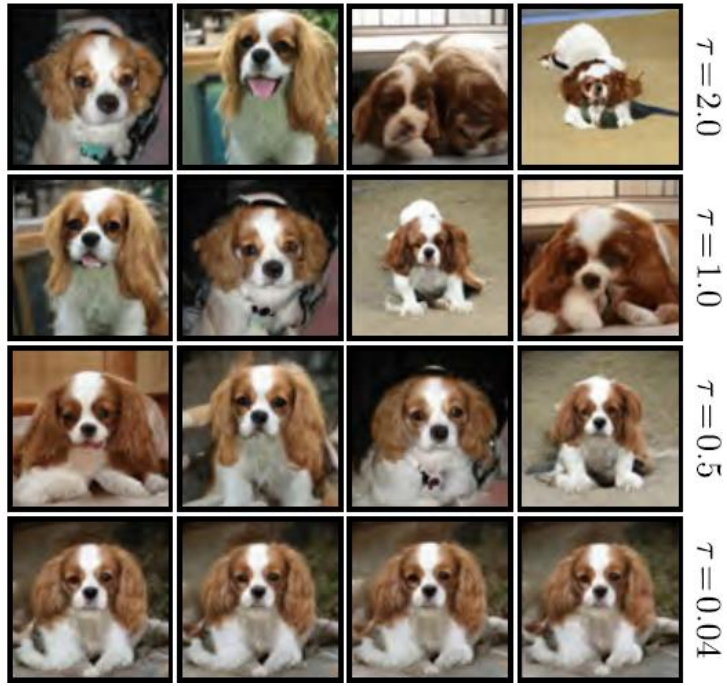
Wasserstein GAN (WGAN)

- A better metric for difference between probability distributions is the *Wasserstein distance*, also known as *earth mover's distance*
- The Wasserstein distance more accurately measures how far apart two distributions are even when either has zero probability in spots
- WGAN modifies the loss function to use the Wasserstein distance
- This improves training stability of GANs



Additional GAN quality improvements

- It's hard to generate high resolution images directly, so a technique used for GANs (and other model types) is progressive growing
  - Generate a smaller image, then add a few upsampling steps
- Mini-batch discrimination encourages variety
  - Feature statistics are calculated for different samples in the batch
  - This allows a variety signal to the generator
- Truncation disallows random latents far from the mean
  - Reduces diversity but helps quality



Conditional generation

- A vanilla GAN will generate a sample from any part of the data distribution
- Conditional generation allows user to specify characteristics
- Conditional GAN gives *c* to both generator and discriminator
- ACGAN gives *class* to generator and discriminator predicts *class*
- InfoGAN gives *c* to generator and discriminator estimates values of *c*

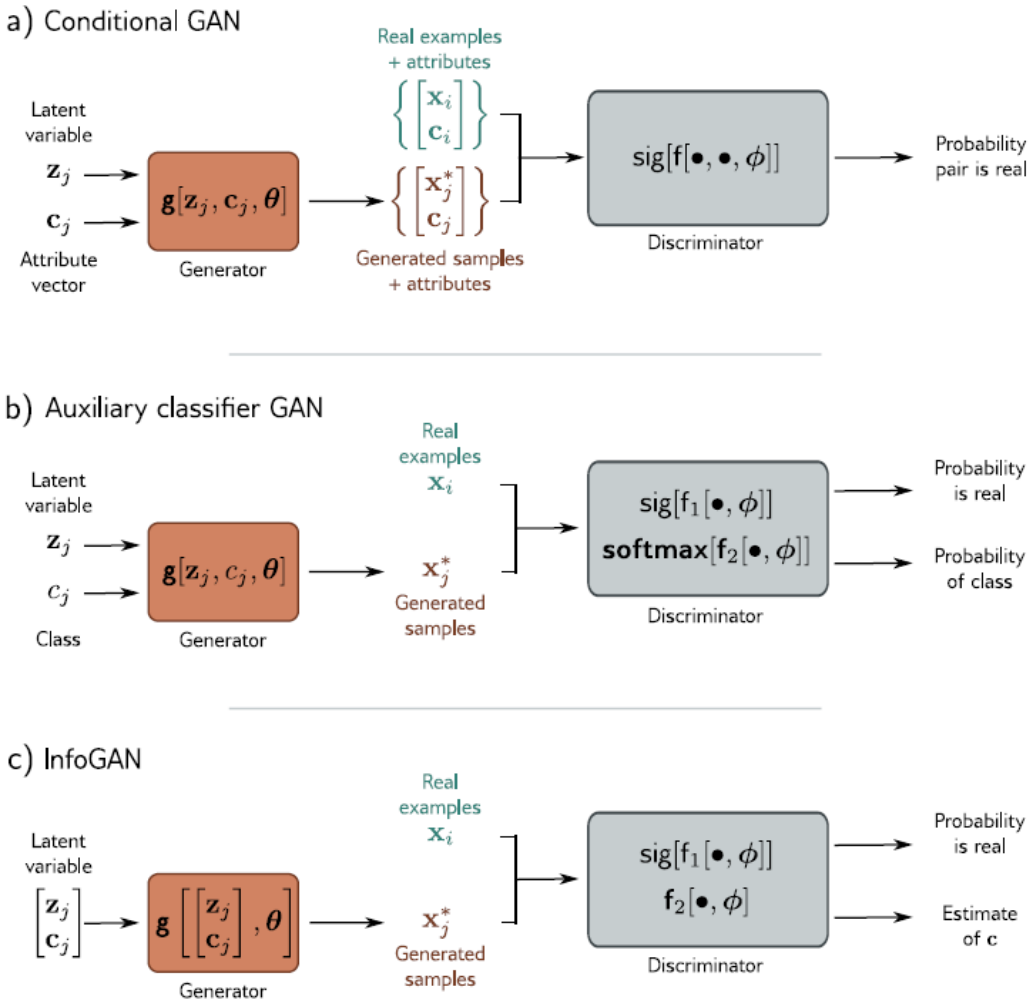


Image translation – Pix2Pix [1]

- A discriminator can be used in ways other than the GAN setup
- Pix2Pix trains on before/after pairs, so it can convert an image into another image, such as turning a grayscale image into a color image
- The first image goes into a U-Net which converts to second format
- The loss has two components:
  - L1 loss between ground truth and converted encourages structural similarity
  - A *PatchGAN* discriminator is applied patch-wise across the entire generated image, indicating if it thinks each patch is realistic

Image translation – Pix2Pix [2]

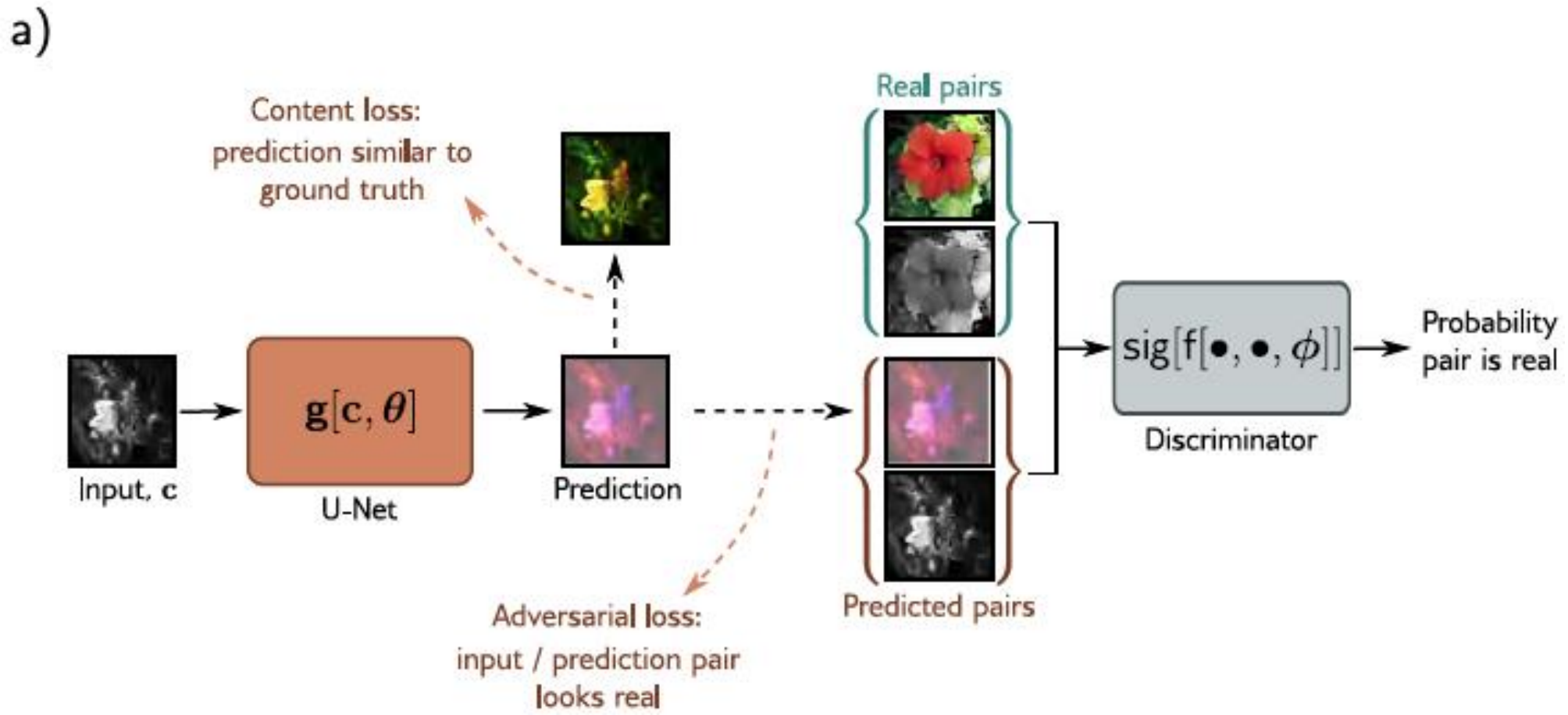
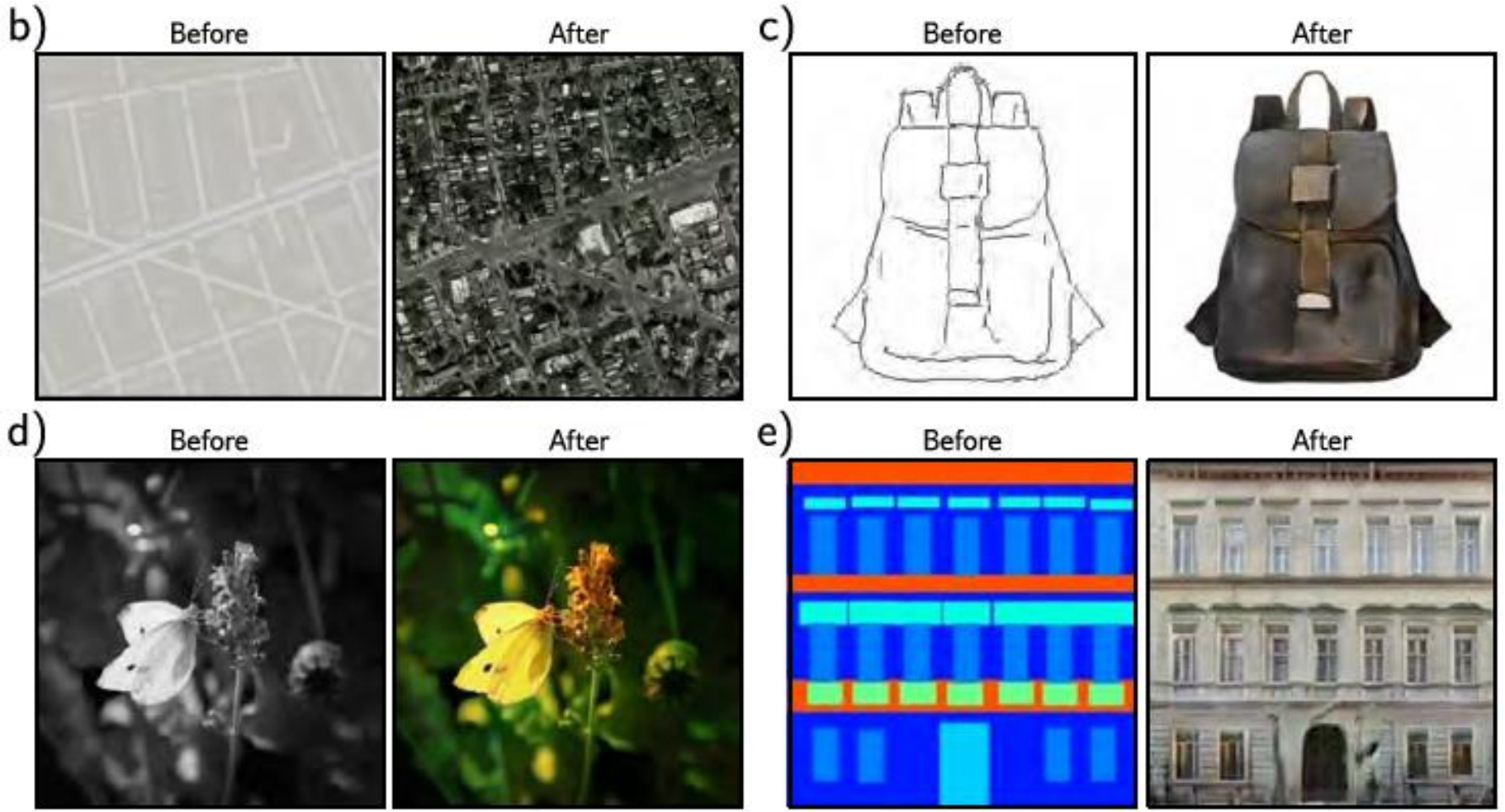


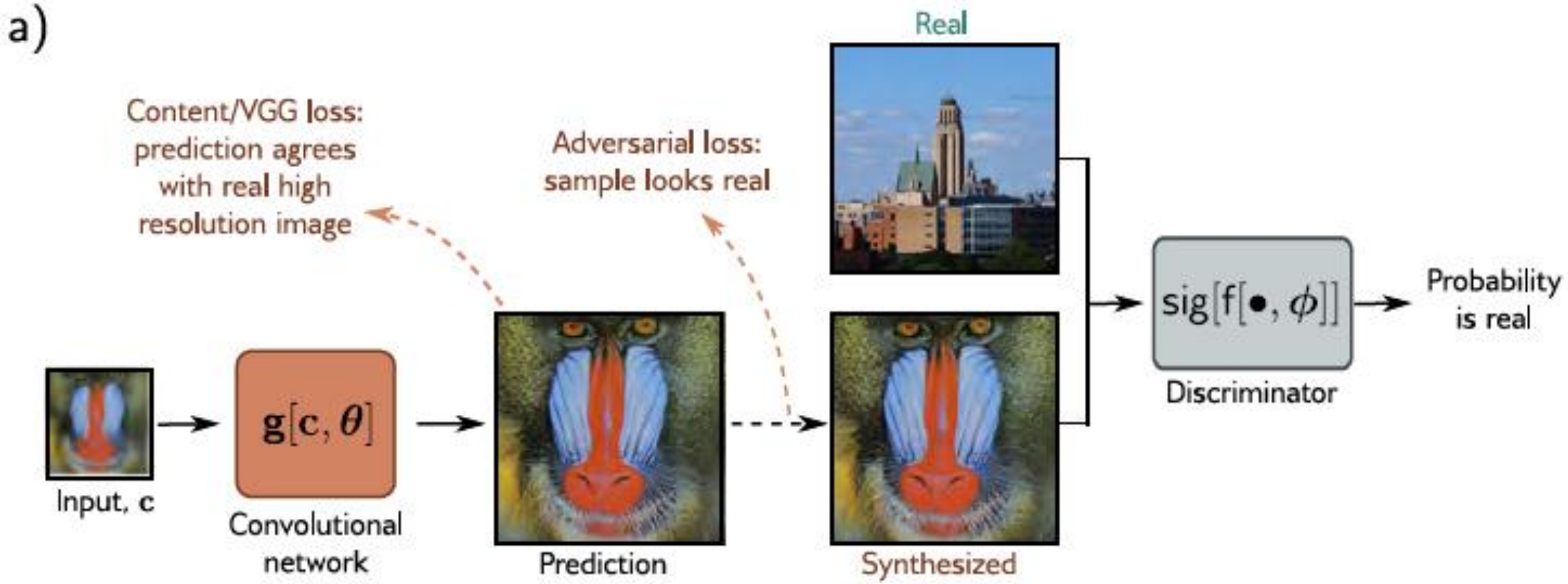
Image translation – Pix2Pix [3]



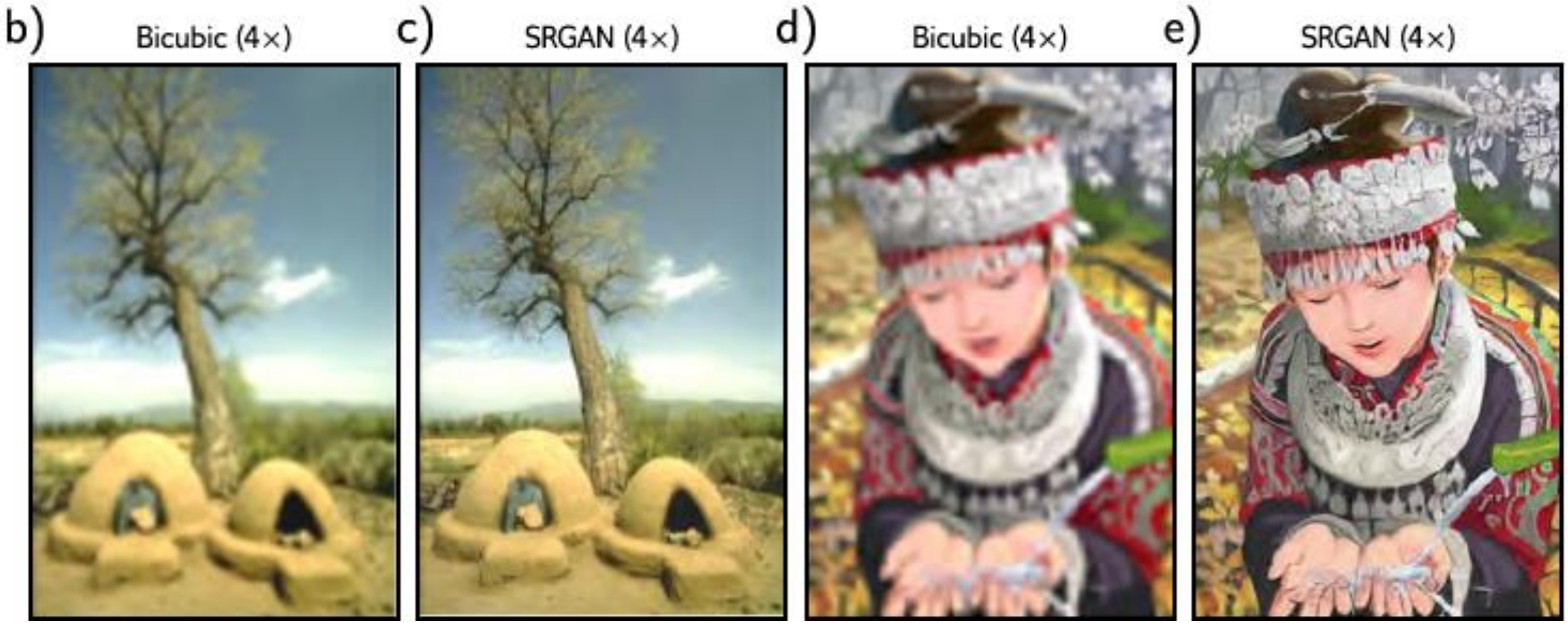
Super-Resolution GAN (SRGAN) [1]

- SRGAN creates a higher resolution version of an image
- Main model is convolutional upsampling layers
- The loss has three components:
  - L2 loss between upsampled image and ground truth high-resolution image
  - A *VGG loss*, also called *perceptual loss*, looks at L2 loss of the VGG bottleneck. This encourages semantic similarity
  - An *adversarial loss* comes from a discriminator that predicts if the image is a real high-resolution image or one that was upsampled from low resolution
- Upsampled images tend to be blurry, but SRGAN looks sharper

Super-Resolution GAN (SRGAN) [2]



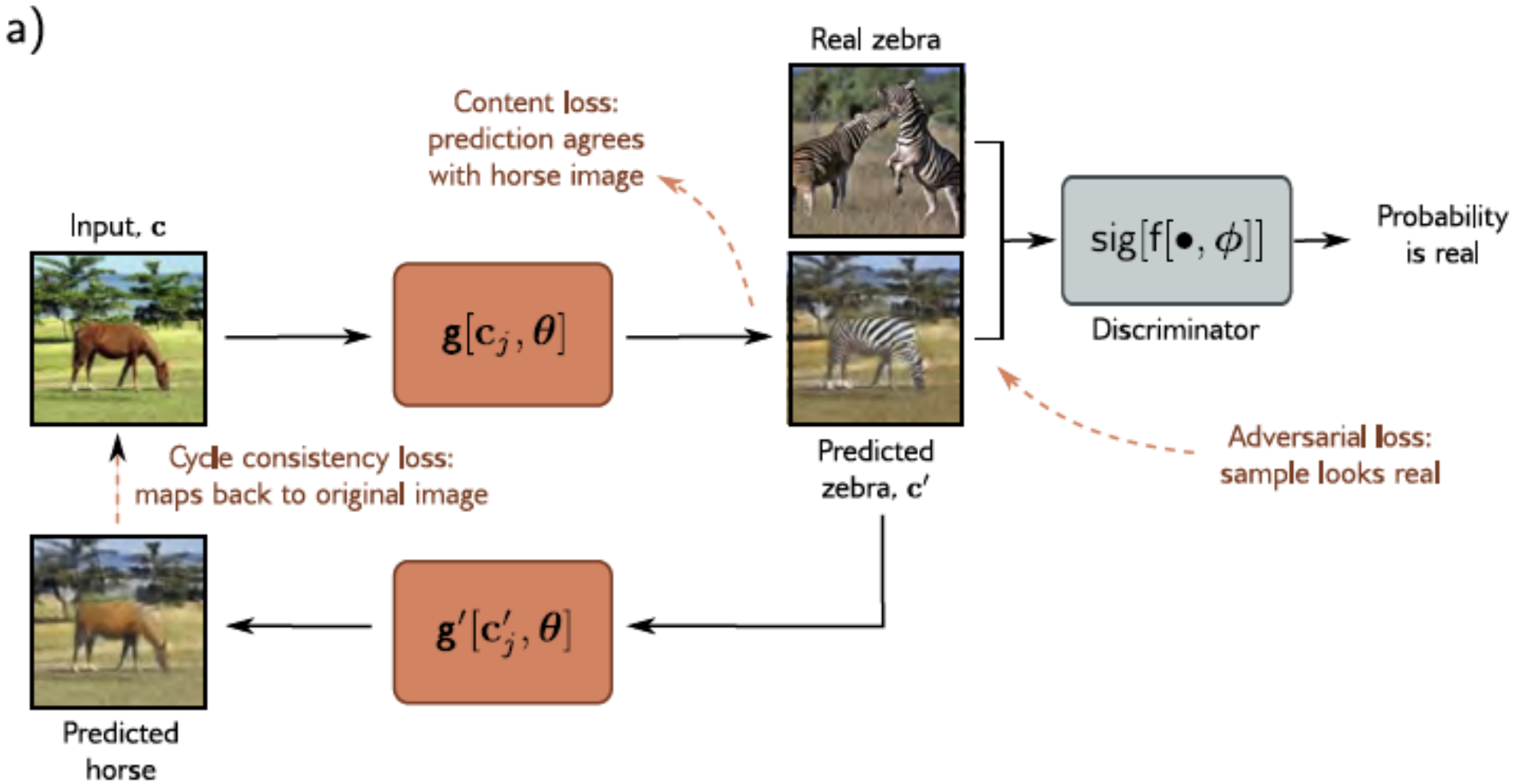
Super-Resolution GAN (SRGAN) [3]



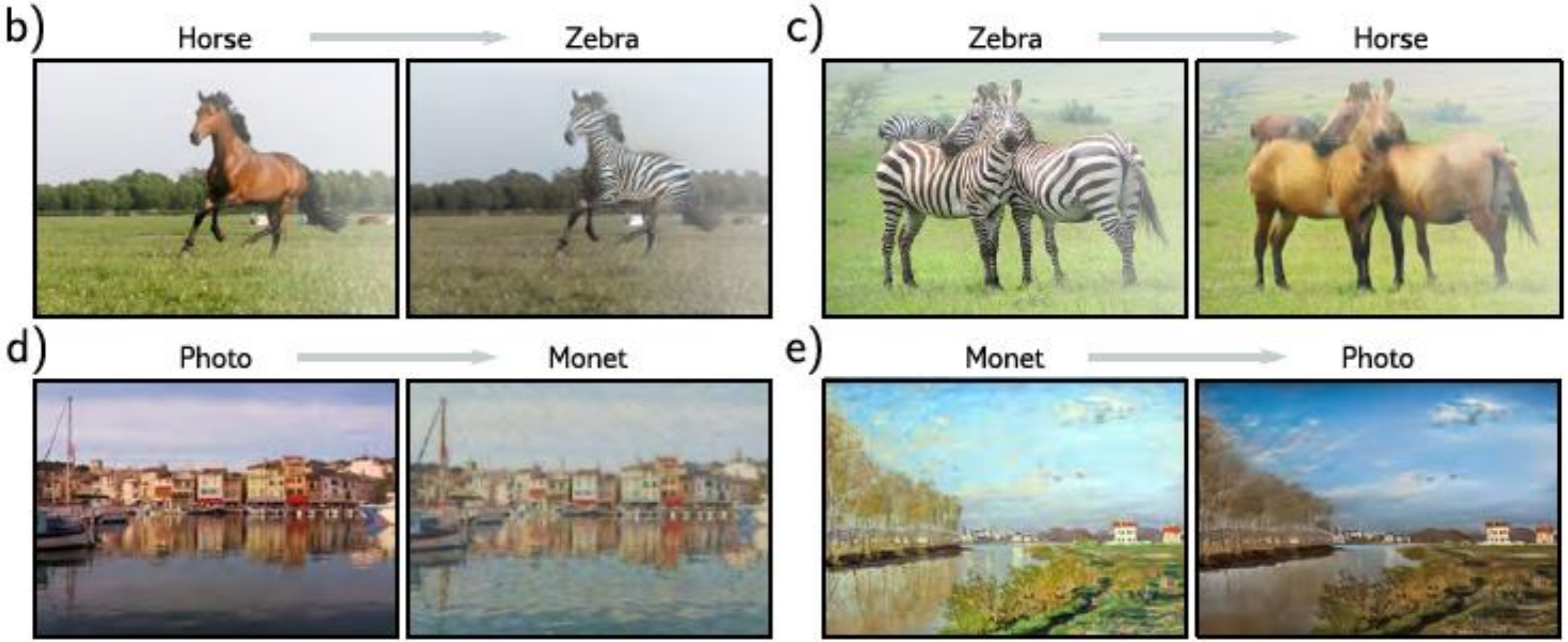
CycleGAN [1]

- CycleGAN does image translation without needing before/after pairs
- Uses the property that converting an image in one direction (e.g., convert a photo to Monet painting) should be reversible
- There are two models, one for each conversion direction
- The loss has three components:
  - L1 loss between the original and the converted image for content similarity
  - An adversarial loss uses a discriminator that predicts if the image is a real image from the domain (e.g., Monet) or a converted image
  - The *cycle-consistency loss* is the loss between original and image converted forward and then back again

CycleGAN [2]


a)
Real zebra
Content loss:
prediction agrees
with horse image
Input, c
$\mathbf{g}[\mathbf{c}_j, \theta]$
$\text{sig}[f[\bullet, \phi]]$
Probability
is real
Discriminator
Predicted
zebra, c′
Adversarial loss:
sample looks real
Cycle consistency loss:
maps back to original image
$\mathbf{g}'[\mathbf{c}'_j, \theta]$
Predicted
horse

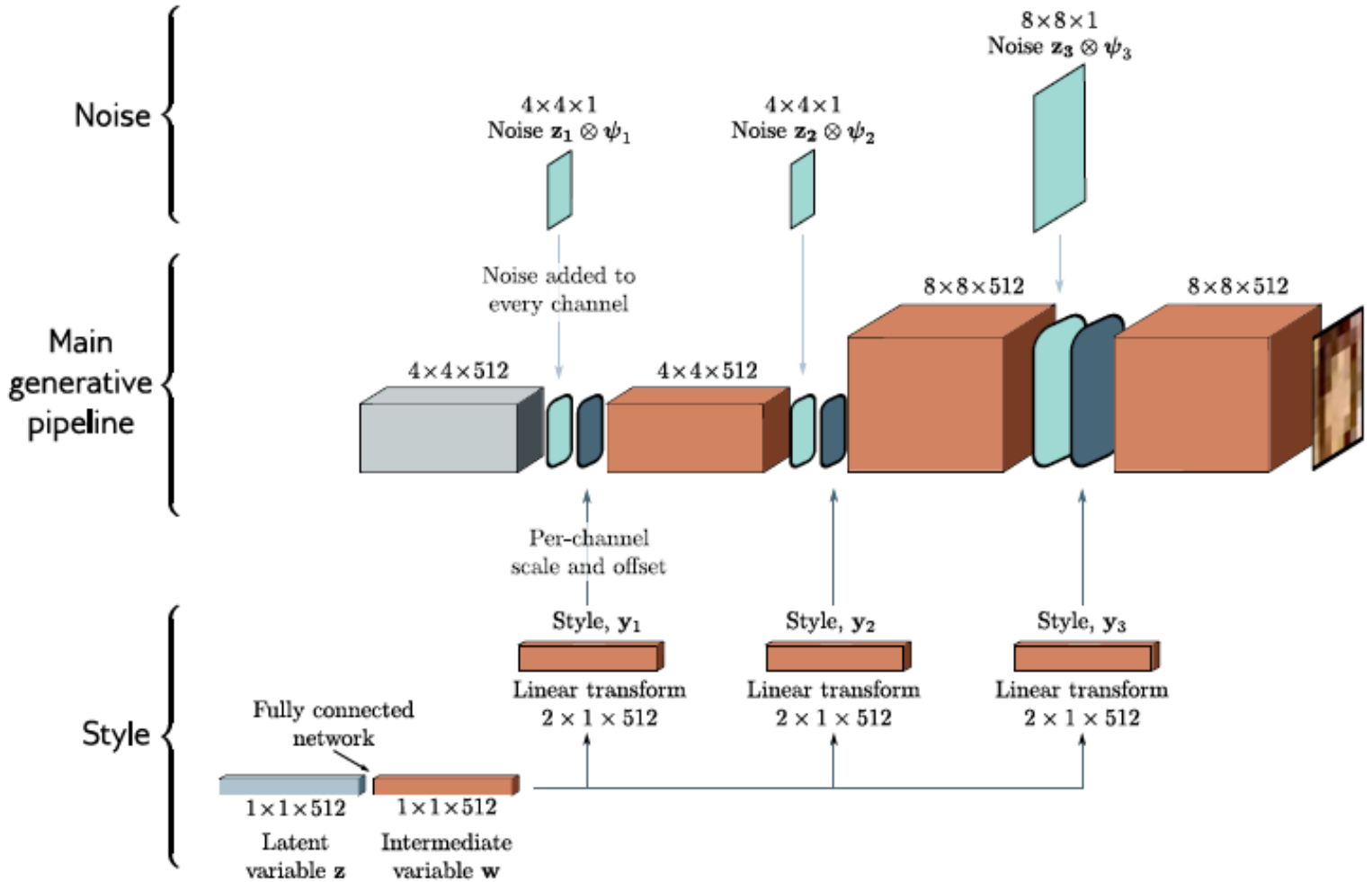
CycleGAN [3]



StyleGAN [1]

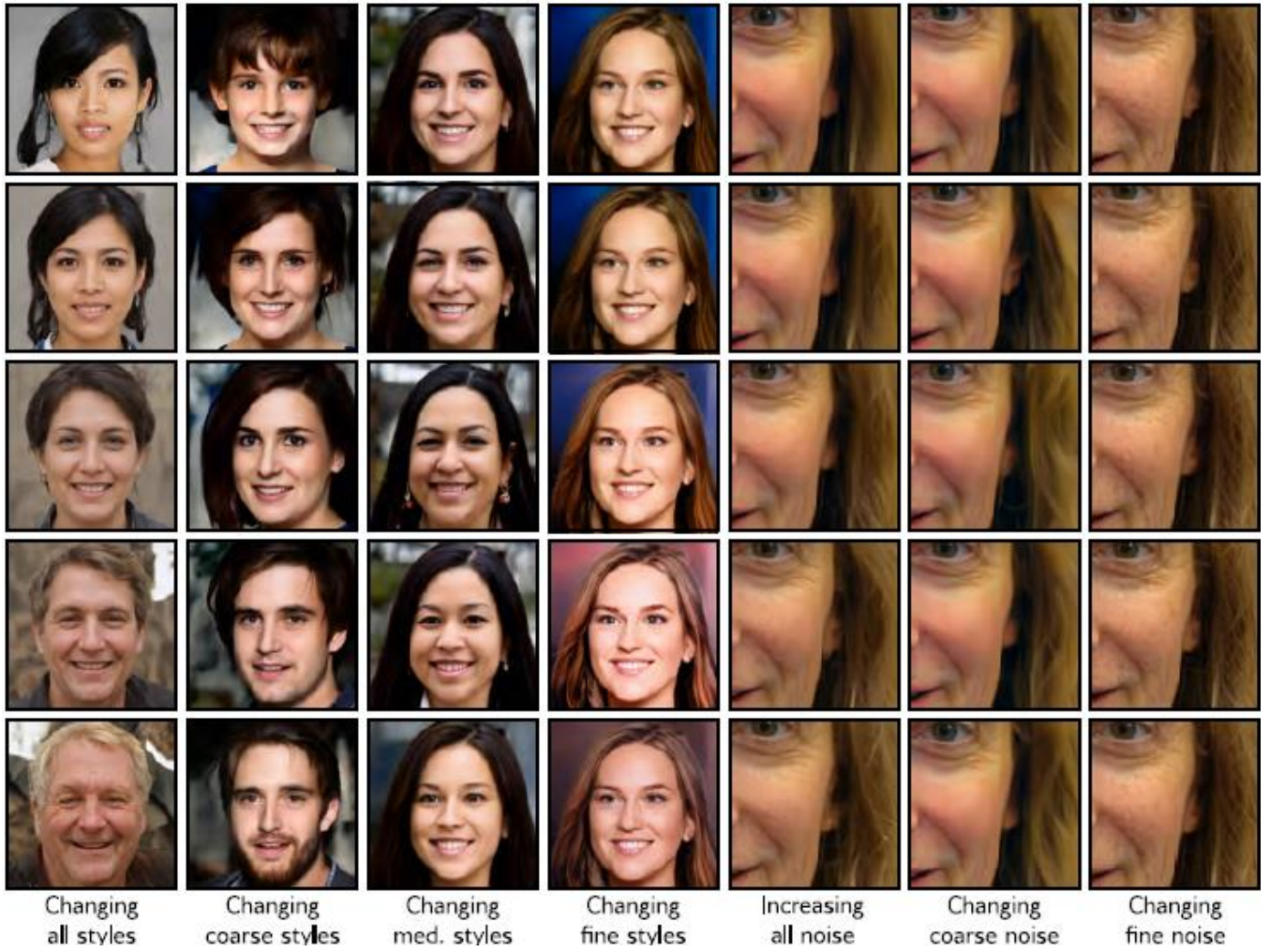
- StyleGAN successfully divided the latent variable space into controllable human-interpretable components
  - For faces, large-scale features include face shape and head pose
    Medium-scale features include the shape of individual features, like the nose
    Fine-scale features include hair and skin details
  - The random/noise component affects unimportant variation such as the exact placement of hairs or freckles
- The architecture is very different
  - Uses progressive growing, starting with low resolution and working up
  - Instead of starting with noise, noise is added at each stage
  - A deep MLP generates latents that control features at each stage
- Additional training tricks help StyleGAN learn interpretable features

StyleGAN [2]


Noise
Main generative pipeline
Style
4×4×1
Noise $\mathbf{z_1} \otimes \psi_1$
4×4×1
Noise $\mathbf{z_2} \otimes \psi_2$
8×8×1
Noise $\mathbf{z_3} \otimes \psi_3$
Noise added to every channel
4×4×512
4×4×512
8×8×512
8×8×512
Per-channel scale and offset
Style, $\mathbf{y}_1$
Style, $\mathbf{y}_2$
Style, $\mathbf{y}_3$
Linear transform 2 × 1 × 512
Linear transform 2 × 1 × 512
Linear transform 2 × 1 × 512
Fully connected network
1×1×512
Latent variable z
1×1×512
Intermediate variable w

StyleGAN [3]

- First four columns show the effect of changing styles at different levels from coarse to fine
- Last three columns show the effect if increasing the amount of noise at different levels from coarse to fine



GAN summary

- We learned about GANs which combine a generator and discriminator that compete, forcing the generator to learn how to generate realistic samples that match the training distribution
- Discussed various improvements and upgrades, including the Wasserstein GAN and conditional GAN architectures
- While we have seen many example of GANs that generate images, GANs have been used to generate every kind of data
- The idea of an adversarial loss has now been used in many different deep learning models, some having nothing to do with GANs
- Note that editing an image requires knowing the latent for it. Much work has been done on *GAN inversion* with mixed results