Autoencoders and GANs

Semester 1, 2021

Kris Ehinger

Outline

- Background: Generative models
- Autoencoders
- Generative Adversarial Networks (GANs)
- Evaluating GANs

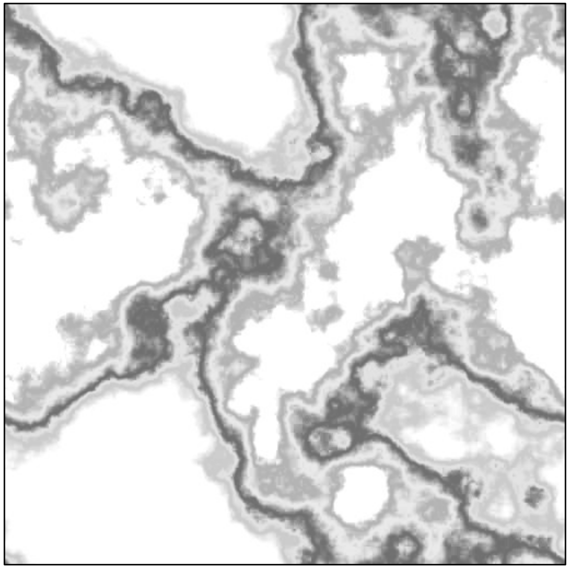
Discriminative vs. Generative

- **Discriminative** models
  - Learn conditional probability of class Y given attributes X: $P(Y|X=x)$
- **Generative** models
  - Learn joint probability of attributes X and class Y: $P(X,Y)$
- Generative model contains discriminative model: you can use the joint probability to get $P(Y|X=x)$
- AND generative can do the reverse: $P(X|Y=y)$

generate new data

Generative models

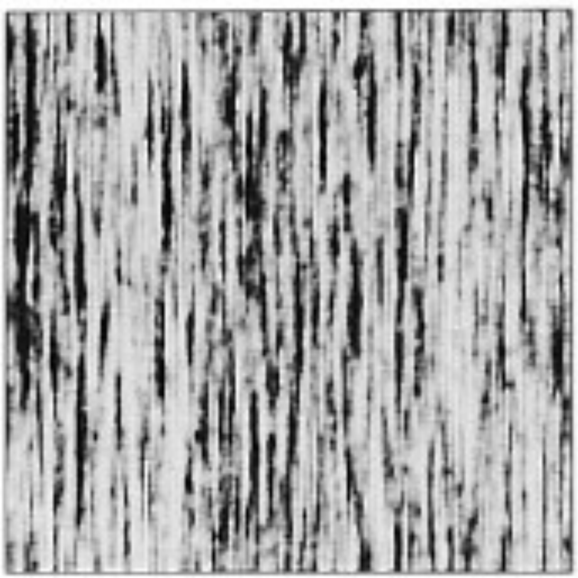
- That means you can generate new samples from the learned distribution



"Marble"
Procedural texture algorithm

Generative models

- That means you can generate new samples from the learned distribution

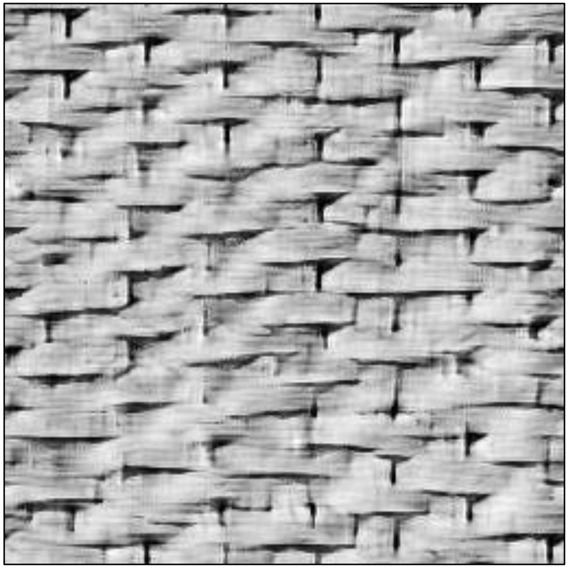


"Wood"
Gaussian mixture model

Image: Portilla, Navarro, Nestares, & Tabernero (1996)

Generative models

- That means you can generate new samples from the learned distribution



"Basket"
More complex model of probability distributions, more features

Portilla & Simoncelli (2000)
https://www.cns.nyu.edu/~lcv/texture/

Generative models

- That means you can generate new samples from the learned distribution

"Face"
More complex model of probability distributions, more features

Portilla & Simoncelli (2000)
https://www.cns.nyu.edu/~lcv/texture/

Generative models of images

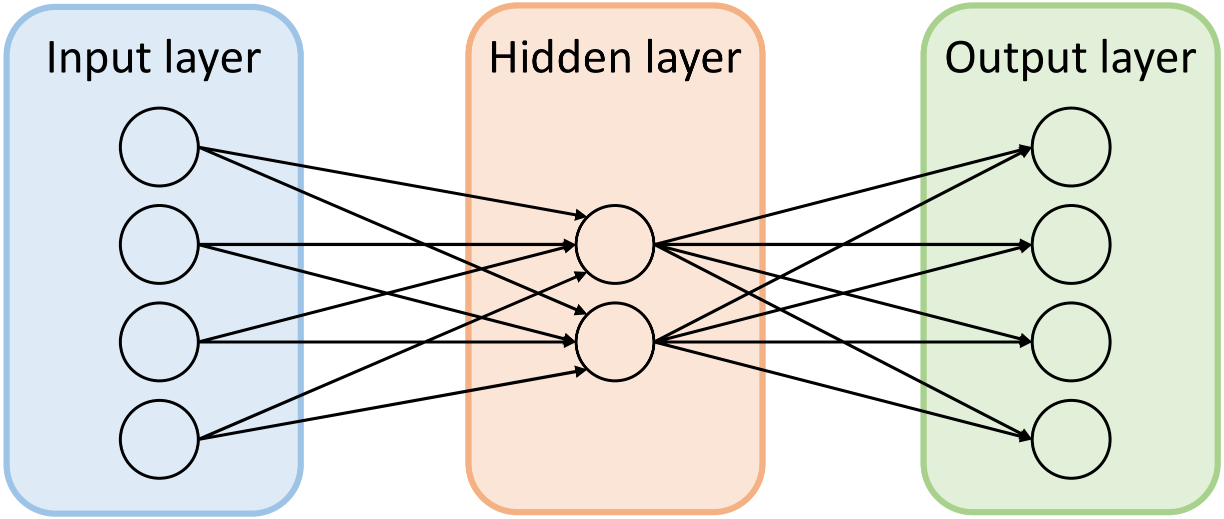
- Building a generative model of images is difficult!
- How to generate objects, faces, scenes?

Autoencoders

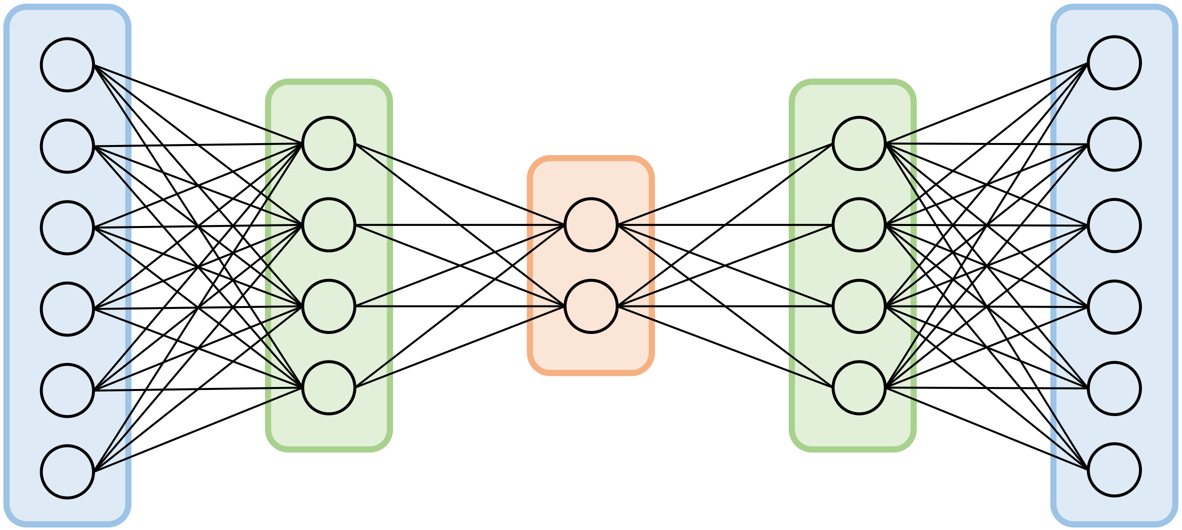
Autoencoders

- Essentially, neural networks for unsupervised learning
- Output of the network is whatever was passed to the network (e.g., an image)
- Hidden layer learns a lower-dimensional representation of the input
- Sometimes called “self-supervised” learning

Basic autoencoder architecture



Deeper autoencoder architecture



---------- Encoder ---------- ---------- Decoder ----------

Autoencoders

- Encoder/decoder architecture
  - **Encode** in a hidden layer
  - Hidden layer is smaller than the input (fewer neurons)
  - **Decode** to an output layer
  - Often the encoding and decoding weights are forced to be the same
- Goal: output the input

Hidden layer

- "Bottleneck" layer – smaller than the input
- Represents the input in terms of **latent variables**
  - In the simplest case (one hidden layer with linear activation functions), this layer learns PCA

hidden layer PCA

- Why does this layer need to be smaller than the input?

Output layer

- Unlike a standard NN, the output is not a class or regression value – it's the same type as the input (e.g., an image)
- Activation function is chosen appropriately:
  - For a binary image, tanh or sigmoid
  - For a grayscale/colour image, linear activation

Example: Variational autoencoder

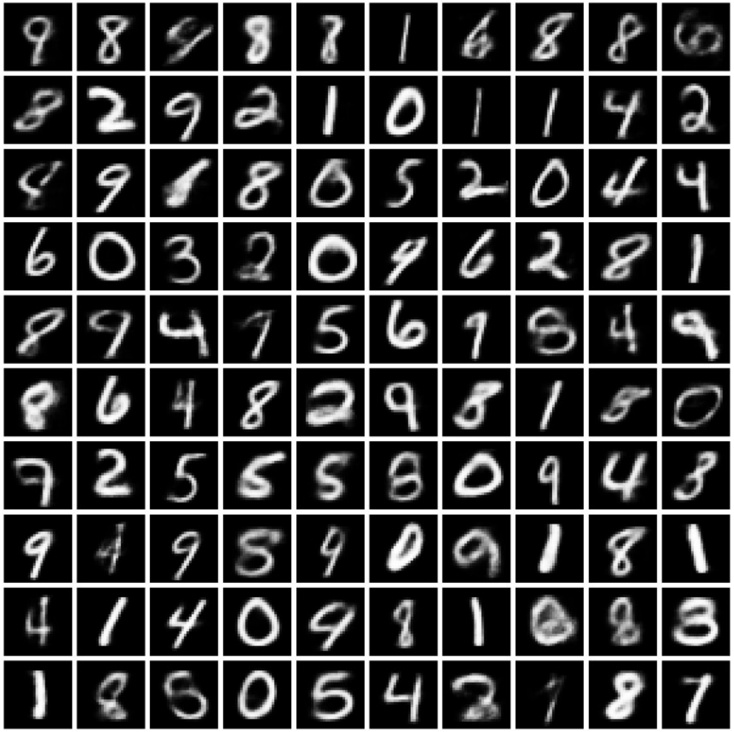


Image: Doersch (2016)

Autoencoders - Summary

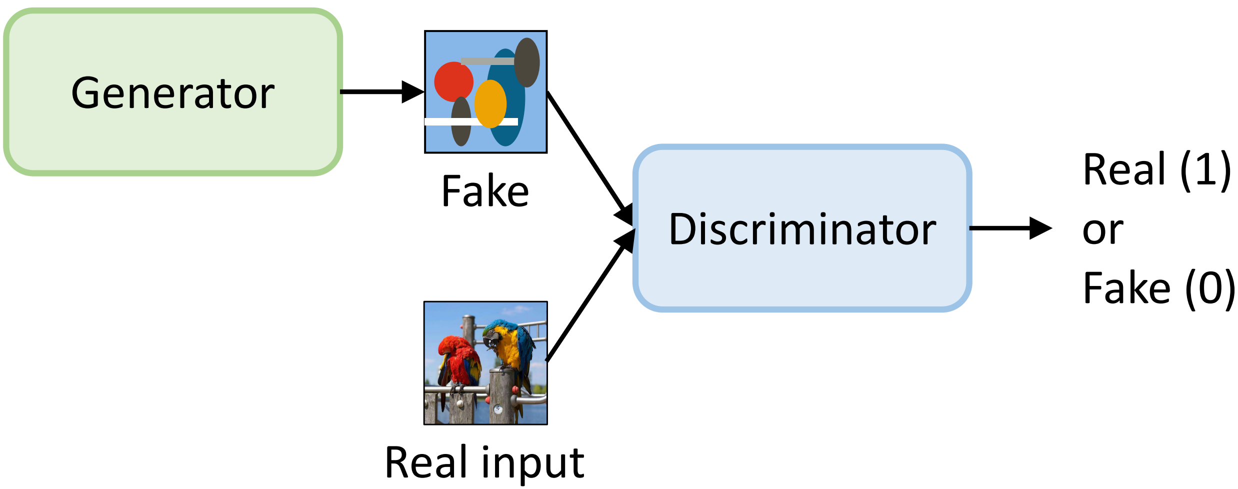
- Advantages
  - Learns a smaller, latent variable representation of the input
  - Can learn this representation over complex features
  - Variational autoencoders can be used to generate new instances
- Disadvantages
  - Deeper versions can be difficult to train

Generative Adversarial Networks (GANs)

GANs

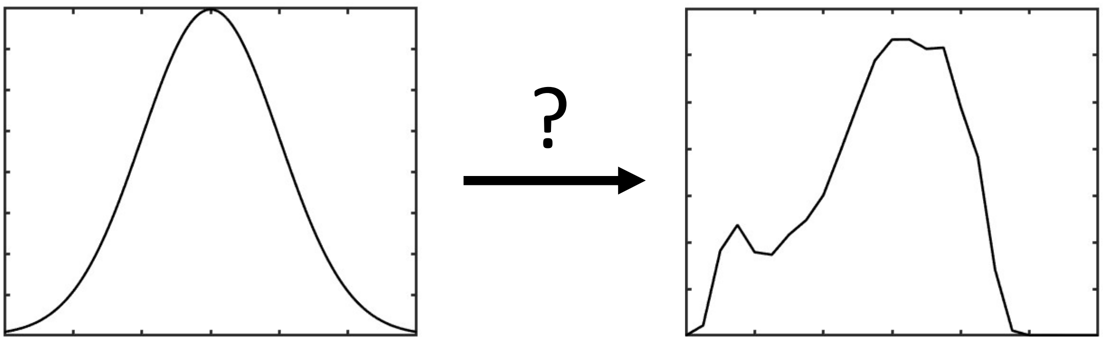
- **Generative Adversarial Networks** (GANs) are neural networks that learn to generate instances from a particular distribution (e.g., images of faces)
- Actually consist of two neural networks: a **generator** and a **discriminator**
- Training involves a sort of competition between the two networks

GAN architecture



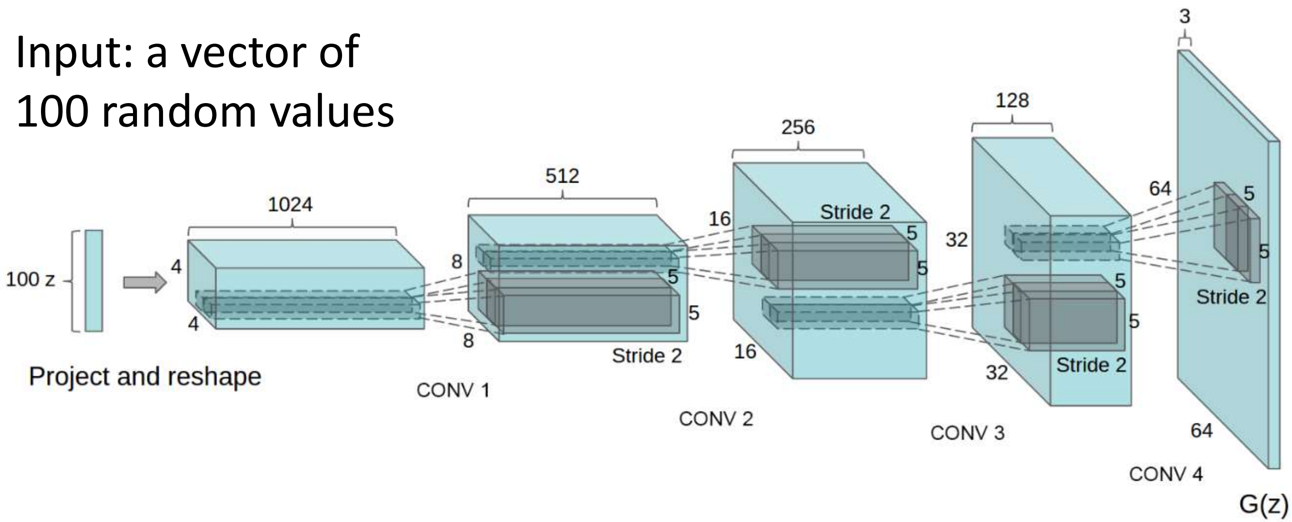
Generator

- Generator learns a probability distribution over inputs
- It does this by sampling from a distribution (e.g., Gaussians) and learning a function to map from this distribution to the input



Generator architecture example

Input: a vector of 100 random values



Output: 64 x 64 pixel colour image

Image: Radford, Metz, & Chintala (2016)

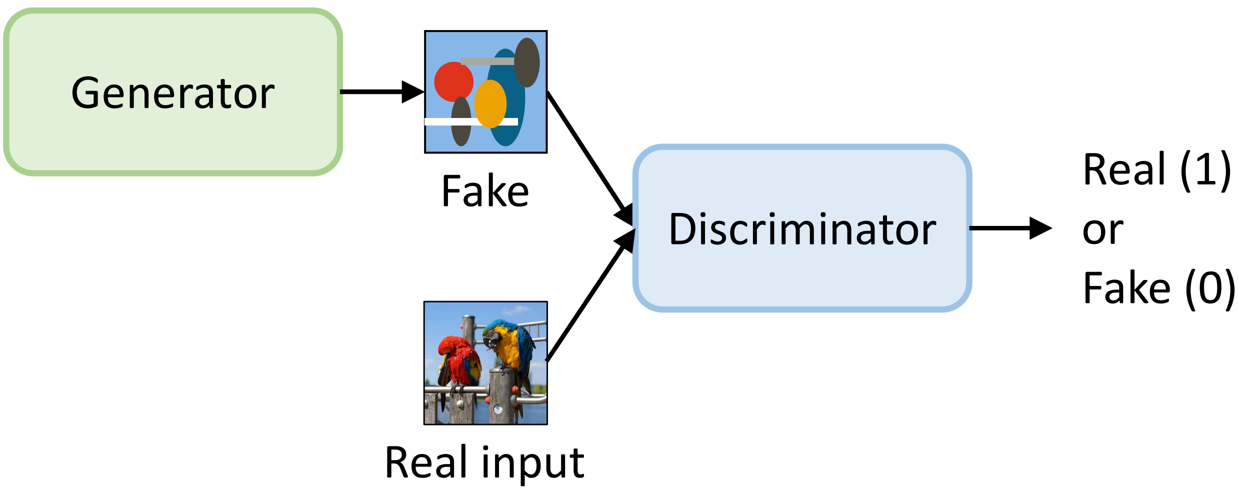
Discriminator

- Discriminator learns to identify real inputs vs. fake inputs created by generator
- Neural network classifier with two output classes (real, fake)
- Architecture depends on task: e.g., for images the discriminator might be a CNN with several convolutional layers

Training

- The networks are trained together on a combination of real data $\mathbf{x}$ and generator input $\mathbf{z}$
- Given a generator G and discriminator D:
  - Discriminator's goal is to correctly classify real vs. fake
  - Discriminator wants to maximize $D(\mathbf{x})$ and minimize $D\big(G(\mathbf{z})\big)$
  - Generator's goal is to fool the Discriminator
  - Generator wants to maximize $D\big(G(\mathbf{z})\big)$
- Can treat this as a zero-sum game with the goal of finding equilibrium between G and D

Training



Training

- If the discriminator is too good:
  - Easily rejects all fake inputs
  - Not much information to train the generator
- If the discriminator is too bad:
  - Easily confused by fake inputs that don’t look real
  - Generator will learn a poor solution
- Training can be difficult – hard to find a balance between discriminator and generator

Evaluating GANs

GAN evaluation

- How to tell if a GAN has learned?
- Ideally:
  - Outputs should look like inputs (look “real” and not “fake”)
  - Outputs should not be identical to inputs (memorized training data)
  - Outputs should be as diverse as real data (avoid **mode collapse** = the generator only creates one or a few outputs)
- First two are easier to evaluate

Memorization?

GAN output:

3 nearest neighbours in training dataset

Karras, Aila, Laine, & Lehtinen (2018)

Realism?

Karras, Aila, Laine, & Lehtinen (2018)

Realism?

Karras, Aila, Laine, & Lehtinen (2018)

Realism?

Karras, Aila, Laine, & Lehtinen (2018)

Diversity?

- The GAN isn't just memorizing training examples
- But does it capture all of the diversity in the training set?
  - How would you measure this?

Birthday paradox for GANs

- Arora, Risteski, & Zhang (2018)
- Suppose a generator that can produce N discrete outputs, all equally likely
- Experiment: take a small sample of s outputs and count duplicates
  - The odds of observing duplicates in a sample of size s can be used to compute N
  - A sample of about $\sqrt{N}$ outputs is likely to contain at least one pair of duplicates

Duplicates and diversity

- Example duplicates (and 1-NN in training dataset):

- Most GANs tested produced about the same diversity (number of different images) as was in their training set

Arora, Risteski, & Zhang (2018)

GANs - Summary

- Advantages
  - Model, and generate samples from, complex probability distributions
- Disadvantages
  - Can be unstable / hard to train
  - Difficult to evaluate
  - Even when the performance looks good, the learned probability distribution may not actually be correct

GAN Applications

- https://www.nvidia.com/en-us/research/ai-playground/