# Generative models

Semester 1, 2024

Ting Dang

Acknowledgement: Kris Ehinger

# Outline

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

# Discriminative vs. Generative

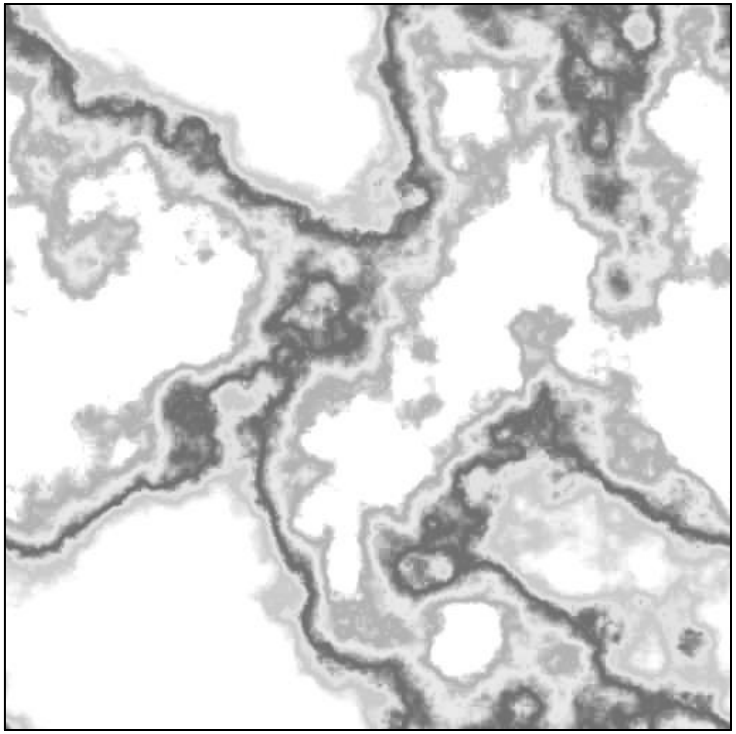
- **Discriminative** models
  - Learn conditional probability of class Y given attributes X: $P(Y|X=x)$
- **Generative** models
  - Learn joint probability of attributes X and class Y: $P(X,Y)$
- Generative model contains discriminative model: you can use the joint probability to get $P(Y|X=x)$
- AND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

# Why generative models?

- In theory, better reasoning because you have a complete model of the world
  - But in practice, if you only need to discriminate between some classes, a discriminative model often performs better
- Model uncertainty, detect out-of-distribution data (data significantly different from training instances, e.g., a totally new class)
- Generate new samples from a distribution

# Generative models

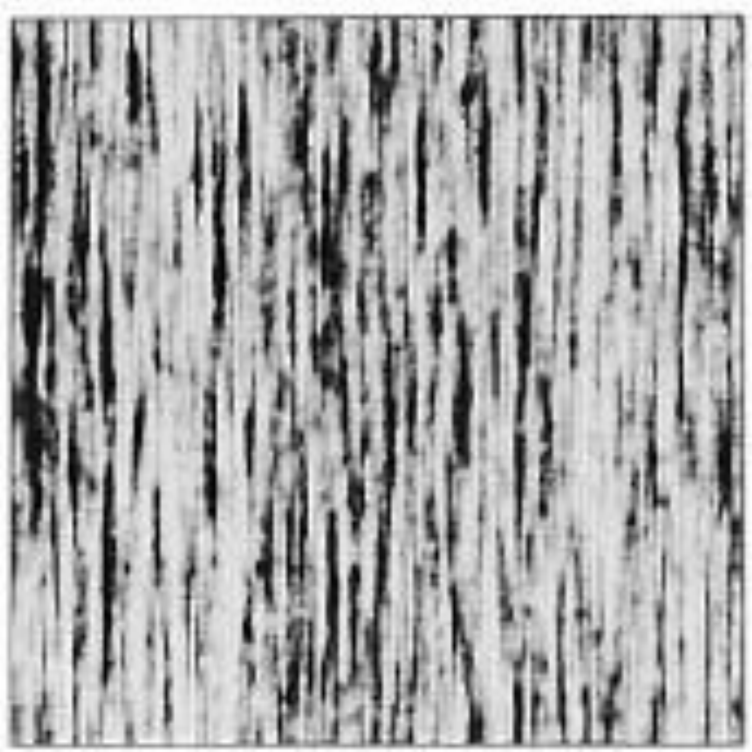
- Sometimes you know the generating process, or can approximate it with a simple model:



"Marble"
Procedural texture algorithm

# Generative models

- Sometimes you know the generating process, or can approximate it with a simple model:

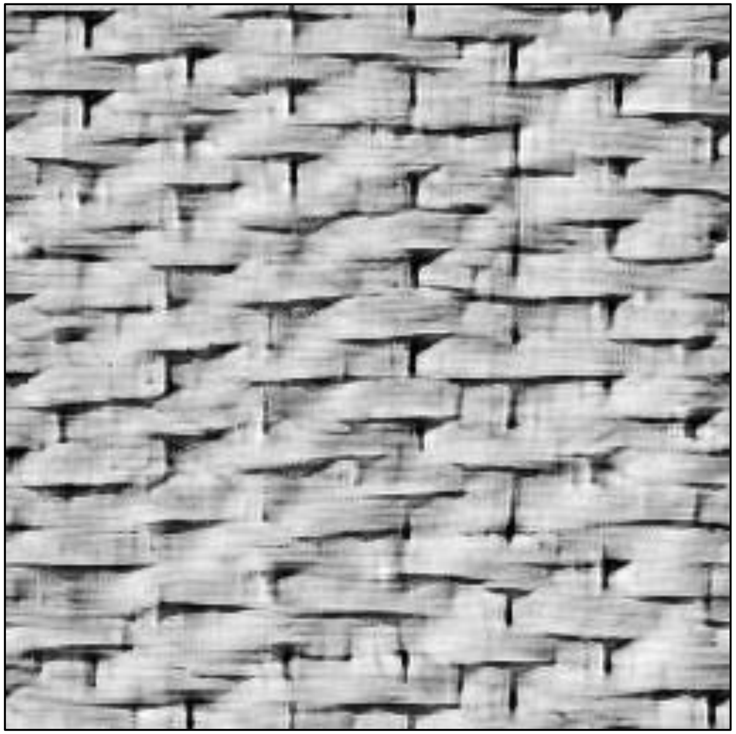


"Wood"
Gaussian mixture model

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

# Generative models

- Sometimes you know the generating process, or can approximate it with a simple model:



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

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

# Generative models

- Sometimes you know the generating process, or can approximate it with a simple model:

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

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


8

# Generative models of images

- Outside of special classes (e.g., textures), image generation can not be approximated by a simple model
- How to generate objects, faces, scenes?

# Generative model approaches

- Goal: model the probability density function (PDF) of what you want to generate (e.g., images)
- Often hard to model the density function directly
- Instead:
  - Map to a lower-dimensional “latent” space – autoencoders, variational autoencoders (VAE)
  - Learn a function to convert samples from a simple PDF (e.g., Gaussian) into the target PDF, to allow sampling from the PDF – generative adversarial networks (GAN)
  - Learn the gradient of the PDF and move along the gradient to generate more-probable samples – score-based models, diffusion models

# Generative model approaches

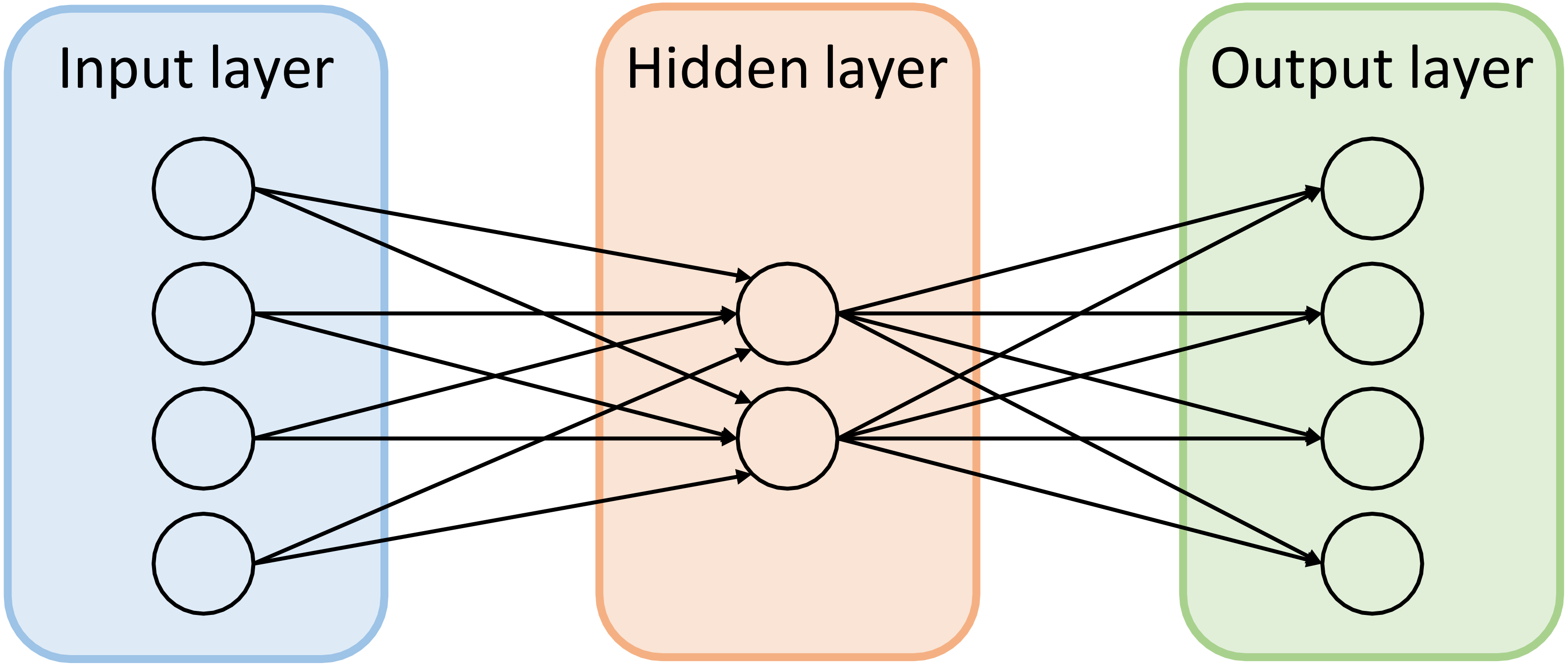
- Training requires a lot of data
  - GPT-3: 570GB of text
  - Stable Diffusion: 2.3 billion images
- Training is unsupervised (or self-supervised), requiring no human labels
- Usually, hide part of the input and ask model to predict it
  - Masked language modelling – predict missing words in sentences
  - Image denoising – remove (Gaussian) noise from images

# Autoencoders

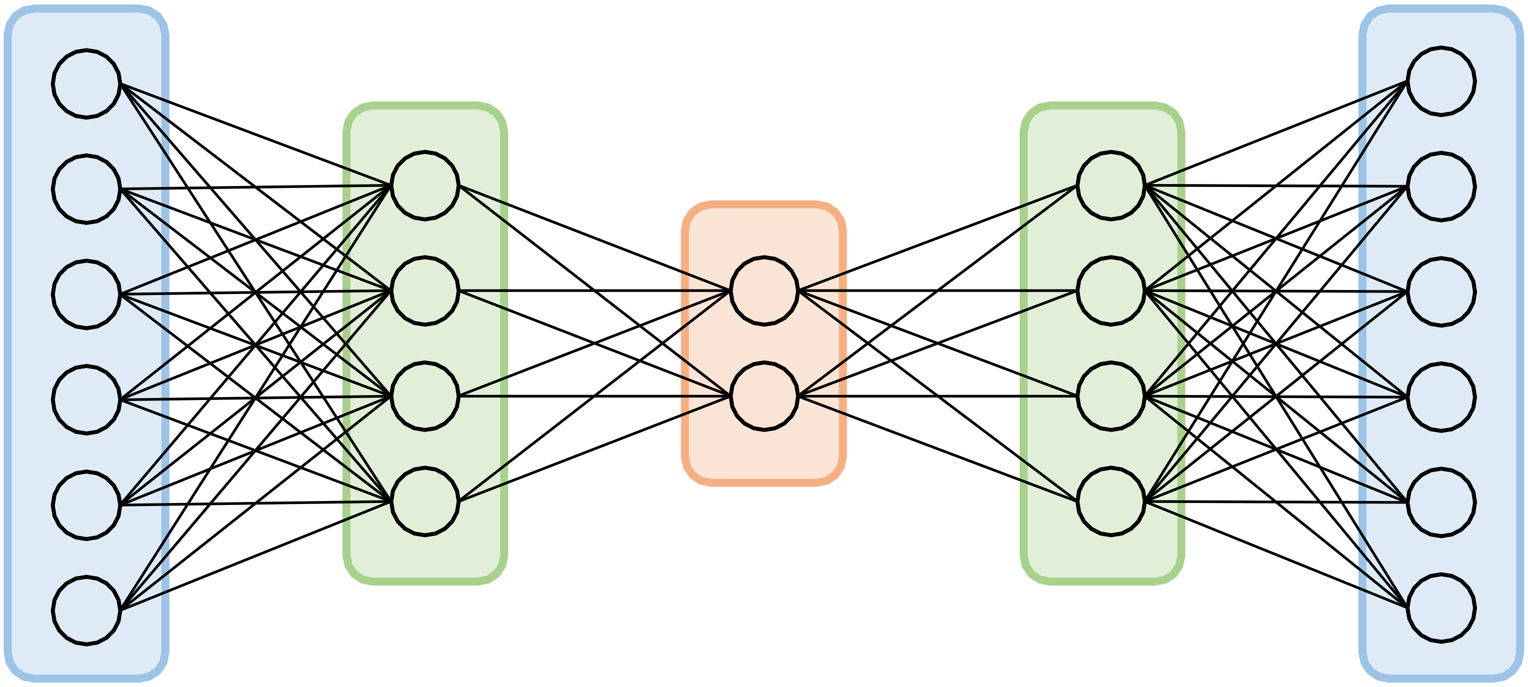
# Autoencoders

- 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 (latent) representation of the input
- Different types – “variational autoencoder” adds constraints to the basic autoencoder so the latent layer can be used to generate new samples

# 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

- 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

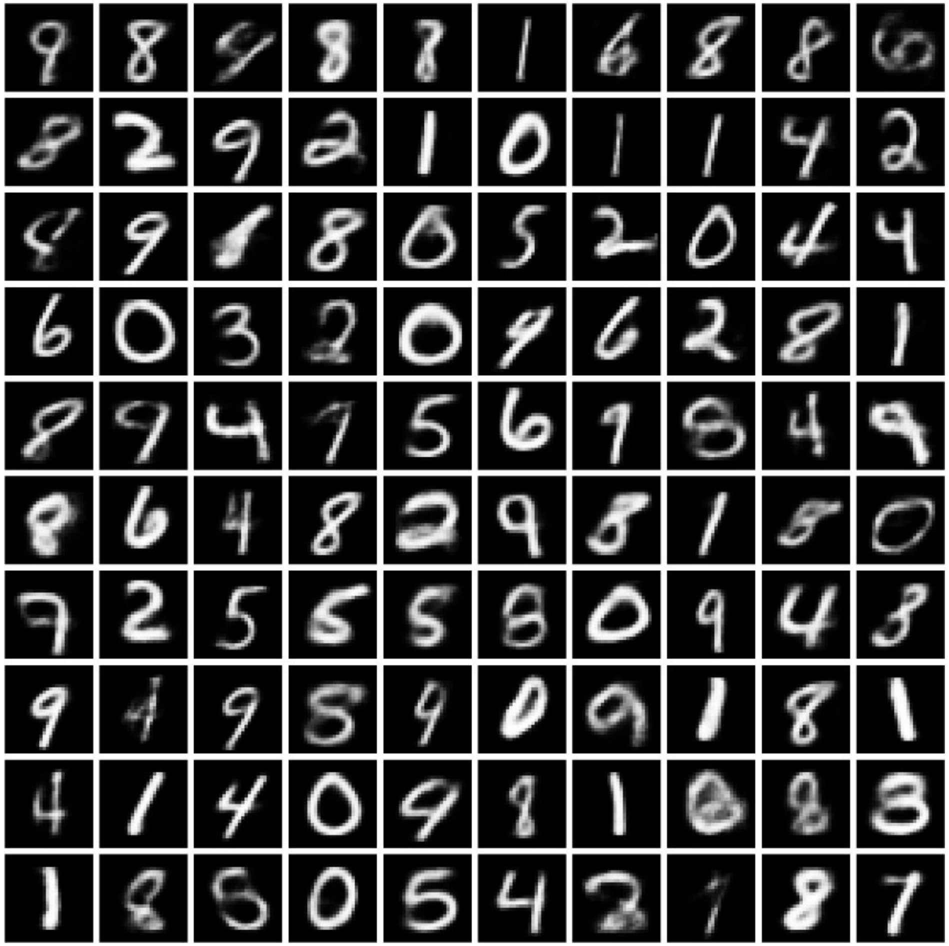


Image: Doersch (2016)

# Autoencoders - Summary

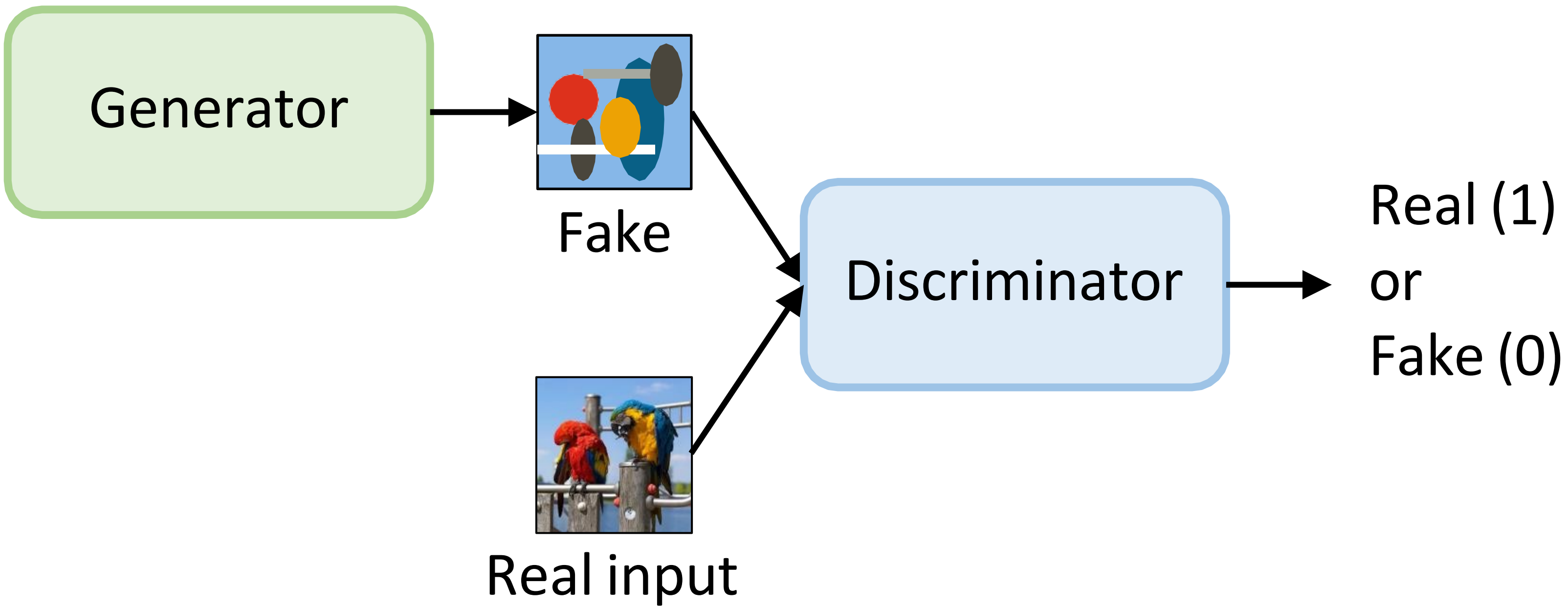
- Advantages
  - Learns a smaller, latent variable representation of the input
  - Can learn this representation over complex features
  - Variational autoencoders can be used to generate new instances
- Disadvantages
  - Deeper versions can be difficult to train

# Generative Adversarial Networks (GANs)

# GANs

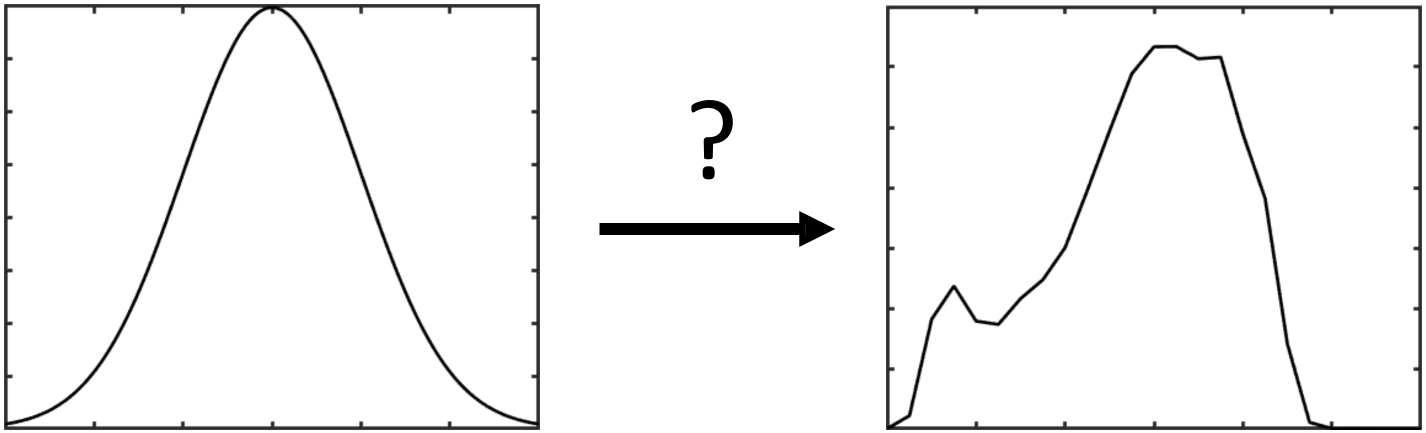
- **Generative Adversarial Networks** (GANs) are neural networks that learn to generate instances from a particular distribution (e.g., images of faces)
- Actually consist of two neural networks: a **generator** and a **discriminator**
- Training involves a sort of competition between the two networks

# GAN architecture



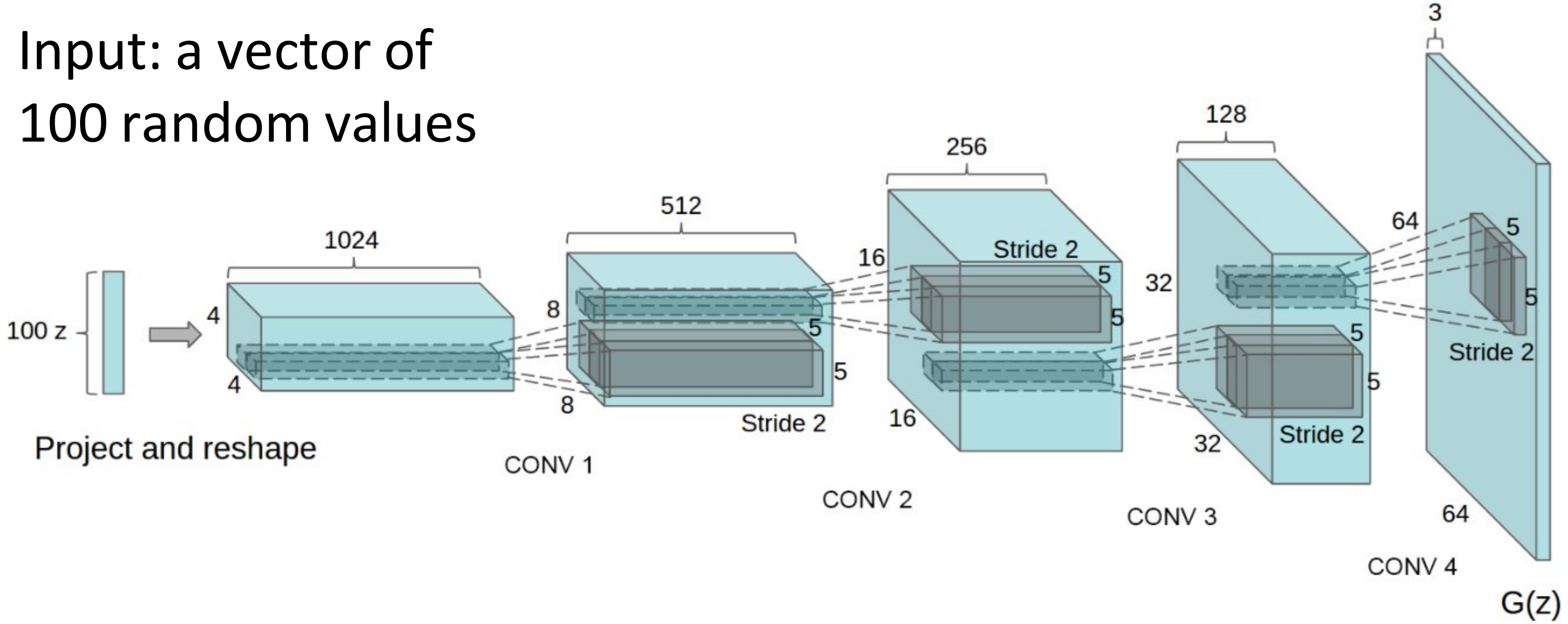
# Generator

- 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)

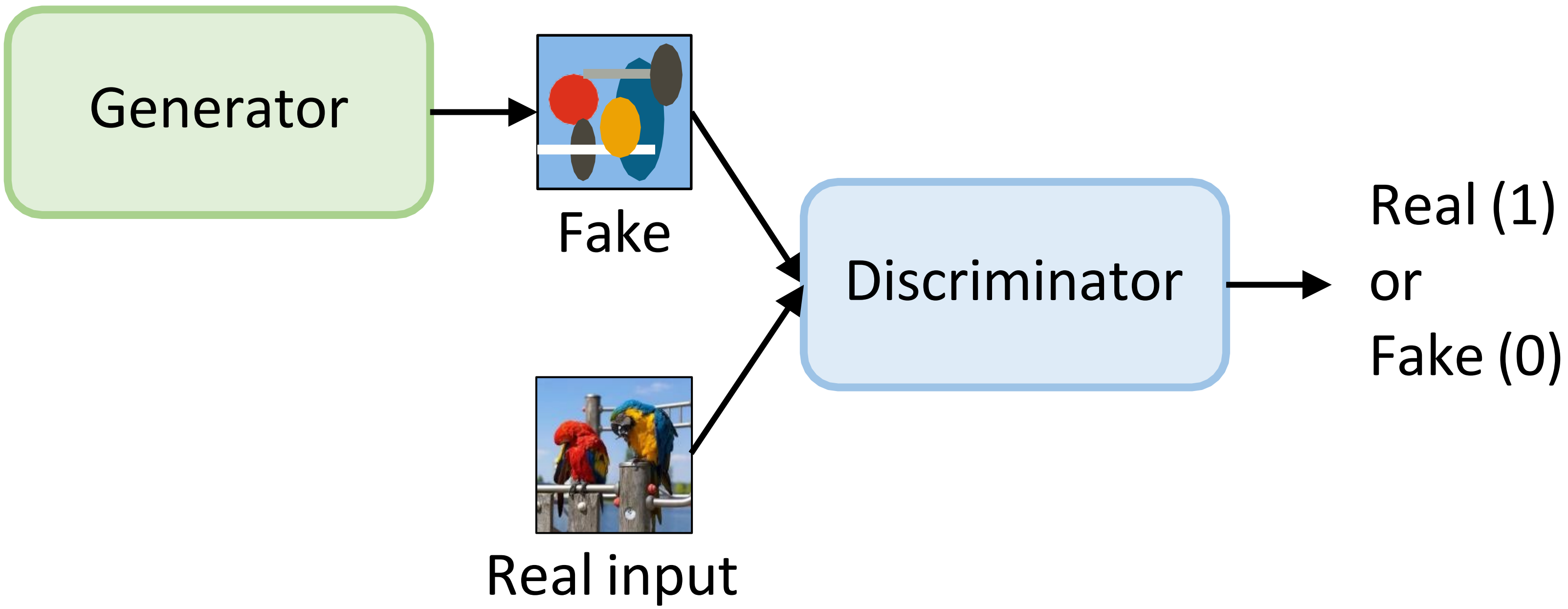
# Discriminator

- 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 $D\left(G(\mathbf{z})\right)$
  - Generator's goal is to fool the Discriminator
  - Generator wants to maximize $D\left(G(\mathbf{z})\right)$
- 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

- What are the odds that someone else in this subject has the same birthday you do?
  - 74% ($= 1 - \ (364/365)^{494}$)
- What's the smallest class size that has at least a 50/50 chance of two people sharing a birthday?

# 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

# Big data

Semester 1, 2024 Ting Dang

Acknowledgement: Kris Ehinger

# Importance of data

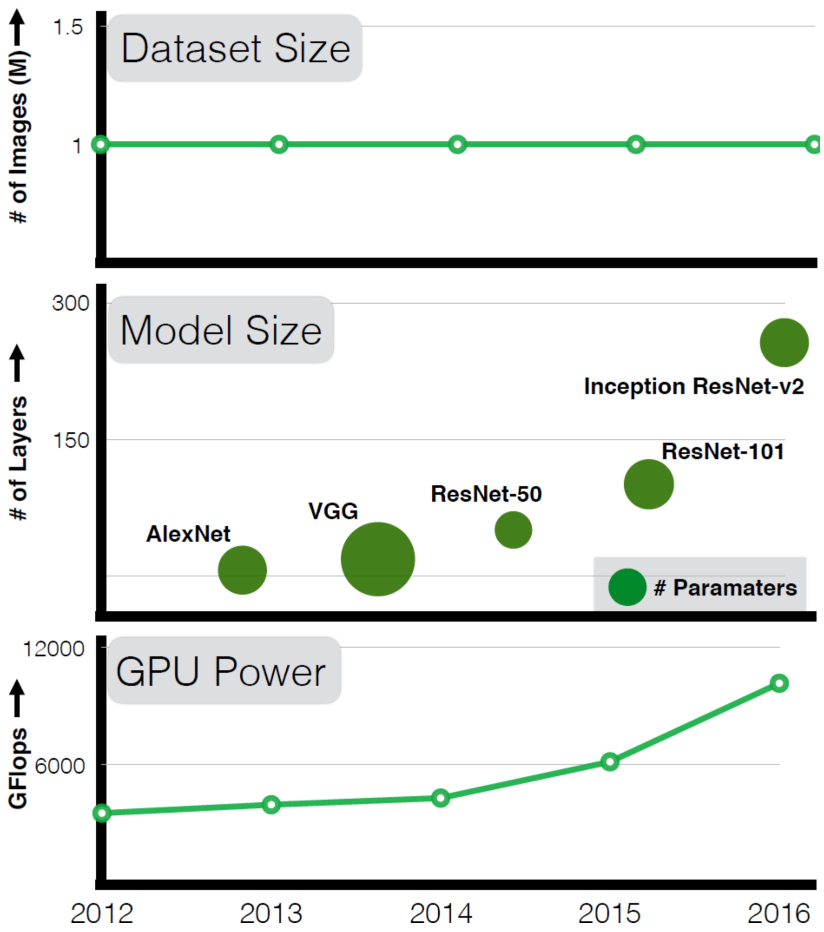
"memorization is a good policy if you have a lot of training data. … simple models and a lot of data trump more elaborate models based on less data."

– Halevy, Norvig, & Pereira (2009)

"The Unreasonable Effectiveness of Data"

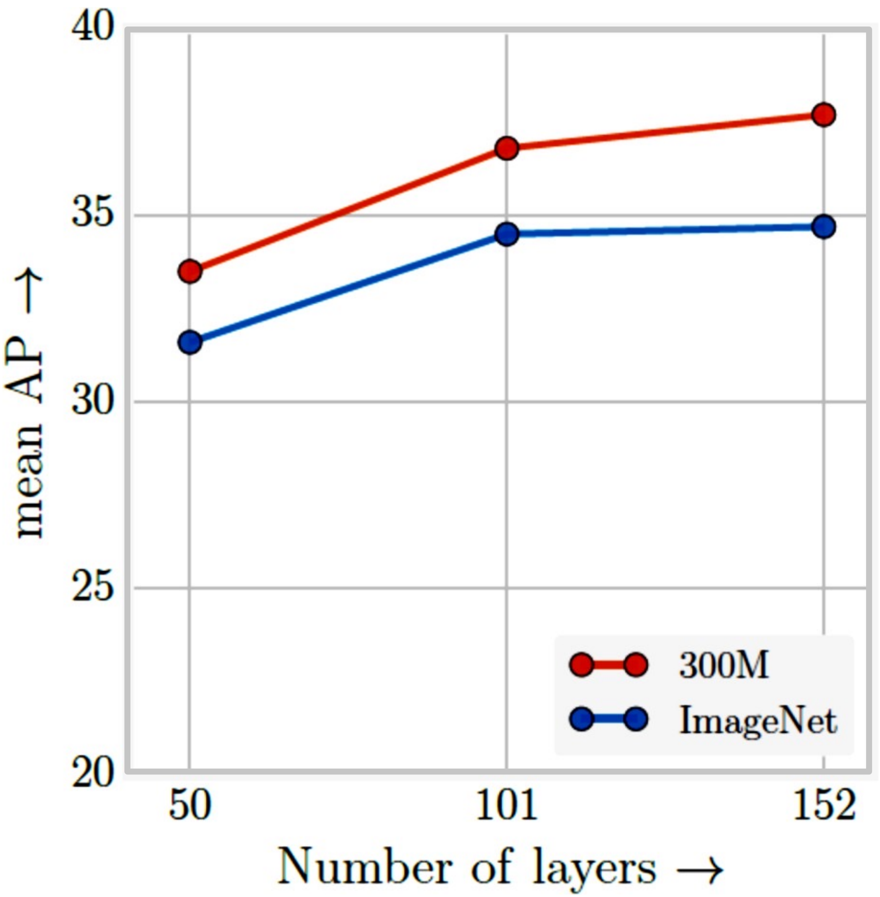
# Importance of data

- (Labelled) data is a bottleneck for machine learning
- In image classification, model depth has increased dramatically, but the size of “large-scale” datasets has not kept pace



# Importance of data

- Adding data is nearly as effective as adding layers
- More parameters are not helpful unless you have more data to train them



# Labelling bottleneck

- Data is (often) abundant but labelling is expensive
  - Switchboard corpus: 400 hours of annotation time per hour of speech data
  - Image labelling: often 30-60 minutes per image for a complete segmentation

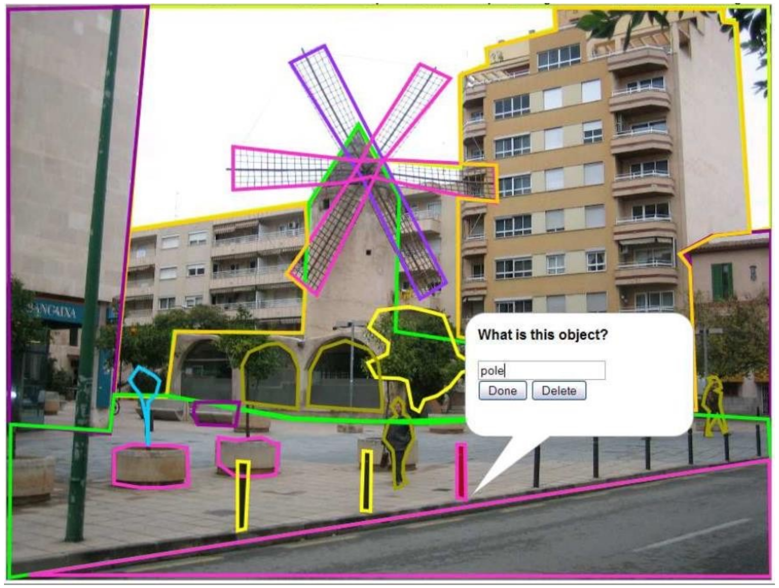



Image: Torralba, Russell, & Yuen (2010)

# Outline

- Data augmentation
- Semi-supervised learning
- Active learning
- Data considerations

# Data Augmentation

# Data augmentation

- There are various ways to expand a labelled training dataset
- General: resampling methods
- Dataset-specific: add artificial variation to each instance, without changing ground truth label

# Bootstrap sampling

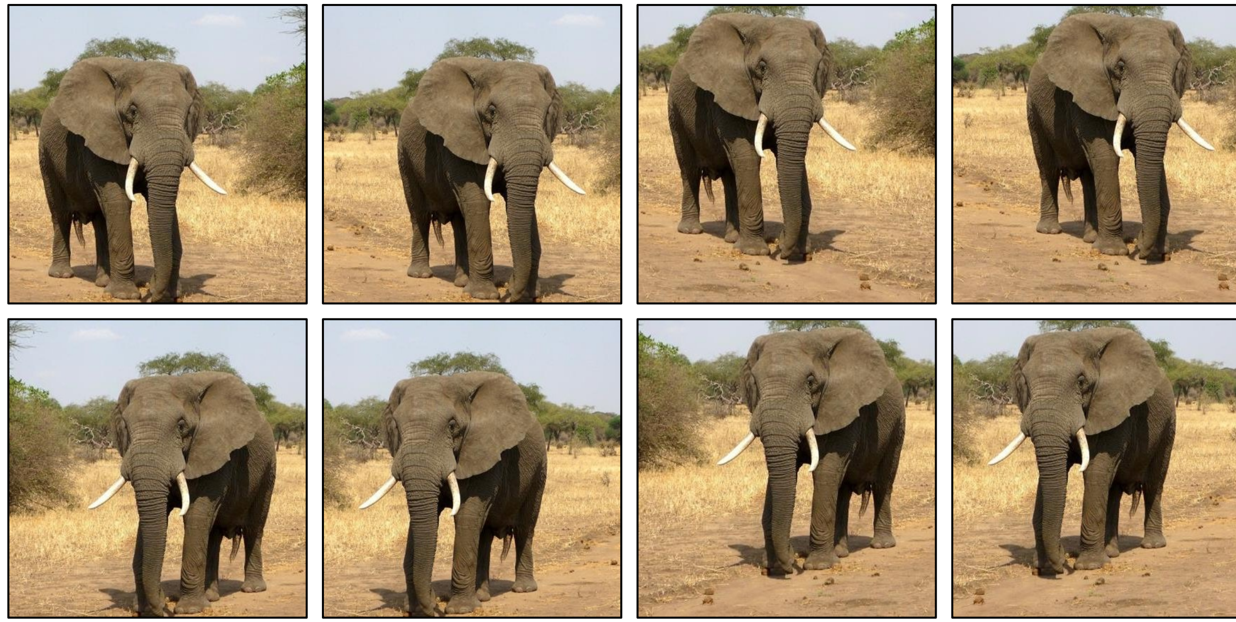
- Bootstrap sampling: create “new” datasets by resampling existing data, with or without replacement
- Common in perceptron and neural network training (“mini-batch,” “batch size”), methods that involve stochastic gradient descent
- Each “batch” has a slightly different distribution of instances, forces model to use different features and not get stuck in local minima

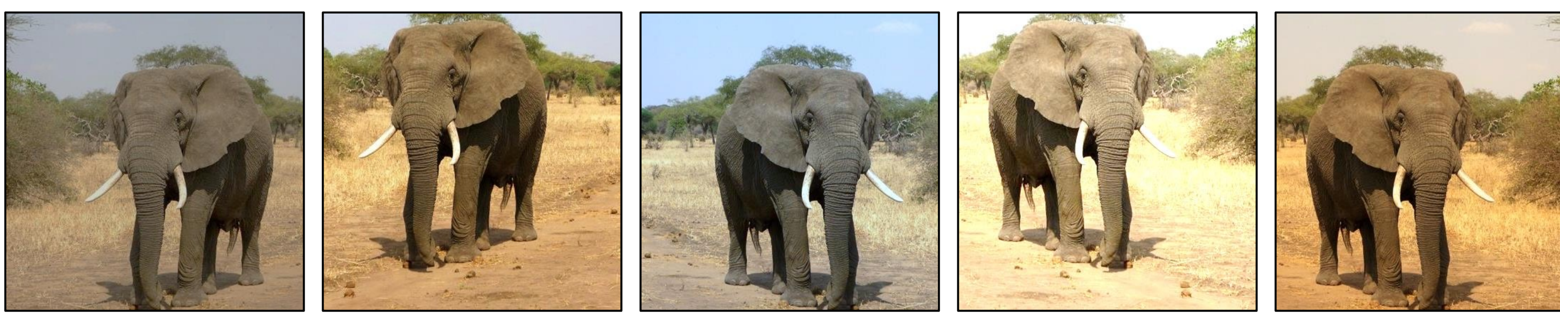
# Data manipulation

- Another option: add a small amount of noise to each instance to create multiple variations:
  - Images: adjust brightness, flip left-right, shift image up / down / left / right, resize, rotate
  - Audio: adjust volume, shift in time, adjust frequencies
  - Text: synonym substitution
- These perturbations should not change the instance's label
- Generally, they should be the same kind of variations you expect in real-world data

# Common image manipulations

Original





# Data synthesis

- Create artificial data using another machine learning method:
  - Train a probability distribution on labelled data
  - Sample the probability distribution to produce new instances
- Generative adversarial network: neural network trained to create samples from a distribution
- Exploit algorithms designed for other tasks, e.g.:
  - Computer-generated images
  - Automatic translation

# Data augmentation

- Advantages:
  - More data nearly always improves learning
  - Most learning algorithms have some robustness to noise (e.g., from machine-translation errors)
- Disadvantages
  - Biased training data
  - May introduce features that don’t exist in the real world
  - May propagate errors
  - Increases problems with interpretability and transparency

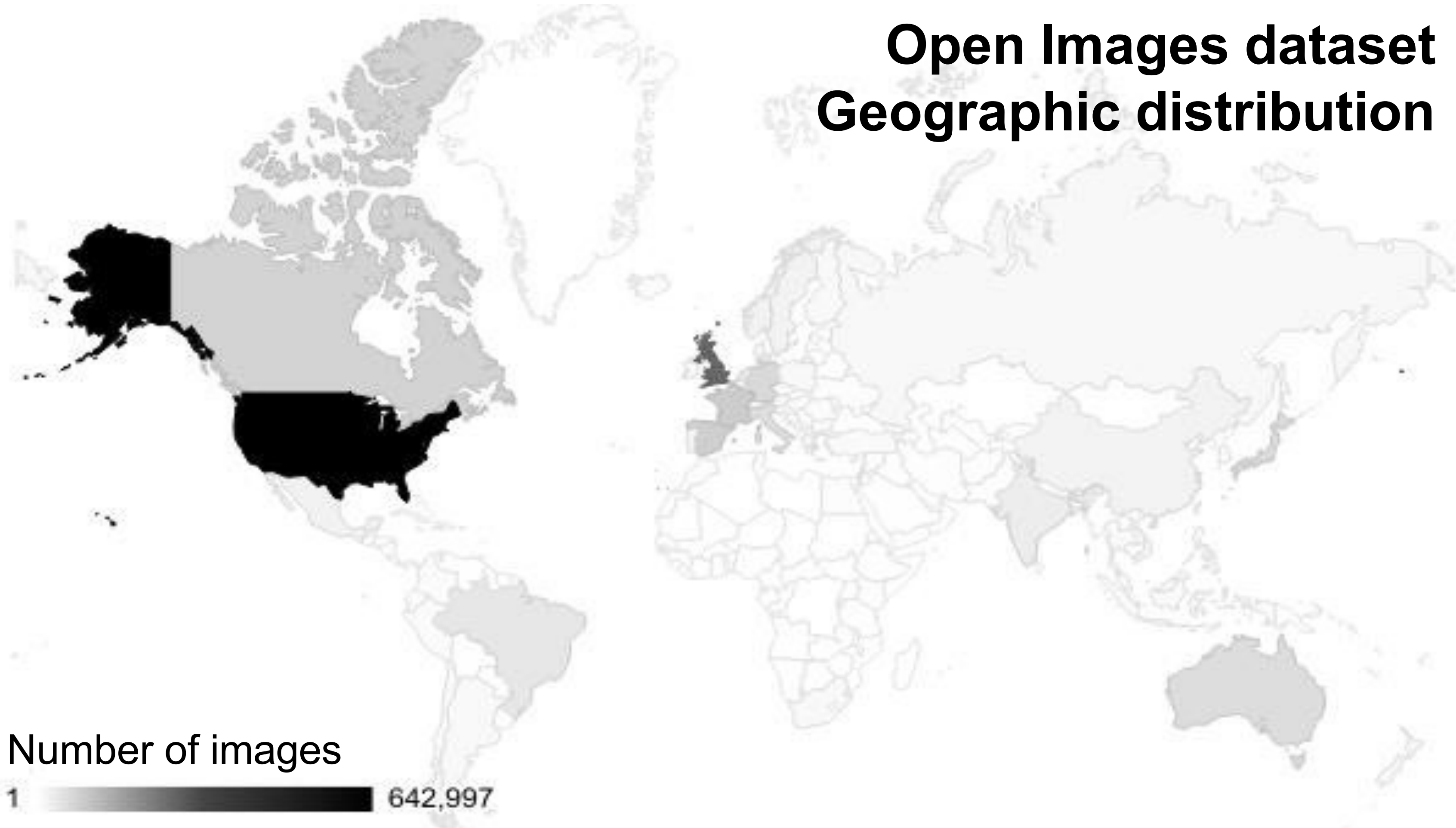
# Data Considerations

# Data considerations

- Does the training dataset accurately reflect the real world?
  - Tendency to use data that is convenient, not necessarily fair or representative; e.g., movie transcripts, Twitter
- Do you want to accurately reflect the real world?
  - Undersample minority groups
  - Replicate historical biases
- Do you have the right to use this data to train an AI model?
  - Data ownership, copyright, privacy

# Example: Object recognition

**Open Images dataset**
**Geographic distribution**




Shankar, Halpern, Breck, Atwood, Wilson, & Sculley (2017)

# Dataset bias

- What happens if some groups are less represented in a dataset?
  - Less contribution to loss function, so potentially more errors on this group
  - Poorer model fit in this group due to limited training data
  - Poorer generalisation to new examples of this group due to lack of training diversity
- Note that continuous learning methods can exacerbate this problem

# Dataset bias

- Does dataset bias lead to worse performance on the test set?

# Mitigating dataset bias

- Treat as an imbalanced dataset problem
  - Use data augmentation or oversample to increase samples from underrepresented group
  - Adjust loss function to put more penalty on errors in underrepresented group
- Force model output to be independent of some variable (e.g., race or gender)
- Drawbacks
  - Requires you to know about the bias in your dataset
  - Doesn’t solve the problem of low diversity

# Recommendations

- Check your dataset for biases that you can think of
- But be aware that datasets may be biased in many ways you haven't thought of

- The fact that something is on the internet doesn't necessarily mean you can use it to train AI – there may be legal/ethical issues around using data in this way, even if it is "publicly available"

# Summary

- Labelled data is a major bottleneck for machine learning
- There are various strategies for making use of unlabelled data, or making more effective use of the data and labelling resources we already have
- These strategies generally improve performance, but sometimes with a trade-off in terms of error propagation, bias, and interpretability