

Neural Networks

For a great supplement, read [this](#) article about how ChatGPT works (it's only 100 pages... 😊)

When professor
says we are
making neural nets



When he starts
explaining them



Why?

Neural networks are **advanced** machine learning models designed to replicate many observed characteristics of human brains

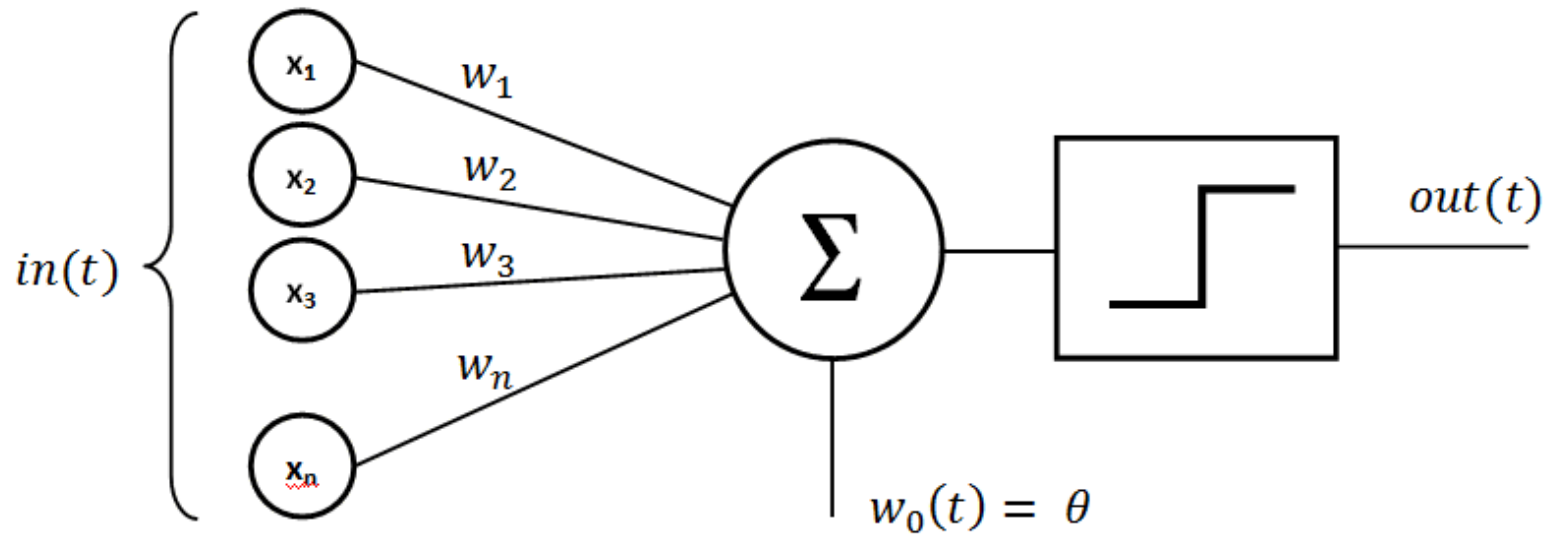
- Hard to use well
- Difficult to explain clearly
- Highly valuable **in specific contexts**

The basics - Perceptrons

A perceptron is the computational equivalent of a single neuron

Let's describe it with a visual

A perceptron



What is an Activation Function?

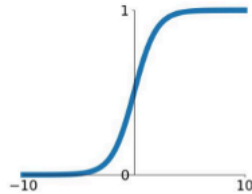
An activation function is a math function that determines when a perceptron moves from "off" to "on".

Again, let's describe this visually

Activation Functions

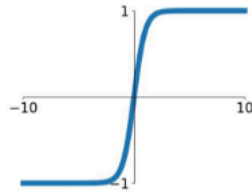
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



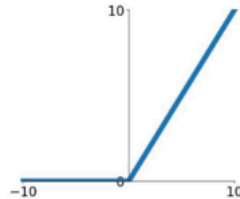
tanh

$$\tanh(x)$$



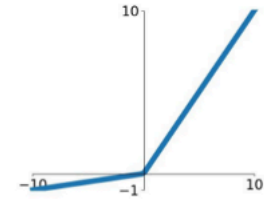
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

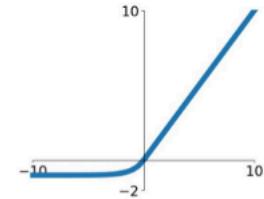


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

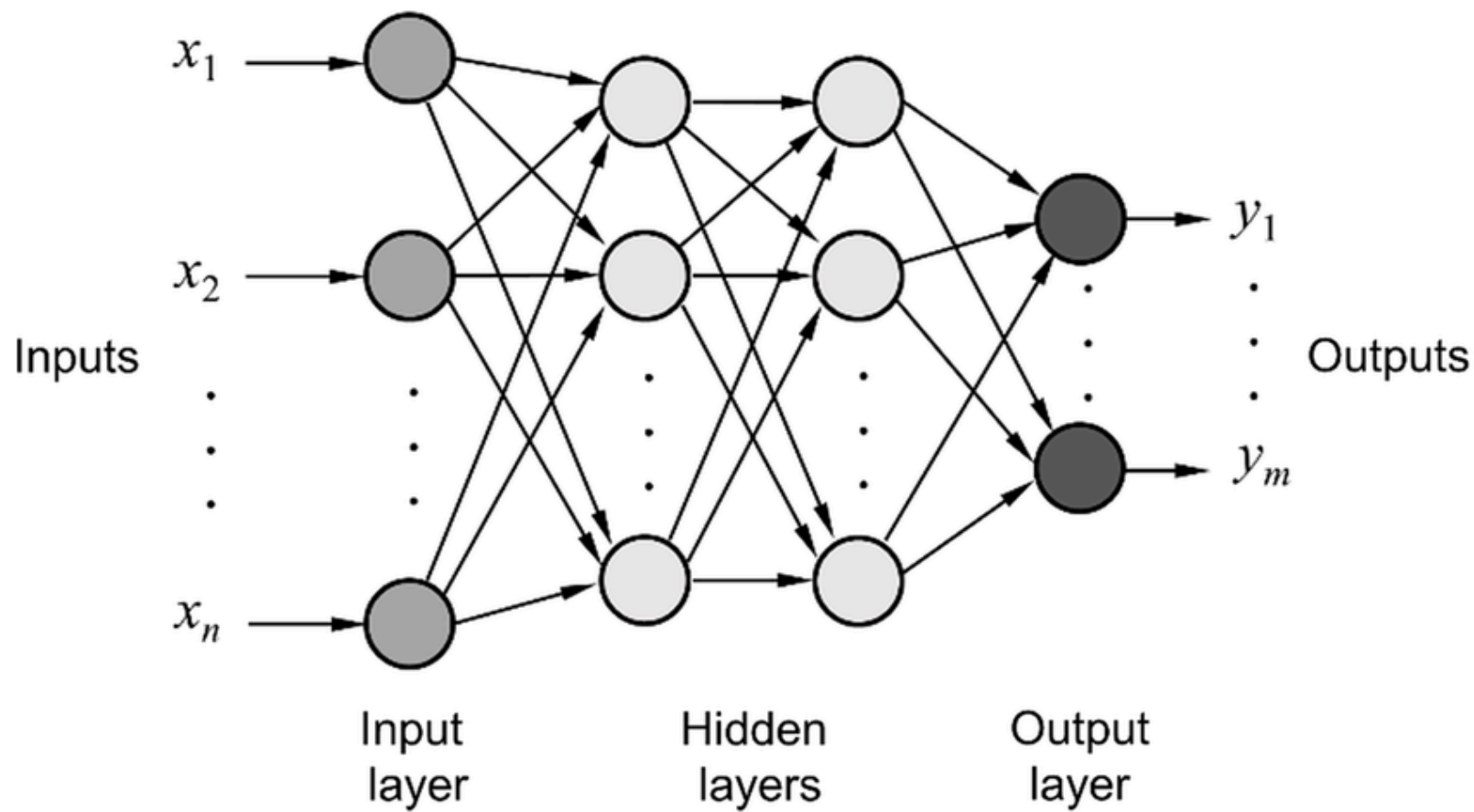
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



How does this make a Neural Network?

Neural networks are made up of **layers** of perceptrons that are interconnected, leading from the **input layer** to the **output layer**.

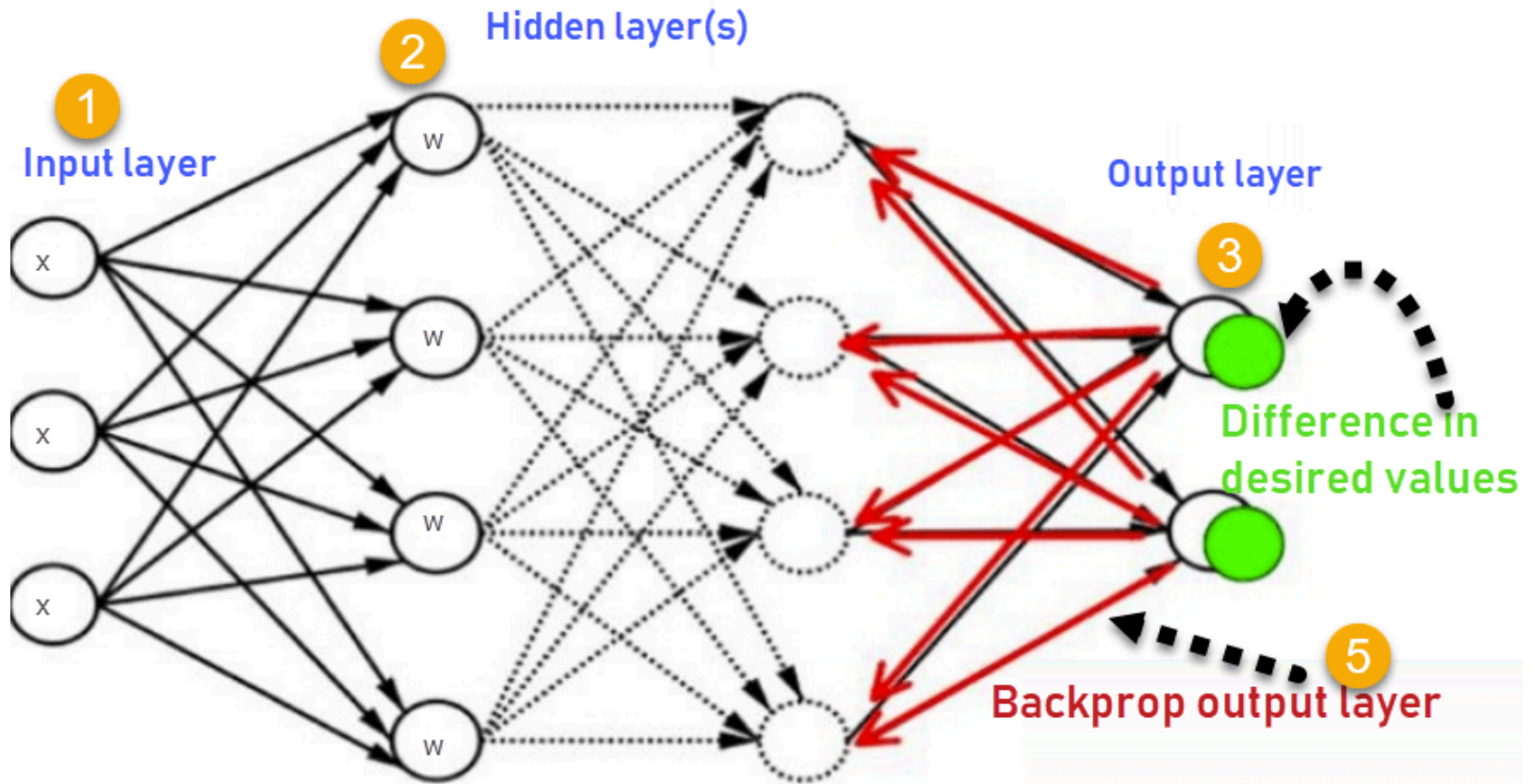
Any layer in between is called a **hidden layer**. Neural networks are often referred to as **deep learning** because of these hidden layers making the neural network "deep".



How do we use them?

Our goal with a neural network is to train our network with **weights** and **biases** (these are what trigger the activation functions, remember?) so that the network is able to represent the complex process of predicting our outcome of interest.

How we learn - Backpropagation



How we learn - Backpropagation

Through inputs and backpropagation, we train our model to perform as well as we can on our task.

How do we choose the right network?

Choosing the right network

1. Use one that someone else designed for the same (or similar) tasks
2. Evolutionary algorithms ("neuroevolution")
 - Here's an example: <https://www.youtube.com/watch?v=qv6UVOQ0F44>
3. Trial and error (probably not efficient)

Using Keras to work with Neural Networks

This simple example comes from the [keras documentation](#)

```
import os
import numpy as np

os.environ["KERAS_BACKEND"] = "torch"

# Note that keras_core should only be imported after the backend
# has been configured. The backend cannot be changed once the
# package is imported.
import keras_core as keras
from keras_core import layers
```

Prep and Load Data

```
# Model / data parameters
num_classes = 10
input_shape = (28, 28, 1)

# Load the data and split it between train and test sets
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

It's better for this example to pull MNIST from the library, since it is already stored on Google's server so it loads the FULL 60,000 observations very quickly

Scale Image Data

```
# Scale images to the [0, 1] range  
x_train = x_train.astype("float32") / 255  
x_test = x_test.astype("float32") / 255
```

Many small images come with data stored in levels from 0 to 255 (8-bits), and we need to scale that to be between 0 and 1

Color channels

```
# Make sure images have shape (28, 28, 1)
print("x_train shape before transform:", x_train.shape)
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
print("x_train shape after transform:", x_train.shape)
print(x_train.shape[0], "train samples")
print(x_test.shape[0], "test samples")
```

Many images have 3 color channels, so our data needs to have three dimensions (height, width, color channels). Since this data is black and white, we only have one channel.

Convert dependent variable

```
# convert class vectors to binary class matrices – one hot encoding  
y_train = keras.utils.to_categorical(y_train, num_classes)  
y_test = keras.utils.to_categorical(y_test, num_classes)
```

Instead of 1 column with values between 0 and 9, we need 10 columns, with ones and zeros, called **one hot encoding**, just like what we did for regressions with categorical data.

Specify the model

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)  
  
model.summary()
```

Specify the model

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dropout (Dropout)	(None, 1600)	0
dense (Dense)	(None, 10)	16,010

Total params: 34,826 (136.04 KB)

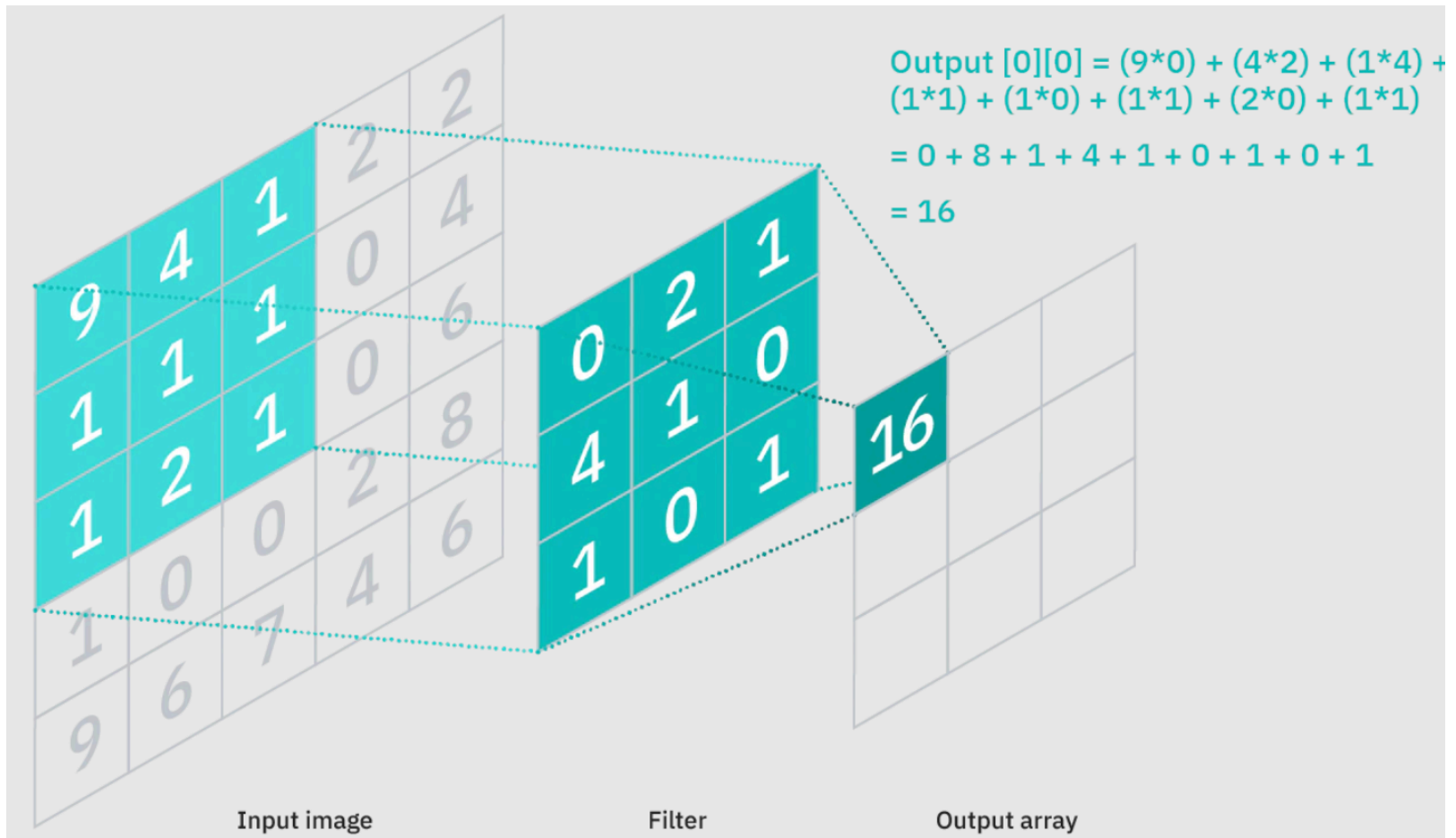
Trainable params: 34,826 (136.04 KB)

Non-trainable params: 0 (0.00 B)

This model is ~10,000,000 times smaller than ChatGPT!

What do all these "layers" mean??

Convolution



Convolution

The goal of convolution is to try to extract patterns in our image through small-scale filters applied to each region of the image. This reduces the size of the image, but the result is a matrix of "pattern" data.

Pooling

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

max pooling

20	30
112	37

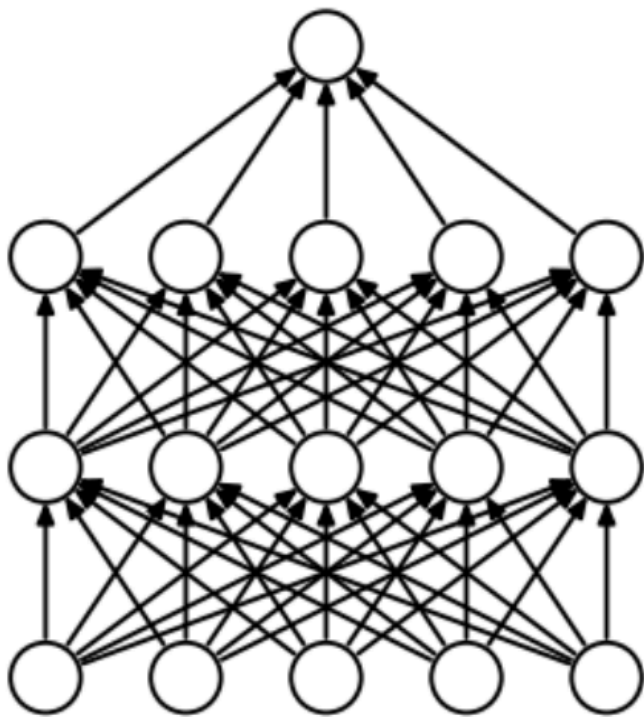
average pooling

13	8
79	20

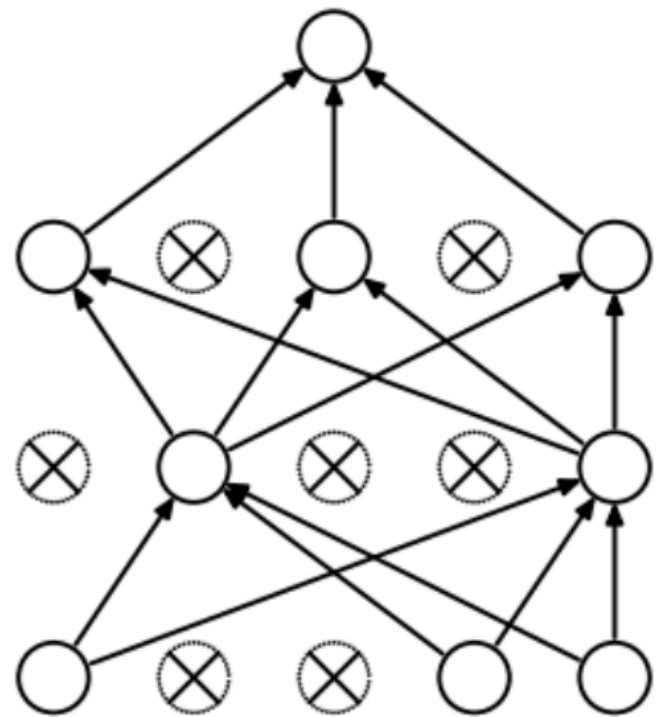
Pooling

Pooling is a simple compression of the data, with our options being to average the value of a group of pixels, or to take the maximum value of the grouped pixels

Dropout



(a) Standard Neural Net



(b) After applying dropout.

Dropout

In a neural network, it is common for each neuron in one layer to connect to **every neuron in the next layer**. This is called a **fully connected** layer. In order to avoid overfitting, dropout "unplugs" a random percentage of neuron connections. The dropped connections are chosen randomly.

Train the model

```
batch_size = 128
epochs = 3

# USING ENTROPY TO TRAIN OUR MODEL!
model.compile(loss="categorical_crossentropy",
              optimizer="adam", metrics=["accuracy"])

model.fit(x_train, y_train,
          batch_size=batch_size, epochs=epochs,
          validation_split=0.1)
```

More definitions

Batch size: The number of image samples provided at once to our model. We have to keep this relatively small, since it takes a lot of computer memory to train the models on a set of images.

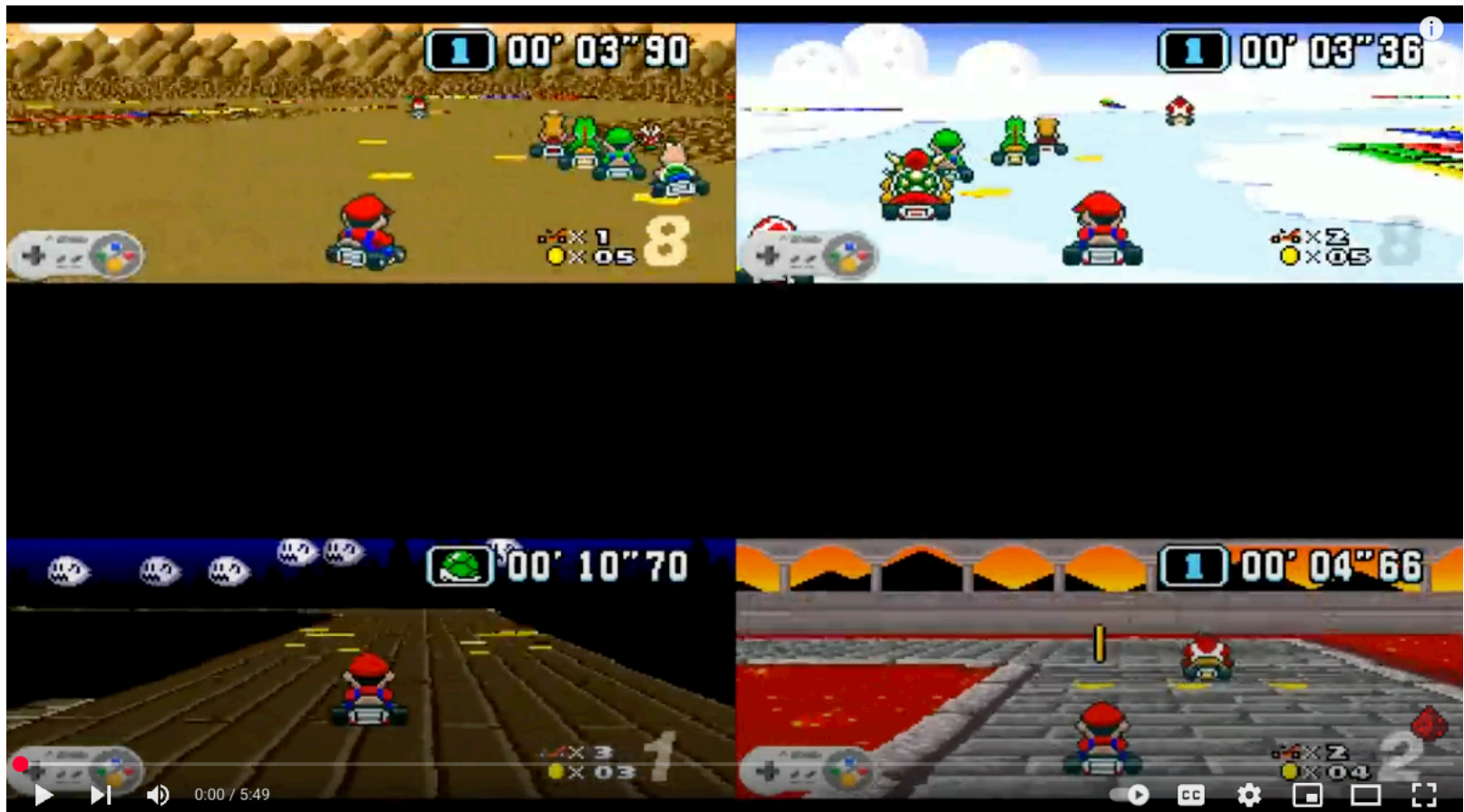
Epochs: The number of times we will loop over the **entire data set** as we train our model. More epochs leads to more refined models, but also takes a long time.

Score the model

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

Does this model outperform our decision trees and random forests? Why?

Time series models with Neural Networks



From time series to LLM

Lab Time!

Work on the project or experiment with the neural network we made. There is no required homework today! 😊

(Nothing needs to be submitted for Assignment 11)