Machine Learning

(Source 1, Source 2, Source 3)

We have heard of "Machine Learning" as a buzzword for the past few years; reasons for this might be the high amount of data produced by everyone, or the increase of computation power in the past few years, or the development of better **algorithms** and the introduction of "deep" learning.

Machine Learning is used anywhere from automating mundane tasks to offering intelligent insights: industries everywhere try to benefit from it. You may already be using a device that utilizes it: for example, a wearable fitness tracker like Fitbit, or an intelligent home assistant like Alexa (actually, in 2021 many data-processing companies such as Google, Amazon, Facebook/Instagram use machine learning techniques). But there are much more examples of ML in use.

- Prediction – weather, banking/stocks, flight patterns, etc.
- Image recognition – facial detection in an image, image classification. Recently, "deep fakes" have highlighted achievements in this area (see video from Rijksmuseum).
- Speech and handwriting recognition – Alexa, Google Home, Siri and various note-taking apps use this.

What are some examples you have heard of?

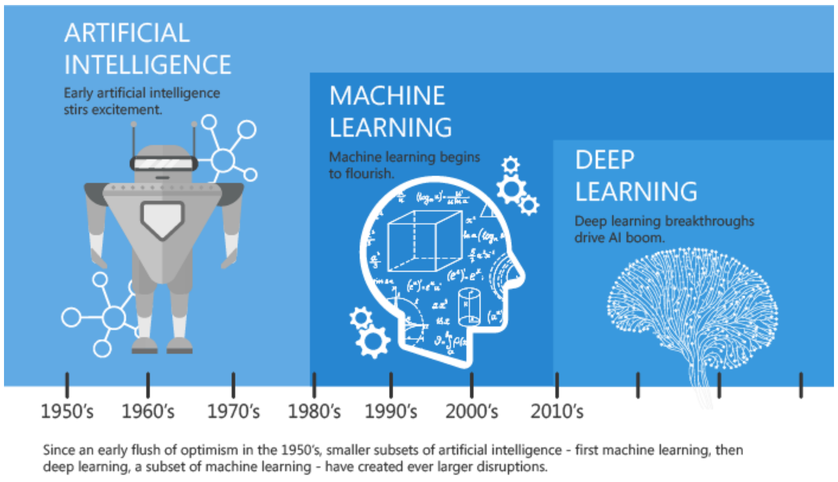
ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's
Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

Figure 1: Image credit.

"Machine learning" is a category of algorithms that allow software applications to become more accurate in predicting outcomes without being explicitly programmed to. The basic premise of machine learning is to build algorithms that can receive input data and use statistics to predict an output while updating outputs as new data becomes available. Broadly, there are three types of ML algorithms:

1. **Supervised Learning algorithms** – the computer is provided with example inputs that are labelled with the desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabelled data.

   For example, with supervised learning, an algorithm may be fed data with thousands of images of sharks labelled as "fish" and thousands of images of oceans labelled as "water". By being trained on this data, the supervised learning algorithm should be able to later identify unlabelled shark images as "fish" and unlabelled ocean images as "water".

2. **Unsupervised Learning algorithms** – data is unlabelled, so the learning algorithm is left to find things in common among its input data. It is commonly used for shopping data: figuring out what purchases people might like to make based on what they have bought previously (e.g. Amazon).

3. **Reinforcement Learning algorithms** – a reinforcement learning algorithm, usually programmed into a physical robot, learns by interacting with its environment. The robot receives rewards by performing correctly and penalties for performing incorrectly. The robot learns without intervention from a human by maximizing its reward and minimizing its penalty.

Do you know any examples of each of these types of learning algorithm?

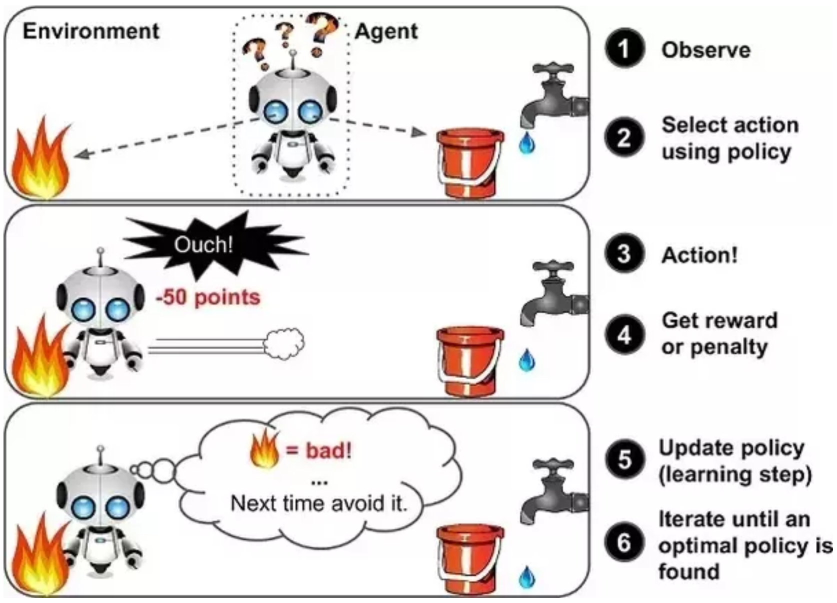
Environment
Agent
1 Observe
2 Select action using policy
Ouch!
-50 points
3 Action!
4 Get reward or penalty
= bad!
...
Next time avoid it.
5 Update policy (learning step)
6 Iterate until an optimal policy is found

Figure 2: Image credit.

Deep supervised learning covers identifying objects in images, recognising handwriting, automatically colouring black-and-white photos, and "deep fakes". It is connected to automatic machine translation and music creation (1, 2).

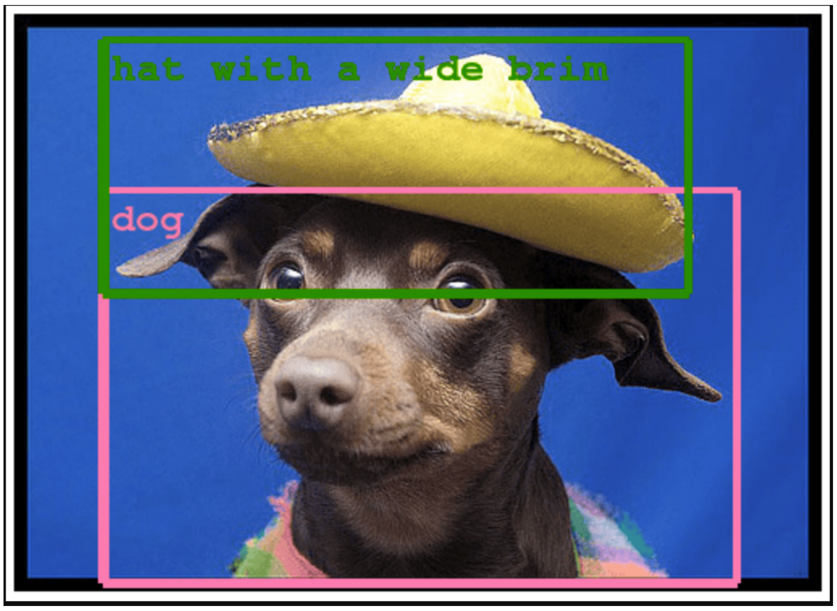

Figure 3: Image credit.

Deep Supervised Learning

Let's focus on the example of recognising a single, hand drawn digit. **How would you make a computer solve this problem?**

The key behind deep learning is its use of "neural nets"; layers of nodes ("neurons") programmed into the machine which are trained to capture specific features of an inputted image. For example, when training a digit recognition algorithm, your inputted date is a 28 x 28 pixel square like the following:

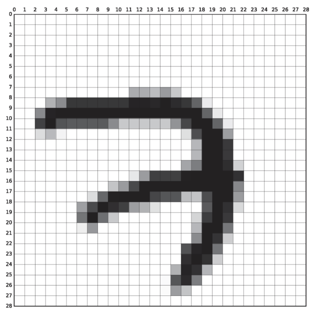

Figure 4: Image credit.

Specifically, for this example, your input data is the colour of $(28 \times 28 =)$ 784 different pixels. There are more complicated ways of feeding data into the neural net, including passing it through several "filters" to reduce the amount of data or highlight certain features. However, the end result is still some numbers feeding into a column of input **neurons**:

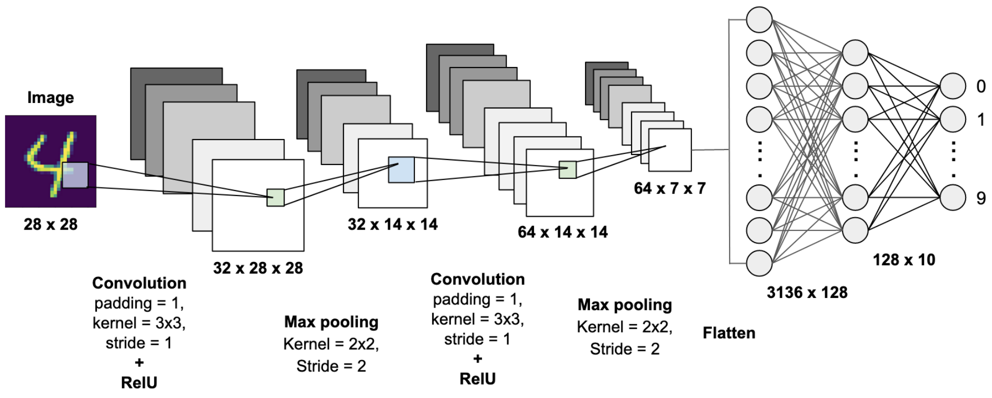

Figure 5: Image credit.

There are several layers between the input neurons, which take in data, and the output neurons, which output either “0”, “1”, “2”, . . . , “9”. These layers are what make the neural network “deep”. **Can anyone guess why they’re called “hidden layers”?**

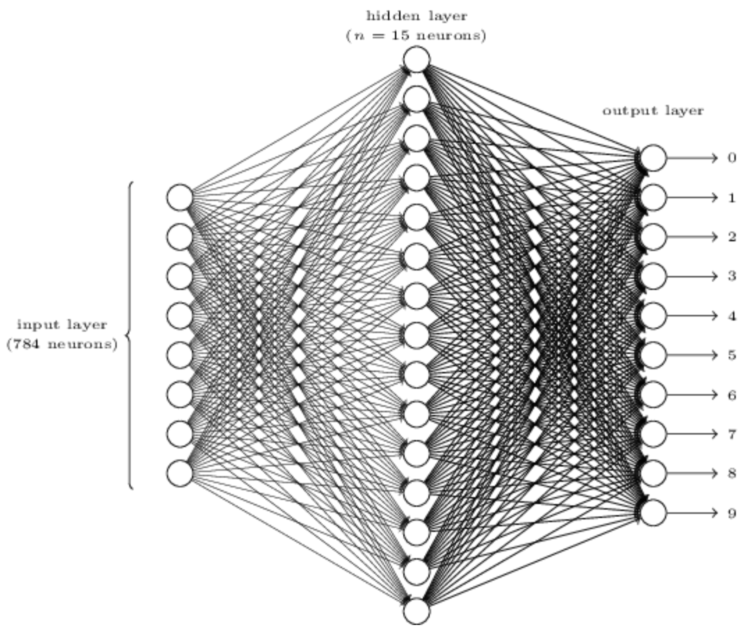

Figure 6: Image credit.

What happens in these “hidden” layers? A very simple neuron could take one inputs, then do two things: *multiply* by a number (“a weight”) and *add* a number (“a bias”).

For example: suppose our weight was the number 7 and our bias was the number 1. What would be the output:

- On input 3?
- On input 5?
- On input 10?

The output then feeds into the next neuron, forming the neural net. This process of passing inputs forward to get an output is known as *feedforward*. Below is an example of a neural network ("neural net") even simpler than our digit recognition example – it has two inputs, a single "hidden" layer, and one output.

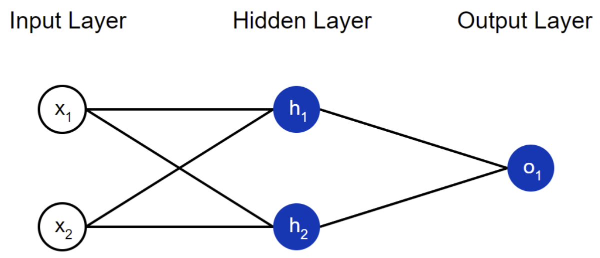

Figure 7: Image credit.

Of course, we need to answer the question: where do we get the weights and bias? These numbers will be crucial to determining whether your deep learning algorithm works as it should, and this is what your algorithm "learns" from the training data.

Training

We want to pick values for the weights and bias that *minimize*, or make as small as possible, any error the computer makes in guess what number you've shown it. This is in general not an easy thing to do, and to do it properly requires mathematics you wouldn't see until you're at university.

However, the basic idea is called *backpropagation*. The method is to start out with some randomly chosenvalues (e.g. weight 1, bias 1), then calculate how many times the computer got the answer wrong on the *training data*. We then make a *small* change to the values of the weights and biases (e.g. weight 2, bias 2) and calculate how many times the computer got the answer wrong again – if it has increased, then we know we have made a bad change, and discard it. If it has decreased, we know we have made a good change, and keep it. We continue doing this until the computer makes as few mistakes as possible – that any change to the weights and bias makes the number of errors bigger. This is the *minimum* (it is often not zero) and with these values the neural net is considered *trained*.

Examples and Applications

Now we know how it works, let's see what we can do.

- Experiments with Google (example: QuickDraw).
- machinelearningforkids.co.uk.
- Jukebox.

Overview

Today, we started by motivating *machine learning* from the perspective of a music experiment at Google. Then, we spoke about how we can see many examples of machine learning in our everyday lives.

We talked about the 3 types of machine learning *algorithm*, and went through the example of how your computer recognises your handwritten numbers. Finally, we looked at some examples and resources for learning more online.