A brief intro into neural networks with pytorch

It really is very brief

The brain

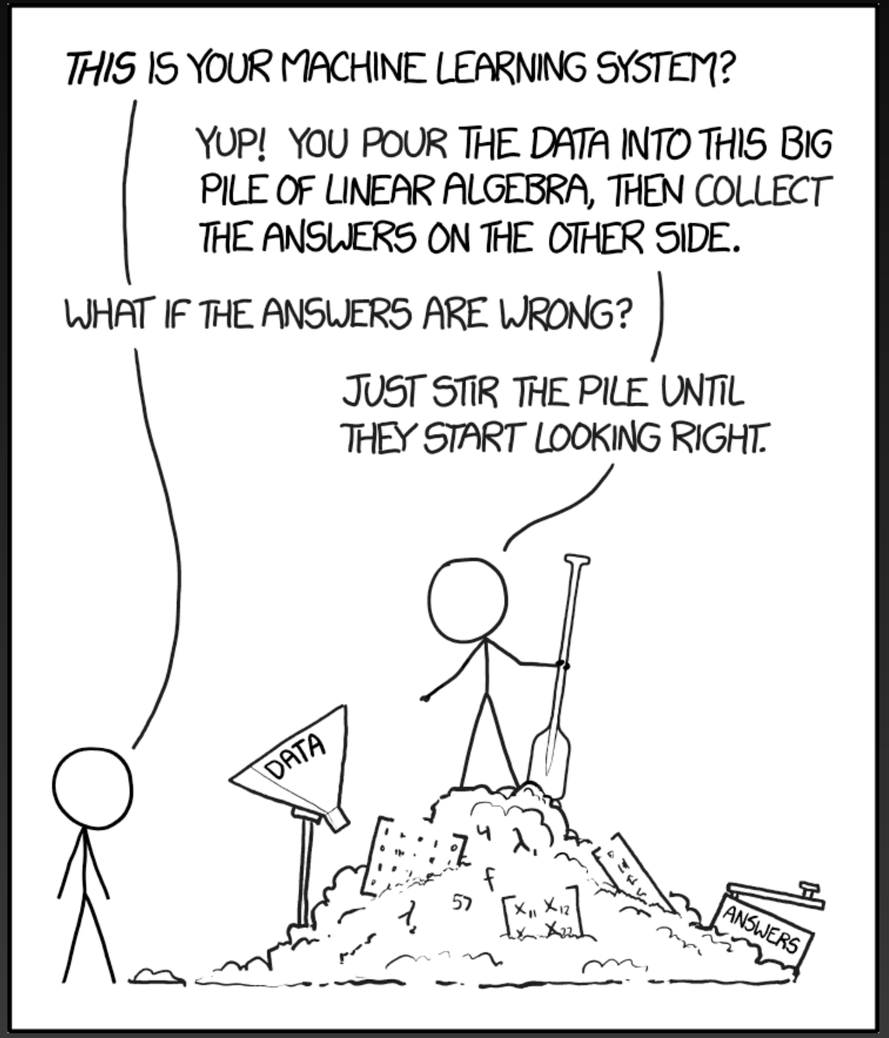
Its pretty useful

- It's the control center of your body, which means you'd be lost without it. Literally.
- Your brain is like a Swiss Army knife It's the ultimate multitasker, letting you listen to music, chat with friends, and browse memes all at the same time.
- Your brain is a master at pattern recognition
- Your brain is incredibly energy-efficient.

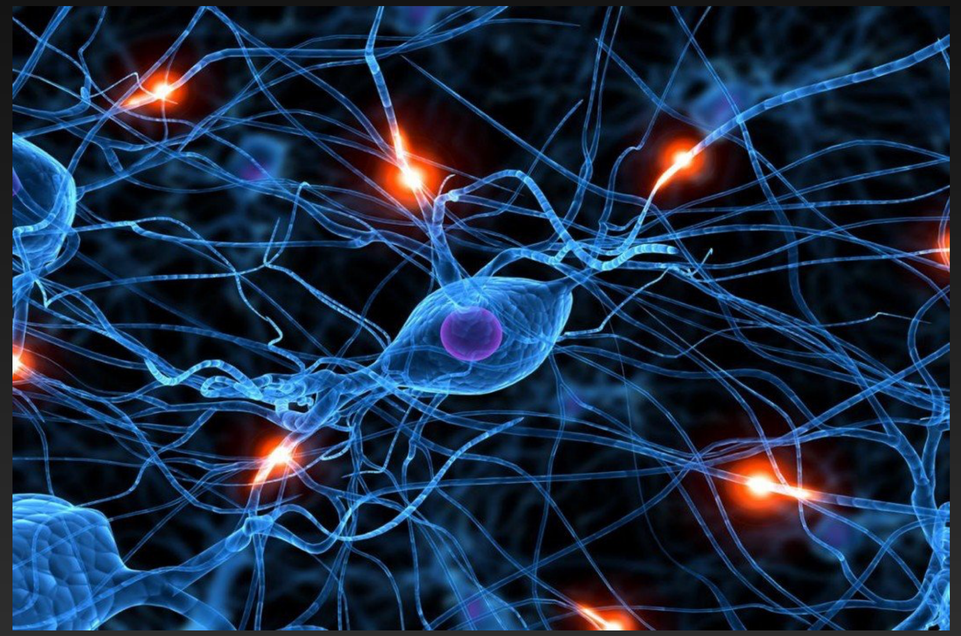
Building the artificial brain

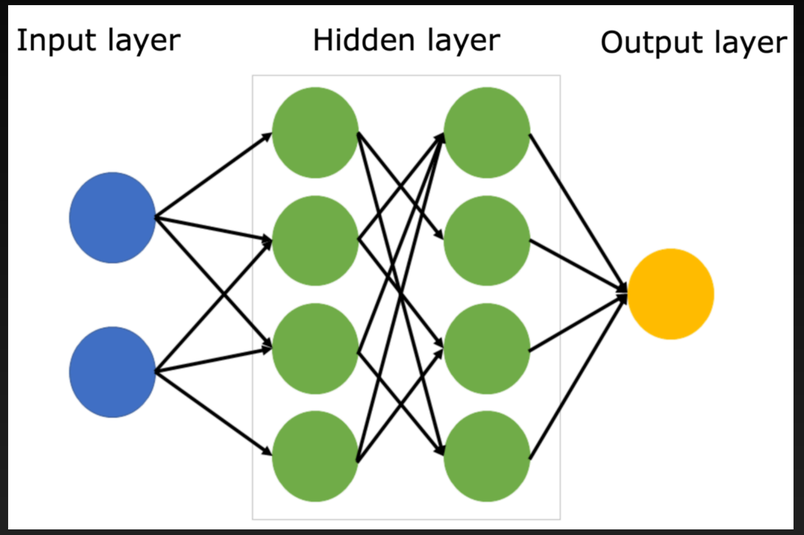
Would be pretty useful too

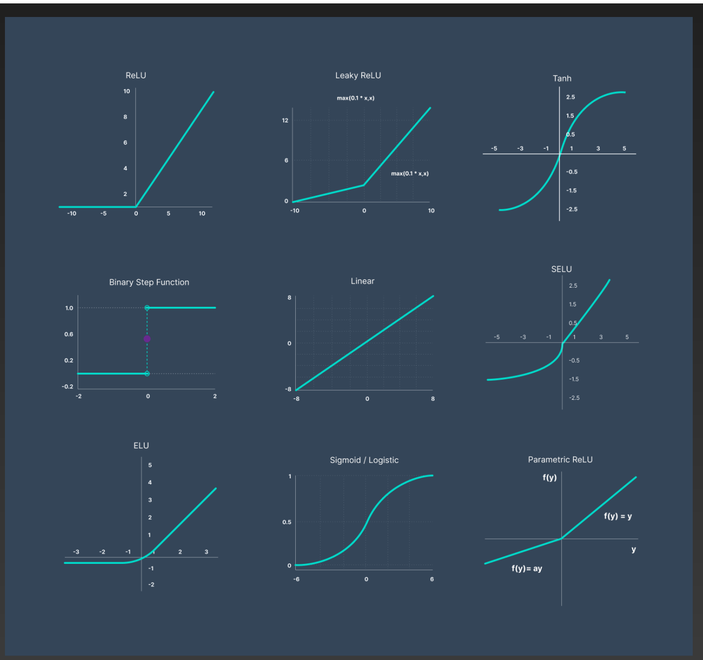
- The term machine learning was coined in 1959 by Arthur Samuel
- AI winter 1970s —> Pretty bad time for ML
- ReDiscovery of Backprop (its just fancy chain rule)



So what do neural networks look like







What are we trying to do

Lets take a step back

- Predict whether the sun will rise:

$$P(\text{Sun} = \text{Rise} \mid \text{Past}) = 1$$

- What about prediction of Neurodegeneration from MRI:

$$F : \text{Input} \longrightarrow \text{Output}$$

- By universal approximation theorem (simplified):
  Given a sufficient amount of depth and width a neural network can approximate a wide array of functions*

What are we trying to do

- By universal approximation theorem (simplified):
  Given a sufficient amount of depth and width a neural network can approximate a wide array of functions*

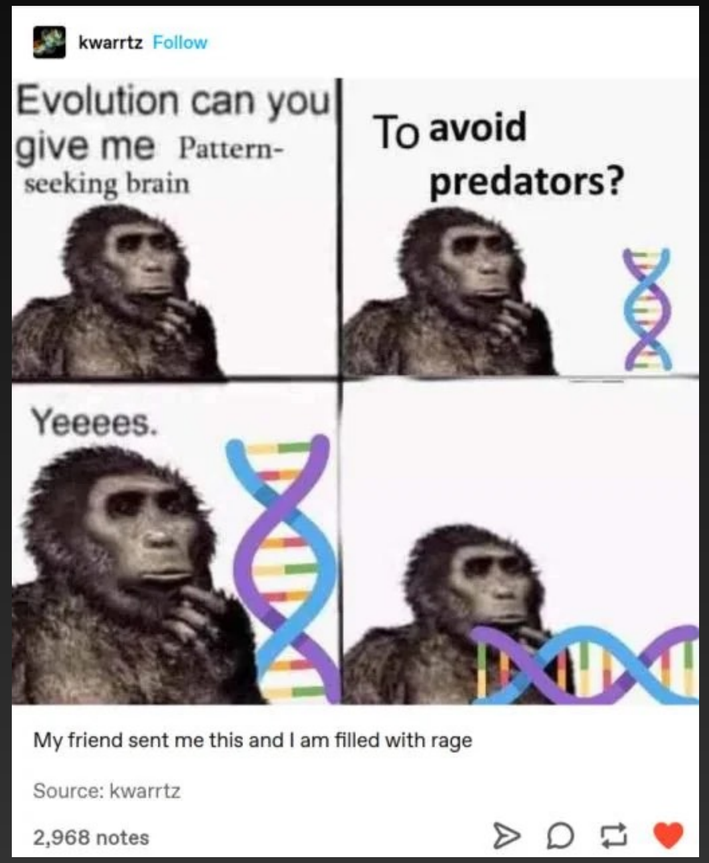
- *This is existence - there is no defined construction!

- Training (its an optimisation problem):
  1. Cost function
  2. Backprop
  3. Data

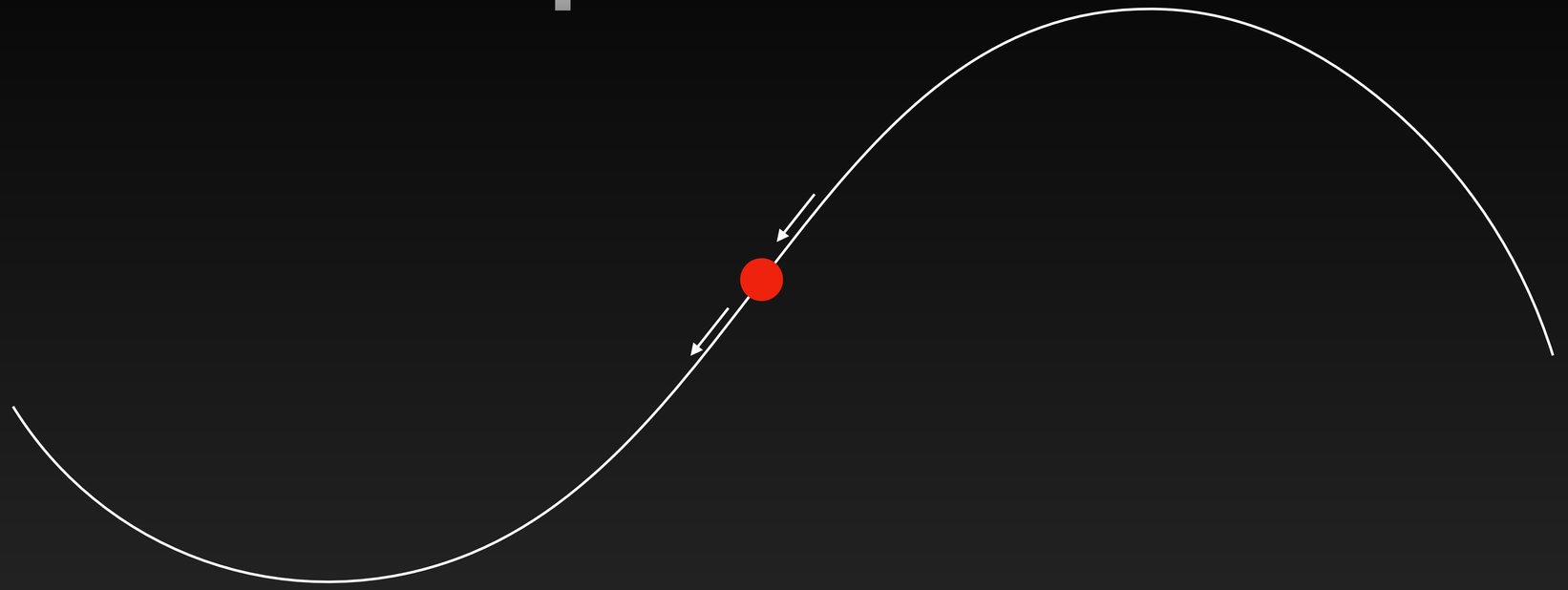
Loss function

- Something to optimise:
  - The line before
  - The amount of money in my bank account
  - How good the prediction of your model is
  - etc. etc. etc.

- We can mathematically define different types of loss which lead to different behaviours when training your neural network:
  - L1 = |y-x|
  - Cross-entropy loss
  - Reward function



Optimisation



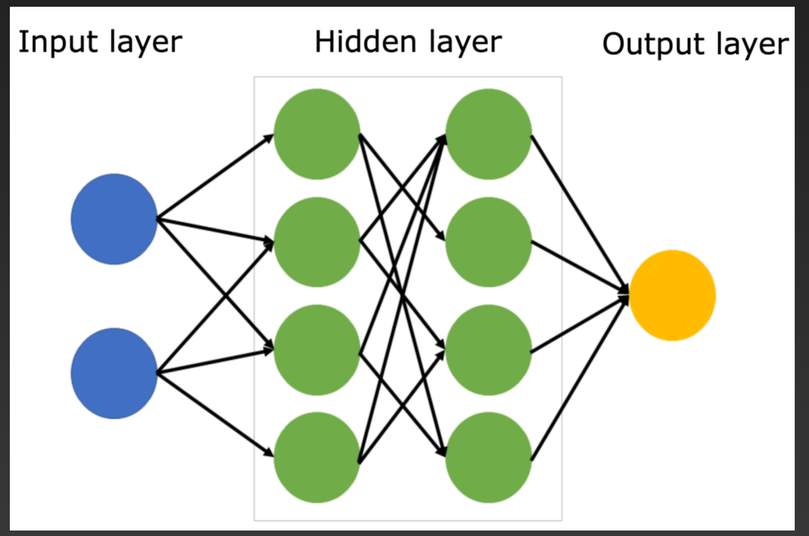
- Many ways to skin a cat - ant colony/particle swarm/Dynamic programming
- The most “popular” way: Gradient-based approaches

Backprop
“fancy chain rule”?

- Given $h(x) = f(g(x))$
- Then (given conditions): $h'(x) = f'(g(x))g'(x)$

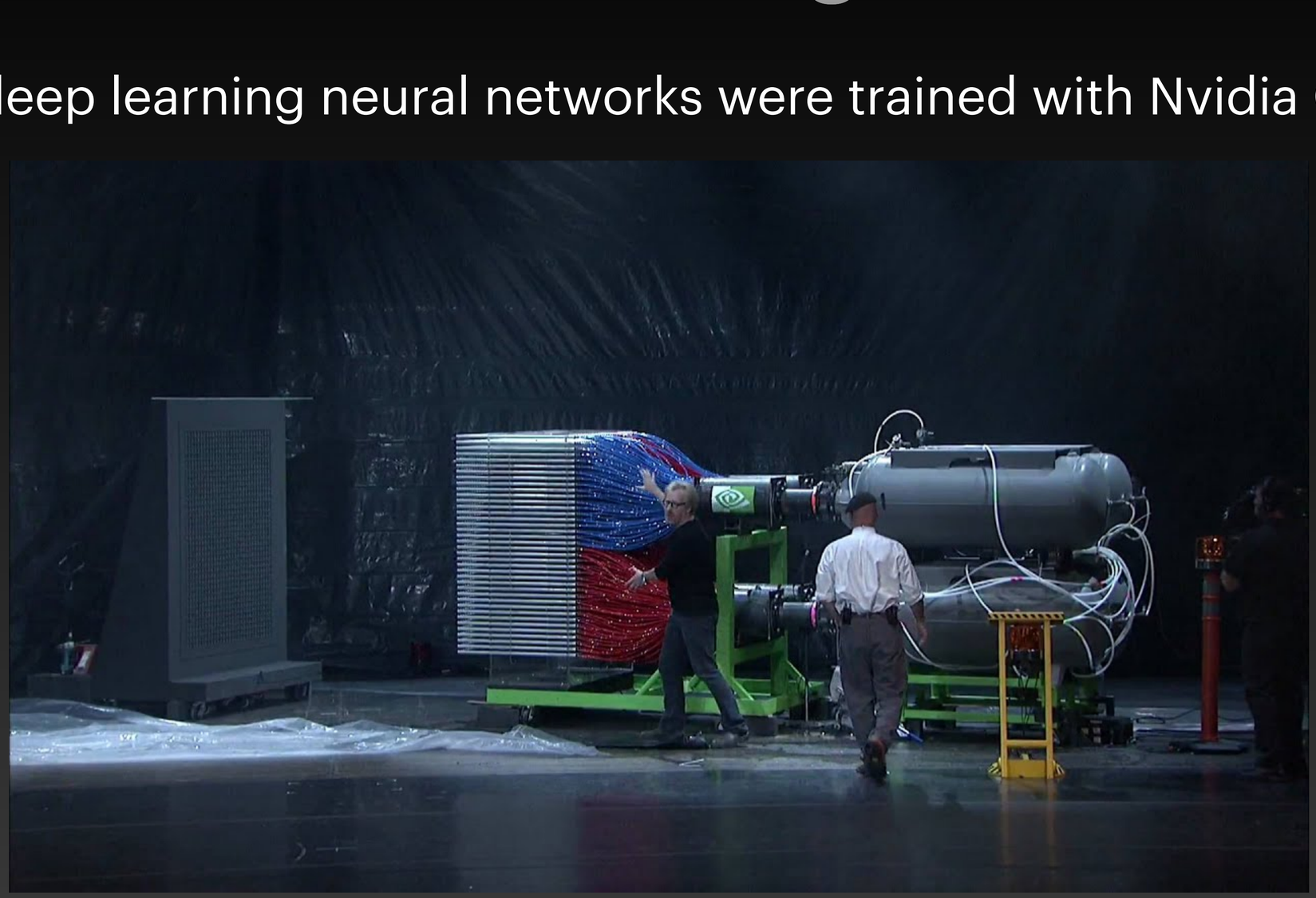
- “Knowing the instantaneous rate of change of z w.r.t y and y w.r.t x allows one to calculate the rate of change of z w.r.t x as the product of the two rates of change.”

- Neural networks: $Output = f_N(f_{N-1}(\ldots(f_0(Input))))$



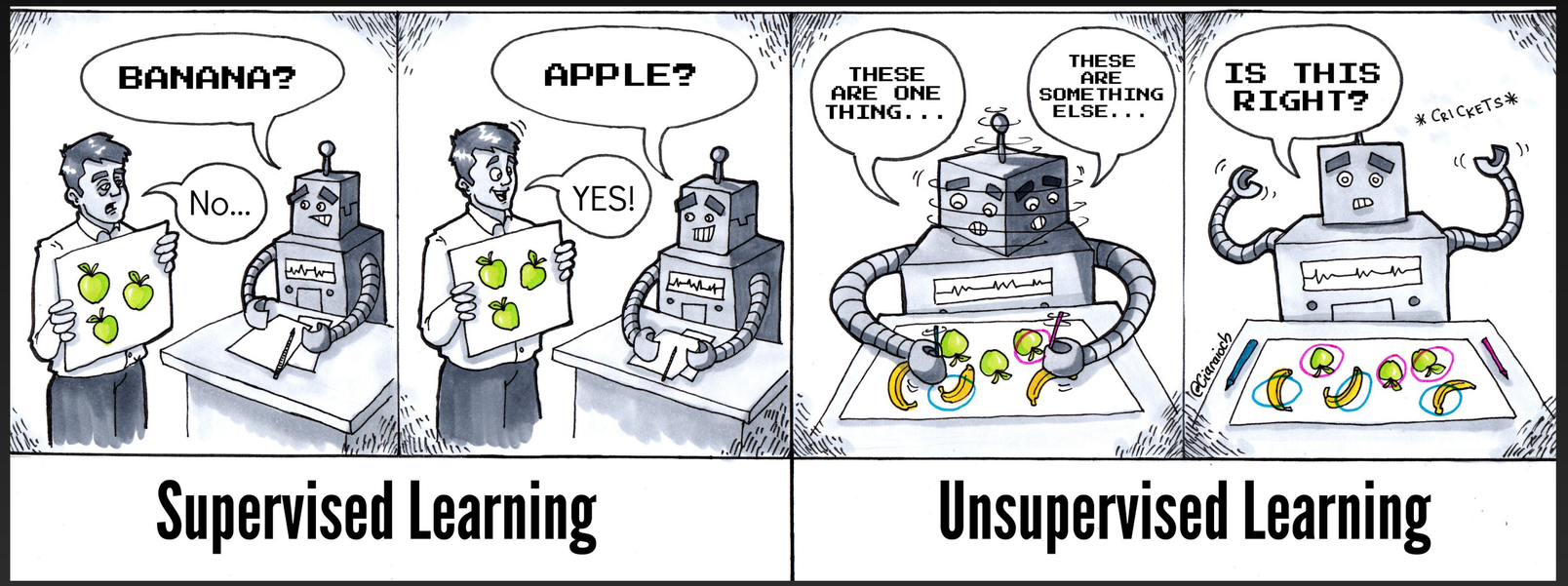
What was the other big breakthrough

- 2009: deep learning neural networks were trained with Nvidia GPUs



Three tracks of ML

Different cost functions lead to different things



How to implement NNs

There are a lot of ways

- From scratch
- JAX, TensorFlow, Keras
- Pytorch:
  - Pythonic
  - Big community
  - Lots of packages