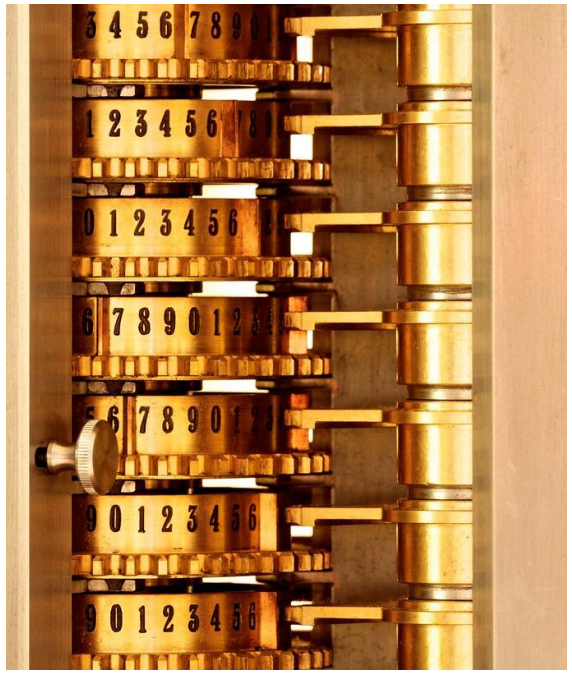
Computers, brains and AI

5: Deep learning





Difference engine by Marcin Wichary / CC BY
Ada Lovelace by Science Museum Group / CC0

1) Intro

2) Foundations
3) Silicon

4) Deep learning — Brains and neural networks
5) Deep learning — Learning

6) ChatGPT
7) ChatGPT
8) ChatGPT

9) The future of AI
10) The future of AI



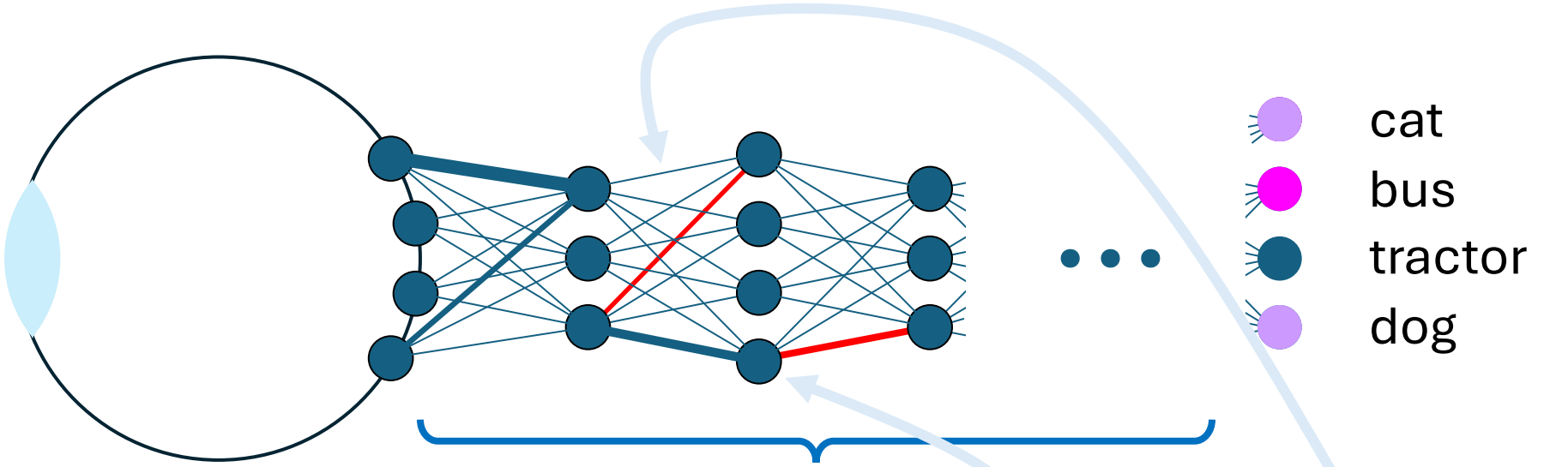
AlexNet

What AlexNet learns









Learning means choosing these numbers

Specifically, it means these weights…

… rather than these values…



DJ equipment by Paulo Guereta / CC BY 2.0

If you've ever tried to regulate the temperature in a hotel shower, you have glimpsed the challenges of deep learning! Instead of two taps, imagine millions of taps operated by squirrels.



If the squirrels have done a good job during training, the system will make the correct classifications for dogs, bus and cats and so on.

Training









dog

bus

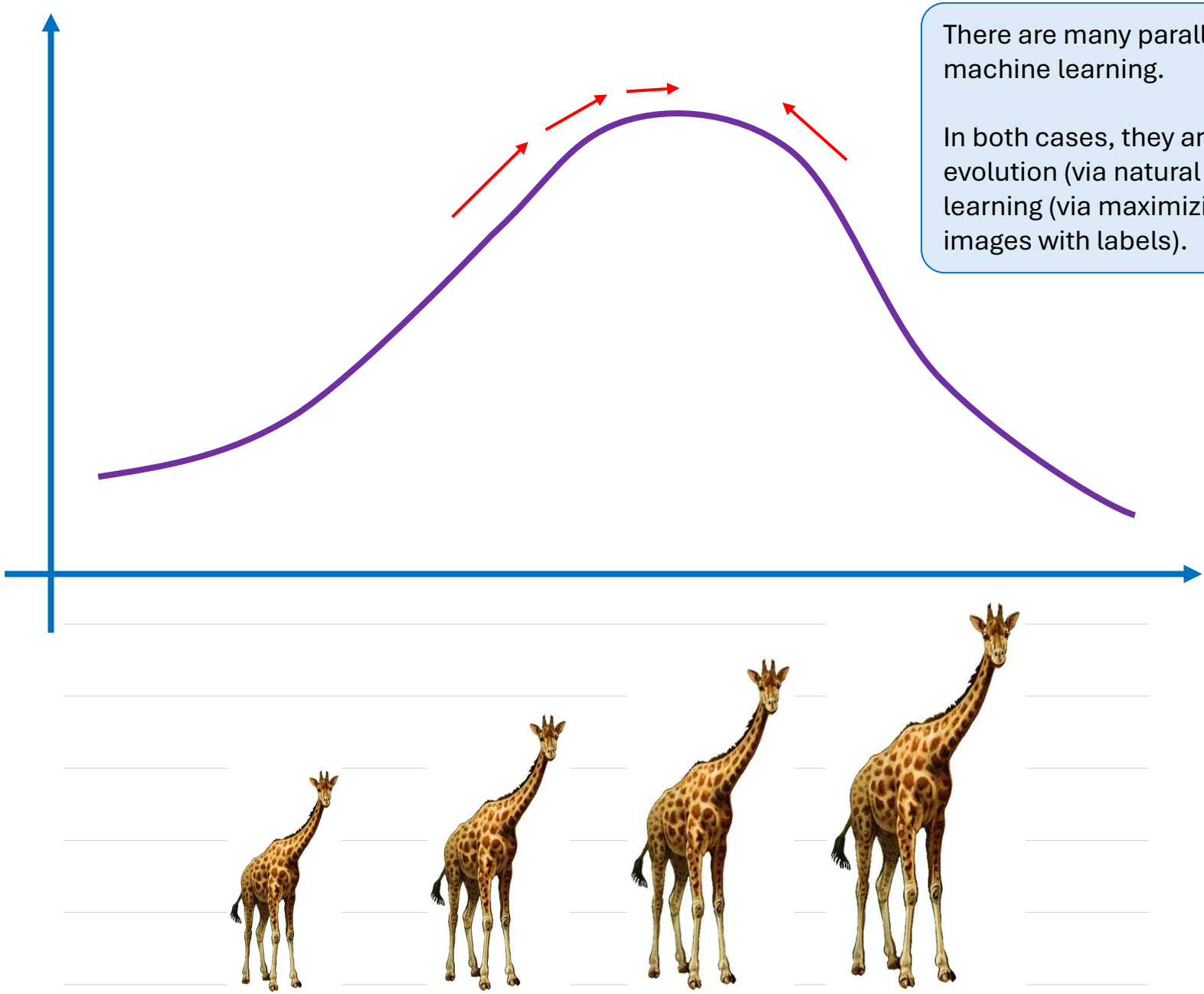
cat

Although AlexNet is shown pictures during training, what it learns is a lot of numbers (the weights). Oddly, it has no recollection of which specific pictures it was shown.

Deep learning
vs
Evolution



success



There are many parallels between evolution and machine learning.

In both cases, they are doing some optimization: evolution (via natural selection); and machine learning (via maximizing, say, its ability to match images with labels).



Giraffe / CC0

badness ?

cost ?

loss

In machine learning, things are usually drawn upside-down.

Gradient descent

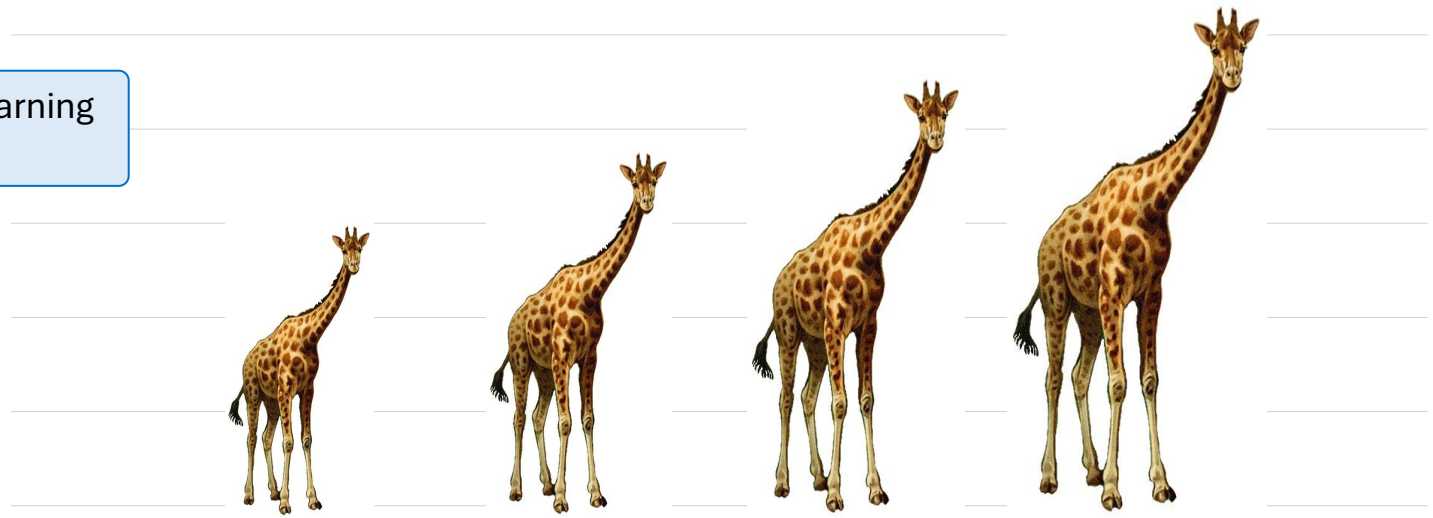
success

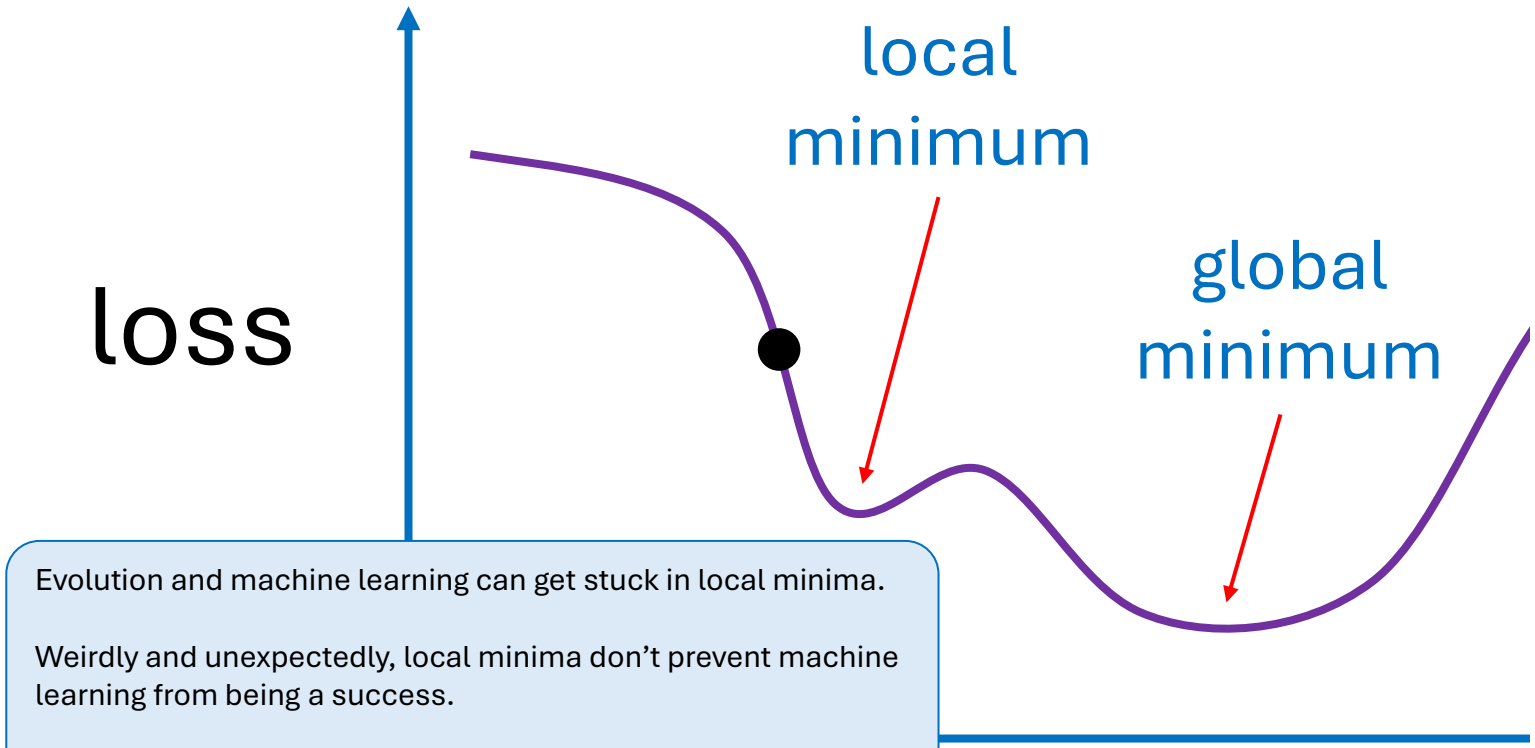
Neither machine learning nor evolution can see very far in either direction (because they are trying small changes).

In other words, it is like they are walking around in the fog.

height

The word, loss, in deep learning says how bad things are.

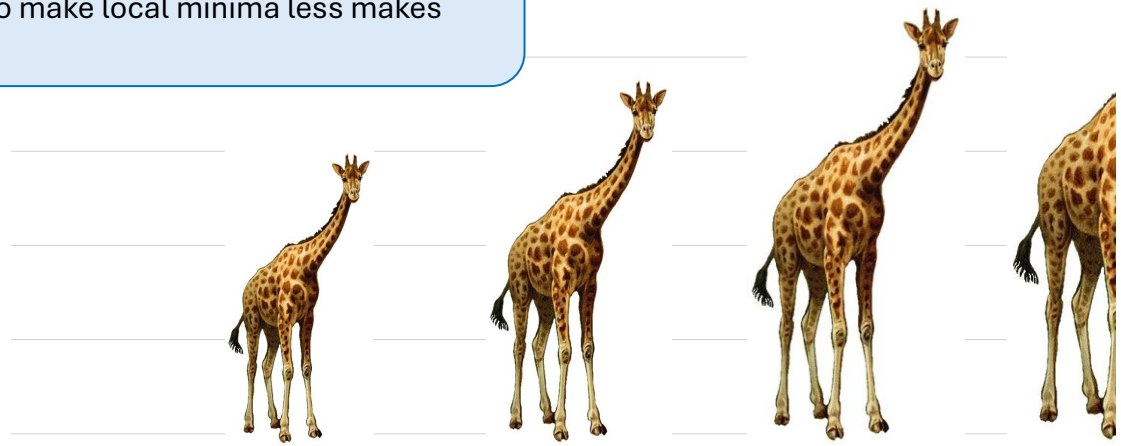


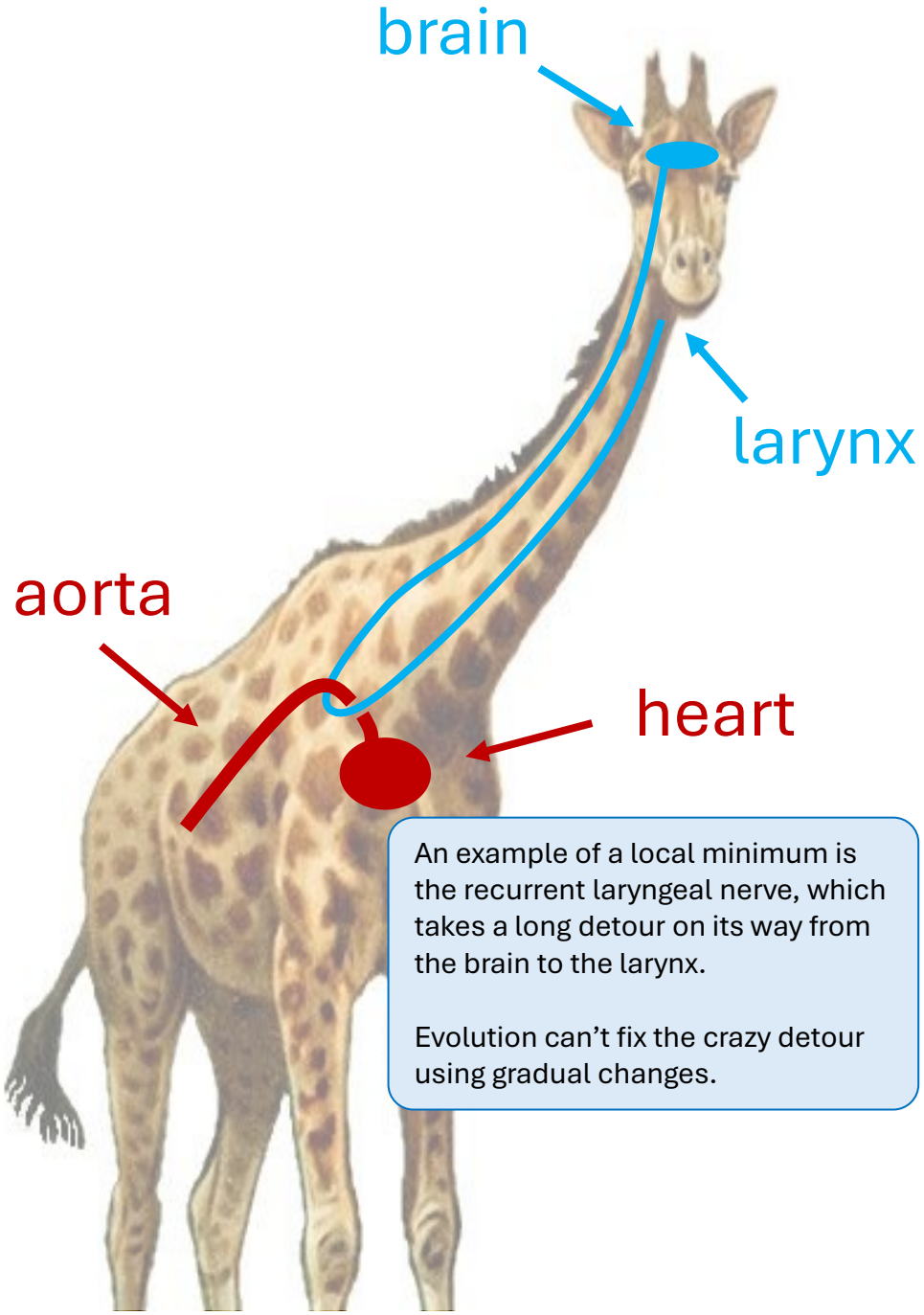


Evolution and machine learning can get stuck in local minima.

Weirdly and unexpectedly, local minima don't prevent machine learning from being a success.

The graph shows the effect of varying just one hypothetical parameter (the giraffe's height). In practice, there will be many parameters, which seems to make local minima less makes less likely.





An example of a local minimum is the recurrent laryngeal nerve, which takes a long detour on its way from the brain to the larynx.

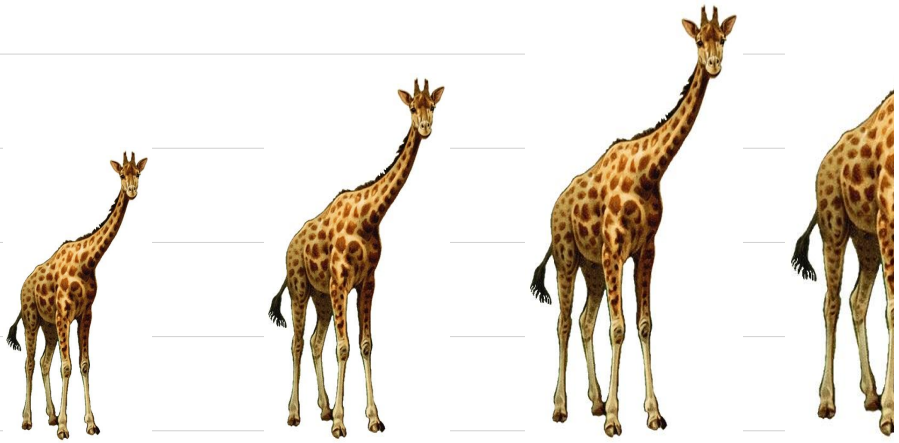
Evolution can't fix the crazy detour using gradual changes.

Giraffe / CC0

loss

local minimum

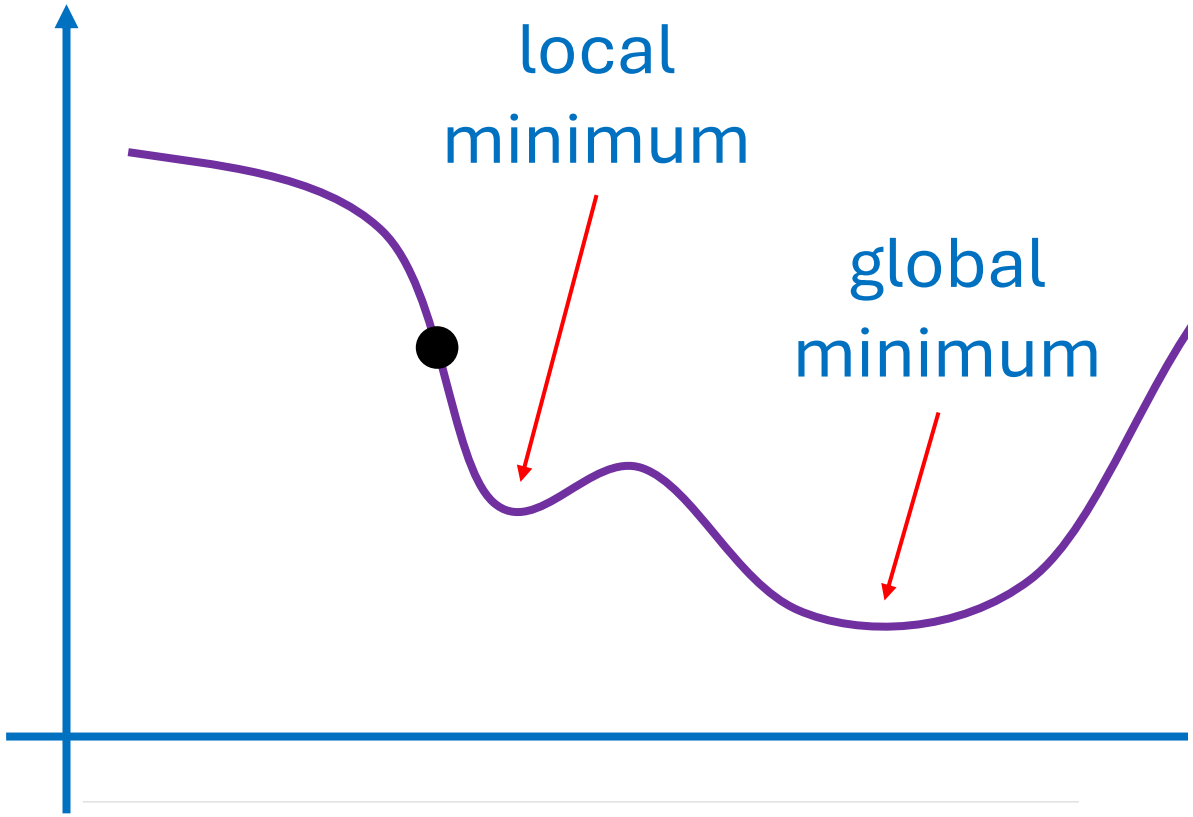
global minimum

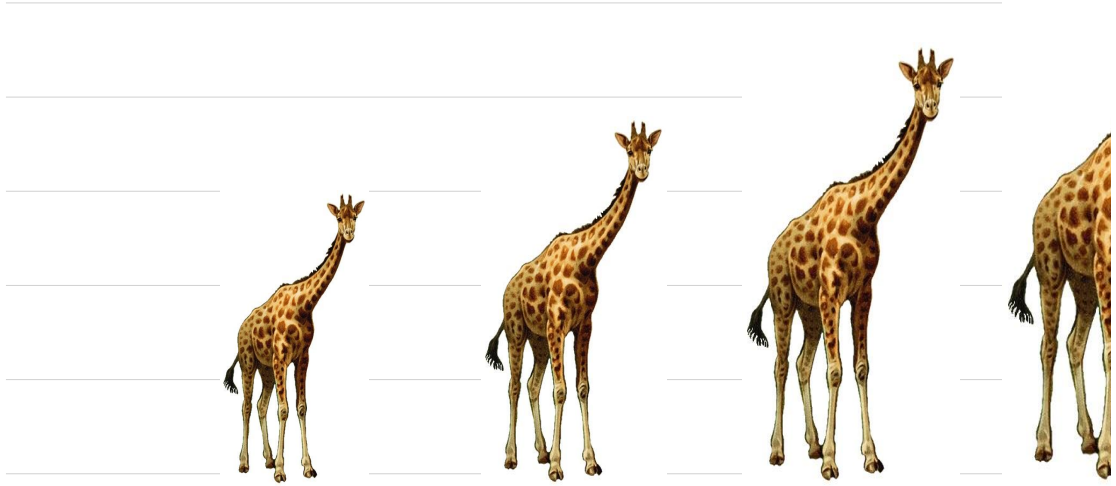


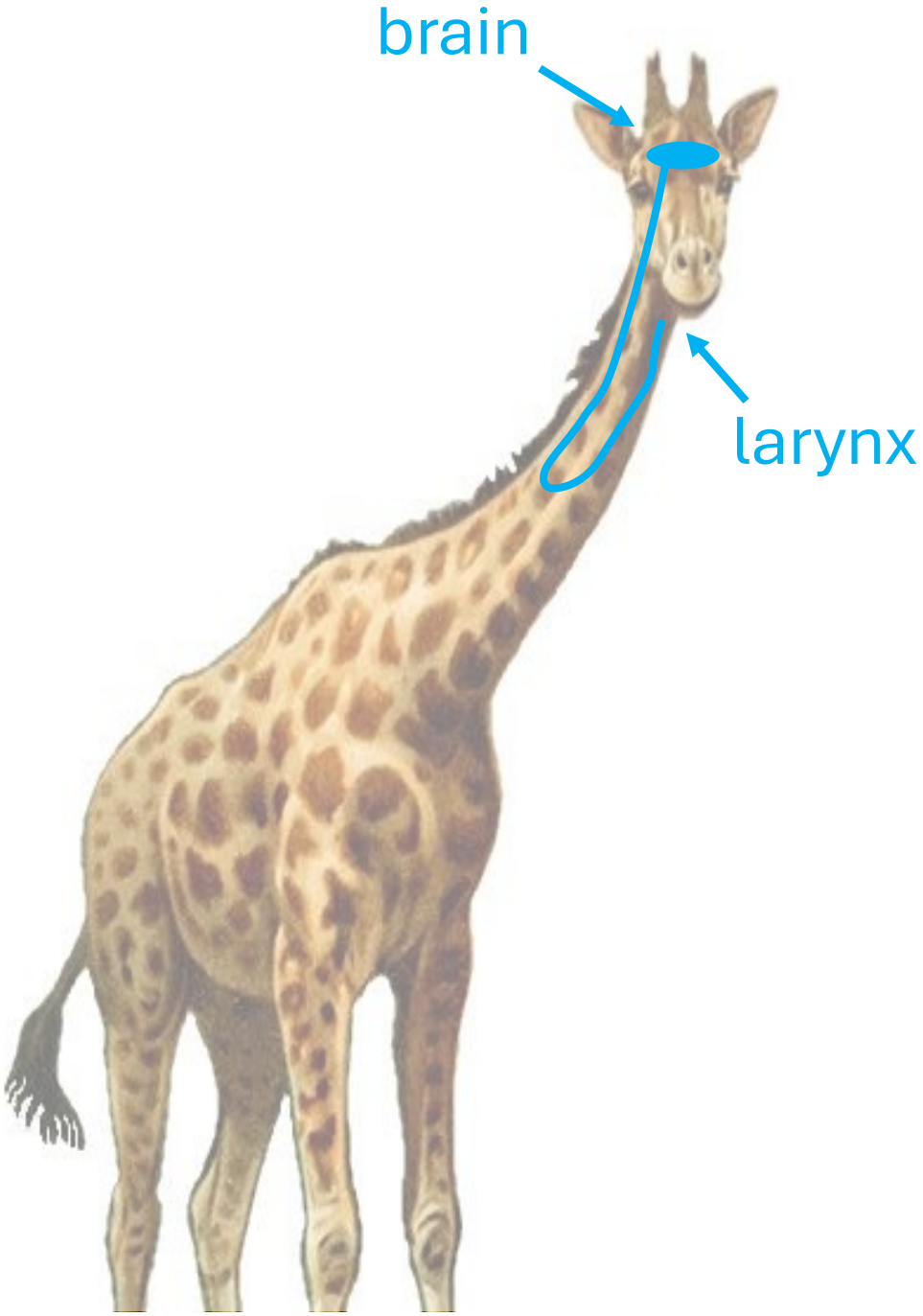
brain

larynx

loss











Giraffe / CC0



These lakes are trapping water in local minima and preventing the water from flowing down to the sea.

Marichini lakes (tarns) / CC0

How AlexNet learns

Here are some ways that don't seem to work.

~~One photo at a time~~

1. Learning about this cat will be undone when it learns about the bus etc.



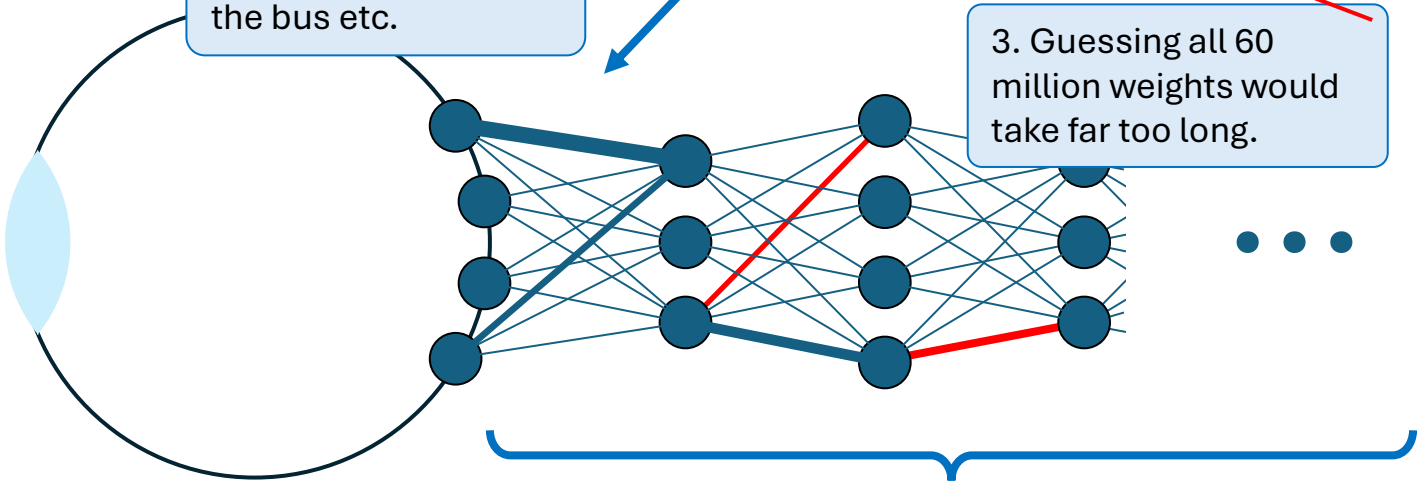




~~One weight at a time~~

~~Guessing ?!~~

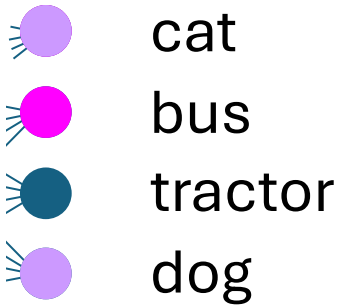
3. Guessing all 60 million weights would take far too long.



choose these numbers!

?

2. Learning about a weight in isolation is no good because it is the combination of weights that matters.



Learning from **labelled data** is called **supervised learning**.

10^{160} permutations of two packs of cards

10^{140}

10^{120}

10^{100} the observable universe as a computer

10^{80} atoms in the observable universe

permutations of a pack of cards

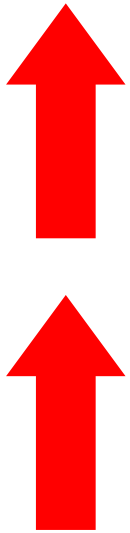
10^{60}



10^{40}

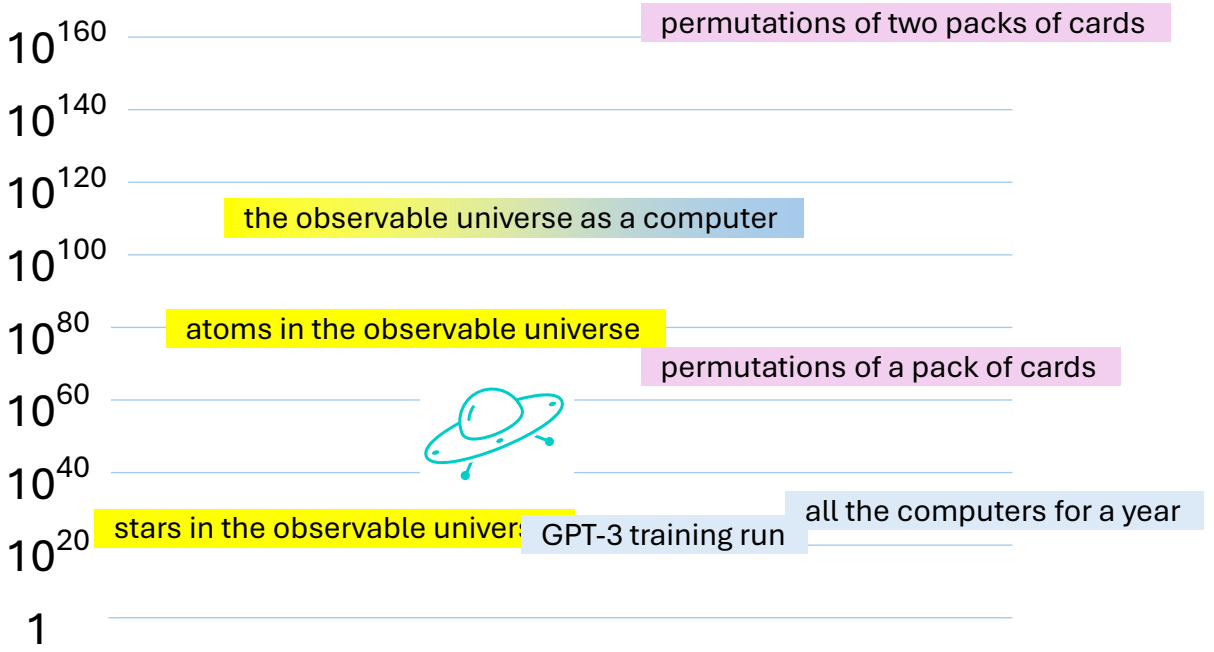
10^{20} stars in the observable universe GPT-3 training run all the computers for a year

1



Number of guesses

The number of possible guesses would dwarf all these other numbers.





~~One photo at a time~~

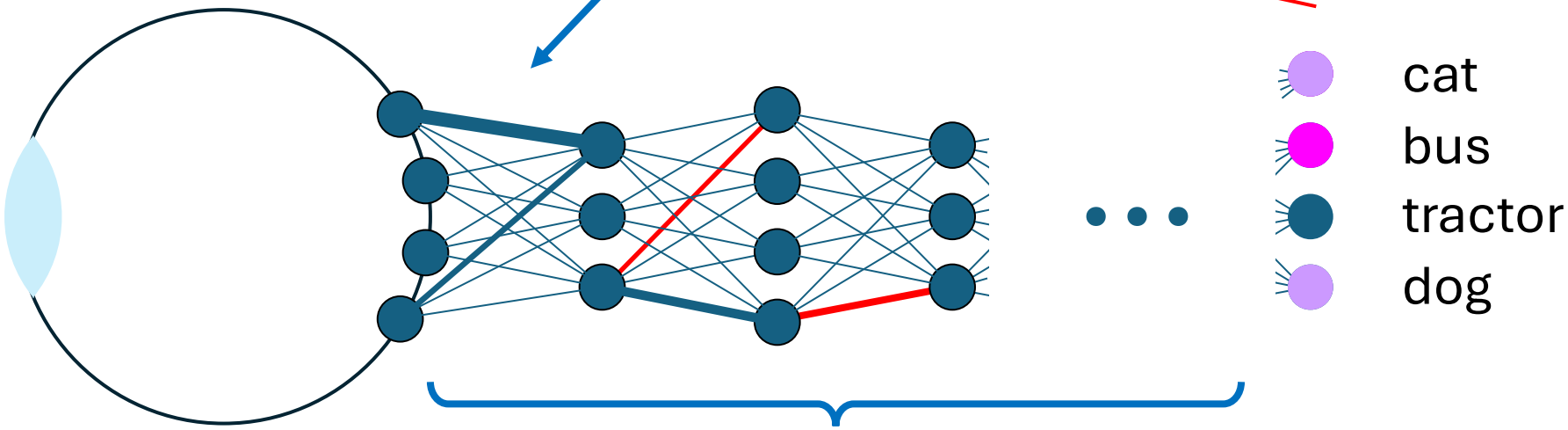
~~One weight at a time~~

~~Guessing ?!~~









choose these numbers!

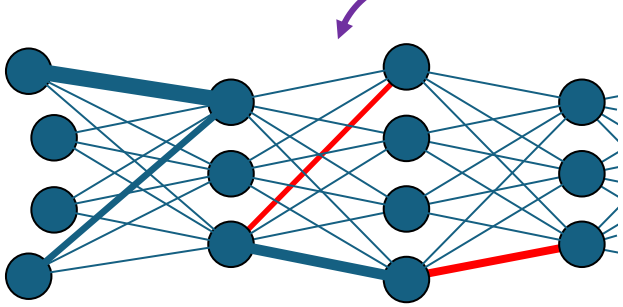
?

The algorithm

A newborn AlexNet hasn't seen any images yet.

We are calling it newborn because, like a baby, it hasn't learnt anything yet.

Random weights





- Start with a newborn AlexNet

During five days of training, AlexNet takes more than 800,000 of these tiny steps. Each step involves nudging all the weights.

- Improve things a *tiny* step at a time



somehow!

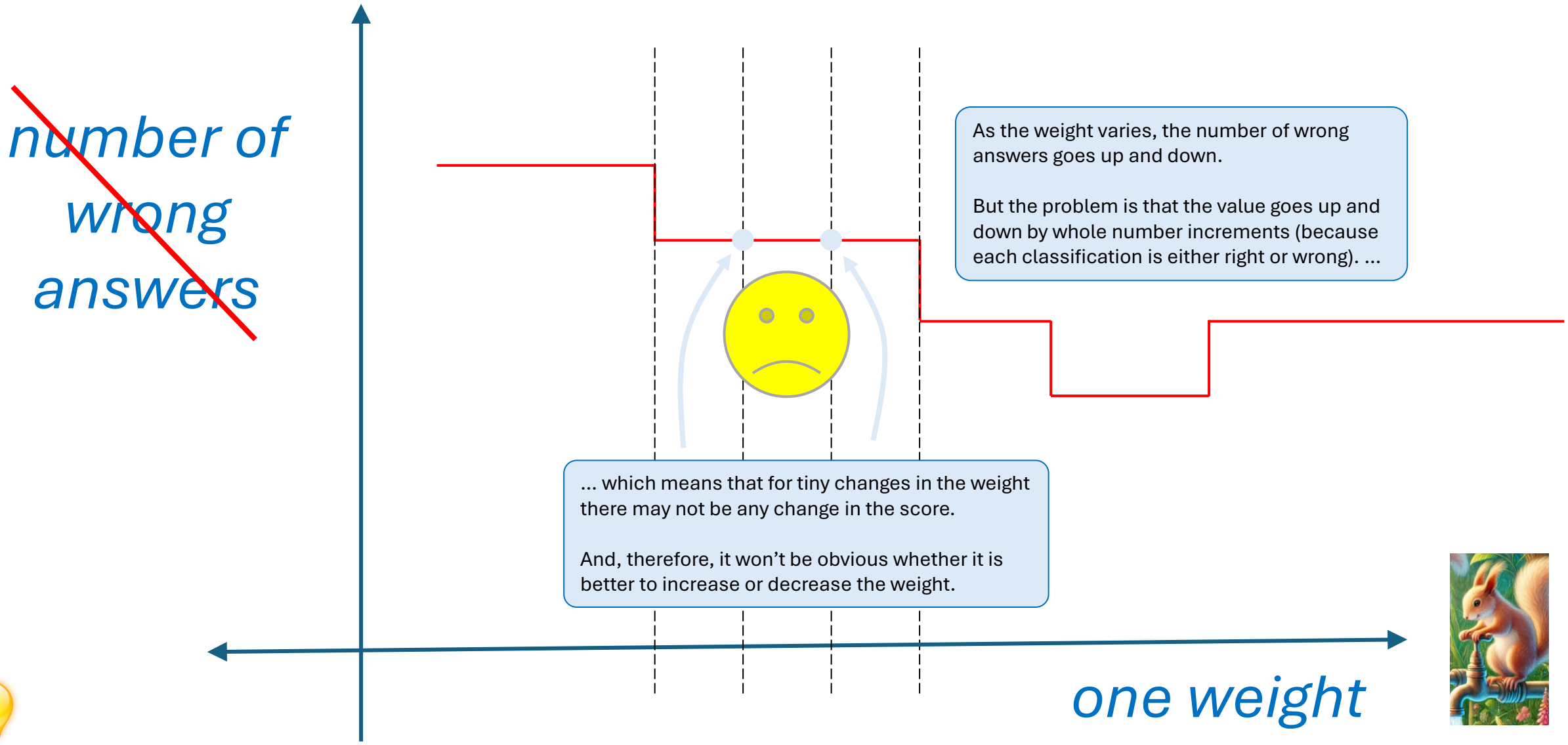
?

- (by adjusting all the weights)

? ❖ (by adjusting all the weights)

? ❖ (by adjusting all the weights)

Although, at each step, all the weights are being adjusted, it is instructive to think about the effect of just one weight at a time.



As the weight varies, the number of wrong answers goes up and down.

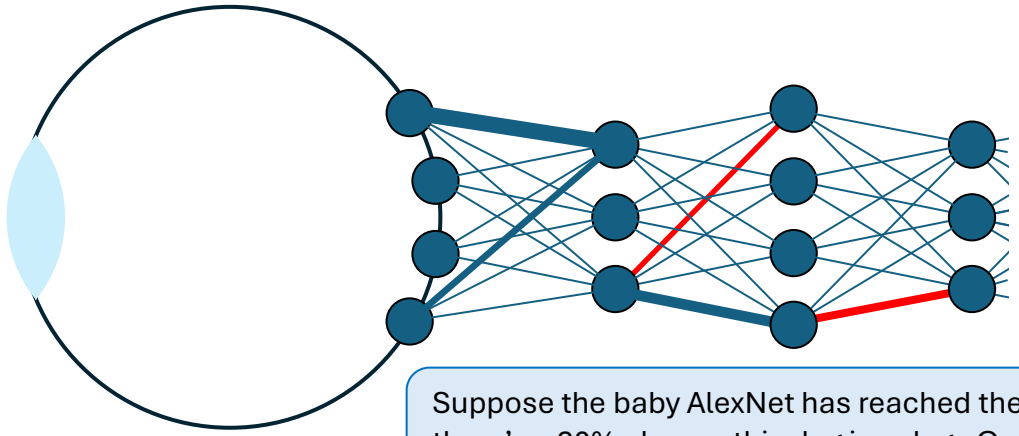
But the problem is that the value goes up and down by whole number increments (because each classification is either right or wrong). …

… which means that for tiny changes in the weight there may not be any change in the score.

And, therefore, it won't be obvious whether it is better to increase or decrease the weight.

dog





50% cat
bus
tractor
20% dog

cat

Suppose the baby AlexNet has reached the stage where it thinks there's a 20% chance this dog is a dog. On the one hand, its final answer, "cat", is wrong. On the other hand, it ought to get some credit for thinking it *might* be a dog. And, importantly, it should get more credit for increasing 20% to, say, 21%, something that might be achievable from those tiny training steps.

1%

20% ⟶ 21% ✓

60% ⟶ 61% ✓

~~number of wrong answers~~

~~tell it when it's wrong~~

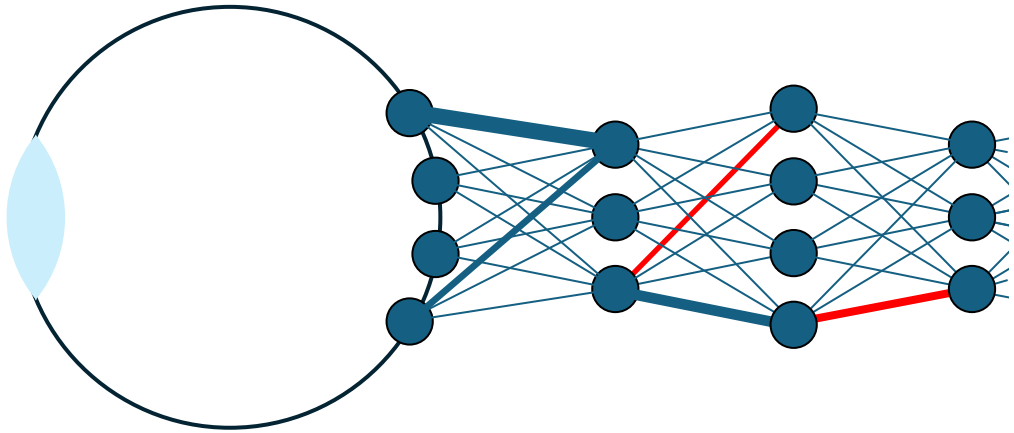
Even if the AlexNet is doing well enough to give "dog" a probability of 60%, it is still a good idea to increase that to probability, say, "61%", for two reasons:

1. Subsequent training on other images may make the 60% decay, and so it is good to have a safety margin.

2. (More importantly) it isn't interested in that particular dog. Instead, it wants to improve its recognition of dogs in general.

dog





50% cat
bus
tractor
20% dog

loss



20% 80% 0.8



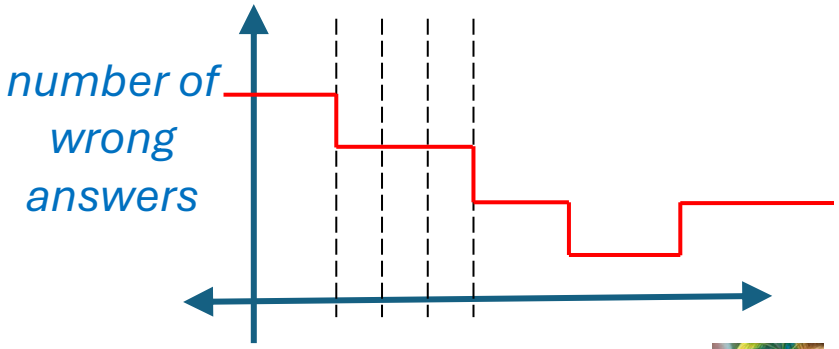
70% 30% 0.3

all the images

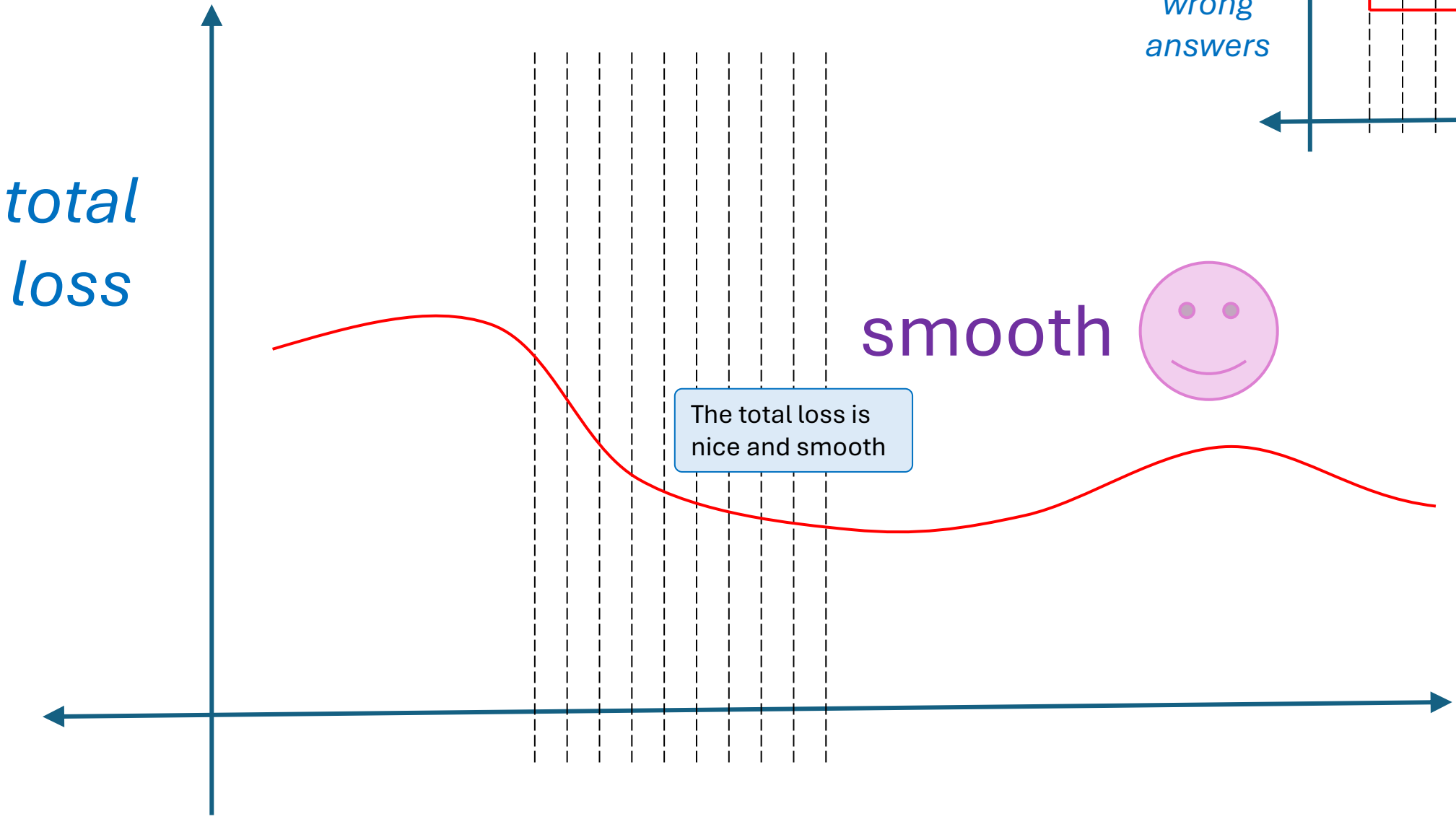
We are interested in minimizing the number of wrong answers, but it turns out to be better to optimize for total loss instead. Total loss is acting as a proxy for the number of wrong answers.

total loss

The **total loss** is the sum of the losses of all one million images.


number of
wrong
answers




total
loss
smooth
The total loss is
nice and smooth

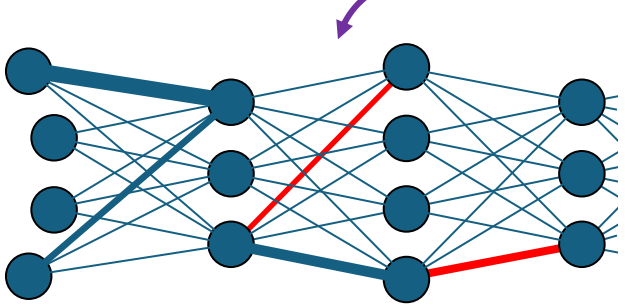




This is a summary of what we've just seen.

The algorithm

Random weights

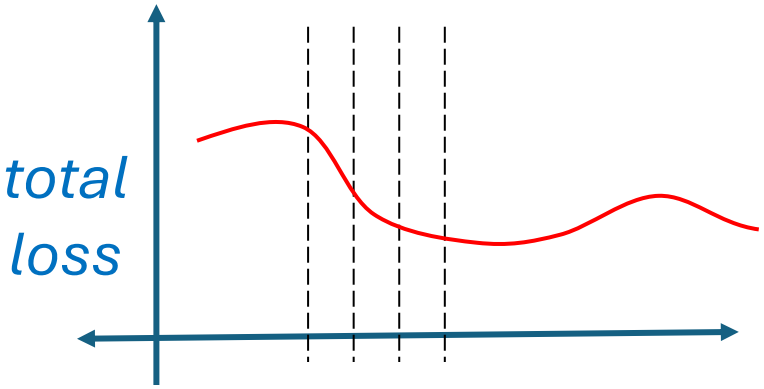




- Start with a newborn AlexNet
- Improve things a *tiny* step at a time



?
- (by adjusting all the weights)





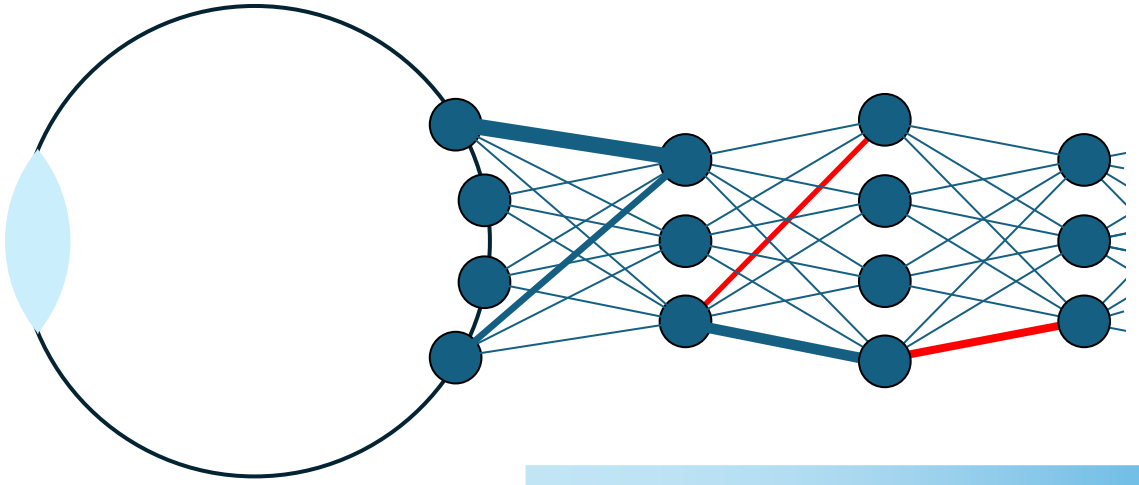


In the next few slides, we will see a couple of important improvements for making the learning faster: (1) **gradient descent** using **backpropagation**, and (2) **stochastic gradient descent**.



dog





50% cat

bus

tractor

20% dog

First the calculations feed forward through the neurons, keeping notes of all the intermediate values.

Then the backpropagation algorithm goes backwards working out the effect each weight has on the overall answer. Or, to put it more mathematically, it calculates the partial derivatives (the gradients) of the loss function with respect to each weight.

backpropagation

Backpropagation was discovered and rediscovered many times.



Gradient descent

backpropagation

total loss

For each parameter, backpropagation calculates the gradient, which says which direction to adjust the parameter.

gradient







Slope by Bouwe Brouwer / CC0

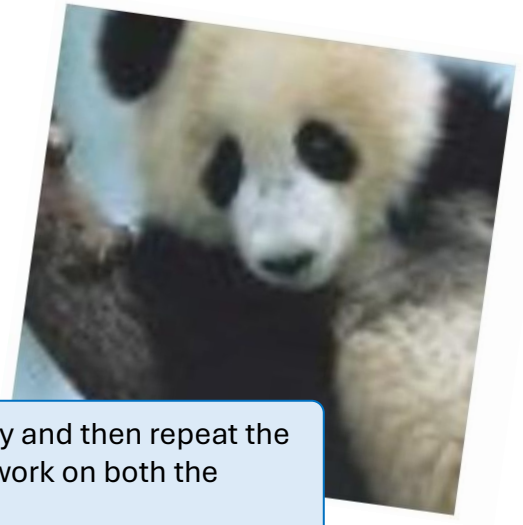
Adversarial machine learning

Shockingly, image classifiers can be fooled with tiny changes. It is shocking, not least, because it demonstrates that machine learning is weird and alien and inscrutable.



\+

carefully chosen tiny changes



=



panda

58% confidence

gibbon

99% confidence

A practical defence is to distort the image slightly and then repeat the classification. The attack is much less likely to work on both the original and the distorted image.

https://arxiv.org/pdf/1412.6572

Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy Google Inc. (2015)