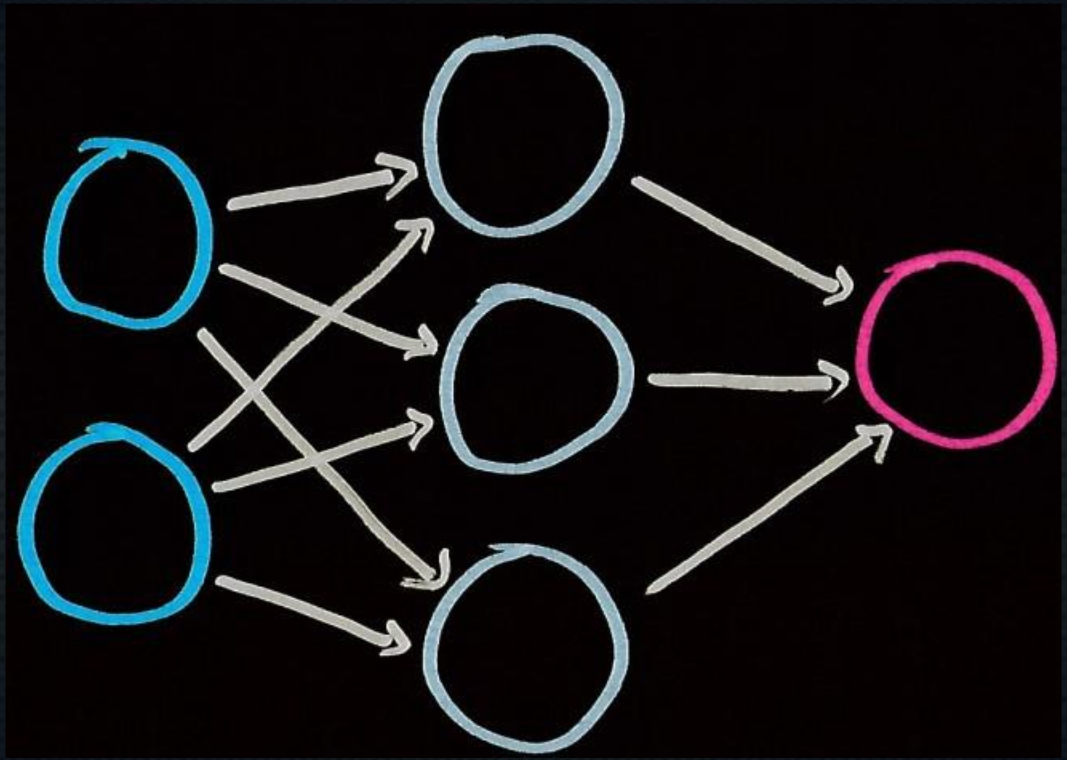
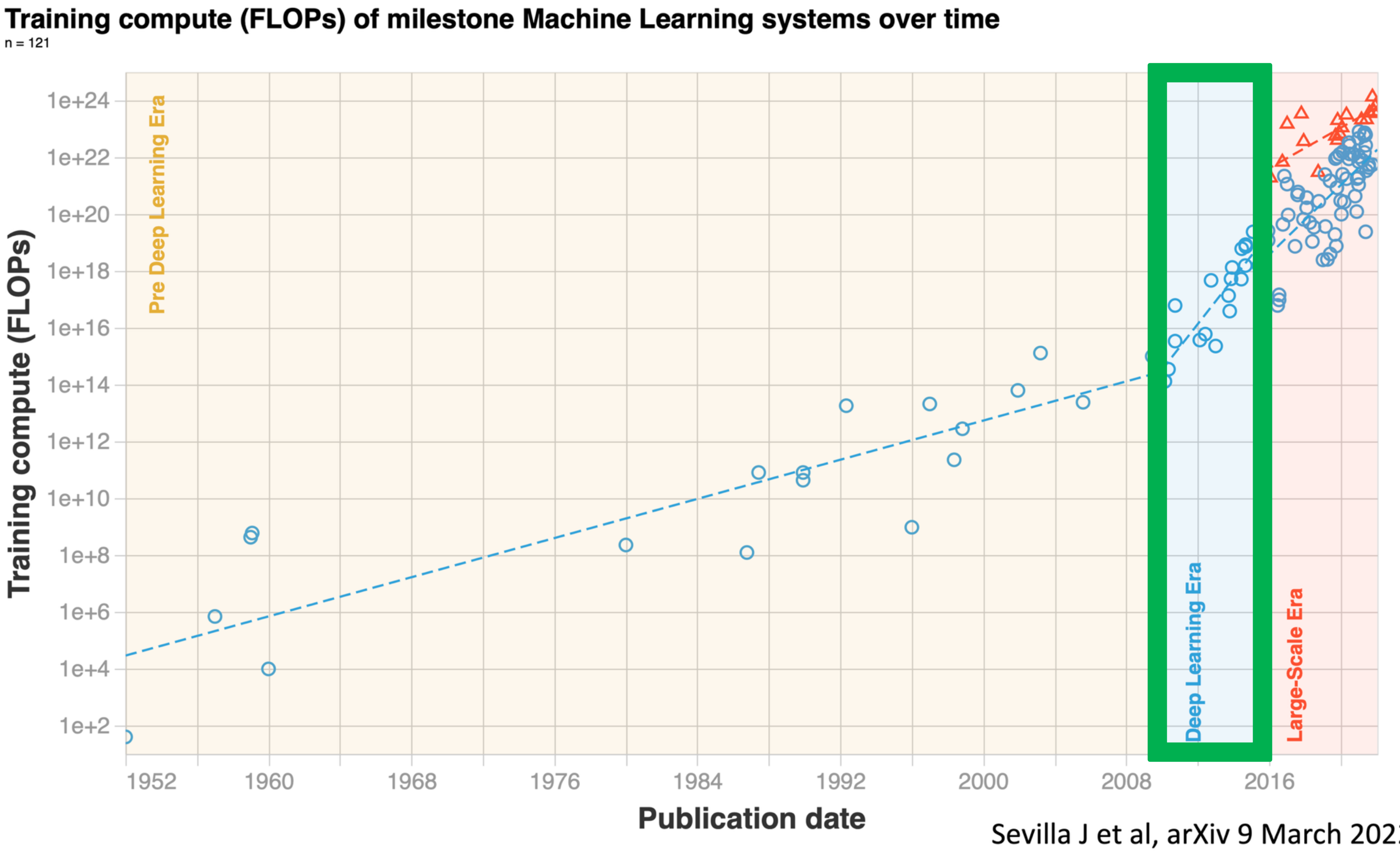
Switching gears: neural networks



Source: https://thesharperdev.com/build-your-first-neural-network-part-2/

Why neural networks?

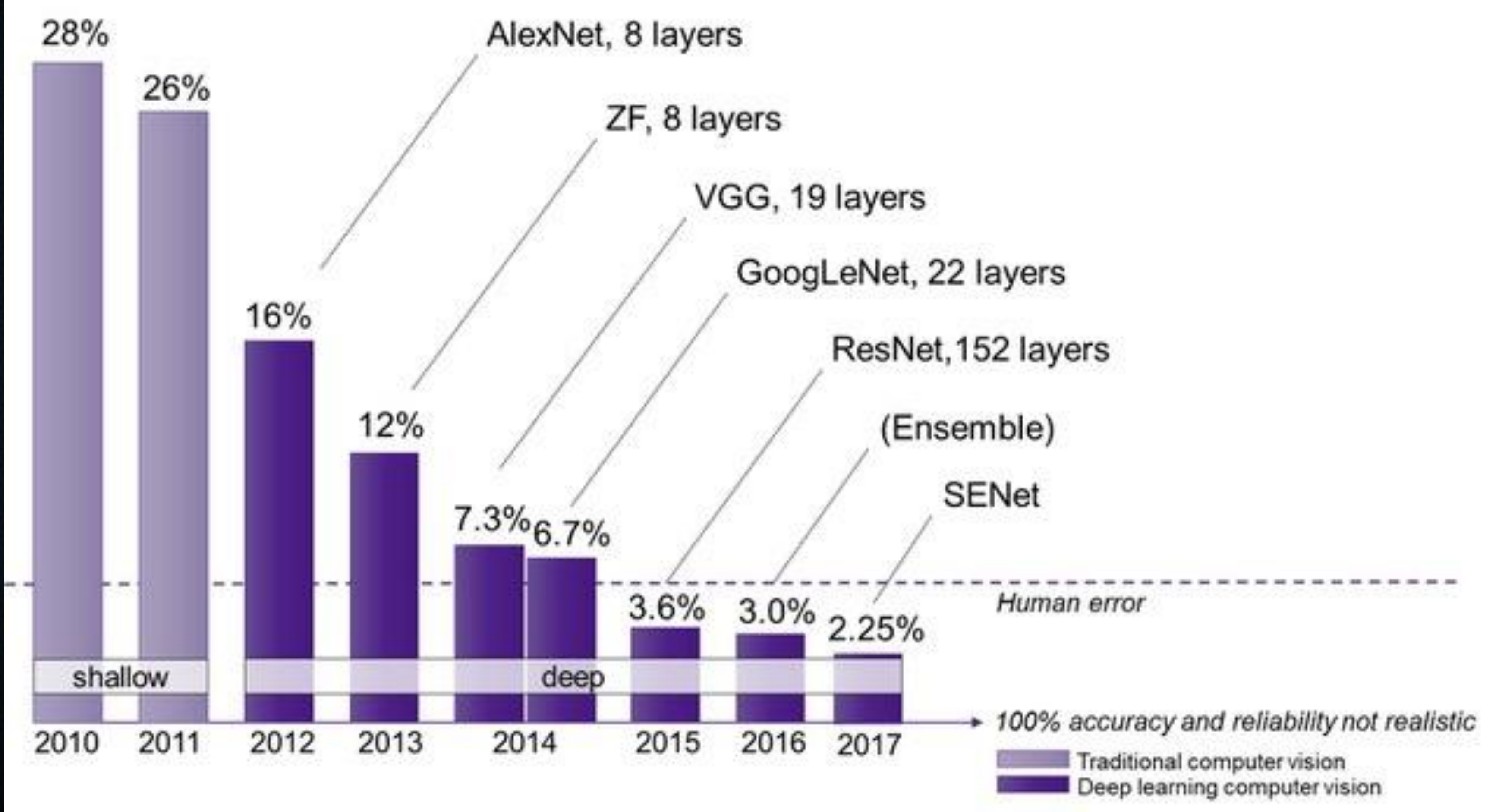
Deep learning revolution



Sevilla J et al, arXiv 9 March 2022

https://epochai.org/research/ml-trends

How did that happen?

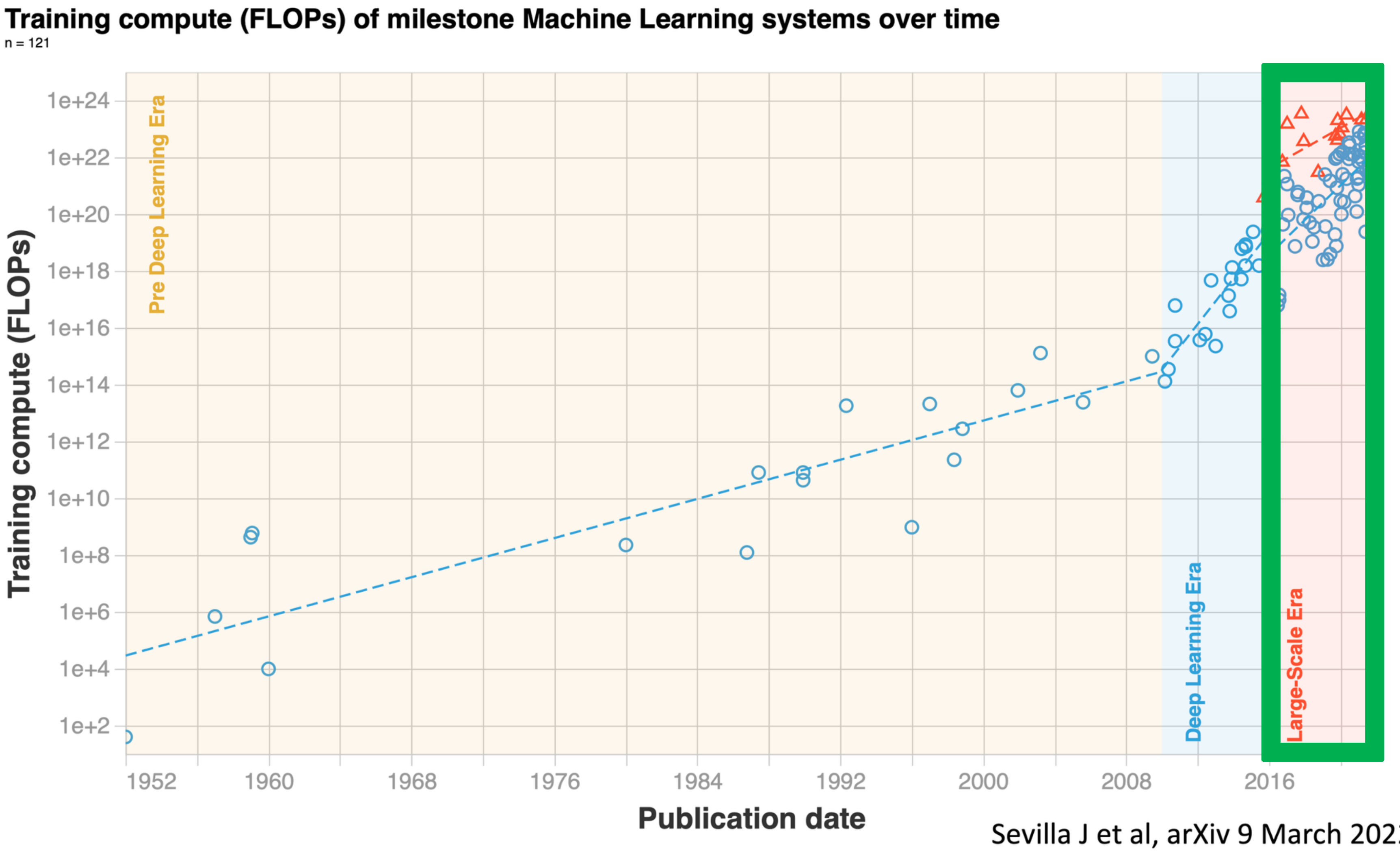


How did that happen?

Much of the credit for this revolution should go to the pioneers who spent many years developing the technology of CNNs, but the essential missing ingredient was supplied by FeiFei et al.[7] who put a huge effort into producing a labeled dataset that was finally large enough to show what neural networks could really do.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90.

What happened here?



Sevilla J et al, arXiv 9 March 2022

https://epochai.org/research/ml-trends

And now we have this

The main soundtrack of an arcade game. It is fast-paced and upbeat, with a catchy electric guitar riff. The music is repetitive and easy to remember, but with unexpected sounds, like cymbal crashes or drum rolls.

https://google-research.github.io/seanet/musiclm/examples/

Linus Ericsson, Henry Gouk, Chen Change Loy, and Timothy M. Hospedales

Self-Supervised Representation Learning

Introduction, advances, and challenges

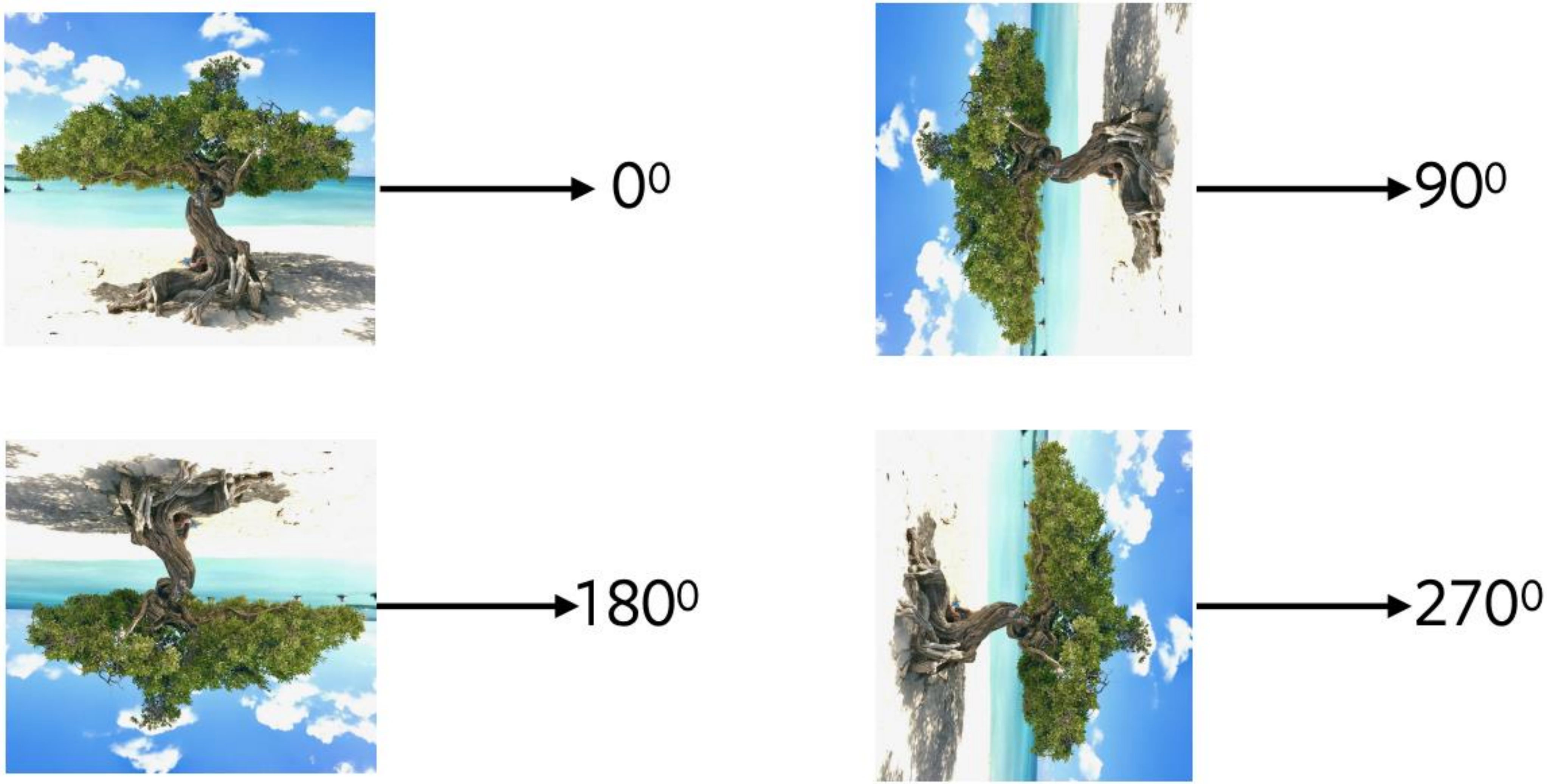
Self-supervised representation learning

We can learn meaningful representations via self-supervision and then use these in downstream tasks that we care about

↓

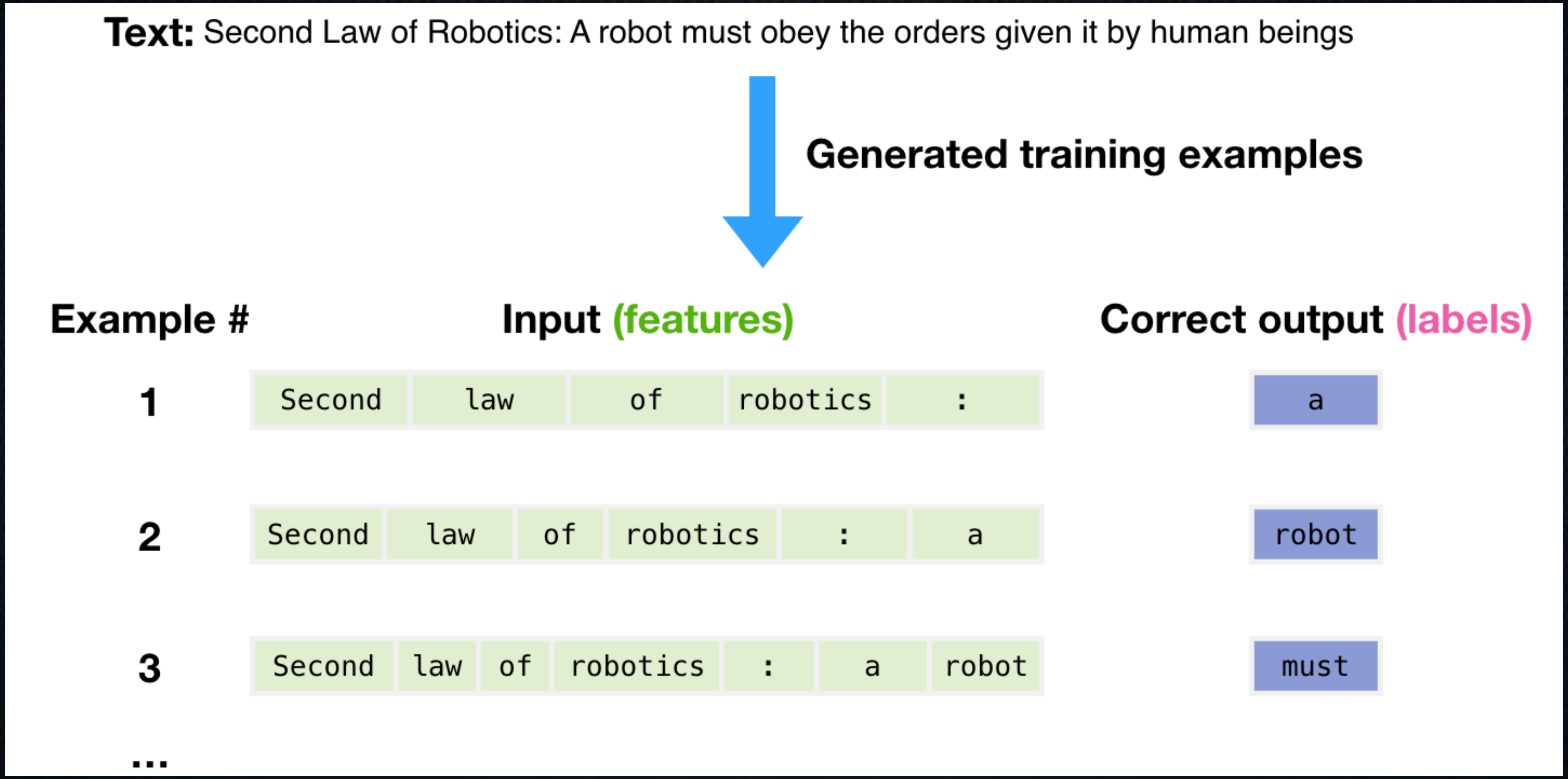
Learn useful stuff from unlabeled data, fine-tune with small amount of labeled data

Self-supervised representation learning



Gidaris et al., 2018, Predicting Image Rotations

Self-supervised representation learning



https://jalammar.github.io/how-gpt3-works-visualizations-animations/

Self-supervised representation learning

1,400,000,000,000 tokens

Common Crawl

70,000,000,000 parameters

Random	25.0%
Average human rater	34.5%
GPT-3 5-shot	43.9%
Gopher 5-shot	60.0%
***Chinchilla* 5-shot**	**67.6%**
Average human expert performance	*89.8%*
June 2022 Forecast	57.1%
June 2023 Forecast	63.4%

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., ... & Sifre, L. (2022). Training Compute-Optimal Large Language Models. arXiv preprint arXiv:2203.15556.

The bitter lesson

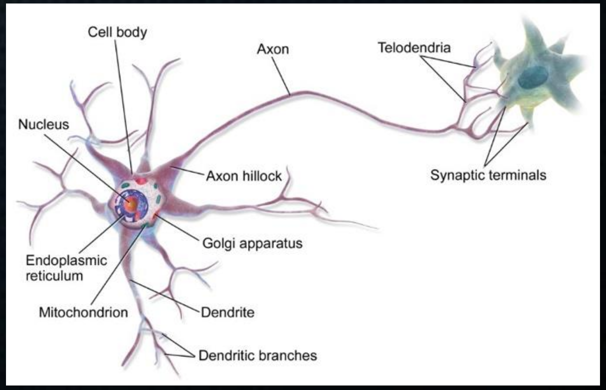
More compute + more data
>> cool human ideas*

http://incompleteideas.net/IncIdeas/BitterLesson.html

*However, note:
- A better lesson (Brooks)
- The use of inductive biases in CNNs and GNNs.

Why neural networks?

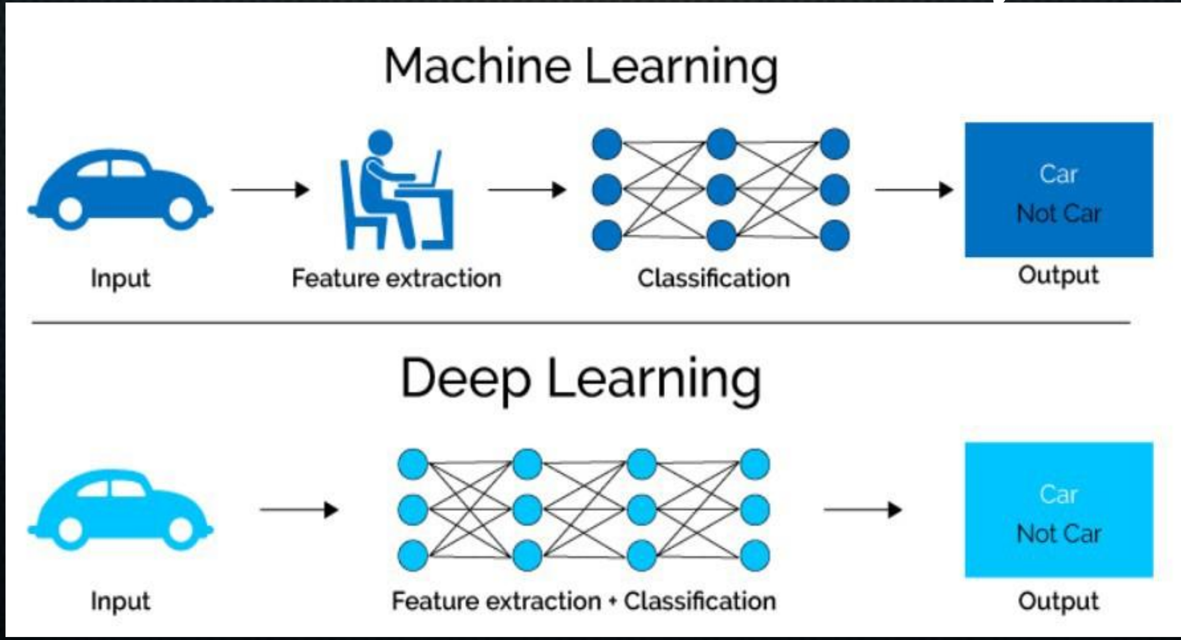
- Best performing algorithms for complex tasks, bar none.
- Known potential of hierarchical organisation of simple units because of biological examples (though neural networks are not good models of actual neurons)
- Observation in frogs and cat visual cortex: there are specific layers of neurons, where earlier layers detect basic shapes (lines, edges) with later layers incorporating this information into more complex features about what is seen.



Source: https://en.wikipedia.org/wiki/Axon#/media/File:Blausen_0657_MultipolarNeuron.png

Why neural networks?

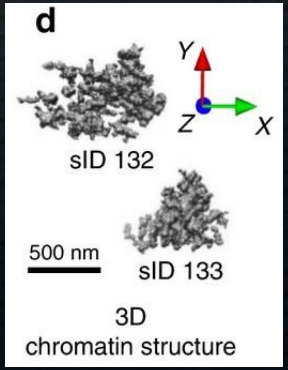
- Until now, we decided on the features to give to our algorithms: think tumour size, biopsy test scores, etc.
- With neural networks and images, the situation changes: we don't arduously describe what is in each image, but rather let the network learn to extract and combine features *so that* it can classify training examples correctly.

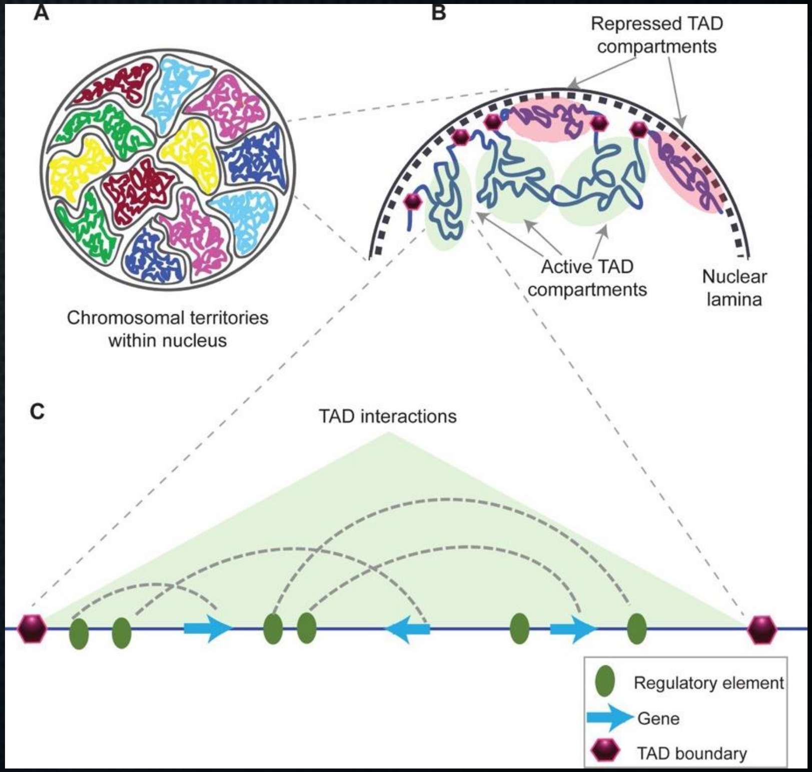


https://towardsdatascience.com/cnn-application-on-structured-data- automated-feature-extraction-8f2cd28d9a7e

Why neural networks?

- Caveat for biology: it can be quite difficult to translate biological problems into a framework fit for deep learning.
- Example: in images, nearby pixels probably hold similar information, i.e. are involved in the same thing. Due to (long-range) 3D-folding of DNA, linearly far DNA can be close together functionally. You need to encode your network or input to accomodate this!





Source: https://en.wikipedia.org/wiki/Topologically_associating_domain#/media/File:Structural_organization_of_chromatin.png

Source: Trzaskoma, P., Ruszczycki, B., Lee, B., Pels, K. K., Krawczyk, K., Bokota, G., ... & Wilczyński, G. M. (2020). Ultrastructural visualization of 3D chromatin folding using volume electron microscopy and DNA in situ hybridization. Nature communications, 11(1), 1-9.

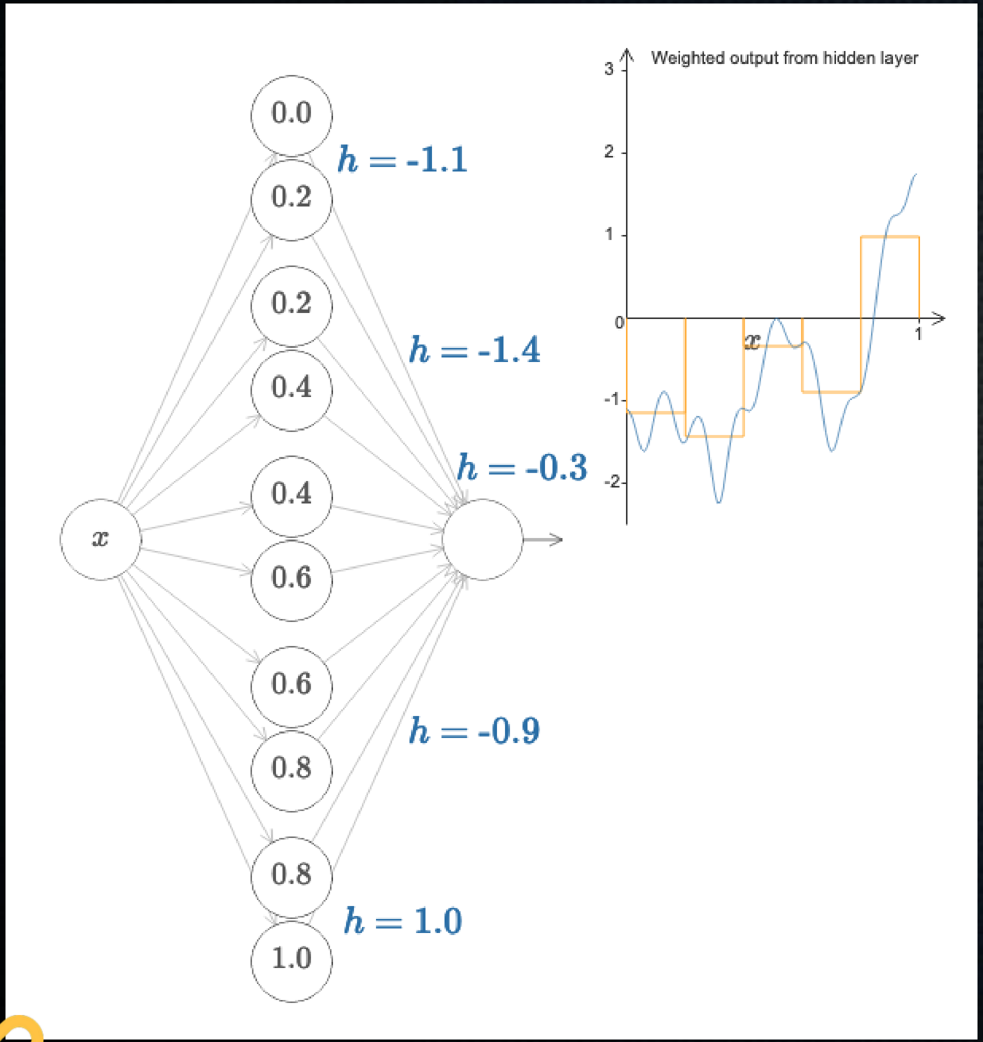
Why neural networks?

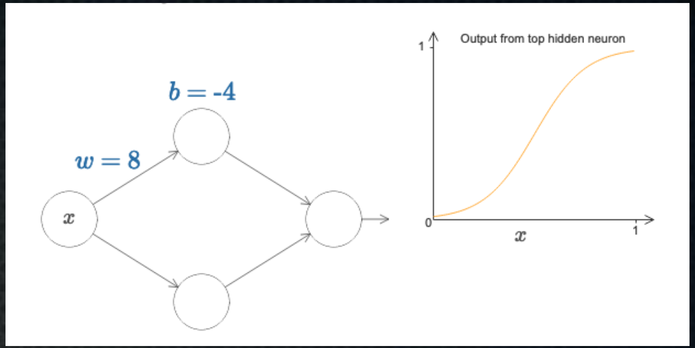
- The mythical property of *universal approximation*. This says that neural networks can approximate any function with arbitrary accuracy, even with only **1** hidden layer (given enough neurons in it).

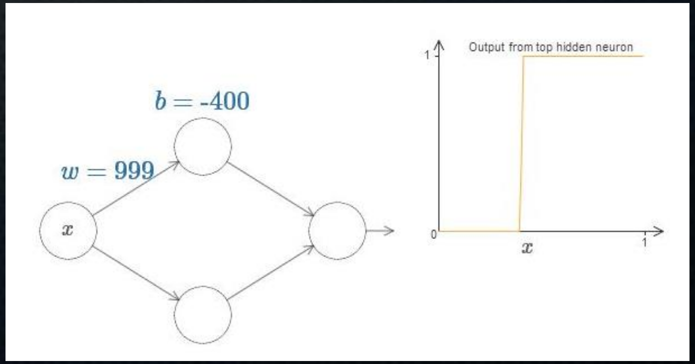
Why neural networks?

- The mythical property of *universal approximation*. This says that neural networks can approximate any function with arbitrary accuracy, even with only **1** hidden layer (given enough neurons in it).
- Of course, that doesn't necessarily mean we would have the data to *train* such a neural network efficiently. Just that it is provable that for *any continuous function* a neural network can exist that approximates it as well as you like.

Why neural networks?



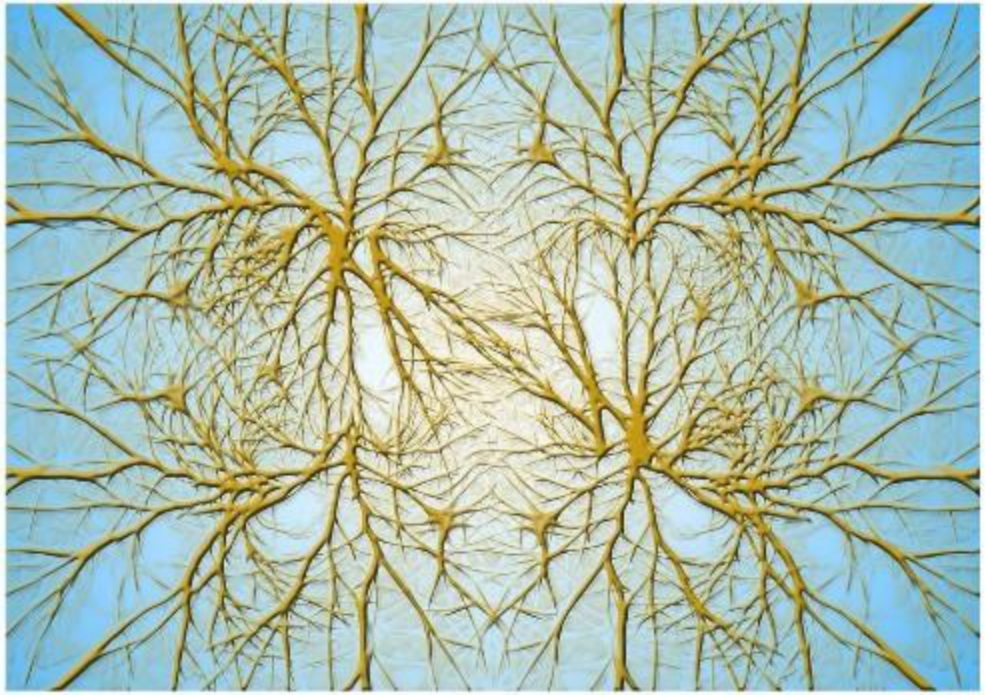




http://neuralnetworksanddeeplearning.com/chap4.html

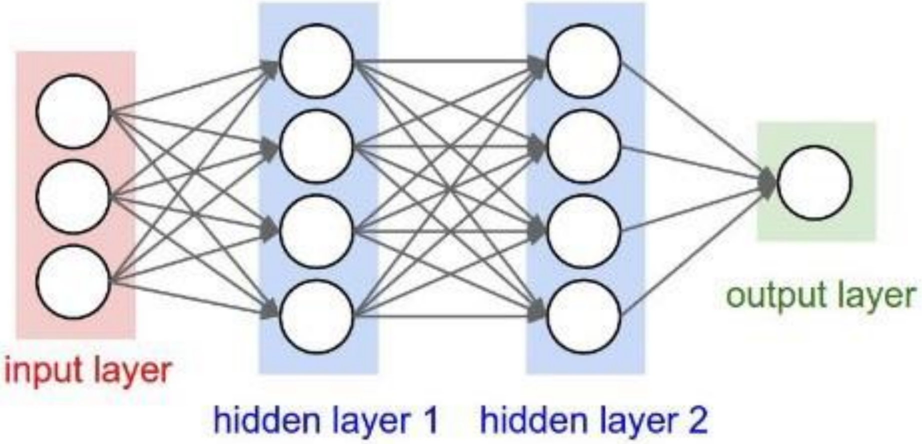
Like biology? **No**

Biological Neurons:
Complex connectivity patterns



This image is CC0 Public Domain

Neurons in a neural network:
Organized into regular layers for computational efficiency



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

Like biology? **No**

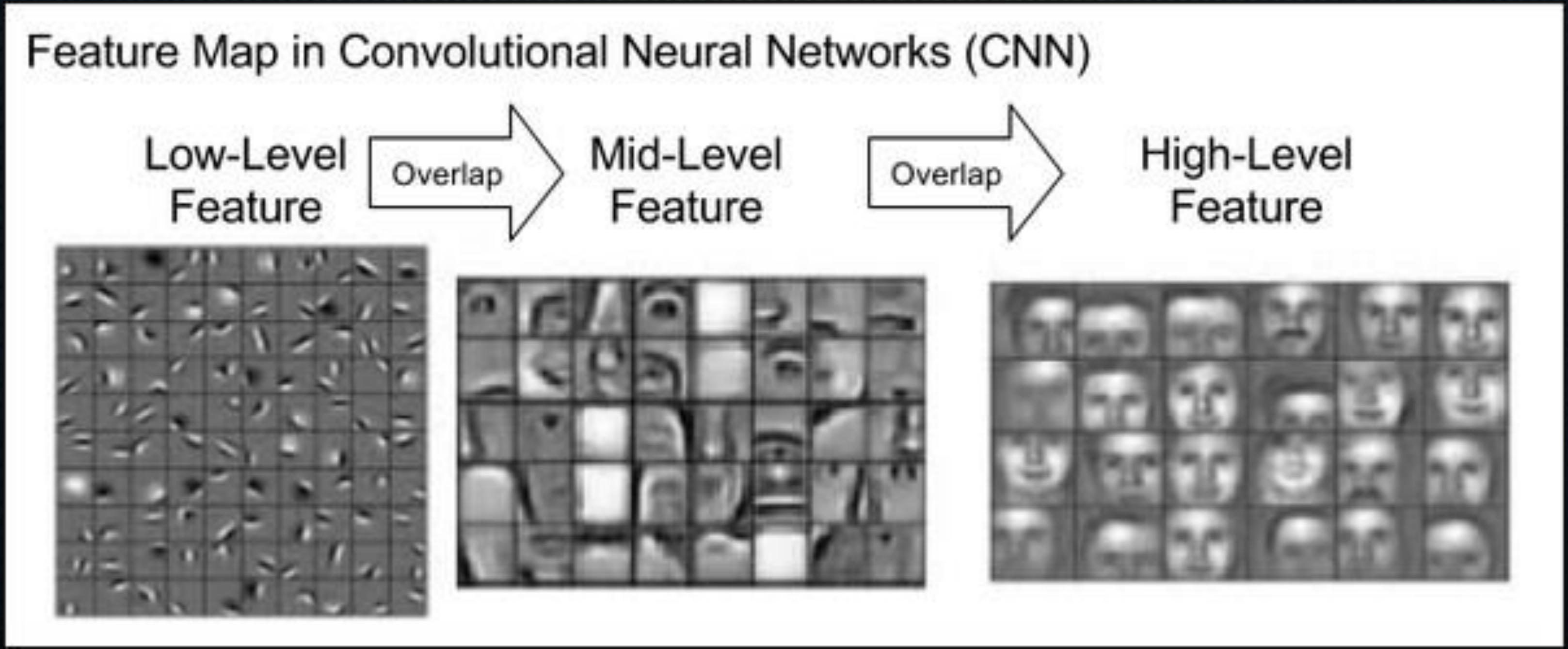
Biological Neurons:
- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system

- Human brains ~a cool 86 billion neurons
- A neuron can have 400.000 dendrites
- Real brains *vastly outclass* their computational analogues

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture04.pdf

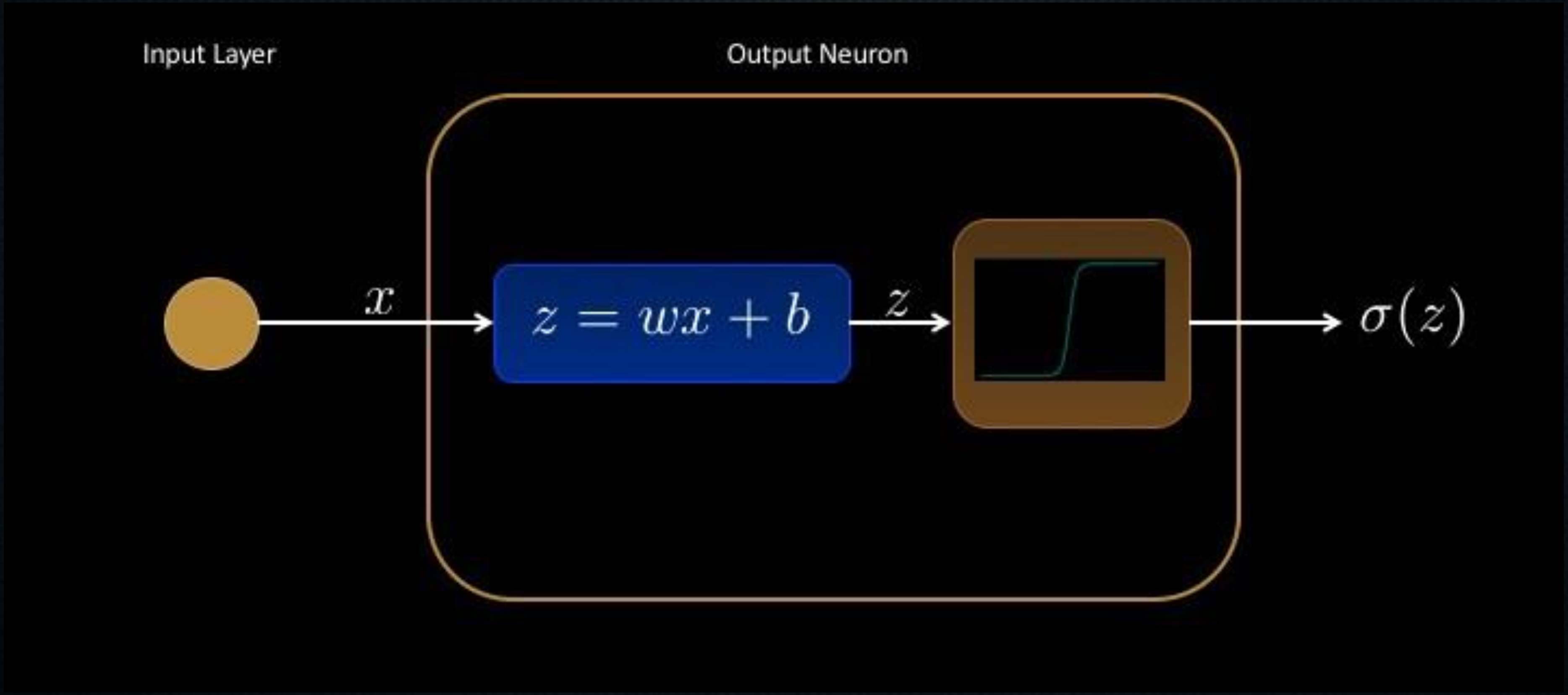
Like biology? **No**

- Still extremely useful
- Parts of how they learn *superficially resemble* how we learn



What do neural nets have to do with logistic regression?

- A single neuron is a logistic regressor!*

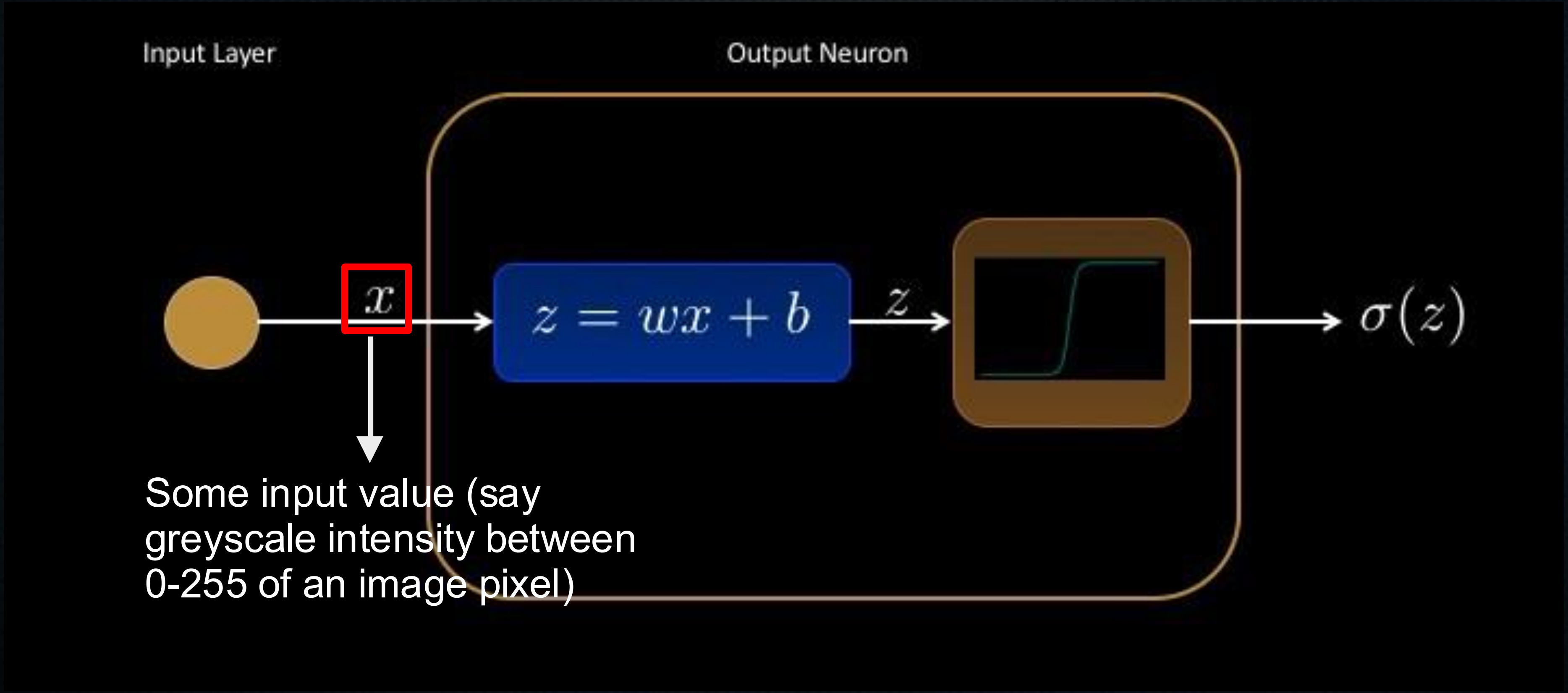


Source: https://thedatafrog.com/en/articles/logistic-regression/

*Given sigmoid activation function (which is not current standard practice anymore, but was at first)

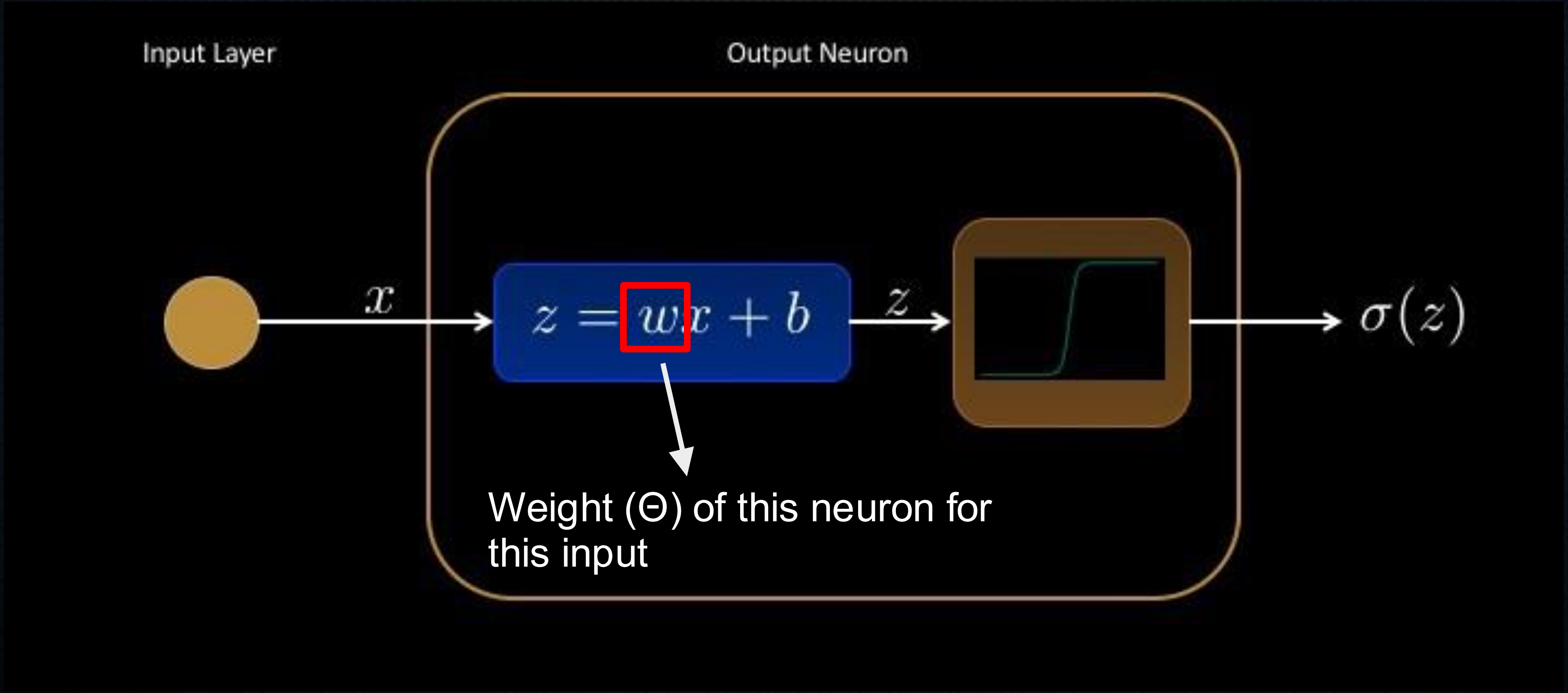
What do neural nets have to do with logistic regression?

- A single neuron is a logistic regressor!



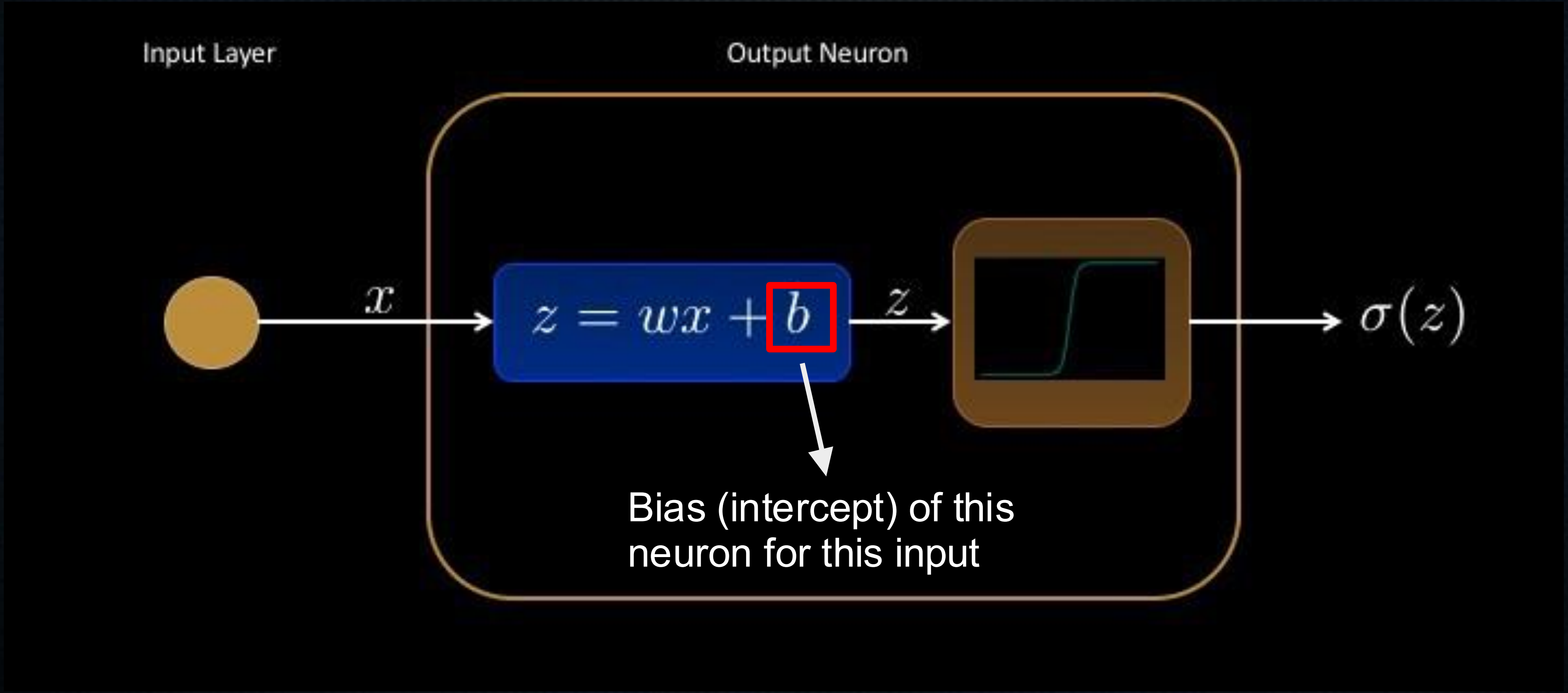
What do neural nets have to do with logistic regression?

- A single neuron is a logistic regressor!



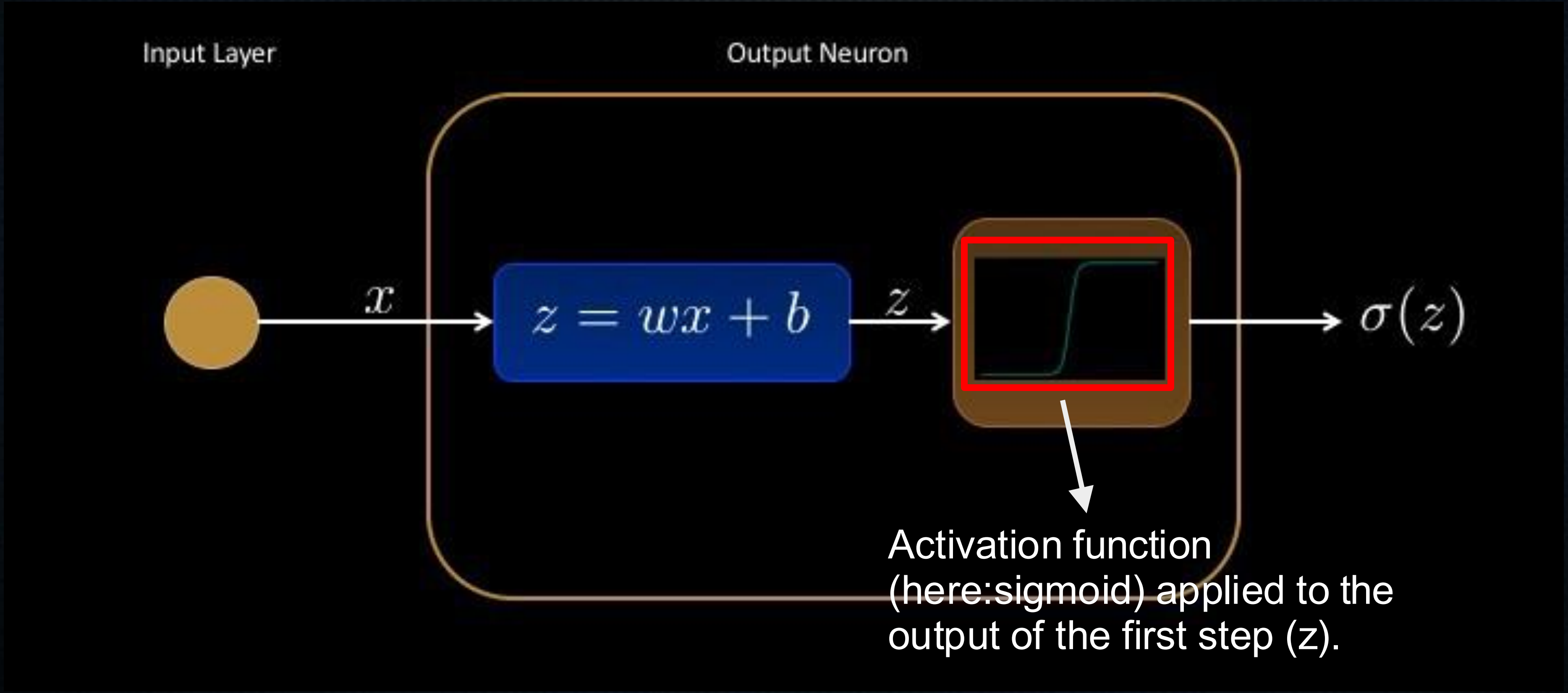
What do neural nets have to do with logistic regression?

- A single neuron is a logistic regressor!



What do neural nets have to do with logistic regression?

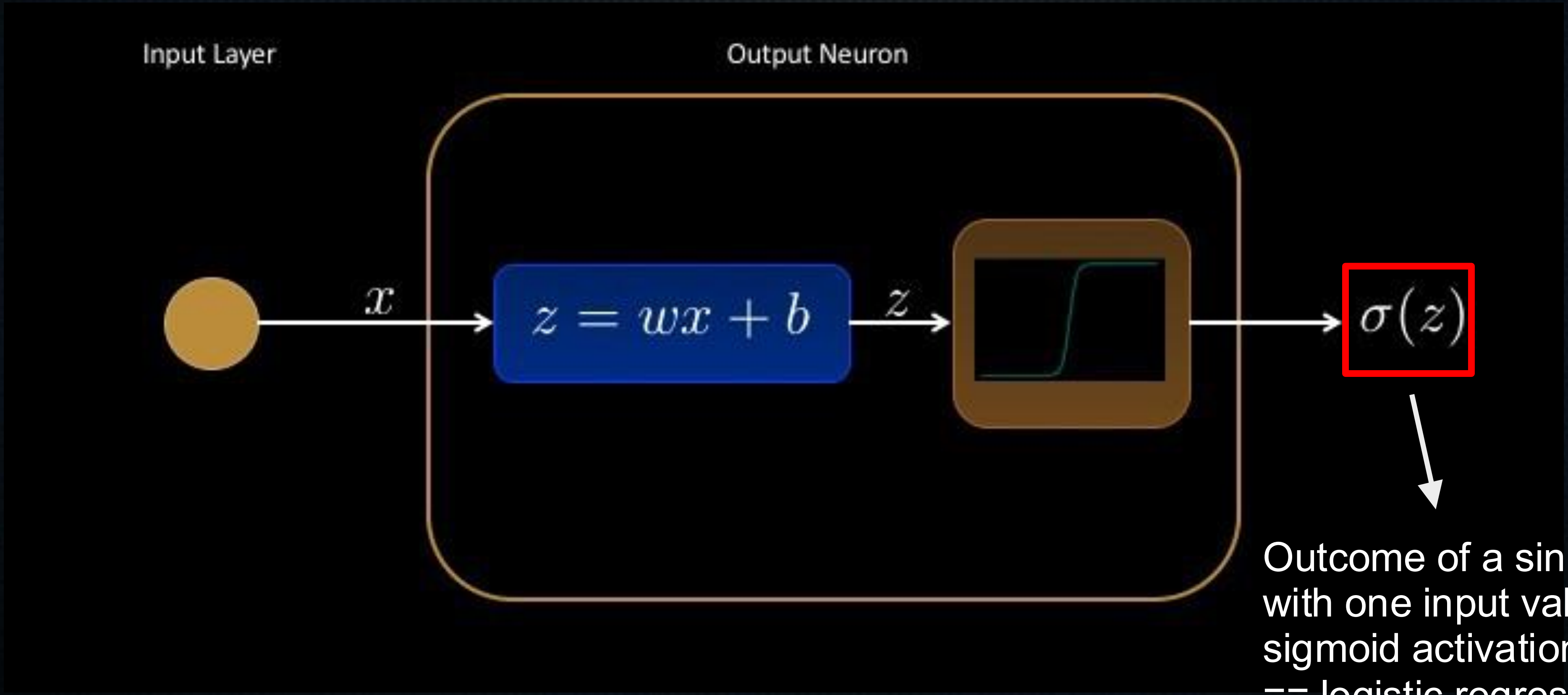
- A single neuron is a logistic regressor!



Activation function (here:sigmoid) applied to the output of the first step (z).

What do neural nets have to do with logistic regression?

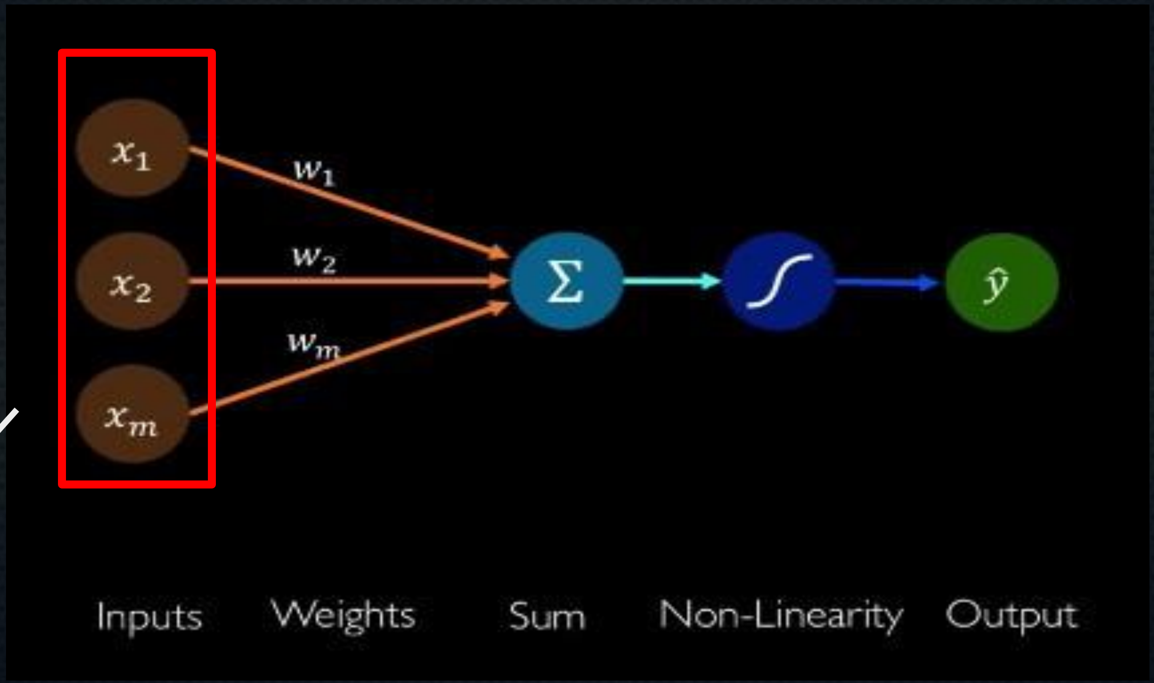
- A single neuron is a logistic regressor!



Outcome of a single neuron with one input value and sigmoid activation function == logistic regressor with 1 parameter (and bias)

What do neural nets have to do with logistic regression?

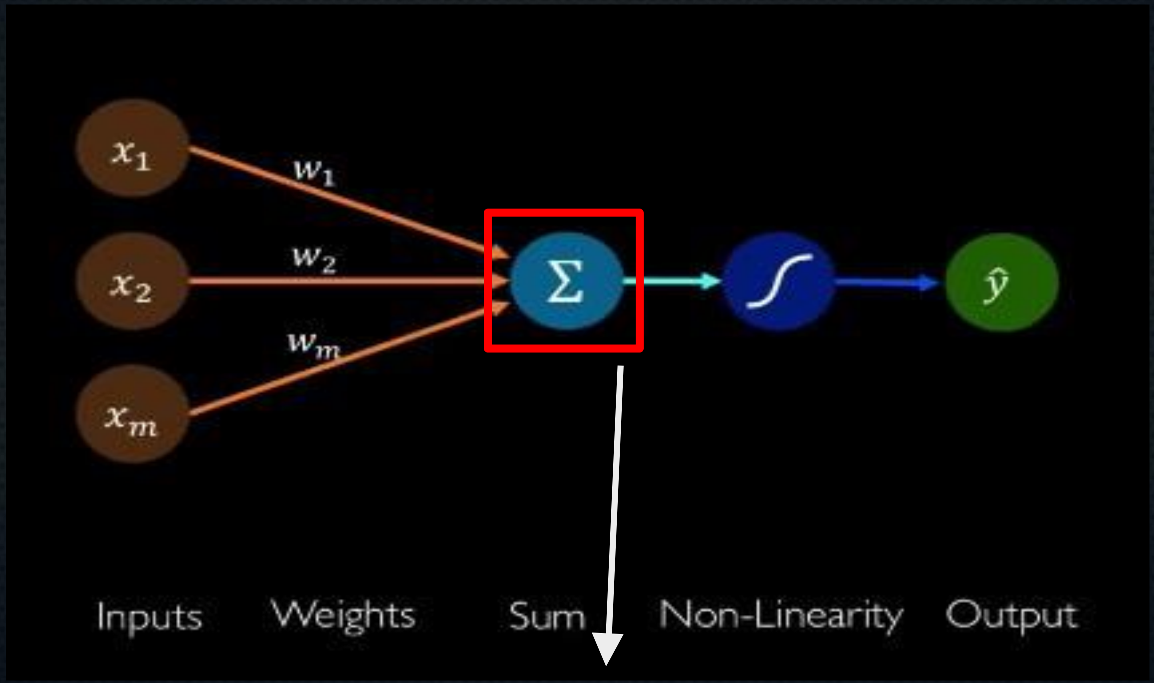
- A single neuron is a logistic regressor!



Multiple inputs: summation of those inputs' weights and the neuron's bias

What do neural nets have to do with logistic regression?

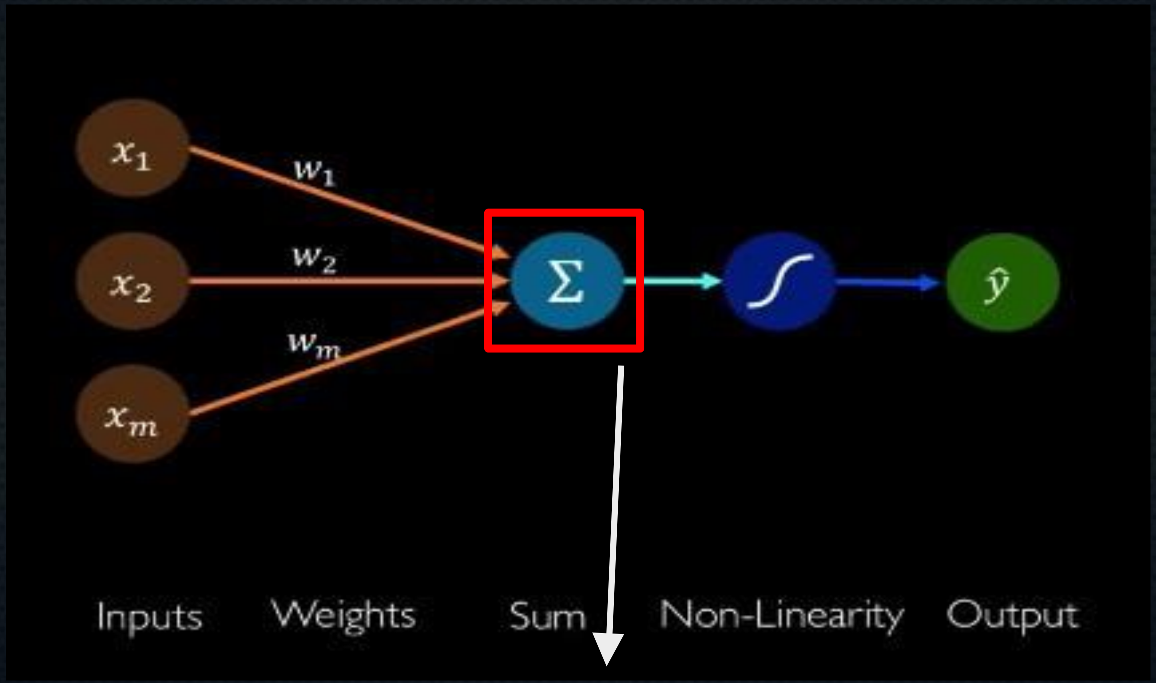
- A single neuron is a logistic regressor!



$$b + \sum_{i=1}^{m} w_i \cdot x_i$$

What do neural nets have to do with logistic regression?

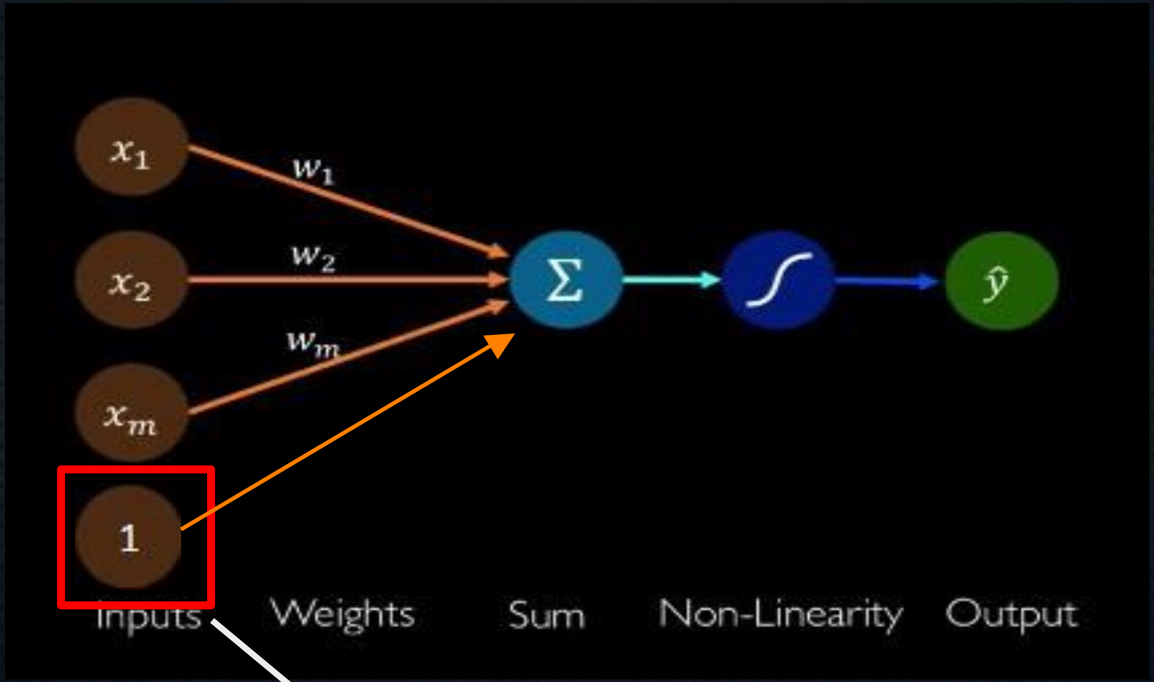
- A single neuron is a logistic regressor!



Or, in linear algebra notation: $\begin{bmatrix} 1 & x_1 & x_2 & x_m \end{bmatrix} \begin{bmatrix} b \\ w_1 \\ w_2 \\ w_m \end{bmatrix}$

What do neural nets have to do with logistic regression?

- A single neuron is a logistic regressor!

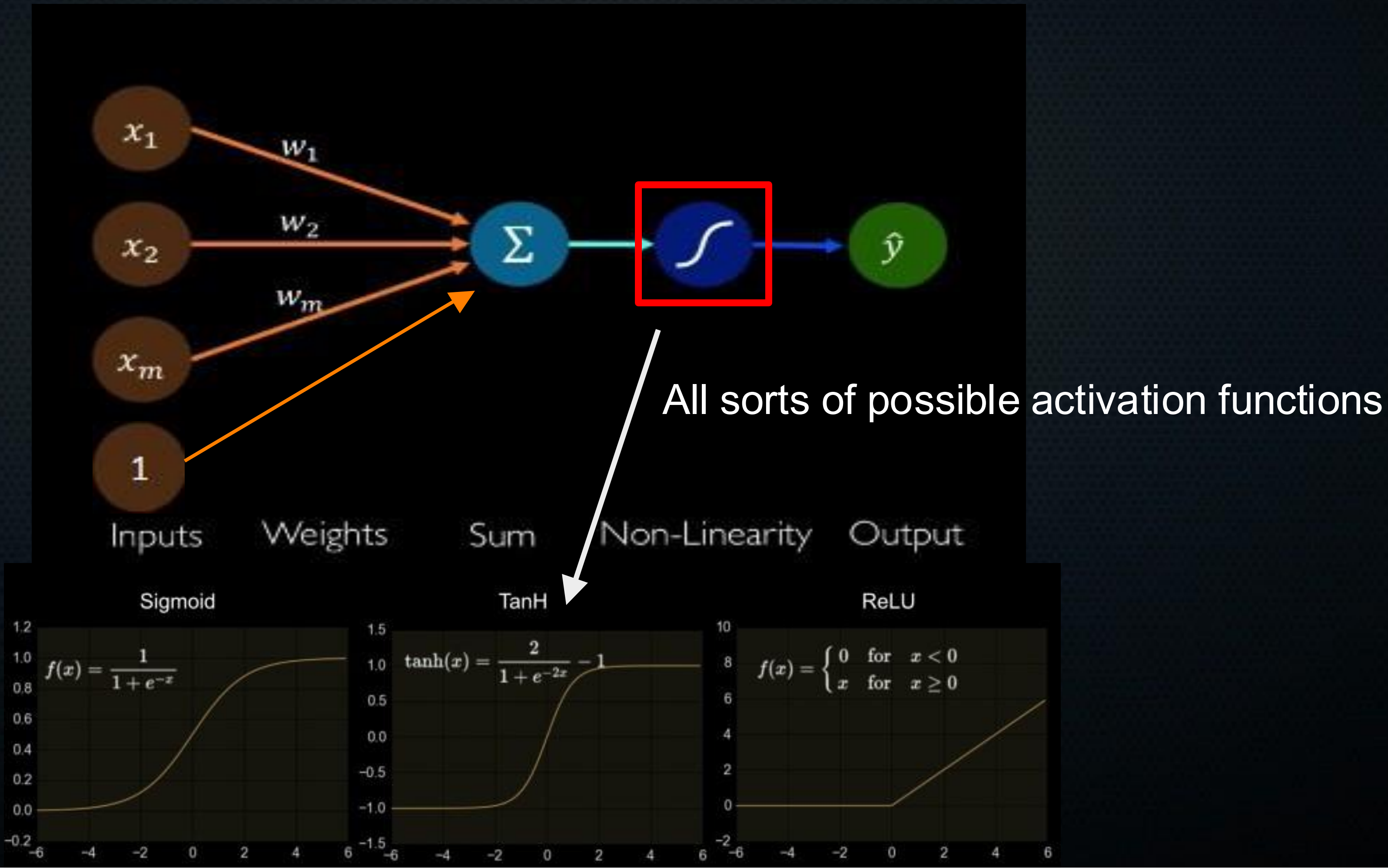


$$[\boxed{1} \quad x_1 \quad x_2 \quad x_m] \begin{bmatrix} b \\ w_1 \\ w_2 \\ w_m \end{bmatrix}$$

Bias unit (always 1, same for every neuron, most often not shown)

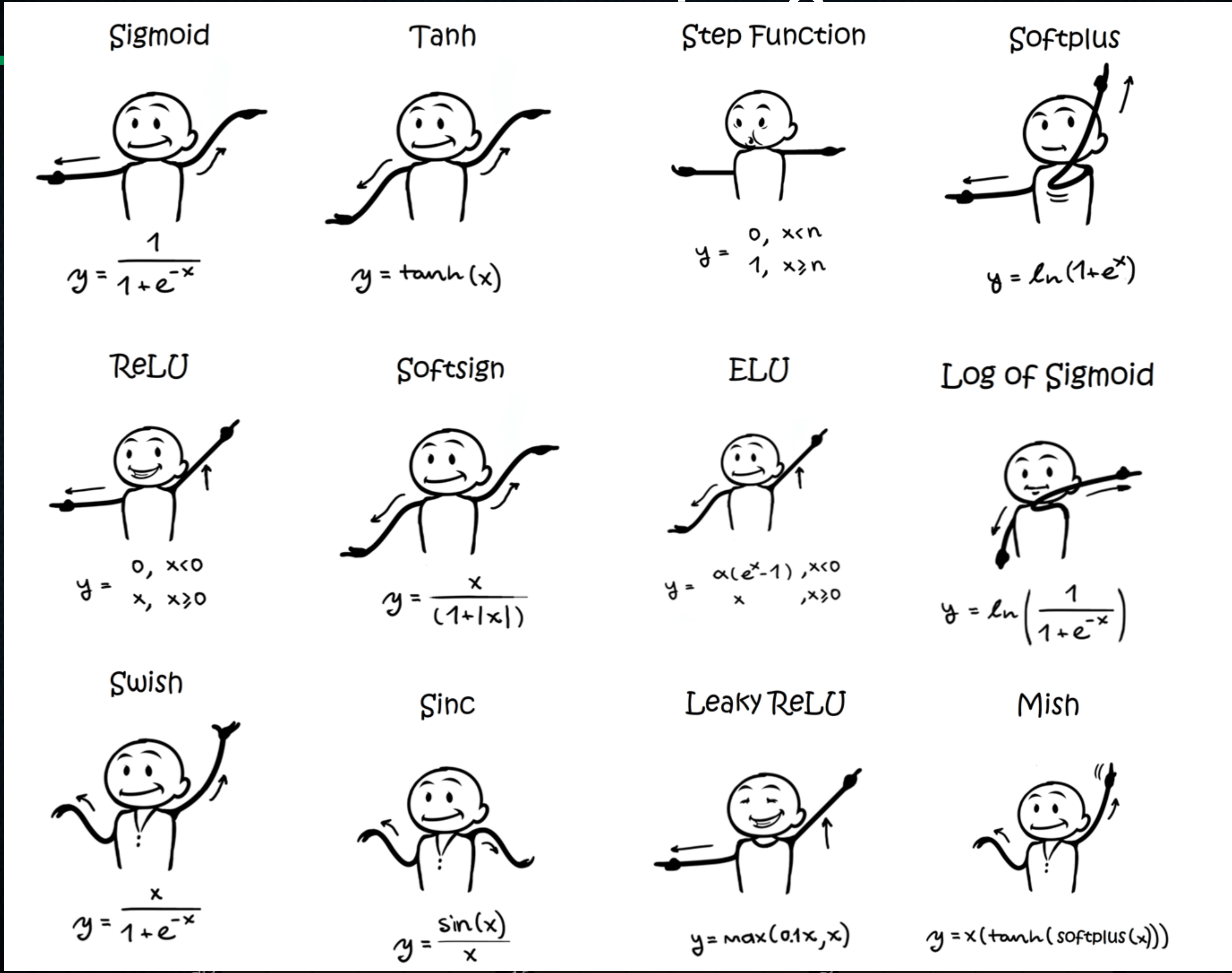
What do neural nets have to do with logistic regression?

- A single neuron is a logistic regressor!



What do neural nets have to do with logistic

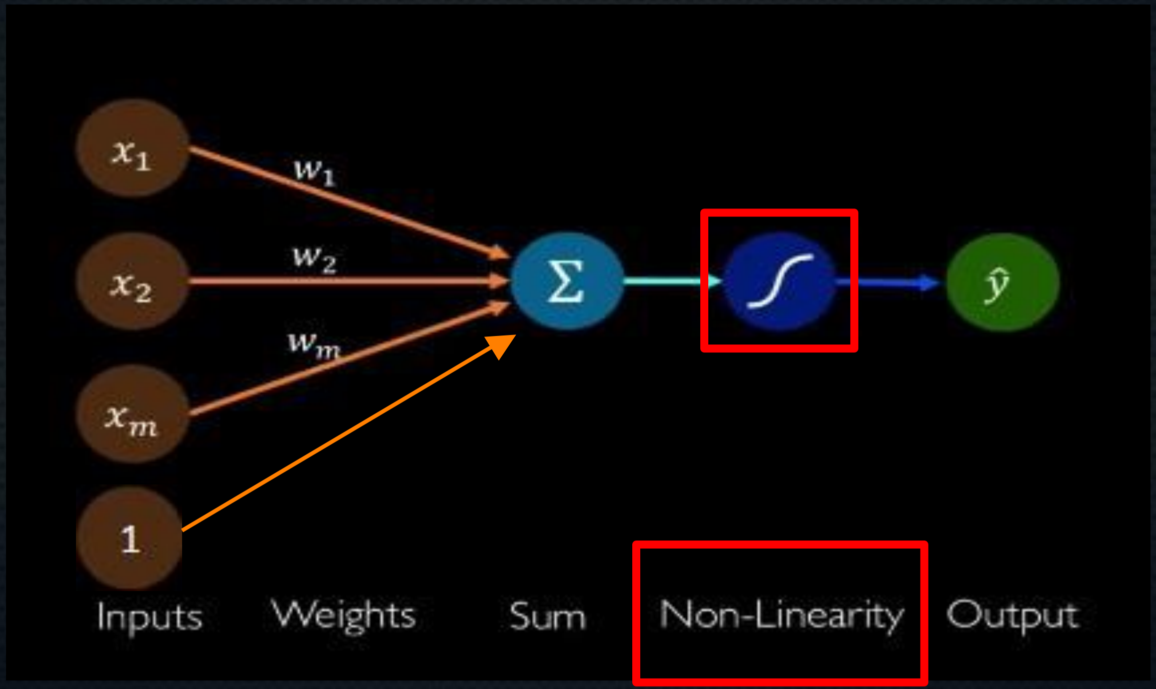
- A single



n functions

What do neural nets have to do with logistic regression?

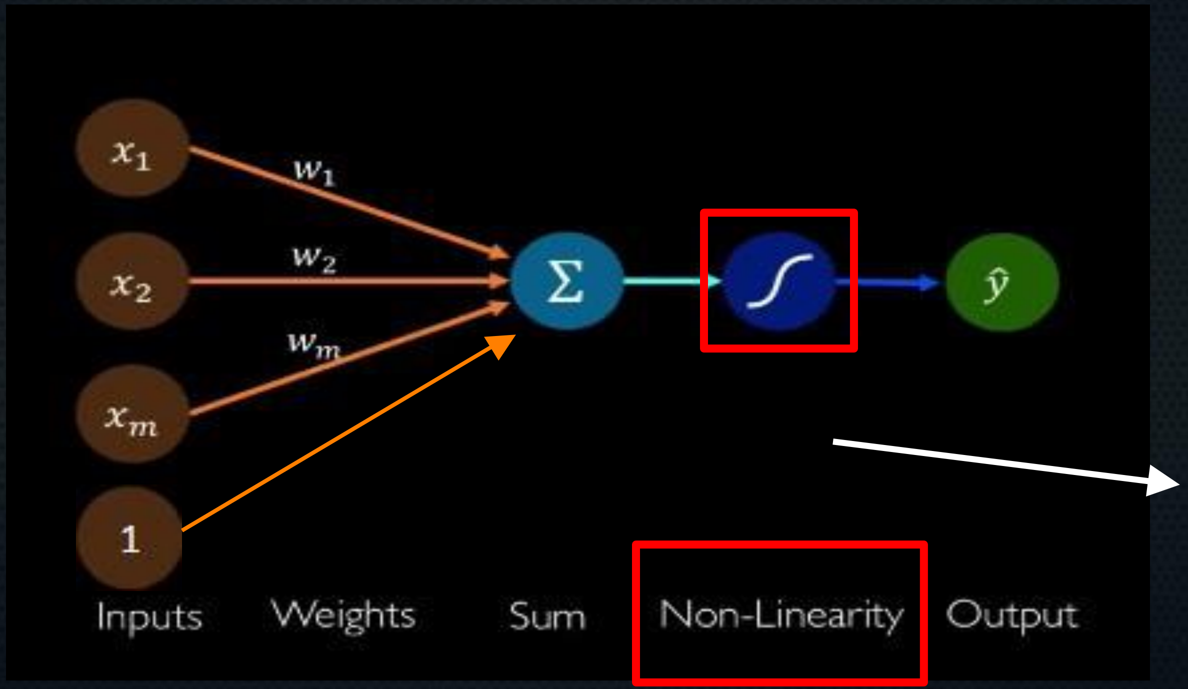
- A single neuron is a logistic regressor!

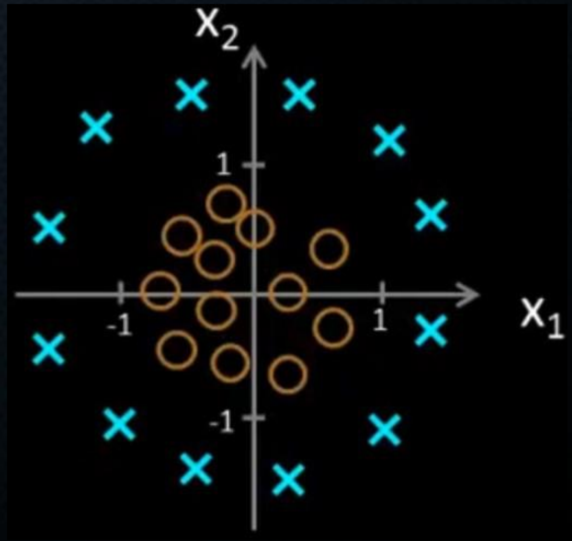


- Why non-linearity?

What do neural nets have to do with logistic regression?

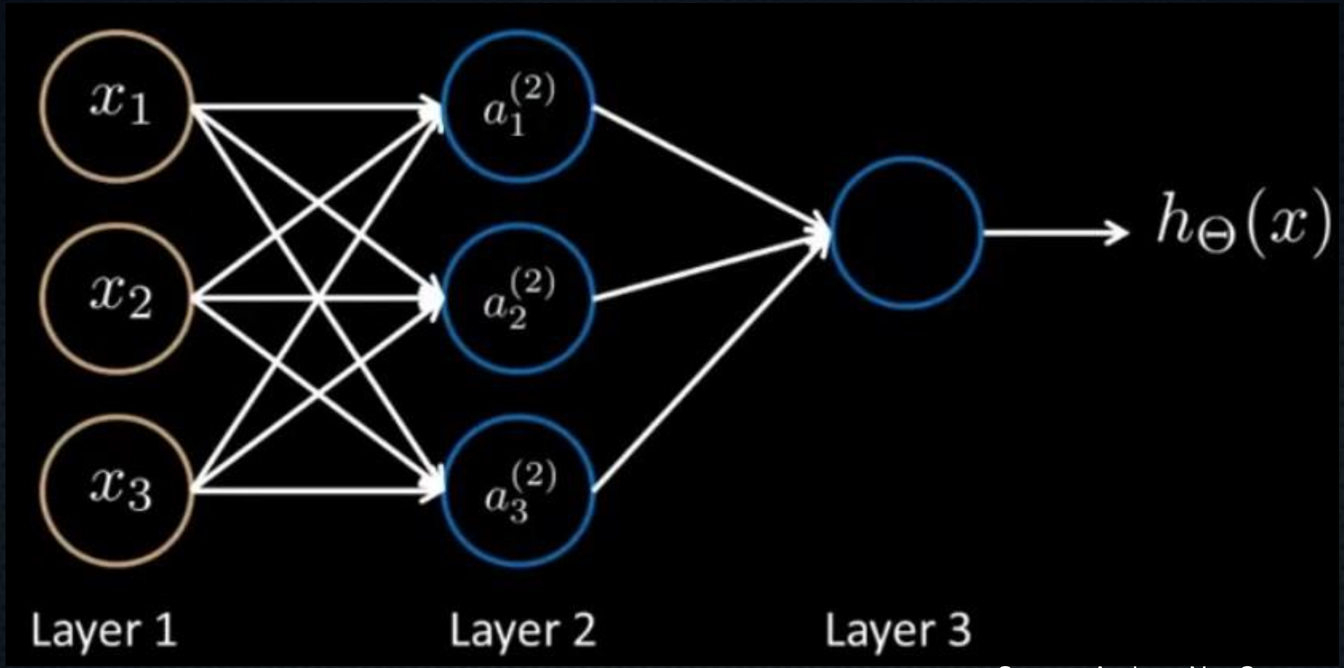
- A single neuron is a logistic regressor!





- Why non-linearity? → without them, a NN (no matter how deep) could only approximate linear functions

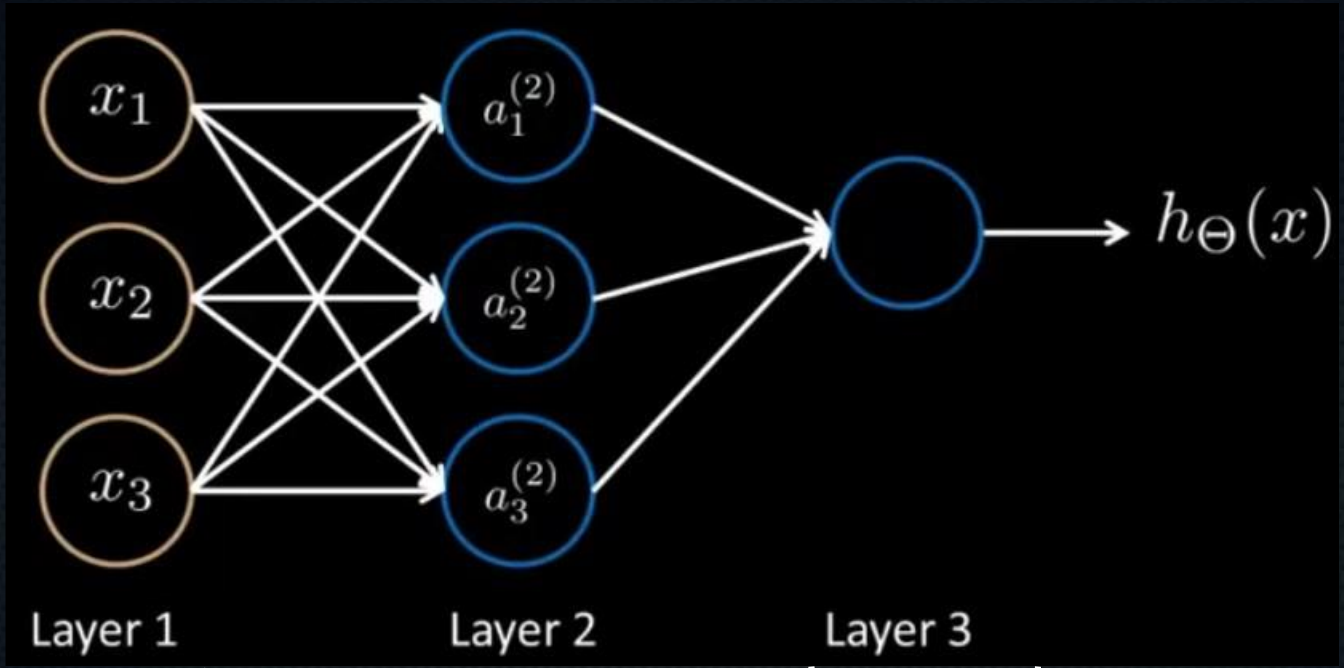
Workings of a simple neural network



Source: Andrew Ng, Coursera

Input layer: shown as neurons, are just features

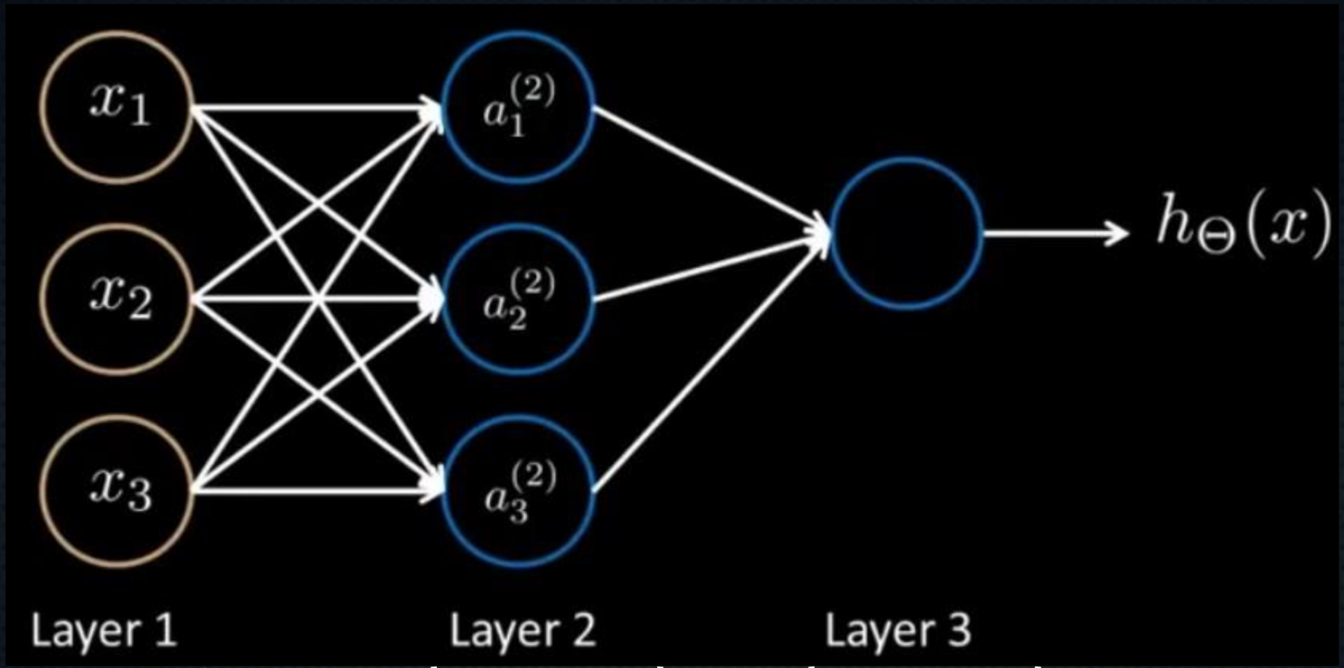
Workings of a simple neural network



Input layer: shown as neurons, are just features

Output layer: final result of the neural net

Workings of a simple neural network

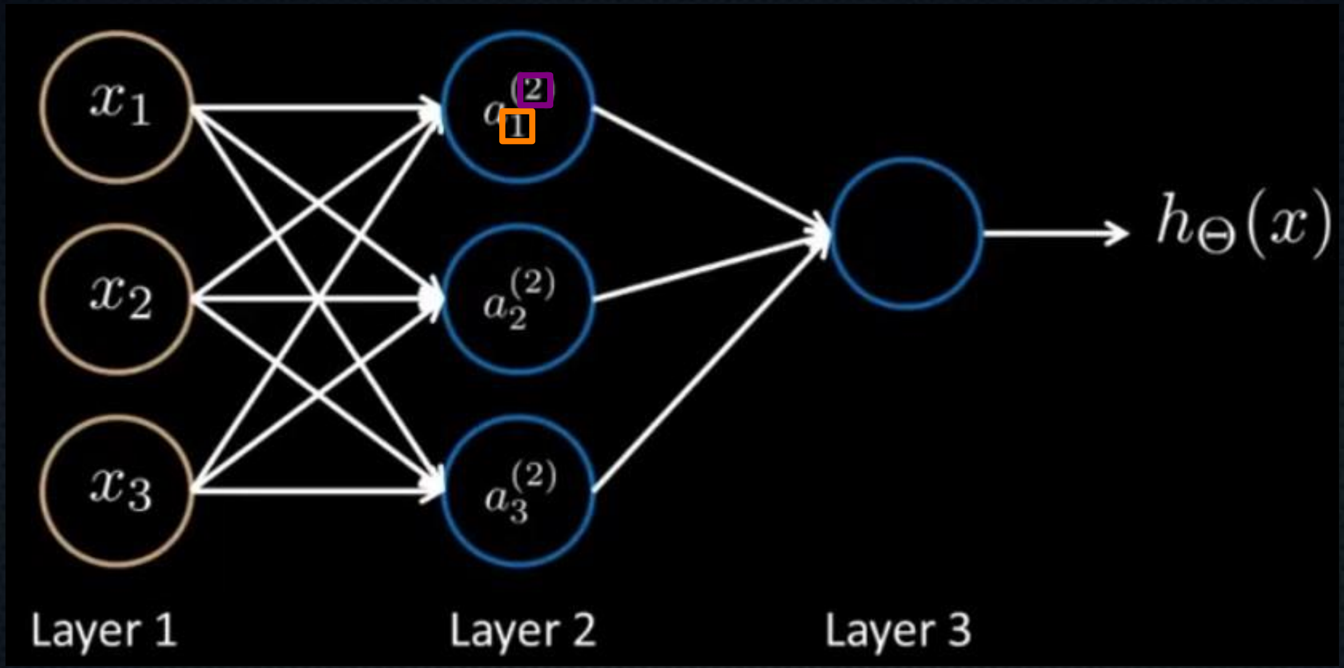


Input layer: shown as neurons, are just features

Output layer: final result of the neural net

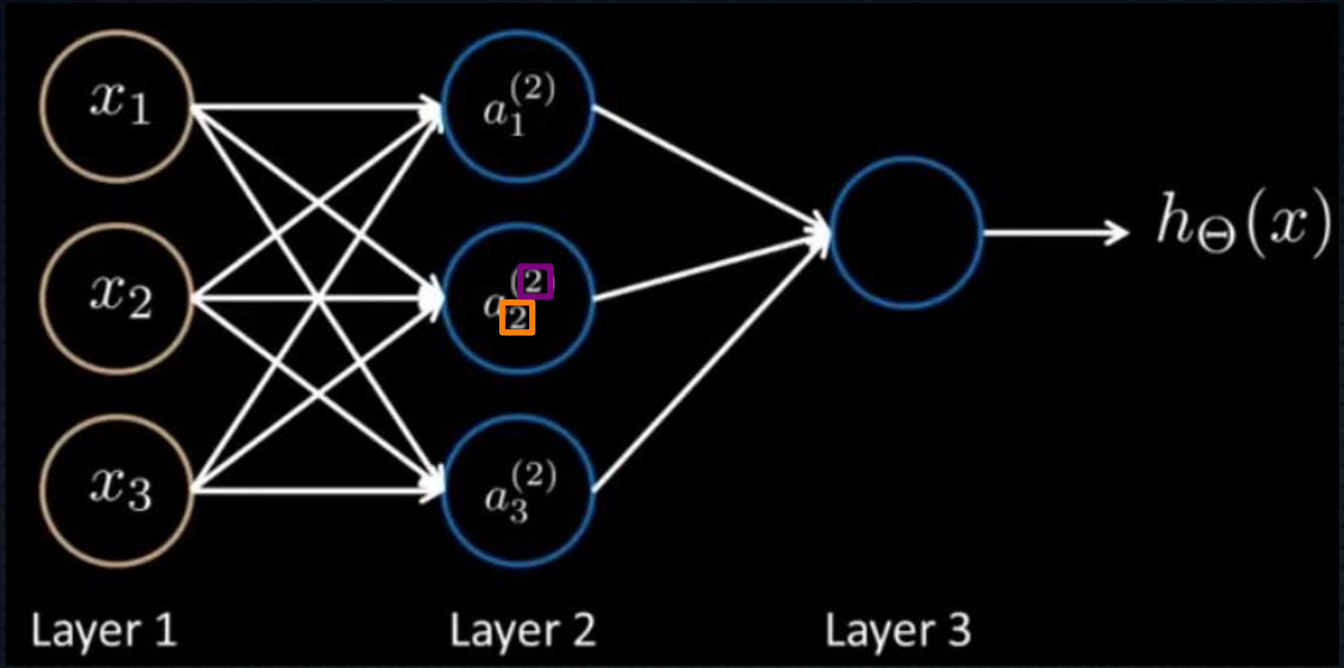
Hidden layer(s): intermediate layers whose outputs are not directly observed (hence hidden). Here: 1 HL. Facebook's DenseNet family of NNs had 121-264 HLs in 2016 (0.8-15.3 million parameters).

Workings of a simple neural network



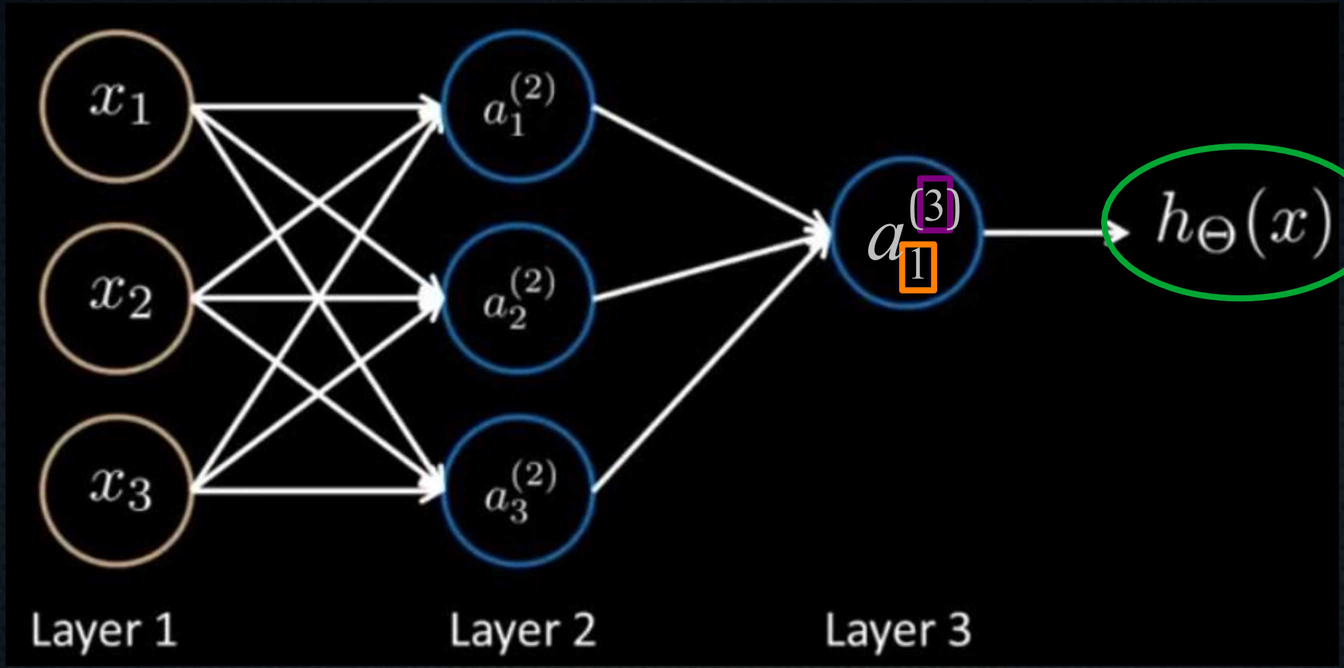
Activation of neuron **1** in the **2**nd layer of the network.

Workings of a simple neural network



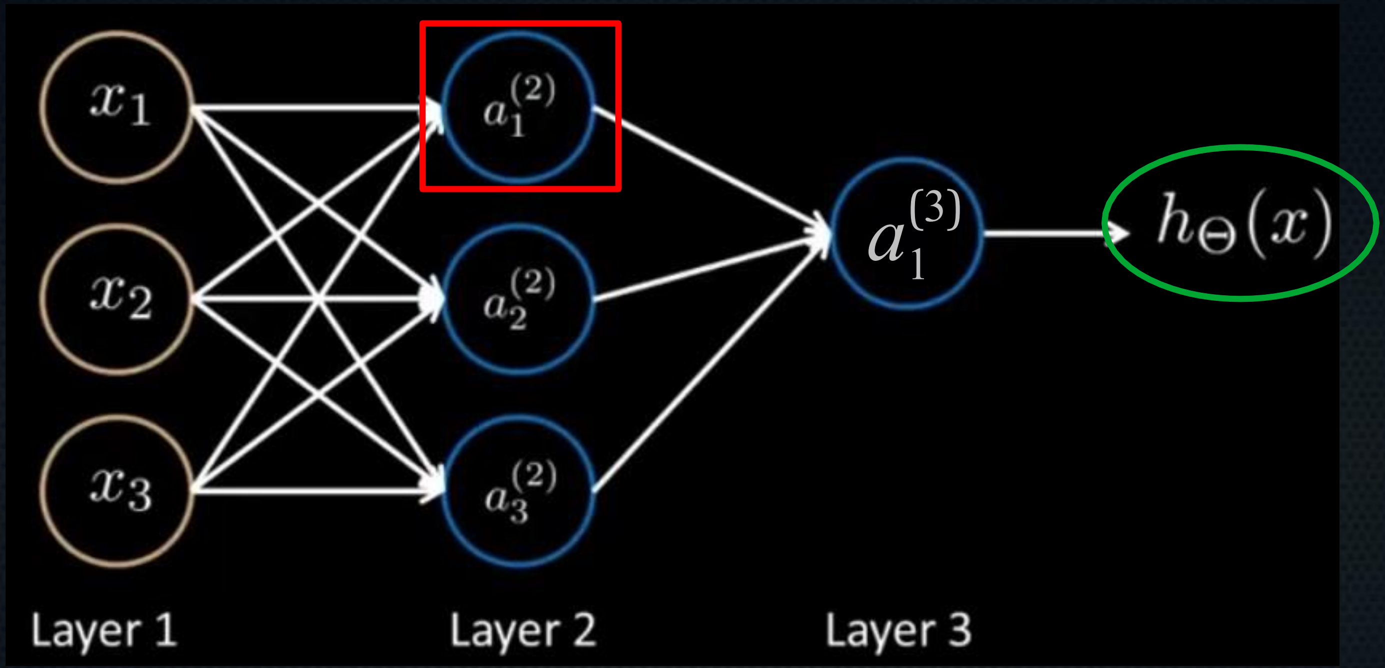
Activation of neuron **2** in the **2**nd layer of the network.

Workings of a simple neural network



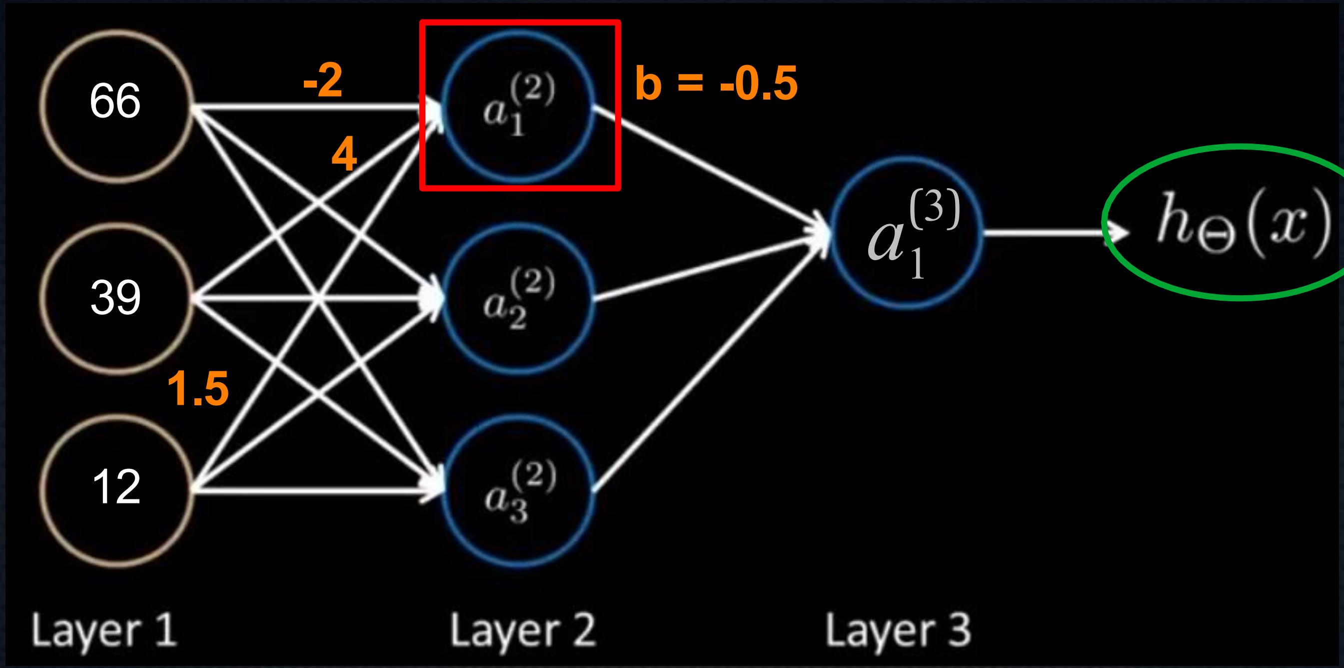
Activation of neuron 1 in the 3rd layer of the network.

What is this network calculating?



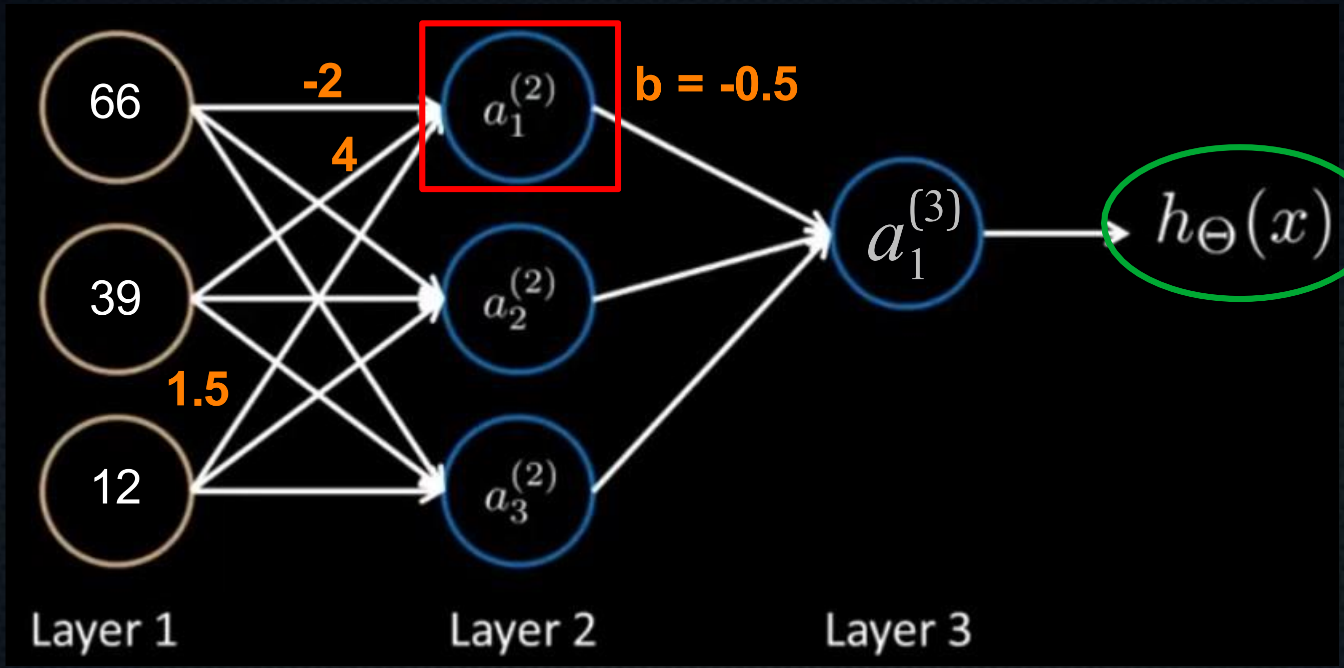
$$\sigma\left(\begin{bmatrix}1 & x_1 & x_2 & x_3\end{bmatrix}\begin{bmatrix}b \\ w_1 \\ w_2 \\ w_3\end{bmatrix}\right)$$

What is this network calculating?



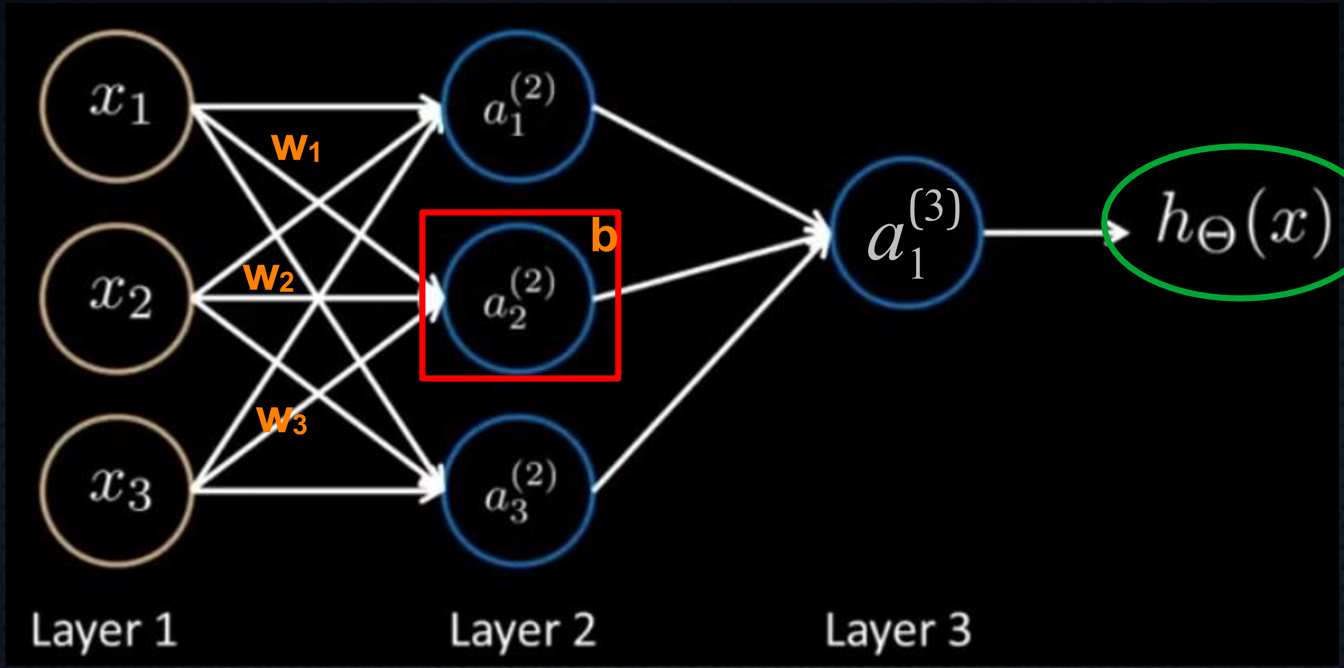
$$\sigma\left(\begin{bmatrix}1 & x_1 & x_2 & x_3\end{bmatrix}\begin{bmatrix}b\\ w_1\\ w_2\\ w_3\end{bmatrix}\right) \rightarrow \sigma\left(\begin{bmatrix}1 & 66 & 39 & 12\end{bmatrix}\begin{bmatrix}-0.5\\ -2\\ 4\\ 1.5\end{bmatrix}\right)$$

What is this network calculating?



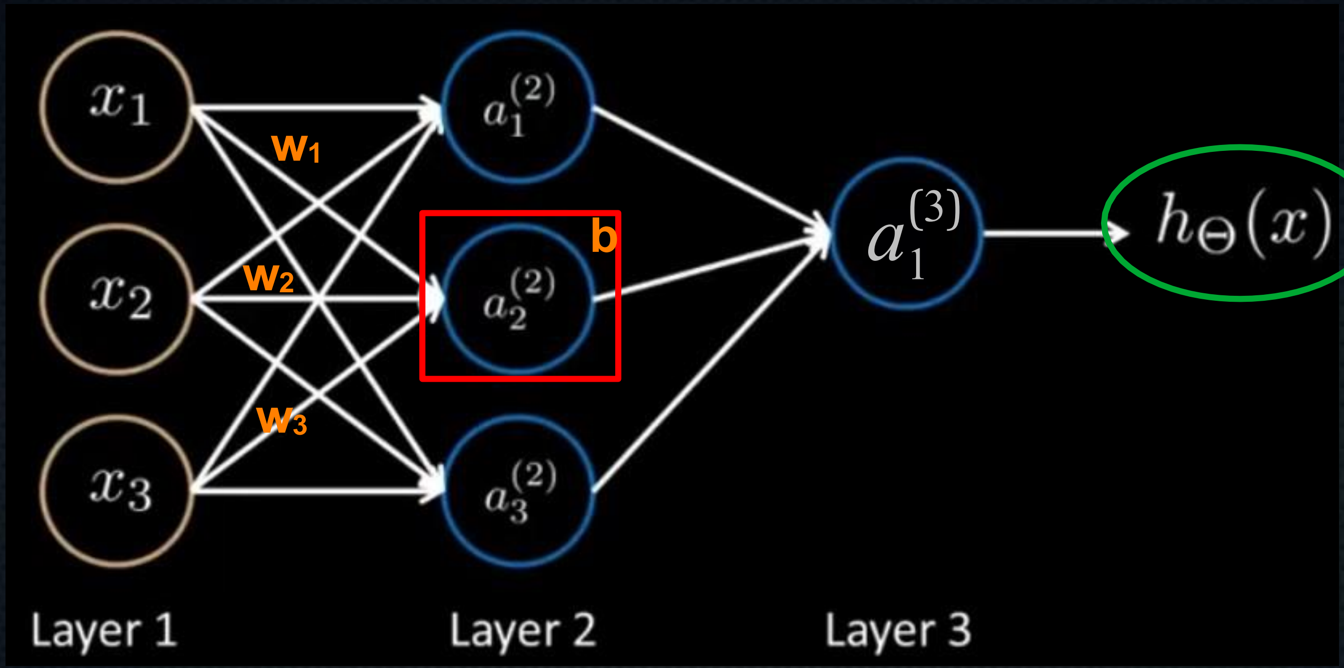
$$\sigma\left(\begin{bmatrix}1 & 66 & 39 & 12\end{bmatrix}\begin{bmatrix}-0.5\\-2\\4\\1.5\end{bmatrix}\right) \rightarrow \sigma(41.5) \rightarrow 0.999\ldots$$

What is this network calculating?



$$\sigma\left(\begin{bmatrix}1 & x_1 & x_2 & x_3\end{bmatrix}\begin{bmatrix}b\\ w_1\\ w_2\\ w_3\end{bmatrix}\right)$$

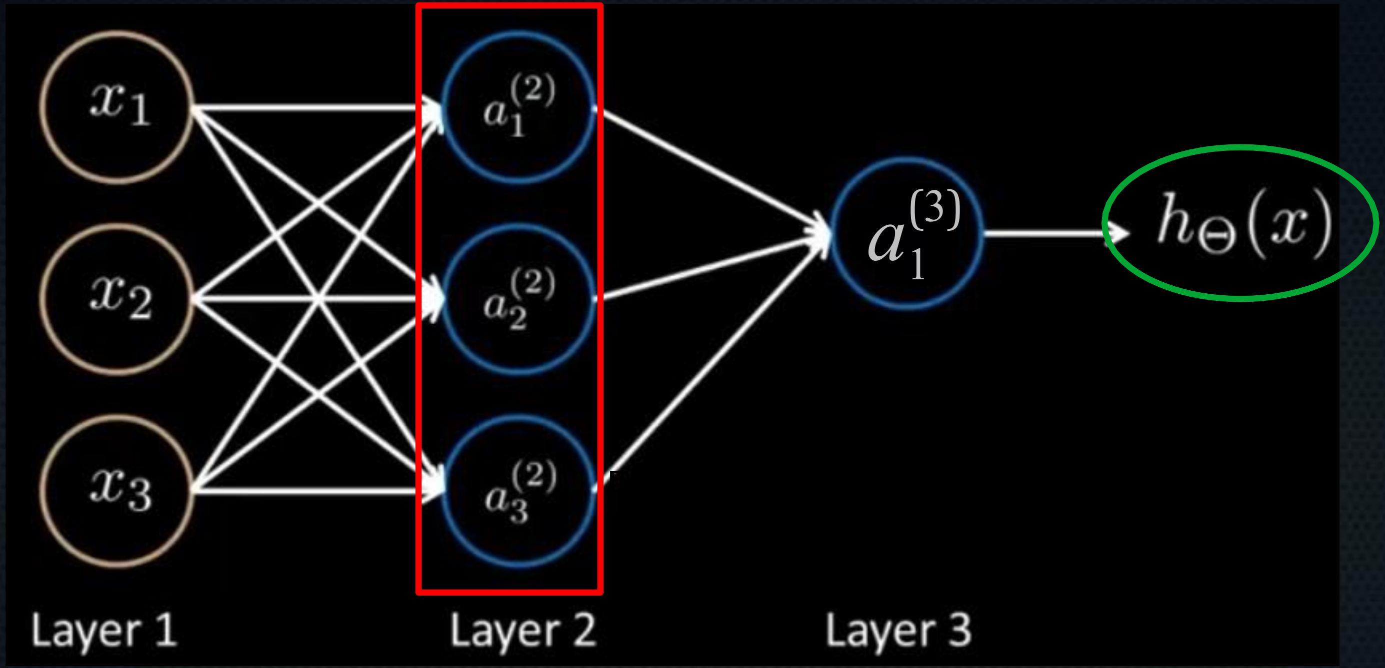
What is this network calculating?



$$\sigma\left(\begin{bmatrix}1 & x_1 & x_2 & x_3\end{bmatrix}\begin{bmatrix}b \\ w_1 \\ w_2 \\ w_3\end{bmatrix}\right) \longrightarrow$$

Calculate for all units at the same time with a theta matrix $\Theta^{(j)}$

What is this network calculating?

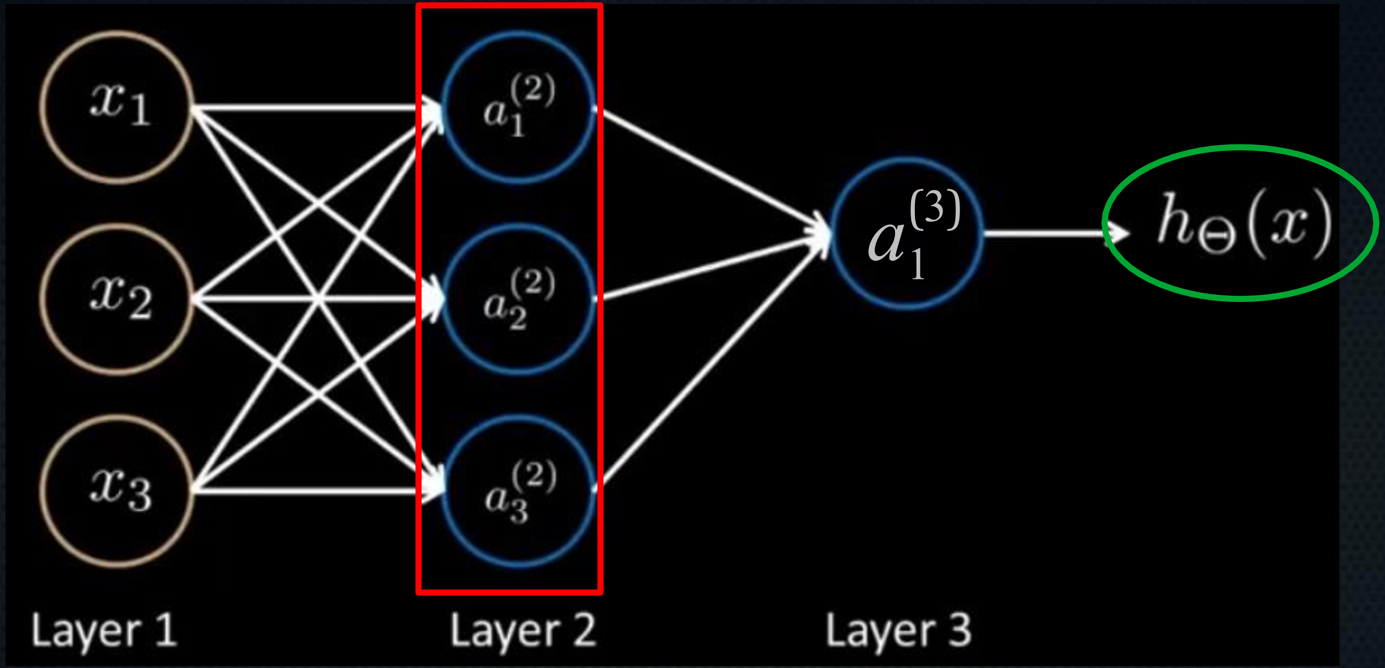


$$\sigma\left([1 \quad x_1 \quad x_2 \quad x_3]\begin{bmatrix} b_1 & b_2 & b_3 \\ w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \\ w_{13} & w_{23} & w_{33} \end{bmatrix}\right) \longrightarrow \sigma([1 \quad 2 \quad 3])$$

$$\downarrow$$

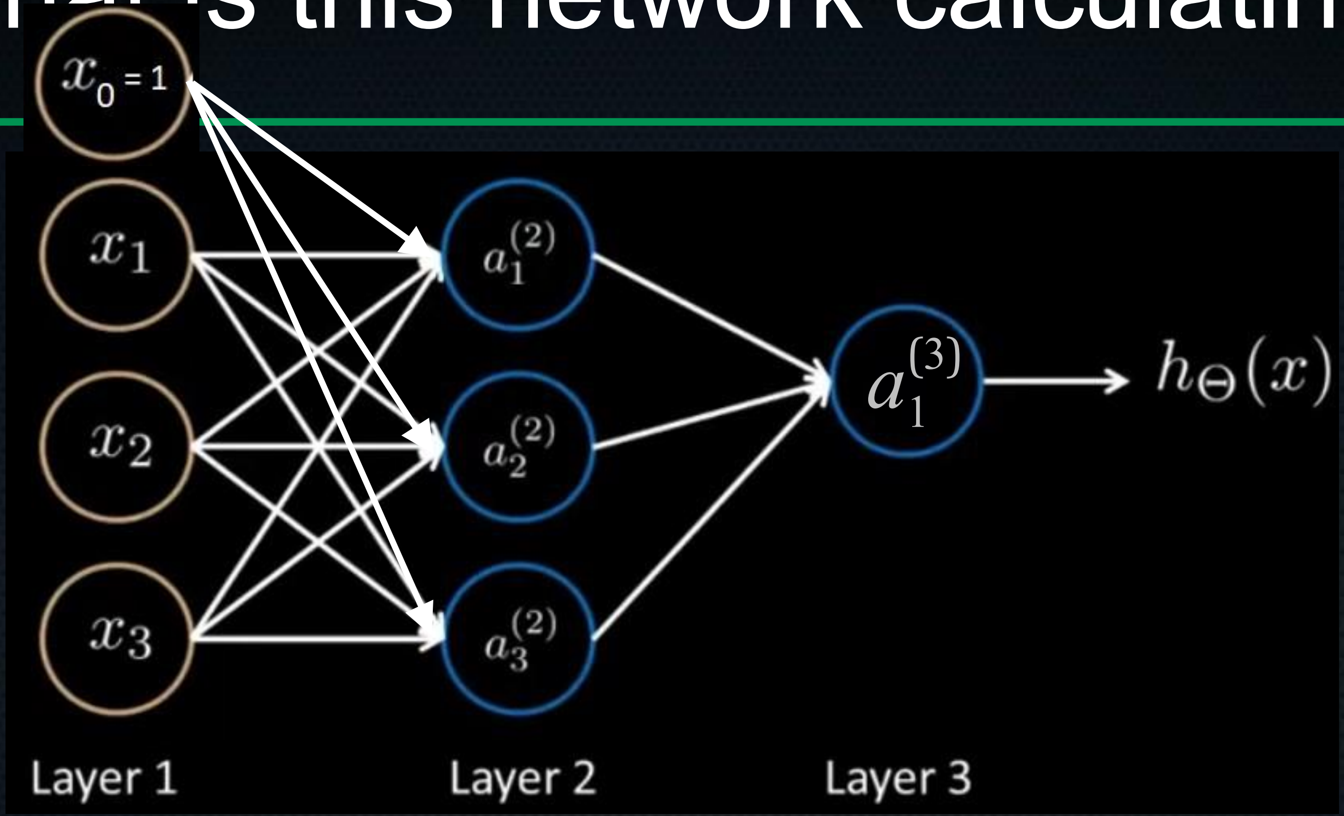
$$\boxed{[\sigma(1) \quad \sigma(2) \quad \sigma(3)]}$$

What is this network calculating?



$$\sigma\left(\begin{bmatrix} b_1 & w_{11} & w_{12} & w_{31} \\ b_2 & w_{21} & w_{22} & w_{23} \\ b_3 & w_{31} & w_{32} & w_{33} \end{bmatrix}\begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) \longrightarrow \sigma\left(\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}\right) \longrightarrow \boxed{\begin{bmatrix} \sigma(1) \\ \sigma(2) \\ \sigma(3) \end{bmatrix}}$$

What is this network calculating?

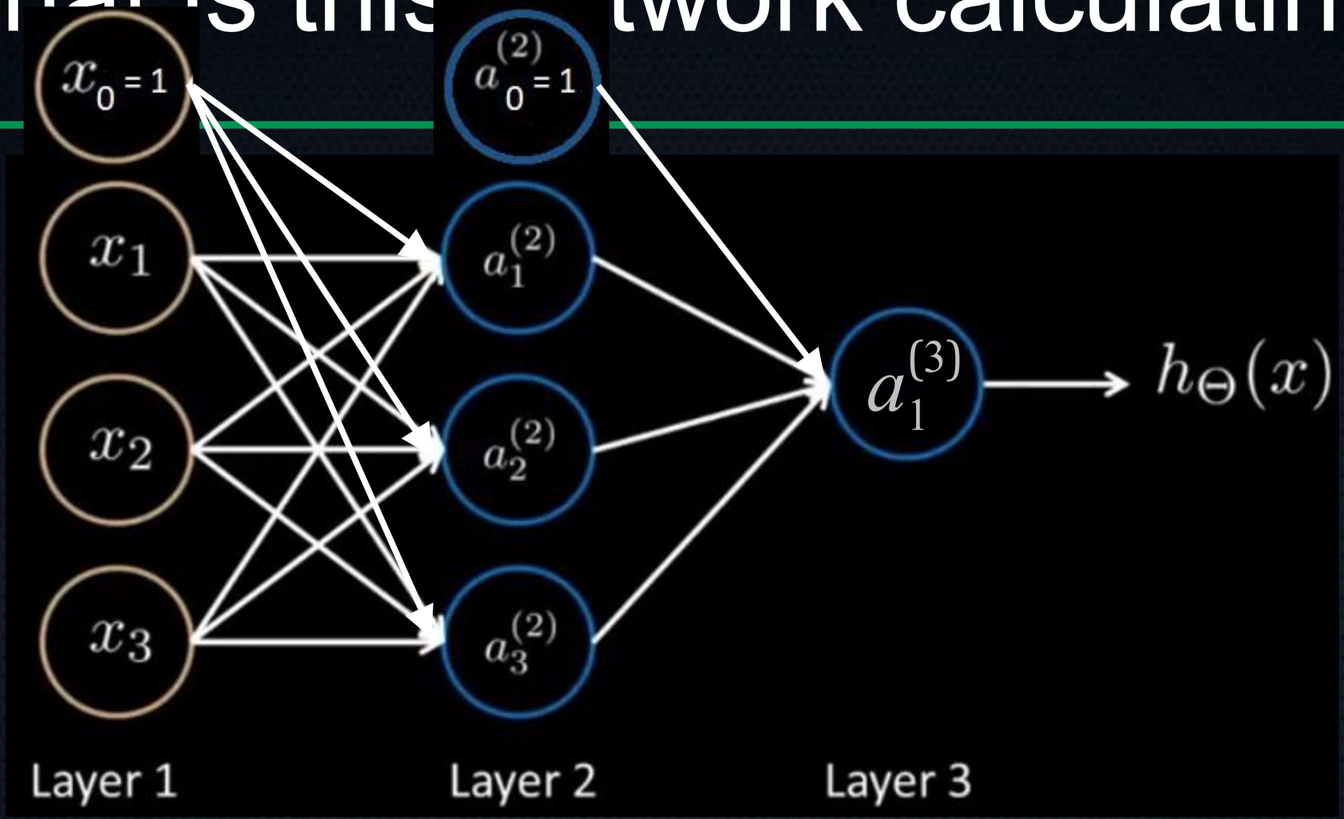


$$
\begin{aligned}
a_1^{(2)} &= g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)\\
a_2^{(2)} &= g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)\\
a_3^{(2)} &= g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)
\end{aligned}
$$

$\Theta^{(1)}$ (layer 1 to layer 2)

$$
\begin{bmatrix}
b_1 & w_{11} & w_{12} & w_{13}\\
b_2 & w_{21} & w_{22} & w_{23}\\
b_3 & w_{31} & w_{32} & w_{33}
\end{bmatrix}
$$

What is this network calculating?

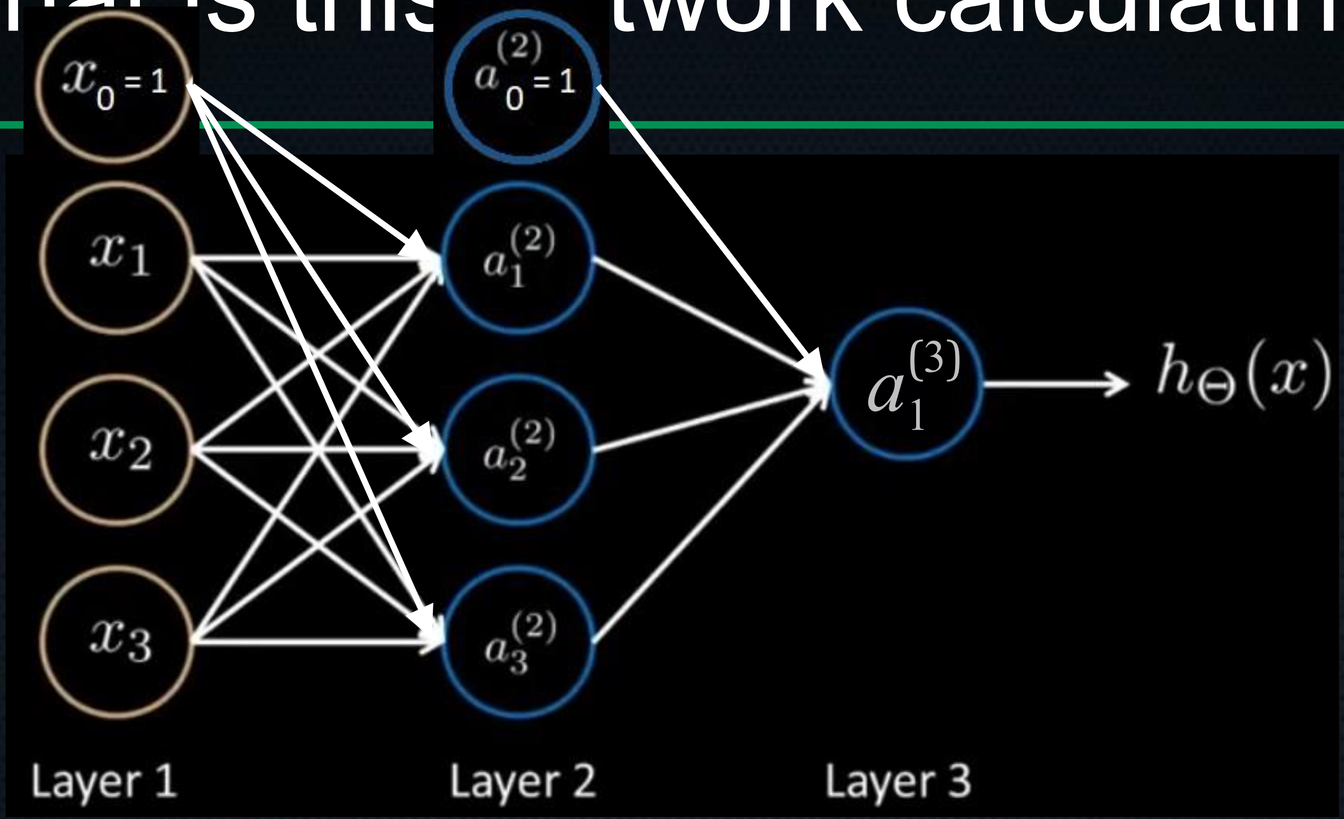


$\Theta^{(2)}$ (layer 2 to layer 3)

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$$\begin{bmatrix} b_1 & w_{11} & w_{12} & w_{13} \end{bmatrix}$$

What is this network calculating?



- This calculation of the output of the network is called forward propagation

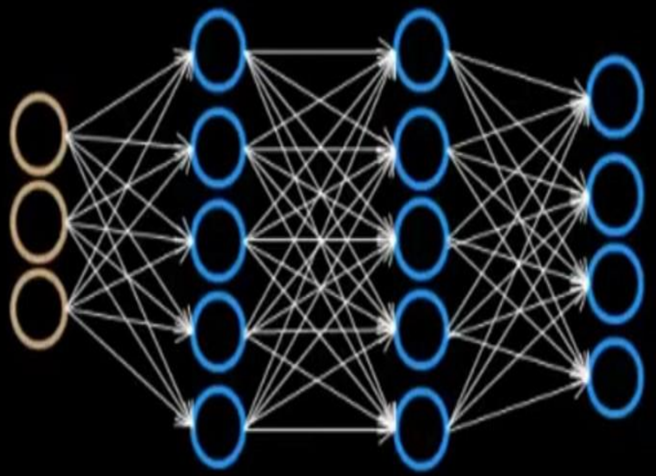
How do we perform?

- Just like before, there is a cost function.
- But we will talk about that and its implementation tomorrow!

How do we get parameters?

- Just like before, there is a cost function and a way to minimise this. But it’s a bit more involved.
- To get parameters, we will use the principle of backpropagation. We’ll get to that tomorrow.

Multiclass classification in neural nets

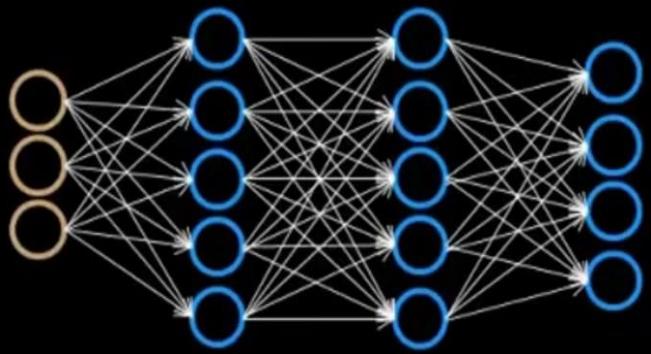


$$h_\Theta(x) \in \mathbb{R}^4$$

Want $h_\Theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Multiclass classification in neural nets



$h_\Theta(x) \in \mathbb{R}^4$

Want $h_\Theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, etc.

when pedestrian   when car   when motorcycle

Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

$y^{(i)}$ one of $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

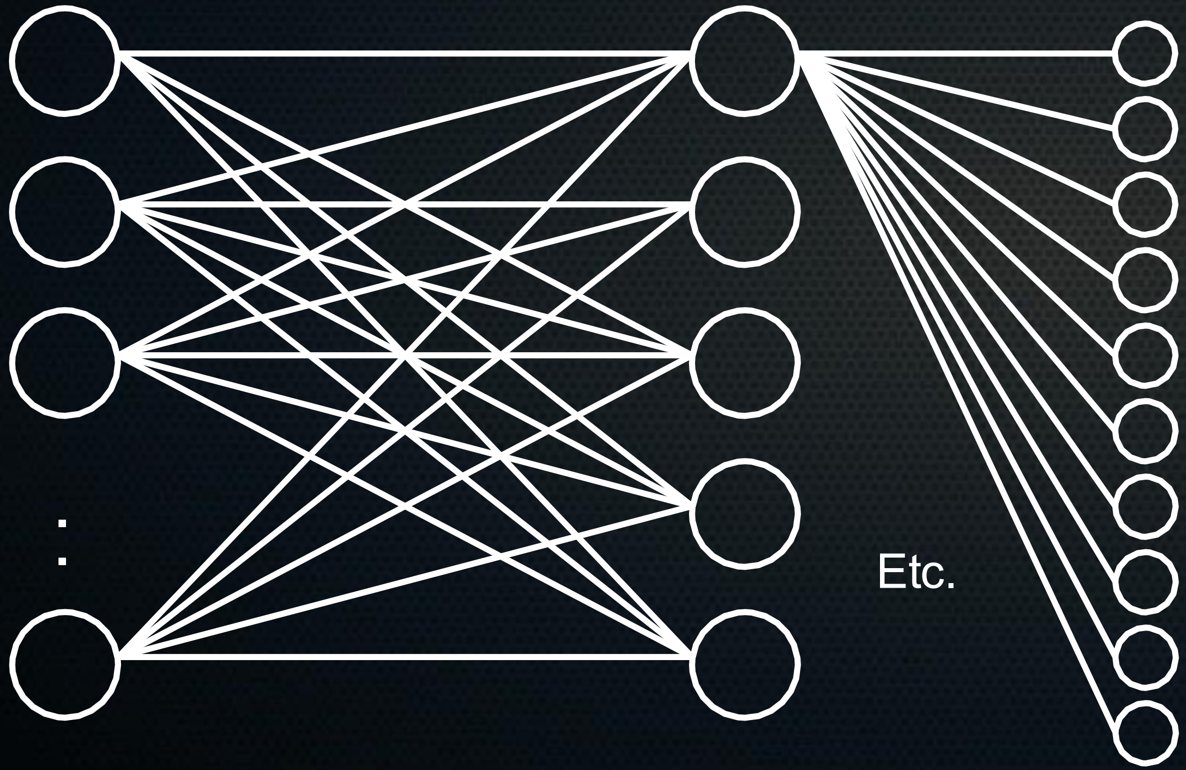
pedestrian   car   motorcycle   truck

~~Previously~~

$\cancel{y \in \{1,2,3,4\}}$

Question to you

- Say we have 10 classes and the following network, how many parameters in $\Theta^{(2)}$?



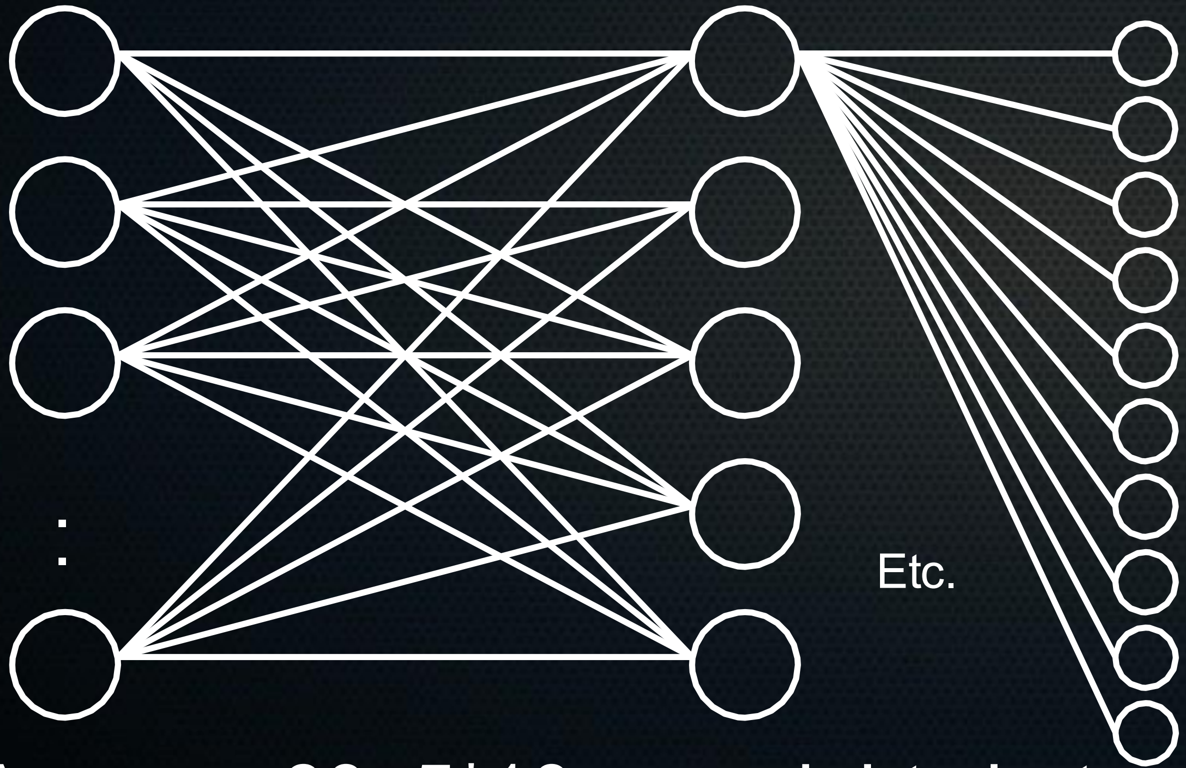
$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

$$y^{(10)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Example:
10th training sample is class 3

Question to you

- Say we have 10 classes and the following network, how many parameters in $\Theta^{(2)}$?



$\Theta^{(j)} =$ matrix of weights controlling function mapping from layer j to layer $j+1$

$$y^{(10)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

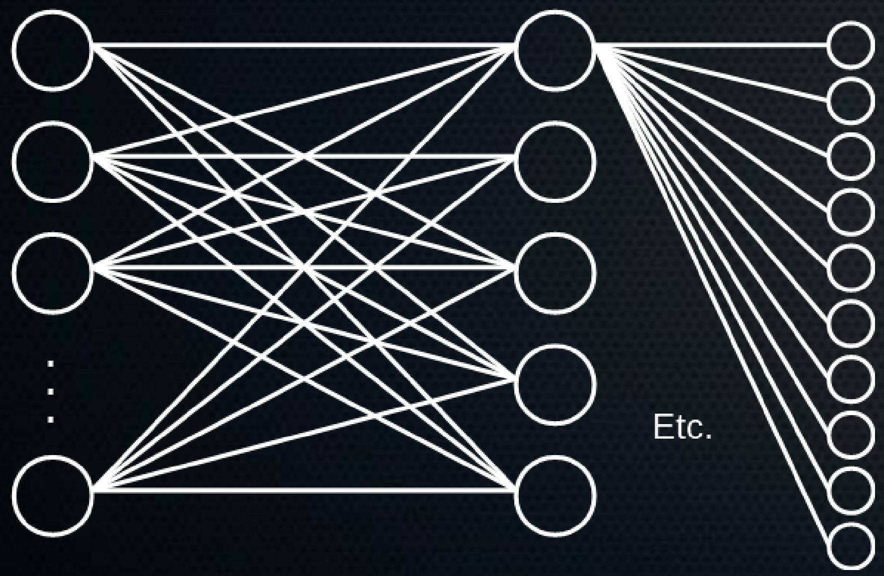
Example:
10th training sample is class 3

- Answer: 60. 5*10 → weights between units.
  + 10 → bias of each unit in output layer

Question to you

- Say we have 10 classes and the following network, how many parameters in $\Theta^{(2)}$?

$\Theta^{(j)} =$ matrix of weights controlling function mapping from layer j to layer $j+1$



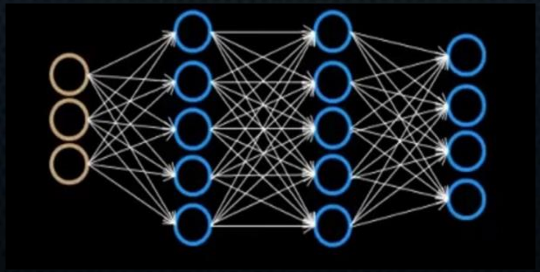
$$\sigma\left(\overbrace{\begin{bmatrix} b_1 & w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ b_2 & w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ b_3 & w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ b_4 & w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ b_5 & w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \end{bmatrix}}^{\Theta^{(2)}} \begin{bmatrix} 1 \\ a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \\ a_5^{(2)} \end{bmatrix}\right)$$

- Answer: 60. 5*10 → weights between units.
  + 10 → bias of each unit in output layer

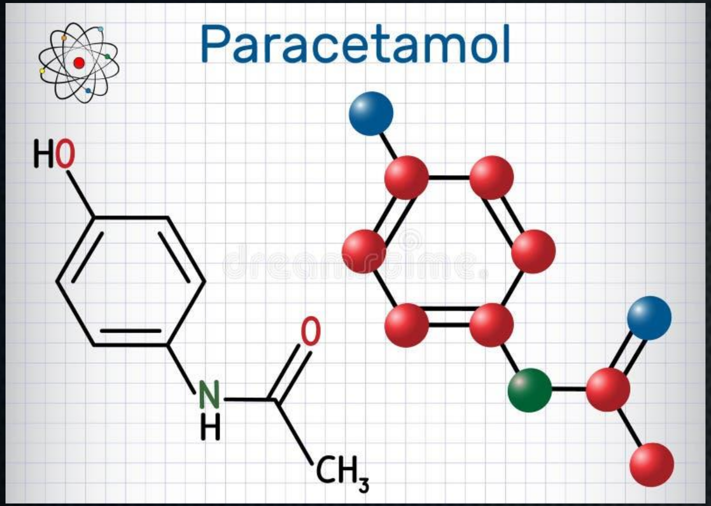
Recap so far

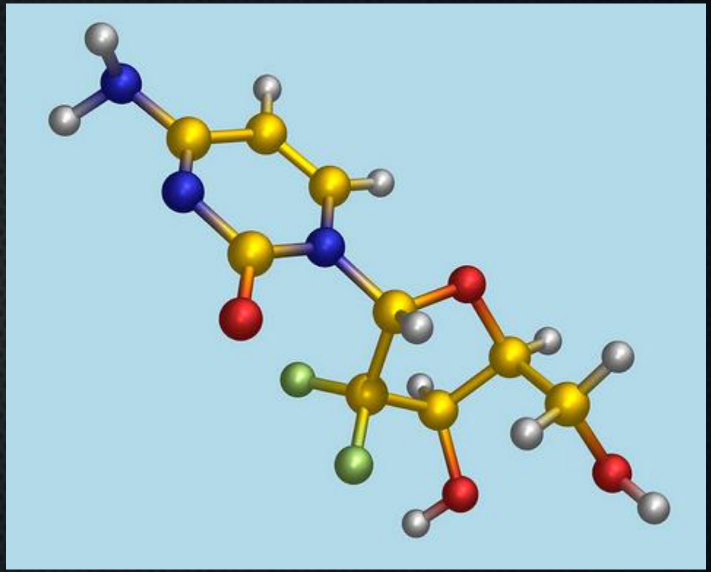
- Neurons in neural networks are not really like biological neurons, except superficially
- Neural networks can be thought of as hiërarchical sets of logistic regressors
- We essentially make earlier layers learn useful features for distinction on their own, and can use these best possible learned features for the classification by the final unit(s)
- Parsing an example through the network and getting the output is called forward propagation
- *Universal approximation* holds that, in principle, neural networks can learn any continuous function arbitrarily well

Adding structure

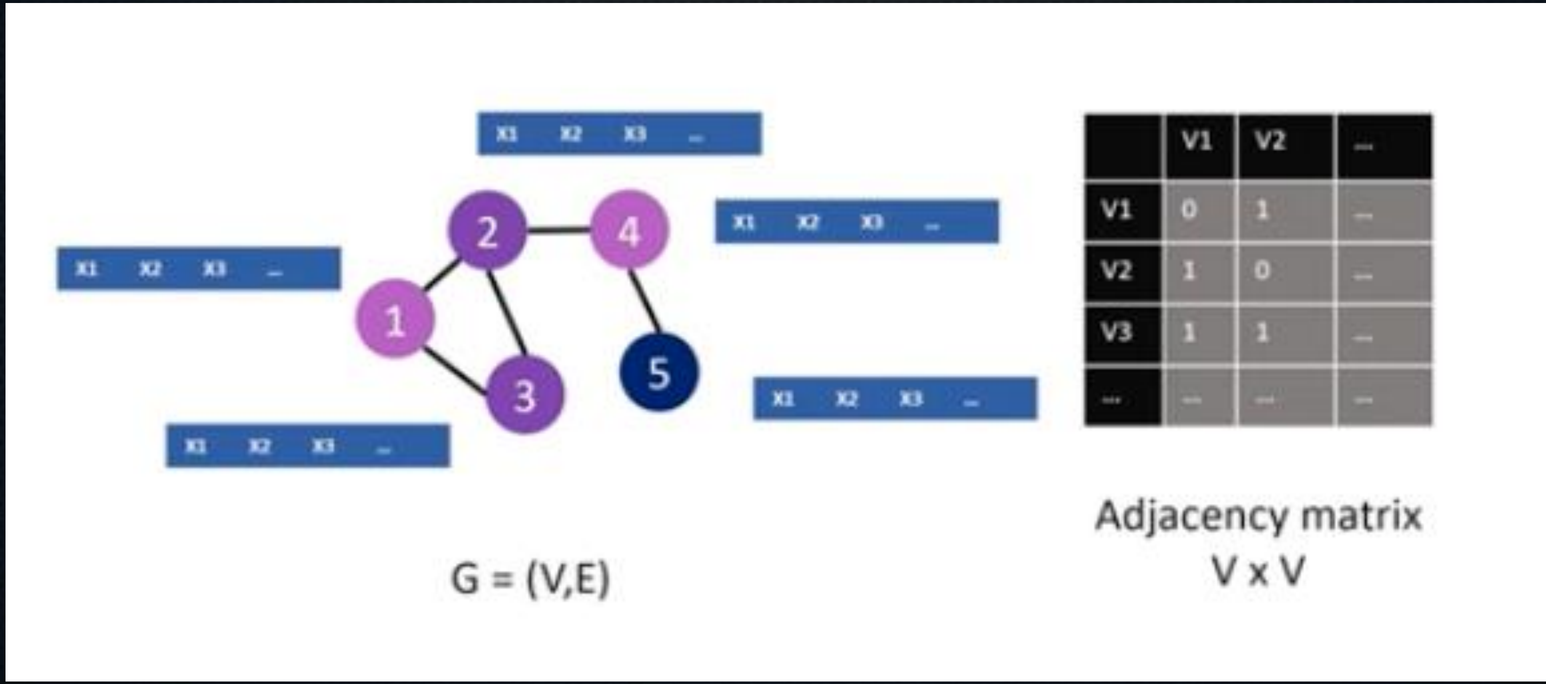


Adding structure



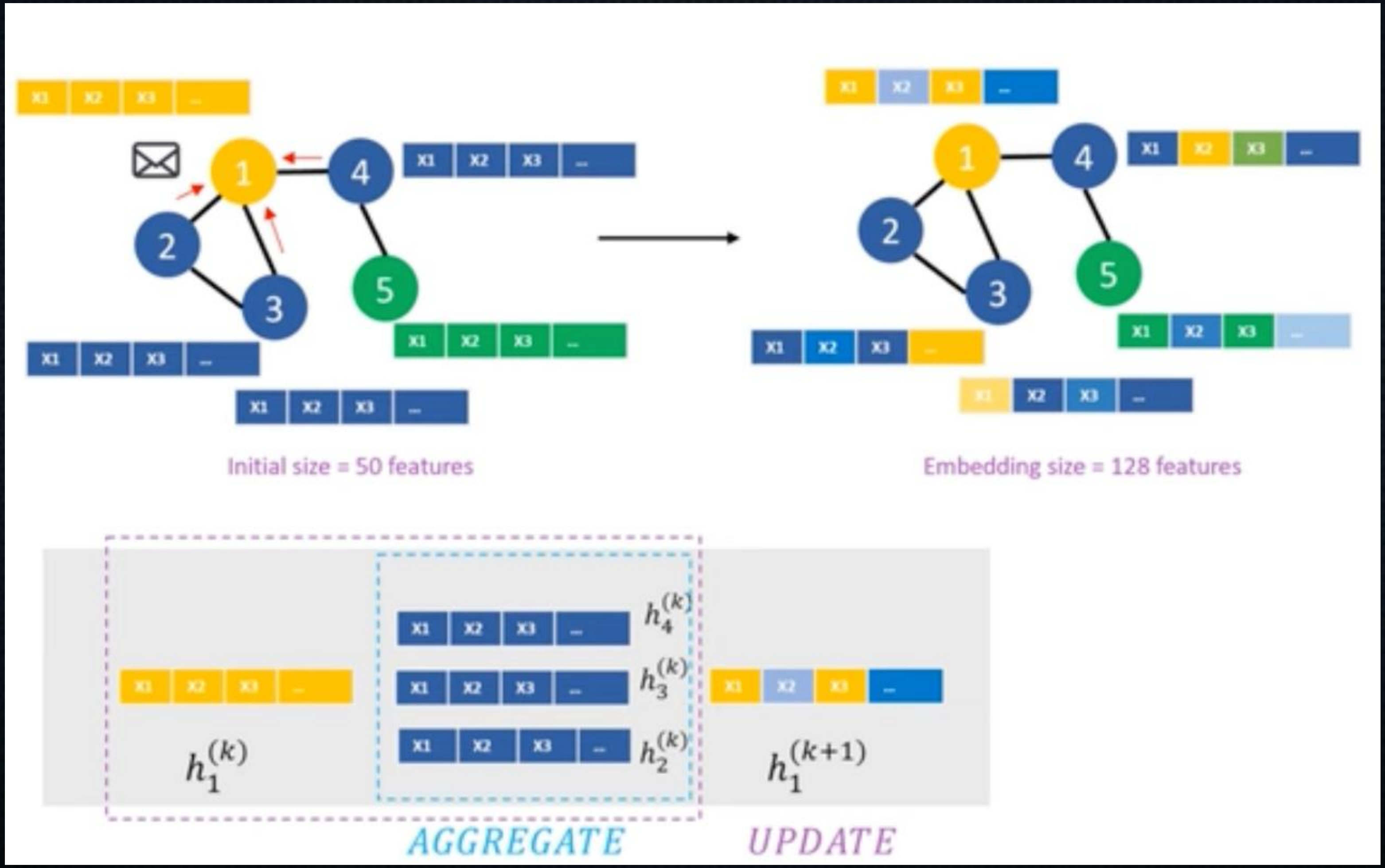


Adding structure



https://www.youtube.com/watch?v=fOctJB4kVlM

Adding structure



https://www.youtube.com/watch?v=ABCGCf8cJOE

Want to learn more?

Relational inductive biases, deep learning, and graph networks

Peter W. Battaglia[1,*], Jessica B. Hamrick[1], Victor Bapst[1], Alvaro Sanchez-Gonzalez[1], Vinicius Zambaldi[1], Mateusz Malinowski[1], Andrea Tacchetti[1], David Raposo[1], Adam Santoro[1], Ryan Faulkner[1], Caglar Gulcehre[1], Francis Song[1], Andrew Ballard[1], Justin Gilmer[2], George Dahl[2], Ashish Vaswani[2], Kelsey Allen[3], Charles Nash[4], Victoria Langston[1], Chris Dyer[1], Nicolas Heess[1], Daan Wierstra[1], Pushmeet Kohli[1], Matt Botvinick[1], Oriol Vinyals[1], Yujia Li[1], Razvan Pascanu[1]

[1]DeepMind; [2]Google Brain; [3]MIT; [4]University of Edinburgh

https://arxiv.org/abs/1806.01261

Super short recap

- Fully-connected neural networks don’t assume any structure in the data at all. No *inductive biases*.

- Often we know some inductive biases:

In images, nearby pixels have similar information, and highly similar picture sub-parts may occur in different locations in the same image.

Many data live on graphs, and probably features need to be locally updated to incorporate this graph structure.

- We will focus on Convolutional Neural Networks tomorrow

Please git pull

To pull changes from git while keeping your own modifications:

`git stash`

remove any .ipynb_checkpoints folder in your practical material if git is mad about them (`rm --rf .ipynb_checkpoints`). These files are saved states of your notebook, but the .ipynb also has a saved state. It's basically a very simple form of backup, which you don't need.

`git pull`

`git pop stash`

Or use github desktop: https://desktop.github.com/download/

Time for the afternoon practical!

*and continuing where you are