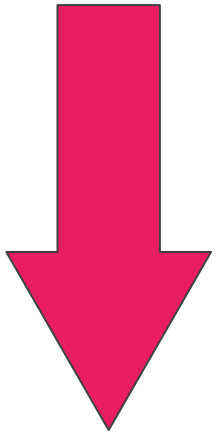
Intro to Artificial Neural Network

3 layers of abstraction

- Level 1: Neuron
- Level 2: Network
- Level 3: Cost function

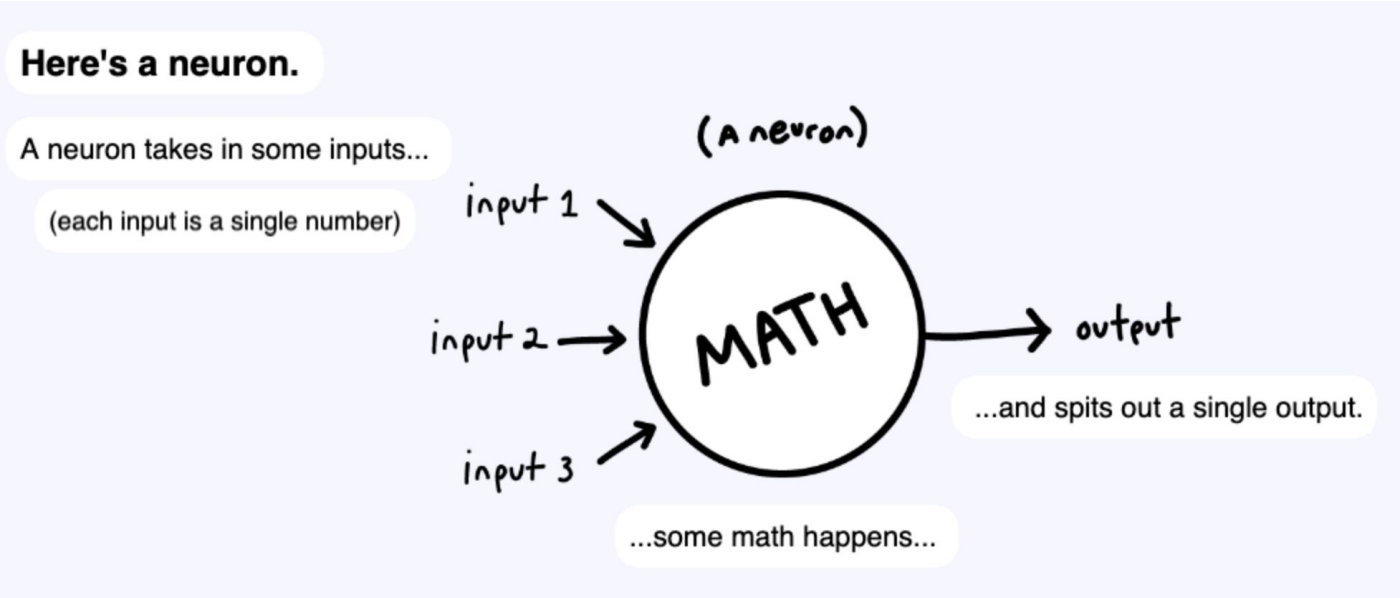


Increasing in complexity

Level 1: Neuron

Neuron == Function

- many inputs
- one output

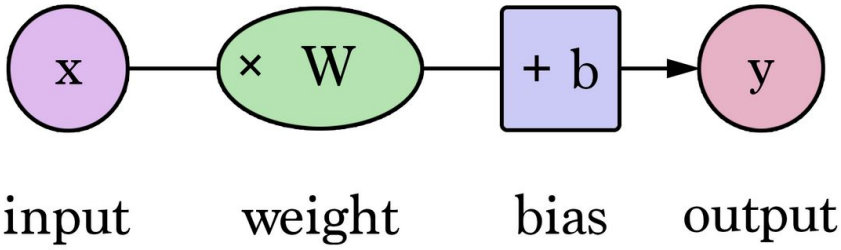


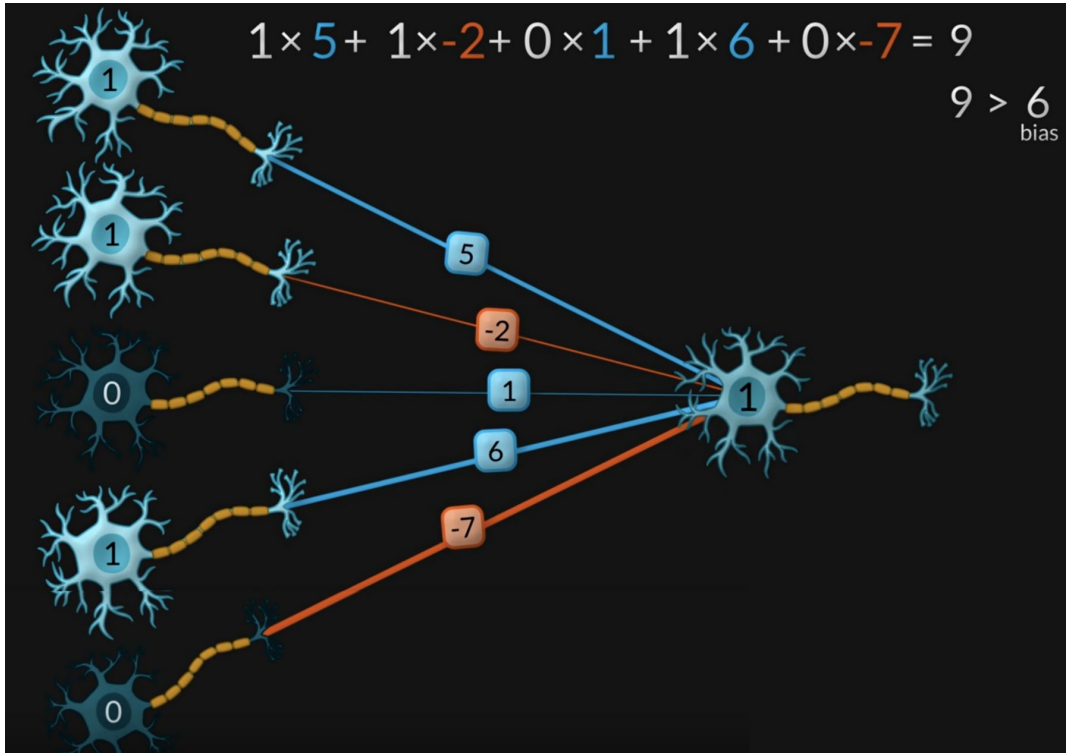
Neuron connections

- loosely based on how neurons fire in the brain

- input - excitatory vs inhibitory connections

- output - activation
(0 vs 1) - depends on bias threshold

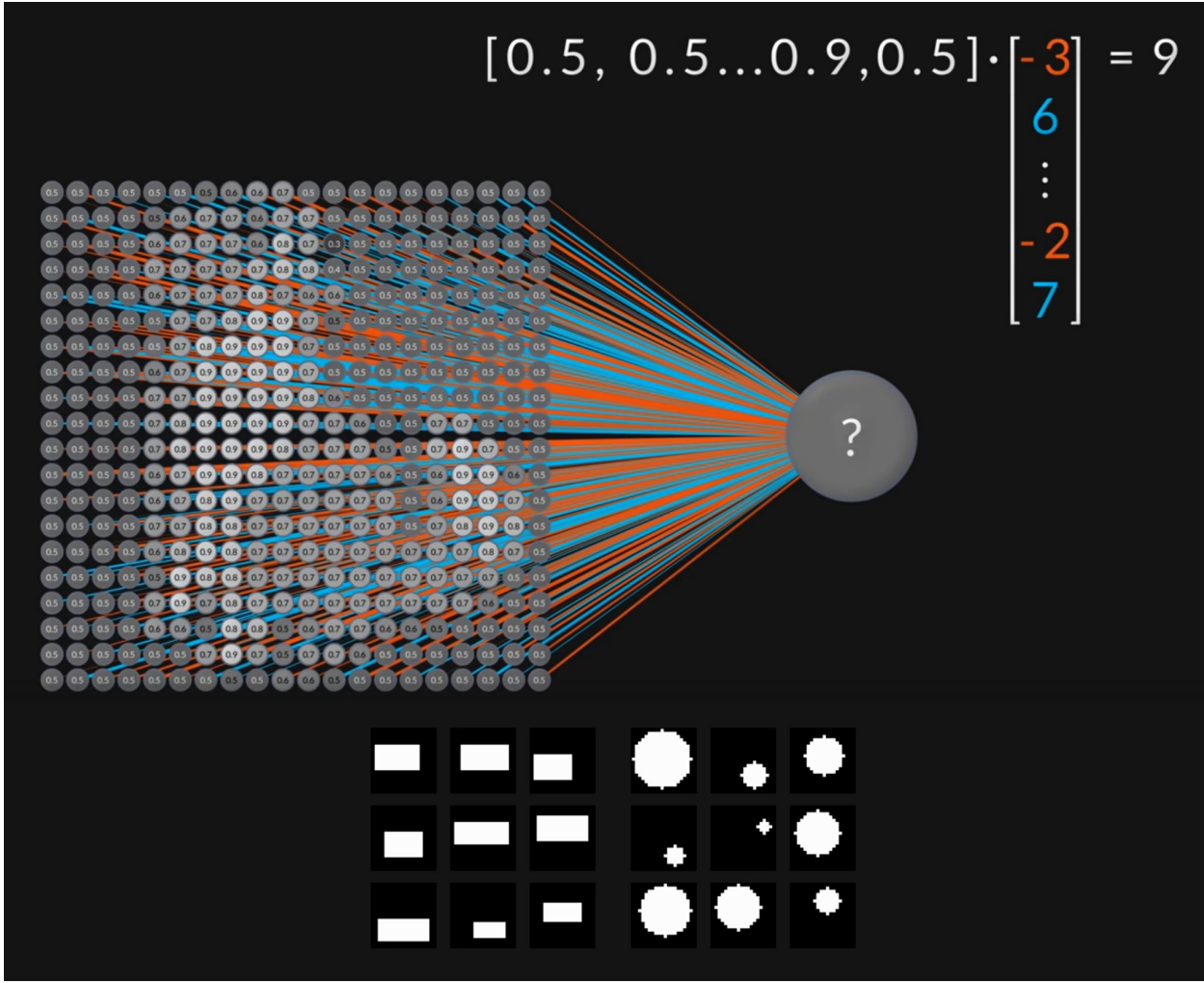




Perceptron

1958 - Cornell University, USA

- 20x20 pixel image input
- single layer
- output:
  - 0: rectangle
  - 1: circle

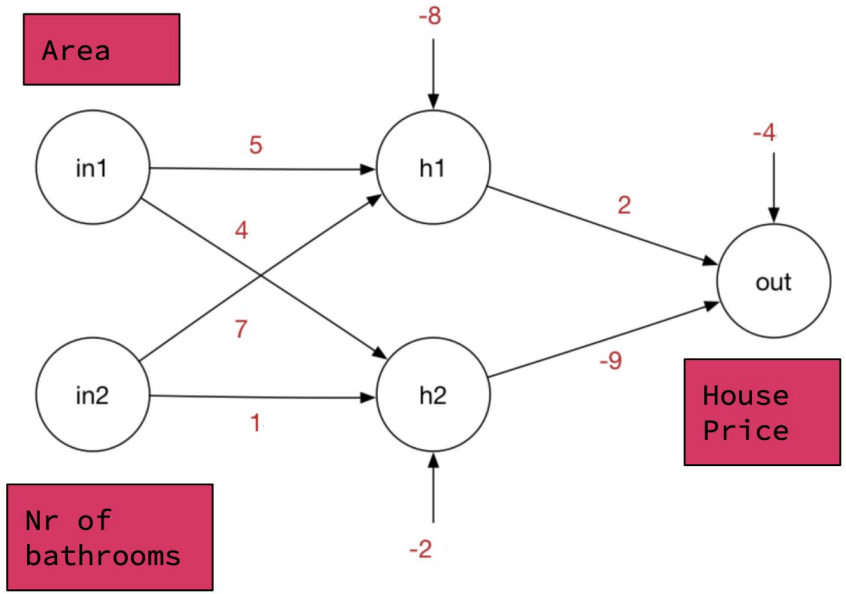


Level 2: Network

Multi-Layer Network == Parameterized Function

- inputs (2)
- output (1)
- parameters (9)
  - weights - strength of connection
  - biases - threshold, tendency to be active

What if we add another neuron in the middle?

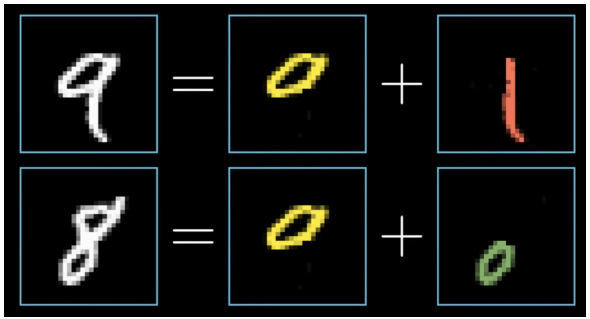


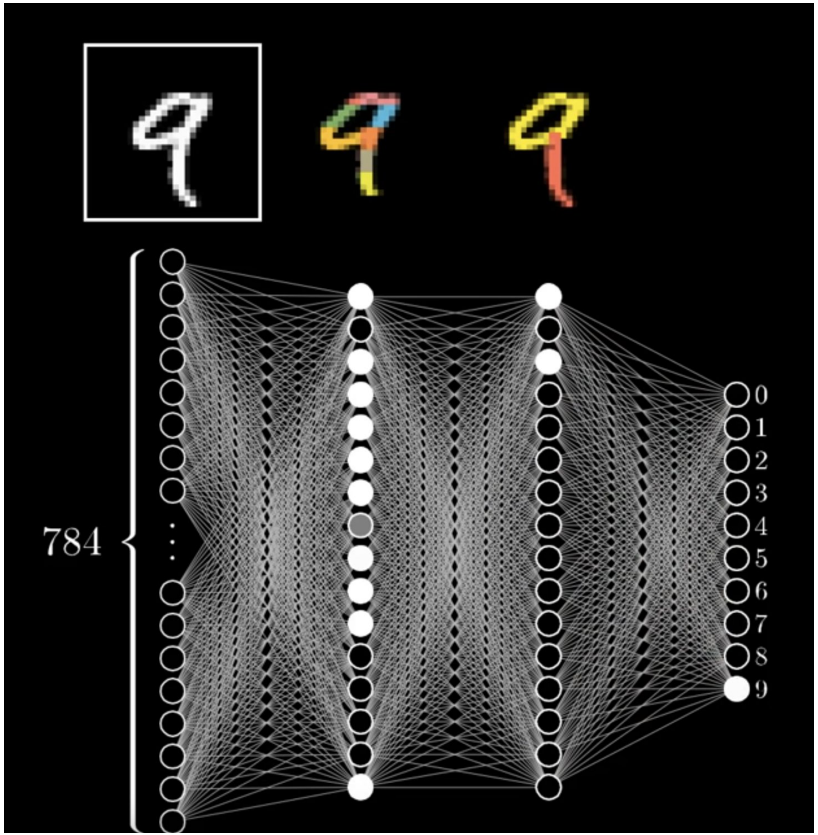
Multiple layers

What is the benefit?

- intuition/expectation - breakdown problem into components

- layered structure allows for increasingly complex operations





Linearity

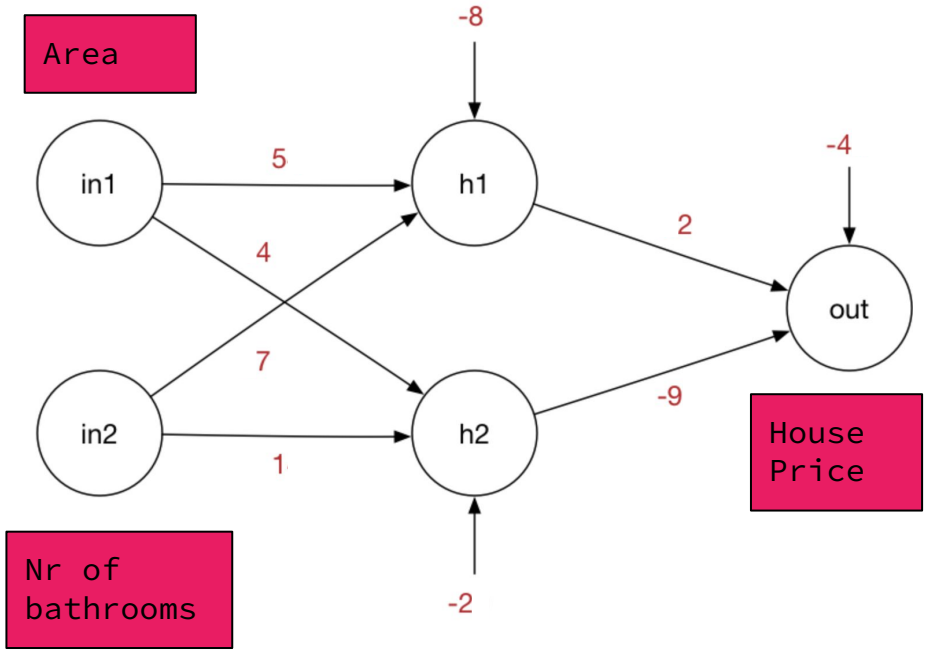
---

a linear combination of
a linear combination is still
a linear combination

$$out = 2 * (5 * in1 - 8) - 4$$

$$\begin{aligned} out &= 10 * in1 - 16 - 4 \\ &= 10 * in1 - 20 \end{aligned}$$

$$out = w * in + b$$



Linearity

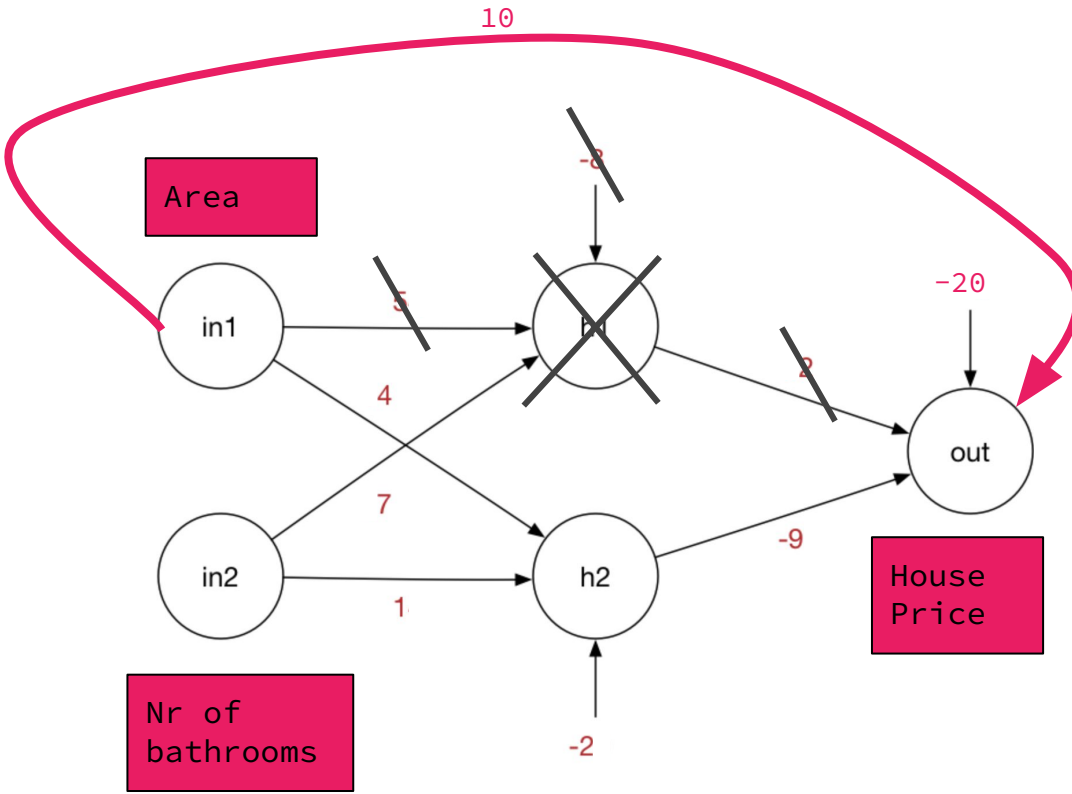
---

a linear combination of
a linear combination is still
a linear combination

$$out = 2 * (5 * in1 - 8) - 4$$

$$out = 10 * in1 - 16 - 4$$
$$= 10 * in1 - 20$$

$$out = w * in + b$$



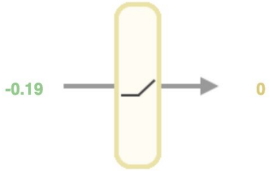
Activation functions

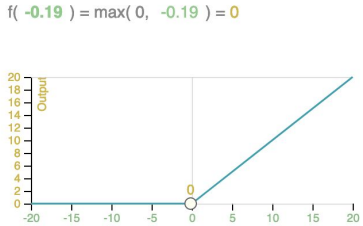
- provide non-linearity, which help neural networks perform more complex tasks

- neural networks can model any given function approximately by having appropriate parameters (weights and biases)

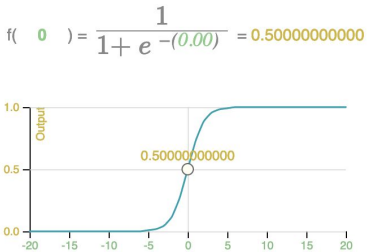
- UAT - Universal Approximation Theorem (video proof intuition)

ReLU Visualization #





Sigmoid Visualization #



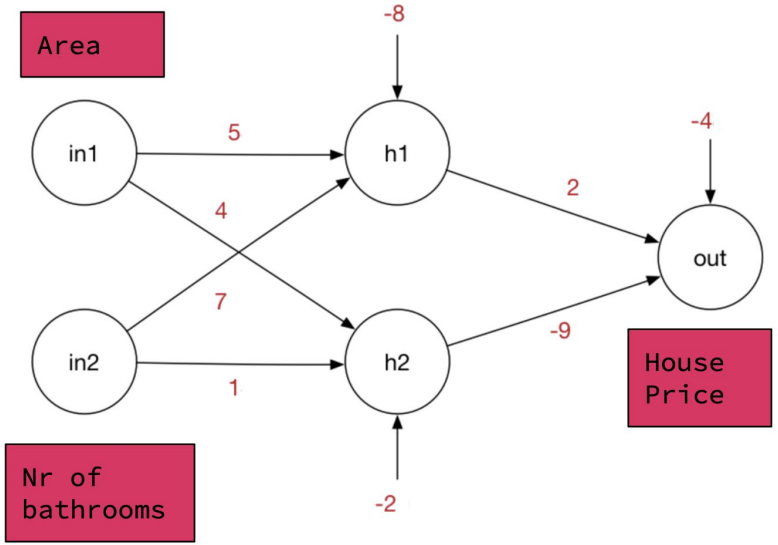
Machine learning?

- What does learning mean?

Finding the best values for the weights and biases

- How can we know if some values of w&b are better than other values?

We need a labeled dataset, where we know what is the correct output
(and we can compare this to the prediction)



Area (in1)	Nr of Bathrooms (in1)	House Price (out)
85 m2	1	200.000 EUR
120 m2	2	350.000 EUR
60 m2	1	100.000 EUR
250 m2	3	750.000 EUR
90 m2	2	175.000 EUR
75 m2	1	140.000 EUR

Prediction error

- difference between predicted output and correct/true output

- diff can be positive or negative - we need to square (SE) or take absolute value (AE)

- diff varies based on the example, so to be able to globally measure the error - we need to average (MSE)

- to have a smaller average value, we can take the square root of the average (RMSE)

House Price (out)	Predicted House Price	Diff / Error	Squared Error
200.000 EUR	190.000 EUR	-10,000	100,000,000
350.000 EUR	390.000 EUR	40,000	1,600,000,000
100.000 EUR	105.000 EUR	5,000	25,000,000
750.000 EUR	600.000 EUR	-90,000	8,100,000,000
175.000 EUR	150.000 EUR	-25,000	625,000,000
140.000 EUR	145.000 EUR	5,000	25,000,000

Mean Squared Error	Root Mean Squared Error
1,745,833,333	41,783

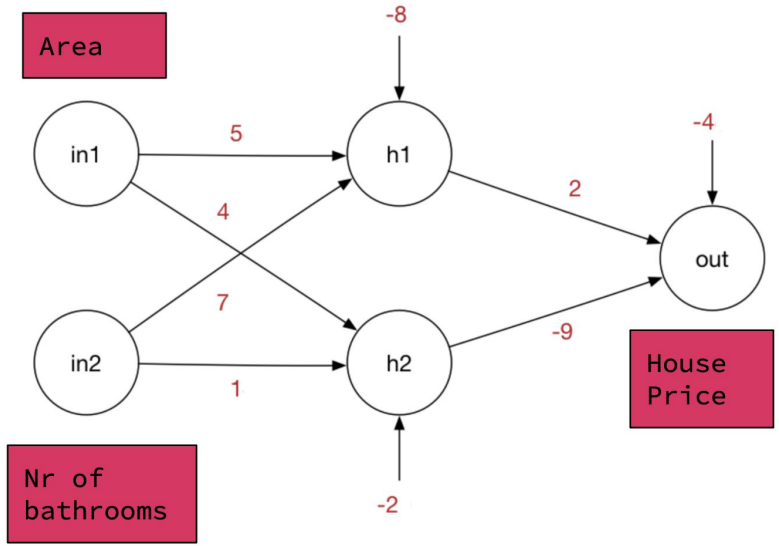
Level 3: Cost function

- measure how bad/good is the performance of the neural network across all training examples

- example: RMSE for all training data

Cost function == Parameterized Function

- inputs (w&b of the network)
- output (1 number - the error/cost)
- parameters: training examples

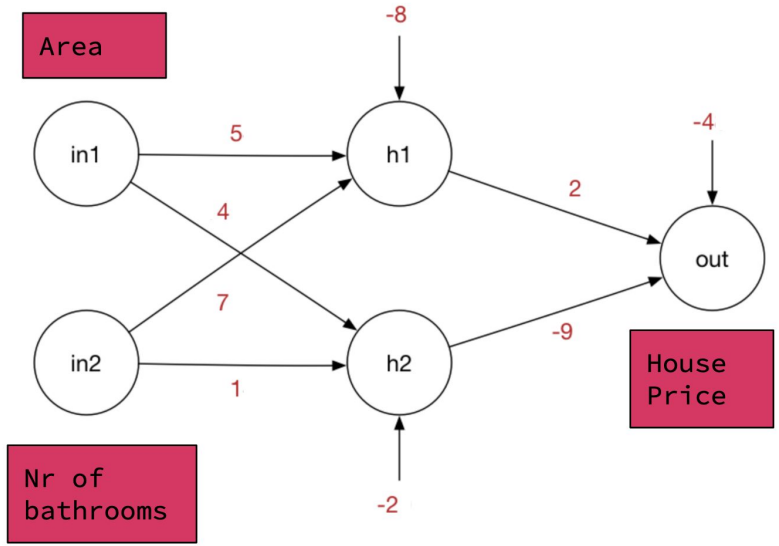


Mean Squared Error	Root Mean Squared Error
1,745,833,333	41,783

Machine Learning == Minimizing Cost Function

- changing the weights and biases - changes the output of the cost function

- how do we need to change the w&b - in order to minimize the cost? minimize RMSE?



Mean Squared Error	Root Mean Squared Error
1,745,833,333	41,783

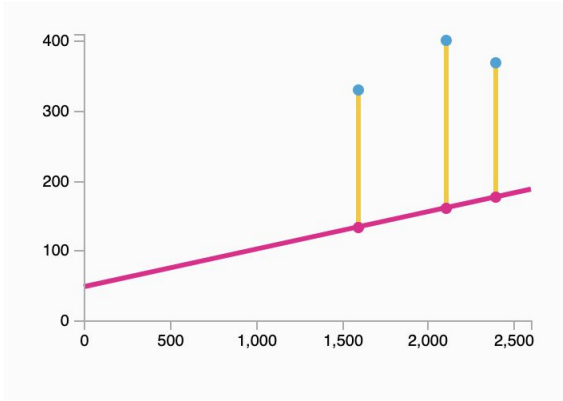
Linear Regression

$$y = Wx + b$$

- fitting a straight line

- minimize error == minimize cost function

Area (sq ft) (x)	Price (y)
2,104	399,900
1,600	329,900
2,400	369,000



Error 44,311

Weight 0.053

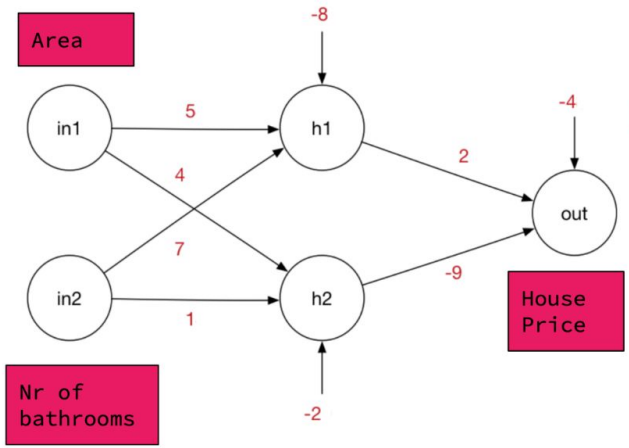
Bias 49.0

Manual Machine Learning

		h1	ReLU(h1)
		424	424
		606	606
		299	299
Weight: in1→h1	5	1263	1263
Weight: in2→h1	7	456	456
Bias: h1	-8	374	374

Weight: h1→out	2
Bias: out	-4

Area (m2) - in1
85
120
60
250
90
75



Bathrooms (nr) - in2
1
2
1
3
2
1

True House Price	Predicted House Price	Squared Error	Root Mean Squared Error
200	-2,207	5,793,649	3,731
350	-3,112	11,985,444	
100	-1,557	2,745,649	
750	-6,487	52,374,169	
175	-2,332	6,285,049	
140	-1,947	4,355,569	

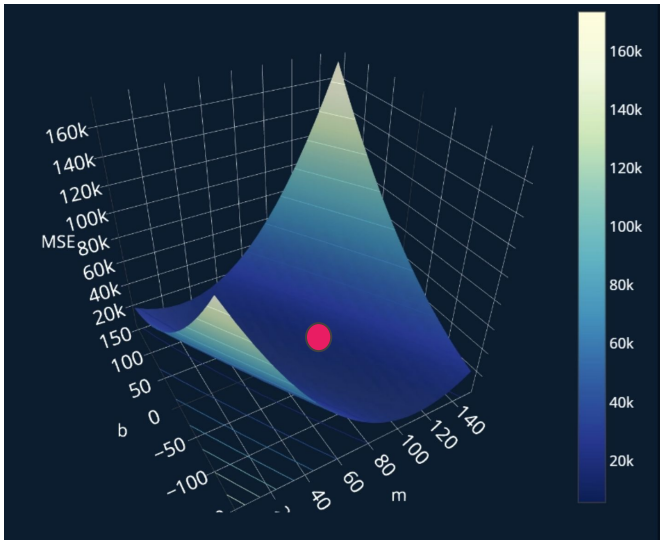
Baseline RMSE	Improvement
3,731	0.0%

		h2	ReLU(h2)
		339	339
		480	480
		239	239
		1001	1001
		360	360
		299	299
Weight: in1→h2	4		
Weight: in2→h2	1		
Bias: h2	-2		

Weight: h2→out	-9

Automatized Machine Learning

Linear Algebra + Calculus/Derivatives



Simple case (linear regression)

- minimum value can be directly calculated with a closed form equation

- using matrix operations - the normal equation

$$\theta = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

Automatized Machine Learning

Linear Algebra + Calculus/Derivatives

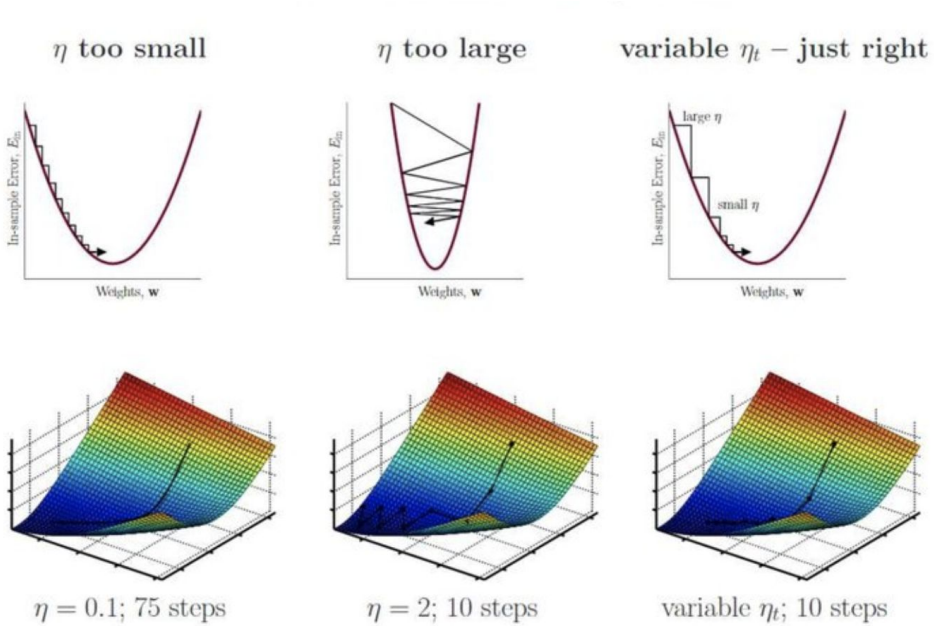
Complex case (neural net)

- optimization algorithm

- starting with random w&b and incrementally adjusting to improve (gradient descent)

- gradient vector - direction of the biggest slope (multivariable derivative - how does a small change in input affect/change the output)

- chain rule & backpropagation



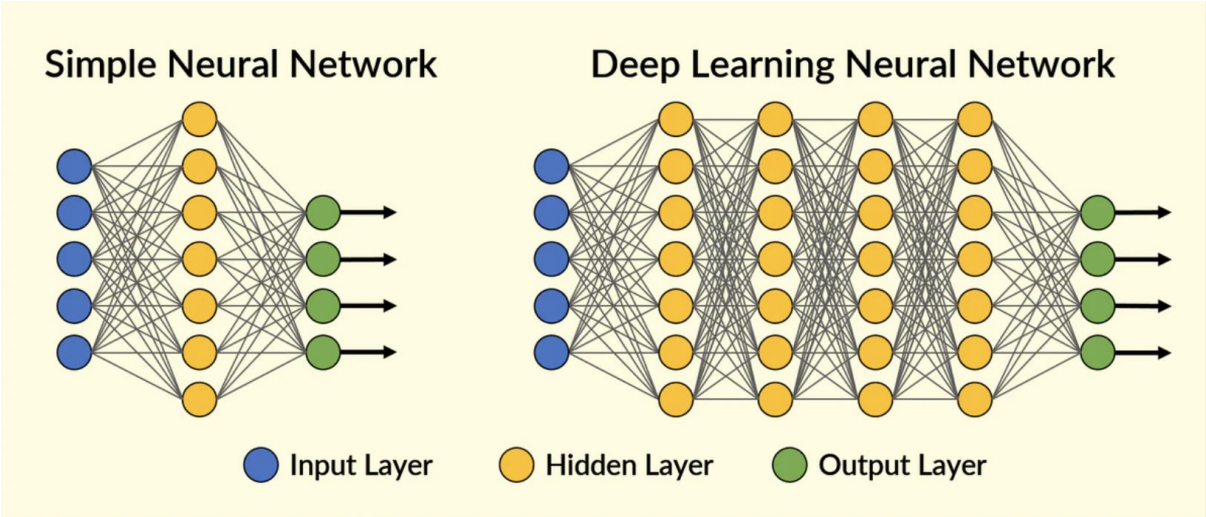
Deep Learning

- lots of hidden layers

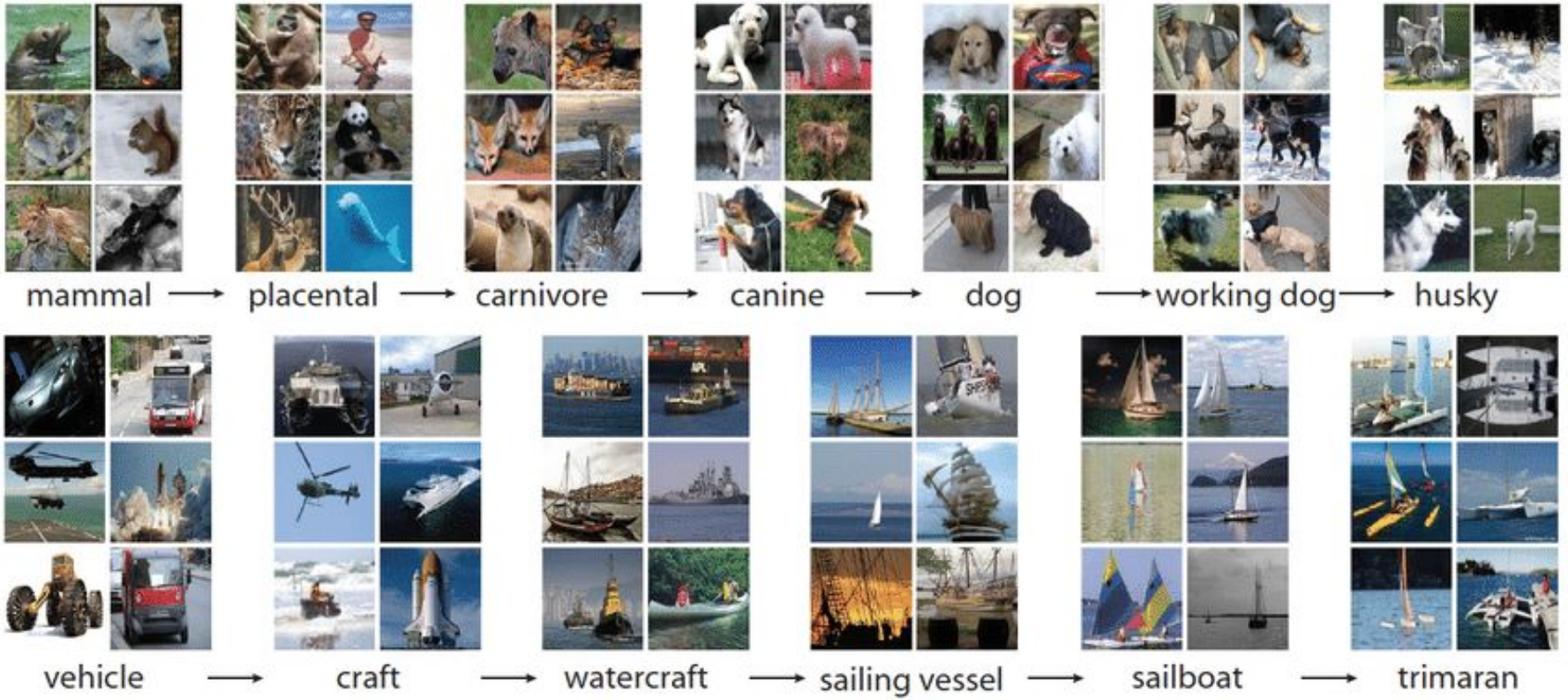
- special techniques to avoid vanishing/exploding gradients

- heavy compute resource needs TPU/GPU

- lots of data



ImageNet Dataset



ImageNet Challenge

- 14 million images - 20.000 categories

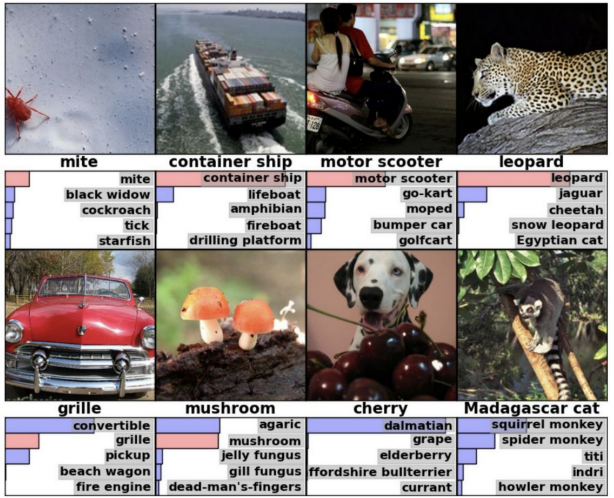
- hand labelled/crowdsourced (2006-2009)

- designed/created to train visual object recognition software

- ImageNet Challenge (ILSVRC) - 2010-2017 Large Scale Visual Recognition Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

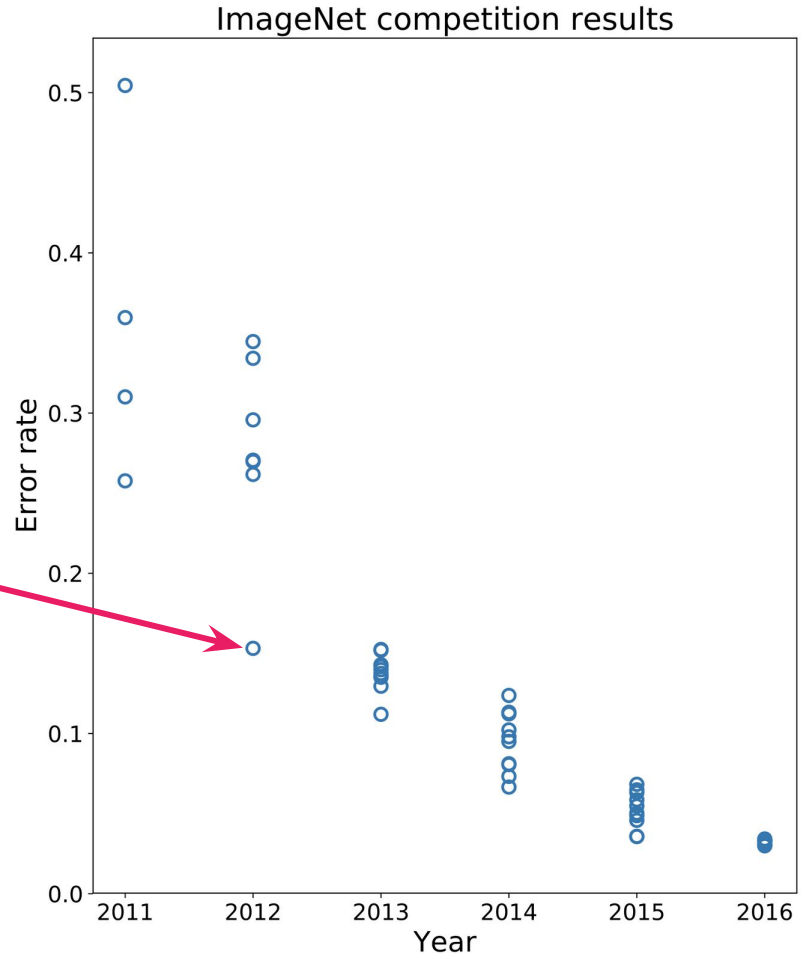


ImageNet Moment 2012

- 2010-2011: top-5 error rate ~25%

- 2012: top-5 error rate 16%
(>10 percentage points improvement)

- 2015 - better than human level performance (<4.5%)

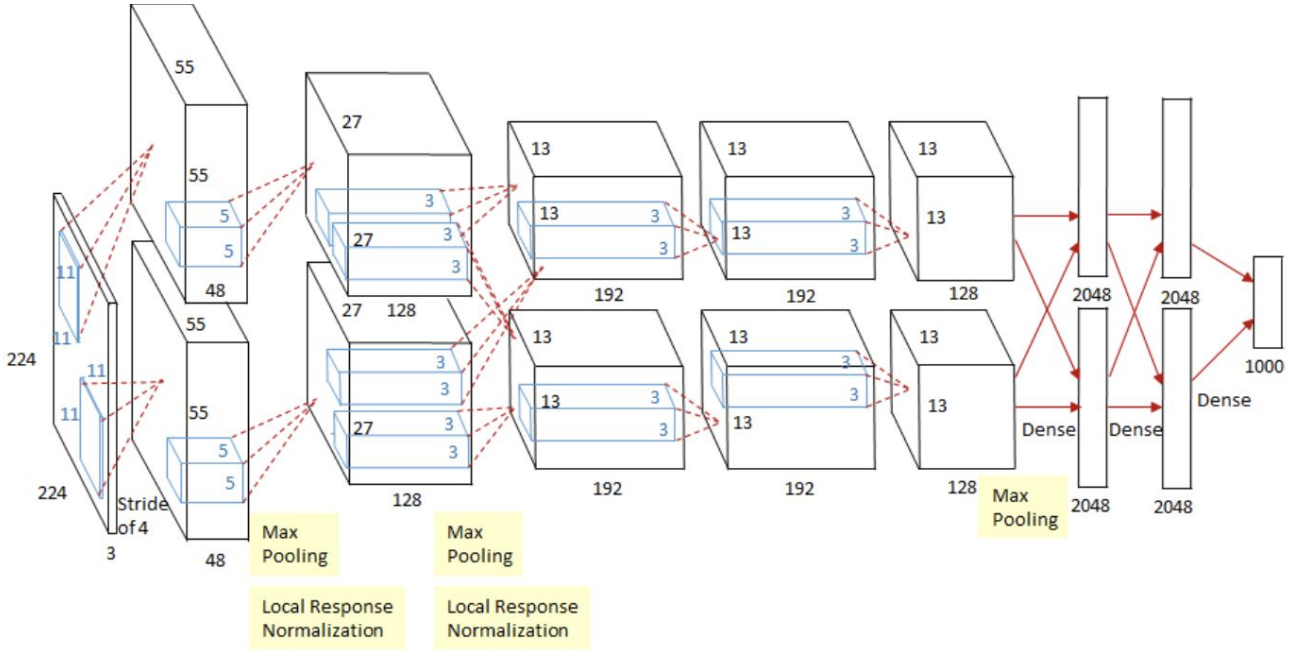


AlexNet

60 million parameters

different types of layers:

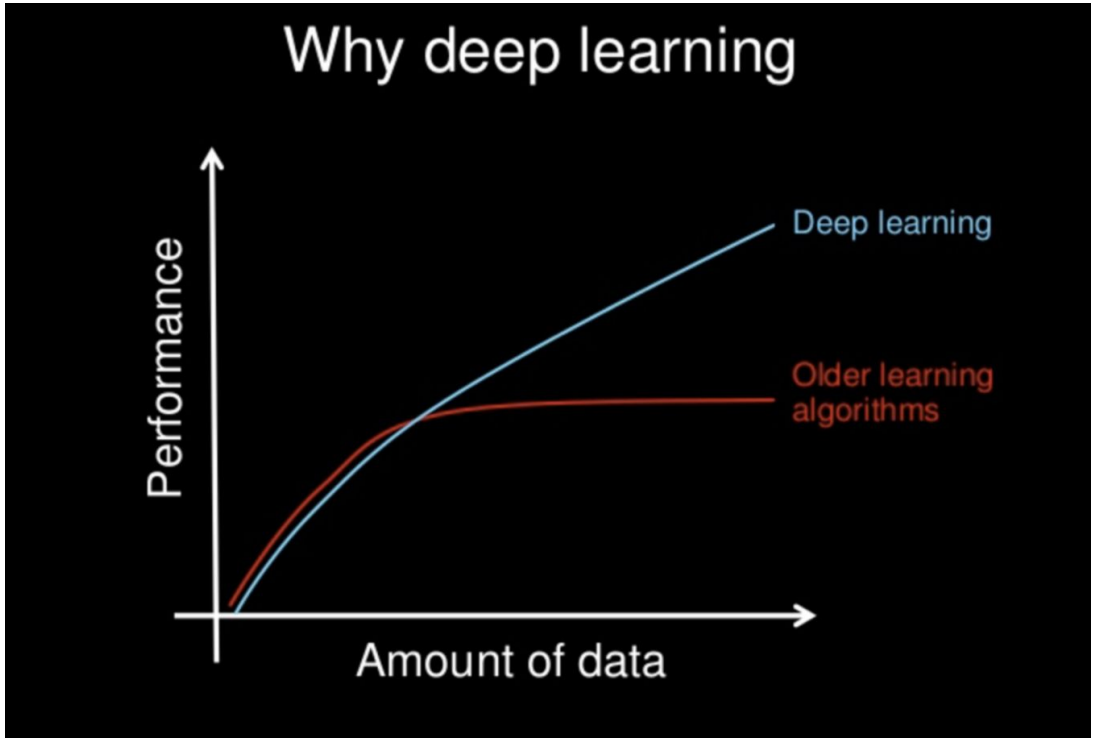
- Convolution
- Pooling
- Dense



Why it took 50 years?

- large dataset

- cheap/efficient compute



References

- Youtube - Veritasium - Perceptron
- Youtube - 3blue1brown - Machine Learning series
- Neural Net playground - Strawberry vs Blueberry
- Jalammar - Interactive activation function
- Kaggle Notebook - Pytorch NN Tutorial
- Towards Data Science article - NN, UAT