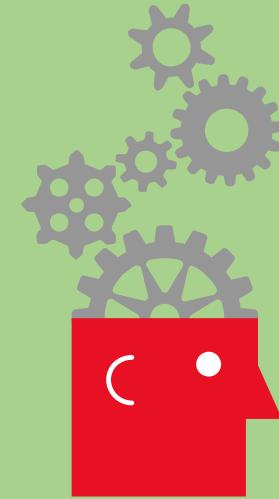


Artificial Intelligence and Machine Learning





1000kg

40mpg



3000kg

20mpg



2000kg



?

inputs



1000kg



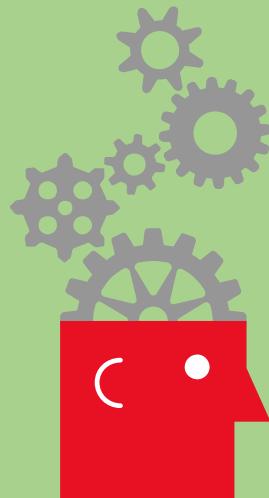
3000kg



2000kg

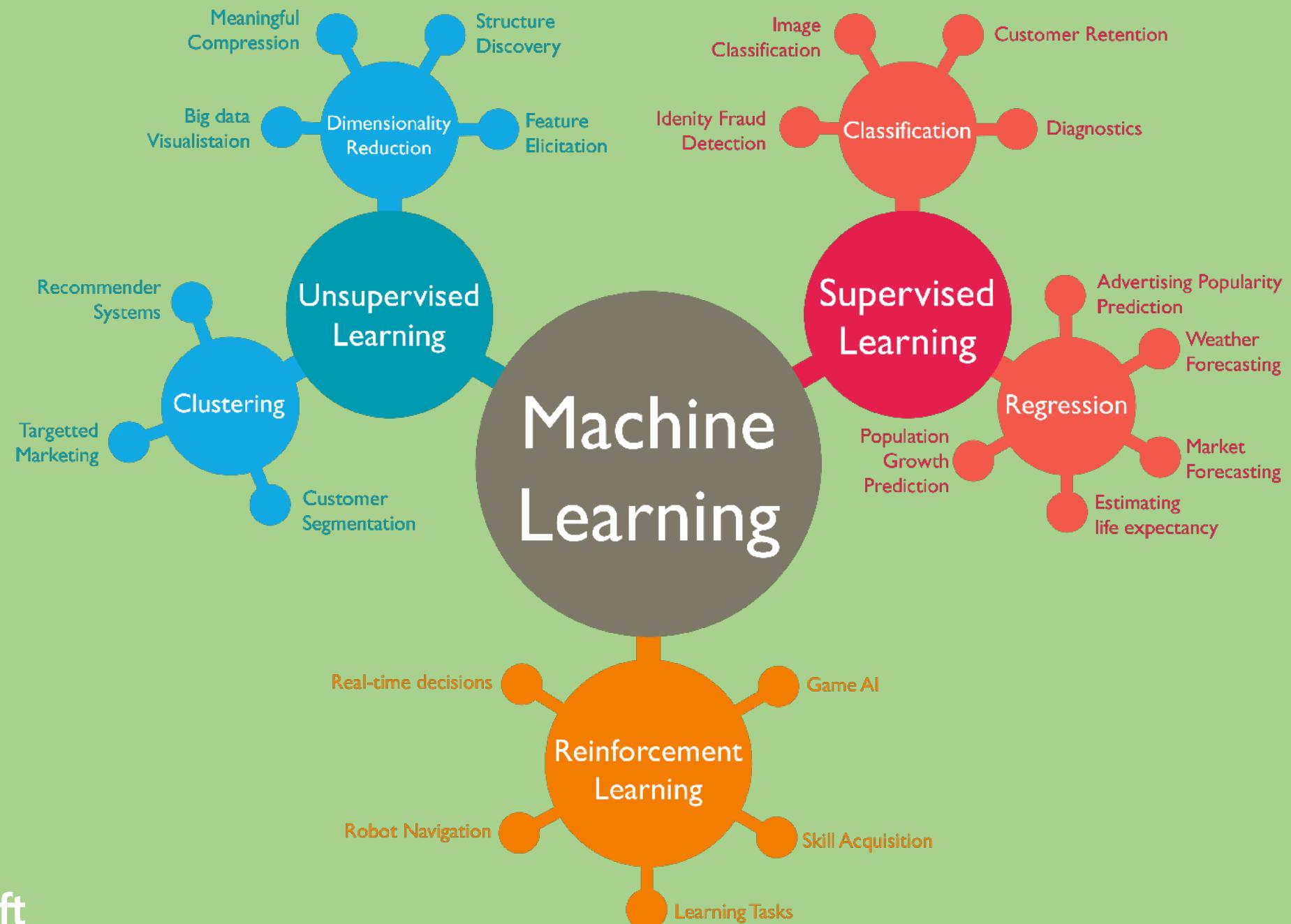
output

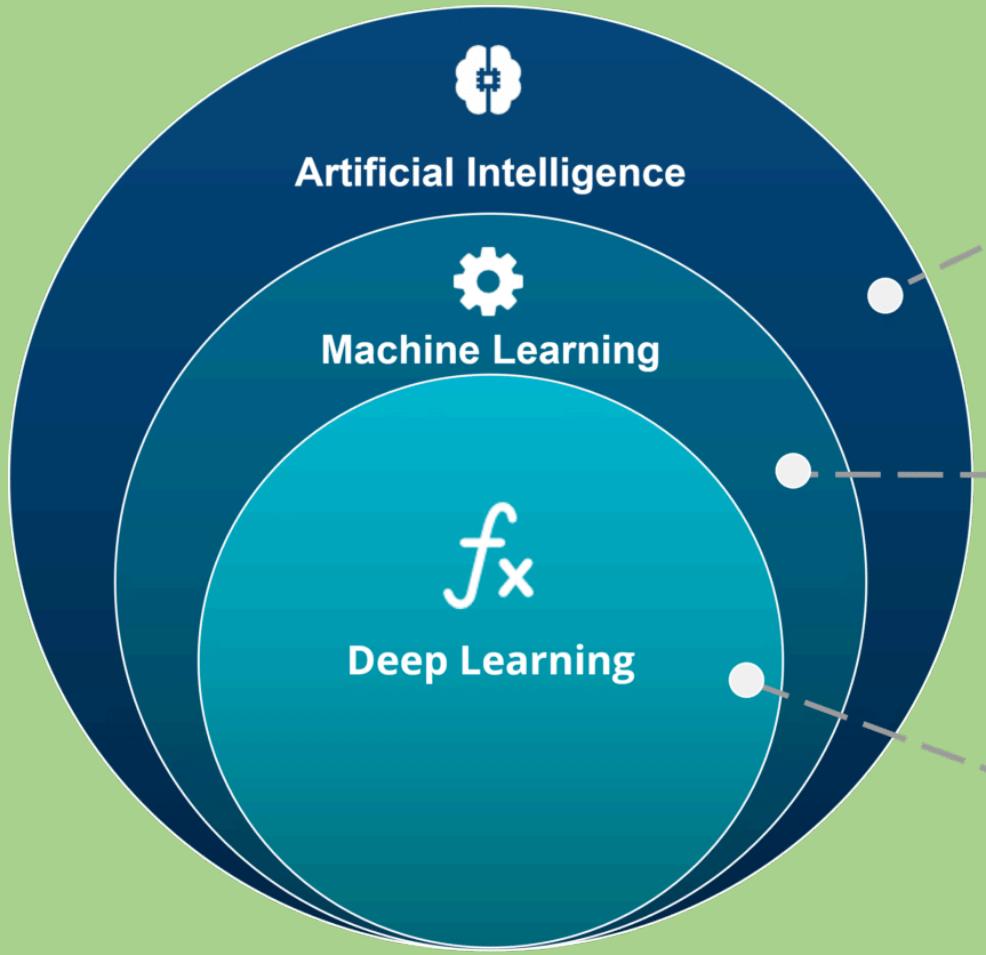
40mpg



20mpg

30mpg





ARTIFICIAL INTELLIGENCE

A technique which enables machines to mimic human behaviour

MACHINE LEARNING

Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING

Subset of ML which make the computation of multi-layer neural network feasible

AI Example

How autonomous cars understand the world around them



Source: FT research Graphic: Ian Bott
© FT



Perception

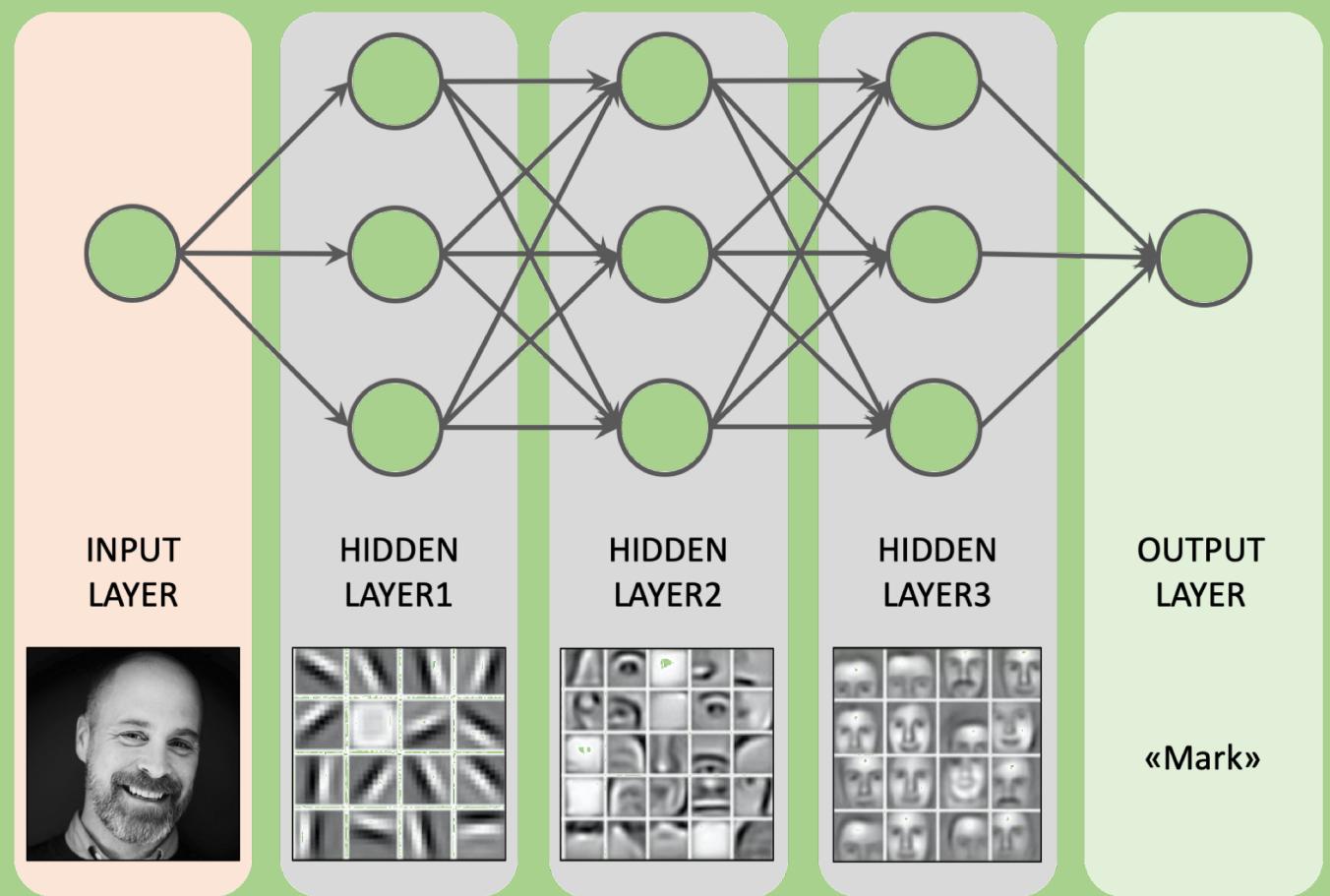
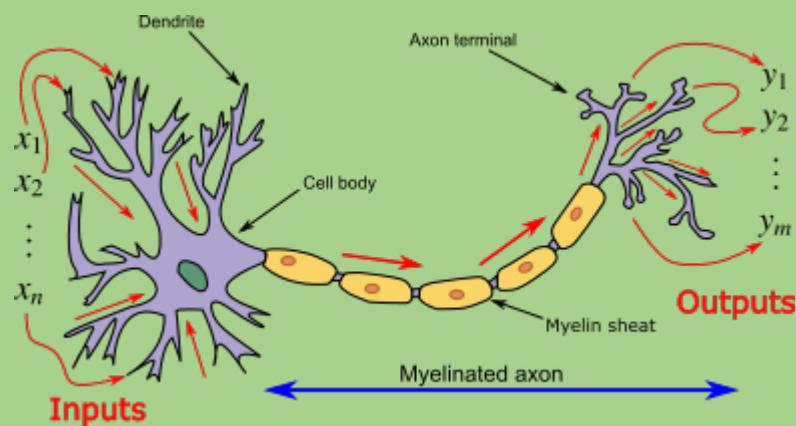


Prediction

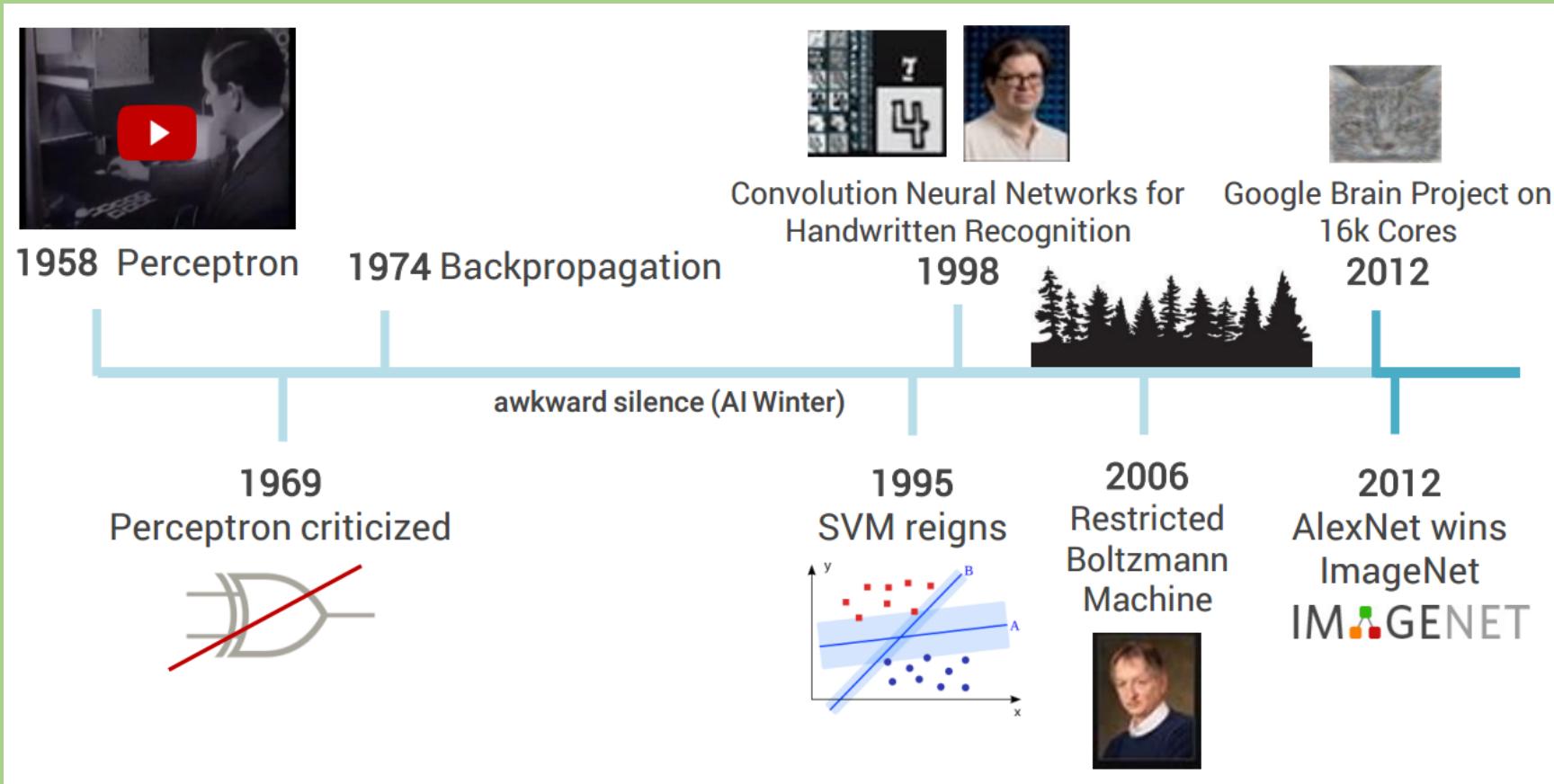


Planning

Deep Learning / Neural Networks



Deep Learning





[NIPS Home](#)

[Overview](#)

[Conference Videos](#)

[Workshop Videos](#)

[Program Highlights](#)

[Tutorials](#)

[Conference Sessions](#)

[Workshops](#)

[Publication Models](#)

[Demonstrations](#)

[Mini Symposia](#)

[Accepted Papers](#)

[Dates](#)

[Committees](#)

[Li Deng, Dong Yu, Geoffrey Hinton](#)

[Microsoft Research; Microsoft Research; University of Toronto](#)

[Deep Learning for Speech Recognition and Related Applications](#)

7:30am - 6:30pm Saturday, December 12, 2009

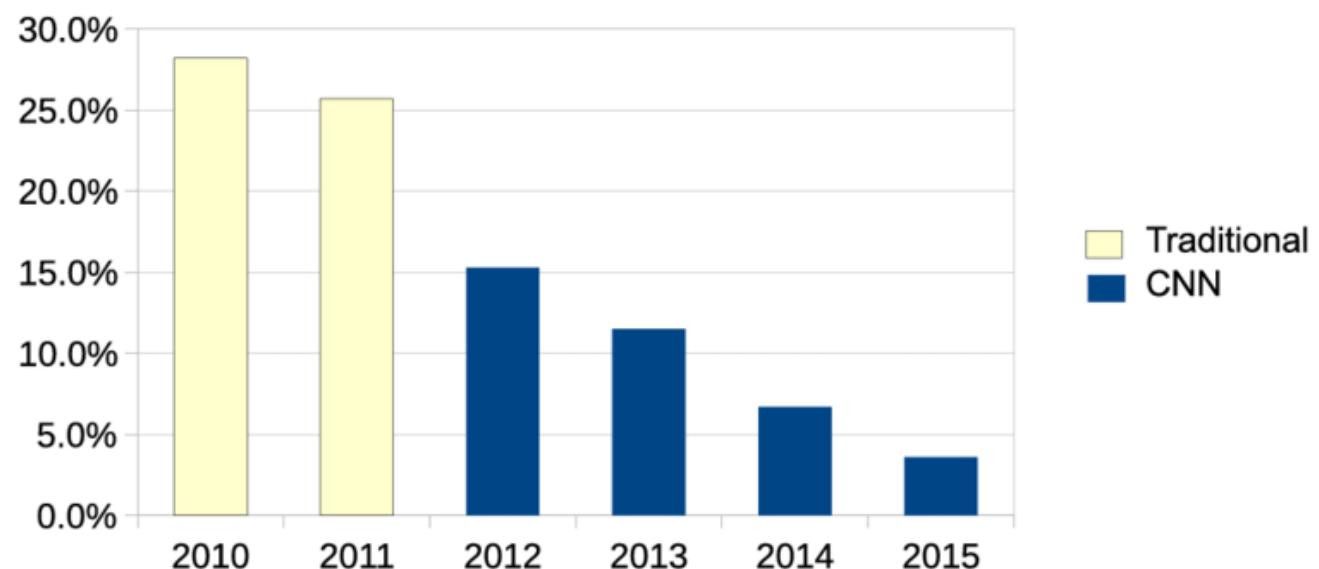
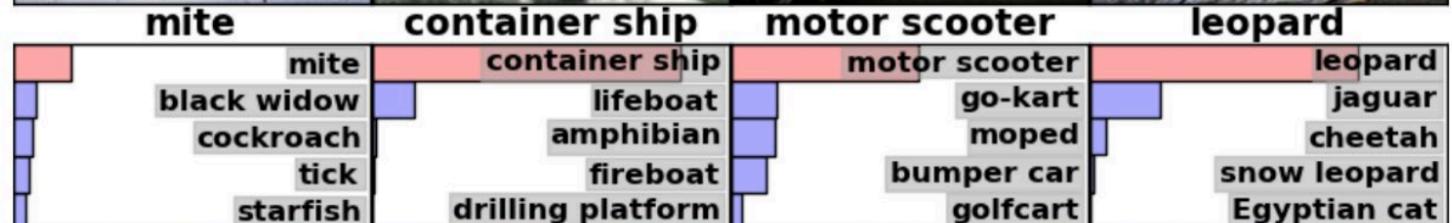
Location: Hilton: Cheakamus

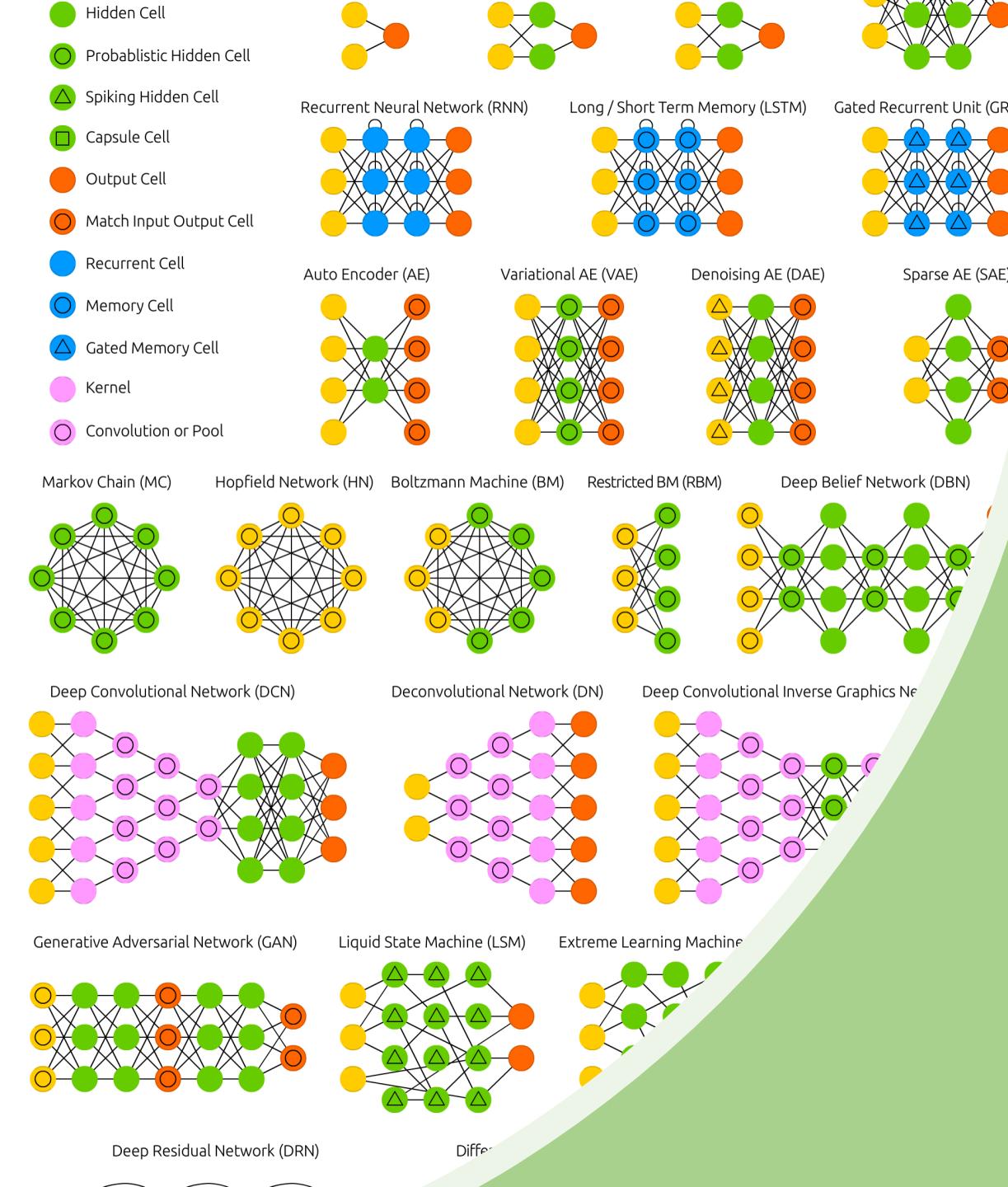
Abstract: Over the past 25 years or so, speech recognition technology has been dominated by a "shallow" architecture --- hidden Markov models (HMMs). Significant technological success has been achieved using complex and carefully engineered variants of HMMs. The next generation of the technology requires solutions to remaining technical challenges under diversified deployment environments. These challenges, not adequately addressed in the past, arise from the many types of variability present in the speech generation process. Overcoming these challenges is likely to require "deep" architectures with efficient learning algorithms. For speech recognition and related sequential pattern recognition applications, some attempts have been made in the past to develop computational architectures that are "deeper" than conventional HMMs, such as

ImageNet Challenge

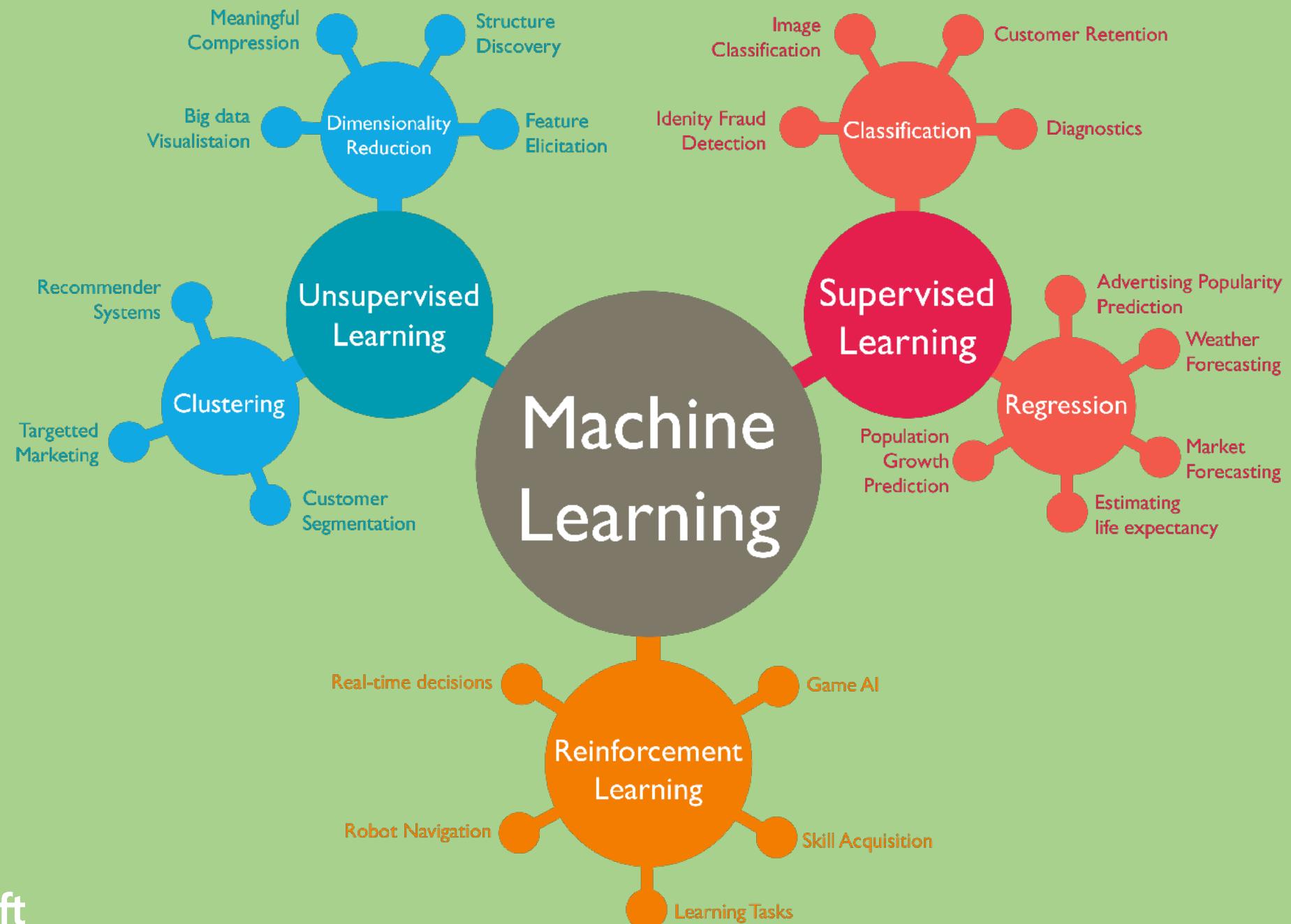
IMAGENET

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

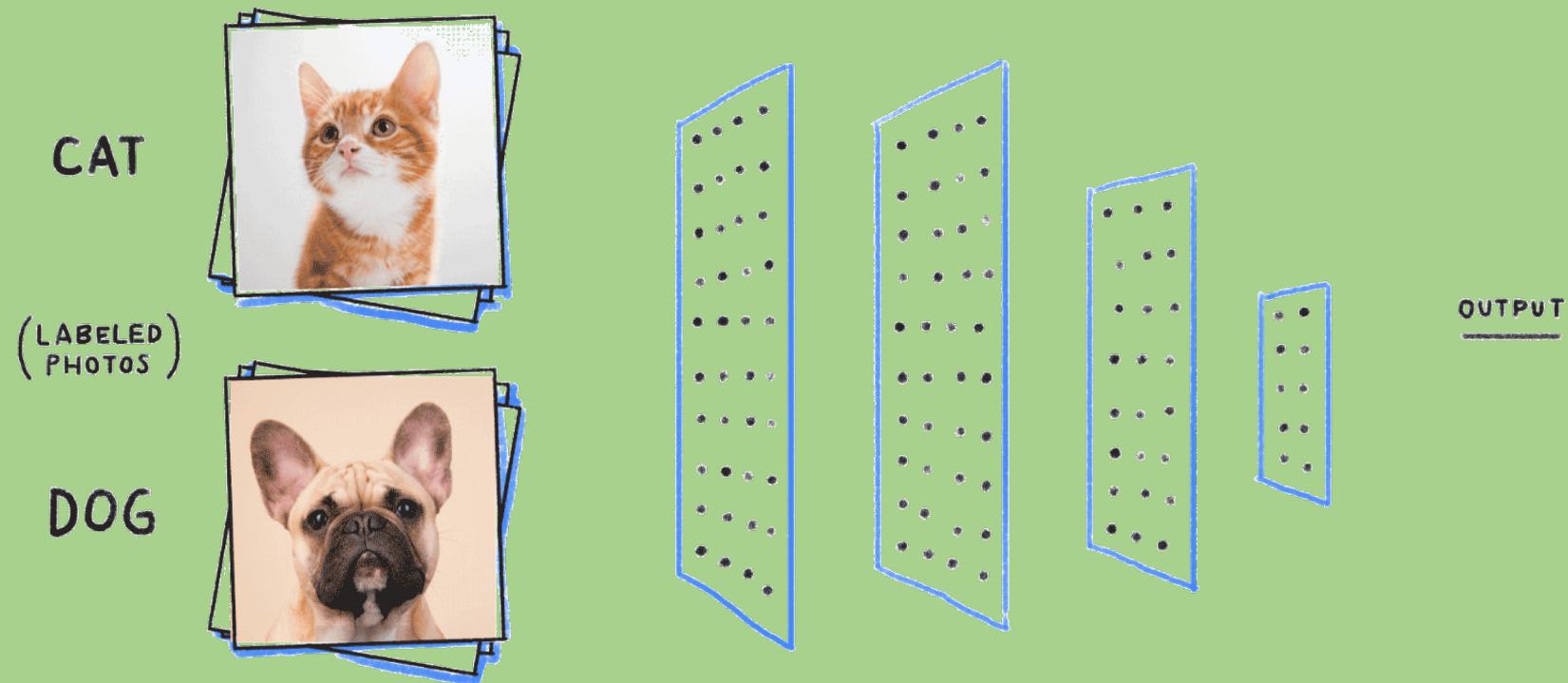
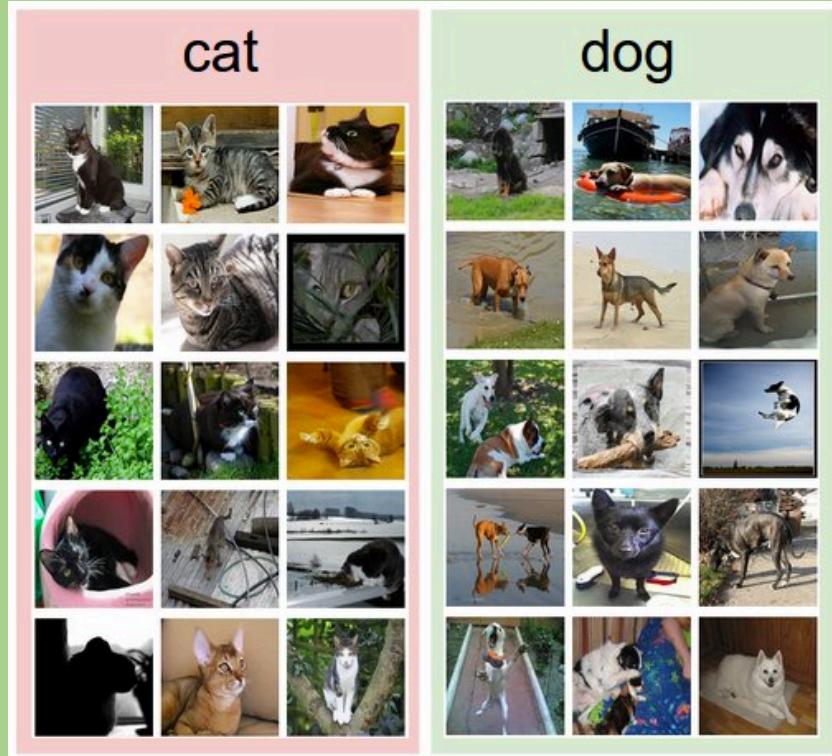




Neural Networks



Convolutional Neural Network









2014



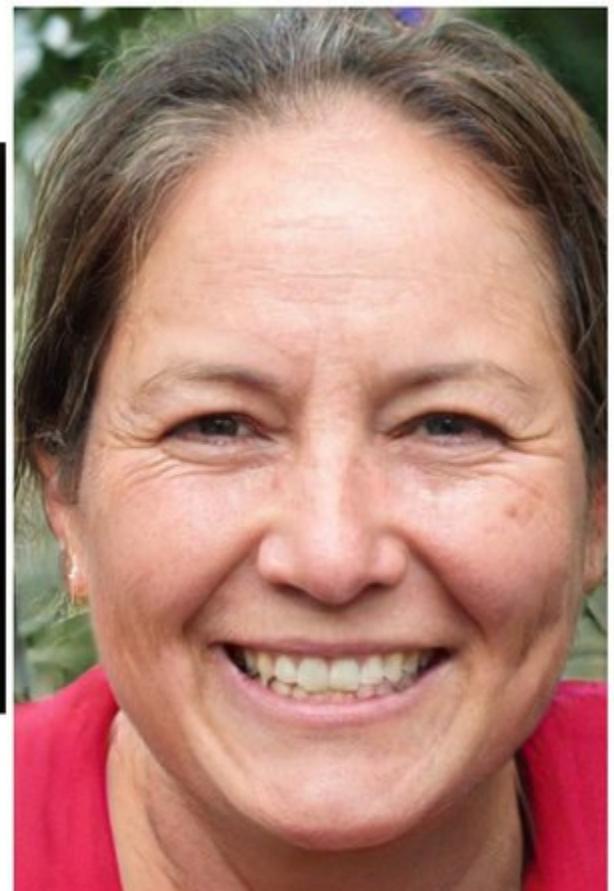
2015



2016

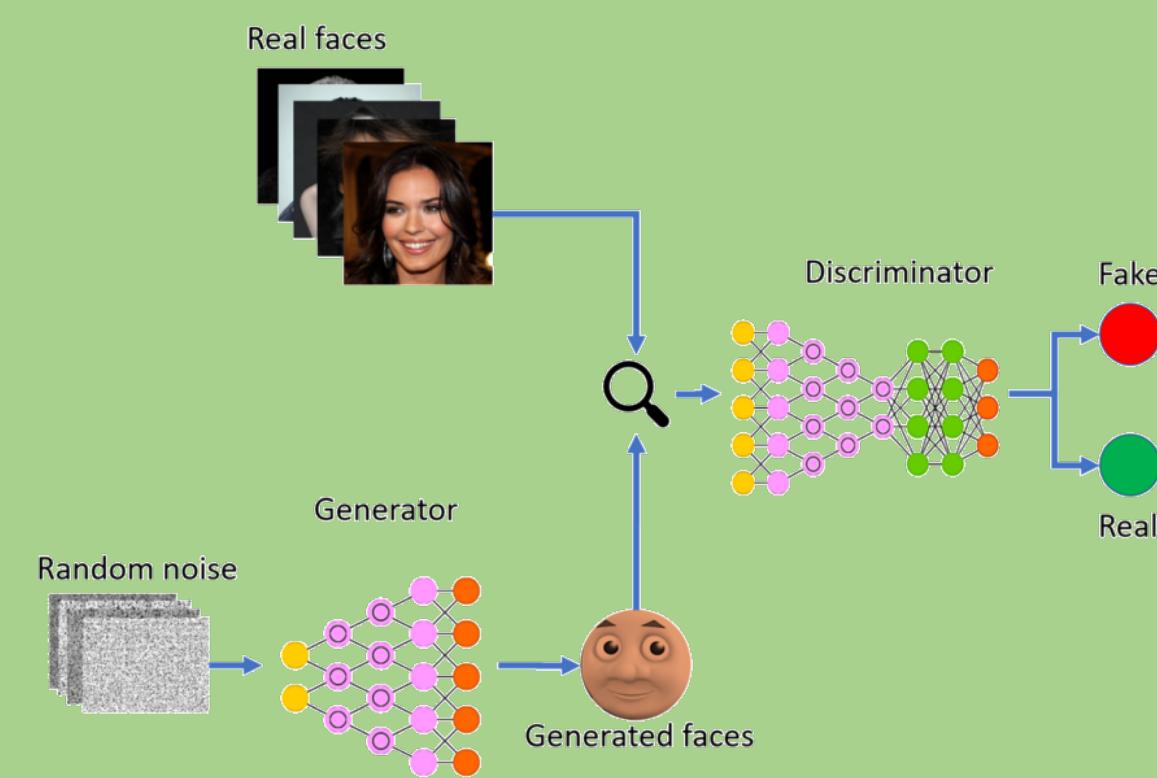


2017

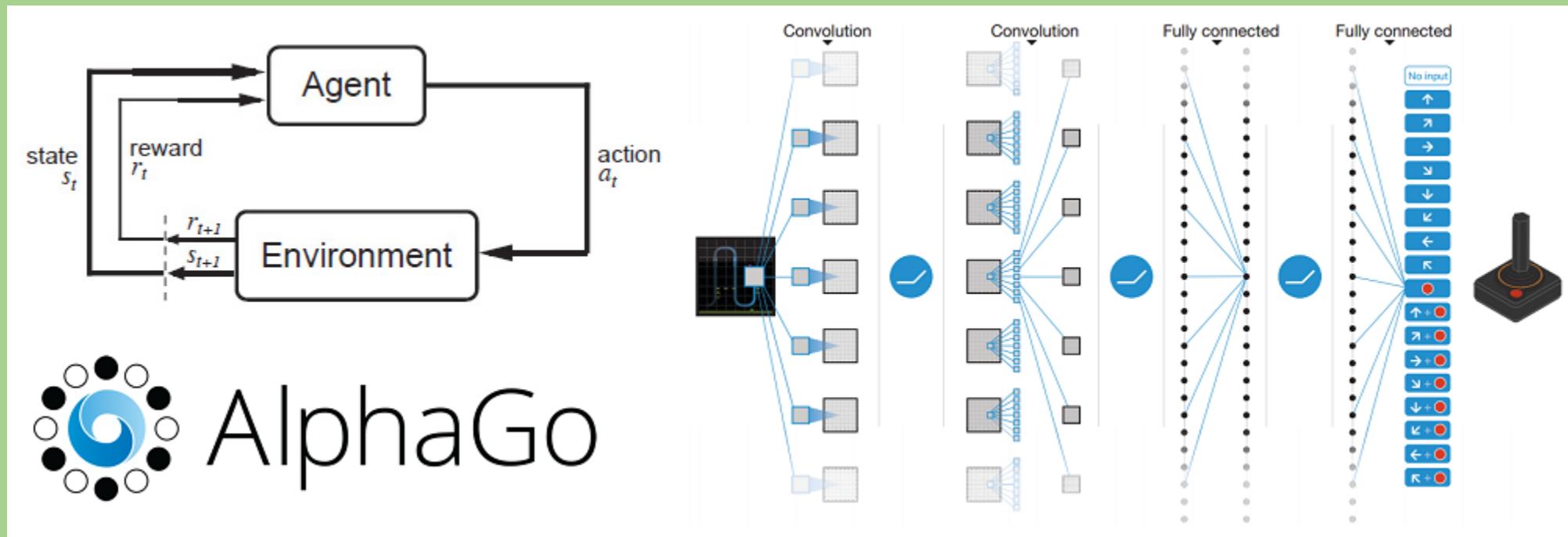


2018

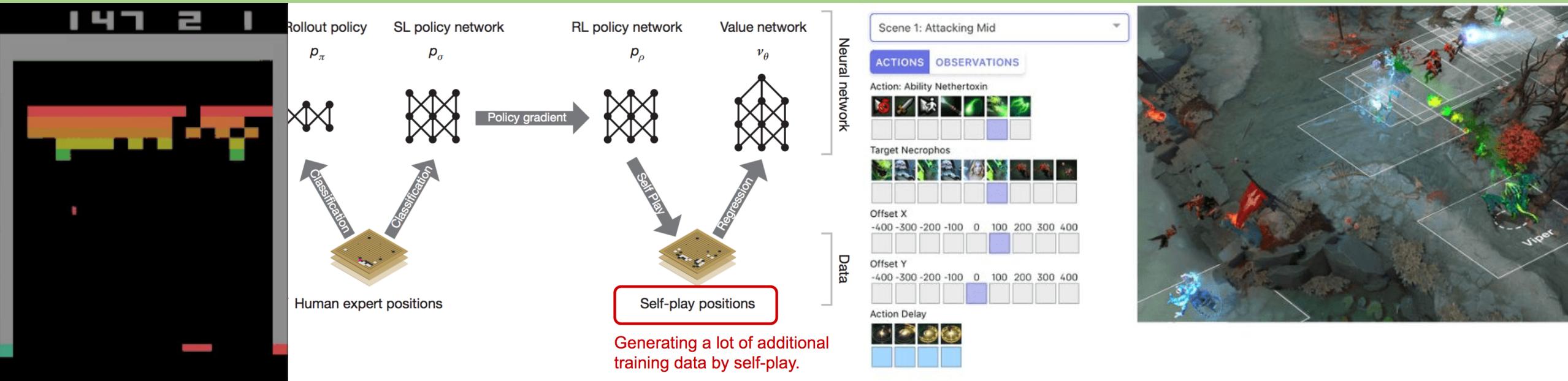
GANs (Generative Adversarial Networks)



Deep Reinforcement Learning



Deep Reinforcement Learning



Time for some Math

Fish and Chips



x 2



x 1

\$12



x 1

Fish and Chips



x 2

\$?



x 1

\$?

\$12



x 1

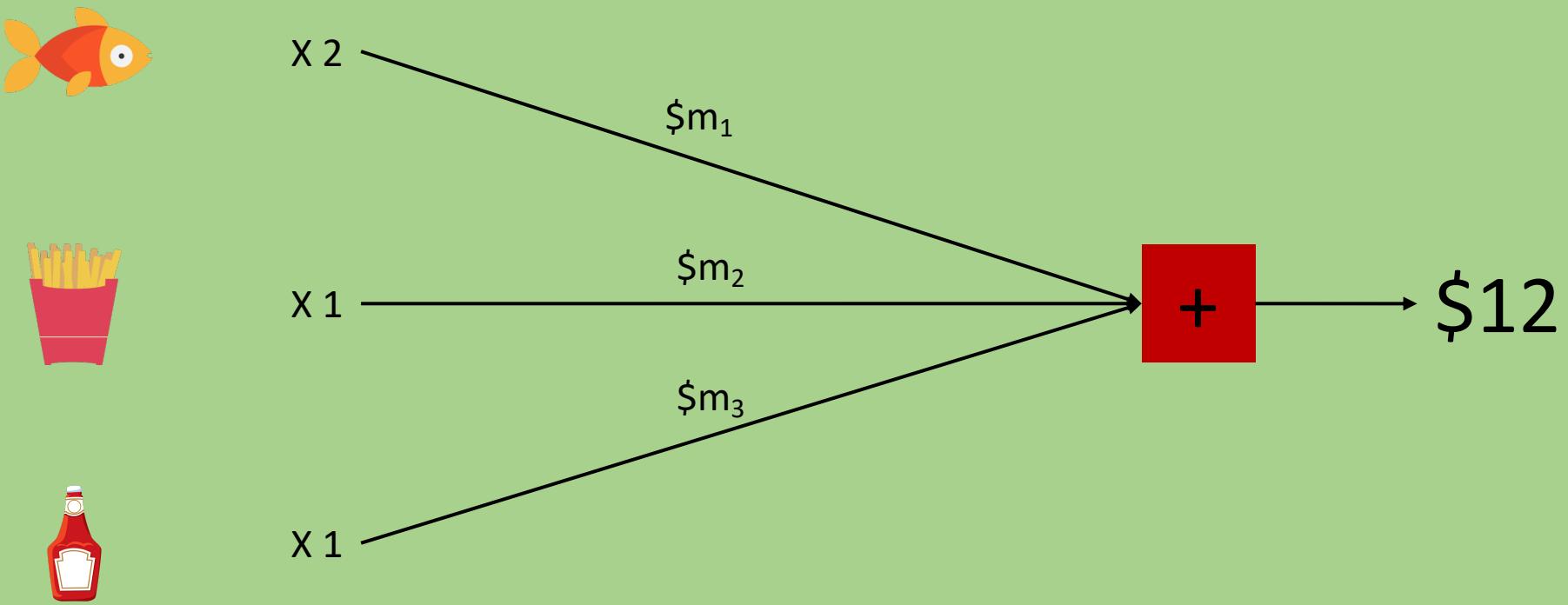
\$?

Dataset

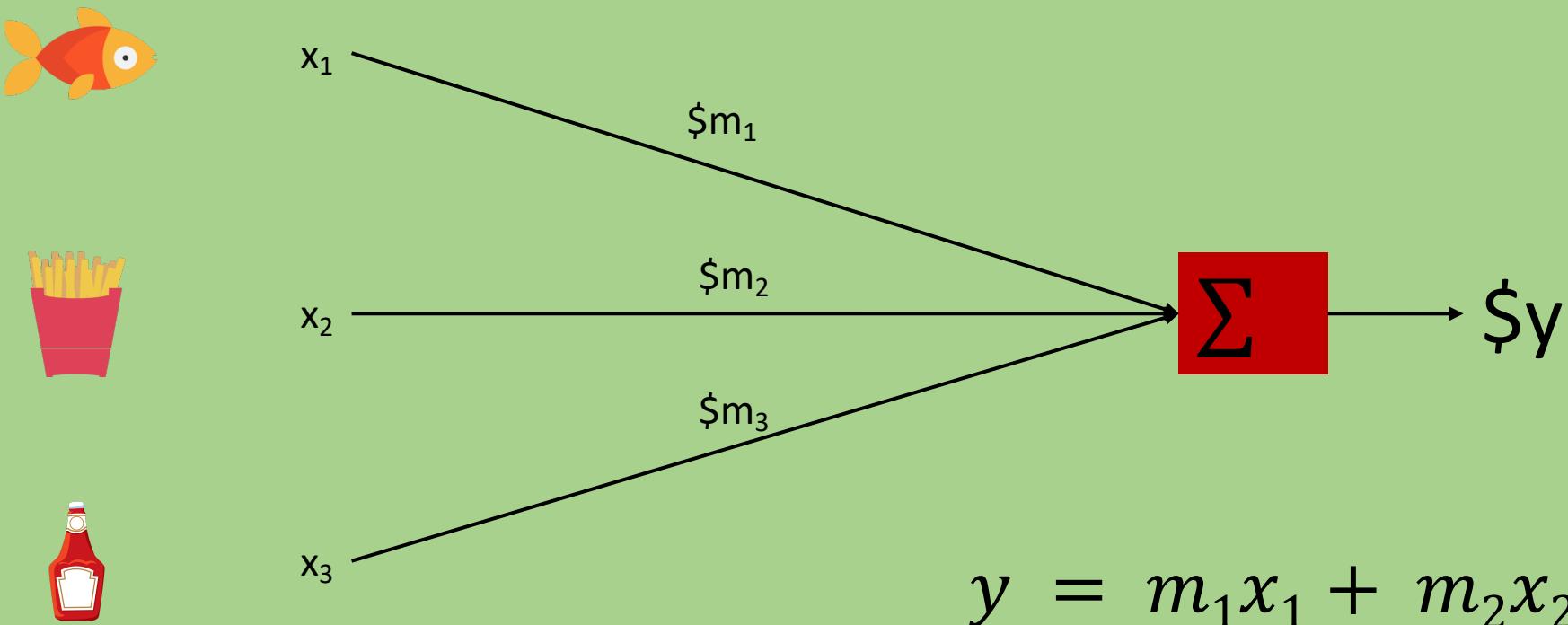


Fish	Chips	Sauce	Total
2	1	1	\$12
3	2	4	\$15
1	1	0	\$7
0	2	2	\$8
1	1	0	\$7

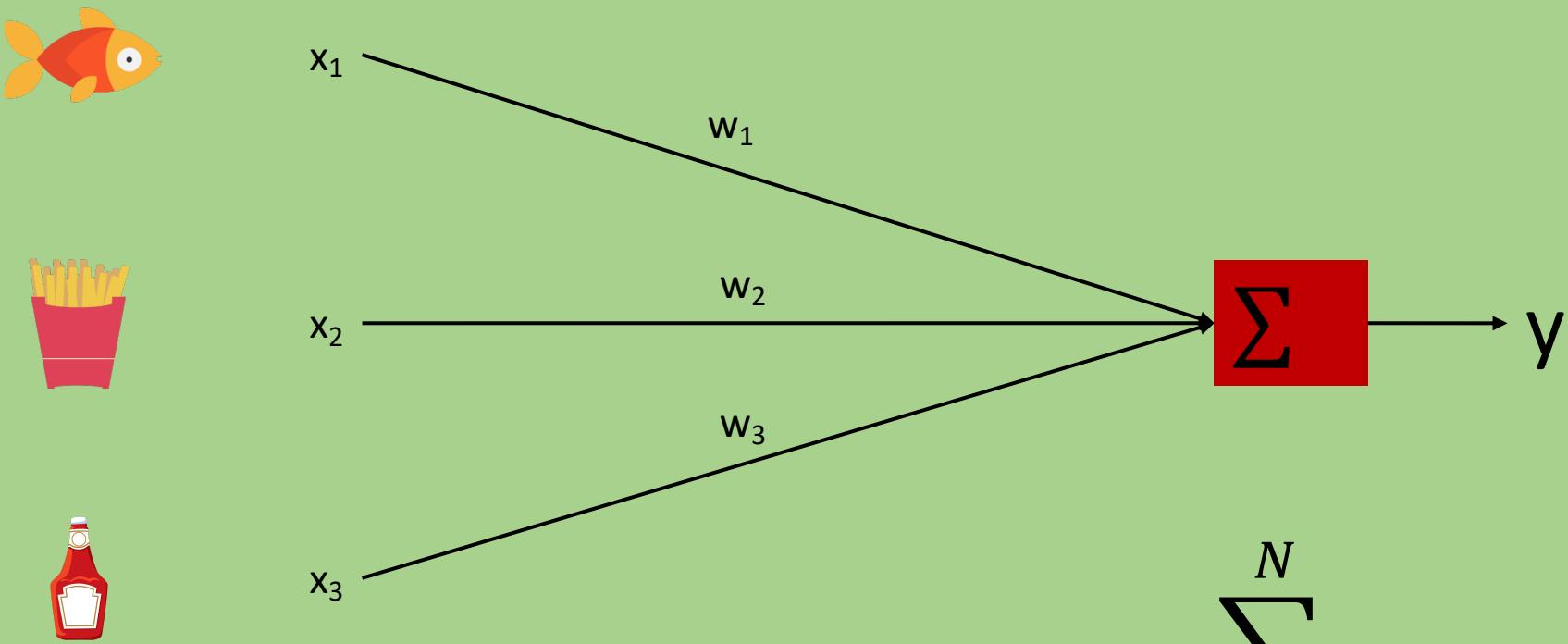
Linear Regression



Linear Regression

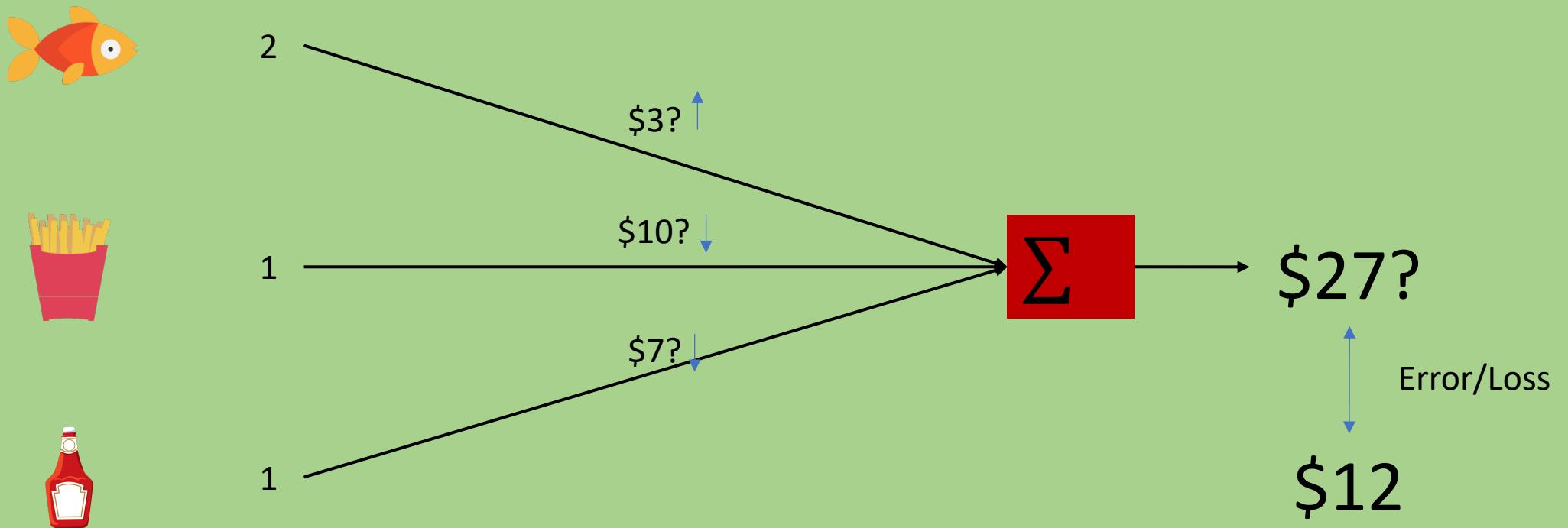


Linear Regression



$$y = \sum_{i=1}^N w_i x_i$$

Fish and Chips



Loss Functions - Regression

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}|$$

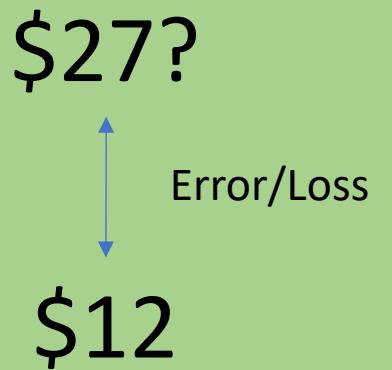
$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

Where,

\hat{y} – predicted value of y
 \bar{y} – mean value of y



Error Metrics

$$MAE = \frac{1}{n} \sum \left| y - \hat{y} \right|$$

Divide by the total number of data points

Predicted output value

Actual output value

Sum of

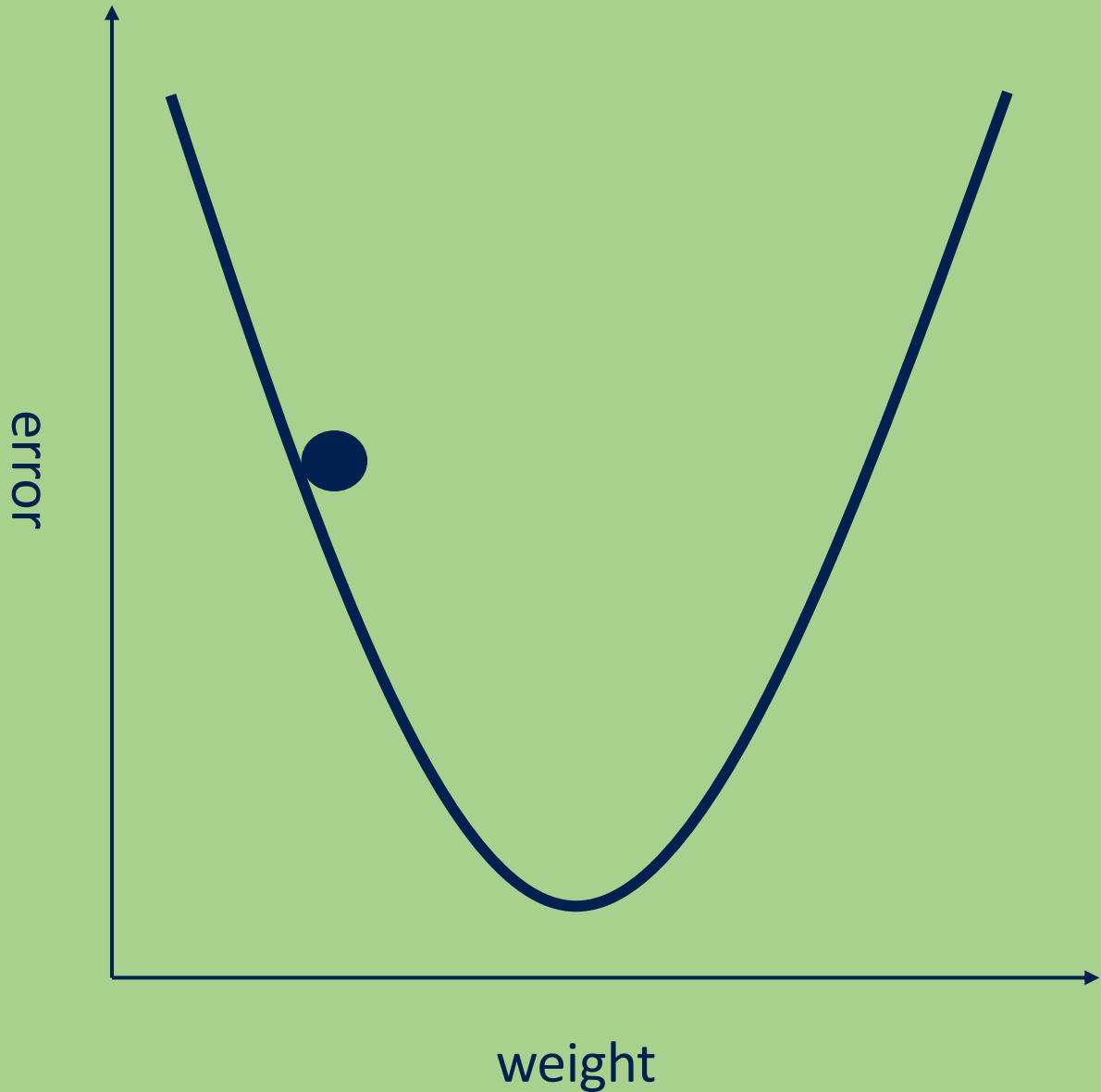
The absolute value of the residual

MAE = $\frac{1}{n} \sum \left| y - \hat{y} \right|$

\$27?
↓
Error/Loss
\$12

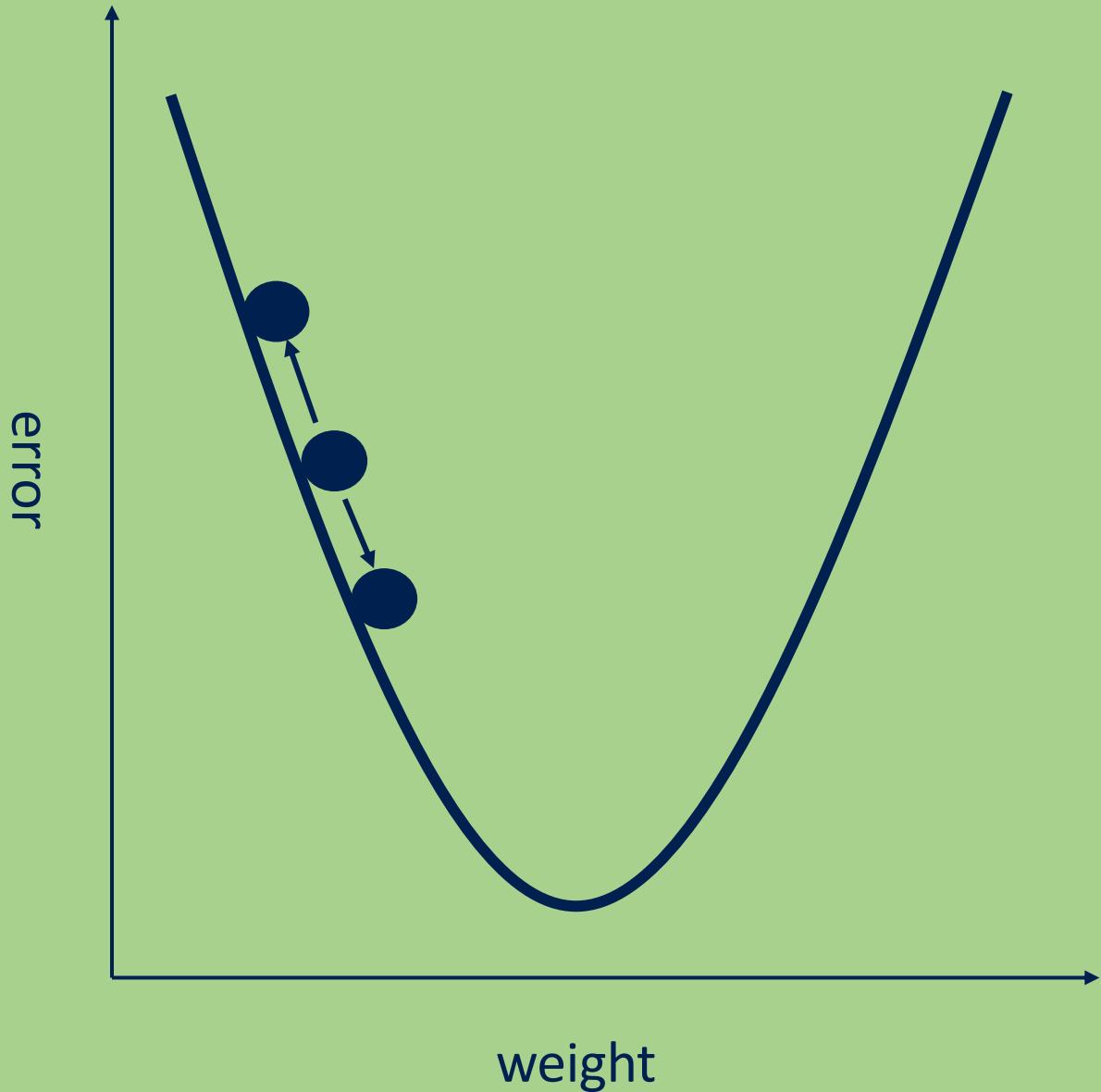
Gradient descent

For each input feature and voting weight, adjust it up and down a bit and see how the error changes.

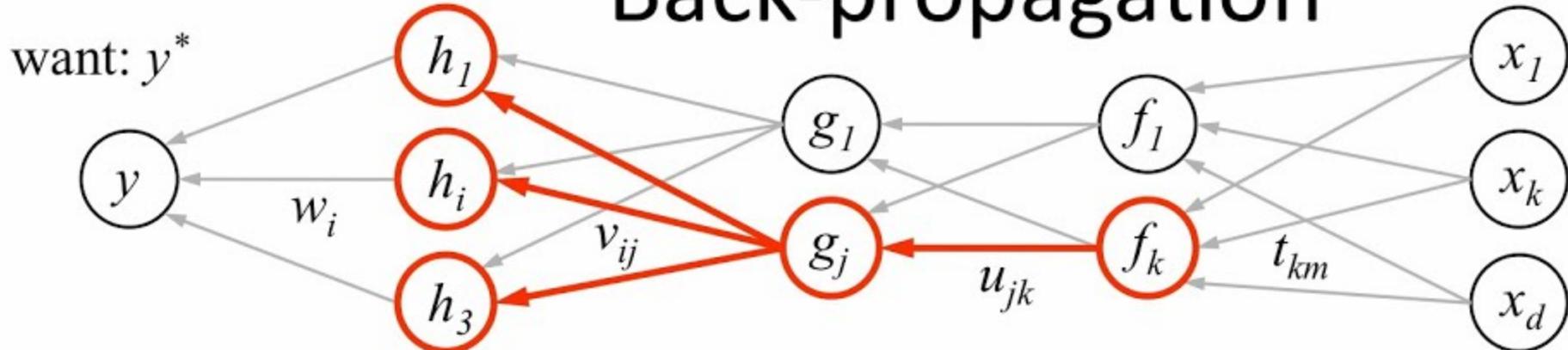


Gradient descent

For each input feature and voting weight, adjust it up and down a bit and see how the error changes.



Back-propagation



1. receive new observation $\mathbf{x} = [x_1 \dots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer $1 \dots L$
compute g_j based on units f_k from previous layer: $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction y and error $(y - y^*)$
4. **back-propagate error:** for each unit g_j in each layer $L \dots 1$

(a) compute error on g_j

$$\frac{\partial E}{\partial g_j} = \sum_i \underbrace{\sigma'(h_i)}_{\text{should } g_j \text{ be higher or lower?}} \underbrace{v_{ij}}_{\text{how } h_i \text{ will change as } g_j \text{ changes}} \underbrace{\frac{\partial E}{\partial h_i}}_{\text{was } h_i \text{ too high or too low?}}$$

(b) for each u_{jk} that affects g_j

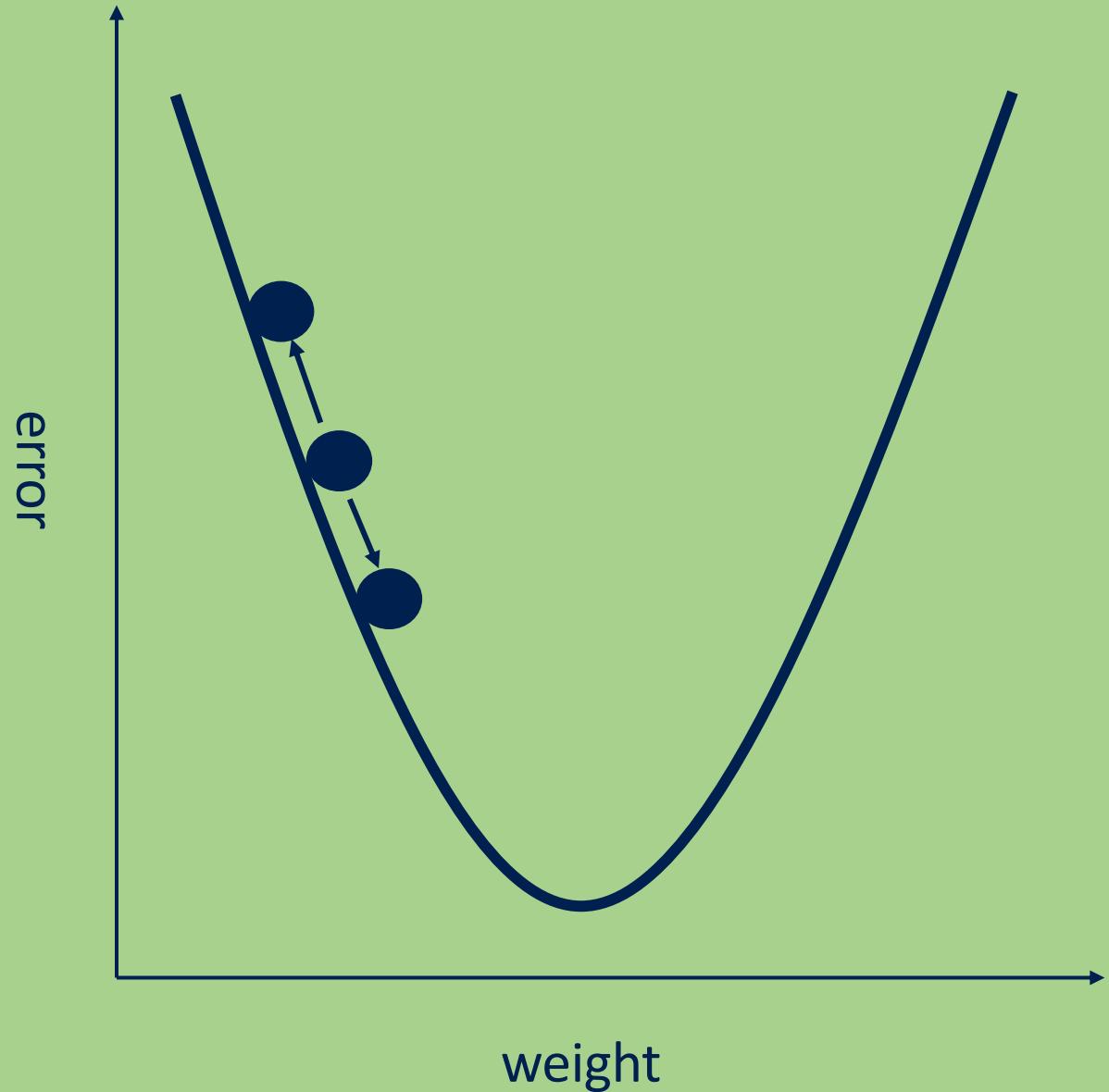
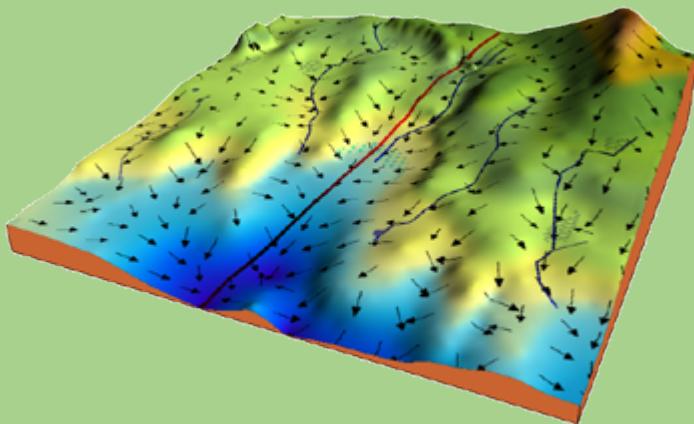
(i) compute error on u_{jk}

$$\frac{\partial E}{\partial u_{jk}} = \underbrace{\frac{\partial E}{\partial g_j}}_{\text{do we want } g_j \text{ to be higher/lower?}} \underbrace{\sigma'(g_j) f_k}_{\text{how } g_j \text{ will change if } u_{jk} \text{ is higher/lower}}$$

(ii) update the weight

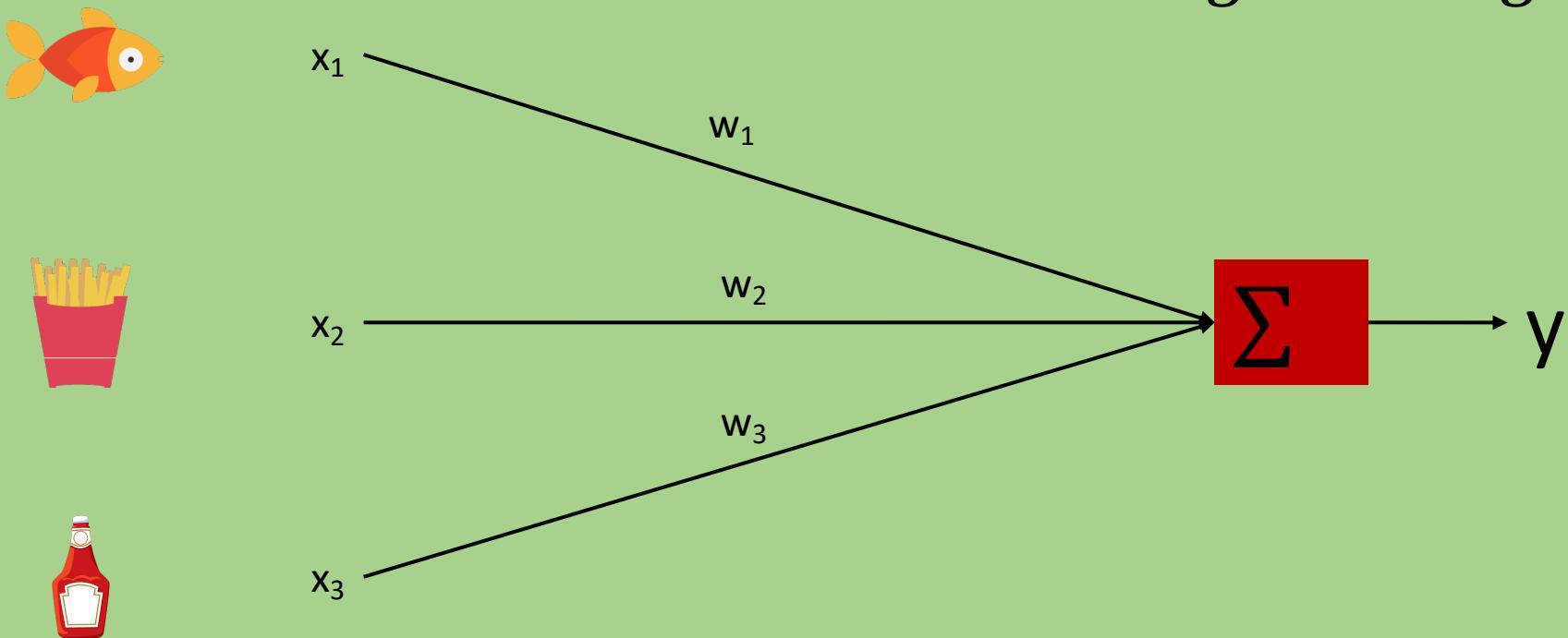
$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$$

Gradient descent



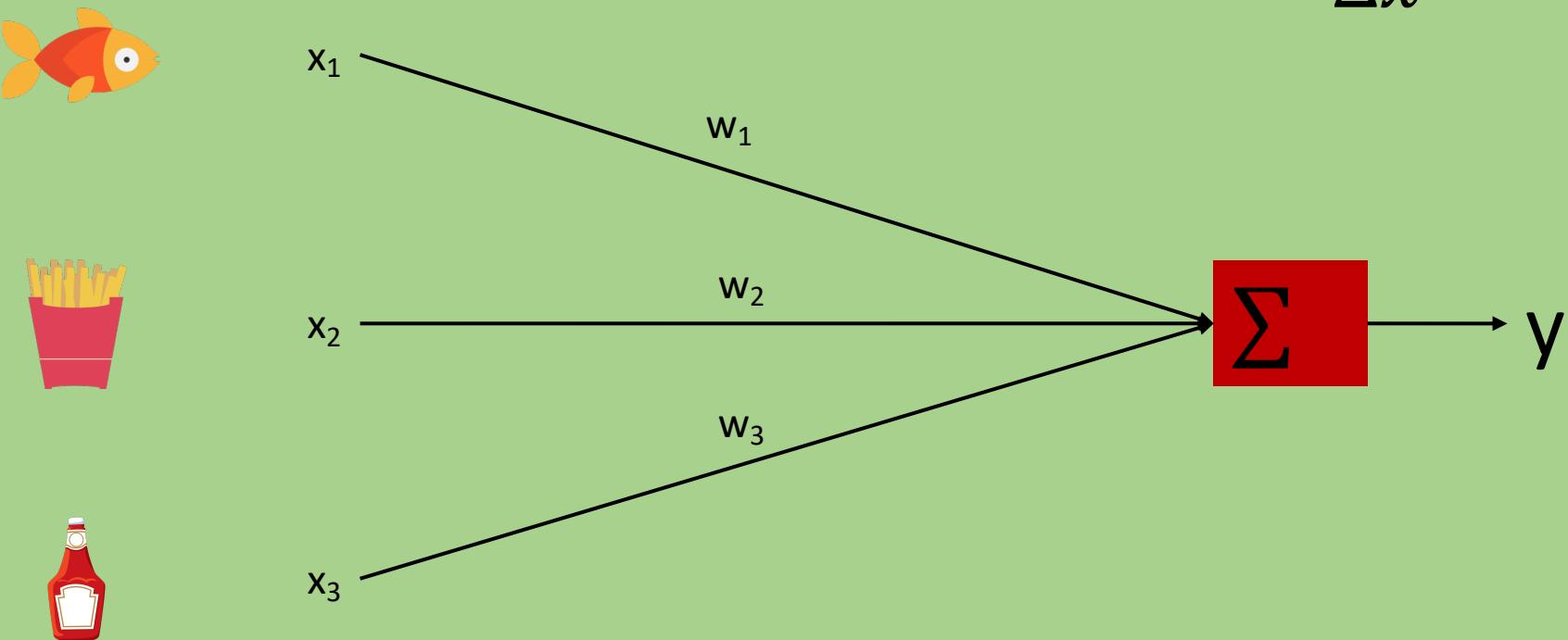
Weight Update

$$w^{new} = w^{old} + learning\ rate * gradient$$

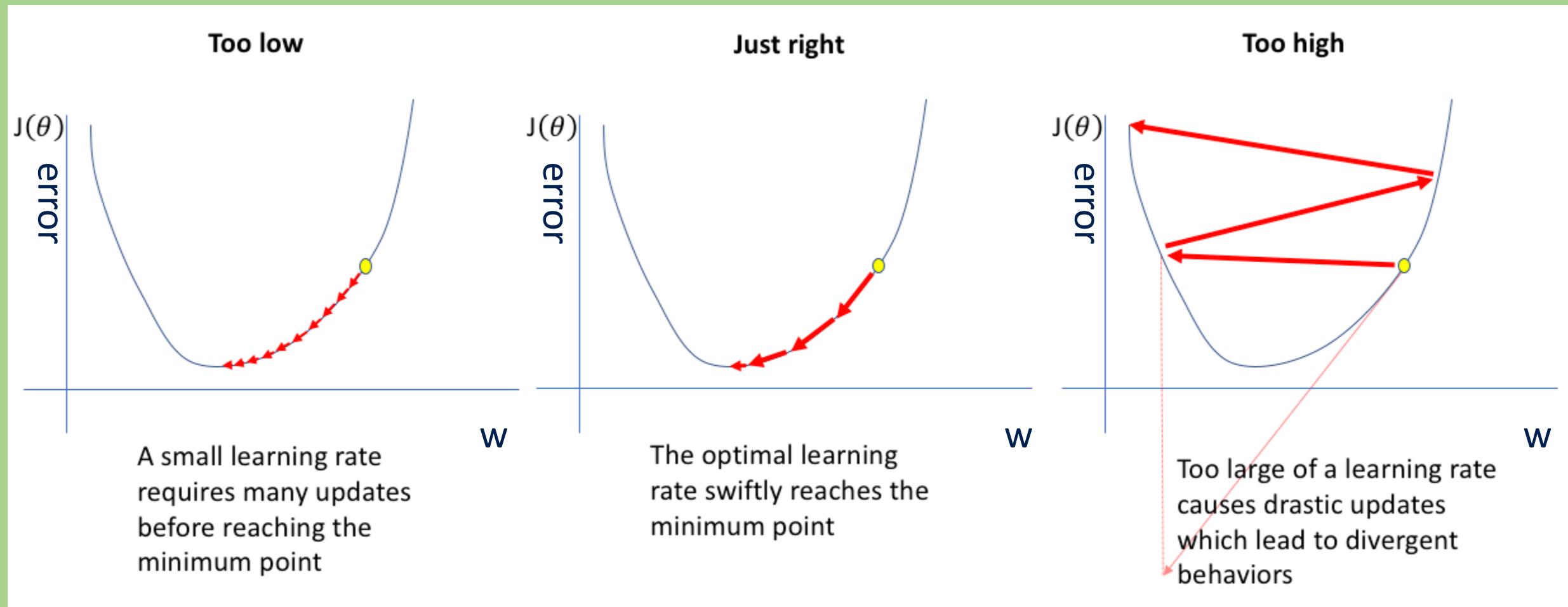


Weight Update

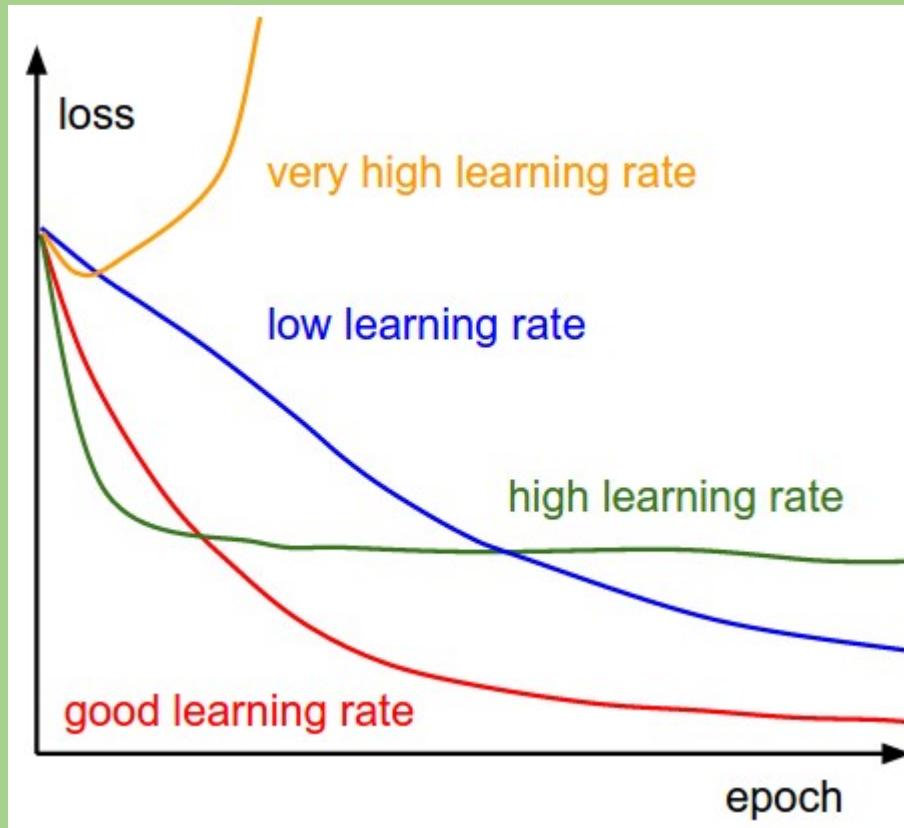
$$w^{new} = w^{old} + \alpha \frac{\Delta y}{\Delta x}$$



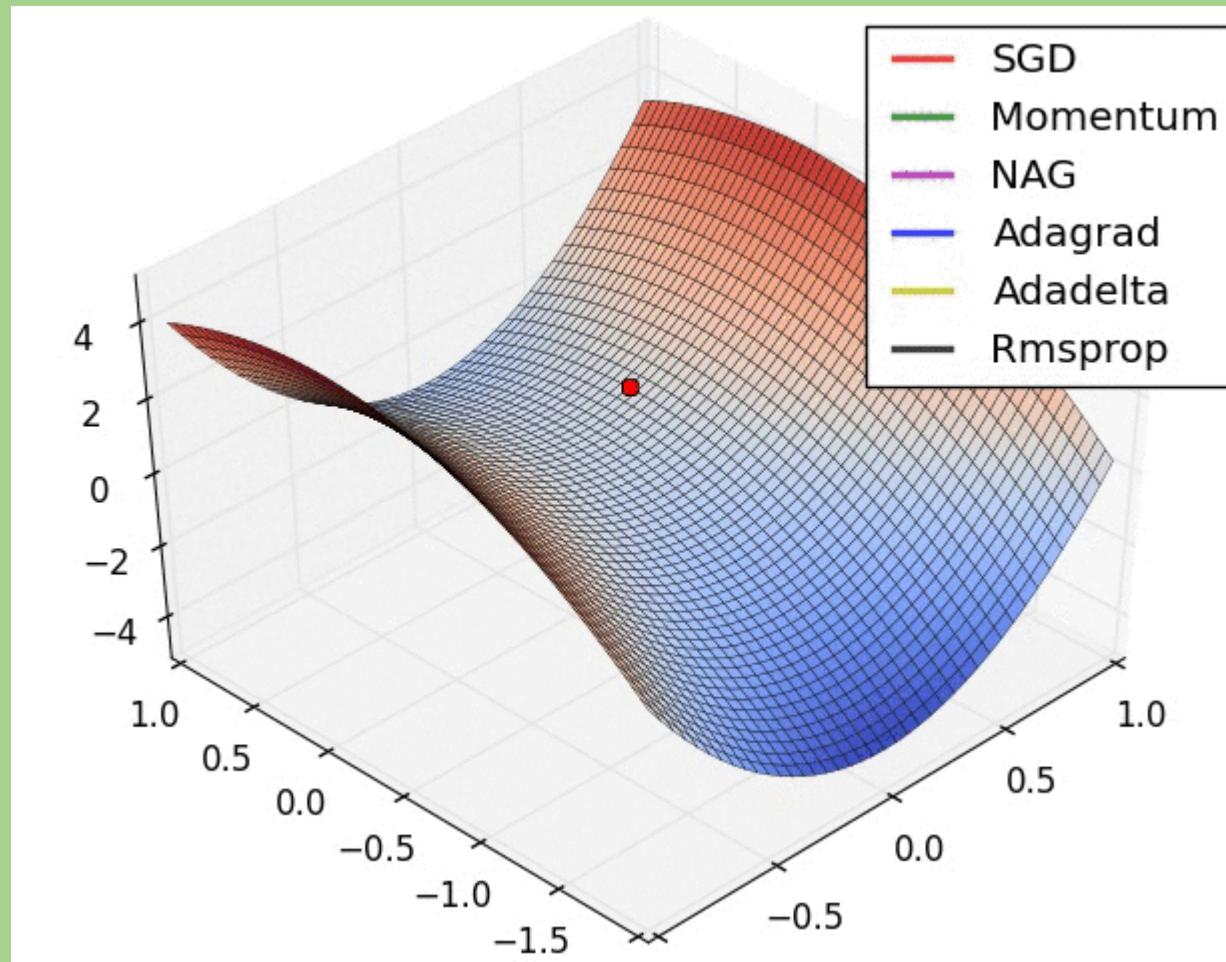
Learning Rate



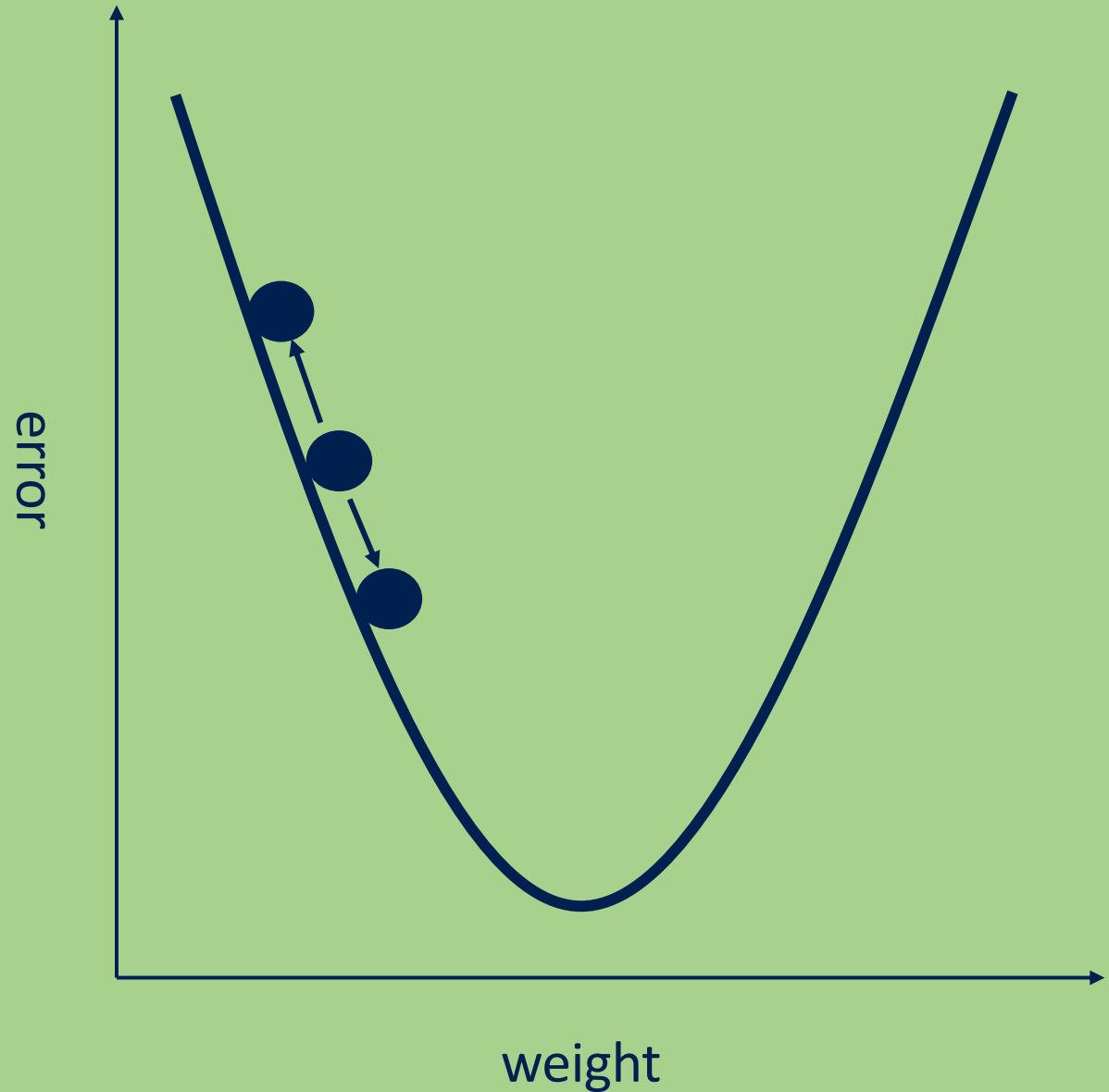
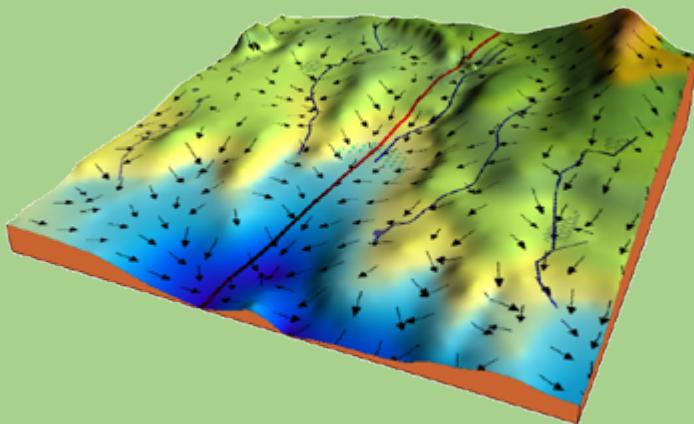
Learning Rate



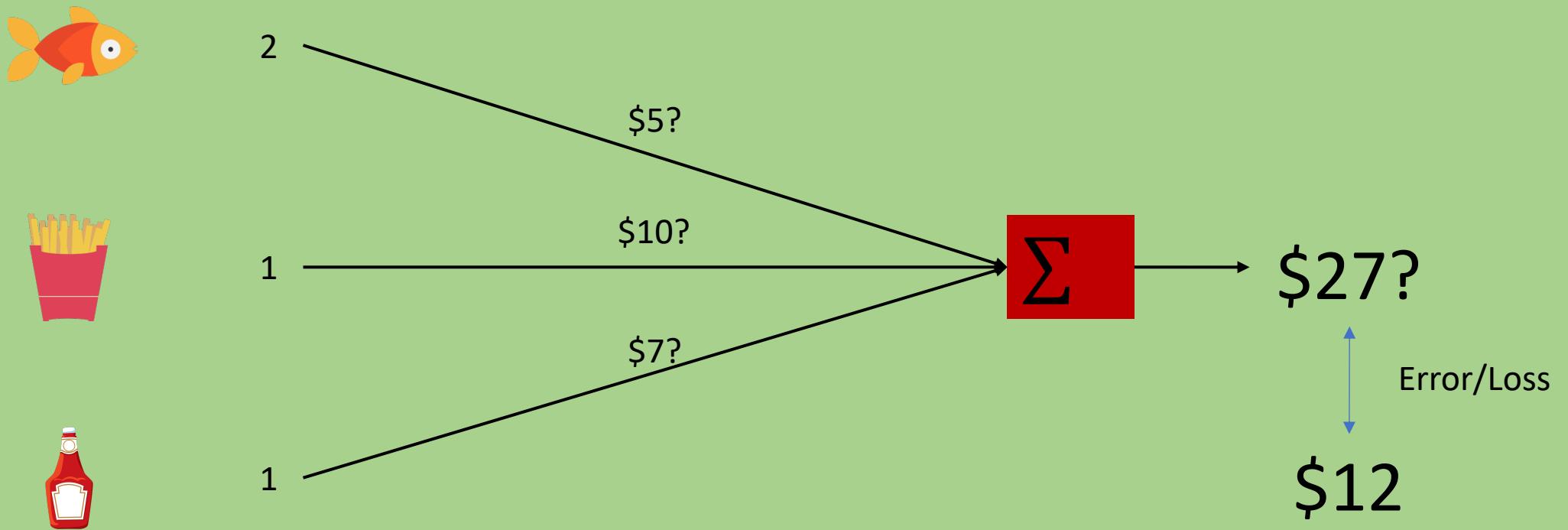
Different Optimizers



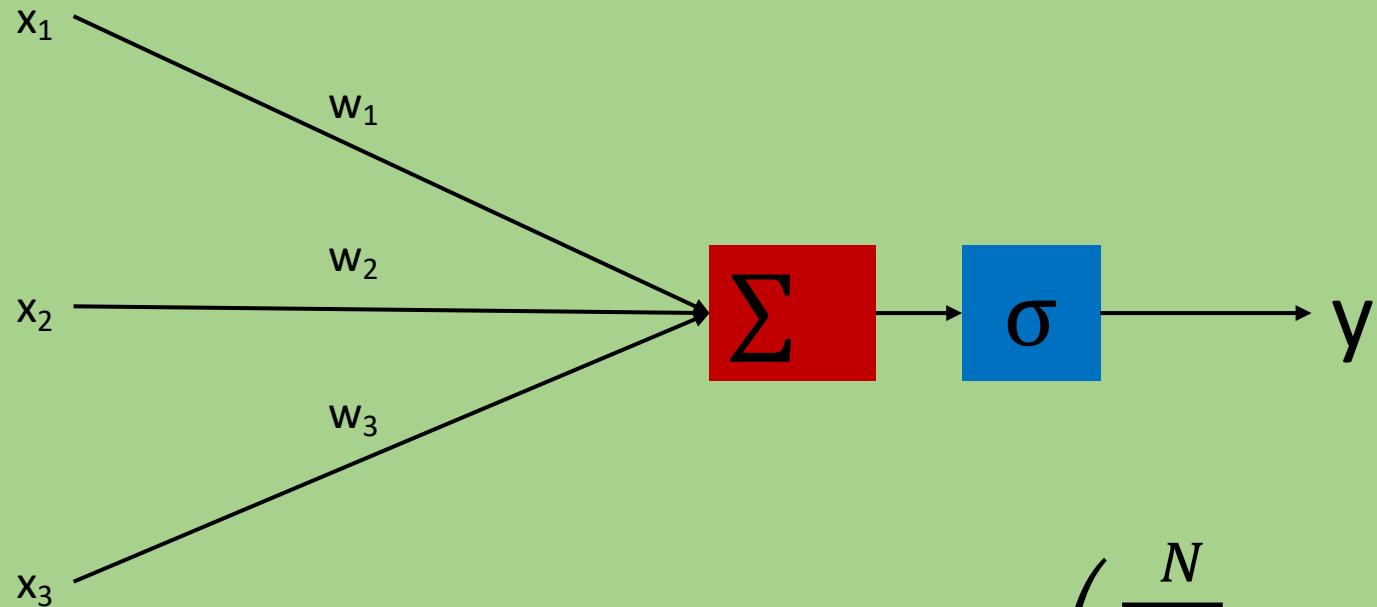
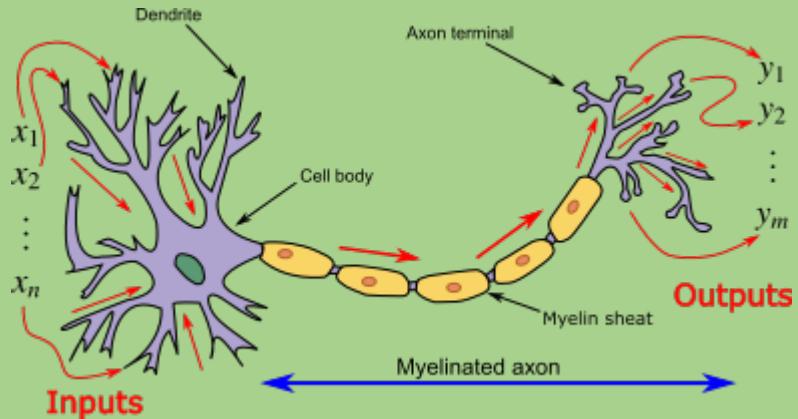
Gradient descent



Fish and Chips

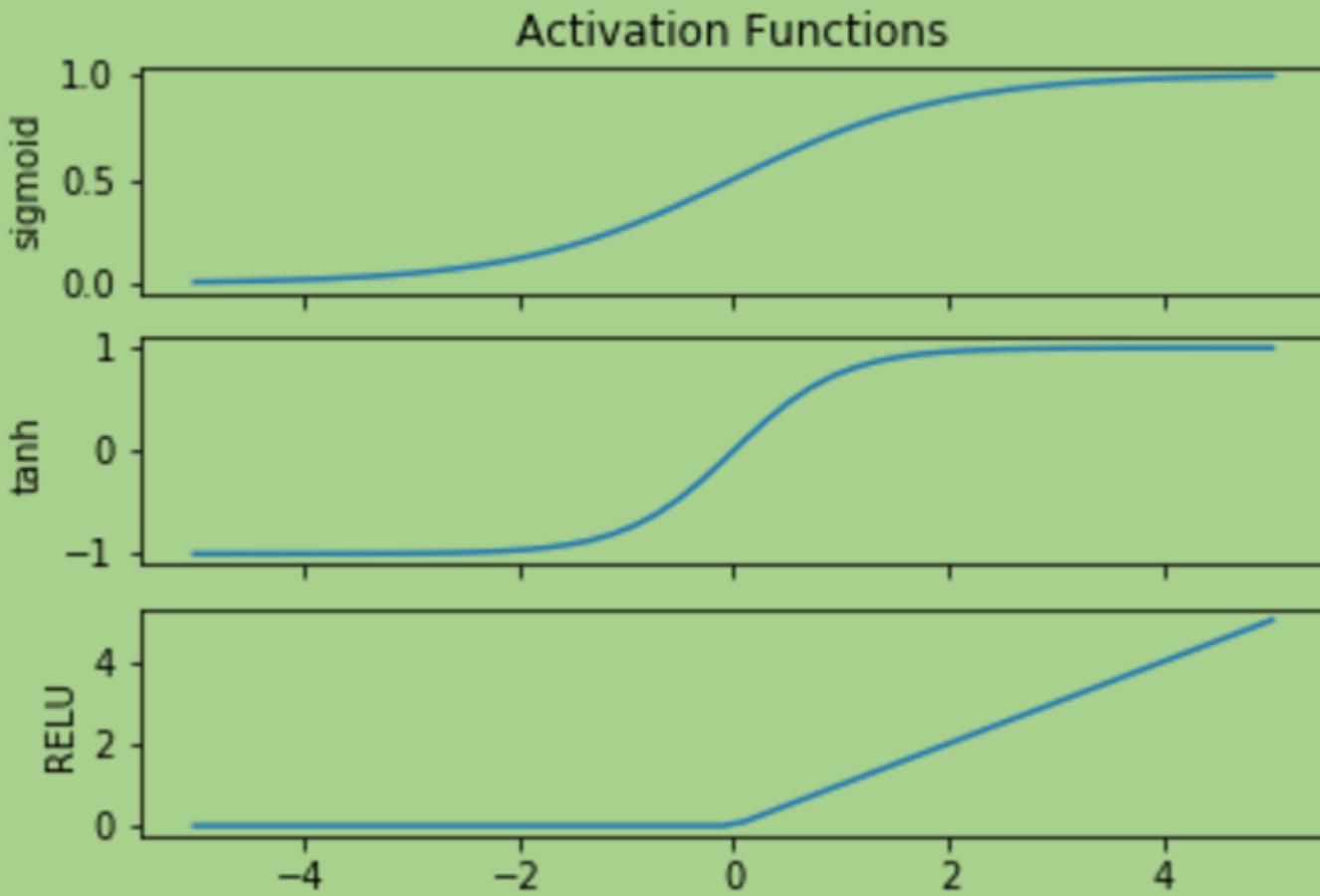


Neuron



$$y = \sigma \left(\sum_{i=1}^N w_i x_i \right)$$

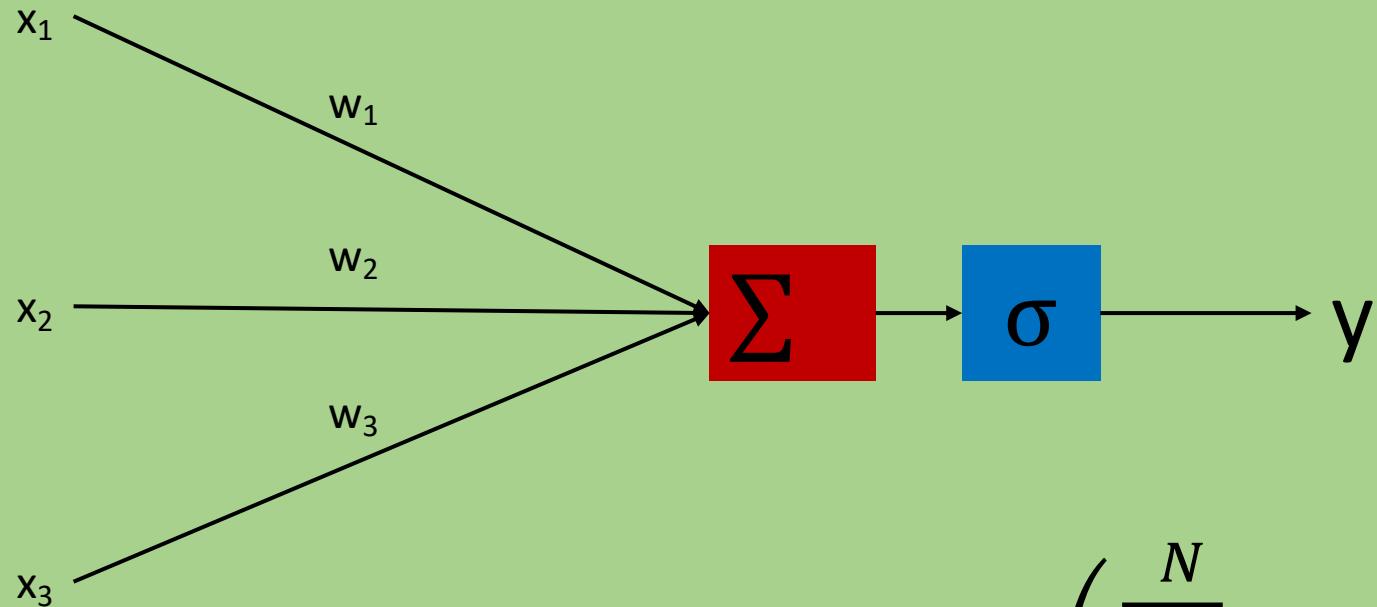
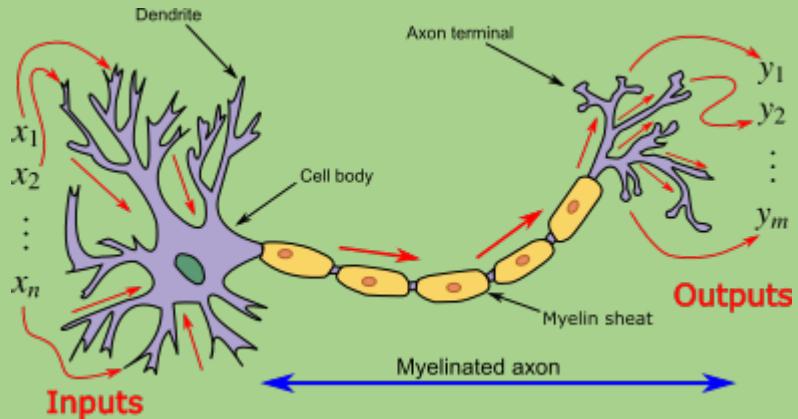
σ



$x \leftarrow \sum_{i=1}^N m_i x_i$

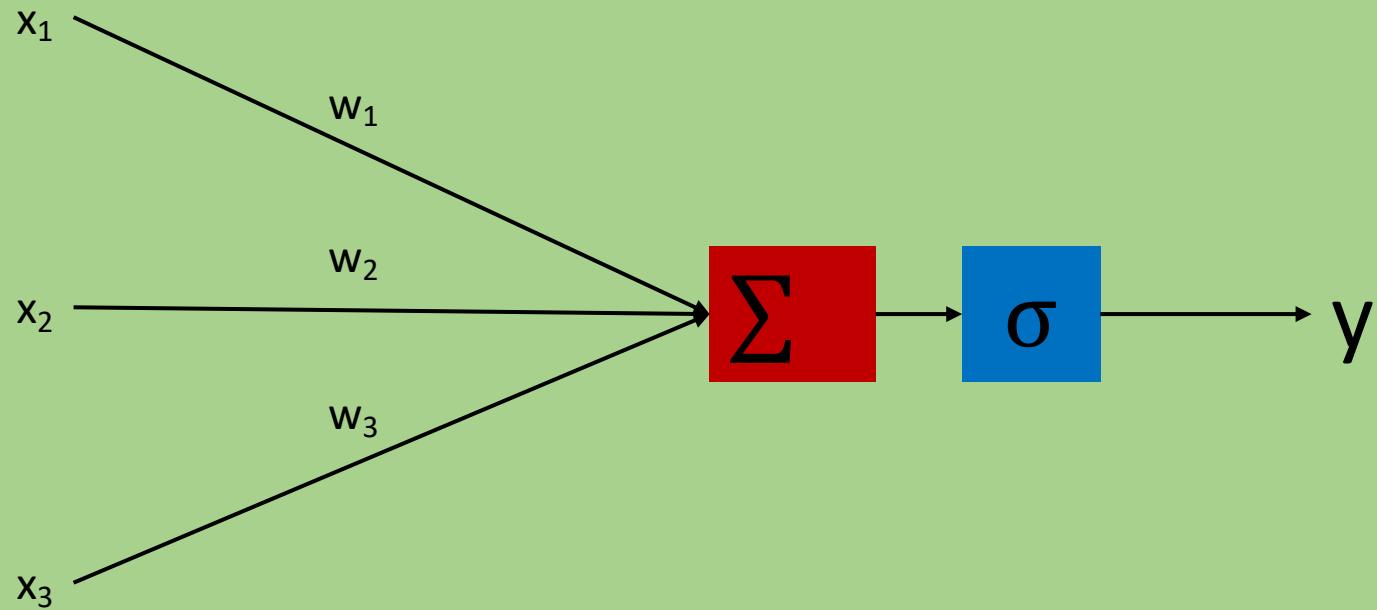
$$y = \sigma\left(\sum_{i=1}^N m_i x_i\right)$$

Neuron

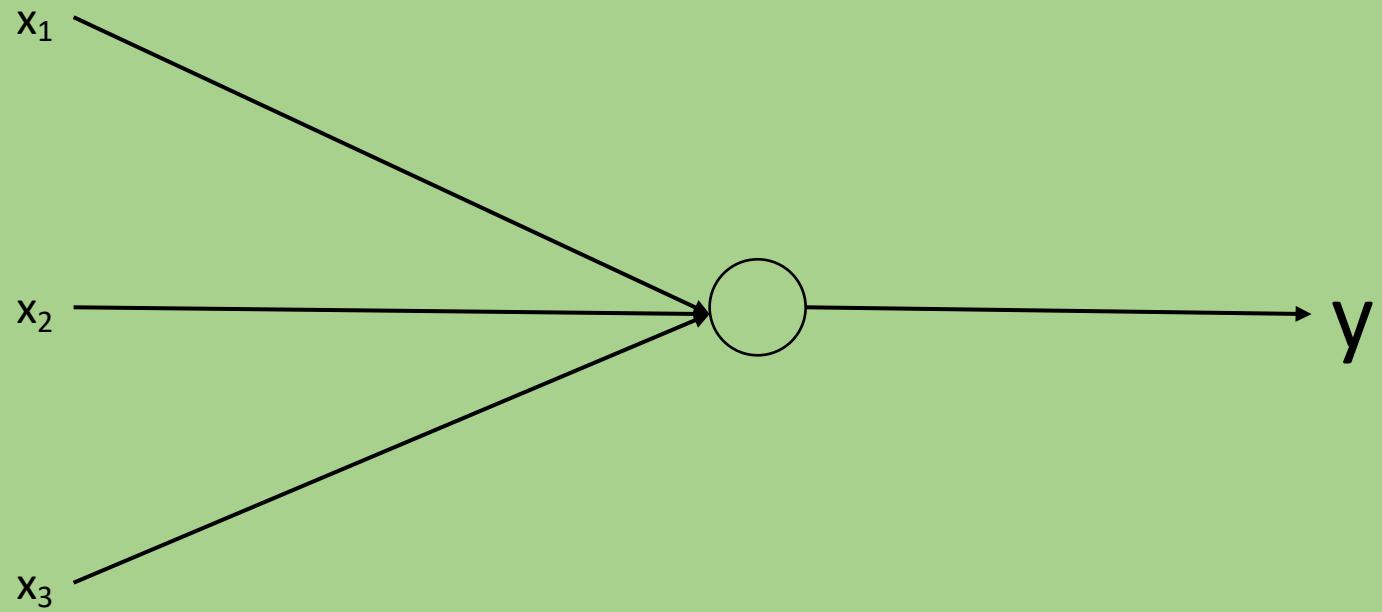


$$y = \sigma \left(\sum_{i=1}^N w_i x_i \right)$$

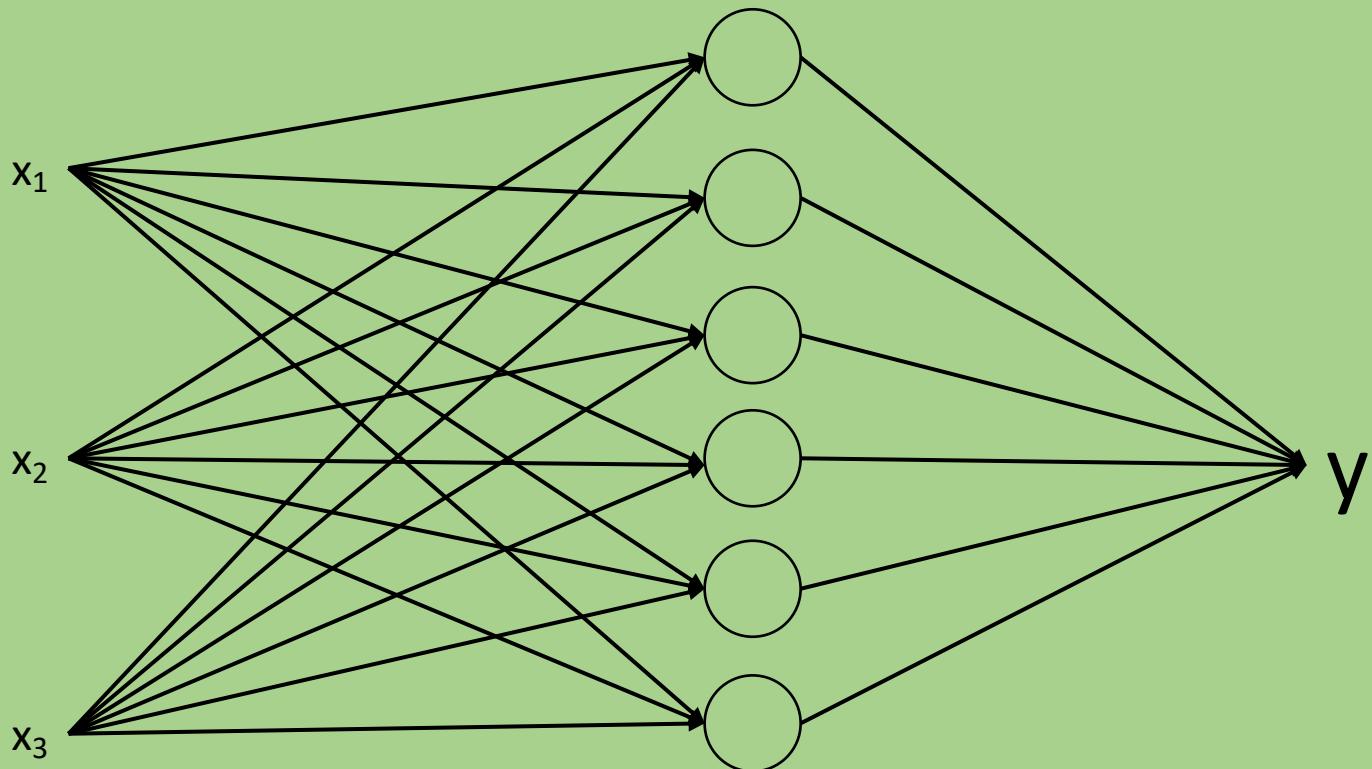
Neuron

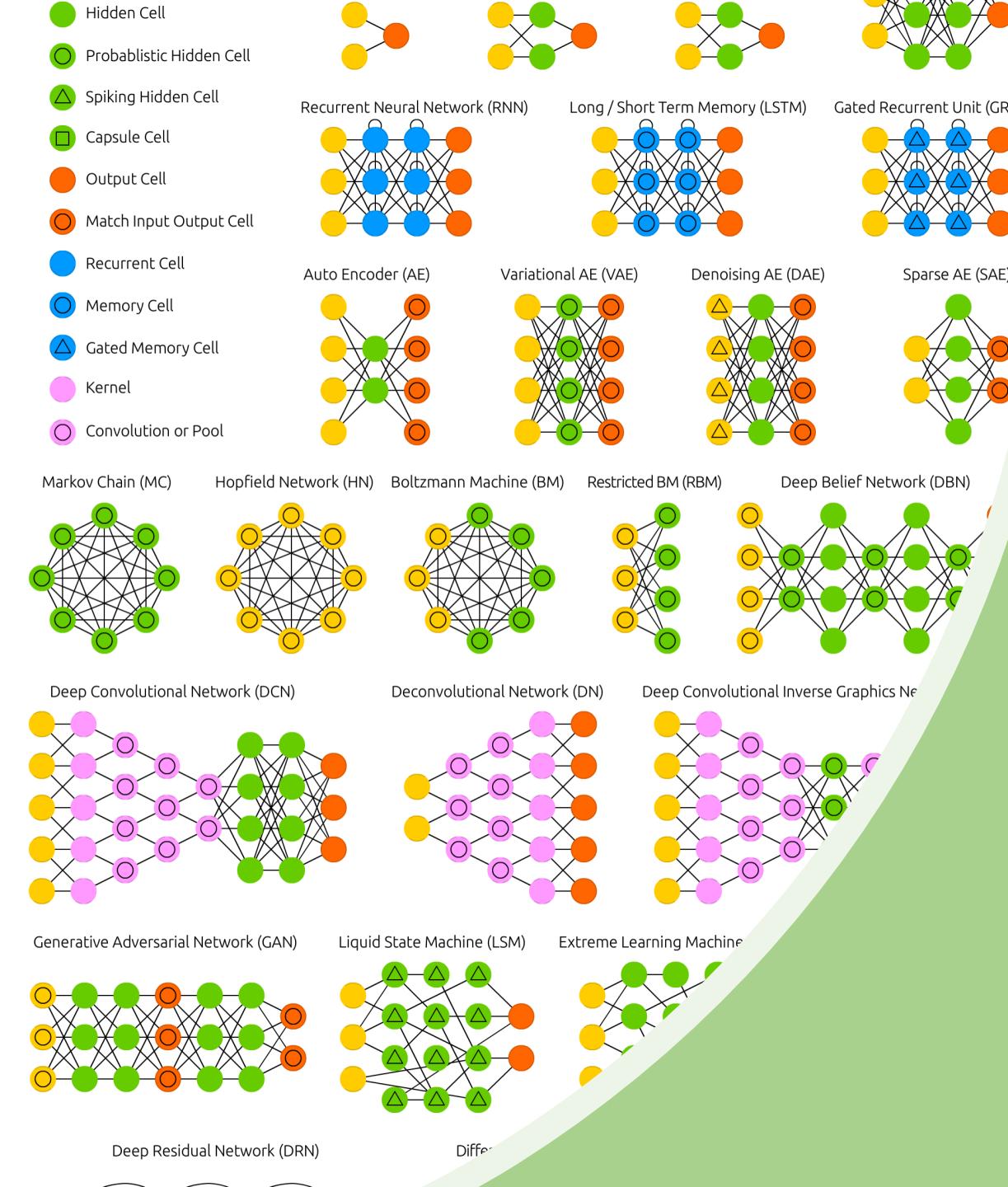


Neuron



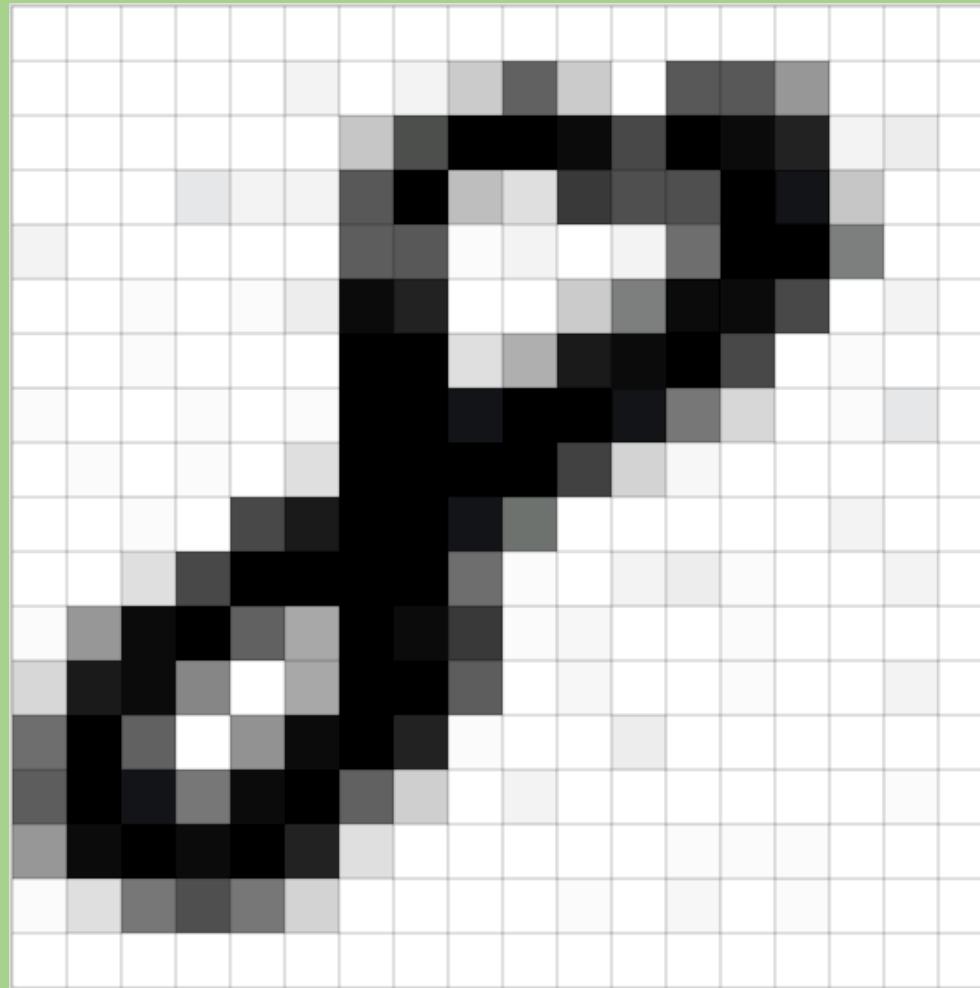
Neural Network



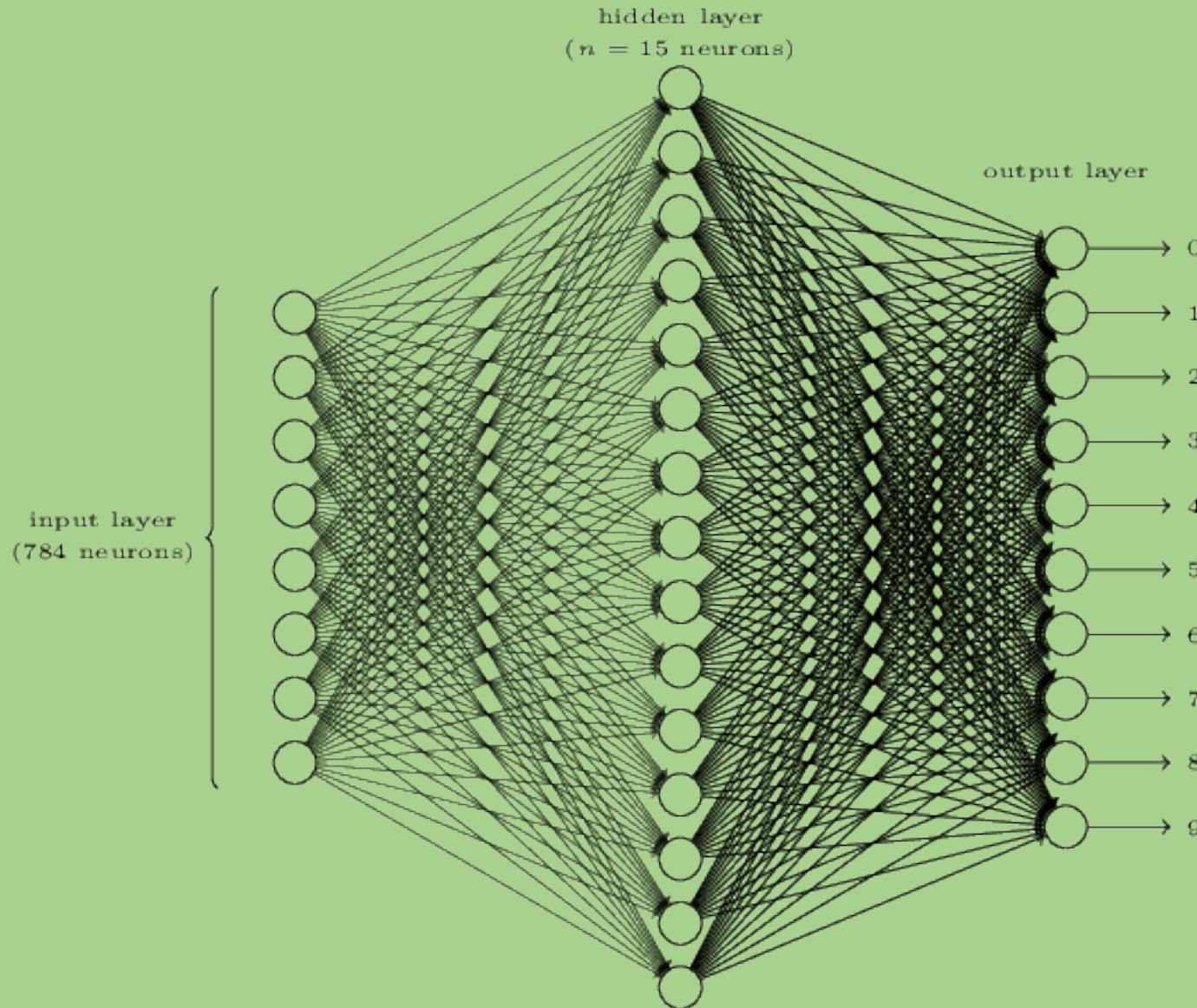


Neural Networks

Can only deal with numbers

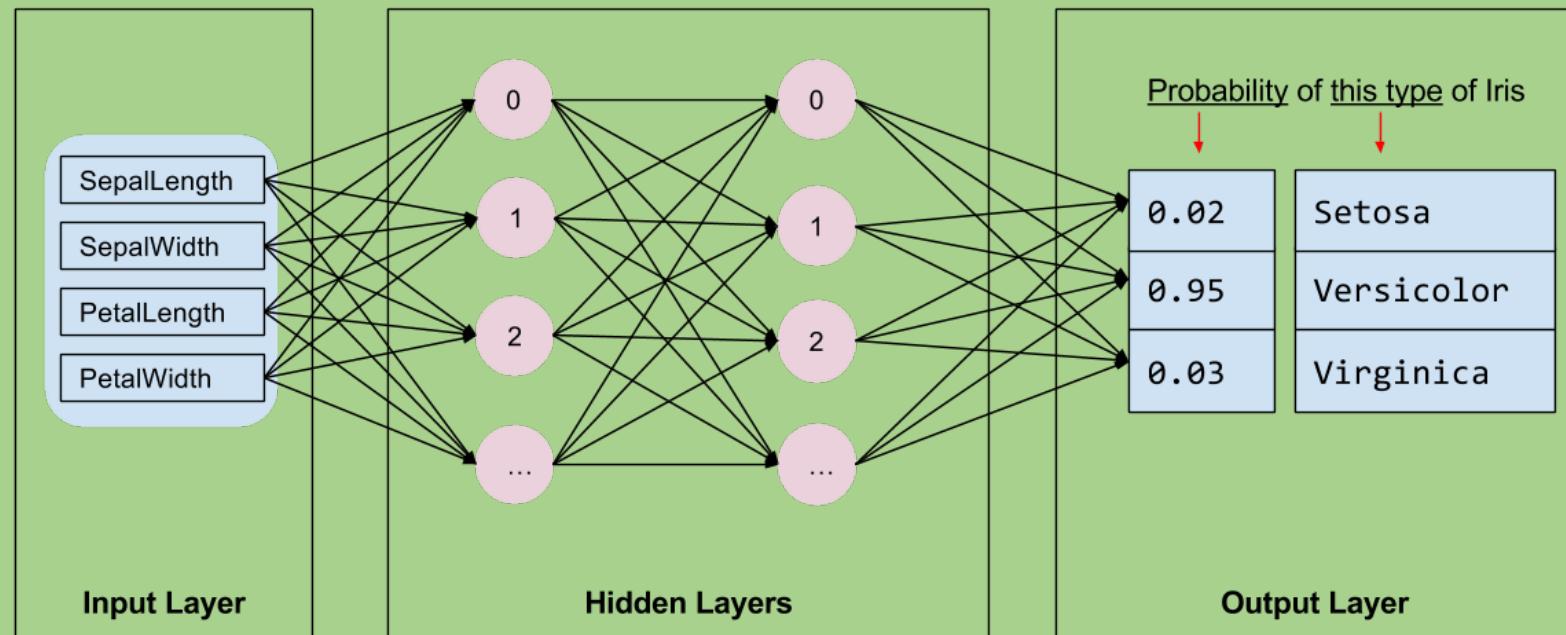


Let's build a Neural Network



Softmax

$$\text{Softmax}(z_j) = \frac{\sum e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$



Cross entropy

<https://www.youtube.com/watch?v=ErfnhcEV1O8>

Training testing and validation

1. Training data

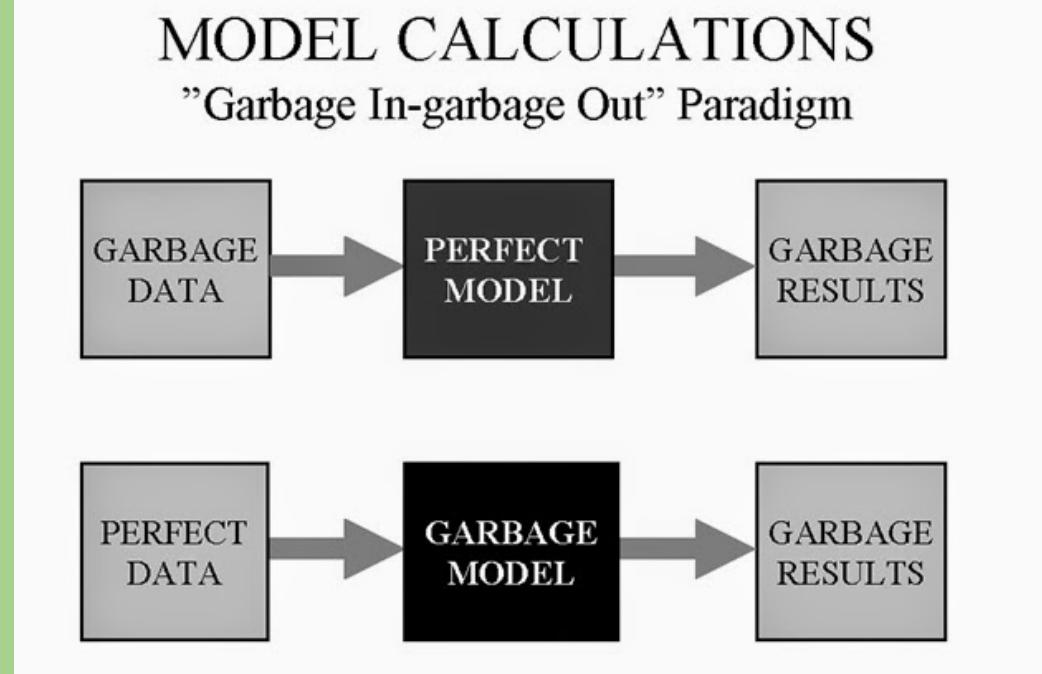
The data your algorithm will ‘learn’ from.
garbage in = garbage out.

2. Validation data

Data used to tune hyperparameters.
Results will be used to tweak the model.

3. Testing data

Used to validate final accuracy.
Should not be used for hyperparameter tuning.



Data composition

1. Training/Test/Validation Split

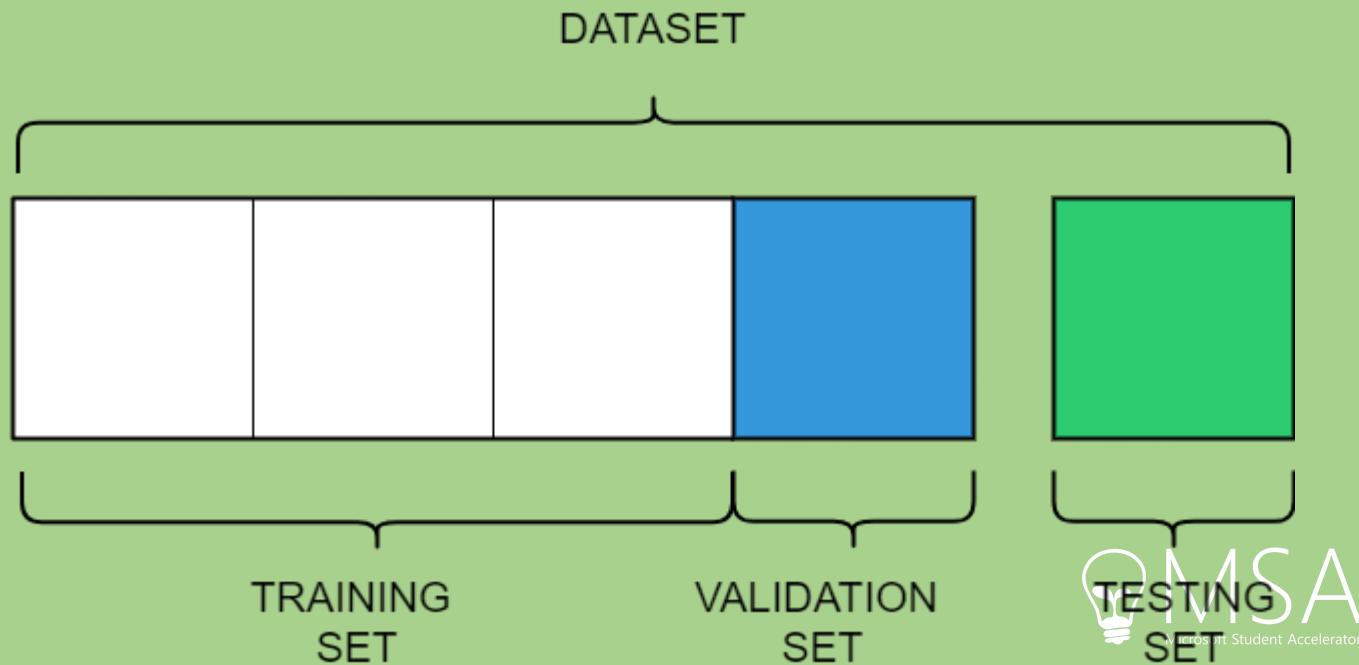
Usually more training than testing examples.

Need a balance.

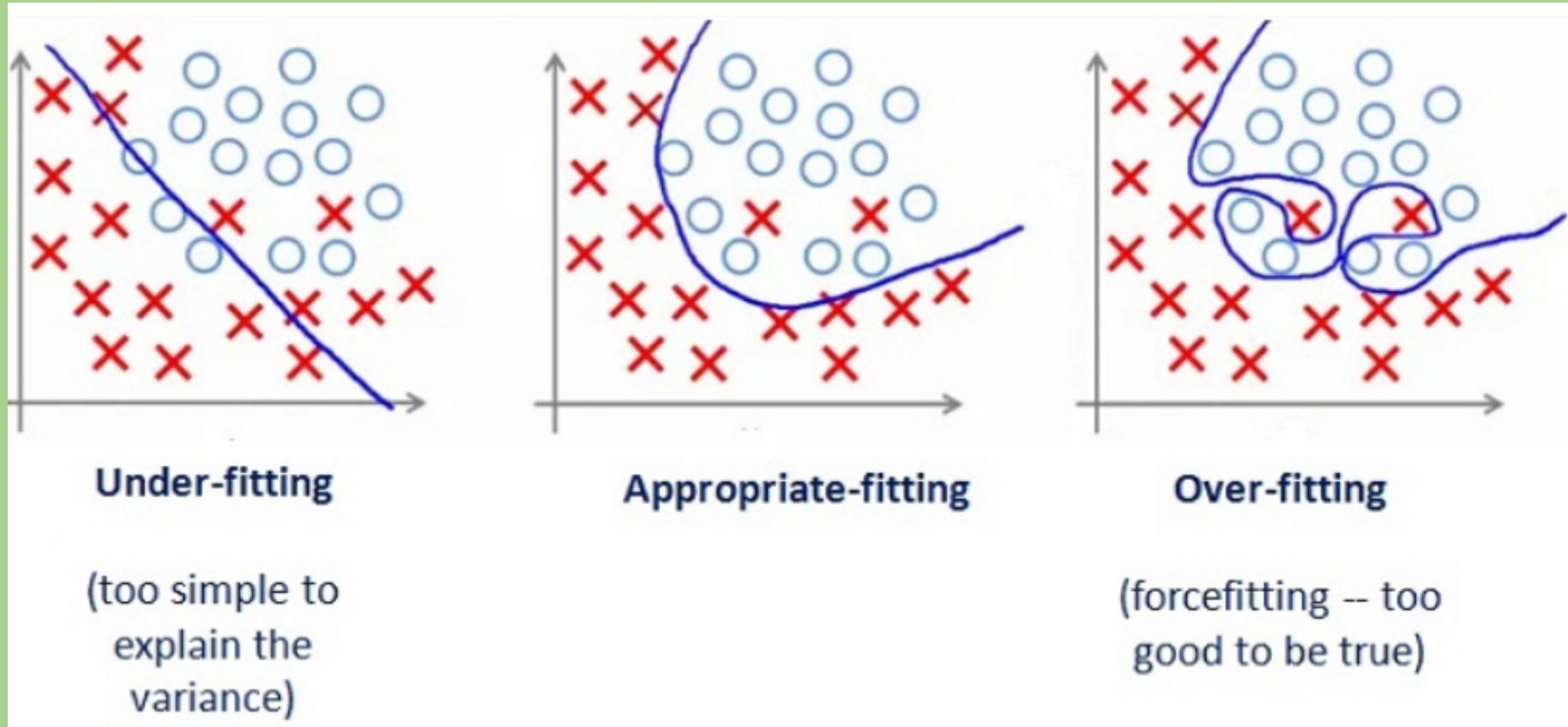
70/20/10 , 60/30/10, etc.

2. Risks in selecting the split

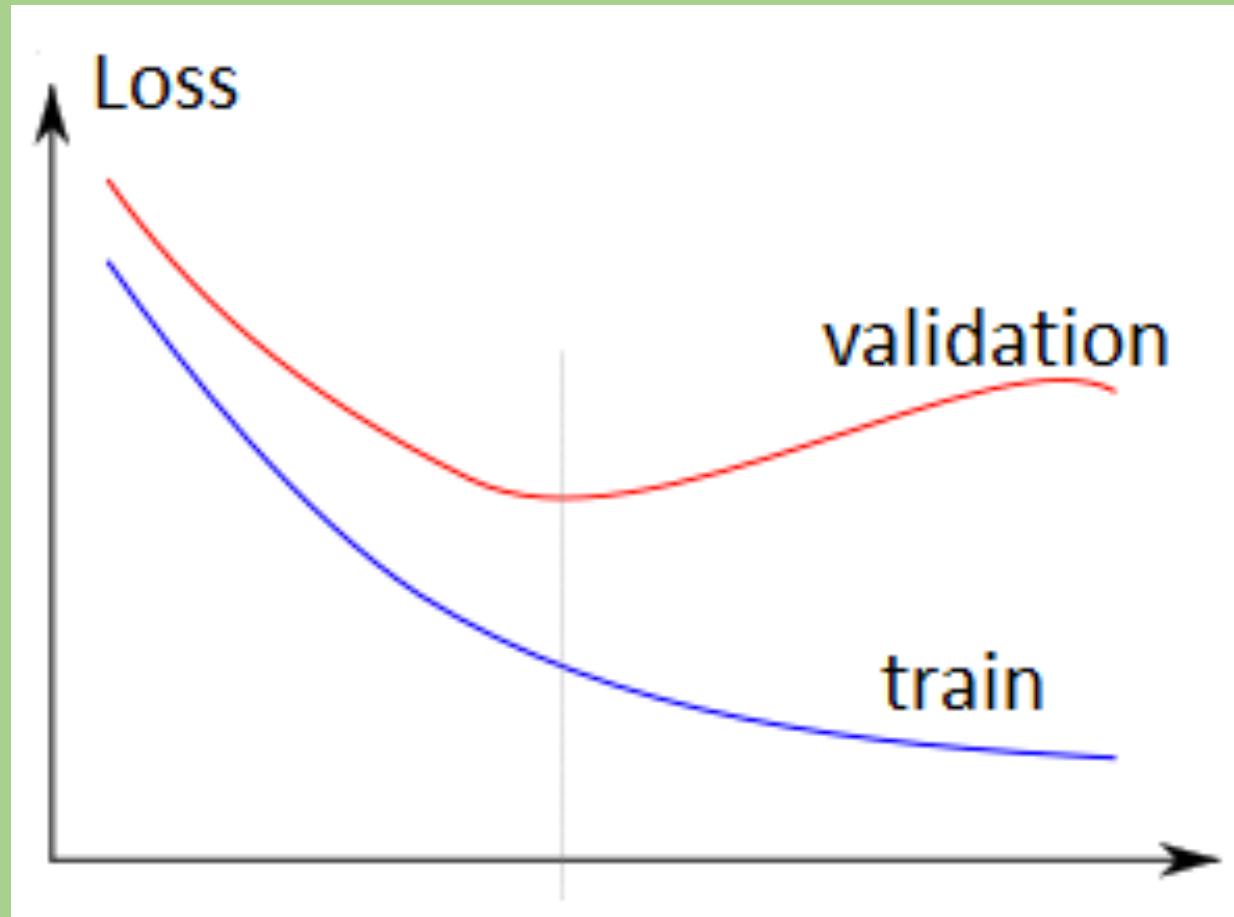
Overfitting, model generalizes too much on training data because of low number of examples.



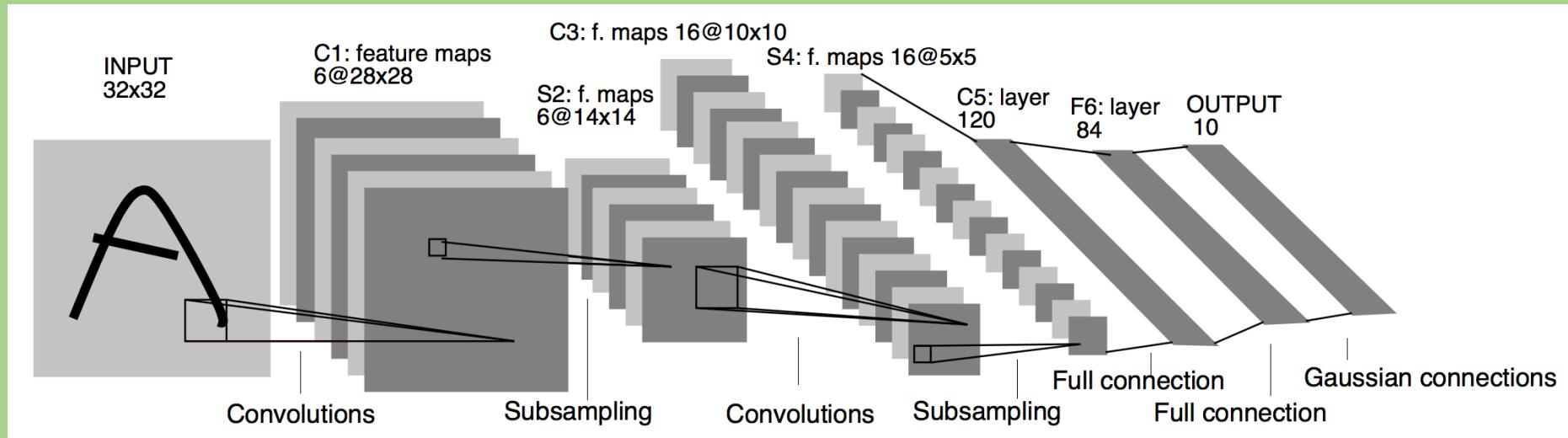
Overfitting/Underfitting



Overfitting



CNNs



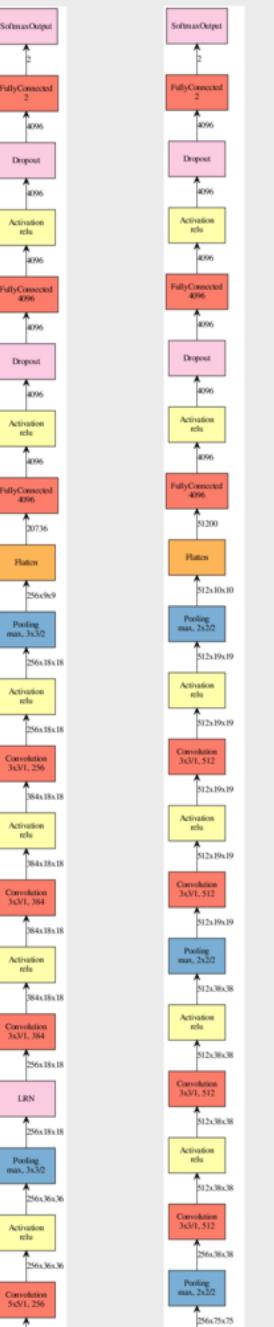
CNN

Architectures

LeNet



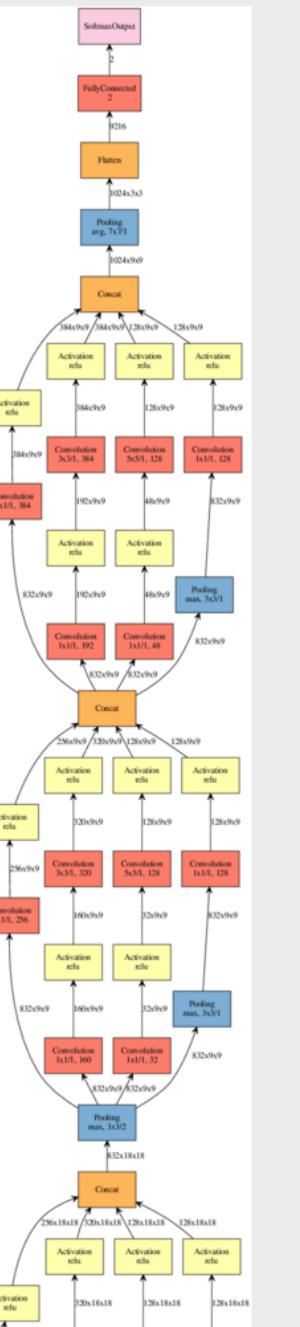
AlexNet



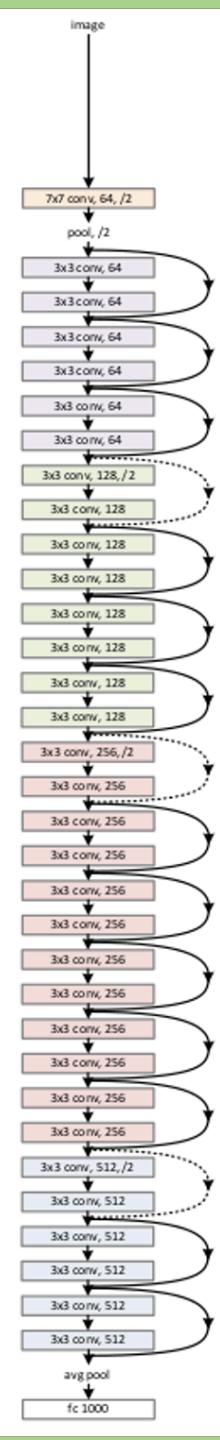
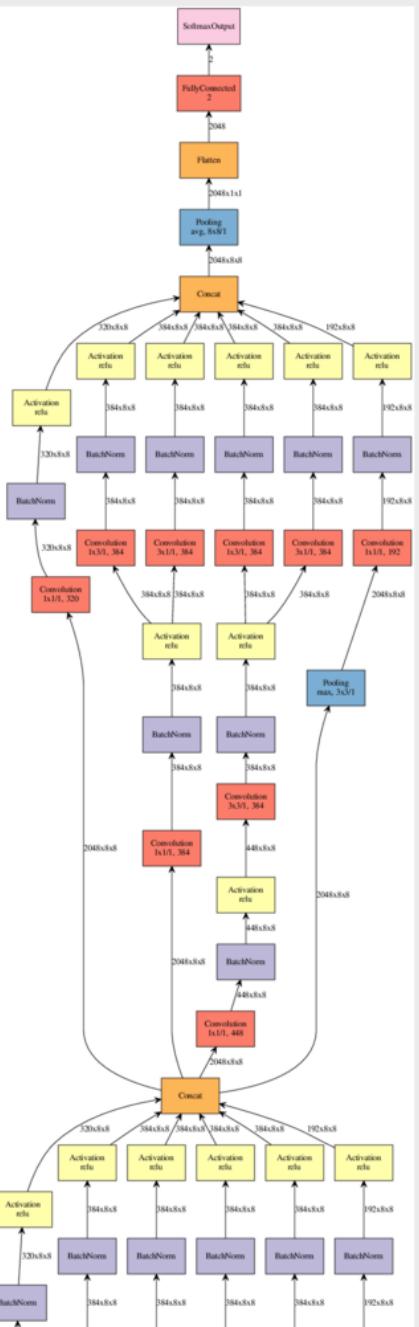
VGG



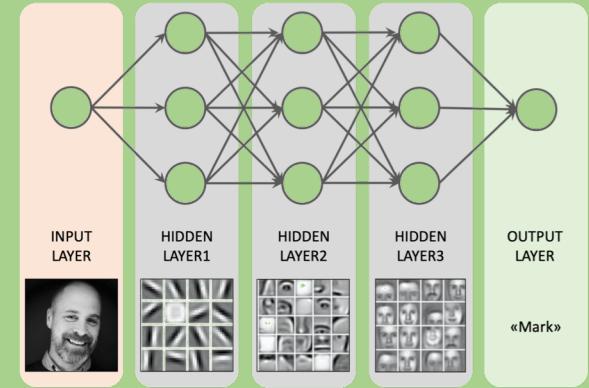
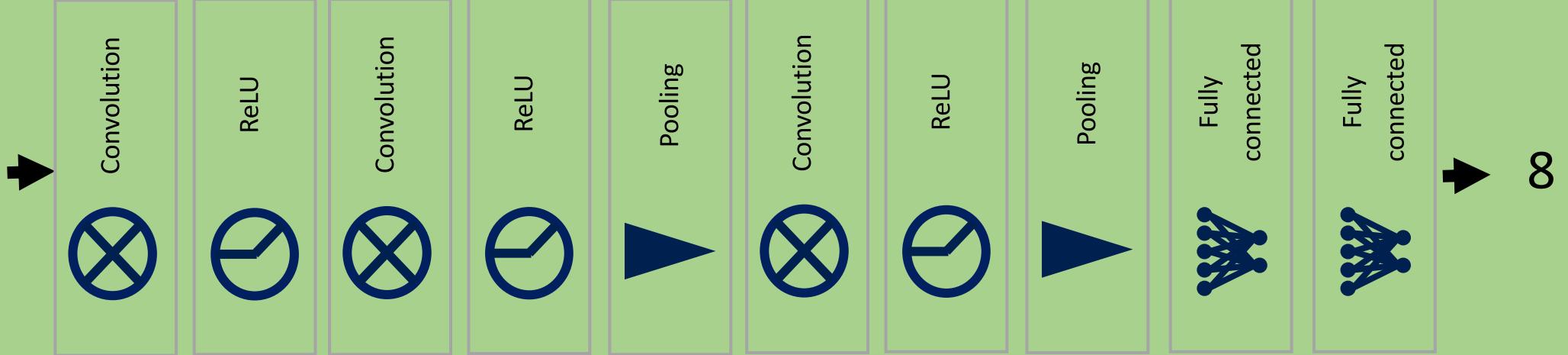
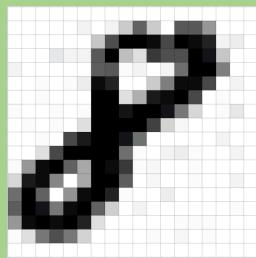
GoogLeNet



Inception V3



CNN



CNN slides from:

[Brandon Rohrer](#)

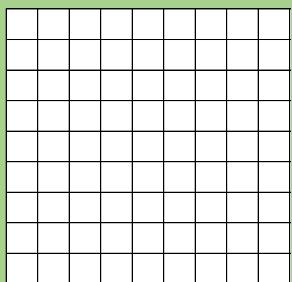
[@ brohrer on Twitter](#)



A toy ConvNet: X's and O's

Says whether a picture is of an X or an O

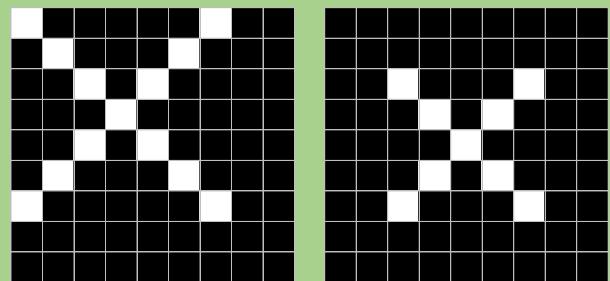
A two-dimensional
array of pixels



For example



Trickier cases

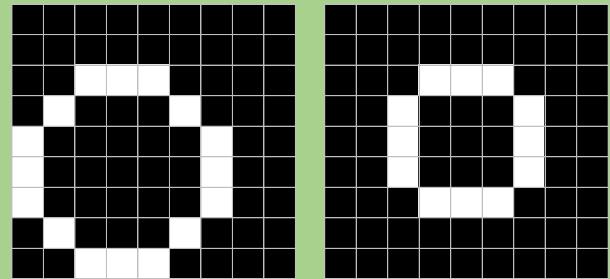


translation

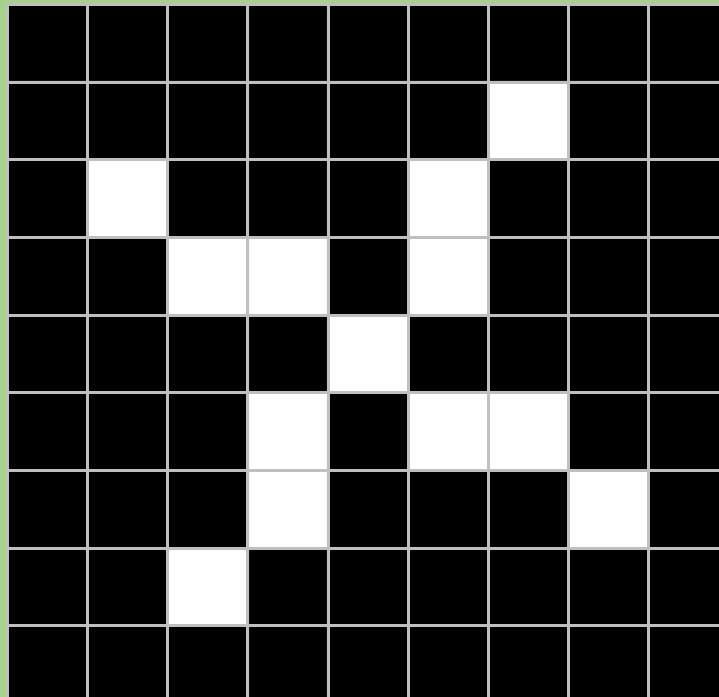
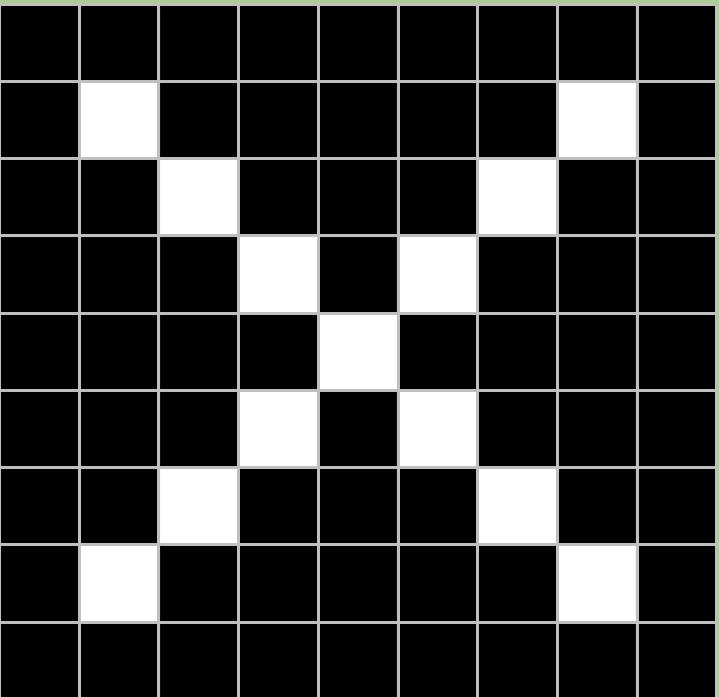
scaling

rotation

weight



Deciding is hard



What computers see



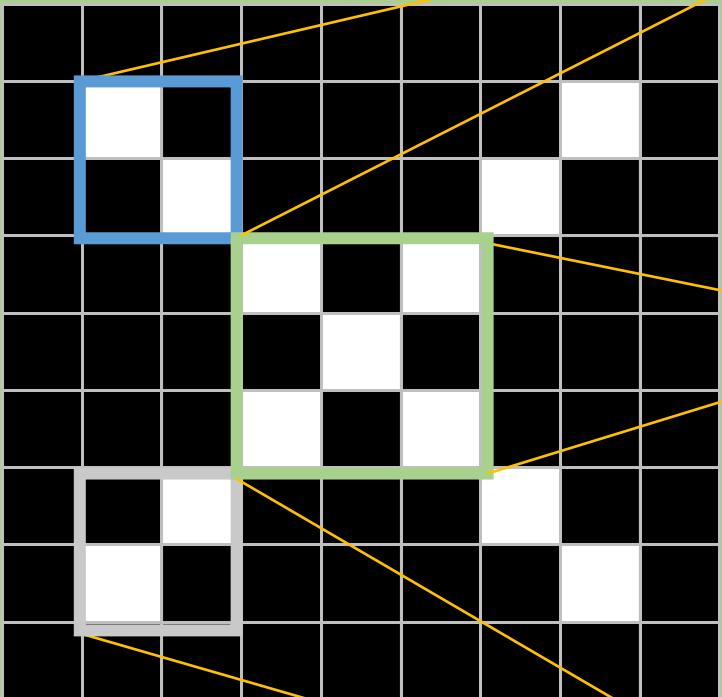
What computers see

Computers are literal

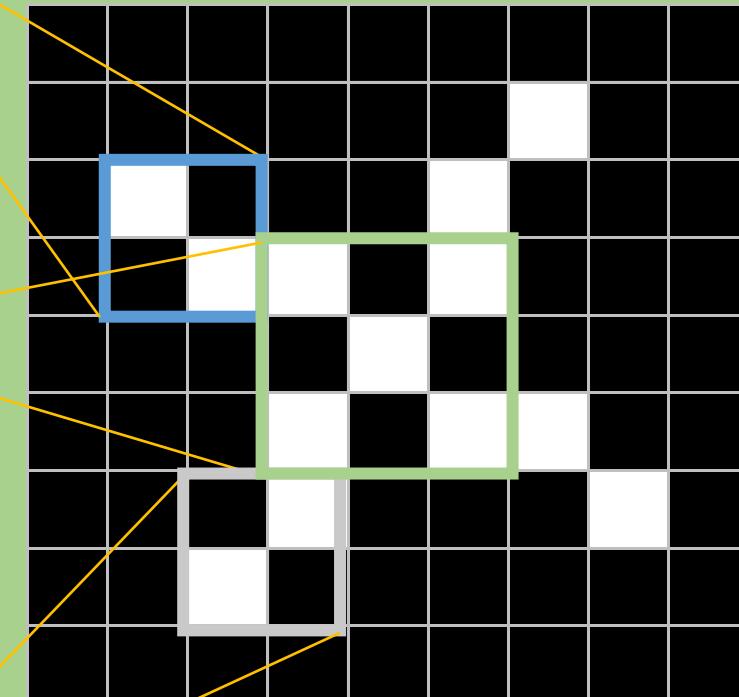


ConvNets match pieces of the image

=



=



=

Features match pieces of the image

1	-1	-1
-1	1	-1
-1	-1	1

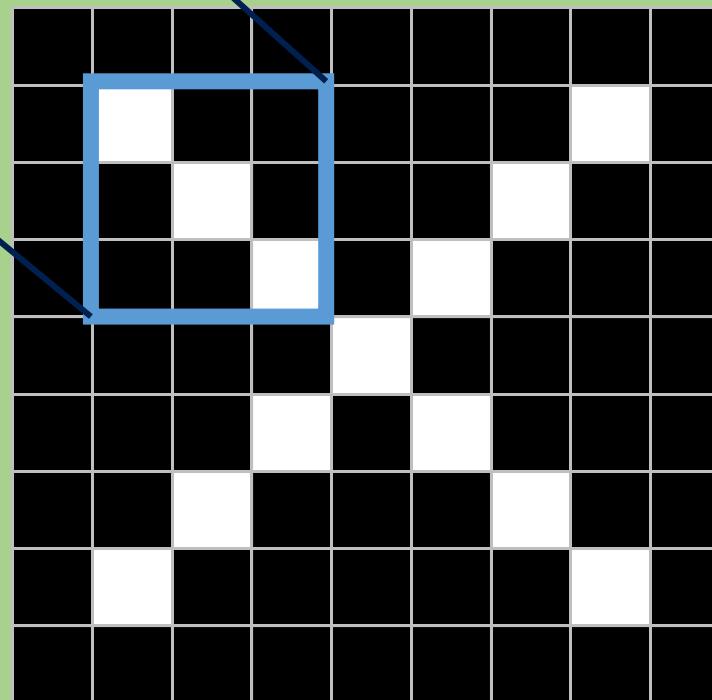
1	-1	1
-1	1	-1
1	-1	1

-1	-1	1
-1	1	-1
1	-1	-1

1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

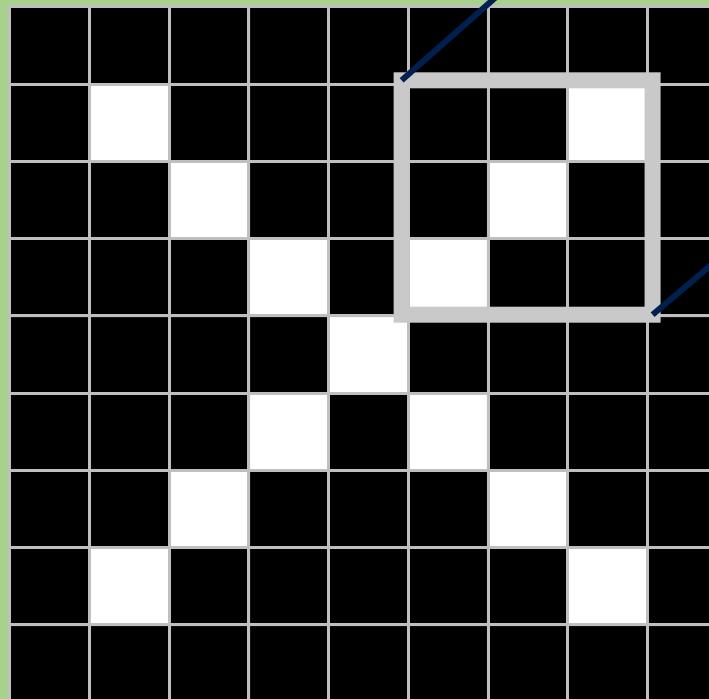
-1	-1	1
-1	1	-1
1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

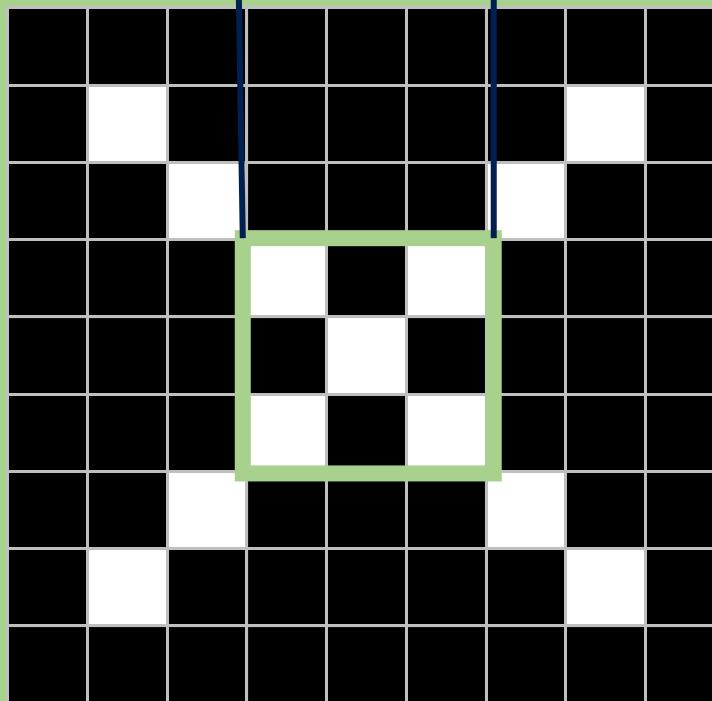
-1	-1	1
-1	1	-1
1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

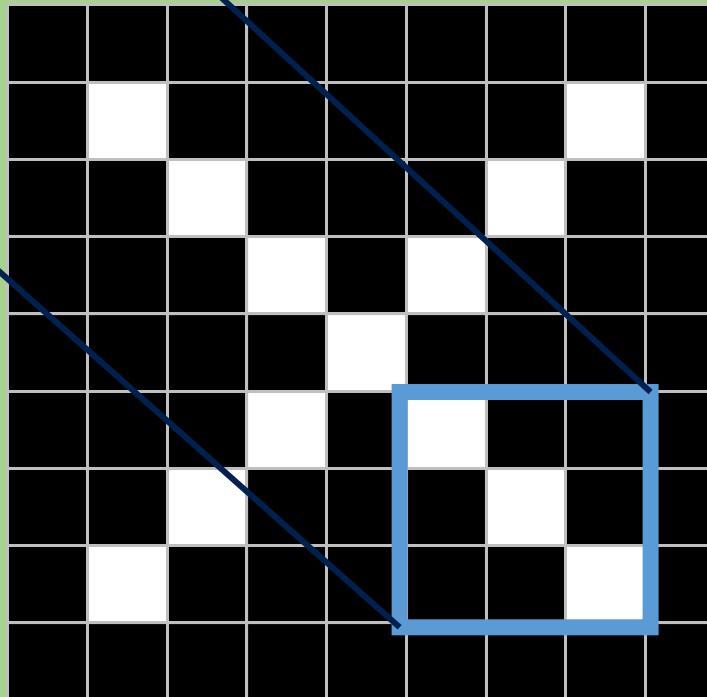
-1	-1	1
-1	1	-1
1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

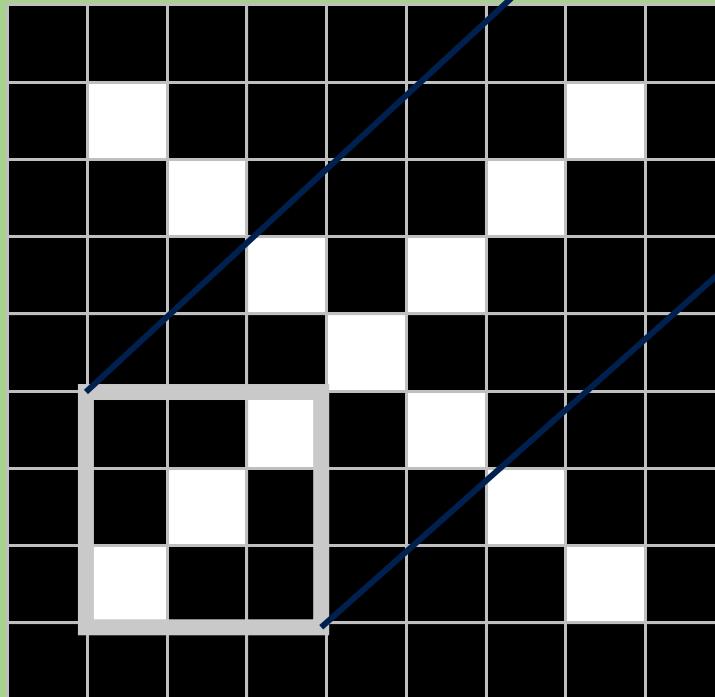
-1	-1	1
-1	1	-1
1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

-1	-1	1
-1	1	-1
1	-1	-1



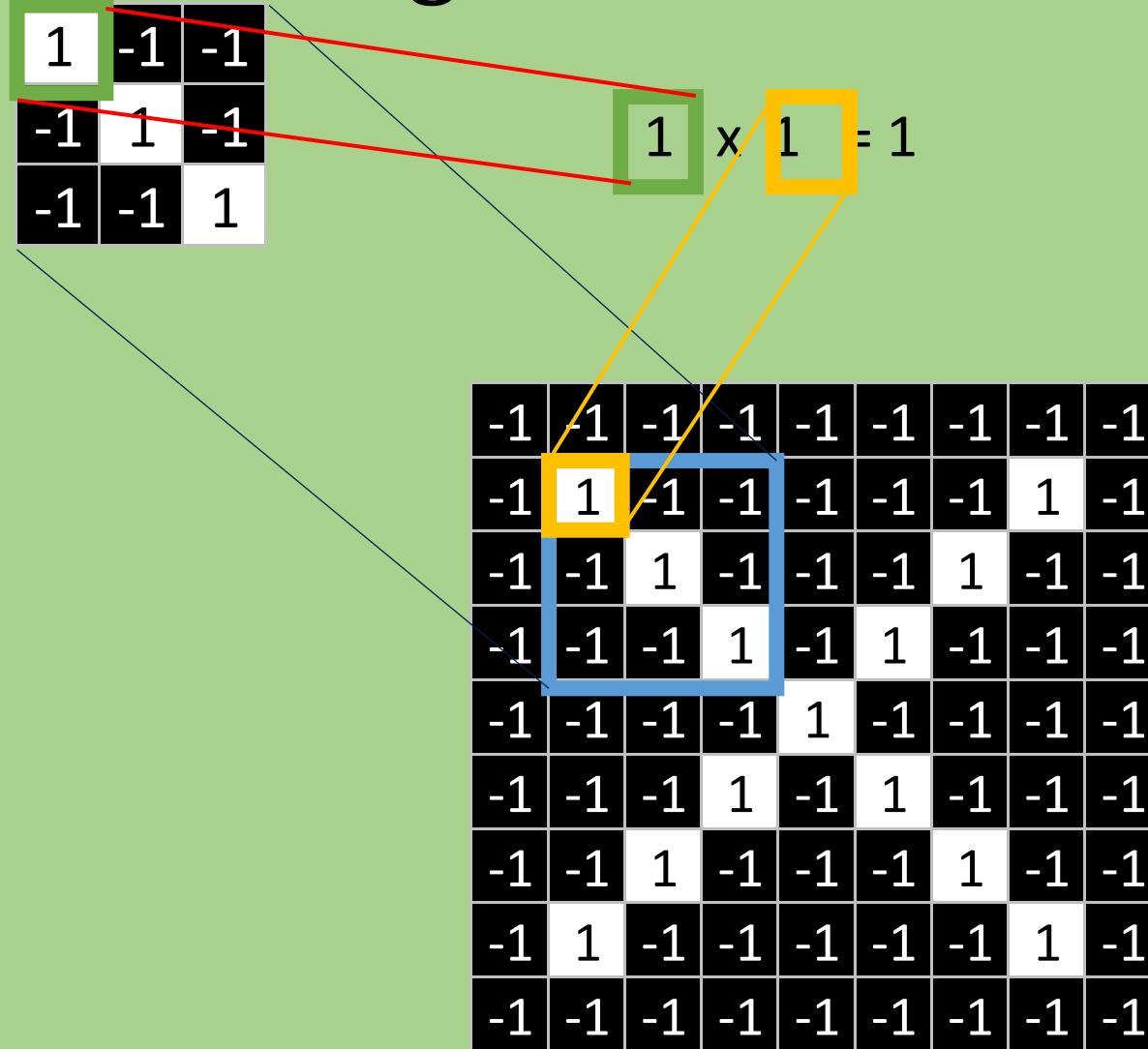
Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

Filtering: The math behind the match

1. Line up the feature and the image patch.
2. Multiply each image pixel by the corresponding feature pixel.
3. Add them up.
4. Divide by the total number of pixels in the feature.

Filtering: The math behind the match



Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

$$1 \times 1 = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	1	-1
-1	-1	-1	1	-1	-1	1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1		

Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

$$-1 \times -1 = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1

Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

$$-1 \times 1 = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	1

Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

$$-1 \times -1 = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	1	-1	-1	1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	1
1		

Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

$$1 \times 1 = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	1
1	1	

Filtering: The math behind the match

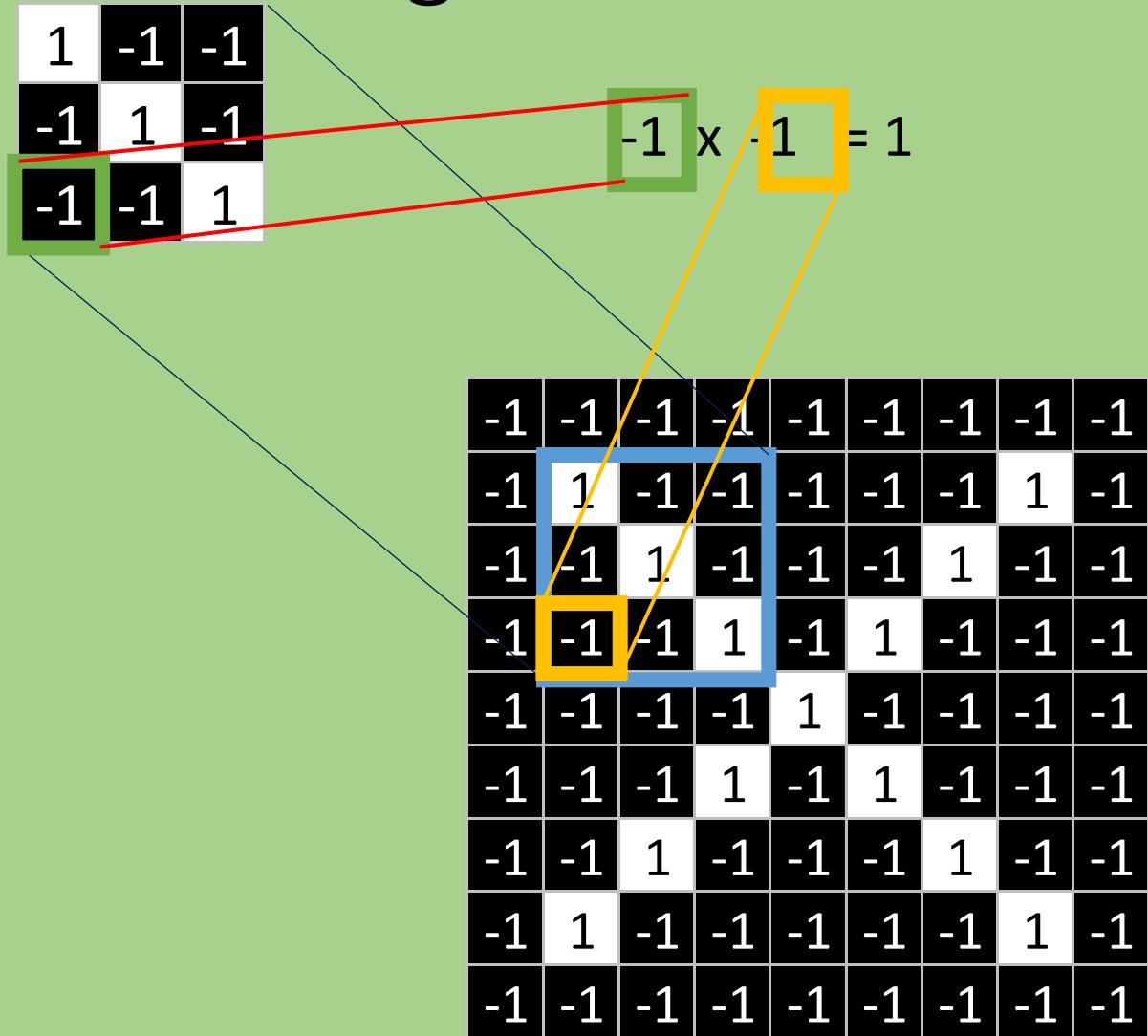
1	-1	-1
-1	1	-1
-1	-1	1

$$-1 \times -1 = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	1
1	1	1

Filtering: The math behind the match



1	1	1
1	1	1
1		

Filtering: The math behind the match

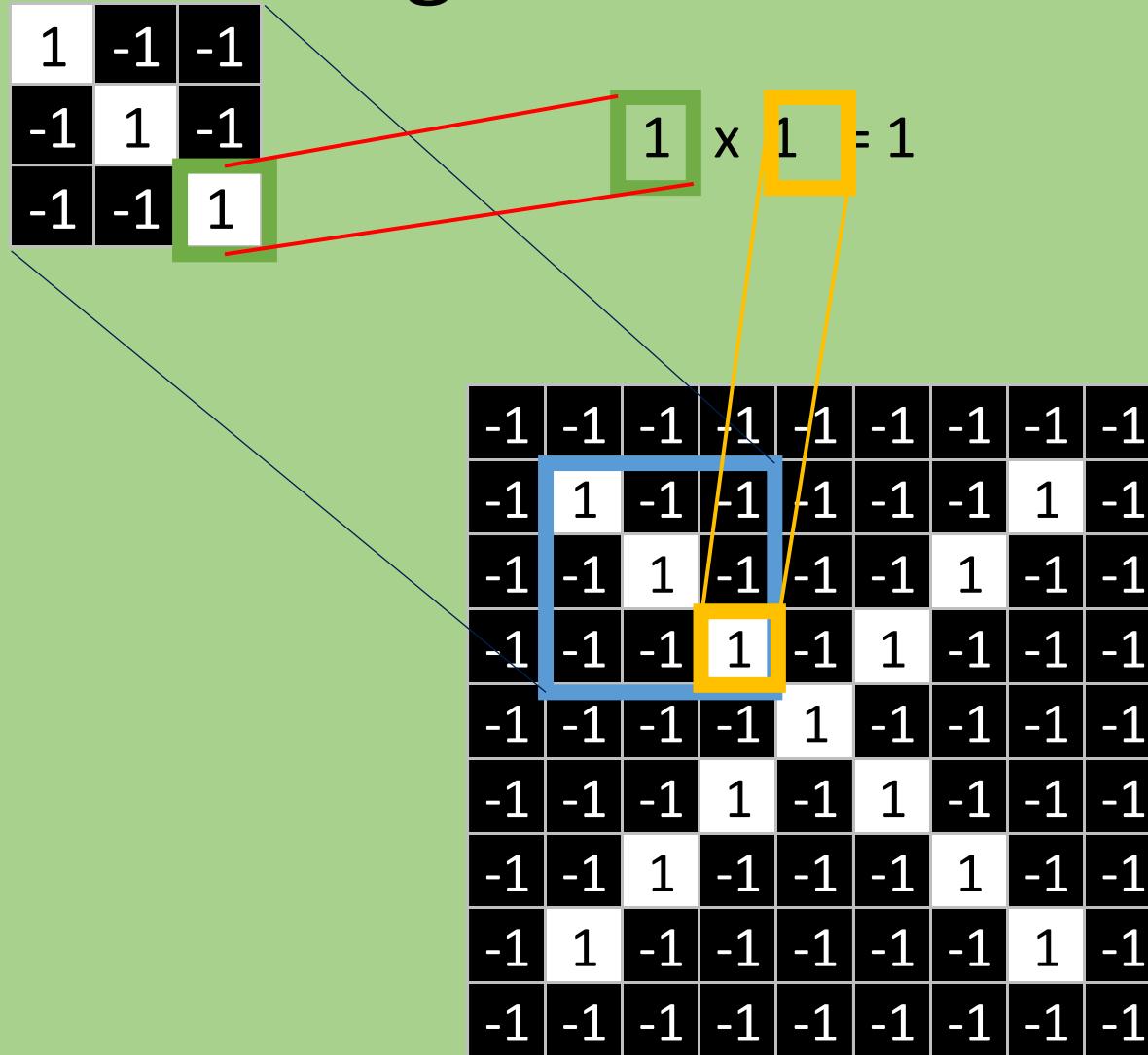
1	-1	-1
-1	1	-1
-1	-1	1

$$-1 \times 1 = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	1
1	1	1
1	1	

Filtering: The math behind the match



1	1	1
1	1	1
1	1	1

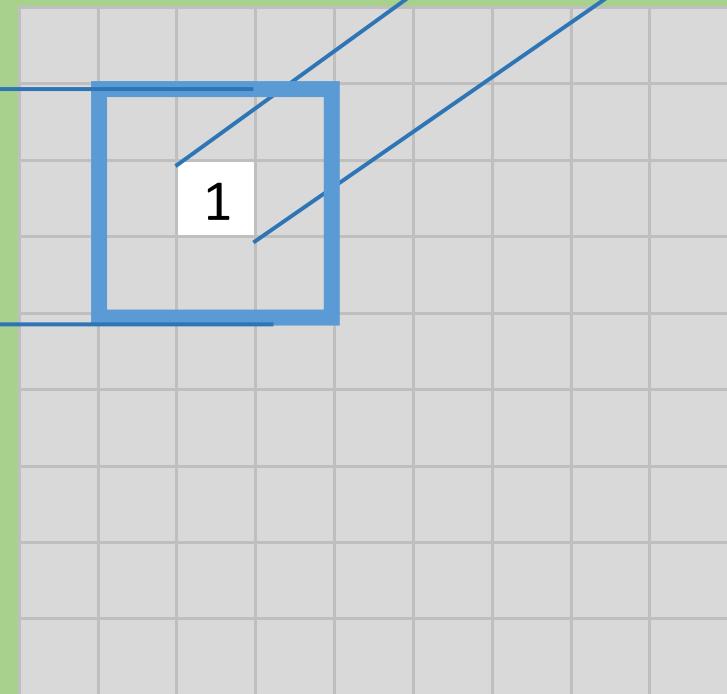
Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

1	1	1
1	1	1
1	1	1

$$\frac{1 + 1 + 1 + 1 + 1 + 1 + 1 + 1}{9} = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

$$1 \times 1 = 1$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

-1

1

1	1	-1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

$$-1 \times 1 = -1$$

Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	-1
1	1	1
-1	1	1

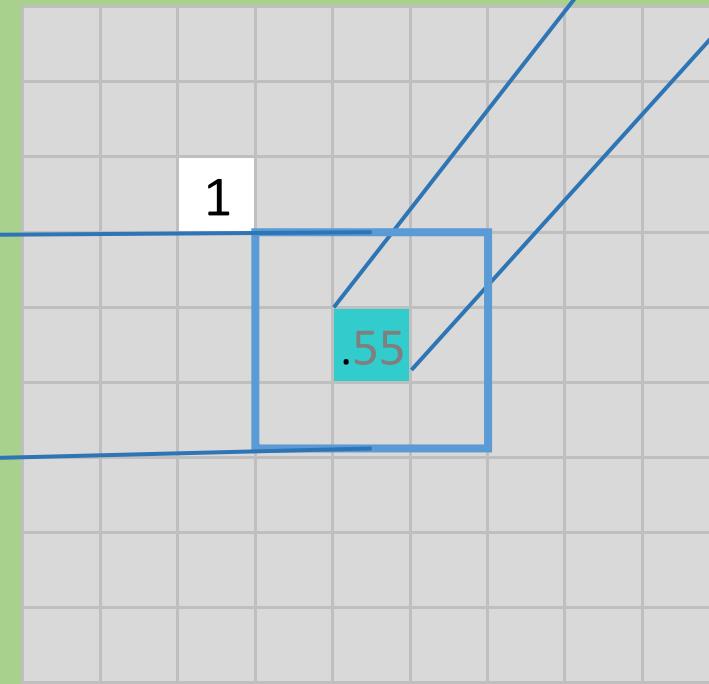
Filtering: The math behind the match

1	-1	-1
-1	1	-1
-1	-1	1

1	1	-1
1	1	1
-1	1	1

$$\frac{1 + 1 - 1 + 1 + 1 + 1 - 1 + 1 + 1}{9} = .55$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



Convolution: Trying every possible match

1	-1	-1
-1	1	-1
-1	-1	1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Convolution: Trying every possible match

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	
-1	-1	1	-1	-1	-1	1	-1	-1	
-1	-1	-1	1	-1	1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	-1	1	-1	1	-1	-1	-1	
-1	-1	1	-1	-1	-1	1	-1	-1	
-1	1	-1	-1	-1	-1	-1	1	-1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	



1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



$$\begin{matrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{matrix}$$

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



$$\begin{matrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{matrix}$$

=

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



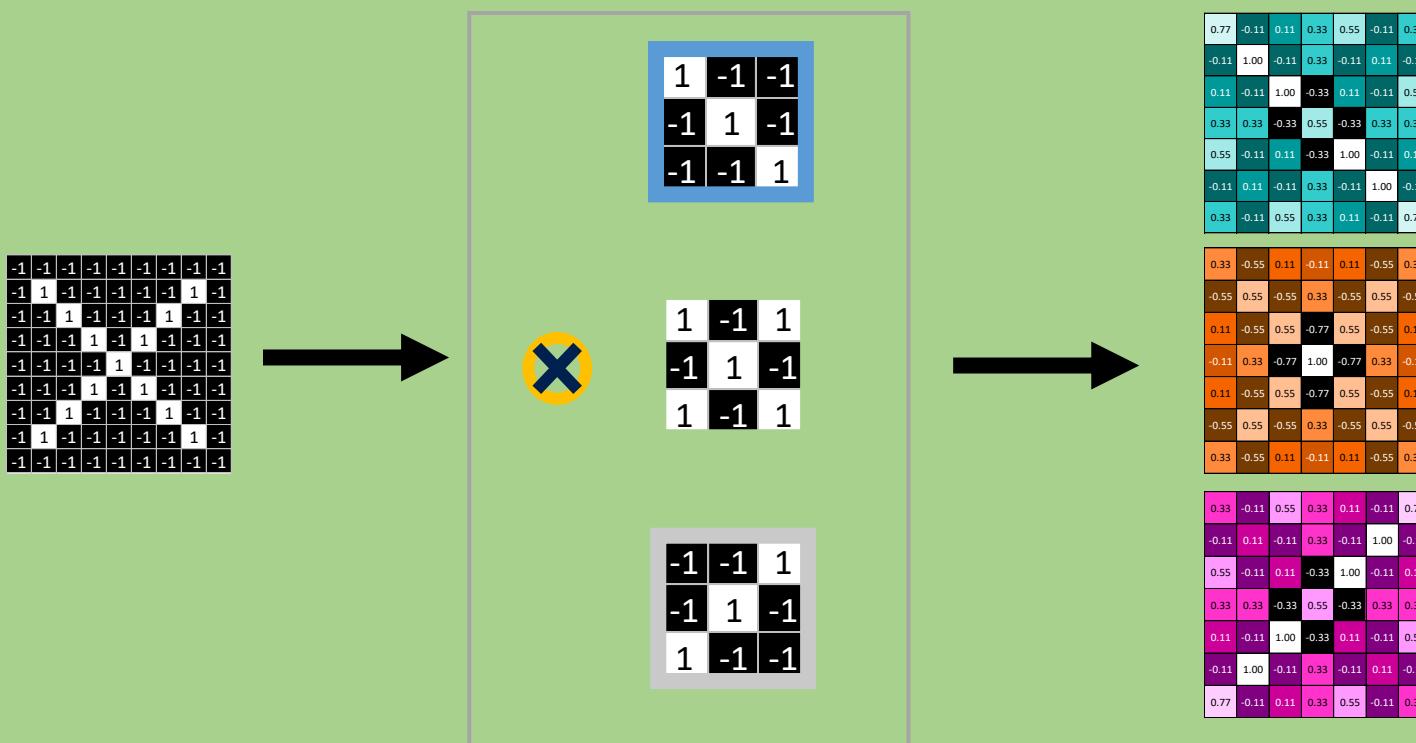
$$\begin{matrix} -1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{matrix}$$

=

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

Convolution layer

One image becomes a stack of filtered images



Convolution layer

One image becomes a stack of filtered images

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	
-1	-1	1	-1	-1	-1	1	-1	-1	
-1	-1	-1	1	-1	1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	-1	-1	-1	1	-1	-1	-1	
-1	-1	-1	-1	-1	-1	1	-1	-1	
-1	1	-1	-1	-1	-1	-1	1	-1	
-1	-1	-1	-1	-1	-1	-1	-1	1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

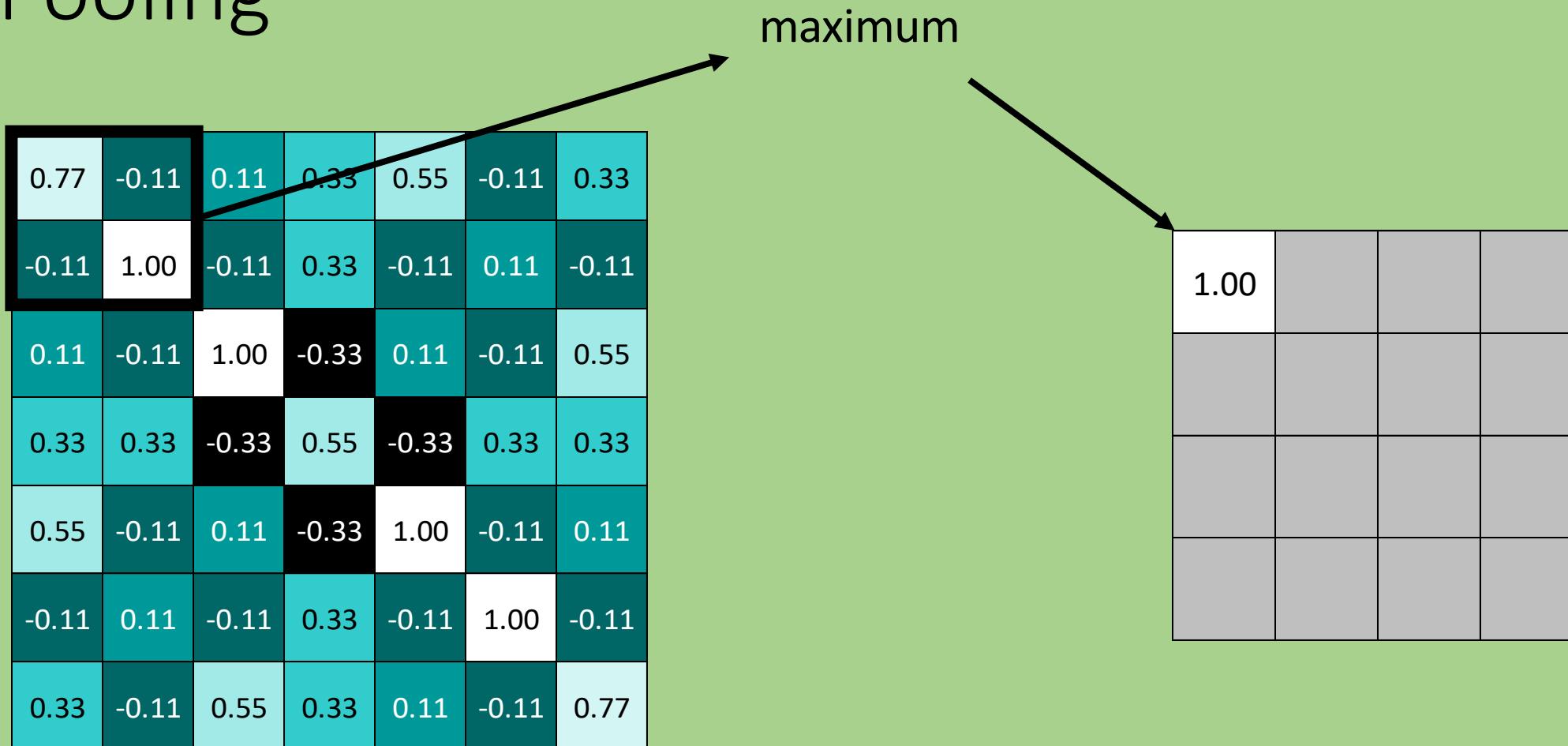
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

Pooling: Shrinking the image stack

1. Pick a window size (usually 2 or 3).
2. Pick a stride (usually 2).
3. Walk your window across your filtered images.
4. From each window, take the maximum value.

Pooling



Pooling

maximum

0.77	-0.11	0.11	0.33	0.55	0.11	0.33	
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11	
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55	
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33	
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11	
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11	
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77	

1.00	0.33		

Pooling

maximum

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33	
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11	
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55	
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33	
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11	
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11	
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77	

1.00	0.33	0.55	

Pooling

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33	
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11	
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55	
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33	
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11	
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11	
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77	

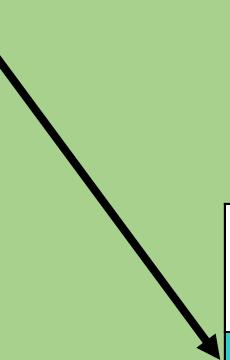
maximum

1.00	0.33	0.55	0.33

Pooling

maximum

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



1.00	0.33	0.55	0.33
0.33			

Pooling

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

max pooling



1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33



0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33



0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

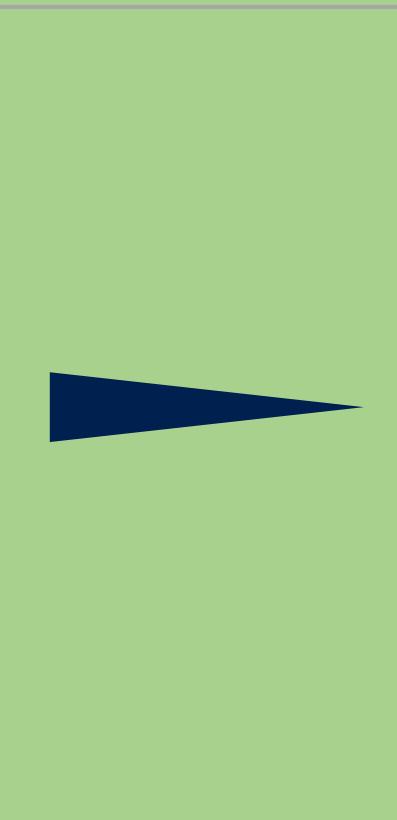
Pooling layer

A stack of images becomes a stack of smaller images.

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33



1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

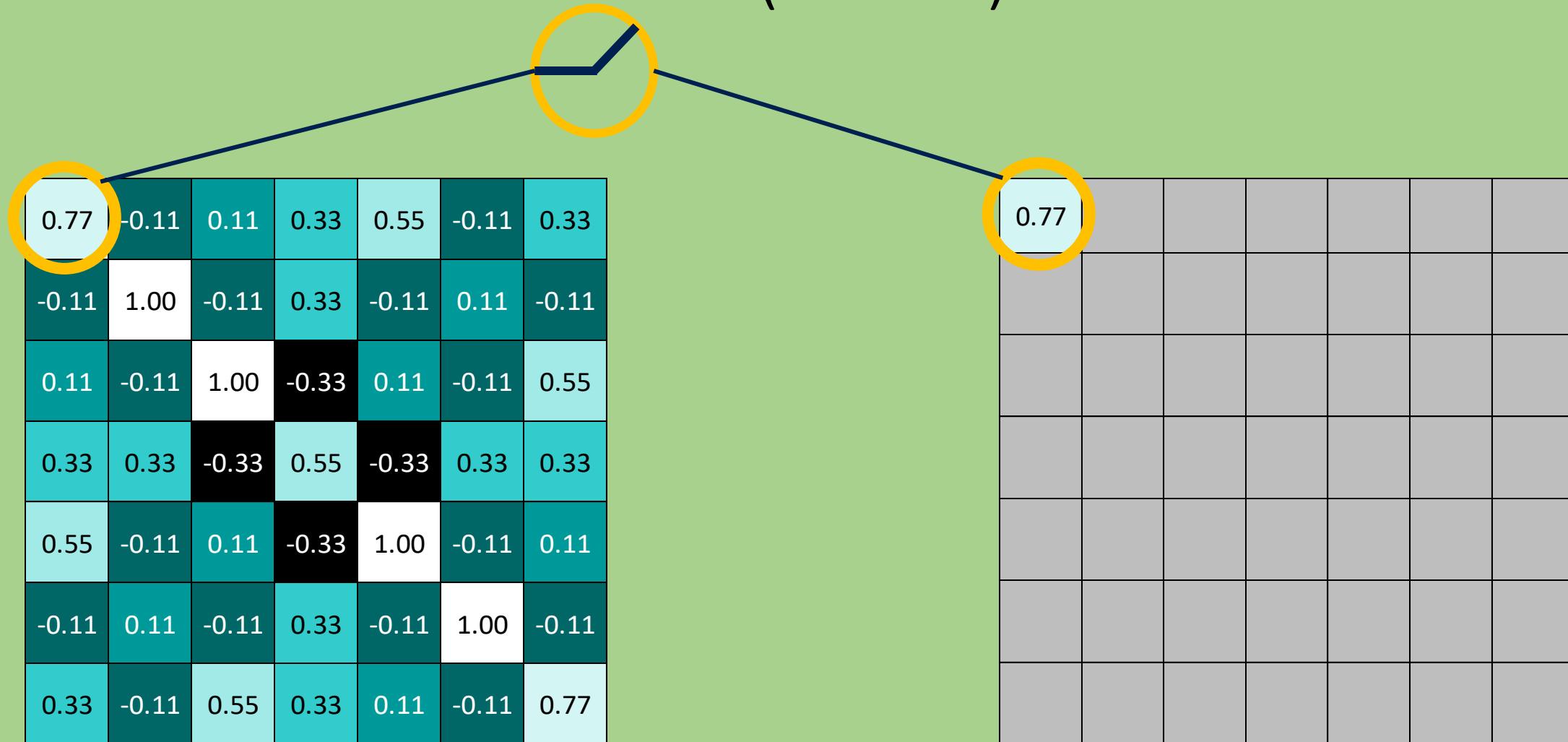
0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

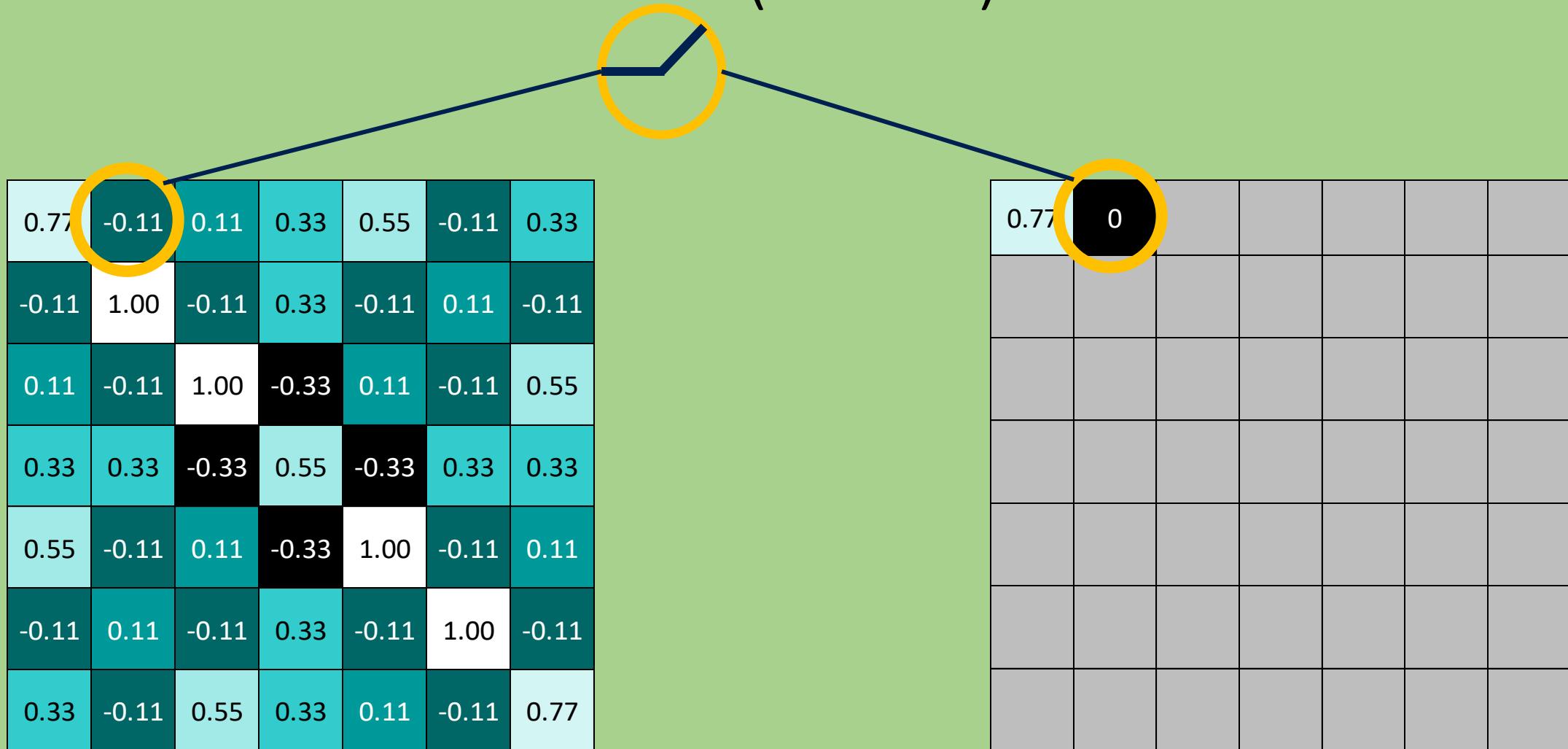
Normalization

Keep the math from breaking by tweaking each of the values just a bit.
Change everything negative to zero.

Rectified Linear Units (ReLUs)



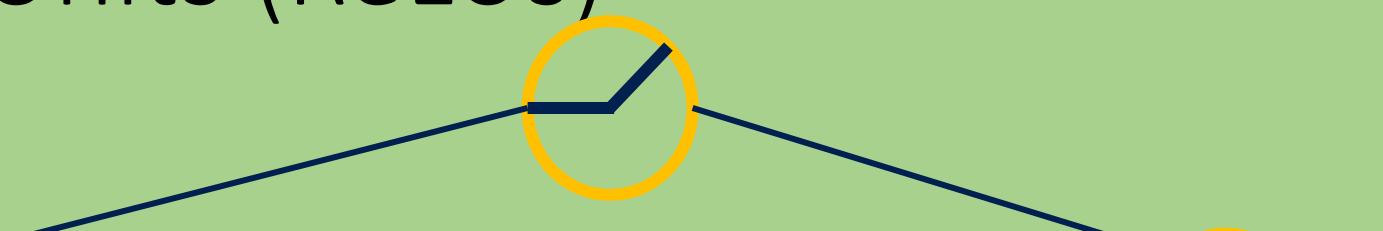
Rectified Linear Units (ReLUs)



Rectified Linear Units (ReLUs)

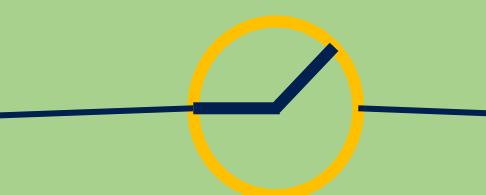
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33



Rectified Linear Units (ReLUs)

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

ReLU layer

A stack of images becomes a stack of images with no negative values.

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33



0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

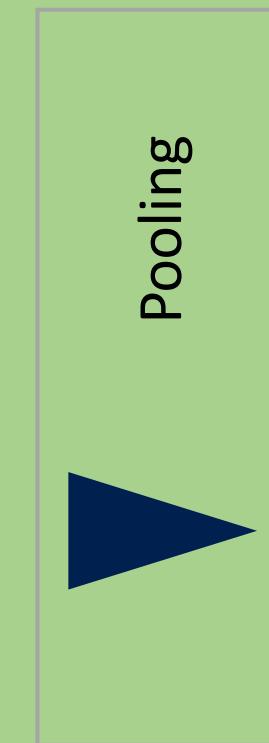
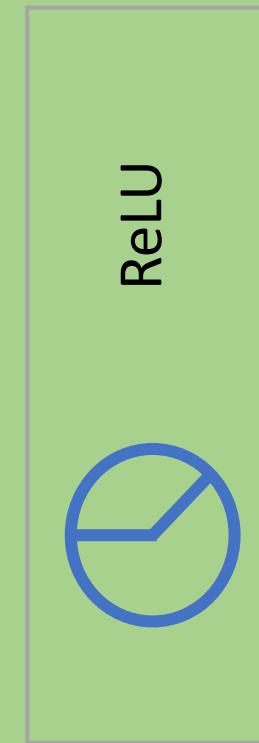
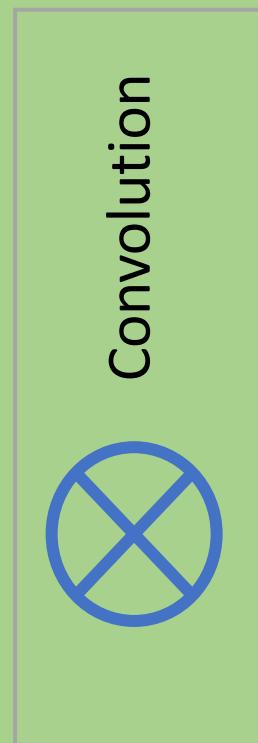
0.33	0	0.11	0	0.11	0	0.33
0	0.55	0	0.33	0	0.55	0
0.11	0	0.55	0	0.55	0	0.11
0	0.33	0	1.00	0	0.33	0
0.11	0	0.55	0	0.55	0	0.11
0	0.55	0	0.33	0	0.55	0
0.33	0	0.11	0	0.11	0	0.33

0.33	0	0.55	0.33	0.11	0	0.77
0	0.11	0	0.33	0	1.00	0
0.55	0	0.11	0	1.00	0	0.11
0.33	0.33	0	0.55	0	0.33	0.33
0.11	0	1.00	0	0.11	0	0.55
0	1.00	0	0.33	0	0.11	0
0.77	0	0.11	0.33	0.55	0	0.33

Layers get stacked

The output of one becomes the input of the next.

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1



1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

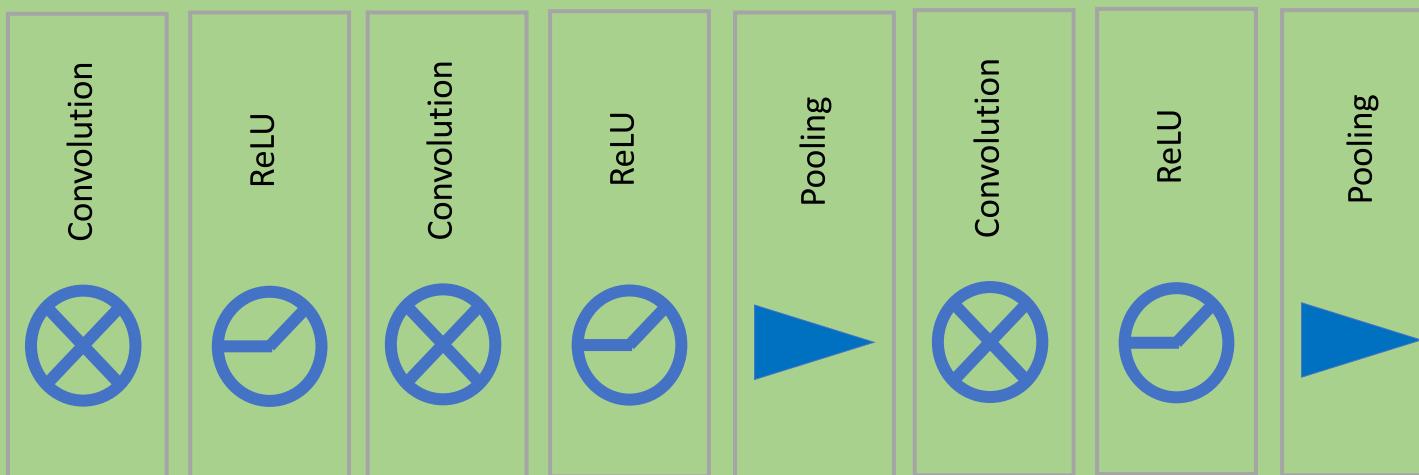
0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

Deep stacking

Layers can be repeated several (or many) times.

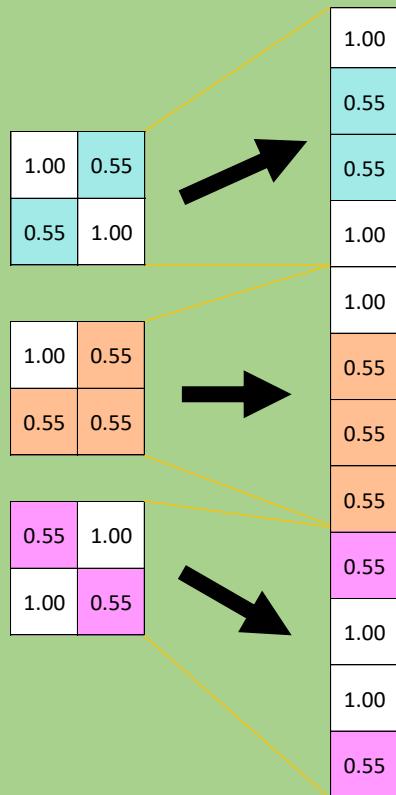
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	
-1	-1	1	-1	-1	-1	1	-1	-1	
-1	-1	-1	1	-1	1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	-1	1	-1	1	-1	-1	-1	
-1	-1	1	-1	-1	-1	1	-1	-1	
-1	1	-1	-1	-1	-1	-1	1	-1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	



1.00	0.55
0.55	1.00
1.00	0.55
0.55	0.55
0.55	1.00
1.00	0.55

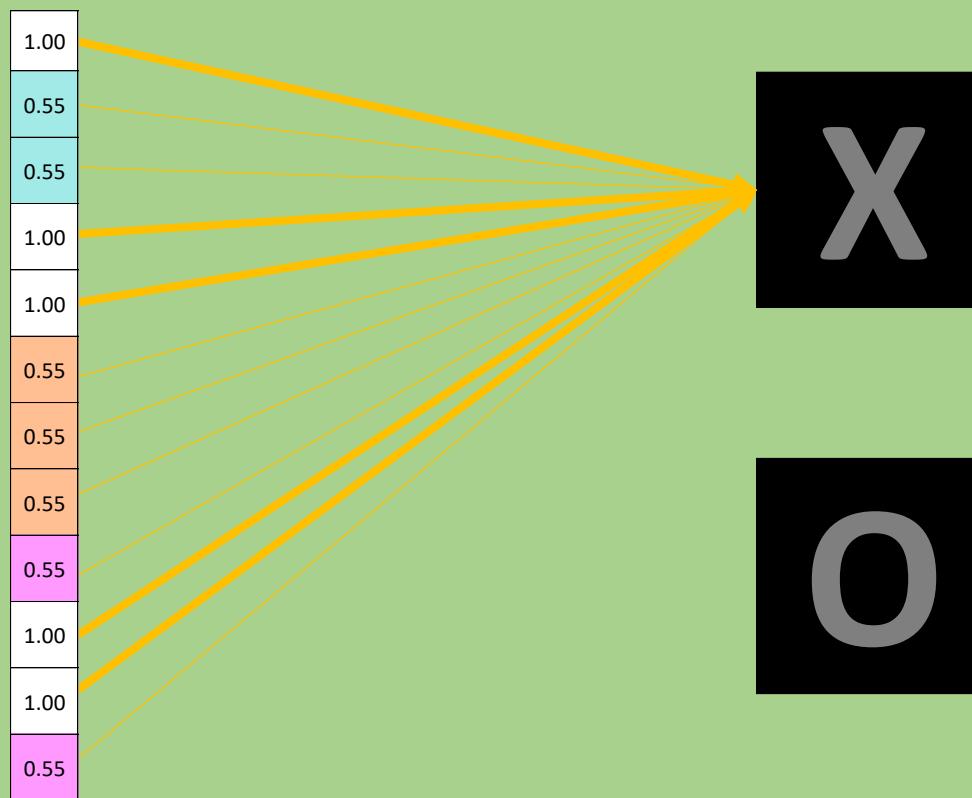
Fully connected layer

Every value gets a vote



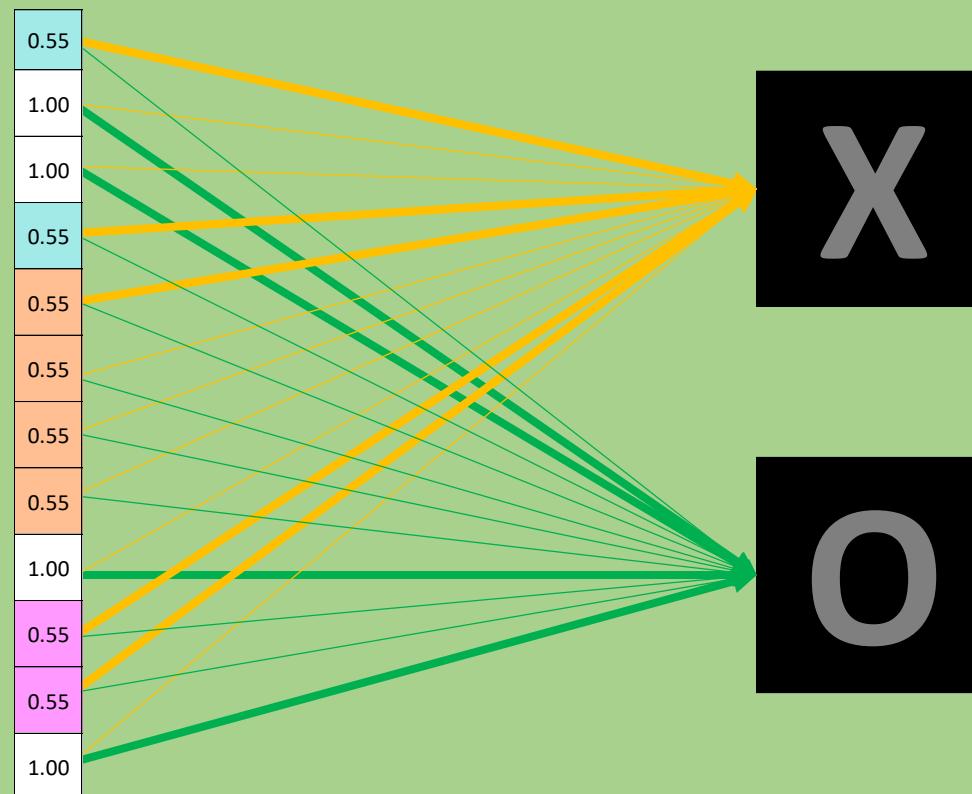
Fully connected layer

Vote depends on how strongly a value predicts X or O



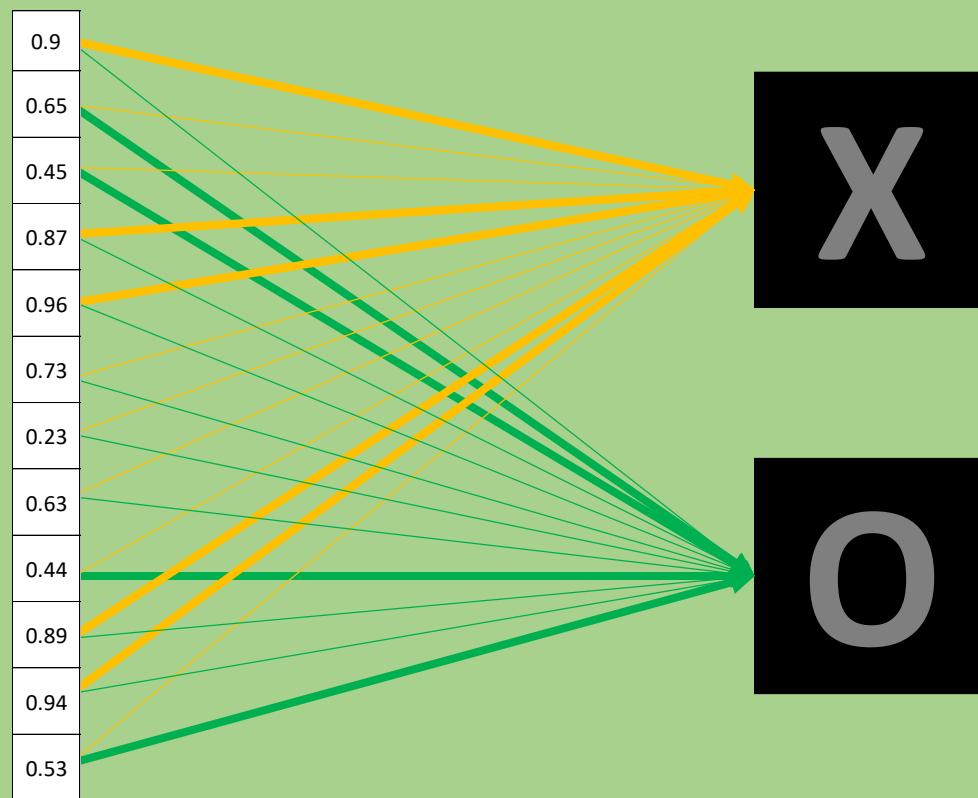
Fully connected layer

Vote depends on how strongly a value predicts X or O



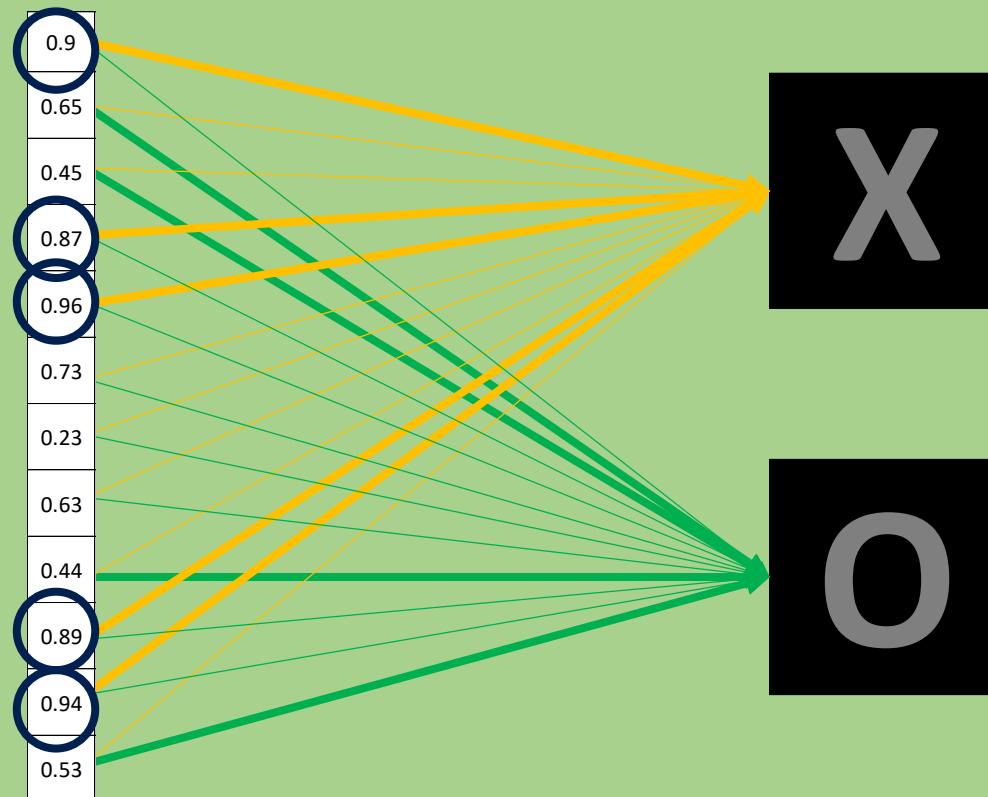
Fully connected layer

Future values vote on X or O



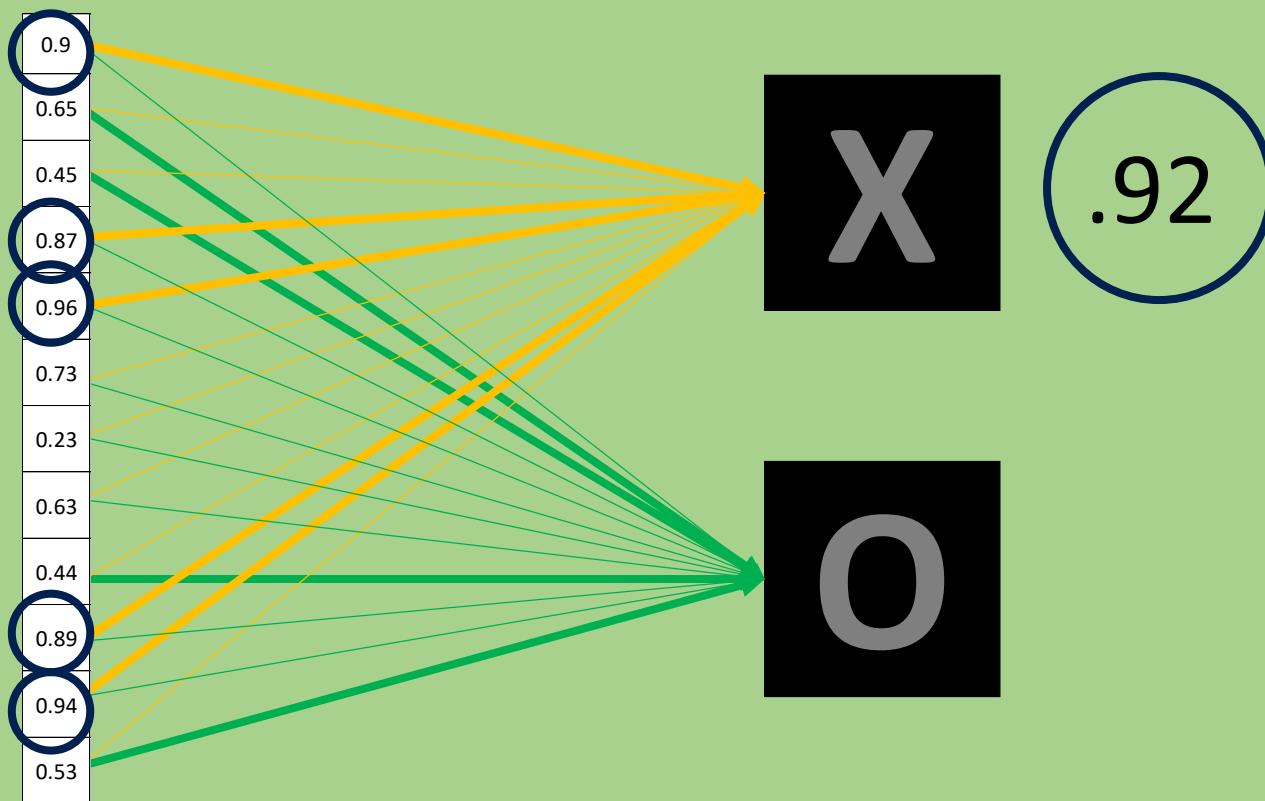
Fully connected layer

Future values vote on X or O



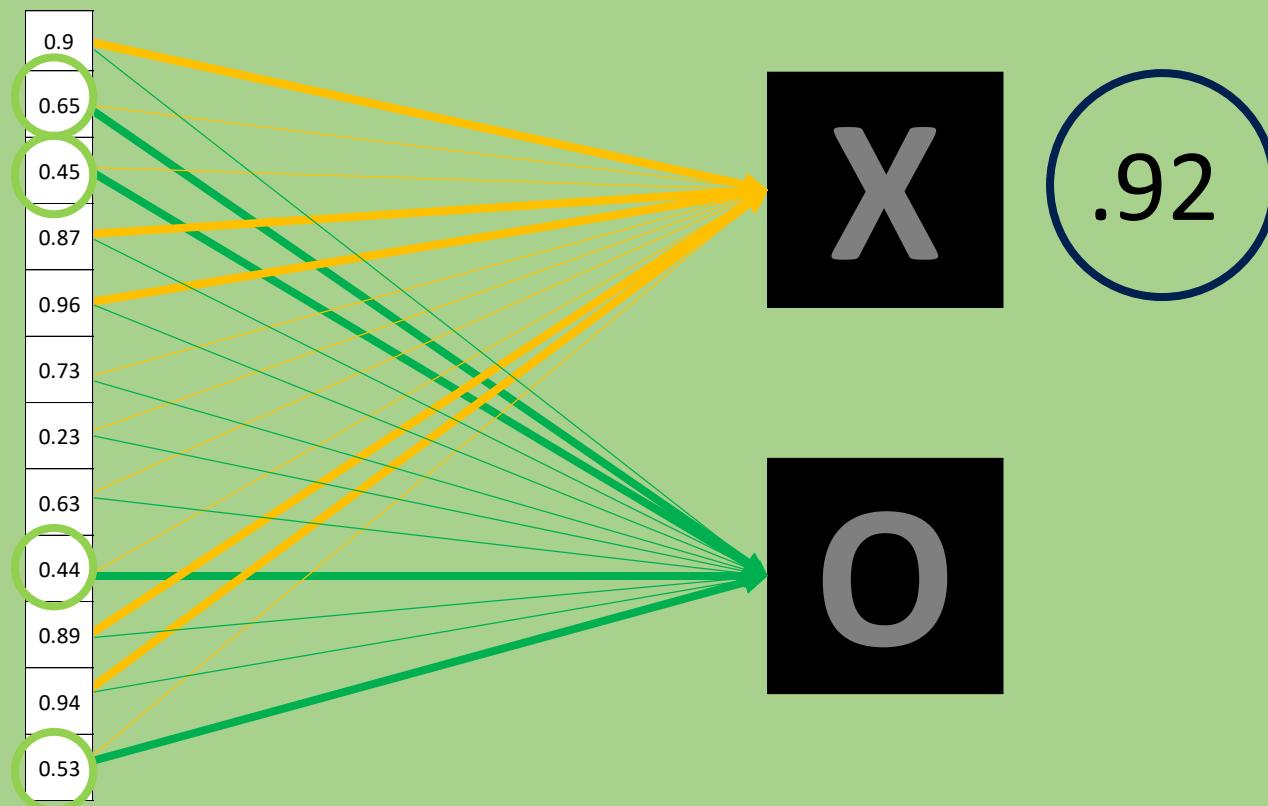
Fully connected layer

Future values vote on X or O



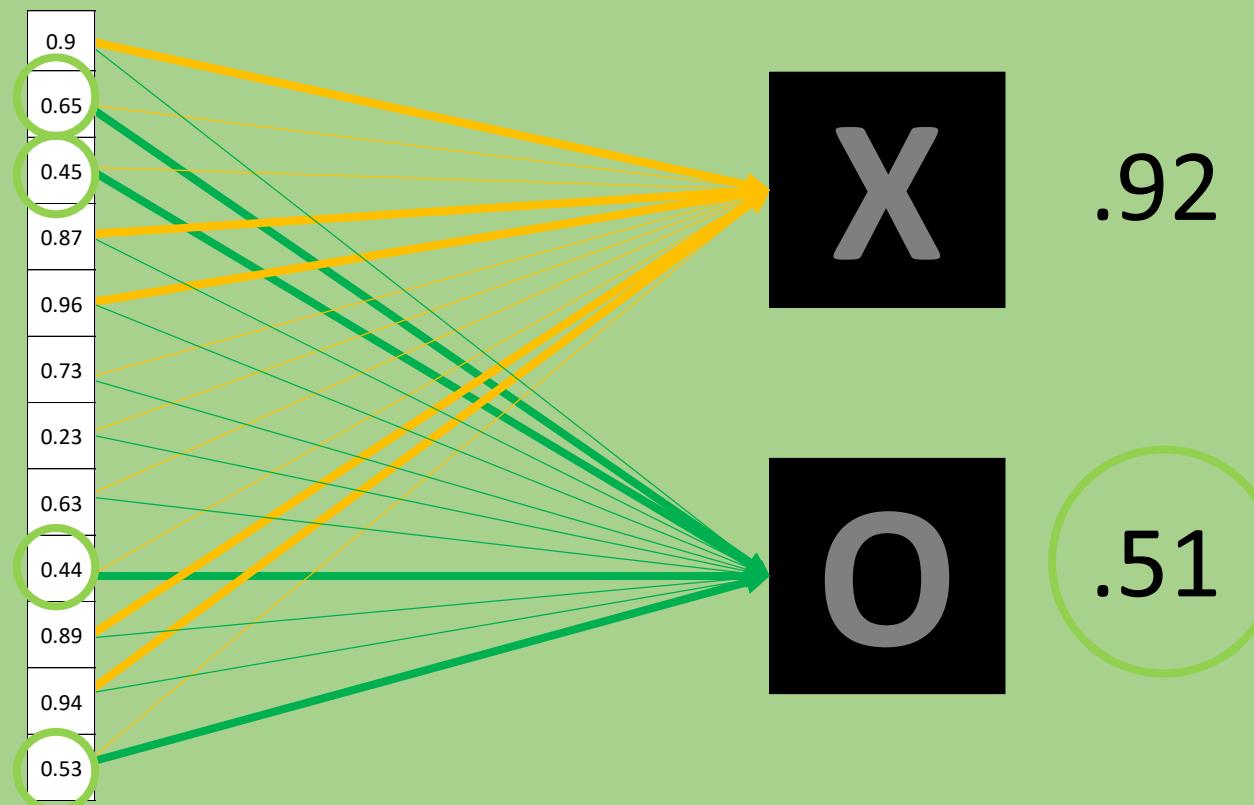
Fully connected layer

Future values vote on X or O



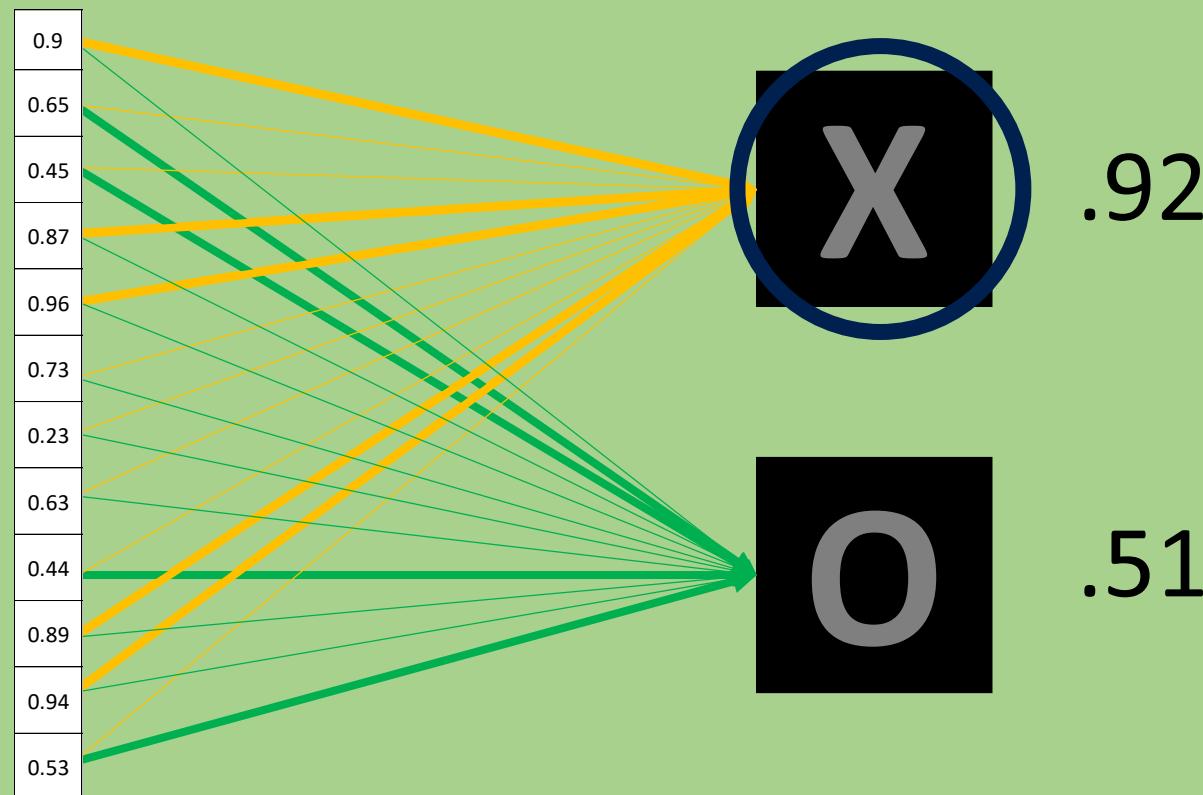
Fully connected layer

Future values vote on X or O



Fully connected layer

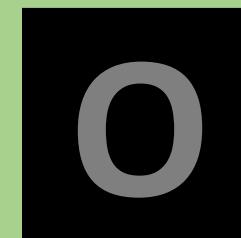
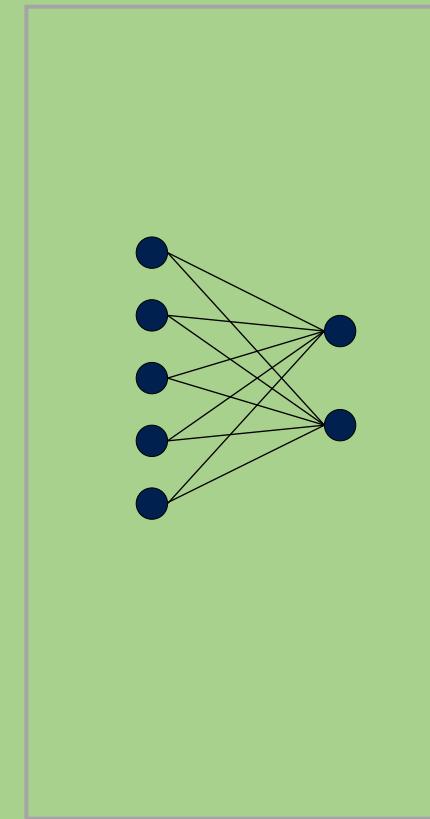
Future values vote on X or O



Fully connected layer

A list of feature values becomes a list of votes.

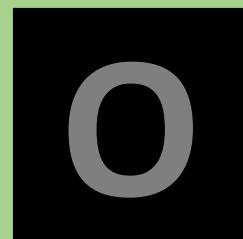
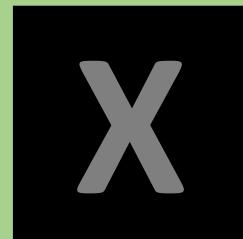
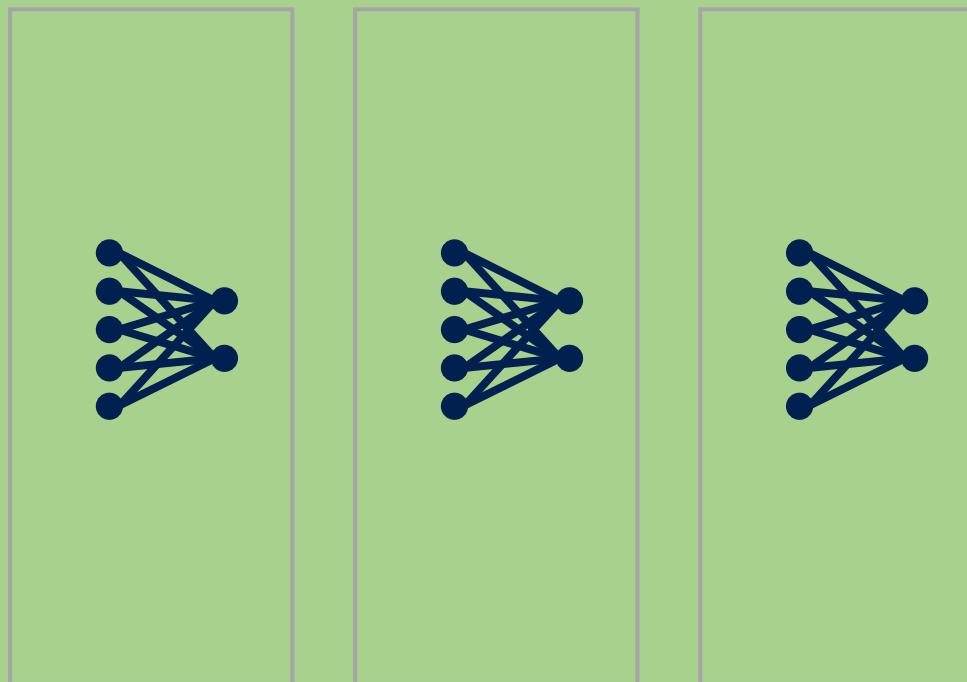
0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53



Fully connected layer

These can also be stacked.

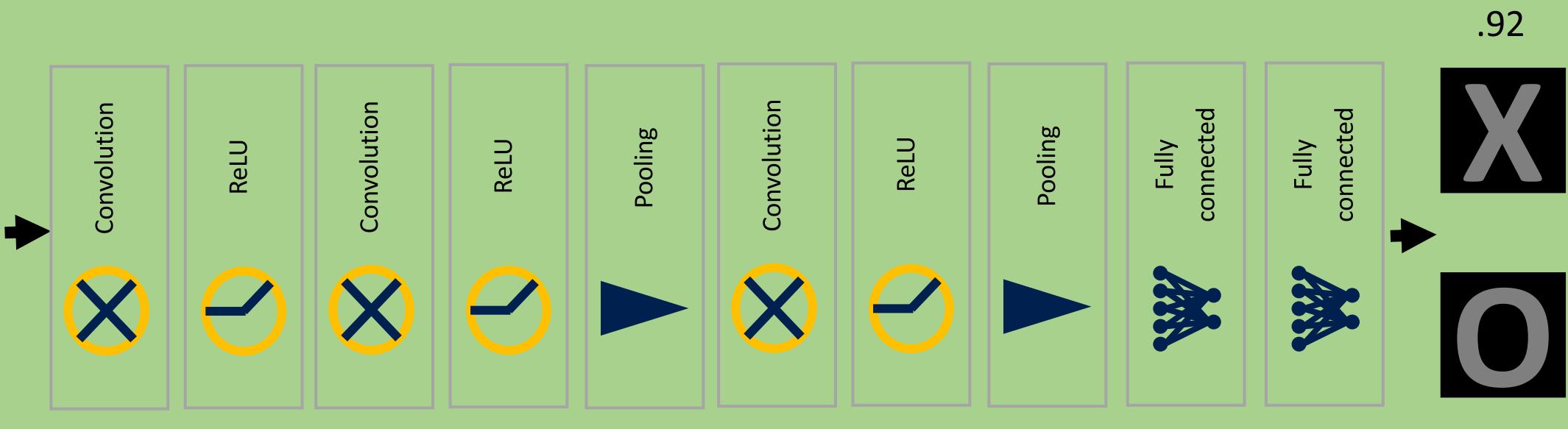
0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53



Putting it all together

A set of pixels becomes a set of votes.

```
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1  
-1 1 -1 -1 -1 -1 -1 1 -1 -1  
-1 -1 1 -1 -1 1 -1 -1 -1 -1  
-1 -1 -1 1 -1 1 -1 -1 -1 -1  
-1 -1 -1 -1 1 -1 -1 -1 -1 -1  
-1 -1 -1 1 -1 1 -1 -1 -1 -1  
-1 -1 -1 1 -1 1 -1 -1 -1 -1  
-1 -1 -1 -1 -1 1 -1 -1 -1 -1  
-1 1 -1 -1 -1 -1 1 -1 -1 -1  
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```



.92

.51

Learning

Q: Where do all the magic numbers come from?

Features in convolutional layers

Voting weights in fully connected layers

A: Backpropagation

Backprop

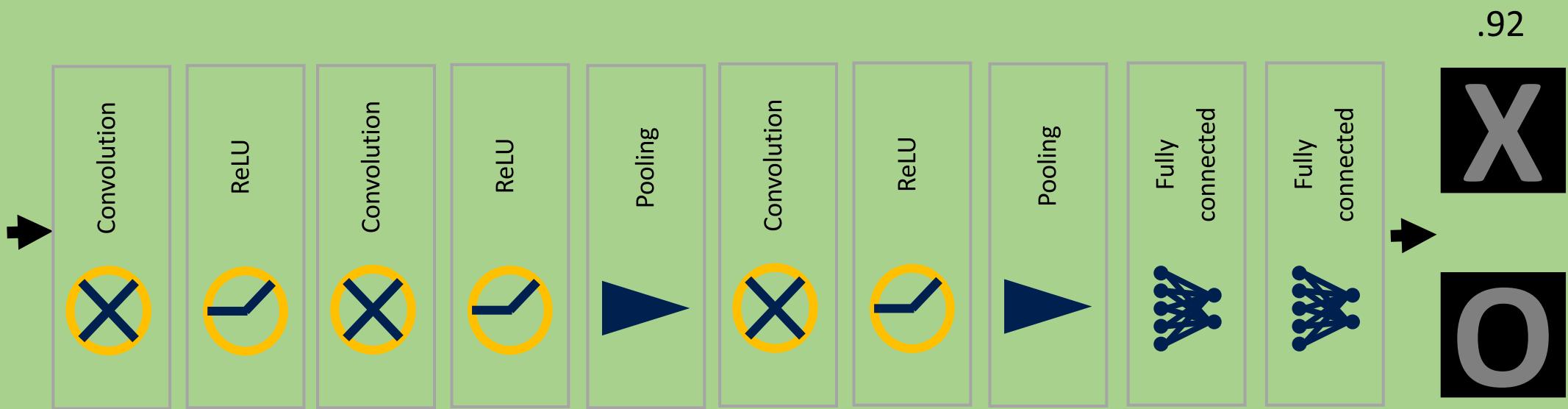
Error = right answer – actual answer



Backprop

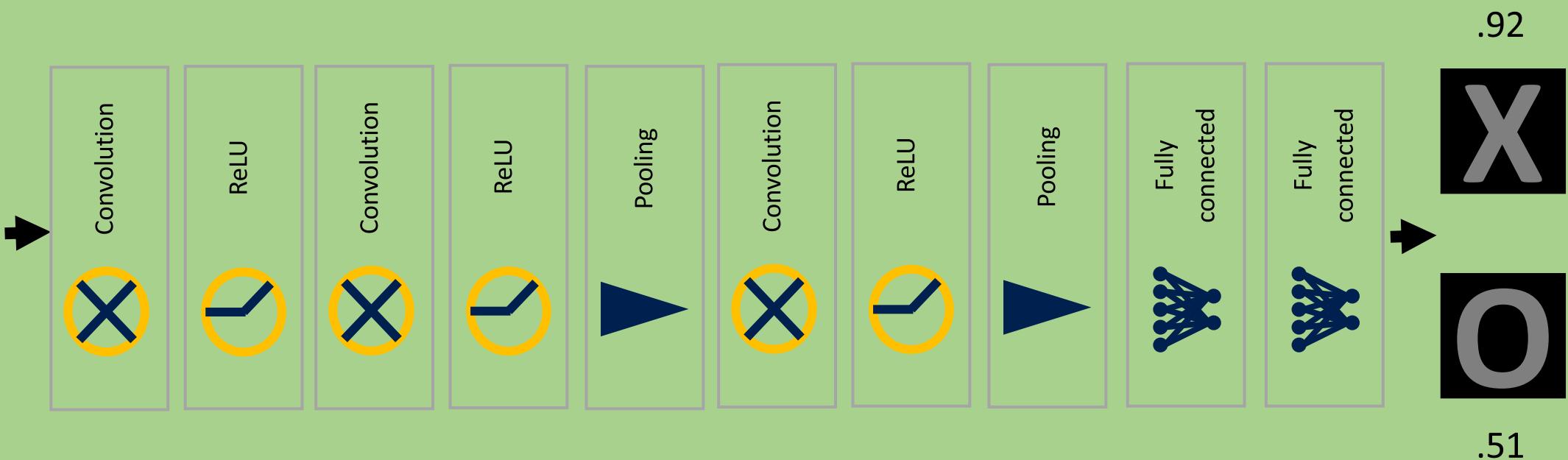
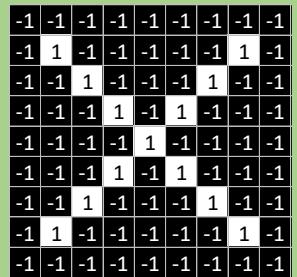
$$\begin{matrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & \textcolor{white}{1} & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{matrix}$$

	Right answer	Actual answer	Error
X	1		
O			

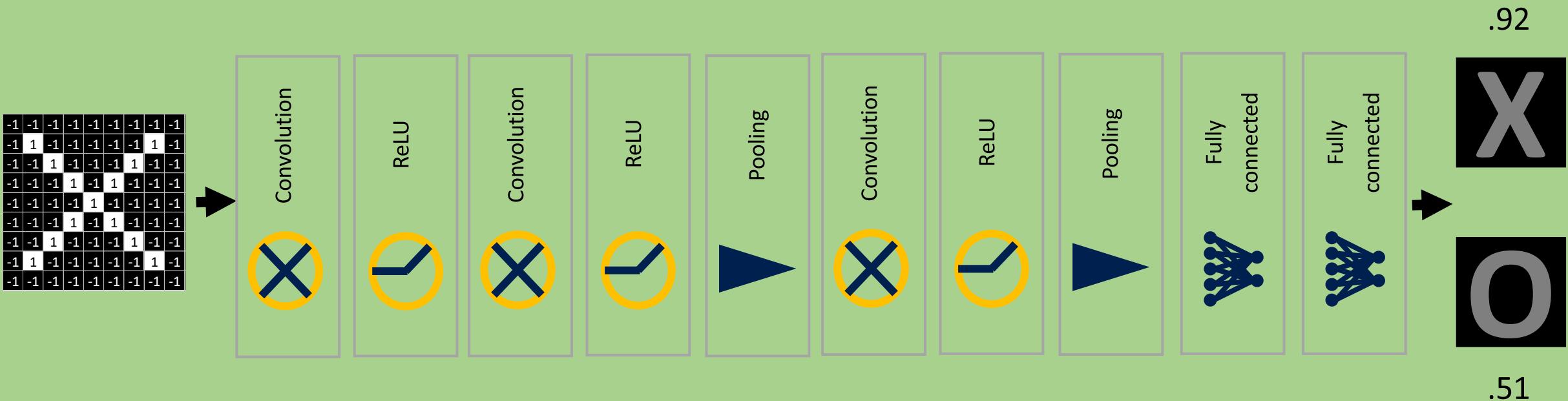


Backprop

	Right answer	Actual answer	Error
X	1	0.92	
O			

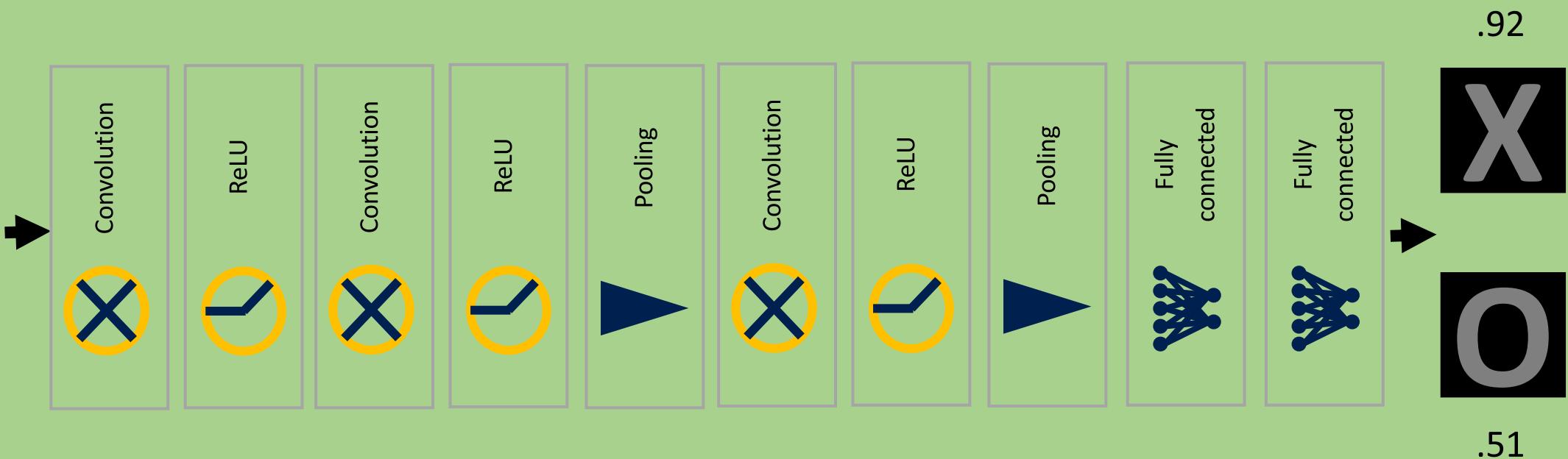
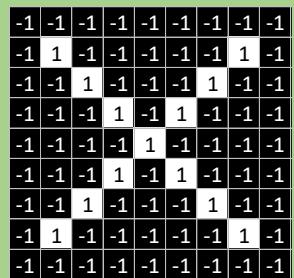


Backprop



Backprop

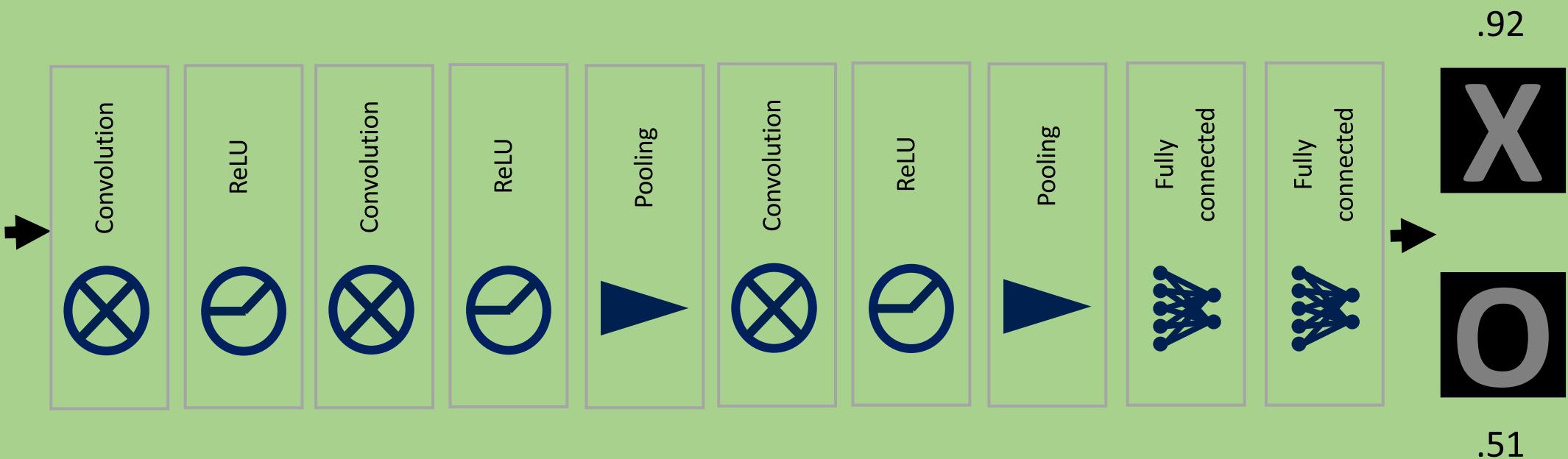
	Right answer	Actual answer	Error
X	1	0.92	0.08
O	0	0.51	0.49



Backprop

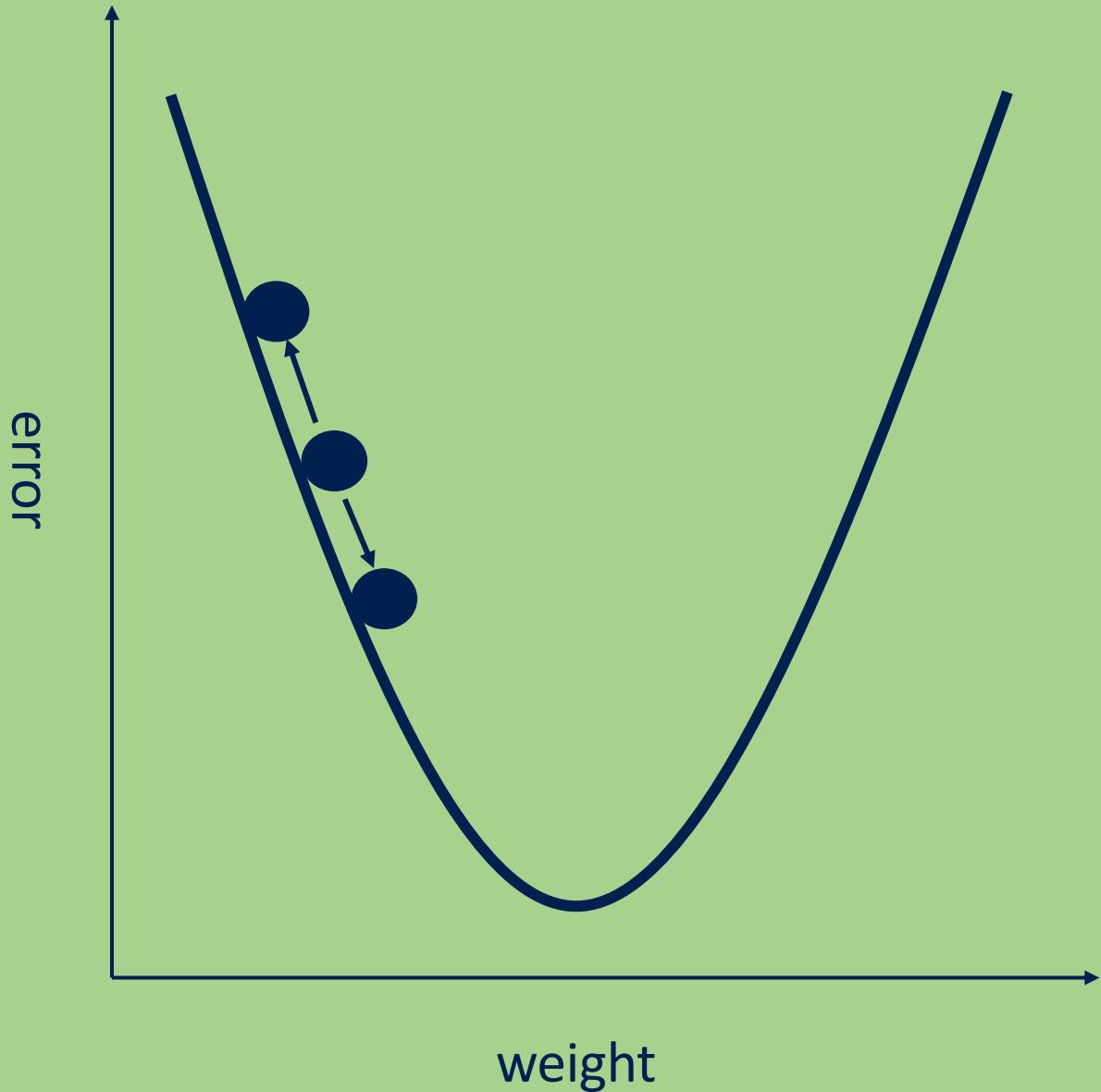
	Right answer	Actual answer	Error
X	1	0.92	0.08
O	0	0.51	0.49
Total			0.57

`-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1`



Gradient descent

For each input feature and voting weight, adjust it up and down a bit and see how the error changes.



CNN slides from:

[Brandon Rohrer](#)

[@ brohrer on Twitter](#)

#HowItWorks #ConvNet

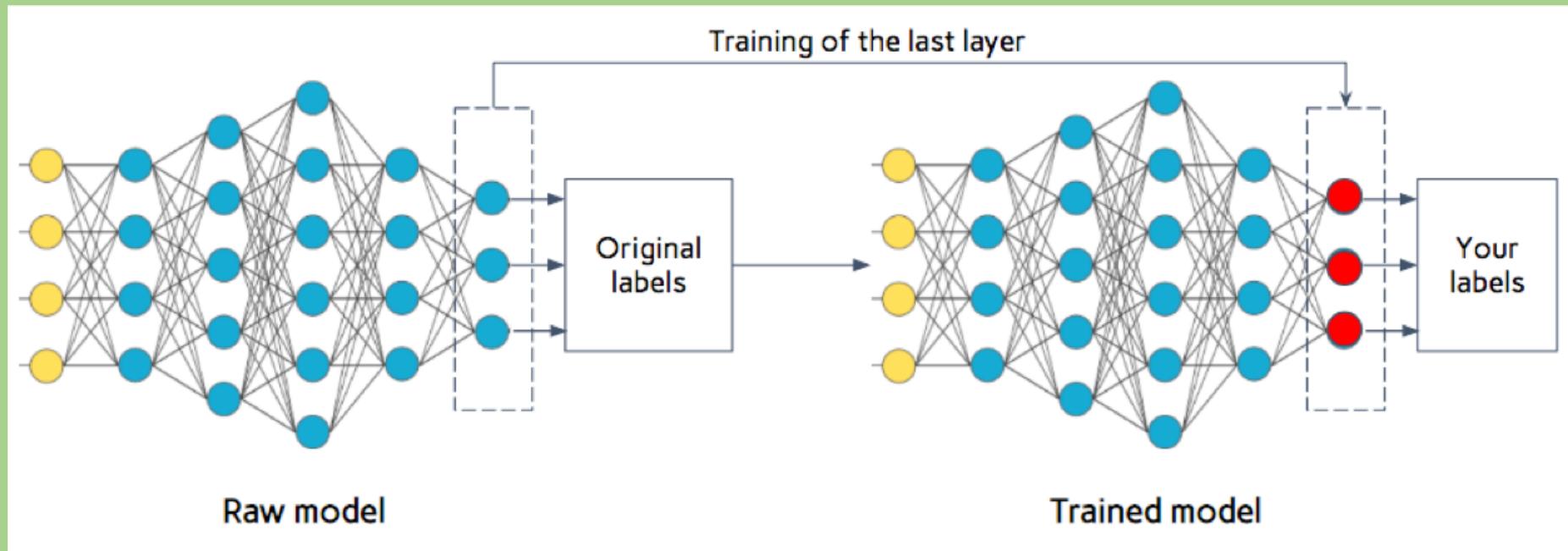
brohrer@microsoft.com



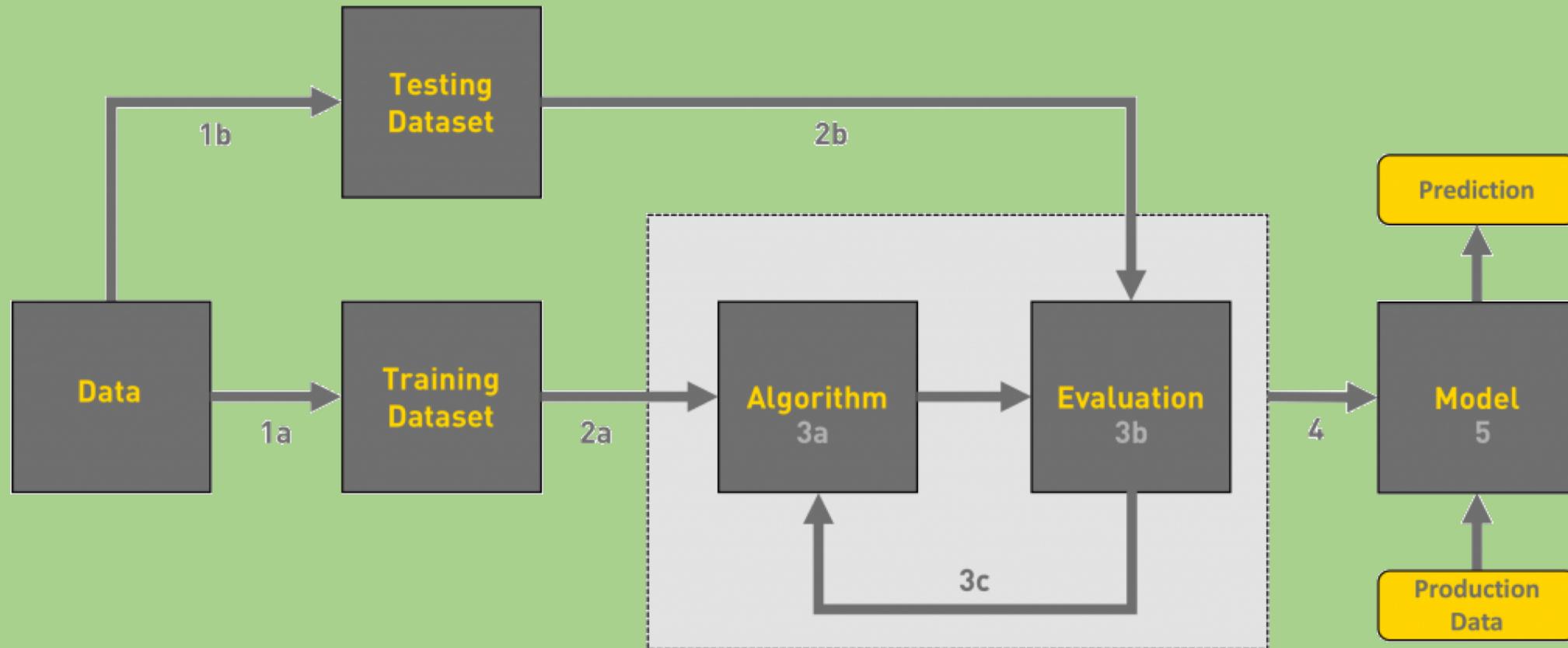
Difficulties with real data

- Data hungry
- Garbage in = Garbage out
 - Need to monitor training data quality
 - Overfitting
 - Bias

Fine Tuning – dealing with small data



ML Workflow



Cloud Architecture

Create

Frameworks

Caffe2 Chainer Cognitive Toolkit

mxnet PyTorch PaddlePaddle

ML.NET MathWorks® XGBoost

M scikit-learn TensorFlow Keras

Services

Azure Custom Vision Service

Native support

Converters

Native support

ONNX Model



Deploy

Azure

Azure Machine Learning services
Ubuntu VM
Windows Server 2019 VM

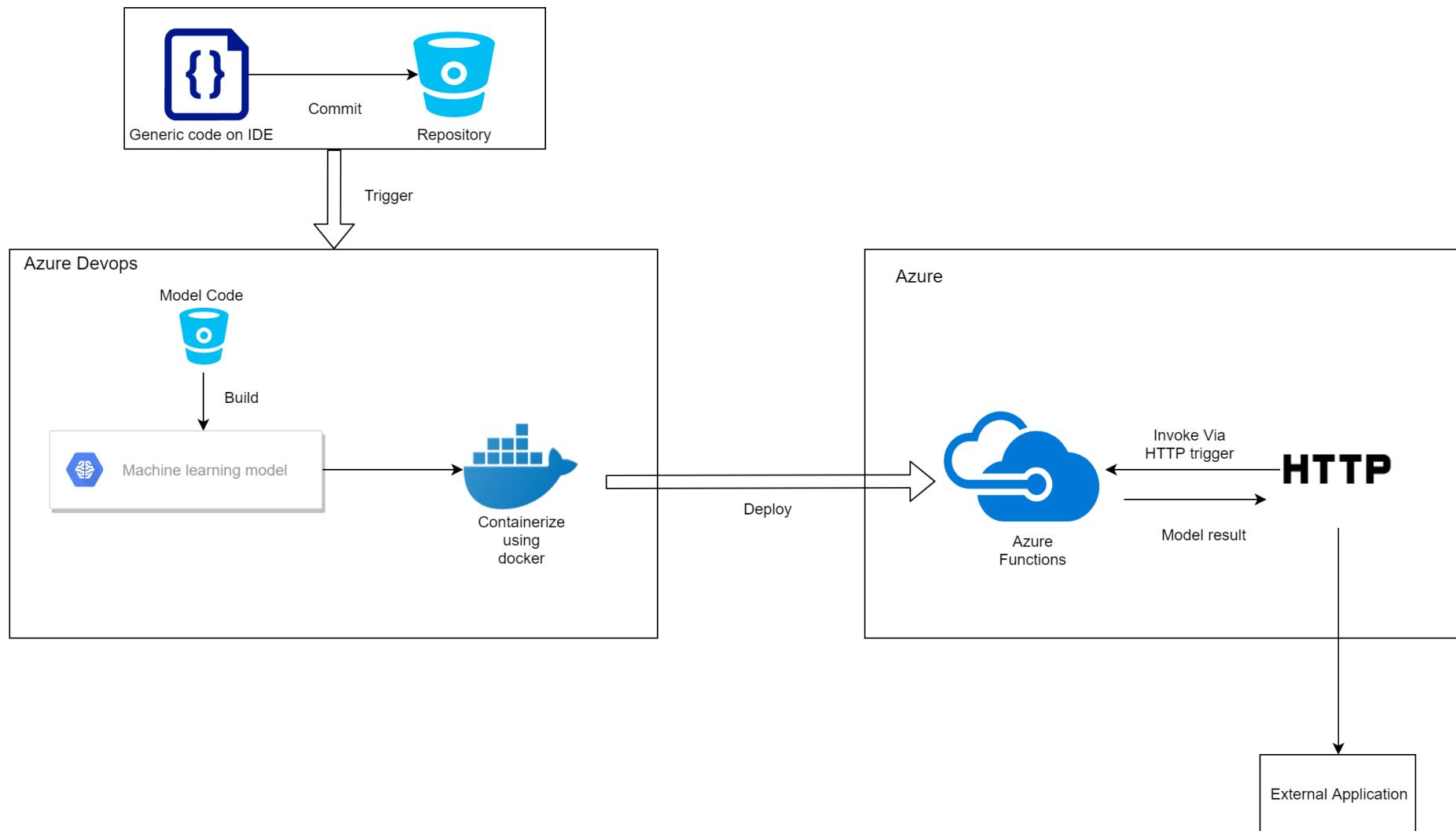
Windows Devices

Native support

Converters

Other Devices (iOS, etc)

Alternative Cloud Architecture



Examples

- Process Automation
 - Data entry, Insurance Risk Assessment
- Sales Optimisation
 - Sales forecasting, churn predictions
- Customer Service
 - Chatbots, VA
- Security and Fraud
 - Predict Analytics vs Behavioural Analytics



Resources

- Youtube
 - <http://3b1b.co/neural-networks>
 - https://www.youtube.com/playlist?list=PLZbbT5o_s2xq7Lwl2y8_QtvuXZedL6tQU
 - https://www.youtube.com/watch?v=JB8T_zN7ZC0
 - ... + a LOT more
- MOOCs
 - fast.ai
 - deeplearning.ai
 - <http://cs231n.stanford.edu>
- Tensorflow Transfer Learning tutorial
 - https://www.tensorflow.org/beta/tutorials/images/transfer_learning
- Free GPU in the cloud
 - <https://colab.research.google.com/>

Resources

- Azure Functions
 - <https://azure.microsoft.com/en-us/resources/videos/ignite-2018-machine-learning-using-python-in-azure-functions/>
 - <https://azure.microsoft.com/en-au/resources/samples/functions-python-tensorflow-tutorial/>

