Introduction to Deep Learning

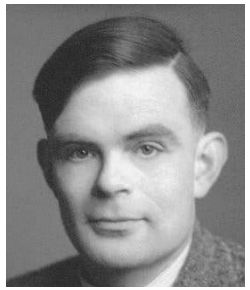
Exploring Deep Neural Networks: A Beginner's Guide

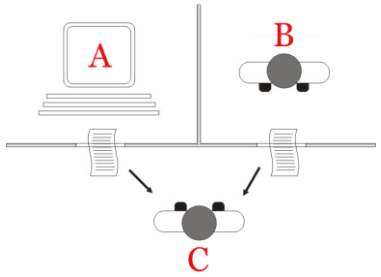
Feng Chen
HPC User Services
LSU HPC & LONI
sys-help@loni.org

Louisiana State University
Baton Rouge

Parts of slides referenced from
Nvidia, ***Deep Learning Institute (DLI) Teaching Kit***
Stanford, ***CS231n: Convolutional Neural Networks for Visual Recognition***
Martin Görner, ***Learn TensorFlow and deep learning, without a Ph.D***
Luis Serrano, ***A friendly introduction to Convolutional Neural Networks and Image Recognition***

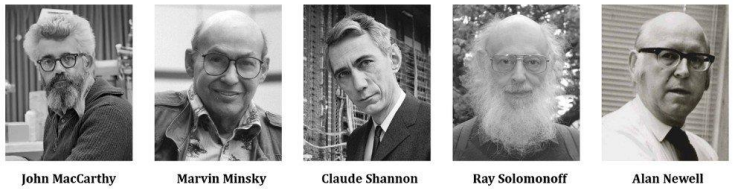
History of AI





1950, Alan Turing proposes the Turing test as a measure of machine intelligence.

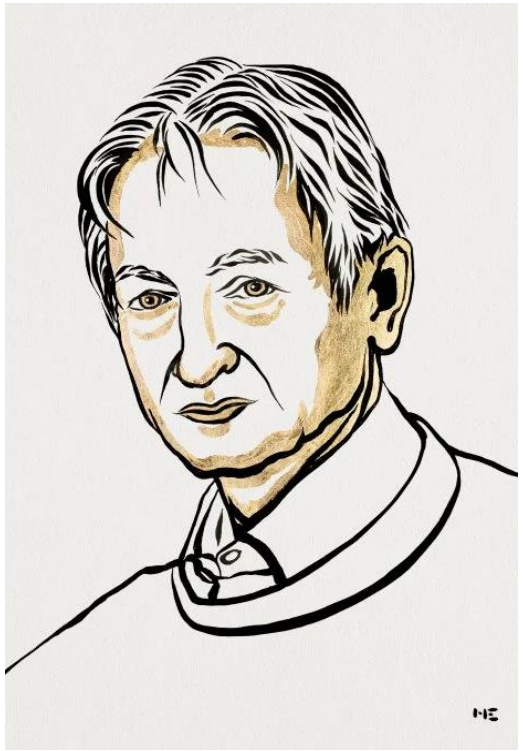
1956 Dartmouth Conference: The Founding Fathers of AI



1956, The Dartmouth College summer AI conference is organized by John McCarthy, Marvin Minsky, Nathan Rochester of IBM and Claude Shannon. McCarthy coins the term artificial intelligence for the conference.

1997, The Deep Blue chess machine (IBM) defeats the (then) world chess champion, Garry Kasparov.

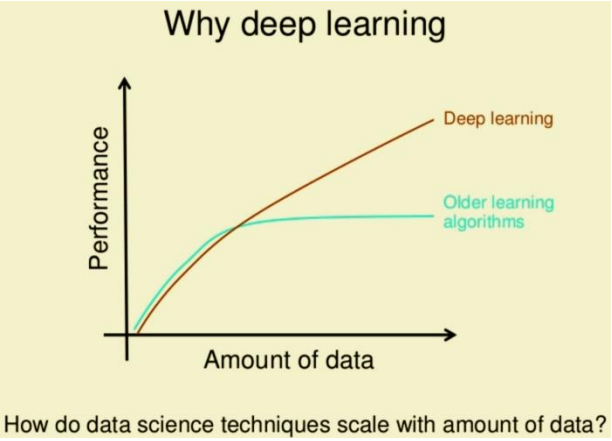
2018, Turing Award
Geoffrey Hinton, Yann LeCun, Yoshua Bengio



Geoffrey E. Hinton
The Nobel Prize in Physics 2024
Born: 6 December 1947, London, United Kingdom
Affiliation at the time of the award: University of Toronto, Toronto, Canada
Prize motivation: "for foundational discoveries and inventions that enable machine learning with artificial neural networks"

Why Deep Learning?

- **Supremacy in terms of accuracy when trained with huge amount of data.**



https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063

- **LeNet-5, a pioneering 7-level convolutional network by LeCun et al in 1998**
- **Popularized by AlexNet (by Alex Krizhevsky) in 2012**
- **Advancement and usage in GPUs (Graphic Processing Unit)**
- **Not limited to just image-based tasks.**

Applications of Machine Learning

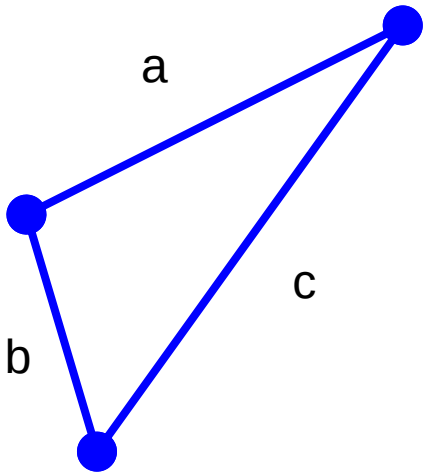
- **Computer Vision (CV)**
  - Image Classification
    - label images with appropriate categories (e.g. Google Photos)
  - Handwriting Recognition
    - convert written letters into digital letters
- **Natural Language Processing (NLP)**
  - Language Translation
    - translate spoken and or written languages (e.g. Google Translate)
  - Speech Recognition
    - convert voice snippets to text (e.g. Siri, Cortana, and Alexa)
- **Autonomous Driving**
  - enable cars to drive

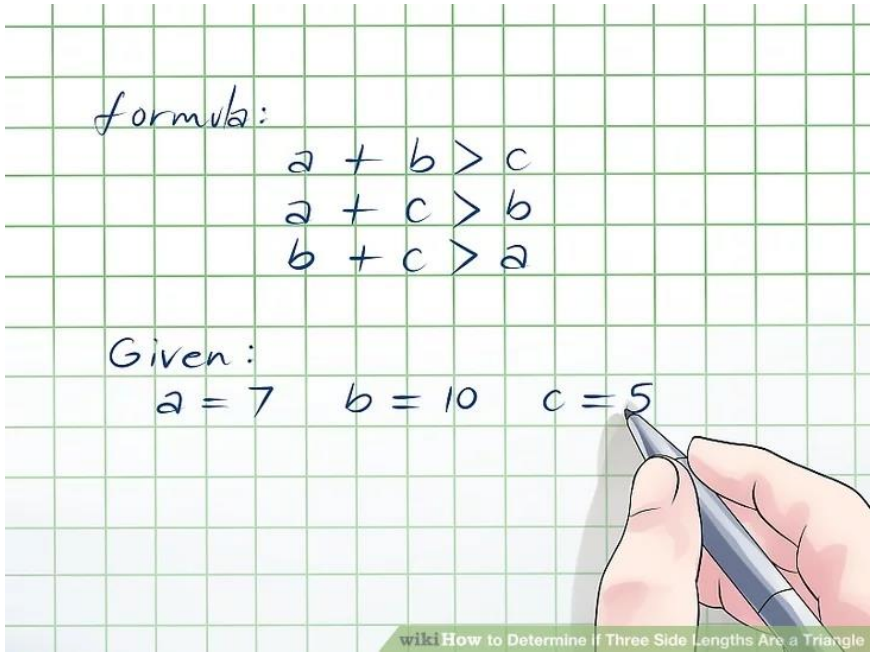
Topics To Be Discussed

- **Fundamentals about Machine Learning**
  - What is ***Deep Learning***?
    - What is a (deep) neural network
    - How to train it
  - Deep Learning Frameworks (Software package)

- **Build a neural network model using Pytorch**
  - MNIST example
    - Softmax classification
    - Cross-entropy cost function
    - Structure of the deep neural network
    - Convolutional networks

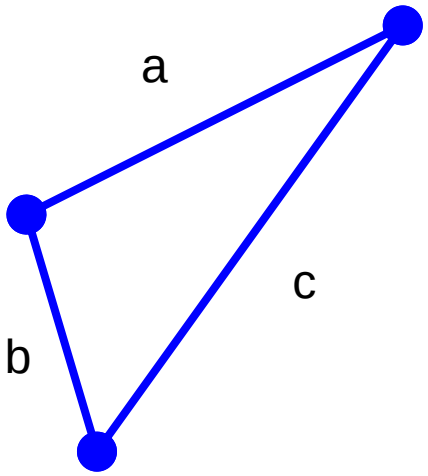
How to recognize a triangle?

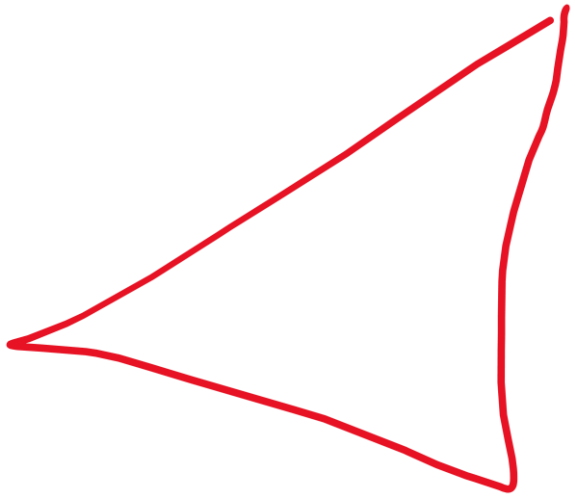




Given the lengths of three sides, a, b, and c, determine if they can form a triangle.

How to recognize a triangle?



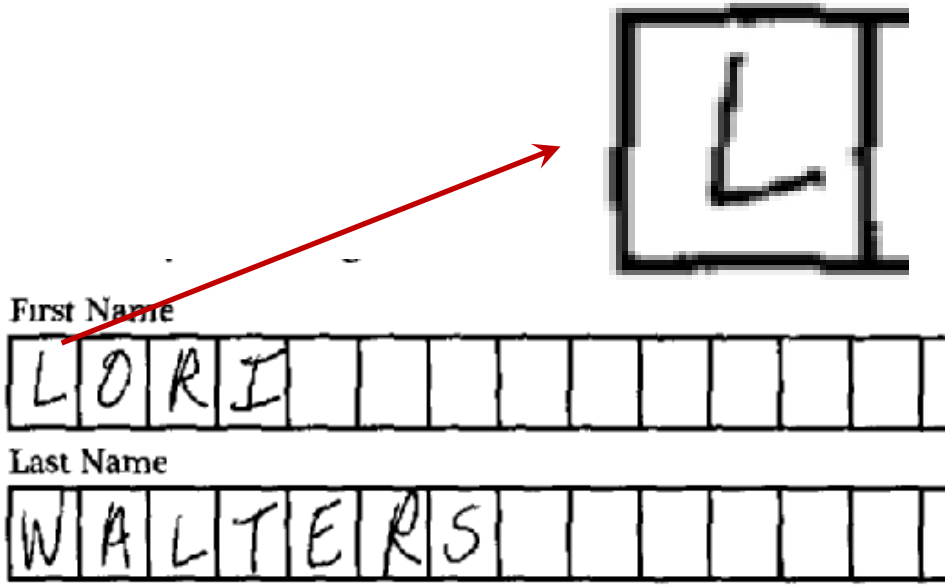


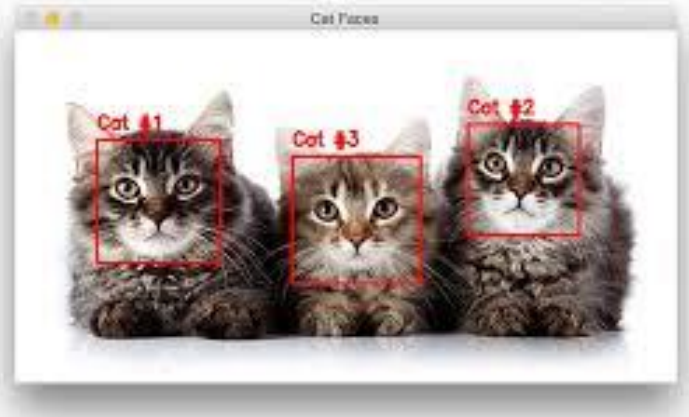
How about this?

Machine Learning

- **Machine Learning is the science of getting computers to learn, without being explicitly programmed.**

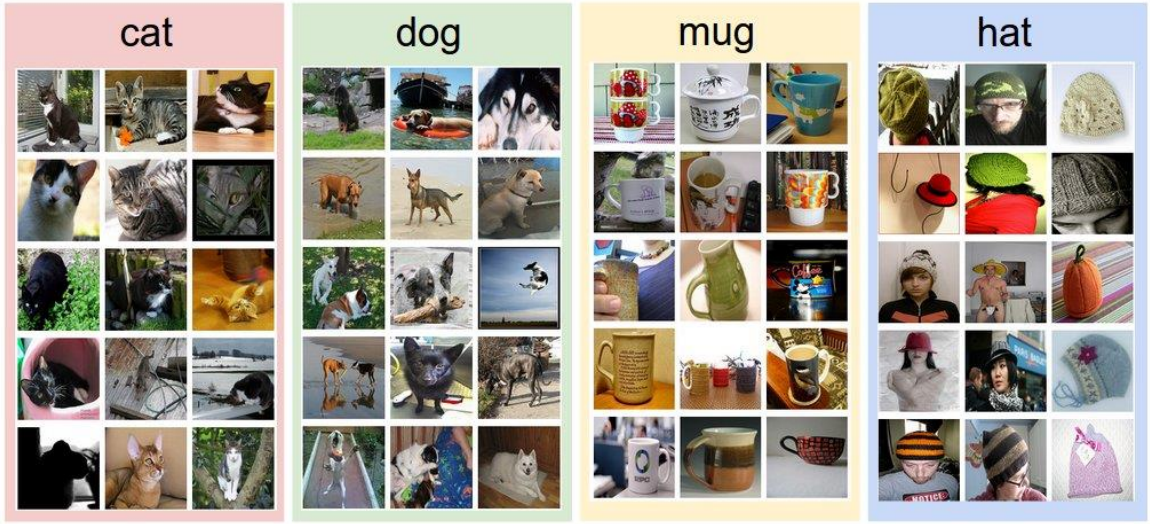
- **Examples are used to train computers to perform tasks that would be difficult to program**





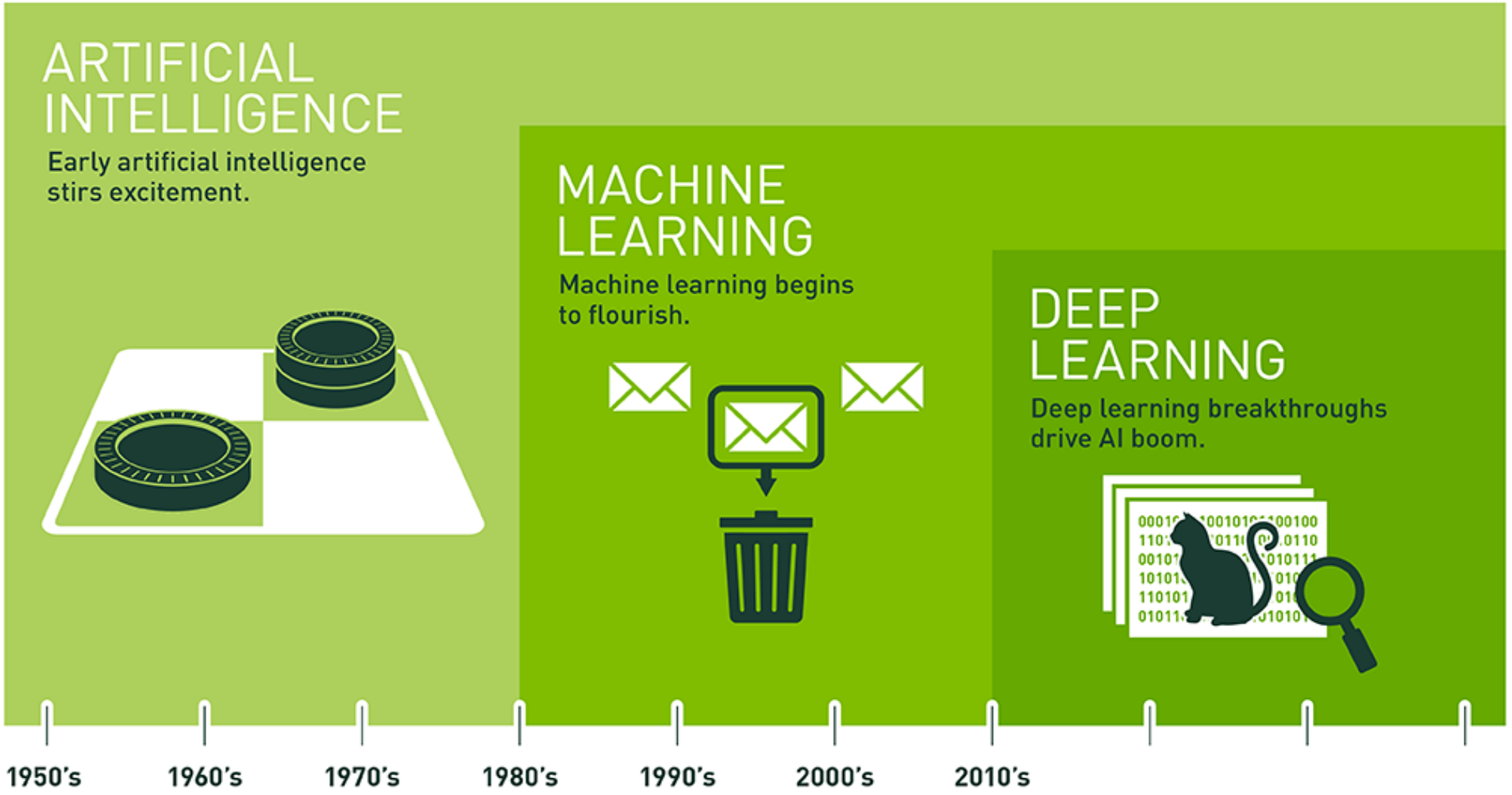
Data-driven Approach

- **Instead of trying to specify what every one of the categories of interest look like directly in code, the approach that we will take is not unlike one you would take with a child:**
  - Provide the computer with many examples of each class
  - Develop learning algorithms that look at these examples and learn about the visual appearance of each class.
- **This approach is referred to as a data-driven approach.**



An example training set for four visual categories. In practice we may have thousands of categories and hundreds of thousands of images for each category. ***(From Stanford *CS231n*)**

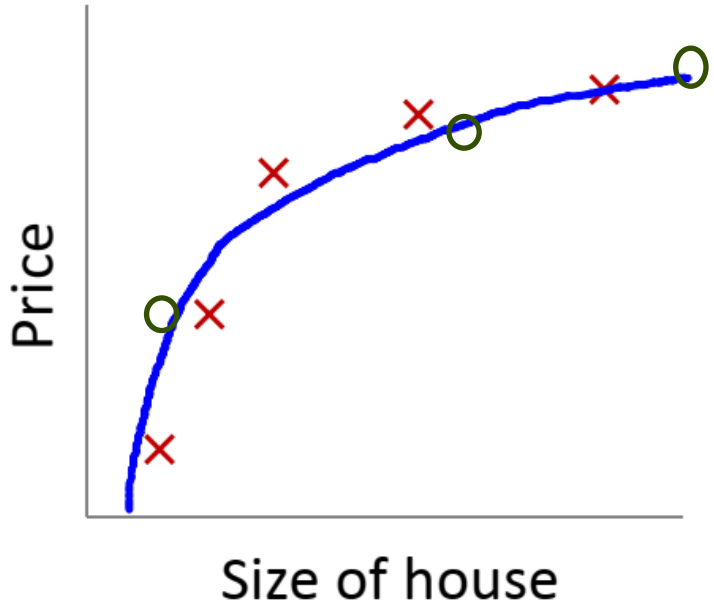
AI, Machine Learning and Deep Learning



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Simplest Case of Machine Learning

E.g.: Price vs Size of house



X - Training Data
O - Test Data

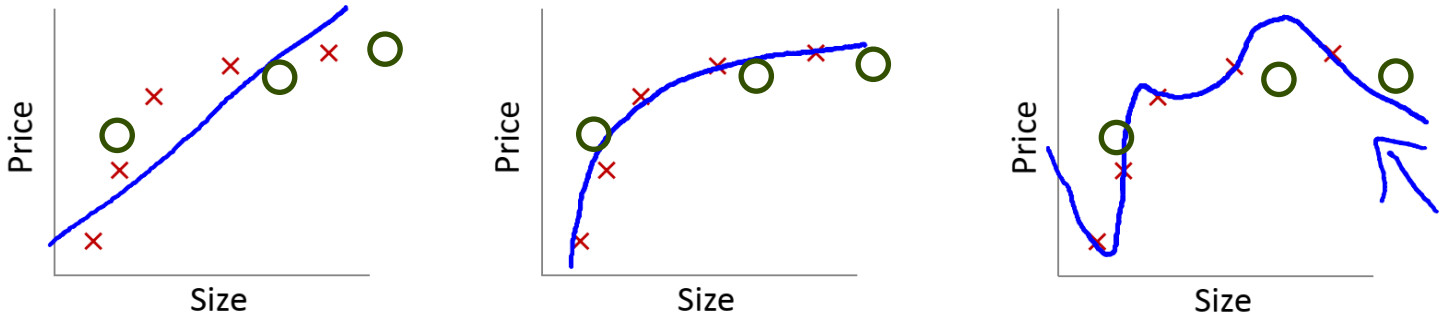
- **In machine learning, datasets are split into two subsets:**
  - ***Training Data*** (Training set)
    - data used to learn a model
  - ***Test Data*** (Test set)
    - data used to assess the accuracy of model

Simplest Case of Machine Learning

Slide from Andrew Ng, Machine Learning, Lecture 7

E.g.: Price vs Size of house

X - Training Data
O - Test Data



Underfitting

"Just Right"

Overfitting

- **Underfitting:**
  - Model is unable to capture the relationship between the input and output variables accurately.
- **Overfitting:**
  - Model performs well on training data but poorly on test data

Types of Machine Learning

- **Supervised Learning**
  - Training data is labeled
  - Goal is correctly labelling new data
- **Unsupervised Learning**
  - Training data is unlabeled
  - Goal is to categorize the observations
- **Reinforcement Learning**
  - Training data is unlabeled
  - System receives feedback for its actions
  - Goal is to perform better actions

Supervised Learning Algorithms

- **Linear Regression**
- **Neural Networks**
  - Deep Learning is the branch of Machine Learning based on Deep Neural Networks (DNNs, i.e., neural networks composed of more than 1 hidden layer).
  - Convolutional Neural Networks (CNNs) are one of the most popular DNN architectures (so CNNs are part of Deep Learning), but by no means the only one.
- **Decision Trees**
- **Support Vector Machines**
- **K-Nearest Neighbor**

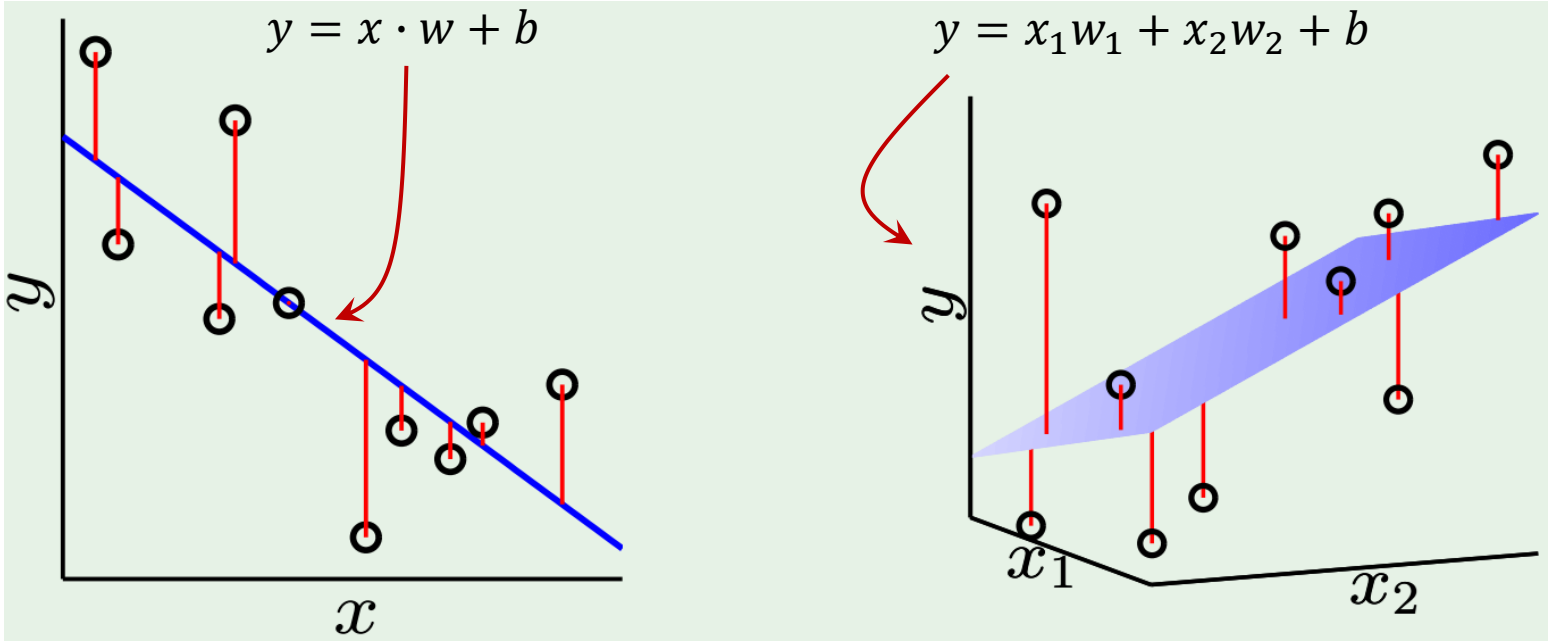
Machine Learning Frameworks

- **TensorFlow:**
  - Developed by Google, it's widely used for deep learning applications.
- ***PyTorch:***
  - A deep learning framework developed by Facebook, known for its flexibility and dynamic computation graph.
- **Keras:**
  - An easy-to-use high-level neural network API, written in Python, that runs on top of TensorFlow.
- **Scikit-Learn:**
  - A Python library for classical machine learning algorithms, popular for its ease of use in data mining and data analysis.
- **MXNet:**
  - Supported by Amazon, it's known for its efficiency and used for deep learning, particularly in large-scale cloud environments.
- **Caffe:**
  - Developed by the Berkeley Vision and Learning Center (BVLC), it's often used in image classification and convolutional neural networks (CNNs).

Machine Learning and Deep Learning

Recall From The Least Square Method

- **Start from least square method...**
- **Trying to find**
  - ***Parameters* (*w, b*)**: minimizes the sum of the squares of the errors
  - ***Errors***: distance between known data points and predictions



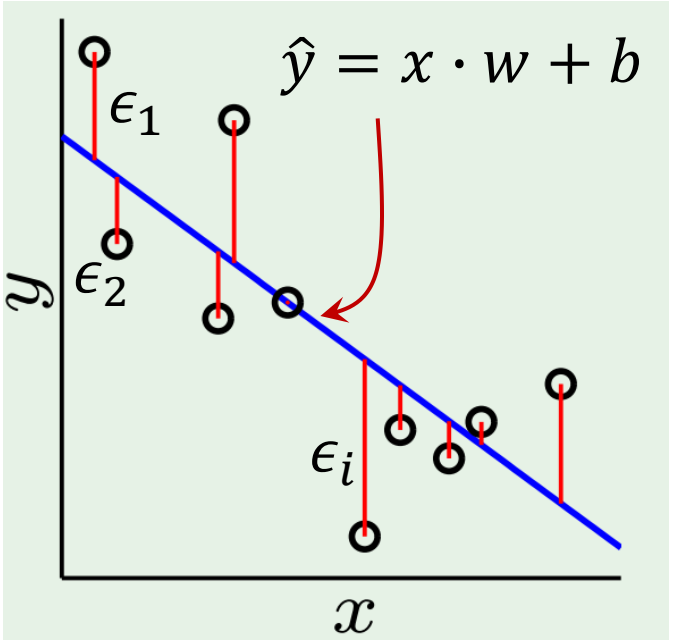
❖ from Yaser Abu-Mustafa “Learning From Data” Lecture 3

Recall Least Square Method - How to?

- **Calculate the sum of square of errors (residual)**

$$E = \sum \epsilon_i^2 = \sum (y_i - \hat{y}_i)^2$$

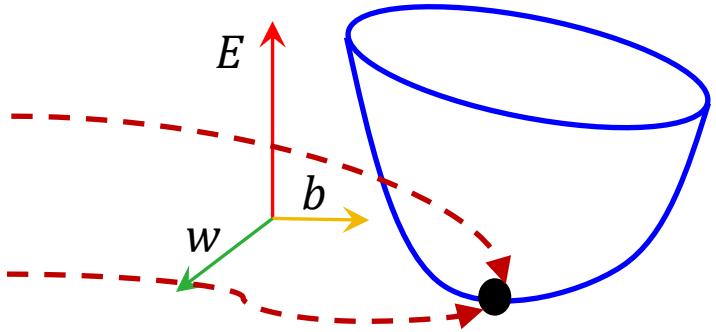
$$= \Sigma \left(y_i - (wx_i + b)\right)^2$$



- **Minimize the error function**
  - How to minimize the error function?

$$\frac{\partial E}{\partial w} = 0$$

$$\frac{\partial E}{\partial b} = 0$$



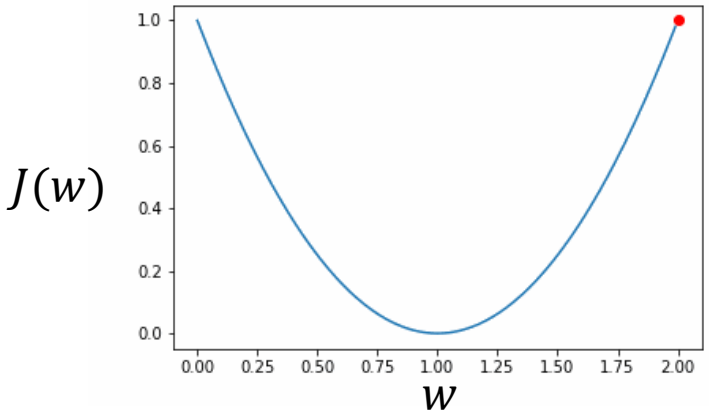
  - We will then get the expression of w and b

$$w = w(x_i, y_i)$$

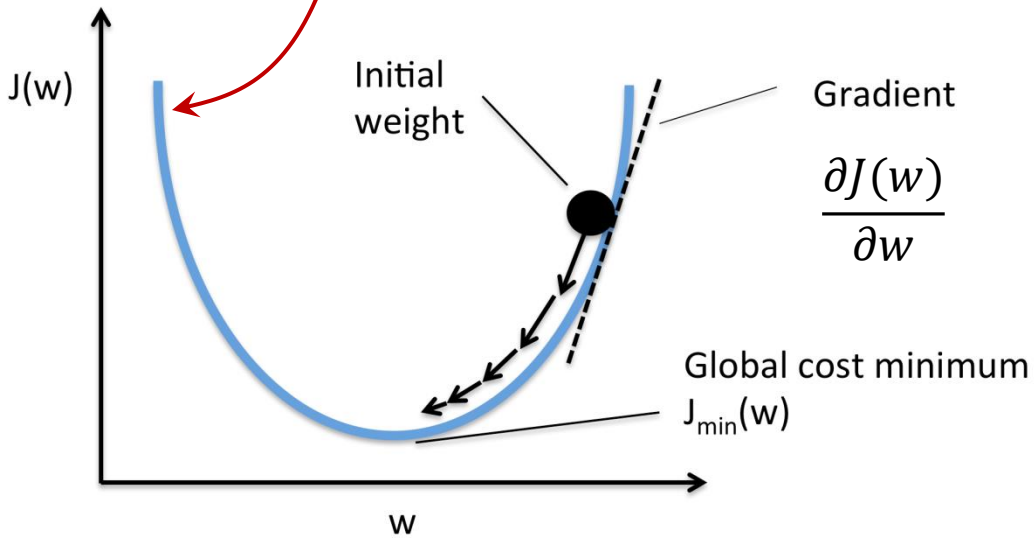
$$b = b(x_i, y_i)$$

To The Machine Learning Language

- **Error (*E*)**
  - Cost Function (Loss): *J*(*w*), *C*, *L*
- **Parameters**
  - Weights and Biases: (*w*, *b*)

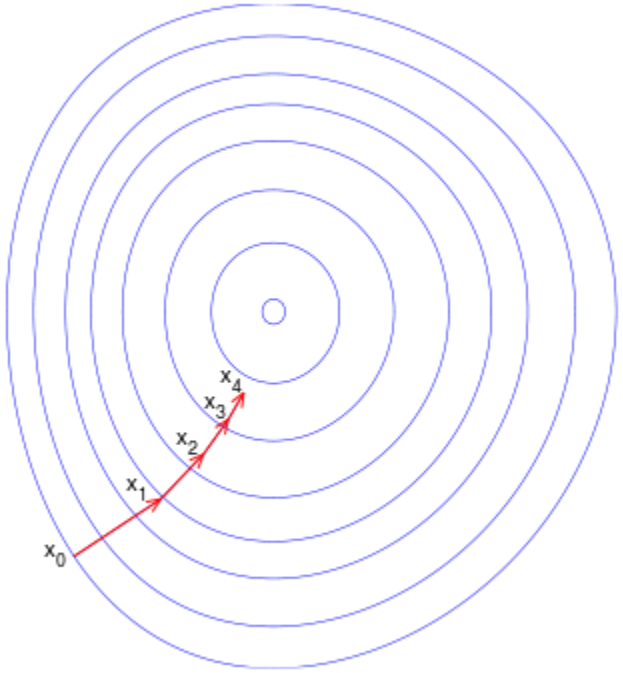


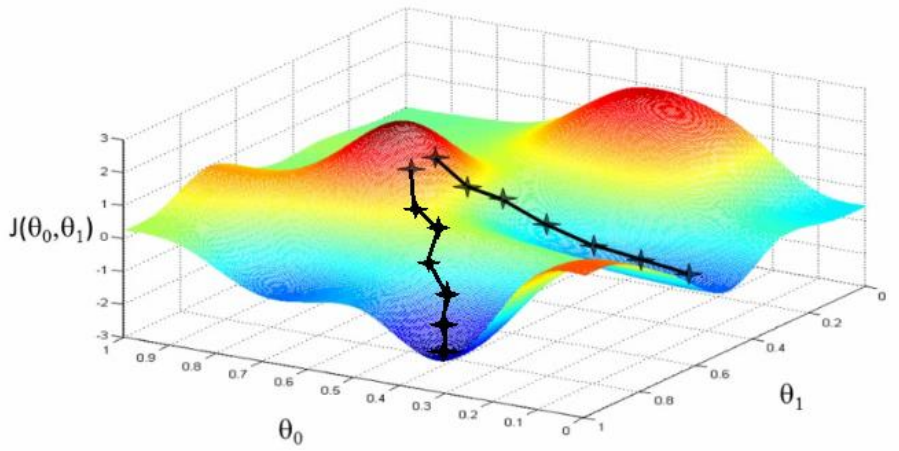
- **Define the cost function of your problem**
- **Find the set of weights that minimizes the cost function (loss)**



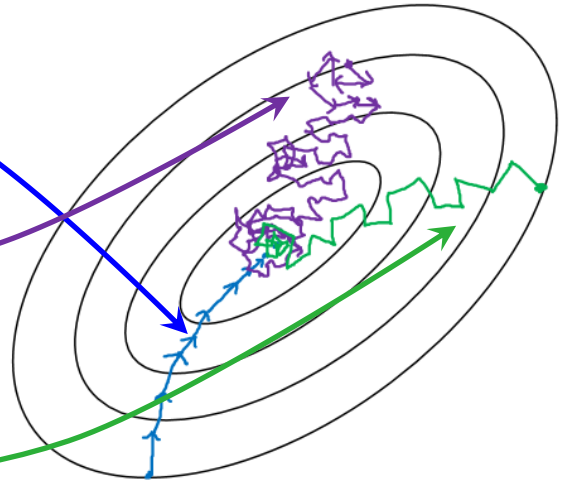
Theory: Gradient Descent

- **Gradient descent is a first-order iterative optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point.**





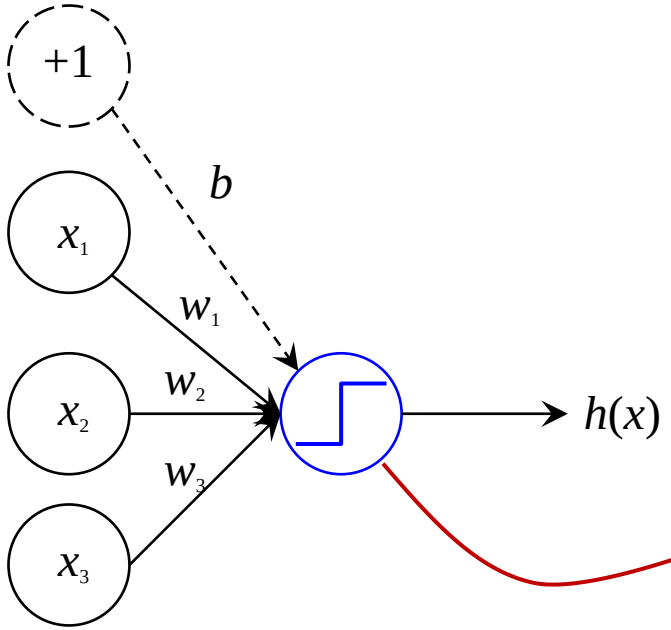
Mini-batch Gradient Descent



- **Batch gradient descent:**
  - Use all examples in each iteration
- **Stochastic gradient descent:**
  - Use one example in each iteration
- **Mini-batch gradient descent**
  - Splits the training dataset into small batches (size *b*) that are used to calculate model error and update model coefficients.
- **In the neural network terminology:**
  - One ***epoch*** consists of one full training cycle on the training set.
  - Using all your batches once is 1 ***epoch***. If you have 10 epochs, it mean that you will use all your data 10 times (split in batches).

What is a Neural Network?

- **Start from a perceptron**



x1	pixel1	0
x2	pixel2	120
x3	pixel3	255
b	threshold	some value

h(x)	Alert, if: `h(x)>0`

Activation function:

$$\sigma(z)=sign(z)$$

Denote as: Z

Feature vector: $\mathbf{x}$ Weight vector: $\mathbf{w}$

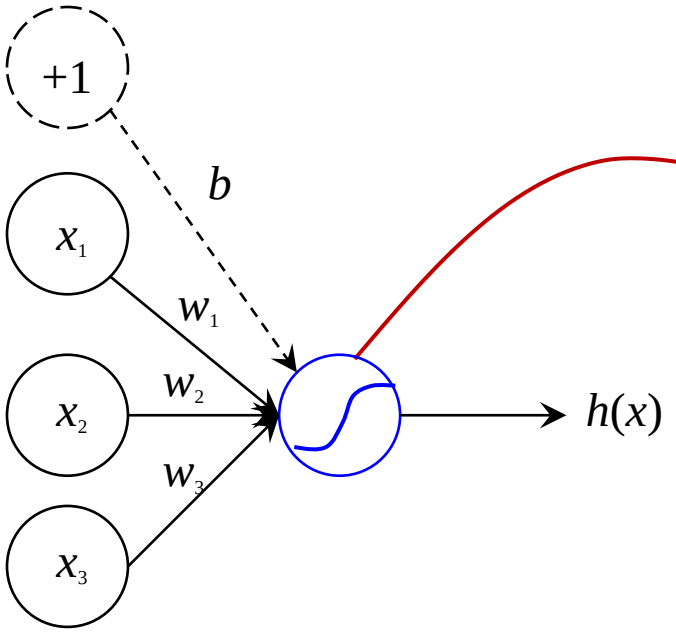
$$\mathbf{x}=\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} \quad \mathbf{w}=\begin{bmatrix}w_1\\w_2\\w_3\end{bmatrix}$$

$$h(x) = \text{sign}(w_1x_1 + w_2x_2 + w_3x_3 + b)$$
$$= \text{sign}\left(\sum_i w_ix_i + b\right)$$
$$= \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Hypothesis (Prediction: y)

Perceptron To Neuron

- **Replace the sign to sigmoid**



Activation function:

$\sigma(z)=sigmoid(z)$

$$h(x) = \text{sigmoid}(w_1x_1 + w_2x_2 + w_3x_3 + b)$$
$$= \text{sigmoid}\left(\sum_i w_i x_i + b\right)$$
$$= \text{sigmoid}(\mathbf{w}^T\mathbf{x} + b)$$

Feature vector: $\mathbf{x}$ Weight vector: $\mathbf{w}$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

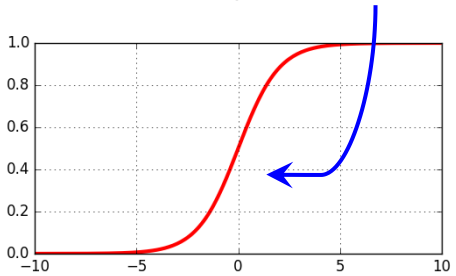
$$z = \mathbf{w}^T\mathbf{x} + b$$

$$y = h(x) = \sigma(z)$$

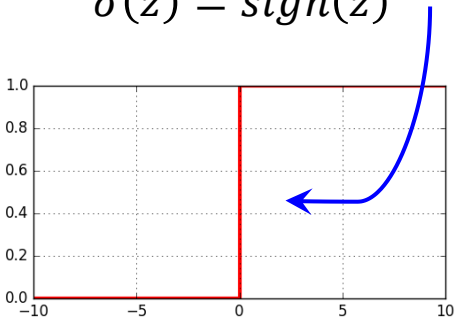
Sigmoid Neurons

- **Sigmoid activation Function**
  - In the field of Artificial Neural Networks, the sigmoid function is a type of activation function for artificial neurons.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

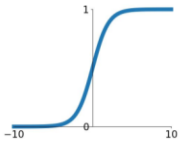


$$\sigma(z) = sign(z)$$

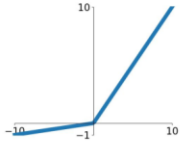


- **There are many other activation functions. (We will touch later.)**

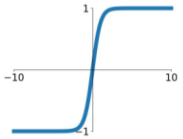
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

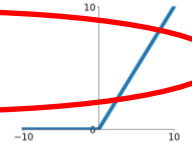


tanh
$\tanh(x)$



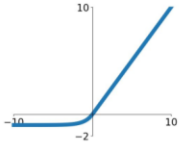
Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$



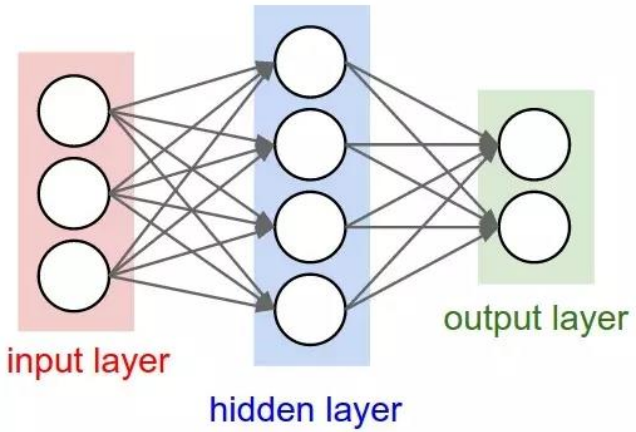
ELU

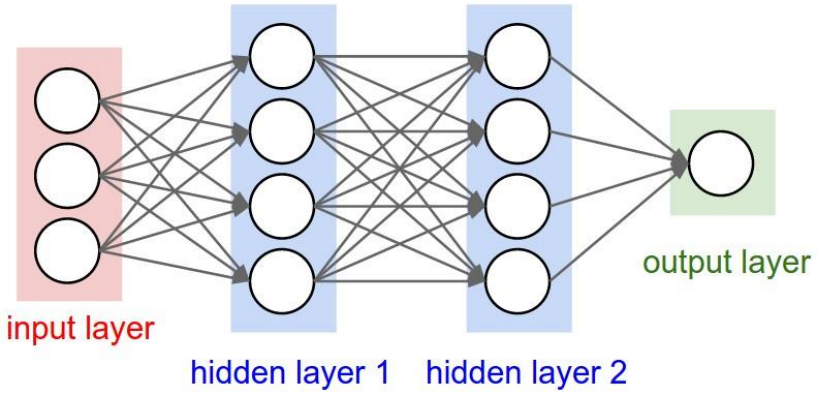
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



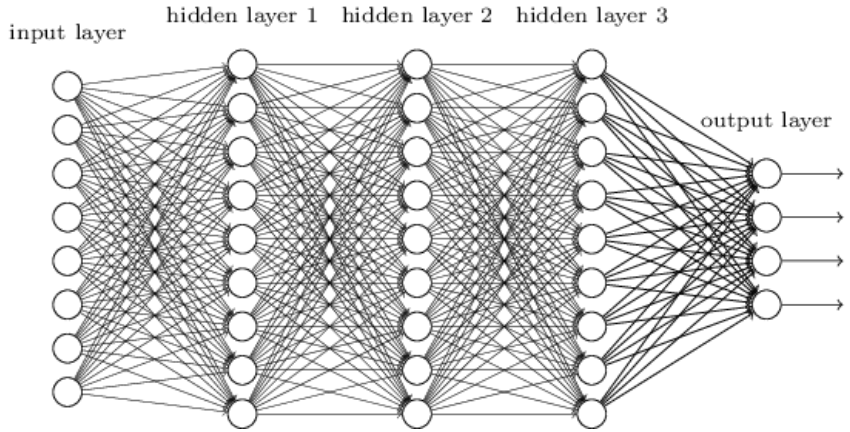
Network Of Neurons

- **A complex network of neurons could make quite subtle decisions**





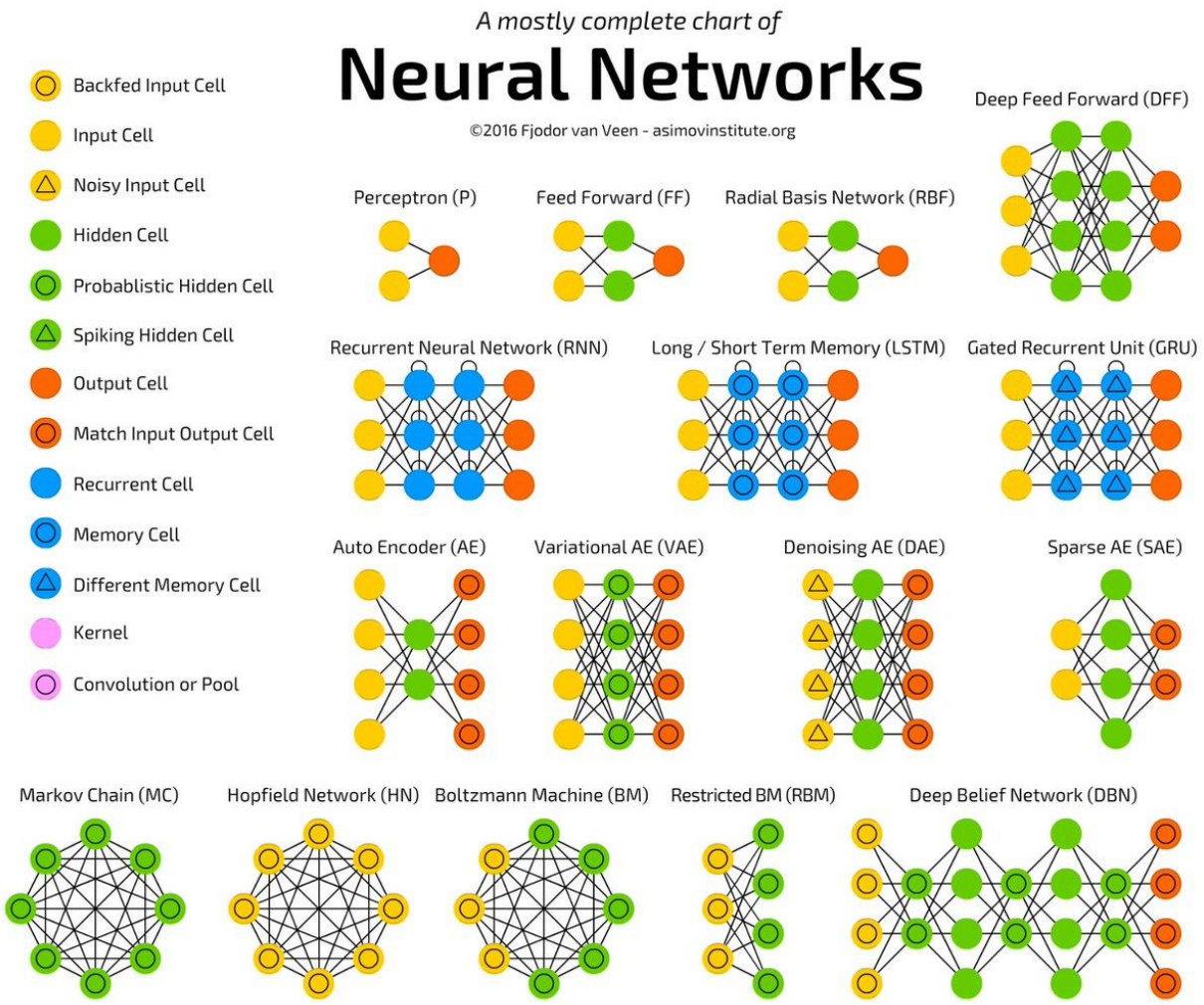
- **Deep Neuron Network: Number of hidden layers >1**



Rise and Fall of Neural Networks

- **Began gaining traction in the 1940s with the development of the perceptron. Early enthusiasm was high, but limitations such as the inability to solve non-linearly separable problems, highlighted by Marvin Minsky and Seymour Papert in their 1969 book "Perceptrons"**
  - Led to a period of decline in interest and funding, known as the AI winter of the 1970s and 1980s.
- **Resurgence (1980s - 1990s)**
  - Backpropagation Algorithm Rediscovery (1986, David Rumelhart, Geoffrey Hinton, and Ronald Williams popularized the backpropagation algorithm).
  - Renewed Interest: Backpropagation revived interest in neural networks by demonstrating that they could be trained to perform complex tasks, including pattern recognition and data classification.
- **Second Decline (Late 1990s - Early 2000s)**
  - Competition from other methods, e.g., Support Vector Machines (SVMs):
  - Overfitting and generalization Issues
  - Challenges in scaling and training challenges.
- **The resurgence of neural networks in the 2000s, often referred to as the deep learning revolution, was fueled by advances in computational power (especially GPUs).**
  - AlexNet (2012), Transformer (2017), Bert (2018)

Types of Neural Networks



Ref: http://www.asimovinstitute.org/neural-network-zoo/

How to Train DNN?

- **Backward Propagation**
  - The backward propagation of errors or backpropagation, is a common method of training artificial neural networks and used in conjunction with an optimization method such as gradient descent.

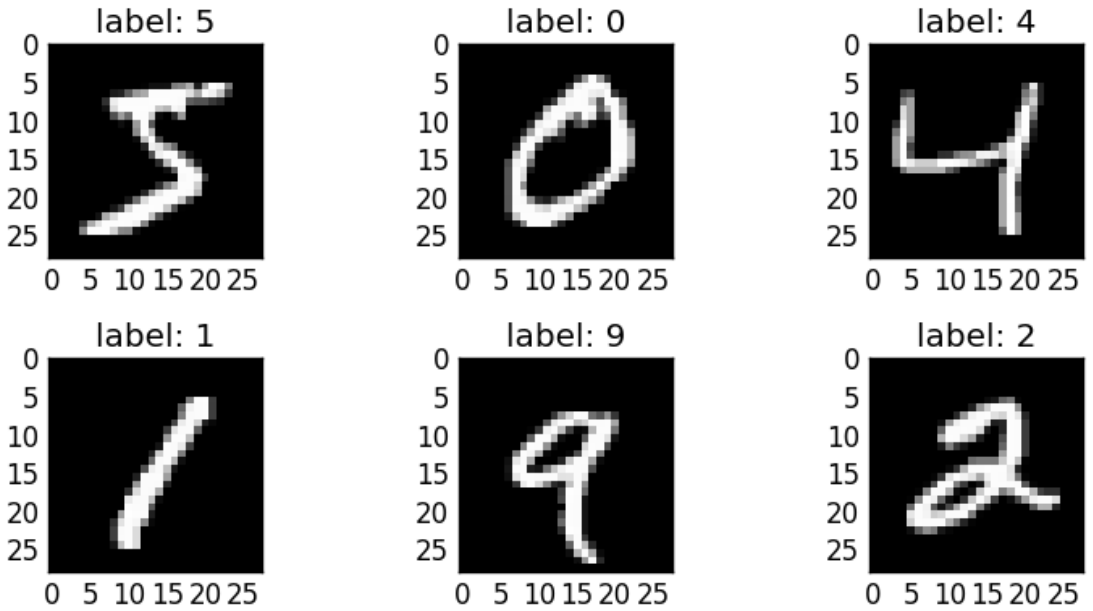
- **Deep Neural Networks are hard to train**
  - learning machines with lots of (typically in range of million) parameters
  - Unstable gradients issue
    - Vanishing gradient problem
    - Exploding gradient problem
  - Choice of network architecture and other hyper-parameters is also important.
  - Many factors can play a role in making deep networks hard to train
  - Understanding all those factors is still a subject of ongoing research

Deep Learning Example

Hello World of Deep Learning: Recognition of MNIST

Introducing the MNIST problem

- **MNIST (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.**
- **It consists of images of handwritten digits like these:**



- **The MNIST database contains 60,000 training images and 10,000 testing images.**

Example Problem - MNIST

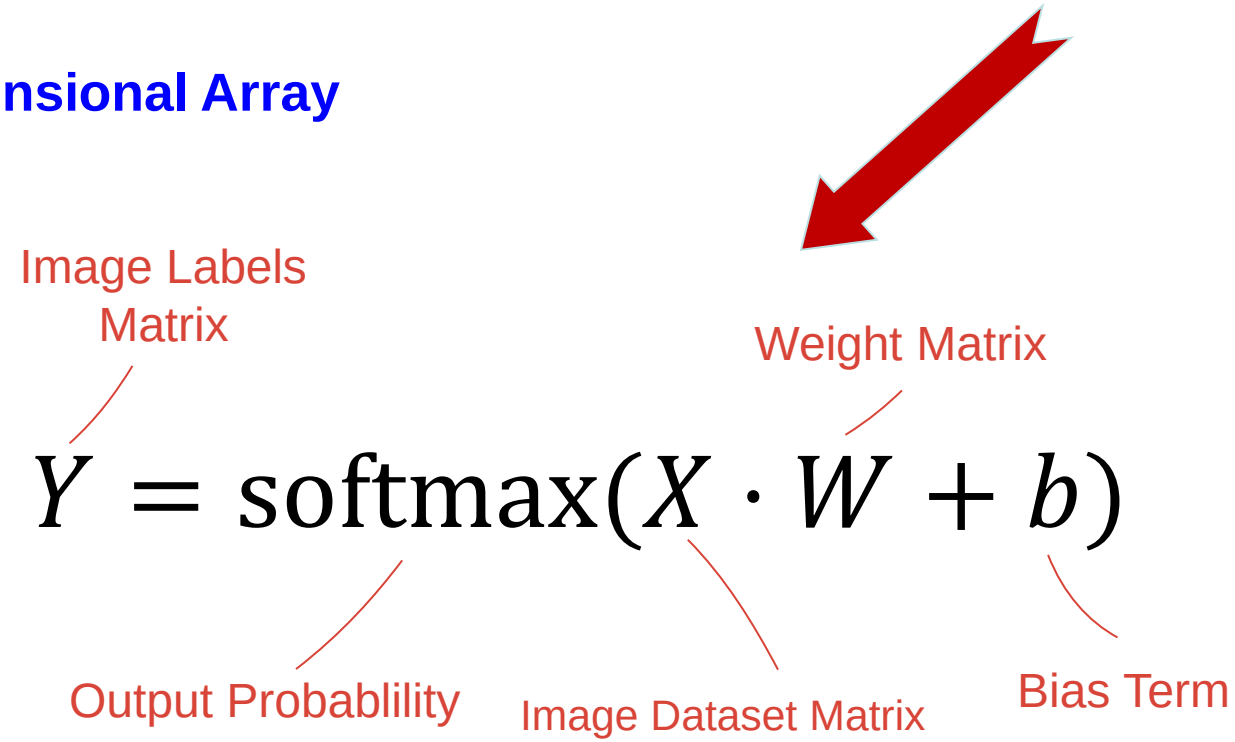
- **Recognizes handwritten digits.**
- **We uses the MNIST dataset, a collection of 60,000 labeled digits that has kept generations of PhDs busy for almost two decades. You will solve the problem with less than 100 lines of Python/Keras/TensorFlow code.**
- **We will gradually enhance the accuracy of the neural network by using the mentioned techniques.**

Steps for MNIST recognition

- **Understand the MNIST data**
- **Build the matrix regression function**
- **Softmax regression layer**
- **The cost function**

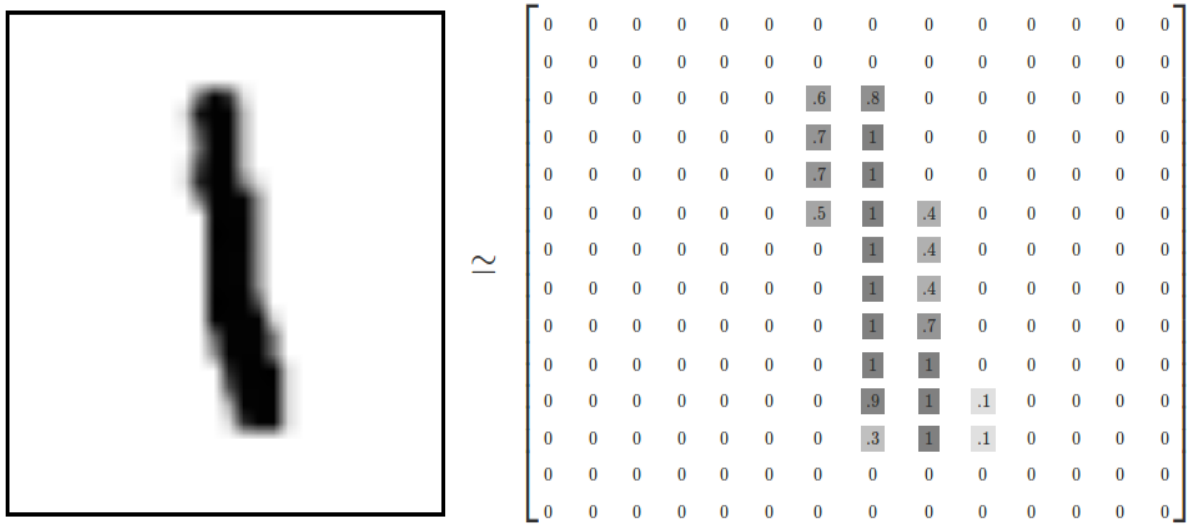
$$y = x \cdot w + b$$

Matrix = Tensor = Multi-dimensional Array



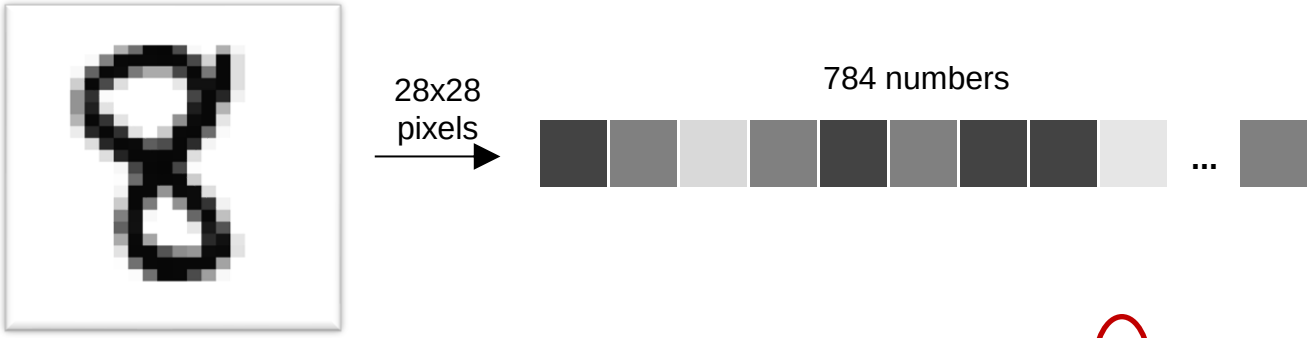
The MNIST Data

- **Every MNIST data point has two parts: an image of a handwritten digit and a corresponding label. We'll call the images "x" and the labels "y". Both the training set and test set contain images and their corresponding labels;**
- **For each grayscale pixel value 0-255, we normalize it to 0-1;**
- **Each image is 28 pixels by 28 pixels. We can interpret this as a big array of numbers:**



One Layer NN for MNIST Recognition

- We will start with a very simple model, called **Softmax Regression**.
- We can flatten each image array into a vector of 28x28 = 784 numbers. It doesn't matter how we flatten the array, as long as we're consistent between images.
- From this perspective, the MNIST images are just a bunch of points in a 784-dimensional vector space.

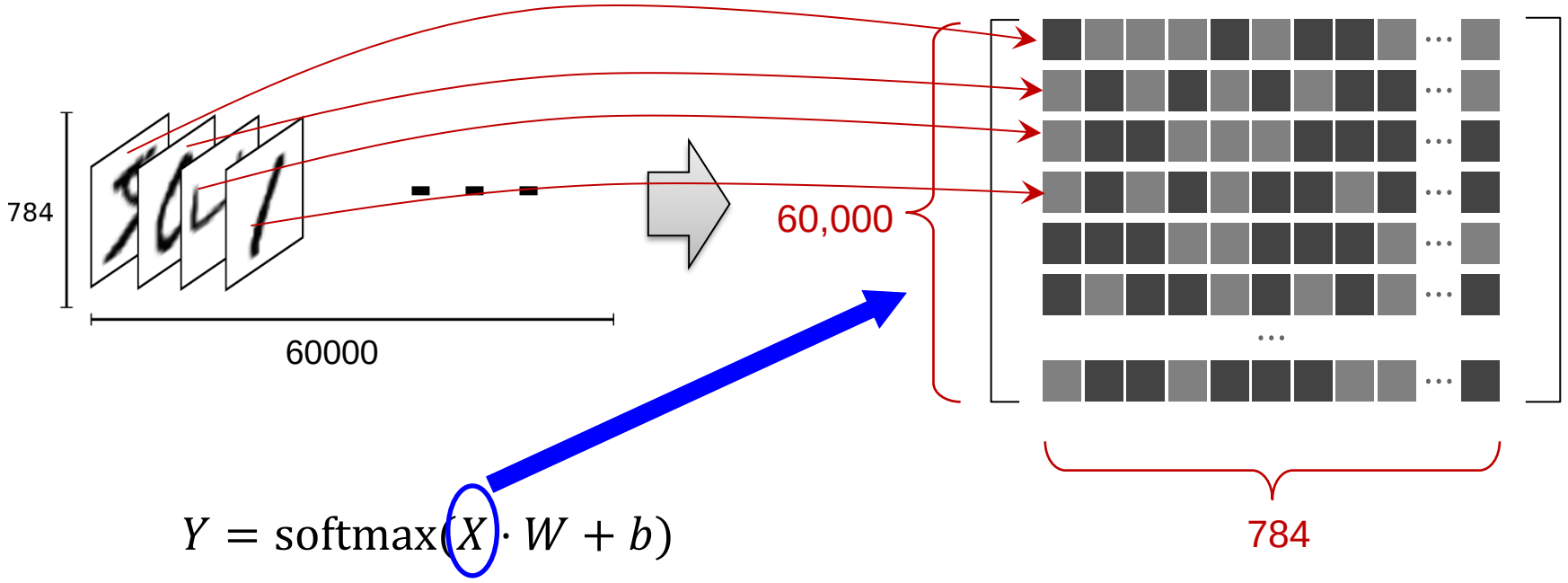


$$Y = \text{softmax}(X \cdot W + b)$$

❖ **What are we missing here?**

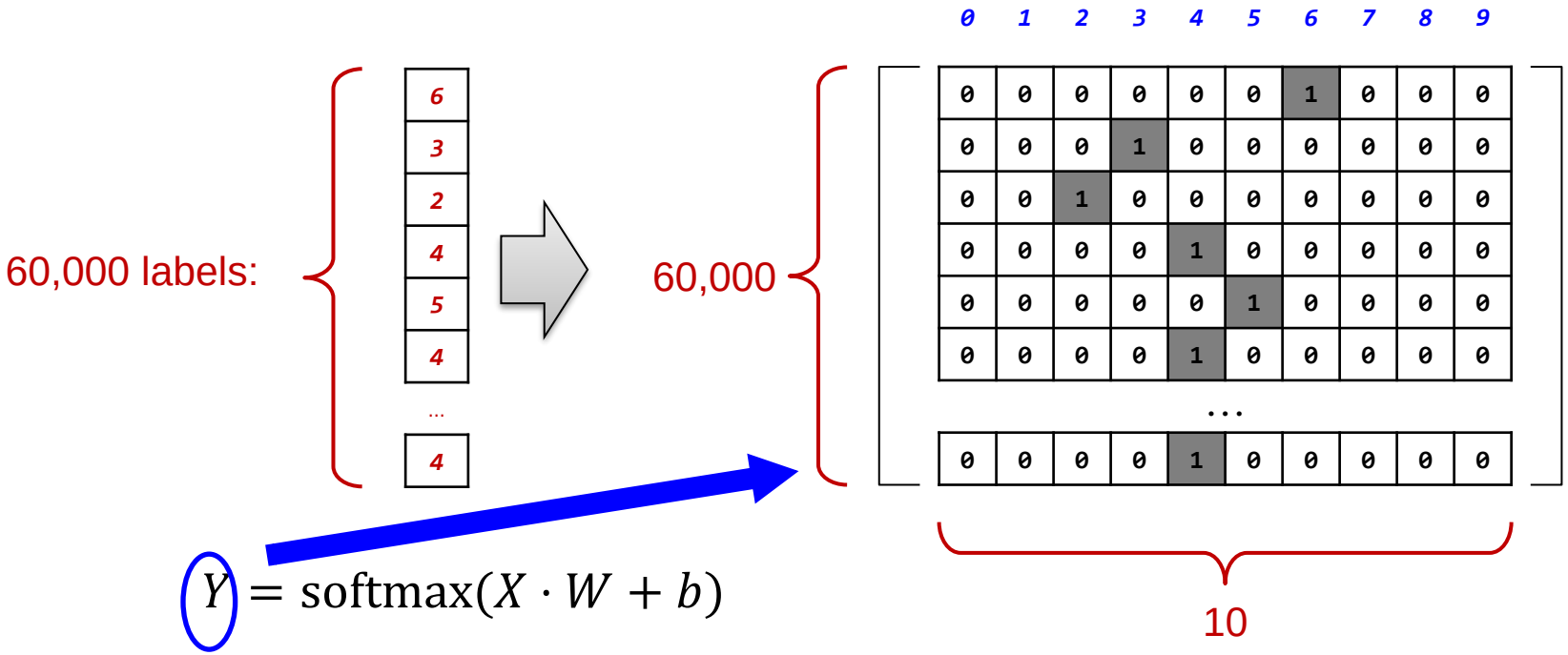
Result of the Flatten Operation

- **The result is that the training images is a matrix (tensor) with a shape of `[60000, 784]`.**
- **The first dimension is an index into the list of images and the second dimension is the index for each pixel in each image.**
- **Each entry in the tensor is a pixel intensity between 0 and 1, for a particular pixel in a particular image.**



One-hot Vector (One vs All)

- For the purposes of this tutorial, we label the y's as "one-hot vectors".
- A one-hot vector is a vector which is 0 in most dimensions, and 1 in a single dimension.
- How to label an "8"?
  - `[0,0,0,0,0,0,0,0,1,0]`
- What is the dimension of our y matrix (tensor)?



Comparison: One-Hot Encoding vs. Integer Encoding

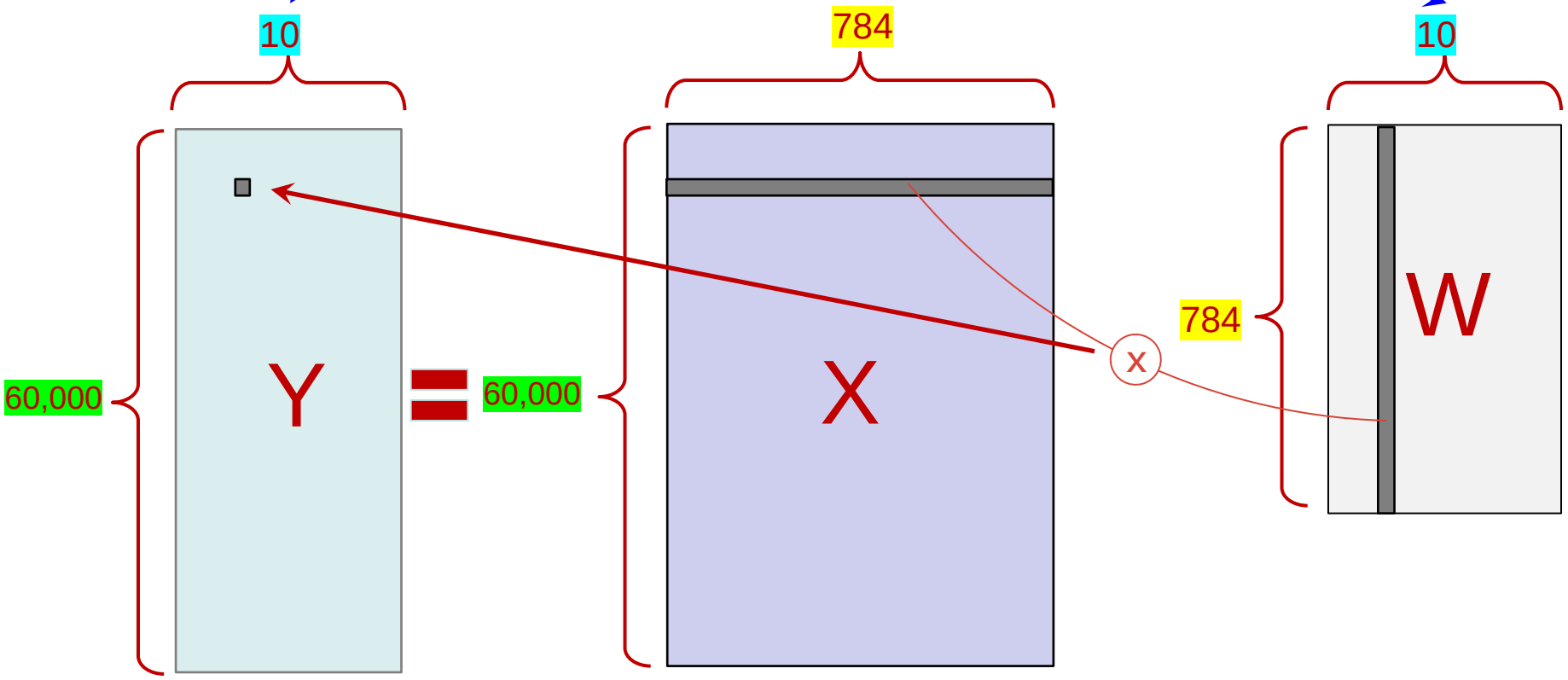
Aspect	One-Hot Encoding	Integer Encoding (Used in PyTorch)
Label Representation	Converts label into a vector where all values are 0 except for the true class index (e.g., class 2 = [0, 0, 1, 0])	Uses an integer for the label (e.g., class 2 = 2)

Example	For 3 classes:
	`class 0 = [1, 0, 0],`
	`class 1 = [0, 1, 0],`
	`class 2 = [0, 0, 1]`
		For 3 classes:
		`class 0 = 0,`
		`class 1 = 1,`
		`class 2 = 2`

Softmax Application	Applied manually before calculating cross-entropy	Handled automatically within CrossEntropyLoss
Final Loss Value	Produces the same loss as integer encoding if calculations are equivalent	

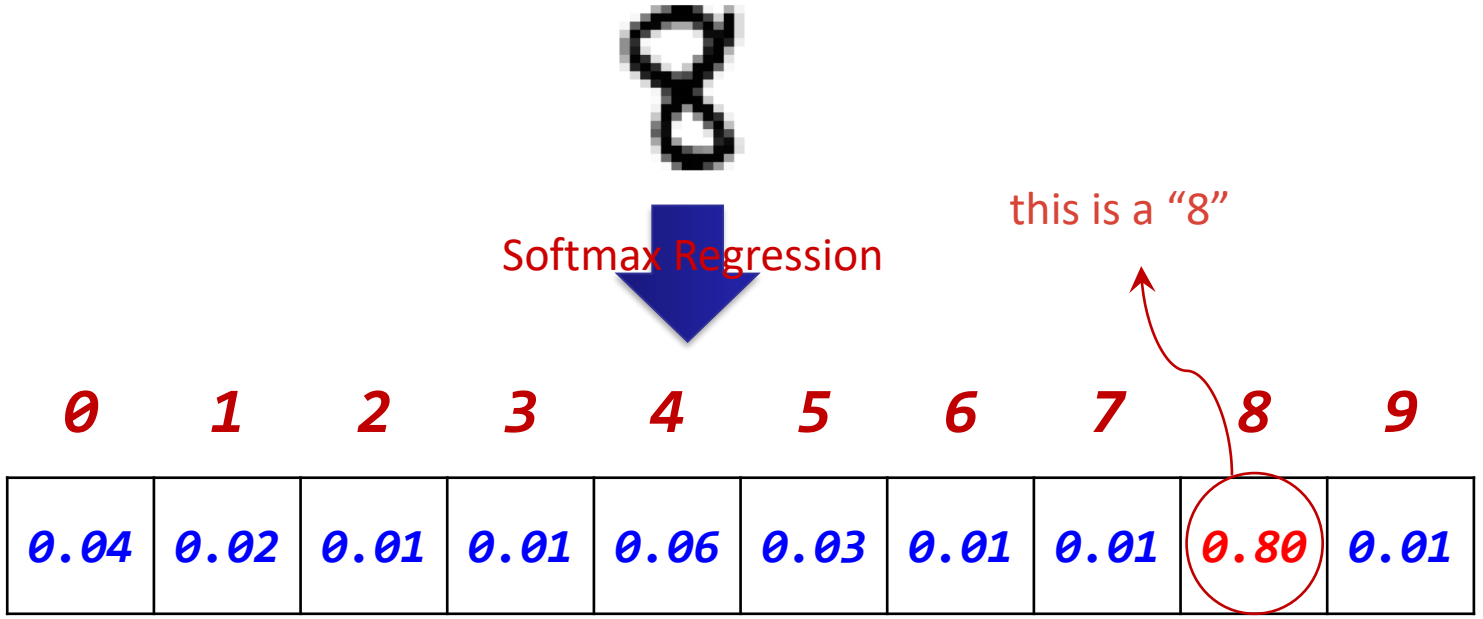
Matrix representation of the equation

$$Y = \text{softmax}(X \cdot W + b)$$



Softmax Regressions

- Every image in MNIST is of a handwritten digit between 0 and 9.
- So, there are only ten possible things that a given image can be. We want to be able to look at an image and give the probabilities for it being each digit.
- For example, our model might look at a picture of an eight and be 80% sure it's an 8, but give a 6% chance to it being a 4 (because of the top loop) and a bit of probability to all the others because it isn't 100% sure.



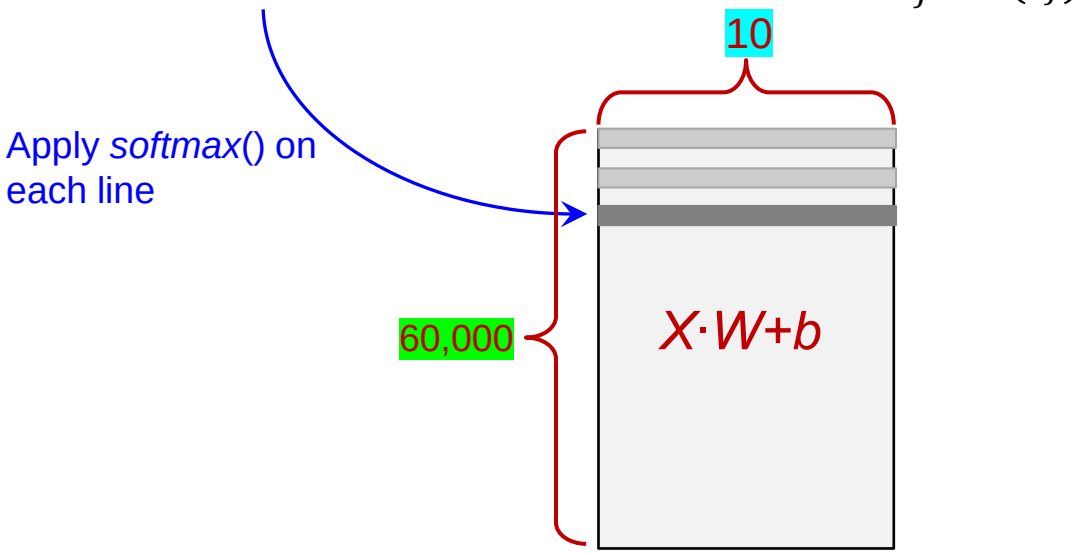
2 steps in softmax regression - Step 2

- **Step 2: Convert the evidence tallies into our predicted probabilities y using the "softmax" function:**

$$h(\mathbf{x}_i) = \text{softmax}(z_i) = \text{softmax}\left(\sum_j W_{i,j} x_j + b_i\right)$$

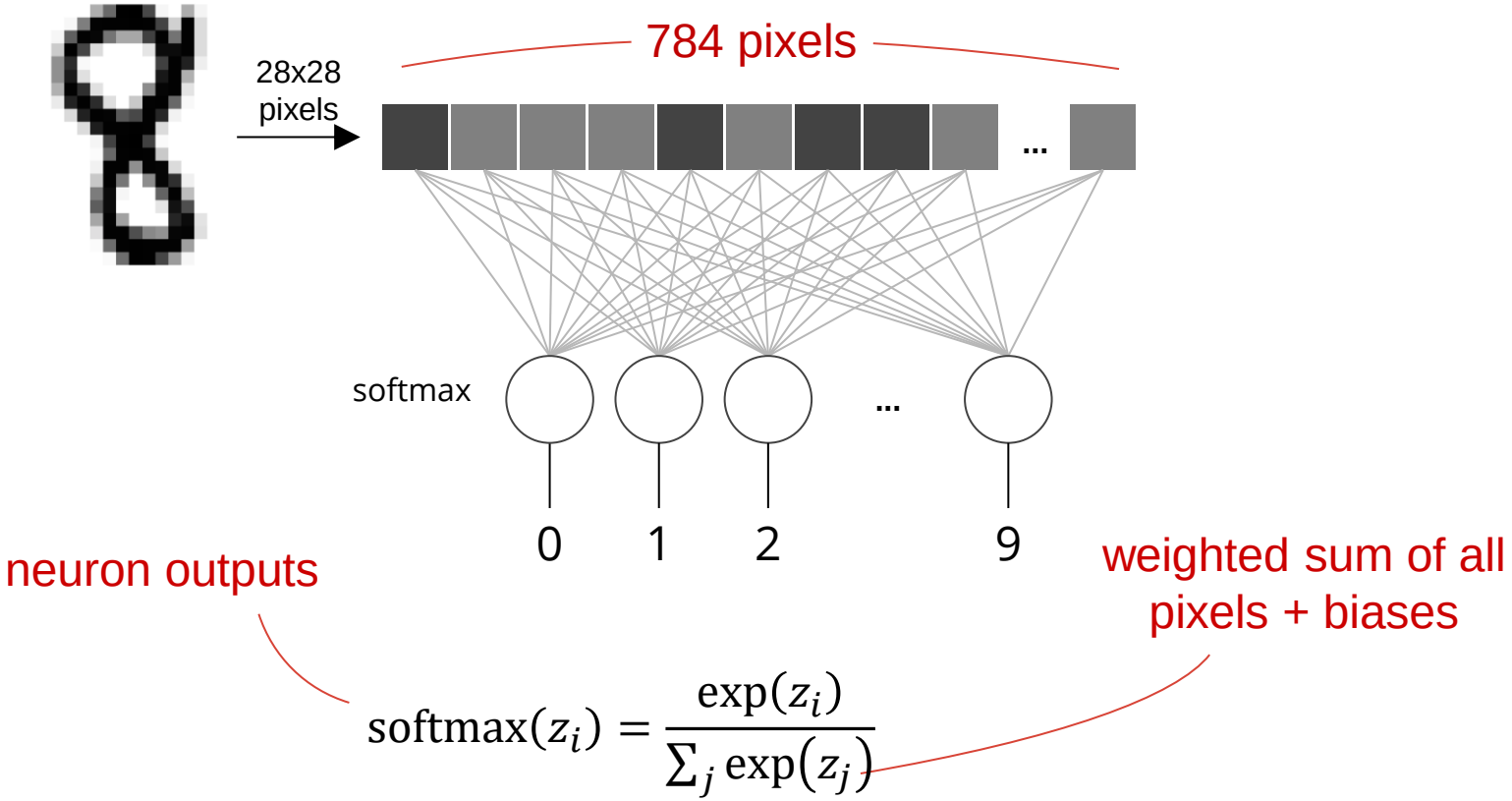
- **Here softmax is serving as an "activation" function, shaping the output of our linear function a probability distribution over 10 cases, defined as:**

$$\text{softmax}(z_i) = \text{normalize}(\exp(z)) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$



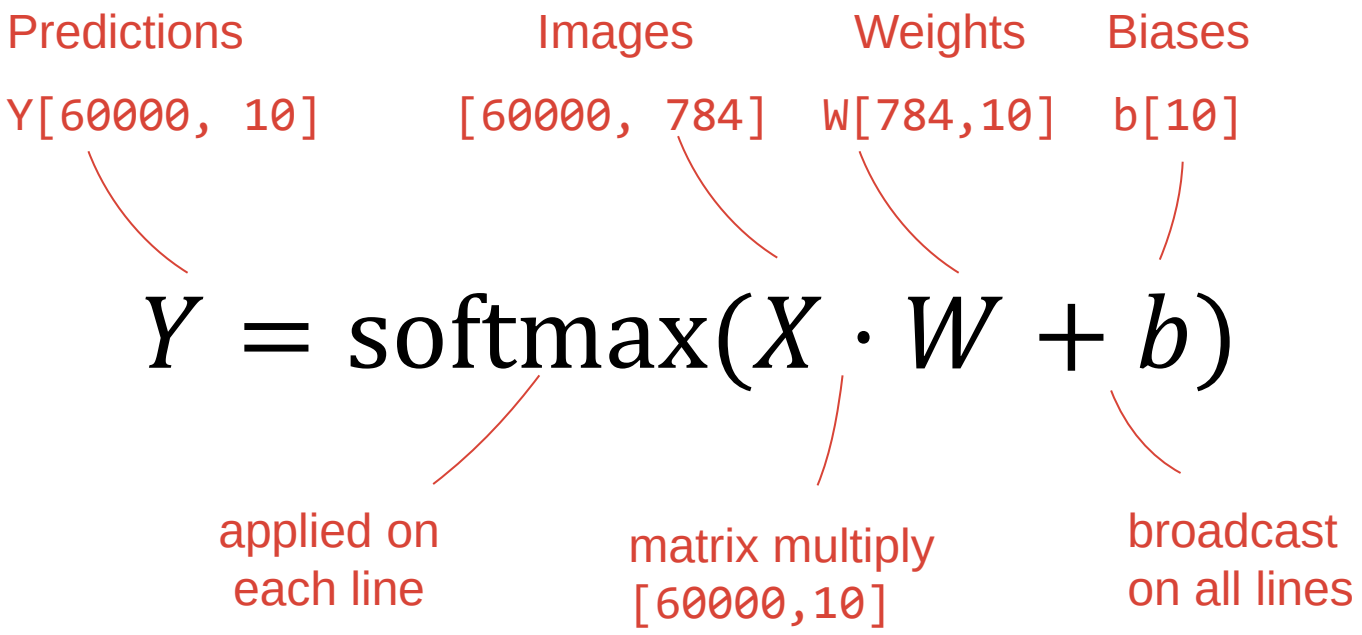
The softmax layer

- **The output from the softmax layer is a set of probability distribution, positive numbers which sum up to 1.**



Softmax on a batch of images

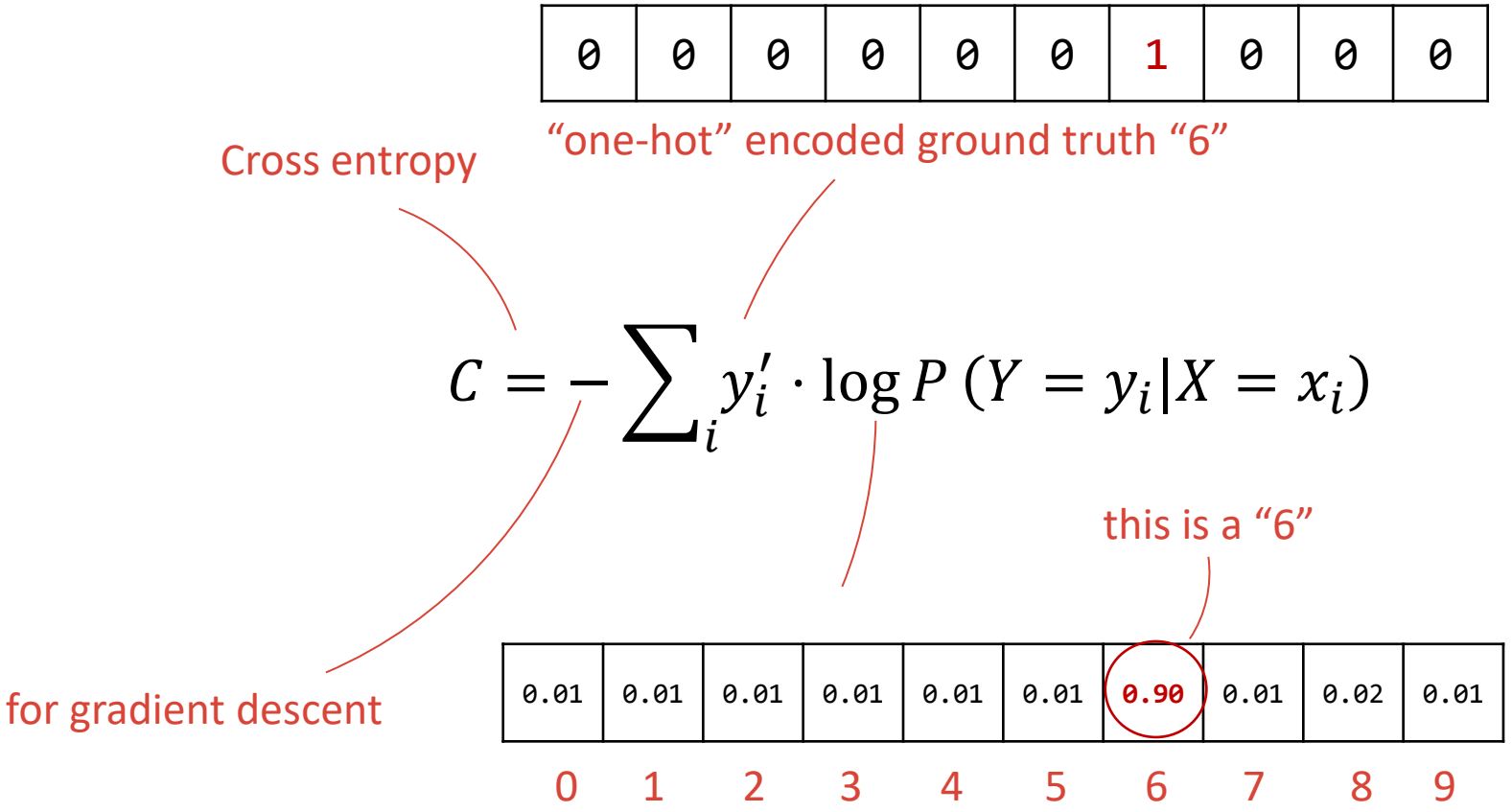
- **More compact representation for "softmaxing" on all the images**



Very similar to 2D linear regression: $y = x \cdot w + b$

The Cross-Entropy Cost Function

- **For classification problems, the Cross-Entropy cost function works better than quadratic cost function.**
- **We define the cross-entropy cost function for the neural network by:**



Short Summary

- **How MNIST data is organized**
  - X:
    - Flattened image pixels matrix
  - Y:
    - One-hot vector
- **Softmax regression layer**
  - Linear regression
  - Output probability for each category
- **Cost function**
  - Cross-entropy

$$Y = \text{softmax}(X \cdot W + b)$$

❖ **How to implement the gradient descent for W?**

Recap

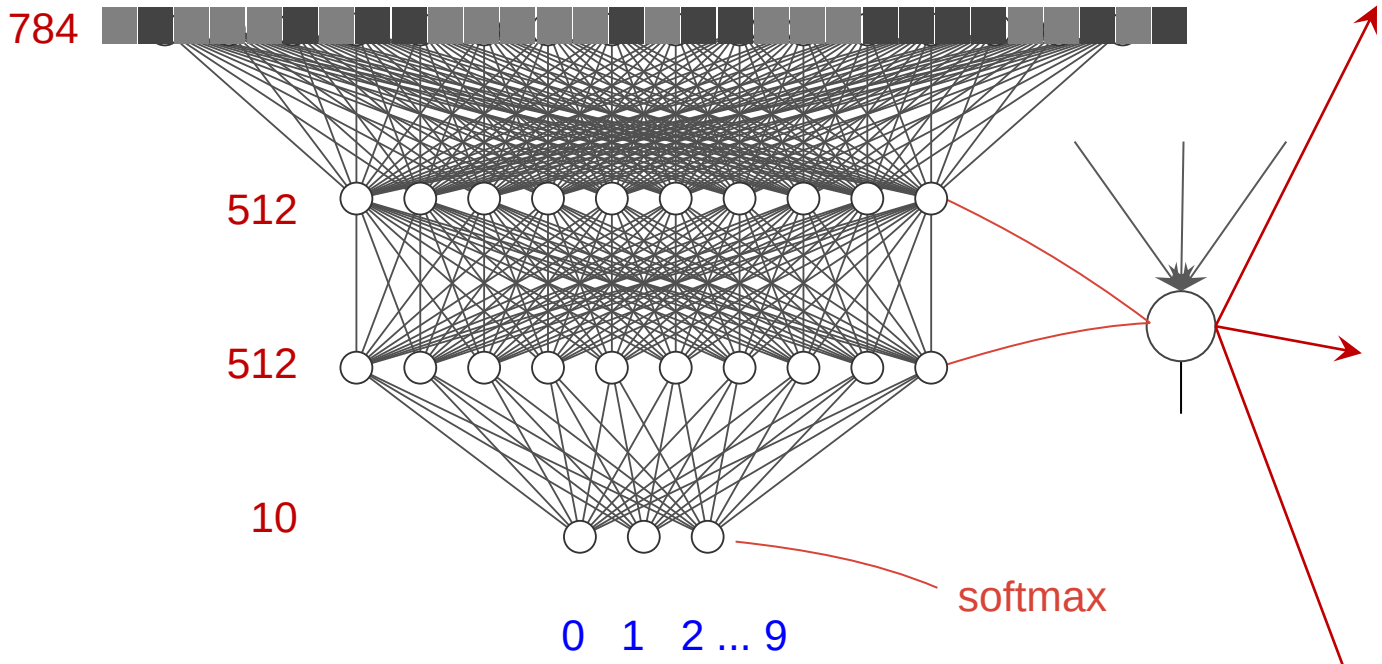
- **Machine learning and deep learning**
  - Data driven approach
  - Simplest case of machine learning
  - Cost function
  - How to train a model?
    - Gradient descent
- **What is a deep neural network?**
  - How to train a deep neural network?
- **MNIST problem**
  - How MNIST data is organized
    - X: Flattened image pixels matrix
    - Y: One-hot vector
  - Softmax regression layer

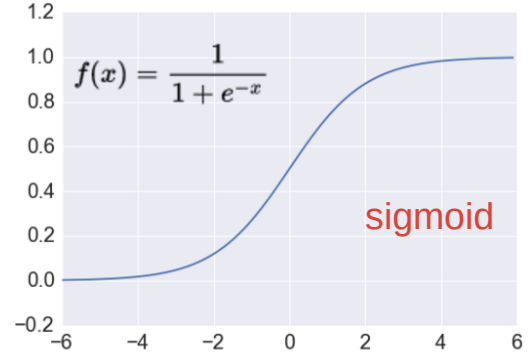
$$Y = \text{softmax}(X \cdot W + b)$$

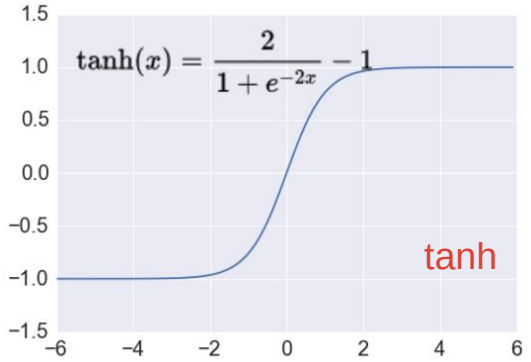
  - Cost function
    - Cross-entropy

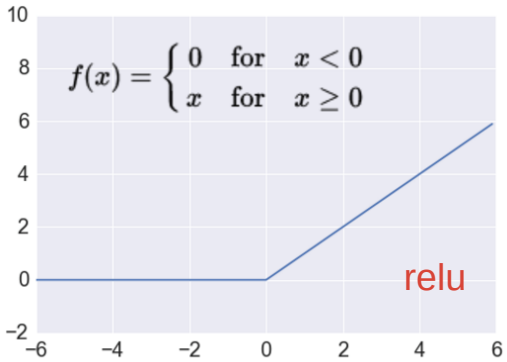
Adding More Layers

- **Using a 2 hidden layer model:**









Introduction to Deep Learning

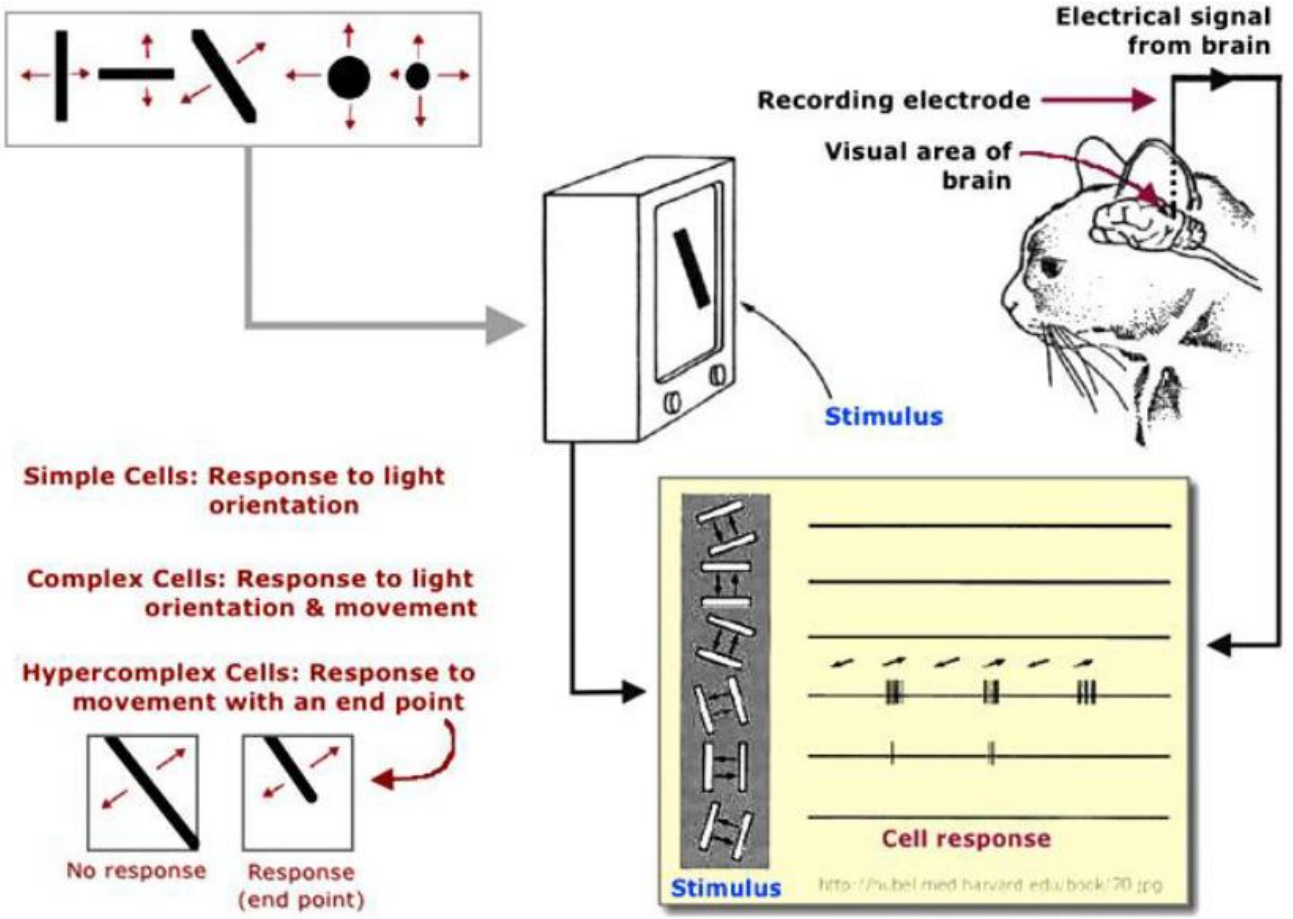
Convolutional Neural Network

Why Using Fully Connected Layers?

- **Such a network architecture does not consider the spatial structure of the images.**
  - For instance, it treats input pixels which are far apart and close together on the same weight.
- **Spatial structure must instead be inferred from the training data.**

- **Is there an architecture which tries to take advantage of the spatial structure?**

History of CNN (1)

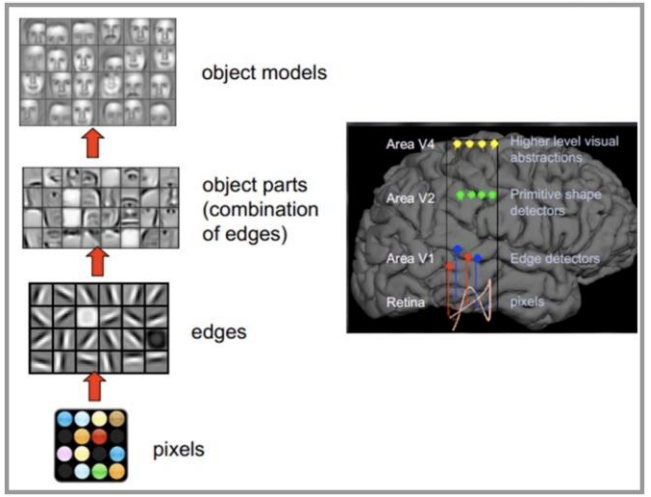


Hubel & Wiesel, 1959

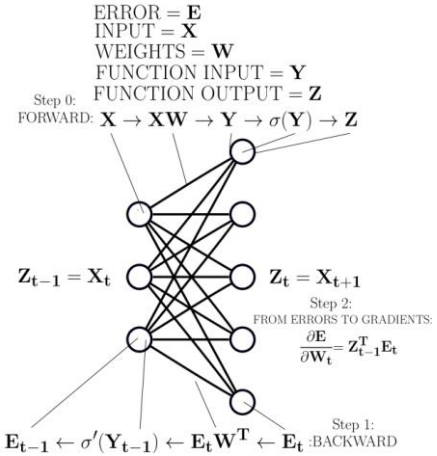
Slide referred from Stanford, CS231n, Winter 2016, Lecture 1

History of CNN (2)

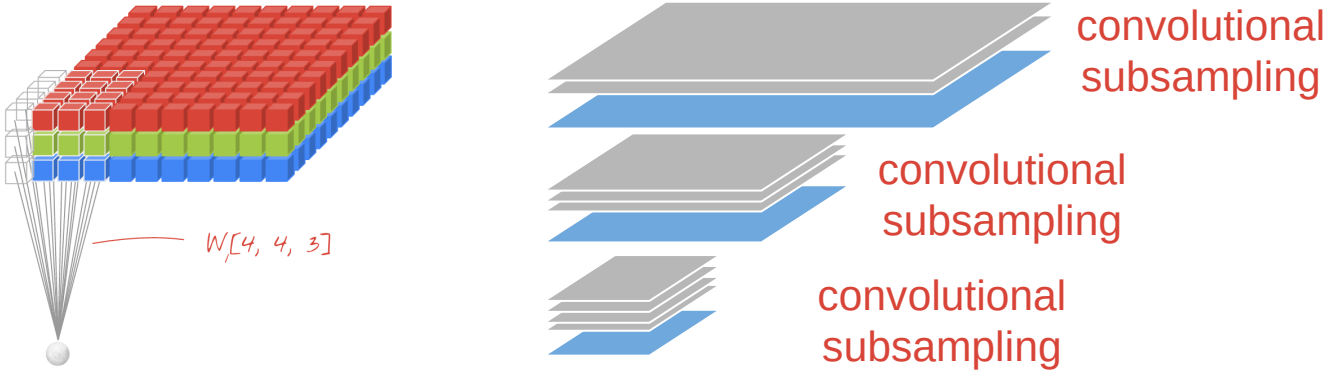
- Kunihiko Fukushima (Japanese: 福島 邦彦, born1936), most noted for his work on artificial neural networks and deep learning.
- In 1980, Fukushima published the neocognitron, the original deep convolutional neural network (CNN) architecture.
- The neocognitron is a hierarchical, multilayered artificial neural network.
- It served as the inspiration for convolutional neural networks.



- Yann LeCun is a renowned computer scientist and a pioneer in the field of deep learning, particularly in the development and advancement of Convolutional Neural Networks (CNNs).
- In the 1980s and 1990s, Development of the LeNet-5 architecture.
- Introduction of the backpropagation algorithm, which allowed for efficient and effective training of deep neural networks.



Convolution Neuron Network (CNN)

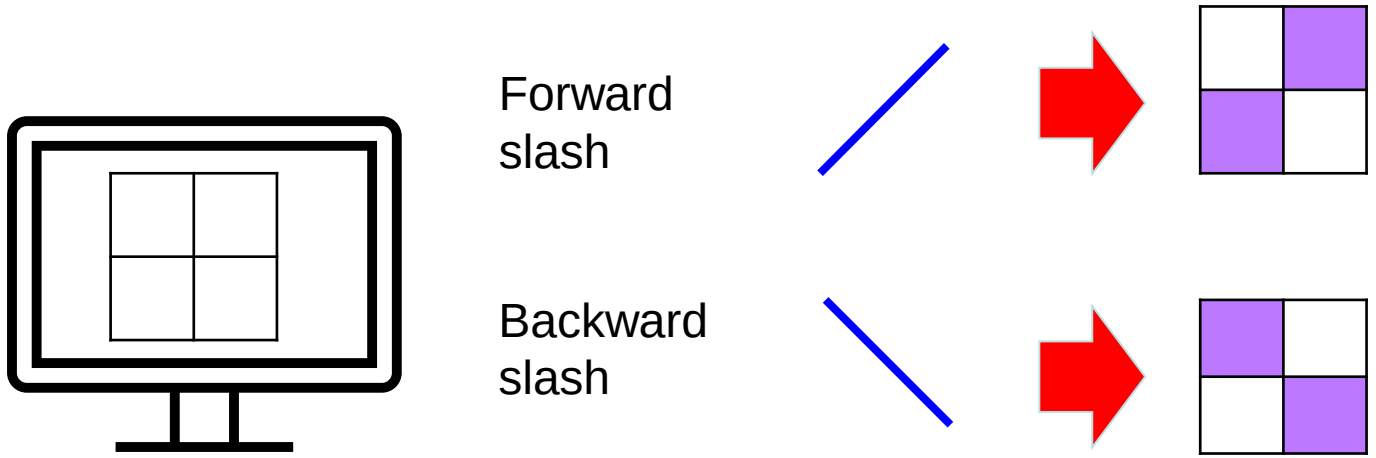


from Martin Görner Learn TensorFlow and deep learning, without a Ph.D

- **Deep convolutional network is one of the most widely used types of deep network.**
- **In a layer of a convolutional network, one "neuron" does a weighted sum of the pixels just above it, across a small region of the image only. It then acts normally by adding a bias and feeding the result through its activation function.**
- **The big difference is that each neuron reuses the same weights whereas in the fully-connected networks seen previously, each neuron had its own set of weights.**

A simple example shows how CNN works

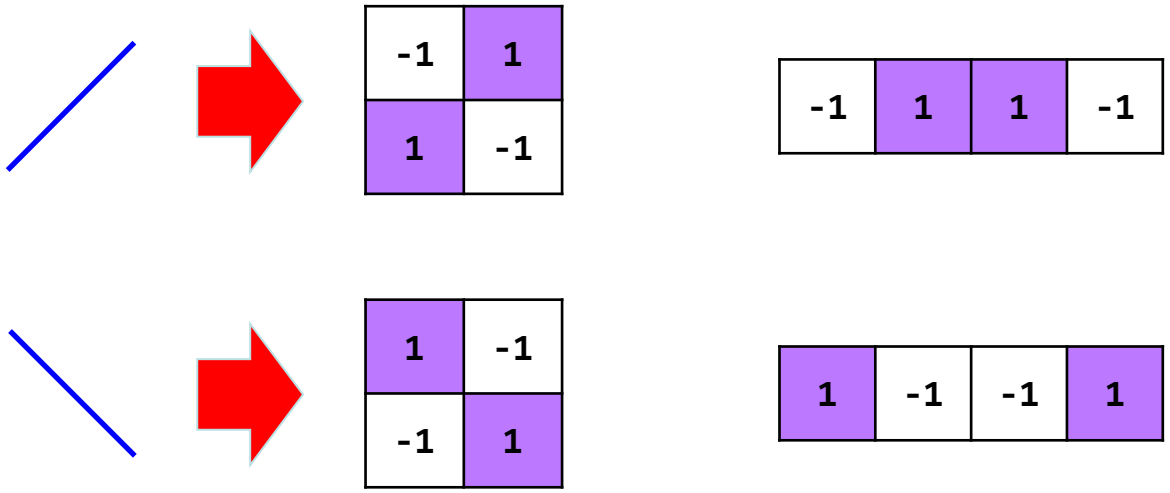
- **Simplified world**
  - Only has two characters: “/” and “\”
  - Image only has 2x2 resolution



Slides referenced from Luis Serrano,
A friendly introduction to Convolutional Neural Networks and Image Recognition,
https://youtu.be/2-OI7ZB0MmU

A simple example shows how CNN works

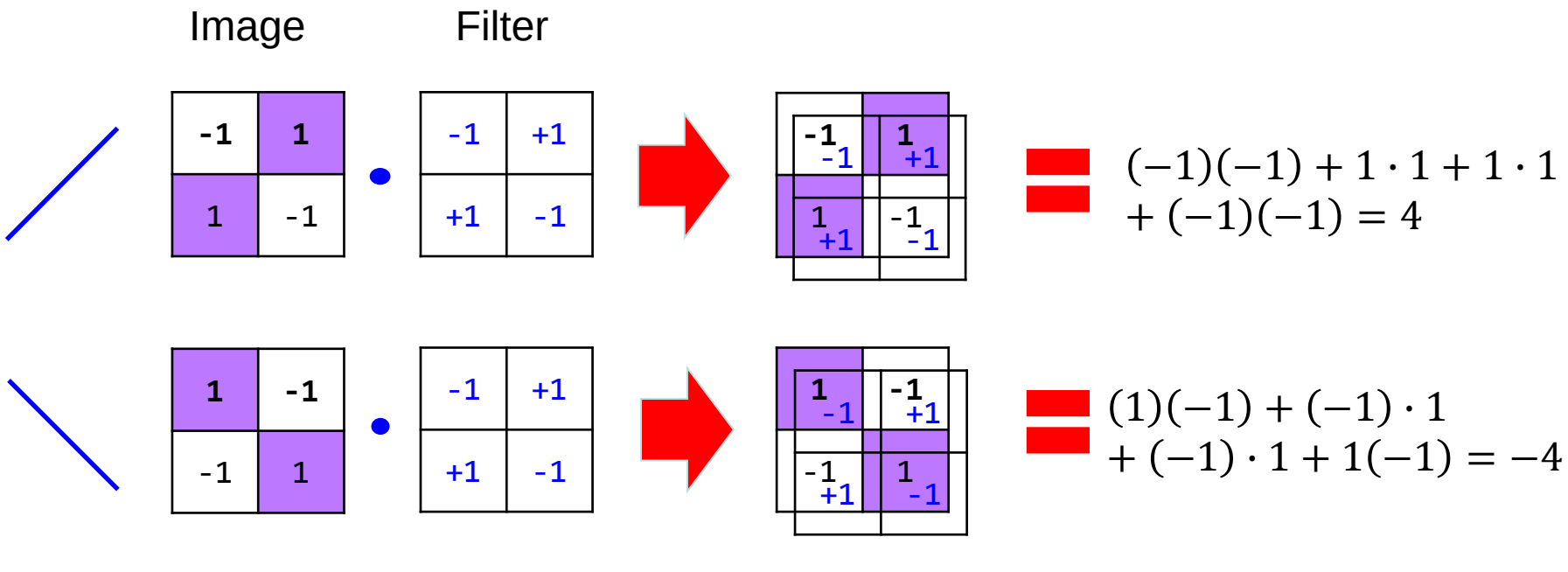
- **Simplified image in binary world**



Think about a way using the above 2x2 matrix elements to differentiate them?

A simple example shows how CNN works

- **Solution:**
  - Use a filter, overlay with the image, do a dot product:

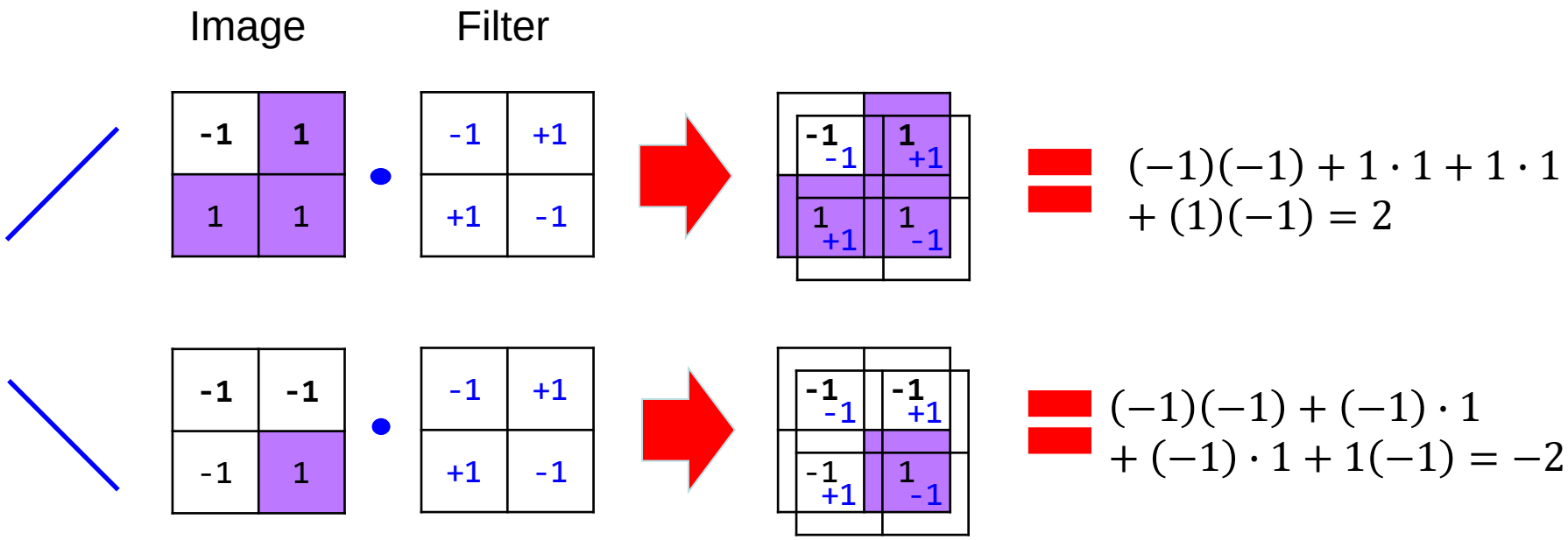


You can use this filter for the same effect:

+1	-1
-1	+1

A simple example shows how CNN works

- **How about the below figures?**
  - Use the same filter, overlay with the image, do a dot product:



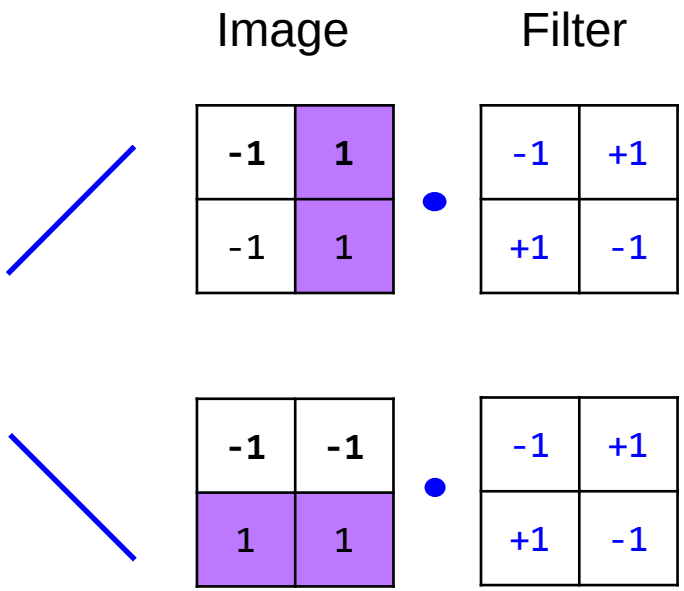
If the result >0, poorly drawn "/"
If the result <0, incomplete "\"

You can use this filter for the same effect:

+1	-1
-1	+1

A simple example shows how CNN works

- **As an exercise?**
  - Use the same filter, to classify below images:

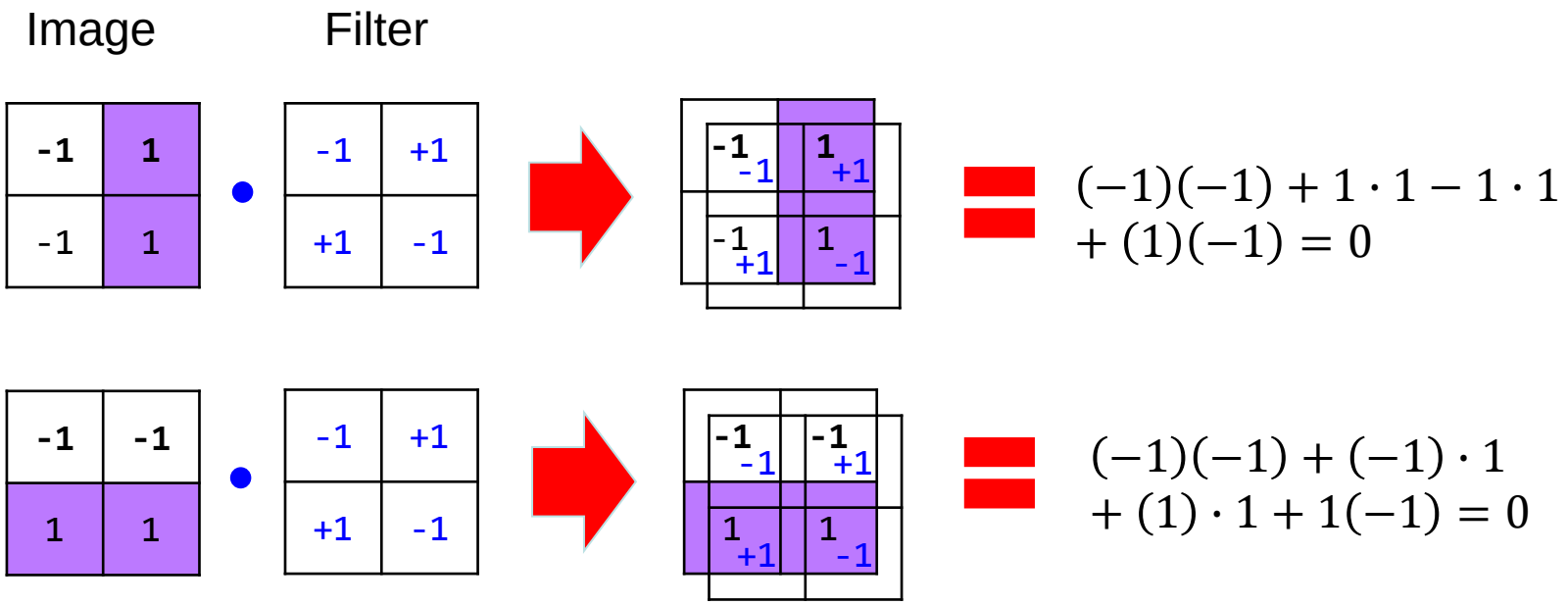


You can use this filter for the same effect:

+1	-1
-1	+1

A simple example shows how CNN works

- **As an exercise?**
  - Use the same filter, overlay with the image, do a dot product:



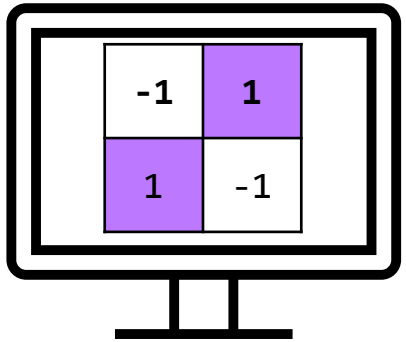
Results are zeros…, No idea!

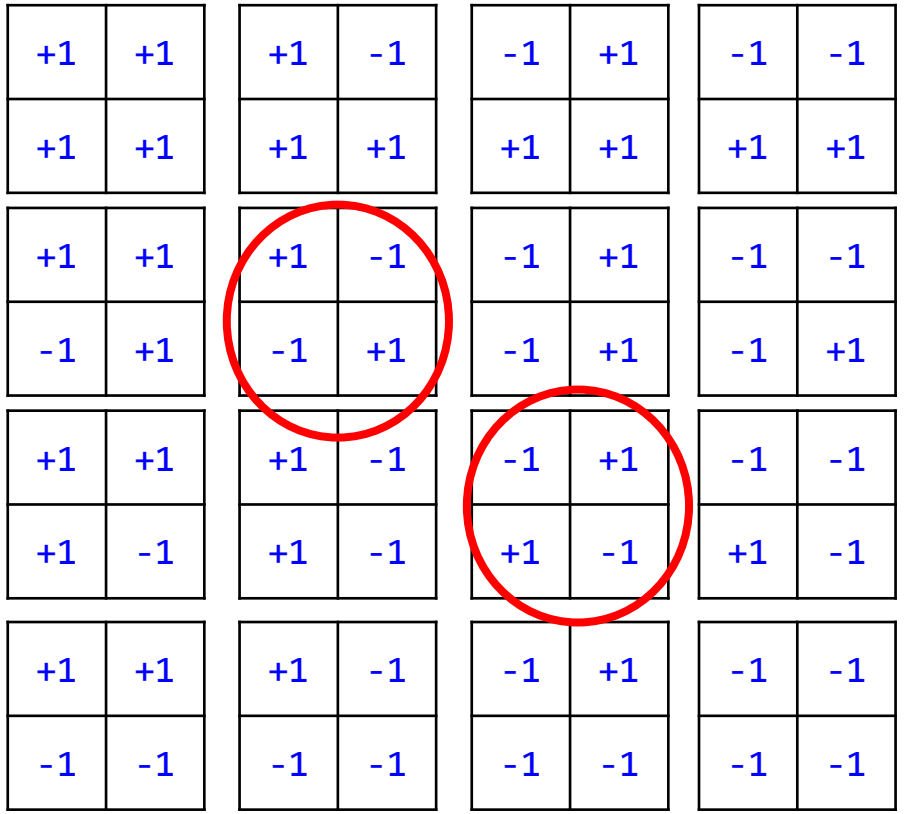
You can use this filter for the same effect:

+1	-1
-1	+1

A simple example shows how CNN works

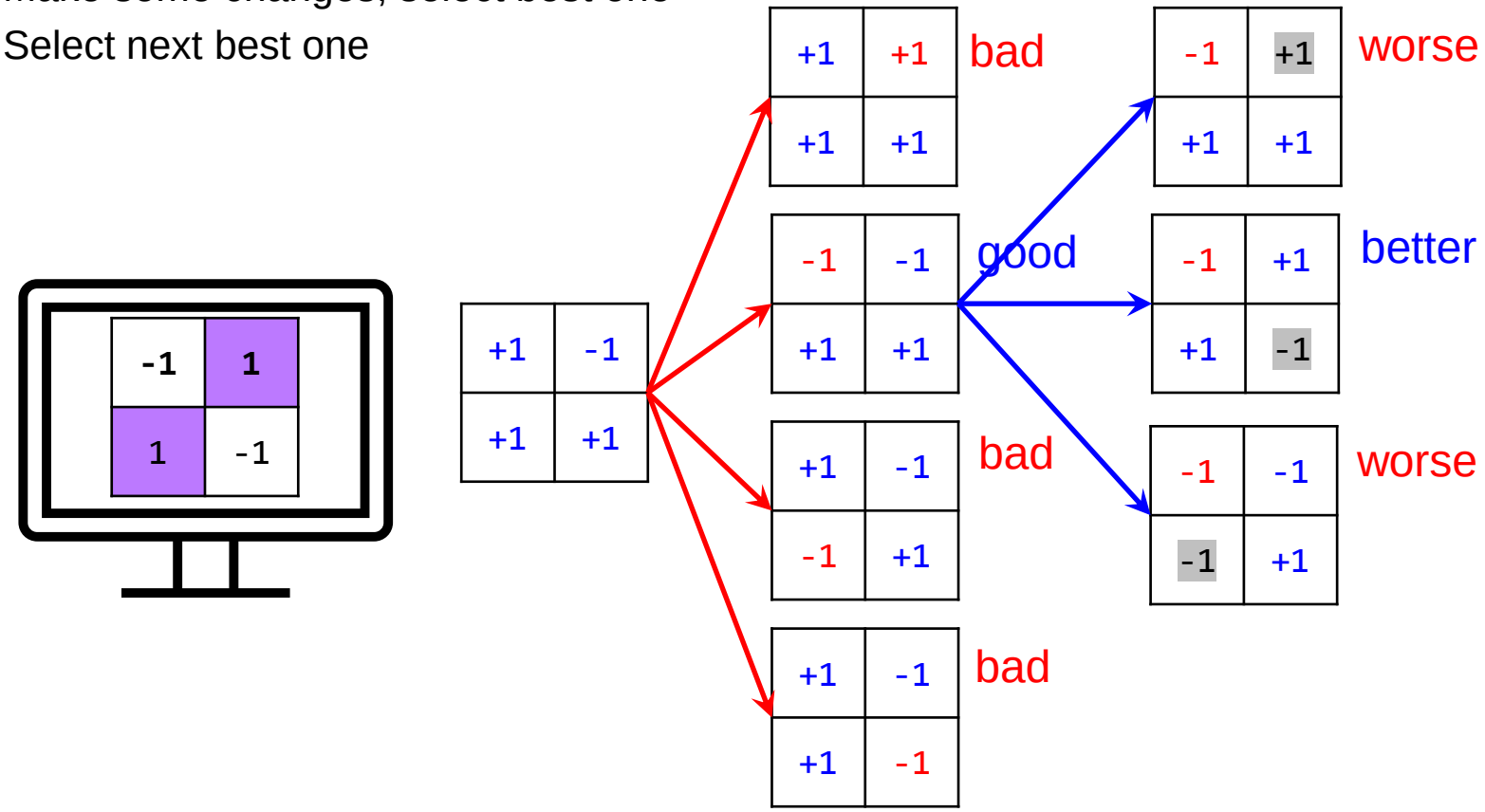
- **How do we find this filter that works?**
  - Vanilla way:
    1. Check all possible combinations (2*2*2*2=16)
    2. Run them
    3. Select the best filter





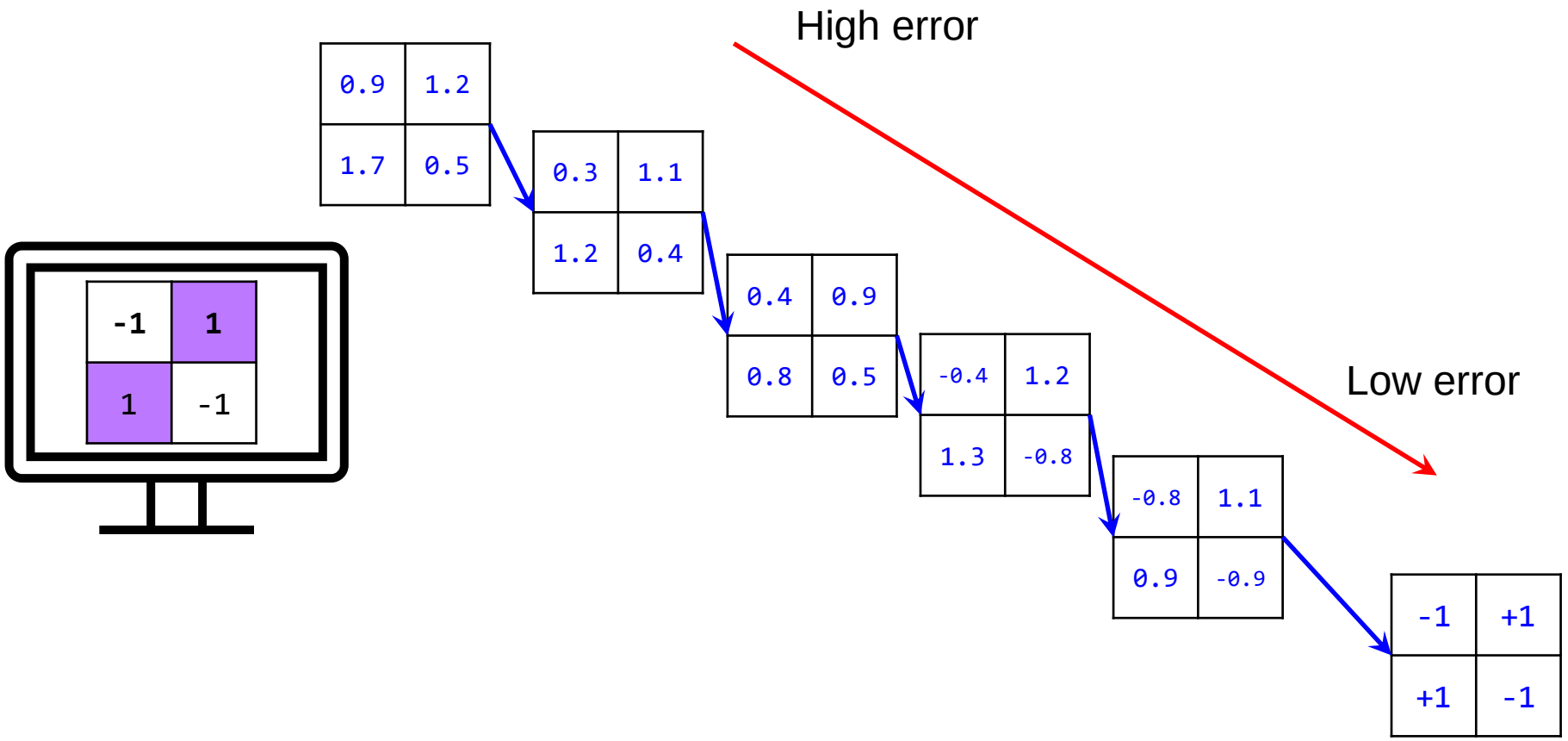
A simple example shows how CNN works

- **How do we find this filter that works?**
  - Better way:
    1. Start from a random filter
    2. Make some changes, select best one
    3. Select next best one



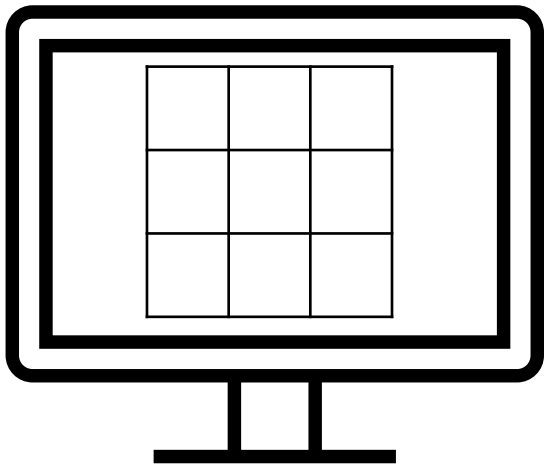
A simple example shows how CNN works

- **How do we find this filter that works?**
  - What did we learn at the beginning?
    - Gradient Descent

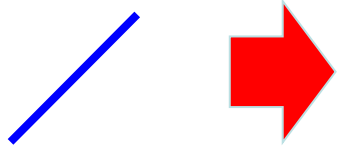


A slightly complex example

- **A more complex world:**
  - Image has 3x3 resolution
  - 4 characters:
    - "/", "\", "X", "O"

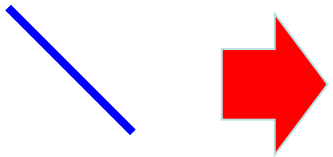


Forward slash



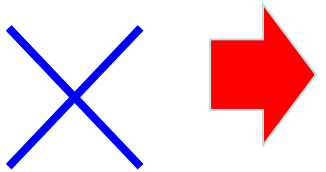
-1	-1	1
-1	1	-1
1	-1	-1

Backward slash



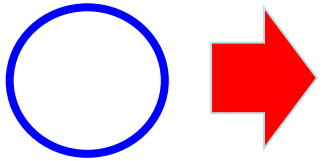
1	-1	-1
-1	1	-1
-1	-1	1

X



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

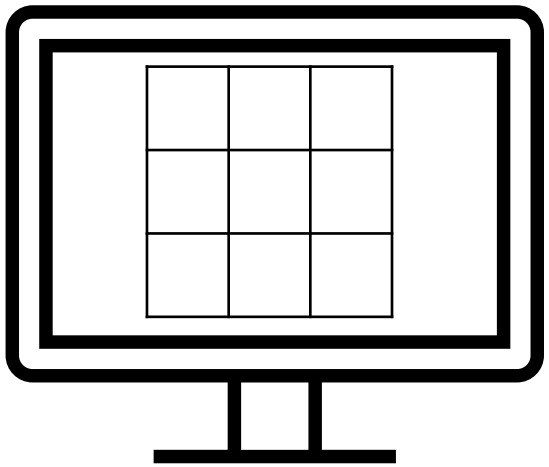
O



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

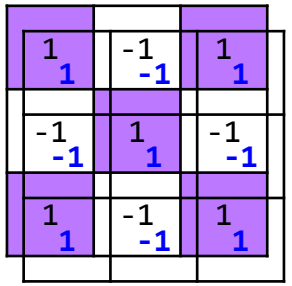
A slightly complex example

- **A more complex world:**
  - Image has 3x3 resolution
  - 4 characters:
    - "/", "\", "X", "O"



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

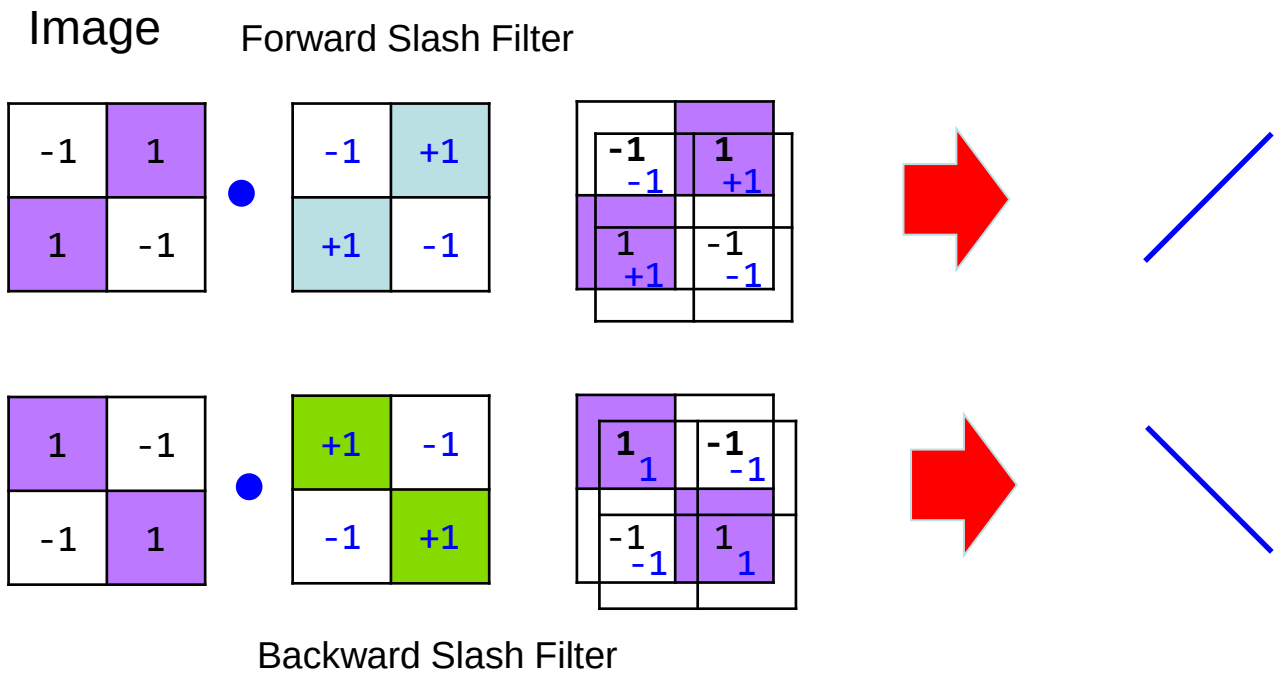
1	-1	1
-1	1	-1
1	-1	1



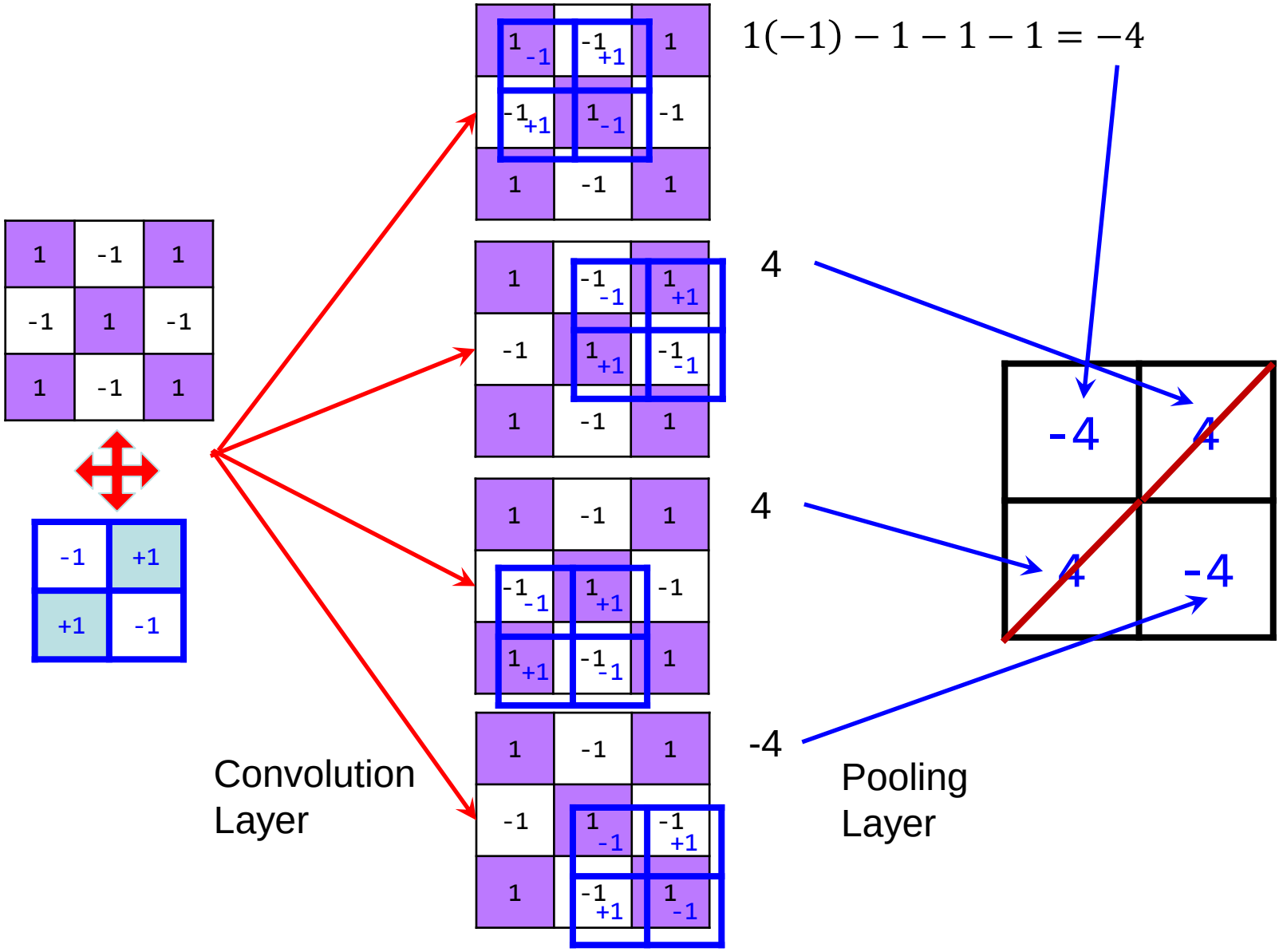
Can we simply use the same strategy as the 2x2 world??

A slightly complex example

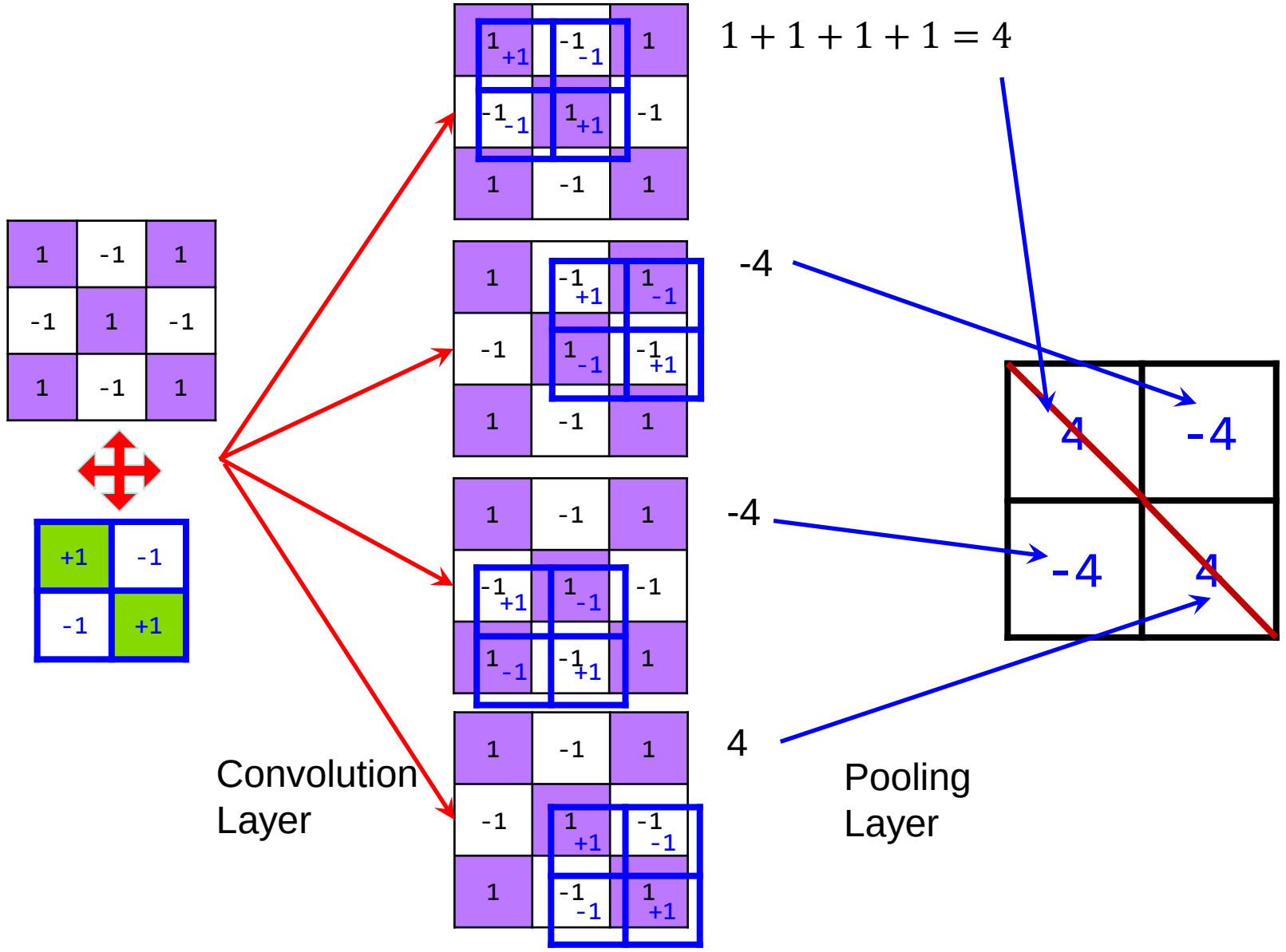
- **Use our previous 2x2 world experience:**



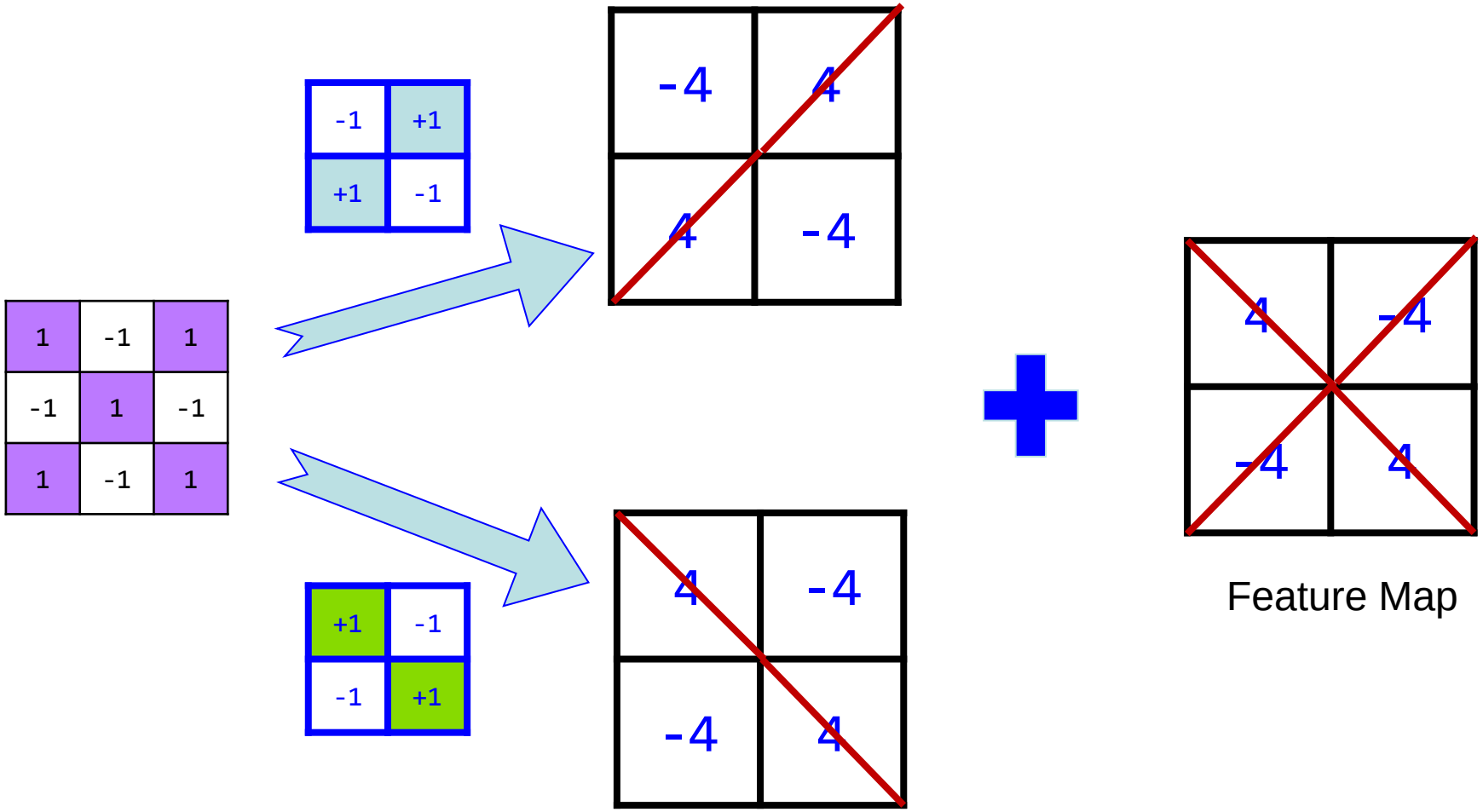
Using our previous knowledge



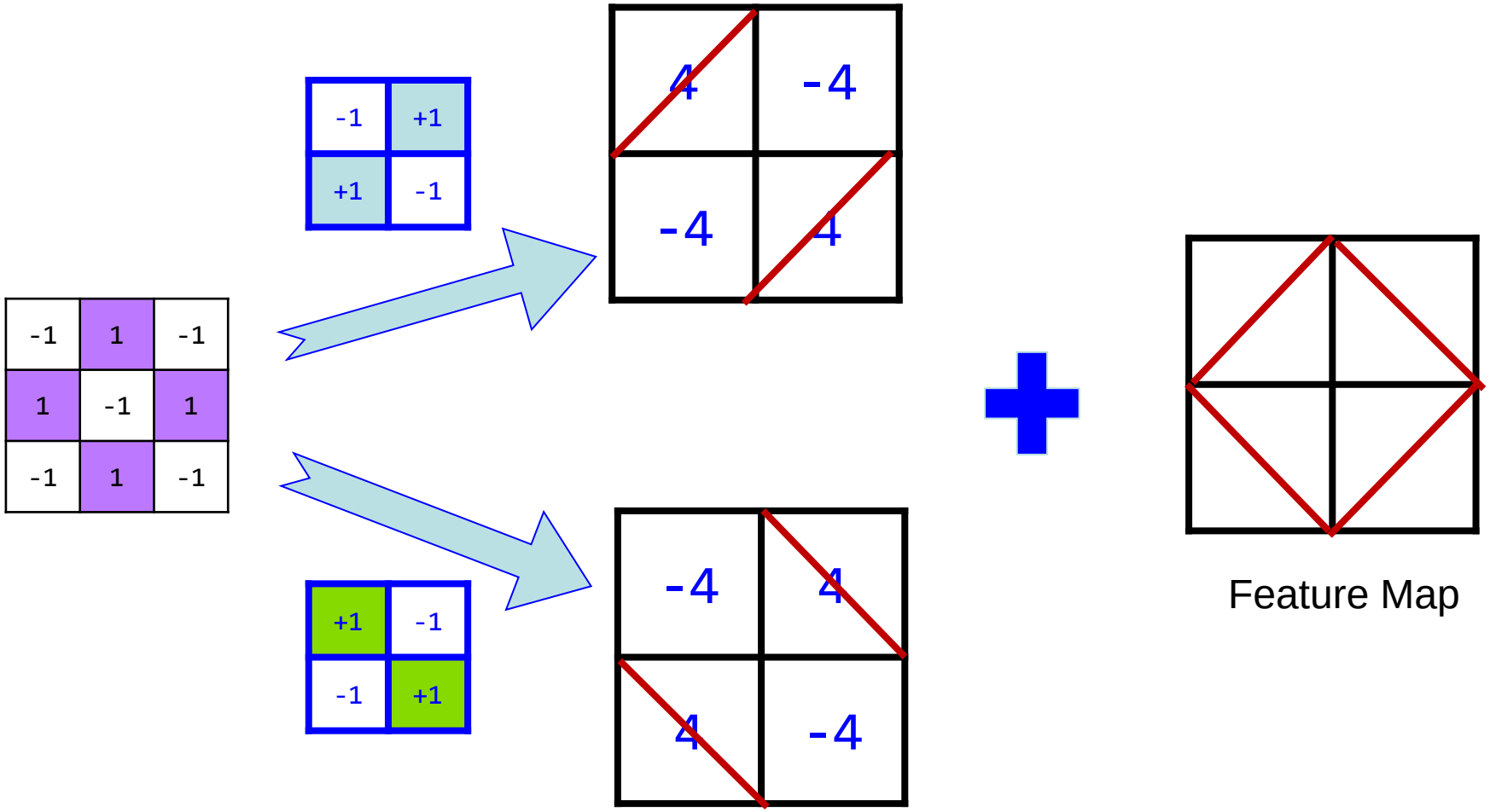
Using our previous knowledge



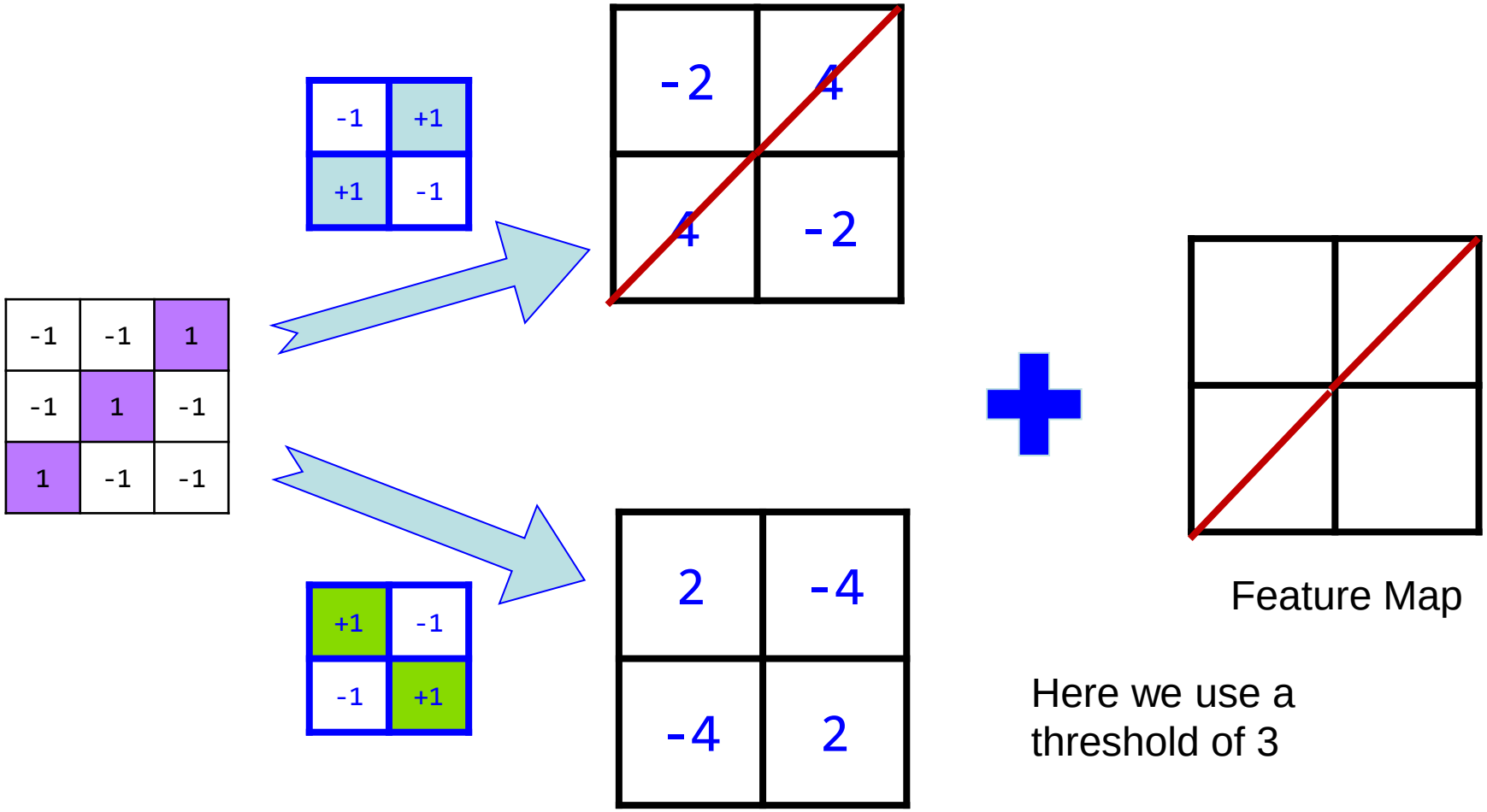
Combine the last two results



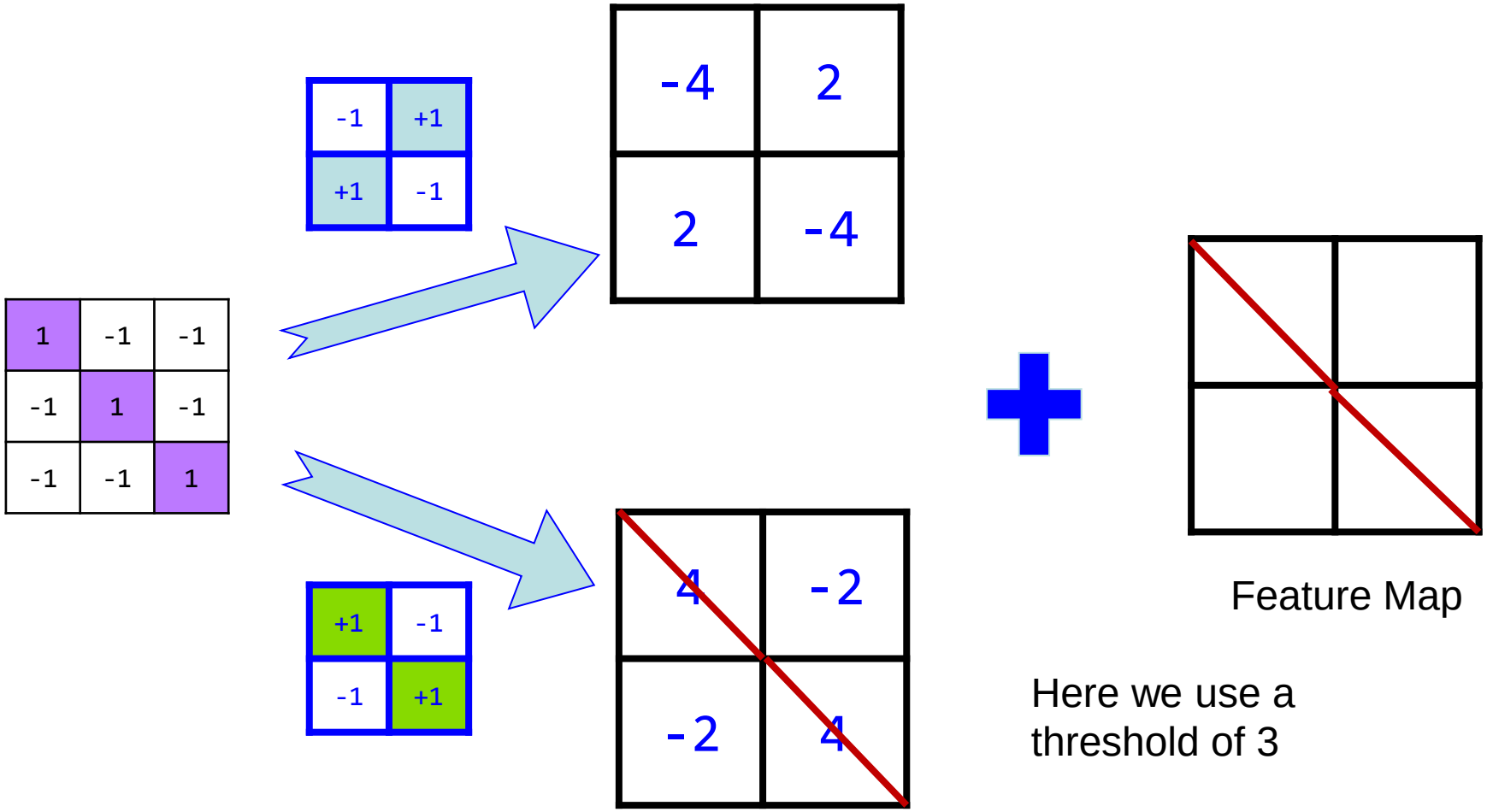
We can do same operation on “O”



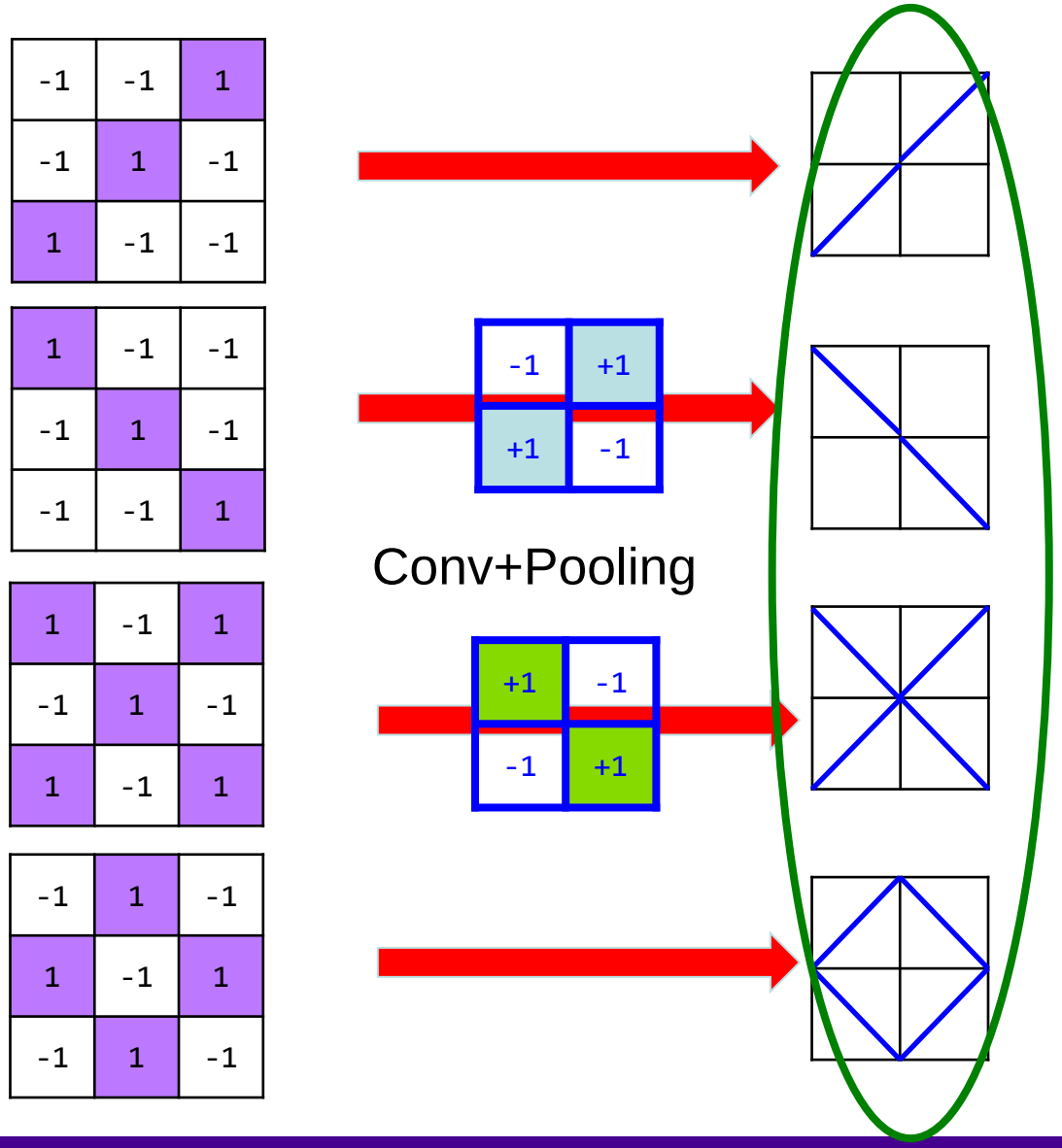
Do the same operation on forward slash "/"



Do the same operation on backward slash "\"



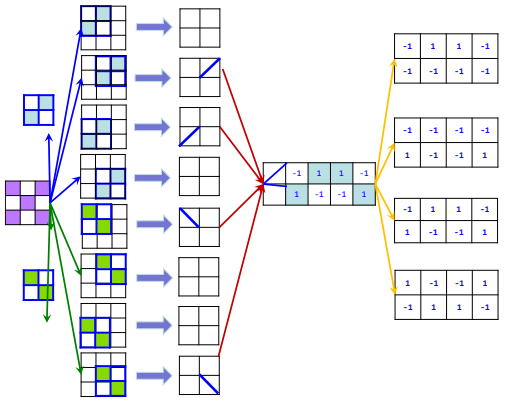
A short summary



- Convolution layer
  - ✓ Use small filter to slide over the large image
- Pooling layer
  - ✓ Check if we find something using a threshold

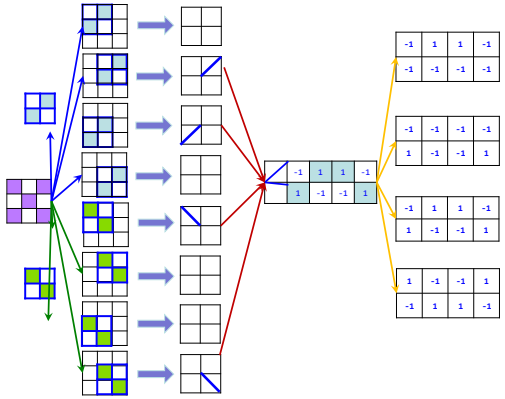
What about the fully connected layer?

How do we obtain filters?

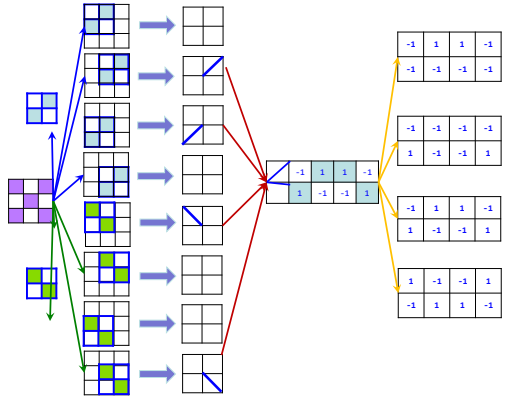


High error

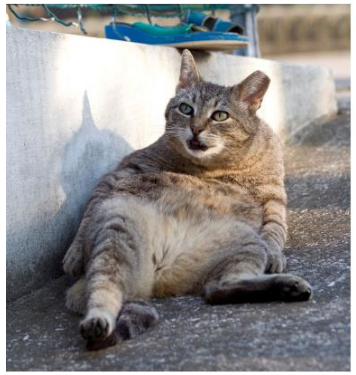
Gradient Descent

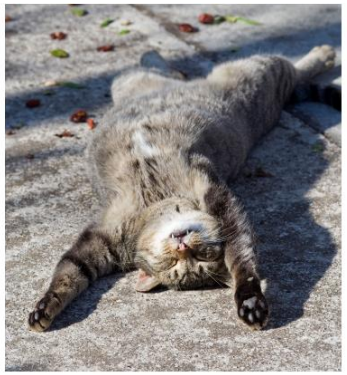


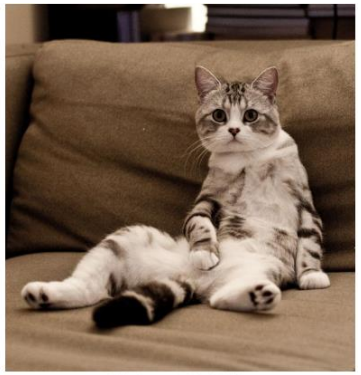
Low error



Let's go back to real world!

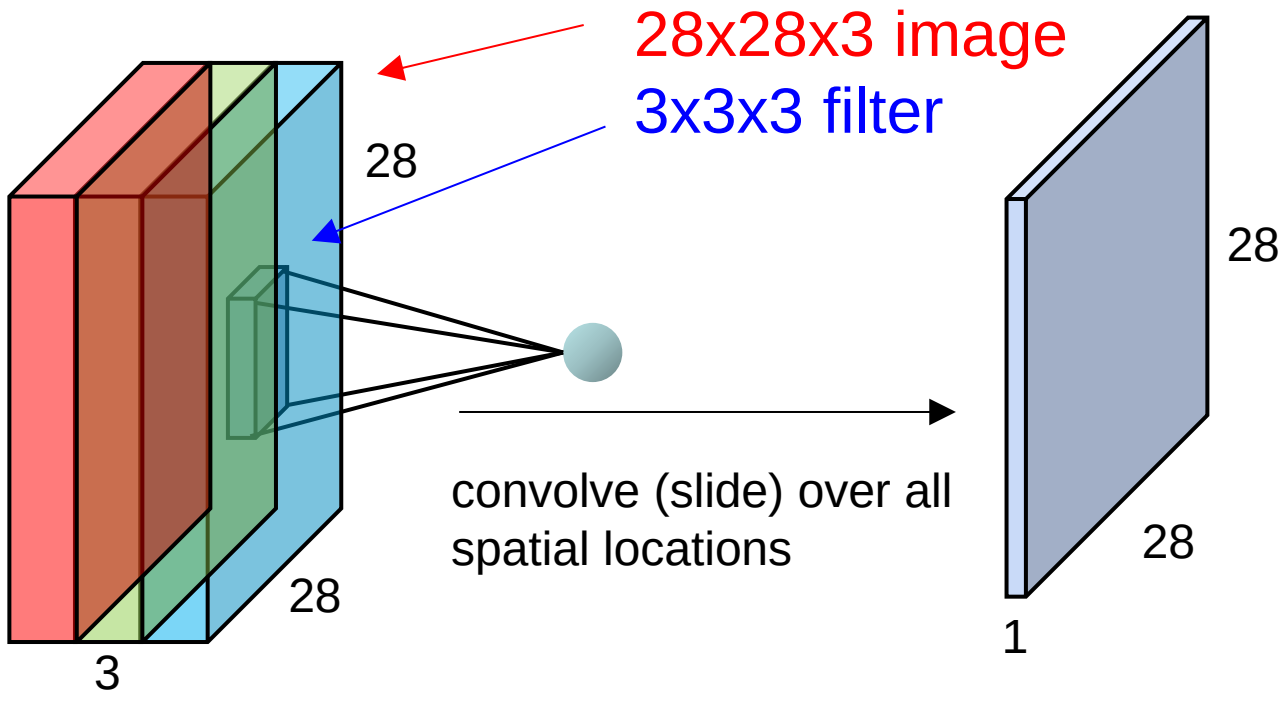


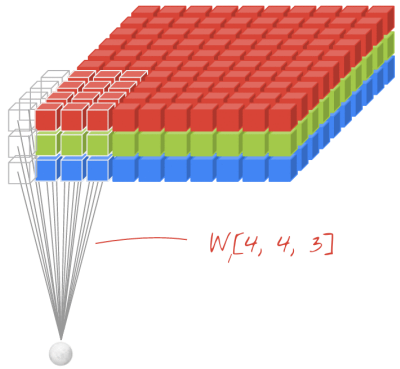




How Does CNN Work?

activation map

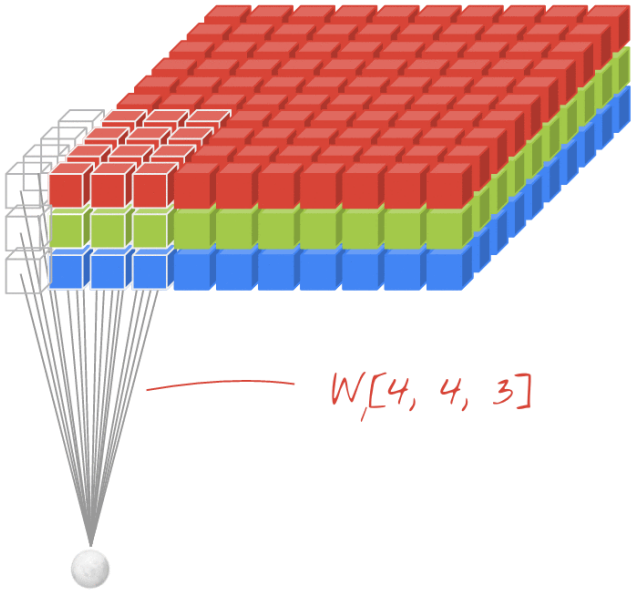


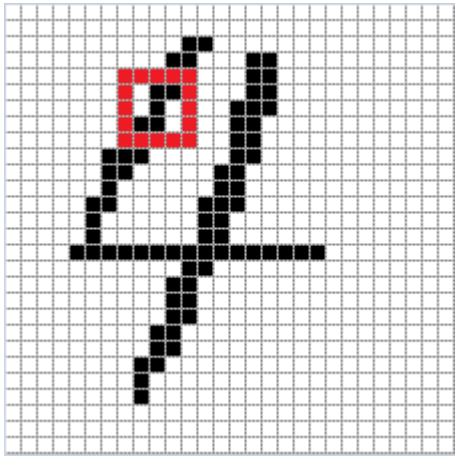


- **By sliding the patch of weights (filter) across the image in both directions (a convolution) you obtain as many output values as there were pixels in the image (some padding is necessary at the edges).**

Three basic ideas about CNN

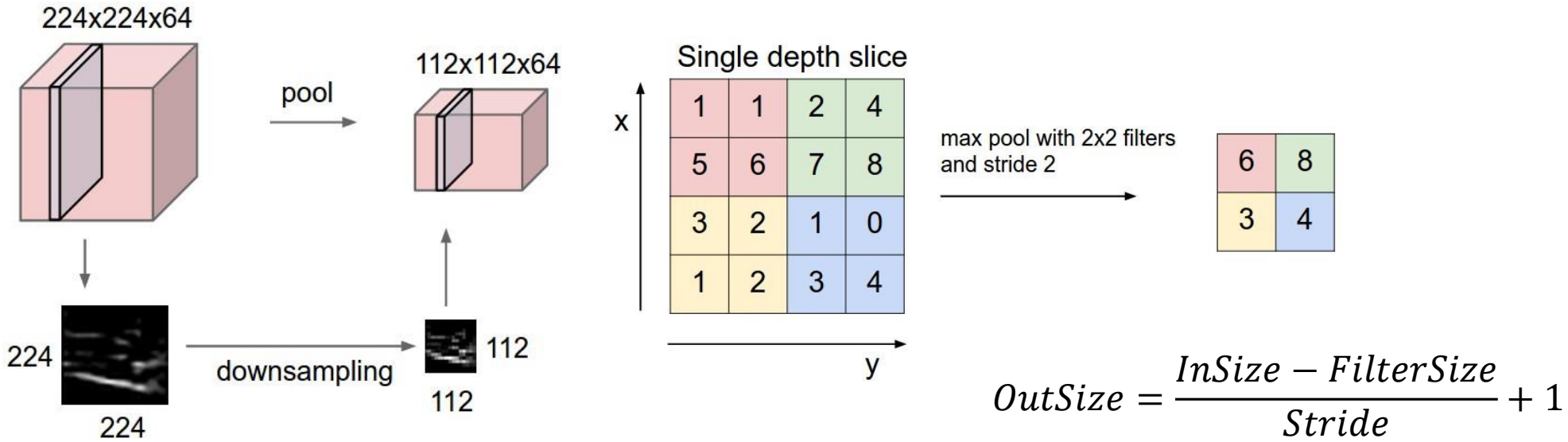
- **Filter (Local receptive fields)**
- **Shared weights and biases:**
- **Pooling**





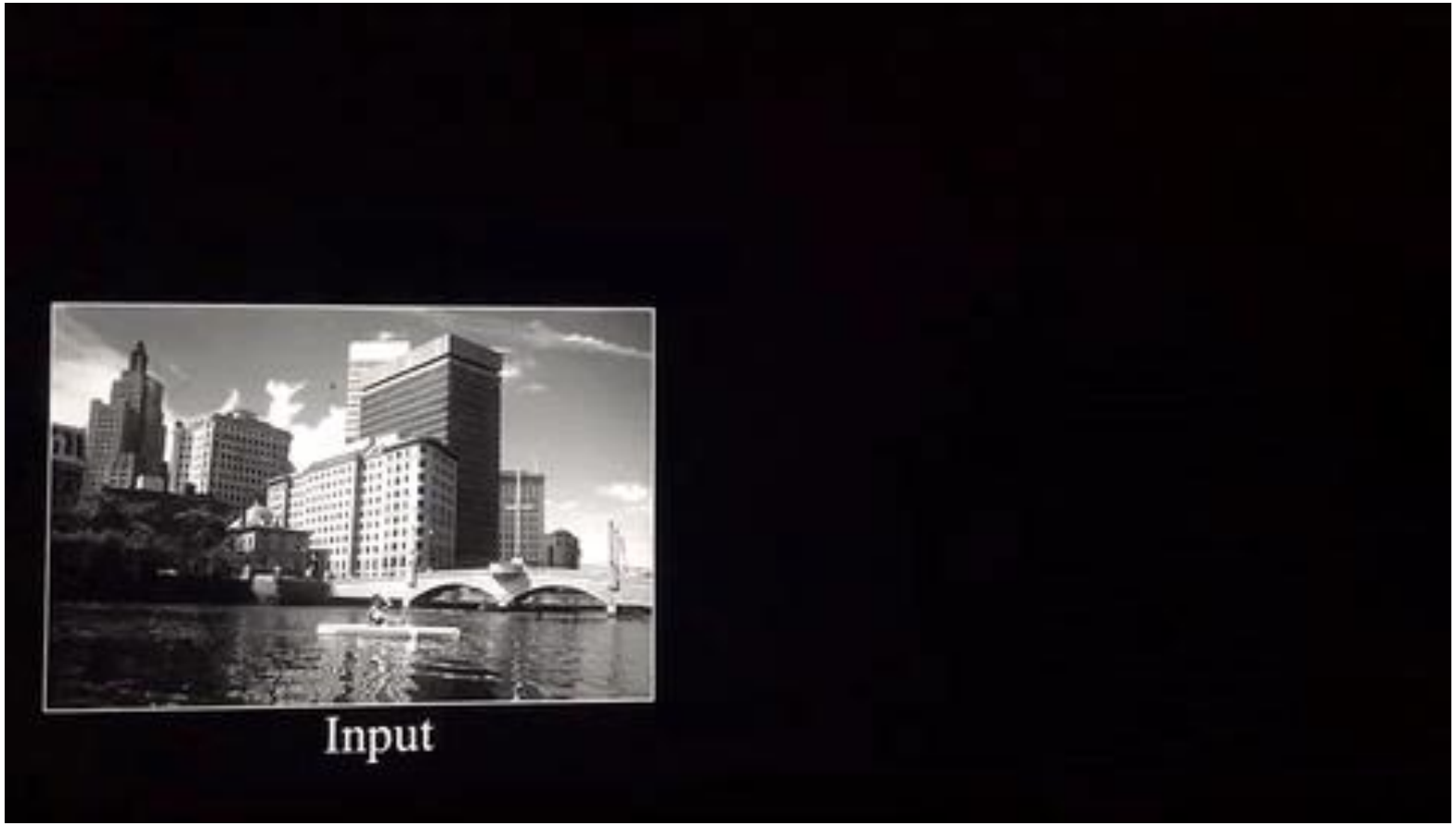
Pooling Layer

- **Convolutional neural networks also contain pooling layers. Pooling layers are usually used immediately after convolutional layers.**
- **What the pooling layers do is simplify the information in the output from the convolutional layer.**
- **We can think of max-pooling as a way for the network to ask whether a given feature is found anywhere in a region of the image. It then throws away the exact positional information.**



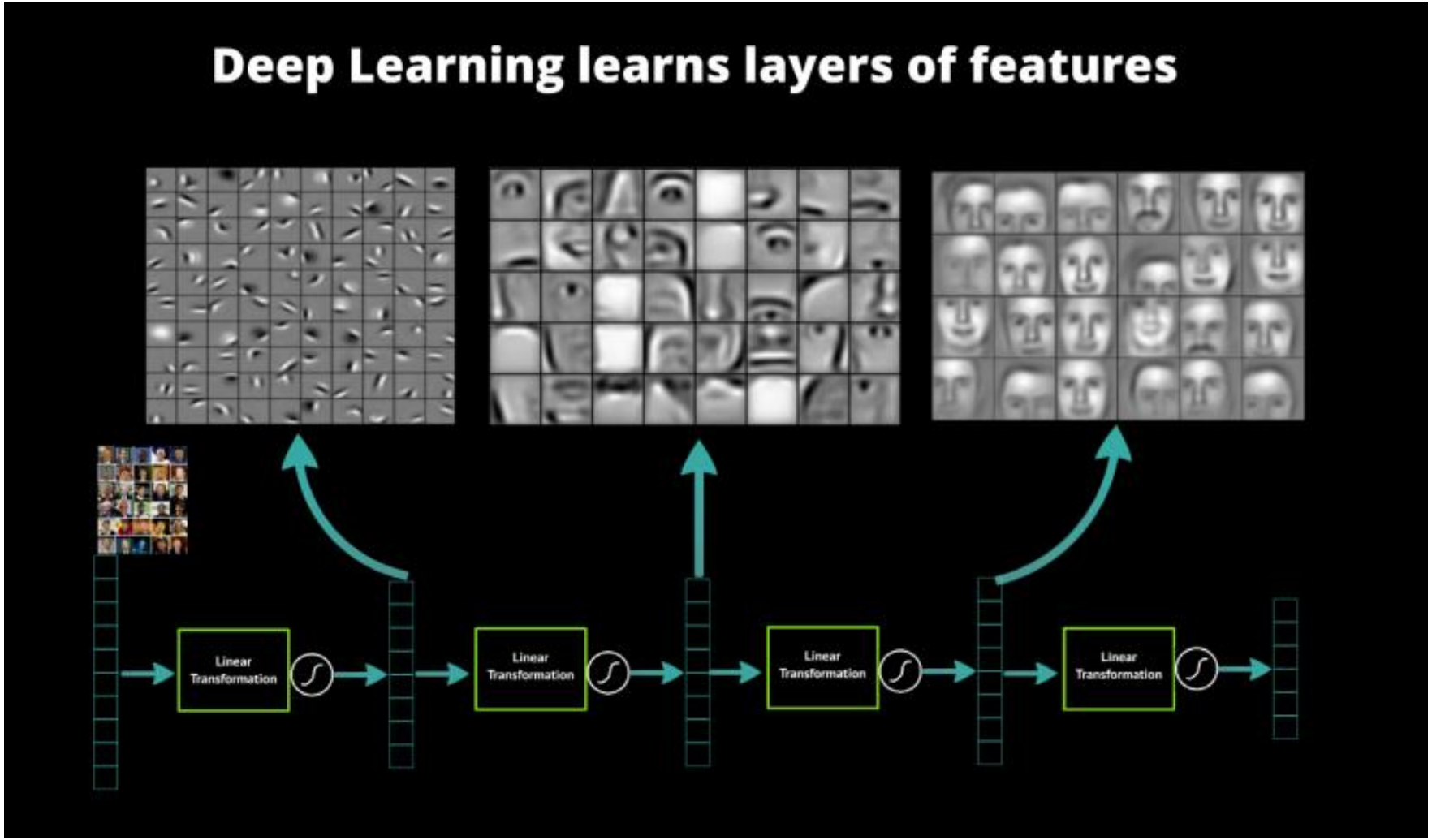
$$OutSize = \frac{InSize - FilterSize}{Stride} + 1$$

Feature Maps

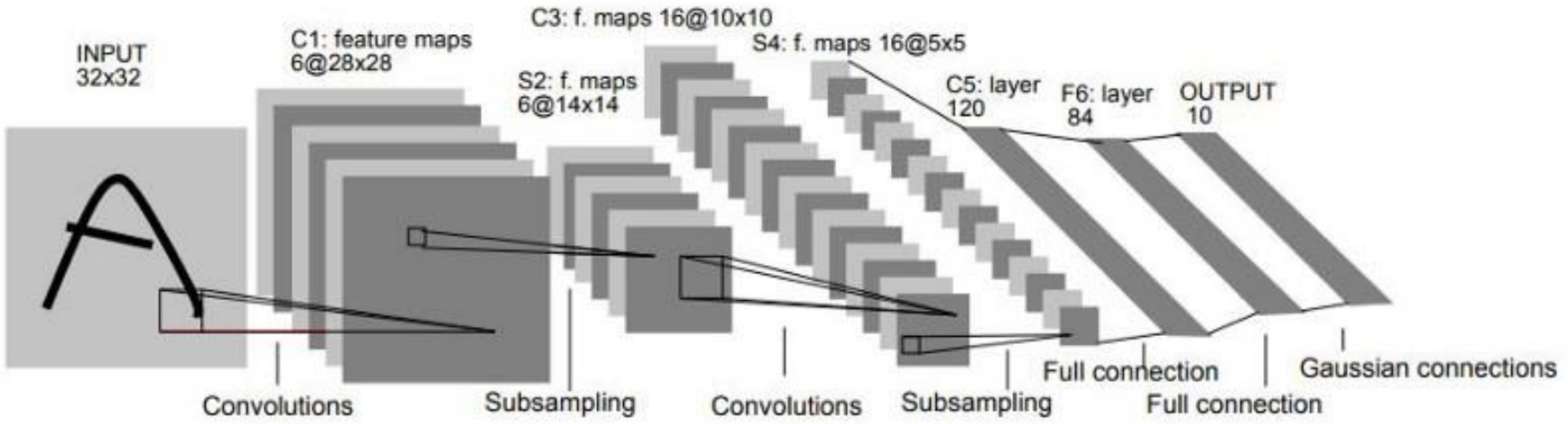


Source: https://medium.com/@chriskevin_80184/feature-maps-ee8e11a71f9e

Deep learning learns layers of features

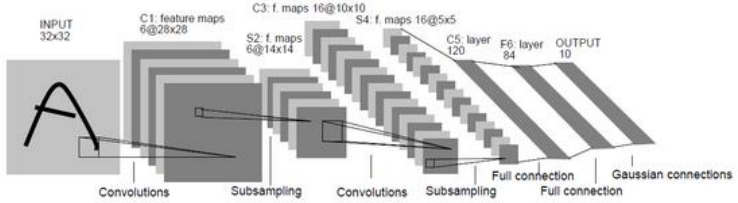


A classical CNN example: LeNet (LeCun et al., 1998)

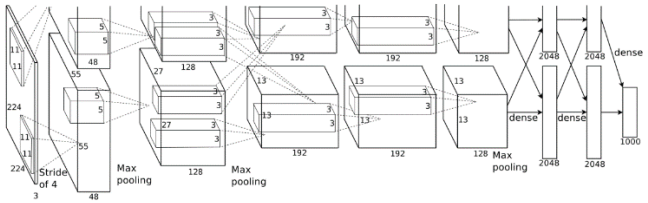


Examples of Convolution NN

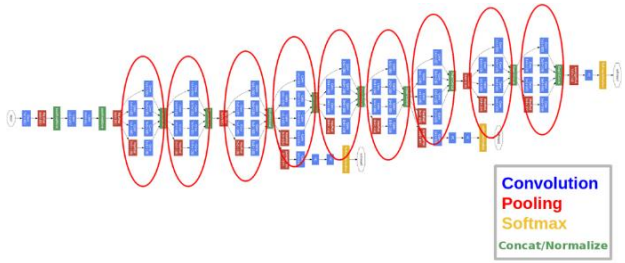
- **LeNet (1998)**



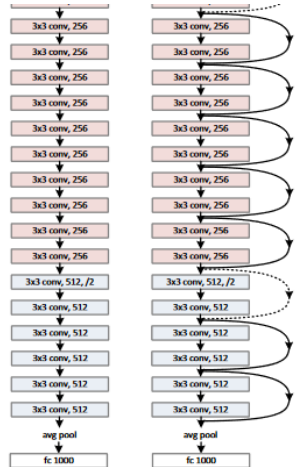
- **AlexNet (2012)**



- **GoogleLeNet (2014)**



- **ResNet (2015)**



- **Transformer (2017)**

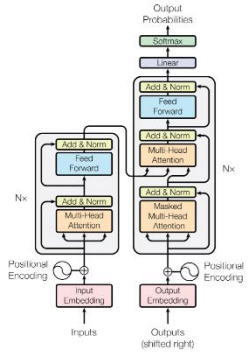


Figure 1: The Transformer - model architecture.

- **Bert (2018)**