



Week 10 Lecture 09

Artificial Neural Network

Wan Fang

Southern University of Science and Technology

Agenda

- Single Artificial Neuron
- Multi-Layer Perceptron (MLP)
- Practice
- Convolutional Neural Networks (CNN)

An example of acceptance at a University



Student A: (8, 2)

Student B: (5, 5)

Linear Regression

- **w**
 - The weight determines the influence of each feature on our prediction, usually a vector form with w_i
- **b**
 - The bias says what value the predicted price should take when all features take 0
- Given a dataset, our goal is
 - To choose the weights **w** and bias b such that on average, the predictions made based on our model best fit the true prices observed in the data.

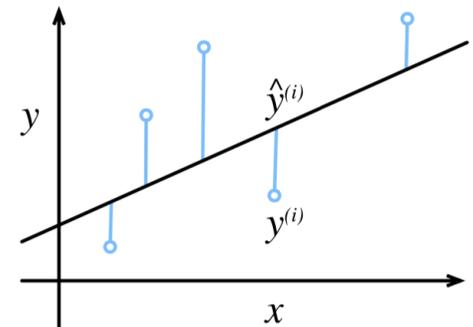
$$\hat{y} = w_1 \cdot x_1 + \dots + w_d \cdot x_d + b \longrightarrow \hat{y} = \mathbf{w}^T \mathbf{x} + b.$$

Loss Function

- To quantify the distance between the *predicted* and *real* value.
 - Perfect predictions incur a loss of 0

- The Sum of Squared Errors $l^{(i)}(\mathbf{w}, b) = \frac{1}{2} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$

- Loss Function as an averaged SSE

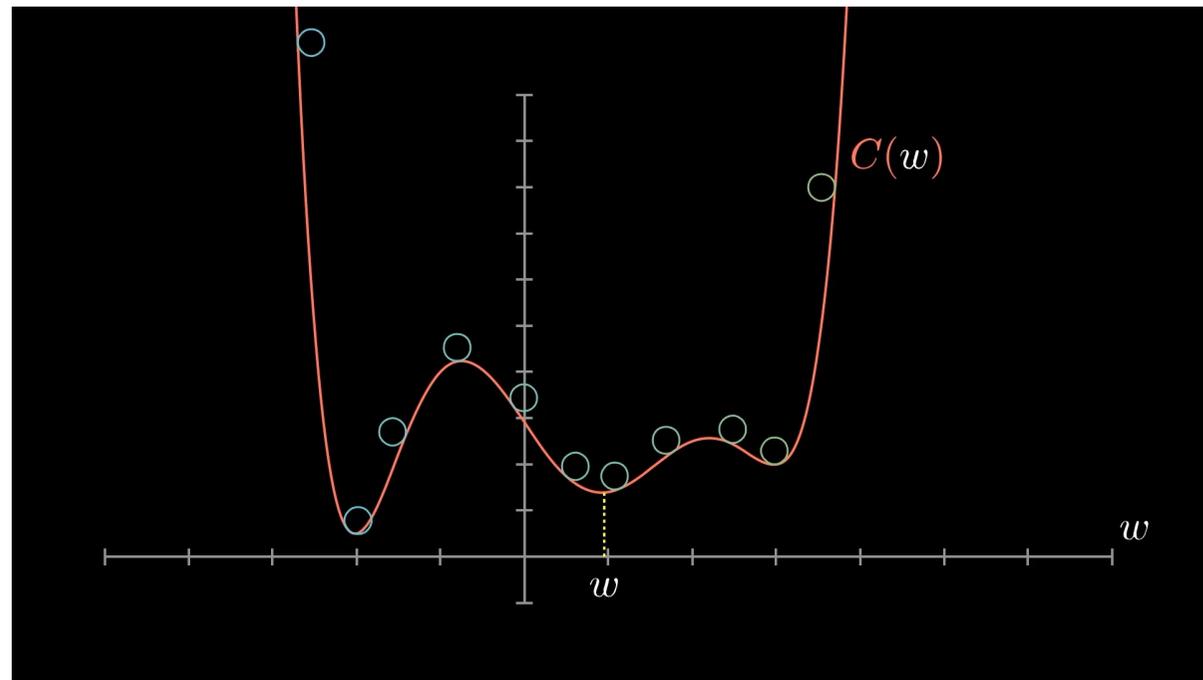
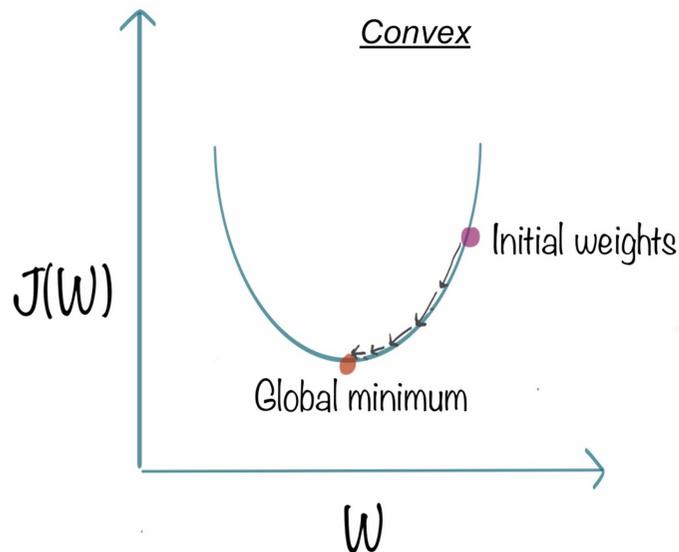


$$L(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^n l^{(i)}(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \left(\mathbf{w}^T \mathbf{x}^{(i)} + b - y^{(i)} \right)^2$$

$$\mathbf{w}^*, b^* = \underset{\mathbf{w}, b}{\operatorname{argmin}} L(\mathbf{w}, b)$$

Gradient Descent

- **Iteratively reducing** the error by updating the parameters in the direction that incrementally lowers the loss function
 - On *convex* loss surfaces, it will eventually converge to a global minimum
 - For *nonconvex* surfaces, it will at least lead towards a (hopefully good) local minimum.

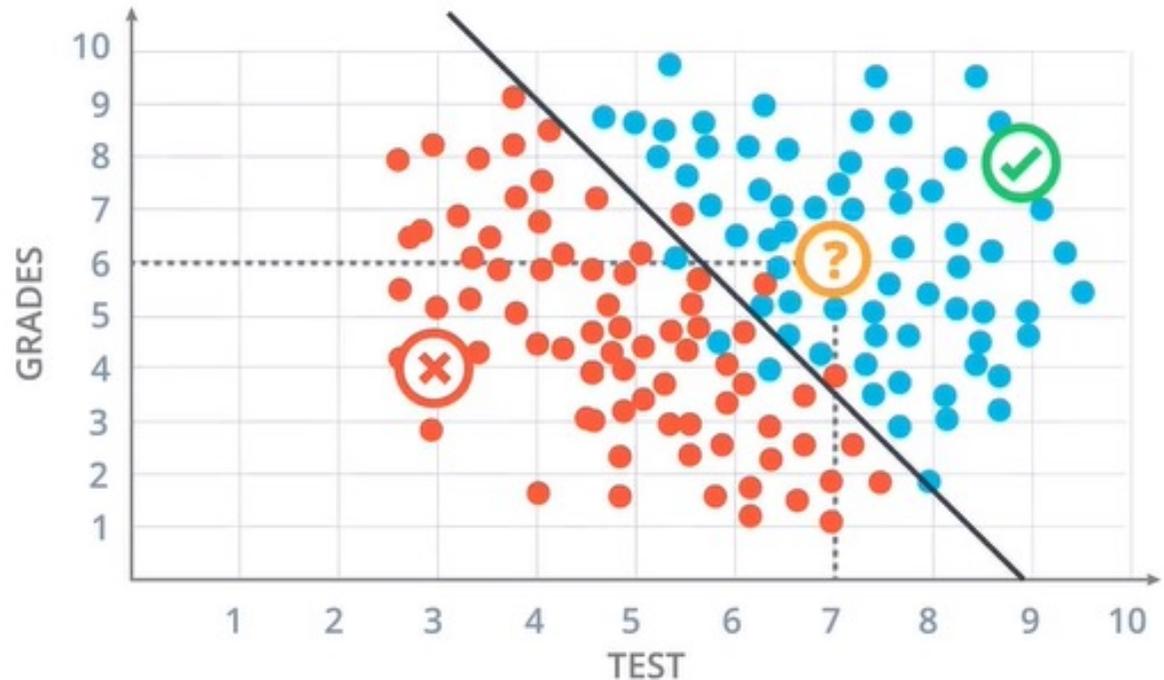


- The key technique for optimizing *nearly any* deep learning model

Linear Classification

- Hypothesis
 - Acceptance depending on Test and Grade
- Data
 - $(x^{(i)}, y^{(i)})$
- Input
 - $x_1^{(i)}$ as test scores and $x_2^{(i)}$ as test scores
- Output
 - $\hat{y}^{(i)}$ as a threshold decision of **Accept** or **Reject**
- Model
 - A linear boundary line to separate the data
 - $w_1x_1 + w_2x_2 + b = 0$
 - A threshold to activate a decision against the line
 - > 0 : **Accept**; < 0 : **Reject**

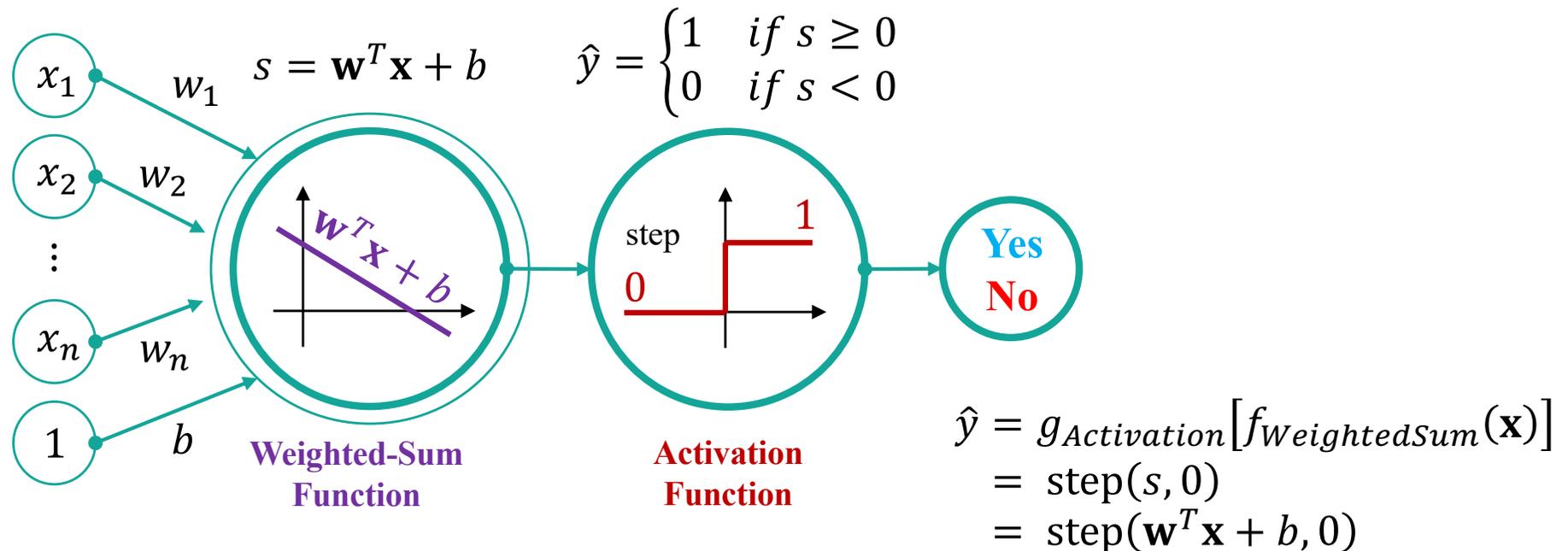
An example of acceptance at a University



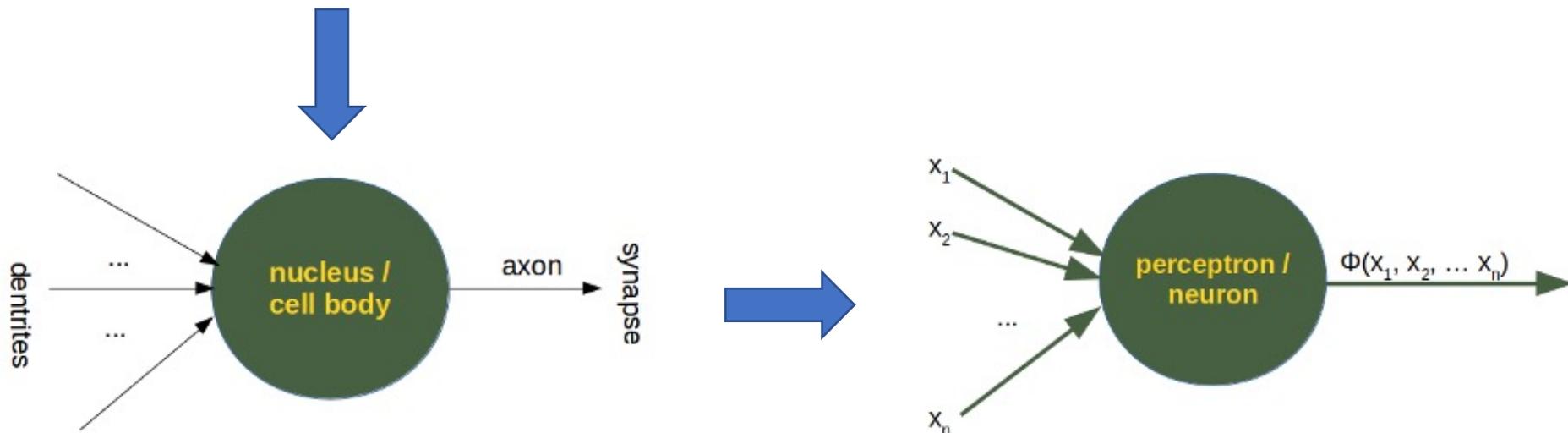
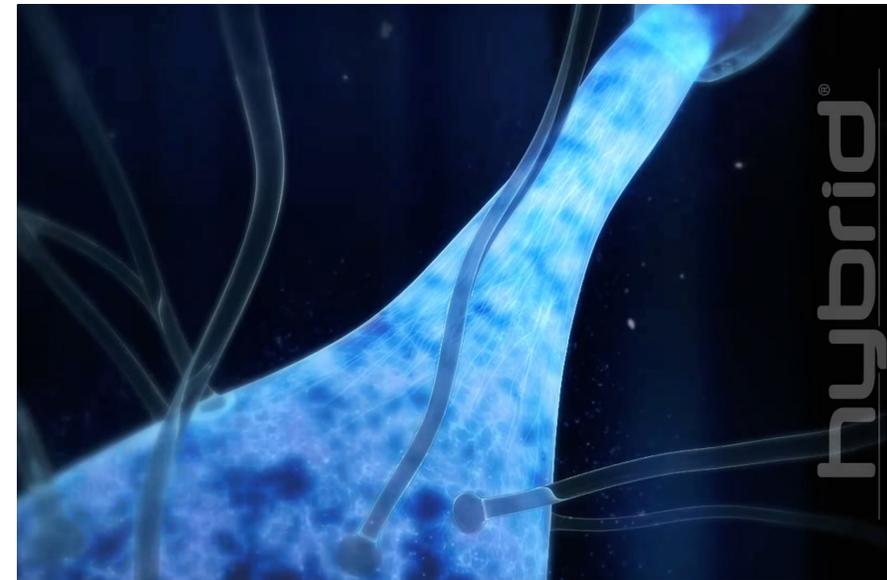
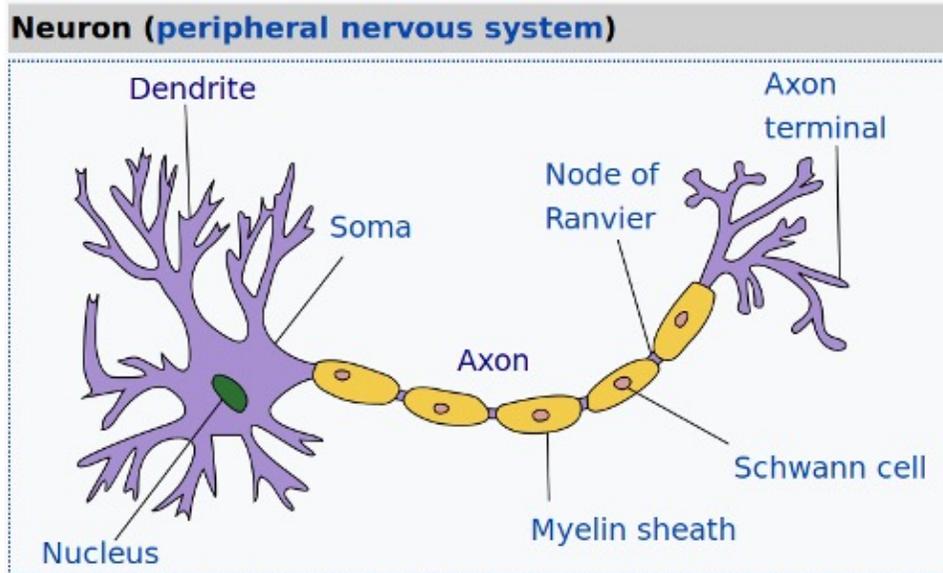
A Linear Boundary Line of $2x_1 + x_2 - 18 = 0$
as a decision criteria from regression to classification

Perceptron with an Activation Function

- An Artificial Neuron with two nodes
 - **Weighted-sum node**
 - Calculate a linear equation $s(x)$ with inputs on the weights plus bias
 - **Activation node**
 - Apply the step function to get the predicted result $\hat{y}(s)$



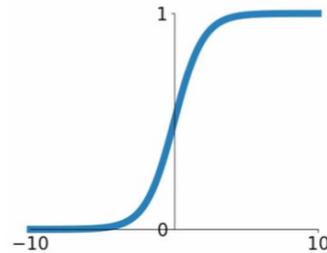
A Perceptron as an Artificial Neuron



Activation Function

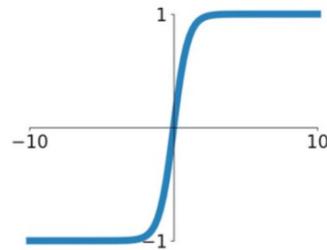
Sigmoid

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



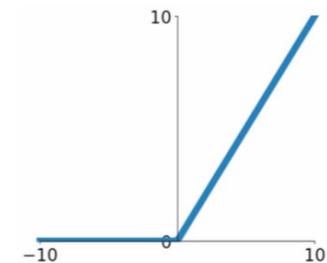
tanh

$$\tanh(x)$$



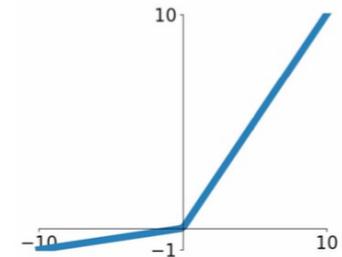
ReLU

$$\max(0, x)$$



Leaky ReLU

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

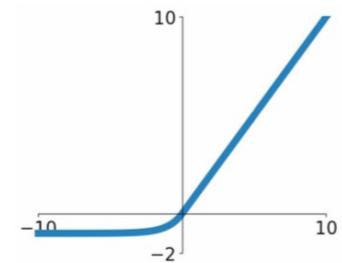


Maxout

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

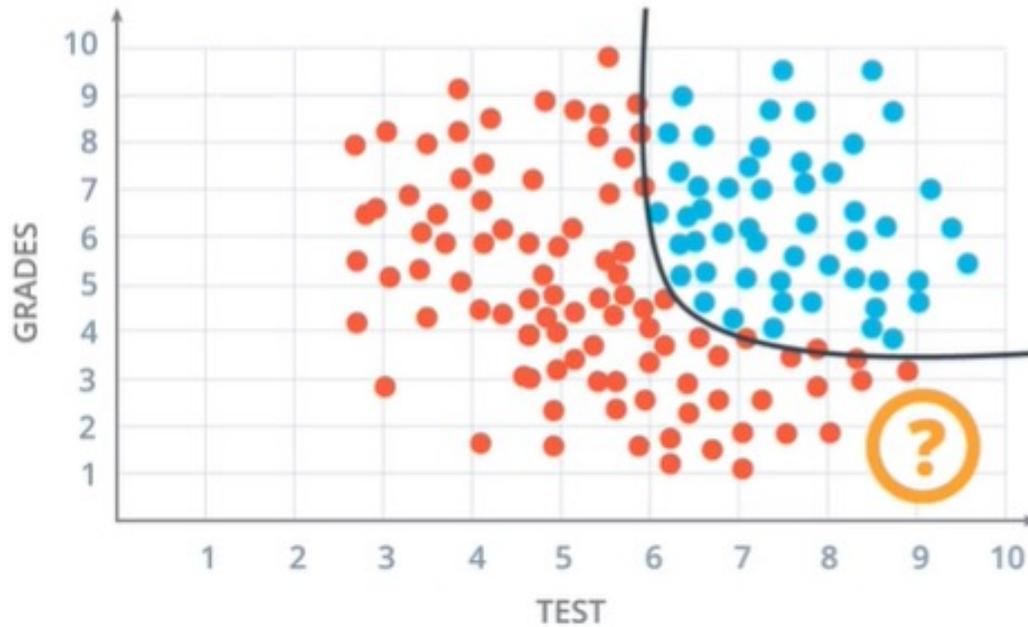
ELU

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



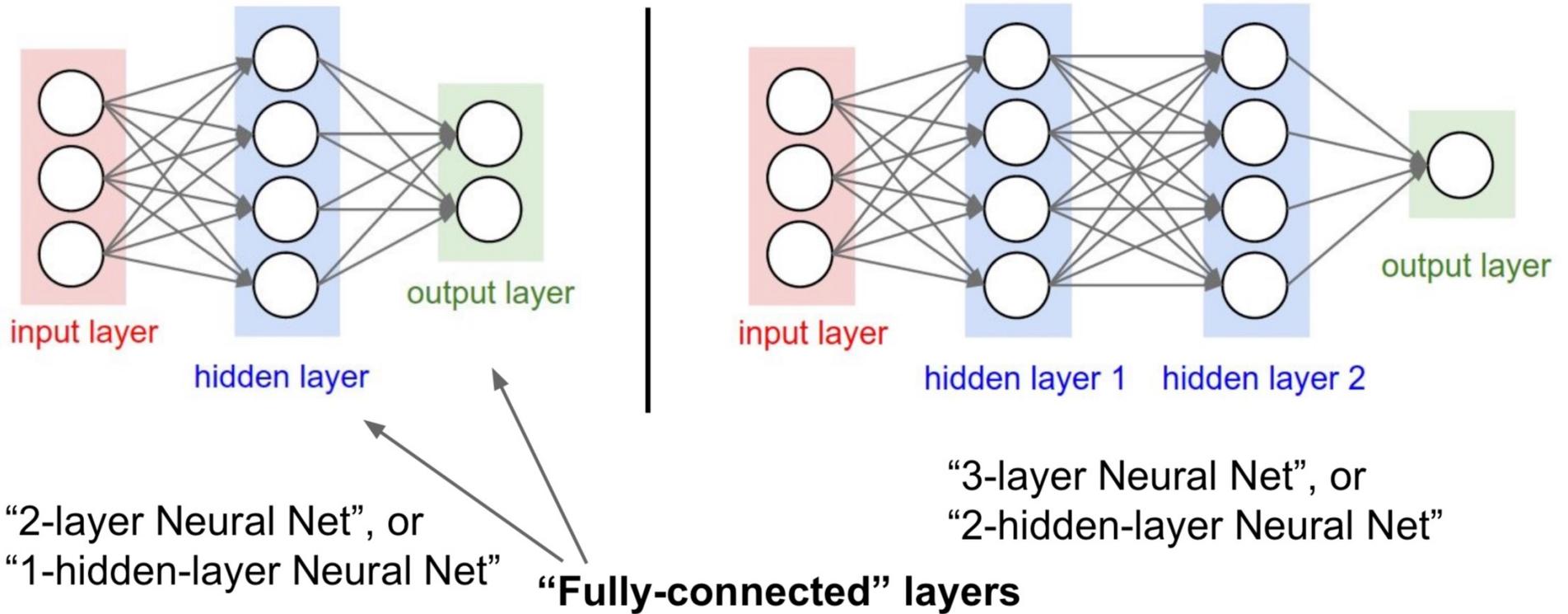
Limitation of Single Perceptron

What if the boundary line is non-linear?



- Unable to classify nonlinear scenarios

Multi-Layer Perceptrons



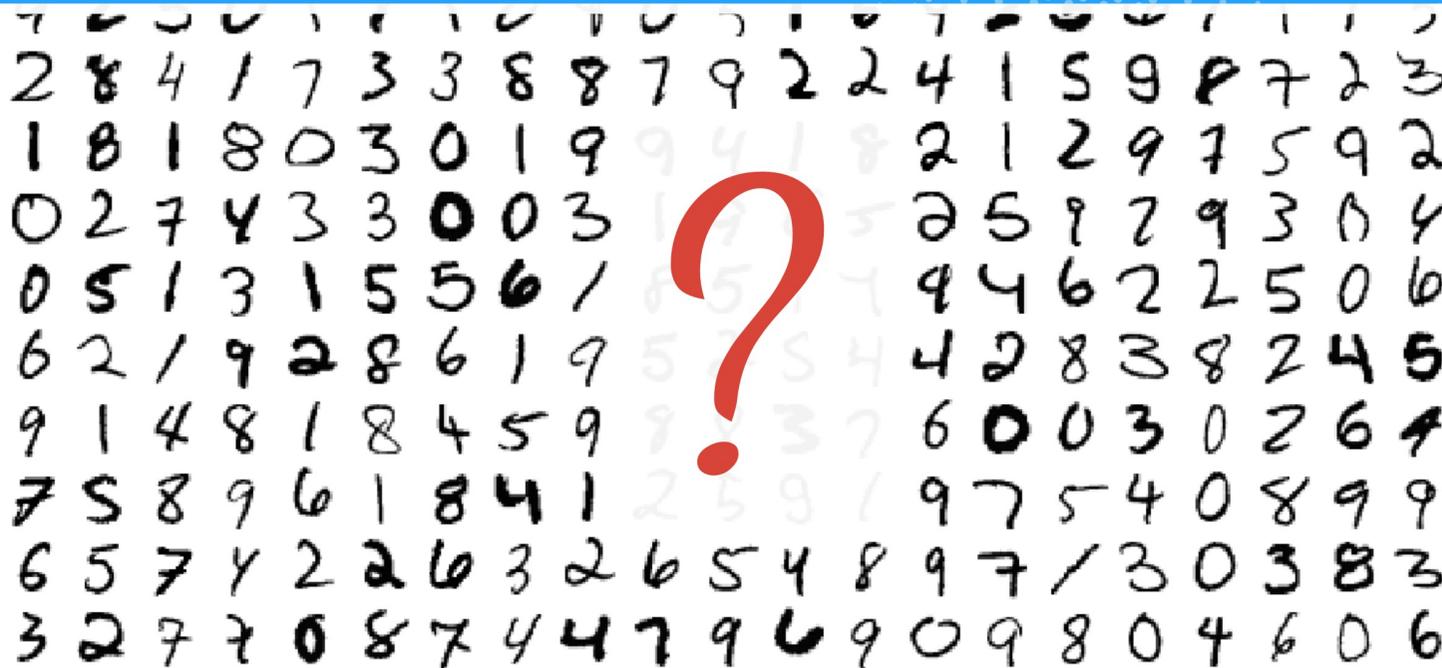
Exercise I

- Build your first neural networks on the website
 - <https://playground.tensorflow.org/>
- Ask the following questions?
 - Play with different data types, features, network structures.
 - Can Neural networks separate nonlinear features?
 - How does nonlinearity come?
 - How important is input features
 - How important is number of neurons and layers of neurons?

Exercise II

A Toy Example of Training a Neural Network

Hello World: handwritten digits classification - MNIST

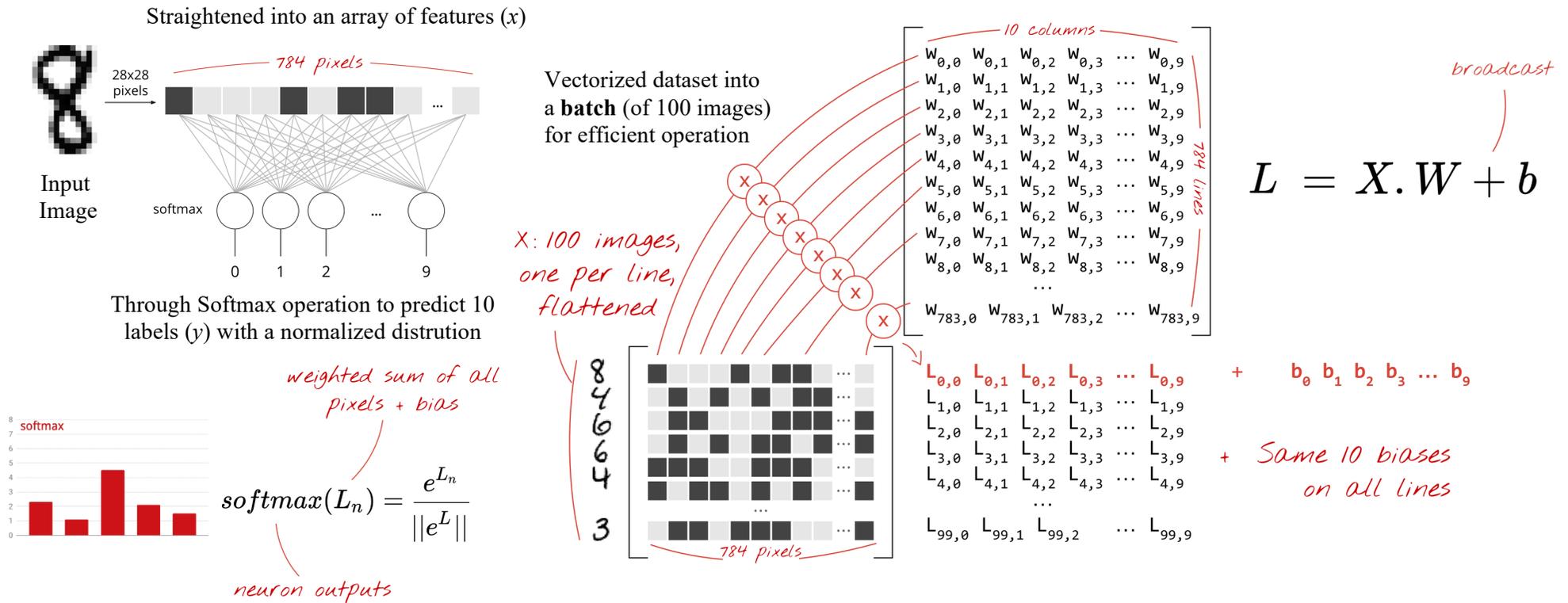


MNIST = Mixed National Institute of Standards and Technology - Download the dataset at <http://yann.lecun.com/exdb/mnist/>

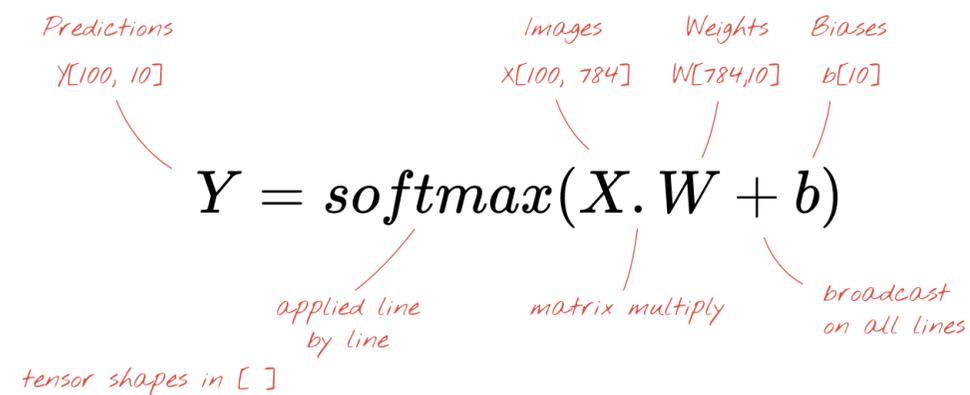
- 28x28 pixel grayscale image
- 60,000 training images
- 10,000 testing images

Multi-Layer Perceptron

A Single-layer Network of Image Classification



Softmax on a Batch of Images



0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0

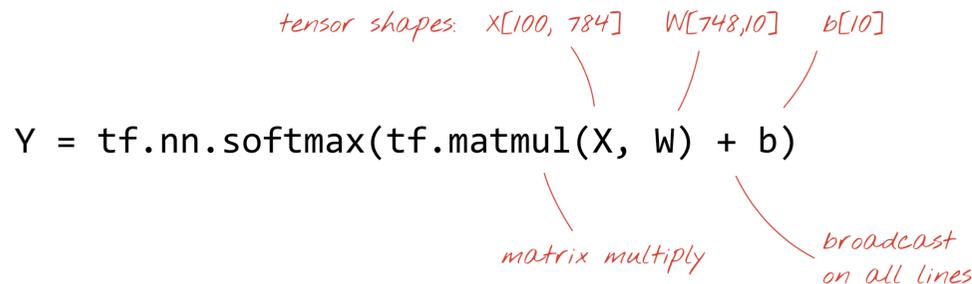
actual probabilities, "one-hot" encoded

$$-\sum Y_i' \cdot \log(Y_i) \text{ Cross entropy}$$

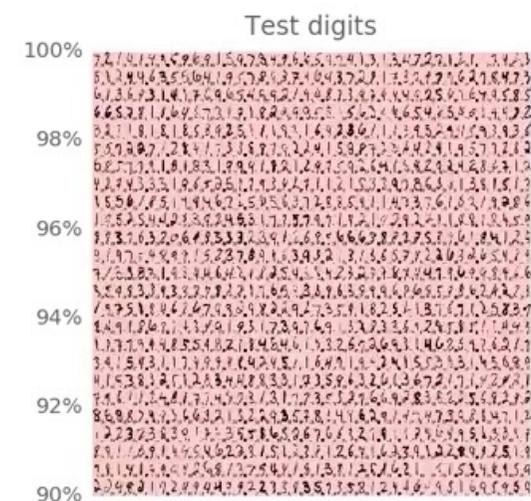
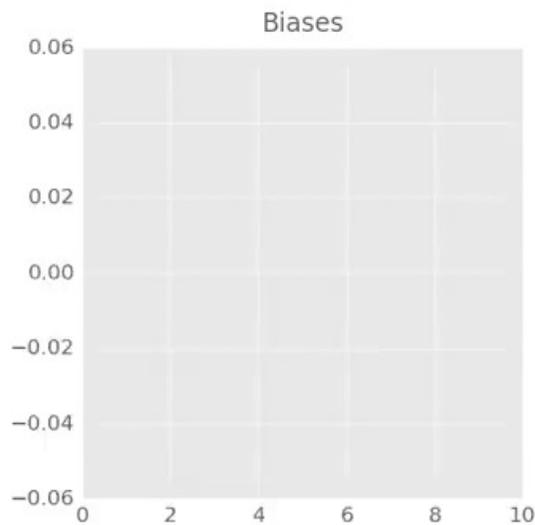
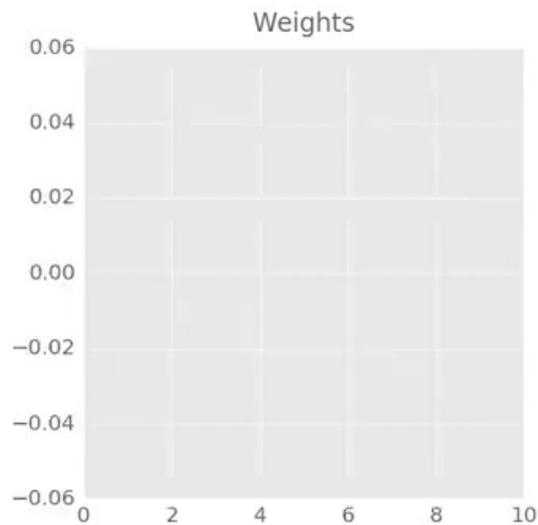
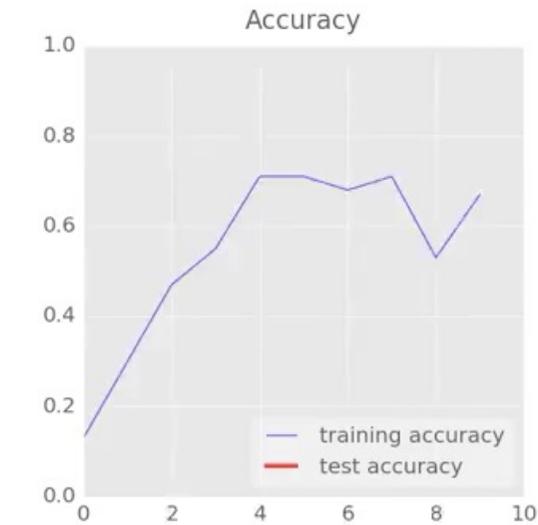
computed probabilities

.01	.03	.00	.04	.03	.05	0.8	.02	.01	.01
0	1	2	3	4	5	6	7	8	9

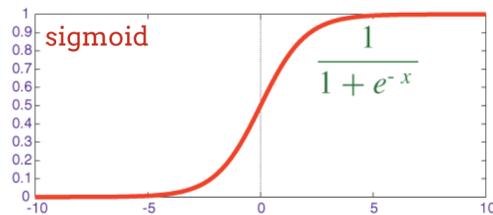
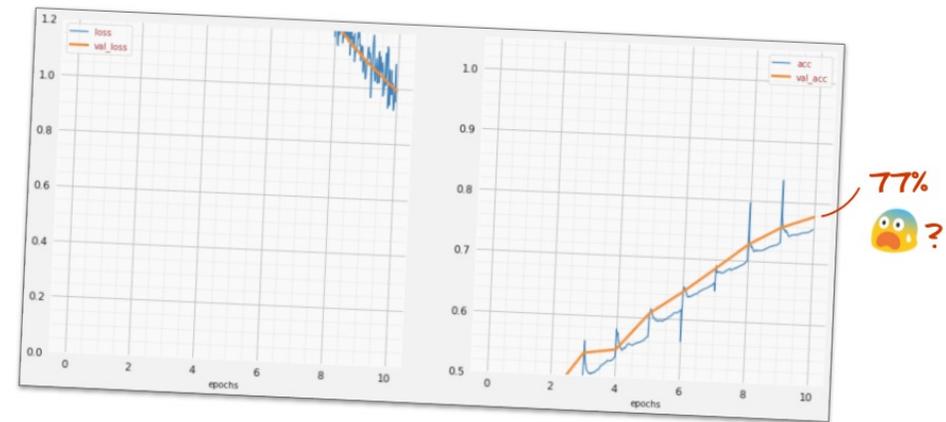
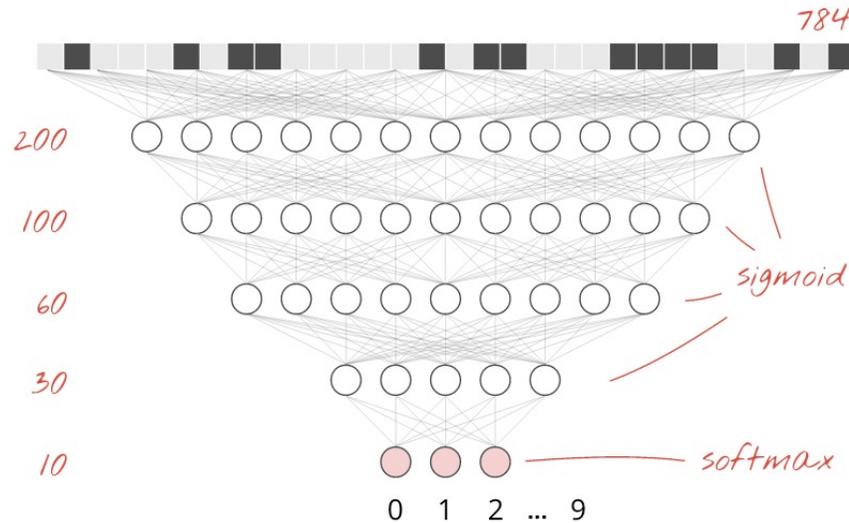
this is a "6"



Training Process



Adding Layers

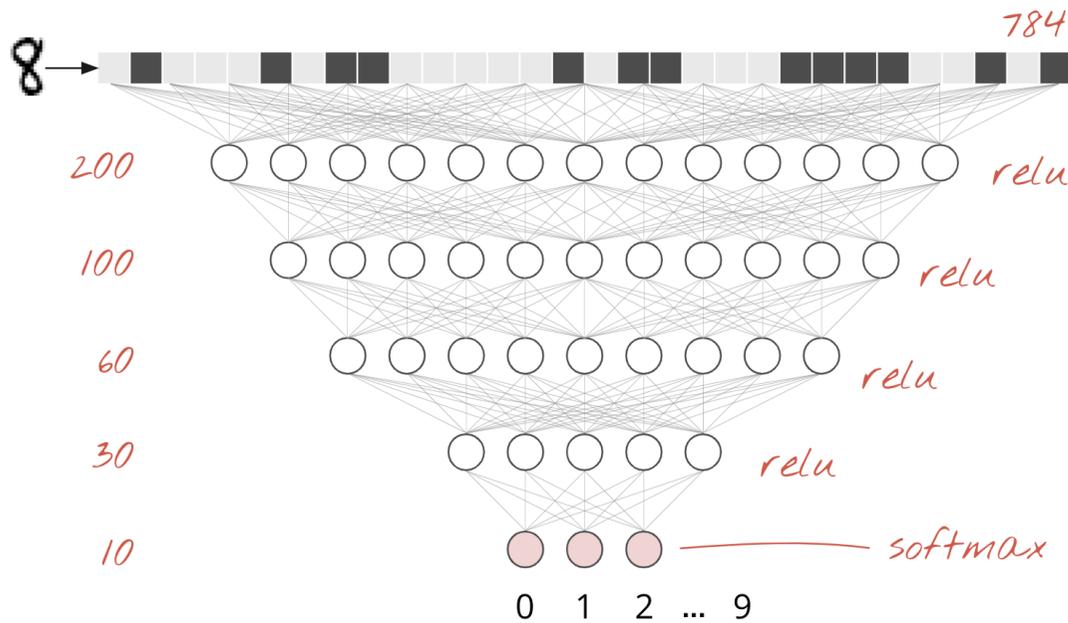


Getting flat

- The gradient can become very small and training get slower and slower.

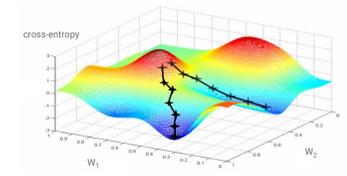
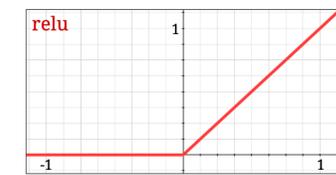
Simply adding more layers with sigmoid activations does not give us the expected results ...

Special Care for Deep Networks

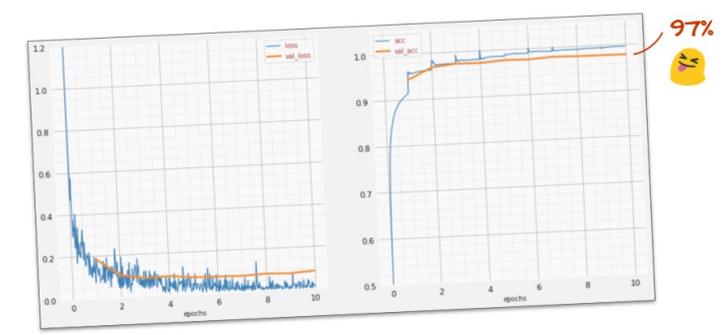


$$Y = \text{relu}\left(\sum_i W_i X_i + b\right)$$

activation weights bias inputs



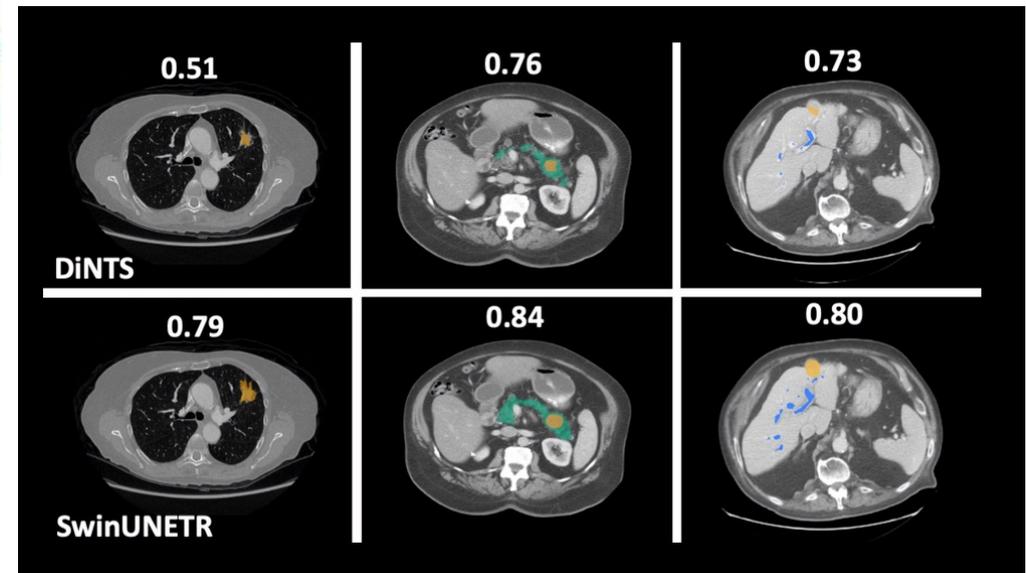
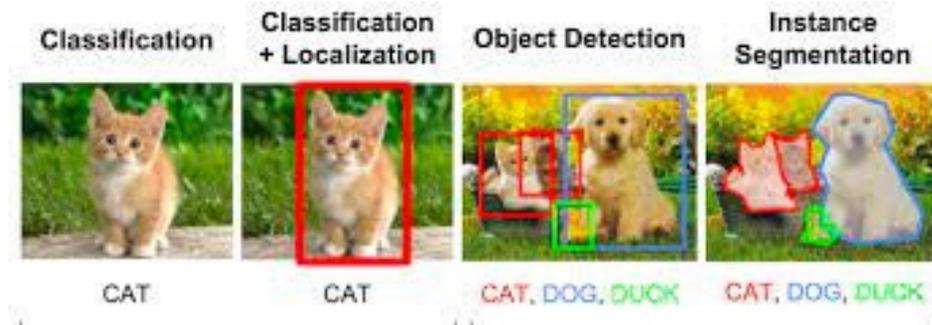
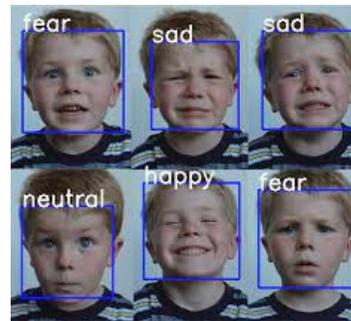
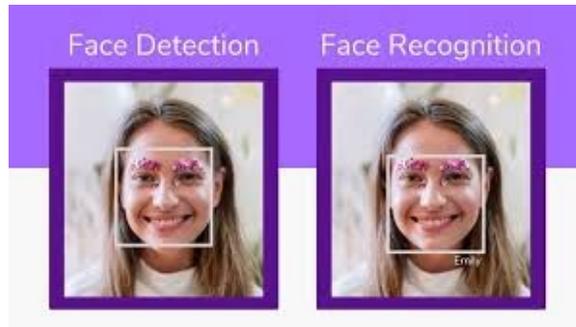
HANDS ON:
Replace the 'sgd' optimizer with a better one, for example 'adam' and train again.



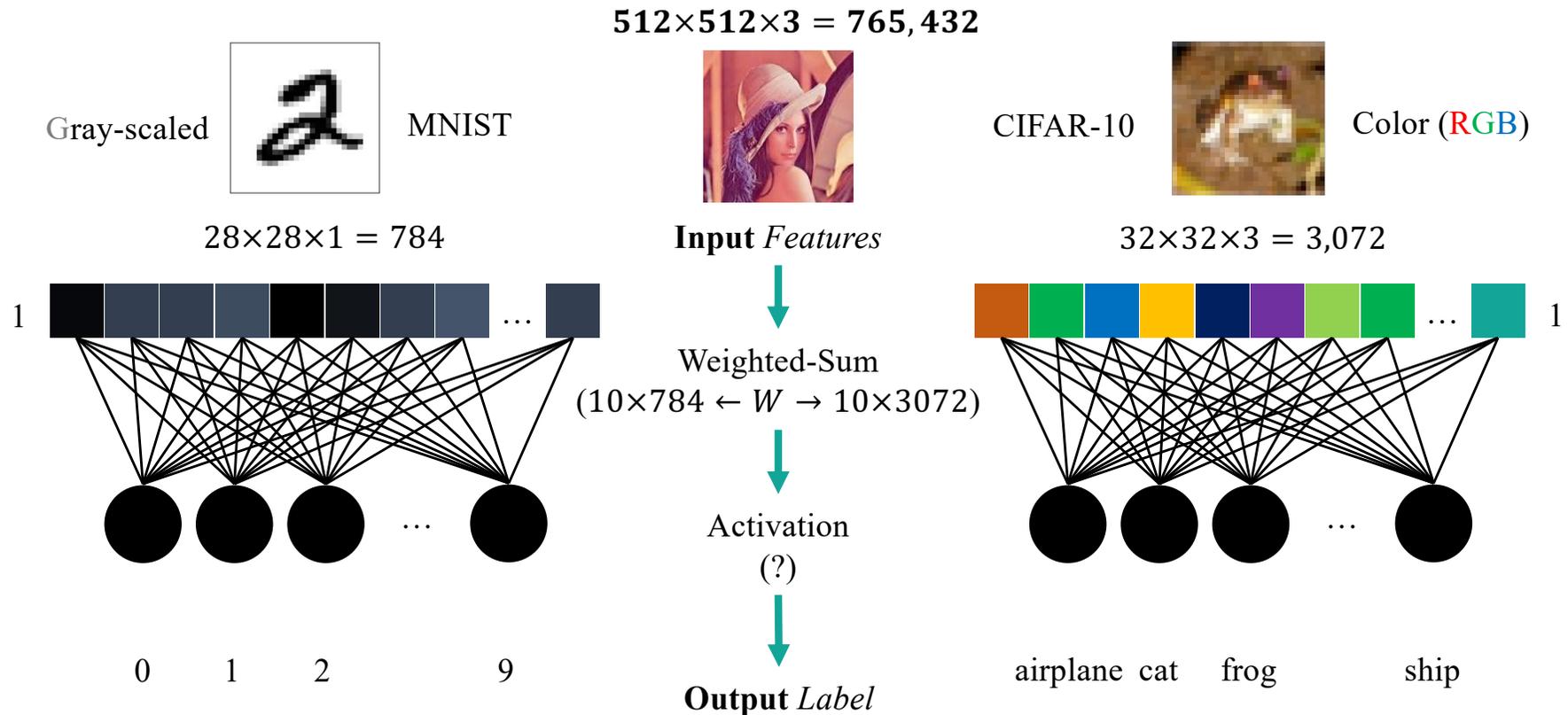
HANDS ON:
Replace all `activation='sigmoid'` with `activation='relu'` in your layers and train again.

Convolutional Neural Network (CNN)

Convolutional Networks Applications

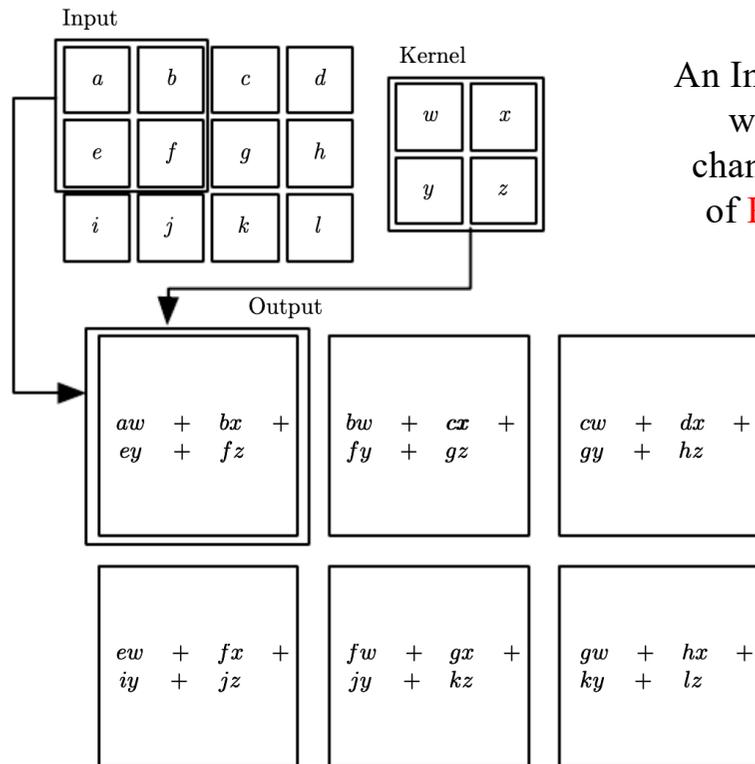


A Design Challenge with Increasing Dimensions

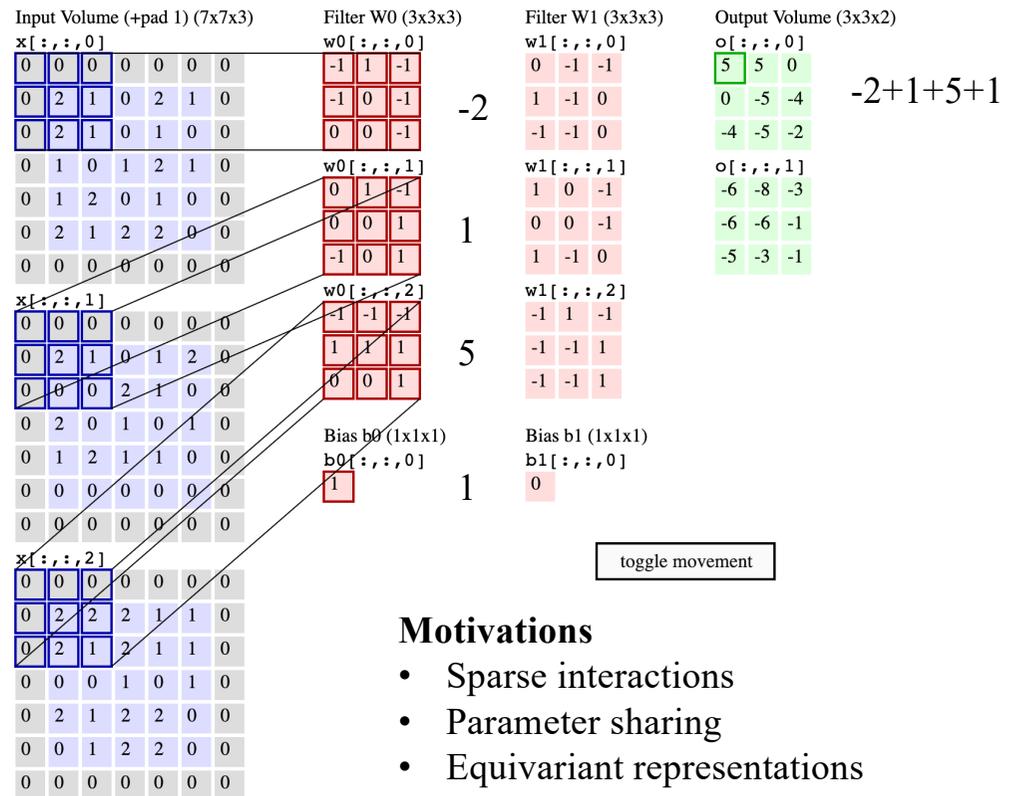


Regular Neural Nets don't scale well to full images

Convolutional Operation

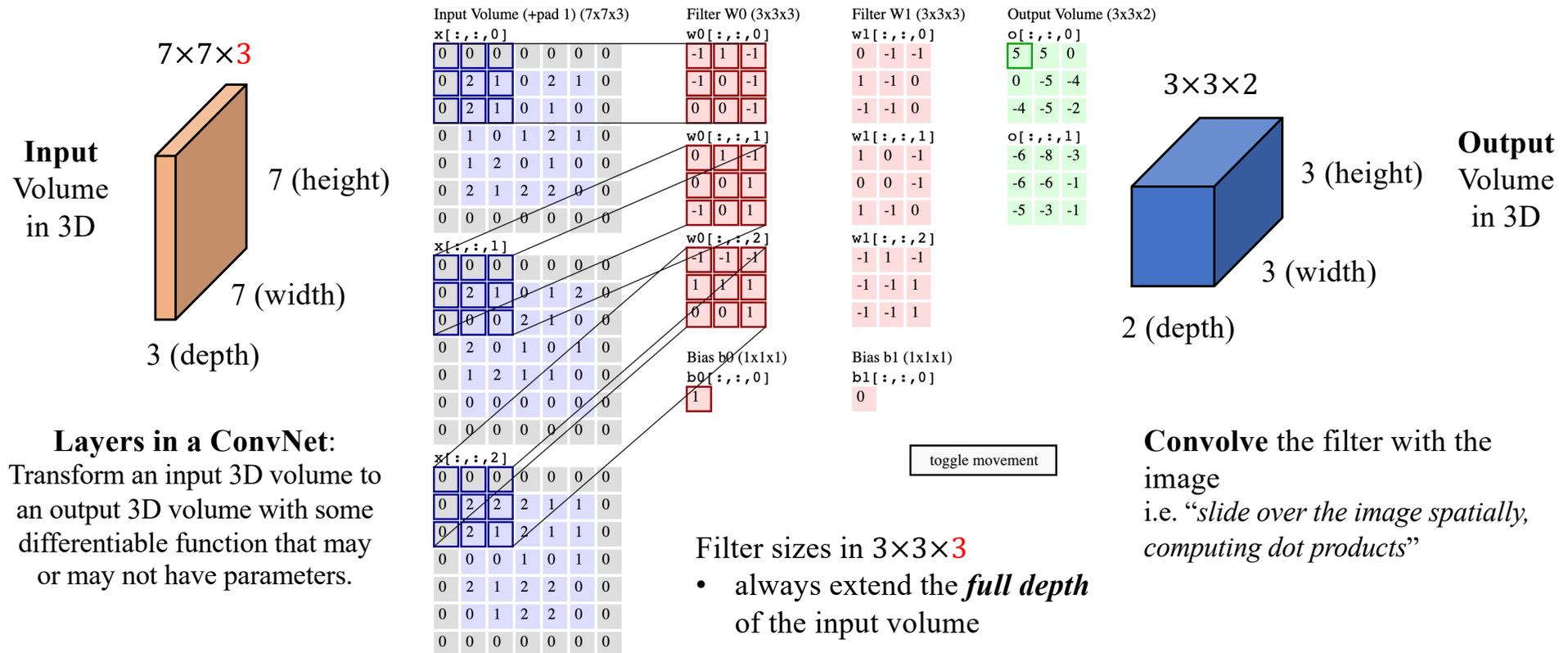


An Image with 3 channels of RGB



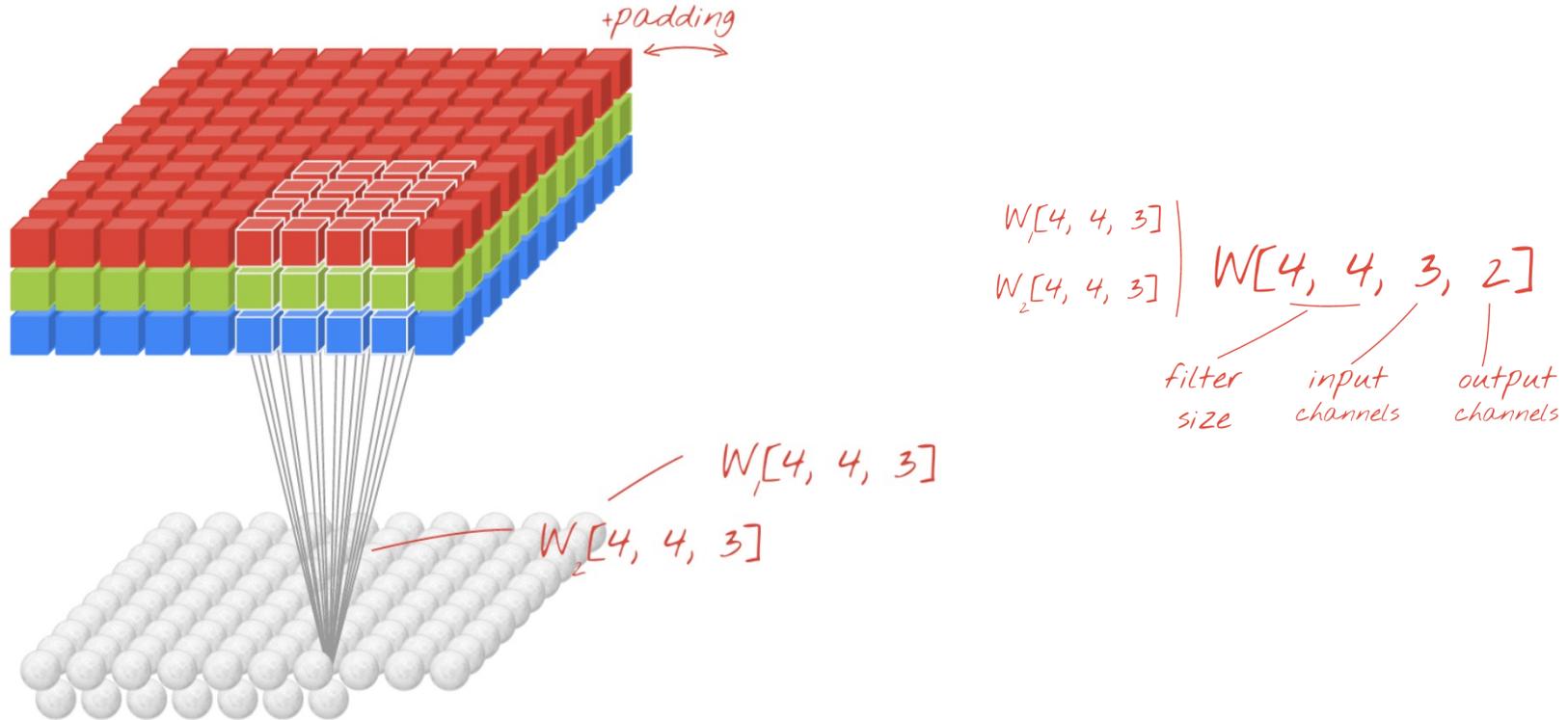
Convolution in 3D Volumes

Preserved spatial structure between the input and output volumes in width, height, number of channels

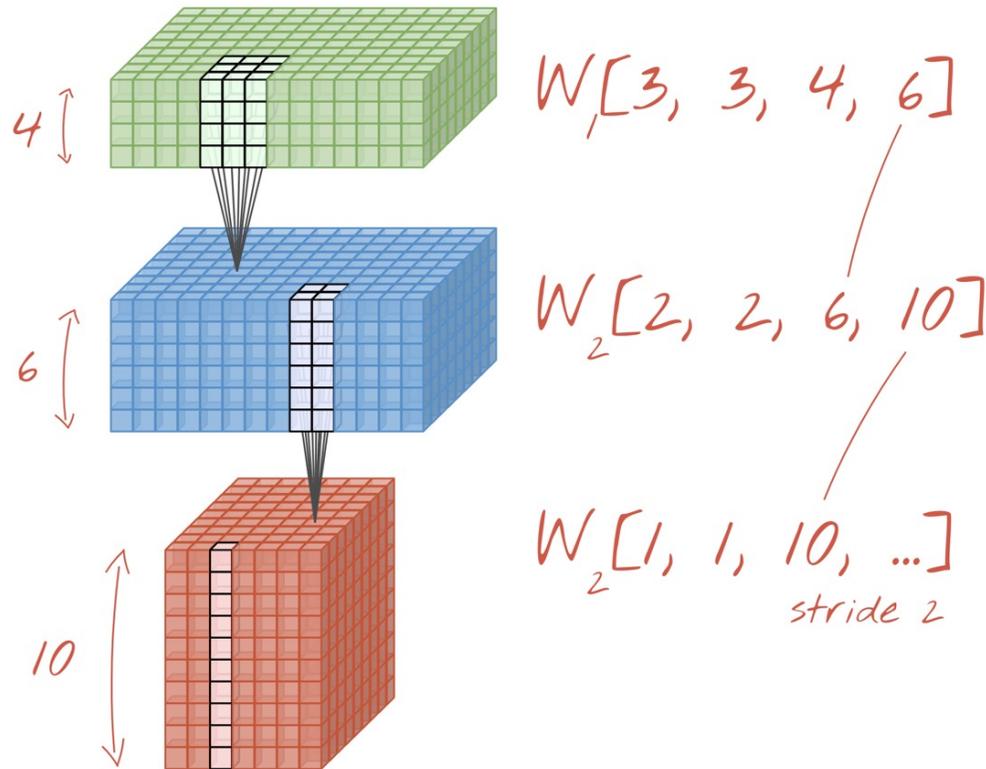


The Design of a Convolutional Layer

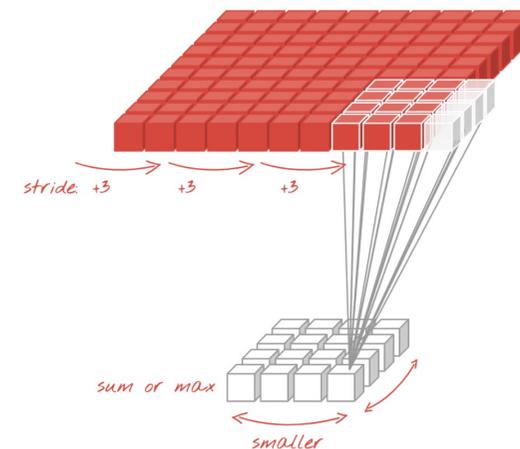
A convolutional layer is defined by the filter (or kernel) size, the number of filters applied and the stride



Output Volume Size



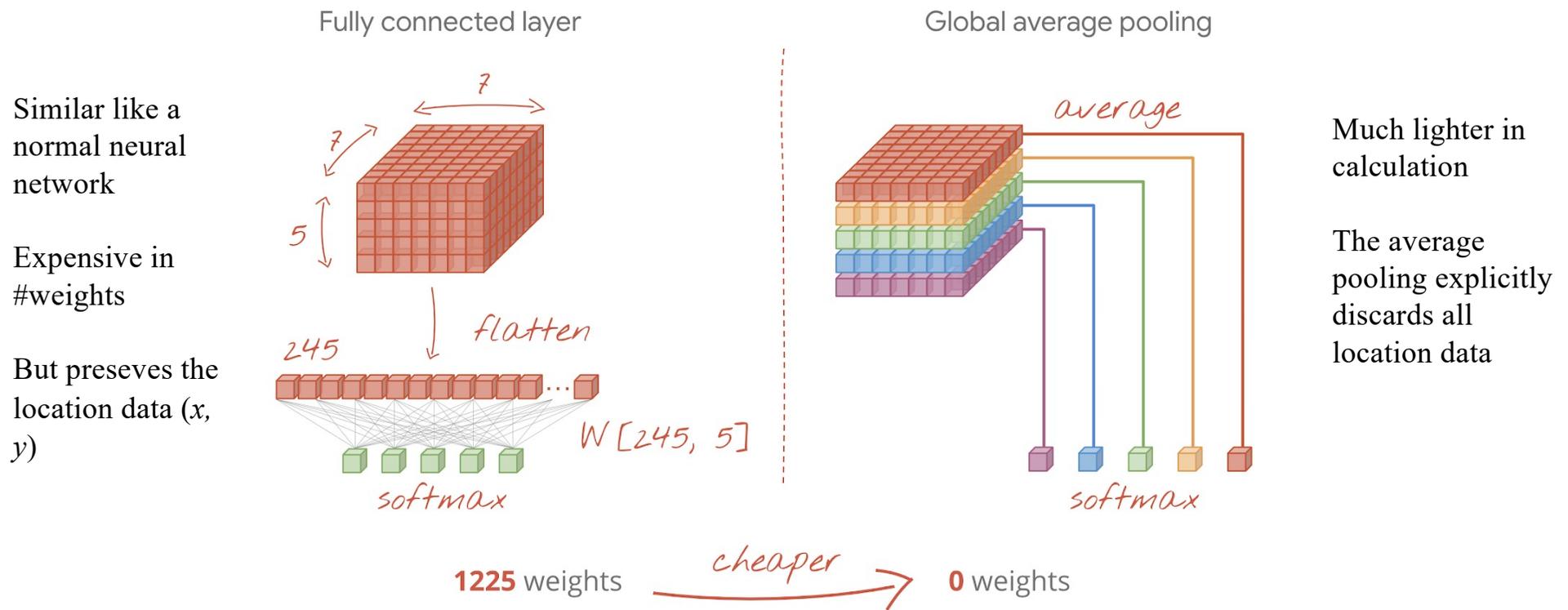
- Depth (number of channels):
 - *adjusted by using more or fewer filters*
- Width & Height:
 - *adjusted by using a stride > 1*
 - *(or with a max-pooling operation)*



Defined by the filter (or kernel) size, the number of filters applied and the stride

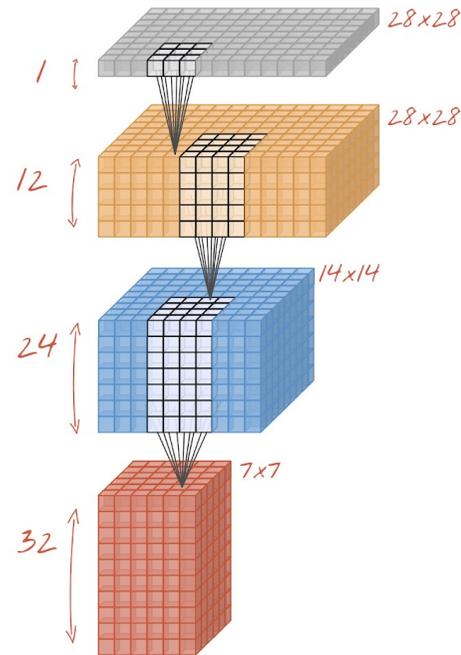
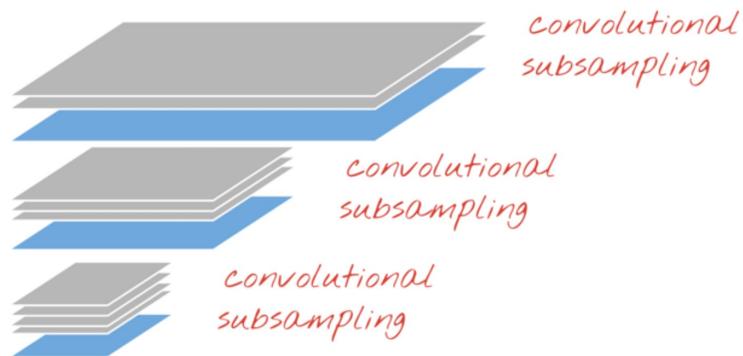
The Last Layer

From a Cubic Volume in 3D to predicted labels



Convolutional Neural Network

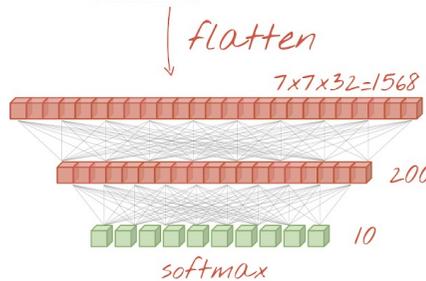
Stacking Up a ConvNet *Layer-by-layer*



Convolutional 3x3 filters=12
 $W_1[3, 3, 1, 12]$

Convolutional 6x6 filters=24
 $W_2[6, 6, 12, 24]$ stride 2

Convolutional 6x6 filters=32
 $W_3[6, 6, 24, 32]$ stride 2



Dense layer

$W_4[1568, 200]$

Softmax dense layer

$W_5[200, 10]$

Exercise II: Handwritten digits classification

- MNIST forward Neural Network:
 - https://ml4a.github.io/demos/forward_pass_mnist/
- MNIST confusion matrix
 - https://ml4a.github.io/demos/confusion_mnist/
- MNIST Convolutional Neural Network: filters
 - https://ml4a.github.io/demos/convolution_all/
- MNIST TensorFlow.js playground:
<https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html>
- Explainable AI Demos:
<https://lrpserver.hhi.fraunhofer.de/image-classification>



Thank you~

Wan Fang
Southern University of Science and Technology