الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

AI for Interdisciplinary Research

Part 3: Opening the Box (2)

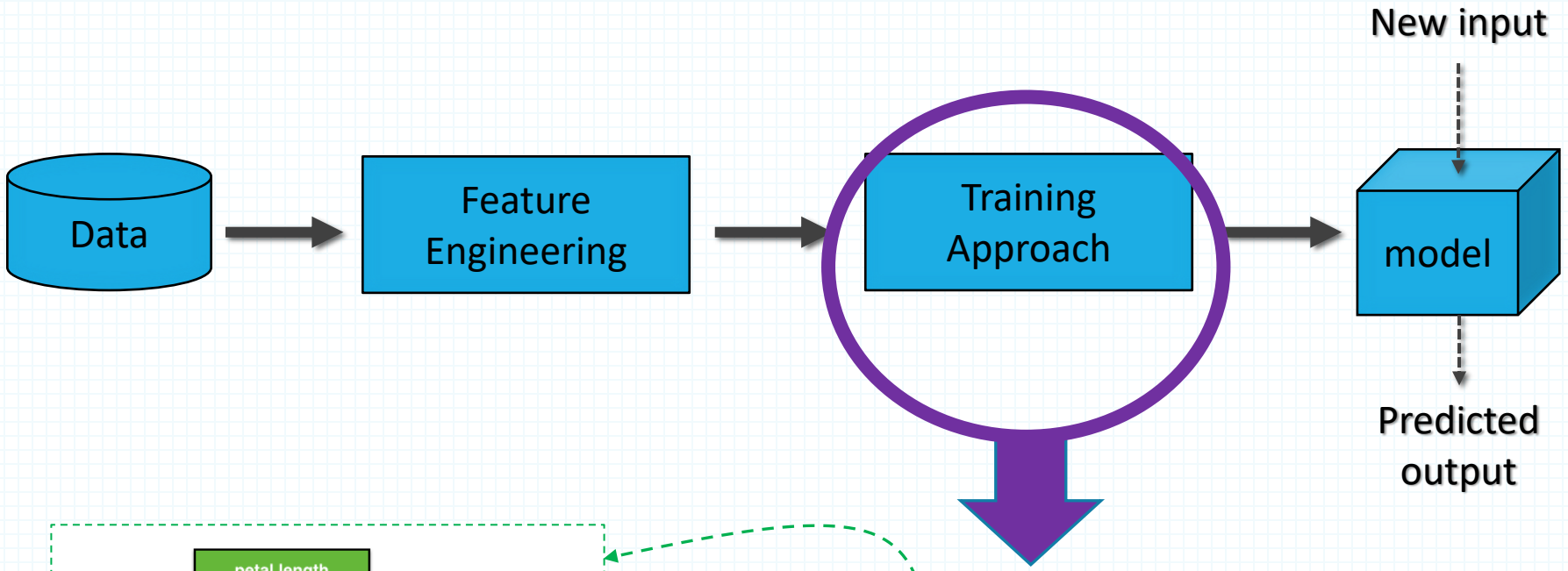
Dr. Riad Sonbol

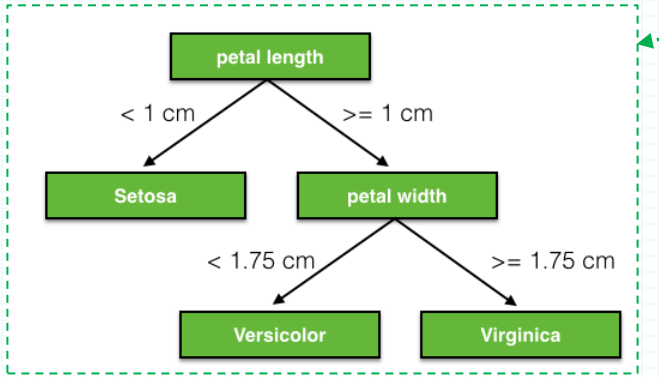
April 2026

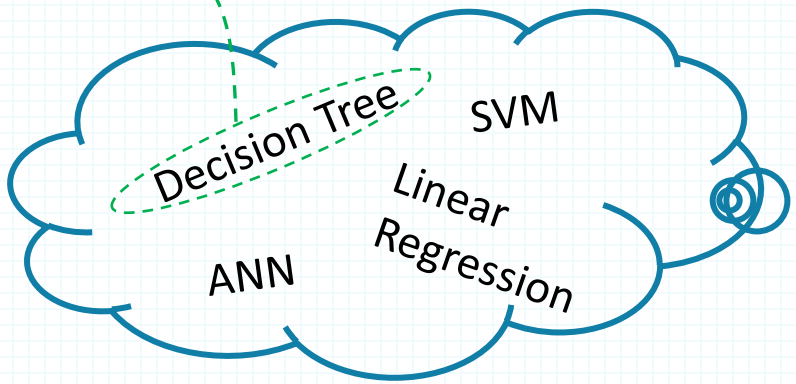
Road Map

- AI as a Black Box – Concepts and Foundations
- Opening the Box (1) – Traditional Machine Learning
- Opening the Box (2) – Deep Learning
- Modern AI and Research Directions

Traditional ML Pipeline



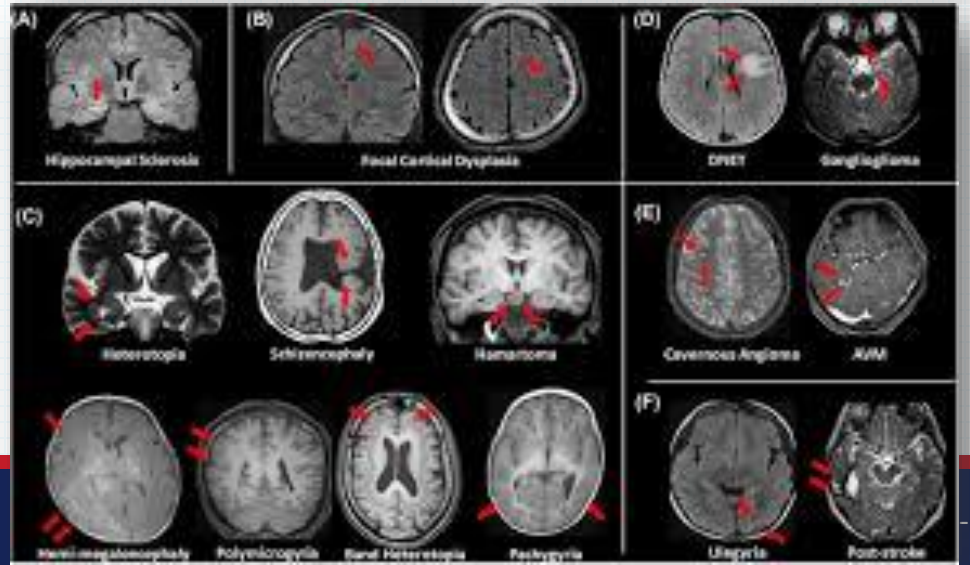




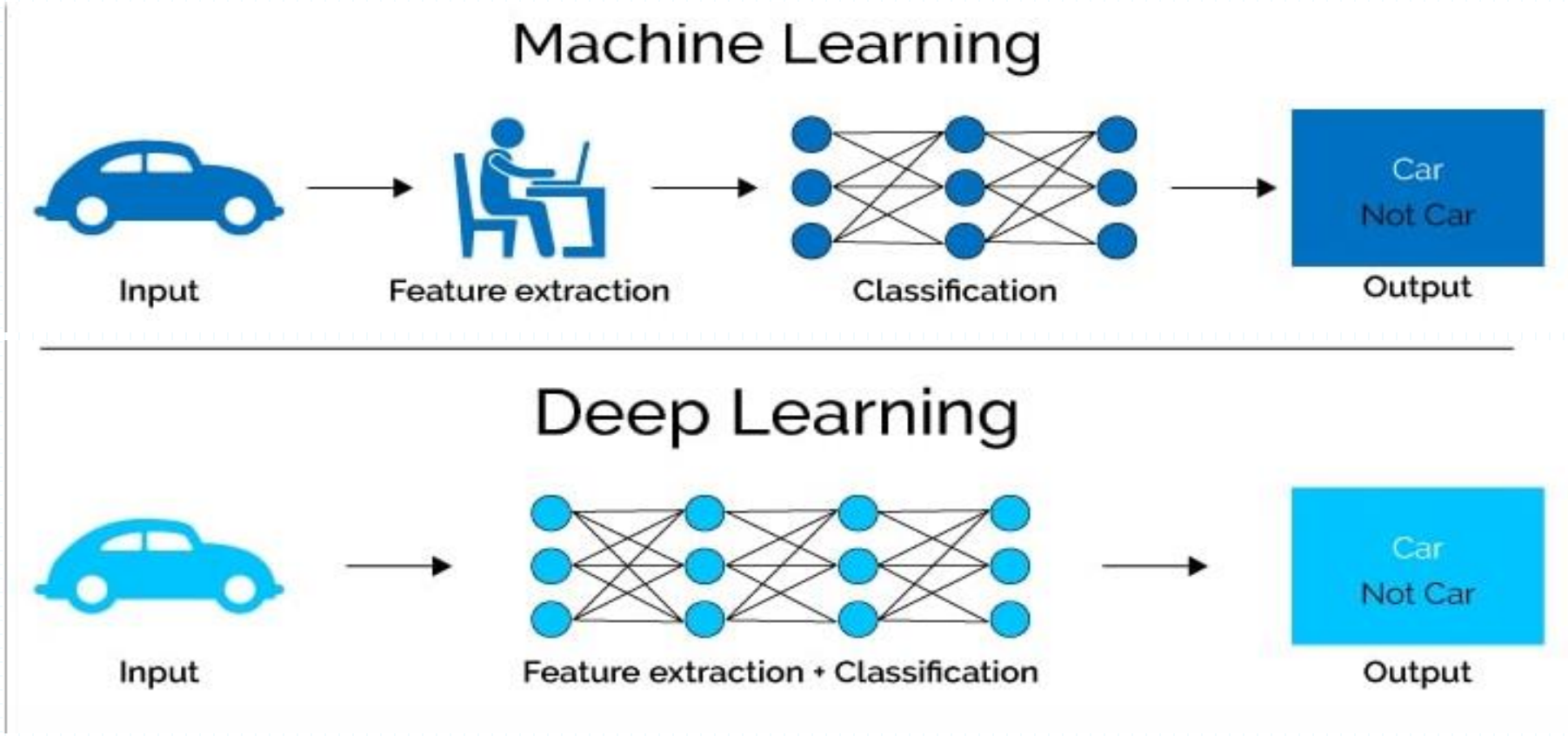
Why Deep Learning!?

Why Deep Learning!?

Riad Sonbol -

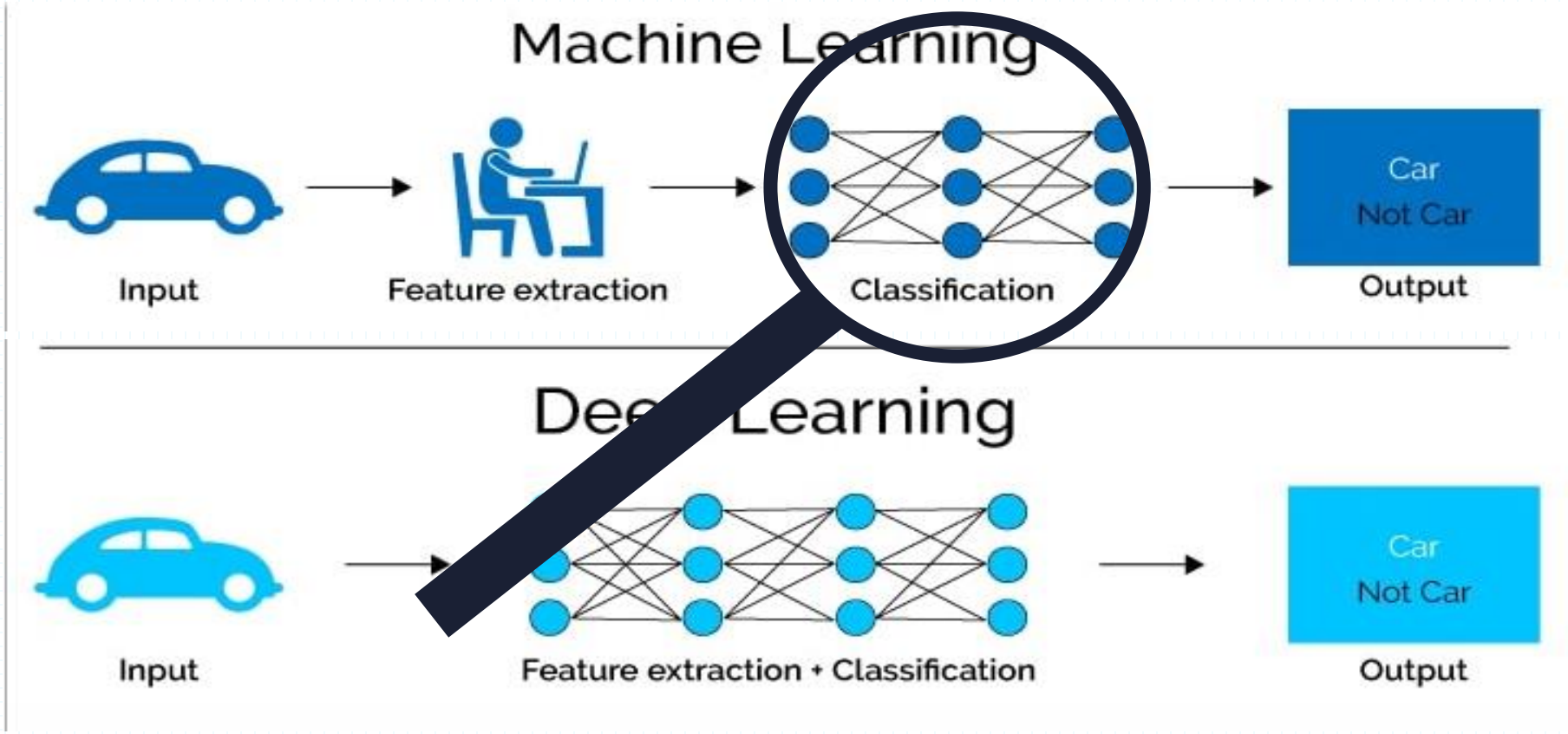


What is Deep Learning?

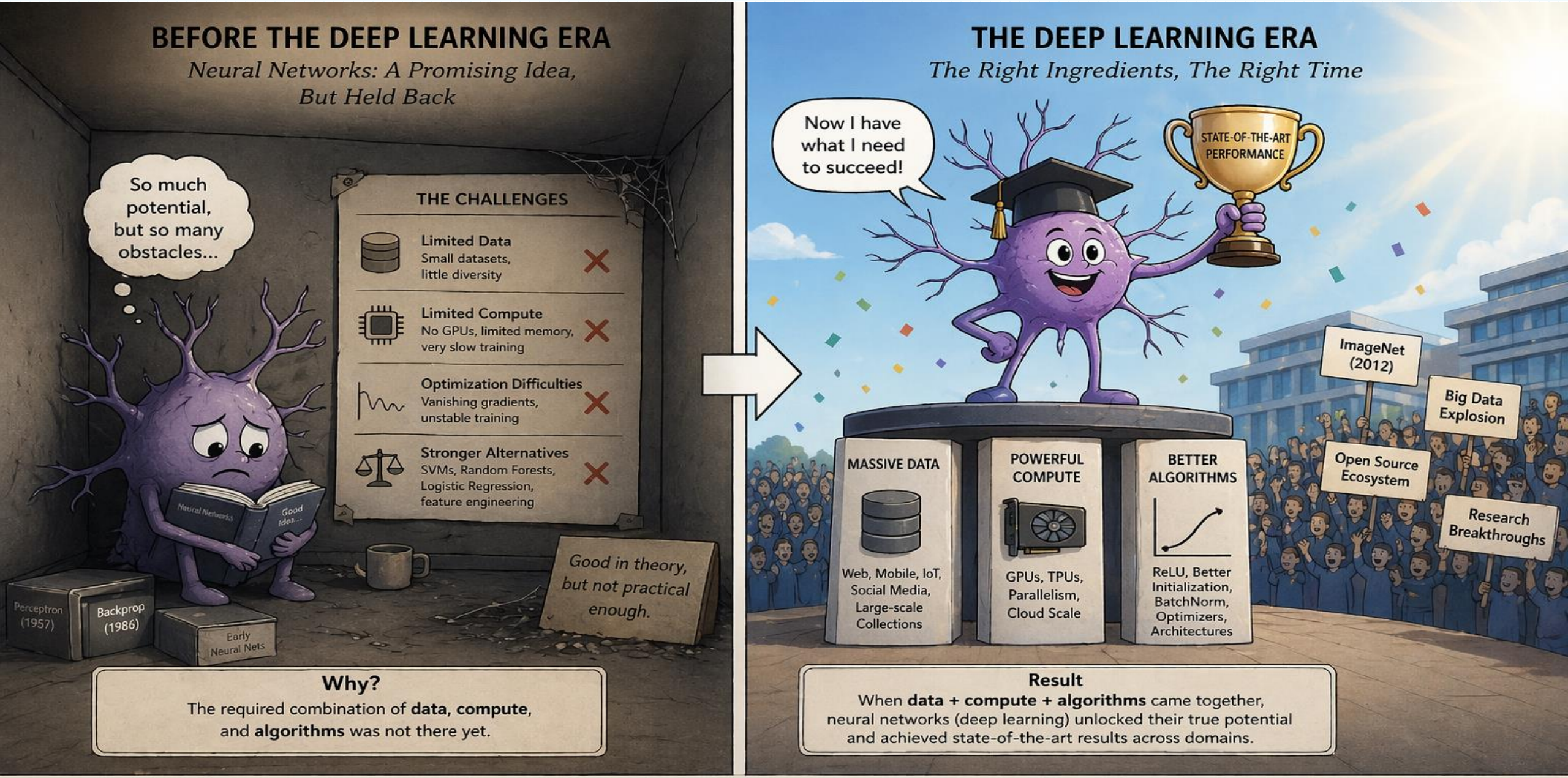


can we learn underlying features directly from data

What is Deep Learning?



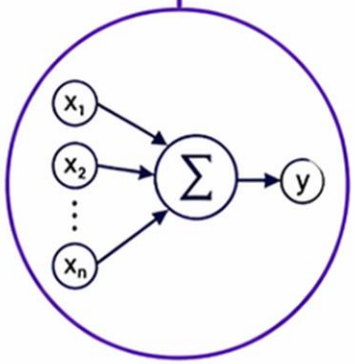
can we learn underlying features directly from data


BEFORE THE DEEP LEARNING ERA
Neural Networks: A Promising Idea, But Held Back
So much potential, but so many obstacles...
THE CHALLENGES
Limited Data
Small datasets, little diversity
Limited Compute
No GPUs, limited memory, very slow training
Optimization Difficulties
Vanishing gradients, unstable training
Stronger Alternatives
SVMs, Random Forests, Logistic Regression, feature engineering
Neural Networks
Good Ideas...
Perceptron (1957)
Backprop (1986)
Early Neural Nets
Good in theory, but not practical enough.
Why?
The required combination of data, compute, and algorithms was not there yet.
THE DEEP LEARNING ERA
The Right Ingredients, The Right Time
Now I have what I need to succeed!
STATE-OF-THE-ART PERFORMANCE
MASSIVE DATA
Web, Mobile, IoT, Social Media, Large-scale Collections
POWERFUL COMPUTE
GPUs, TPUs, Parallelism, Cloud Scale
BETTER ALGORITHMS
ReLU, Better Initialization, BatchNorm, Optimizers, Architectures
ImageNet (2012)
Big Data Explosion
Open Source Ecosystem
Research Breakthroughs
Result
When data + compute + algorithms came together, neural networks (deep learning) unlocked their true potential and achieved state-of-the-art results across domains.
1950s–1980s
Foundations (Perceptron, Backprop)
1990s
Traditional ML Dominates
2000s
More Data, More Compute, But Still Limited
2010s
Deep Learning Breakthrough
2020s
Broad Impact Across Industries and Science

The Rise of Deep Learning: A Timeline

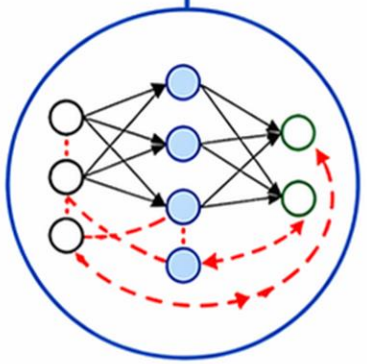
From foundational ideas to practical, scalable deep learning

1958 — Perceptron



The first neural network model proposed. A simple idea that laid the foundation for artificial neural networks.

1986 — Backpropagation



Introduction of the backpropagation algorithm, the key breakthrough that enabled training of multi-layer neural networks effectively.

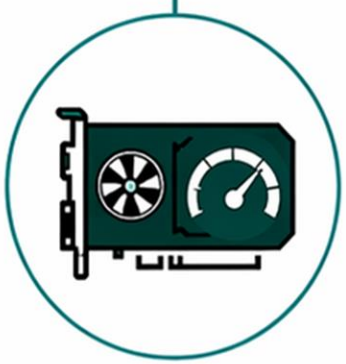
2000–2010 — Internet + Social Media

Explosion of digital data from the internet and social media platforms provided the massive amounts of data needed to train deep learning models.

2006 — CUDA (NVIDIA)

NVIDIA launched CUDA, a parallel computing platform that enabled GPUs to be used for general-purpose computing, dramatically boosting performance.

2010–2012 — GPU Training Becomes Practical



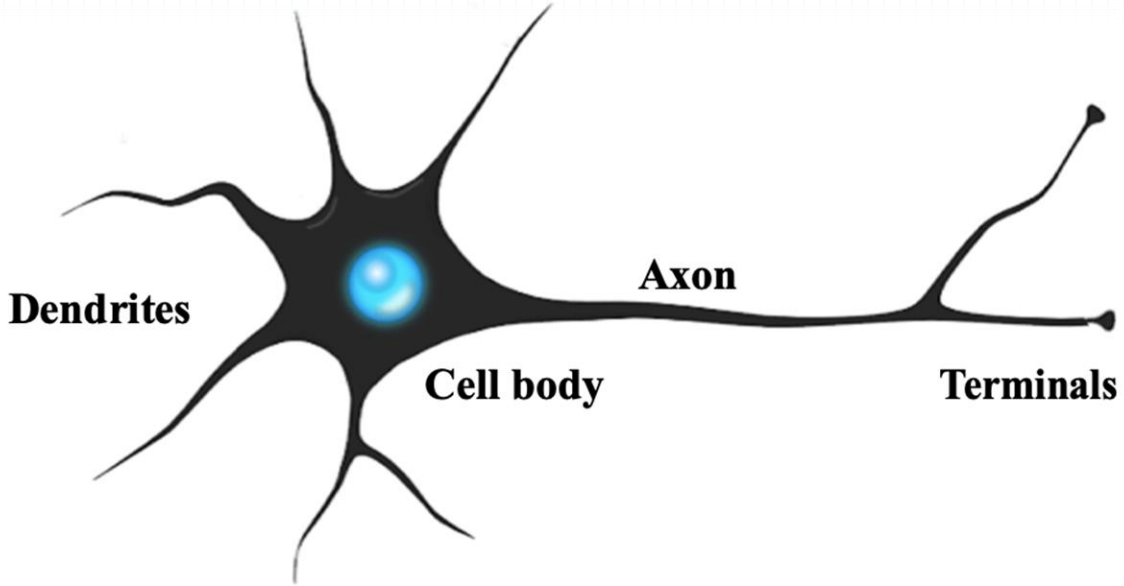
With big data + powerful GPUs + improved algorithms, training deep neural networks became practical, efficient, and achievable at scale for the first time.

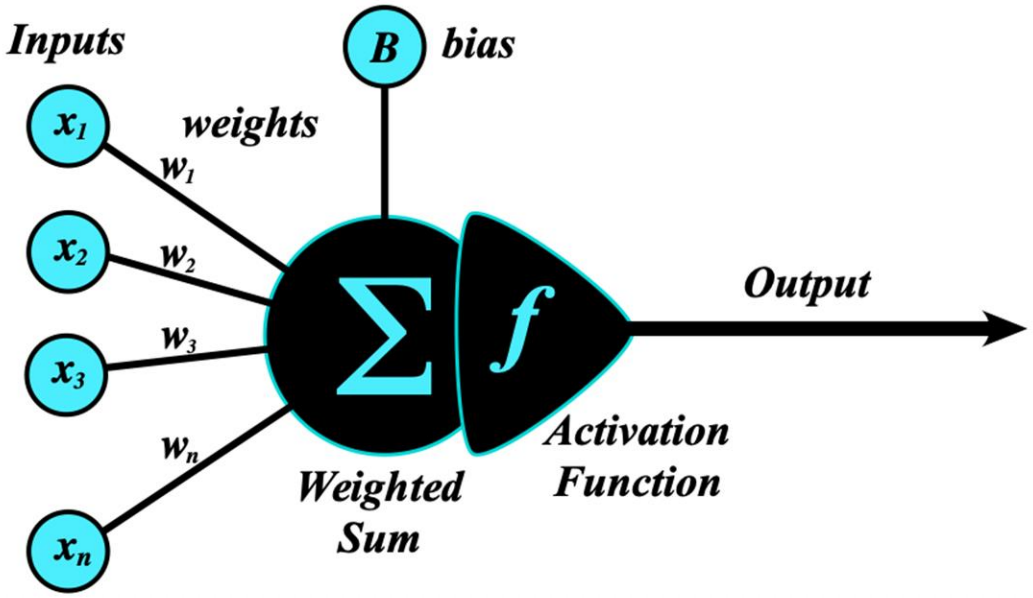
Data (Massive datasets) + **Compute** (High-performance GPUs) + **Algorithms** (Advances in techniques) = **Deep Learning Revolution** — Transforming AI and the world

The Concept

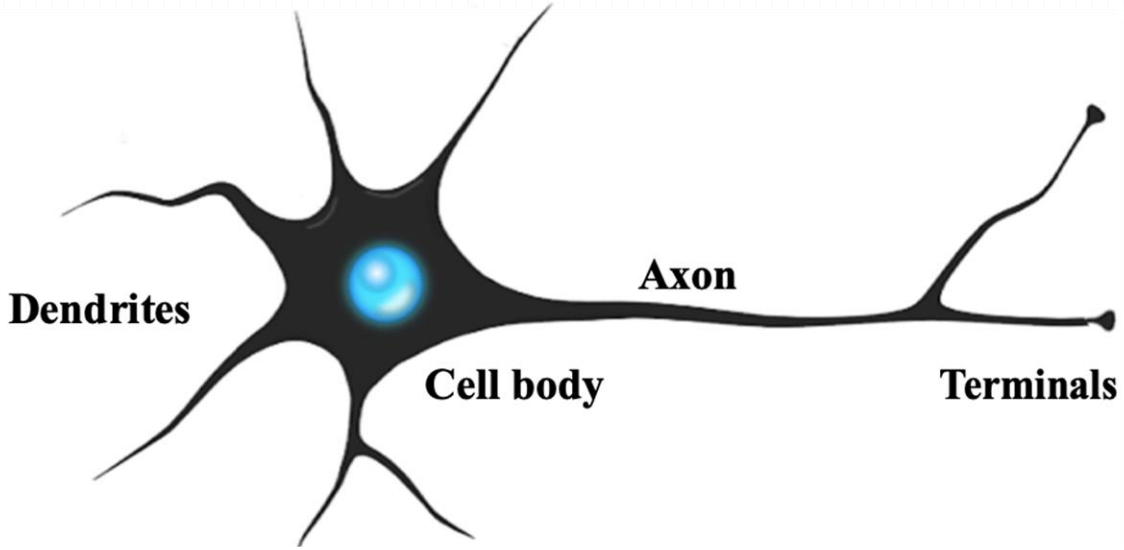
The Concept

Biological Neuron
vs
artificial neuron





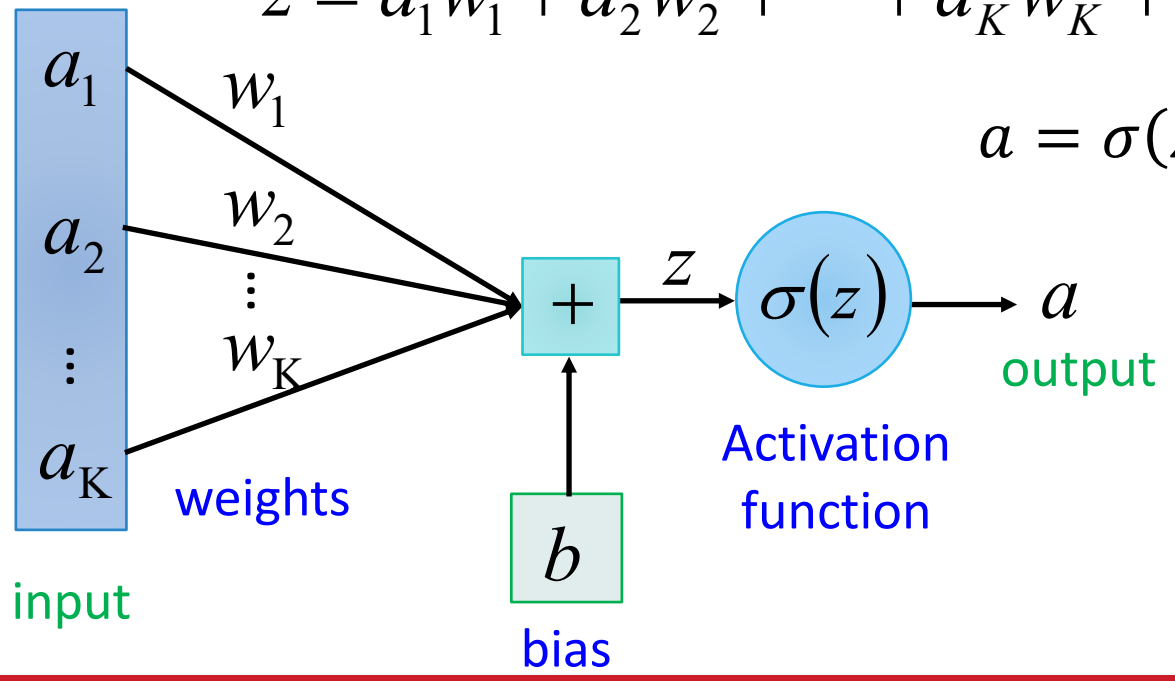
The Concept



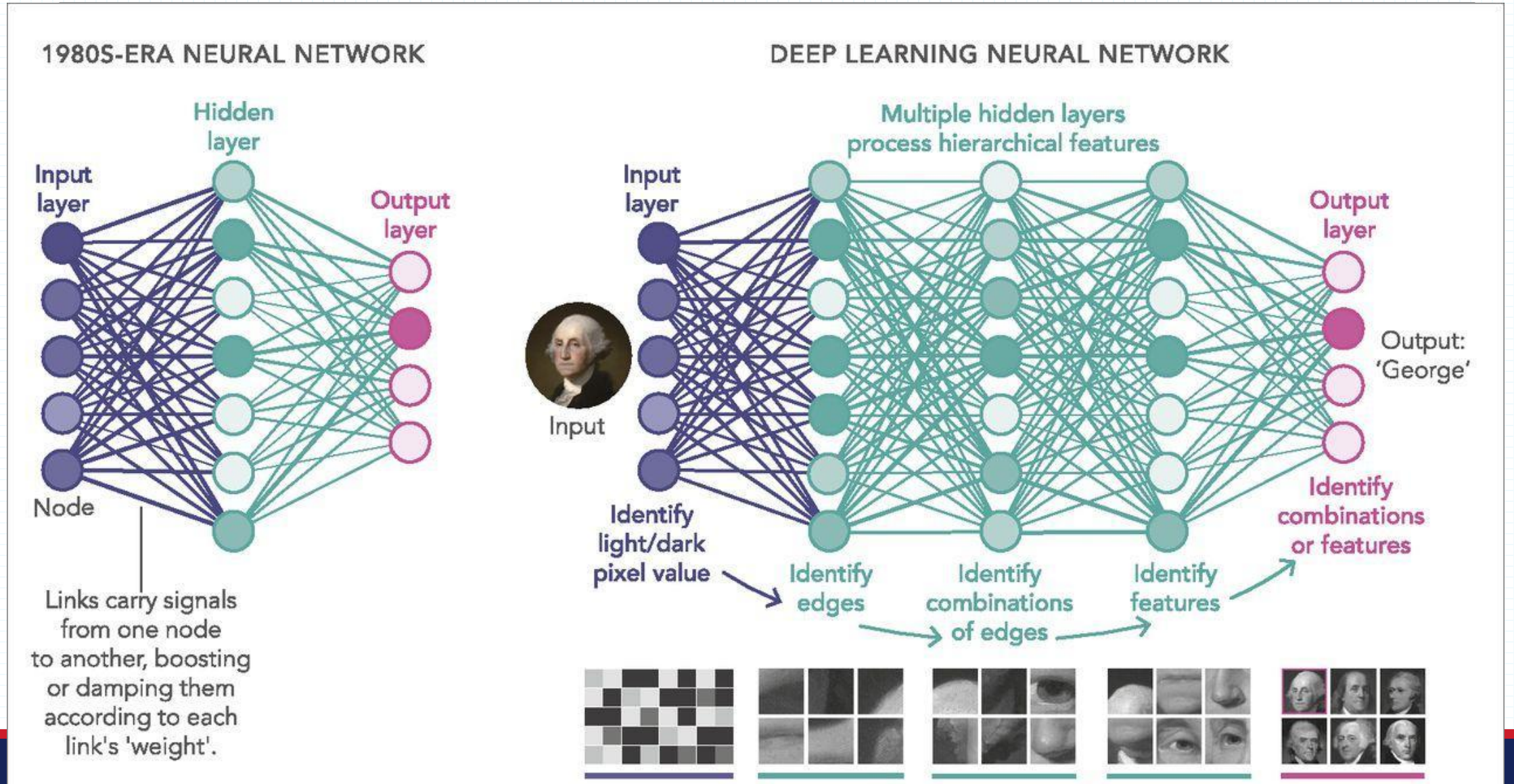
Biological Neuron
vs
artificial neuron

$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$

$$a = \sigma(z)$$

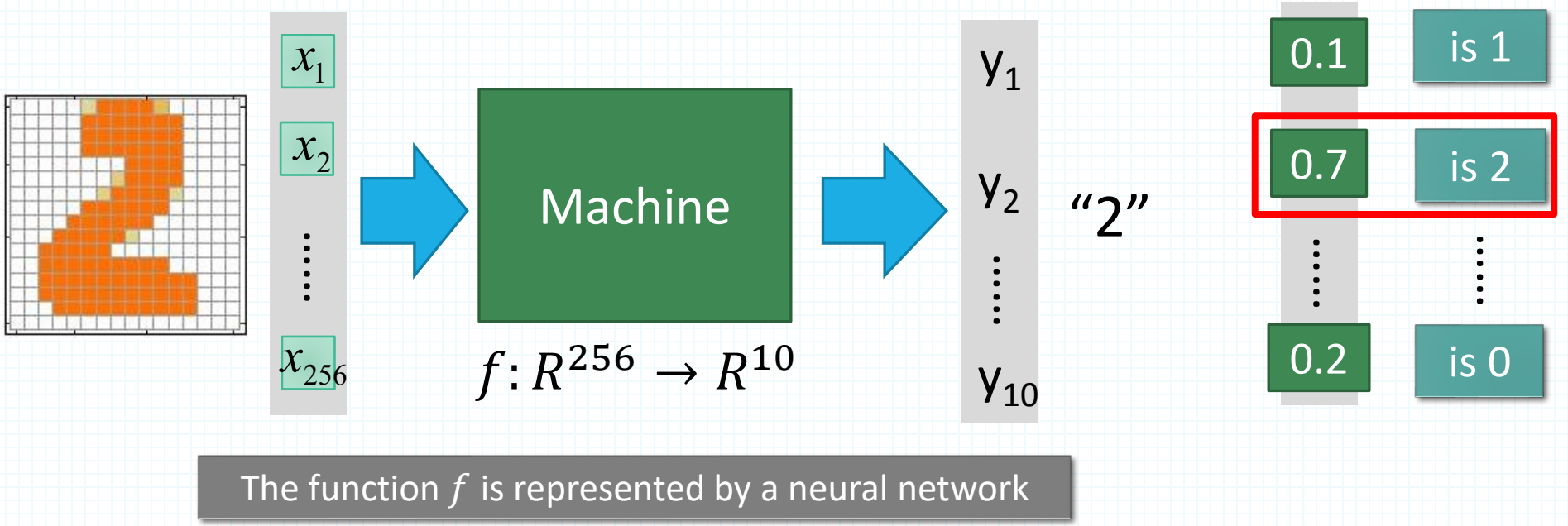


Deeper Neural Networks!



Example:

- **Input:** Raw data—such as images—is represented as a set of numerical features (e.g., pixel intensities in an image).
- **Output:** a predicted class label representing the category or type of the input data.

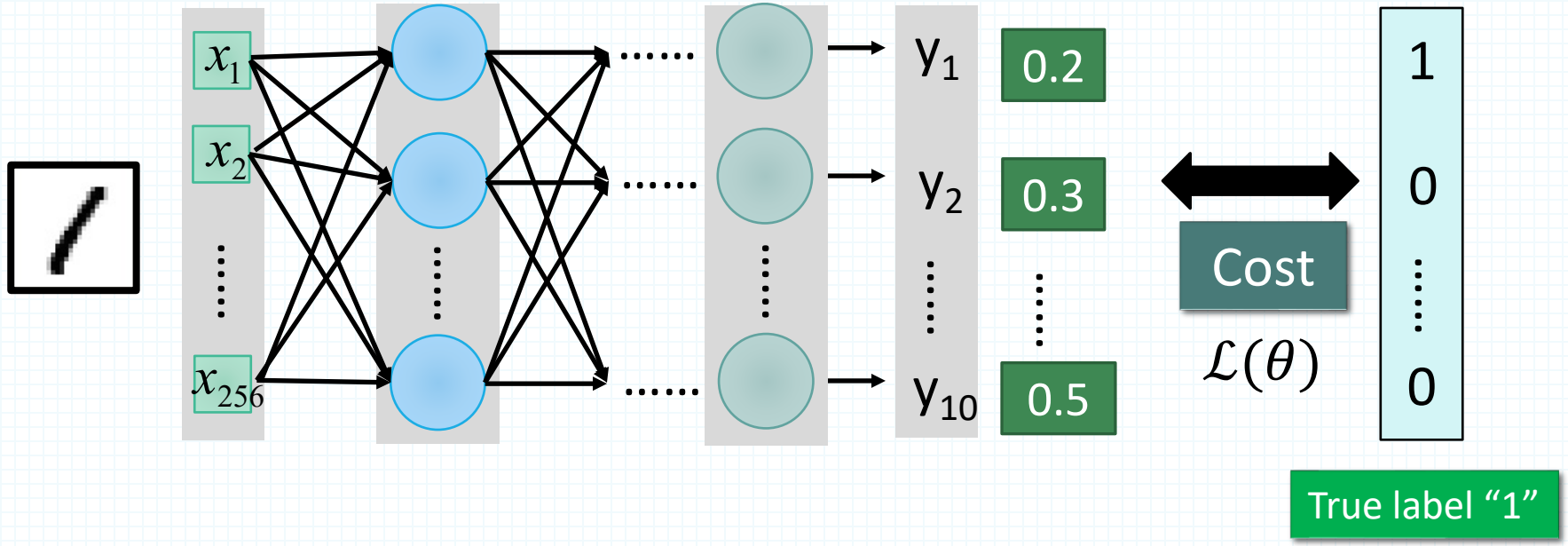


The function f is represented by a neural network

Training Algorithm

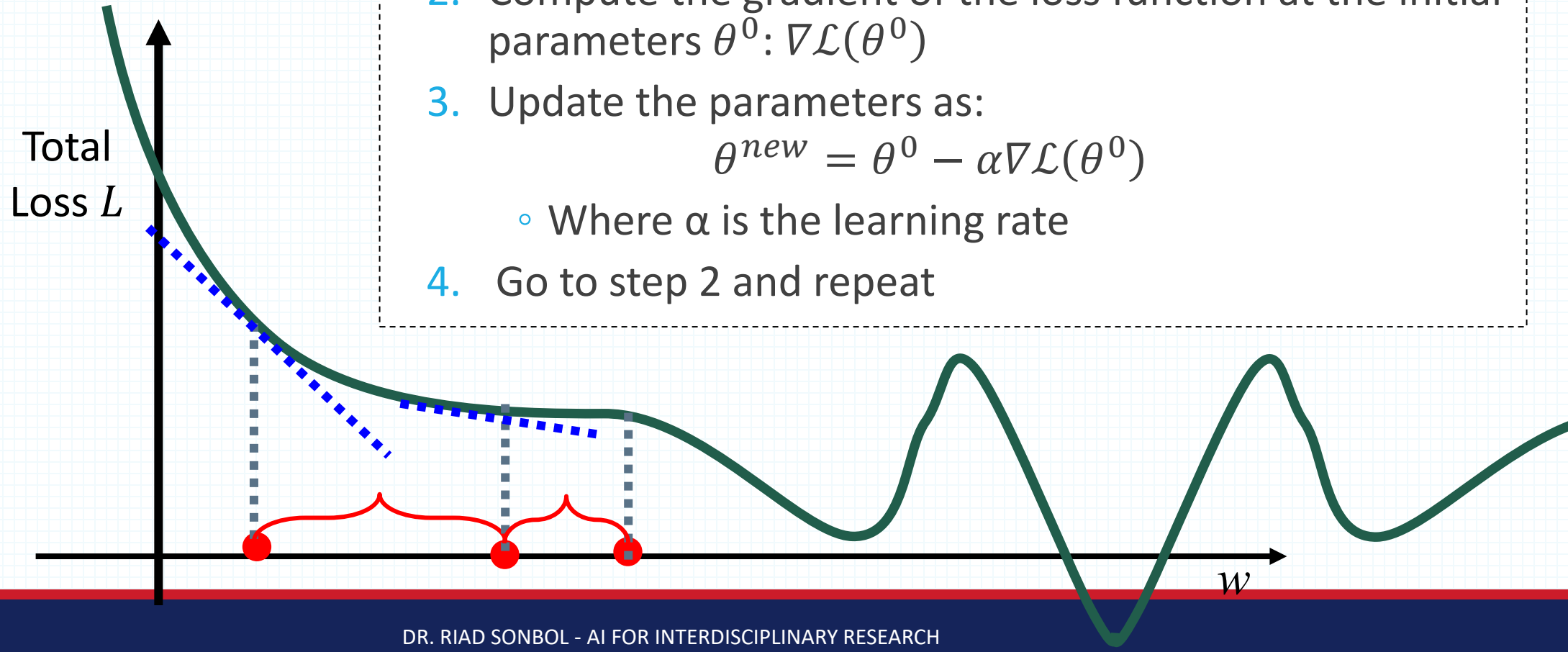
Training NNs

- The network ***parameters*** θ include $\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$
- Define a ***loss function***/objective function/cost function $\mathcal{L}(\theta)$ that calculates the **difference (error) between the model prediction and the true label**

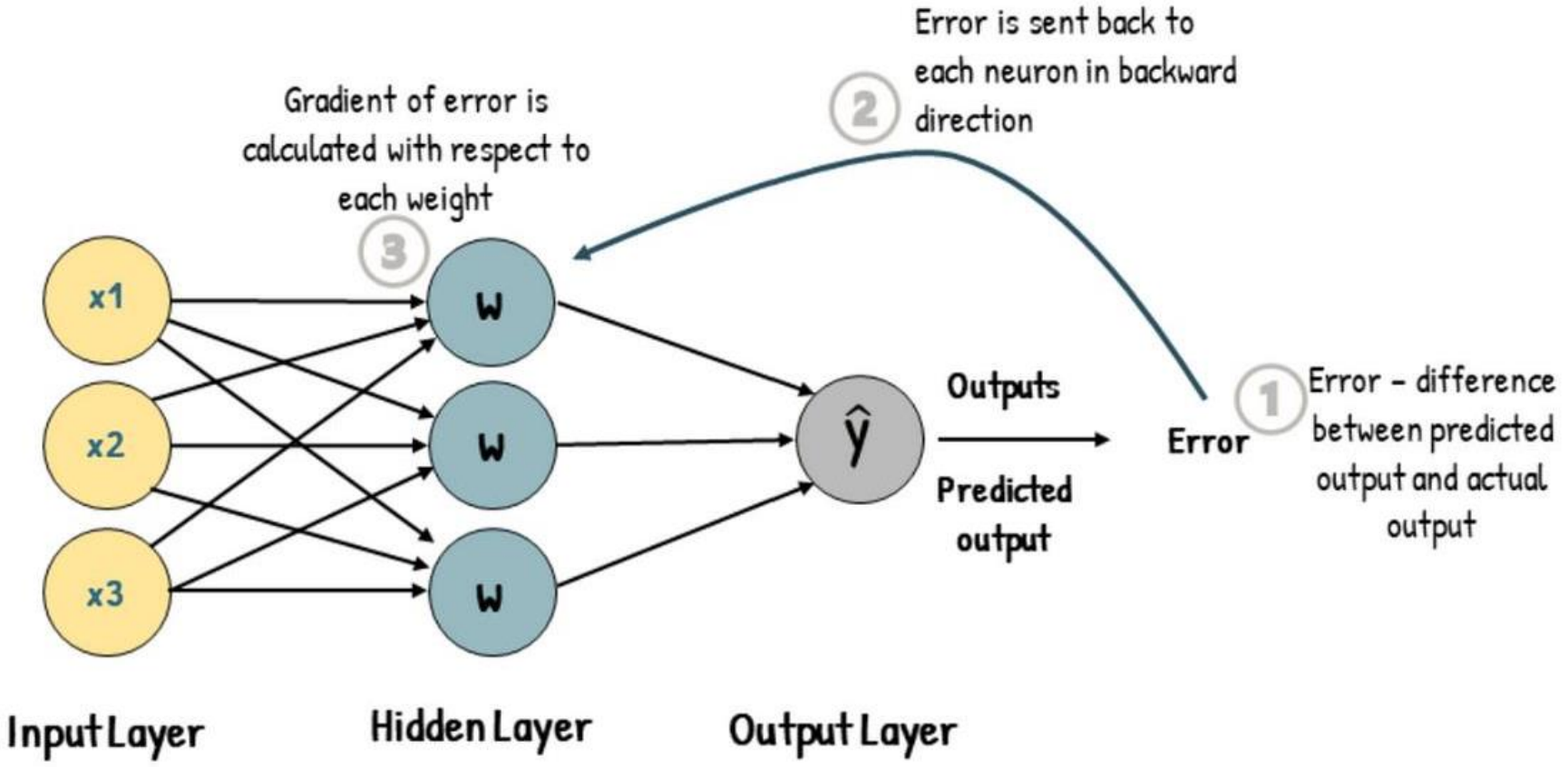


Gradient Descent Algorithm

- Steps in the ***gradient descent algorithm***:
  1. Randomly initialize the model parameters, θ^0
  2. Compute the gradient of the loss function at the initial parameters θ^0: $\nabla\mathcal{L}(\theta^0)$
  3. Update the parameters as:
     $$\theta^{new} = \theta^0 - \alpha\nabla\mathcal{L}(\theta^0)$$
     - Where α is the learning rate
  4. Go to step 2 and repeat



Backpropagation



Batch vs Epoch

It is **wasteful** to compute the loss over the entire training dataset to perform a single parameter update for large datasets

E.g., ImageNet has 14M images

Compute the loss on a mini-batch of data, update the parameters θ, and repeat until all data are used

At the next epoch, shuffle the training data, and repeat the above process

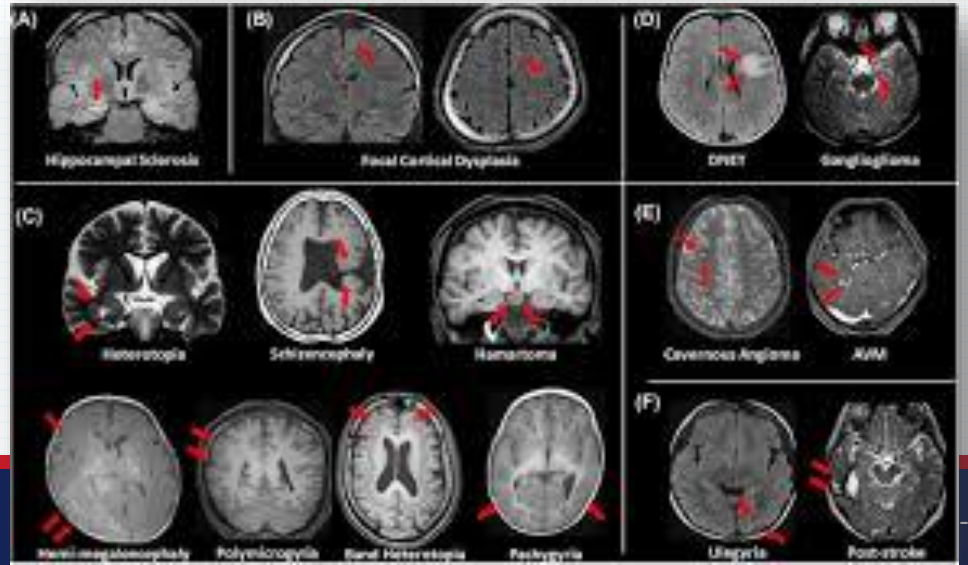
Batch vs Epoch

An **epoch** is one complete pass of the entire training dataset through a neural network,

A **batch** is a subset of that dataset processed before the model's internal parameters are updated

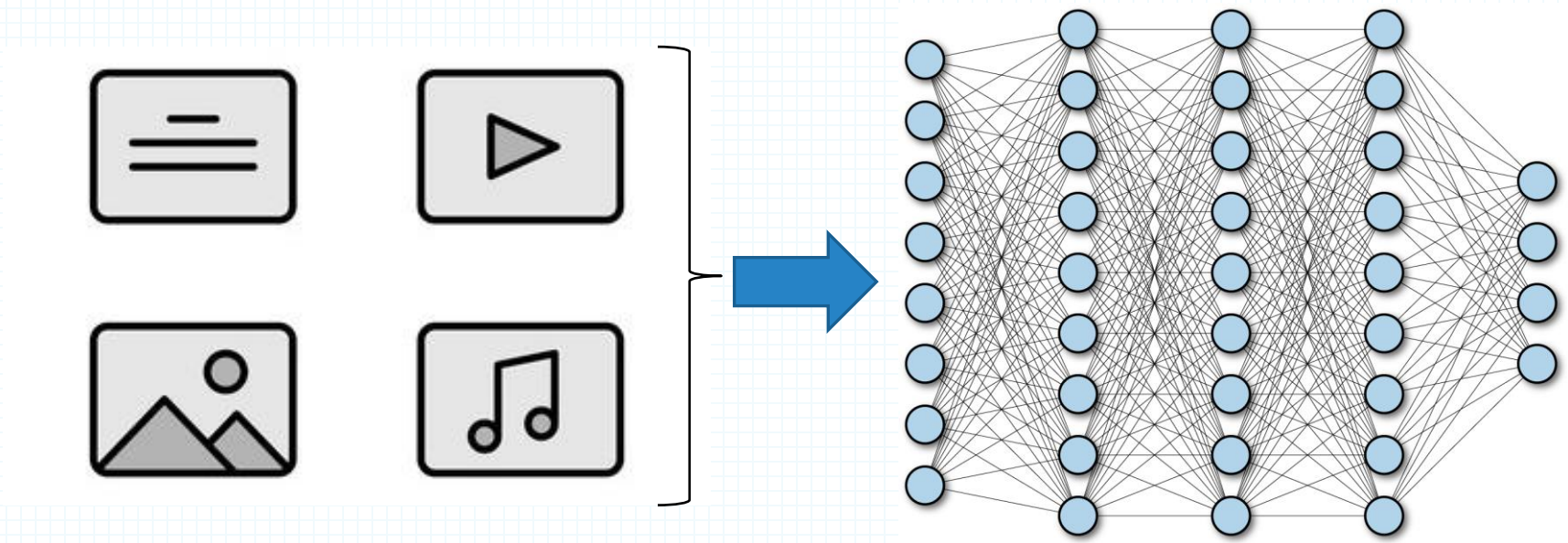
Well Known DL Architecture
CNN, RNN, LSTM

Well Known DL Architecture



Well Known DL Architecture

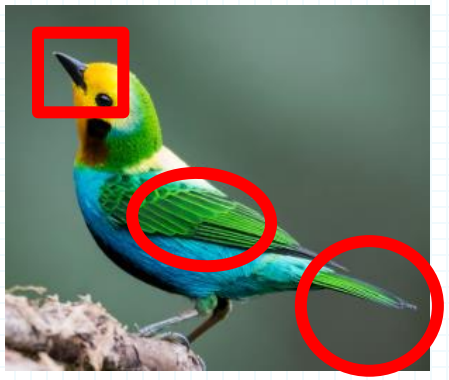
- What kind of neural networks can be used for large or variable length input vectors (e.g., time series)?
- Common networks:
  - CNN, RNN, etc



Convolutional Neural Network (CNN)

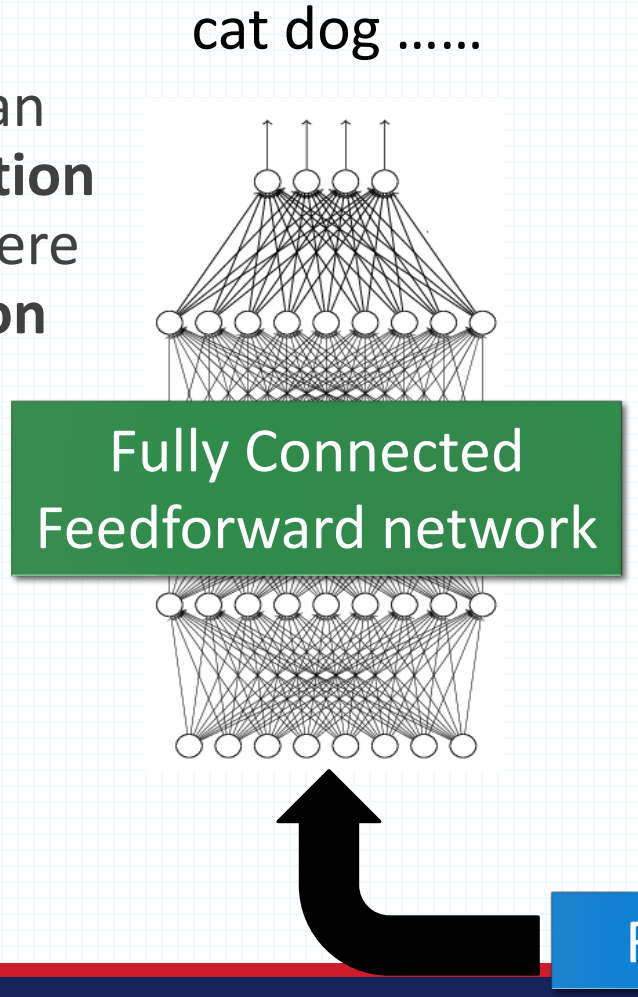
Convolutions for feature extraction

The same patterns appear in different regions.



Convolution Neural Network

- A **convolutional neural network** refers to any network that includes an **alternation of convolution and pooling layers**, where **some of the convolution weights are shared.**
- Architecture:



The whole CNN

Convolution

Max Pooling

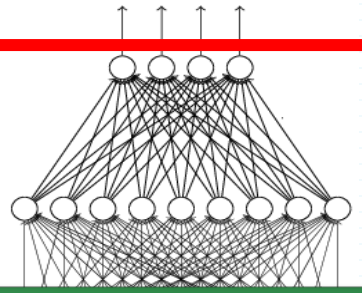
A new image

Convolution

Max Pooling

A new image

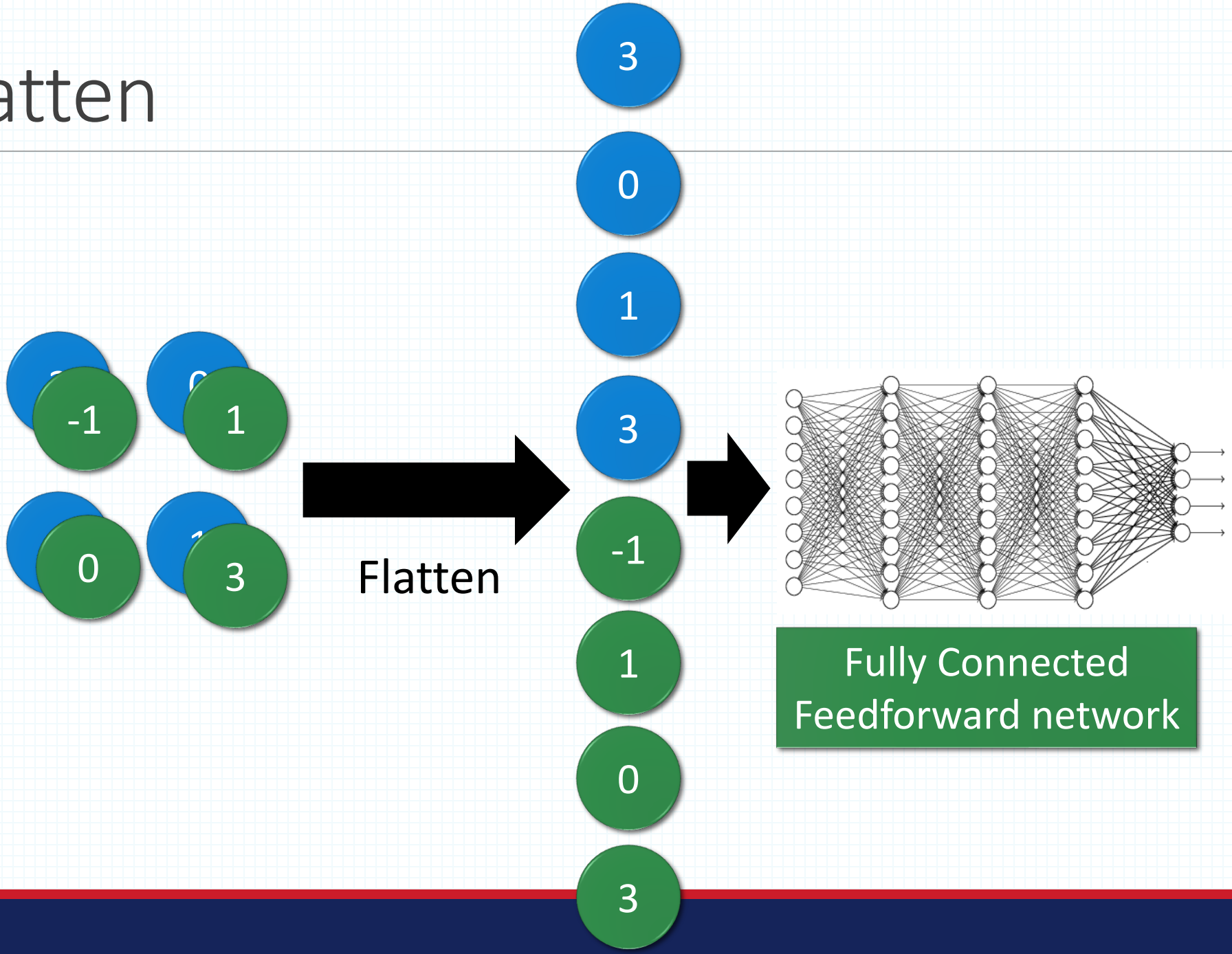
Flatten



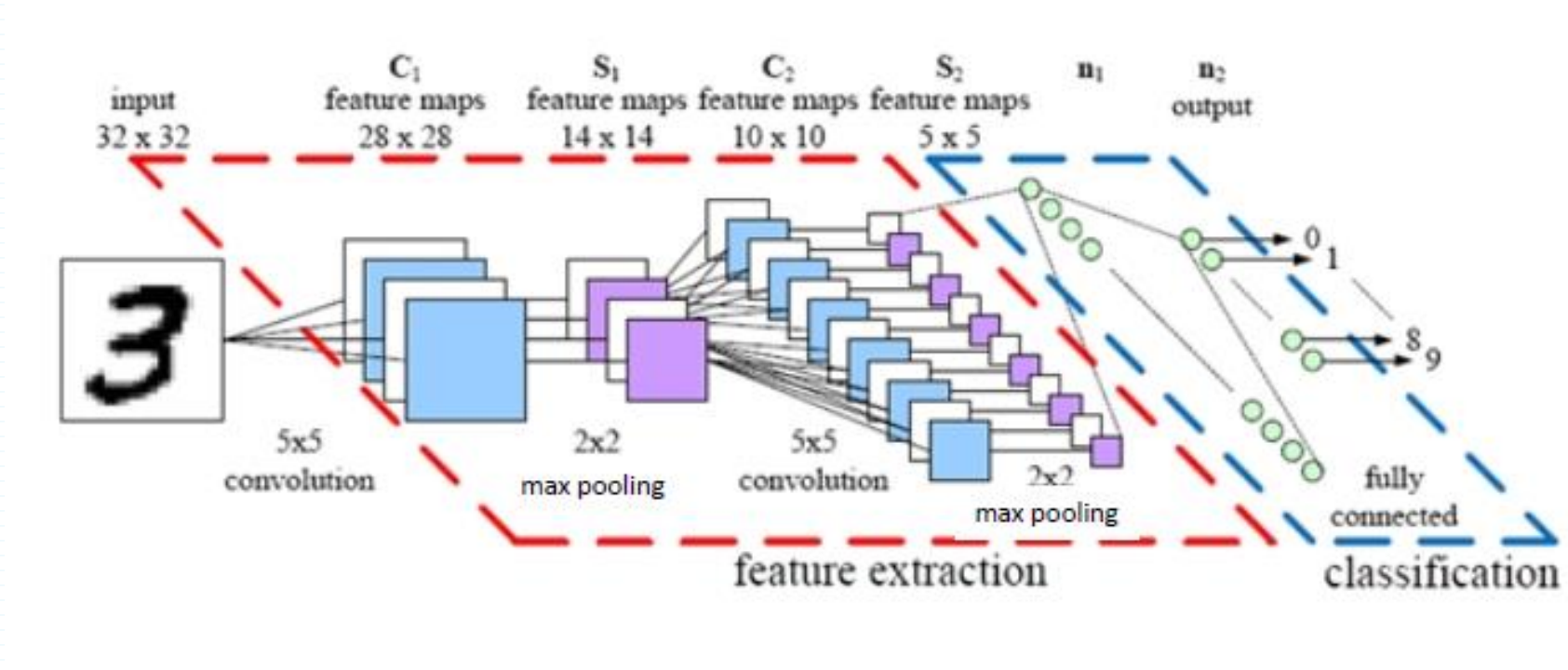
Fully Connected Feedforward network

cat dog ……

Flatten


-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

Example

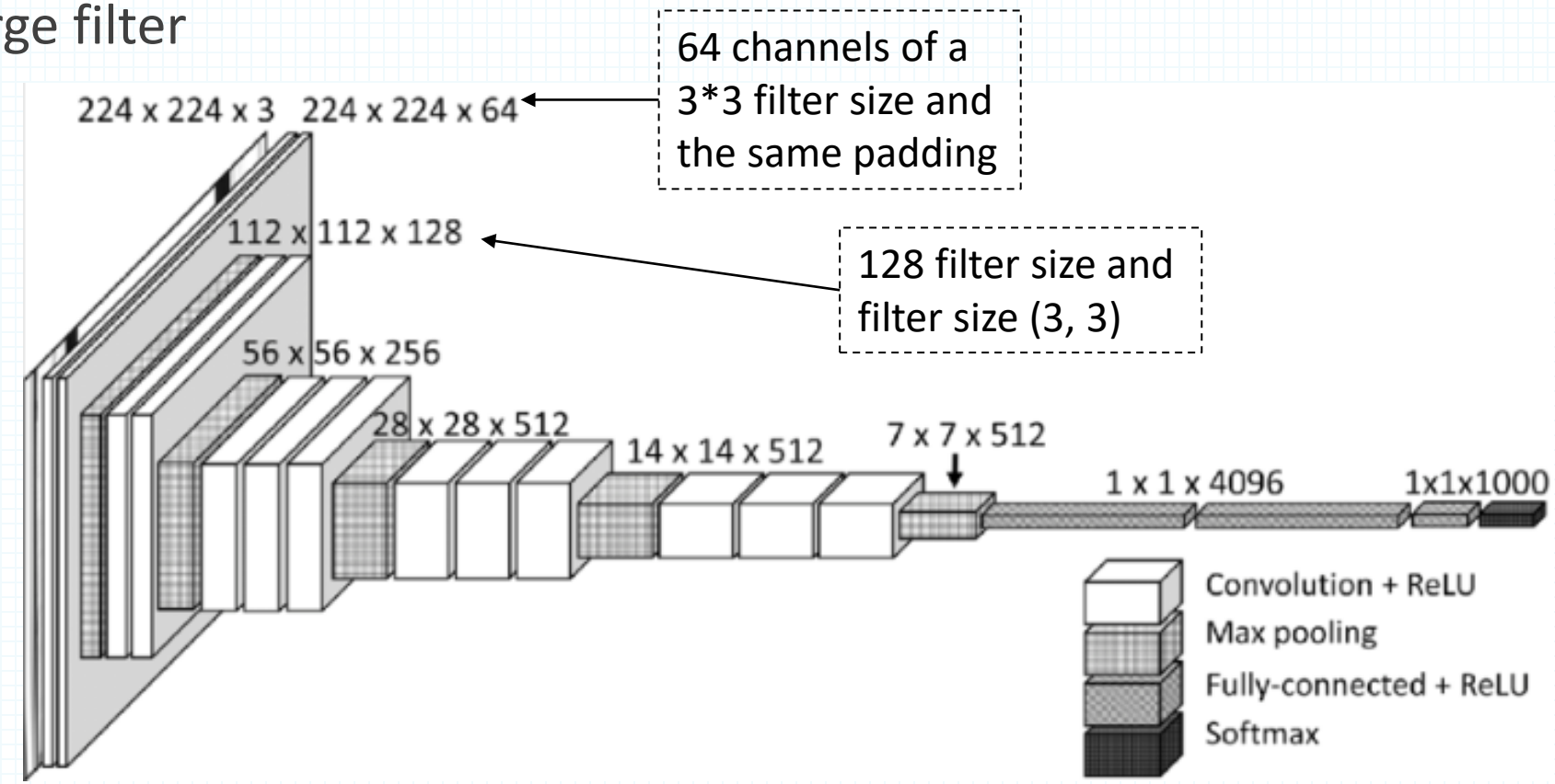

input
32 x 32
C1
feature maps
28 x 28
S1
feature maps
14 x 14
C2
feature maps
10 x 10
S2
feature maps
5 x 5
n1
n2
output
0
1
8
9
5x5
convolution
2x2
max pooling
5x5
convolution
2x2
max pooling
fully
connected
feature extraction
classification

Parameters

- **# of filters**: integer indicating the # of filters applied to each window.
- **kernel size:** tuple (width, height) indicating the size of the window.
- **Stride:** tuple (horizontal, vertical) indicating the horizontal and vertical shift between each window.
- **Padding:** “valid” or ”same”. Valid indicates no input padding. Same indicates that the input is padded with a border of zeros to ensure that the output has the same size as the input.

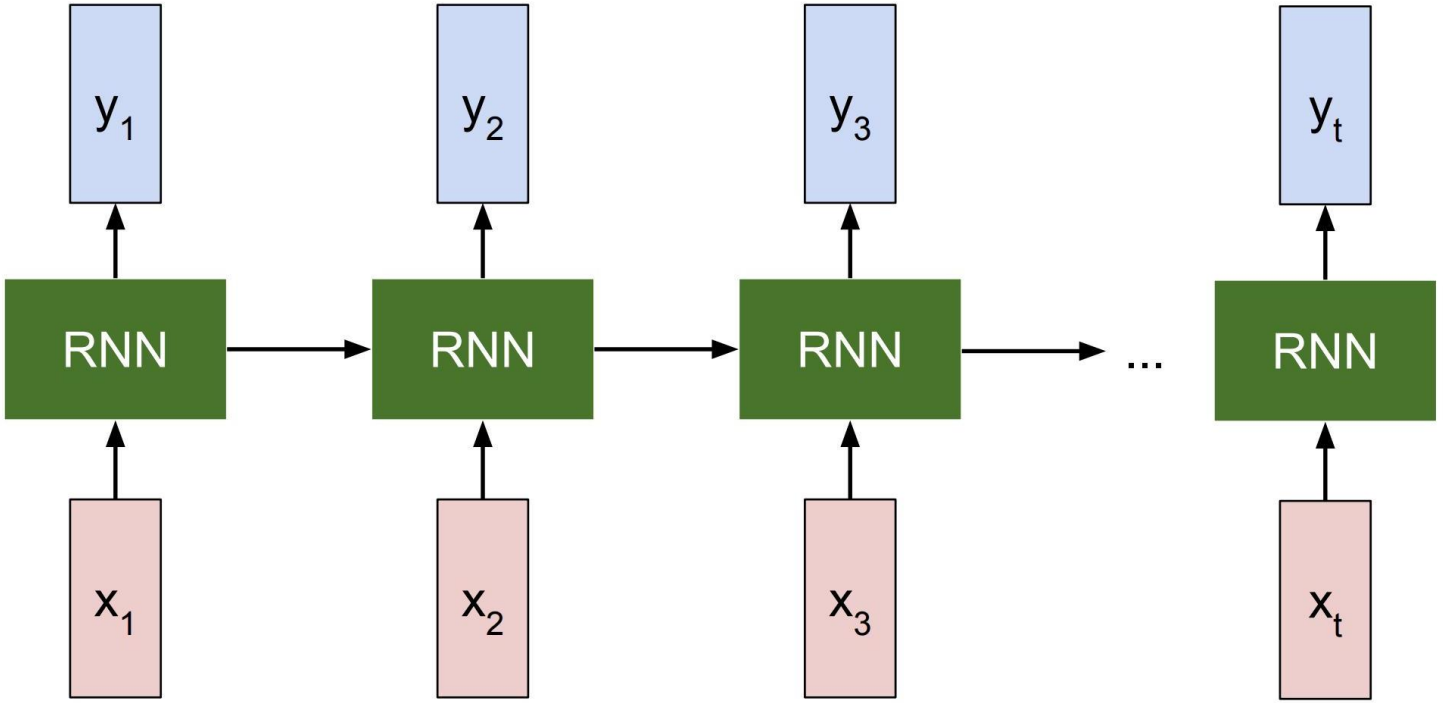
Architecture design: VGG

- What is the preferred filter size?
- VGG (Visual Geometry Group at Oxford, 2014): stack of small filters is often preferred to single large filter
  - Fewer parameters
  - Deeper network



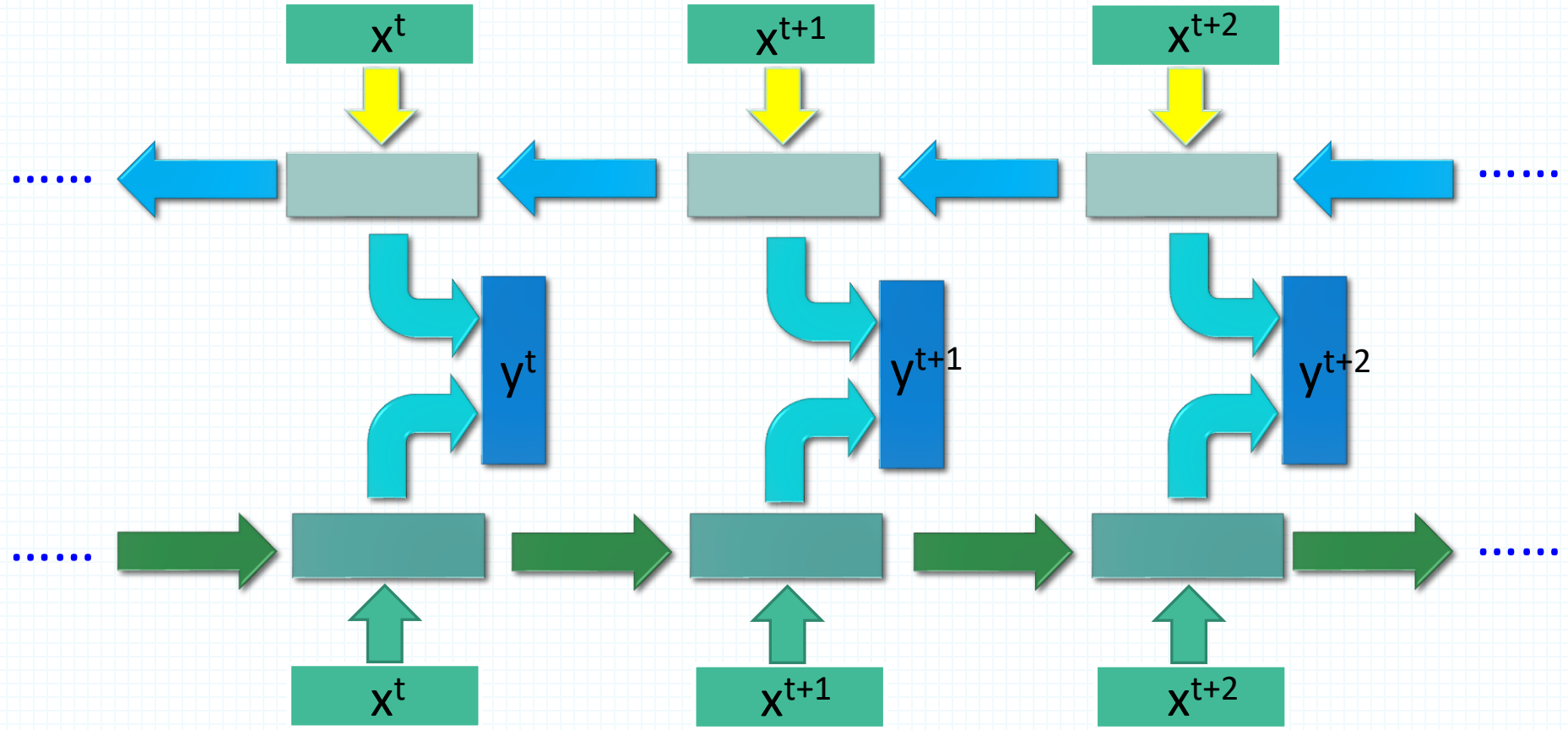
Recurrent Neural Networks

RNN



Bidirectional RNN

We can combine past and future evidence in separate chains



Recurrent Neural Networks