



# WORKSHOP

# INTRODUCTION TO DEEP LEARNING

Tue Vu, PhD  
Research & Data Science Services  
SMU OIT



# Research & Data Science Service

**Who we are?**





# Research & Data Science Service

## What we do?

- Onboard students and researchers with High Performance Computing using M2 (and other HPC)
- Consultation and Scale up research to a new level
- Data analytics, Data Sciences services
- Application of Machine Learning & Deep Learning to advance research
- Conduct workshop on Advance Research Computing and Data Science to SMU communities
- Internet of Things (IOT)



# WORKSHOP

# INTRODUCTION TO DEEP LEARNING

Tue Vu, PhD  
Research & Data Science Services  
SMU OIT

# Schedule

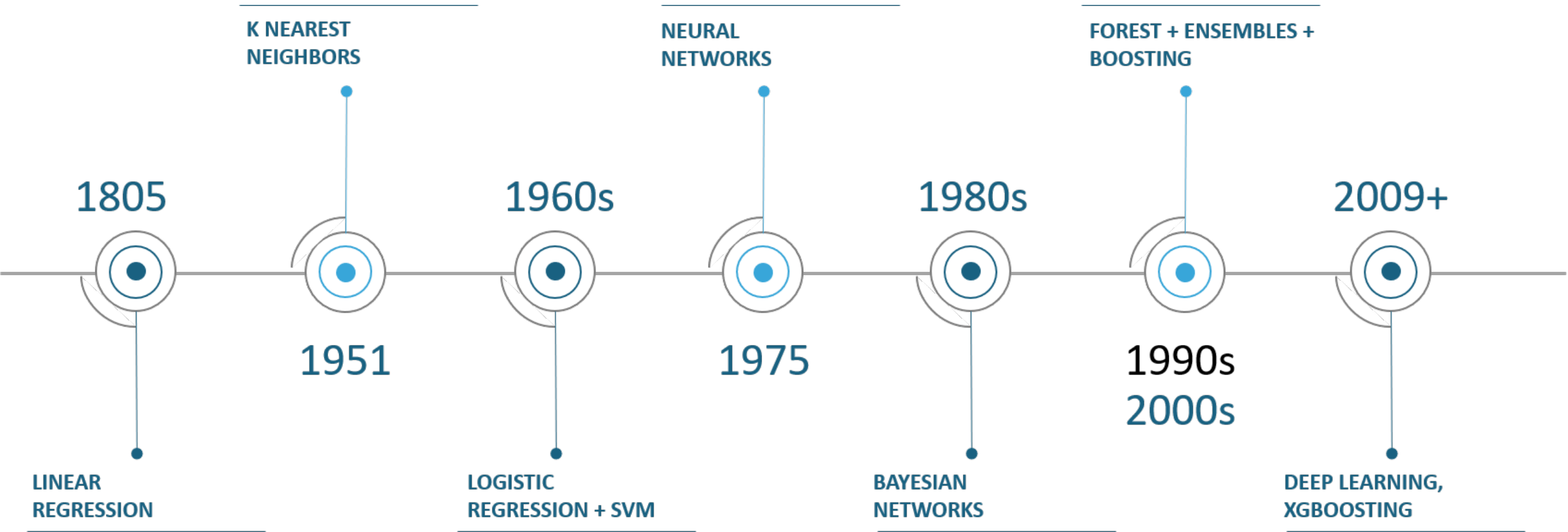
1. Introduction
2. History of Deep Learning
3. Applications of Deep Learning
4. Deep Learning Framework
5. Recap with Neural Network
6. Introduction to Keras API
7. Hands-on section

# Introduction





# Introduction

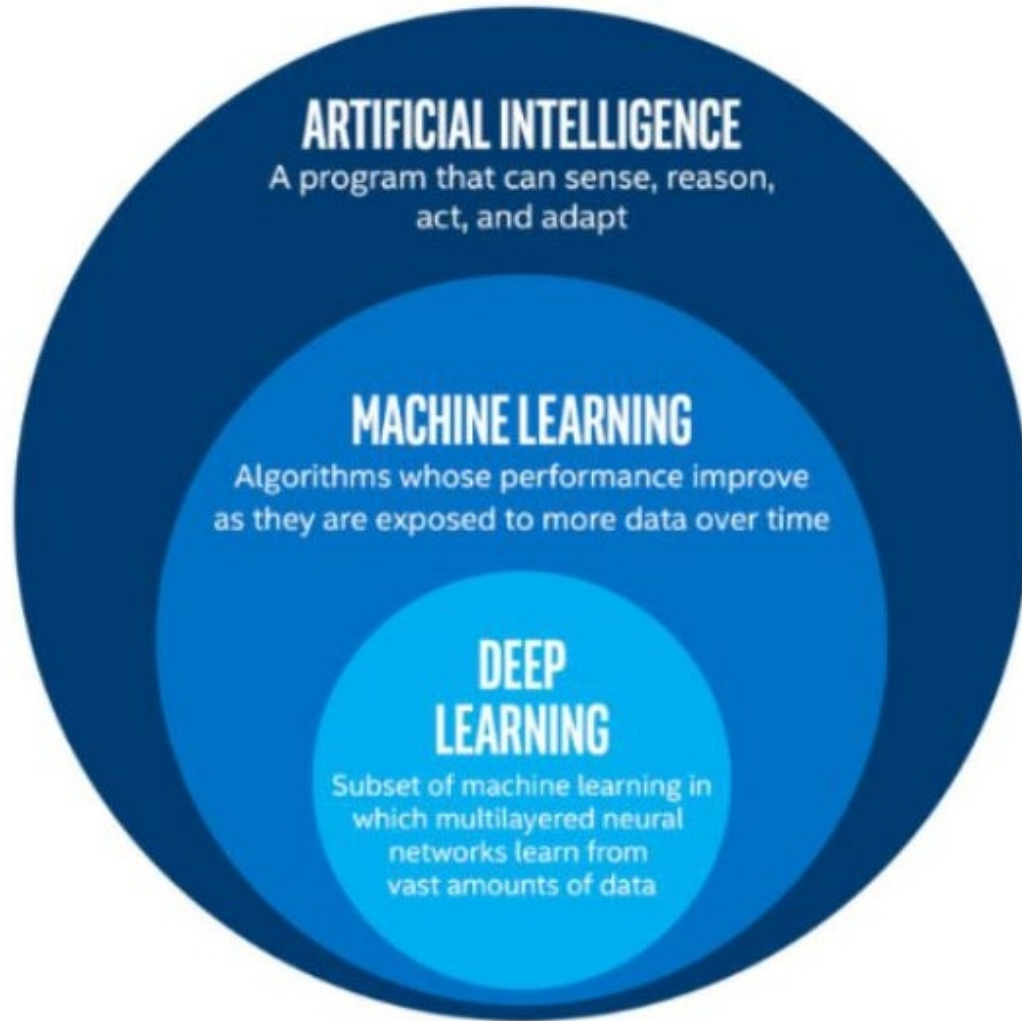


**Machine Learning Timeline:**





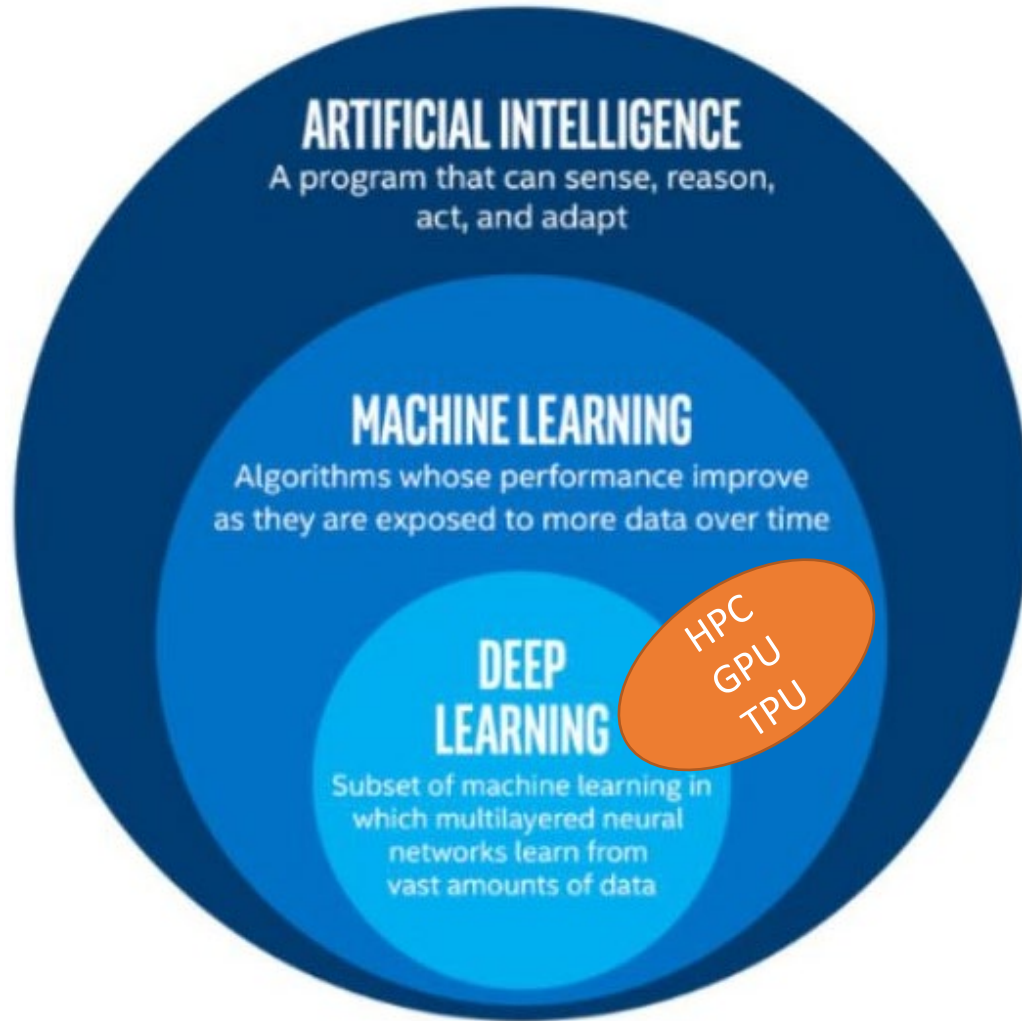
# Introduction





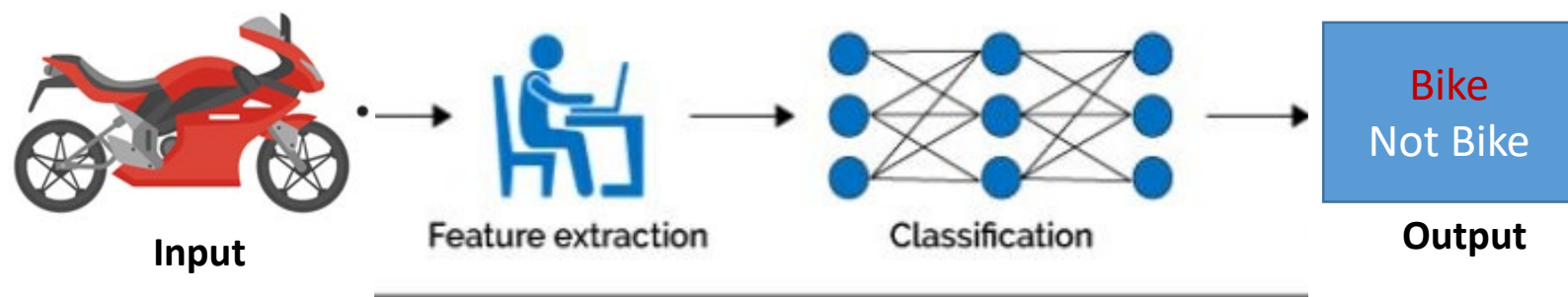


# Introduction

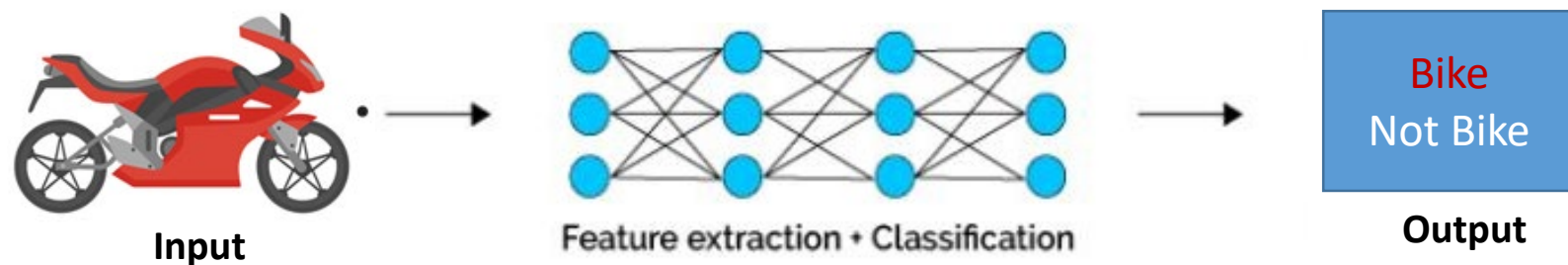


# Introduction

## Machine Learning

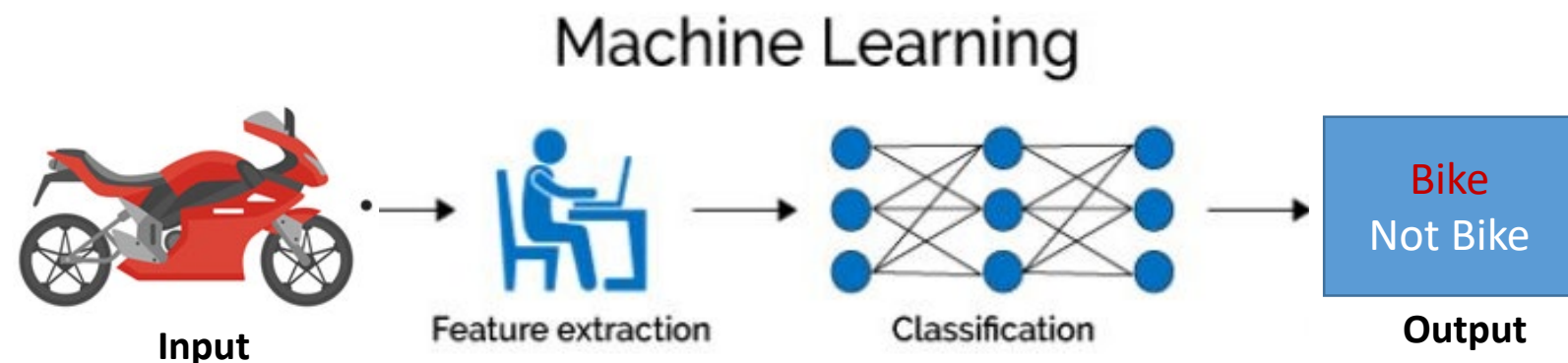


## Deep Learning

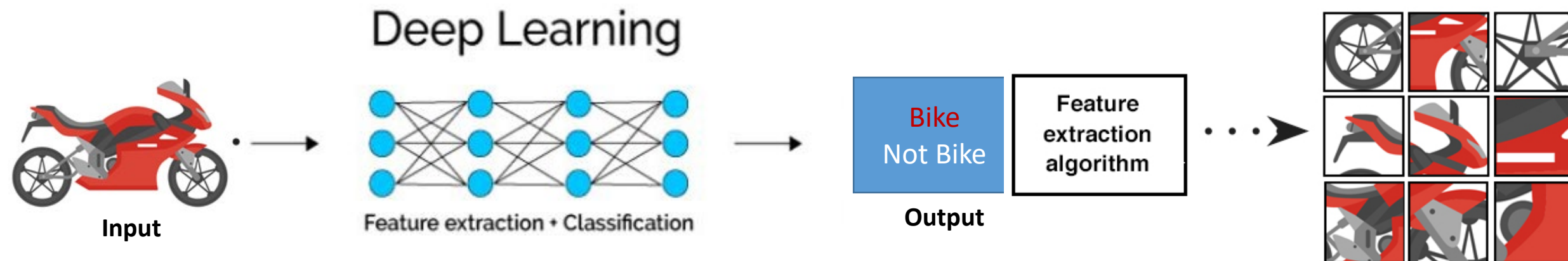


Deep Learning models don't need Feature Extraction

# Introduction

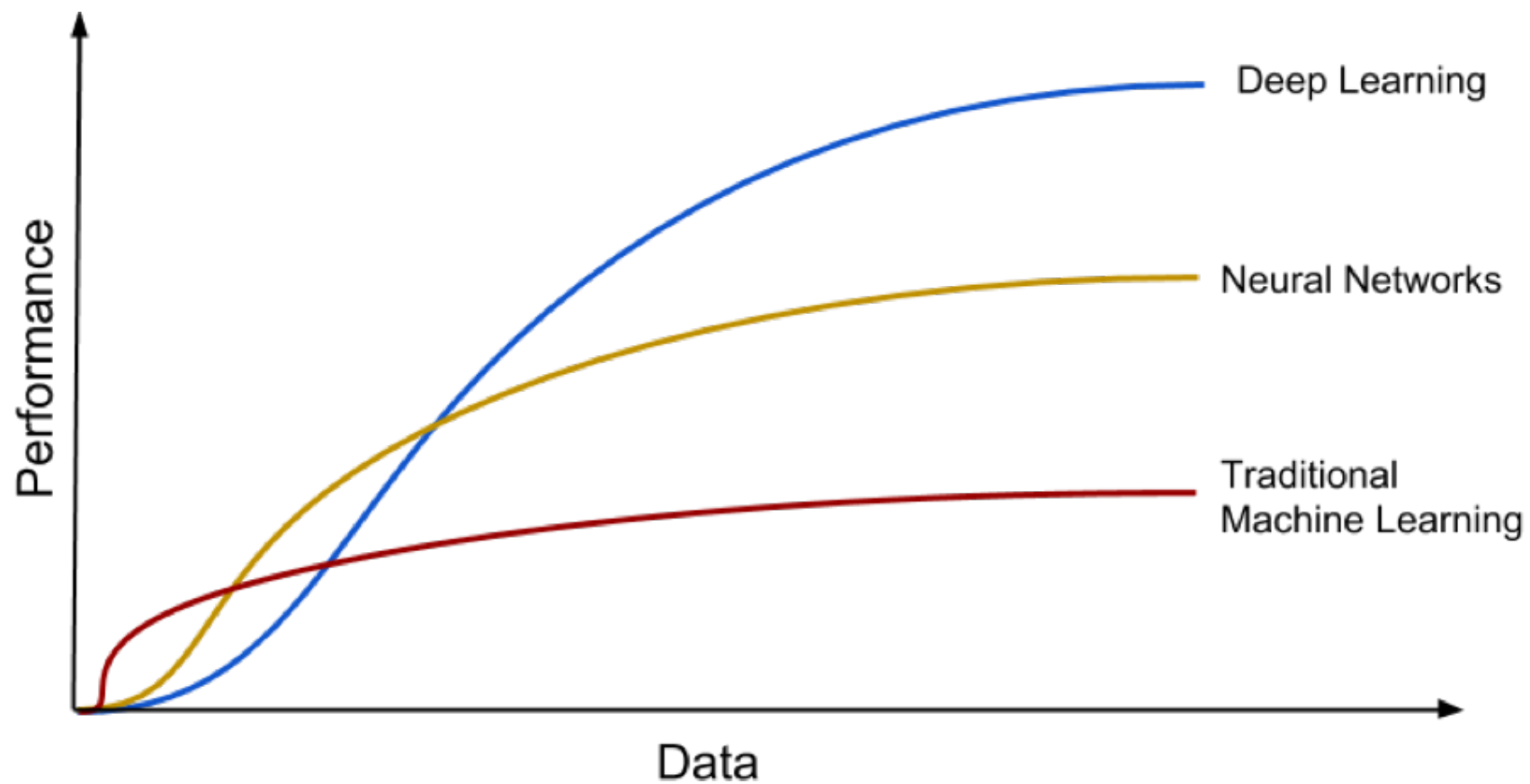


Has wheel	Yes
Weight	3000 lbs
Has door	Yes
Using gas?	No

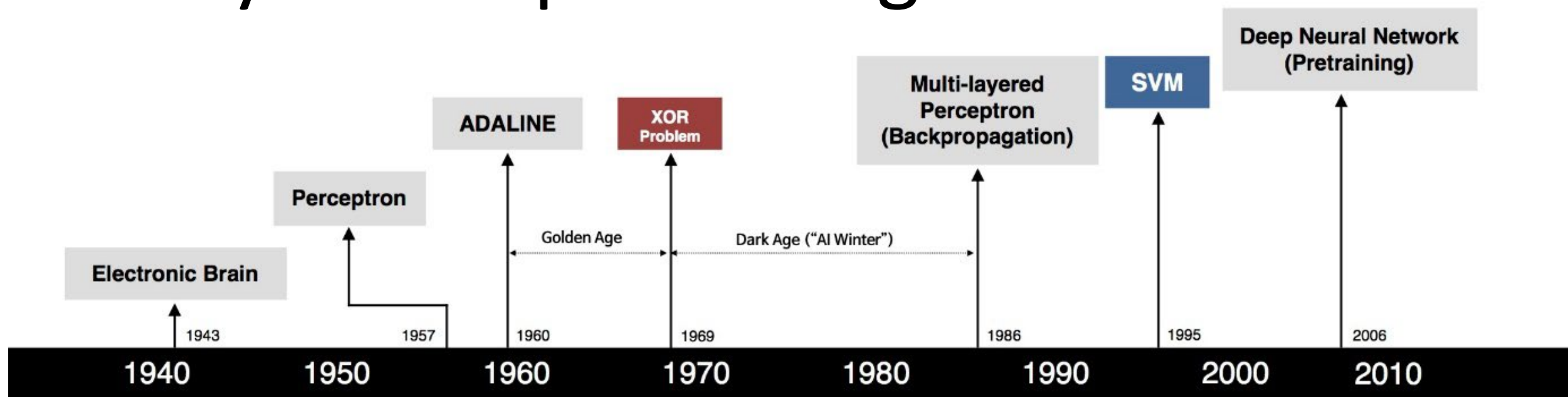


Deep Learning models don't need Feature Extraction

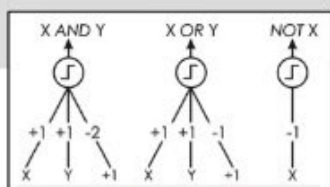
# Introduction



# History of Deep Learning



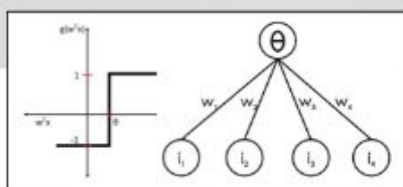
S. McCulloch – W. Pitts



- Adjustable Weights
- Weights are not Learned



F. Rosenblatt



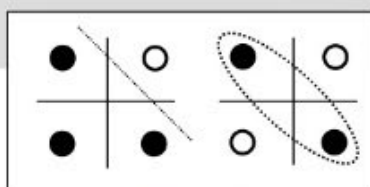
- Learnable Weights and Threshold



B. Widrow – M. Hoff



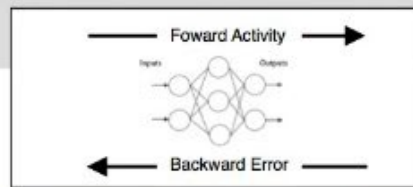
M. Minsky – S. Papert



- XOR Problem



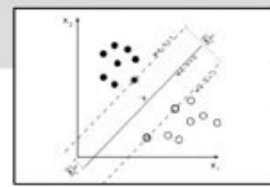
D. Rumelhart – G. Hinton – R. Williams



- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting



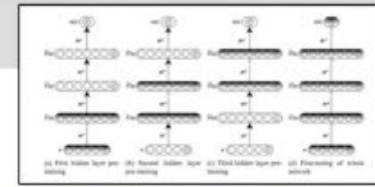
V. Vapnik – C. Cortes



- Limitations of learning prior knowledge
- Kernel function: Human Intervention



G. Hinton – S. Ruslan

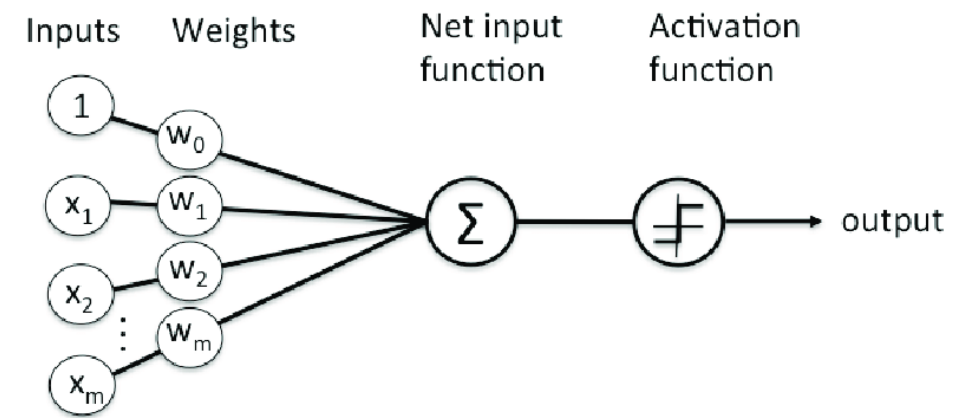


- Hierarchical feature Learning



# 1957-1958: PERCEPTRON

- Perceptron is one of the very first base of DL.
- It was developed by Rosenblatt in 1957 (funded by US Naval Research)
- It is a supervised learning model and applicable if the 2 datasets are linearly separable



Organization of Perceptron (Rosenblatt, 1958)

*1958: [The perceptron is] the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence. (New York Times)*



F. Rosenblatt



## 1969: The First AI WINTER

*However, I started to worry about what such a machine could not do. For example, it could tell 'E's from 'F's, and '5's from '6's—things like that. But when there were disturbing stimuli near these figures that weren't correlated with them the recognition was destroyed.*

- Perceptron gains a decade of attention until Marvin Minsky – often thought of as one of the father of AI, sensed something **off**:
- In a book named “Perceptron”, Minsky and Seymour Papert proved that perceptron was **incapable of learning the simple exclusive-or (XOR) function** (*Now it is obvious that Perceptron is a linear model while XOR is non-linear*)
- This finding shocked the AI community and **Perceptron was interrupted for 20 years**, all research related Perceptron was halt.



M. Minsky





# 1986: The BACKPROPAGANDISTS emerge

- Geoffrey Hinton (PhD in Neural Science) published in Nature: “Learning representations by **back-propagating** errors”.
- The authors showed that NN with many hidden layers (**Multi-Layer Perceptron - MLP**) could be effectively trained by a relatively simple procedure called “Backpropagation”.
- This allows NN to overcome the Perceptron’s weakness with the ability to learn nonlinear function (with nonlinear **activation function like sigmoid, ReLU**)
- NN has ability as “**universal approximation theorem**” and it gets back to the race

## Learning representations by back-propagating errors

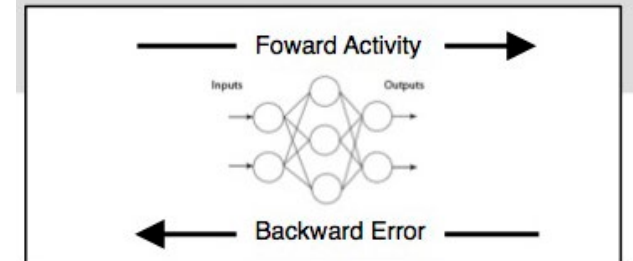
David E. Rumelhart\*, Geoffrey E. Hinton†  
& Ronald J. Williams\*

\* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA

† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA



D. Rumelhart – G. Hinton – R. Williams



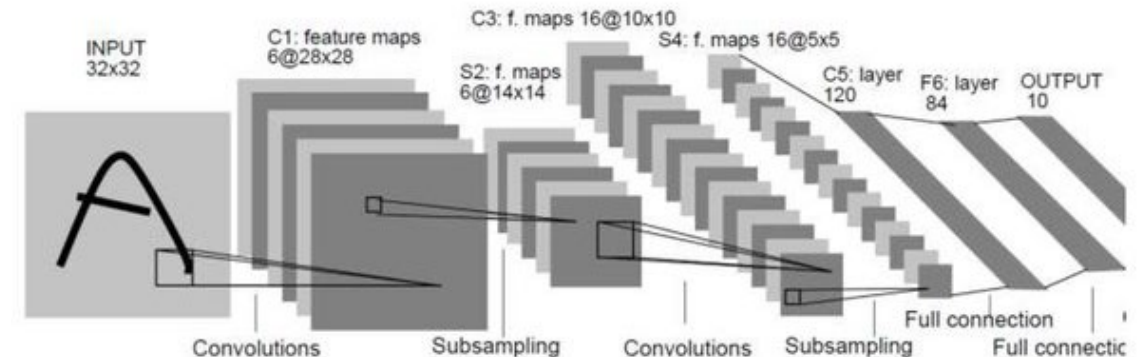
Solution to nonlinearly separable problems •  
Big computation, local optima and overfitting •

# 1986: CNN - LeNet

- Backpropagation leads to some early success (notably Convolution Neural Network: CNN)
- CNN was spearheaded by Yann LeCun at ATT Bell Labs to recognize handwritten digits.
- This model has been used widely in US Banks and Post Office to read handwritten check and postal code
- LeNet was the best algorithm at that time. It is better than regular MLP (with fully connected layers) with the ability to filter in 2D



Lenet-5 (1998)





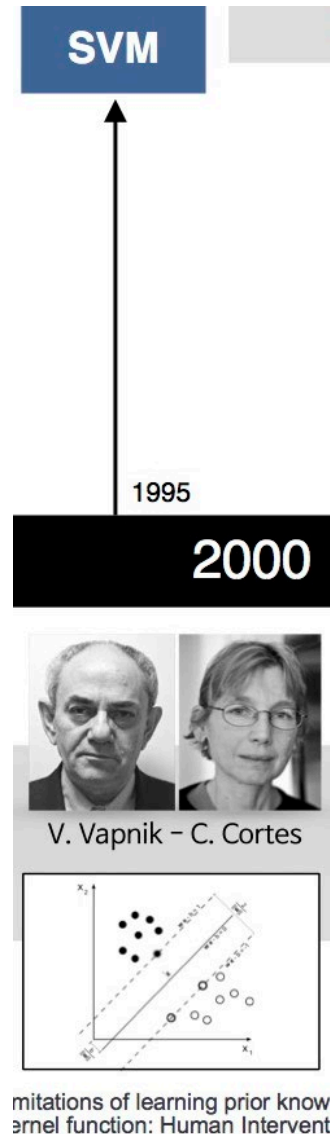
# 1990-2000: The 2nd AI Winter (Deep Freeze)

Several disadvantage of CNN and MLP:

- **Limited in 2D data** (images) for training (digital camera was not popular since then)
- The ability of **computation** machine still limited
- The NN were trained using **stochastic gradient descent** that made it **difficult to optimize** with less data and power
- Many hidden layers with activation function led to **vanishing gradient**

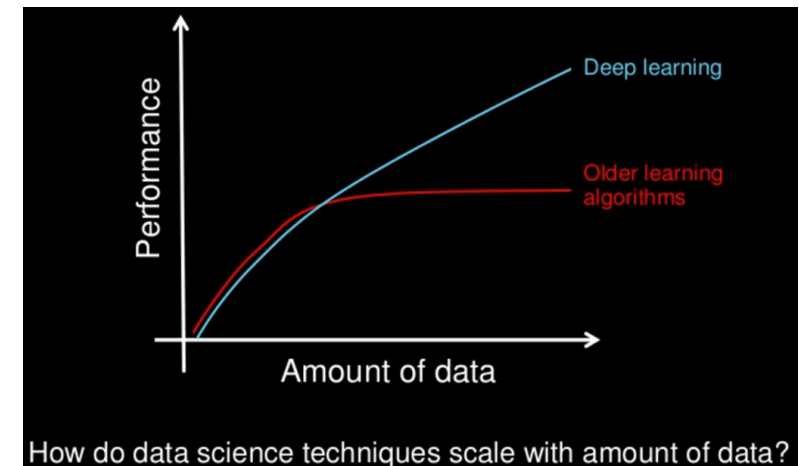
That leads to the Deep Freeze of NN in this decade.

- **Support Vector Machine** emerged during this time with kernel technology to find the optimal parameters
- NN went back to storage and researchers started to move to SVM, except few stubborn researcher ...



## 2006: Deep Learning (Rebranding)

- Hinton once again declared he knew how brain works
- He introduced: “Unsupervised Pretraining” and “Deep Belief Nets” – DBN
- These technique could partially resolve the “vanishing gradient” problem.
- Using this technique, people could train the NN that were deeper than previous attempt => **rebranding to “Deep Learning”**



# 2010: ImageNet

- Feifei Li (a Stanford CS professor) and her group created the dataset named “ImageNet”
- The “ImageNet” has millions of images with thousand of assigned labels
- The project was supported by the booming of internet, digital cameras.
- This dataset has been updated annually and used in the competition named: ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- 2010 & 2011: many teams participated in the competition, they used mostly SVM and the top models had error rates of 28% (2010) and 26% (2011)

IMAGENET

14,197,122 images, 21841 synsets indexed



# 2012: Breakthrough

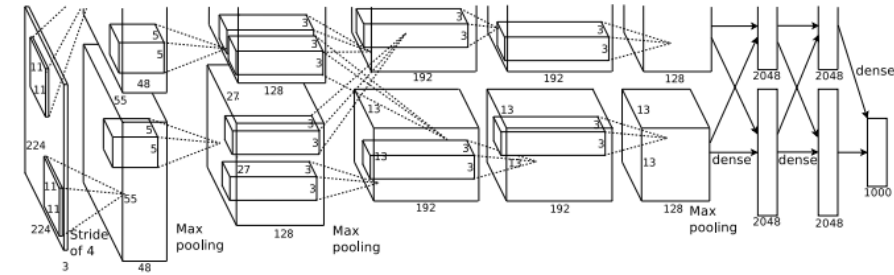
- 2012, Hinton and his student Alex Krizhevsky attended and got first prize in ILSVRC with error rate of **16%** (!!!) that surprised the community: the born of Deep CNN or “**AlexNet**”
- In this competition, there are few key contributions:
  - Introduction of Rectified Linear Activation function (ReLU) to partially overcome “**vanishing gradient**” and increase the computation speed
  - Introducing “**Dropout**” method to shutoff some unused unit, help **to avoid Overfitting** (Similar to Ensemble in ML)
  - AlexNet has great contribution from using **GPUs** that accelerated the computational speed from using CPUs

## ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky  
University of Toronto  
kriz@cs.utoronto.ca

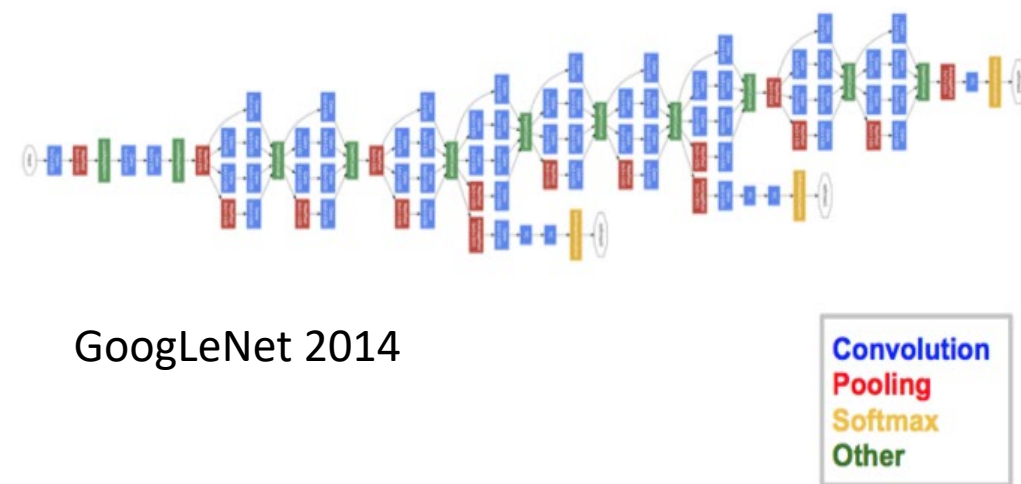
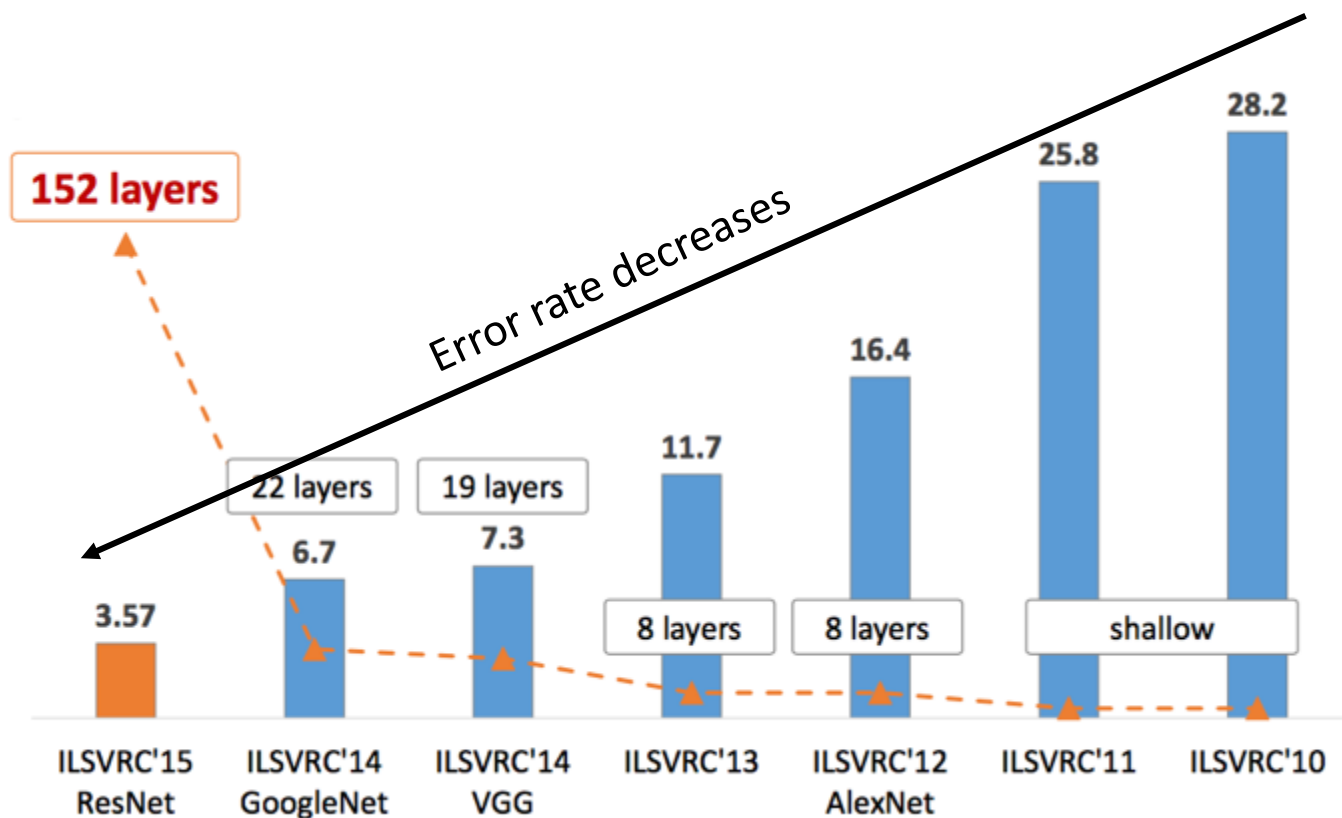
Ilya Sutskever  
University of Toronto  
ilya@cs.utoronto.ca

Geoffrey E. Hinton  
University of Toronto  
hinton@cs.utoronto.ca



# 2013: Since then ...

- After AlexNet, deeper and deeper NN have been applied to ILSVRC
- Increase in hidden layers => decrease error rate





# Discussions:

What brings up the success of Deep Learning?

- New and new **heavy datasets** born with labels (e.g: ImageNet)
- Ability of **GPUs in parallel computing**
- **More activation function** to **overcome vanishing gradient** (ReLU), **exploding gradient** (Gradient clipping)
- Contribution of **tech company** (GoogLeNet, ResNet,...) and other technology like transfer learning, finetuning
- More and more **regularization/ensemble technique** to avoid overfitting: Dropout, Batch Normalization, Data Augmentation
- More and **more library** (many developed by Tech Company): Tensorflow, Pytorch, ...
- **New Optimization technique**: Adam, RMSProp,...

# What can SMU HPC help you with Deep Learning?

SMU M2 HPC features with

- 33 nodes equipped with 1 Tesla P100 GPU/node
- 5 nodes equipped with 8 Tesla V100 GPU/node
- SuperPODS from NVIDIA with 20 nodes equipped with 8 A100 GPUs/node (available this summer)
- High infiniband connection with nearly unlimited storage for your big data model



# APPLICATIONS OF DEEP LEARNING

## Deep Learning Vs Machine Learning

### Factors

Data Requirement

Accuracy

Training Time

Hardware Dependency

Hyperparameter Tuning

### Deep Learning

Requires large data

Provides high accuracy

Takes longer to train

Requires GPU to train properly

Can be tuned in various  
different ways.

### Machine Learning

Can train on lesser data

Gives lesser accuracy

Takes less time to train

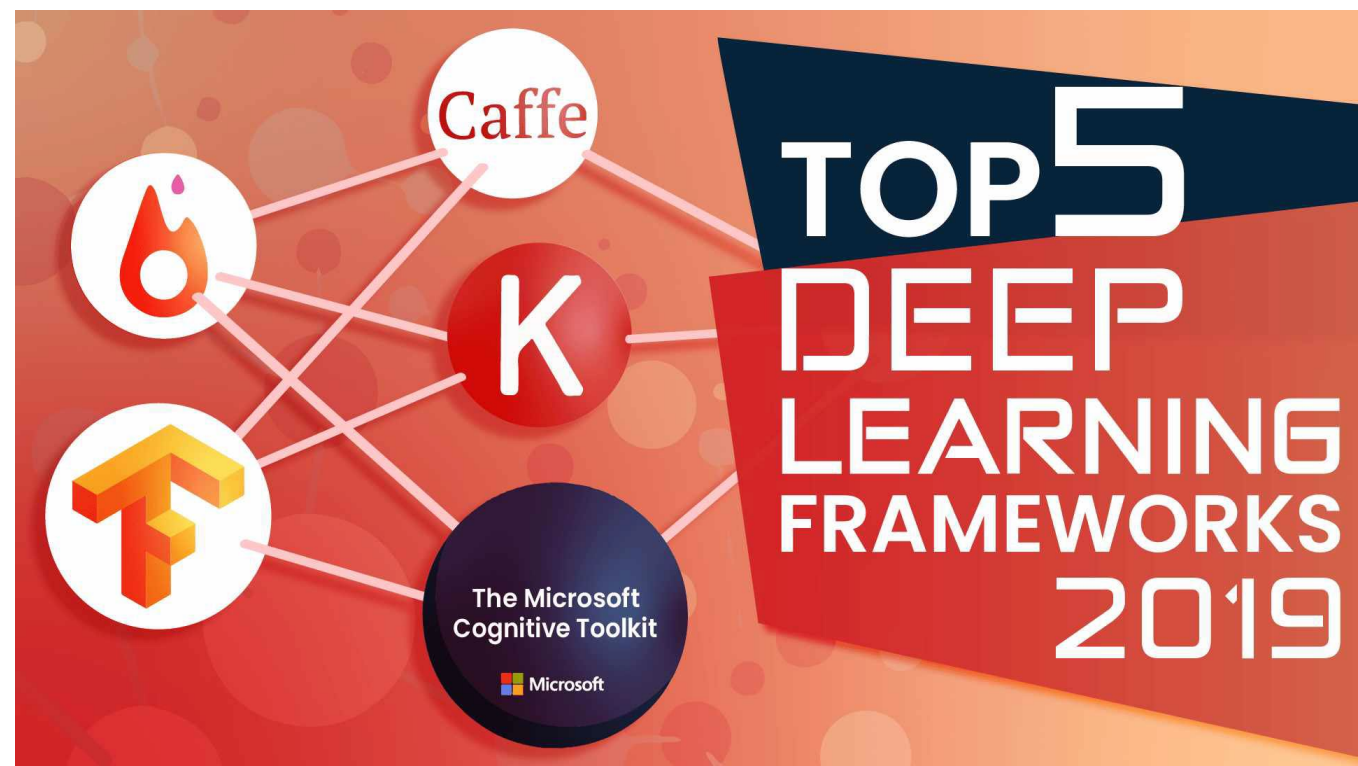
Trains on CPU

Limited tuning capabilities

# 10 Fascinating Applications of Deep Learning



# DEEP LEARNING FRAMEWORK



## (1) Keras

- Keras is an effective high-level neural network [Application Programming Interface](#) (API) written in Python. This open-source neural network library is designed to provide fast experimentation with deep neural networks, and it can run on top of CNTK, TensorFlow, and Theano.
- Keras focuses on being ***modular, user-friendly, and extensible***. It doesn't handle low-level computations; instead, it hands them off to another library called the Backend.
- Keras was [adopted and integrated into TensorFlow](#) in mid-2017. Users can access it via the ***tensorflow.keras*** module. However, the Keras library can still operate separately and independently.



## (2) Tensorflow



- TensorFlow is an end-to-end open-source deep learning framework developed by **Google** and released in **2015**.
- It is known for documentation and training support, scalable production and deployment options, multiple abstraction levels, and support for different platforms, such as Android.
- TensorFlow is a symbolic math library used for neural networks and is best ***suited for dataflow programming across a range of tasks***. It offers multiple abstraction levels for building and training models.

## (3) Pytorch



- Pytorch is a relatively new deep learning framework based on Lua Torch.
- Developed by **Facebook (Meta)**'s AI research group and open-sourced on GitHub in **2017**, it's used for ***natural language processing*** applications.
- Pytorch has a reputation for simplicity, ease of use, flexibility, efficient memory usage, and dynamic computational graphs. It also feels native, making coding more manageable and increasing processing speed.
- Strong competitor to Tensorflow

## (4) Theano

- Theano **used to be one** of the more popular deep learning libraries, an open-source project that lets programmers define, evaluate, and optimize mathematical expressions, including multi-dimensional arrays and matrix-valued expressions.
- Theano was developed by the **Universite de Montreal in 2007** and is a key foundational library used for deep learning in Python. It's considered the grandfather of deep learning frameworks and has fallen out of favor by most researchers outside academia.

(5 and so on)

The number of Deep Learning framework is not just standing still, other tech giants are into the game:

- Microsoft Cognitive Toolkit (CNTK) launched in 2017



- Meta/FB launched Caffe2 as successor to the wellknown Caffe framework (developed by the Berkeley Vision and Learning Center)



- Apache MXnet supported by Microsoft and Amazon and it supports many languages including R (Yes R)



- And many more: Sonnet (from Google DeepMind), H2O.ai, Spark



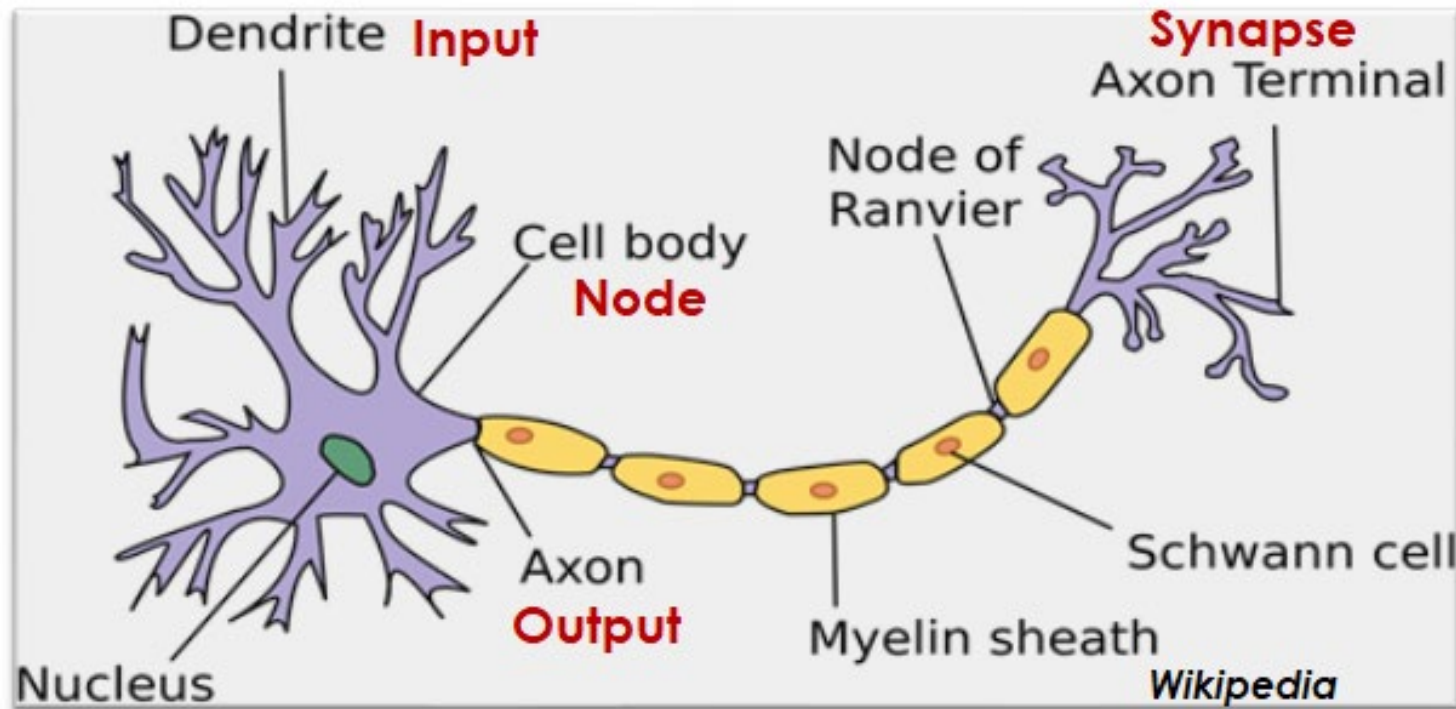
... in comparison

Framework	Keras	Pytorch	TensorFlow
API Level	High	Low	High and Low
Architecture	Simple, concise, readable	Complex, less readable	Not easy to use
Datasets	Smaller datasets	Large datasets, high performance	Large datasets, high performance
Debugging	Simple network, so debugging is not often needed	Good debugging capabilities	Difficult to conduct debugging
Does It Have Trained Models?	Yes	Yes	Yes
Popularity	Most popular	Third most popular	Second most popular
Speed	Slow, low performance	Fast, high-performance	Fast, high-performance
Written In	Python	Lua	C++, CUDA, Python

# Conclusions:

- Keras is the easiest Deep Learning Library API. It is easy to transit from scikit-learn based model
- Tensorflow is the most popular DL library developed by Google. It has the most number of user.
- Pytorch is developed by Facebook and gaining significant attention from users all over the world

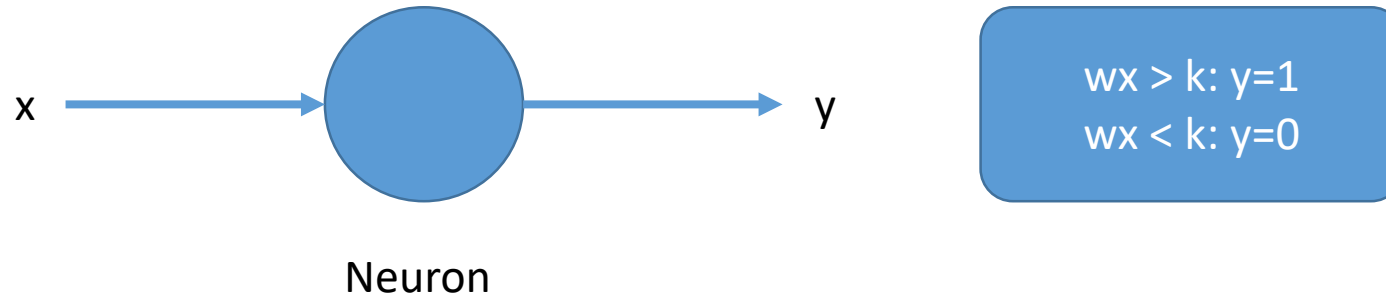
# Recap on ANN: Biological Neural Network



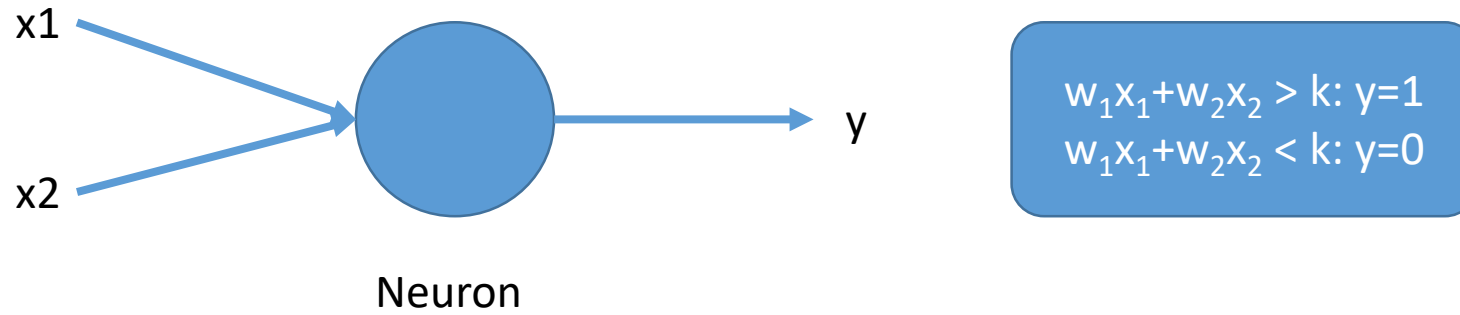
Biological Neural Network



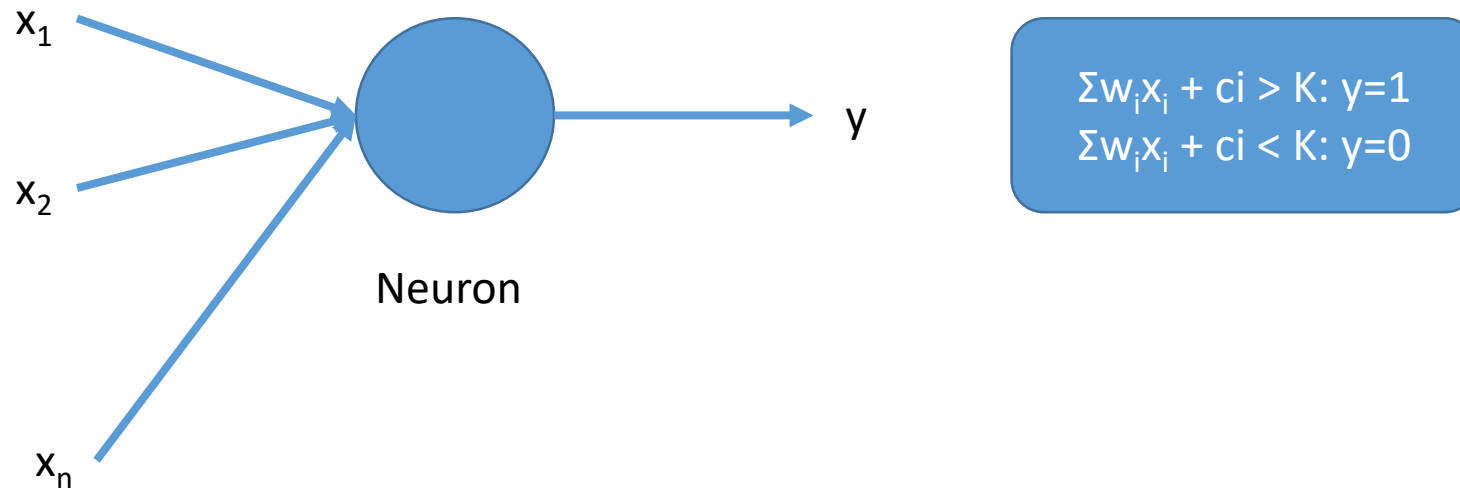
# Recap on ANN: Perceptron



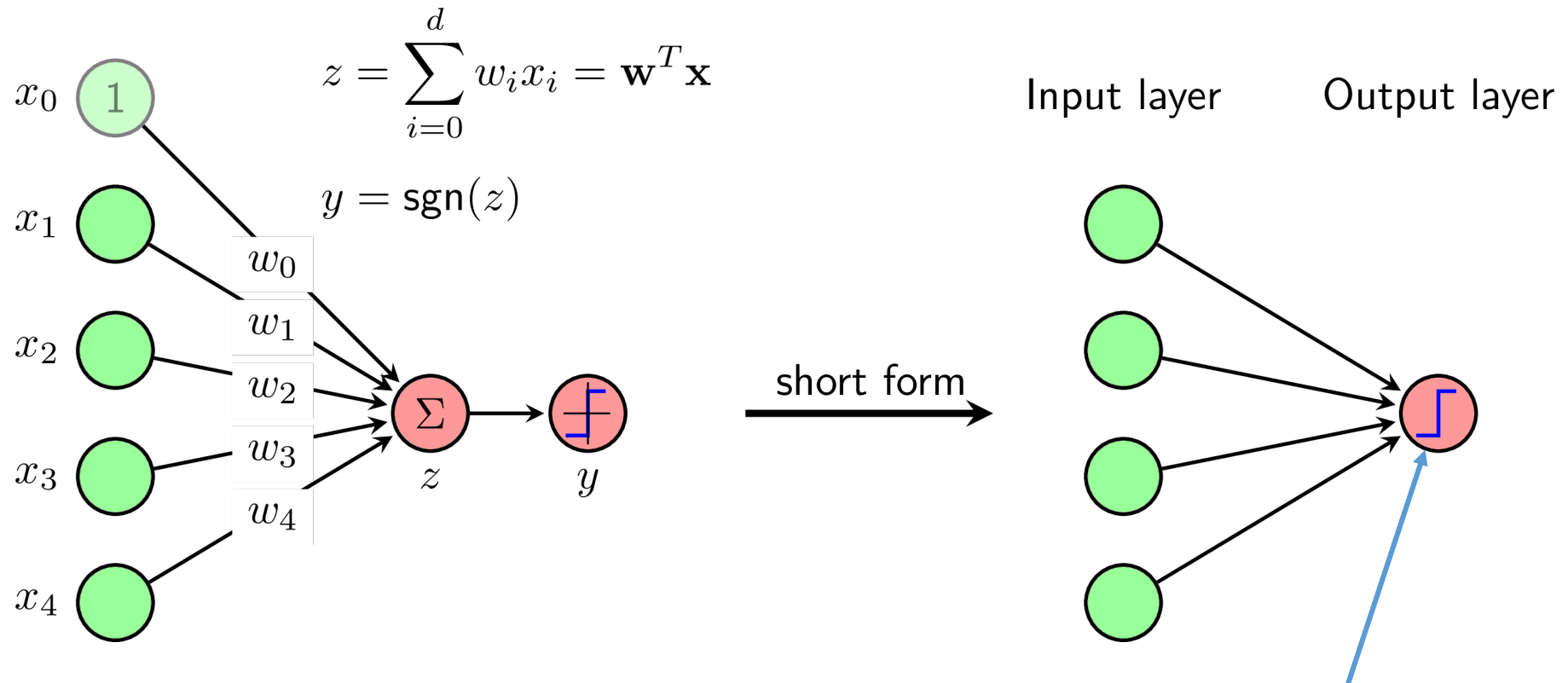
# Recap on ANN: Perceptron



# Recap on ANN: Perceptron



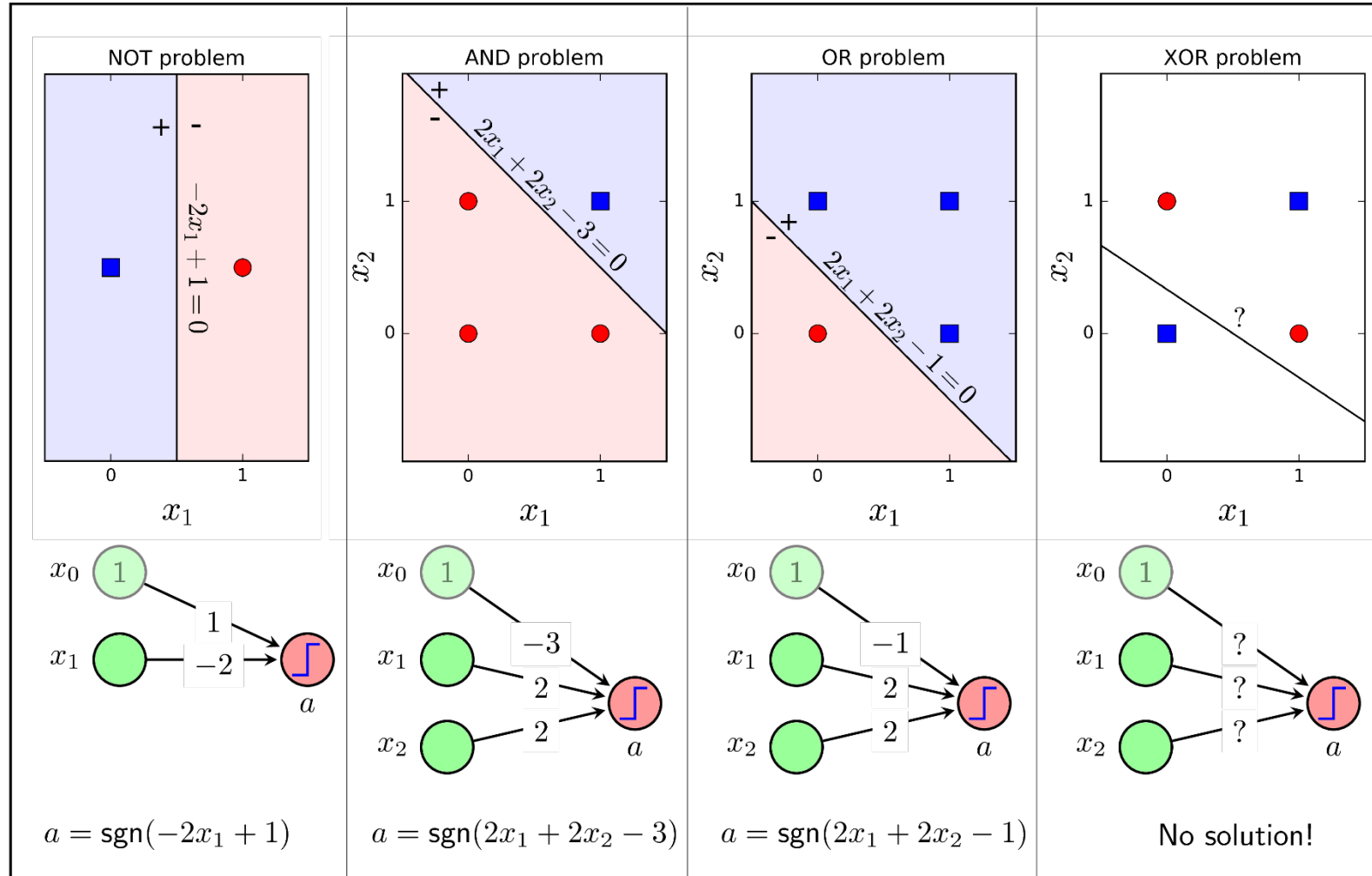
# Recap on ANN: The first Neural Network



Note: Linear function cannot capture the Non-linearity

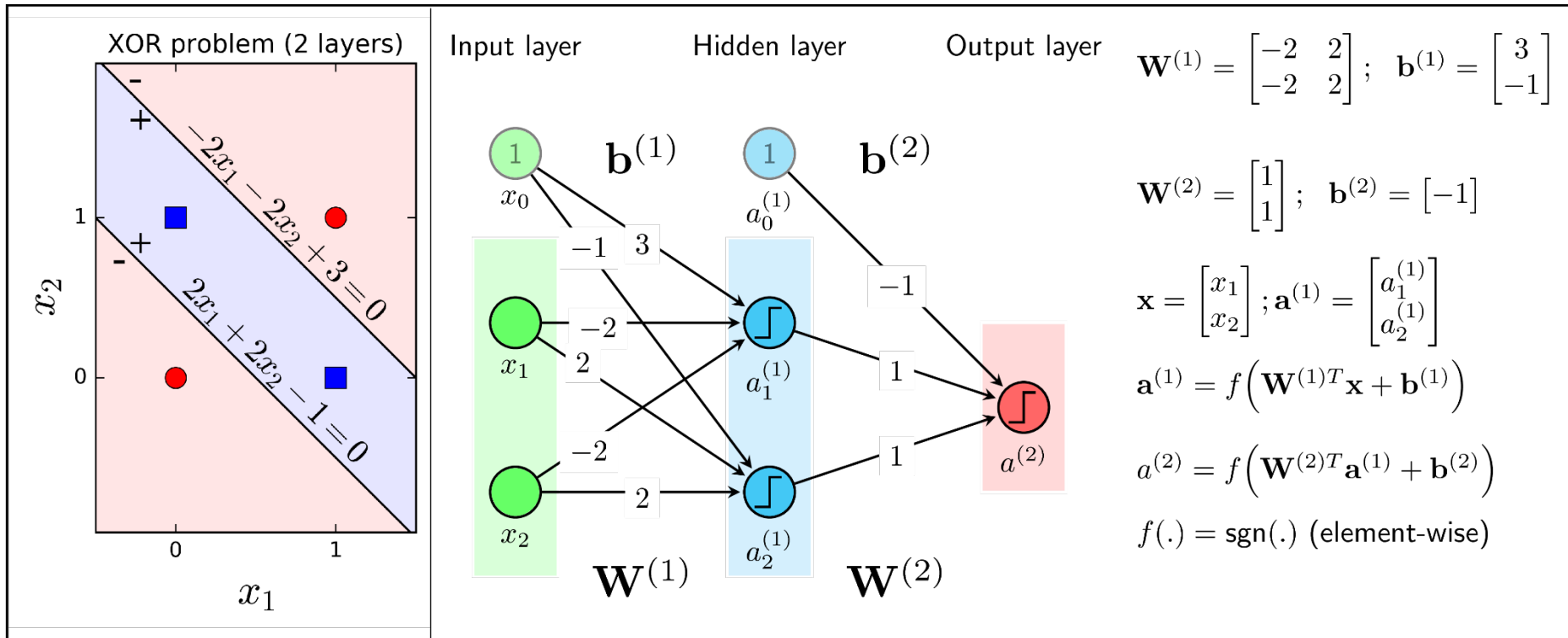
The sign activation function: the simplest form of Neural Network

# Recap on ANN: Multi-Layer Perceptron



The Perceptron Layer Algorithm works well for linear separable problem with AND, OR problem but not with nonlinear

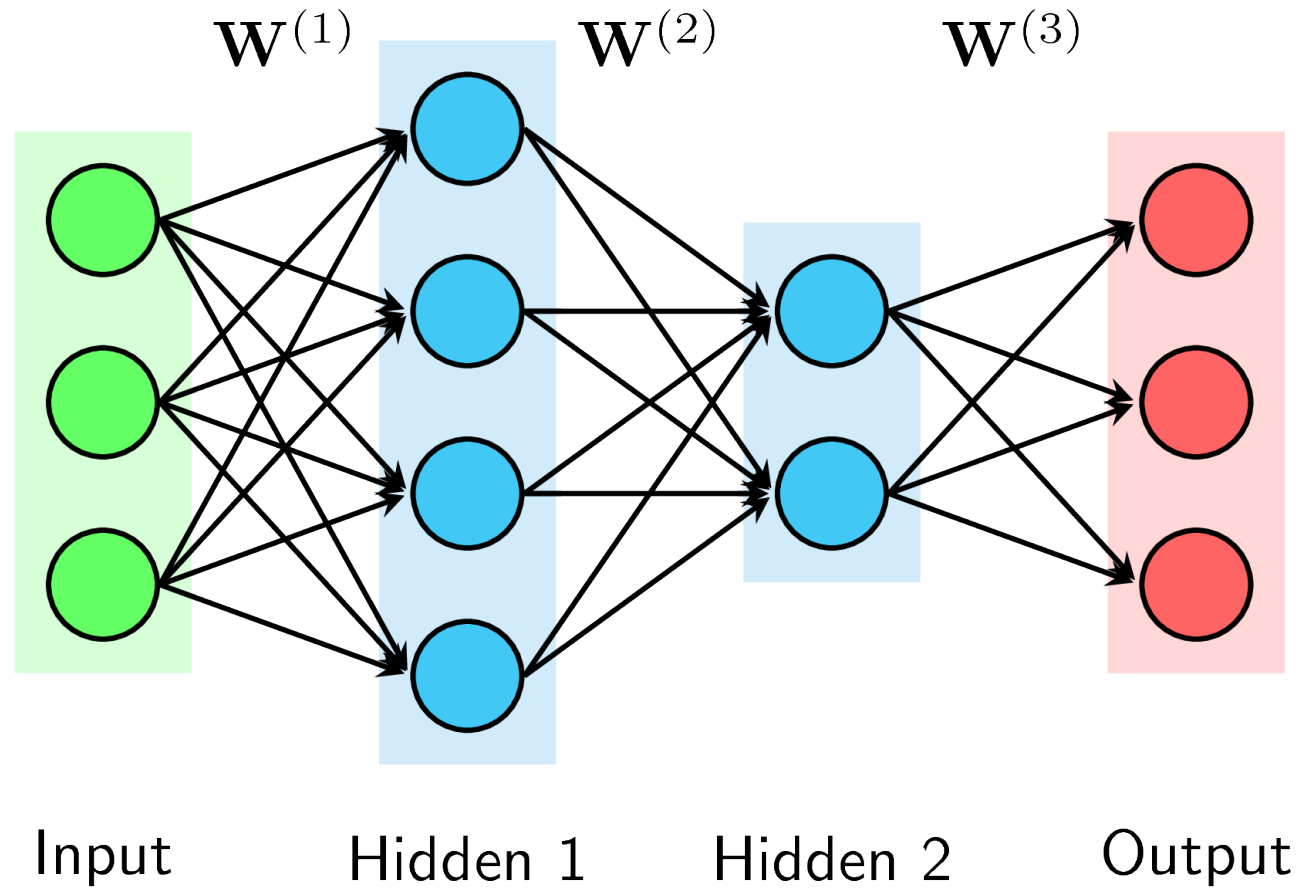
# Recap on ANN: Multi-Layer Perceptron



For Nonlinear separable, one more layer should be added: **hidden layer**.

The Model is called **Multi-Layer Perceptron**

# Recap on ANN: Multi-Layer Perceptron



MLP can have more hidden layers.

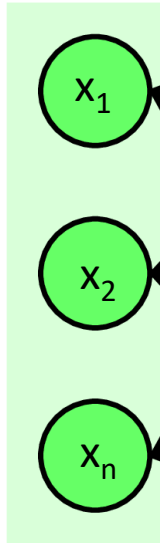
When you refer to the number of layers (**L**) of MLP, you should NOT count the input layers.

In the above figures: the  $L = 3$



# Recap on ANN: Multi-Layer Perceptron

**Input Layer:** The number of neurons in input layers equal to the input vector size

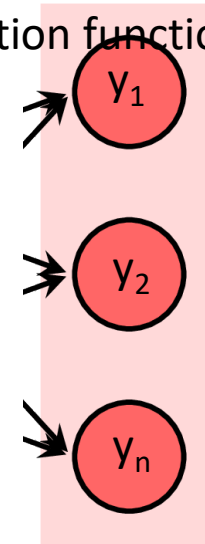


Input

# Recap on ANN: Multi-Layer Perceptron

## Output Layer:

- Classifier: The number of neurons in output layers equal to the number of categories
- Regression: one or many neurons in output layers but the activation function needs to produce continuous values

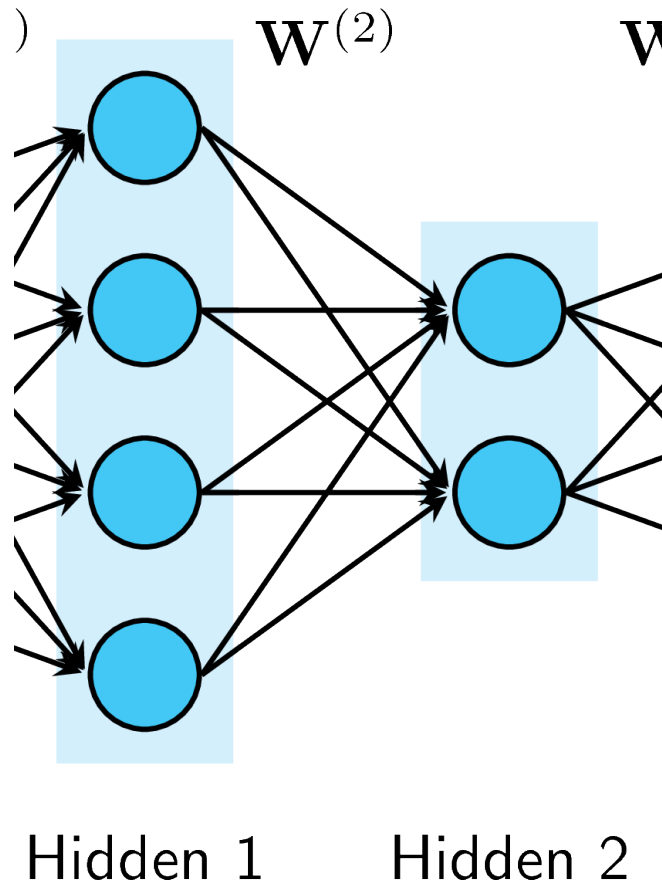


Output

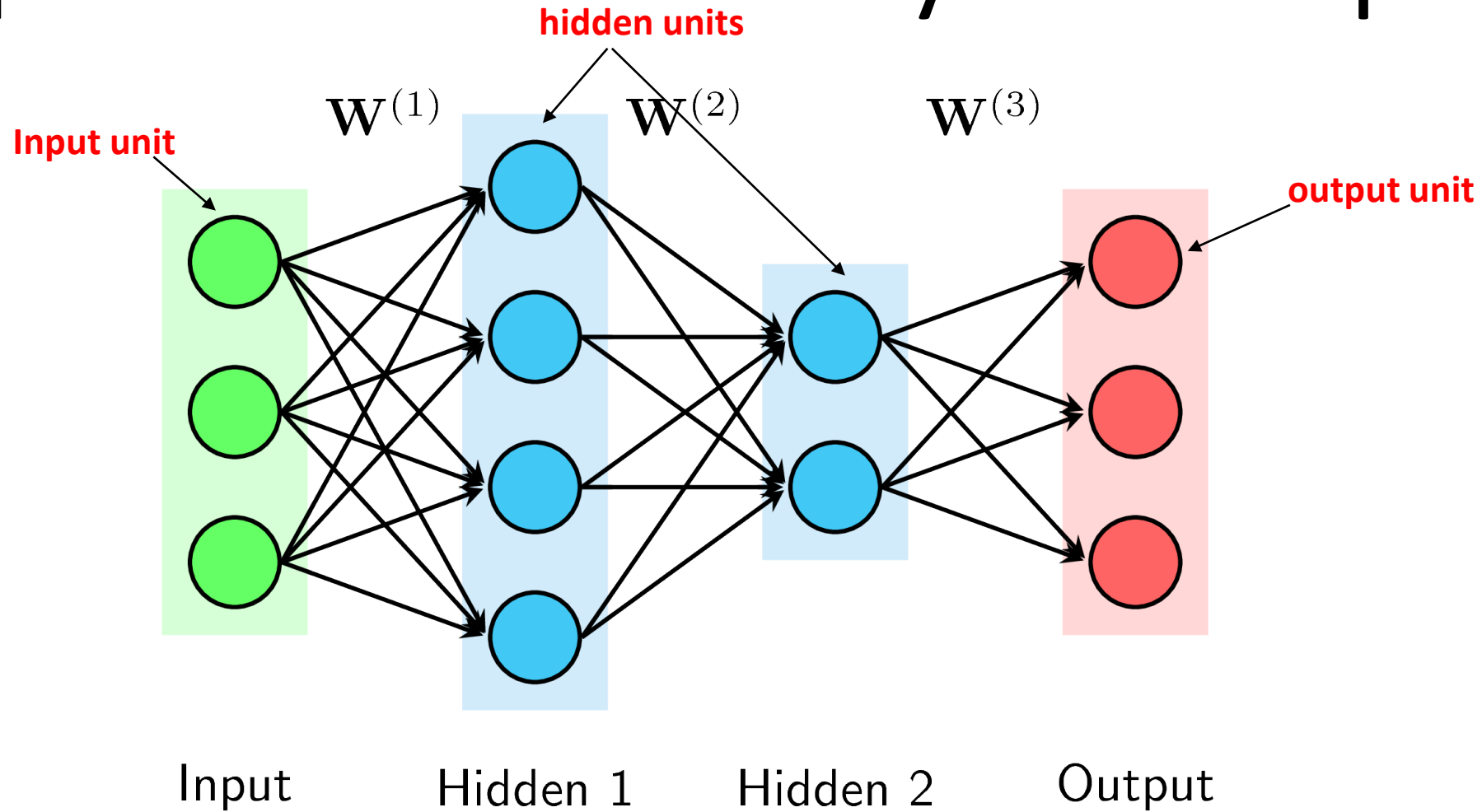
# Recap on ANN: Multi-Layer Perceptron

## Hidden Layers:

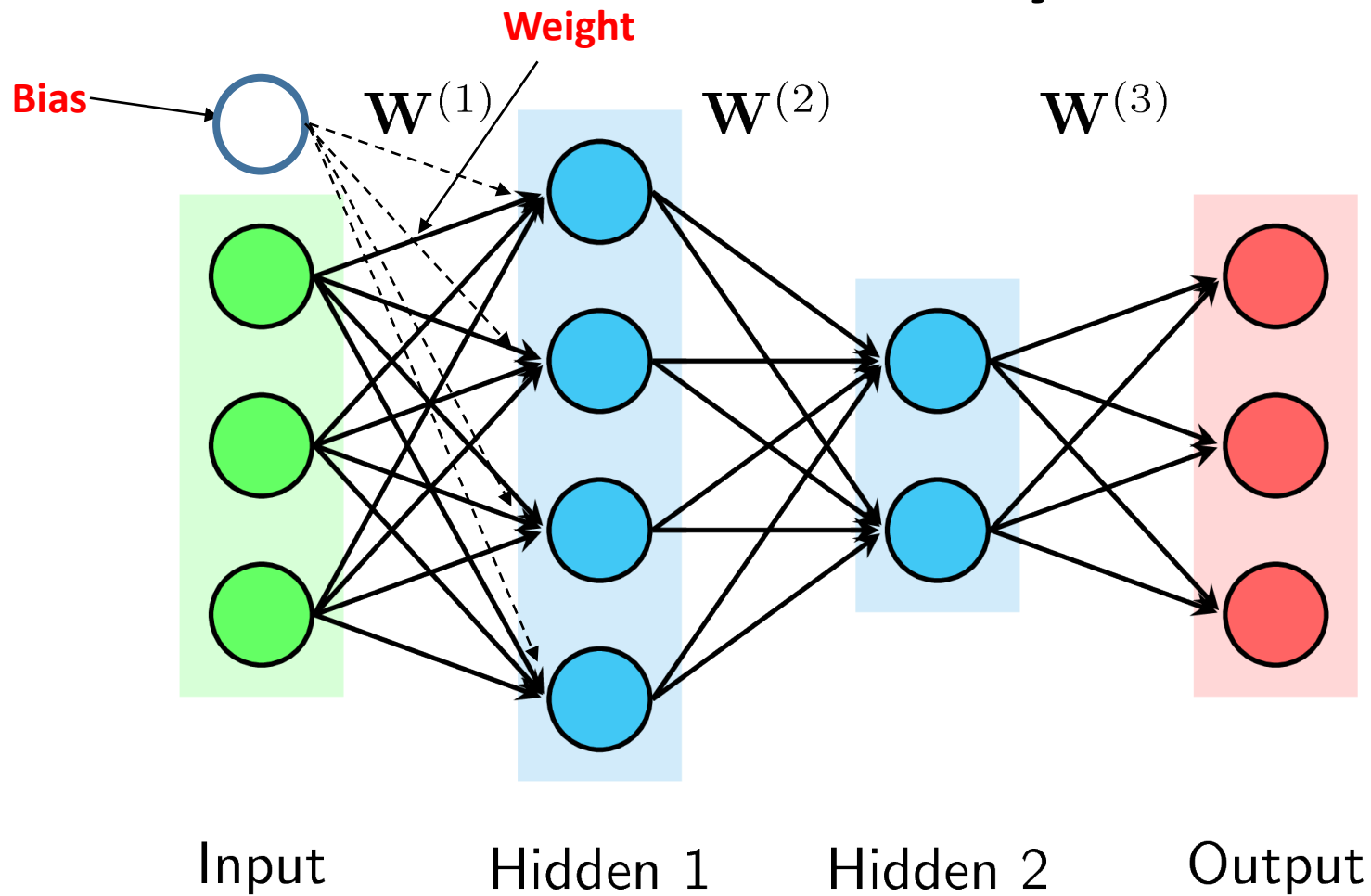
- Weights are updated



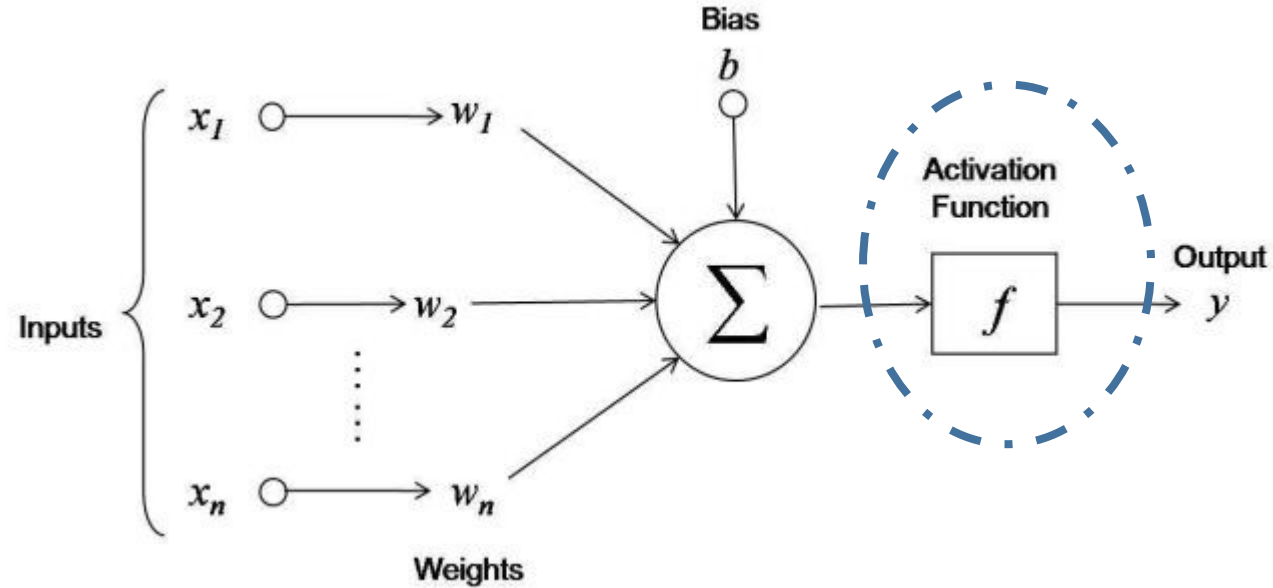
# Recap on ANN: Multi-Layer Perceptron



# Recap on ANN: Multi-Layer Perceptron



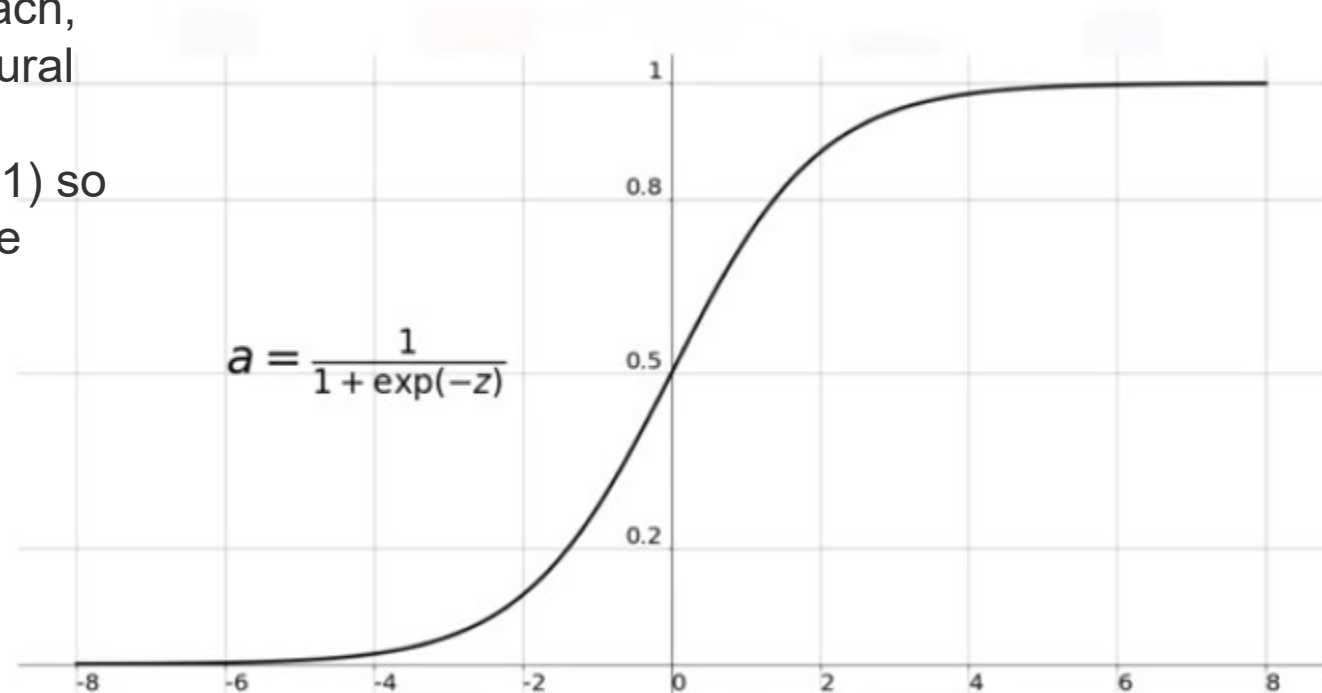
# Recap on ANN: Activation functions



# Recap on ANN: Activation functions

## Sigmoid Function:

- One of the widely used activation function in the hidden layer of NN
- However, it is flat with  $\text{abs}(z) > 3$ , therefore it might lead to “vanishing gradient” in Backpropagation approach, that slowdown the optimization of NN in Deep Neural Network.
- Sigmoid function converts the output to range (0, 1) so it is not symmetric around the origin. All values are positive.
- Application in binary classification problems.





# Recap on ANN: Activation functions

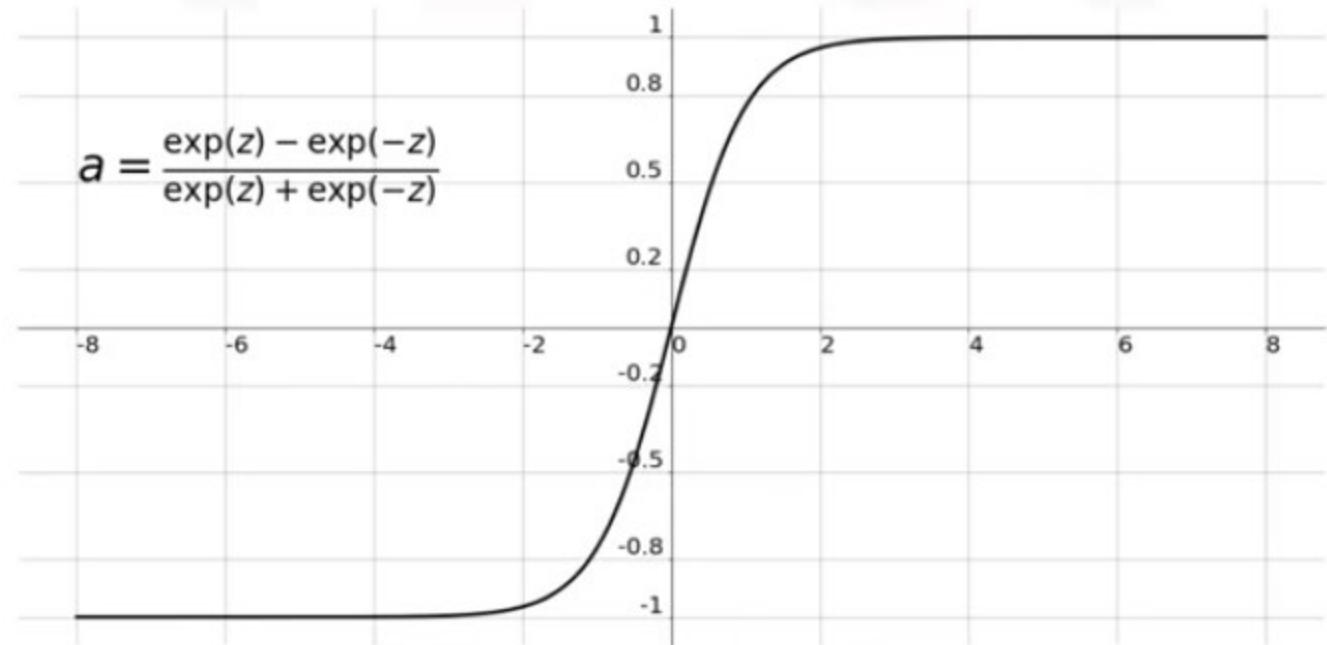
## Softmax

- The softmax is a more generalised form of the sigmoid.
- It is used in multi-class classification problems.
- Similar to sigmoid, it produces values in the range of 0–1 therefore it is used as the final layer in classification models.
- Application in classification with more categories

# Recap on ANN: Activation functions

## Hyperbolic Tangent (Tanh)

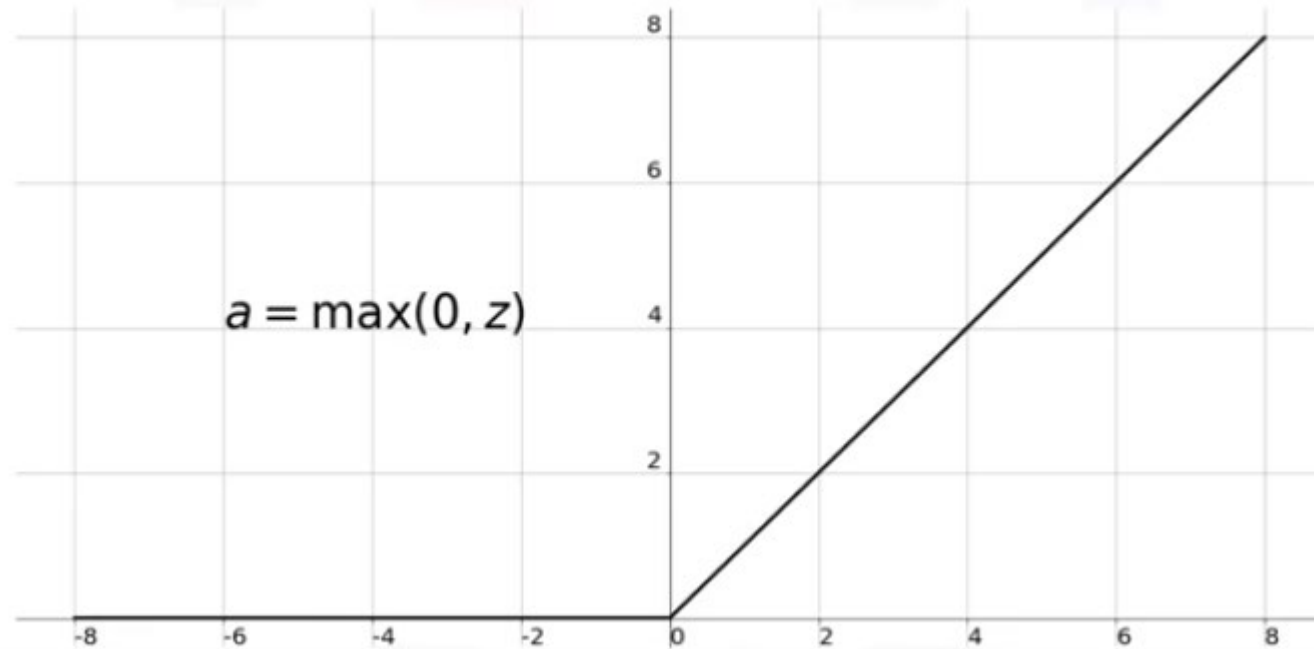
- Tanh is quite similar to Sigmoid but it is symmetric around the origin
- However, it also flat with  $\text{abs}(z) > 3$  and also lead to “vanishing gradient” problem in Deep Neural Network



# Recap on ANN: Activation functions

## Rectangular Linear Unit (ReLU)

- The most widely used Activation Function in Deep Neural Network
- It is nonlinear
- It does not activate all neuron at the same time: If the input is negative, the neuron is not activated
- Therefore, it overcomes the “vanishing gradient” problem



# Recap on ANN: Gradient Problems

## Vanishing gradient

- In a network of  $n$  hidden layers,  $n$  derivatives will be multiplied together. If the derivatives are small then the gradient will decrease exponentially as we propagate through the model until it eventually vanishes
- The accumulation of small gradients results in a model that is incapable of learning meaningful insights since the weights and biases of the initial layers, which tends to learn the core features from the input data ( $X$ ), will not be updated effectively. In the worst case scenario the gradient will be 0 which in turn will stop the network and stop further training.

# Recap on ANN: Gradient Problems

## Exploding gradient

- If the derivatives are large then the gradient will increase exponentially as we propagate down the model until they eventually explode, and this is what we call the problem of exploding gradient
- The accumulation of large derivatives results in the model being very unstable and incapable of effective learning, The large changes in the models weights creates a very unstable network, which at extreme values the weights become so large that it causes overflow resulting in NaN weight values of which can no longer be updated.

# Recap on ANN: Gradient Problems

## Solutions

- Reduce the amount of layer
- Proper weights for initialization