Neural Networks & Deep Learning

Sina K. Maram, M.Sc., P.Eng.

Session 1 – Exploring & Applications

By the end of this lecture you'd be able to:

- Understand the progression of Neural Networks from Simple Classification to Advanced Predictor
- Become familiar with structure of a simple neuron
- Define Forward and Backward Propagation on a single node
- Different architectures of utilizing neurons and their applications
- Challenges in Developing a Neural Network Model
- Safeguards available to protect your model from common shortcomings

Agenda:

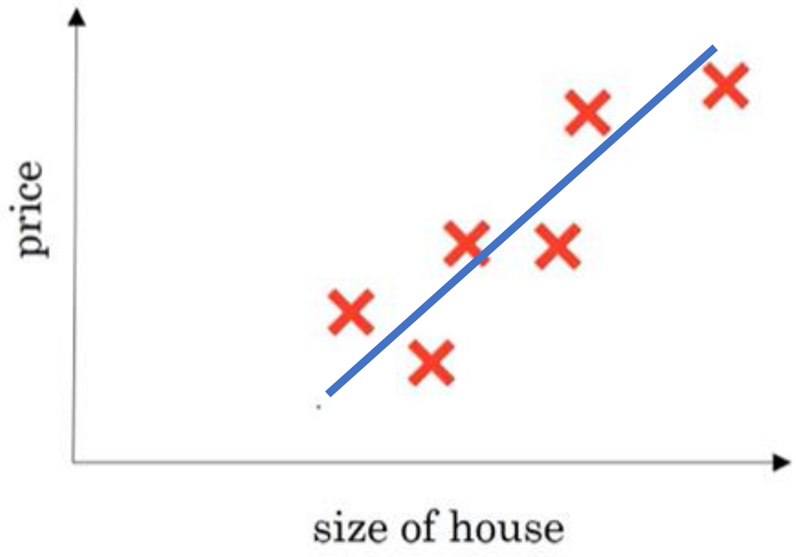
01 **Introduction to Neural Networks**
History, Evolution & Building Blocks

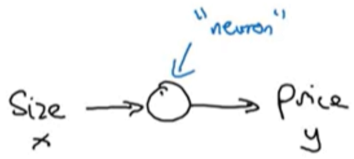
02 **How Neural Networks Work?**
Gradient Descent, Forward & Backward Propagation

03 **Types of Neural Networks**
Perceptrons and FCN's, RNN's, CNN's

04 **Training Challenges and Strategies**
Safeguards against Gradient Vanishing/Exploding

- What is a Neural Network?

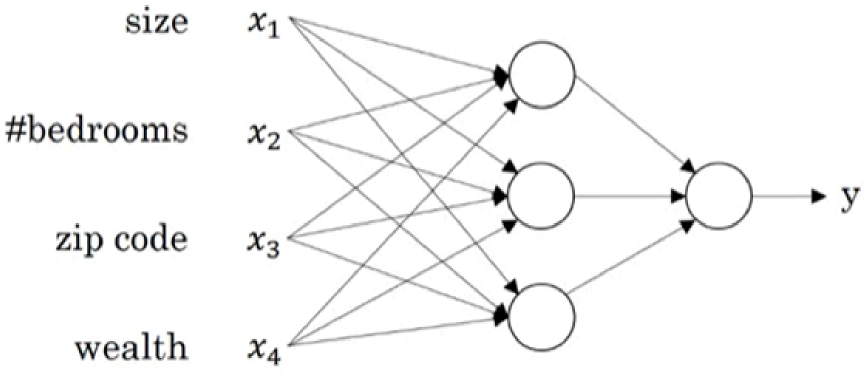




Single Perceptron

Qualities:

- Simple node structure
- Highly customizable architecture
- High performer in supervised learning use-cases
- An adaptation of human neural network



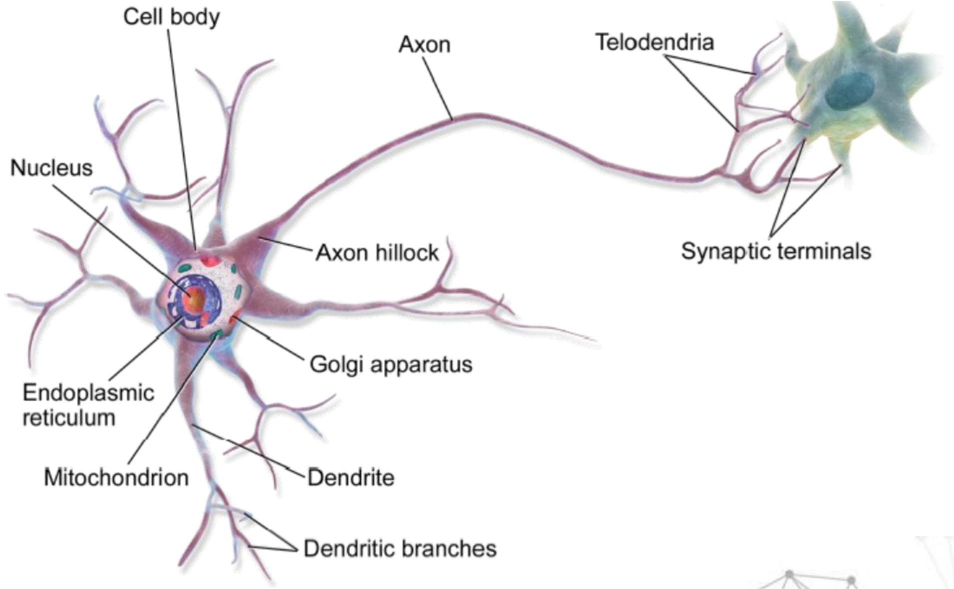
Multi-Layer Network

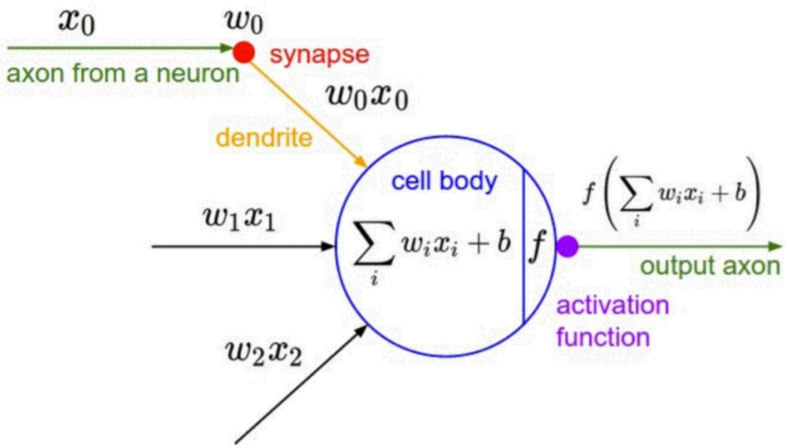
01

Introduction to Neural Networks

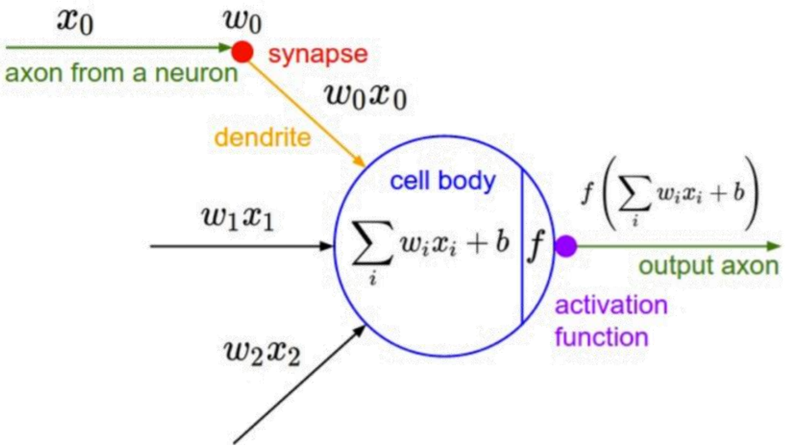
History, Evolution & Building Blocks

- Proposed units in Neural Networks are inspired by Biological Neural Networks (BNN's) that exist in the brain.
- Simplified model of Neurons was first proposed by McCulloch and Pitts (1943). These Binary (2 inputs → 1 output) models were able to replicated logical expressions

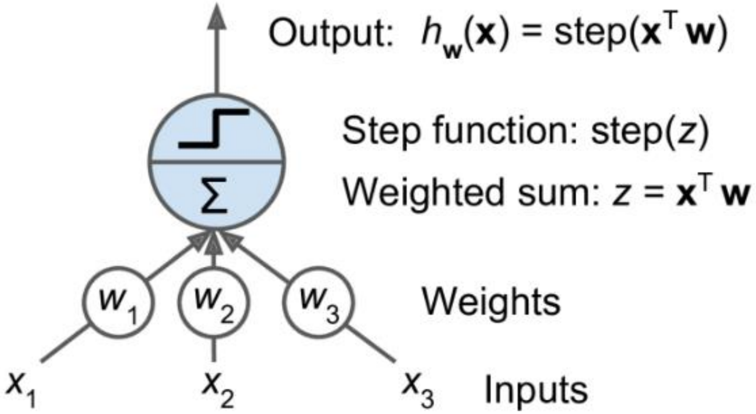




Perceptron model was introduced by Rosenblatt (1957) where inputs could be a float value transformed through a heavside (step) function.



Perceptron model was introduced by Rosenblatt (1957) where inputs could be a float value transformed through a heavside (step) function.

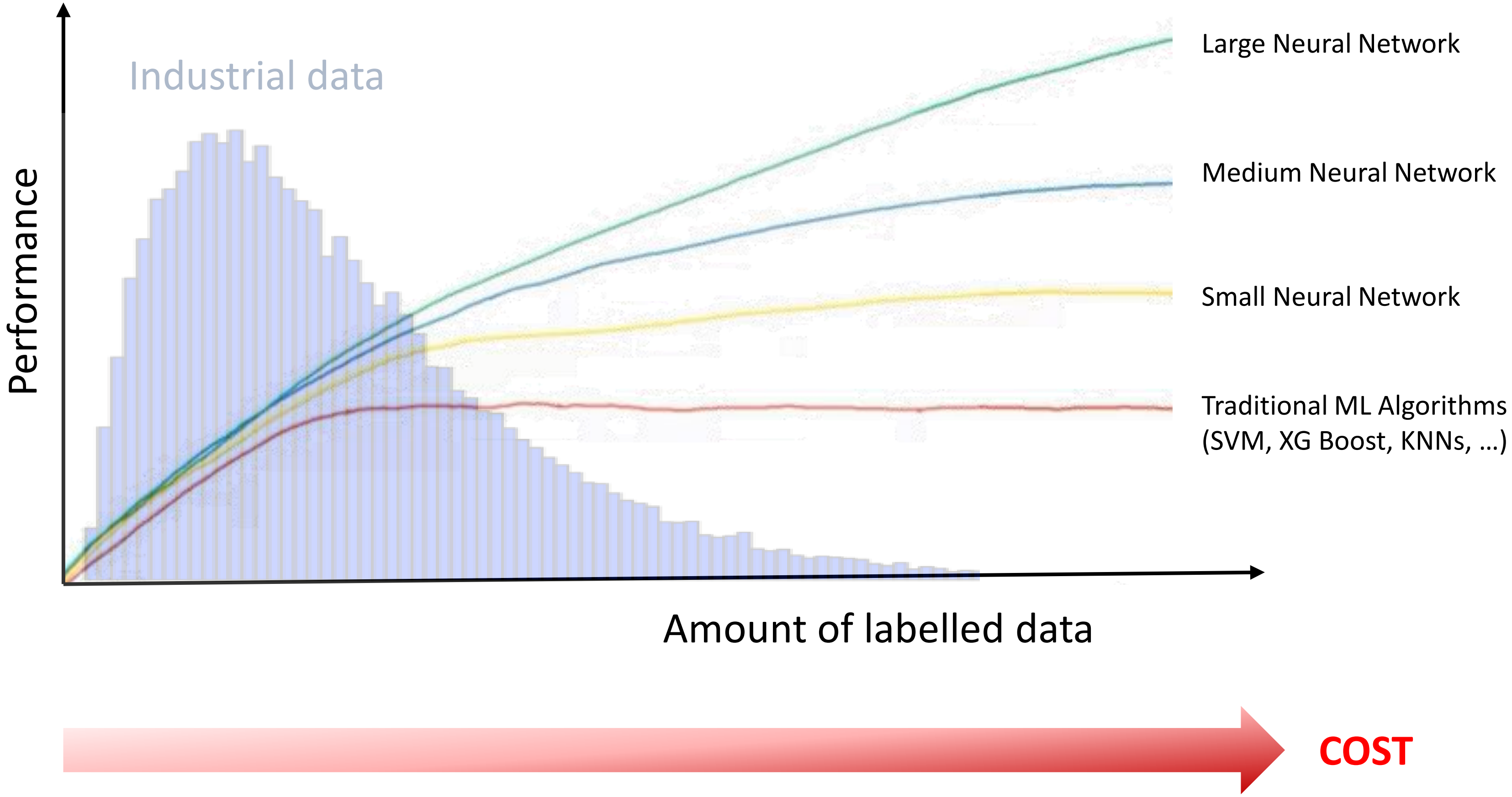


$$\text{heaviside}\ (z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

$$\text{sgn}\ (z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

BNN Image by Bruce Blaus (Creative Commons 3.0) Reproduced from http://en.Wikipedia.org/wiki/Neuron

Why is Deep Learning becoming popular:
The main driving factor of deep learning is scale!

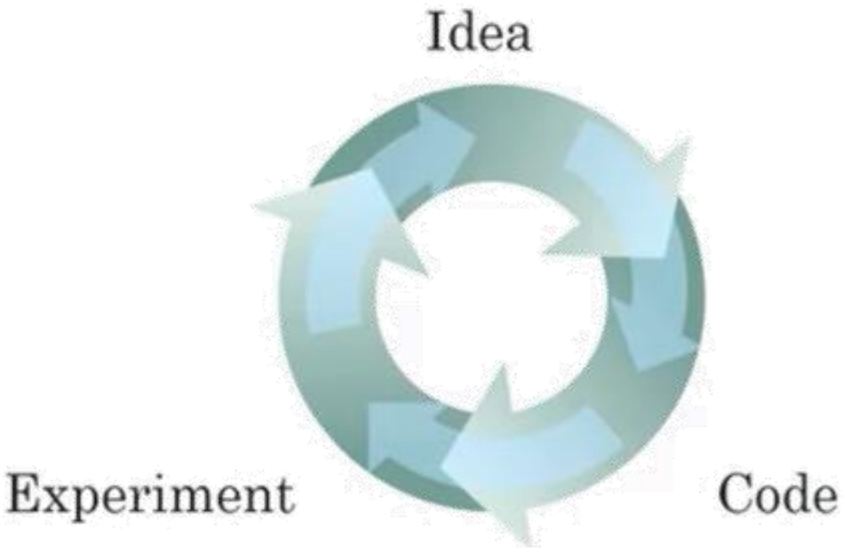


Reference, Andrew Ng on Deep Learning

Why is Deep Learning becoming popular:

Other factors:

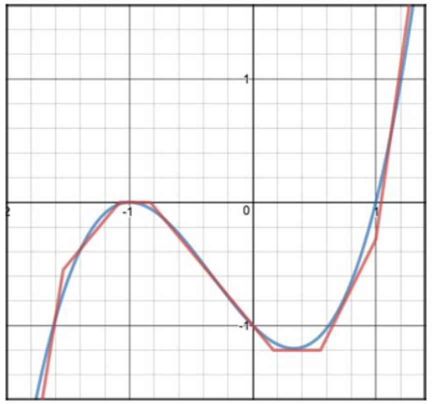
1. **Data:** Businesses start to leverage technology to collect data
2. **Computation:** Cloud Computing, distributed ML Computation
3. **Algorithms:** Innovative architectures, improvements in core node structure



Reference, Andrew Ng on Deep Learning

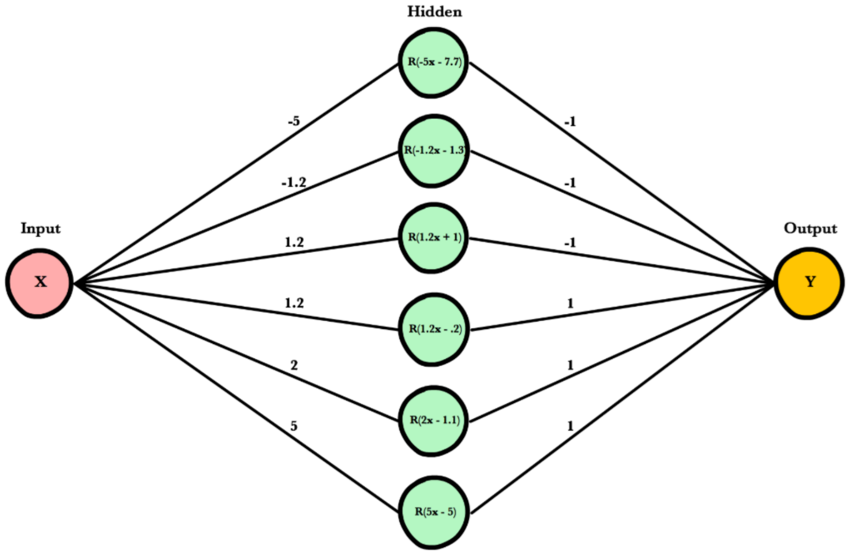
Universal Approximation Theorem:

A common principle throughout computer science is that we can build complicated systems from minimal components. E.g. Turing Machine’s memory needs only to be able to store 0 or 1 states, we can build a universal function approximator from rectified linear functions*



$$
\begin{aligned}
n_1(x) &= Relu(-5x - 7.7)\\
n_2(x) &= Relu(-1.2x - 1.3)\\
n_3(x) &= Relu(1.2x + 1)\\
n_4(x) &= Relu(1.2x - .2)\\
n_5(x) &= Relu(2x - 1.1)\\
n_6(x) &= Relu(5x - 5)
\end{aligned}
$$

$$
\begin{aligned}
Z(x) = &-n_1(x) - n_2(x) - n_3(x)\\
&+ n_4(x) + n_5(x) + n_6(x)
\end{aligned}
$$

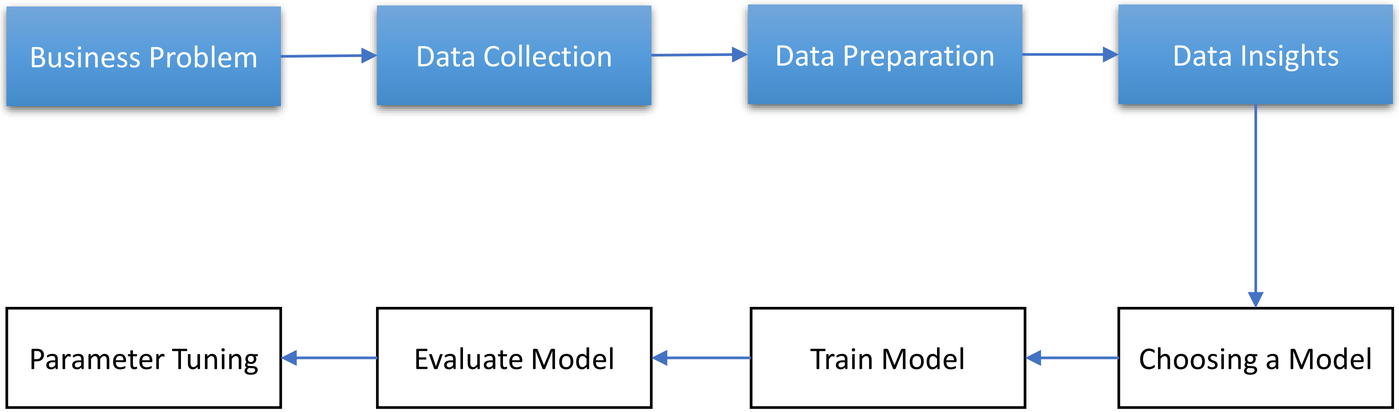


Attempt to replicated $y = (x^3 + x^2 - x \ - 1)$ function using ReLU step functions (https://towardsdatascience.com/can-neural-networks-really-learn-any-function-65e106617fc6)

Chapter 6, Deep Learning Book (http://Deeplearningbook.org/contents/mlp.html)

Some warm-up Classroom Assignment

Assignment: Anomaly Detection in Cellular network
Group size: Maximum 3

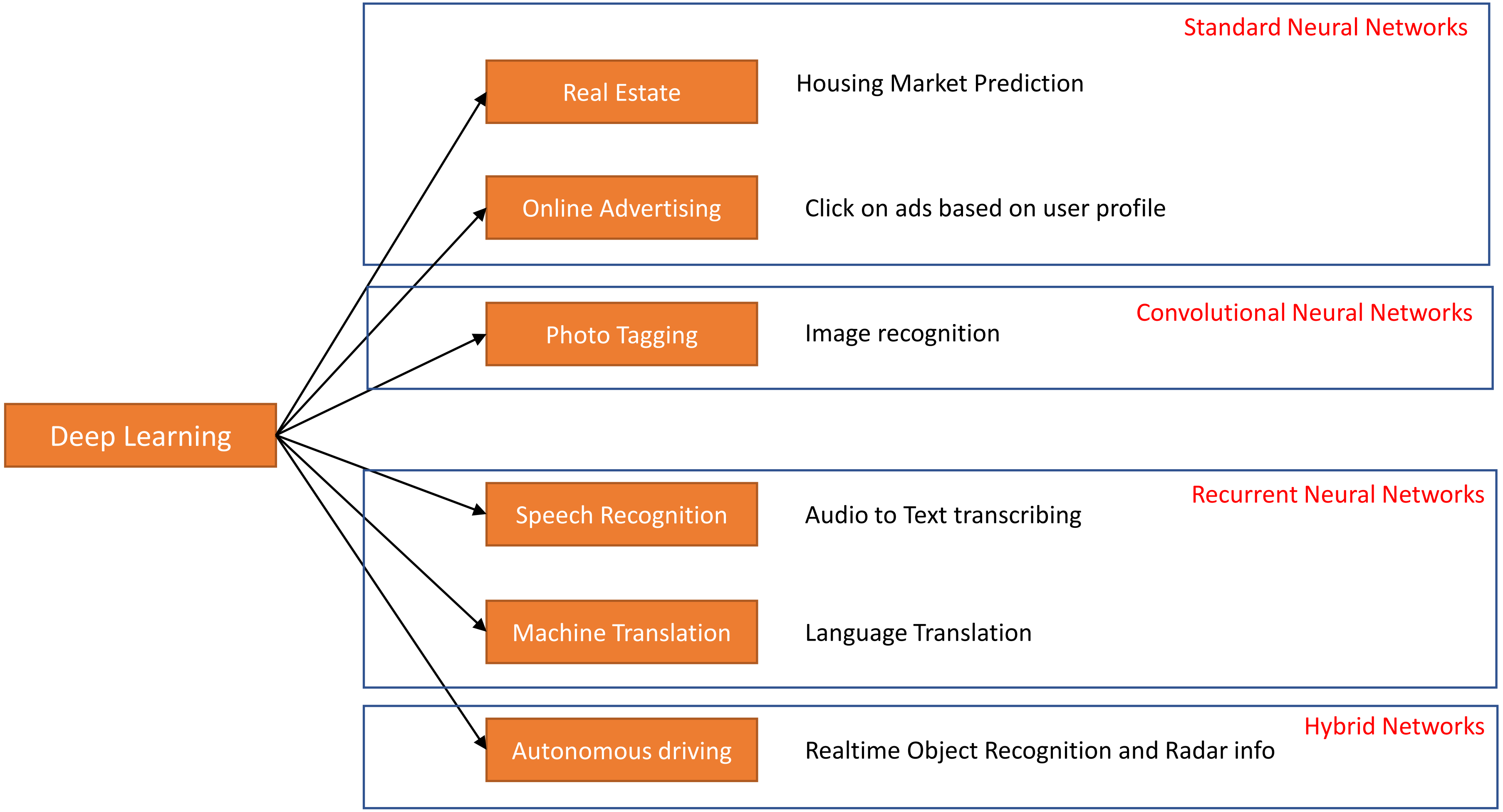


Please write and run your code using Google Collab

01

Introduction to Neural Networks

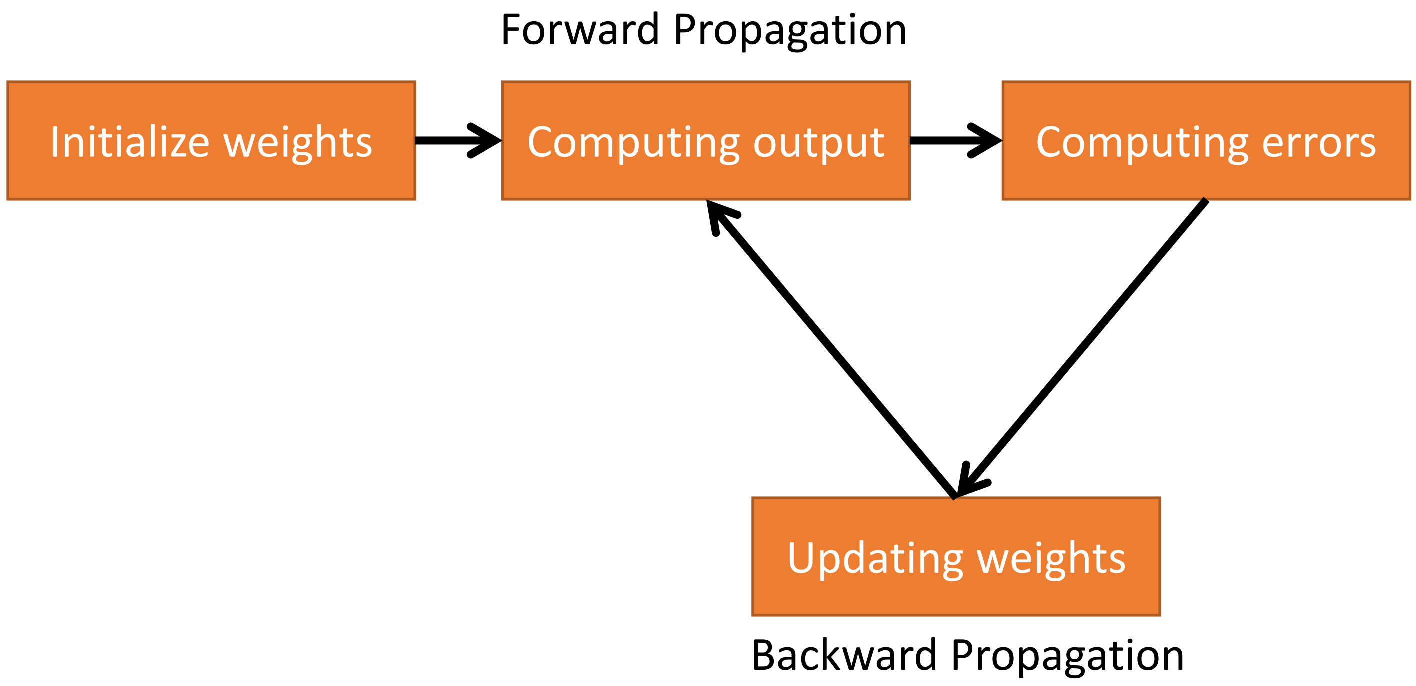
Use-cases

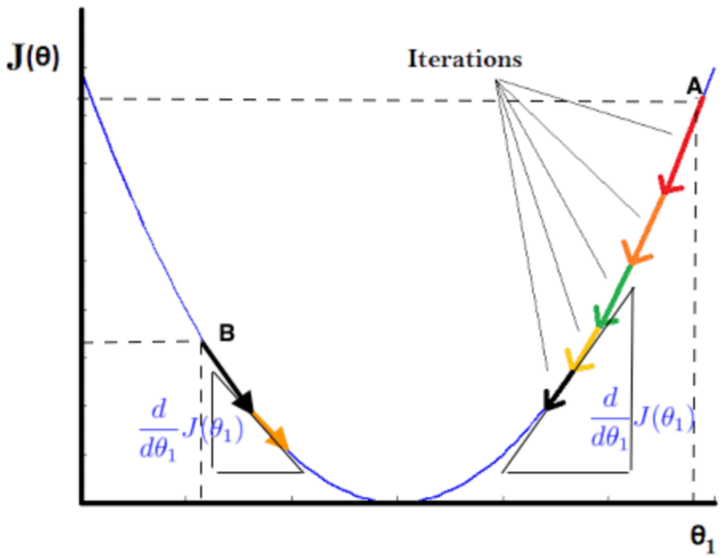


02

How Neural Networks Work?

Gradient Descent, Forward & Backward Propagation



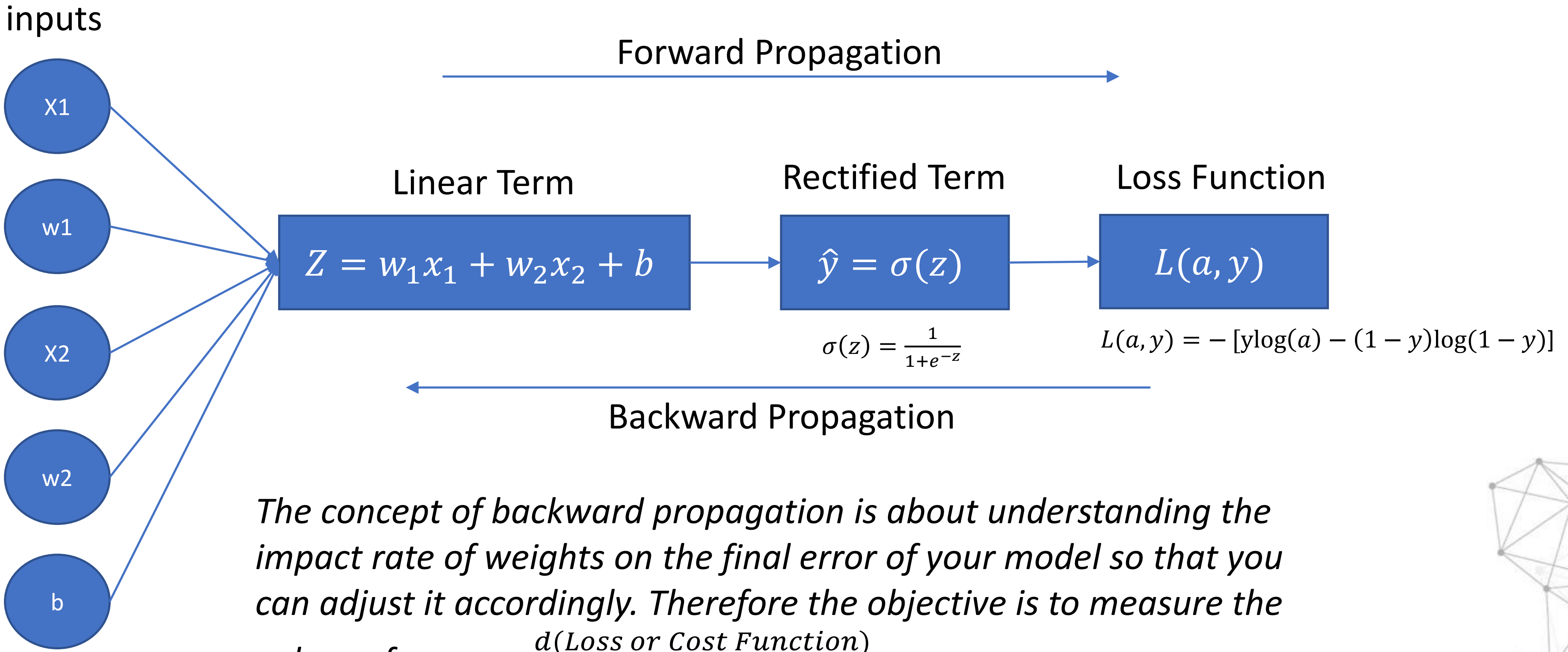


Gradient Descent:

Landing on the minimum of the error function through updating independent variables based on how drastic the error increase/decrease rate is. The drasticisty! of this change is governed by the slope of the curve (Gradient Nature) multiplied by a predefined factor (Learning Rate)

02 How Neural Networks Work?

A simple case of Logistic Regression Node



The concept of backward propagation is about understanding the impact rate of weights on the final error of your model so that you can adjust it accordingly. Therefore the objective is to measure the values of $\frac{d(Loss\ or\ Cost\ Function)}{d(Individual\ weights\ across\ your\ network)}$

In case of a simple Logistic Regression with a sigmoid activation function and two features (x1, x2) we can prove that:

dL/dw1 = dw1 = x1.dZ

dL/dw2 = dw2 = x2.dZ

dL/db = db = dZ

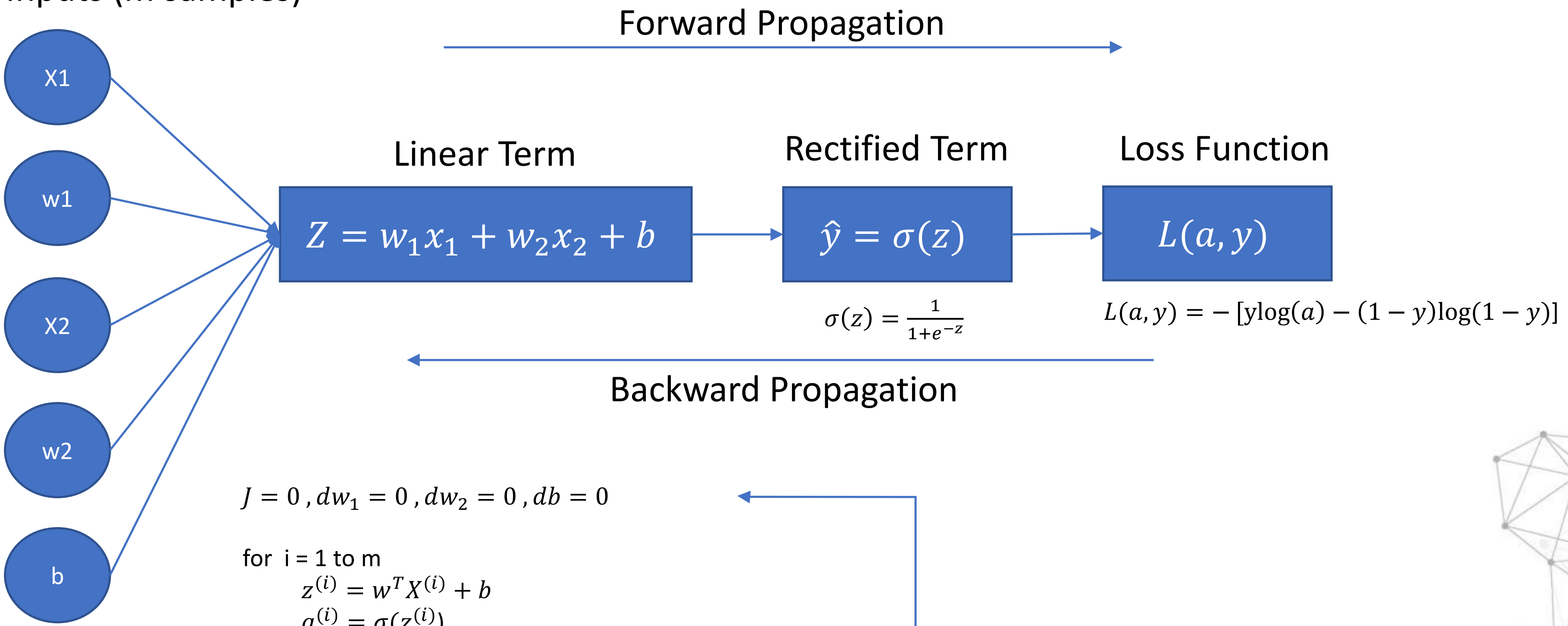
dL/dz = dz = a-y

02

How Neural Networks Work?

A simple case of Logistic Regression Node

Inputs (m samples)



$J = 0\,, dw_1 = 0\,, dw_2 = 0\,, db = 0$

for i = 1 to m

$z^{(i)} = w^T X^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J \mathrel{+}= -[y^{(i)} \log(a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 \mathrel{+}= X_1^{(i)} dz^{(i)}$

$dw_2 \mathrel{+}= X_2^{(i)} dz^{(i)}$

$db = dz^{(i)}$

$J = \frac{J}{m}, dw_1 = \frac{dw_1}{m}\,, dw_2 = \frac{dw_2}{m}\,, db = \frac{db}{m}$

Repeat until the errors are minimized

03

Types of Neural Networks

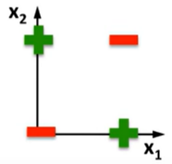
Perceptrons and FCN's, RNN's, CNN's

Circuit Theory and Deep Learning:

There are functions you can compute with a small L-layer deep network that shallow networks require exponentially more hidden units to compute. The best example for this is an *XOR* function where:

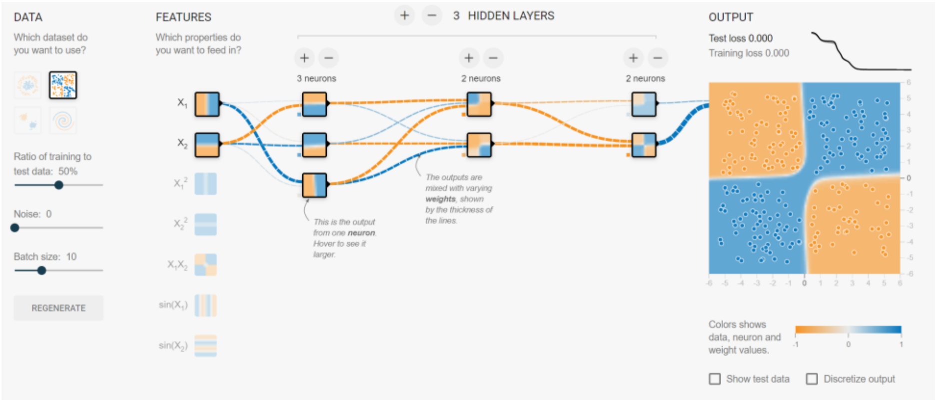
$$y = X_1 XOR\ X_2 XOR\ \ldots XOR\ X_n$$

Computing the above function requires e^n nodes in a single but can be computed in $\log(n)$ layers. This significantly reduces the computation power required to perform Neural Network Modeling.

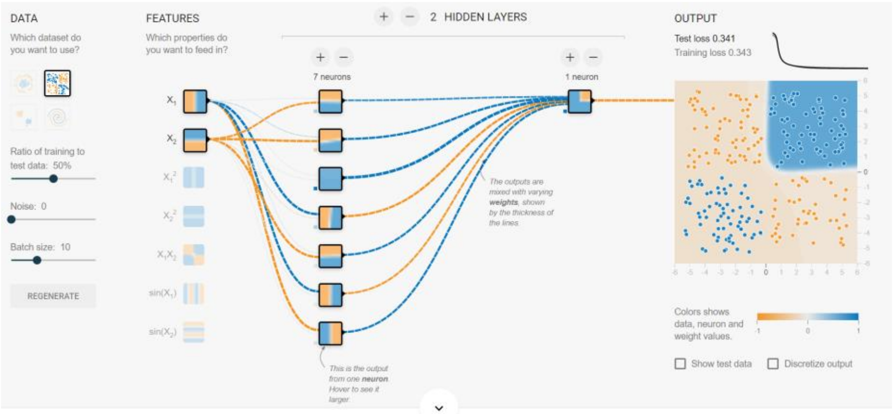


XOR Classification

Successful Multilayer Classification



Failed Single Layer Classification



https://playground.tensorflow.org/

Circuit Theory and Deep Learning:

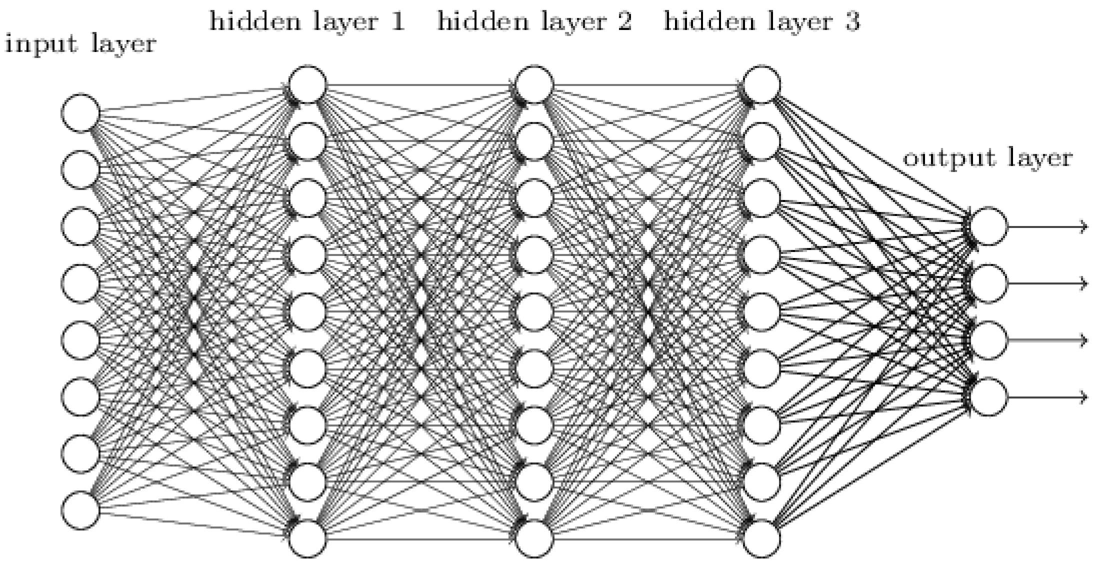
There are functions you can compute with a small L-layer deep network that shallow networks require exponentially more hidden units to compute. The best example for this is an *XOR* function where:

$$y = X_1 XOR\ X_2 XOR\ \ldots XOR\ X_n$$

Computing the above function requires e^n nodes in a single but can be computed in $\log(n)$ layers. This significantly reduces the computation power required to perform Neural Network Modeling.

Fully Connected Networks:

Fully Connected Networks are networks where all of your input features from any nodes is connected to all of the nodes connected to the subsequent layer.



03

Types of Neural Networks

Perceptrons and FCN’s, RNN’s, CNN’s

Circuit Theory and Deep Learning:

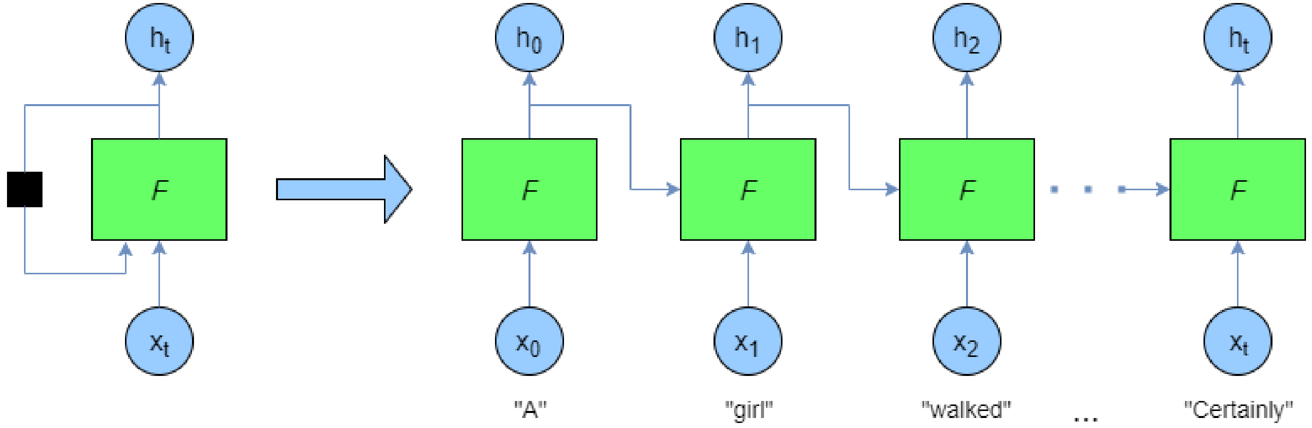
There are functions you can compute with a small L-layer deep network that shallow networks require exponentially more hidden units to compute. The best example for this is an *XOR* function where:

$$y = X_1 XOR\ X_2 XOR\ \ldots XOR\ X_n$$

Computing the above function requires e^n nodes in a single but can be computed in $\log(n)$ layers. This significantly reduces the computation power required to perform Neural Network Modeling.

Recurrent Neural Networks (RNN’s):

A **recurrent neural network** (RNN) is a class of artificial **neural networks** where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Unlike feedforward **neural networks**, RNNs can use their internal state (memory) to process sequences of inputs. A wide application of RNN’s are at Time Series Classification as well as Natural Language Processing (NLP)



03

Types of Neural Networks

Perceptron and FCN's, RNN's, CNN's

Circuit Theory and Deep Learning:

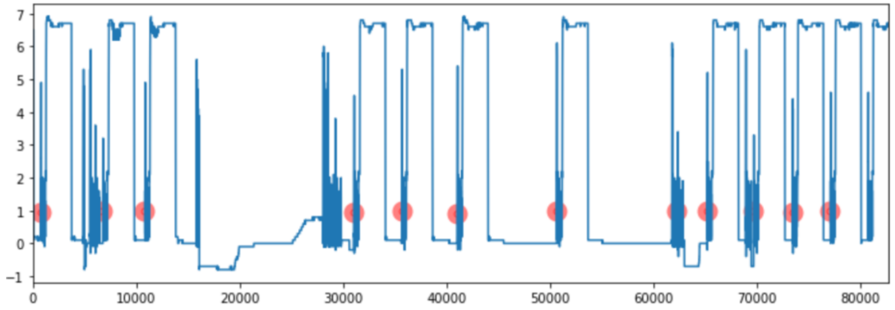
There are functions you can compute with a small L-layer deep network that shallow networks require exponentially more hidden units to compute. The best example for this is an *XOR* function where:

$$y = X_1 XOR\ X_2 XOR\ \ldots XOR\ X_n$$

Computing the above function requires e^n nodes in a single but can be computed in $\log(n)$ layers. This significantly reduces the computation power required to perform Neural Network Modeling.

Recurrent Neural Networks (RNN's):

A **recurrent neural network** (RNN) is a class of artificial **neural networks** where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Unlike feedforward **neural networks**, RNNs can use their internal state (memory) to process sequences of inputs. A wide application of RNN's are at Time Series Classification as well as Natural Language Processing (NLP)



Mold Time Prediction (LSTM) Model

03

Types of Neural Networks

Perceptrons and FCN's, RNN's, CNN's

Circuit Theory and Deep Learning:

There are functions you can compute with a small L-layer deep network that shallow networks require exponentially more hidden units to compute. The best example for this is an *XOR* function where:

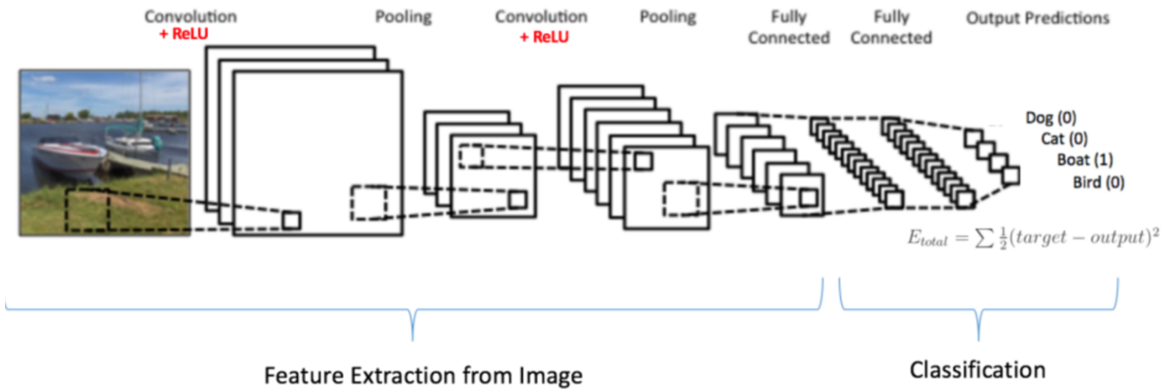
$$y = X_1 XOR\ X_2 XOR\ \ \ldots XOR\ X_n$$

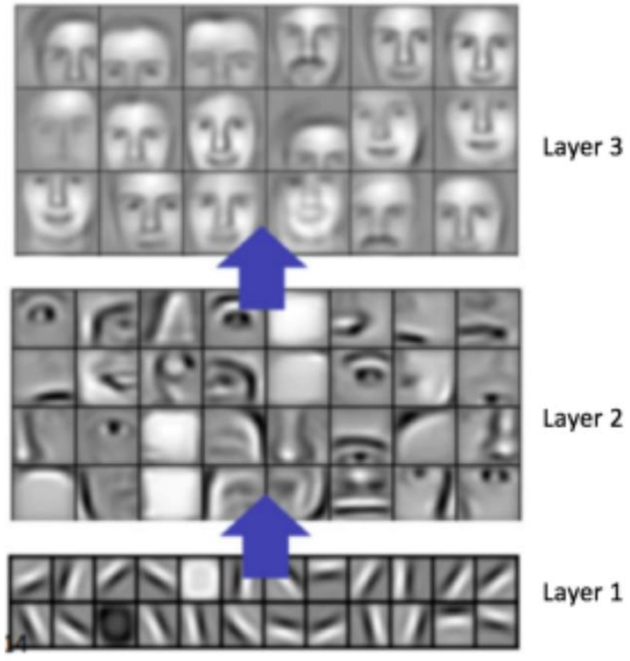
Computing the above function requires e^n nodes in a single but can be computed in $\log(n)$ layers. This significantly reduces the computation power required to perform Neural Network Modeling.

Convolutional Neural Networks (CNN's):

In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. It involves:

1. Convolutions: To extract features by applying kernels(filters) across the image
2. Non-Linearity (ReLU): Like any Neural Network
3. Pooling and Sub-Sampling: To reduce vector size
4. Classification: Output Layer





03

Types of Neural Networks

Perceptrons and FCN's, RNN's, CNN's

Circuit Theory and Deep Learning:

There are functions you can compute with a small L-layer deep network that shallow networks require exponentially more hidden units to compute. The best example for this is an *XOR* function where:

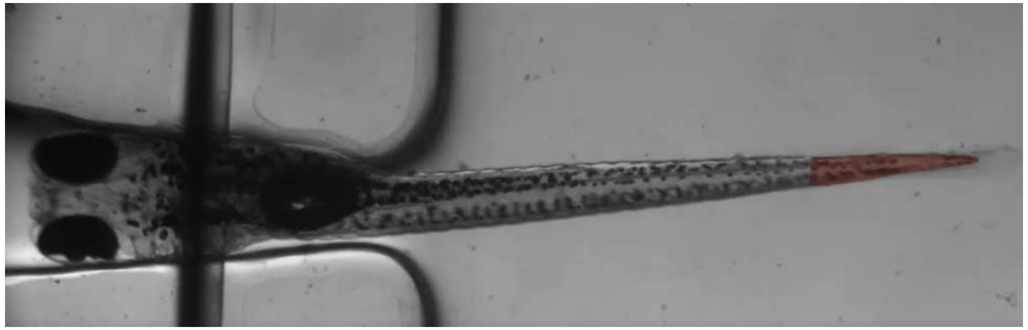
$$y = X_1 XOR\ X_2 XOR\ \dots XOR\ X_n$$

Computing the above function requires e^n nodes in a single but can be computed in $\log(n)$ layers. This significantly reduces the computation power required to perform Neural Network Modeling.

Convolutional Neural Networks (CNN's):

In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. It involves:

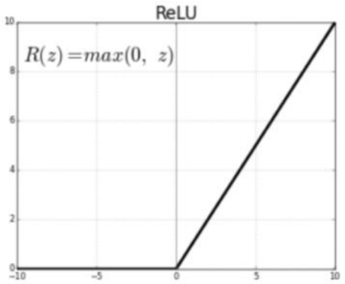
1. Convolutions: To extract features by applying kernels(filters) across the image
2. Non-Linearity (ReLU): Like any Neural Network
3. Pooling and Sub-Sampling: To reduce vector size
4. Classification: Output Layer

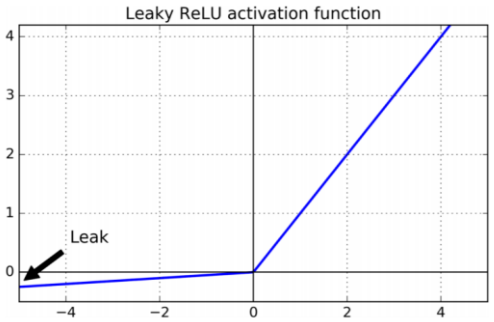


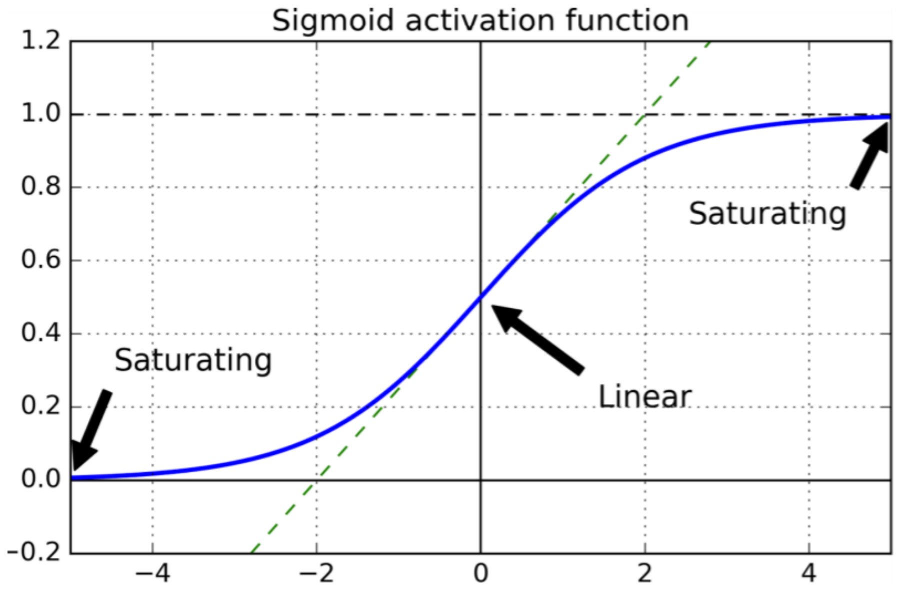
Recognition of Zebra fish tail movement behaviour in Microfluidic Channel

Deep Learning Challenges:

- **Vanishing Gradient Problems:** Gradients often get smaller and smaller as the algorithm progresses down to the lower layers.
- **Slow Computation:** For large networks, training can be extremely slow
- **Overfitting:** A Model with millions of parameters would severely risk overfitting the training set.







Safeguards:

- **Batch Normalization:** This method consists of adding normalization operation before the activation step at each node of the layer. This way, distribution of each layer's activation input will be centered at zero and will be spread within 3 standard deviation. This will reduce the risk of co-adaptation during training

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(300, kernel_initializer="he_normal", use_bias=False),
    keras.layers.BatchNormalization(),
    keras.layers.Activation("elu"),
    keras.layers.Dense(100, kernel_initializer="he_normal", use_bias=False),
    keras.layers.BatchNormalization(),
    keras.layers.Activation("elu"),
    keras.layers.Dense(10, activation="softmax")
])
```

- **Gradient Clipping:** A popular technique to lessen the exploding gradients problem is to simply clip the gradients so that they never exceed some threshold.

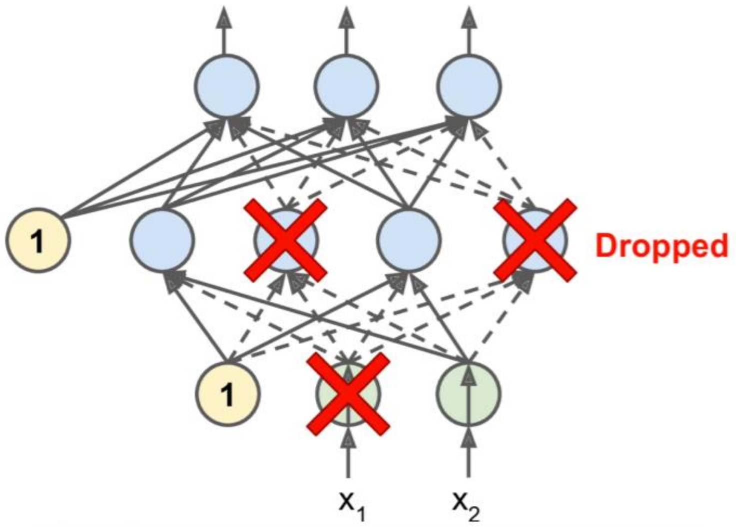
```
optimizer = keras.optimizers.SGD(clipvalue=1.0)
model.compile(loss="mse", optimizer=optimizer)
```

04

Training Challenges and Strategies

Safeguards against Gradient Vanishing/Exploding

Safeguards:



- **Dropout:** At each iteration, the model will drop some nodes based on a probability (Drop out rate ~ 50%). This is also to avoid co-adaptation behaviour of neurons.

- **Early Stopping:** Stop training when performance on validation set starts to drop

- **L1/L2 Regularization:** add a term to the loss that penalizes the L1 or L2 norm of the weights

- **Pre-trained Models:** Transfer Learning is extremely popular particularly in image segmentation architectures due to commonalities between early layers