University of Guilan

# Computational Intelligence

## Subject2: Artificial Neural Networks

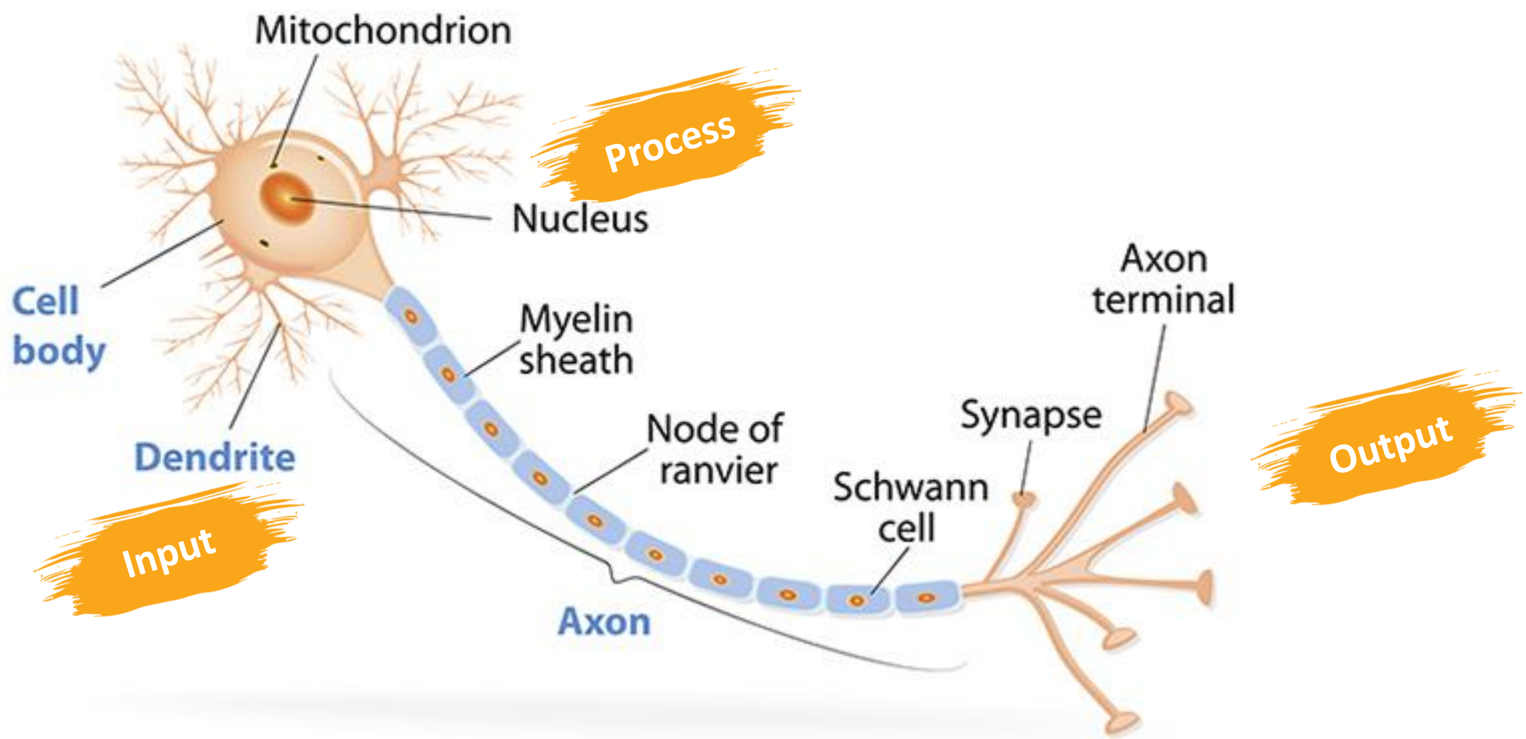

Instructor: Ali Tourani

A.Tourani1991@gmail.Com

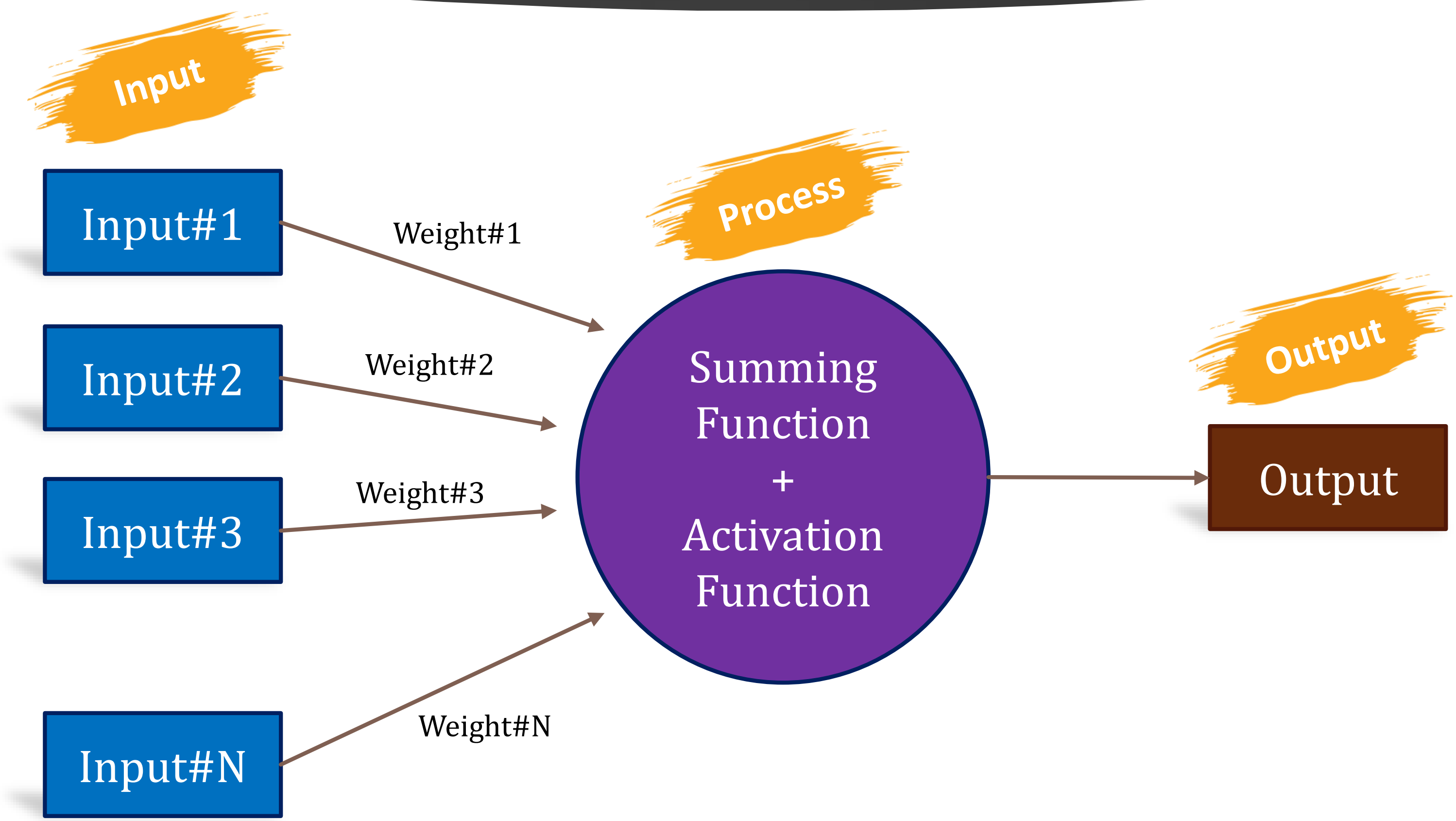
# Agenda

- Intro
- How do they work?
- Activation Functions
- Applications of ANNs

# Intro


Mitochondrion
Process
Nucleus
Axon
terminal
Cell
body
Myelin
sheath
Node of
ranvier
Synapse
Output
Dendrite
Schwann
cell
Input
Axon

# Intro



Input
Input#1
Input#2
Input#3
Input#N
Weight#1
Weight#2
Weight#3
Weight#N
Process
Summing
Function
+
Activation
Function
Output
Output

# Intro

University of Guilan

## Natural vs. Artificial Neural Networks

| NNNs | ANNs |
|---|---|
| Dendrites | Links |
| The strength of the signal | Weights |
| Accumulated signals in the Soma | Sum of the weighted inputs |
| Signal transmission from an Axon | Activation Function (AF) |

# Intro

University of Guilan

**What makes humans to be so efficient at learning things?**

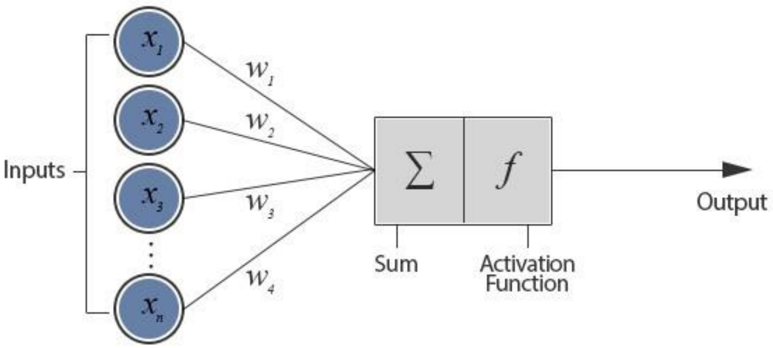
- The great structure and innate capabilities of the human brain
- The dream of creating thinking machines
- Emulate the learning behavior of organic brains
- **Result**: the advent of Artificial Neural Networks (ANNs)

# Intro

University of Guilan

**Main specifications of ANNs**

- Using neuron-like elements
  - Neurons do the processing of data
  - Interconnected like a web
- Sending signals through the links
  - Each link has a weight
- Using a summation function
- Using an Activation Function



# Intro

**Main specifications of ANNs**

- Learning process in ANNs:
  1. Extracting statistical regularities from the input data
  2. Encoding the structure into the parameters of the network
- Self-learning capabilities
  - Producing better results as more data becomes available

# Intro

University of Guilan

**Artificial Neuron (AN)**

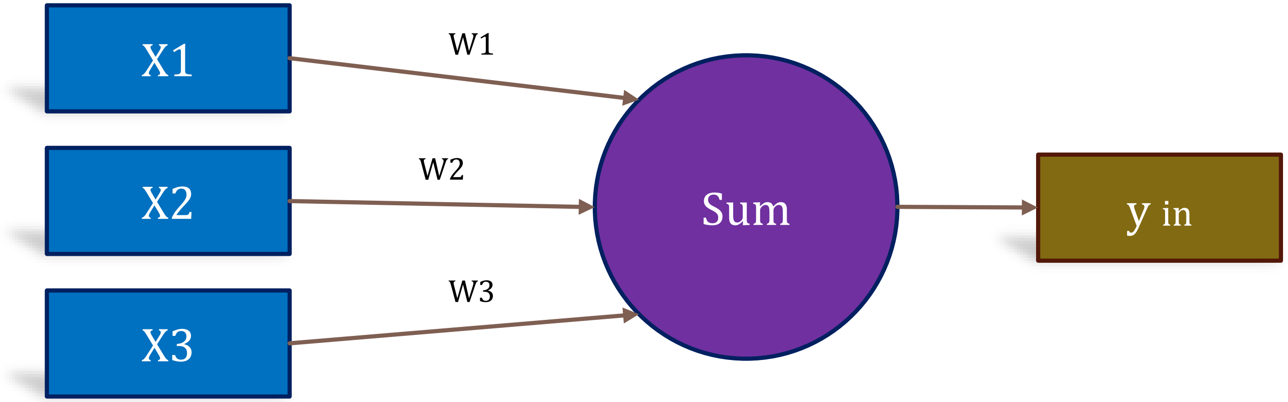
- Implements a non-linear mapping from to [0,1] or [-1,1], depending on the Activation Function

$$f_{AN}: R^I \rightarrow [0,1] \qquad f_{AN}: R^I \rightarrow [-1,1]$$

  - Where $I$ is the number of inputs
- Each input signal is associated to a weight
- The AN computes the **Net** (summation), and then, passes it through the **Activation Function** to compute the output signal

# Intro

## Artificial Neuron (AN)




$$y_{in} = \sum_{i=1}^{3} x_i w_i$$

# Intro

University of Guilan

**Artificial Neuron (AN)**

- How to calculate the Net Input Signal ($y_{in}$)?
- Method 1 - Summation Units:
  - More common!

$$net = \sum_{i=1}^{I} x_i w_i$$

- Method 2 – Product Units:
  - Allows higher-order combinations
  - Increased information capacity

$$net = \sum_{i=1}^{I} {x_i}^{w_i}$$

# Intro

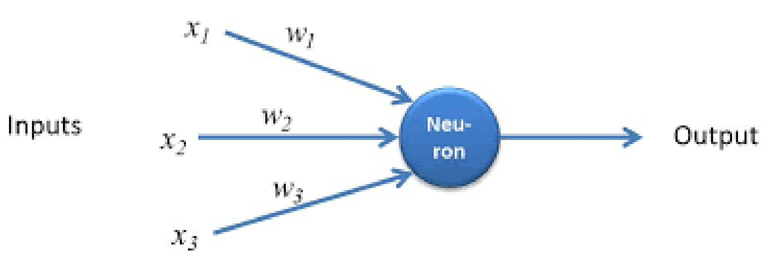
**In an Artificial Neuron with three inputs:**

- Input signals are $x_1, x_2, x_3$
- Weights are $w_1, w_2, w_3$
- The value of summation is equal to

$$y_{in} = net = \sum_{i=1}^{3} x_i w_i$$

- The final output after processing in Activation Function is:

$$output = f(y_{in})$$



# Intro

University of Guilan

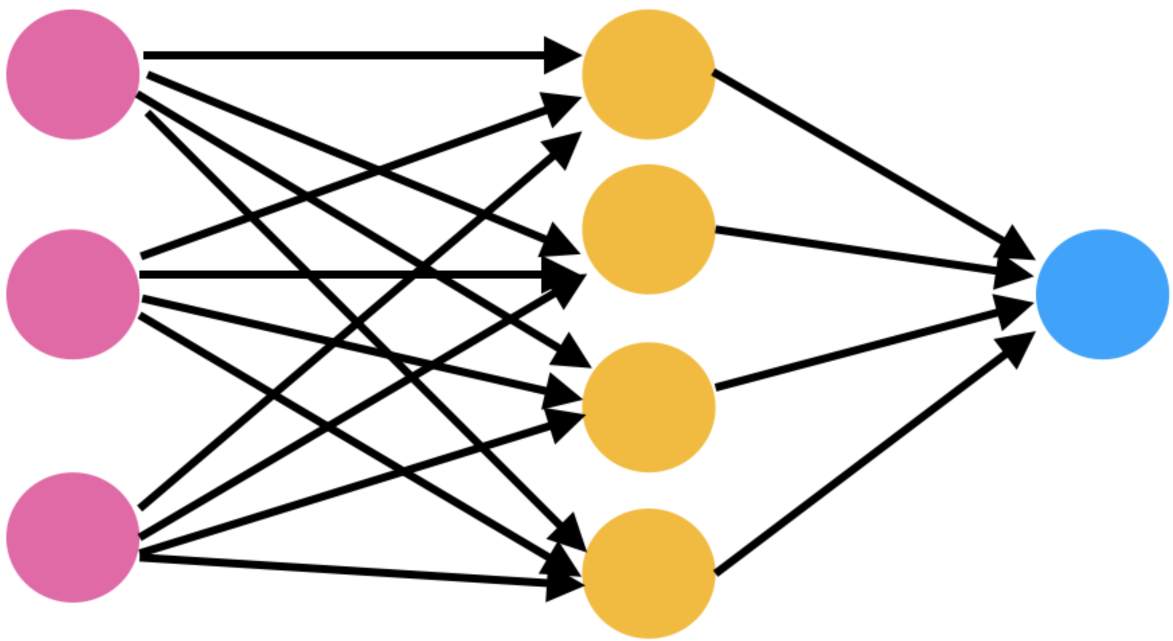
**The functionality of an Artificial Neuron:**

- The strength of each input signal depends on its weight
  - The weight $w_1 = 0.2$ weakens the input signal
  - The weight $w_1 = 5$ empowers the input signal
  - The weight $w_1 = 0$ makes the input signal ineffective
- In some cases, we might need a threshold θ in the neuron

# Intro

University of Guilan

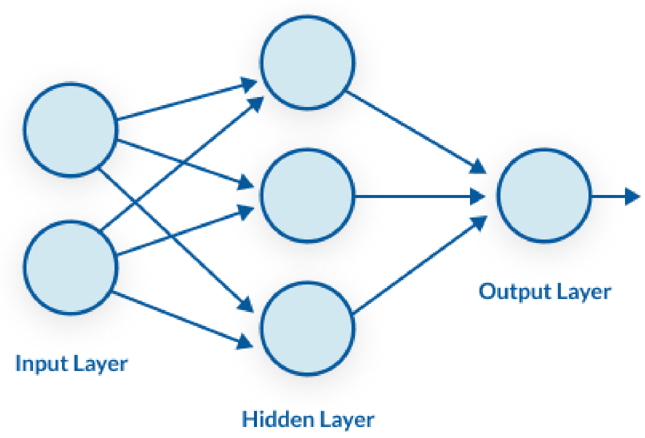
**An Artificial Neural Network (ANN):**



# Intro

**An Artificial Neural Network (ANN):**

- A supervised learning system built of a large number of neurons
  - Organized in interconnected layers
- Each neuron can make simple decisions
  - Decisions are fed to other neurons



# How do they work?

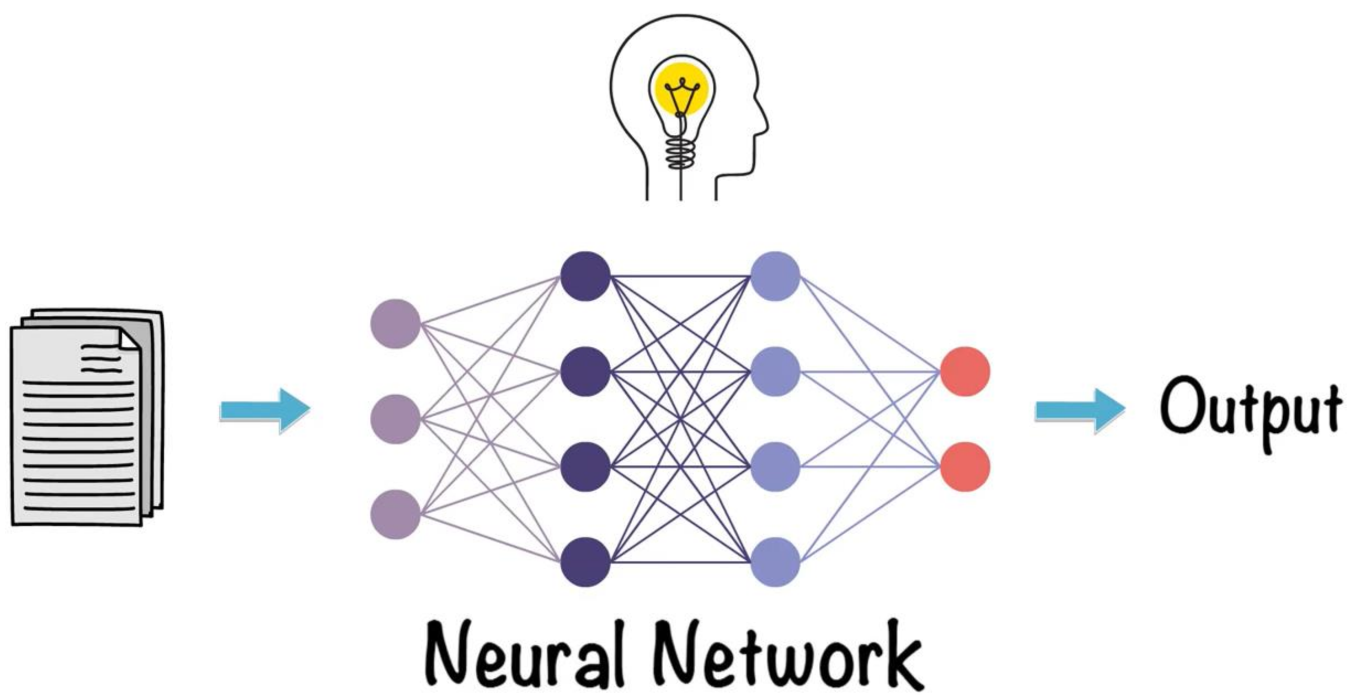
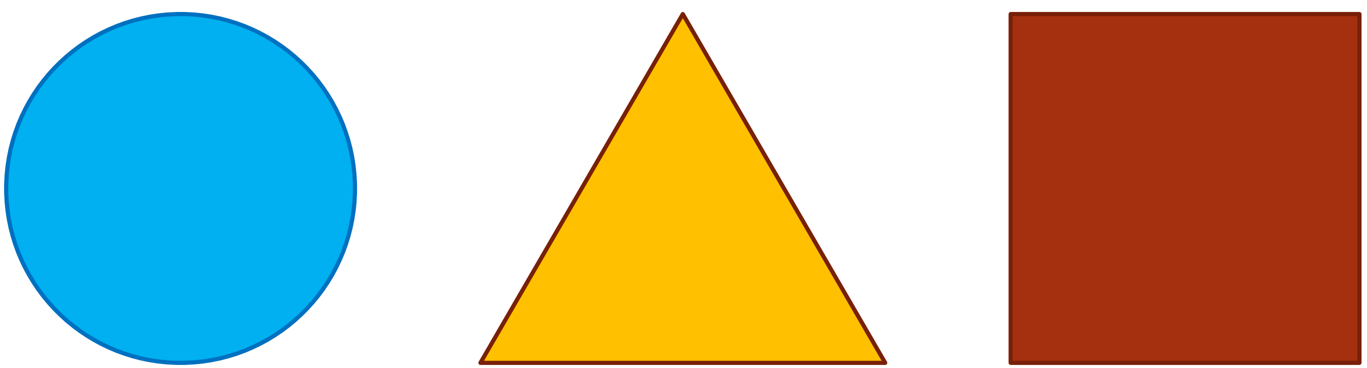



Image from *Simplilearn*

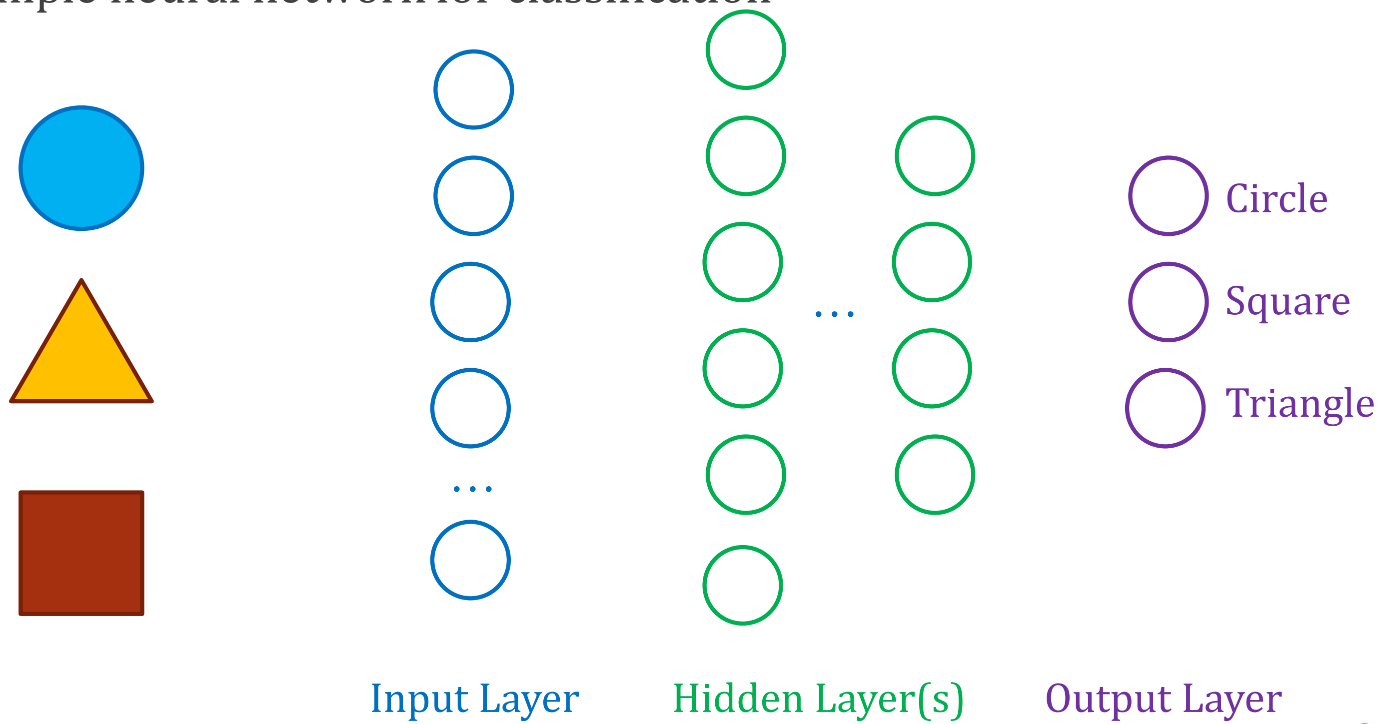
# How do they work?

- Imagine we want to classify different objects using ANNs
  - Inputs: Rectangle, Circle, Triangle
  - Goal: Correct classification



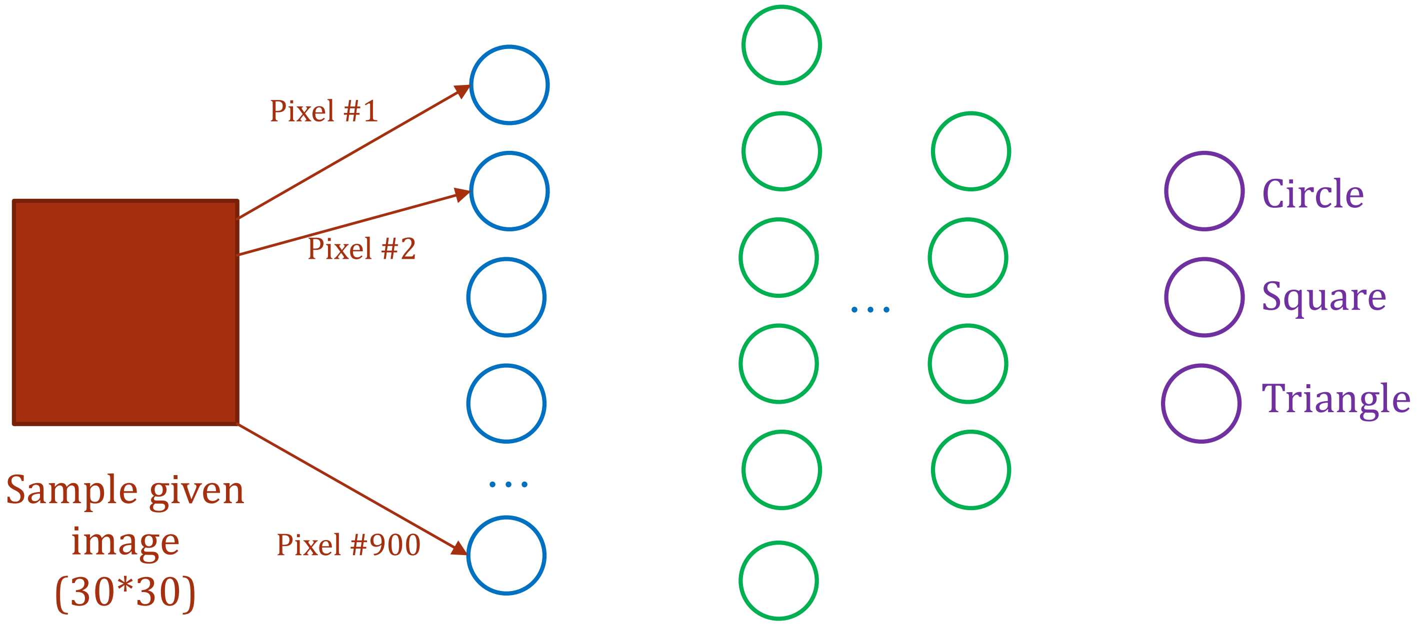
# How do they work?

- A simple neural network for classification



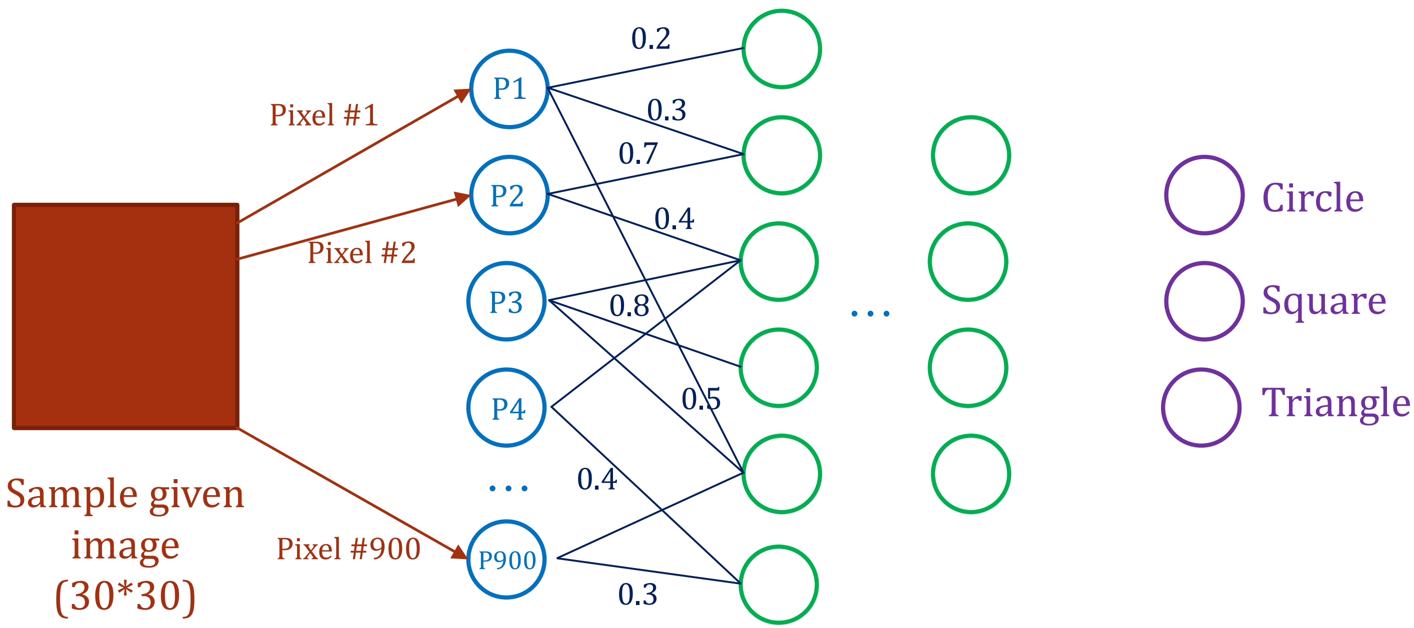
# How do they work?

- Each pixel should be mapped to a single neuron from the input layer



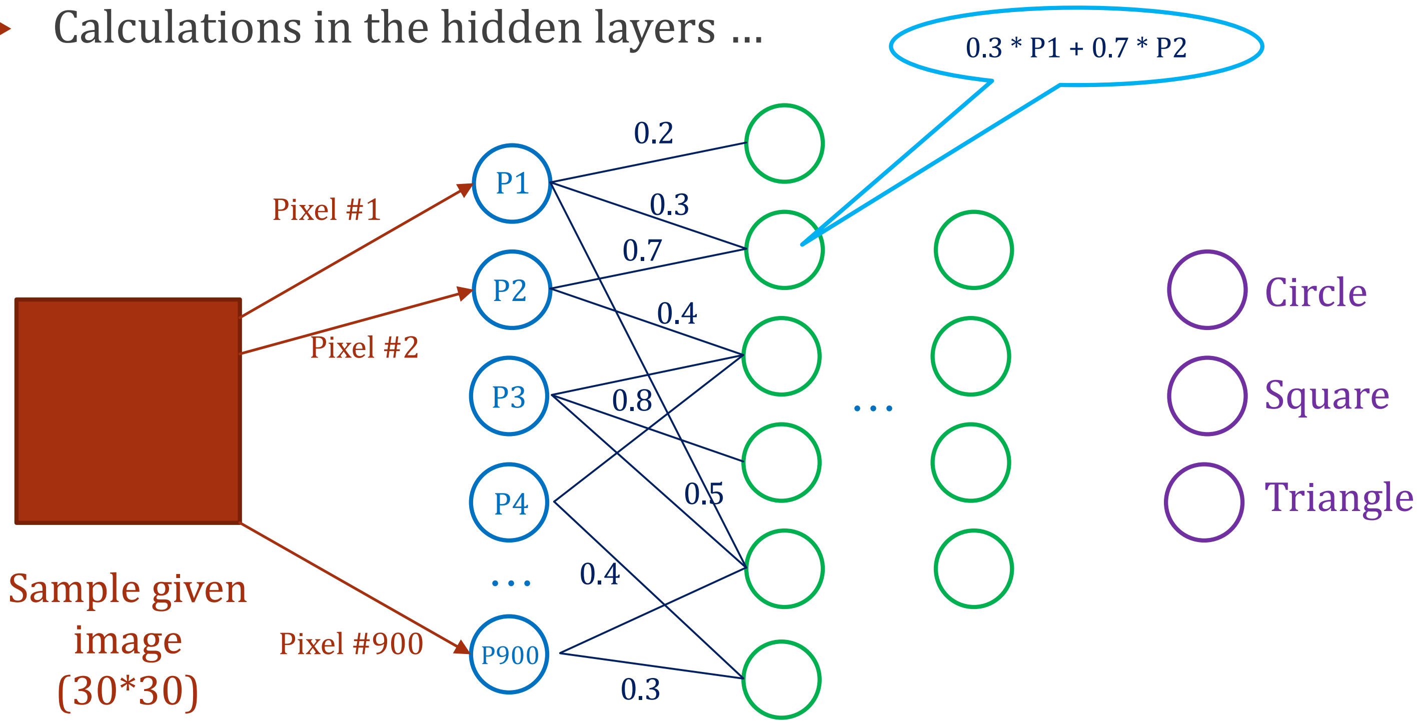
# How do they work?

- Calculations in the hidden layers …



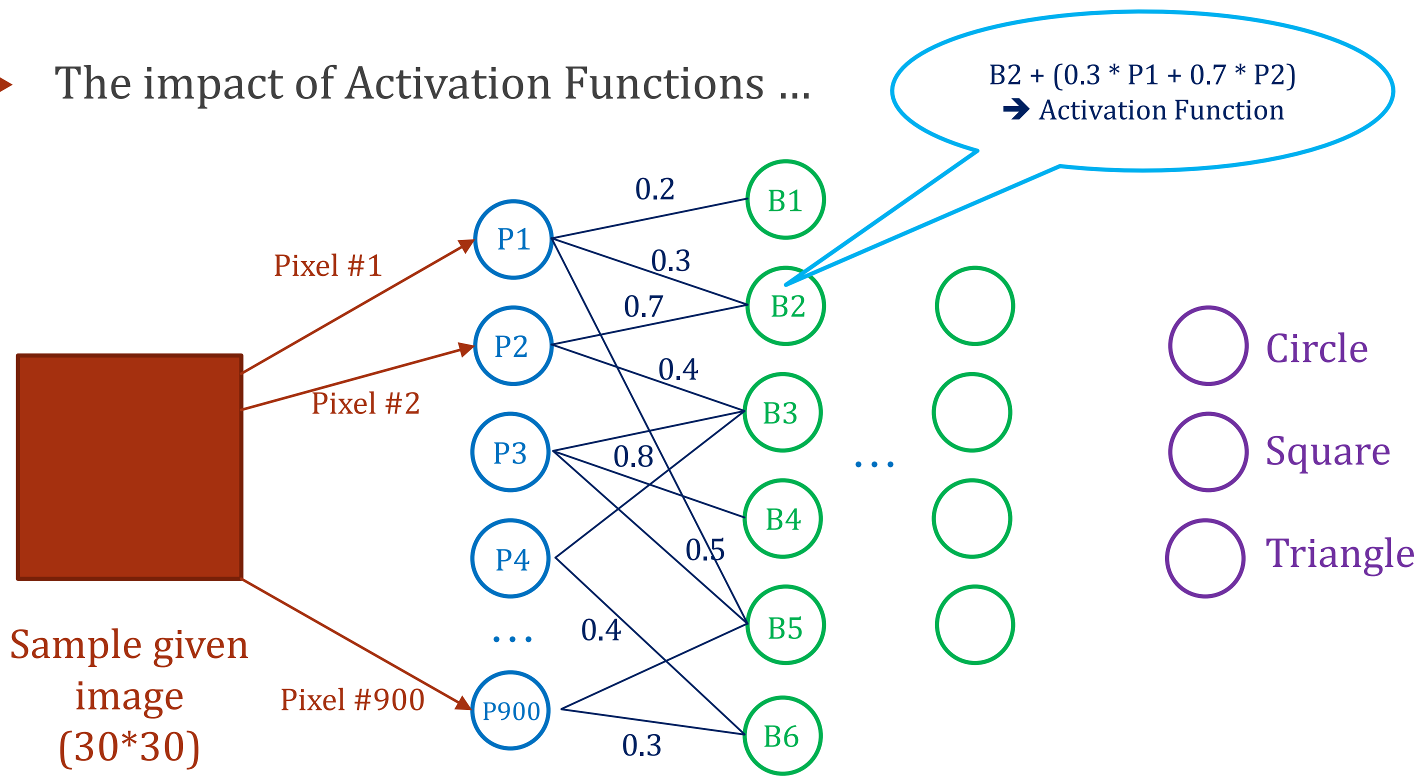
# How do they work?

- Calculations in the hidden layers …



# How do they work?

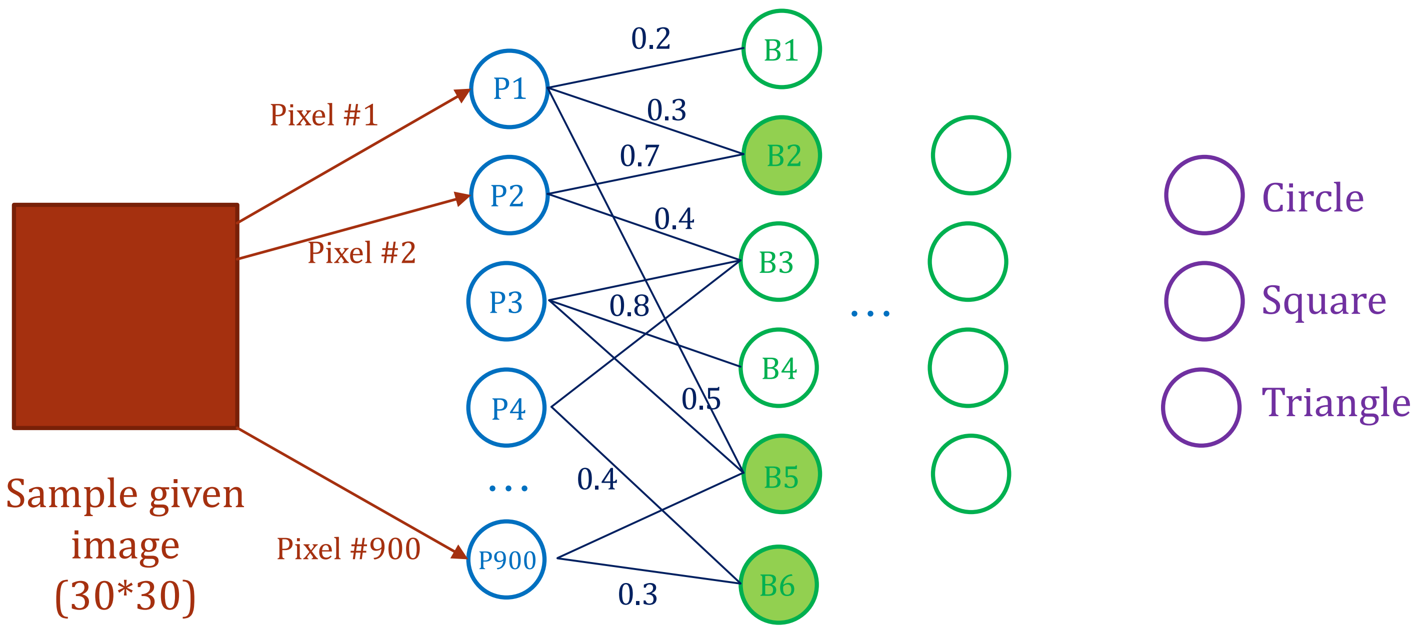
- The impact of Activation Functions …



University of Guilan

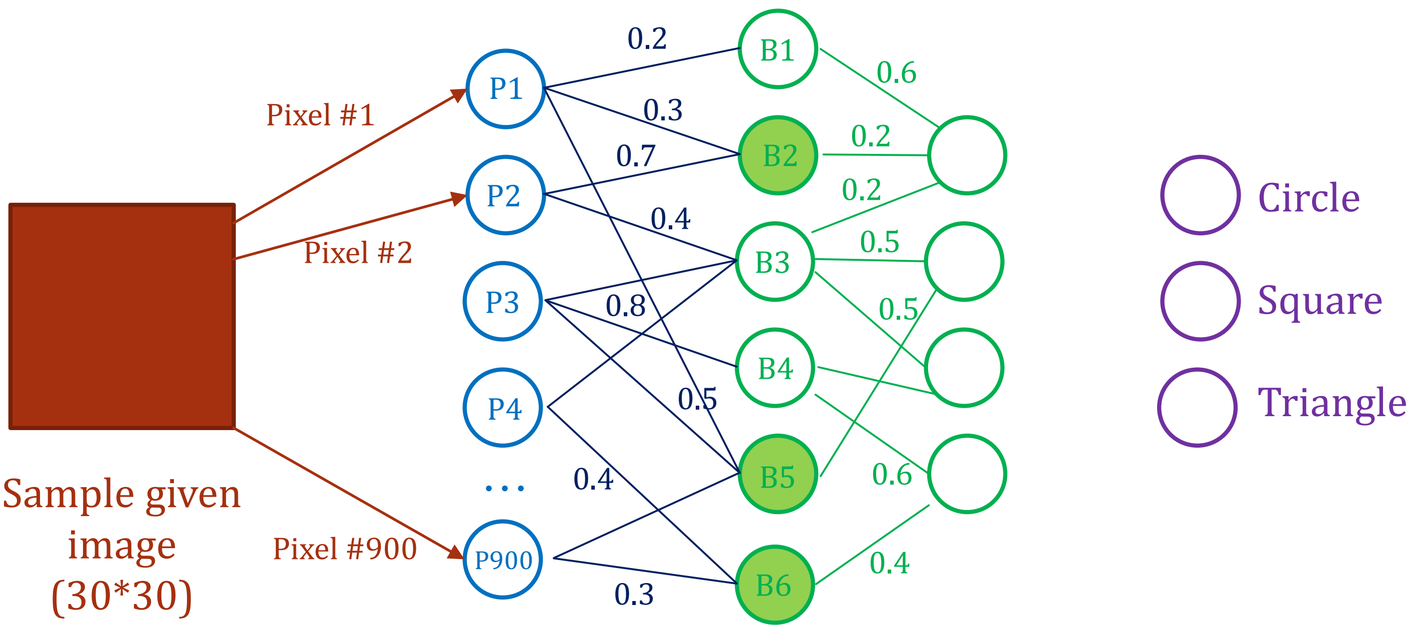
# How do they work?

- The impact of Activation Functions (Active nodes have green backgrounds) …



# How do they work?

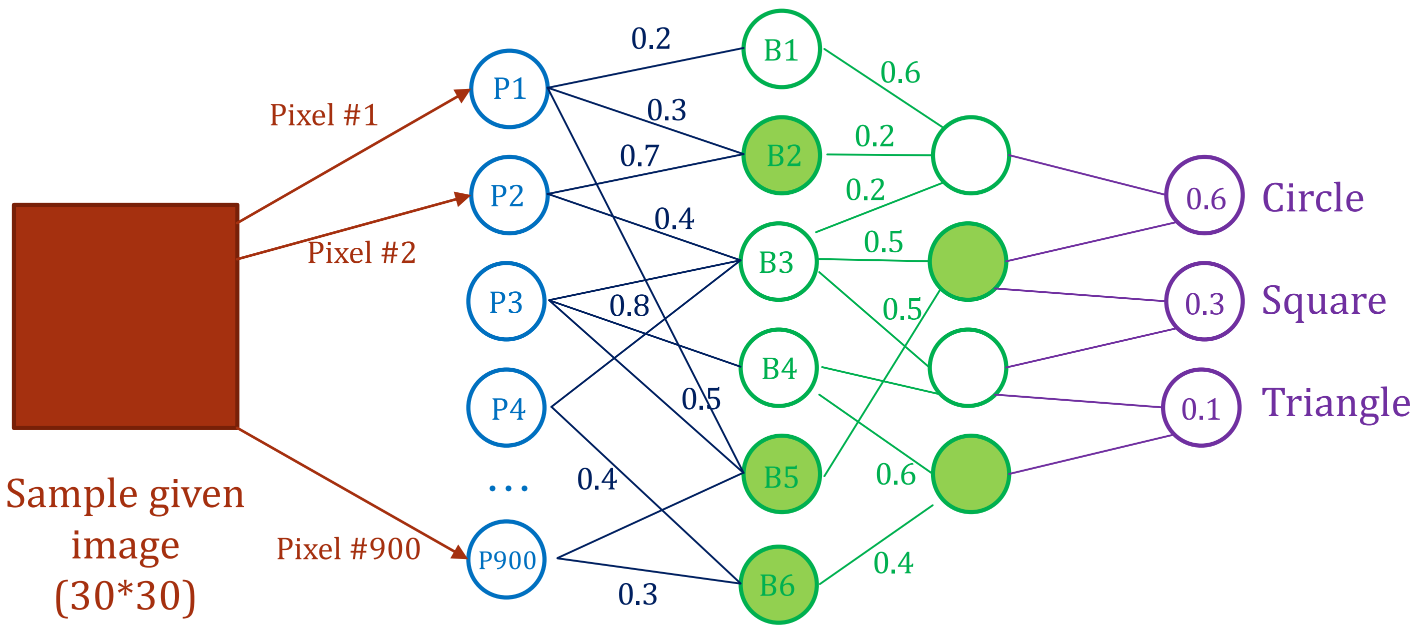
- Forward propagation ...



# How do they work?

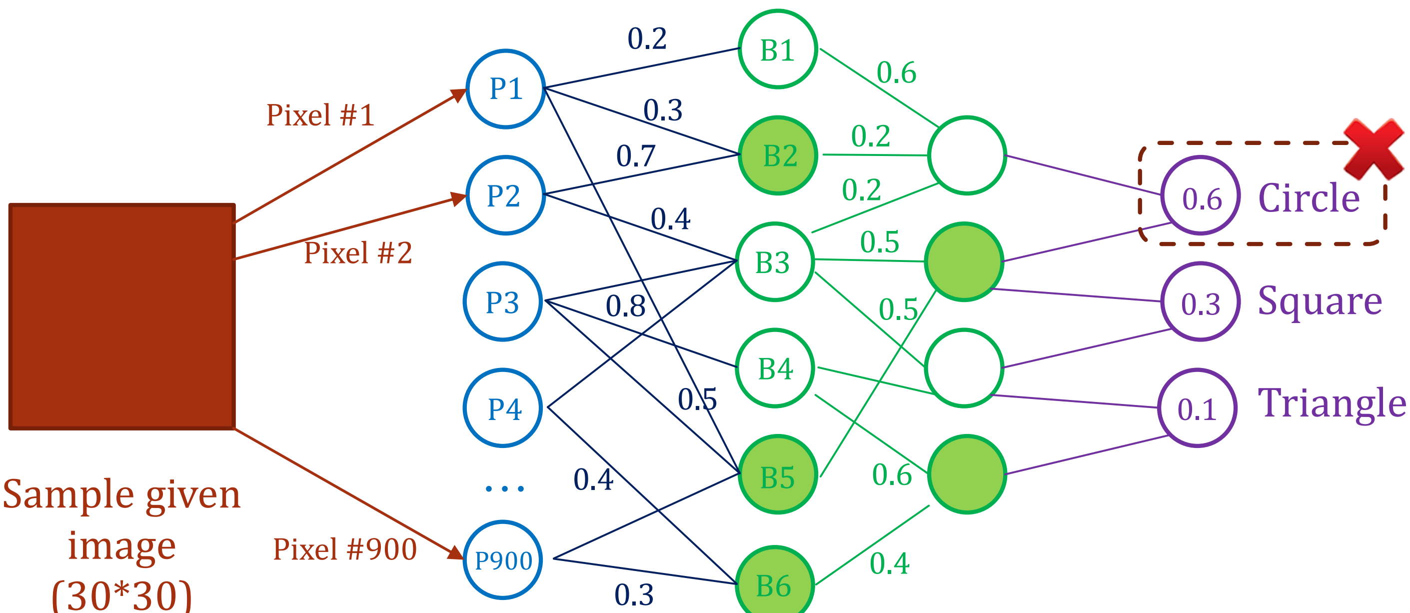
University of Guilan

- Forward propagation …



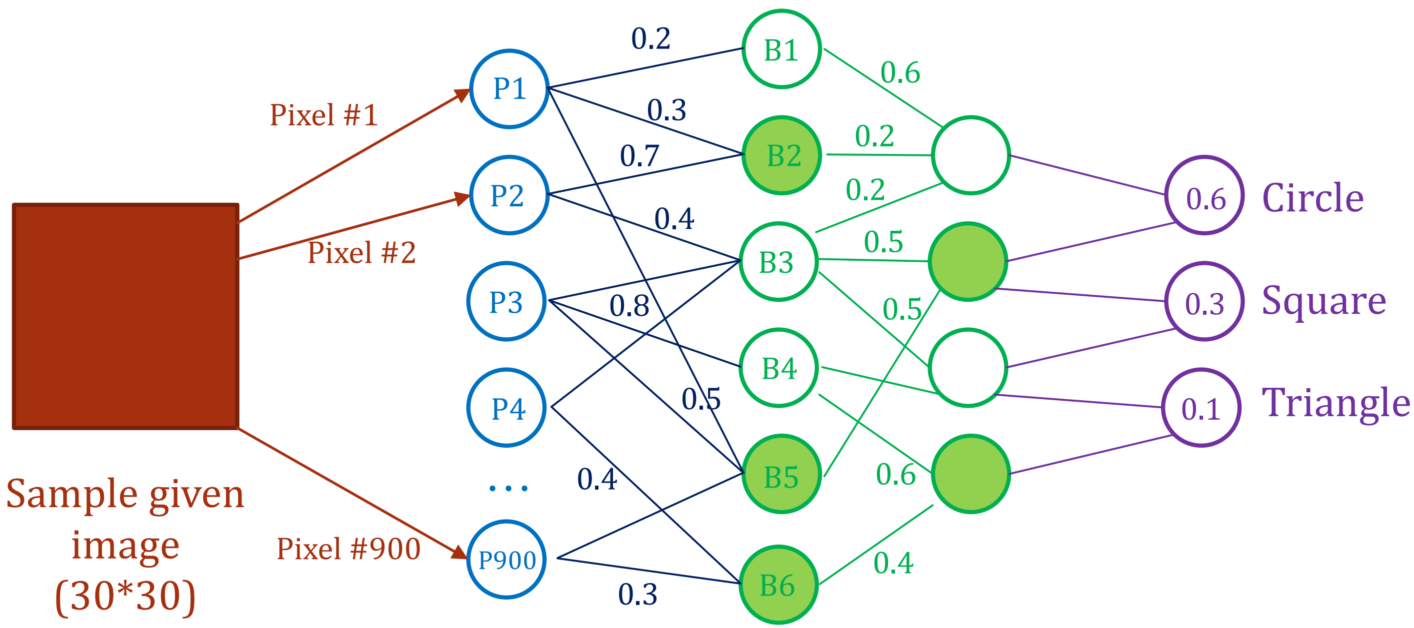
# How do they work?

- Wrong prediction! (The highest probability is circle with 0.6)



# How do they work?

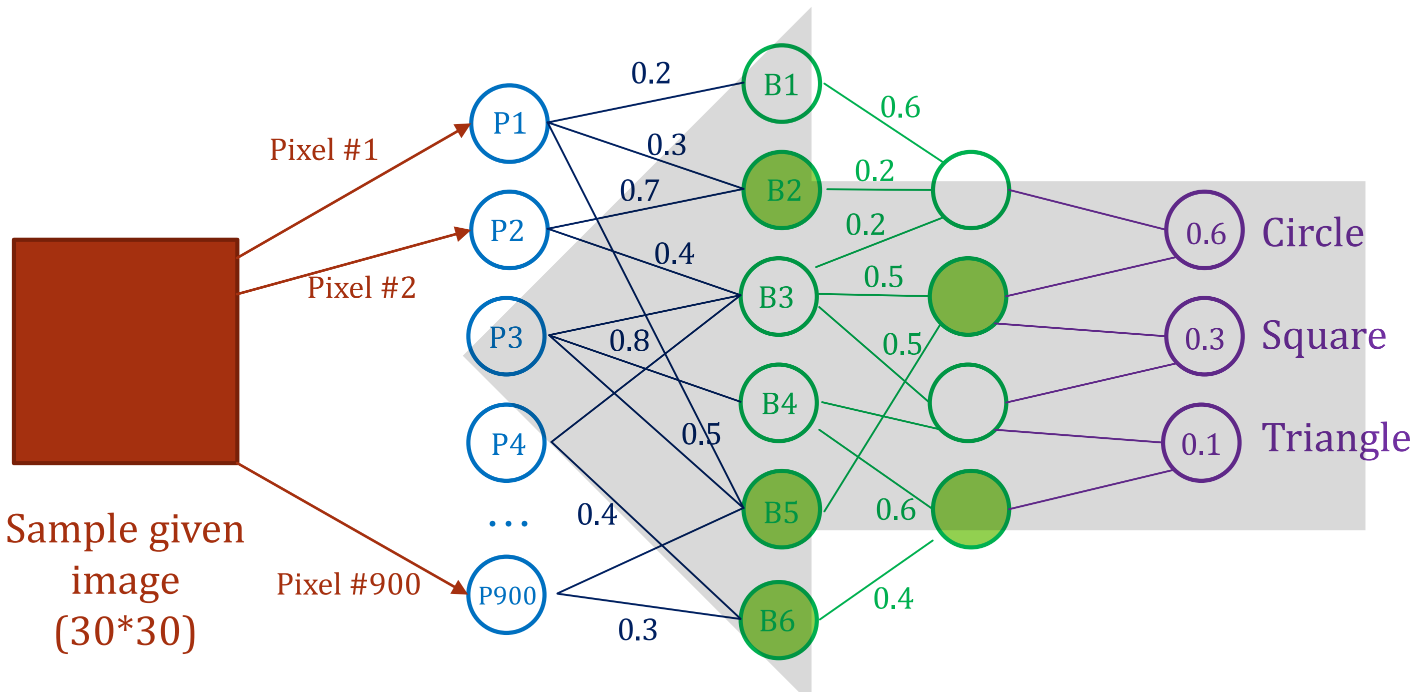
- How to know? Comparing to the Actual (expected) Outputs




| Actual | Error |
|---|---|
| 0 | -0.6 |
| 1 | 0.7 |
| 0 | -0.1 |

# How do they work?

- How to resolve? Backpropagation …




| Actual | Error |
|---|---|
| 0 | -0.6 |
| 1 | 0.7 |
| 0 | -0.1 |

# Configurations

**Artificial Neuron Network (ANN) - Configurations**

- Architecture
  - Order of neurons in different layers with different connections
  - Shallow 3-layered architecture: input-hidden-output
  - Deep architecture: many hidden layers
- Training process
  - How to set weights so that the learning process results in correct prediction

# Configurations

University of Guilan

**Artificial Neuron Network (ANN) - Configurations**

- How many hidden layers do we need?
  - At least one! The more, the deeper architecture
- How many nodes (neurons) in each hidden layer?
  - No exact science or formula
  1. *Midway between the number of inputs and outputs*
  2. *2/3 the input nodes + output nodes*
  3. *Less than 2x the inputs*
- What type of activation function do we need?

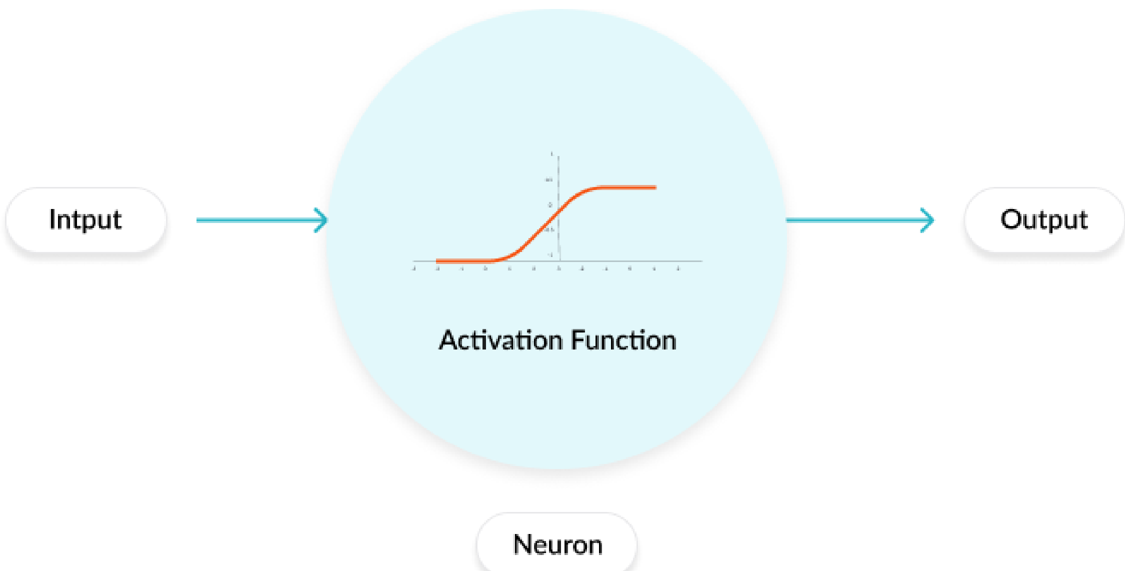
# Configurations

**Artificial Neuron Network (ANN) - Configurations**

- How should I choose the learning rate?
- How many iterations can solve the problem?
  - How many times it goes over data in the dataset
  - Maybe 1000, 2000, 10000+
- How do I choose the desired error level?
  - How accurate you want your network to be
- How can I choose the weights?
  - Randomly at first! But they should be updated

# Activation Functions

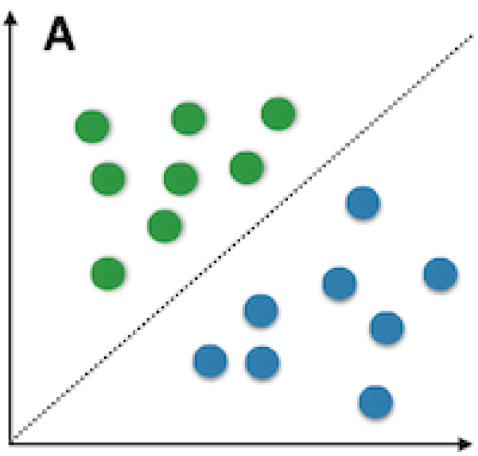
## What are they?!



# Activation Functions

**Linear data vs. Non-linear data**

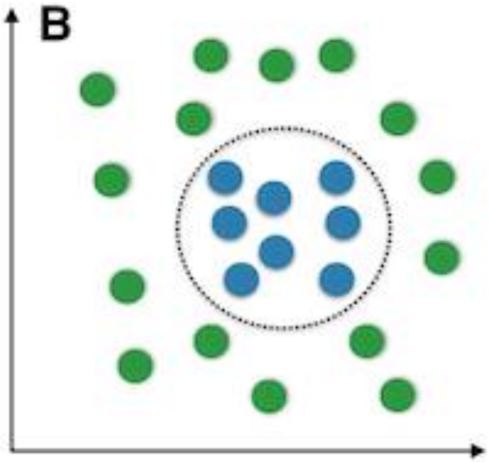
- In linear data structure
  - We can simply predict the outcome
  - Data elements are arranged in a linear order
  - They can be stored in array, stack, queue, etc.
  - Elements are attached to their adjacent



# Activation Functions

**Linear data vs. Non-linear data**

- In non-linear data structure
  - We need to analyze different aspects
  - Their implementations are complex
  - They can be stored in trees, graphs, etc.
  - Elements are attached in hierarchically manner



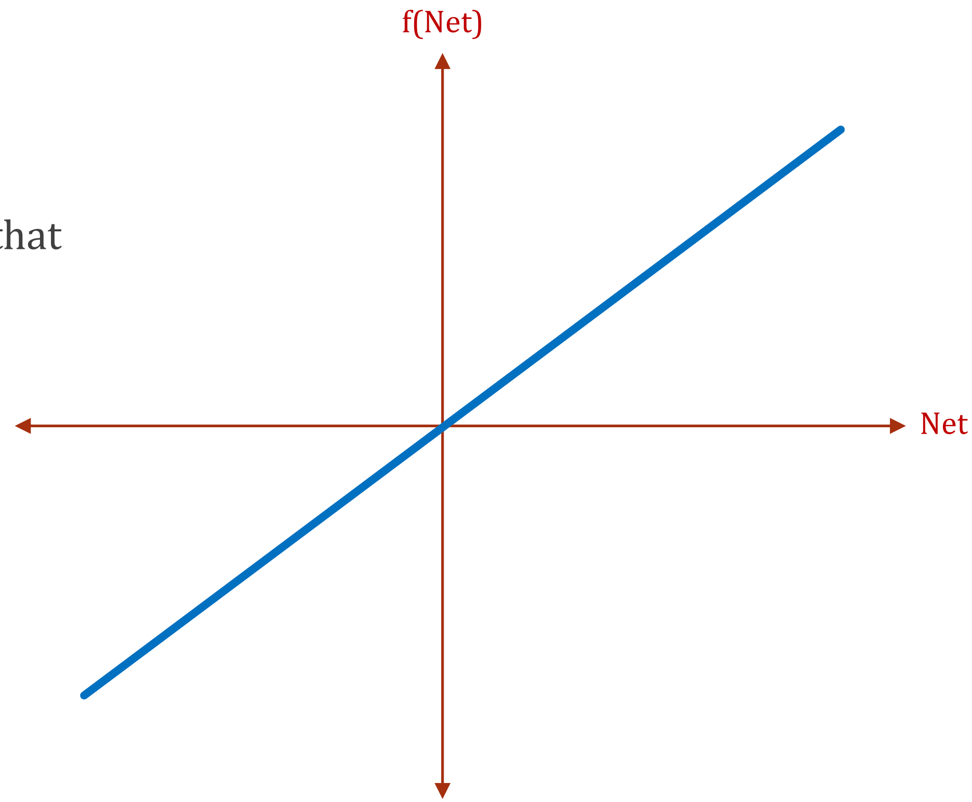
# Activation Functions

University of Guilan

**Common types of AFs**

- Linear (Identity function)
  - Returns the same value that was used as its argument
  - Sample NNs: *Adaline*

$$f(x) = x$$



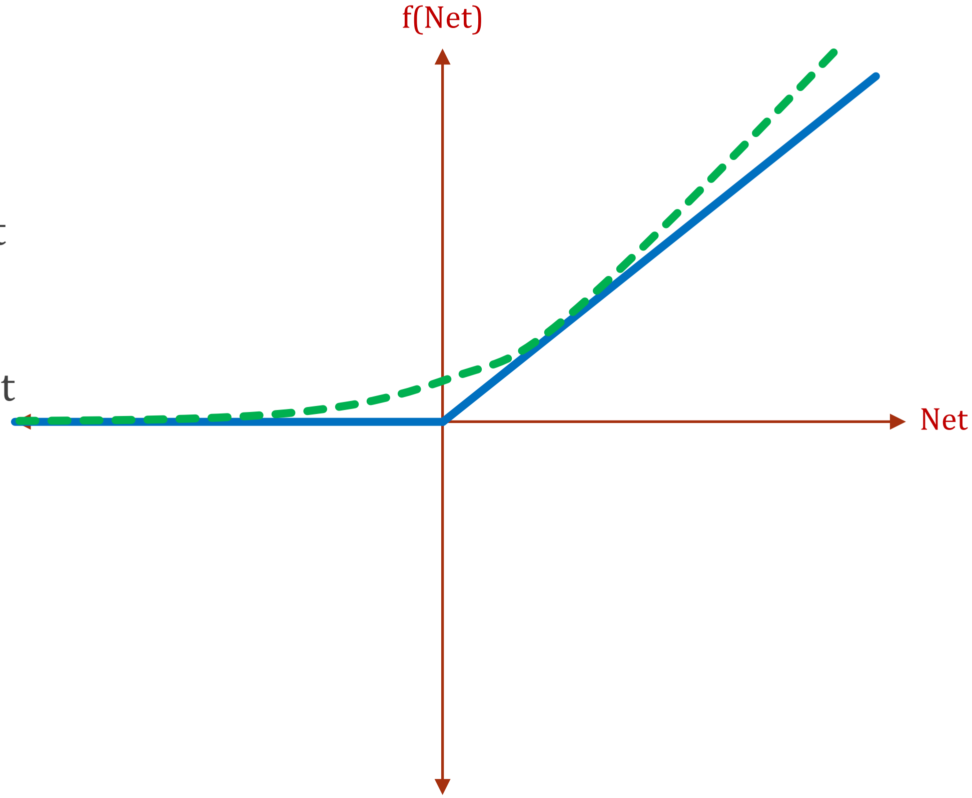
# Activation Functions

**Common types of AFs**

- ReLU (Rectifier)
  - Returns the positive part of its argument
  - Computationally efficient
  - Sample NNs: *Multi-layer*

$$f(x) = \max(0, z)$$

$$f(x) = \ln(1 + e^{z})$$

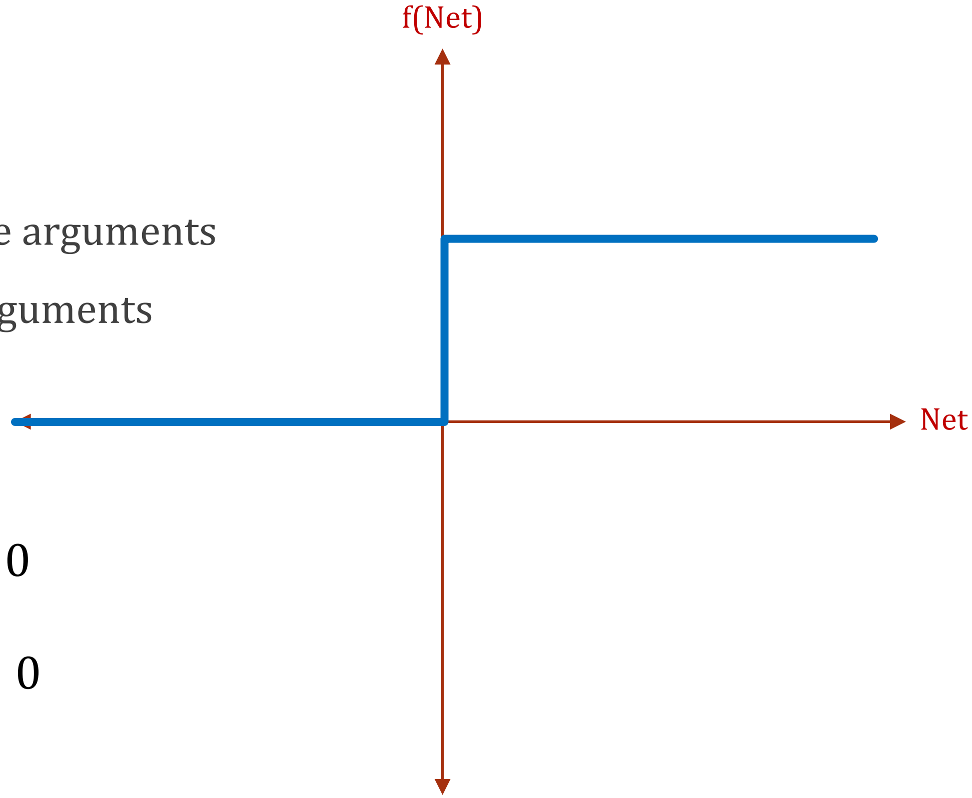


# Activation Functions

University of Guilan

**Common types of AFs**

- Unit step (Heaviside)
  - Returns zero for negative arguments and one for non-negative arguments
  - Sample NNs: *Perceptron*



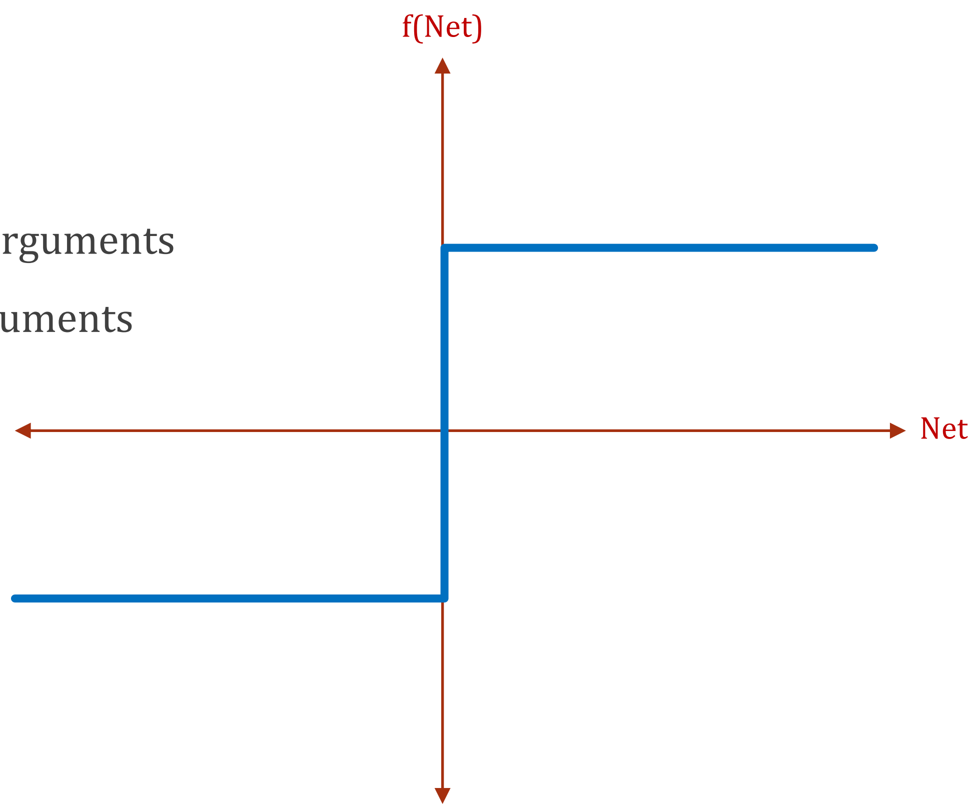

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

# Activation Functions

**Common types of AFs**

- Sign (Signum)
  - Returns -1 for negative arguments and +1 for non-negative arguments
  - Sample NNs: *Perceptron*

$$f(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$$

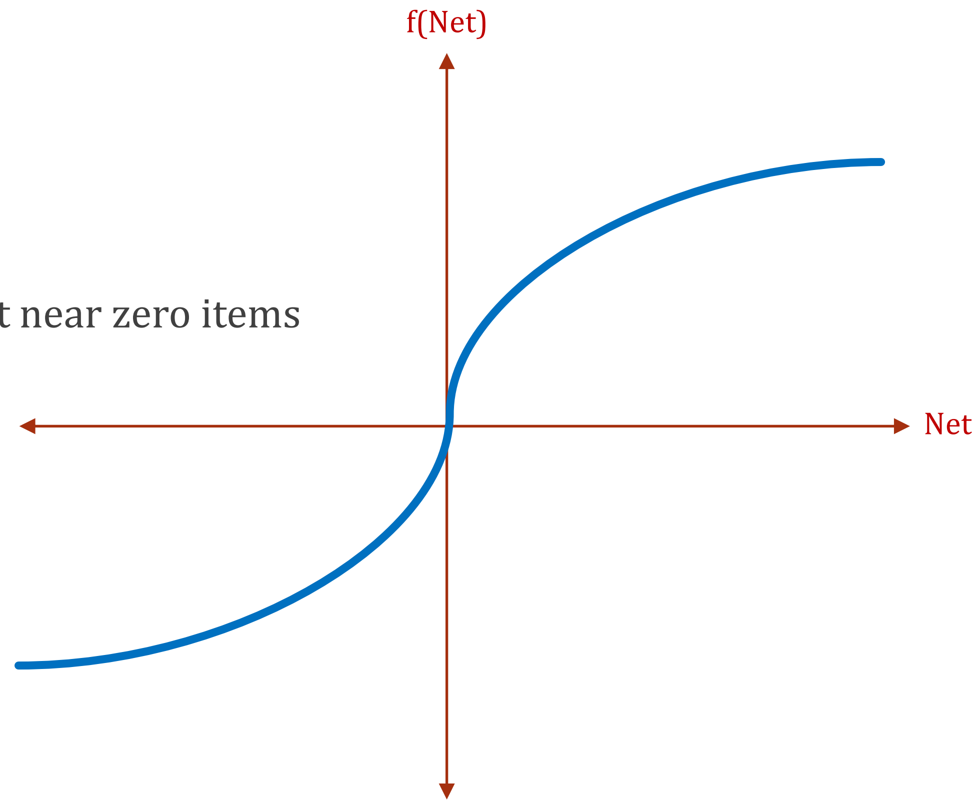


# Activation Functions

**Common types of AFs**

- Tanh (hyperbolic tangent)
  - Zero centered
  - Intensifies inputs, except near zero items
  - Range: (-1 to 1)
  - Sample NNs: *Multi-layer*

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

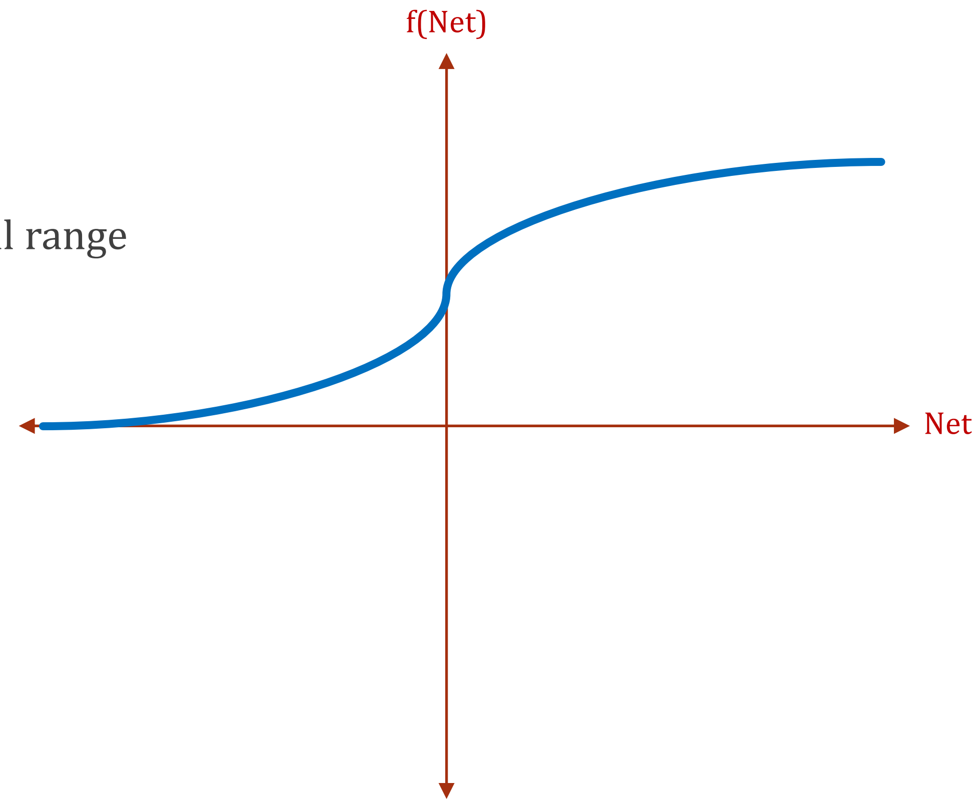


# Activation Functions

**Common types of AFs**

- Sigmoid (logistic)
  - Maps real values to small range
  - Clear predictions
  - Vanishing gradient
  - Range: (0 to 1)
  - Sample NNs: *Multi-layer*

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

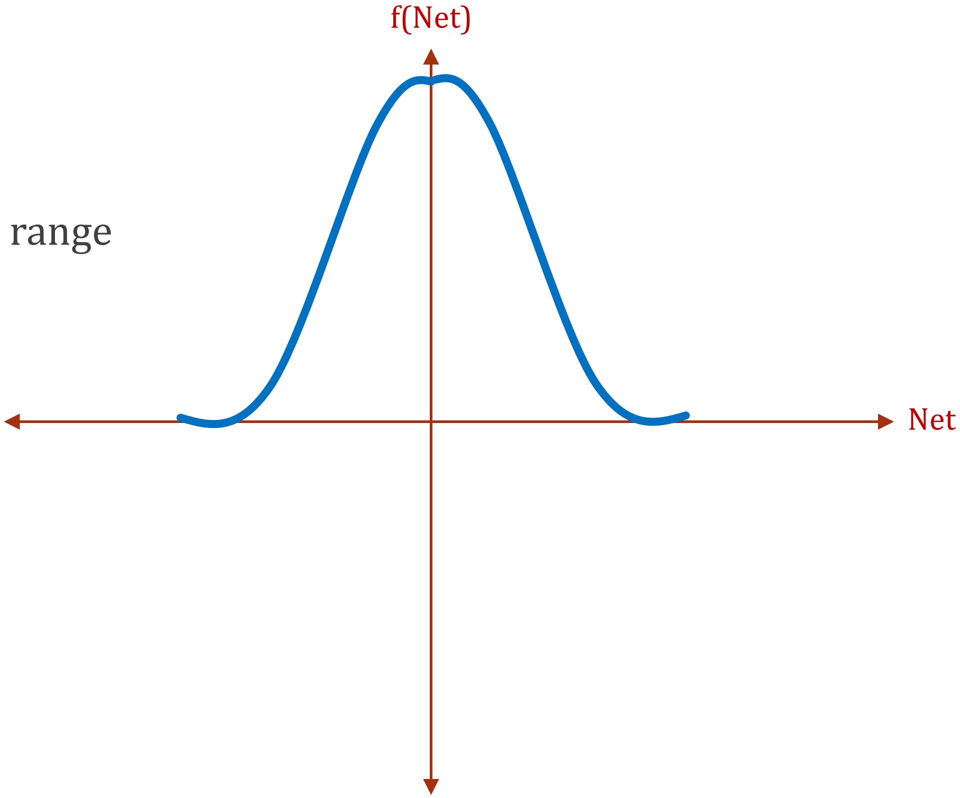


# Activation Functions

University of Guilan

**Common types of AFs**

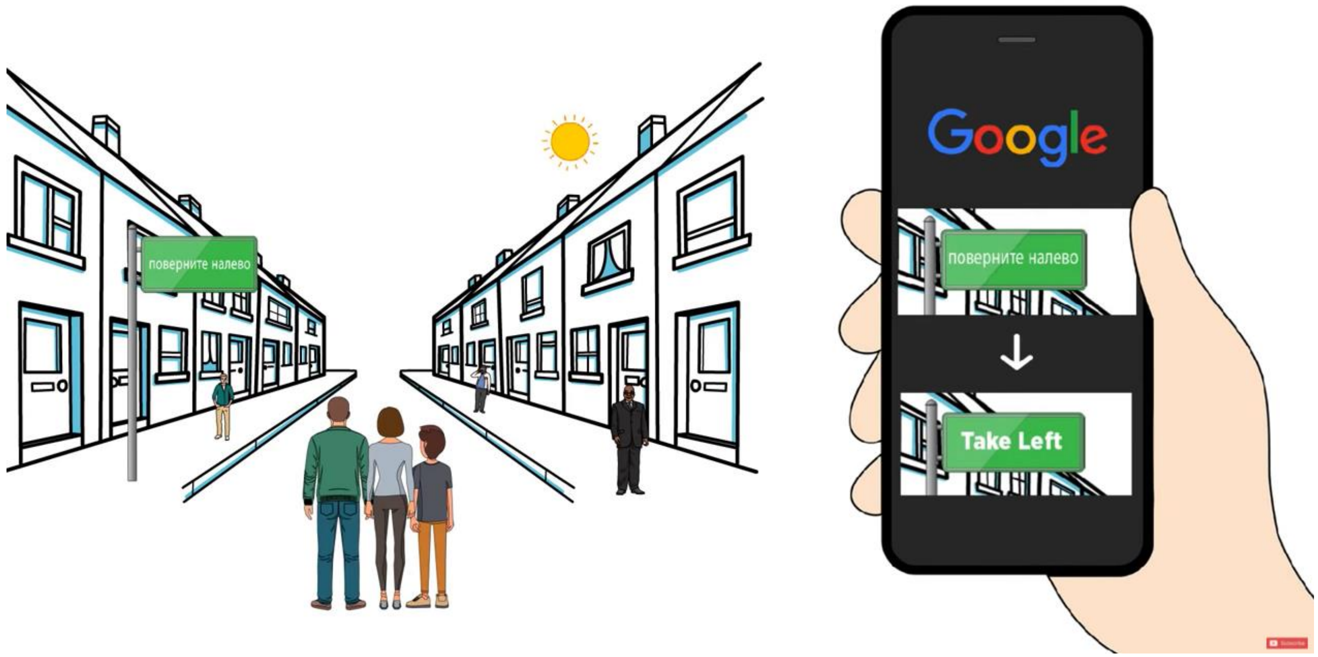
- Gaussian
  - Maps real values to small range
  - Range: (0, 1]
  - Sample NNs: *RBF*




$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\delta^2}}$$

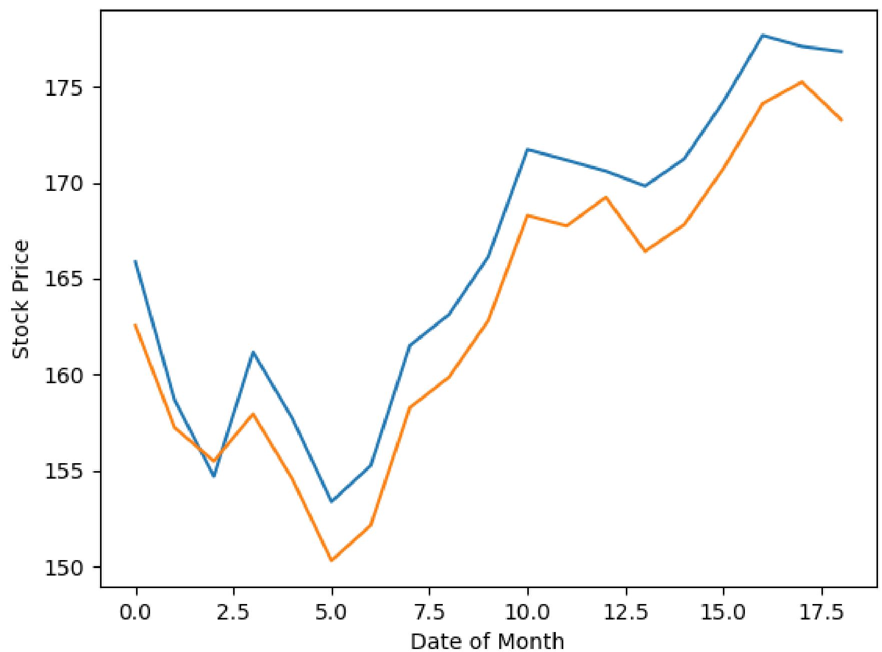
# Applications of ANNs

- Google Online Translation tool



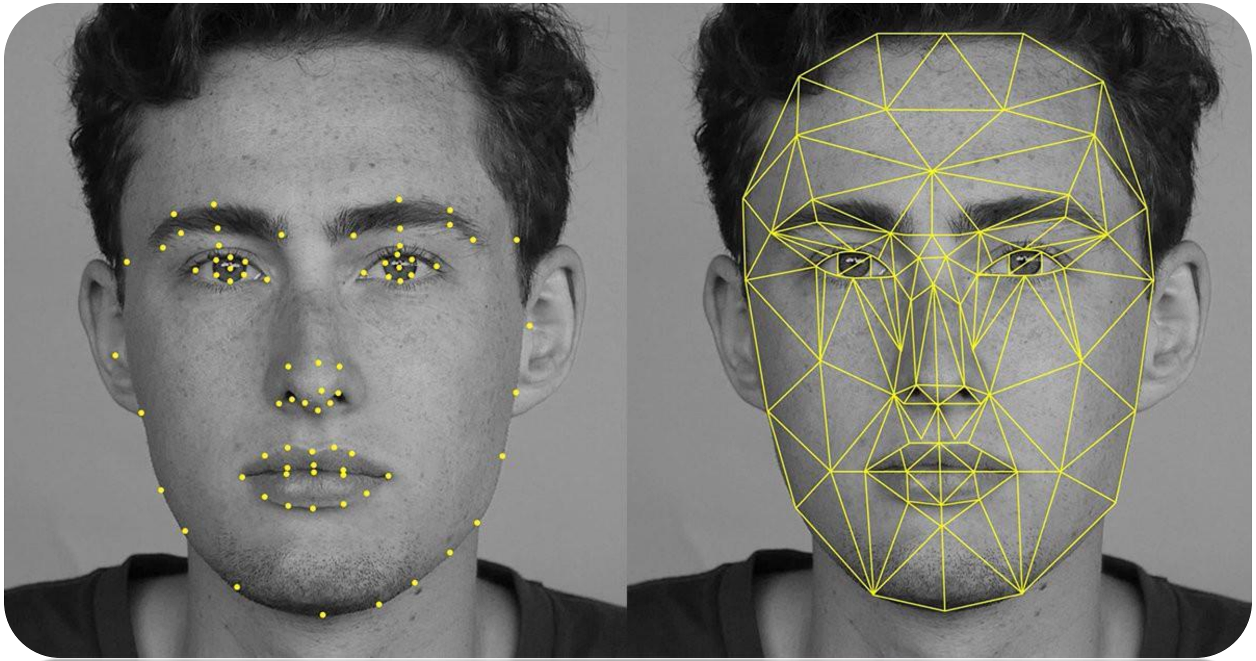
# Applications of ANNs

- Prediction of stock prices



# Applications of ANNs

- Human face recognition



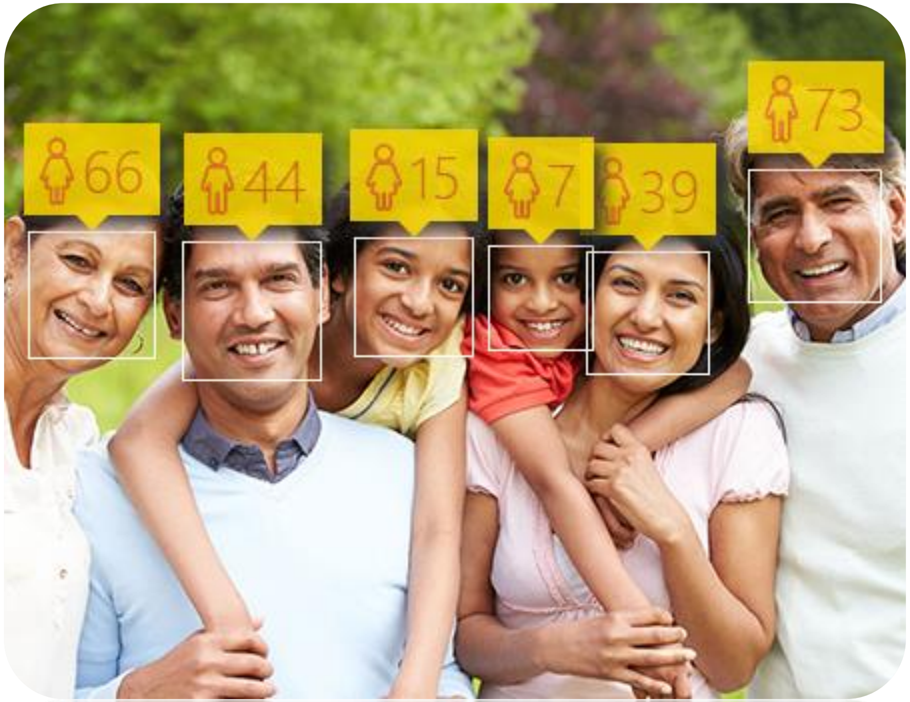
# Applications of ANNs

- Music composition

Music Generation Using Deep Learning

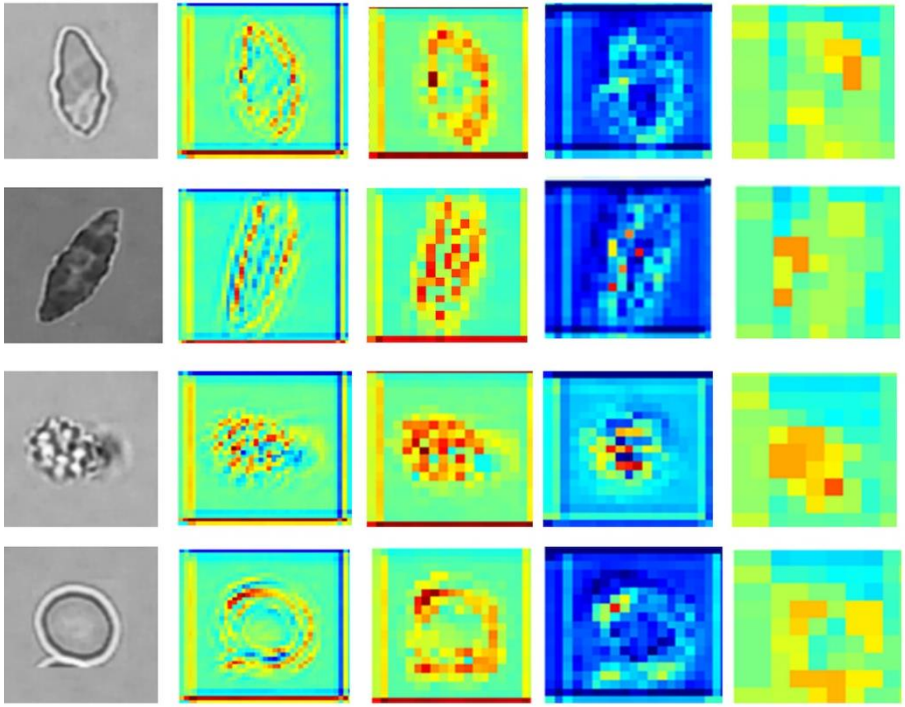
# Applications of ANNs

- Age estimation



# Applications of ANNs

- Pattern matching



# Applications of ANNs

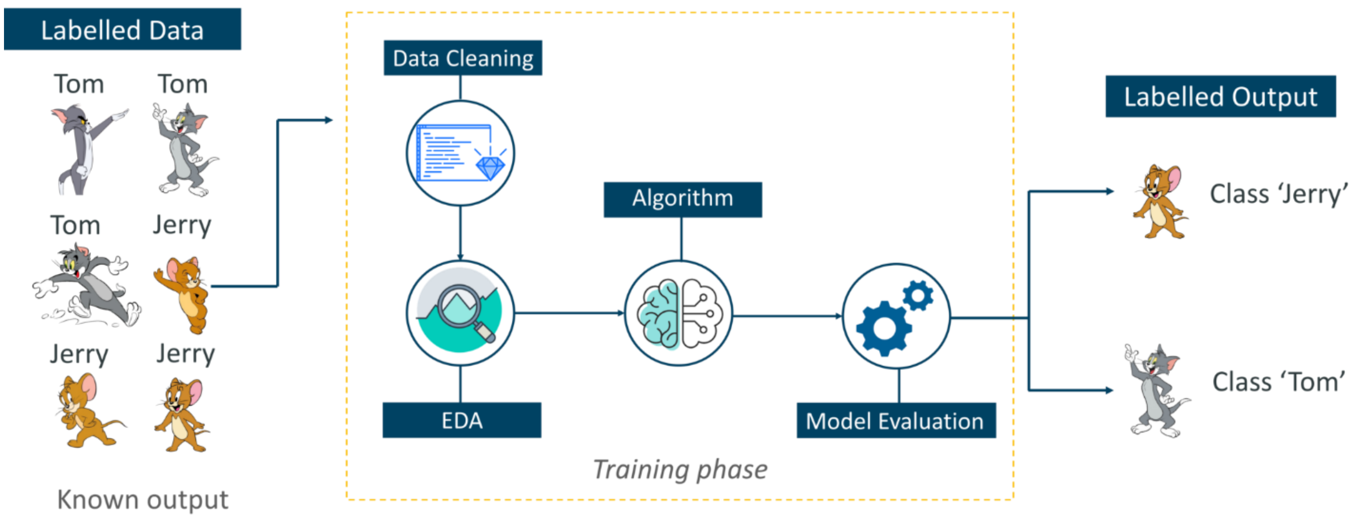
- Pattern completion

# Applications of ANNs

- Data mining

# What's Next?

- ANNs architectures, models, and parameters



# Questions?