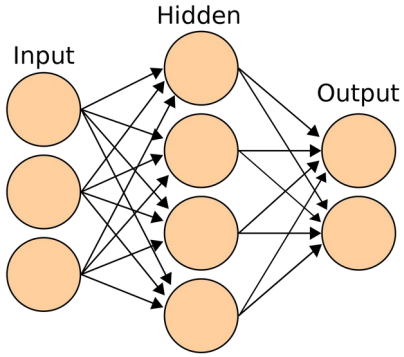
Multi Layer Perceptron

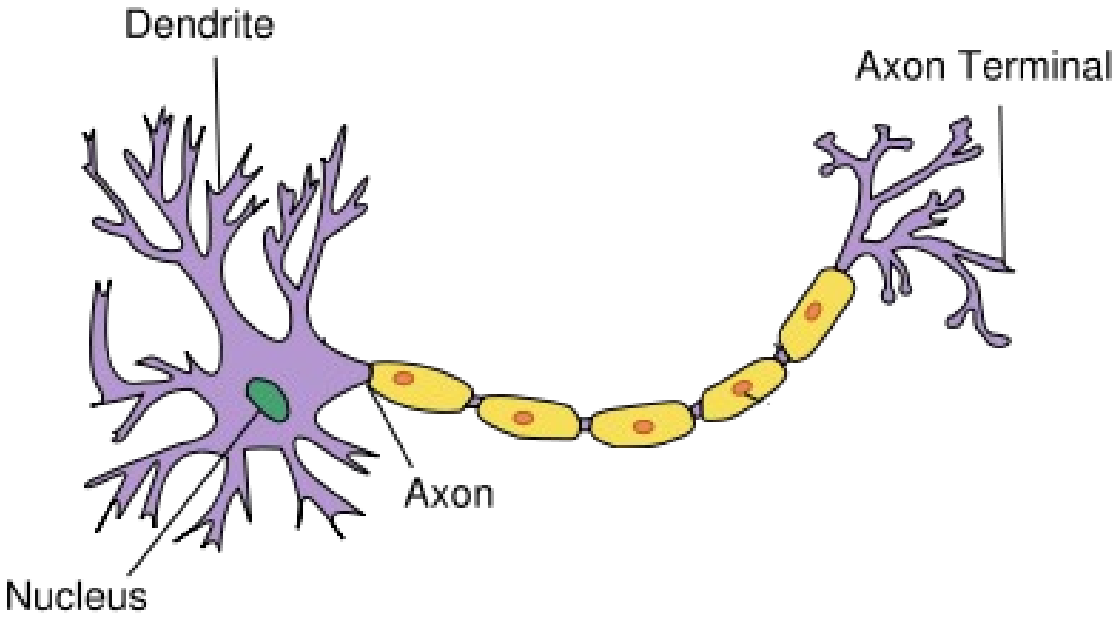
Tushar B. Kute,
http://tusharkute.com



Neural Network

- Humans have an ability to identify patterns within the accessible information with an astonishingly high degree of accuracy.
- Whenever you see a car or a bicycle you can immediately recognize what they are. This is because we have learned over a period of time how a car and bicycle looks like and what their distinguishing features are.
- Artificial neural networks are computation systems that intend to imitate human learning capabilities via a complex architecture that resembles the human nervous system.

Human Nervous System



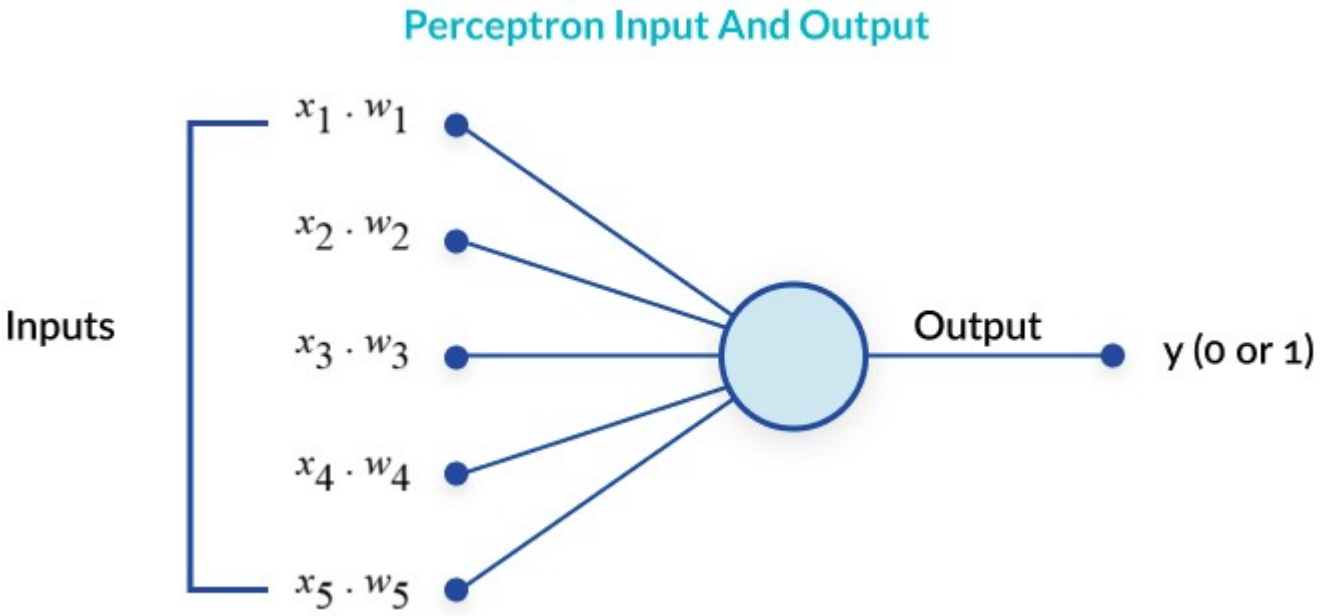
Human Nervous System

- Human nervous system consists of billions of neurons. These neurons collectively process input received from sensory organs, process the information, and decides what to do in reaction to the input.
- A typical neuron in the human nervous system has three main parts: dendrites, nucleus, and axons.
  - The information passed to a neuron is received by dendrites.
  - The nucleus is responsible for processing this information.
  - The output of a neuron is passed to other neurons via the axon, which is connected to the dendrites of other neurons further down the network.

Perceptron

- A perceptron is a simple binary classification algorithm, proposed by Cornell scientist Frank Rosenblatt.
- It helps to divide a set of input signals into two parts—“yes” and “no”.
- But unlike many other classification algorithms, the perceptron was modeled after the essential unit of the human brain—the neuron and has an uncanny ability to learn and solve complex problems.

Perceptron


Perceptron Input And Output
Inputs
$x_1 \cdot w_1$
$x_2 \cdot w_2$
$x_3 \cdot w_3$
$x_4 \cdot w_4$
$x_5 \cdot w_5$
Output
y (0 or 1)

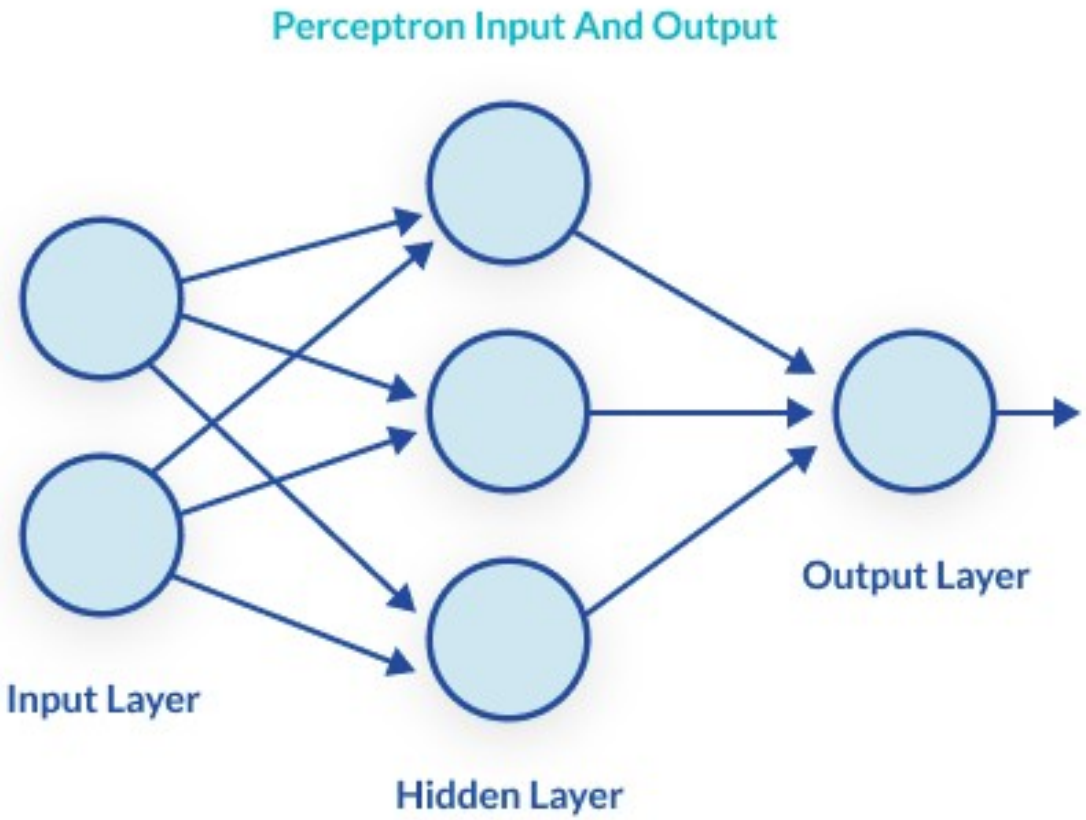
Perceptron

- A perceptron is a very simple learning machine. It can take in a few inputs, each of which has a weight to signify how important it is, and generate an output decision of “0” or “1”.
- However, when combined with many other perceptrons, it forms an artificial neural network.
- A neural network can, theoretically, answer any question, given enough training data and computing power.

Multilayer Perceptron

- A multilayer perceptron (MLP) is a perceptron that teams up with additional perceptrons, stacked in several layers, to solve complex problems.
- Each perceptron in the first layer on the left (the input layer), sends outputs to all the perceptrons in the second layer (the hidden layer), and all perceptrons in the second layer send outputs to the final layer on the right (the output layer).

Multilayer Perceptron



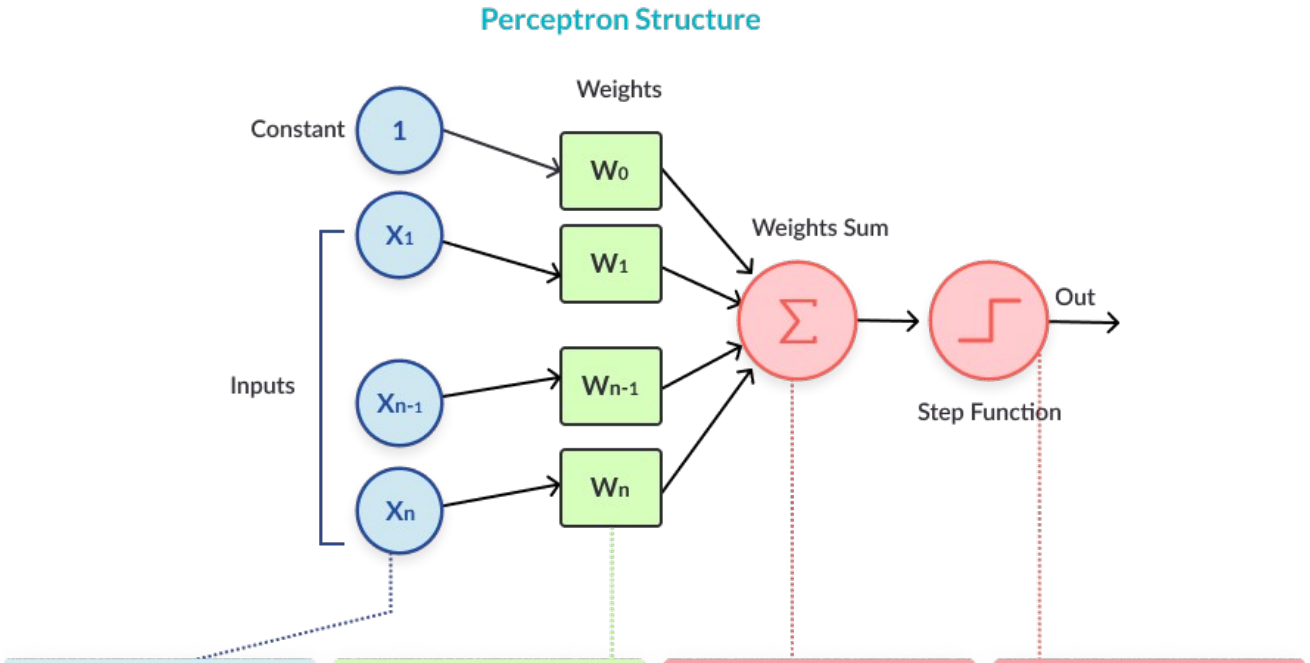
Multilayer Perceptron

- Each layer can have a large number of perceptrons, and there can be multiple layers, so the multilayer perceptron can quickly become a very complex system.
- The multilayer perceptron has another, more common name—a neural network.
- A three-layer MLP, like the diagram in previous slide, is called a Non-Deep or Shallow Neural Network.
- An MLP with four or more layers is called a Deep Neural Network.

Multilayer Perceptron

- One difference between an MLP and a neural network is that in the classic perceptron, the decision function is a step function and the output is binary.
- In neural networks that evolved from MLPs, other activation functions can be used which result in outputs of real values, usually between 0 and 1 or between -1 and 1.
- This allows for probability-based predictions or classification of items into multiple labels.

Structure of a Perceptron



The Percpetron Learning Process

1. Takes the inputs, multiplies them by their weights, and computes their sum
2. Adds a bias factor, the number 1 multiplied by a weight
3. Feeds the sum through the activation function
4. The result is the perceptron output

Step-1 Backpropogation

- Takes the inputs, multiplies them by their weights, and computes their sum
- Why It's Important ?
  - The weights allow the perceptron to evaluate the relative importance of each of the outputs.
  - Neural network algorithms learn by discovering better and better weights that result in a more accurate prediction.
  - There are several algorithms used to fine tune the weights, the most common is called backpropagation.

Step-2 Neural Network Bias

- Adds a bias factor, the number 1 multiplied by a weight
- This is a technical step that makes it possible to move the activation function curve up and down, or left and right on the number graph.
- It makes it possible to fine-tune the numeric output of the perceptron.

Step-3 Activation Function

- Feeds the sum through the activation function
- The activation function maps the input values to the required output values.
- For example, input values could be between 1 and 100, and outputs can be 0 or 1. The activation function also helps the perceptron to learn, when it is part of a multilayer perceptron (MLP).
- Certain properties of the activation function, especially its non-linear nature, make it possible to train complex neural networks.

Step-4 Output

- The perceptron output is a classification decision.
- In a multilayer perceptron, the output of one layer's perceptrons is the input of the next layer.
- The output of the final perceptrons, in the "output layer", is the final prediction of the perceptron learning model.

Transformation

- From the Classic Perceptron to a Full-Fledged Deep Neural Network
- Although multilayer perceptrons (MLP) and neural networks are essentially the same thing, you need to add a few ingredients before an MLP becomes a full neural network. These are:
  - Backpropagation
  - Hyperparameters
  - Advanced structures

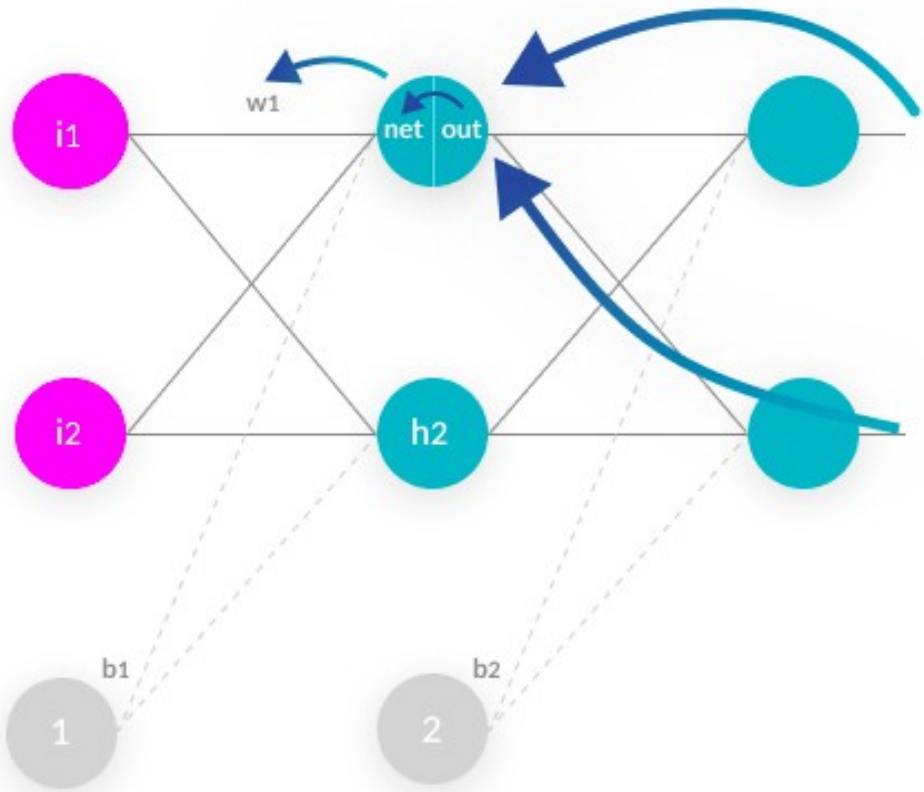
Backpropogation

- The backpropagation algorithm allows you to perform a “backward pass”, which helps tune the weights of the inputs.
- Backpropagation performs iterative backward passes which attempt to minimize the “loss”, or the difference between the known correct prediction and the actual model prediction.
- With each backward pass, the weights move towards an optimum that minimizes the loss function and results in the most accurate prediction.

Backpropogation

- Backpropagation is an algorithm commonly used to train neural networks.
- When the neural network is initialized, weights are set for its individual elements, called neurons.
- Inputs are loaded, they are passed through the network of neurons, and the network provides an output for each one, given the initial weights.
- Backpropagation helps to adjust the weights of the neurons so that the result comes closer and closer to the known true result.

Backpropogation



Hyperparameters

- In a modern neural network, aspects of the multilayer structure such as the number of layers, initial weights, the type of activation function, and details of the learning process, are treated as parameters and tuned to improve the performance of the neural network.
- Tuning hyperparameters is an art, and can have a huge impact on the performance of a neural network.

Model vs. Hyperparameters

- Model parameters are internal to the neural network – for example, neuron weights. They are estimated or learned automatically from training samples. These parameters are also used to make predictions in a production model.
- Hyperparameters are external parameters set by the operator of the neural network – for example, selecting which activation function to use or the batch size used in training.
- Hyperparameters have a huge impact on the accuracy of a neural network, there may be different optimal values for different values, and it is non-trivial to discover those values.

Hyperparameters of Neural N/W

- Number of hidden layers
- Dropout
- Neural network activation function
- Weights initialization

Hyperparameters of Neural N/W

- Number of hidden layers –
  - adding more hidden layers of neurons generally improves accuracy, to a certain limit which can differ depending on the problem.
- Dropout –
  - what percentage of neurons should be randomly “killed” during each epoch to prevent overfitting.
- Neural network activation function –
  - which function should be used to process the inputs flowing into each neuron. The activation function can impact the network’s ability to converge and learn for different ranges of input values, and also its training speed.

Hyperparameters of Neural N/W

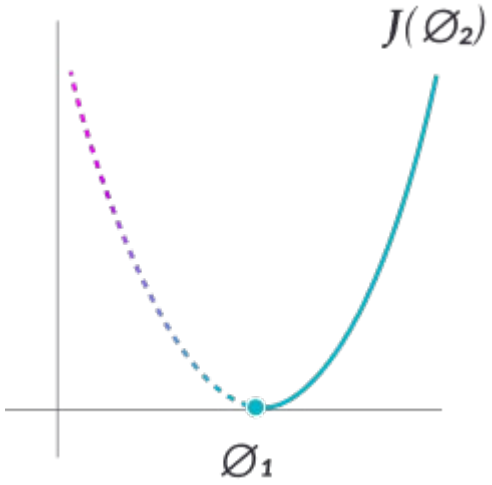
- Weights initialization –
  - it is necessary to set initial weights for the first forward pass. Two basic options are to set weights to zero or to randomize them.
  - However, this can result in a vanishing or exploding gradient, which will make it difficult to train the model.
  - To mitigate this problem, you can use a heuristic (a formula tied to the number of neuron layers) to determine the weights.
  - A common heuristic used for the Tanh activation is called Xavier initialization.

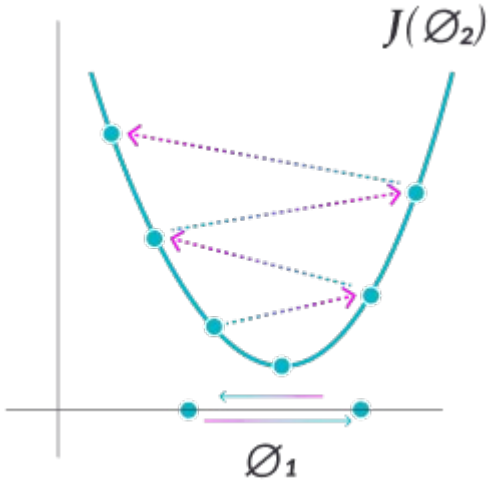
Hyperparameters of training algo

- Neural network learning rate
- Deep learning epoch, iterations and batch size
- Optimizer algorithm and neural network momentum

Neural Network Learning Rate

- How fast the backpropagation algorithm performs gradient descent.
- A lower learning rate makes the network train faster but might result in missing the minimum of the loss function.

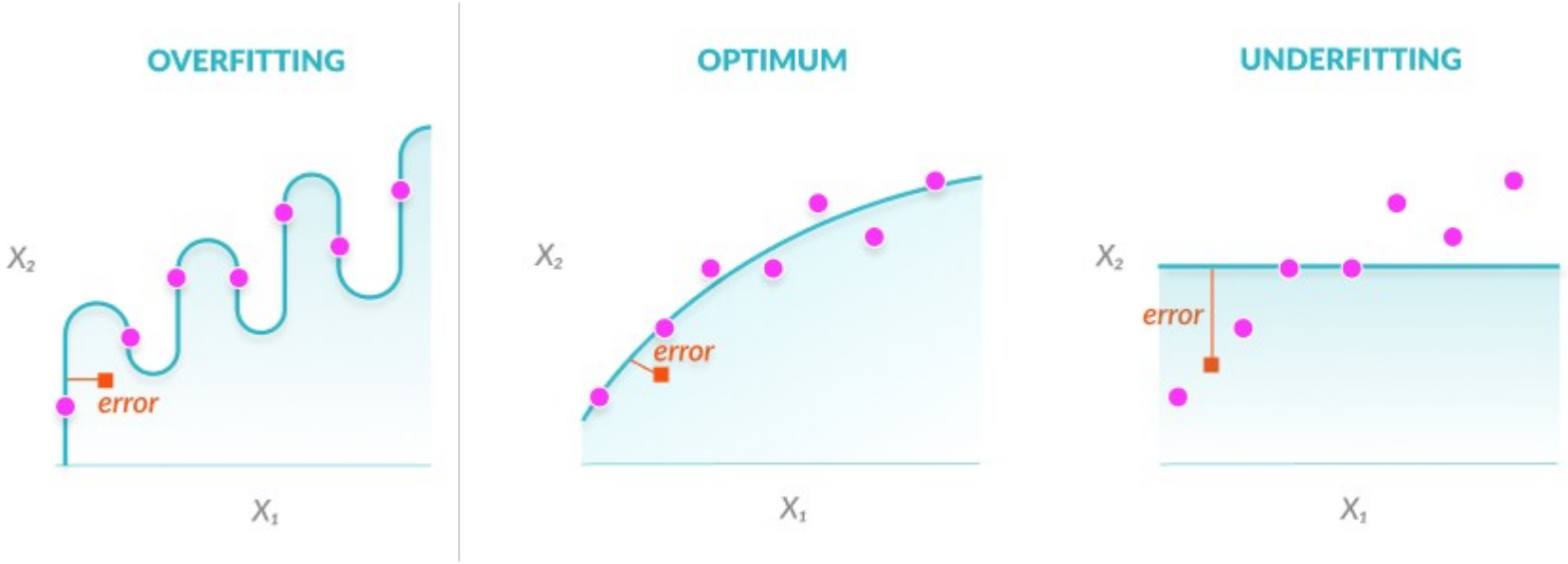




Epoch, iterations, batch size

- Deep learning epoch, iterations and batch size – these parameters determine the rate at which samples are fed to the model for training.
- An epoch is a group of samples which are passed through the model together (forward pass) and then run through backpropagation (backward pass) to determine their optimal weights.
- If the epoch cannot be run all together due the size of the sample or complexity of the network, it is split into batches, and the epoch is run in two or more iterations.
- The number of epochs and batches per epoch can significantly affect model fit, as shown (next slide).

Epoch, iterations, batch size



Optimizer Algorithm

- Optimizer algorithm and neural network momentum – when a neural network trains, it uses an algorithm to determine the optimal weights for the model, called an optimizer.
- The basic option is Stochastic Gradient Descent, but there are other options.
- Another common algorithm is Momentum, which works by waiting after a weight is updated, and updating it a second time using a delta amount.
- This speeds up training gradually, with a reduced risk of oscillation. Other algorithms are Nesterov Accelerated Gradient, AdaDelta and Adam.

Hyperparameter Tuning Methods

- Manual Hyperparameter Tuning
- Grid Search
- Random Search
- Bayesian Optimization

Manual Tuning

- Traditionally, hyperparameters were tuned manually by trial and error.
- This is still commonly done, and experienced operators can “guess” parameter values that will achieve very high accuracy for deep learning models.
- However, there is a constant search for better, faster and more automatic methods to optimize hyperparameters.
- **Pros**: Very simple and effective with skilled operators
- **Cons**: Not scientific, unknown if you have fully optimized hyperparameters

Grid Search

- Grid search is slightly more sophisticated than manual tuning. It involves systematically testing multiple values of each hyperparameter, by automatically retraining the model for each value of the parameter.
- For example, you can perform a grid search for the optimal batch size by automatically training the model for batch sizes between 10-100 samples, in steps of 20.
- The model will run 5 times and the batch size selected will be the one which yields highest accuracy.
- **Pros**: Maps out the problem space and provides more opportunity for optimization
- **Cons**: Can be slow to run for large numbers of hyperparameter values

Random Search

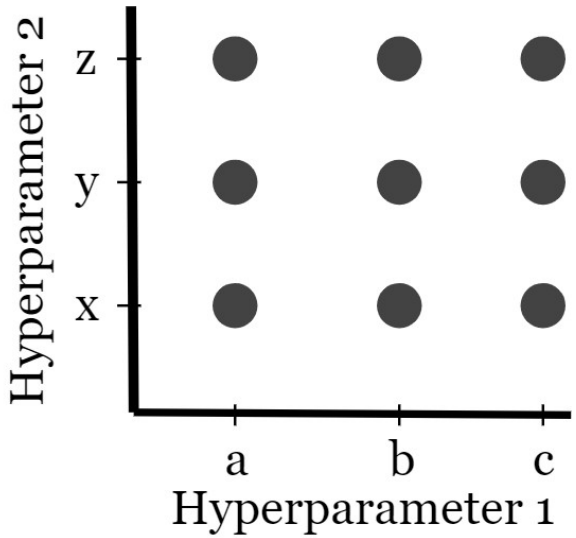
- According to a 2012 research study by James Bergstra and Yoshua Bengio, testing randomized values of hyperparameters is actually more effective than manual search or grid search.
- In other words, instead of testing systematically to cover “promising areas” of the problem space, it is preferable to test random values drawn from the entire problem space.
- **Pros**: According to the study, provides higher accuracy with less training cycles, for problems with high dimensionality
- **Cons**: Results are unintuitive, difficult to understand “why” hyperparameter values were chosen

Comparing

Grid Search

Pseudocode

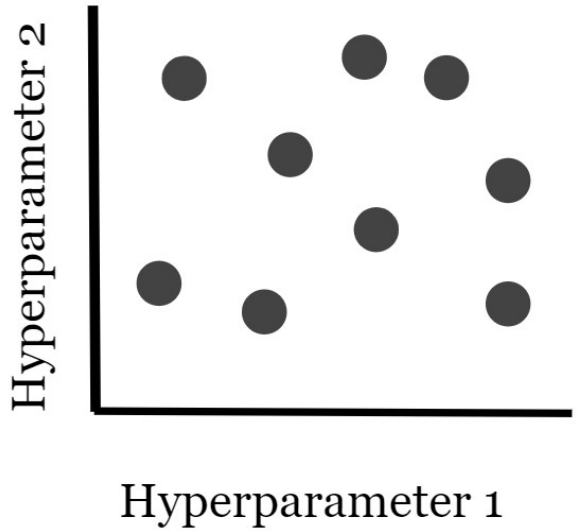
```
Hyperparameter_One = [a, b, c]
Hyperparameter_Two = [x, y, z]
```



Random Search

Pseudocode

```
Hyperparameter_One = random.num(range)
Hyperparameter_Two = random.num(range)
```



Baysian Optimization

- Bayesian optimization (described by Shahriari, et al) is a technique which tries to approximate the trained model with different possible hyperparameter values.
- To simplify, bayesian optimization trains the model with different hyperparameter values, and observes the function generated for the model by each set of parameter values.
- It does this over and over again, each time selecting hyperparameter values that are slightly different and can help plot the next relevant segment of the problem space.

Baysian Optimization

- Similar to sampling methods in statistics, the algorithm ends up with a list of possible hyperparameter value sets and model functions, from which it predicts the optimal function across the entire problem set.
- **Pros**: The original study and practical experience from the industry shows that bayesian optimization results in significantly higher accuracy compared to random search.
- **Cons**: Like random search, results are not intuitive and difficult to improve on, even by trained operators

In real world...

- In a real neural network project, you will have three practical options:
  - Performing manual optimization
  - Leveraging hyperparameter optimization techniques in the deep learning framework of your choice. The framework will report on hyperparameter values discovered, their accuracy and validation scores
  - Using third party hyperparameter optimization tools

Advanced Strutures

- Many neural networks use a complex structure that builds on the multilayer perceptron.
- For example, a Recurrent Neural Network (RNN) uses two neural networks in parallel—one runs the training data from beginning to end, the other from the end to the beginning, which helps with language processing.
- A Convolutional Neural Network (CNN) uses a three-dimensional MLP—essentially, three multilayer perceptron structures that learn the same data point.
- This is useful for color images which have three layers of “depth”—red, green and blue.

Neural Network in Real World

- In the real world, perceptrons work under the hood. You will run neural networks using deep learning frameworks such as TensorFlow, Keras, and PyTorch.
- These frameworks ask you for hyperparameters such as the number of layers, activation function, and type of neural network, and construct the network of perceptrons automatically.
- When you work on real, production-scale deep learning projects, you will find that the operations side of things can become a bit daunting:

Neural Network in Real World

- Running experiments at scale and tracking results, source code, metrics, and hyperparameters.
  - To succeed at deep learning you need to run large numbers of experiments and manage them correctly to see what worked.
- Running experiments across multiple machines—
  - in most cases neural networks are computationally intensive. To work efficiently, you'll need to run experiments on multiple machines. This requires provisioning these machines and distributing the work.

Neural Network in Real World

- Manage training data—
  - The more training data you provide, the better the model will learn and perform.
  - There are files to manage and copy to the training machines.
  - If your model's input is multimedia, those files can weigh anywhere from Gigabytes to Petabytes.

Activation Function

- Neural network activation functions are a crucial component of deep learning.
- Activation functions determine the output of a deep learning model, its accuracy, and also the computational efficiency of training a model—which can make or break a large scale neural network.
- Activation functions also have a major effect on the neural network's ability to converge and the convergence speed, or in some cases, activation functions might prevent neural networks from converging in the first place.

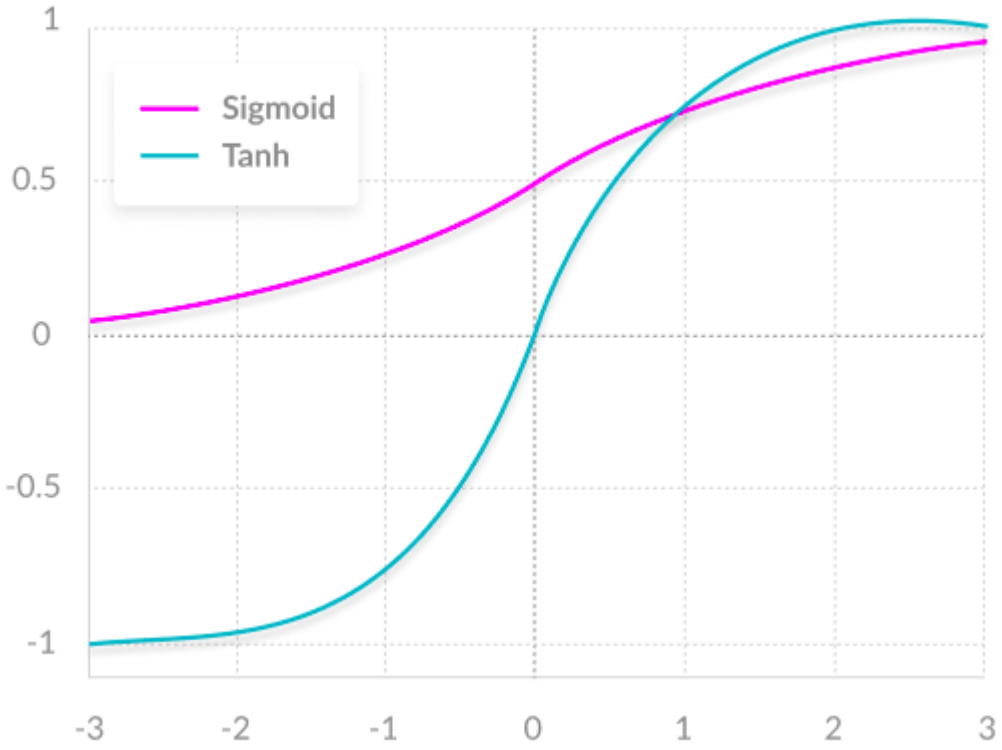
Activation Function

- Activation functions are mathematical equations that determine the output of a neural network.
- The function is attached to each neuron in the network, and determines whether it should be activated (“fired”) or not, based on whether each neuron’s input is relevant for the model’s prediction.
- Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

Activation Function

- An additional aspect of activation functions is that they must be computationally efficient because they are calculated across thousands or even millions of neurons for each data sample.
- Modern neural networks use a technique called backpropagation to train the model, which places an increased computational strain on the activation function, and its derivative function.

Common Activation Function

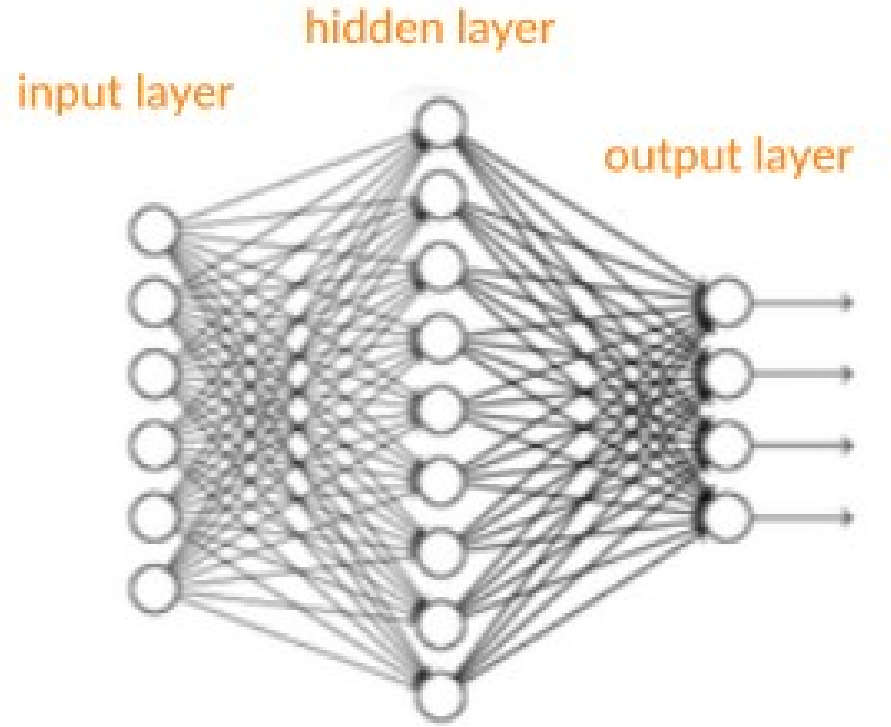


Two common neural network activation functions - Sigmoid and Tanh

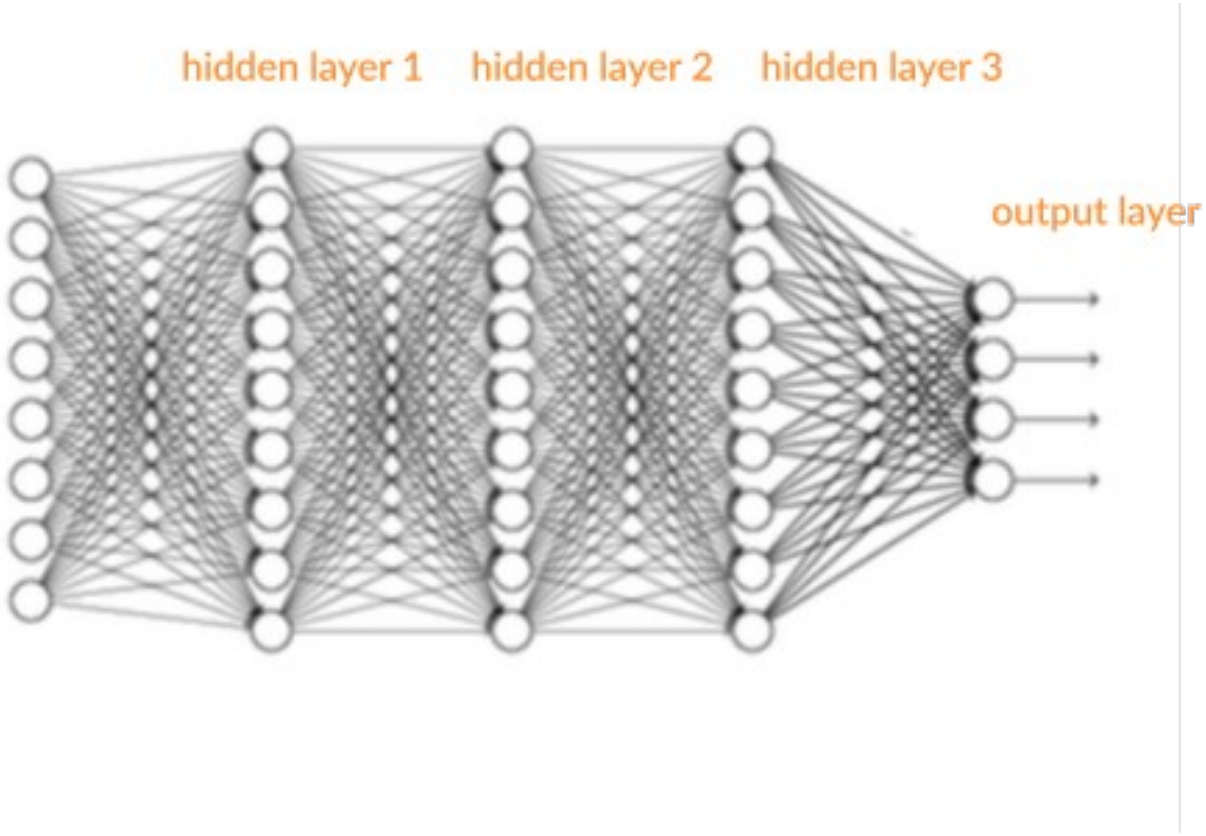
ANN and DNN

- Artificial Neural Networks (ANN) are comprised of a large number of simple elements, called neurons, each of which makes simple decisions. Together, the neurons can provide accurate answers to some complex problems, such as natural language processing, computer vision, and AI.
- A neural network can be “shallow”, meaning it has an input layer of neurons, only one “hidden layer” that processes the inputs, and an output layer that provides the final output of the model.
- A Deep Neural Network (DNN) commonly has between 2-8 additional layers of neurons.

Non-Deep Feed Forward Neural N/W



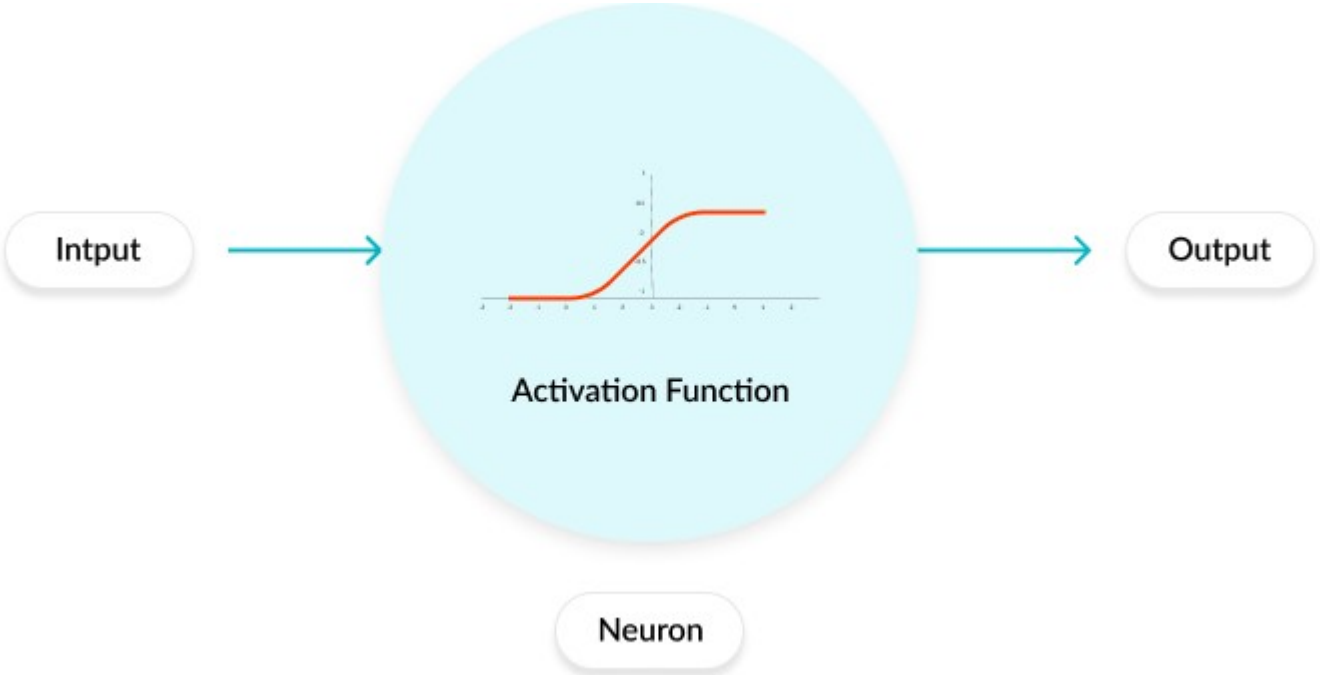
Deep Neural Network



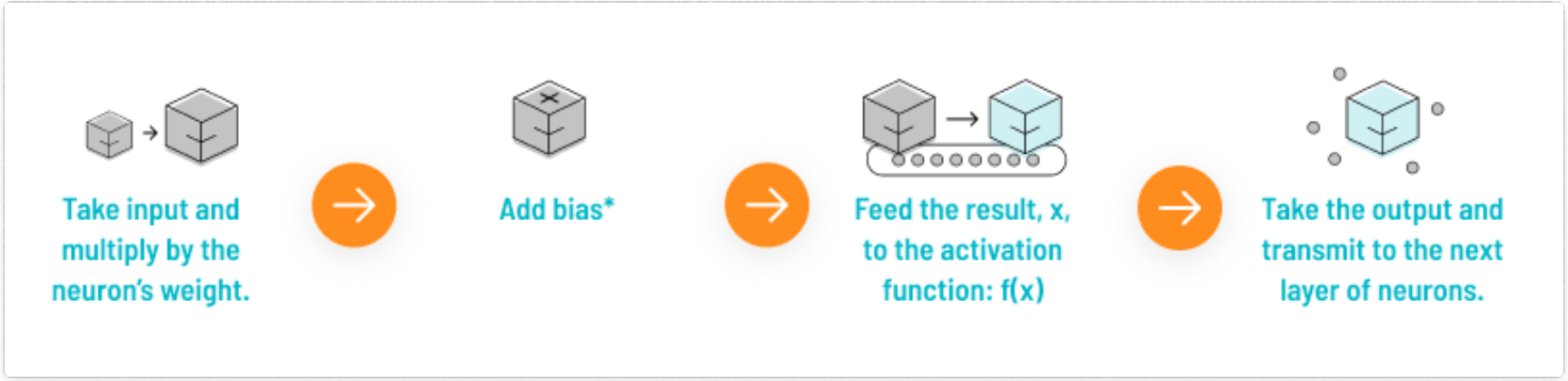
Role of Activation Function

- In a neural network, numeric data points, called inputs, are fed into the neurons in the input layer. Each neuron has a weight, and multiplying the input number with the weight gives the output of the neuron, which is transferred to the next layer.
- The activation function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.
- It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold. Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function.

Role of Activation Function



Process Carried out by Neuron

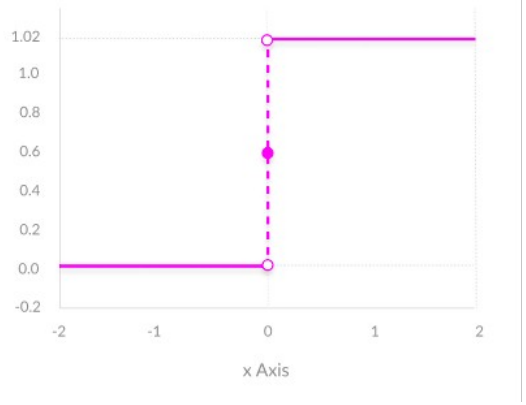


Types of Activation Function

- Binary Step Function
- Linear Activation Function
- Non Linear Activation Function

Binary Step Function

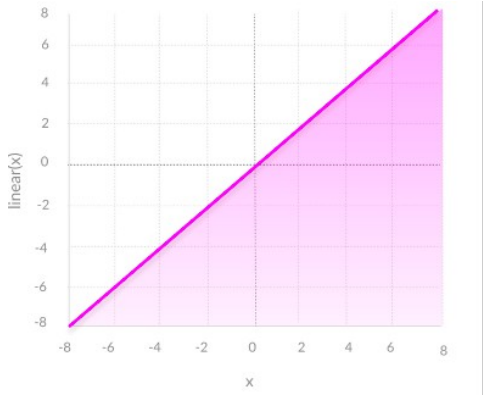
- A binary step function is a threshold-based activation function. If the input value is above or below a certain threshold, the neuron is activated and sends exactly the same signal to the next layer.



- The problem with a step function is that it does not allow multi-value outputs—for example, it cannot support classifying the inputs into one of several categories.

Linear Activation Function

- It takes the inputs, multiplied by the weights for each neuron, and creates an output signal proportional to the input.
- In one sense, a linear function is better than a step function because it allows multiple outputs, not just yes and no.



Problems: Linear Activation Function

- Not possible to use backpropagation (gradient descent) to train the model—
  - The derivative of the function is a constant, and has no relation to the input, X. So it's not possible to go back and understand which weights in the input neurons can provide a better prediction.
- All layers of the neural network collapse into one—
  - With linear activation functions, no matter how many layers in the neural network, the last layer will be a linear function of the first layer (because a linear combination of linear functions is still a linear function). So a linear activation function turns the neural network into just one layer.

Non Linear Activation Function

- Modern neural network models use non-linear activation functions. They allow the model to create complex mappings between the network's inputs and outputs, which are essential for learning and modeling complex data, such as images, video, audio, and data sets which are non-linear or have high dimensionality.
- Almost any process imaginable can be represented as a functional computation in a neural network, provided that the activation function is non-linear.

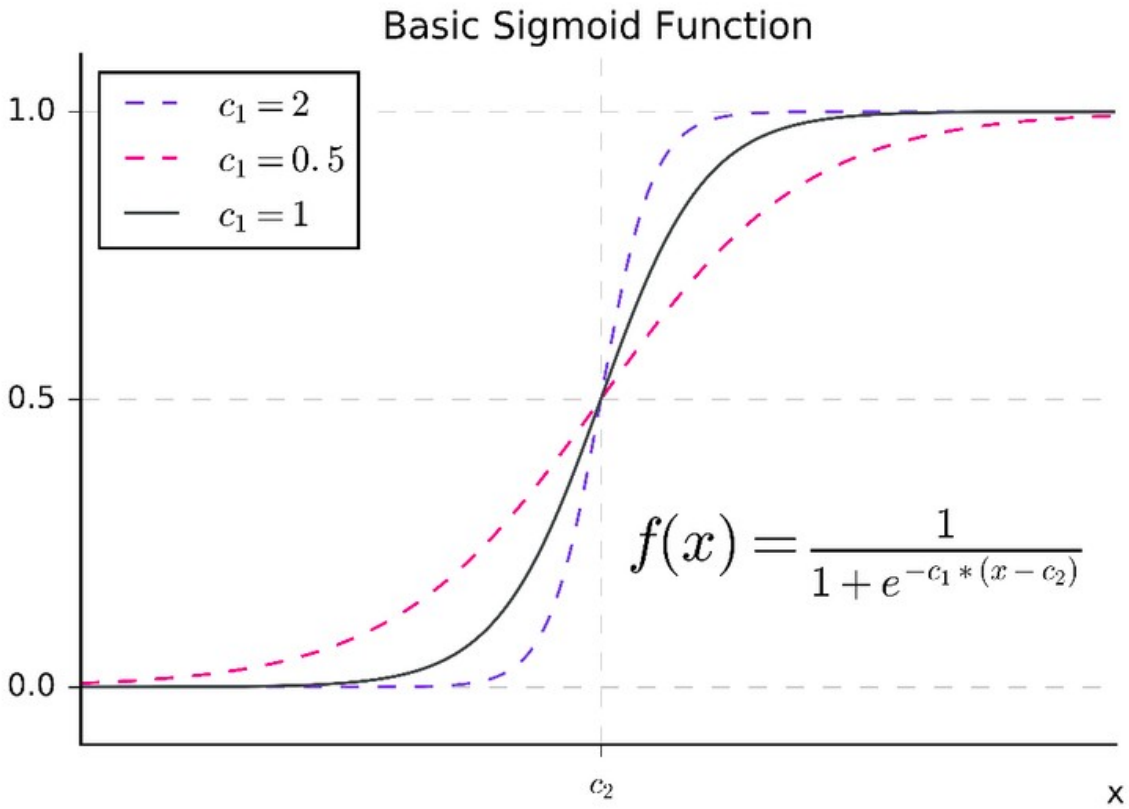
Problems: Non Linear Activation Function

- Non-linear functions address the problems of a linear activation function:
  - They allow backpropagation because they have a derivative function which is related to the inputs.
  - They allow “stacking” of multiple layers of neurons to create a deep neural network. Multiple hidden layers of neurons are needed to learn complex data sets with high levels of accuracy.

Common Non-Linear Functions

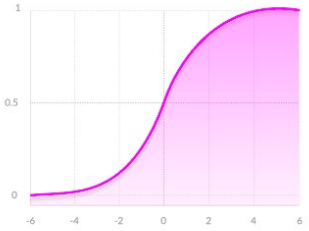
- Sigmoid / Logistic
- Tanh / Hyperbolic Tangent
- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Parametric ReLU
- Softmax
- Swish

Sigmoid / Logistic



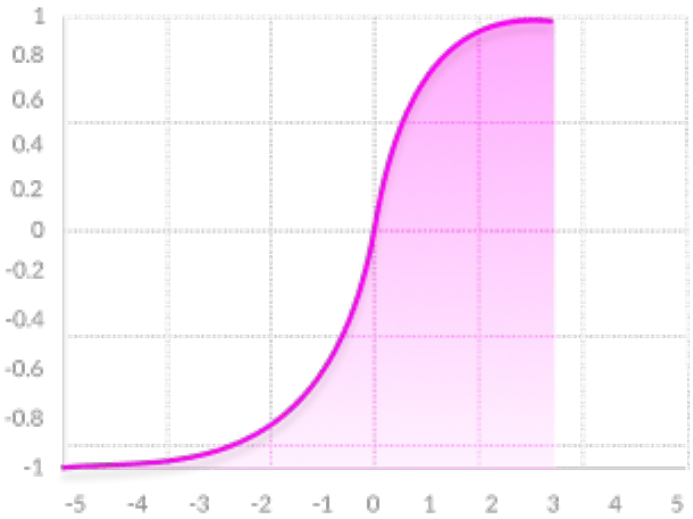
Sigmoid / Logistic

- Advantages
  - Smooth gradient, preventing “jumps” in output values.
  - Output values bound between 0 and 1, normalizing the output of each neuron.
  - Clear predictions—For X above 2 or below -2, tends to bring the Y value (the prediction) to the edge of the curve, very close to 1 or 0. This enables clear predictions.
- Disadvantages
  - Vanishing gradient—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
  - Outputs not zero centered.
  - Computationally expensive

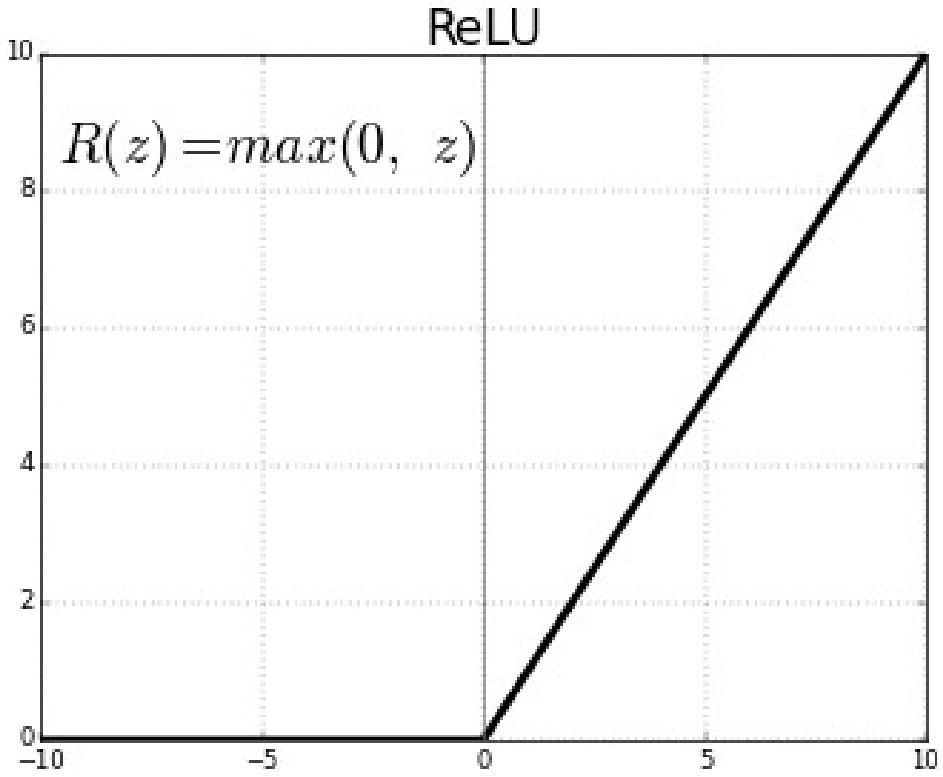


Tanh

- Advantages
  - Zero centered—making it easier to model inputs that have strongly negative, neutral, and strongly positive values.
  - Otherwise like the Sigmoid function.
- Disadvantages
  - Like the Sigmoid function

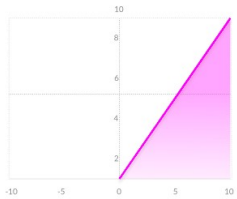


ReLU (Rectified Linear Unit)



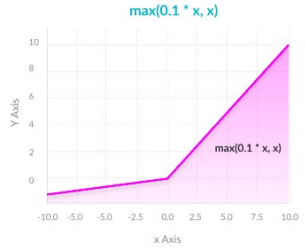
ReLU (Rectified Linear Unit)

- Advantages
  - Computationally efficient—allows the network to converge very quickly
  - Non-linear—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation
- Disadvantages
  - The Dying ReLU problem—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.

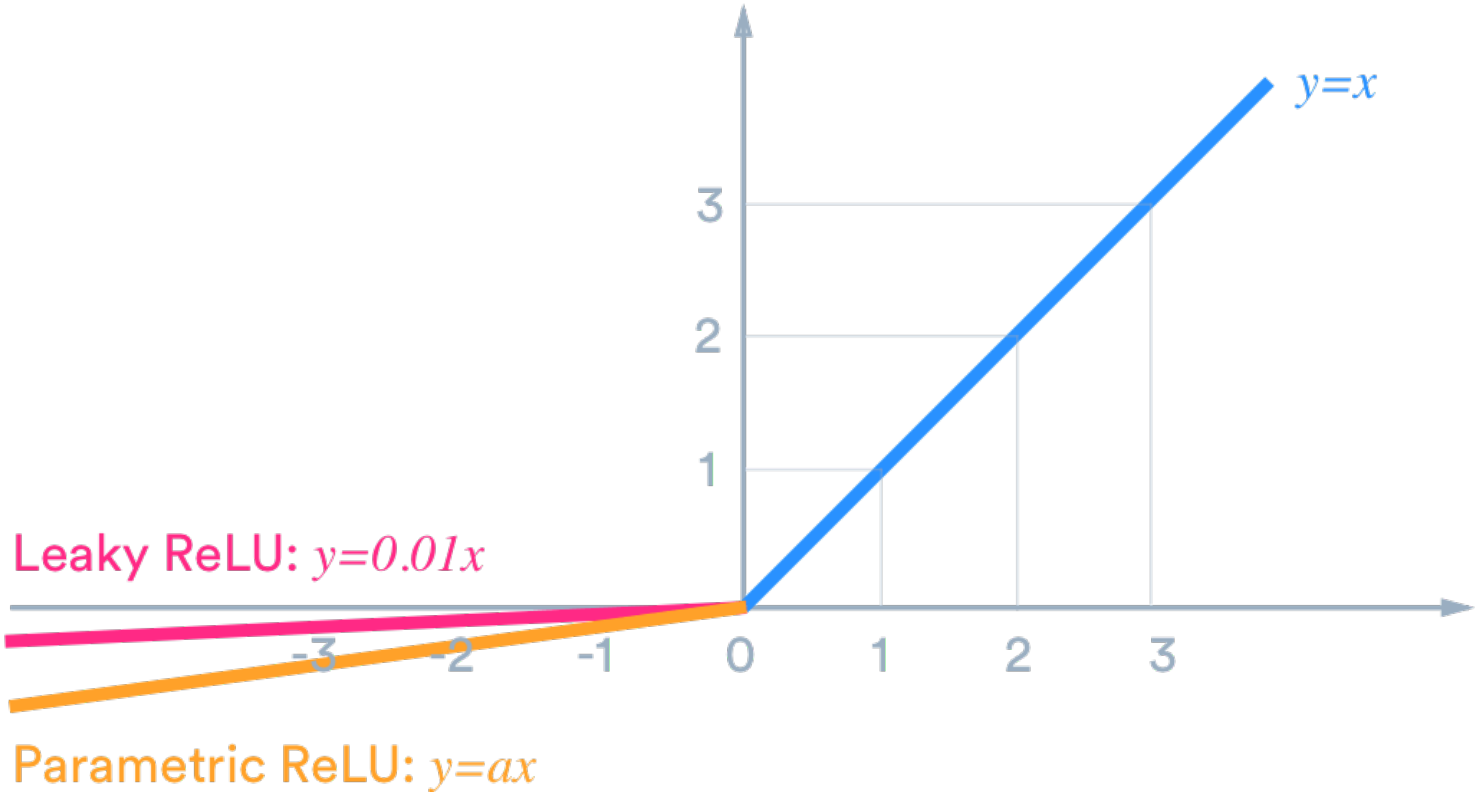


Leaky ReLU

- Advantages
  - Prevents dying ReLU problem—this variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values
  - Otherwise like ReLU
- Disadvantages
  - Results not consistent—leaky ReLU does not provide consistent predictions for negative input values.



Leaky ReLU

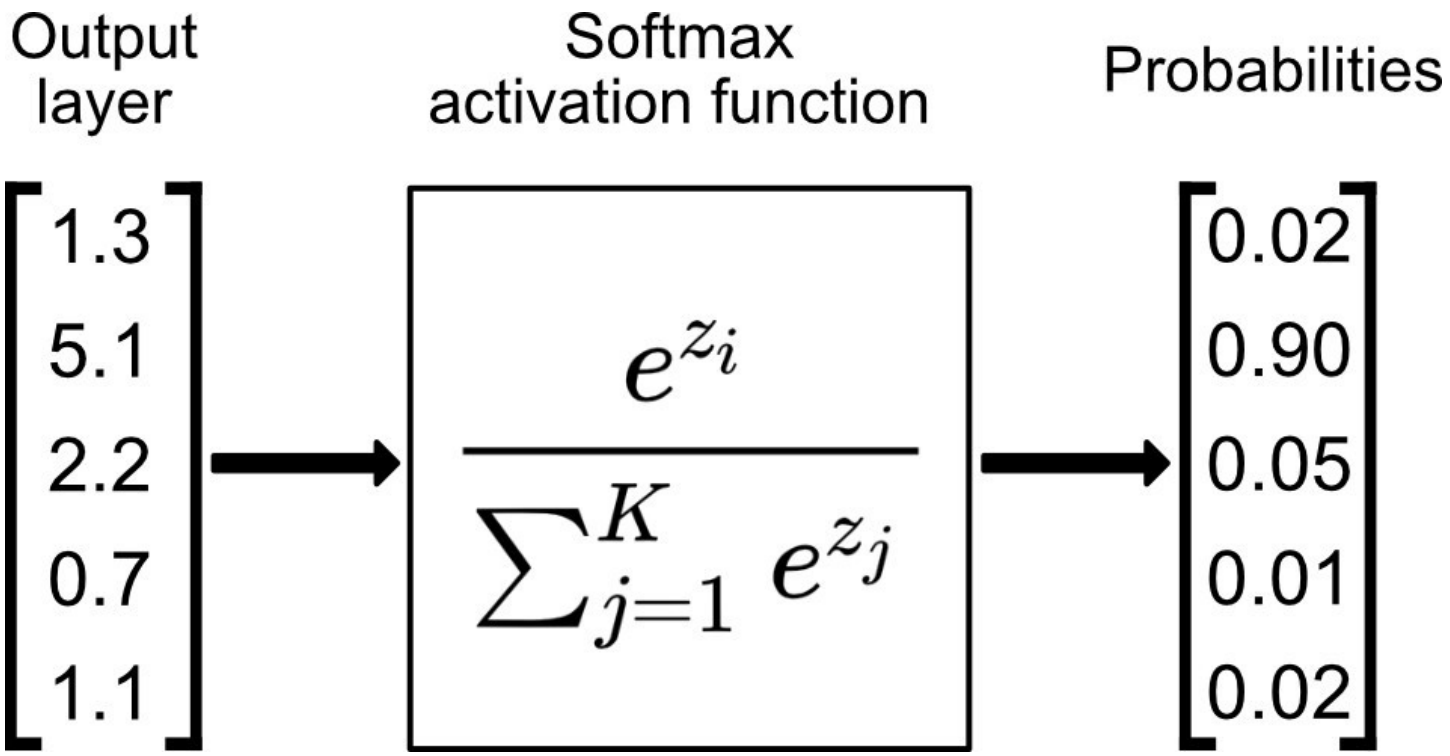


Parametric ReLU

- Advantages
  - Allows the negative slope to be learned—unlike leaky ReLU, this function provides the slope of the negative part of the function as an argument.
  - It is, therefore, possible to perform backpropagation and learn the most appropriate value of α.
  - Otherwise like ReLU
- Disadvantages
  - May perform differently for different problems.

$$f(x)=max(\alpha\partial,\partial)$$

Softmax


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

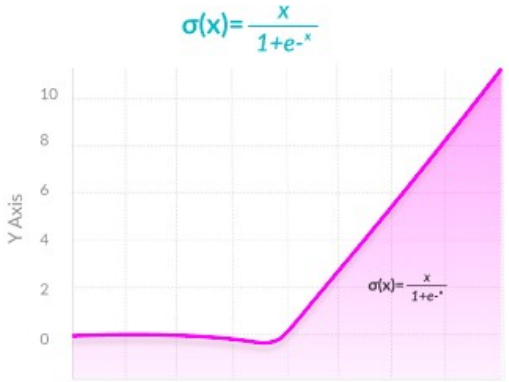
Softmax

- Advantages
  - Able to handle multiple classes only one class in other activation functions—normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class.
  - Useful for output neurons—typically Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories.

$$\sigma(z)j= \frac{ezj}{\sum_{k=1}^{k} ezk} \quad for\ j= 1....k.$$

Swish

- Swish is a new, self-gated activation function discovered by researchers at Google.
- According to their paper, it performs better than ReLU with a similar level of computational efficiency.
- In experiments on ImageNet with identical models running ReLU and Swish, the new function achieved top -1 classification accuracy 0.6-0.9% higher.

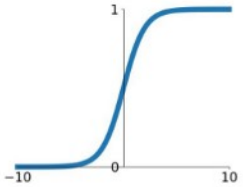


Summary

Activation Functions

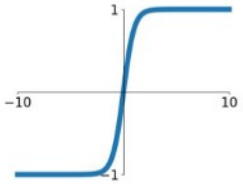
Sigmoid

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



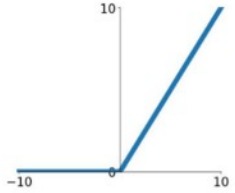
tanh

$$\tanh(x)$$



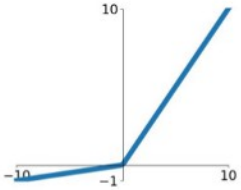
ReLU

$$\max(0, x)$$



Leaky ReLU

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

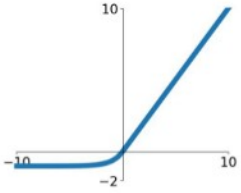


Maxout

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

ELU

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



Challenges

- While selecting and switching activation functions in deep learning frameworks is easy, you will find that managing multiple experiments and trying different activation functions on large test data sets can be challenging.
- It can be difficult to:
  - Track experiment progress
  - Run experiments across multiple machines
  - Manage training data

Let’s Start with an example

Perfect Roommate

Apple pie

Burger

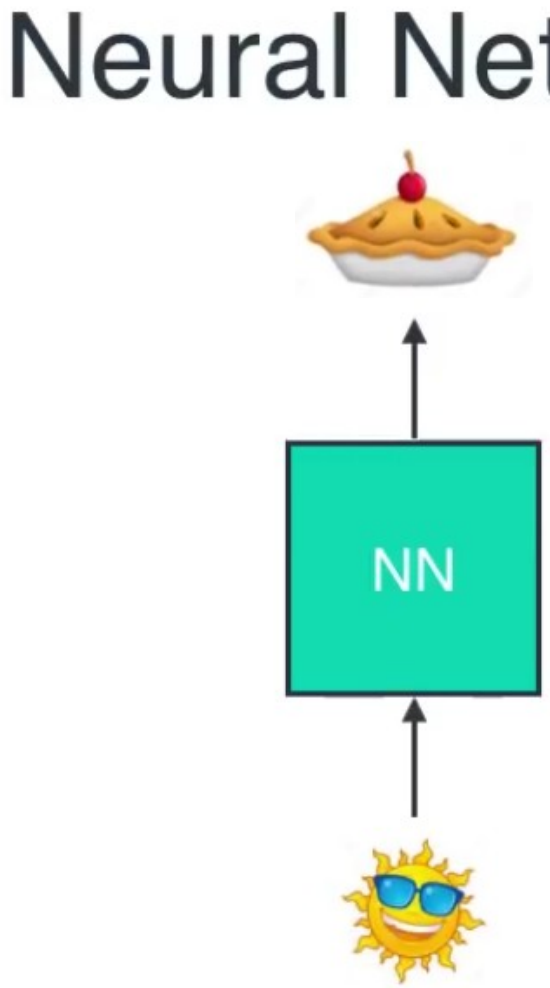
Chicken

Reference: Friendly introduction to RNN by Luis Serrano

Conditional Outputs

Weather

Basic NN



Let’s do some maths

Vectors

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

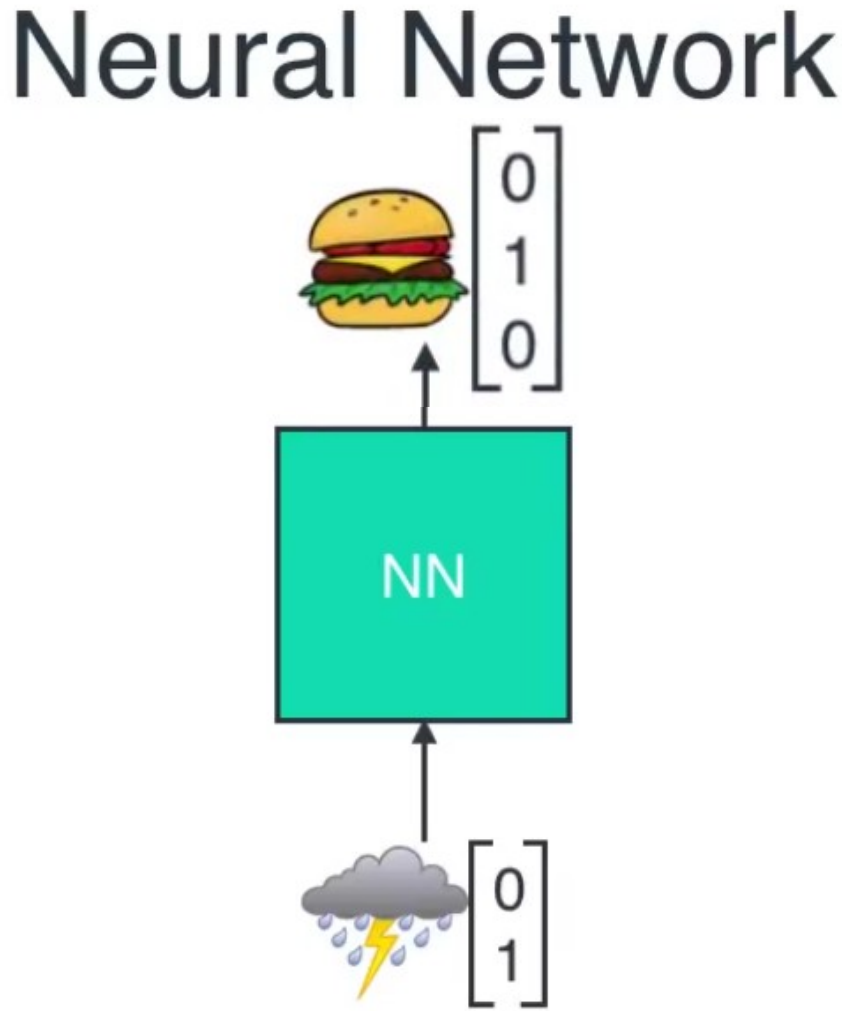
$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

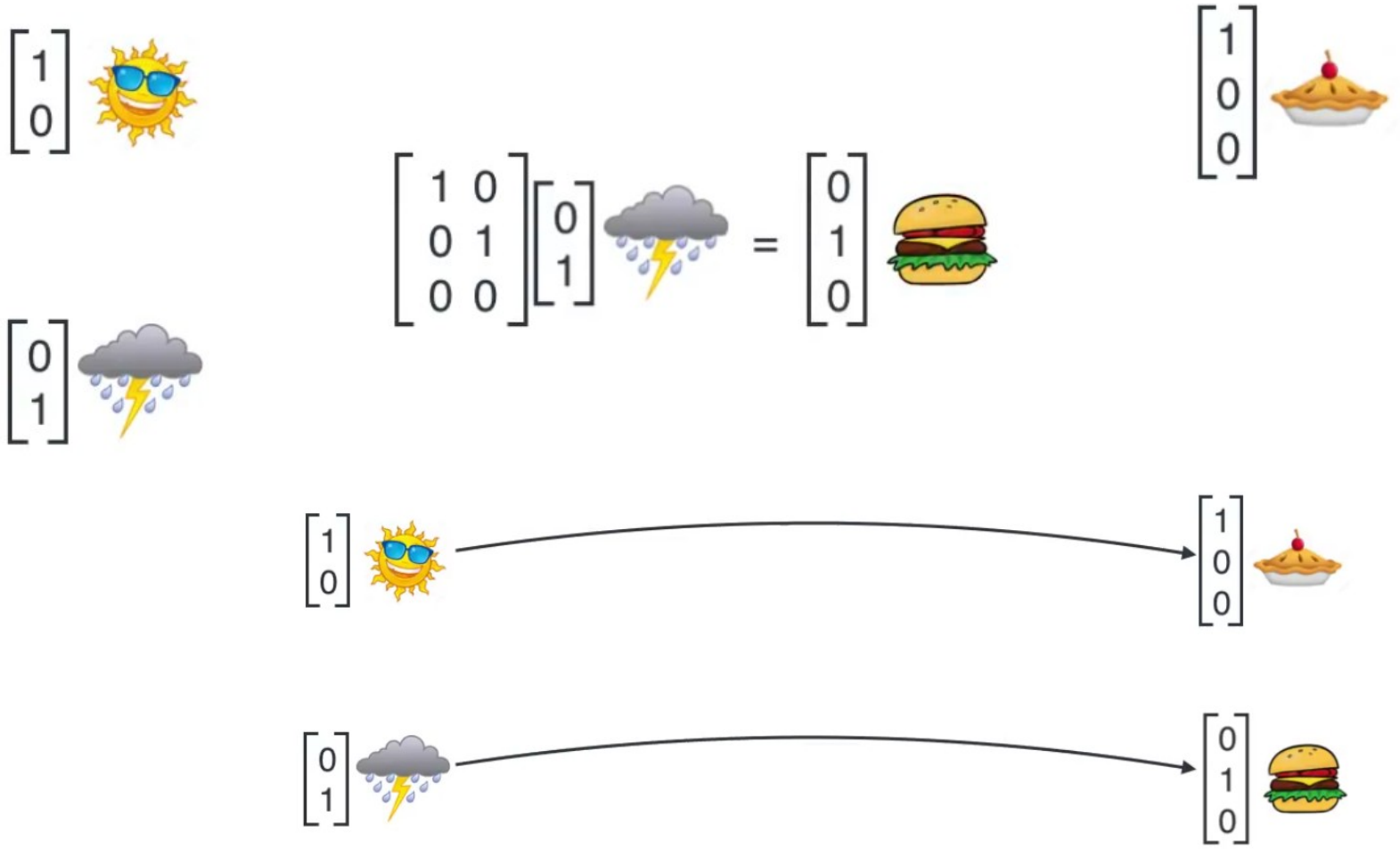
$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Now in NN



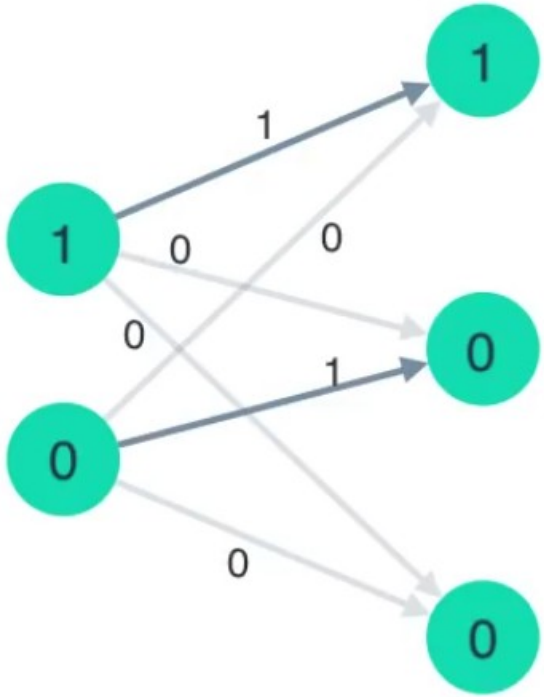
Conceptualizing



Adding to NN

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Useful resources

- https://missinglink.ai
- https://machinelearningmastery.com
- https://www.allaboutcircuits.com
- https://medium.com

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
http://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com