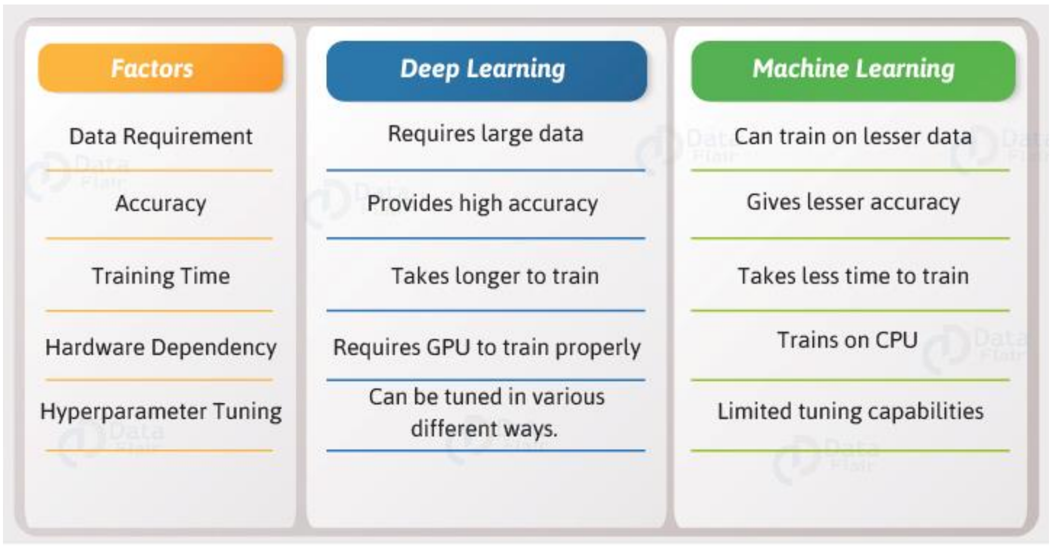
Deep Learning

A type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher-level features from data.

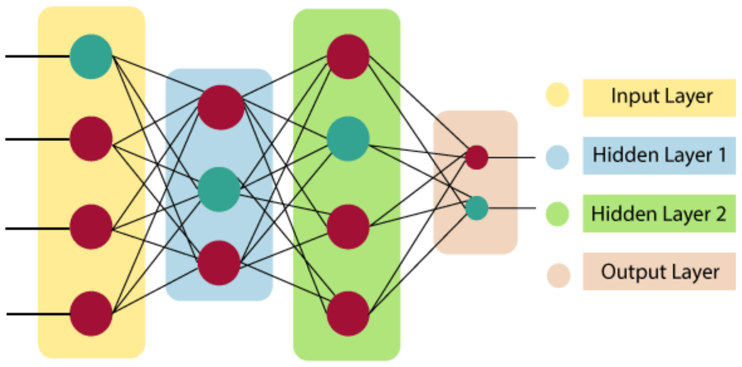
Deep Learning V/S Machine Learning



Factors	Deep Learning	Machine Learning
Data Requirement	Requires large data	Can train on lesser data
Accuracy	Provides high accuracy	Gives lesser accuracy
Training Time	Takes longer to train	Takes less time to train
Hardware Dependency	Requires GPU to train properly	Trains on CPU
Hyperparameter Tuning	Can be tuned in various different ways.	Limited tuning capabilities

Neural Network's

- Also Known as ANN (Artificial Neural Network)
- An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain
- Artificial Neural Network primarily consists of three layers:



- **Input Layer:**
  As the name suggests, it accepts inputs in several different formats provided by the programmer.

- **Hidden Layer:**
  The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.
  Multiple hidden layers is what the term deep learning refers to
- **Output Layer:**
  The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

- **Weights:** weights are the real values that are associated with each feature which tells the importance of that feature in predicting the final value.

- **Bias:** Bias is used for shifting the activation function towards left or right, it can be referred to as a y-intercept in the line equation.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output.

Activation Function:

- **An Activation Function** decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations
- The purpose of an activation function is to add non-linearity to the neural Network. Activation functions introduce an additional step at each layer during the forward propagation, but its computation is worth it.

Let's Suppose we have Neural Network Working without an activation function:

In that case, every neuron will only be performing a linear transformation on the inputs using the weights and biases. It's because it doesn't matter how many hidden layers we attach in the neural network; all layers will behave in the same way because the composition of two linear functions is a linear function itself. Although the neural network becomes simpler, learning any complex task is impossible, and our model would be just a linear regression model.

Choosing the right activation Function:

1. **Regression** – Linear Activation Function
2. **Binary Classification** – Sigmoid or Logistic Activation Function
3. **Multiclass Classification** – SoftMax
4. **Multilabel Classification** – Sigmoid

- **Convolutional Neural Network (CNN)**: ReLU activation function.
- **Recurrent Neural Network**: Tanh or Sigmoid activation function

Learning Rate:

The learning rate is a value that speeds up or slow down how quickly an algorithm learns, it determines the size of steps an algorithm takes when moving towards the global minima, i.e. The lowest error rates.

Types of Neural Networks:

1. Feed forward
2. Feed back

Feedforward:

- With Feedforward neural network signals travel only one way, from input to output
- These types of networks are used extensively in pattern recognition
- Example – CNN

Feedback:

- With feedback neural network signals travel in both directions and there can be loops
- Feedback networks are more powerful and complex
- Example – RNN

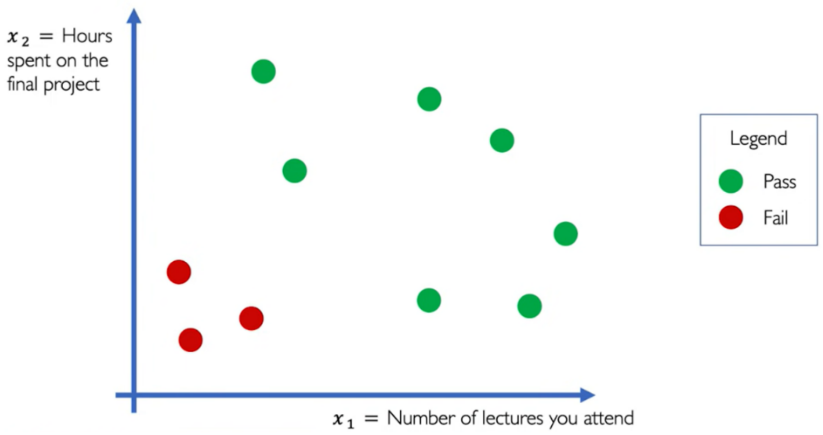
Working of Neural Networks:

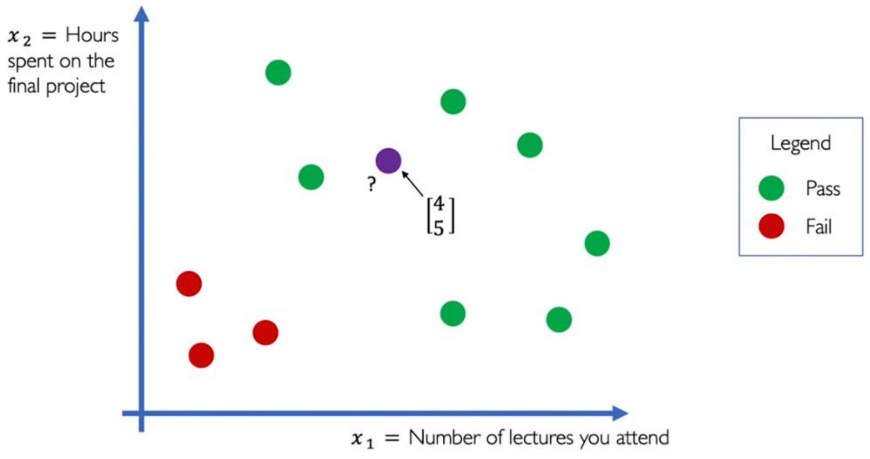
Will I pass this class?

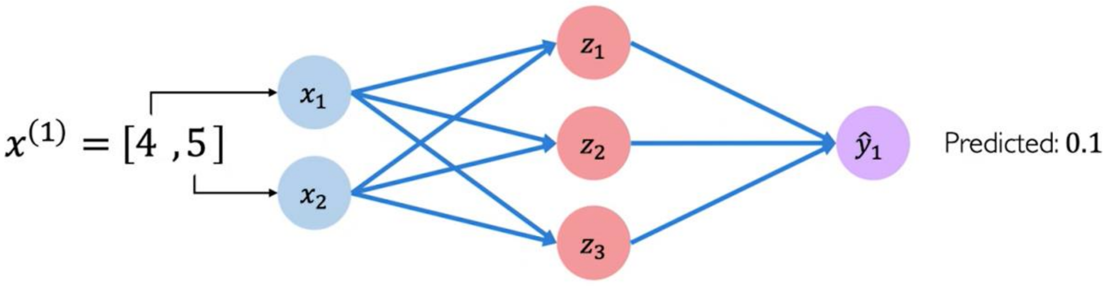
Let's start with a simple two feature model

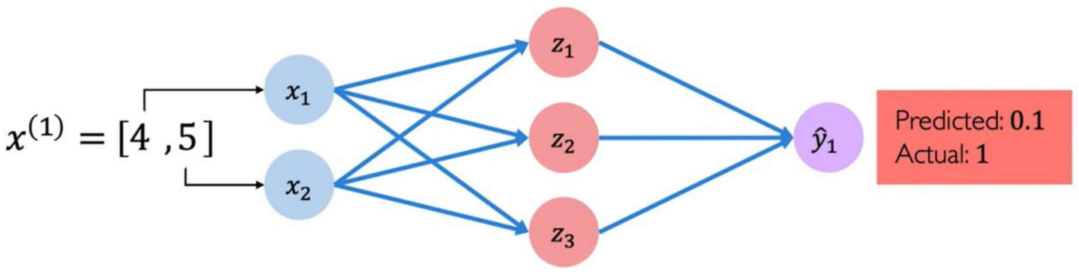
x_1 = Number of lectures you attend

x_2 = Hours spent on the final project

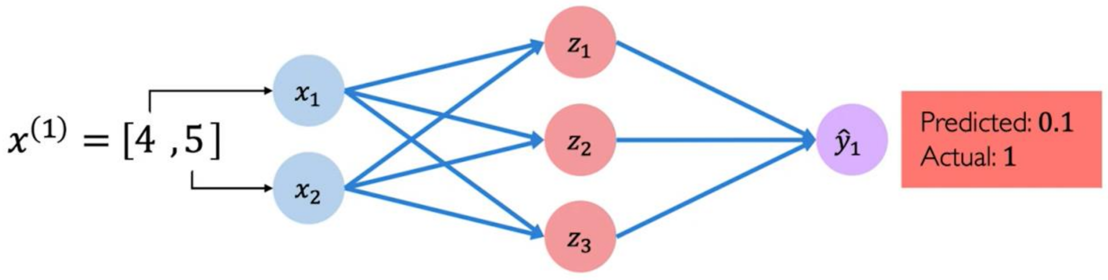






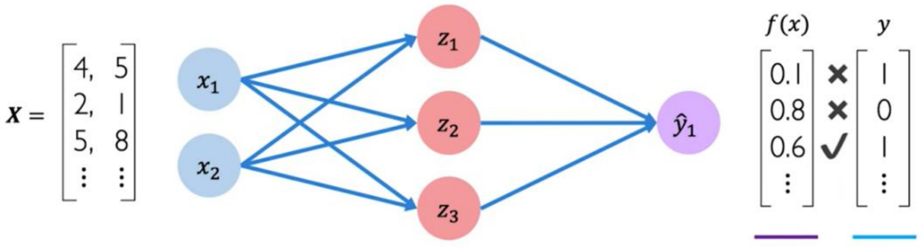


*The **loss** of our network measures the cost incurred from incorrect predictions*



$$\mathcal{L}\left(\underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\right)$$

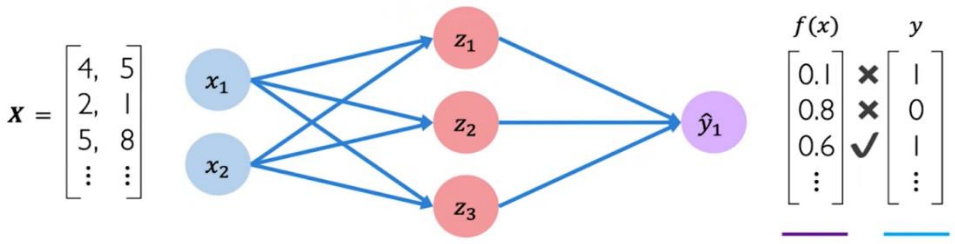
*The **empirical loss** measures the total loss over our entire dataset*



Also known as:
- Objective function
- Cost function
- Empirical Risk

$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\big)$$

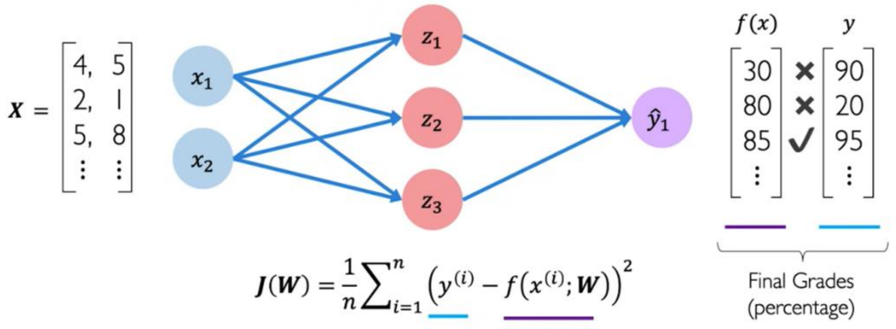
***Cross entropy loss** can be used with models that output a probability between 0 and 1*



$$J(\boldsymbol{W}) = -\frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\Big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big) + (1-\underbrace{y^{(i)}}_{\text{Actual}}) \log\Big(1-\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)$$

```
loss = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(y, predicted) )
```

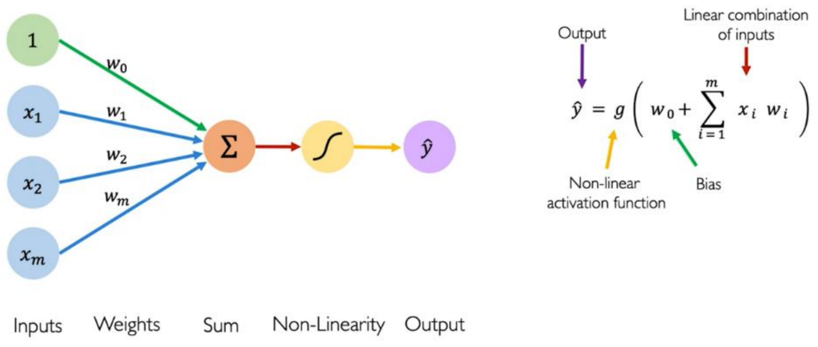
***Mean squared error loss** can be used with regression models that output continuous real numbers*



$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \Big(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)^2$$

```
loss = tf.reduce_mean( tf.square(tf.subtract(y, predicted)) )
loss = tf.keras.losses.MSE( y, predicted )
```

Summary

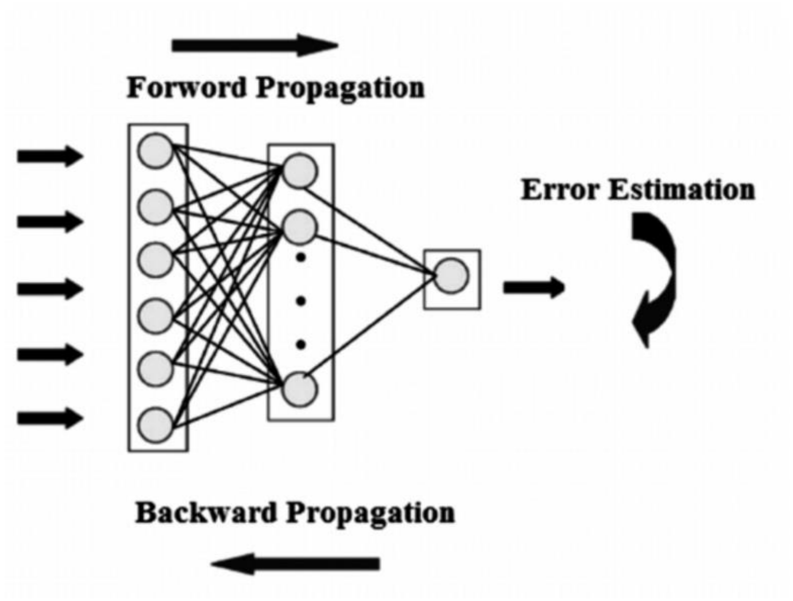

1
w_0
x_1
w_1
w_2
x_2
w_m
x_m
Σ
$\hat{y}$
Inputs
Weights
Sum
Non-Linearity
Output
Output
Linear combination of inputs
$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right)$
Non-linear activation function
Bias

In neural networks, you basically do forward-propagation to get the output of your model and check if this output is correct or incorrect, to get the error. Backpropagation is nothing but going backwards through your neural network to find the partial derivatives of the error with respect to the weights, which enables you to subtract this value from the weights.

Those derivatives are then used by gradient descent, an algorithm that can iteratively minimize a given function. Then it adjusts the weights up or down, depending on which decreases the error. That is exactly how a neural network learns during the training process.

So, with backpropagation you basically try to tweak the weights of your model while training.

The image below illustrates the concept of forward propagation and backpropagation in a feed-forward neural network:



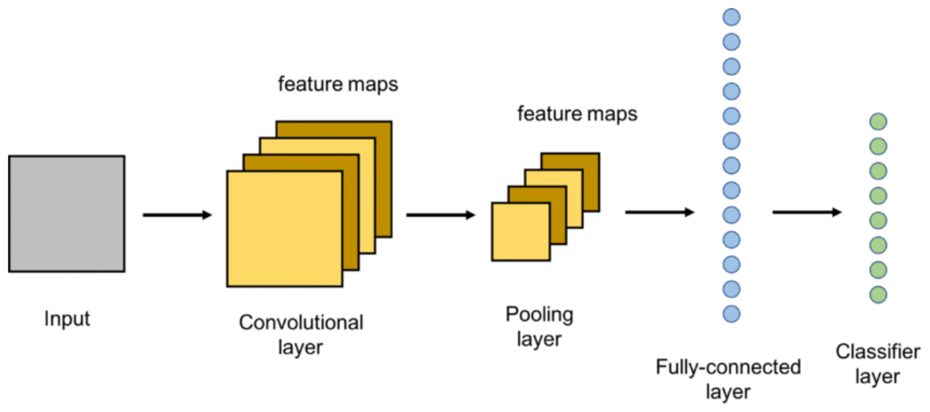
For Image classification ANN is not preferred as the output of ANN is a single dimensional array in which we have the pixel values to overcome this drawback CNN is preferred for image classification.

Convolution Neural Network (CNN)

Convolutional Neural Network is used to do image classification and image recognition in neural networks. Scene labelling, objects detections, and face recognition, etc., are some of the areas where convolutional neural networks are widely used.

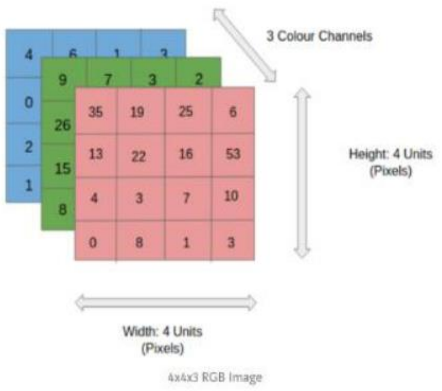
Convolutional Neural Network have several types of layers:

1. Input Layer
2. Convolution Layer
3. Pooling Layer
4. Fully Connected Layer



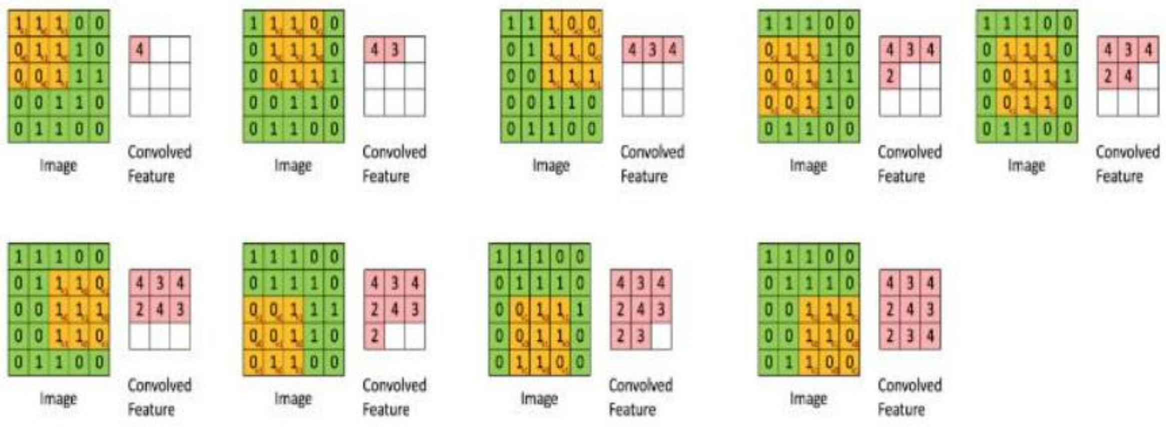
Input Layer

In this figure we have a RGB image which have been separated by three colour planes, Red, Green and Blue



Convolution Layer:

Convolution layer extract features from an input image. By learning image features using a small square of input data. It is a mathematical operation which takes two inputs such as image matrix and a kernel or filter



- **Strides**

Stride is the number of pixels which are shift over the input matrix. When the stride is equalled to 1, then we move the filters to 1 pixel at a time and similarly, if the stride is equalled to 2, then we move the filters to 2 pixels at a time.

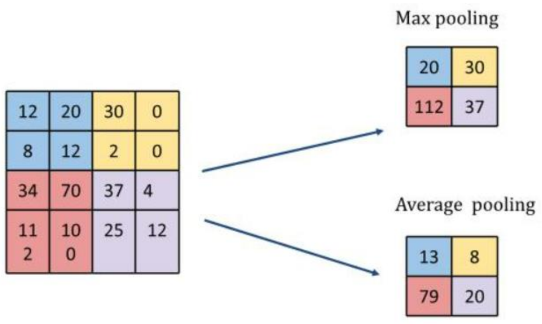
In the above Image we have used stride = 1

Pooling Layer:

Reduces the amount of information in each feature obtained in the convolution layer while maintaining the most important information

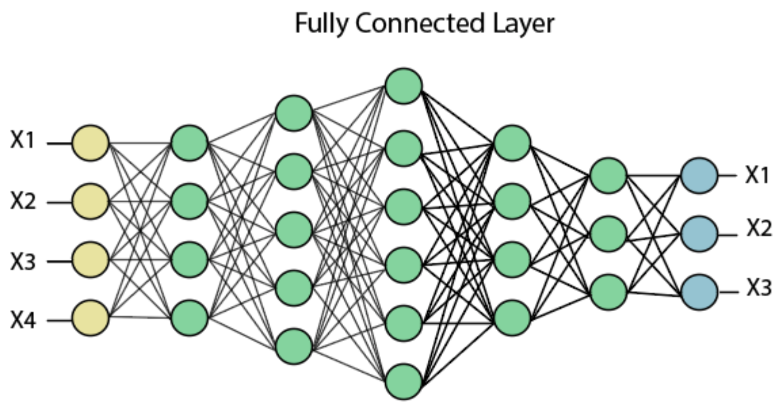
Two common functions used in pooling operation are:

1. Average pooling – Calculate the average value for each patch on the feature map
2. Max pooling - Calculate the maximum value for each patch on the feature map



Fully Connected Layer

The fully connected layer is a layer in which the input from the other layers will be flattened into a vector and sent. It will transform the output into the desired number of classes by the network.



In the above diagram, the feature map matrix will be converted into the vector such as x1, x2, x3... xn with the help of fully connected layers. We will combine features to create a model and apply the activation function such as softmax or sigmoid to classify the outputs as a car, dog, truck, etc.

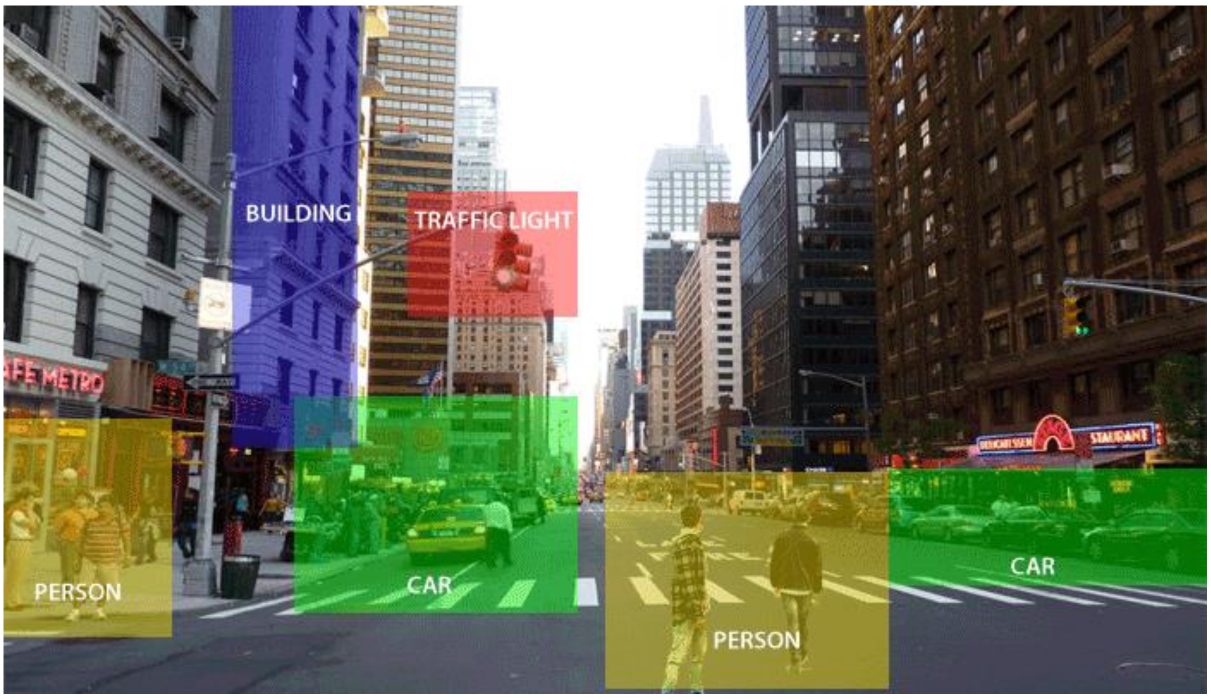
OpenCV (Computer Vision)

What is OpenCV?

OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc.

In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.

The purpose of computer vision is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, and three-dimension model, and so on. For example, cars can be facilitated with computer vision, which will be able to identify and different objects around the road, such as traffic lights, pedestrians, traffic signs, and so on, and acts accordingly.



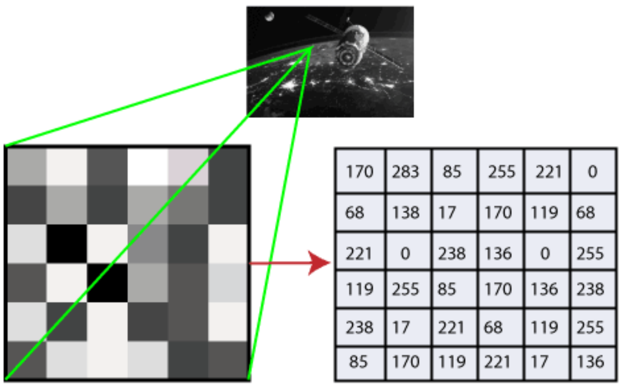
Computer vision allows the computer to perform the same kind of tasks as humans with the same efficiency. There are a two main task which are defined below:

- **Object Classification** - In the object classification, we train a model on a dataset of particular objects, and the model classifies new objects as belonging to one or more of your training categories.
- **Object Identification** - In the object identification, our model will identify a particular instance of an object - for example, parsing two faces in an image and tagging one as face1 and other one as face 2.

How does computer recognize the image?

Machines convert the vision into numbers and store in the memory. Here the question arises how computer convert images into numbers. So, the answer is that the pixel value is used to convert images into numbers.

A pixel is the smallest unit of a digital image or graphics that can be displayed and represented on a digital display device.



170	283	85	255	221	0
68	138	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

The picture intensity at the particular location is represented by the numbers. In the above image, we have shown the pixel values for a grayscale image consist of only one value, the intensity of the black colour at that location.

Natural Language Processing (NLP)

- NLP stands for Natural Language Processing, which is a part of Computer Science, Human language, and Artificial Intelligence.
- It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages.
- It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation.

Components of NLP

There are the following two components of NLP -

1. Natural Language Understanding (NLU)

Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

2. Natural Language Generation (NLG)

Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

Difference between NLU and NLG

NLU	NLG
NLU is the process of reading and interpreting language.	NLG is the process of writing or generating language.
It produces non-linguistic outputs from natural language inputs.	It produces constructing natural language outputs from non-linguistic inputs.

Applications of NLP

1. Question Answering → Alexa, Siri
2. Spam Detection → Spam detection is used to detect unwanted e-mails getting to a user's inbox.
3. Sentiment Analysis
4. Machine Translation → Google Translator
5. Spelling correction
6. Speech Recognition
7. Chatbot

There are the following steps to build an NLP pipeline -

Step1: Sentence Segmentation

Sentence Segment is the first step for building the NLP pipeline. It breaks the paragraph into separate sentences.

Independence Day is one of the important festivals for every Indian citizen. It is celebrated on the 15th of August each year ever since India got independence from the British rule. The day celebrates independence in the true sense.

Sentence Segment produces the following result:

1. "Independence Day is one of the important festivals for every Indian citizen."
2. "It is celebrated on the 15th of August each year ever since India got independence from the British rule."
3. "This day celebrates independence in the true sense."

Step2: Word Tokenization

Word Tokenizer is used to break the sentence into separate words or tokens.

Example: JavaTpoint offers Corporate Training, Summer Training, Online Training, and Winter Training.

Word Tokenizer generates the following result:

"JavaTpoint", "offers", "Corporate", "Training", "Summer", "Training", "Online", "Training", "and", "Winter", "Training", "."

Step3: Stemming

Stemming is used to normalize words into its base form or root form. For example, celebrates, celebrated and celebrating, all these words are originated with a single root word "celebrate." The big problem with stemming is that sometimes it produces the root word which may not have any meaning.

For Example, intelligence, intelligent, and intelligently, all these words are originated with a single root word "intelligen." In English, the word "intelligen" do not have any meaning.

Step 4: Lemmatization

Lemmatization is quite similar to the Stemming. It is used to group different inflected forms of the word, called Lemma. The main difference between Stemming and lemmatization is that it produces the root word, which has a meaning.

For example: In lemmatization, the words intelligence, intelligent, and intelligently has a root word intelligent, which has a meaning.

Step 5: Identifying Stop Words

In English, there are a lot of words that appear very frequently like "is", "and", "the", and "a". NLP pipelines will flag these words as stop words. Stop words might be filtered out before doing any statistical analysis.

Example: He is a good boy, Becomes → good boy

Step 6: Dependency Parsing

Dependency Parsing is used to find that how all the words in the sentence are related to each other.

Step 7: POS tags

POS stands for parts of speech, which includes Noun, verb, adverb, and Adjective. It indicates that how a word functions with its meaning as well as grammatically within the sentences. A word has one or more parts of speech based on the context in which it is used.

Example: "Google" something on the Internet.

In the above example, Google is used as a verb, although it is a proper noun.

Step 8: Named Entity Recognition (NER)

Named Entity Recognition (NER) is the process of detecting the named entity such as person name, movie name, organization name, or location.

Example: Steve Jobs introduced iPhone at the Macworld Conference in San Francisco, California.

Step 9: Chunking

Chunking is used to collect the individual piece of information and grouping them into bigger pieces of sentences.

Creating Vectors from Text:

1. **Bag of Words (BoW) Model**

The Bag of Words (BoW) model is the simplest form of text representation in numbers. Like the term itself, we can represent a sentence as a bag of words vector (a string of numbers).

Example:

- Review 1: This movie is very scary and long
- Review 2: This movie is not scary and is slow
- Review 3: This movie is spooky and good

We will first build a vocabulary from all the unique words in the above three reviews. The vocabulary consists of these 11 words:

‘This’, ‘movie’, ‘is’, ‘very’, ‘scary’, ‘and’, ‘long’, ‘not’, ‘slow’, ‘spooky’, ‘good’.

We can now take each of these words and mark their occurrence in the three movie reviews above with 1s and 0s. This will give us 3 vectors for 3 reviews:

	1 This	**2 movie**	**3 is**	**4 very**	**5 scary**	**6 and**	**7 long**	**8 not**	**9 slow**	**10 spooky**	**11 good**	**Length of the review(in words)**
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Vector of Review 1: [1 1 1 1 1 1 1 0 0 0 0]

Vector of Review 2: [1 1 2 0 0 1 1 0 1 0 0]

Vector of Review 3: [1 1 1 0 0 0 1 0 0 1 1]

And that's the core idea behind a Bag of Words (BoW) model.

- **Drawbacks of using a Bag-of-Words (BoW) Model**

In the above example, we can have vectors of length 11. However, we start facing issues when we come across new sentences:

1. If the new sentences contain new words, then our vocabulary size would increase and thereby, the length of the vectors would increase too.
2. We are retaining no information on the grammar of the sentences nor on the ordering of the words in the text.

Term Frequency-Inverse Document Frequency (TF-IDF)

"Term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus."

Term Frequency (TF)

It is a measure of how frequently a term, t, appears in a document, d:

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

Here, in the numerator, n is the number of times the term "t" appears in the document "d". Thus, each document and term would have its own TF value.

We will again use the same vocabulary we had built in the Bag-of-Words model to show how to calculate the TF for Review #2:

Review 2: This movie is not scary and is slow

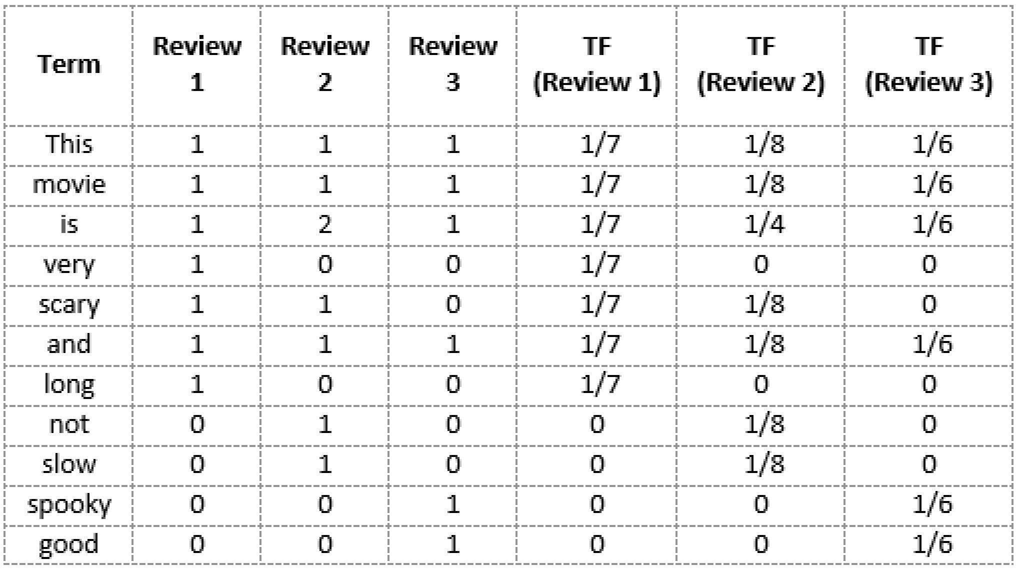
Here,

- Vocabulary: 'This', 'movie', 'is', 'very', 'scary', 'and', 'long', 'not', 'slow', 'spooky', 'good'
- Number of words in Review 2 = 8
- TF for the word 'this' = (number of times 'this' appears in review 2)/(number of terms in review 2) = 1/8

Similarly,

- TF('movie') = 1/8
- TF('is') = 2/8 = 1/4
- TF('very') = 0/8 = 0
- TF('scary') = 1/8
- TF('and') = 1/8
- TF('long') = 0/8 = 0
- TF('not') = 1/8
- TF('slow') = 1/8
- TF('spooky') = 0/8 = 0
- TF('good') = 0/8 = 0

We can calculate the term frequencies for all the terms and all the reviews in this manner:



Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

Inverse Document Frequency (IDF)

IDF is a measure of how important a term is. We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words:

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

We can calculate the IDF values for the all the words in Review 2:

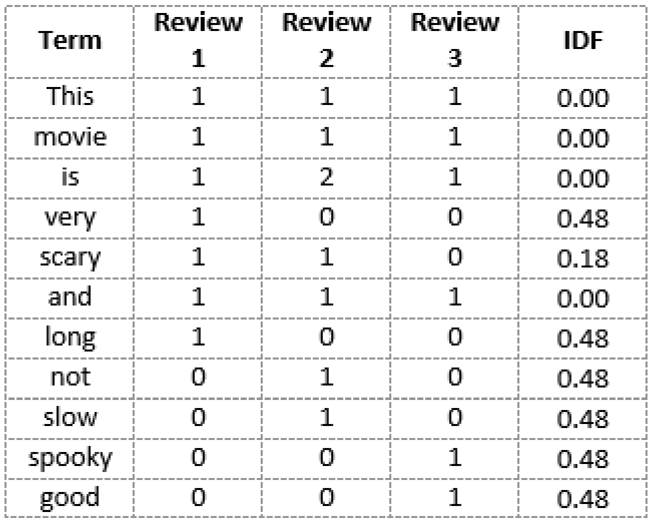
IDF(‘this’) = log (number of documents/number of documents containing the word ‘this’) =

log (3/3) = log (1) = 0

Similarly,

- IDF (‘movie’,) = log (3/3) = 0
- IDF(‘is’) = log (3/3) = 0
- IDF(‘not’) = log (3/1) = log (3) = 0.48
- IDF(‘scary’) = log (3/2) = 0.18
- IDF(‘and’) = log (3/3) = 0
- IDF(‘slow’) = log (3/1) = 0.48

We can calculate the IDF values for each word like this. Thus, the IDF values for the entire vocabulary would be:



Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	1	0.48

Hence, we see that words like “is”, “this”, “and”, etc., are reduced to 0 and have little importance; while words like “scary”, “long”, “good”, etc. are words with more importance and thus have a higher value.

We can now compute the TF-IDF score for each word in the corpus. Words with a higher score are more important, and those with a lower score are less important:

$$(tf\_idf)_{t,d} = tf_{t,d} * idf_t$$

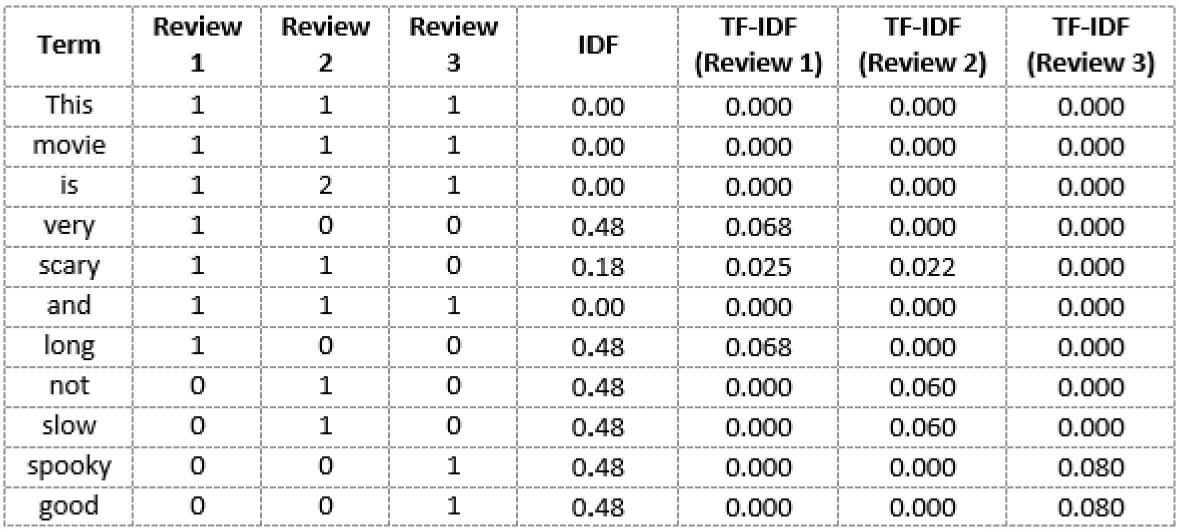
We can now calculate the TF-IDF score for every word in Review 2:

TF-IDF (‘this’, Review 2) = TF (‘this’, Review 2) * IDF(‘this’) = 1/8 * 0 = 0

Similarly,

- TF-IDF (‘movie’, Review 2) = 1/8 * 0 = 0
- TF-IDF (‘is’, Review 2) = 1/4 * 0 = 0
- TF-IDF (‘not’, Review 2) = 1/8 * 0.48 = 0.06
- TF-IDF (‘scary’, Review 2) = 1/8 * 0.18 = 0.023
- TF-IDF (‘and’, Review 2) = 1/8 * 0 = 0
- TF-IDF (‘slow’, Review 2) = 1/8 * 0.48 = 0.06

Similarly, we can calculate the TF-IDF scores for all the words with respect to all the reviews:



Term	Review 1	Review 2	Review 3	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	0.18	0.025	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0.48	0.000	0.060	0.000
slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
good	0	0	1	0.48	0.000	0.000	0.080

We have now obtained the TF-IDF scores for our vocabulary. TF-IDF also gives larger values for less frequent words and is high when both IDF and TF values are high i.e. the word is rare in all the documents combined but frequent in a single document.

Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a special type of an artificial neural network adapted to work for time series data or data that involves sequences.

Ordinary feed forward neural networks are only meant for data points, which are independent of each other. However, if we have data in a sequence such that one data point depends upon the previous data point, we need to modify the neural network to incorporate the dependencies between these data points.

RNNs have the concept of 'memory' that helps them store the states or information of previous inputs to generate the next output of the sequence.

Sequential data is basically just ordered data in which related things follow each other. Examples are financial data or the DNA sequence. The most popular type of sequential data is perhaps time series data, which is just a series of data points that are listed in time order.

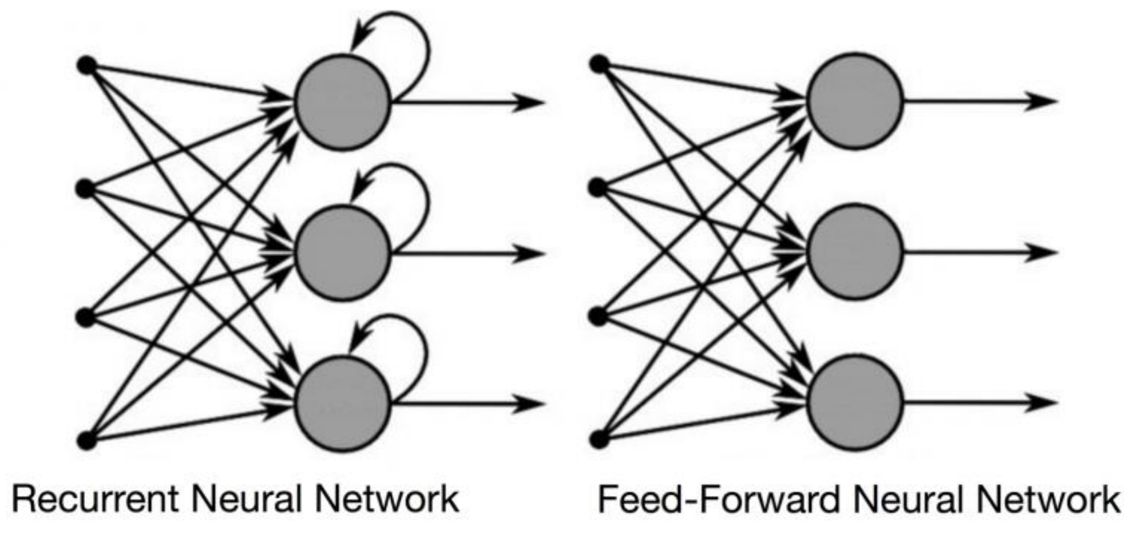
RNN VS. FEED-FORWARD NEURAL NETWORKS

RNN's and feed-forward neural networks get their names from the way they channel information.
In a feed-forward neural network, the information only moves in one direction — from the input layer, through the hidden layers, to the output layer. The information moves straight through the network and never touches a node twice.

Feed-forward neural networks have no memory of the input they receive and are bad at predicting what's coming next. Because a feed-forward network only considers the current input, it has no notion of order in time. It simply can't remember anything about what happened in the past except its training.

In a RNN the information cycles through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously.

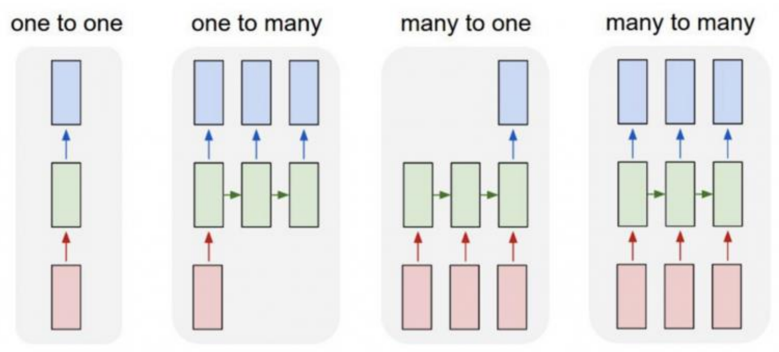
The two images below illustrate the difference in information flow between a RNN and a feed-forward neural network.



- **A usual RNN has a short-term memory**
- **Recurrent neural networks add the immediate past to the present.**
- **RNN has two inputs: the present and the recent past**

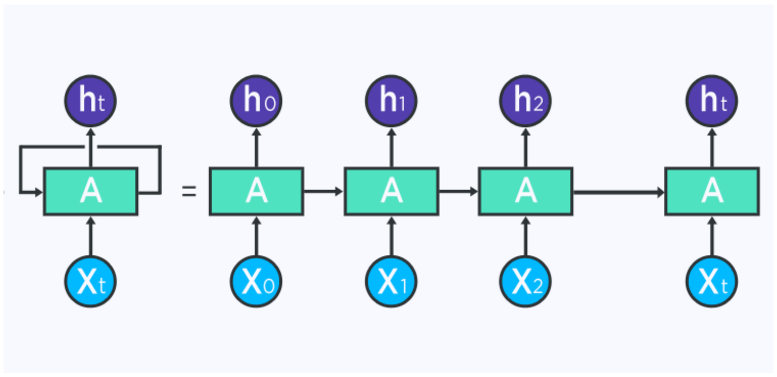
TYPES OF RNNS

- One to One
- One to Many
- Many to One
- Many to Many



Backpropagation Through Time

The image below illustrates an unrolled RNN. On the left, the RNN is unrolled after the equal sign. Note there is no cycle after the equal sign since the different time steps are visualized and information is passed from one time step to the next. This illustration also shows why a RNN can be seen as a sequence of neural networks.



If you do BPTT, the conceptualization of unrolling is required since the error of a given timestep depends on the previous time step.

Within BPTT the error is backpropagated from the last to the first timestep, while unrolling all the timesteps. This allows calculating the error for each timestep, which allows updating the weights. Note that BPTT can be computationally expensive when you have a high number of timesteps.

Two issues of standard RNN's

EXPLODING GRADIENTS

Exploding gradients are when the algorithm, without much reason, assigns a stupidly high importance to the weights.

VANISHING GRADIENTS

Vanishing gradients occur when the values of a gradient are too small and the model stops learning or takes way too long as a result