**What is Unsupervised Learning?**

Unsupervised Learning, as discussed earlier, can be thought of as **self-learning** where the algorithm can find previously **unknown patterns** in datasets that do not have any sort of labels. It helps in modelling probability density functions, finding anomalies in the data, and much more. To give you a simple example, think of a student who has textbooks and all the required material to study but has no teacher to guide. Ultimately, the student will have to learn by himself or herself to pass the exams. This sort of self-learning is what we have scaled into Unsupervised Learning for machines.

## ****Why is it important?****

So what does Unsupervised Learning help us obtain? Let me tell you all about it.

* Unsupervised Learning algorithms work on datasets that are unlabelled and find patterns which would previously not be known to us.
* These patterns obtained are helpful if we need to categorize the elements or find an [association](https://www.edureka.co/blog/apriori-algorithm/) between them.
* They can also help detect anomalies and defects in the data which can be taken care of by us.

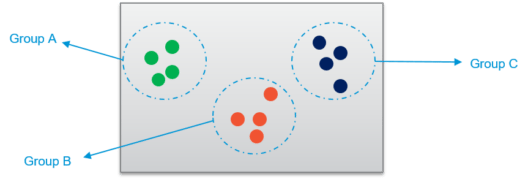
Lastly and most importantly, data which we collect is usually unlabelled which makes work easier for us when we use these algorithms.

**Types of Unsupervised Learning**

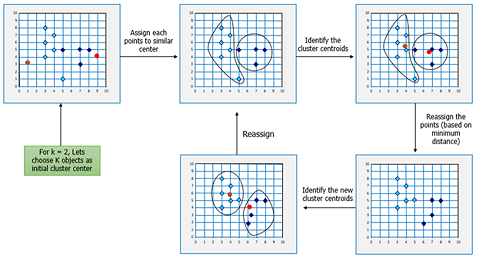
Unsupervised Learning has been split up majorly into 2 types:

* **Clustering**
* **Association**

Clustering is the type of Unsupervised Learning where you find patterns in the data that you are working on. It may be the shape, size, colour etc. which can be used to group data items or create clusters. Some popular algorithms in Clustering are discussed below:

****

* **Hierarchical Clustering** – This algorithm builds clusters based on the similarity between different data points in the dataset. It goes over the various features of the data points and looks for the similarity between them. If the data points are found to be similar, they are grouped together. This continues until the dataset has been grouped which creates a hierarchy for each of these clusters.
* [**K-Means Clustering**](https://www.edureka.co/blog/k-means-clustering-algorithm/) – This algorithm works step-by-step where the main goal is to achieve clusters which have labels to identify them. The algorithm creates clusters of different data points which are as homogenous as possible by calculating the centroid of the cluster and making sure that the distance between this centroid and the new data point is as less as possible. The smallest distance between the data point and the centroid determines which cluster it belongs to while making sure the clusters do not interlay with each other. The centroid acts like the heart of the cluster. This ultimately gives us the cluster which can be labelled as needed.



[**K-NN Clustering**](https://www.edureka.co/blog/knn-algorithm-in-r/)**–**This is probably the most simple of the Machine Learning algorithms as the algorithm does not really learn but rather classifies the new data point based on the datasets that have been stored by it. This algorithm is also called as a lazy learner because it learns only when the algorithm is given a new data point. It works well with smaller datasets as huge datasets take time to learn.

Association is the kind of Unsupervised Learning where you find the dependencies of one data item to another data item and map them such that they help you profit better. Some popular algorithms in Association Rule Mining are discussed below:



* [**Apriori algorithm**](https://www.edureka.co/blog/apriori-algorithm/)**–**The Apriori Algorithm is a breadth-first search based which calculates the support between items. This support basically maps the dependency of one data item with another which can help us understand what data item influences the possibility of something happening to the other data item. For example, bread influences the buyer to buy milk and eggs. So that mapping helps increase profits for the store. That sort of mapping can be learnt using this algorithm which yields rules as for its output.
* **FP-Growth Algorithm –**The Frequency Pattern (FP) algorithm finds the count of the pattern that has been repeated, adds that to a table and then finds the most plausible item and sets that as the root of the tree. Other data items are then added into the tree and the support is calculated. If that particular branch fails to meet the threshold of the support, it is pruned. Once all the iterations are completed, a tree with the root to the item will be created which are then used to make the rules of the association. This algorithm is faster than Apriori as the support is calculated and checked for increasing iterations rather than creating a rule and checking the support from the dataset.

**Applications of Unsupervised Learning**

Unsupervised Learning helps in a variety of ways which can be used to solve various real-world problems.

* They help us in understanding patterns which can be used to cluster the data points based on various features.
* Understanding various defects in the dataset which we would not be able to detect initially.
* They help in mapping the various items based on the dependencies of each other.
* Cleansing the datasets by removing features which are not really required for the machine to learn from.

This ultimately leads to applications which are helpful to us. Certain examples of where Unsupervised Learning algorithms are used are discussed below:

* **AirBnB** – This is a great application which helps host stays and experiences connecting people all over the world. This application uses Unsupervised Learning where the user queries his or her requirements and Airbnb learns these patterns and recommends stays and experiences which fall under the same group or cluster.
* **Amazon** – Amazon also uses unsupervised learning to learn the customer’s purchase and recommend the products which are most frequently bought together which is an example of association rule mining.
* **Credit-Card Fraud Detection** – Unsupervised Learning algorithms learn about various patterns of the user and their usage of the credit card. If the card is used in parts that do not match the behaviour, an alarm is generated which could possibly be marked fraud and calls are given to you to confirm whether it was you using the card or not.

Those were some of the applications where Unsupervised Learning algorithms have shined and shown their grit. Now that we have finished the applications of Unsupervised Learning, let’s move ahead to the differences between Supervised and Unsupervised Learning.

**Disadvantages of Unsupervised Learning**

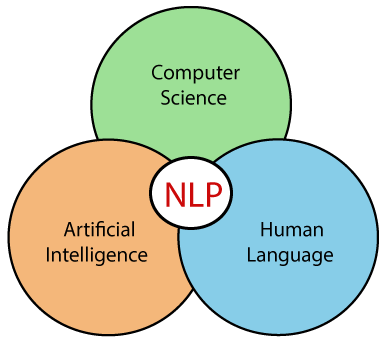
Even though Unsupervised Learning is used in many well-known applications and works brilliantly, there are still many disadvantages to it.

* There is no way of obtaining the way or method the data is sorted as the dataset is unlabelled.
* They may be less accurate as the input data is not known and labelled by the humans making the machine do it.
* The information obtained by the algorithm may not always correspond to the output class that we required.
* The user has to understand and map the output obtained with the corresponding labels.

**NLP**

## What is 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**.



Advantages of NLP

* NLP helps users to ask questions about any subject and get a direct response within seconds.
* NLP offers exact answers to the question means it does not offer unnecessary and unwanted information.
* NLP helps computers to communicate with humans in their languages.
* It is very time efficient.
* Most of the companies use NLP to improve the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

Disadvantages of NLP

A list of disadvantages of NLP is given below:

* NLP may not show context.
* NLP is unpredictable
* NLP may require more keystrokes.
* NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

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.

NLU involves the following tasks -

* It is used to map the given input into useful representation.
* It is used to analyze different aspects of the 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**
  2. **Spam Detection**
  3. **Sentiment Analysis**
  4. **Machine Translation**
  5. **Spelling correction**
  6. **Speech Recognition**
  7. **Chatbot**
  8. **Information extraction**
  9. **Natural Language Understanding (NLU**

## How to build an NLP pipeline

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.

**Example:** Consider the following paragraph -

**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 Stamming. 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.

**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.

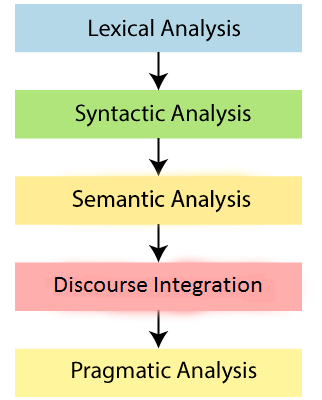
**Example: Steve Jobs** introduced iPhone at the Macworld Conference in San Francisco, California.

**Step 9: Chunking**

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

## Phases of NLP

There are the following five phases of NLP:



**1. Lexical Analysis and Morphological**

The first phase of NLP is the Lexical Analysis. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. It divides the whole text into paragraphs, sentences, and words.

**2. Syntactic Analysis (Parsing)**

Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

**Example:** Agra goes to the Poonam

In the real world, Agra goes to the Poonam, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

**3. Semantic Analysis**

Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases, and sentences.

**4. Discourse Integration**

Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

**5. Pragmatic Analysis**

Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

**For Example:** "Open the door" is interpreted as a request instead of an order.

Why NLP is difficult?

NLP is difficult because Ambiguity and Uncertainty exist in the language.

**Ambiguity**

There are the following three ambiguity -

* **Lexical Ambiguity**

Lexical Ambiguity exists in the presence of two or more possible meanings of the sentence within a single word.

**Example:**

Manya is looking for a **match**.

In the above example, the word match refers to that either Manya is looking for a partner or Manya is looking for a match. (Cricket or other match)

* **Syntactic Ambiguity**

Syntactic Ambiguity exists in the presence of two or more possible meanings within the sentence.

**Example:**

I saw the girl with the binocular.

In the above example, did I have the binoculars? Or did the girl have the binoculars?

* **Referential Ambiguity**

Referential Ambiguity exists when you are referring to something using the pronoun.

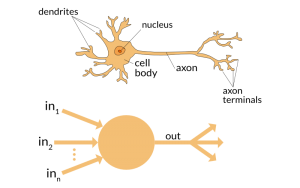
**Example:** Kiran went to Sunita. She said, "I am hungry."

In the above sentence, you do not know that who is hungry, either Kiran or Sunita.

**Deep Learning**

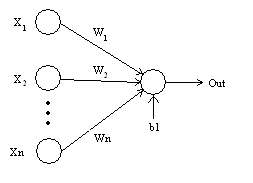
## ****Basics of Neural Networks****

**1) Neuron-** Just like a neuron forms the basic element of our brain, a neuron forms the basic structure of a neural network. Just think of what we do when we get new information. When we get the information, we process it and then we generate an output. Similarly, in case of a neural network, a neuron receives an input, processes it and generates an output which is either sent to other neurons for further processing or it is the final output.



**2) Weights –** When input enters the neuron, it is multiplied by a weight. For example, if a neuron has two inputs, then each input will have has an associated weight assigned to it. We initialize the weights randomly and these weights are updated during the model training process. The neural network after training assigns a higher weight to the input it considers more important as compared to the ones which are considered less important. A weight of zero denotes that the particular feature is insignificant.

Let’s assume the input to be a, and the weight associated to be W1. Then after passing through the node the input becomes a\*W1

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2017/05/21145421/Perceptron.bmp)

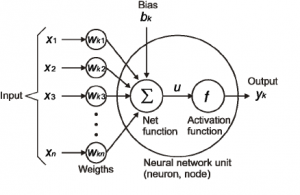
**3) Bias –** In addition to the weights, another linear component is applied to the input, called as the bias. It is added to the result of weight multiplication to the input. The bias is basically added to change the range of the weight multiplied input. After adding the bias, the result would look like a\*W1+bias. This is the final linear component of the input transformation.

**4) Activation Function –** Once the linear component is applied to the input, a non-linear function is applied to it. This is done by applying the activation function to the linear combination.The activation function translates the input signals to output signals. The output after application of the activation function would look something like f(a\*W1+b) where f() is the activation function.

In the below diagram we have “n” inputs given as X1 to Xn and corresponding weights Wk1 to Wkn. We have a bias given as bk. The weights are first multiplied to its corresponding input and are then added together along with the bias. Let this be called as u.

u=∑w\*x+b

The activation function is applied to u i.e. f(u) and we receive the final output from the neuron as yk = f(u)

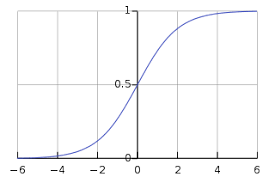


### **Commonly applied Activation Functions**

The most commonly applied activation functions are – Sigmoid, ReLU and softmax

**a) Sigmoid –** One of the most common activation functions used is Sigmoid. It is defined as:

sigmoid(x) = 1/(1+e-x)

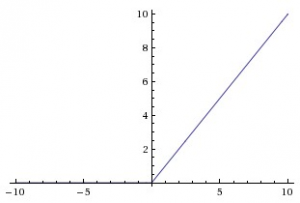


The sigmoid transformation generates a more smooth range of values between 0 and 1. We might need to observe the changes in the output with slight changes in the input values. Smooth curves allow us to do that and are hence preferred over step functions.

**b) ReLU(Rectified Linear Units) –** Instead of sigmoids, the recent networks prefer using ReLu activation functions for the hidden layers. The function is defined as:

f(x) = max(x,0).

The output of the function is X when X>0 and 0 for X<=0. The function looks like this:

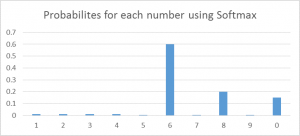


###### **source: cs231n**

The major benefit of using ReLU is that it has a constant derivative value for all inputs greater than 0. The constant derivative value helps the network to train faster.

**c) Softmax –**Softmax activation functions are normally used in the output layer for classification problems. It is similar to the sigmoid function, with the only difference being that the outputs are normalized to sum up to 1. The sigmoid function would work in case we have a binary output, however in case we have a multiclass classification problem, softmax makes it really easy to assign values to each class which can be easily interpreted as probabilities.

It’s very easy to see it this way – Suppose you’re trying to identify a 6 which might also look a bit like 8. The function would assign values to each number as below. We can easily see that the highest probability is assigned to 6, with the next highest assigned to 8 and so on…

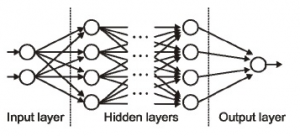


**5) Neural Network –**Neural Networks form the backbone of deep learning.The goal of a neural network is to find an approximation of an unknown function. It is formed by interconnected neurons. These neurons have weights, and bias which is updated during the network training depending upon the error. The activation function puts a nonlinear transformation to the linear combination which then generates the output. The combinations of the activated neurons give the output.

A neural network is best defined by “Liping Yang” as –

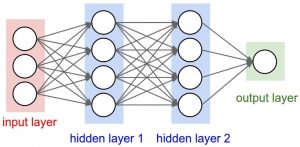
“*Neural networks are made up of numerous interconnected conceptualized artificial neurons, which pass data between themselves, and which have associated weights which are tuned based upon the network’s “experience.” Neurons have activation thresholds which, if met by a combination of their associated weights and data passed to them, are fired; combinations of fired neurons result in “learning”.*

**6) Input / Output / Hidden Layer –** Simply as the name suggests the input layer is the one which receives the input and is essentially the first layer of the network. The output layer is the one which generates the output or is the final layer of the network. The processing layers are the hidden layers within the network. These hidden layers are the ones which perform specific tasks on the incoming data and pass on the output generated by them to the next layer. The input and output layers are the ones visible to us, while are the intermediate layers are hidden.



###### **Source: cs231n**

**7) MLP (Multi Layer perceptron) –** A single neuron would not be able to perform highly complex tasks. Therefore, we use stacks of neurons to generate the desired outputs. In the simplest network we would have an input layer, a hidden layer and an output layer. Each layer has multiple neurons and all the neurons in each layer are connected to all the neurons in the next layer. These networks can also be called as fully connected networks.



**8) Forward Propagation –**Forward Propagation refers to the movement of the input through the hidden layers to the output layers. In forward propagation, the information travels in a single direction FORWARD. The input layer supplies the input to the hidden layers and then the output is generated. There is no backward movement.

**9) Cost Function –** When we build a network, the network tries to predict the output as close as possible to the actual value. We measure this accuracy of the network using the cost/loss function. The cost or loss function tries to penalize the network when it makes errors.

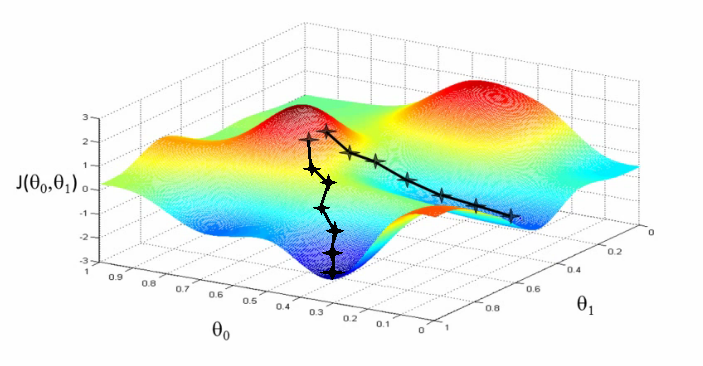
Our objective while running the network is to increase our prediction accuracy and to reduce the error, hence minimizing the cost function. The most optimized output is the one with least value of the cost or loss function.

If I define the cost function to be the mean squared error, it can be written as –

C= 1/m ∑(y – a)2 where m is the number of training inputs, a is the predicted value and y is the actual value of that particular example.

The learning process revolves around minimizing the cost.

**10) Gradient Descent –** Gradient descent is an optimization algorithm for minimizing the cost. To think of it intuitively, while climbing down a hill you should take small steps and walk down instead of just jumping down at once. Therefore, what we do is, if we start from a point x, we move down a little i.e. delta h, and update our position to x-delta h and we keep doing the same till we reach the bottom. Consider bottom to be the minimum cost point.



[Source](https://www.youtube.com/watch?v=5u4G23_OohI)

Mathematically, to find the local minimum of a function one takes steps proportional to the negative of the gradient of the function.

You can go through [this article](https://www.analyticsvidhya.com/blog/2017/03/introduction-to-gradient-descent-algorithm-along-its-variants/) for a detailed understanding of gradient descent.

**11) Learning Rate –** The learning rate is defined as the amount of minimization in the cost function in each iteration. In simple terms, the rate at which we descend towards the minima of the cost function is the learning rate. We should choose the learning rate very carefully since it should neither be very large that the optimal solution is missed and nor should be very low that it takes forever for the network to converge.



**12) Backpropagation –** When we define a neural network, we assign random weights and bias values to our nodes. Once we have received the output for a single iteration, we can calculate the error of the network. This error is then fed back to the network along with the gradient of the cost function to update the weights of the network. These weights are then updated so that the errors in the subsequent iterations is reduced. This updating of weights using the gradient of the cost function is known as back-propagation.

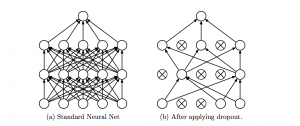
In back-propagation the movement of the network is backwards, the error along with the gradient flows back from the out layer through the hidden layers and the weights are updated.

**13) Batches –** While training a neural network, instead of sending the entire input in one go, we divide in input into several chunks of equal size randomly. Training the data on batches makes the model more generalized as compared to the model built when the entire data set is fed to the network in one go.

**14) Epochs –** An epoch is defined as a single training iteration of all batches in both forward and back propagation. This means 1 epoch is a single forward and backward pass of the entire input data.

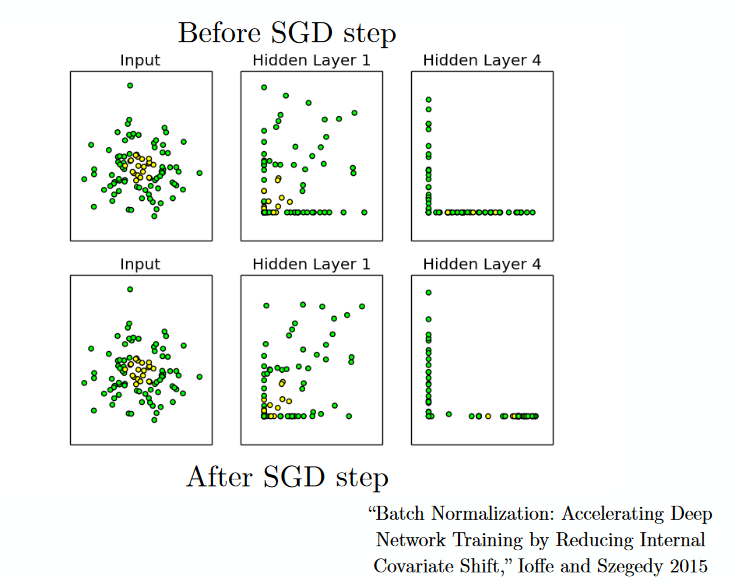
The number of epochs you would use to train your network can be chosen by you. It’s highly likely that more number of epochs would show higher accuracy of the network, however, it would also take longer for the network to converge. Also you must take care that if the number of epochs are too high, the network might be over-fit.

**15) Dropout –** Dropout is a regularization technique which prevents over-fitting of the network. As the name suggests, during training a certain number of neurons in the hidden layer is randomly dropped. This means that the training happens on several architectures of the neural network on different combinations of the neurons. You can think of drop out as an ensemble technique, where the output of multiple networks is then used to produce the final output.

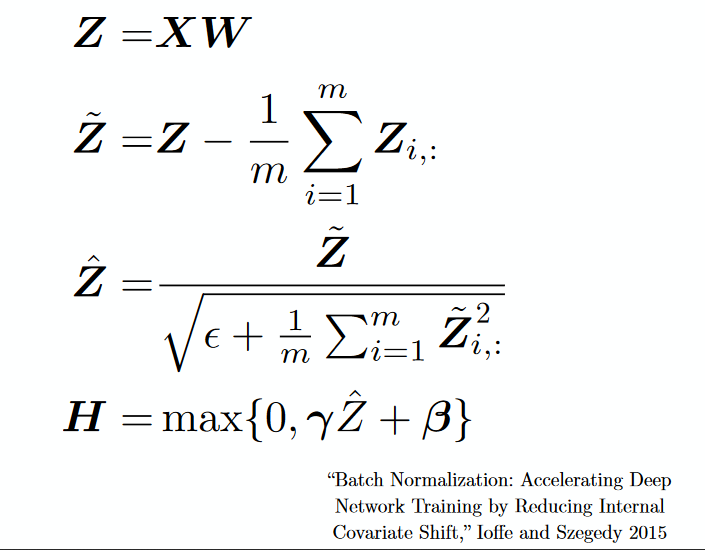


###### **Source:**[**Original paper**](https://arxiv.org/pdf/1207.0580.pdf)

**16) Batch Normalization –** As a concept, batch normalization can be considered as a dam we have set as specific checkpoints in a river. This is done to ensure that distribution of data is the same as the next layer hoped to get. When we are training the neural network, the weights are changed after each step of gradient descent. This changes the how the shape of data is sent to the next layer.



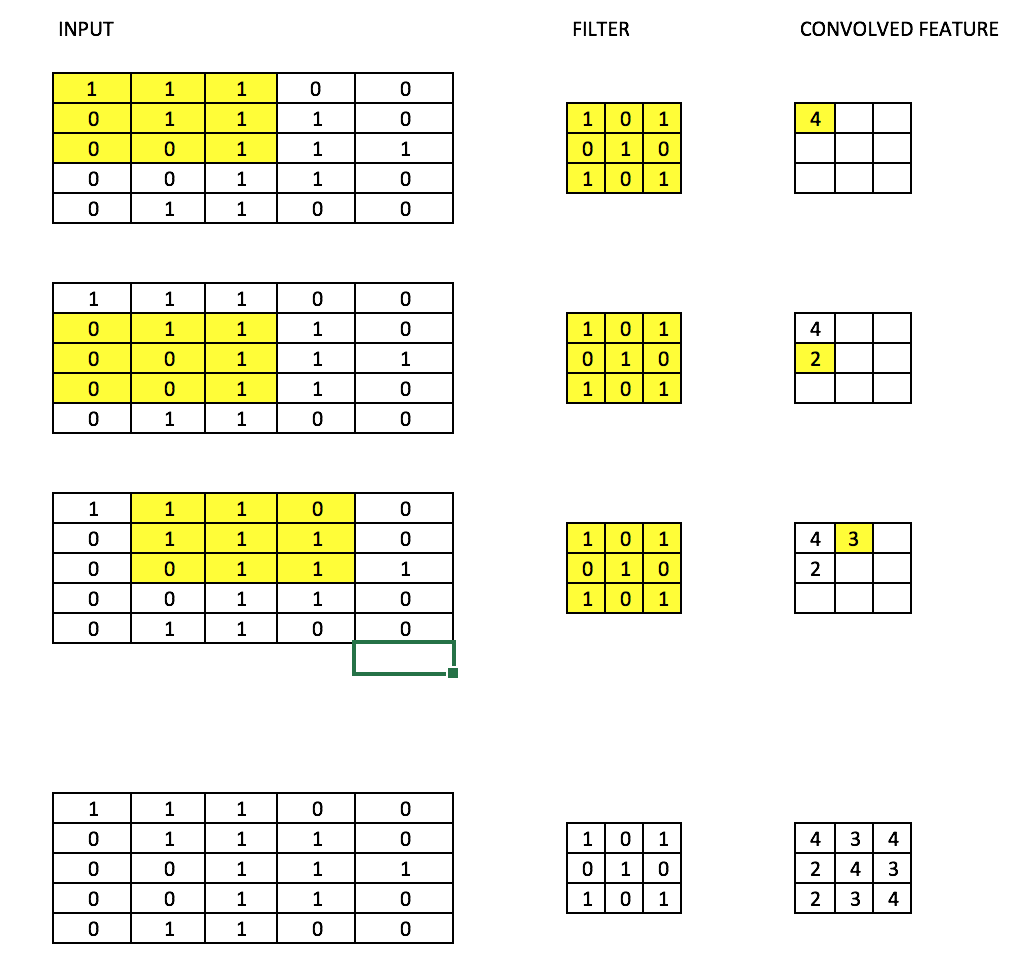
But the next layer was expecting the distribution similar to what it had previously seen. So we explicitly normalize the data before sending it to the next layer.



## Convolutional Neural Networks

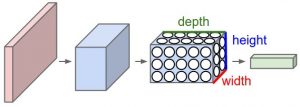
**17) Filters –**A filter in a CNN is like a weight matrix with which we multiply a part of the input image to generate a convoluted output. Let’s assume we have an image of size 28\*28. We randomly assign a filter of size 3\*3, which is then multiplied with different 3\*3 sections of the image to form what is known as a convoluted output. The filter size is generally smaller than the original image size. The filter values are updated like weight values during backpropagation for cost minimization.

Consider the below image. Here filter is a 3\*3 matrix    which is multiplied with each 3\*3 section of the image to form the convolved feature.

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2017/05/20174841/Screen-Shot-2017-05-20-at-11.18.08-PM.png)

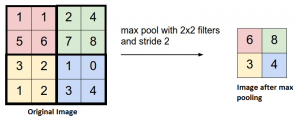
**18) CNN (Convolutional neural network) –** Convolutional neural networks are basically applied on image data. Suppose we have an input of size (28\*28\*3), If we use a normal neural network, there would be 2352(28\*28\*3) parameters. And as the size of the image increases the number of parameters becomes very large. We “convolve” the images to reduce the number of parameters (as shown above in filter definition). As we slide the filter over the width and height of the input volume we will produce a **2-dimensional activation map** that gives the output of that filter at every position. We will stack these activation maps along the depth dimension and produce the output volume.

You can see the below diagram for a clearer picture.



###### **Source: cs231n**

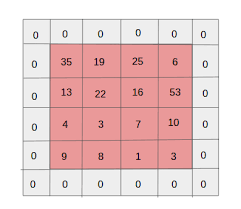
**19) Pooling –**It is common to periodically introduce pooling layers in between the convolution layers. This is basically done to reduce a number of parameters and prevent over-fitting. The most common type of pooling is a pooling layer of filter size(2,2) using the MAX operation. What it would do is, it would take the maximum of each 4\*4 matrix of the original image.



###### **Source: cs231n**

You can also pool using other operations like Average pooling, but max pooling has shown to work better in practice.

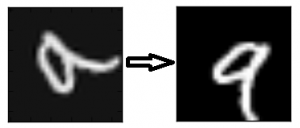
**20) Padding** – Padding refers to adding extra layer of zeros across the images so that theoutput image has the same size as the input. This is known as same padding.



After the application of filters  the convolved layer in the case of same padding has the size equal to the actual image.

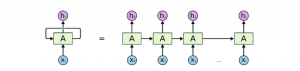
Valid padding refers to keeping the image as such an having all the pixels of the image which are actual or “valid”. In this case after the application of filters the size of the length and the width of the output keeps getting reduced at each convolutional layer.

**21) Data Augmentation –**Data Augmentation refers to the addition of new data derived from the given data, which might prove to be beneficial for prediction. For example, it might be easier to view the cat in a dark image if you brighten it, or for instance, a 9 in the digit recognition might be slightly tilted or rotated. In this case, rotation would solve the problem and increase the accuracy of our model. By rotating or brightening we’re improving the quality of our data. This is known as Data augmentation.



## Recurrent Neural Network

**22)** **Recurrent Neuron –** A recurrent neuron is one in which the output of the neuron is sent back to it for t time stamps. If you look at the diagram the output is sent back as input t times. The unrolled neuron looks like t different neurons connected together. The basic advantage of this neuron is that it gives a more generalized output.



###### **Source: cs231n**

**23) RNN(Recurrent Neural Network) –** Recurrent neural networks are used especially for sequential data where the previous output is used to predict the next one. In this case the networks have loops within them. The loops within the hidden neuron gives them the capability to store information about the previous words for some time to be able to predict the output. The output of the hidden layer is sent again to the hidden layer for t time stamps. The unfolded neuron looks like the above diagram. The output of the recurrent neuron goes to the next layer only after completing all the time stamps. The output sent is more generalized and the previous information is retained for a longer period.

The error is then back propagated according to the unfolded network to update the weights. This is known as **backpropagation through time(BPTT).**

**24) Vanishing Gradient Problem –** Vanishing gradient problem arises in cases where the gradient of the activation function is very small. During back propagation when the weights are multiplied with these low gradients, they tend to become very small and “vanish” as they go further deep in the network. This makes the neural network to forget the long range dependency. This generally becomes a problem in cases of recurrent neural networks where long term dependencies are very important for the network to remember.

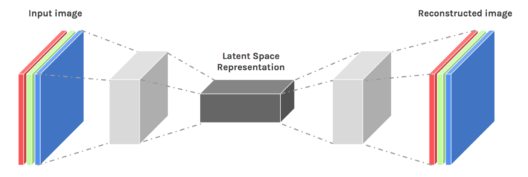
This can be solved by using activation functions like ReLu which do not have small gradients.

**25) Exploding Gradient Problem –** This is the exact opposite of the vanishing gradient problem, where the gradient of the activation function is too large. During back propagation, it makes the weight of a particular node very high with respect to the others rendering them insignificant. This can be easily solved by clipping the gradient so that it doesn’t exceed a certain value.

**AutoEncoders :-**

**What are Autoencoders?**

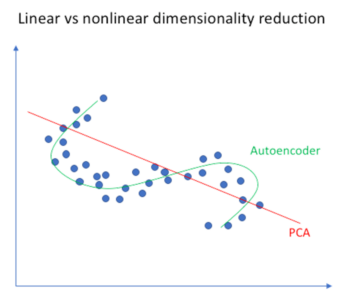
An autoencoder [**neural network**](https://www.edureka.co/blog/neural-network-tutorial/) is an [**Unsupervised Machine learning**](https://www.edureka.co/blog/what-is-machine-learning/) algorithm that applies backpropagation, setting the target values to be equal to the inputs. Autoencoders are used to reduce the size of our inputs into a smaller representation. If anyone needs the original data, they can reconstruct it from the compressed data.



We have a similar machine learning algorithm ie. PCA which does the same task. So you might be thinking why do we need Autoencoders then? Let’s continue this Autoencoders Tutorial and find out the reason behind using Autoencoders.

**Autoencoders Tutorial: Its Emergence**

Autoencoders are preferred over PCA because:



* An autoencoder can learn **non-linear** **transformations** with a **non-linear activation function** and multiple layers.
* It doesn’t have to learn dense layers. It can use **convolutional layers** to learn which is better for video, image and series data.
* It is more efficient to learn several layers with an autoencoder rather than learn one huge transformation with PCA.
* An autoencoder provides a representation of each layer as the output.
* It can make use of **pre-trained layers** from another model to apply transfer learning to enhance the encoder/decoder.

Now let’s have a look at a few Industrial Applications of Autoencoders.

## ****Applications of Autoencoders****

### **Image Coloring**



Autoencoders are used for converting any black and white picture into a colored image. Depending on what is in the picture, it is possible to tell what the color should be.

### **Autoencoders Tutorial - Feature Variation**

### **Feature variation**

It extracts only the required features of an image and generates the output by removing any noise or unnecessary interruption.

### **Autoencoders Tutorial - Dimensionality Reduction**

### **Dimensionality Reduction**

The reconstructed image is the same as our input but with reduced dimensions. It helps in providing the similar image with a reduced pixel value.

### **Autoencoders Tutorial - Denoising image**

### **Denoising Image**

The input seen by the autoencoder is not the raw input but a stochastically corrupted version. A denoising autoencoder is thus trained to reconstruct the original input from the noisy version.

### **Autoencoders Tutorial - Watermark Removal**

### **Watermark Removal**

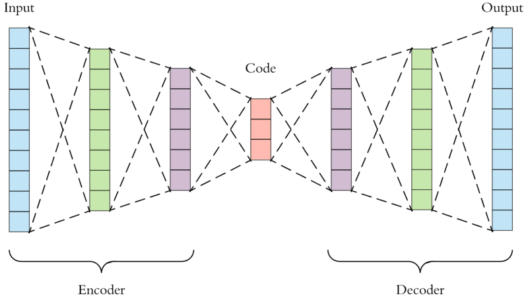
It is also used for removing watermarks from images or to remove any object while filming a video or a movie.

Now that you have an idea of the different industrial applications of Autoencoders, let’s continue our Autoencoders Tutorial Blog and understand the complex architecture of Autoencoders.

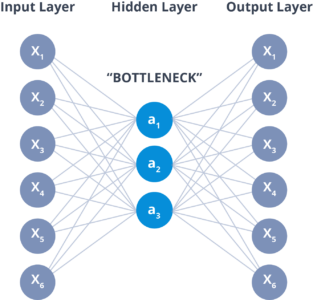
## ****Architecture of Autoencoders****

An Autoencoder consist of three layers:

1. **Encoder**
2. **Code**
3. **Decoder**



* **Encoder:** This part of the network compresses the input into a **latent space representation**. The encoder layer **encodes** the input image as a compressed representation in a reduced dimension. The compressed image is the distorted version of the original image.
* **Code:** This part of the network represents the compressed input which is fed to the decoder.
* **Decoder:** This layer **decodes** the encoded image back to the original dimension. The decoded image is a lossy reconstruction of the original image and it is reconstructed from the latent space representation.



 The layer between the encoder and decoder, ie. the code is also known as **Bottleneck**. This is a well-designed approach to decide which aspects of observed data are relevant information and what aspects can be discarded. It does this by balancing two criteria :

* Compactness of representation, measured as the compressibility.
* It retains some behaviourally relevant variables from the input.

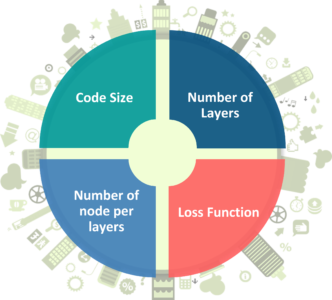
Now that you have an idea of the architecture of an Autoencoder. Let’s continue our Autoencoders Tutorial and understand the different properties and the Hyperparameters involved while training Autoencoders.

**Properties and Hyperparameters**

**Properties of Autoencoders:**

* **Data-specific**: Autoencoders are only able to compress data similar to what they have been trained on.
* **Lossy:** The decompressed outputs will be degraded compared to the original inputs.
* **Learned automatically from examples:**It is easy to train specialized instances of the algorithm that will perform well on a specific type of input.

**Hyperparameters of Autoencoders:**



There are **4** hyperparameters that we need to set before training an autoencoder:

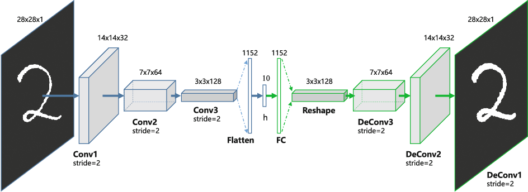
* **Code size**: It represents the number of nodes in the middle layer. Smaller size results in more compression.
* **Number of layers**: The autoencoder can consist of as many layers as we want.
* **Number of nodes per layer**: The number of nodes per layer decreases with each subsequent layer of the encoder, and increases back in the decoder. The decoder is symmetric to the encoder in terms of the layer structure.
* **Loss function:** We either use mean squared error or binary cross-entropy. If the input values are in the range [0, 1] then we typically use cross-entropy, otherwise, we use the mean squared error.

Now that you know the properties and hyperparameters involved in the training of Autoencoders. Let’s move forward with our Autoencoders Tutorial and understand the different types of autoencoders and how they differ from each other.

**Types of Autoencoders**

**Convolution Autoencoders**

Autoencoders in their traditional formulation does not take into account the fact that a signal can be seen as a sum of other signals. Convolutional Autoencoders use the convolution operator to exploit this observation. They learn to encode the input in a set of simple signals and then try to reconstruct the input from them, modify the geometry or the reflectance of the image.

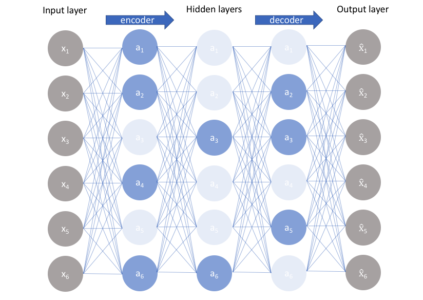


Use cases of CAE:

* Image Reconstruction
* Image Colorization
* latent space clustering
* generating higher resolution images

**Sparse Autoencoders**

Sparse autoencoders offer us an alternative method for introducing an information bottleneck **without requiring a reduction in the number of nodes** at our hidden layers. Instead, we’ll construct our loss function such that we penalize activations within a layer.

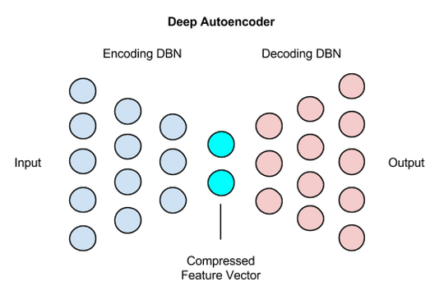


**Deep Autoencoders**

The extension of the simple Autoencoder is the **Deep Autoencoder**. The first layer of the Deep Autoencoder is used for first-order features in the **raw input**. The second layer is used for second-order features corresponding to **patterns** in the appearance of first-order features. Deeper layers of the Deep Autoencoder tend to learn even higher-order features.

A **deep autoencoder**is composed of two, symmetrical deep-belief networks-

1. First four or five shallow layers representing the encoding half of the net.
2. The second set of four or five layers that make up the decoding half.

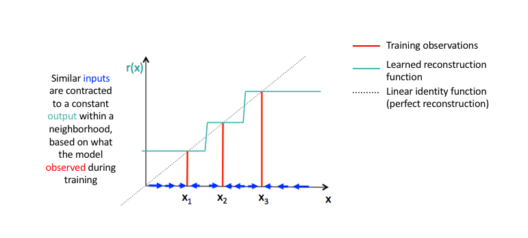


Use cases of Deep Autoencoders

* Image Search
* Data Compression
* Topic Modeling & Information Retrieval (IR)

**Contractive Autoencoders**

A **contractive autoencoder**is an unsupervised deep learning technique that helps a neural network encode unlabeled training data. This is accomplished by constructing a **loss term** which penalizes large derivatives of our hidden layer activations with respect to the input training examples, **essentially penalizing** instances where a small change in the input leads to a large change in the encoding space.



Now that you have an idea of what Autoencoders is, it’s different types and it’s properties.