ABSTRACT

Data is an entity or any attribute or characteristic that can describe particular aspect. Data science is a multi-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data. Data science is the same concept as data mining and big data: "use the most powerful hardware, the most powerful programming systems, and the most efficient algorithms to solve problems".

Data science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena" with data.[4] It employs techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, and information science. By the implementation of the above domains one can easily develop models that are Artificial Intelligent and Machine Learnable.

These highly efficient models can increase the scope for the automation just by the implementation in a software programming language like Python. They can be easily implemented such that one can use it as per the requirement. The Deep Learning models can be implemented by the use of packages like tensorflow, keras, opencv etc from python programming language. For any AI based application to be developed aneural network needs to be developed such that it can be used to mimic the activities of the human brain.

Keywords- Data, Data Science, Artificial Intelligence, Machine Learning, Deep Learning, Neural Networks, Python, Tensorflow, Keras.

CHAPTER 1

INTRODUCTION

1.1 DATA

Data is any sequence of one or more symbols given meaning by specific act(s) of interpretation or it can be considered as a real entity that has some characteristics. **Data** (or datum – a single unit of data) requires interpretation to become information. To translate data to information, there must be several known factors considered and proper methods must be applied such that in order to convert the haphazard data to information. The term metadata is used to reference the data about the data which can be implied, specified or given. The data can have it been recorded by the date, time and if required its purpose and size.

Digital data is data that is represented using the binary number system of ones (1) and zeros (0), as opposed to analog representation. Data can be stored in the storage devices like RAM and ROM and can be used accordingly based on the application. Physical computer memory elements consist of an address and a byte/word of data storage. Digital data are often stored in relational databases, like tables or SQL databases, and can generally be represented as abstract key/value pairs which makes it easy to access the data accordingly with an ease .Data can be organized in many different types of data structures, including arrays, graphs, and objects. Data pass in and out of computers via peripheral devices. Digital data comes in these three states: data at rest, data in transit and data in use.

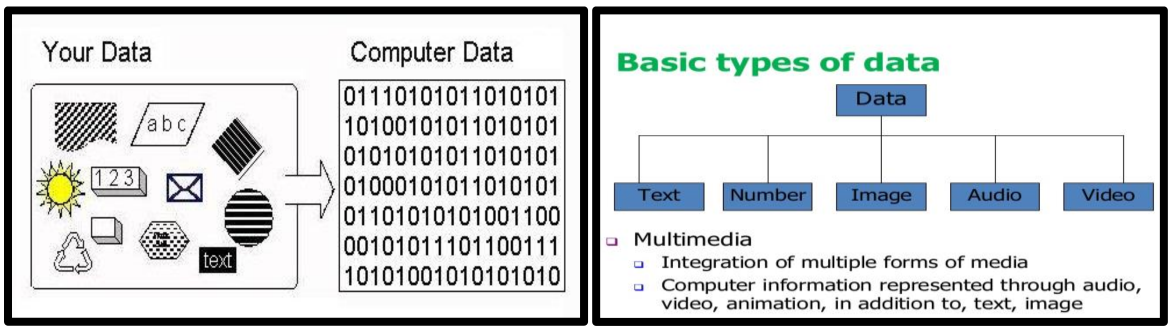
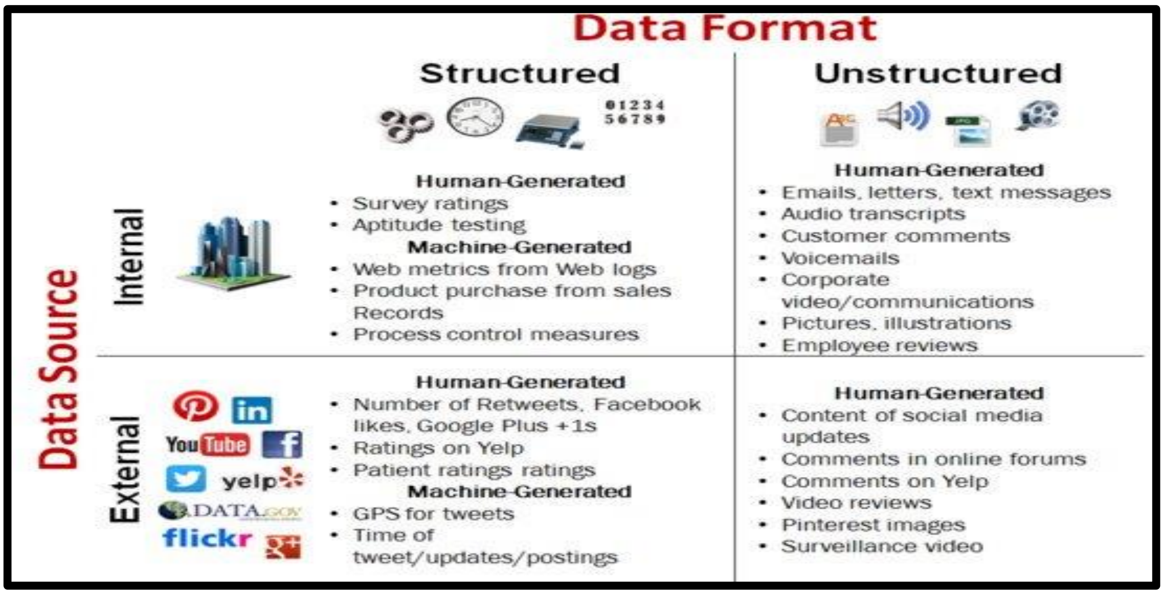


Fig 1.1: DATA

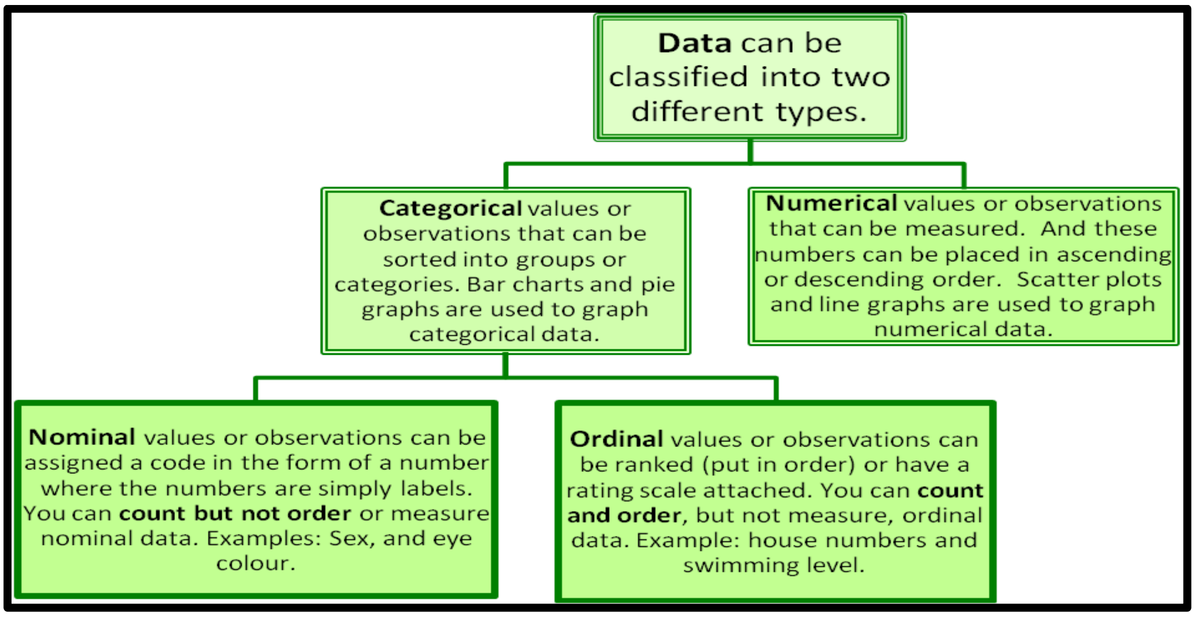


Data Format

Data Source		Structured	Unstructured
Internal		**Human-Generated** • Survey ratings • Aptitude testing **Machine-Generated** • Web metrics from Web logs • Product purchase from sales Records • Process control measures	**Human-Generated** • Emails, letters, text messages • Audio transcripts • Customer comments • Voicemails • Corporate video/communications • Pictures, illustrations • Employee reviews
External	yelp DATA.GOV flickr	**Human-Generated** • Number of Retweets, Facebook likes, Google Plus +1s • Ratings on Yelp • Patient ratings ratings **Machine-Generated** • GPS for tweets • Time of tweet/updates/postings	**Human-Generated** • Content of social media updates • Comments in online forums • Comments on Yelp • Video reviews • Pinterest images • Surveillance video

Fig 1.2: Sources of DATA

❖ TYPES OF DATA



Data can be classified into two different types.

Categorical values or observations that can be sorted into groups or categories. Bar charts and pie graphs are used to graph categorical data.

Numerical values or observations that can be measured. And these numbers can be placed in ascending or descending order. Scatter plots and line graphs are used to graph numerical data.

Nominal values or observations can be assigned a code in the form of a number where the numbers are simply labels. You can **count but not order** or measure nominal data. Examples: Sex, and eye colour.

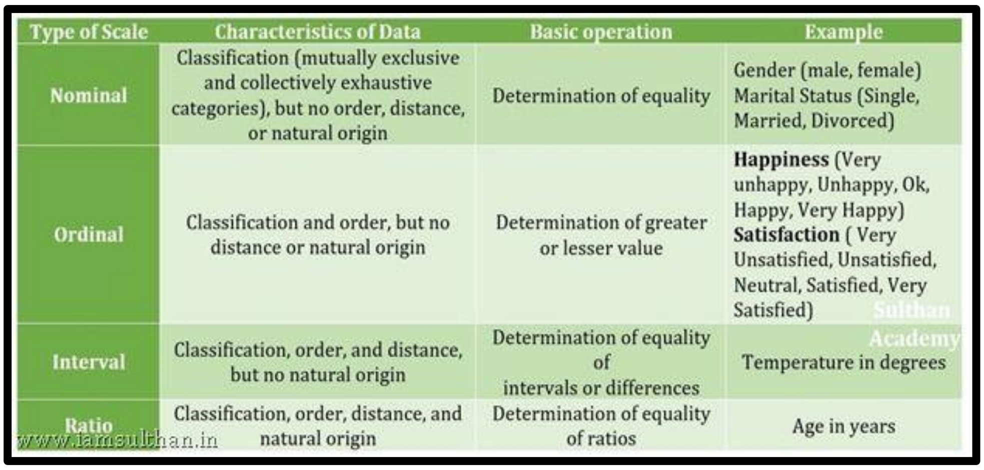
Ordinal values or observations can be ranked (put in order) or have a rating scale attached. You can **count and order**, but not measure, ordinal data. Example: house numbers and swimming level.

Fig 1.3: Types of DATA

- **Numerical -** Numerical data is information that is measurable, and it is, of course, data represented as numbers and not words or text. Continuous numbers are numbers that don't have a logical end to them. Examples include variables that represent money or height. Discrete numbers are the opposite; they have a logical end to them. Some examples include variables for days in the month, or number of bugs logged.

- **Categorical**

  For categorical data, this is any data that isn't a number, which can mean a string of text or date. These variables can be broken down into nominal and ordinal values, though you won't often see this done. Ordinal values are values that have a set order to them. Nominal values are the opposite of ordinal values, and they represent values with no set order to them. In addition to ordinal and nominal values, there is a special type of categorical data called binary. Binary data types only have two values – yes or no. This can be represented in different ways such as "True" and "False" or 1 and 0. Binary data is used heavily for classification machine learning models.



Type of Scale	Characteristics of Data	Basic operation	Example
Nominal	Classification (mutually exclusive and collectively exhaustive categories), but no order, distance, or natural origin	Determination of equality	Gender (male, female) Marital Status (Single, Married, Divorced)
Ordinal	Classification and order, but no distance or natural origin	Determination of greater or lesser value	**Happiness** (Very unhappy, Unhappy, Ok, Happy, Very Happy) **Satisfaction** (Very Unsatisfied, Unsatisfied, Neutral, Satisfied, Very Satisfied)
Interval	Classification, order, and distance, but no natural origin	Determination of equality of intervals or differences	Temperature in degrees
Ratio	Classification, order, distance, and natural origin	Determination of equality of ratios	Age in years

Fig 1.4: Comparison between Types of DATA

1.2 DATA MINING

Data mining is the process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems. Data mining is an interdisciplinary subfield of computer science and statistics with an overall goal to extract information (with intelligent methods) from a data set and transform the information into a comprehensible structure for further use.

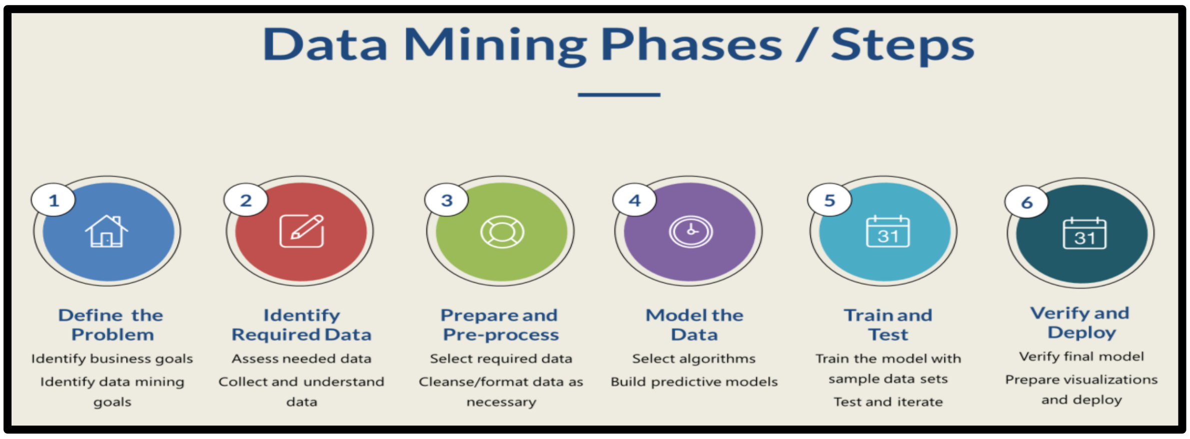


Fig.1.5: DATA Mining Process

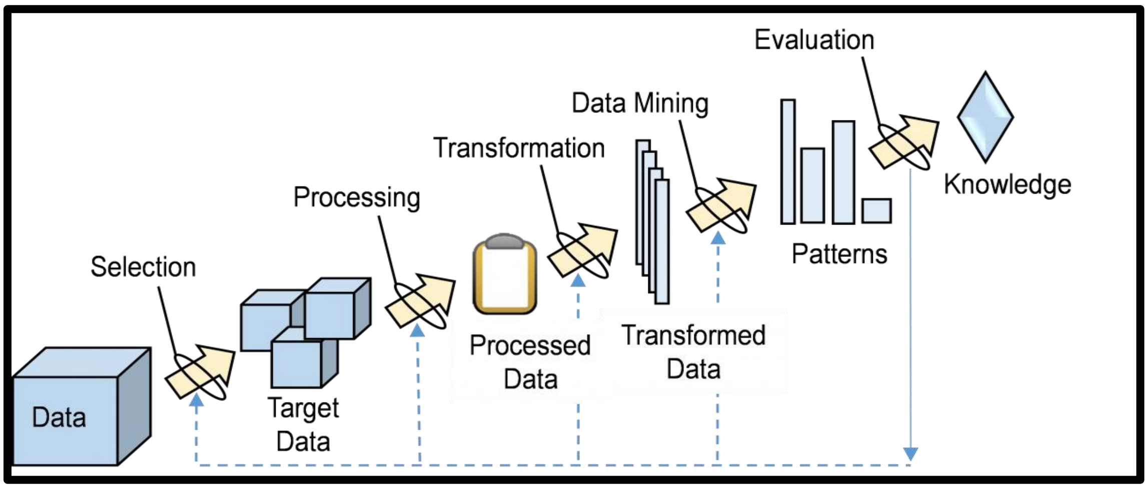


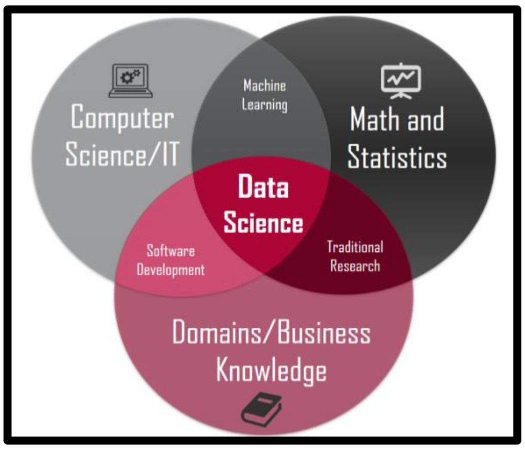
Fig 1.6: KDD Process

Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD. Aside from the raw analysis step, it also involves database and data management aspects, data preprocessing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. The difference between data analysis and data mining is that data analysis is used to test models and hypotheses on the dataset. The term "data mining" is in fact a misnomer, because the goal is the extraction of patterns and knowledge from large amounts of data, not the extraction (mining) of data itself. It also is a buzzword and is frequently applied to any form of large-scale data or information processing (collection, extraction, warehousing, analysis, and statistics) as well as any application of computer decision support system, including artificial intelligence (e.g., machine learning) and business intelligence. The representation and quality of data is first and foremost before running an analysis. Often, data preprocessing is the most important phase of a machine learning project, especially in computational phases.

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data preprocessing includes cleaning, instance selection, normalization, transformation, feature extraction and selection, etc. The product of data preprocessing is the final training set. Data pre-processing may affect the way in which outcomes of the final data processing can be interpreted. This aspect should be carefully considered when interpretation of the results is a key point, such in the multivariate processing of chemical data. The actual data mining task is the semi-automatic or automatic analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection), and dependencies (association rule mining, sequential pattern mining).

1.3 DATA SCIENCE

Data science is a multi-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data. Data science is the same concept as data mining and big data but the use of the most powerful hardware and the most powerful programming systems with efficient algorithms are used to solve the problems. It is implemented by the use of statistics, data analysis, machine learning and along with proper understanding of the software languages and mathematics.



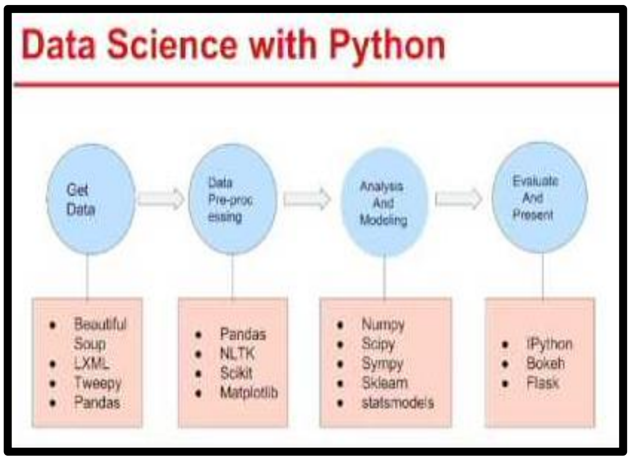
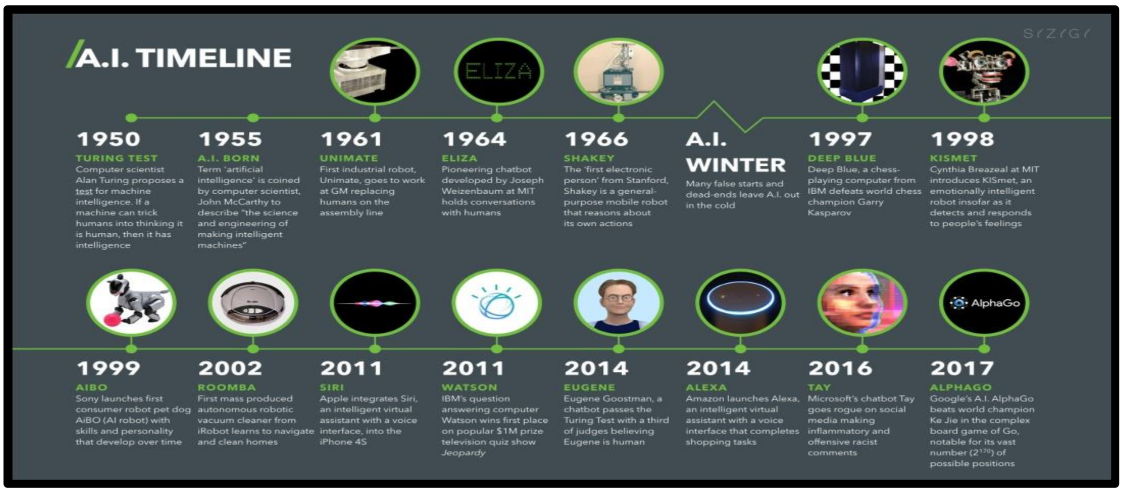


Fig 1.7: Data Science

1.4 ARTIFICIAL INTELLIGENCE

Artificial intelligence (**AI**) or **machine intelligence** is the intelligence demonstrated by machines, in contrast to the **natural intelligence** displayed by humans. Colloquially, the term "artificial intelligence" is often used to describe machines (or computers) that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving". As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. Approaches include statistical methods, computational intelligence, and traditional symbolic AI. Many tools are used in AI, including versions of search and mathematical optimization, artificial neural networks, and methods based on statistics, probability and economics. The AI field draws upon computer science, information engineering, mathematics, psychology, linguistics, philosophy, and many other fields. AI techniques have experienced a resurgence following concurrent advances in computer power, large amounts of data, and theoretical understanding; and AI techniques have become an essential part of the technology industry, helping to solve many challenging problems in computer science, software engineering and operations research.



1950
TURING TEST
Computer scientist Alan Turing proposes a test for machine intelligence. If a machine can trick humans into thinking it is human, then it has intelligence

1955
A.I. BORN
Term 'artificial intelligence' is coined by computer scientist, John McCarthy to describe "the science and engineering of making intelligent machines"

1961
UNIMATE
First industrial robot, Unimate, goes to work at GM replacing humans on the assembly line

1964
ELIZA
Pioneering chatbot developed by Joseph Weizenbaum at MIT holds conversations with humans

1966
SHAKEY
The 'first electronic person' from Stanford, Shakey is a general-purpose mobile robot that reasons about its own actions

A.I. WINTER
Many false starts and dead-ends leave A.I. out in the cold

1997
DEEP BLUE
Deep Blue, a chess-playing computer from IBM defeats world chess champion Garry Kasparov

1998
KISMET
Cynthia Breazeal at MIT introduces KISmet, an emotionally intelligent robot insofar as it detects and responds to people's feelings

1999
AIBO
Sony launches first consumer robot pet dog AiBO (AI robot) with skills and personality that develop over time

2002
ROOMBA
First mass produced autonomous robotic vacuum cleaner from iRobot learns to navigate and clean homes

2011
SIRI
Apple integrates Siri, an intelligent virtual assistant with a voice interface, into the iPhone 4S

2011
WATSON
IBM's question answering computer Watson wins first place on popular $1M prize television quiz show *Jeopardy*

2014
EUGENE
Eugene Goostman, a chatbot passes the Turing Test with a third of judges believing Eugene is human

2014
ALEXA
Amazon launches Alexa, an intelligent virtual assistant with a voice interface that completes shopping tasks

2016
TAY
Microsoft's chatbot Tay goes rogue on social media making inflammatory and offensive racist comments

2017
ALPHAGO
Google's A.I. AlphaGo beats world champion Ke Jie in the complex board game of Go, notable for its vast number (2^{170}) of possible positions

Fig 1.8: AI TIMELINE

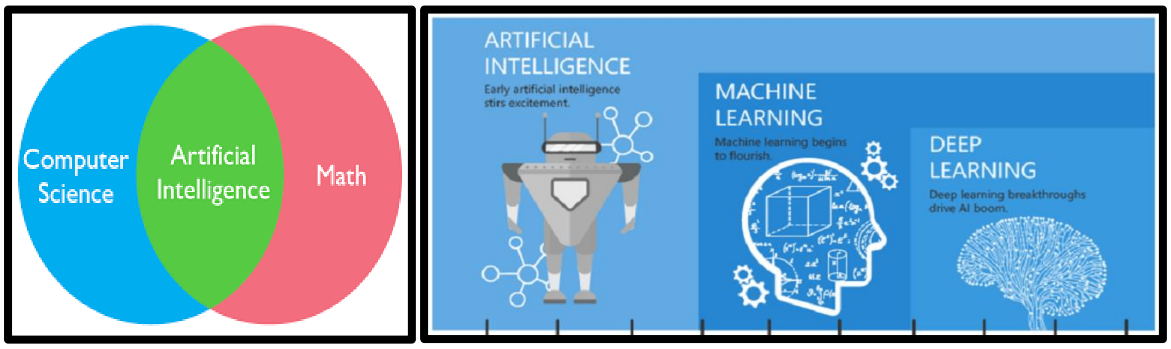

Computer Science
Artificial Intelligence
Math
ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.

Fig 1.9: Artificial Intelligence

Artificial intelligence can be classified into three different types of systems: analytical, human-inspired, and humanized artificial intelligence. Analytical AI has only characteristics consistent with cognitive intelligence; generating cognitive representation of the world and using learning based on past experience to inform future decisions. Human-inspired AI has elements from cognitive and emotional intelligence; understanding human emotions, in addition to cognitive elements, and considering them in their decision making. Humanized AI shows characteristics of all types of competencies (i.e., cognitive, emotional, and social intelligence), is able to be self-conscious and is self-aware in interactions with others. The goals of AI research is to include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence is among the field's long-term goals.

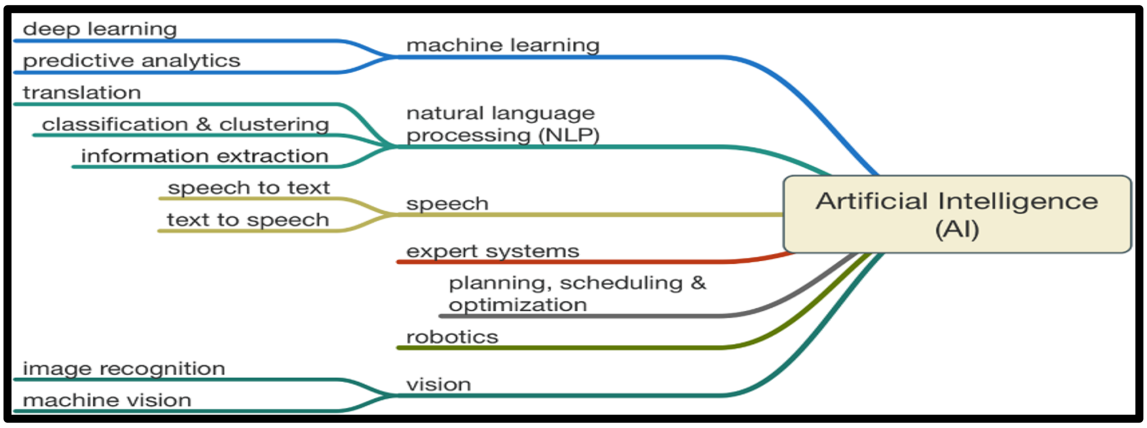


Fig 1.10: Applications of AI

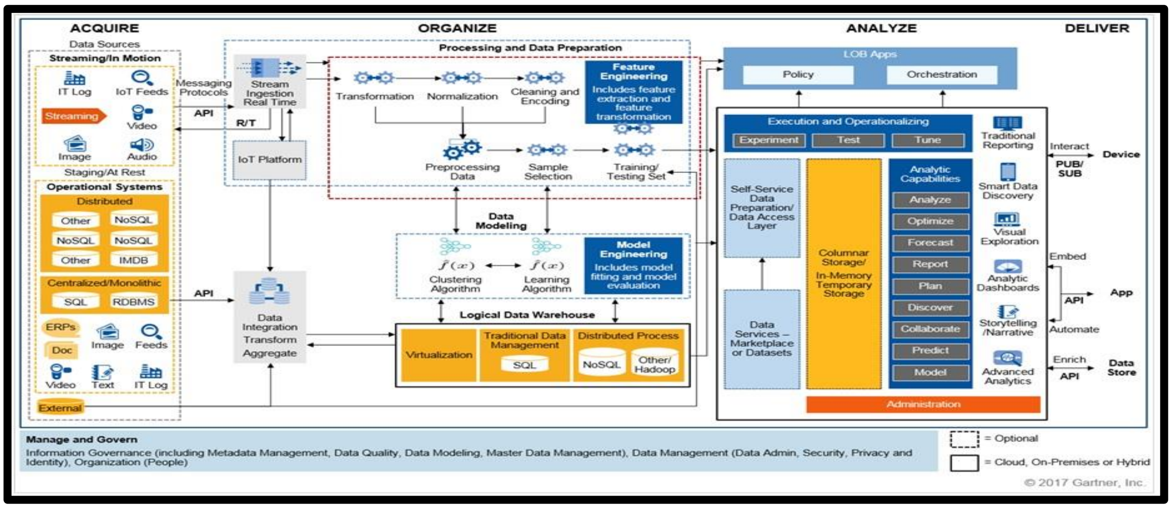


Fig 1.11: AI Architecture

A typical AI analyzes its environment and takes actions that maximize its chance of success by using components like organizer, analyzer and optimizer. These components work on the data that has been acquired in a pre-processed manner that is free from noises/errors. An AI's intended utility function (or goal) can be simple ("1 if the AI wins a game, 0 otherwise") or complex ("Do mathematically similar actions to the ones succeeded in the past"). Goals can be explicitly defined, or induced.

AI often revolves around the use of algorithms. An algorithm is a set of unambiguous instructions that a mechanical computer can execute. A complex algorithm is often built on top of other, simpler, algorithms. Many AI algorithms are capable of learning from data and then they can enhance themselves by learning new heuristics. Much of AI research involves figuring out how to identify and avoid considering broad range of possibilities that are unlikely to be beneficial. Learning algorithms work on the basis that strategies, algorithms, and inferences that worked well in the past are likely to continue working well in the future. There are chances for the model to overfit and cause the entire operation to generate wrong results. Hence certain measures are to be taken to pervent overfit or underfit of the data that is taken to analyze and process.

1.5 MACHINE LEARNING

Machine learning is a sort of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves. The process of learning begins with observations or data by looking for the patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly. Machine learning algorithms are often categorized as supervised or unsupervised..

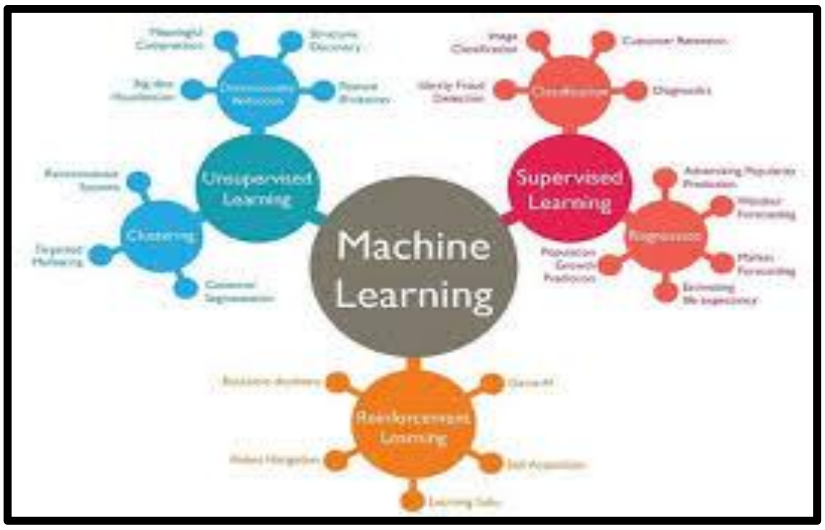


Fig 1.12: Types of Learning in ML

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information. The following types of machine learning algorithms are:

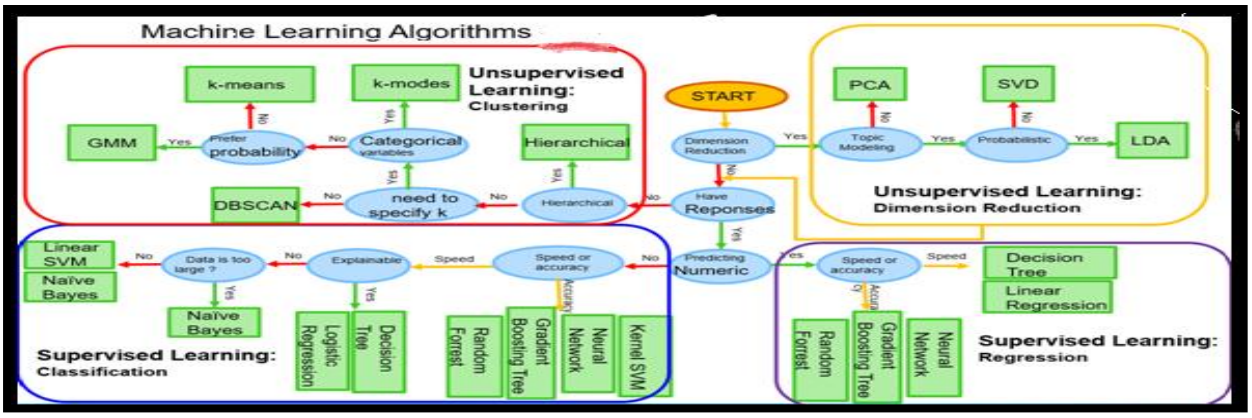


Fig 1.13: Machine Learning Algorithms

- **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
- **Unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
- **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant

resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

- **Reinforcement machine learning algorithms** is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

1.6 DEEP LEARNING

Deep learning/ Deep structured learning/H**ierarchical learning** is part of a broader family of machine learning methods based on artificial neural networks. Learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts.

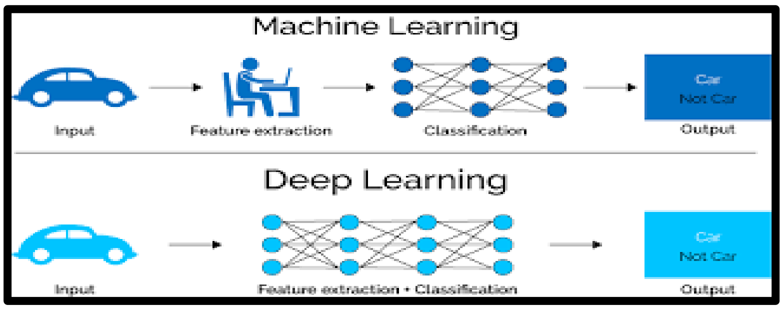


Fig 1.14: Machine Learning v/s Deep Learning

In order to implement them packages like TensorFlow and Keras of python programming language makes it easier to code and execute. Artificial Neural Networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains.

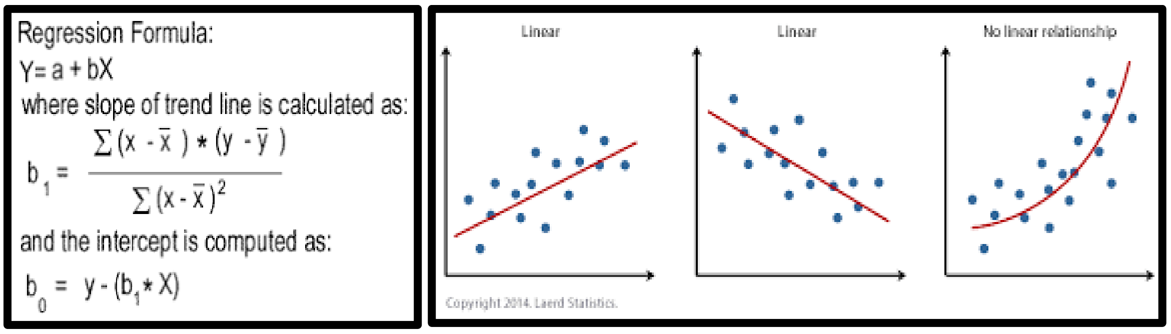
CHAPTER-2

REVIEW OF LITERATURE

2.1 REGRESSION ALGORITHM

Regression analysis is a reliable method of identifying which variables have impact on a particular topic of interest. The process of performing a regression allows you to confidently determine which factors matter the most, which factors can be ignored, and how these factors influence each other. In order to understand regression analysis fully, it's essential to comprehend the following terms:

- **Dependent Variable:** This is the main factor that you're trying to understand or predict.
- **Independent Variables:** These are the factors that you hypothesize have an impact on your dependent variable.



Regression Formula:

$Y = a + bX$

where slope of trend line is calculated as:

$$b_1 = \frac{\sum (x - \bar{x}) * (y - \bar{y})}{\sum (x - \bar{x})^2}$$

and the intercept is computed as:

$$b_0 = y - (b_1 * X)$$

Fig 2.1: Linear Regression

In order to conduct a regression analysis, you'll need to define a dependent variable that you hypothesize is being influenced by one or several independent variables either in a linear or in a polynomial way. The data upon which it is to be operated in continuous in nature and not categorical. The data is collected either by means of survey or by means of using random data as per the requirement.

Logistic regression (or **logit regression**) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name.

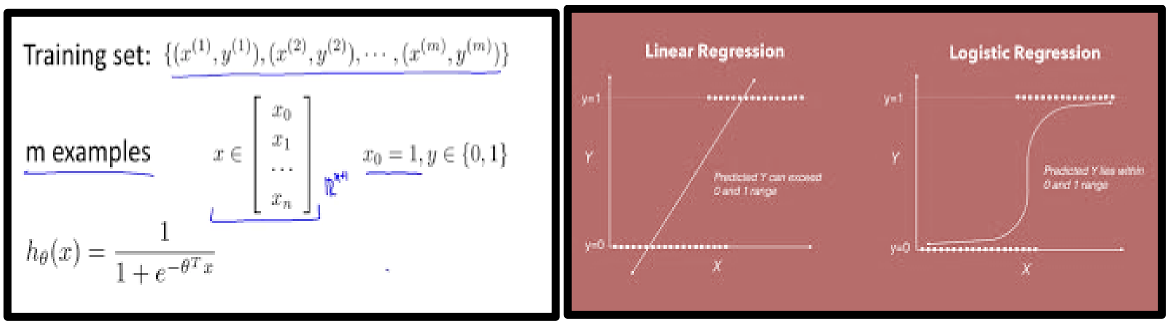

Training set: $\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\cdots,(x^{(m)},y^{(m)})\}$
m examples
$x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$
$x_0 = 1, y \in \{0,1\}$
$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$
Linear Regression
Logistic Regression
y=1
Y
y=0
X
Predicted Y can exceed 0 and 1 range
Predicted Y lies within 0 and 1 range

Fig 2.2: Logistic Regression

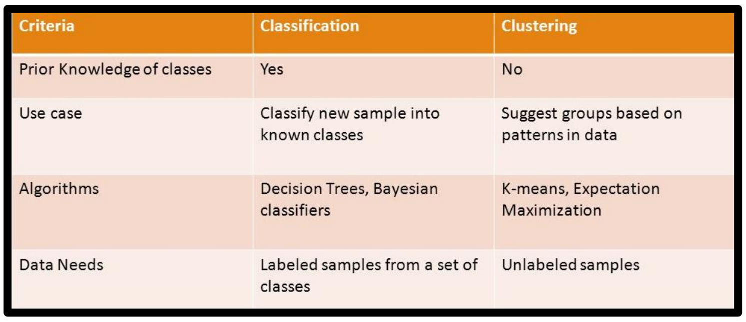
The unit of measurement for the log-odds scale is called a logit, from **log**istic un**it**, hence the alternative names. Analogous models with a different **sigmoid function** instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter; for a binary dependent variable this generalizes the odds ratio.

2.2 CLASSIFICATION ALGORITHM

The Classification Algorithms are easier to implement where in the target class is predicted by analyzing the training dataset**.** We use the training dataset to get better boundary conditions which could be used to determine each target class. Once the boundary conditions are determined, the next task is to predict the target class. The whole process is known as classification. It is a supervised learning model wherein it can be made to operate on the categorical data like **yes or no or to generate subclasses from the collected data.** Whereas in clustering, the idea is not to predict the target class as in classification, it's more trying to group the similar kind of things by considering the most satisfied condition, **all the items in the same group should be similar and no two**

different group items should not be similar. Basic Terminology in Classification Algorithms are:

- **Classifier-** An algorithm that maps the input data to a specific category.
- **Classification model-** A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data.
- **Feature-** A feature is an individual measurable property of a phenomenon being observed.
- **Binary Classification-** Classification task with two possible outcomes. **Eg: Gender classification (Male / Female)**
- **Multi-class classification-** Classification with more than two classes. In multi-class classification, each sample is assigned to one and only one target label. **Eg: An animal can be a cat or dog but not both at the same time.**
- **Multi-label classification-** Classification task where each sample is mapped to a set of target labels (more than one class). **Eg: A news article can be about sports, a person, and location at the same time.**



Criteria	Classification	Clustering
Prior Knowledge of classes	Yes	No
Use case	Classify new sample into known classes	Suggest groups based on patterns in data
Algorithms	Decision Trees, Bayesian classifiers	K-means, Expectation Maximization
Data Needs	Labeled samples from a set of classes	Unlabeled samples

Fig 2.3: Difference between Classification and Clustering

➢ **Applications of Classification Algorithms**

- Email spam classification
- Bank customers loan pay willingness prediction.
- Cancer tumor cells identification.
- Sentiment analysis
- Drugs classification
- Facial key points detection
- Pedestrians detection in an automotive car driving.

➢ **Types of Classification Algorithms**

❖ **Linear Classifiers**

  - Logistic regression

    - Naive Bayes classifier
    - Fisher’s linear discriminant
- **Support vector machines**
    - Least squares support vector machines
- **Quadratic classifiers**
- **Kernel estimation**
    - k-nearest neighbor
- **Decision trees**
    - Random forests
- **Neural networks**
- **Learning vector quantization**

2.2.1 DECISION TREE ALGORITHM

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too. It creates a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data(training data). The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label. Decision Trees follow Sum of Product (SOP) representation. The Sum of product(SOP) is also known as Disjunctive Normal Form. For a class, every branch from the root of the tree to a leaf node having the same class is a conjunction(product) of values, different branches ending in that class form a disjunction(sum).

- **Decision Tree Algorithm Pseudocode**

- Place the best attribute of the dataset at the root of the tree.
- Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
- Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

In decision trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.We continue comparing our record's attribute values with other internal nodes of the tree until we reach a leaf node with predicted class value. As we know how the modeled decision tree can be used to predict the target class or the value. Now let's understanding how we can create the decision tree model. The primary challenge in the decision tree implementation is to identify which attributes do we need to consider as the root node and each level. Handling this is know the attributes selection. We have different attributes selection measure to identify the attribute which can be considered as the root note at each level.The assumptions made while using Decision tree are at the beginning, the whole training set is considered as the root, Feature values are preferred to be categorical, If the values are continuous then they are discretized prior to building the model, Records are distributed recursively on the basis of attribute values, Order to placing attributes as root or internal node of the tree is done by using some statistical approach. The popular attribute selection measures are:

- Entropy

$$\boxed{E(T,X) = \sum_{c \in X} P(c)E(c)}$$

- Information gain

$$\boxed{Gain(T,X) = Entropy(T) - Entropy(T,X)}$$

Overfitting is a practical problem while building a decision tree model. The model is having an issue of overfitting is considered when the algorithm continues to go deeper and deeper in the to reduce the training set error but results with an increased test set error i.e, Accuracy of prediction for our model goes down. It generally happens when it builds many branches due to outliers and irregularities in data.Two approaches which we can use to avoid overfitting are- Pre-Pruning, Post-Pruning

- **Pre-Pruning-** In pre-pruning, it stops the tree construction bit early. It is preferred not to split a node if its goodness measure is below a threshold value. But it's difficult to choose an appropriate stopping point.

- **Post-Pruning-** In post-pruning first, it goes deeper and deeper in the tree to build a complete tree. If the tree shows the overfitting problem then pruning is done as a post-pruning step. We use a cross-validation data to check the effect of our pruning. Using cross-validation data, it tests whether expanding a node will make an improvement or not.

❖ **Decision Tree Algorithm Advantages and Disadvantages**

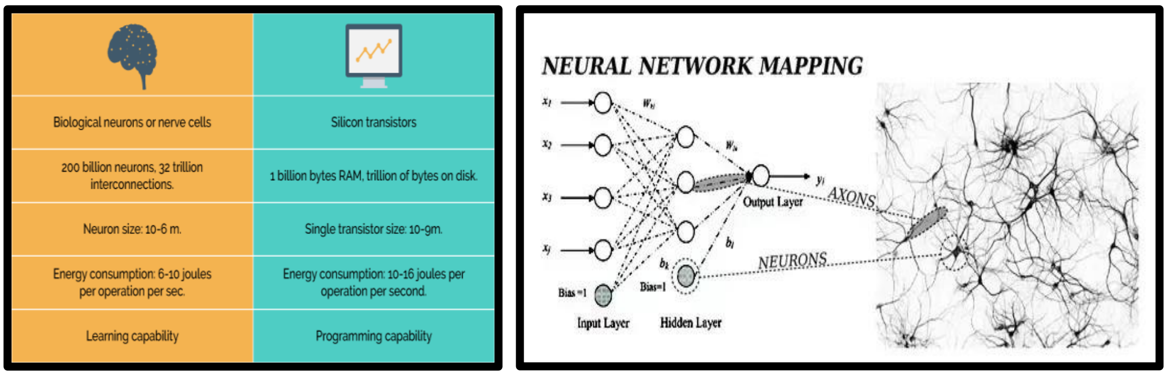
➢ **Advantages**

- Decision Trees are easy to explain. It results in a set of rules.
- It follows the same approach as humans generally follow while making decisions.
- Interpretation of a complex Decision Tree model can be simplified by its visualizations. Even a naive person can understand logic.
- The Number of hyper-parameters to be tuned is almost null.

➢ **Disadvantages**

- There is a high probability of overfitting in Decision Tree.
- Generally, it gives low prediction accuracy for a dataset as compared to other machine learning algorithms.
- Information gain in a decision tree with categorical variables gives a biased response for attributes with greater no. of categories.
- Calculations can become complex when there are many class labels.

2.3 NEURAL NETWORKS



Biological neurons or nerve cells	Silicon transistors
200 billion neurons, 32 trillion interconnections.	1 billion bytes RAM, trillion of bytes on disk.
Neuron size: 10-6 m.	Single transistor size: 10-9m.
Energy consumption: 6-10 joules per operation per sec.	Energy consumption: 10-16 joules per operation per second.
Learning capability	Programming capability

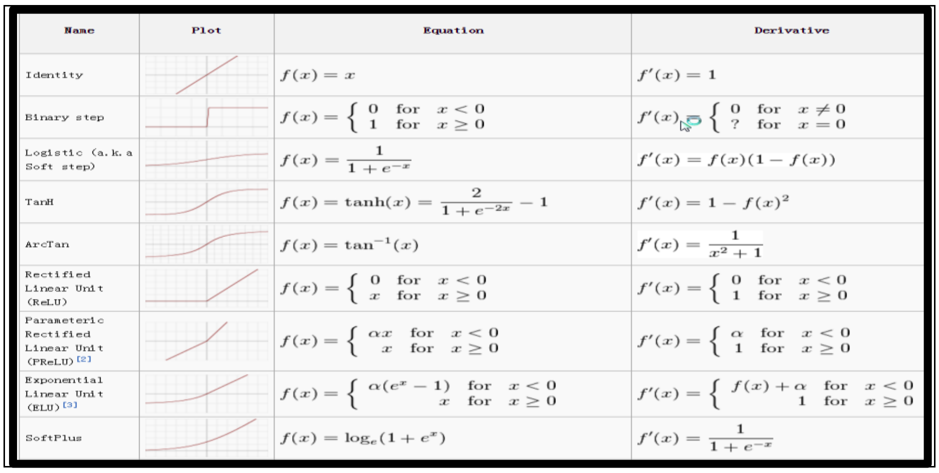
Fig 2.4 : Neural Network -1

A **neural network** is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. An effort is made to develop a network in order to mimic the cognitive abilities of the human brain such that the entire hardware can have the potential to make decisions that help the entire process to execute very efficiently An artificial neural network involves a network of simple processing elements (artificial neurons) which can exhibit complex global behavior, determined by the connections between the processing elements and element parameters. Artificial neurons were first proposed in 1943 by Warren McCulloch, a neurophysiologist, and Walter Pitts, a logician, who first collaborated at the University of Chicago. One classical type of artificial neural network is the recurrent Hopfield network. The concept of a neural network appears to have first been proposed by Alan Turing in his 1948 paper Intelligent Machinery in which called them "B-type unorganised machines".

The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations and also to use it. A neural network (NN), in the case of artificial neurons called artificial neural network (ANN) or simulated neural network (SNN), is an interconnected group of natural or artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation. In most cases, an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network. In more practical terms neural networks are non-linear statistical data modeling or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. Unsupervised neural networks can also be used to learn representations of the input that capture the salient characteristics of the input distribution, and more recently, deep learning algorithms, which can implicitly learn the distribution function of the observed data. Learning in neural networks is particularly useful in applications where the complexity of the data or task makes the design of such functions by hand impractical.. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and then summed. This activity is referred as a linear combination. Finally, an activation function controls the amplitude of the output. The tasks to which artificial neural networks are applied tend to fall within the following broad categories-

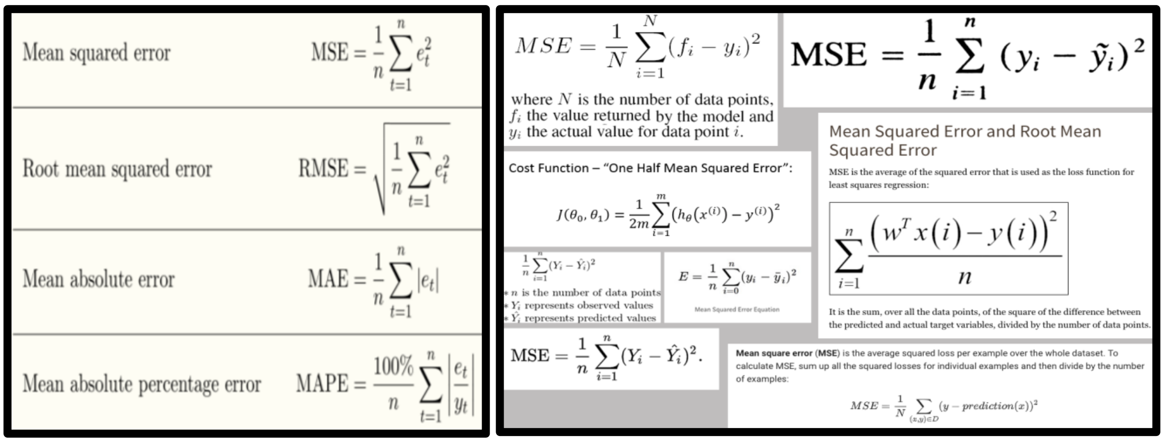
- Function approximation, or regression analysis, including time series prediction and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind signal separation and compression.

In any neural network, there are 2 propagations namely forward and backward propagation. The sum of the products of the inputs and their weights are calculated. This is then fed to the output. Hence the propagation when used together at once, it helps to update the weights in order to reduce the Mean Square Errors (MSE) or the Cost Function which are calculated based on the present value and the predicted value. Application areas of ANNs include nonlinear system identification and control (vehicle control, process control), game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications, data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering. These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Fig 2.5 : Activation Functions in Neural Networks



Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$

where N is the number of data points, f_i the value returned by the model and y_i the actual value for data point i.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

Cost Function – "One Half Mean Squared Error":

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Mean Squared Error and Root Mean Squared Error

MSE is the average of the squared error that is used as the loss function for least squares regression:

$$\sum_{i=1}^{n}\frac{\left(w^T x(i) - y(i)\right)^2}{n}$$

It is the sum, over all the data points, of the square of the difference between the predicted and actual target variables, divided by the number of data points.

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- n is the number of data points
- Y_i represents observed values
- $\hat{Y}_i$ represents predicted values

$$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)^2$$

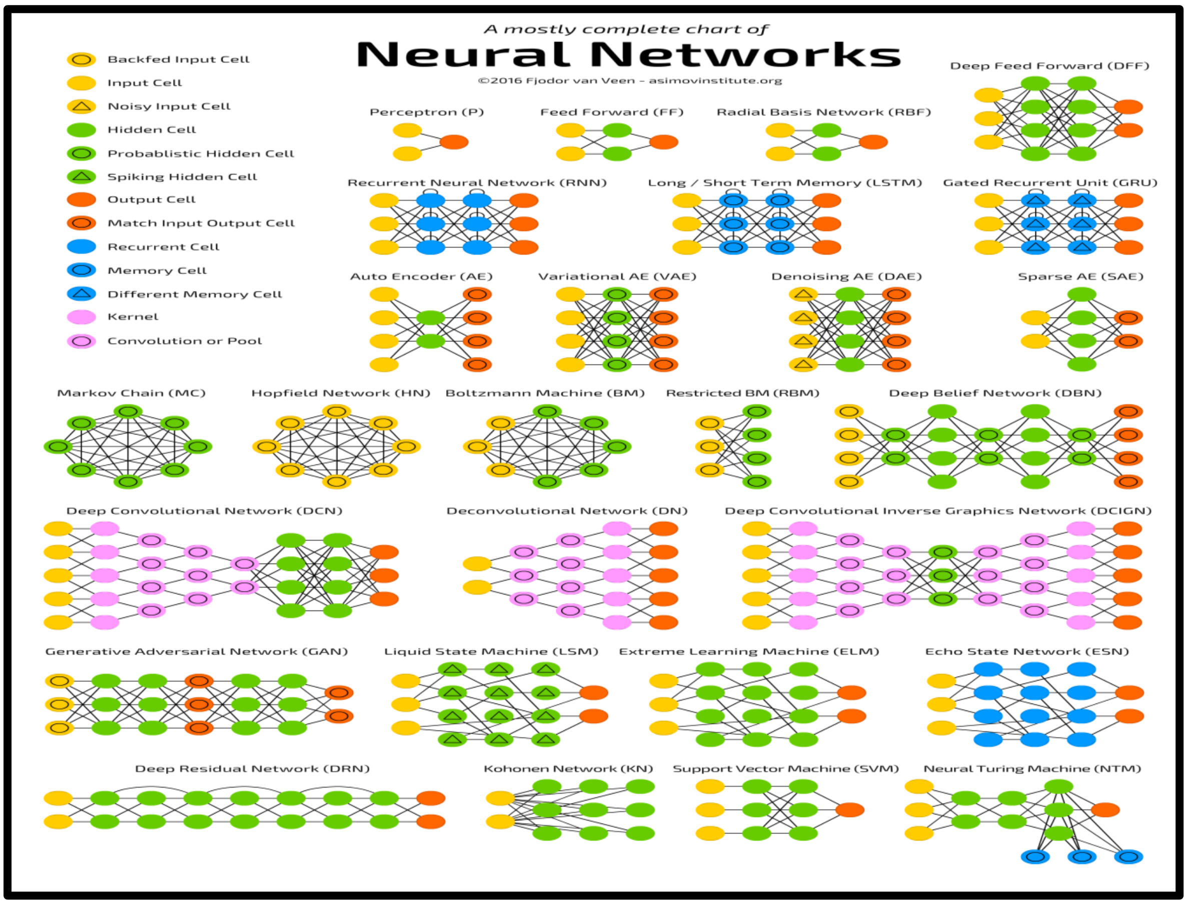
Mean Squared Error Equation

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Mean square error (**MSE**) is the average squared loss per example over the whole dataset. To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

$$MSE = \frac{1}{N}\sum_{(x,y)\in D}(y - prediction(x))^2$$

Fig 2.6: Error calculation in neural network



A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Fig 2.7: Types of Neural Networks applicable in Deep Learning

2.3.1 TYPES OF NEURAL NETWORKS

❖ **Feed Forward Neural Network – Artificial Neuron-** This is one of the simplest types of artificial neural networks. In a feed forward neural network, the data passes through the different input nodes till it reaches the output node. In other words, data moves in only one direction from the first tier onwards until it reaches the output node. This is also known as a front propagated wave which is usually achieved by using a classifying activation function. Unlike in more complex types of neural networks, there is no back propagation and data moves in one direction only. It may have a single or many have hidden layers. Feed forward neural networks are used in technologies like face recognition and computer vision. This is because the target classes in these applications are hard to classify. A simple feed forward neural network is equipped to deal with data which contains a lot of noise. Feed forward neural networks are also relatively simple to maintain.

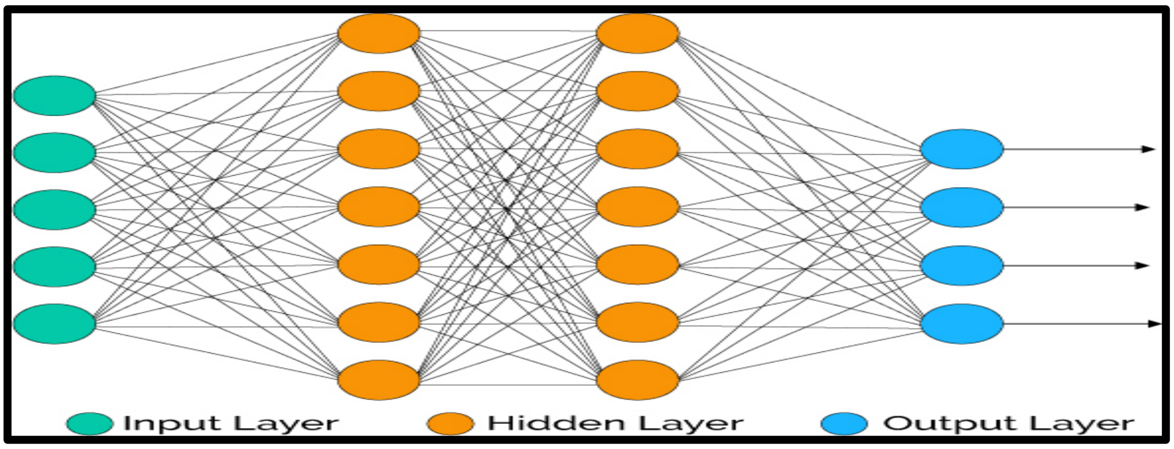


Fig 2.8: Artificial Neural Network (ANN)/ Feed Forward

❖ **Radial Basis Function Neural Network-** A radial basis function considers the distance of any point relative to the centre. Such neural networks have two layers. In the inner layer, the features are combined with the radial basis function. Then the output of these features is taken into account when calculating the same output in the next time-step. The radial basis function neural network is applied extensively in power restoration systems. Power systems have become bigger and more complex. This increases the risk of a blackout. This neural network is used in the power restoration systems in order to restore power in the shortest possible time.

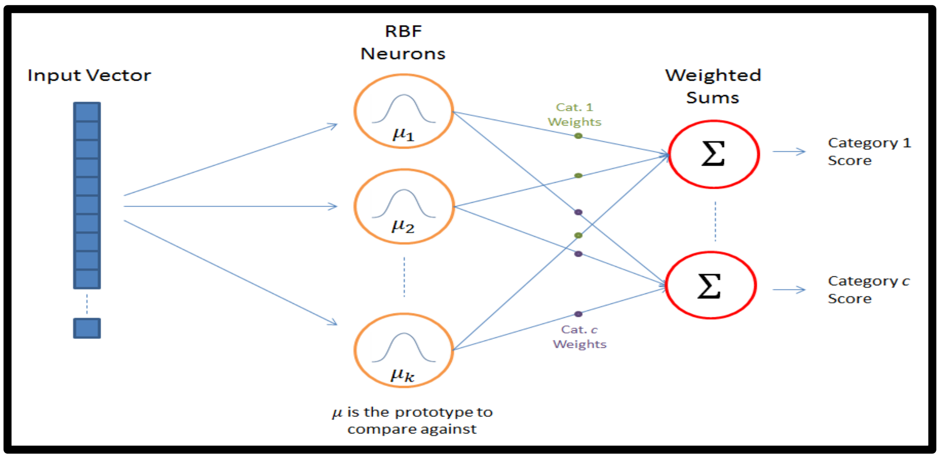


Fig 2.9: Radial Basis Function Neural Network

- **Multilayer Perceptron-** A multilayer perceptron has three or more layers. It is used to classify data that cannot be separated linearly. It is a type of artificial neural network that is fully connected. This is because every single node in a layer is connected to each node in the following layer. A multilayer perceptron uses a nonlinear activation function (mainly hyperbolic tangent or logistic function). It is applied extensively in speech recognition and machine translation technologies.

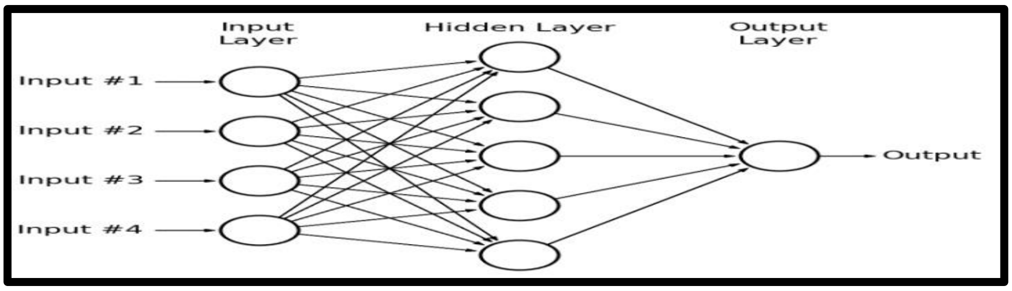


Fig 2.10: Multi Layer Perceptron

- **Recurrent Neural Network (RNN) – Long Short Term Memory-** A Recurrent Neural Network is a type of artificial neural network in which the output of a particular layer is saved and fed back to the input. This helps predict the outcome of the layer. The first layer is formed in the same way as it is in the feed forward network. That is, with the product of the sum of the weights and features. However, in subsequent layers, the recurrent neural network process begins. From each time-step to the next, each node will remember some information that it had in the previous time-step. In other words, each node acts as a memory cell while computing and carrying out operations. The neural network begins with the front propagation as usual but remembers the information it may need to use later. If the prediction is wrong, the system self-learns and works towards making the right prediction during the back propagation. This type of neural network is very effective in text-to-speech conversion technology.

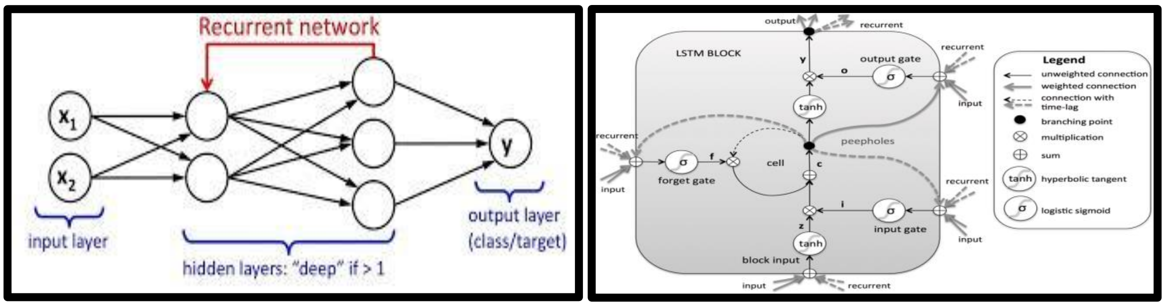


Fig 2.11: Recurrent Neural Network and LSTM

❖ **Deep Learning Neural Network**

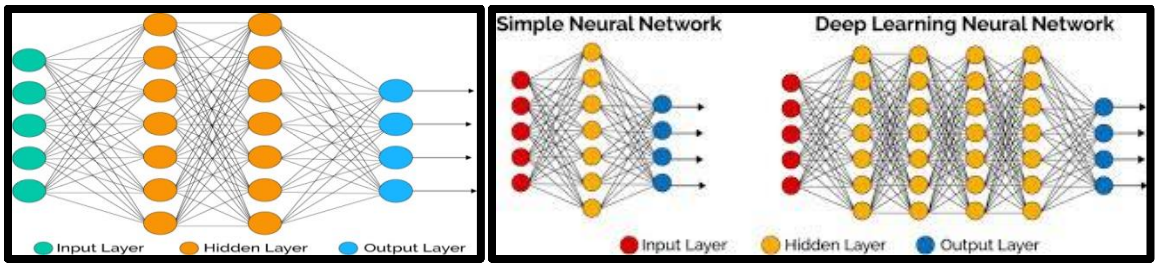


Fig 2.12: Deep Learning Neural Network (DNN)

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. For example, a DNN that is trained to recognize dog breeds will go over the given image and calculates the probability that the dog in the image is a certain breed. The user can review the results and select which probabilities the network should display (above a certain threshold, etc.) and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNN have many layers, hence the name "deep" networks.

DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives. The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network. Deep architectures include many variants of a few basic approaches. Each architecture has found success in specific domains. It is not always possible to compare the performance of multiple architectures, unless they have been evaluated on the same data sets.

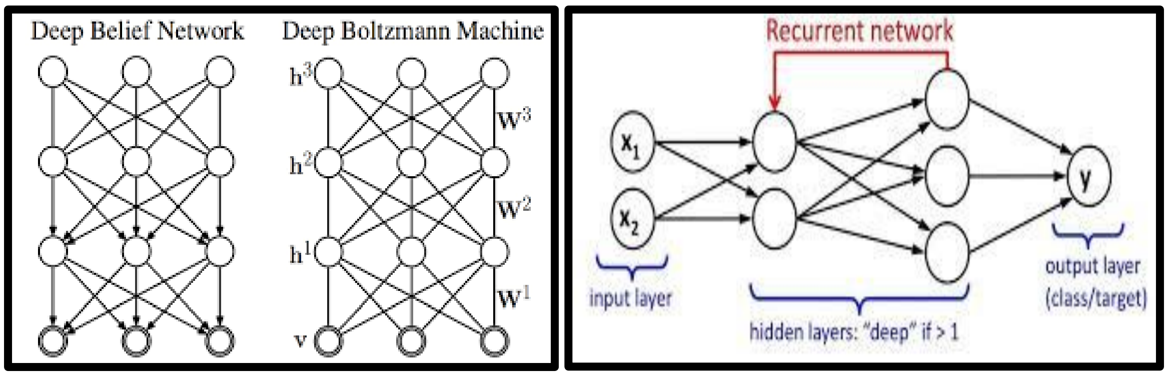


Fig 2.13: Deep Learning Neural Network

DNNs are typically feed forward networks in which data flows from the input layer to the output layer without looping back. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network didn’t accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data. Recurrent neural networks (RNNs), in which data can flow in any direction, are used for applications such as language modeling. Long short-term memory is particularly effective for this use. Convolutional deep neural networks (CNNs) are used in computer vision. CNNs also have been applied to acoustic modeling for automatic speech recognition (ASR).

- **Modular Neural Network-** A modular neural network has a number of different networks that function independently and perform sub-tasks. The different networks do not really interact with or signal each other during the computation process. They work independently towards achieving the output. As a result, a large and complex computational process can be done significantly faster by breaking it down into independent components. The computation speed increases because the networks are not interacting with or even connected to each other.

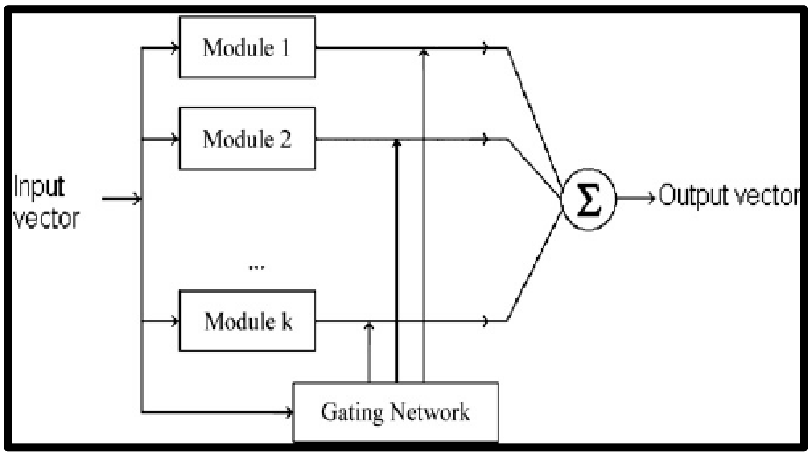


Fig 2.14: Modular Neural Network

- **Sequence-To-Sequence Models -** A sequence to sequence model consists of two recurrent neural networks. There's an encoder that processes the input and a decoder that processes the output. The encoder and decoder can either use the same or different parameters. This model is particularly applicable in those cases where the length of the input data is not the same as the length of the output data. Sequence-to-sequence models are applied mainly in chatbots, machine translation, and question answering systems.

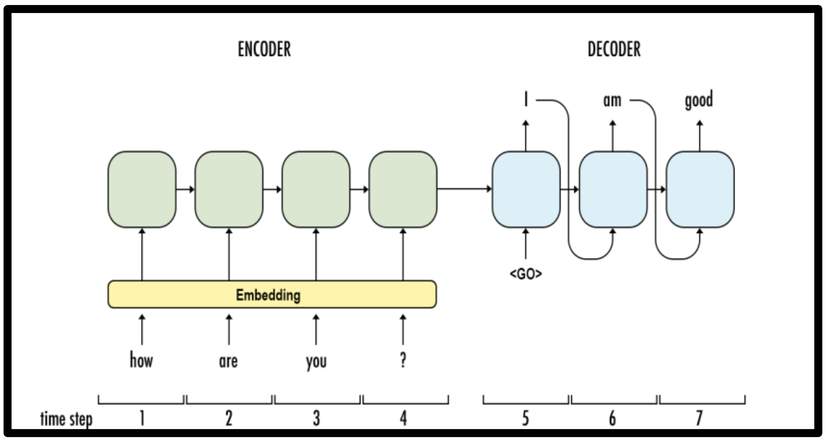


Fig 2.15: Sequence to Sequence Neural Network

- **Convolutional Neural Network-** A convolutional neural network(CNN) uses a variation of the multilayer perceptrons. A CNN contains one or more than one convolutional layers. These layers can either be completely interconnected or pooled. Before passing the result to the next layer, the convolutional layer uses a convolutional operation on the input. Due to this convolutional operation, the network can be much deeper but with much fewer parameters. Due to this ability, convolutional neural networks show very effective results in image and video recognition, natural language processing, and recommender systems.

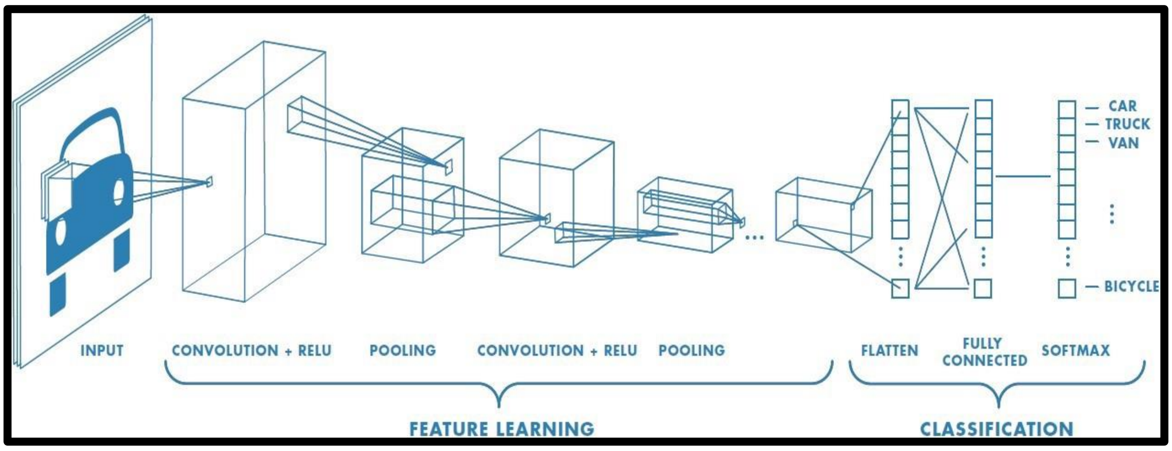


Fig 2.16: Convolutional Neural Network

Convolutional neural networks also show great results in semantic parsing and paraphrase detection. They are also applied in signal processing and image classification. CNNs are also being used in image analysis and recognition in agriculture where weather features are extracted from satellites like LSAT to predict the growth and yield of a piece of land. Here's an image of what a Convolutional Neural Network looks like.

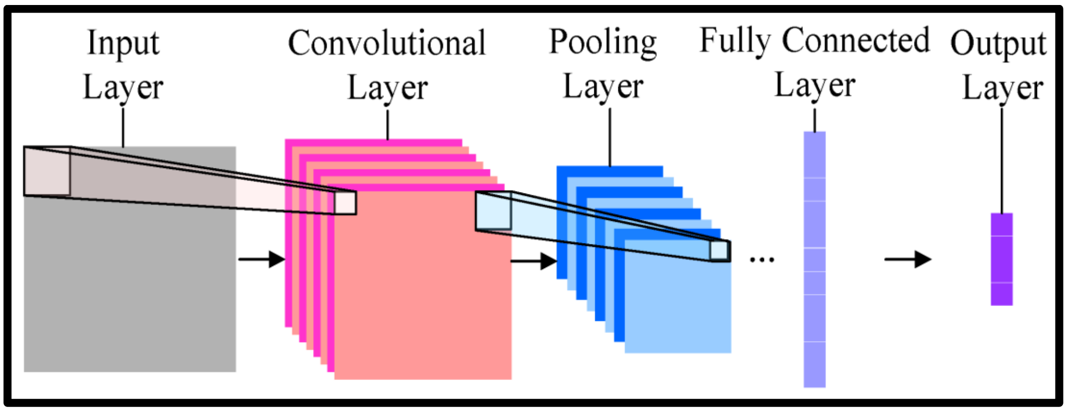


Fig 2.17: Stages in Convolutional Neural Network

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The final convolution, in turn, often involves back propagation in order to more accurately weight the end product. Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point.

- **Convolutional**

When programming a CNN, each convolutional layer within a neural network should have the following attributes: **Input is a tensor with shape (number of images) x (image width) x (image height) x (image depth).**

Convolutional kernels whose width and height are hyper-parameters, and whose depth must be equal to that of the image. Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable.

- **Pooling**

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the prior layer.

- **Fully connected**

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

- **Receptive field**

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

- **Weights**

Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning, in a neural network, progresses by making iterative adjustments to these biases and weights. The vector of weights and the bias are called filters and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, as opposed to each receptive field having its own bias and vector weighting.[1

2.4 NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) gives machines the ability to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level, but continue to lack the semantic understanding required to classify isolated sentences well. Besides the usual difficulties with encoding semantic commonsense knowledge, existing semantic NLP sometimes scales too poorly to be viable in business applications. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning.

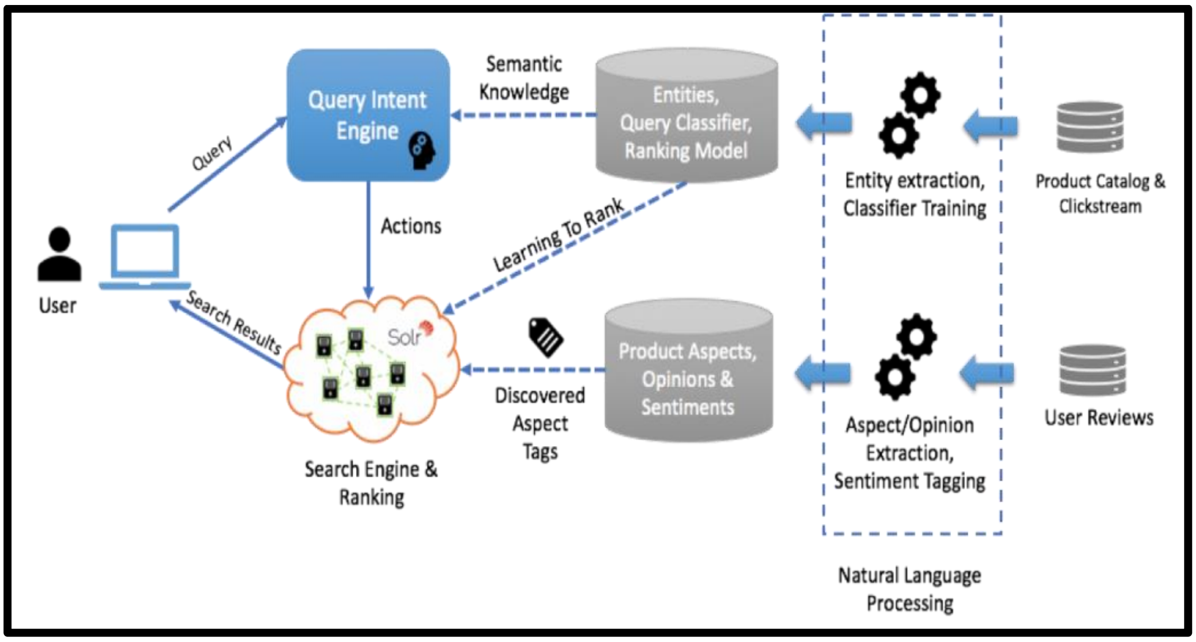

Query Intent Engine
Semantic Knowledge
Entities, Query Classifier, Ranking Model
Entity extraction, Classifier Training
Product Catalog & Clickstream
Query
Actions
Learning To Rank
User
Search Results
Solr
Product Aspects, Opinions & Sentiments
Discovered Aspect Tags
Aspect/Opinion Extraction, Sentiment Tagging
User Reviews
Search Engine & Ranking
Natural Language Processing

Fig 2.18: Natural Language Processing

2.5 BOLTZMANN MACHINE

A **Boltzmann machine** (also called **stochastic Hopfield network with hidden units**) is a type of stochastic recurrent neural network and Markov random field. Boltzmann machines can be seen as the stochastic, generative counterpart of Hopfield networks. They were one of the first neural networks capable of learning internal representations, and are able to represent and (given sufficient time) solve difficult combinatorial problems. They are named after the Boltzmann distribution in statistical mechanics, which is used in their sampling function. That's why they are called "energy based models" (EBM). They were invented in 1985 by Geoffrey Hinton.

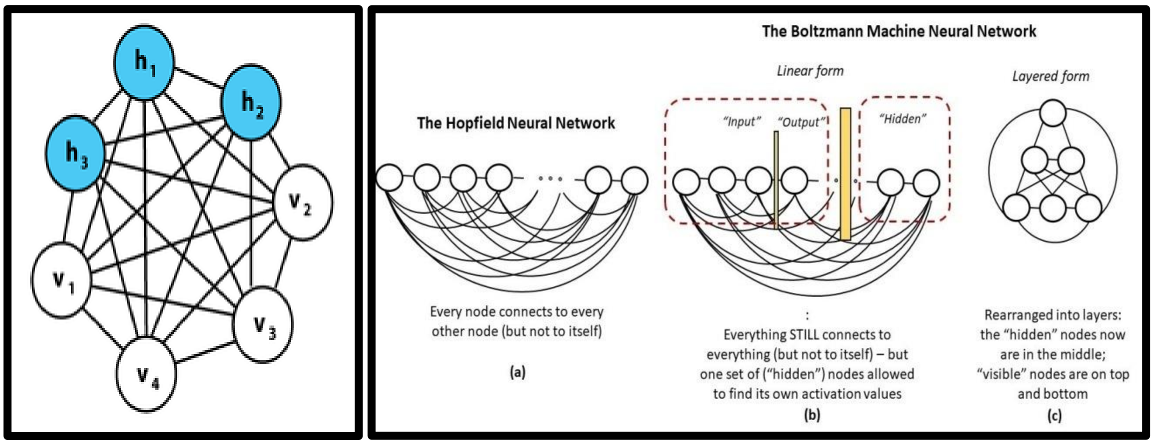
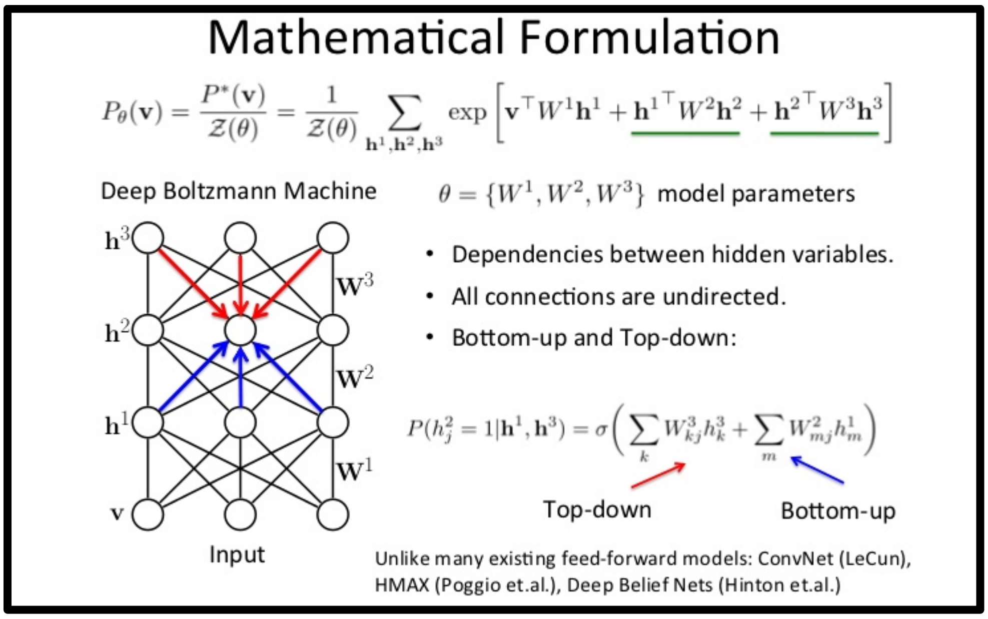

h1
h2
h3
v1
v2
v3
v4
The Boltzmann Machine Neural Network
The Hopfield Neural Network
Linear form
Layered form
"Input"
"Output"
"Hidden"
Every node connects to every other node (but not to itself)
(a)
Everything STILL connects to everything (but not to itself) – but one set of ("hidden") nodes allowed to find its own activation values
(b)
Rearranged into layers: the "hidden" nodes now are in the middle; "visible" nodes are on top and bottom
(c)

Fig 2.19: Boltzmann Neural Network



Mathematical Formulation

$$P_\theta(\mathbf{v}) = \frac{P^*(\mathbf{v})}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + \underline{\mathbf{h}^{1\top} W^2 \mathbf{h}^2} + \underline{\mathbf{h}^{2\top} W^3 \mathbf{h}^3}\right]$$

Deep Boltzmann Machine

$\theta = \{W^1, W^2, W^3\}$ model parameters

- Dependencies between hidden variables.
- All connections are undirected.
- Bottom-up and Top-down:

$$P(h_j^2 = 1 | \mathbf{h}^1, \mathbf{h}^3) = \sigma\left(\sum_k W_{kj}^3 h_k^3 + \sum_m W_{mj}^2 h_m^1\right)$$

Top-down

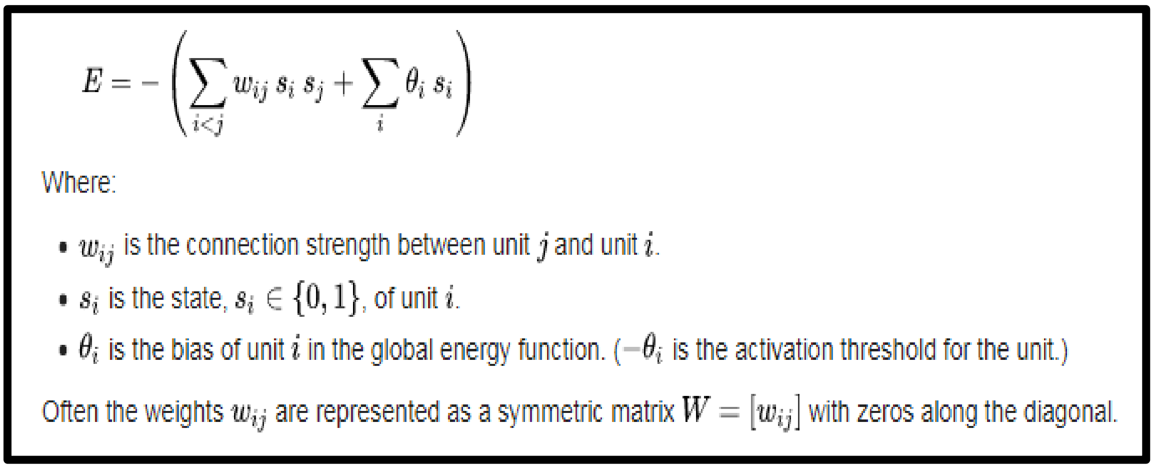
Bottom-up

Unlike many existing feed-forward models: ConvNet (LeCun), HMAX (Poggio et.al.), Deep Belief Nets (Hinton et.al.)

Fig 2.19: Mathematical Formulation of the BNN.

They are theoretically intriguing because of the locality and Hebbian nature of their training algorithm (being trained by Hebb's rule), and because of their parallelism and the resemblance of their dynamics to simple physical processes. Boltzmann machines with unconstrained connectivity have not proven useful for practical problems in machine learning or inference, but if the connectivity is properly constrained, the learning can be made efficient enough to be useful for practical problems.

A Boltzmann machine, like a Hopfield network, is a network of units with an "energy" (Hamiltonian) defined for the overall network. Its units produce binary results.Unlike Hopfield nets, Boltzmann machine units are stochastic.



$$E = -\left(\sum_{i<j} w_{ij}\, s_i\, s_j + \sum_i \theta_i\, s_i\right)$$

Where:

- w_{ij} is the connection strength between unit j and unit i.
- s_i is the state, $s_i \in \{0, 1\}$, of unit i.
- θ_i is the bias of unit i in the global energy function. ($-\theta_i$ is the activation threshold for the unit.)

Often the weights w_{ij} are represented as a symmetric matrix $W = [w_{ij}]$ with zeros along the diagonal.

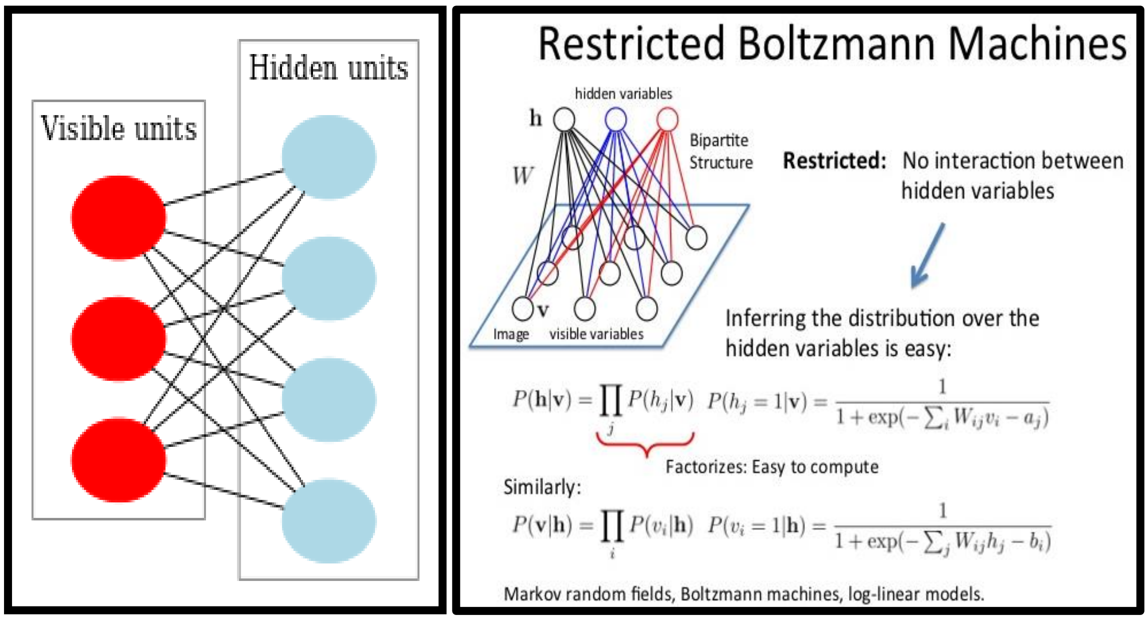
Fig 2.20: Weight function and the Cost function.

The network runs by repeatedly choosing a unit and resetting its state. After running for long enough at a certain temperature, the probability of a global state of the network depends only upon that global state's energy, according to a Boltzmann distribution, and not on the initial state from which the process was started. This means that log-probabilities of global states become linear in their energies. This relationship is true when the machine is "at thermal equilibrium", meaning that the probability distribution of global states has converged. Running the network beginning from a high temperature, its temperature gradually decreases until reaching a thermal equilibrium at a lower temperature. It then may converge to a distribution where the energy level fluctuates around the global minimum. This process is called simulated annealing.To train the network so that the chance it will converge to a global state is according to an external distribution over these states, the weights must be set so that the global states with the highest probabilities get the lowest energies. This is done by training.

Remarkably, this learning rule is fairly biologically plausible because the only information needed to change the weights is provided by "local" information. That is, the connection (or synapse biologically speaking) does not need information about anything other than the two neurons it connects. This is far more biologically realistic than the information needed by a connection in many other neural network training algorithms, such as backpropagation.

The training of a Boltzmann machine does not use the EM algorithm, which is heavily used in machine learning. By minimizing the KL-divergence, it is equivalent to maximizing the log-likelihood of the data. Therefore, the training procedure performs gradient ascent on the log-likelihood of the observed data. This is in contrast to the EM algorithm, where the posterior distribution of the hidden nodes must be calculated before the maximization of the expected value of the complete data likelihood during the M-step. Training the biases is similar, but uses only single node activity:

❖ RESTRICTED BOLTZMANN MACHINE



Restricted: No interaction between hidden variables

Inferring the distribution over the hidden variables is easy:

$$P(\mathbf{h}|\mathbf{v}) = \underbrace{\prod_j P(h_j|\mathbf{v})}_{\text{Factorizes: Easy to compute}} \quad P(h_j = 1|\mathbf{v}) = \frac{1}{1+\exp(-\sum_i W_{ij}v_i - a_j)}$$

Similarly:

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1+\exp(-\sum_j W_{ij}h_j - b_i)}$$

Markov random fields, Boltzmann machines, log-linear models.

Fig 2.21: Restricted Boltzmann Neural Network

Graphical representation of a restricted Boltzmann machine. The four blue units represent hidden units, and the three red units represent visible states. In restricted Boltzmann machines there are only connections (dependencies) between hidden and visible units, and none between units of the same type (no hidden-hidden, nor visible-visible connections).

Although learning is impractical in general Boltzmann machines, it can be made quite efficient in an architecture called the "restricted Boltzmann machine" or "RBM" which does not allow intra-layer connections between hidden units. After training one RBM, the activities of its hidden units can be treated as data for training a higher-level RBM. This method of stacking RBMs makes it possible to train many layers of hidden units efficiently and is one of the most common deep learning strategies. As each new layer is added the overall generative model gets better. There is an extension to the restricted Boltzmann machine that affords using real valued data rather than binary data. Along with higher order Boltzmann machines, it is outlined here. One example of a practical application of Restricted Boltzmann machines is the performance improvement of speech recognition software.

- **DEEP BOLTZMANN MACHINE**

A deep Boltzmann machine (DBM) is a type of binary pair-wise Markov random field (undirected probabilistic graphical model) with multiple layers of hidden random variables. It is a network of symmetrically coupled stochastic binary units. In a DBN only the top two layers form a restricted Boltzmann machine (which is an undirected graphical model), while lower layers form a directed generative model. In a DBM all layers are symmetric and undirected.

Like DBNs, DBMs can learn complex and abstract internal representations of the input in tasks such as object or speech recognition, using limited, labeled data to fine-tune the representations built using a large supply of unlabeled sensory input data. However, unlike DBNs and deep convolutional neural networks, they adopt the inference and training procedure in both directions, bottom-up and top-down pass, which allow the Deep Boltzmann machine to better unveil the representations of the input structures.

However, the slow speed of DBMs limits their performance and functionality. Because exact maximum likelihood learning is intractable for DBMs, only approximate maximum likelihood learning is possible. Another option is to use mean-field inference to estimate data-dependent expectations and approximate the expected sufficient statistics by using Markov chain Monte Carlo (MCMC).[6] This approximate inference, which must be done for each test input, is about 25 to 50 times slower than a single bottom-up pass in DBMs. This makes joint optimization impractical for large data sets, and restricts the use of DBMs for tasks such as feature representation.

2.6 AUTO ENCODERS

An **autoencoder** is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal “noise”. Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input, hence its name. Several variants exist to the basic model, with the aim of forcing the learned representations of the input to assume useful properties. Examples are the regularized autoencoders (Sparse, Denoising and Contractive autoencoders), proven effective in learning representations for subsequent classification tasks, and Variational autoencoders, with their recent applications as generative models.

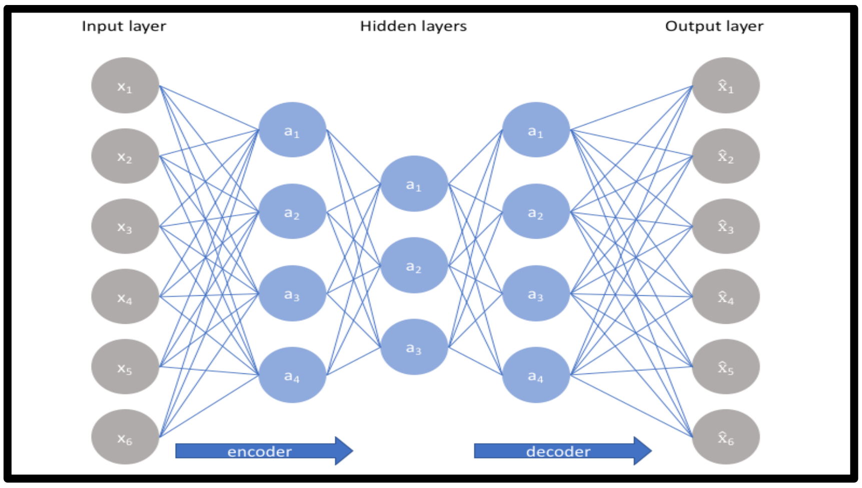


Fig 2.22: Auto Encoders

An autoencoder is a neural network that learns to copy its input to its output. It has an internal (hidden) layer that describes a code used to represent the input, and it is constituted by two main parts: an encoder that maps the input into the code, and a decoder that maps the code to a reconstruction of the original input. Performing the copying task per se would be meaningless, and this is why usually autoencoders are restricted in ways that force them to reconstruct the input only approximately, prioritizing the most relevant aspects of the data to be copied.

- **Types of Autoencoders :**

1. **Denoising autoencoder**

2. **Sparse autoencoder**

3. **Variational autoencoder** (VAE)

4. **Contractive autoencoder (CAE)**

A. Denoising autoencoder :

It is one of the basic autoencoder which takes a partially corrupted inputs randomly to address the identity-function risk, which autoencoder has to recover or denoise. This technique has been introduced with a specific approach to **good**representation. A good representation is one that can be obtained robustly from a corrupted input and that will be useful for recovering the corresponding clean input. The idea behind denoising autoencoders is simple. In order to force the hidden layer to discover more robust features and prevent it from simply learning the identity, we train the autoencoder to reconstruct the input from a corrupted version of it. The amount of noise to apply to the input takes the form of a percentage. Typically, 30 percent, or 0.3, is fine, but if you have very little data, you may want to consider adding more.

B. Stacked Denoising Autoencoders (SDA):

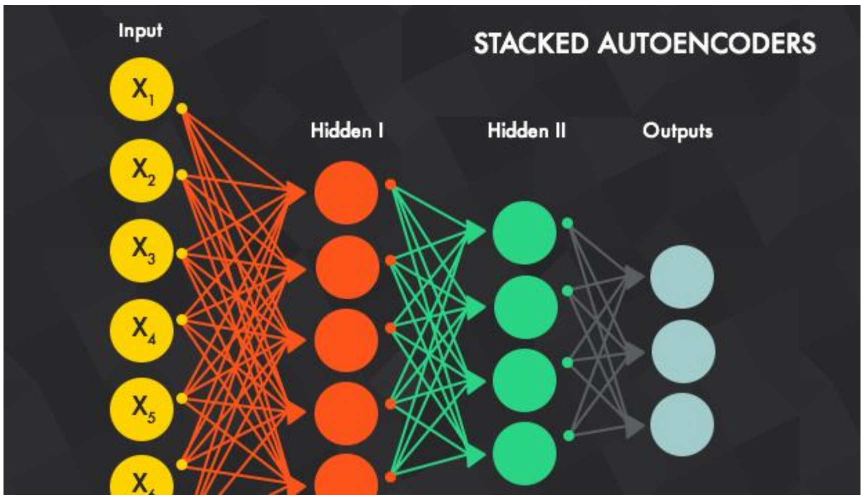


Fig 2.23: Stacked Denoising Auto Encoders

It a kind of denoising encoders which uses unsupervised pre-training mechanism on their layers, where once each layer is pre-trained to conduct feature selection and extraction on the input from the preceding layer, a second stage of supervised fine-tuning can follow. SDA is simply a multiple denoising autoencoders strung together. Once the first k layers are trained, we can train the k+1-th layer because we can now compute the code or latent representation from the layer below. Once all layers are pre-trained, the network goes through a second stage of training called **fine-tuning**. We go for supervised learning mechanism her to fine-tune in order to minimise the prediction error on supervised task. We then train the entire network as we would train a multilayer perceptron. At this point, we only consider the encoding parts of each auto-encoder. This stage is supervised, since now we use the target class during training. During pre-training we use the first facade, i.e., we treat our model as a list of autoencoders, and train each autoencoder seperately. In the second stage of training, we use the second facade. These two facades are linked because:

- the autoencoders and the sigmoid layers of the MLP share parameters, and
- the latent representations computed by intermediate layers of the MLP are fed as input to the autoencoders.

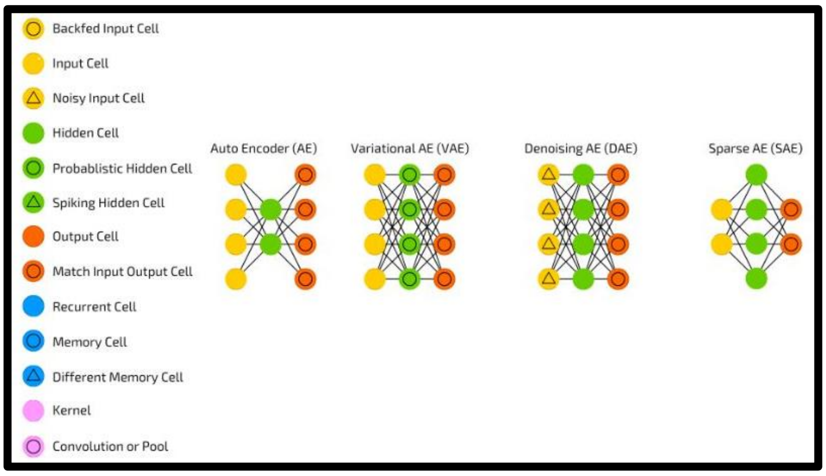


Fig 2.24: Types of Auto Encoders

The idea of auto encoders has been popular in the field of neural networks for decades, and the first applications date back to the '80s. Their most traditional application was dimensionality reduction or feature learning, but more recently the autoencoder concept has become more widely used for learning generative models of data. Some of the most powerful AIs in the 2010s involved sparse autoencoders stacked inside of deep neural networks.

CHAPTER-3

PROCEDURAL DESCRIPTION AND PROGRAM CODE AND OBJECTIVES

3.1 TENSOR FLOW PACKAGE

Tensor flow is a free and open source software library used for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. Among the applications for which Tensor Flow is the foundation, are automated image-captioning software, such as DeepDream. Deep learning relies on a lot of matrix multiplication. Tensor Flow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, Tensor Flow can be accessed and controlled by other languages mainly, Python. Tensor Flow is the best library of all because it is built to be accessible for everyone. Tensor flow library incorporates different API to built at scale deep learning architecture like CNN or RNN. Tensor Flow is based on graph computation; it allows the developer to visualize the construction of the neural network with Tensorboad. This tool is helpful to debug the program. Finally, Tensor flow is built to be deployed at scale. It runs on CPU and GPU. Finally, a significant feature of Tensor Flow is the TensorBoard. The TensorBoard enables to monitor graphically and visually what Tensor Flow is doing.

Tensor Flow can hardware, and software requirements can be classified into

- **Development Phase-** This is when you train the mode. Training is usually done on your Desktop or laptop.
- **Run Phase or Inference Phase-** Once training is done Tensor flow can be run on many different platforms. You can run it on

➢ **Tensor flow architecture works in three parts**

- Preprocessing the data
- Build the model
- Train and estimate the model

It is called Tensorflow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

❖ COMPONENTS OF TENSORFLOW

- **Tensor- Tensor** flow’s name is directly derived from its core framework: Tensor. In Tensor flow, all the computations involve tensors. A tensor is a vector or matrix of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) shape. The shape of the data is the dimensionality of the matrix or array. A tensor can be originated from the input data or the result of a computation. In Tensor Flow, all the operations are conducted inside a graph. The graph is a set of computation that takes place successively. Each operation is called an op node and are connected to each other. The graph outlines the ops and connections between the nodes. However, it does not display the values. The edge of the nodes is the tensor, i.e., a way to populate the operation with data.
- **Graphs-** Tensor Flow makes use of a graph framework. The graph gathers and describes all the series computations done during the training. The graph has lots of advantages as it was done to run on multiple CPUs or GPUs and even mobile operating system. The portability of the graph allows preserving the computations for immediate or later use. The graph can be saved to be executed in the future. All the computations in the graph are done by connecting tensors together. A tensor has a node and an edge. The node carries the mathematical operation and produces an endpoints outputs. The edges the edges explain the input/output relationships between nodes.

3.2 KERAS PACKAGE

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensor flow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. So Keras is high-level API wrapper for the low-level API, capable of running on top of Tensor Flow, CNTK, or Theano. Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine. Backend is a term in Keras that performs all low-level computation such as tensor products, convolutions and many other things with the help of other libraries such as Tensor flow or Theano. So, the "backend engine" will perform the computation and development of the models. Tensor flow is the default "backend engine" but we can change it in the configuration.

Parameters	Keras	Tensor flow
Type	High-Level API Wrapper	Low-Level API
Complexity	Easy to use if you Python language	You need to learn the syntax of using some of Tensor flow function
Purpose	Rapid deployment for making model with standard layers	Allows you to make an arbitrary computational graph or model layers
Tools	Uses other API debug tool such as TFDBG	You can use Tensor board visualization tools
Community	Large active communities	Large active communities and widely shared resources

Table No.1: Keras v/s Tensor flow

Advantages of Keras is Fast Deployment and Easy to understand, Keras is very quick to make a network model. If you want to make a simple network model with a few lines, Keras can help you with that.

```
from keras.models import Sequential
from keras.layers import Dense, Activation
model = Sequential()
model.add(Dense(64, activation='relu', input_dim=50)) #input shape of 50
```

model.add(Dense(28, activation='relu')) #input shape of 50
model.add(Dense(10, activation='softmax'))

Because of friendly the API, we can easily understand the process. Writing the code with a simple function and no need to set multiple parameters.

Large Community Support, There are lots of AI communities that use Keras for their Deep Learning framework. Many of them publish their codes as well tutorial to the general public. Have multiple Backend, You can choose Tensor flow, CNTK, and Theano as your backend with Keras. You can choose a different backend for different projects depending on your needs. Each backend has its own unique advantage. Cross-Platform and Easy Model Deployment, with a variety of supported devices and platforms, you can deploy Keras on any device like iOS with CoreML, Android with Tensor flow Android, Web browser with .js support, Cloud engine, Raspberry Pi. Multi GPUs Support, You can train Keras with on a single GPU or use multiple GPUs at once. Because Keras has a built-in support for data parallelism so it can process large volumes of data and speed up the time needed to train it.

Disadvantages of Keras is that it cannot handle low-level API, Keras only handles high-level API which runs on top other framework or backend engine such as Tensor flow, Theano, or CNTK. So it's not very useful if you want to make your own abstract layer for your research purposes because Keras already have pre-configured layers.

3.3 OPENCV PACKAGE

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language it is library used for Image Processing. It is mainly used to do all the operation related to Images. **OpenCV** (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license. OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe. It supports languages like

C++, JAVA, Android SDK, Python and C. It can do operations like:
1. Read and Write Images.

2. Detection of faces and its features.

3. Detection of shapes like Circle, rectangle etc in a image. E.

4. Text recognition in images. e.g Reading Number Plates

5. Modifying image quality and colors e.g Instagram, CamScanner.

6. Developing Augmented reality apps.

Advantages of the Opencv package is that it is simple to learn and lots of tutorial are available, adaptable to any programming language and it is free to use.

OpenCV's application areas include 2D and 3D feature toolkits, Egomotion estimation, Facial recognition system, Gesture recognition, Human–computer interaction (HCI), Mobile robotics, Motion understanding, Object identification, Segmentation and recognition, Stereopsis stereo vision: depth perception from 2 cameras, Structure from motion (SFM), Motion tracking, Augmented reality.

To support some of the above areas, OpenCV includes a statistical machine learning library that contains Boosting, Decision tree learning, Gradient boosting trees, Expectation-maximization algorithm, k-nearest neighbor algorithm, Naive Bayes classifier, Artificial neural networks, Random forest, Support vector machine (SVM), Deep neural networks (DNN).

These packages are used in our project to determine the age and gender of the individual from the image dataset. The model is trained where the train dataset is larger than the test dataset such that, the overall accuracy achieved is ranging between 75 to 80 %. It will be useful in real time applications like in the ticket bookings, shopping malls and various other applications that need to input the personal data of the individual or any living being. The name of the input dataset is net.caffemodel that has been taken for the data to be trained for the model

3.4 PROGRAM CODE

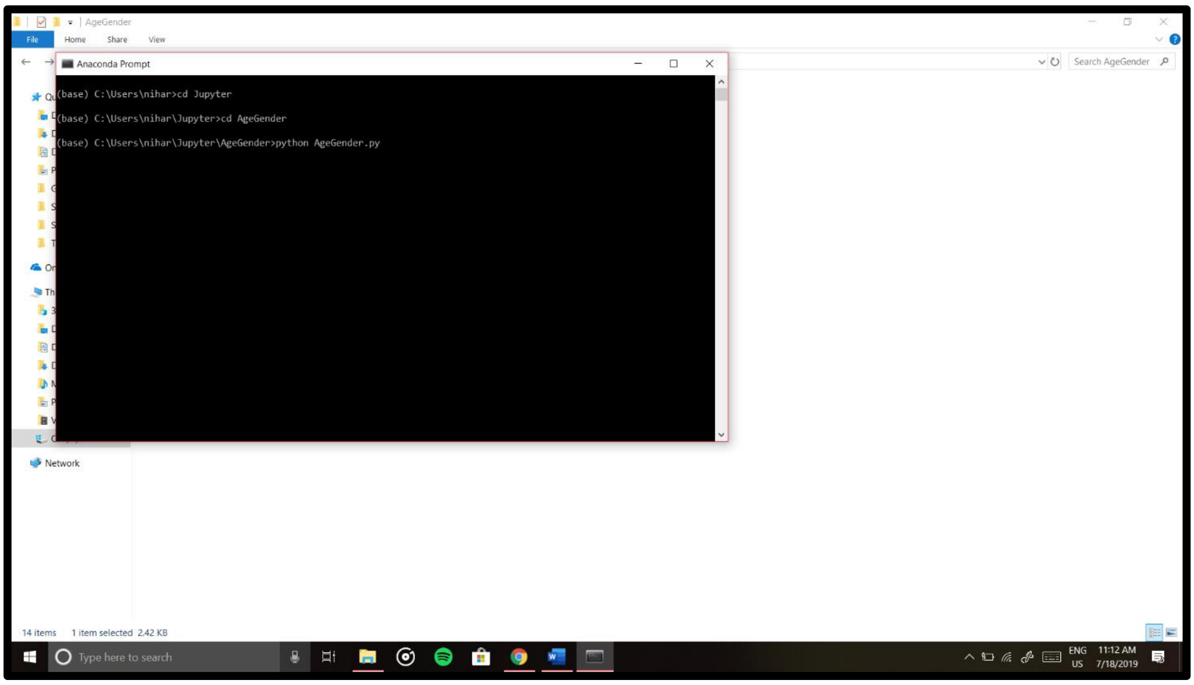
```
# Import required modules

import cv2 as cv

import math

import time

import argparse



def getFaceBox(net, frame, conf_threshold=0.7):

frameOpencvDnn = frame.copy()

frameHeight = frameOpencvDnn.shape[0]

frameWidth = frameOpencvDnn.shape[1]

    blob = cv.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], True,
False)


net.setInput(blob)

    detections = net.forward()

bboxes = []

    for i in range(detections.shape[2]):

        confidence = detections[0, 0, i, 2]

        if confidence >conf_threshold:

            x1 = int(detections[0, 0, i, 3] * frameWidth)

            y1 = int(detections[0, 0, i, 4] * frameHeight)

            x2 = int(detections[0, 0, i, 5] * frameWidth)

            y2 = int(detections[0, 0, i, 6] * frameHeight)

bboxes.append([x1, y1, x2, y2])

cv.rectangle(frameOpencvDnn, (x1, y1), (x2, y2), (0, 255, 0),
int(round(frameHeight/150)), 8)

    return frameOpencvDnn, bboxes
```

```
parser = argparse.ArgumentParser(description='Use this script to run age and gender
recognition using OpenCV.')

parser.add_argument('--input', help='Path to input image or video file. Skip this argument
to capture frames from a camera.')


args = parser.parse_args()


faceProto = "opencv_face_detector.pbtxt"

faceModel = "opencv_face_detector_uint8.pb"


ageProto = "age_deploy.prototxt"

ageModel = "age_net.caffemodel"


genderProto = "gender_deploy.prototxt"

genderModel = "gender_net.caffemodel"


MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)

ageList = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']

genderList = ['Male', 'Female']


# Load network

ageNet = cv.dnn.readNet(ageModel, ageProto)

genderNet = cv.dnn.readNet(genderModel, genderProto)

faceNet = cv.dnn.readNet(faceModel, faceProto)


# Open a video file or an image file or a camera stream

cap = cv.VideoCapture(args.input if args.input else 0)

padding = 20
```

```
while cv.waitKey(1) < 0:
    # Read frame
    t = time.time()
hasFrame, frame = cap.read()
    if not hasFrame:
cv.waitKey()
        break


frameFace, bboxes = getFaceBox(faceNet, frame)
    if not bboxes:
print("No face Detected, Checking next frame")
        continue


    for bbox in bboxes:
        # print(bbox)
        face = frame[max(0,bbox[1]-padding):min(bbox[3]+padding,frame.shape[0]-
1),max(0,bbox[0]-padding):min(bbox[2]+padding, frame.shape[1]-1)]


        blob = cv.dnn.blobFromImage(face, 1.0, (227, 227), MODEL_MEAN_VALUES,
swapRB=False)
genderNet.setInput(blob)
genderPreds = genderNet.forward()
        gender = genderList[genderPreds[0].argmax()]
        # print("Gender Output : {}".format(genderPreds))
print("Gender : {}, conf = {:.3f}".format(gender, genderPreds[0].max()))


ageNet.setInput(blob)
agePreds = ageNet.forward()
        age = ageList[agePreds[0].argmax()]
print("Age Output : {}".format(agePreds))
print("Age : {}, conf = {:.3f}".format(age, agePreds[0].max()))


        label = "{},{}".format(gender, age)
cv.putText(frameFace, label, (bbox[0], bbox[1]-10), cv.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2, cv.LINE_AA)
cv.imshow("Age Gender Demo", frameFace)
        # cv.imwrite("age-gender-out-{}".format(args.input),frameFace)
print("time : {:.3f}".format(time.time() - t))
```

Command line to give an image as an input: **python AgeGender.py --input <image path>**

Command line to give input through webcam: **python AgeGender.py**



```
(base) C:\Users\nihar>cd Jupyter

(base) C:\Users\nihar\Jupyter>cd AgeGender

(base) C:\Users\nihar\Jupyter\AgeGender>python AgeGender.py
```

Fig 3.1: Input Command window

CHAPTER-4

RESULTS

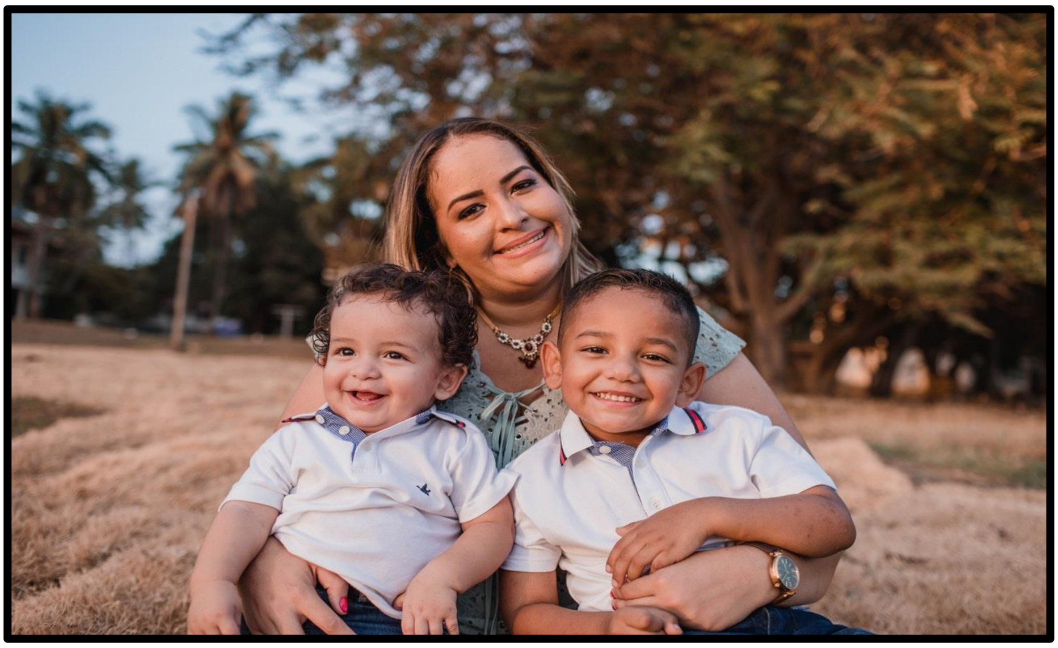


Fig 4.1: Input test image-1

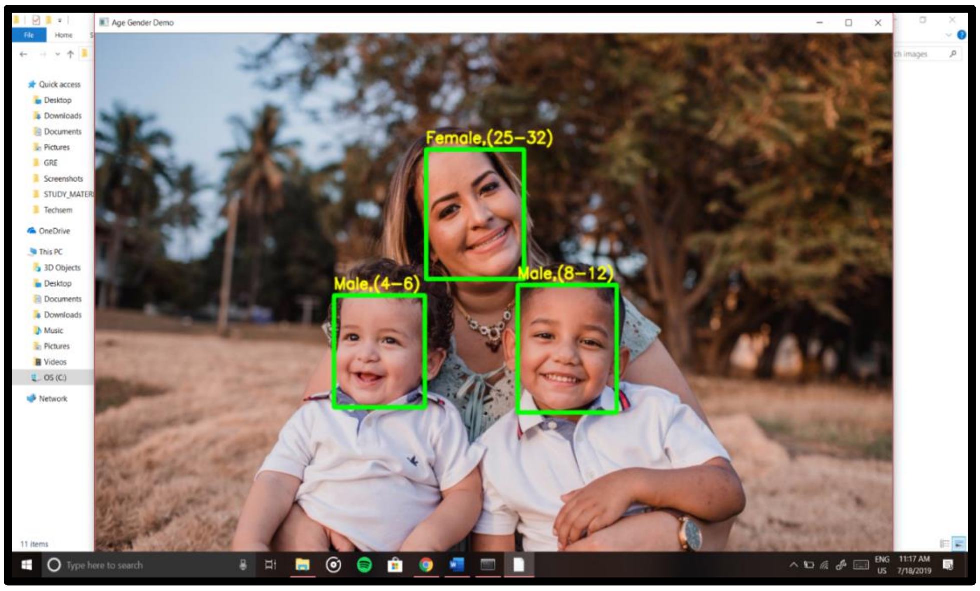


Fig 4.2: Predicted Output

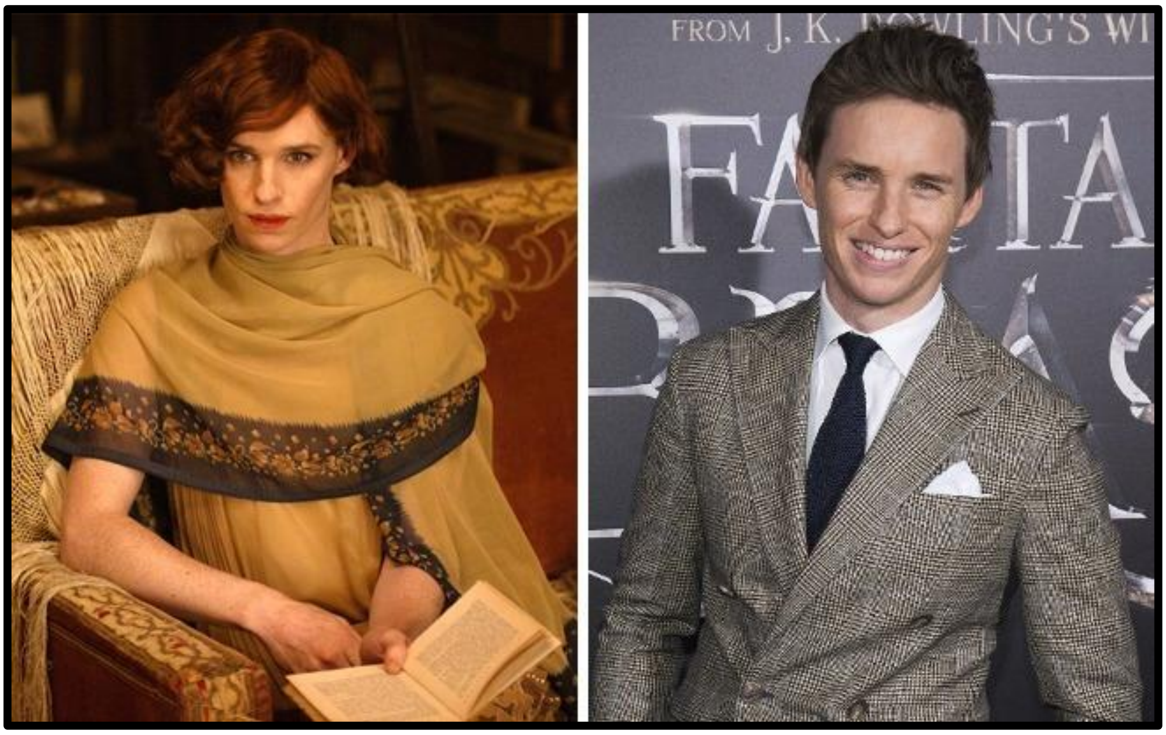
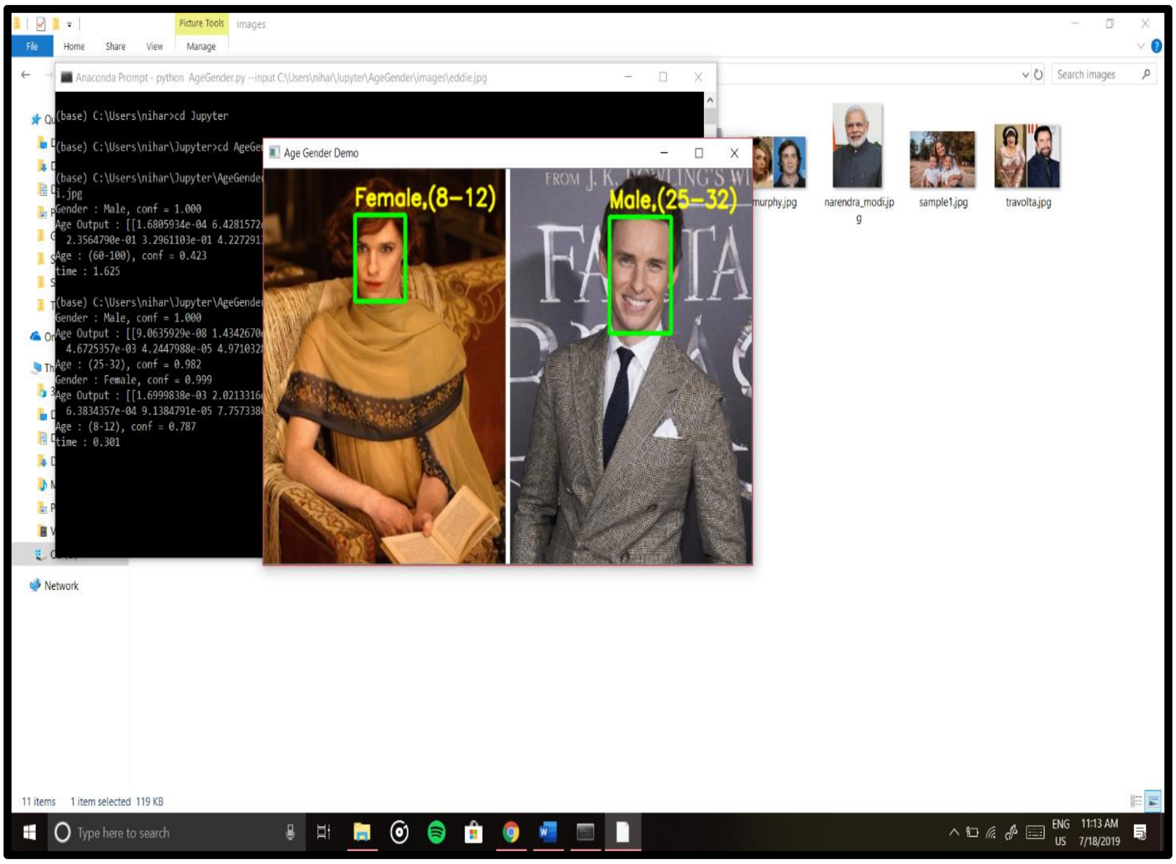


Fig 4.2: Input Test image-2



```
(base) C:\Users\nihar>cd Jupyter

(base) C:\Users\nihar\Jupyter>cd AgeGe

(base) C:\Users\nihar\Jupyter\AgeGende
i.jpg
Gender : Male, conf = 1.000
Age Output : [[1.6805934e-04 6.4281572
  2.3564790e-01 3.2961103e-01 4.227291
Age : (60-100), conf = 0.423
time : 1.625

(base) C:\Users\nihar\Jupyter\AgeGende
Gender : Male, conf = 1.000
Age Output : [[9.0635929e-08 1.4342670
  4.6725357e-03 4.2447988e-05 4.971032
Age : (25-32), conf = 0.982
Gender : Female, conf = 0.999
Age Output : [[1.6999838e-03 2.0213316
  6.3834357e-04 9.1384791e-05 7.757338
Age : (8-12), conf = 0.787
time : 0.301
```

Fig 4.4: Predicted Output.

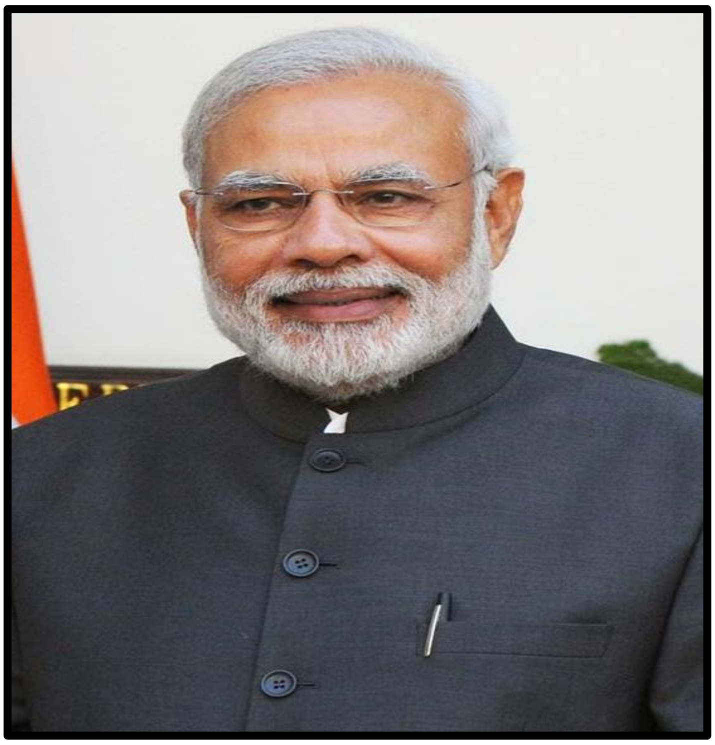
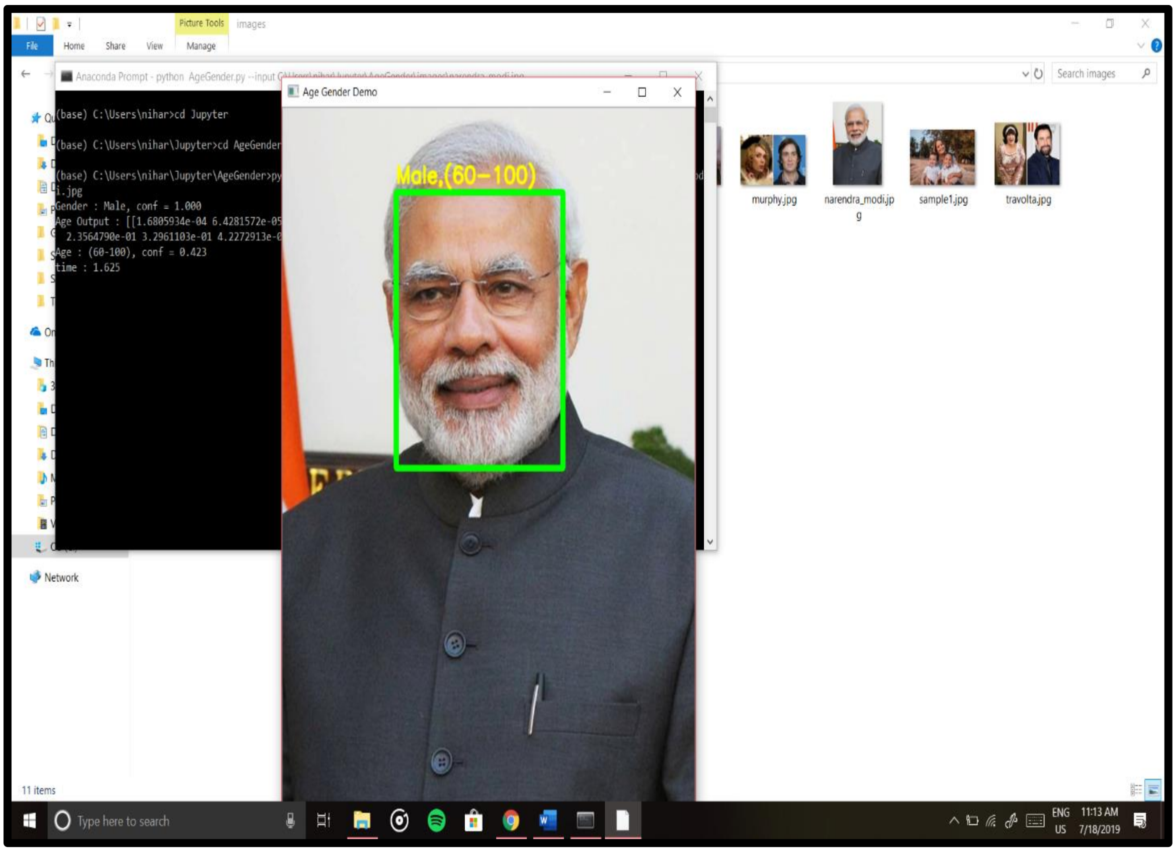


Fig 4.5: Input test image-3



```
(base) C:\Users\nihar>cd Jupyter

(base) C:\Users\nihar\Jupyter>cd AgeGender

(base) C:\Users\nihar\Jupyter\AgeGender>py
i.jpg
Gender : Male, conf = 1.000
Age Output : [[1.6805934e-04 6.4281572e-05
  2.3564790e-01 3.2961103e-01 4.2272913e-0
Age : (60-100), conf = 0.423
time : 1.625
```

Fig 4.6: Predicted Output

CHAPTER-5
ADVANTAGES AND DISADVANTAGES

❖ ADVANTAGES

- AI would have a low error rate compared to humans, if coded properly. They would have incredible precision, accuracy, and speed.
- Highly efficient.
- They won't be affected by hostile environments, thus able to complete dangerous tasks, explore in space, and endure problems that would injure or kill us.
- This can even mean mining and digging fuels that would otherwise be hostile for humans.
- Replace humans in repetitive, tedious tasks and in many laborious places of work.
- Predict what a user will type, ask, search, and do. They can easily act as assitants and can recommend or direct various actions. An example of this can be found in the smartphone.
- Organized and manages records. Can detect fraud in card-based systems, and possibly other systems in the future.
- Interact with humans for entertainment. Example is videogames.
- Can assess people both mentally and physically.
- This can be for medical purposes, such as health risks and emotional state. Can simulate medical procedures and give info on side effects. Robotic radiosurgery, and other types of surgery in the future, can achieve precision that humans can't.

❖ DISADVANTAGES

- Can cost a lot of money and time to build, rebuild, and repair. Robotic repair can occur to reduce time and humans needing to fix it, but that'll cost more money and resources.
- Storage is expansive, but access and retrieval may not lead to connections in memory as well as humans could.
- They can learn and get better with tasks if coded to, but it's questionable as to if this can ever become as good as humans can do such.
- They cannot work outside of what they were programmed for.
- They could never, or, at least, seemingly never with our technological perceptions, receive creativity that humans have.
- Difficulty in implementing "Common Sense".

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

Artificial intelligence is the key to future's development in the filed of technology as it supports the cause of automation. Many applications can be developed based on this AI and increase the efficiency of the entire operation. It may seem costlier to implement it at first but in the growing stages of the task, it tends to have its own imapct. For example, Robots, with them replacing jobs, can lead to severe unemployment, unless if humans can fix the unemployment with jobs AI can't do or severly change the government to communism. There may be loss in jobs in a particular sector but it may provide jobs in its own sector.Machines can easily lead to destruction, if put in the wrong hands. That is, at least a fear of many humans.

CHAPTER-7

REFERENCES

- www.tensorflow.org
- https://www.guru99.com/keras-tutorial.html
- https://www.surveygizmo.com/resources/blog/regression-analysis/
- https://www.researchgate.net/post/What_are_the_advantages_and_disadvantages_of_artificial_intelligence
- www.analyticsvidhya.com
- https://www.tutorialspoint.com/python/python_data_science
- https://www.edureka.co/blog/what-is-data-science/
- https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1