18CSC305J – ARTIFICIAL INTELLIGENCE
End Semester Notes

UNIT – 1

Types of Agents:

Agent	Description	Example
Simple Reflex Agent	Makes decisions based on the current input without considering the history or future consequences.	A thermostat that turns on or off based on a temperature threshold.
Model-Based Agent	Maintains an internal model or representation of the world and uses it to make decisions considering both current input and knowledge about the world.	A self-driving car that uses maps, sensor data, and its internal model to navigate and plan routes.
Goal-Based Agent	Pursues specific goals or objectives, selecting actions based on their expected outcome in achieving the goals.	A personal assistant that schedules appointments based on user preferences, availability, and efficiency constraints.
Utility-Based Agent	Maximizes a utility or reward function to select actions that yield the highest expected utility or reward.	A recommendation system that suggests movies based on user preferences, ratings, and predicted utility scores.
Learning Agent	Improves performance over time by learning from experience, updating its knowledge or model based on feedback.	A spam filter that learns from user-marked emails to improve its accuracy in identifying and filtering spam messages.
Hybrid Agent	Combines multiple agent types or approaches to benefit from their complementary strengths.	A robot vacuum cleaner that uses reactive reflexes for immediate obstacle avoidance and a model-based approach for planning efficient cleaning routes.

Crypto Arithmetic Problem:

Crypto arithmetic, also known as alphametic or verbal arithmetic, is a type of puzzle where letters or symbols represent digits, and the objective is to find the numerical value of each letter to solve the equation. In terms of AI, this is one of the constraint satisfaction problems (CSPs).

Some examples of the CSP are:
SEND+MORE=MONEY
TWO+TWO=FOUR
FORTY+TEN+TEN=SIXTY, etc.

(PRACTICE)

Problem Solving with AI:

Problem solving in AI refers to the process of using artificial intelligence techniques and algorithms to address complex problems or challenges. It involves designing and implementing computational methods that can analyse data, reason about it, and generate solutions or make decisions based on the problem at hand.

Stage	Description
Problem Definition	Clearly defining the problem is the first crucial step. It involves understanding the problem statement, identifying the desired outcome, and setting specific goals or objectives.
Problem Analysis and Representation	Once the problem is defined, it's essential to analyse and break it down into smaller components. This step may involve gathering data, conducting research, identifying constraints or dependencies, and representing the problem in a structured format.
Planning	Planning involves developing a strategy or roadmap to solve the problem. This includes determining the sequence of actions, allocating resources, setting priorities, and considering potential alternative approaches.
Execution	Execution refers to implementing the planned solution. It involves carrying out the actions outlined in the plan, utilizing appropriate tools, techniques, or technologies, and addressing any obstacles or challenges encountered during the process.
Evaluating Solution	After executing the solution, it's crucial to evaluate its effectiveness. This step involves assessing whether the solution achieves the desired outcome, measuring its performance against predefined criteria or metrics, and analysing any potential drawbacks.
Consolidating Gains	Once a solution is successfully implemented, it's important to consolidate the gains achieved. This step involves documenting lessons learned, capturing best practices, and integrating the solution into existing processes or systems to ensure long-term benefits.

AI Models:

Let us learn about semiotic models and statistical models in brief.

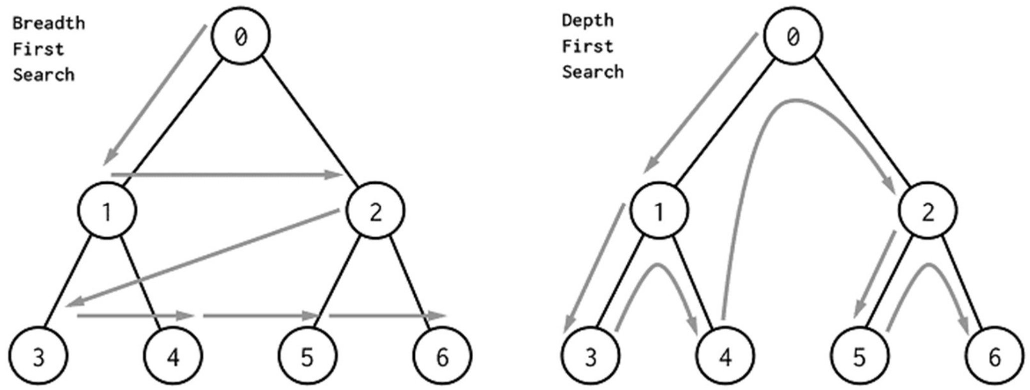
Semiotic Models	Statistical Models
Semiotic models focus on the interpretation and representation of meaning, symbols, and signs in a given context.	Statistical models aim to analyse and understand patterns, relationships, and trends within data using mathematical techniques.
Semiotic models are based on semiotics, which is the study of signs, symbols, and their interpretation in communication.	Statistical models are based on mathematical and probabilistic principles, utilizing statistical methods for analysis and modelling.
Semiotic models can incorporate a wide range of data types, including textual, visual & cultural data.	Statistical models rely on numerical data, as they are designed to perform quantitative analysis & inference.
Semiotic models focus on understanding the meaning and context of symbols, signs, and communication in a given domain.	Statistical models provide insights through numerical analysis, quantifying relationships and making predictions based on data.
Semiotic models are commonly used in areas such as linguistics, literature, cultural studies, and visual arts.	Statistical models find application across various domains, including finance, healthcare, marketing, and social sciences.
Roland Barthes' semiotic analysis, Peircean semiotics, symbolic interactionism, visual semiotics.	Linear regression, logistic regression, decision trees, neural networks, clustering algorithms, time series analysis.

UNIT – 2

For the examples of each algorithm, kindly refer to YouTube videos.

BFS vs DFS:

Breadth-First Search (BFS)	Depth-First Search (DFS)
BFS explores nodes in a breadth-wise manner, starting from the root or a given node, visiting all neighbours at the current depth before moving to the next depth level.	DFS explores nodes in a depth-wise manner, starting from the root or a given node, traversing as far as possible along each branch before backtracking.
BFS typically uses a queue data structure to maintain the order of nodes to be visited.	DFS typically uses a stack or recursion to keep track of nodes to be visited and backtracked.
BFS is implemented iteratively or using a queue data structure.	DFS can be implemented using recursion or a stack data structure.
BFS may require more memory as it needs to store all nodes at each level of the tree or graph.	DFS typically uses less memory as it only needs to store nodes along the current path being explored.
BFS is complete for finite graphs or trees if there is a goal state, as it guarantees finding the shallowest goal state first.	DFS is not guaranteed to find a goal state unless the search space is finite or the goal state is reachable from the start node.
Example: Finding the shortest path in a maze.	Example: Finding a path in a maze without considering its length.



A* Algorithm:

A* is an informed search algorithm that combines the features of both Dijkstra's algorithm (which guarantees finding the shortest path) and Greedy Best-First Search (which focuses on reaching the goal quickly). It uses heuristics to guide the search and make more informed decisions about which paths to explore. The algorithm maintains a priority queue of nodes to visit, prioritizing nodes with lower estimated costs to reach the goal.

Let's say we have a grid-based map representing a maze. Each cell in the grid can be either open or blocked. We want to find the shortest path from a starting cell to a target cell. The A* algorithm can help us achieve this.

Min-Max Algorithm:

The Min-Max algorithm is a decision-making algorithm commonly used in game theory and artificial intelligence to determine the optimal move for a player in a two-player, zero-sum game. It aims to minimize the maximum possible loss (hence the name "Min-Max") for a player assuming that the opponent plays optimally.

Step	Description
Define the game state	Represents the current state of the game, including positions of pieces, scores, or any other relevant information.
Generate the game tree	Creates a tree structure representing all possible moves and their outcomes. Each node represents a game state, and edges depict moves between states.
Evaluate terminal states	Assigns a score to each terminal state (game over) based on its favourability for the player. Positive score indicates a win, negative score indicates a loss, and zero represents a draw.
Perform recursive search	Recursively traverses the game tree, alternating between maximizing and minimizing scores. Maximizing player selects the move with the highest score, while the minimizing player chooses the move with the lowest score.
Backpropagation	Updates parent nodes' scores based on the scores of their children as the recursive search progresses back up the tree.
Choose the best move	Selects the move leading to the child node with the highest or lowest score once the search reaches the root node.

The Min-Max algorithm assumes perfect play from both players and explores the entire game tree up to a certain depth, which is typically limited to make the algorithm computationally feasible. The Min-Max algorithm provides a systematic way to search through the possibilities in a game and make optimal decisions based on the current state and predicted future outcomes.

Hill Climbing Algorithm:

The Hill Climbing algorithm is a local search algorithm used to find an optimal solution in an optimization problem. It is a simple iterative algorithm that starts with an initial solution and incrementally improves it by making small adjustments.

Step	Description
Initialize a solution	Start with an initial solution.
Evaluate the solution	Calculate the quality or fitness of the current solution using an objective function or cost measurement.
Generate solutions	Create a set of neighbouring solutions by making small modifications to the current solution.
Evaluate solutions	Calculate the quality or fitness of each neighbouring solution.
Select the best solution	Choose the neighbouring solution with the highest improvement or lowest cost compared to the current solution.
Update the current solution	If the best neighbouring solution is an improvement, update the current solution to be the best neighbouring solution.

Iterate the process by evaluating and generating neighbouring solutions until a stopping criterion is met.

UNIT – 3

For the examples of each algorithm, kindly refer to YouTube videos.

Forward Chaining vs Backward Chaining:

Forward Chaining	Backward Chaining
From known facts to new conclusions.	From the goal or desired outcome to supporting facts.
Deductive reasoning method that starts with known facts and uses rules to derive new conclusions until a goal is reached.	Deductive reasoning method that starts with the goal or desired outcome and works backward to find the supporting facts or conditions.
Top-down inference order.	Bottom-up inference order.
Based on availability and applicability of rules.	Based on the relevance and applicability of rules.
Data-driven.	Goal-driven.
Occurs at the end of the process.	Occurs at the beginning of the process.
More efficient for problems with a large number of facts.	More efficient for problems with a small number of facts.
Used in diagnostic systems, decision support systems and rule based expert systems.	Used in theorem proving, planning systems, program analysis and debugging.

Predicate Logic vs First Order Logic:

<table>
<tr><th>Predicate Logic</th><th>First-Order Logic</th></tr>
<tr><td colspan="2">Universal (∀) & Existential (∃) quantifiers are used.</td></tr>
<tr><td colspan="2">Variables represent objects or individuals.</td></tr>
<tr><td colspan="2">Constants represent specific objects or individuals.</td></tr>
<tr><td colspan="2">Predicates represent properties or relations between objects.</td></tr>
<tr><td colspan="2">Functions represent operations or transformations.</td></tr>
<tr><td>Less expressive compared to FOL.</td><td>More expressive compared to predicate logic.</td></tr>
<tr><td>Limited to individual sentences or formulas.</td><td>Variables can have scope over multiple sentences or formulas.</td></tr>
<tr><td>Simple and limited syntax.</td><td>More complex and rich syntax.</td></tr>
<tr><td>Examples:
P(x): "x is a prime number."
Q(x, y): "x is taller than y."</td><td>Examples:
P(x): "x is a prime number."
Q(x, y): "x is taller than y."
∀x: "For all x."
∃x: "There exists an x."</td></tr>
<tr><td>Predicate logic allows us to express statements about properties or relations between objects using predicates.</td><td>First-order logic expands on predicate logic by introducing quantifiers that allow us to make general statements about objects in a domain.</td></tr>
</table>

Unification:

Unification is a process in logic and artificial intelligence that aims to find a common substitution for variables in two or more logical expressions, making them syntactically equivalent. It involves finding values for variables such that the expressions can be unified or made equal. Unification is commonly used in automated theorem proving, logic programming, and natural language processing.

The unification process involves the following steps:

Variable Binding	When encountering variables in logical expressions, they can be bound to specific terms or values during the unification process.
Matching	The unification algorithm attempts to match terms or literals between the expressions being unified. It checks if the terms are identical or if one can be instantiated to make them identical.
Substitution	If a match is found, a substitution is performed, where variables are replaced with the corresponding terms. This substitution is applied consistently throughout the unification process.
Failure	If a match cannot be found or if there is a conflict during unification, the unification process fails, indicating that the expressions cannot be unified.

Consider the following expressions:

```
Expression 1: P(a, X, Y)
Expression 2: P(Z, b, c)
```

To unify these expressions, we try to find a substitution that makes them equal. In this case, the unification algorithm would find the following bindings:

```
X = b
Y = c
Z = a
```

After unification, the expressions become syntactically equivalent:

```
Unified Expression 1: P(a, b, c)
Unified Expression 2: P(a, b, c)
```

UNIT – 4

STRIPS (Block World Problem):

STRIPS (Stanford Research Institute Problem Solver) is a classic planning system for artificial intelligence developed at the Stanford Research Institute in the late 1960s and early 1970s. It was one of the earliest planning systems and had a significant influence on subsequent research in this area.

It represents the world as a set of predicates, each of which is either true or false. An action is represented by a set of preconditions, which must be true for the action to be applicable, and a set of effects, which describe the changes that occur in the world when the action is performed. The key idea behind STRIPS is to use a backward-chaining algorithm to search for a sequence of actions that will achieve a desired goal state.

```
Initial state: Lights Off
Goal state: Lights On

Actions:
//Flip the switch
Precondition: switch is off;
Effect: switch is on;
```

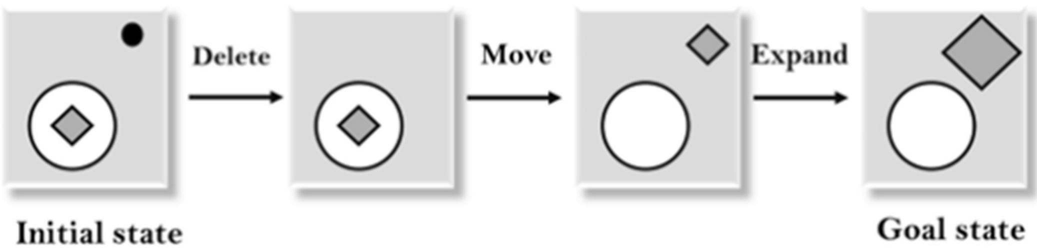
Mean End Analysis:

Mean-End Analysis (MEA) is a problem-solving strategy in artificial intelligence that involves identifying differences between the current state and the desired goal state and then developing subgoals that can help bridge the gap between the two states. It is often used in conjunction with other problem-solving techniques, such as STRIPS planning or heuristic search.

To use MEA, the problem solver first identifies the differences between the current state and the goal state. Next, the problem solver develops subgoals and bring the current state closer to the goal state.

MEA can be a powerful problem-solving technique, but it does have some limitations. For example, it can be difficult to identify all the differences between the current state and the goal state, and the subgoals developed may not always be the most efficient or effective way to reach the goal.

Nonetheless, MEA is still widely used in artificial intelligence and other problem-solving domains.



Types of Learning:

Learning Type	Description	Example
Supervised Learning	The model learns from labelled input-output pairs.	Image classification: Training a model to recognize objects based on labelled images of different classes.
Unsupervised Learning	The model learns from unlabelled data to discover patterns or structures.	Clustering: Grouping similar data points together without prior knowledge of their labels or classes.
Reinforcement Learning	The model learns through trial-and-error interactions with an environment, receiving rewards or penalties.	Game playing: Training an AI agent to play chess, where it receives rewards for winning and penalties for losing or making bad moves.
Semi-supervised Learning	The model learns from a combination of labelled and unlabelled data.	Text classification: Utilizing a large amount of unlabelled text data along with a small labelled dataset for training a sentiment analysis model.
Transfer Learning	The model leverages knowledge from one task to improve performance on another related task.	Pretrained models: Using a pre-trained image classification model as a starting point for a new image recognition task, fine-tuning it on a smaller labelled dataset.
Deep Learning	Utilizes neural networks with multiple layers to learn hierarchical representations of data.	Natural language processing: Training a deep learning model to understand and generate human-like text from huge data.

Support Vector Machine (SVM):

Support Vector Machine (SVM) is a popular supervised machine learning algorithm used for classification and regression tasks. It is particularly effective in cases where the data is separable into distinct classes or can be transformed into a higher-dimensional space where separation is possible. SVM finds an optimal hyperplane that maximally separates the classes by maximizing the margin between the data points of different classes.

SVM works by representing data points as vectors in a multi-dimensional space and finding the optimal hyperplane that separates the classes with the largest margin. The hyperplane is chosen in such a way that it maximizes the distance between the closest data points of different classes, known as support vectors.

Classification: Given a dataset of emails labelled as "spam" or "non-spam" based on their content, SVM can be trained to classify new, unlabelled emails as either spam or non-spam.

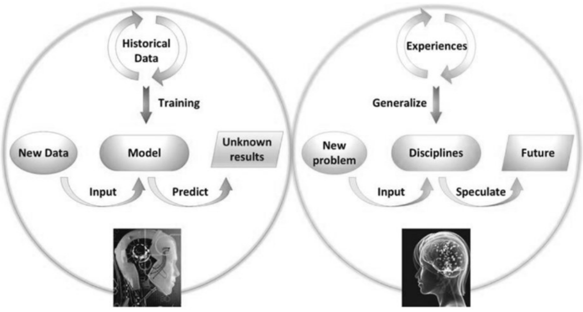
Regression: SVM can also be used for regression tasks, where the goal is to predict a continuous value. For example, given a dataset of housing prices with features such as area, number of bedrooms, and location, SVM can be trained to predict the price of a new house based on its features.

SVMs are known for their ability to handle high-dimensional data, work well with limited training samples, and have solid theoretical foundations. They can handle both linearly separable and non-linearly separable data by using kernel functions to transform the data into a higher-dimensional space.

Goal Stack Planning vs Non-Linear Planning:

Goal Stack Planning	Non-Linear Planning
Hierarchy of goals.	Directed Acyclic Graph (DAG) of actions and states.
Top-down decomposition.	Finding a sequence of actions.
Decomposes main goal into subgoals.	Focuses on finding actions to reach the goal state.
Backtracks and repairs plan if subgoal is not immediately achievable.	Utilizes forward and backward search to expand the graph.
Effective for decomposable problems.	Suitable for problems requiring finding a sequence of actions.
Used in automated reasoning, hierarchical task planning.	Used in graph-based planning, STRIPS planning.

Machine Learning vs Human Learning:



Machine Learning (ML)	Human Learning (HL)
Requires structured and labelled data.	Learns from unstructured data and experiences.
Can process vast amounts of data quickly.	Learning speed varies between individuals.
Can adapt to new data patterns and make predictions.	Adapts through reasoning, intuition, and cognitive flexibility.
Lacks common sense and contextual understanding.	Learns through context, background knowledge, and real-world cues.
Lacks creativity and cannot generate new ideas.	Creative thinking and innovation are fundamental to human learning.
Lacks emotional intelligence and empathy.	Emotional intelligence plays a crucial role in human learning.
Can generalize patterns based on training data.	Human learning can generalize and transfer knowledge across domains.
Requires specific training for each domain.	Humans can acquire expertise across multiple domains.
Raises ethical questions regarding data privacy and biases.	Ethical considerations in learning are shaped by societal values.

UNIT – 5

Expert System Architecture:

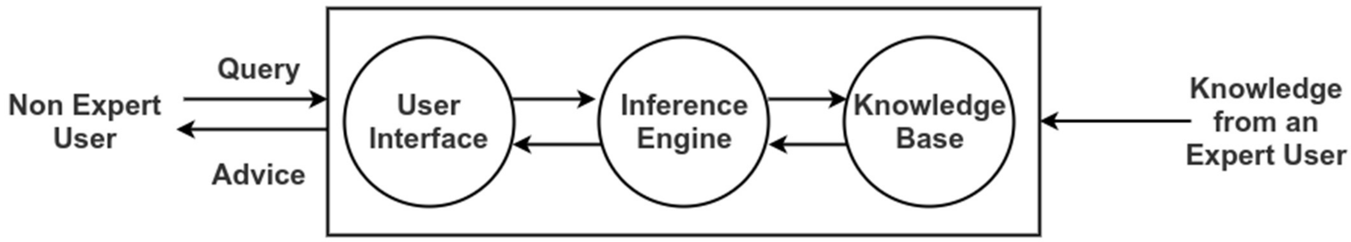
An expert system architecture typically consists of several components that work together to provide intelligent reasoning and decision-making capabilities. The following are some of the key components of an expert system architecture:

Knowledge Base: This is the component that stores all the domain-specific knowledge that the expert system uses to make decisions and provide recommendations. The knowledge base is usually organized in a structured manner, using a knowledge representation language such as frames or rules.

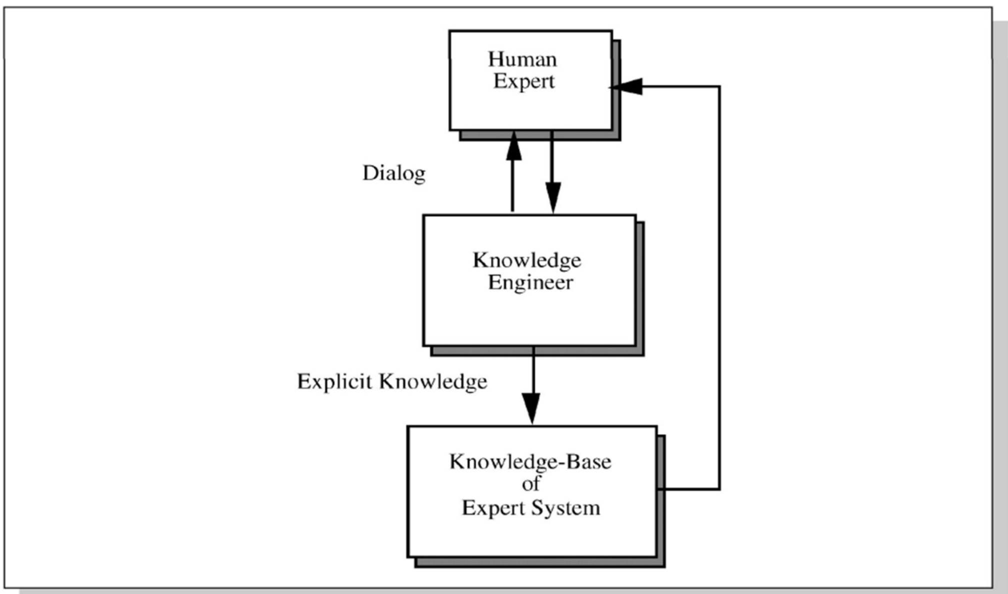
Inference Engine: This is the component that performs reasoning on the knowledge base to make decisions or provide recommendations. The inference engine uses various techniques such as forward chaining, backward chaining, and fuzzy logic to perform reasoning.

User Interface: This is the component that allows users to interact with the expert system. The user interface can be text-based or graphical, and it typically includes tools for entering inputs, displaying outputs, and navigating the knowledge base.

Explanation Facility: This is the component that allows the expert system to explain its reasoning and recommendations to the user. The explanation facility can generate natural language explanations, visualizations, or other forms of output that help the user understand how the expert system arrived at its conclusion.



Developing an Expert System



Rule-Based Expert Systems	Frame-Based Expert Systems
Represent knowledge as if-then rules.	Represent knowledge as frames with slots and values.
Use forward or backward chaining to apply rules.	Use slot-filling and inheritance to reason about data.
Can handle a wide variety of problem domains and knowledge structures.	More specialized and typically used for structured domains.
Requires explicit knowledge to be coded into rules by a knowledge engineer.	Can acquire knowledge from a domain expert through interviews or other means.
Efficient at processing large amounts of data.	May require more computational resources to process large numbers of frames.

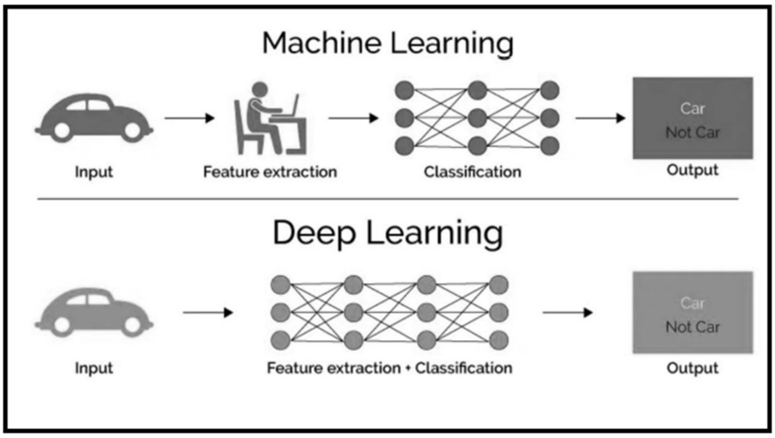
Business Intelligence and Business Analytics:

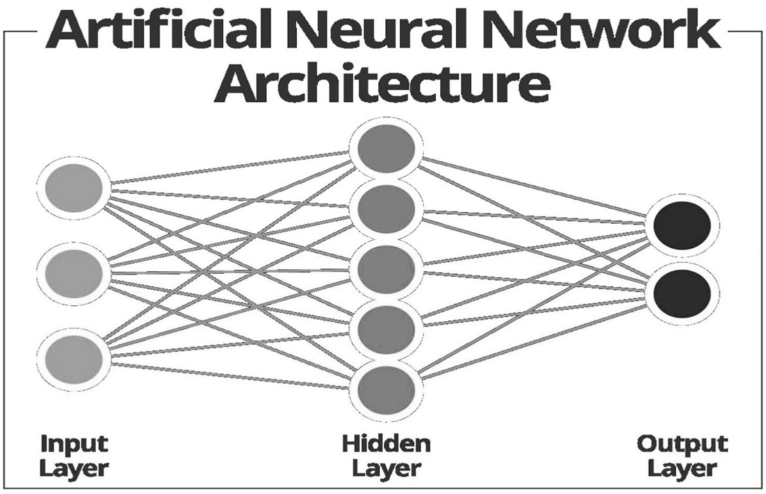
Business Intelligence (BI)	Business Analytics (BA)
Historical data analysis to understand past performance and identify trends.	Use of data analytics to make data-driven decisions for future actions.
Limited to the analysis of structured data from internal systems.	Broader scope including both structured and unstructured data from both internal and external sources.
Internal systems such as ERP, CRM, SCM, etc.	Internal and external sources including social media, customer feedback, market data, etc.
Focused on descriptive analytics, providing insights into what happened in the past.	Includes descriptive, predictive, and prescriptive analytics to gain insights and take actions based on data.
Traditional BI tools such as dashboards, reporting, and OLAP.	Advanced analytics tools such as data mining, machine learning, and predictive modelling.
Primarily used by managers and executives to monitor and analyse performance.	Used by managers, executives, and data scientists to make data-driven decisions across all levels of the organization.
Help organizations make better decisions by providing historical data analysis and insights.	Help organizations make data-driven decisions for future actions to improve business performance and achieve strategic goals.

Deep Learning:

Deep learning is a subfield of machine learning that uses neural networks with multiple layers to model and solve complex problems. Neural networks are a type of algorithm that are inspired by the structure and function of the human brain. In deep learning, these neural networks are composed of many layers of interconnected nodes, which can learn to represent complex features and patterns in the input data.

Deep learning algorithms are designed to automatically learn and extract features from raw data, without the need for manual feature engineering. This makes them well-suited for a wide range of tasks, including image recognition, speech recognition, natural language processing, and predictive analytics.



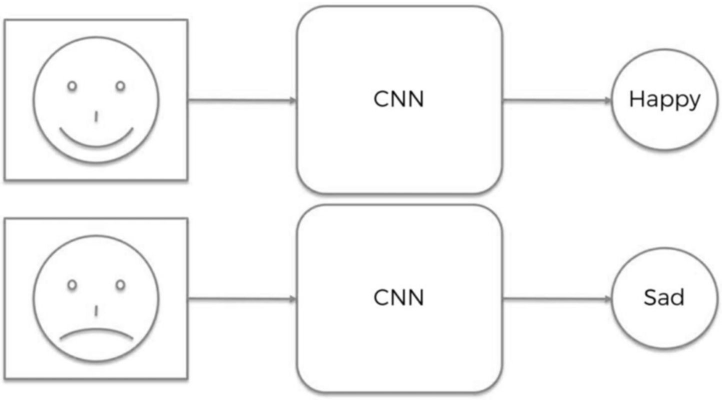


Convolutional Neural Network:

A convolutional neural network (CNN) is a type of deep neural network that is designed to process and analyse images and other multidimensional data. CNNs are widely used in computer vision applications, such as image recognition, object detection, and facial recognition.

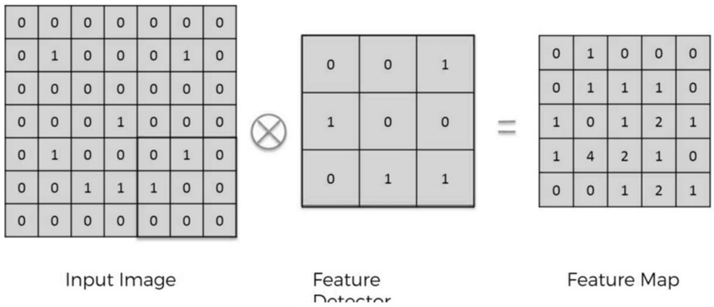
Some of the key features of a CNN include:

- Convolutional layers that perform local feature extraction using filters.
- Pooling layers that down sample the output and reduce the dimensionality of the data.
- Activation functions, such as ReLU, that introduce non-linearity into the model.
- Dropout and regularization techniques to prevent overfitting.
- Backpropagation algorithm to optimize the weights of the network during training.

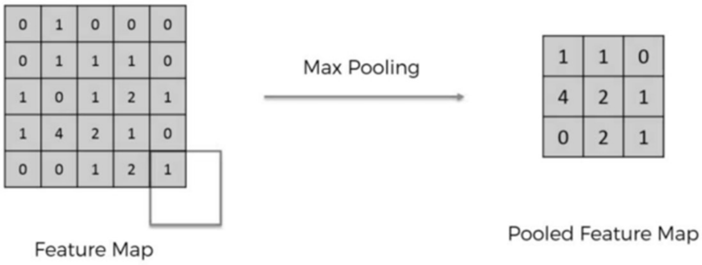


Here are the steps of creating a CNN model.

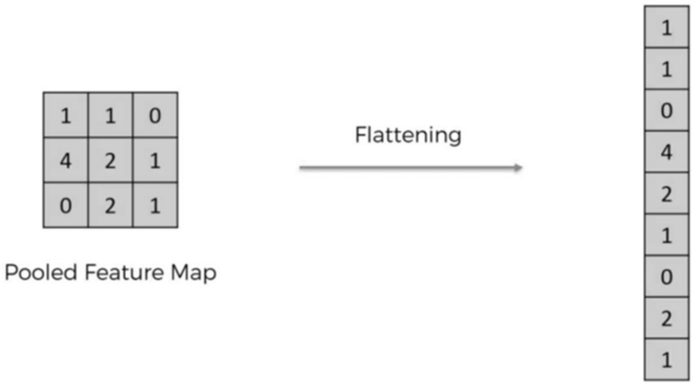
Convolution:



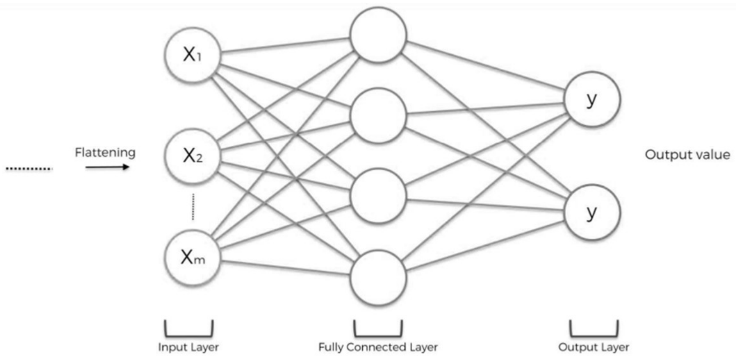
Max Pooling:



Flattening:

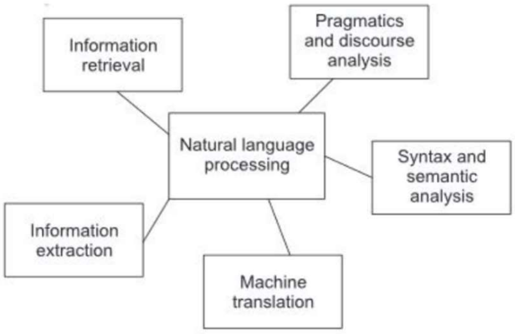


Full Connection:



Natural Language Processing(NLP):

It is a subfield of artificial intelligence and linguistics that focuses on the interaction between computers and human language. NLP involves the development of algorithms and models that enable computers to understand, interpret, and generate human language in a way that is meaningful and useful.



NLP Task	Description
Text classification	Categorizing and organizing text into predefined categories or topics.
Sentiment analysis	Determining the sentiment or emotion expressed in a piece of text, such as positive, negative, or neutral.
Named entity recognition	Identifying and classifying named entities in text, such as names of people, organizations, locations, or dates.
Machine translation	Automatically translating text from one language to another.
Speech recognition	Converting spoken language into written text.
Question answering	Providing accurate answers to questions posed in natural language.
Text summarization	Generating concise summaries of longer texts, such as articles or documents.
Chatbots and virtual assistants	Creating conversational agents that can interact with users in natural language.

To accomplish these tasks, NLP relies on various techniques and methods, including statistical models, machine learning, deep learning, and linguistic rules. Large language models like GPT-3 have significantly advanced the capabilities of NLP by capturing the nuances of human language and generating coherent and contextually relevant responses.

Advantages	Disadvantages
Automates language-related tasks and saves time.	Difficulty in handling ambiguous or context dependent language.
Enables efficient analysis of large volumes of text data.	Challenges with languages and dialects other than major ones.
Enhances customer support and user experience.	Reliance on availability of high-quality training data.
Improves information retrieval and search capabilities.	Limited understanding of sarcasm, irony, or figurative language.
Supports sentiment analysis for market research and feedback.	Privacy concerns when processing personal data.

Pattern Analytics & Knowledge Acquisition System:

Pattern Analytics & Knowledge Acquisition System (PAKAS) refers to a system or framework that combines pattern analytics and knowledge acquisition techniques to gain insights and extract valuable information from data. PAKAS is designed to analyse patterns, trends, and relationships within data to uncover hidden knowledge and make informed decisions.

The components of PAKAS are as follows:

Component	Description
Pattern Analytics	Utilizes statistical analysis, machine learning algorithms, and data mining techniques to identify patterns and relationships within datasets.
Knowledge Acquisition	Extracts explicit and tacit knowledge from domain experts, documents, databases, and other relevant sources.
Data Integration	Integrates and consolidates data from diverse sources to provide a comprehensive view of the information landscape.
Data Visualization	Utilizes visualization techniques, such as charts, graphs, and interactive dashboards, to present patterns and insights in a visually appealing manner.

The benefits of PAKAS are as follows:

Benefit	Description
Decision Making	Supports evidence-based decision making by providing valuable insights and patterns from data.
Knowledge Discovery	Helps uncover hidden patterns and trends in large volumes of data, leading to new discoveries and actionable knowledge.
Efficiency and Accuracy	Automates pattern identification and knowledge extraction, reducing manual effort and potential errors.
Predictive Analytics	Provides predictive analytics capabilities by leveraging historical data and patterns.
Domain Expertise Capture	Facilitates the capture and preservation of domain knowledge from experts for decision making and problem-solving.
Data-Driven Insights	Transforms raw data into meaningful insights, enabling a deeper understanding of operations, customers, and market dynamics.