B Tech CSE(AI-DS)

TY Semester V

Subject: AI Systems and Applications

Disclaimer:

1. Information included in this slides came from multiple sources. We have tried our best to cite the sources. Please refer to the References to learn about the sources, when applicable.

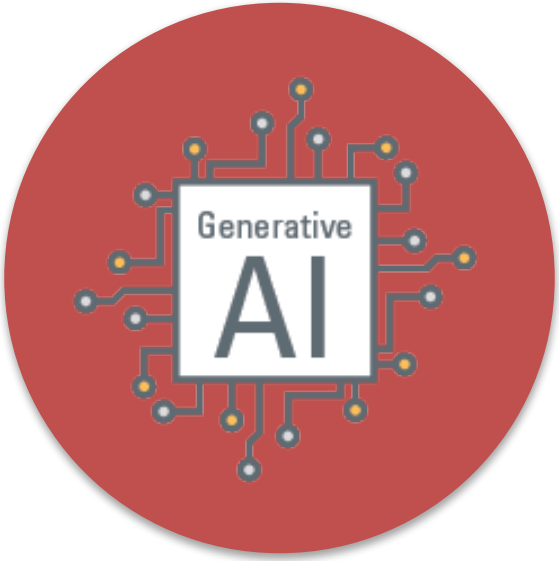
2. The slides should be used only for academic purposes (e.g., in teaching a class), and should not be used for commercial purposes.

Unit III: Advanced Technologies

- Generative AI,
- Transfer learning,
- Large Language Models (LLMs),
- Time series analysis,
- Graph Theory,
- Explainable AI (XAI),
- Edge AI,
- AI Ethics,
- Case Studies - Image Generation with GANs

What is Generative AI?

ChatGPT, Bing, Bard, DallE…



- Generative AI, like ChatGPT, uses machine learning to create new content. While generative AI tools can help explore new ideas, write text, and get feedback, there are important limitations to these tools to keep in mind.
- You can learn more about how generative AI works on LinkedIn Learning - u.mcmaster.ca/genai-linkedin

Generative AI

Generative AI is a branch of artificial intelligence that focuses on creating new content–such as text, images, audio, video, or even code–based on patterns learned from existing data.

Instead of just analyzing or classifying data (like traditional AI), generative models learn the distribution of data and can produce original outputs that resemble the training data.

Key Concepts in Generative AI

Concept	Description
Generative Model	An AI model that can produce new data instances similar to its training set.
Training Data	The dataset used to teach the model the structure and patterns of data.
Latent Space	A compressed representation of data that models use to generate new samples.
Prompting	Giving the model instructions or examples to guide its output.

Popular Generative AI Models

1. Language Models (Text Generation)

Examples: GPT (OpenAI), LLaMA (Meta), Claude (Anthropic), Gemini (Google)
Use cases: Chatbots, article writing, summarization, code generation

2. Image Generation Models

Examples: DALL·E (OpenAI), Midjourney, Stable Diffusion
Use cases: Art creation, product design, photo editing

3. Audio Generation Models

Examples: VALL-E, Suno AI, MusicLM
Use cases: Voice cloning, music generation, sound effects

4. Video Generation Models

Examples: Sora (OpenAI), Runway Gen-2, Pika Labs
Use cases: Movie clips, ads, animations

5. Other Generative Models

GANs (Generative Adversarial Networks) – Good for realistic images
VAEs (Variational Autoencoders) – Learn compressed data representation

Diffusion Models – Generate high-quality images via noise removal
Transformers – Used in large language models and multimodal AI

How Generative AI Works

Data Collection → Gather training data (text, images, audio, etc.)

Model Training → Learn patterns and relationships in the data

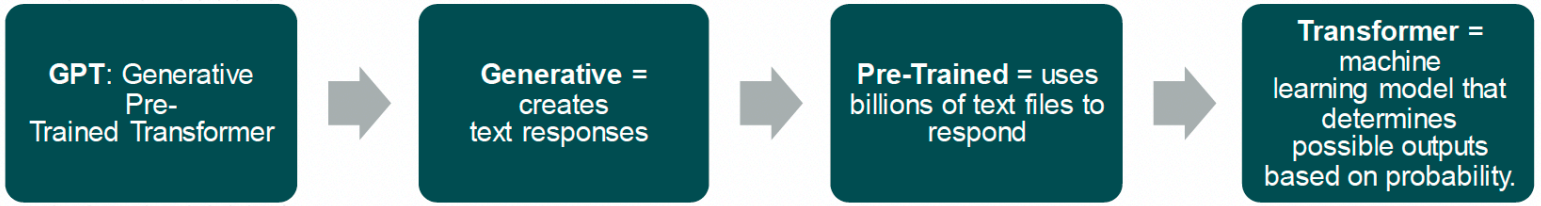
Latent Space Learning → Encode complex patterns into a compressed representation

Generation → Sample from the learned distribution to create new content

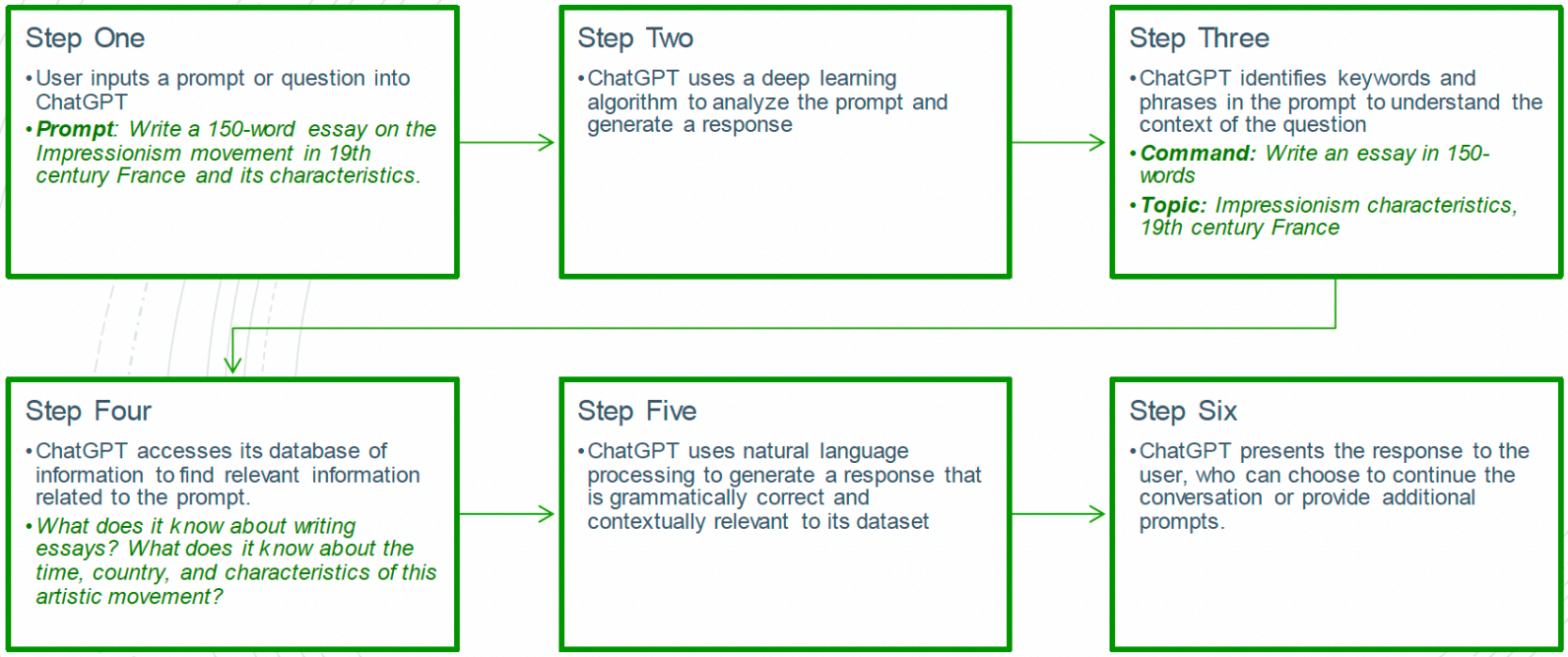
Fine-tuning / Prompting → Guide generation toward specific styles or tasks

HOW DOES GENERATIVE AI WORK?

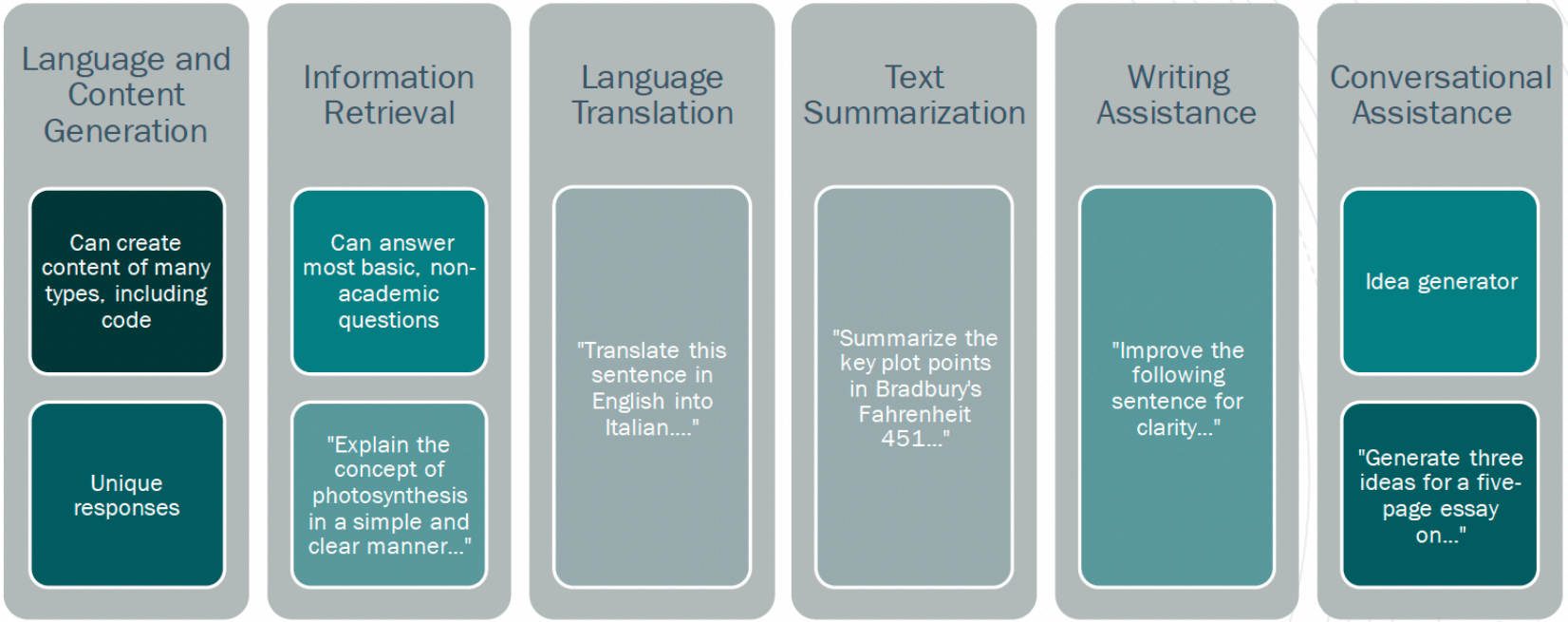
- **Generative AI (Gen AI):** Refers to deep-learning models that can generate high-quality text, images, and other content based on the data they are trained on. (IBM)
- **ChatGPT**: Large Language Model-based chatbot



HOW DOES GENERATIVE AI WORK?



GENERATIVE AI: FUNCTIONS



EXPLORING GEN AI IN THE CLASSROOM

Bias
- Discriminatory outputs

Environment
- Mining, energy consumption, and waste

Academic Integrity
- Plagiarism, cheating

Copyright
- Infringement on intellectual property rights

Privacy
- Personal data collection

Datafication
- Commodification of personal data

Human Labor
- Job automation and exploitation

Power
- Global power imbalances, structural inequalities

Applications of Generative AI

Domain	Examples
Text	ChatGPT, story writing, translation, summarization
Images	DALL·E art, fashion design, product mockups
Audio	AI singers, voice assistants, podcast editing
Video	Short films, explainer videos, synthetic actors
Code	GitHub Copilot, automated debugging
Science	Drug discovery, molecule design

Advantages & Challenges

Advantages

1. Creative content generation
2. Time and cost savings
3. Personalization at scale

Challenges

1. Bias in outputs
2. Misinformation / deepfakes
3. Ethical and copyright issues
4. High computational cost

Transfer learning

Transfer Learning is a machine learning technique where a model trained on one task is reused (transferred) for a different but related task.

Instead of training a model from scratch (which needs lots of data and time), we start with a pre-trained model and fine-tune it for our specific problem.

Key Idea

- Models learn generic features in early layers (e.g., edges, colors in images; syntax in text).
- These features are useful for many tasks.
- By reusing them, we save training time and often get better accuracy.

Steps in Transfer Learning (Deep Learning)

Choose a Pre-Trained Model

Image tasks → VGG, ResNet, Inception, MobileNet, EfficientNet
NLP tasks → BERT, GPT, RoBERTa

Freeze Base Layers

Keep existing weights for early layers (generic features).
This prevents overfitting and reduces computation.

Replace the Output Layer

Add a new layer for your custom classes/outputs.

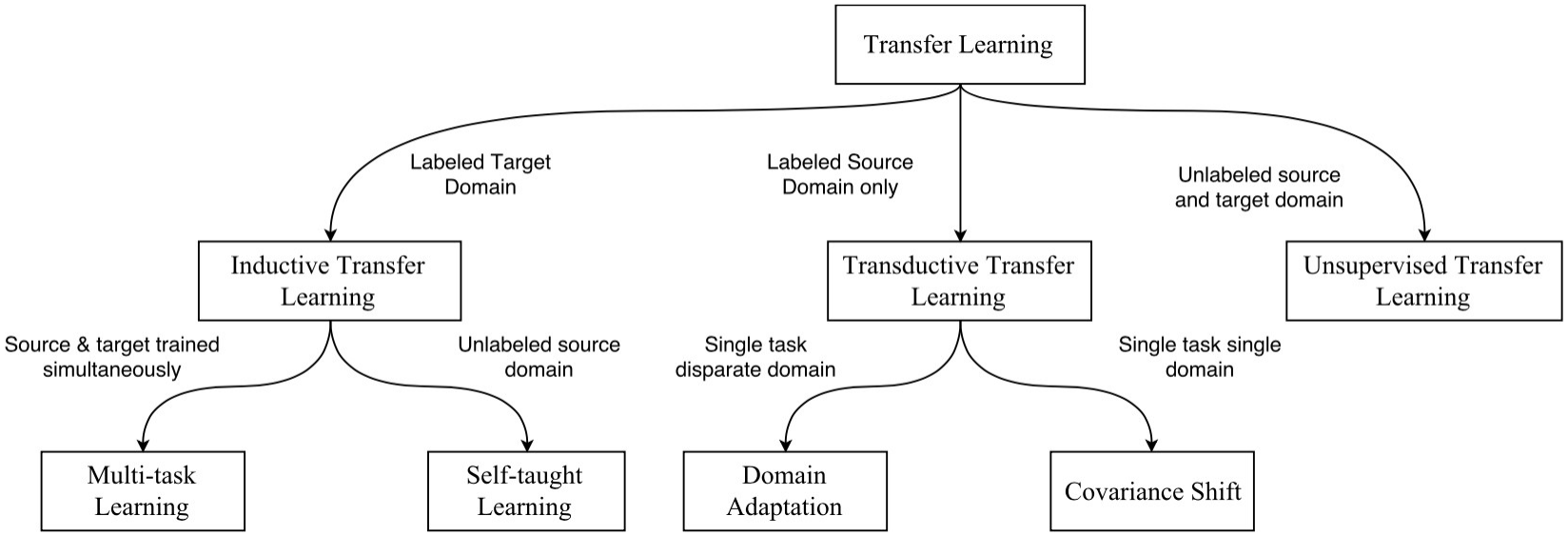
Fine-Tune

Optionally unfreeze some layers and train at a low learning rate.

Types of Transfer Learning

Type	Description
Feature Extraction	Use pre-trained model as a fixed feature extractor.
Fine-Tuning	Retrain some or all layers of the pre-trained model.
Domain Adaptation	Adjust a model trained on one domain to work on another domain.

Transfer Learning Approaches



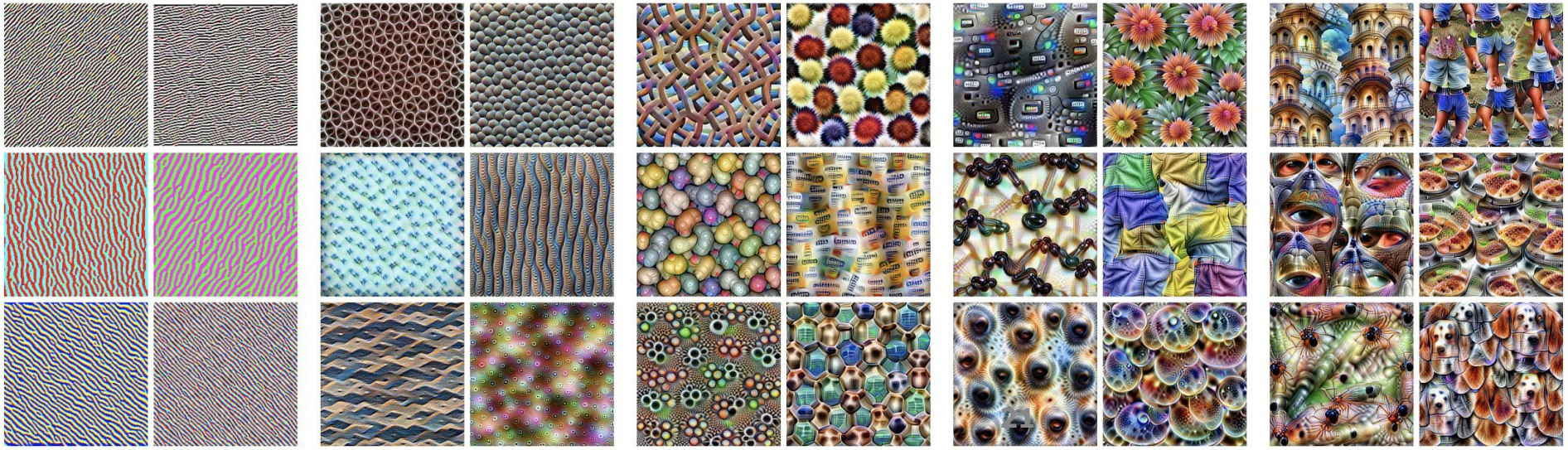
Source: https://arxiv.org/pdf/1802.05934.pdf

Transductive vs Inductive Transfer Learning

- **Transductive** transfer
  - No labeled target domain data available
  - Focus of most transfer research in NLP
  - E.g. Domain adaptation
- **Inductive** transfer
  - Labeled target domain data available
  - Goal: improve performance on the target task by training on other task(s)
    - Jointly training on >1 task (multi-task learning)
    - Pre-training (e.g. word embeddings)

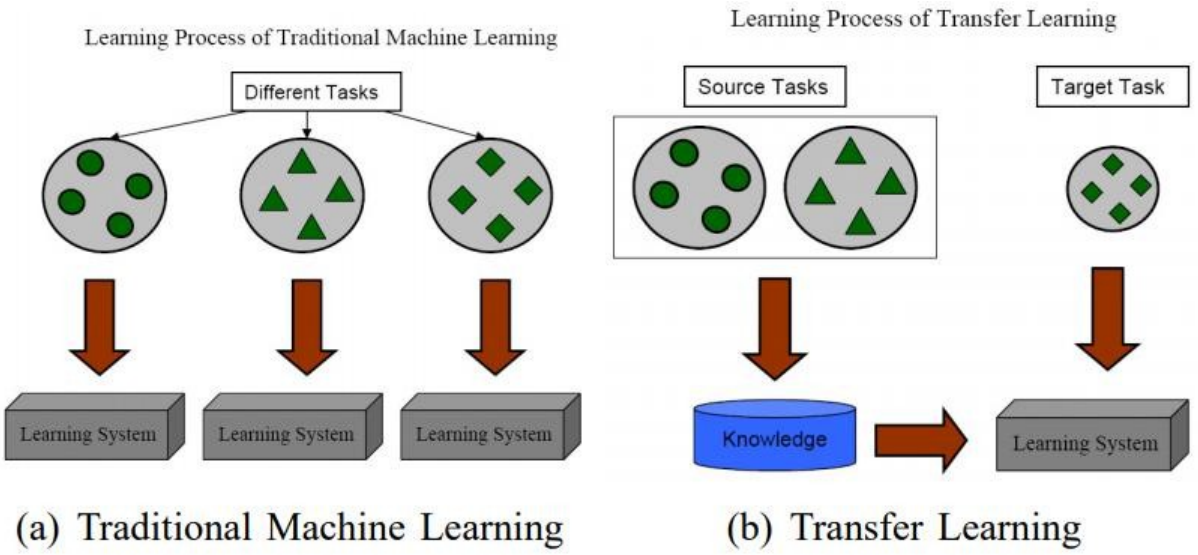
Recent paradigm shift in pre-training for NLP

- Inspired by the success of models pre-trained on ImageNet in Computer Vision (CV)
- The use of models pre-trained on ImageNet is now standard in CV



Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

Traditional vs. Transfer Learning



Transfer Learning Applications

- Image classification (most common): learn new image classes
- Text sentiment classification
- Text translation to new languages
- Speaker adaptation in speech recognition
- Question answering

Large Language Models (LLMs),

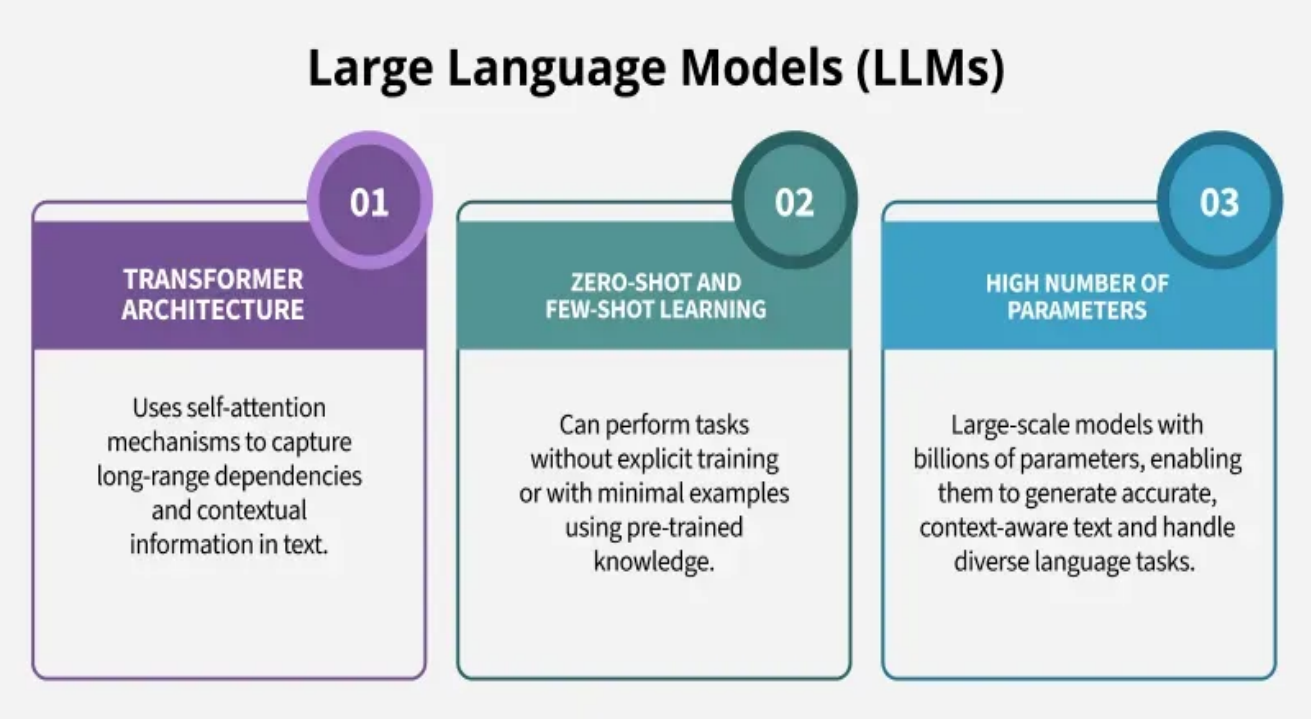
A large language model (LLM) is a type of artificial intelligence (AI) that is trained on a massive dataset of text and code. This allows the model to learn the statistical relationships between words and phrases, which in turn allows it to generate text, translate languages, write different kinds of creative content, and answer your questions in an informative way.

Here are common LLMs

- GPT3.5 and GPT4
- Bard
- LLama, Llama 2 - 65 B parameters
- BERT Large by Hugging Face - 340M parameters

Large Language Model (LLM)

Large Language Models (LLMs) are machine learning models trained on vast amount of textual data to generate and understand human-like language. These models can perform a wide range of natural language processing tasks from text generation to sentiment analysis and summarization.



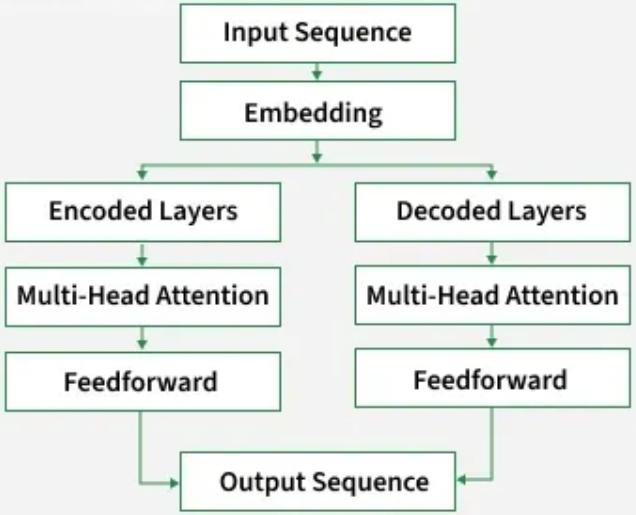
Transformers

Transformers are the foundational architecture behind most modern large language models that rely on attention mechanisms to process the entire sequence of the data simultaneously.

Transformer is a neural network architecture used for performing machine learning tasks particularly in natural language processing (NLP) and computer vision. In 2017 Vaswani et al. published a paper " Attention is All You Need" in which the transformers architecture was introduced. The article explores the architecture, workings and applications of transformers.

What are Transformers?

- Neural network architecture for NLP & vision
- Introduced in 2017: "Attention is All You Need"
- Processes whole sentences in parallel



Need For Transformers Model in Machine Learning

Transformer Architecture uses self-attention to transform one whole sentence into a single sentence. This is useful where older models work step by step and it helps overcome the challenges seen in models like RNNs and LSTMs. Traditional models like RNNs (Recurrent Neural Networks) suffer from the vanishing gradient problem which leads to long-term memory loss. RNNs process text sequentially meaning they analyze words one at a time.

For example:

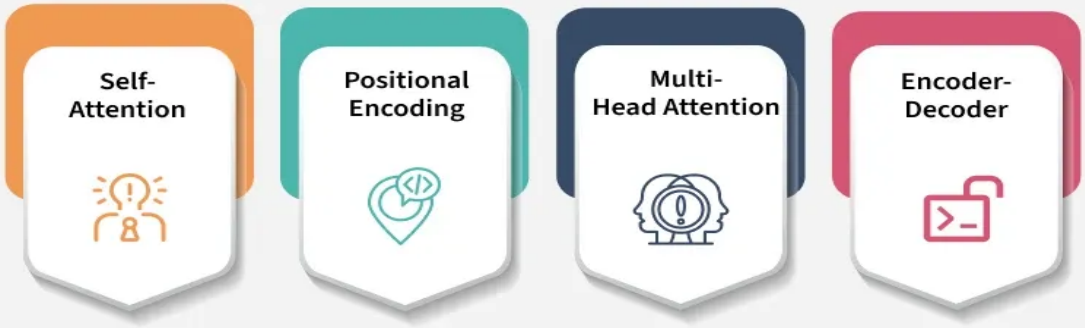
In the sentence: "XYZ went to France in 2019 when there were no cases of COVID and there he met the president of that country" the word "that country" refers to "France".
However RNN would struggle to link "that country" to "France" since it processes each word in sequence leading to losing context over long sentences. This limitation prevents RNNs from understanding the full meaning of the sentence.

While adding more memory cells in LSTMs (Long Short-Term Memory networks) helped address the vanishing gradient issue they still process words one by one. This sequential processing means LSTMs can't analyze an entire sentence at once.

For example:

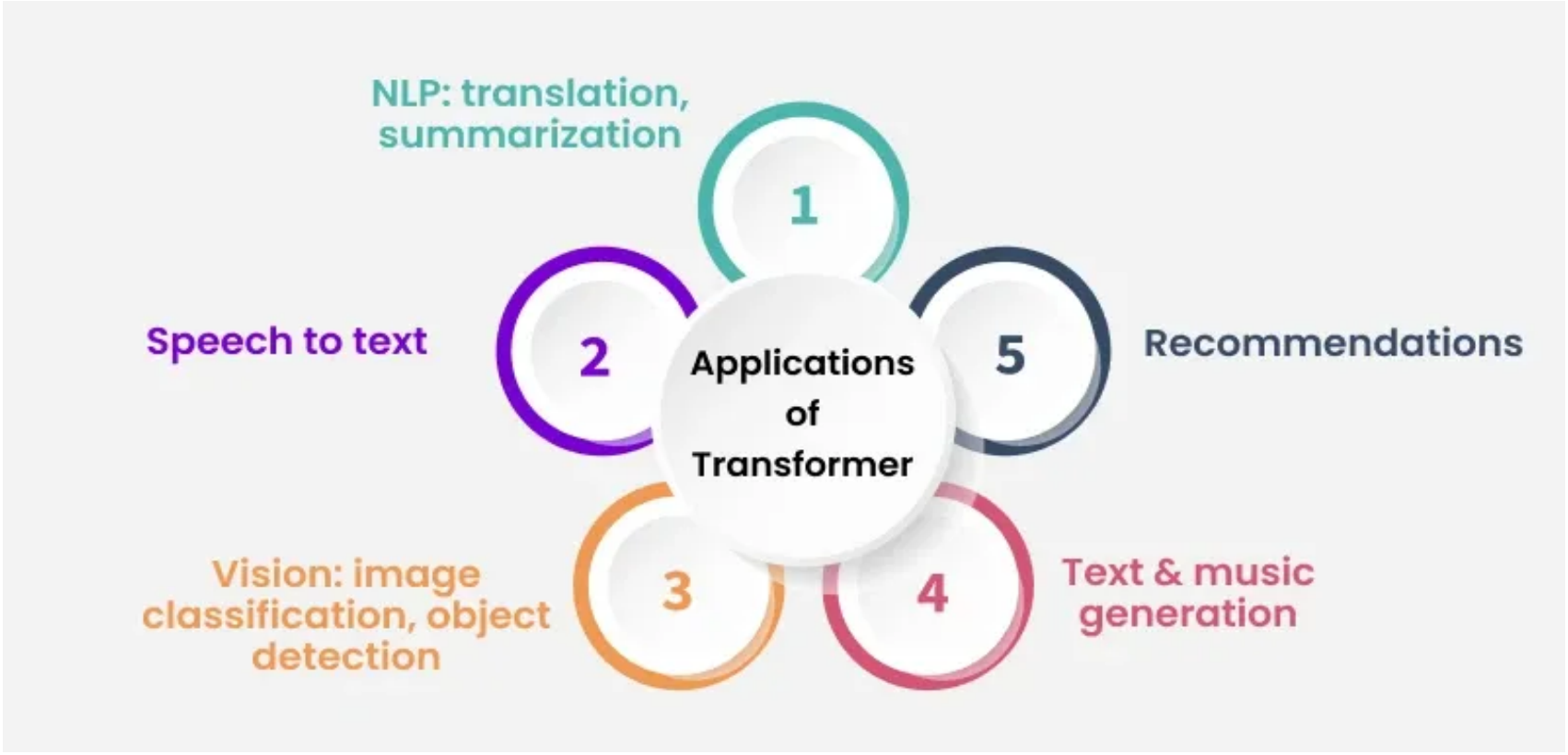
The word "point" has different meanings in these two sentences:
"The needle has a sharp point." (Point = Tip)
"It is not polite to point at people." (Point = Gesture)

Core Concepts Of Transformers



Why Transformers ?

- RNNs lose context in long sentences
- Transformers use self-attention for full context
- Example: "point" means different things in different sentences


NLP: translation, summarization
1
Speech to text
2
Applications of Transformer
5
Recommendations
Vision: image classification, object detection
3
4
Text & music generation

Core Concepts of Transformers

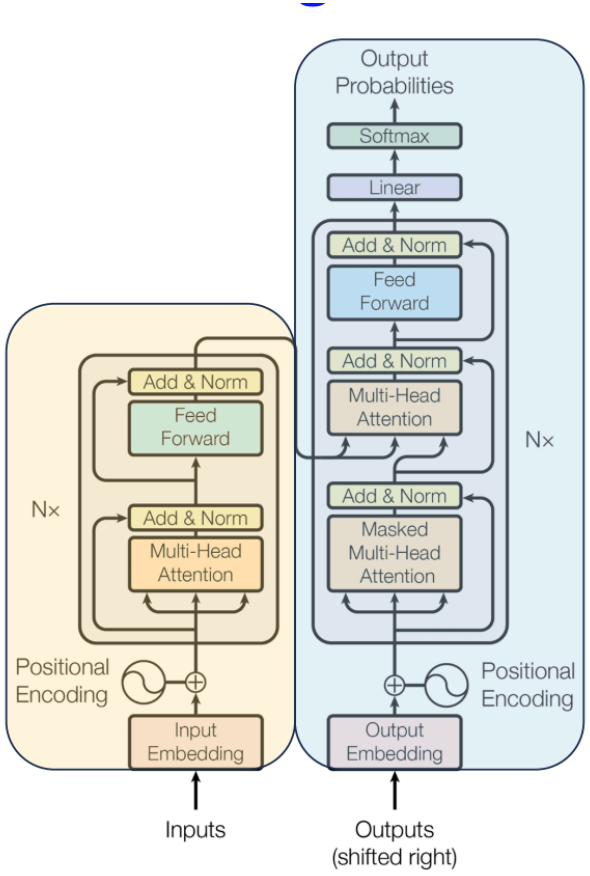
Traditional models struggle with this context dependence, whereas Transformer model through its self-attention mechanism processes the entire sentence in parallel addressing these issues and making it significantly more effective at understanding context.

Review: The LLM Era – Paradigm Shift in Machine Learning

Language models can be used to not just perform a single task, but multiple tasks by learning to predict the next token or sentence

BERT
Oct 2018

Representation



GPT
Jun 2018

Generation

1. Self Attention Mechanism

The self attention mechanism allows transformers to determine which words in a sentence are most relevant to each other. This is done using a scaled dot-product attention approach:

Each word in a sequence is mapped to three vectors:

Query (Q)
Key (K)
Value (V)

Attention scores are computed

as: Attention(Q,K,V)= $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

These scores determin[illegible]ion each word should pay to others.

2. Positional Encoding

Unlike RNNs, transformers lack an inherent understanding of word order since they process data in parallel. To solve this problem Positional Encodings are added to token embeddings providing information about the position of each token within a sequence.

3. Multi-Head Attention

Instead of one attention mechanism, transformers use multiple attention heads running in parallel. Each head captures different relationships or patterns in the data, enriching the model's understanding.

4. Position-wise Feed-Forward Networks

The Feed-Forward Networks consist of two linear transformations with a ReLU activation. It is applied independently to each position in the sequence.

Mathematically: $\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$

This transformation helps refine the encoded representation at each position.

5. Encoder-Decoder Architecture

The encoder-decoder structure is key to transformer models. The encoder processes the input sequence into a vector, while the decoder converts this vector back into a sequence. Each encoder and decoder layer includes self-attention and feed-forward layers. In the decoder, an encoder-decoder attention layer is added to focus on relevant parts of the input.

For example, a French sentence "Je suis étudiant" is translated into "I am a student" in English.

The encoder consists of multiple layers (typically 6 layers). Each layer has two main components:
Self-Attention Mechanism: Helps the model understand word relationships.
Feed-Forward Neural Network: Further transforms the representation.
The decoder also consists of 6 layers but with an additional encoder-decoder attention mechanism. This allows the decoder to focus on relevant parts of the input sentence while generating output.

Intuition with Example

For instance in the sentence "The cat didn't chase the mouse, because it was not hungry" the word 'it' refers to 'cat'. The self-attention mechanism helps the model correctly associate 'it' with 'cat' ensuring an accurate understanding of sentence structure.

Applications of Transformers

Some of the applications of transformers are:

NLP Tasks: Transformers are used for machine translation, text summarization, named entity recognition and sentiment analysis.
Speech Recognition: They process audio signals to convert speech into transcribed text.
Computer Vision: Transformers are applied to image classification, object detection and image generation.
Recommendation Systems: They provide personalized recommendations based on user preferences.
Text and Music Generation: Transformers are used for generating text like articles and composing music.

Transformers have redefined deep learning across NLP, computer vision and beyond. With advancements like BERT, GPT and Vision Transformers (ViTs) they continue to push the boundaries of AI and language understanding and multimodal learning.

Applications of LLMs

- Chatbots (ChatGPT, Claude, Gemini)
- Text Summarization
- Language Translation
- Code Generation (GitHub Copilot)
- Question Answering
- Content Creation
- Data Analysis & Insights

Time Series Analysis (TSA)

Time Series Analysis (TSA) is a method of analyzing data points collected over time to extract meaningful patterns, trends, and insights. It's widely used for forecasting, anomaly detection, and understanding temporal relationships.

Key Characteristics of Time Series

Time Order Matters → The sequence of data points is crucial.

Autocorrelation → Past values can influence future values.

Seasonality → Repeating patterns over a specific period (daily, monthly, yearly).

Trend → Long-term upward or downward movement.

Noise → Random variation in the data.

Components of a Time Series

Trend (T) → Long-term progression (upward, downward, or flat)

Seasonal (S) → Repeating patterns within fixed time frames

Cyclic (C) → Fluctuations without a fixed period (economic cycles)

Irregular/Noise (I) → Random variations not explained by the model

Types of Time Series Data

Univariate → One variable recorded over time (e.g., daily temperature)

Multivariate → Multiple variables recorded over time (e.g., temperature, humidity, wind speed)

Common Methods in Time Series Analysis

Method	Description
Moving Average (MA)	Smooths short-term fluctuations to highlight trends
Exponential Smoothing	Weighted averages where recent data gets more weight
ARIMA (AutoRegressive Integrated Moving Average)	Combines autoregression, differencing, and moving averages
SARIMA	ARIMA + seasonal components
Holt-Winters	Triple exponential smoothing (trend + seasonality)
State Space Models / Kalman Filter	Dynamic modeling of systems over time
Prophet (by Meta)	Easy-to-use forecasting with trend and seasonality

Graph Theory

Graph Theory is a branch of mathematics and computer science that studies graphs, which are structures made up of vertices (nodes) connected by edges (links). Graphs are powerful tools for modeling relationships and interactions in real-world systems, from computer networks to social interactions.

Key Concepts in Graph Theory

Graph

A graph G=(V,E)G = (V, E)G=(V,E) consists of:

VVV: set of vertices (nodes).

EEE: set of edges (connections between nodes).

Types of Graphs

1. Undirected Graph: edges have no direction (e.g., friendship network).
2. Directed Graph (Digraph): edges have a direction (e.g., Twitter followers).
3. Weighted Graph: edges carry weights (e.g., distance in road maps).
4. Unweighted Graph: edges only represent connections, no weights.
5. Simple Graph: no loops or multiple edges.
6. Multigraph: multiple edges can connect the same pair of nodes.

Special Graphs

Tree: a connected graph with no cycles.
Bipartite Graph: vertices can be divided into two sets, with edges only across sets.
Complete Graph: every pair of vertices is connected by an edge.
Planar Graph: can be drawn on a plane without edges crossing.

Basic Terminology

Degree of a vertex: number of edges connected to it.
Path: sequence of vertices connected by edges.
Cycle: path that starts and ends at the same vertex.
Connected graph: all vertices are reachable from any other.
Component: a maximally connected subgraph.

What is a graph G?

It is a pair G = (V, E), where

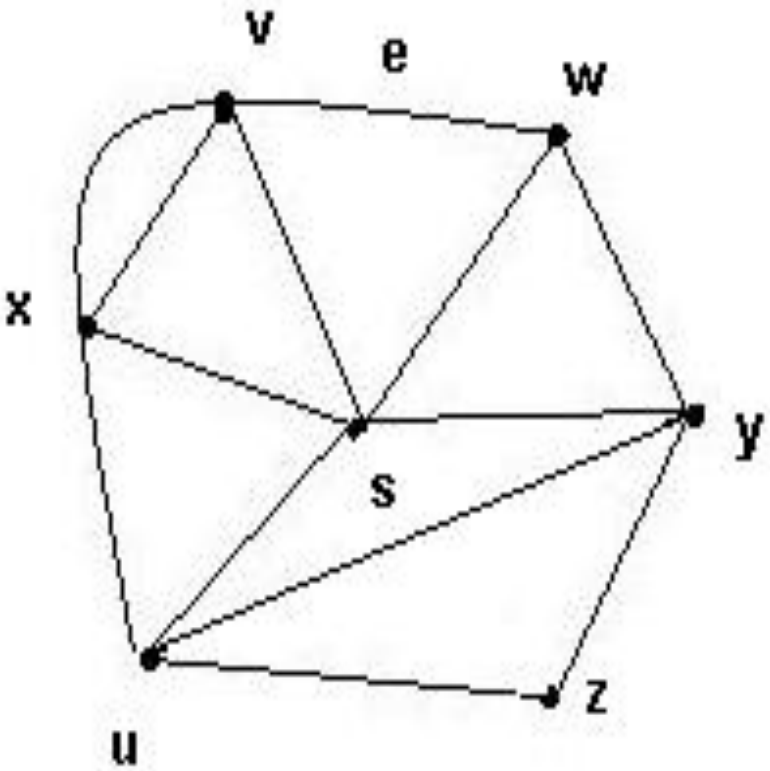
V = V(G) = set of vertices

E = E(G) = set of edges

Example:

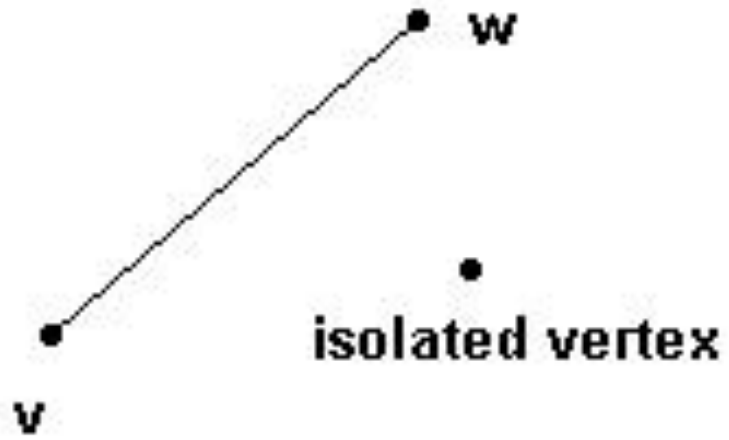
V = {s, u, v, w, x, y, z}

E = {(x,s), $(x,v)_1$, $(x,v)_2$, (x,u), (v,w), (s,v), (s,u), (s,w), (s,y), (w,y), (u,y), (u,z),(y,z)}



Edges

An edge may be labeled by a pair of vertices, for instance e = (v,w).
e is said to be *incident* on v and w.
Isolated vertex = a vertex without incident edges.



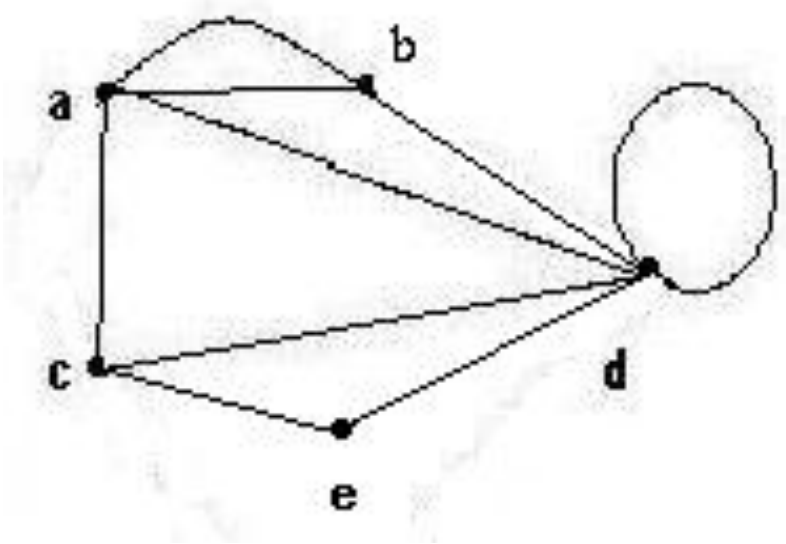
Special edges

Parallel edges

- Two or more edges joining a pair of vertices
  - in the example, **a** and **b** are joined by two parallel edges

Loops

- An edge that starts and ends at the same vertex
  - In the example, vertex **d** has a loop



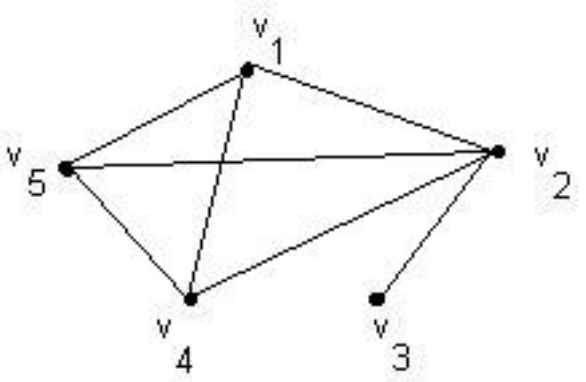
Special graphs

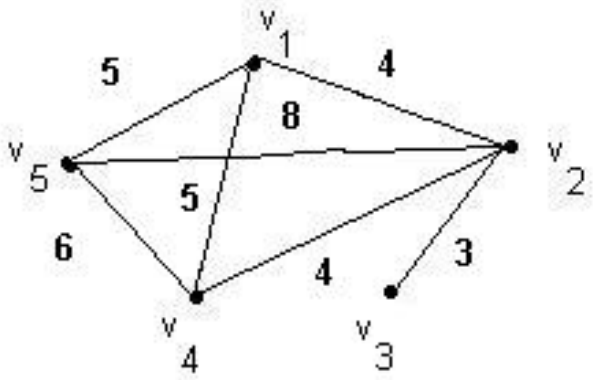
Simple graph

A graph without loops or parallel edges.

Weighted graph

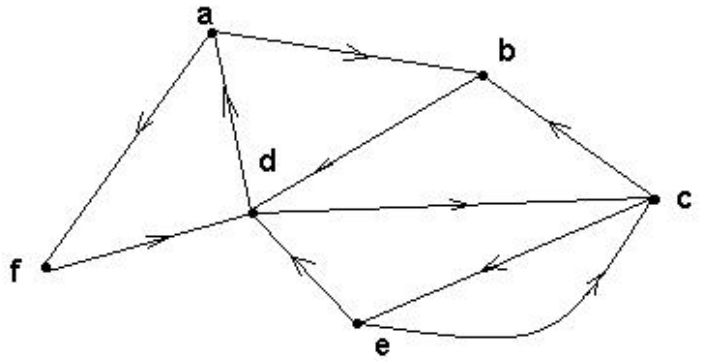
A graph where each edge is assigned a numerical label or “weight”.





Directed graphs (digraphs)

G is a *directed graph* or *digraph* if each edge has been associated with an ordered pair of vertices, i.e. each edge has a direction



Similarity graphs (1)

Problem: grouping objects into similarity classes based on various properties of the objects.

Example:

Computer programs that implement the same algorithm have properties k = 1, 2 or 3 such as:

1. Number of lines in the program
2. Number of “return” statements
3. Number of function calls

Similarity graphs (3)

A graph G is constructed as follows:

V(G) is the set of programs {v_1, v_2, v_3, v_4, v_5 }.

Each vertex v_i is assigned a triple (p_1, p_2, p_3),

where p_k is the value of property k = 1, 2, or 3

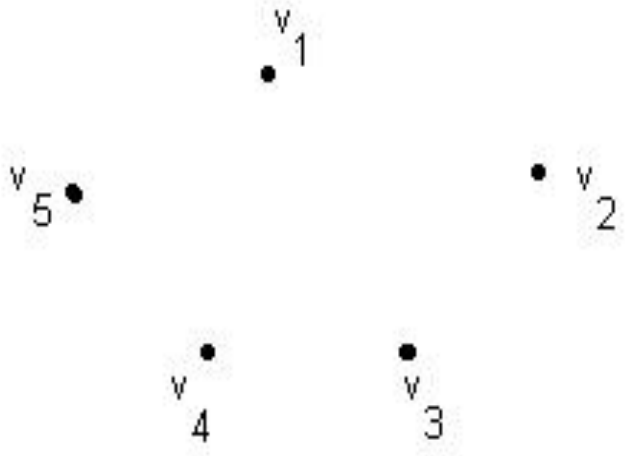
v_1 = (66,20,1)

v_2 = (41, 10, 2)

v_3 = (68, 5, 8)

v_4 = (90, 34, 5)

v_5 = (75, 12, 14)



Bipartite graphs

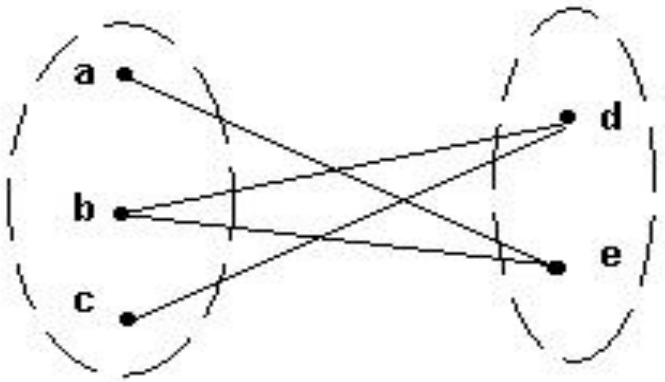
A bipartite graph G is a graph such that

$V(G) = V(G_1) \cup V(G_2)$

$|V(G_1)| = m$, $|V(G_2)| = n$

$V(G_1) \cap V(G_2) = \varnothing$

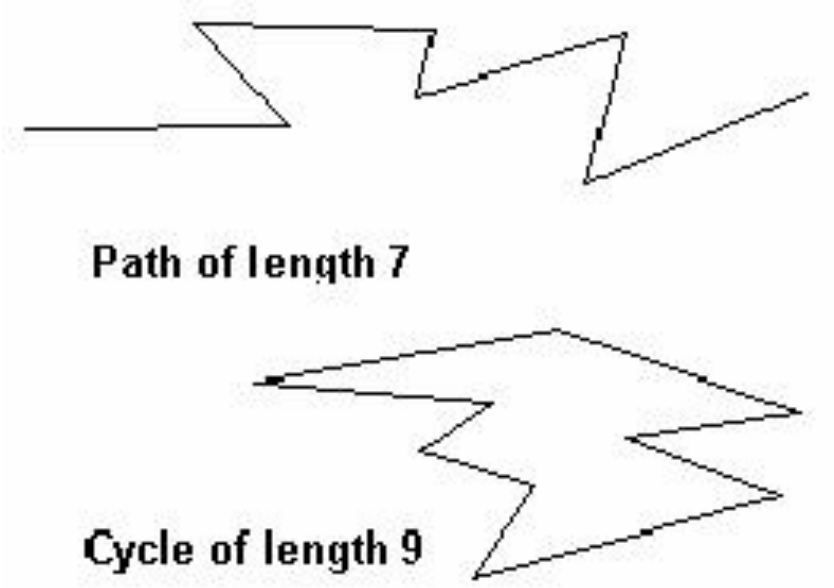
No edges exist between any two vertices in the same subset $V(G_k)$, k = 1,_



Connected graphs

A graph is *connected* if every pair of vertices can be connected by a path
Each connected subgraph of a non-connected graph G is called a *component* of G

6.2 Paths and cycles



A *path of length n* is a sequence of n + 1 vertices and n consecutive edges

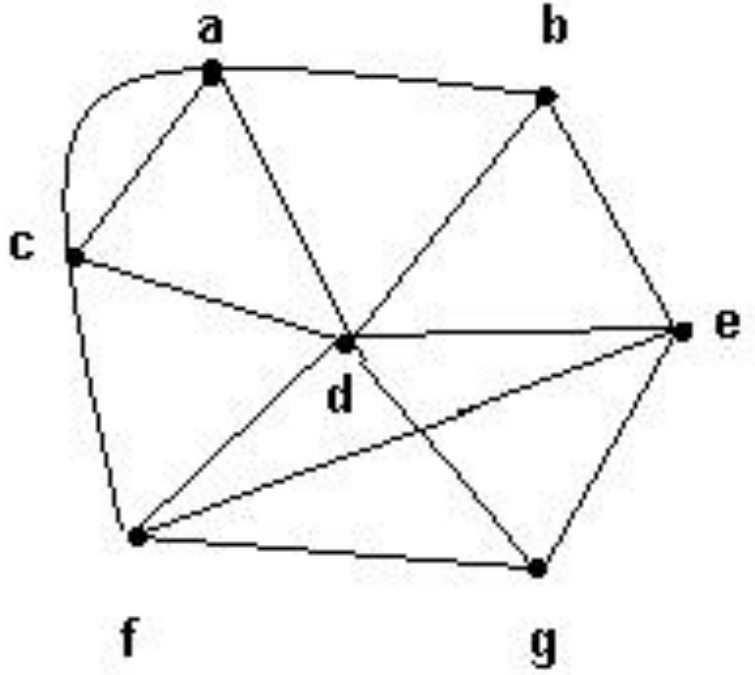
A *cycle* is a path that begins and ends at the same vertex

Degree of a vertex

The *degree* of a vertex v, denoted by $\delta(v)$, is the number of edges incident on v

Example:

$\delta(a) = 4$, $\delta(b) = 3$,
$\delta(c) = 4$, $\delta(d) = 6$,
$\delta(e) = 4$, $\delta(f) = 4$,
$\delta(g) = 3$.



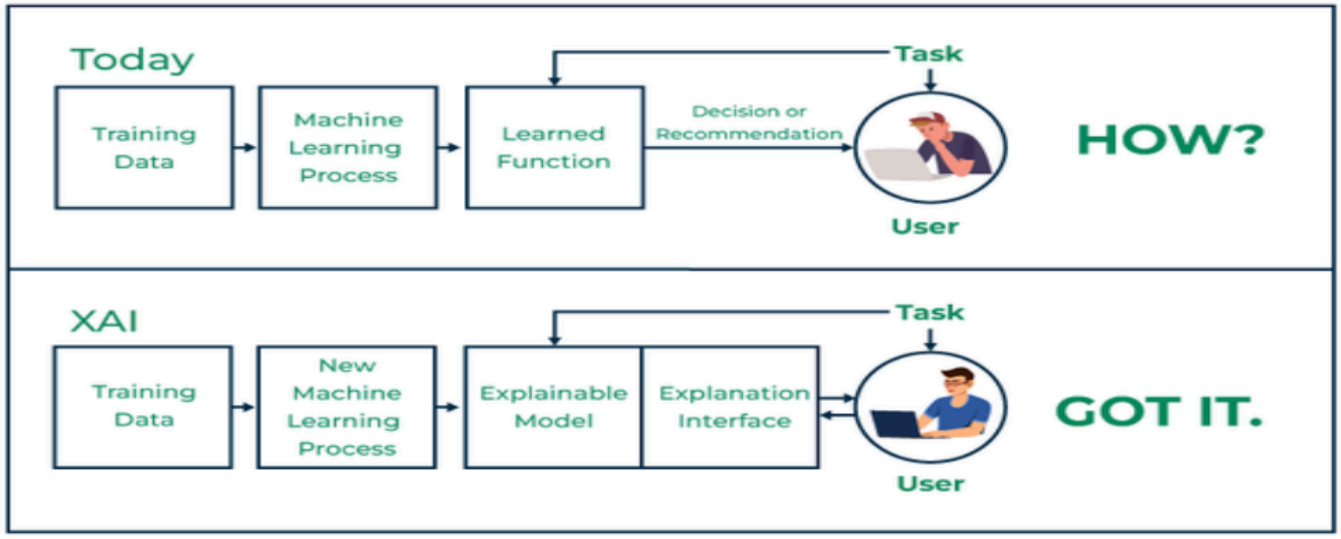
What is Explainable AI?

Explainable AI

Explainable AI (XAI) refers to artificial intelligence systems that can provide clear and understandable explanations for their decisions and outputs, making them transparent to humans. This field aims to develop techniques that allow humans to comprehend, trust, and manage AI systems by understanding the underlying mechanisms and reasoning behind AI-generated results.

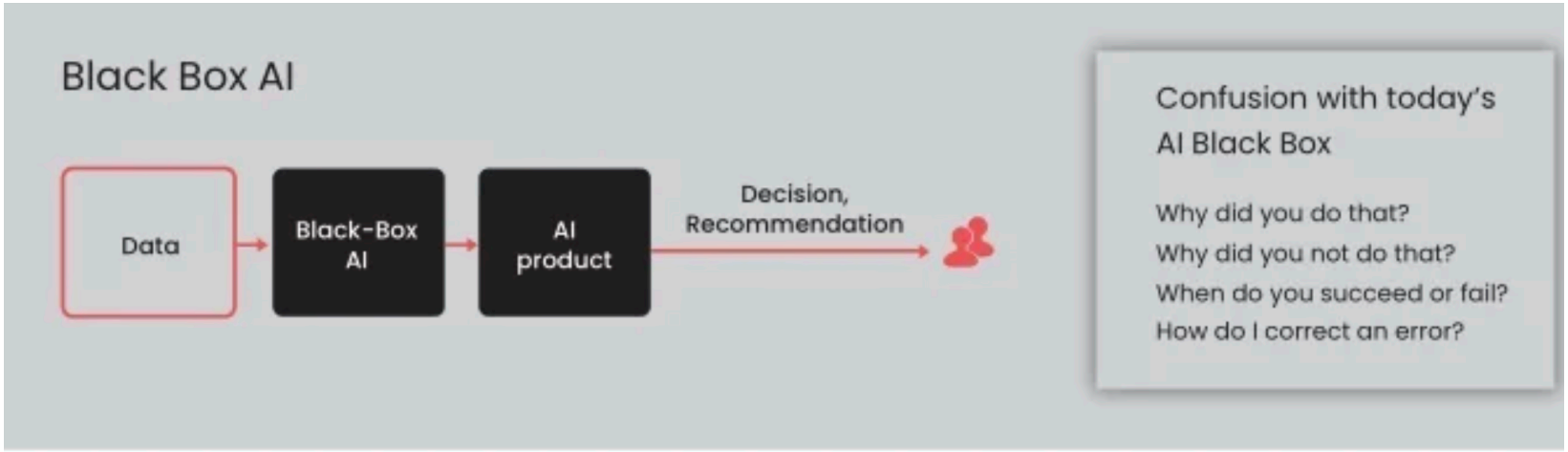
Explainable AI

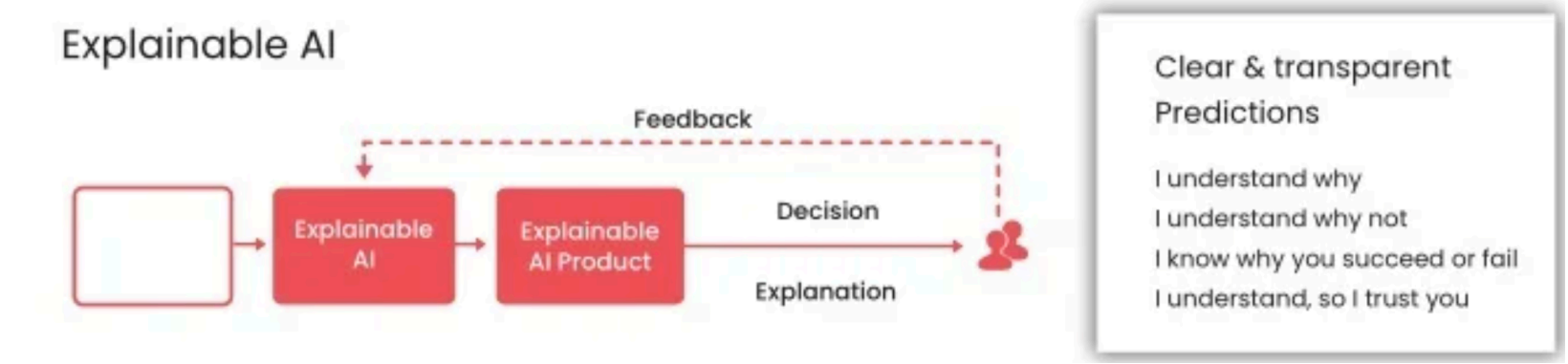
Explainable AI (XAI) is a field of artificial intelligence that focuses on making the decisions and outputs of AI systems understandable to humans. The goal is to move beyond "black box" AI models, which provide a result without a clear explanation, to "glass box" models where the reasoning behind a prediction is transparent. This transparency is crucial for building trust, ensuring ethical behavior, and complying with regulations



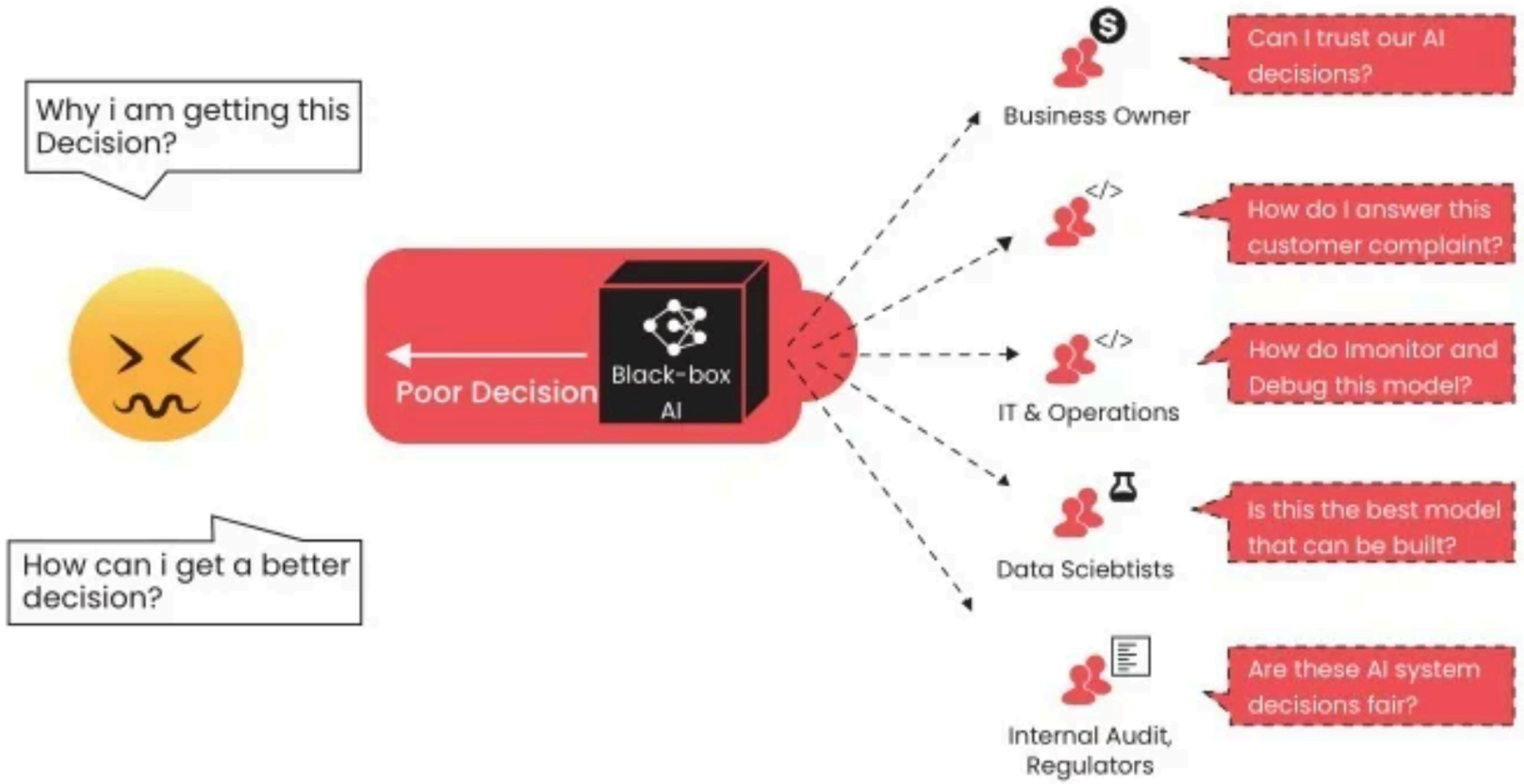
Explainable Artificial Intelligence Concept

What is Explainable AI?





Black-box AI Creates Confusion and Doubt



Why Do We Need It?

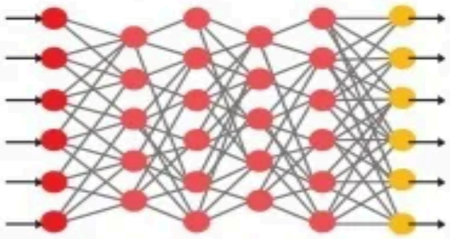
Artificial Intelligence are increasingly implemented in our everyday lives to assist humans in making decisions.

These trivial decisions can vary from a lifestyle choices to more complex decisions such as loan approvals, investments, court decisions and selection of job candidates.

Many AI algorithms are a Blackbox that is not transparent. This leads to trustability concerns. In order to trust these systems, humans want accountability and explanation.

AI System

AI System



We are entering a new age of AI applications

Machine learning is the core technology

Machine learning models are opaque, non-intuitive, and difficult for people to understand

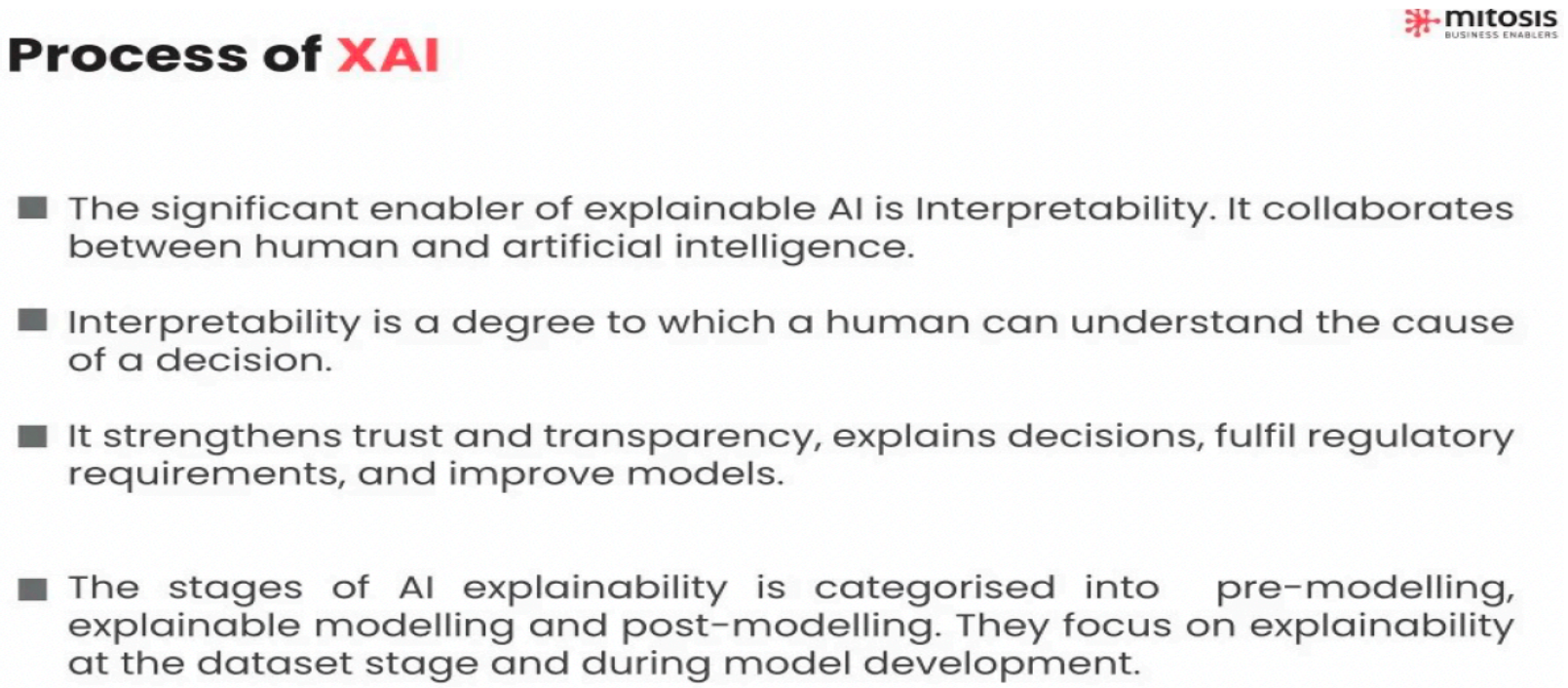
DoD and non-DoD Application

Transportation | Security | Medicine | Finance | Legal | Military

- Why did you do that?
- Why not something else?
- When do you succeed?
- When do you fail?
- When can I trust you?
- How do I correct an error?

Benefits of explainable AI

1. Improved decision-making:-
2. Increased trust and acceptance:-
3. Reduced risks and liabilities:-



Why XAI is Important

The importance of XAI stems from the increasing use of complex AI models in high-stakes fields like healthcare, finance, and criminal justice. In these domains, a decision's reasoning is just as important as the decision itself. XAI helps to:

1. Build Trust and Accountability:

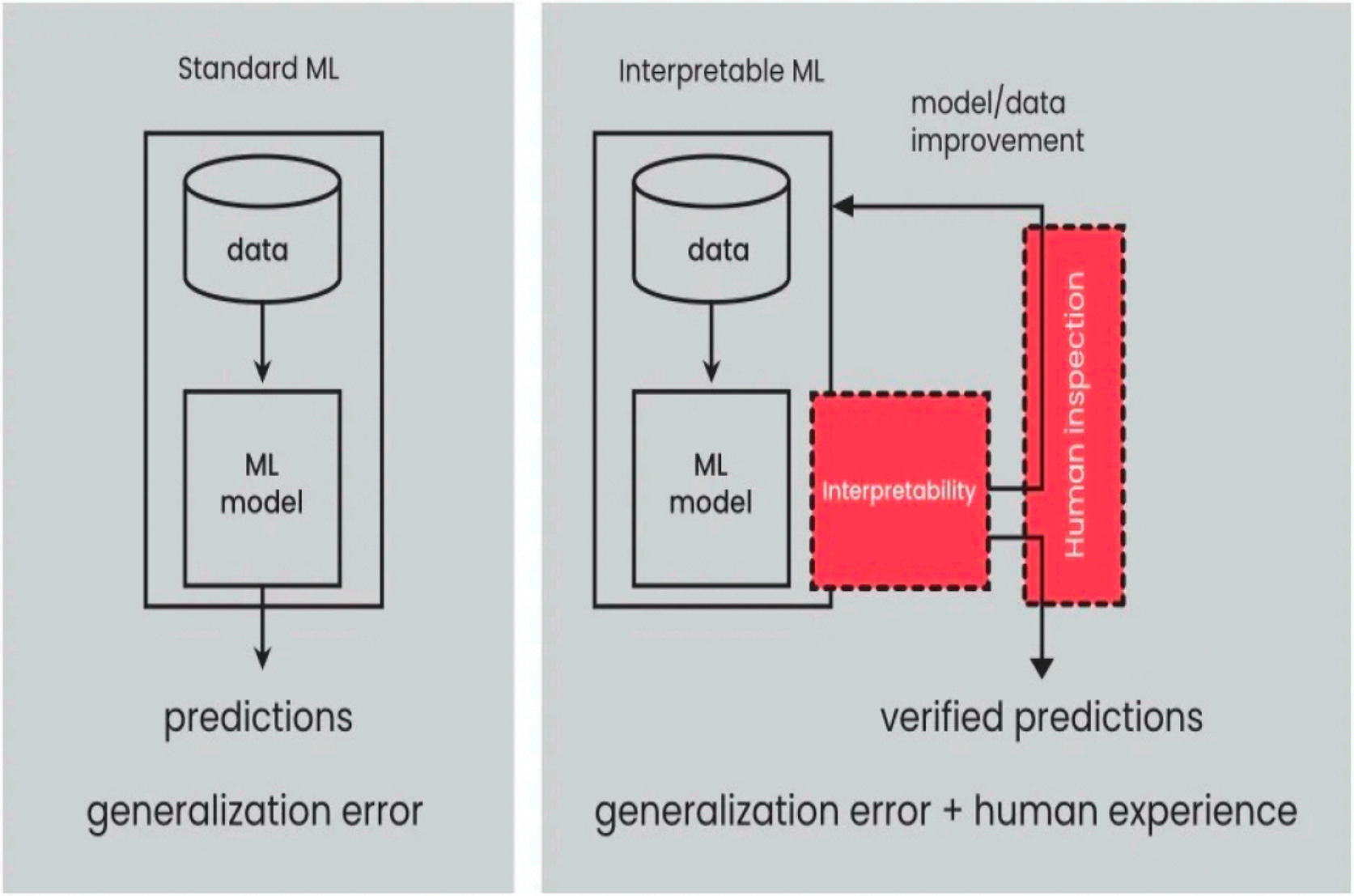
When a system can explain its decisions, users and stakeholders are more likely to trust it. For example, a doctor needs to understand why an AI diagnosed a patient with a certain illness before they can act on that diagnosis.

2. Identify and Mitigate Bias:

AI models can unintentionally learn biases present in their training data. XAI techniques can reveal if a model is making decisions based on unfair or irrelevant criteria, like race or gender, allowing developers to correct the bias.

Why Explainability: Improve ML Model

mitosis
BUSINESS ENABLERS



How does Explainable AI work?

Machine learning model:- The machine learning model is the core component of explainable AI, and represents the underlying algorithms and techniques that are used to make predictions and inferences from data. This component can be based on a wide range of machine learning techniques, such as supervised, unsupervised, or reinforcement learning, and can be used in a range of applications, such as medical imaging, natural language processing, and computer vision.

Explanation algorithm:- The explanation algorithm is the component of explainable AI that is used to provide insights and information about the factors that are most relevant and influential in the model's predictions. This component can be based on different explainable AI approaches, such as feature importance, attribution, and visualization, and can provide valuable insights into the workings of the machine learning model.

Interface:- The interface is the component of explainable AI that is used to present the insights and information generated by the explanation algorithm to humans. This component can be based on a wide range of technologies and platforms, such as web applications, mobile apps, and visualizations, and can provide a user-friendly and intuitive way to access and interact with the insights and information generated by the explainable AI system.

Explainable AI principles

Transparency:- XAI should be transparent and should provide insights and information about the factors that are most relevant and influential in the model's predictions. This transparency can help to build trust and acceptance of XAI and can provide valuable insights and benefits in different domains and applications.

Interpretability:- XAI should be interpretable and should provide a clear and intuitive way to understand and use the insights and information generated by XAI. This interpretability can help to overcome the challenges and limitations of traditional machine learning models, which are often opaque and inscrutable, and can provide valuable insights and benefits in different domains and applications.

Accountability:- XAI should be accountable and should provide a framework for addressing the regulatory and ethical considerations of machine learning. This accountability can help to ensure that XAI is used in a responsible and accountable manner, and can provide valuable insights and benefits in different domains and applications.

Explainable AI (XAI) Techniques

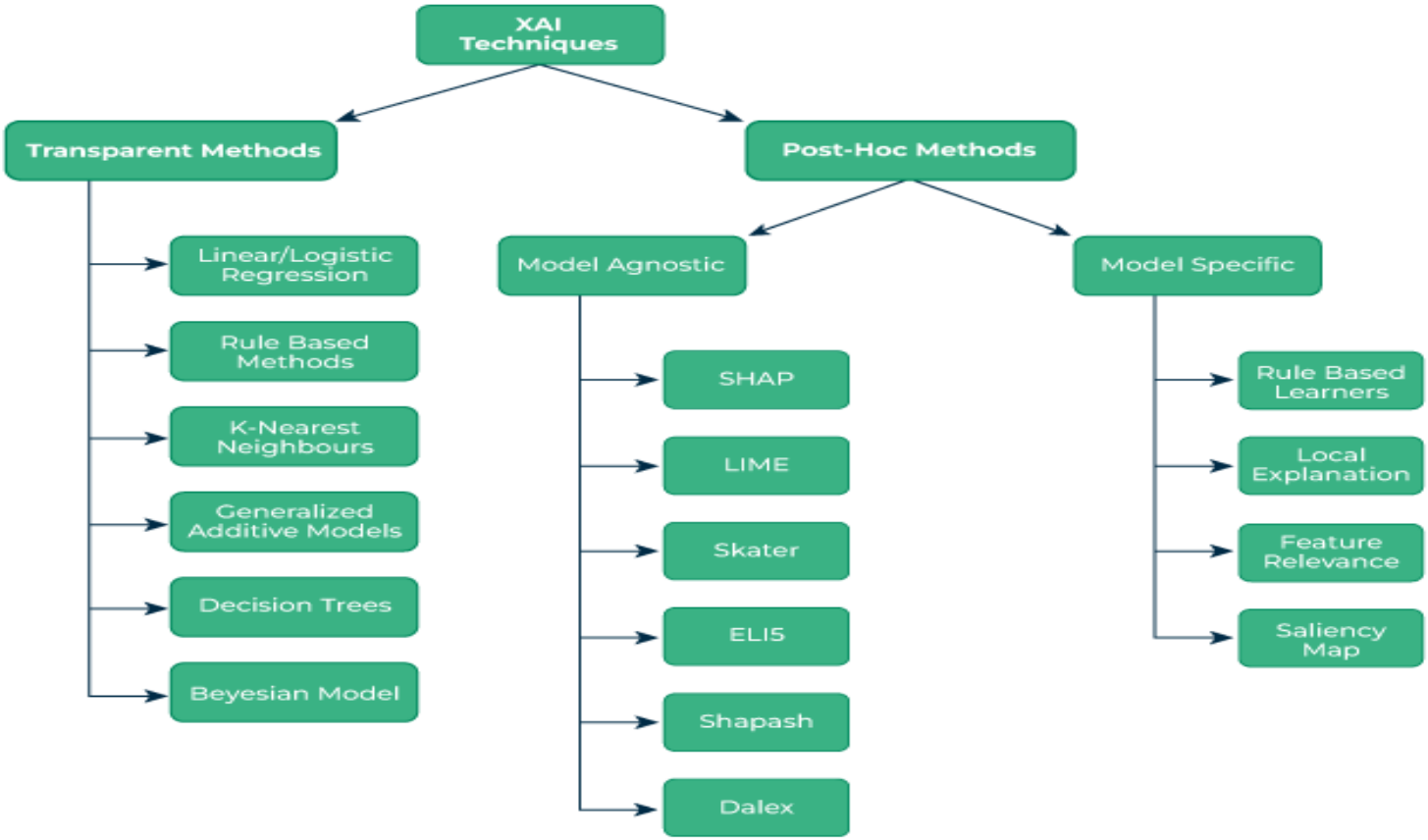
To implement explainable AI (XAI) in python, you can use one of the following approaches:

LIME (Local Interpretable Model-agnostic Explanations):- LIME is a popular XAI approach that uses a local approximation of the model to provide interpretable and explainable insights into the factors that are most relevant and influential in the model's predictions. To implement LIME in python, you can use the lime package, which provides a range of tools and functions for generating and interpreting LIME explanations.

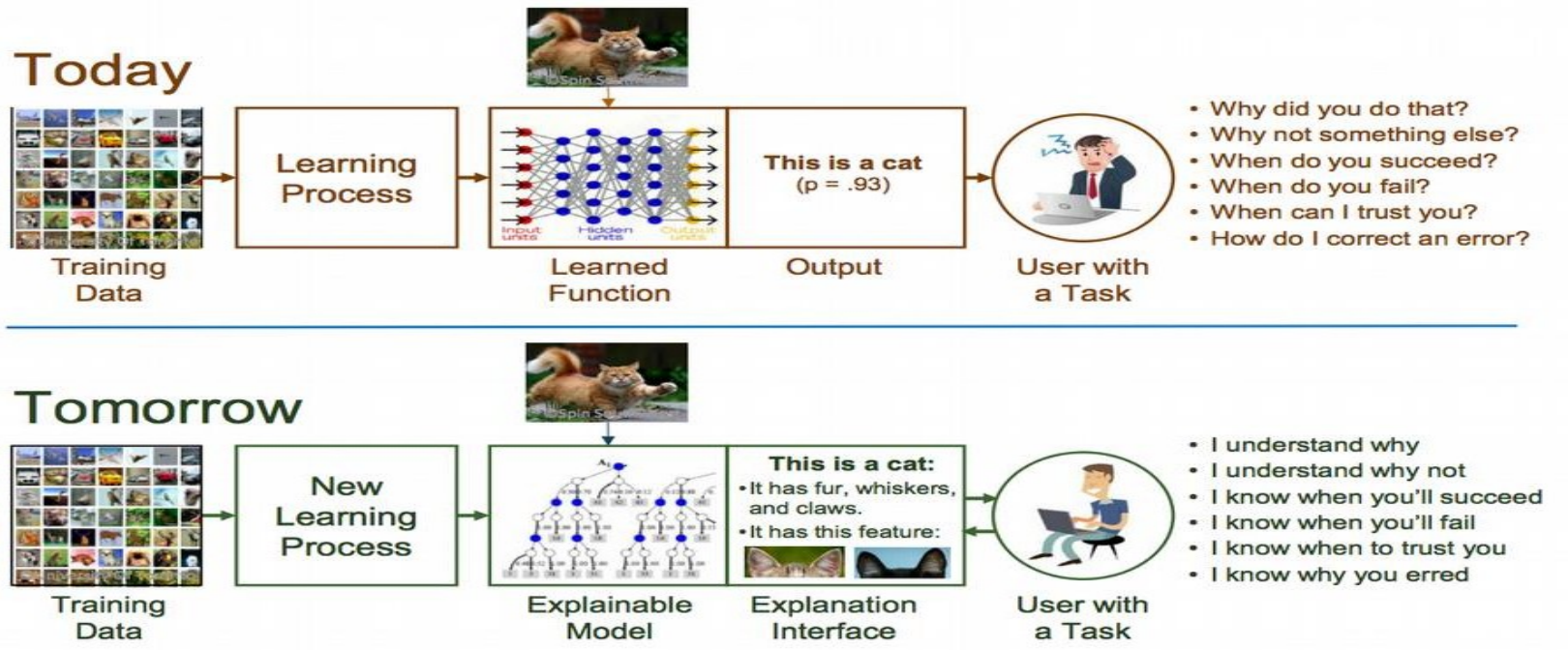
SHAP (SHapley Additive exPlanations):- SHAP is an XAI approach that uses the Shapley value from game theory to provide interpretable and explainable insights into the factors that are most relevant and influential in the model's predictions. To implement SHAP in python, you can use the shap package, which provides a range of tools and functions for generating and interpreting SHAP explanations.

ELI5 (Explain Like I'm 5):- ELI5 is an XAI approach that provides interpretable and explainable insights into the factors that are most relevant and influential in the model's predictions, using a simple and intuitive language that can be understood by non-experts. To implement ELI5 in python, you can use the eli5 package, which provides a range of tools and functions for generating and interpreting ELI5 explanations.

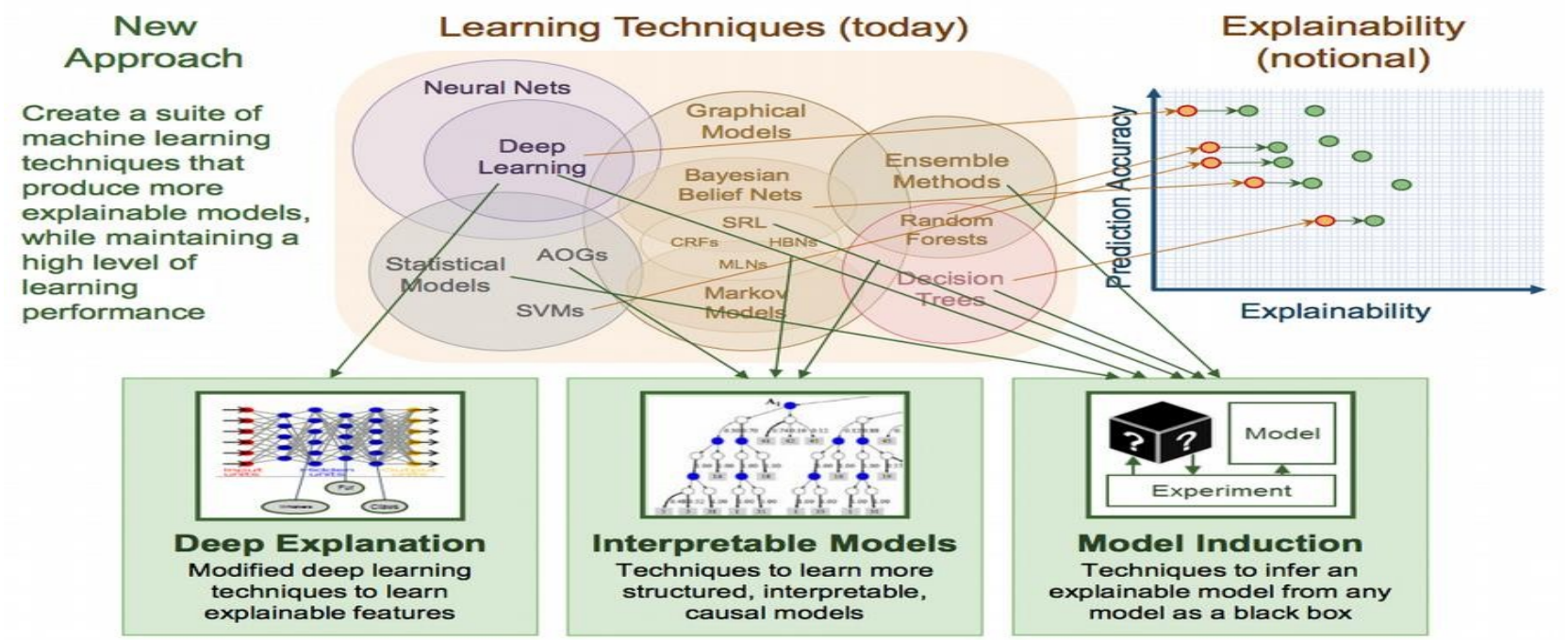
Overall, there are several approaches that you can use to implement XAI in python, including LIME, SHAP, and ELI5. These approaches provide different levels of interpretability and explainability and can be used in a range of applications and domains.



Explainable AI



XAI Approaches



References

- Stuart Russell and Peter Norvig, "Artificial Intelligence: A Modern Approach", Third edition, Pearson, 2013.
- E. Rich and K.Knight, Artificial Intelligence, Tata McGraw Hill, 1992.
- Tom Mitchell , Machine Learning, McGraw hill publication
- Ethem Alpaydin, " Introduction to Machine Learning", PHI 2nd Edition-2013
- Karan Kathpalia, Reinforcement Learning, https://www.cs.princeton.edu/courses/archive/spring17/cos598F/lectures/RL.pptx
- "Reinforcement Learning – An Introduction" by Richard Sutton and Andrew Barto
- Transfer Learning in keras with Computer Vision Models, https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/
- Transfer Learning, https://lisaong.github.io/mldds-courseware/03_TextImage/transfer-learning.slides.html
- Explainable AI: Opening up the Blackbox, Ryan Wesslen, UNC Charllotte, bitly.com/xai-Davidson
- Edge AI: the Future of Artificial Intelligence and Edge Computing, https://www.itbusinessedge.com/data-center/developments-edge-ai/
- The Future of Computing: How Edge Computing is Revolutionizing the Tech Industry - Telecom Reseller/?cid=em&source=elo&campid=smggmo_APJ_gmocoma_EMNL_EN_2022_EN_BitFeed_IOT_C-MKA-30175_T-MKA-30221_20230217&content=smggmo_APJ_gmocoma_EMNL_EN_2022_EN_BitFeed_IOT_C-MKA-30175_T-MKA-30221_20230217&elq_cid=5273033&em_id=89755&elqrid=87a4195da4a14949aa9e75277464150c&elqcampid=55747&erpm_id=8199278
- Chatbot, https://www.cse.iitb.ac.in/~cs626-449/cs626-460-2008/public_html/2008/student_seminars/group7-Chatbots.ppt
- Applications of AI-chatbot, http://tusharkute.com
- How to build a chatbot, https://www.ibm.com/products/watsonx-assistant/resources/how-to-build-a-chatbot

Edge AI

What is Edge AI?

Edge AI, or Artificial Intelligence at the Edge, is the practice of running AI algorithms and machine learning models directly on local devices, such as sensors, smartphones, cameras, and other IoT (Internet of Things) devices. Instead of sending all data to a centralized cloud server for processing and analysis, Edge AI allows for real-time decision-making and data processing to happen right where the data is generated.

How It Works

The process typically involves these steps:

Model Training: The AI model is initially trained on a large dataset in a powerful, centralized environment, like the cloud. This is where the heavy computational lifting happens.

Model Deployment: The pre-trained, optimized AI model is then deployed to the "edge device" (e.g., a smart camera, a factory robot, or a drone). These models are often made smaller and more efficient to run on the limited processing power of the device.

Local Inference: The edge device uses its on-board AI model to process real-time data it collects from its surroundings (e.g., video, audio, or sensor data).

Action and Feedback: Based on the results of the local processing, the device can take immediate action (e.g., stop a machine, send an alert, or adjust settings). While the device can operate autonomously, it may still periodically send back filtered data or insights to the cloud for further analysis and for the central model to be updated.

Key Benefits of Edge AI

Edge AI offers several significant advantages over traditional cloud-based AI:

Low Latency and Real-Time Processing: By processing data locally, Edge AI eliminates the delays that come with transmitting data to a distant server and waiting for a response. This is critical for applications where split-second decisions are necessary, such as in autonomous vehicles or industrial automation.

Enhanced Data Privacy and Security: Sensitive data is processed on the device itself and may never be sent over a network. This reduces the risk of data breaches and helps organizations comply with privacy regulations.

Reduced Bandwidth and Cost: Only necessary data, like insights or alerts, is sent back to the cloud, rather than massive amounts of raw data. This lowers network bandwidth usage and associated costs.

Improved Reliability: Edge AI systems can function without a constant internet connection, making them more reliable in remote locations or in situations with poor network connectivity.

Energy Efficiency: Local processing on the device often requires less power compared to continuously sending data to a power-intensive cloud data center.

Edge AI vs. Cloud AI

Feature	Edge AI	Cloud AI
Data Processing	On-device, at the source of data generation.	In a centralized, remote data center.
Latency	Extremely low, enabling real-time responses.	Higher, due to network transmission and processing time.
Bandwidth	Low, as only filtered data or results are sent.	High, as all raw data must be transmitted.
Privacy & Security	High, as sensitive data remains on the device.	Lower, as data is transferred over a network.
Computational Power	Limited by the device's hardware.	Nearly unlimited and scalable.
Primary Role	Inference and real-time decision-making.	Training, large-scale data analysis, and model updates.

Real-World Applications

Edge AI is being adopted across a wide range of industries:

Autonomous Vehicles: Cars use Edge AI to process sensor data (from cameras, LiDAR, and radar) in real-time to detect objects, navigate roads, and avoid collisions.

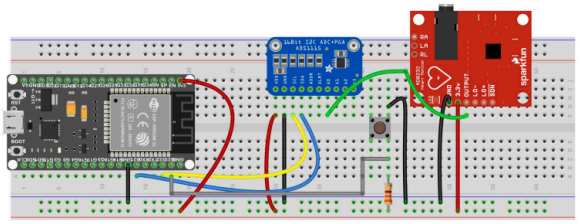
Smart Homes: Devices like smart speakers and security cameras use Edge AI to process voice commands and detect motion locally, improving speed and privacy.

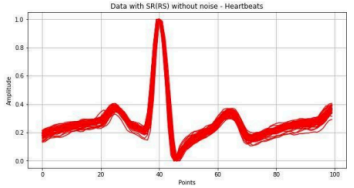
Industrial Automation: In manufacturing, Edge AI can be used for predictive maintenance, analyzing vibrations or sounds from machinery to predict failures before they occur. It's also used for quality control, where cameras with Edge AI can instantly detect defects on a production line.

Healthcare: Wearable devices and medical sensors use Edge AI to monitor a patient's vital signs and send real-time alerts to healthcare providers for anomalies.

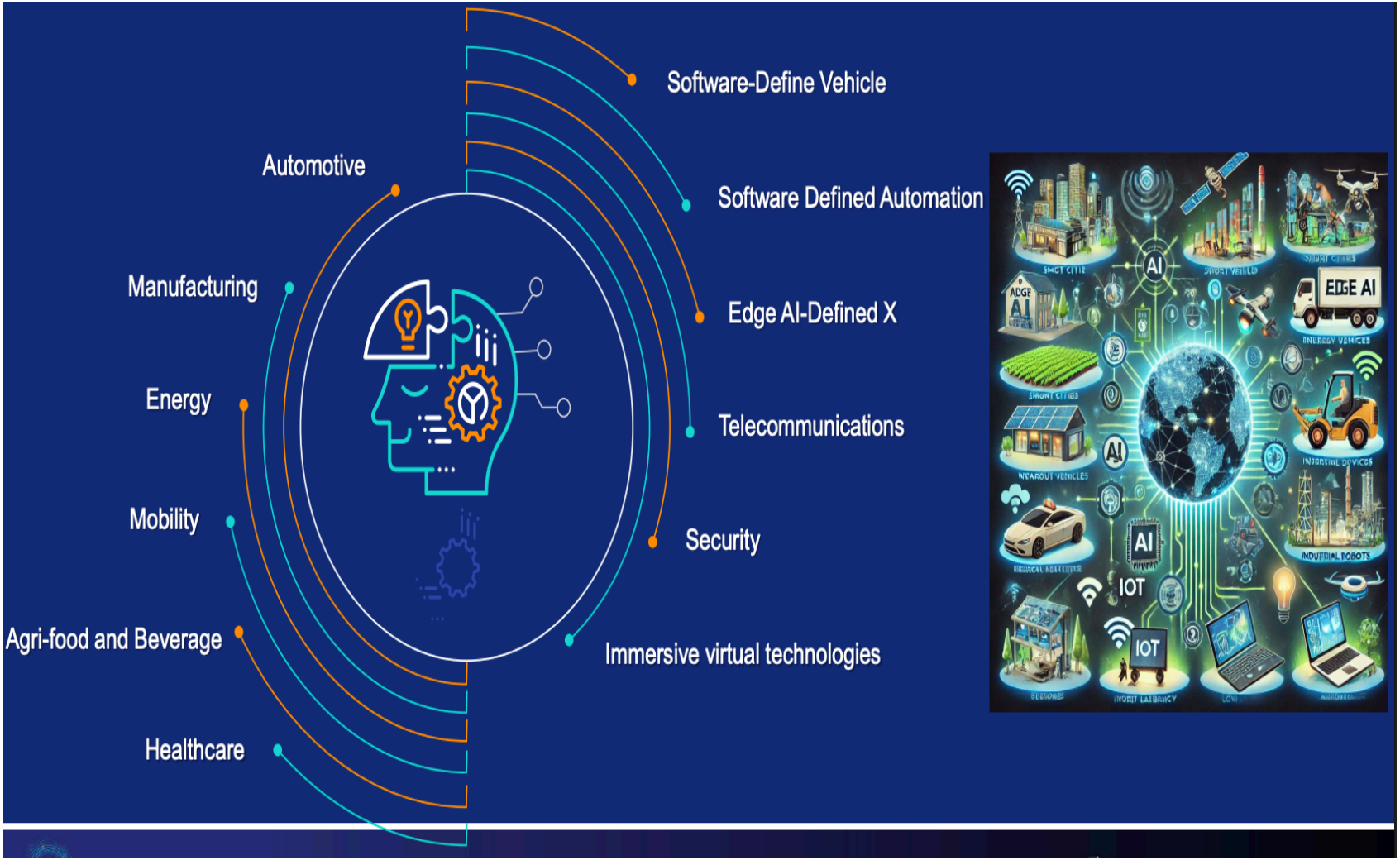
Smart Cities: Edge AI in traffic lights can adjust their timing based on real-time traffic conditions to reduce congestion. Surveillance cameras can also use it to monitor public safety without sending all video streams to the cloud.

Health - Human Sensing





Guilherme Silva
Engenheiro - UNIFEI


Software-Define Vehicle
Automotive
Software Defined Automation
Manufacturing
Edge AI-Defined X
Energy
Telecommunications
Mobility
Security
Agri-food and Beverage
Immersive virtual technologies
Healthcare
AI
EDGE AI
IOT
IOT

AI Ethics

- Ethics is a set of moral principles which help us discern between right and wrong.
- Since the middle of the 2010s, a visible discourse on the ethics of artificial intelligence (AI) has developed.
- AI ethics is a multidisciplinary field that studies how to optimize AI's beneficial impact while reducing risks and adverse outcomes.
- Provide guidance in the development of practical interventions.
- Most prominent AI ethics issues are also human rights issues (privacy, equality and non-discrimination).
- We study AI ethics to ensure that AI is developed and used in a way that is beneficial to society, reduces unfavourable outcomes and aligned with human values.

Introduction to AI Ethics

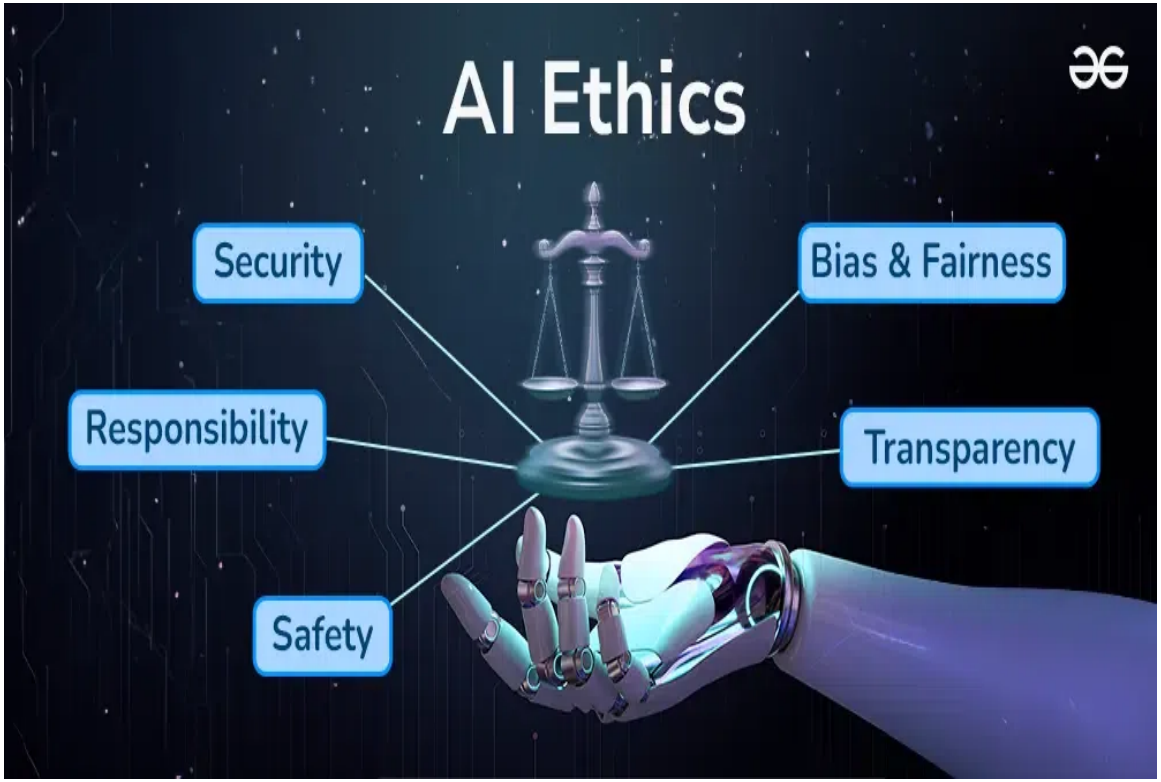
➢AI ethics deals with moral principles guiding AI development and use

➢Ensures AI benefits society while minimizing harm

➢Focuses on fairness, accountability, transparency, and human values

Key Principles of AI Ethics

1. Fairness & Non-discrimination
2. Transparency & Explainability
3. Accountability
4. Privacy & Data Protection
5. Safety & Security
6. Human-Centric Values
7. Sustainability



Example: Fairness in Recruitment AI

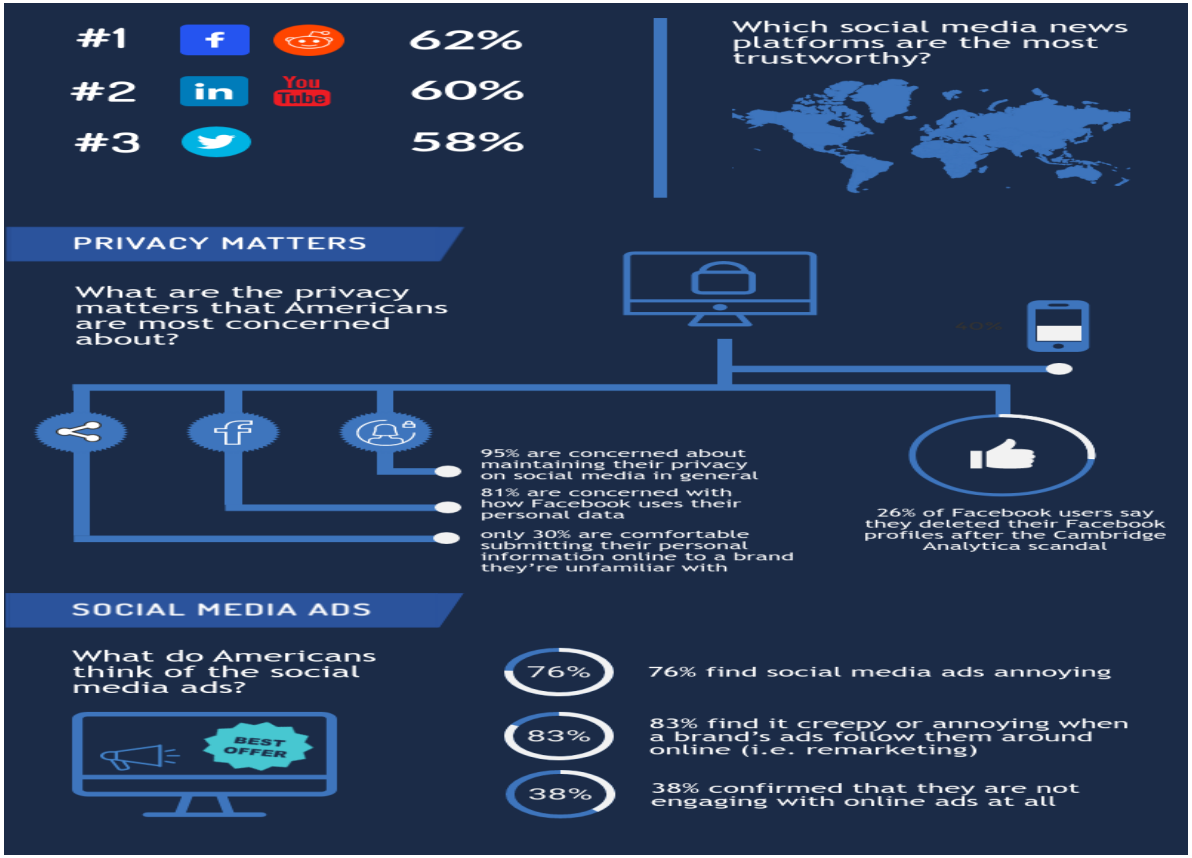
- AI tools in hiring can unintentionally favor certain groups
- Example: Amazon's AI recruitment tool was found biased against women
- Ethical AI must ensure equal opportunity and remove bias in datasets

Example: Transparency in Healthcare

- AI models used for diagnosis must be explainable
- Doctors and patients need to understand AI reasoning
- Black-box decisions risk patient trust and safety

Example: Privacy in Social Media AI

- AI algorithms track user data to personalize ads
- Raises ethical concerns about consent and surveillance
- Example: Cambridge Analytica scandal highlighted misuse of personal data



Applications of AI

- Healthcare – AI for early diagnosis & drug discovery
- Finance – Fraud detection, credit scoring
- Education – Personalized learning tools
- Law Enforcement – Predictive policing (ethical concerns)
- Environment – Climate modeling and smart energy use

Challenges and Future Directions

- Bias and discrimination remain major risks
- Lack of global AI ethical standards
- Need for AI governance and regulation
- Future: Building responsible, explainable, and human-centered AI

Conclusion

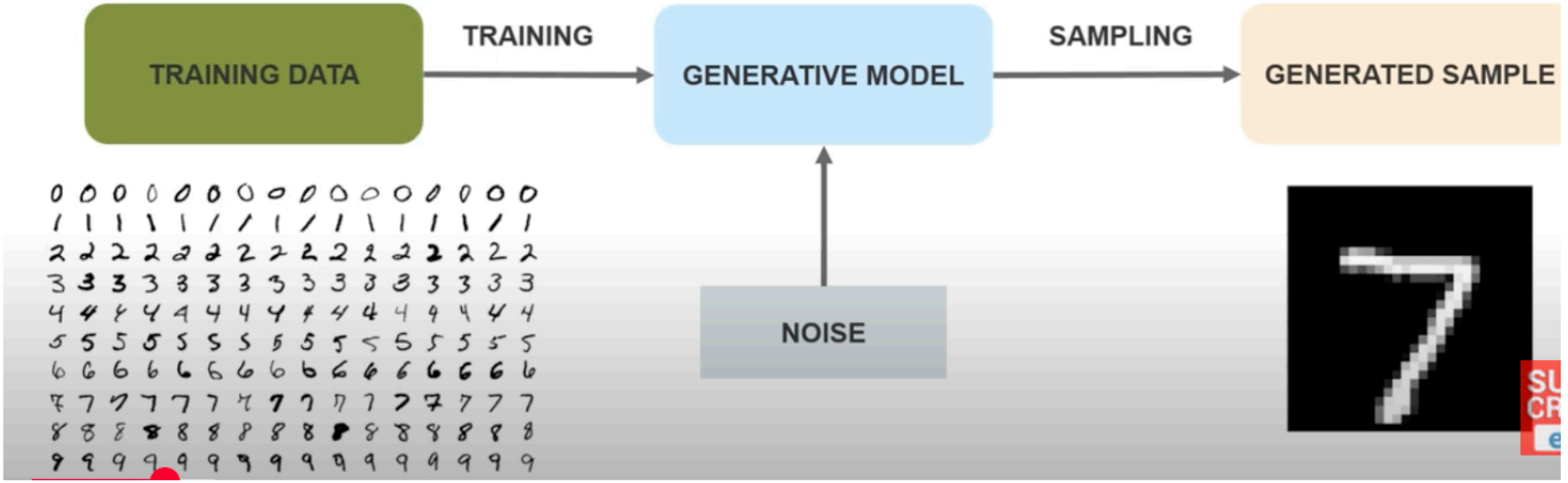
- AI ethics is essential for responsible innovation
- Must balance technological progress with human values
- AI should serve society fairly, transparently, and sustainably

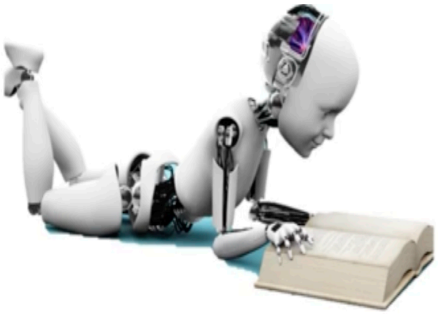
Case Studies - Image Generation with GANs

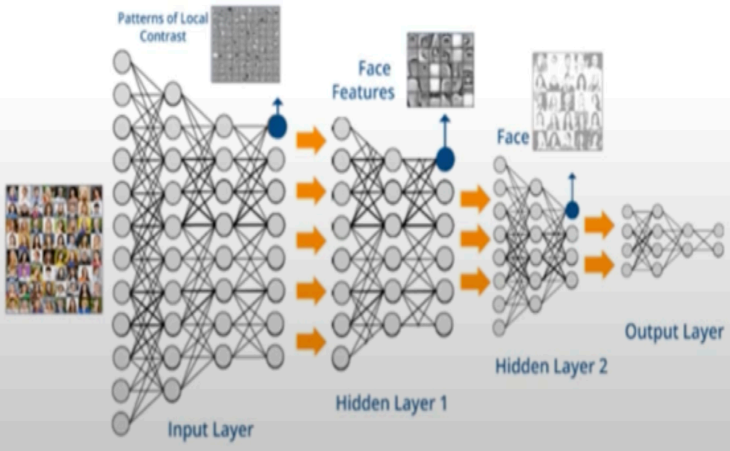
Generative Adversarial Networks (GANs) are a type of deep learning model that have revolutionized image generation by creating highly realistic and novel images. They consist of a generator network that creates synthetic images and a discriminator network that evaluates their authenticity. Through an adversarial process, the generator learns to produce images that are increasingly difficult for the discriminator to distinguish from real ones, leading to high-fidelity outputs

Generative Model

Generative models are nothing but those models that use an unsupervised learning approach in a generative model there are samples in the data that is input variables X but it lacks the output variable Y and we use the only input variables to train the generative model and it recognizes patterns from the input variables to generate an output that is unknown and based on the training data only in supervised learning we are more aligned towards creating predictive models from the input variables and this type of modeling is also known as discriminative modeling and in a classification problem





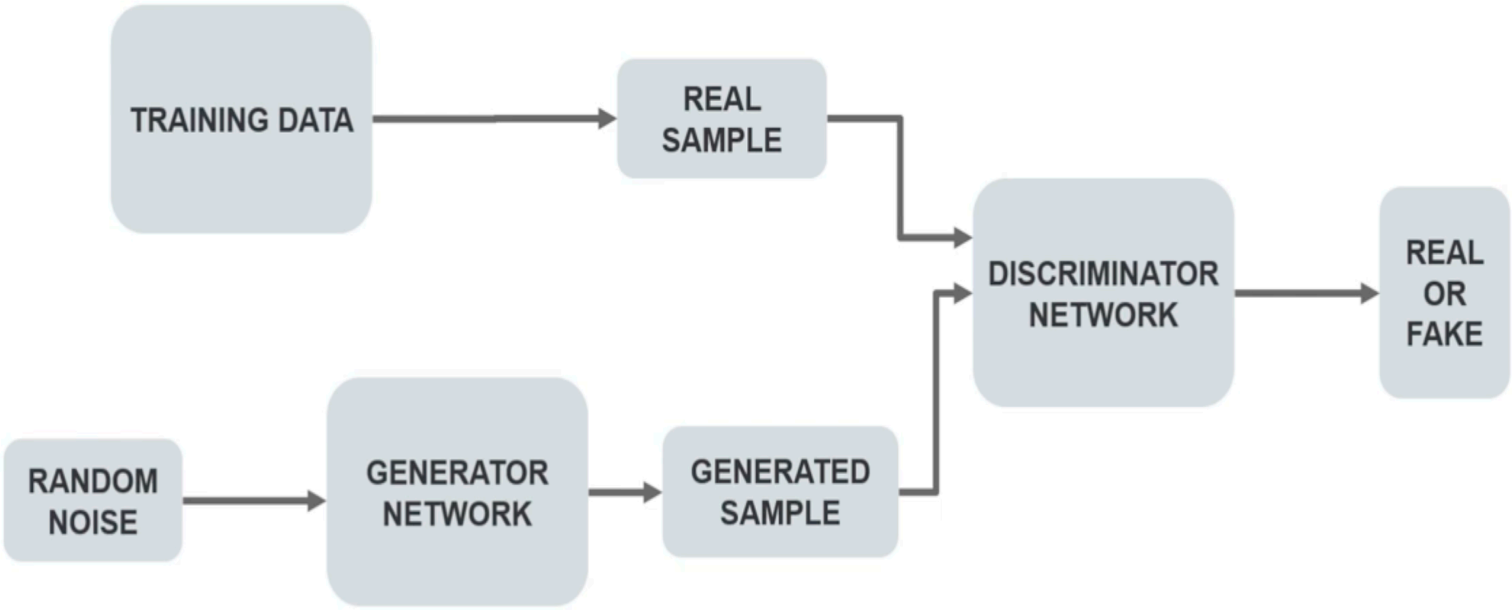


GENERATOR

Generator network that takes a sample and generates a sample of data

DISCRIMINATOR

Discriminator network decides whether the data is generated or taken from the real sample using a binary classification pr oblem with the help of a sigmoid function that gives the output in the range 0 to 1.


TRAINING DATA
REAL SAMPLE
DISCRIMINATOR NETWORK
REAL OR FAKE
RANDOM NOISE
GENERATOR NETWORK
GENERATED SAMPLE

$$V(D,G) = E_{x\sim Pdata(x)}\ [\log D(x)] + E_{z\sim pz(z)}\ [\log(1 - D(G(z))]$$

G = Generator

D = Discriminator

Pdata(x) = Distribution of real data

Pdata(z) = Distributor of generator

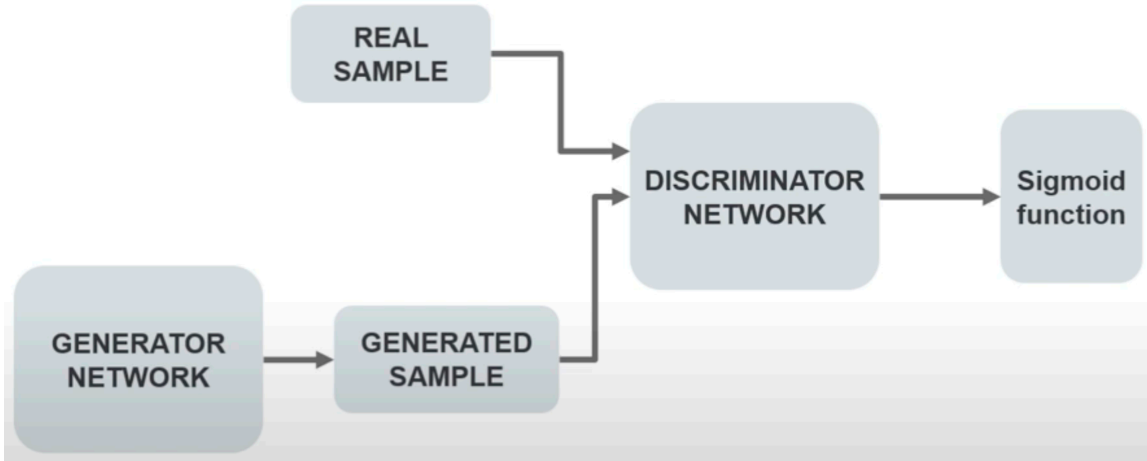
x = sample from real data

z = sample from generator

D(x) = Discriminator Network

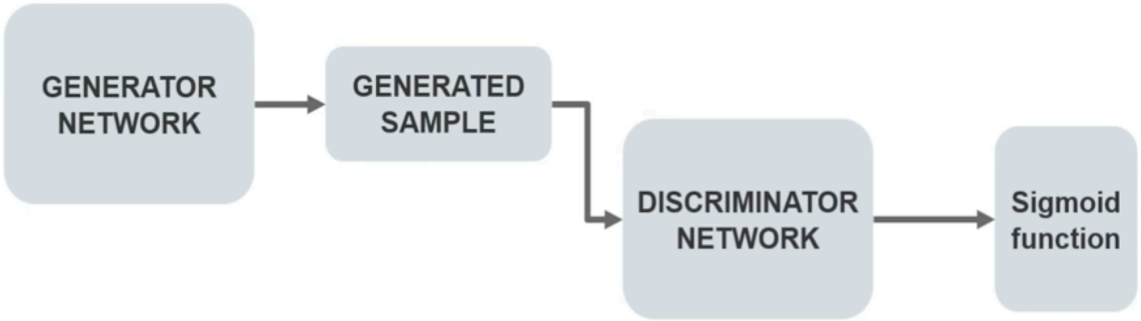
G(z) = Generator Network

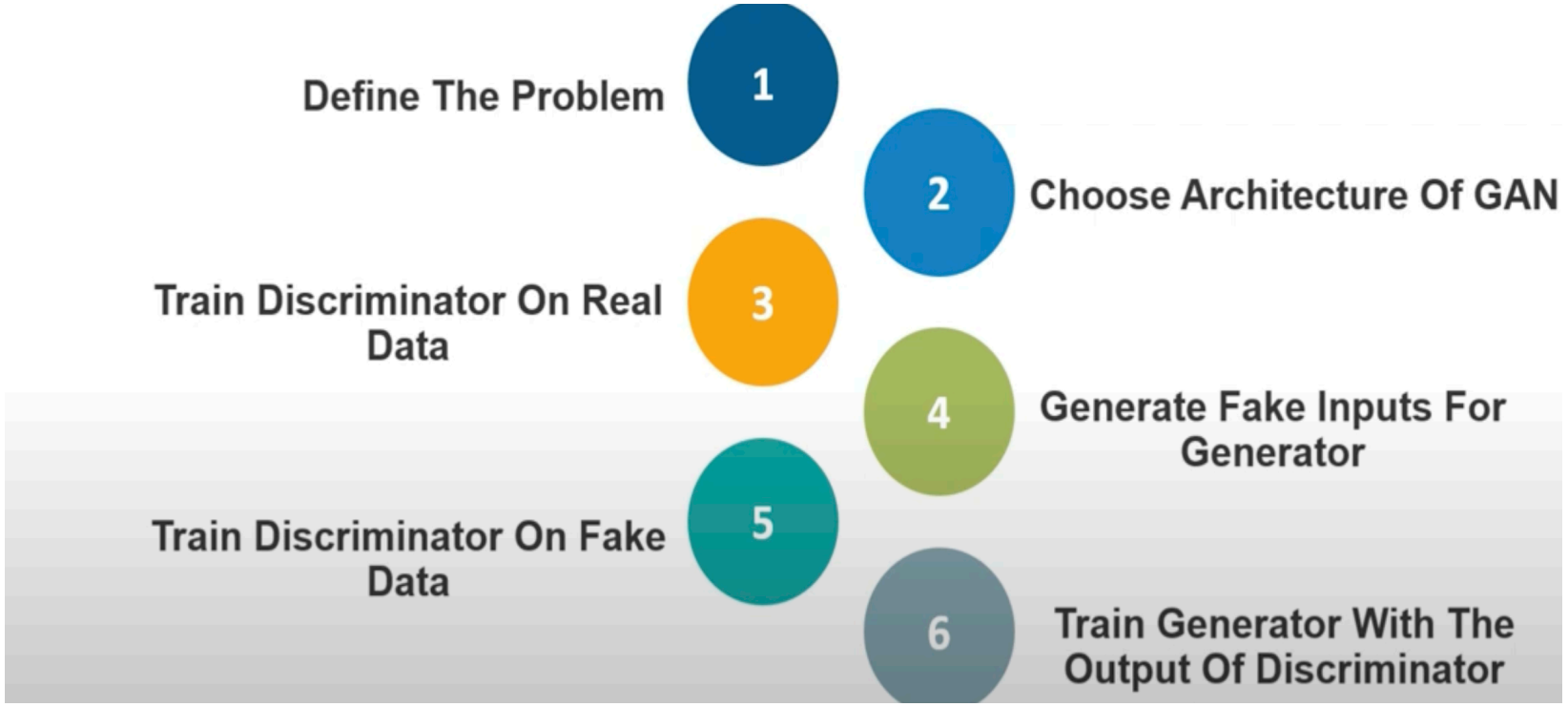
How To Train A GAN?



Train the discriminator and freeze the generator, which means the training set for the generator is turned as False and the network will only do the forward pass and no back-propagation will be applied

Train the generator and freeze the discriminator. In this phase, we get the results from the first phase and can use them to make better from the previous state to try and fool the discriminator better.




Define The Problem
1
2
Choose Architecture Of GAN
Train Discriminator On Real Data
3
4
Generate Fake Inputs For Generator
Train Discriminator On Fake Data
5
6
Train Generator With The Output Of Discriminator

Challenges Faced By GANs

- Problem of stability between generator and discriminator
- Problem to determine positioning of the objects
- Problem in understanding the perspective
- Problem in understanding global objects

GANs Applications

Image To Image Translation

Real

Generated

Reconstructed

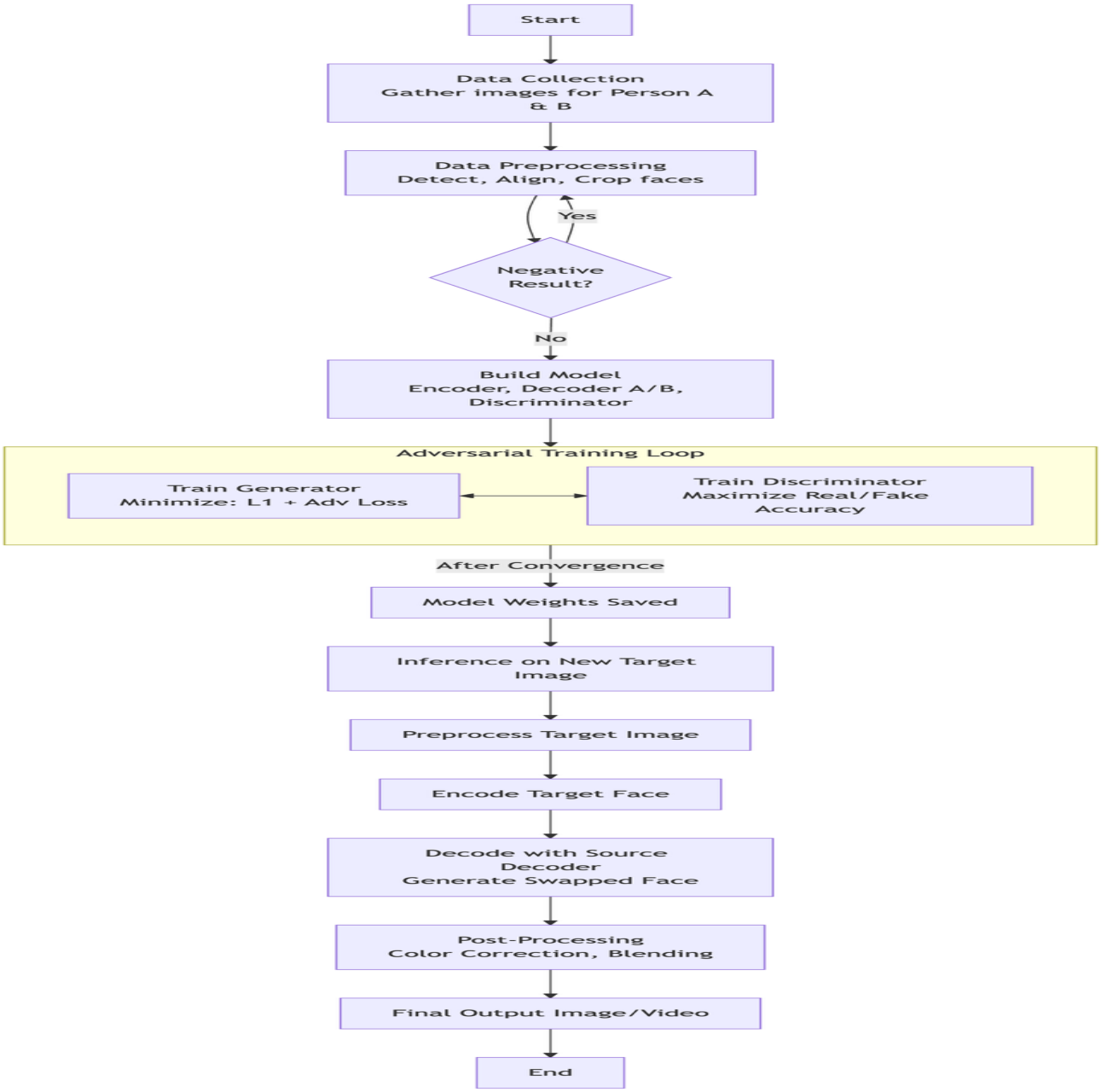
Case Study: Implementing a DeepFake Image Generation System

The entire process can be divided into three main phases:

Data Preparation: Gathering and processing the raw images.

Model Training: The core adversarial training process.

Inference & Post-Processing: Using the trained model to generate the final output.


Start
Data Collection
Gather images for Person A & B
Data Preprocessing
Detect, Align, Crop faces
Yes
Negative Result?
No
Build Model
Encoder, Decoder A/B, Discriminator
Adversarial Training Loop
Train Generator
Minimize: L1 + Adv Loss
Train Discriminator
Maximize Real/Fake Accuracy
After Convergence
Model Weights Saved
Inference on New Target Image
Preprocess Target Image
Encode Target Face
Decode with Source Decoder
Generate Swapped Face
Post-Processing
Color Correction, Blending
Final Output Image/Video
End

Thank you