GENERATIVE AI AND LARGE LANGUAGE MODELS

SCSB4014

UNIT - 1

INTRODUCTION TO GENERATIVE AI

[History of AI - Key Milestones - Basics of Neural Network - Generative Models Vs Autoregressive Models, Applications of Generative AI - Text Generation, Image Manipulation - AI Agents – Gen AI Framework: Langchain]

HISTORY OF AI

Artificial Intelligence is not a new word and not a new technology for researchers. This technology is much older than you would imagine. Even there are the myths of Mechanical men in Ancient Greek and Egyptian Myths. Following are some milestones in the history of AI which defines the journey from the AI generation to till date development.

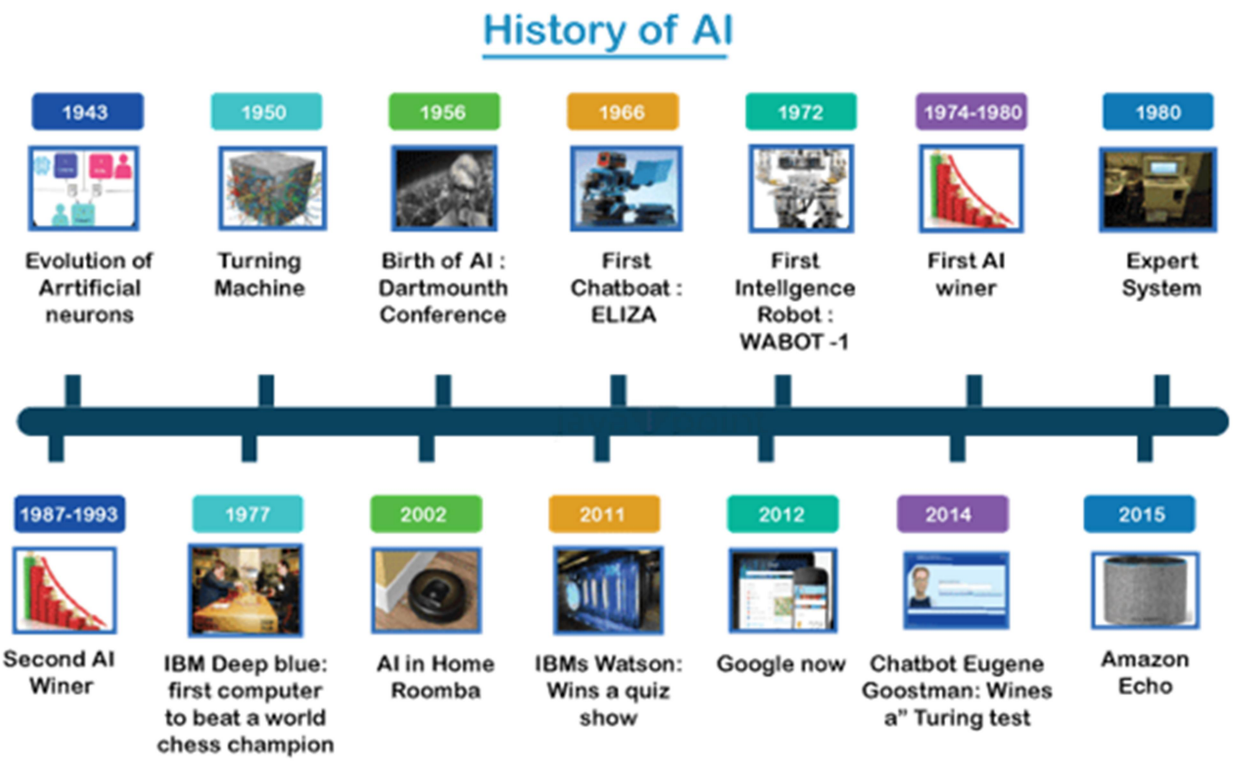


Fig. History of AI

Maturation of Artificial Intelligence (1943-1952)

Between 1943 and 1952, there was notable progress in the expansion of artificial intelligence (AI). Throughout this period, AI transitioned from a mere concept to tangible experiments and practical applications. Here are some key events that happened during this period:

- **Year 1943:** The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of **artificial neurons**.
- **Year 1949:** Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called Hebbian learning.

- **Year 1950:** The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes **"Computing Machinery and Intelligence"** in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a **Turing test**.
- **Year 1951:** Marvin Minsky and Dean Edmonds created the initial artificial neural network (ANN) named SNARC. They utilized 3,000 vacuum tubes to mimic a network of 40 neurons.

The birth of Artificial Intelligence (1952-1956)

From 1952 to 1956, AI surfaced as a unique domain of investigation. During this period, pioneers and forward-thinkers commenced the groundwork for what would ultimately transform into a revolutionary technological domain. Here are notable occurrences from this era:

- **Year 1952:** Arthur Samuel pioneered the creation of the Samuel Checkers-Playing Program, which marked the world's first self-learning program for playing games.
- **Year 1955:** An Allen Newell and Herbert A. Simon created the "first artificial intelligence program"Which was named as **"Logic Theorist"**. This program had proved 38 of 52 Mathematics theorems, and find new and more elegant proofs for some theorems.
- **Year 1956:** The word "Artificial Intelligence" first adopted by American Computer scientist John McCarthy at the Dartmouth Conference. For the first time, AI coined as an academic field.

At that time high-level computer languages such as FORTRAN, LISP, or COBOL were invented. And the enthusiasm for AI was very high at that time.

The golden Years-Early enthusiasm (1956-1974)

The period from 1956 to 1974 is commonly known as the "Golden Age" of artificial intelligence (AI). In this timeframe, AI researchers and innovators were filled with enthusiasm and achieved remarkable advancements in the field. Here are some notable events from this era:

- **Year 1958:** During this period, Frank Rosenblatt introduced the perceptron, one of the early artificial neural networks with the ability to learn from data. This invention laid the foundation for modern neural networks. Simultaneously, John McCarthy developed the

Lisp programming language, which swiftly found favor within the AI community, becoming highly popular among developers.

- **Year 1959:** Arthur Samuel is credited with introducing the phrase "machine learning" in a pivotal paper in which he proposed that computers could be programmed to surpass their creators in performance. Additionally, Oliver Selfridge made a notable contribution to machine learning with his publication "Pandemonium: A Paradigm for Learning." This work outlined a model capable of self-improvement, enabling it to discover patterns in events more effectively.
- **Year 1964:** During his time as a doctoral candidate at MIT, Daniel Bobrow created STUDENT, one of the early programs for natural language processing (NLP), with the specific purpose of solving algebra word problems.
- **Year 1965:** The initial expert system, Dendral, was devised by Edward Feigenbaum, Bruce G. Buchanan, Joshua Lederberg, and Carl Djerassi. It aided organic chemists in identifying unfamiliar organic compounds.
- **Year 1966:** The researchers emphasized developing algorithms that can solve mathematical problems. Joseph Weizenbaum created the first chatbot in 1966, which was named ELIZA. Furthermore, Stanford Research Institute created Shakey, the earliest mobile intelligent robot incorporating AI, computer vision, navigation, and NLP. It can be considered a precursor to today's self-driving cars and drones.
- **Year 1968:** Terry Winograd developed SHRDLU, which was the pioneering multimodal AI capable of following user instructions to manipulate and reason within a world of blocks.
- **Year 1969:** Arthur Bryson and Yu-Chi Ho outlined a learning algorithm known as backpropagation, which enabled the development of multilayer artificial neural networks. This represented a significant advancement beyond the perceptron and laid the groundwork for deep learning. Additionally, Marvin Minsky and Seymour Papert authored the book "Perceptrons," which elucidated the constraints of basic neural networks. This publication led to a decline in neural network research and a resurgence in symbolic AI research.

- **Year 1972:** The first intelligent humanoid robot was built in Japan, which was named WABOT-1.
- **Year 1973:** James Lighthill published the report titled "Artificial Intelligence: A General Survey," resulting in a substantial reduction in the British government's backing for AI research.

The first AI winter (1974-1980)

The initial AI winter, occurring from 1974 to 1980, is known as a tough period for artificial intelligence (AI). During this time, there was a substantial decrease in research funding, and AI faced a sense of letdown.

- The duration between years 1974 to 1980 was the first AI winter duration. AI winter refers to the time period where computer scientist dealt with a severe shortage of funding from government for AI researches.
- During AI winters, an interest of publicity on artificial intelligence was decreased.

A boom of AI (1980-1987)

Between 1980 and 1987, AI underwent a renaissance and newfound vitality after the challenging era of the First AI Winter. Here are notable occurrences from this timeframe:

- In 1980, the first national conference of the American Association of Artificial Intelligence was held at Stanford University.
- **Year 1980:** After AI's winter duration, AI came back with an "Expert System". Expert systems were programmed to emulate the decision-making ability of a human expert. Additionally, Symbolics Lisp machines were brought into commercial use, marking the onset of an AI resurgence. However, in subsequent years, the Lisp machine market experienced a significant downturn.
- **Year 1981:** Danny Hillis created parallel computers tailored for AI and various computational functions, featuring an architecture akin to contemporary GPUs.
- **Year 1984:** Marvin Minsky and Roger Schank introduced the phrase "AI winter" during a gathering of the Association for the Advancement of Artificial Intelligence. They cautioned the business world that exaggerated expectations about AI would result in

disillusionment and the eventual downfall of the industry, which indeed occurred three years later.

- **Year 1985:** Judea Pearl introduced Bayesian network causal analysis, presenting statistical methods for encoding uncertainty in computer systems.

The second AI winter (1987-1993)

- The duration between the years 1987 to 1993 was the second AI Winter duration.
- Again Investors and government stopped in funding for AI research as due to high cost but not efficient result. The expert system such as XCON was very cost effective.

The emergence of intelligent agents (1993-2011)

Between 1993 and 2011, there were significant leaps forward in artificial intelligence (AI), particularly in the development of intelligent computer programs. During this era, AI professionals shifted their emphasis from attempting to match human intelligence to crafting pragmatic, ingenious software tailored to specific tasks. Here are some noteworthy occurrences from this timeframe:

- **Year 1997:** In 1997, IBM's Deep Blue achieved a historic milestone by defeating world chess champion Gary Kasparov, marking the first time a computer triumphed over a reigning world chess champion. Moreover, Sepp Hochreiter and Jürgen Schmidhuber introduced the Long Short-Term Memory recurrent neural network, revolutionizing the capability to process entire sequences of data such as speech or video.
- **Year 2002:** for the first time, AI entered the home in the form of Roomba, a vacuum cleaner.
- **Year 2006:** AI came into the Business world till the year 2006. Companies like Facebook, Twitter, and Netflix also started using AI.
- **Year 2009:** Rajat Raina, Anand Madhavan, and Andrew Ng released the paper titled "Utilizing Graphics Processors for Extensive Deep Unsupervised Learning," introducing the concept of employing GPUs for the training of expansive neural networks.
- **Year 2011:** Jürgen Schmidhuber, Dan Claudiu Cire?an, Ueli Meier, and Jonathan Masci created the initial CNN that attained "superhuman" performance by emerging as the victor in the German Traffic Sign Recognition competition. Furthermore, Apple launched

Siri, a voice-activated personal assistant capable of generating responses and executing actions in response to voice commands.

Deep learning, big data and artificial general intelligence (2011-present)

From 2011 to the present moment, significant advancements have unfolded within the artificial intelligence (AI) domain. These achievements can be attributed to the amalgamation of deep learning, extensive data application, and the ongoing quest for artificial general intelligence (AGI). Here are notable occurrences from this timeframe:

- **Year 2011:** In 2011, IBM's Watson won Jeopardy, a quiz show where it had to solve complex questions as well as riddles. Watson had proved that it could understand natural language and can solve tricky questions quickly.
- **Year 2012:** Google launched an Android app feature, "Google Now", which was able to provide information to the user as a prediction. Further, Geoffrey Hinton, Ilya Sutskever, and Alex Krizhevsky presented a deep CNN structure that emerged victorious in the ImageNet challenge, sparking the proliferation of research and application in the field of deep learning.
- **Year 2013:** China's Tianhe-2 system achieved a remarkable feat by doubling the speed of the world's leading supercomputers to reach 33.86 petaflops. It retained its status as the world's fastest system for the third consecutive time. Furthermore, DeepMind unveiled deep reinforcement learning, a CNN that acquired skills through repetitive learning and rewards, ultimately surpassing human experts in playing games. Also, Google researcher Tomas Mikolov and his team introduced Word2vec, a tool designed to automatically discern the semantic connections among words.
- **Year 2014:** In the year 2014, Chatbot "Eugene Goostman" won a competition in the infamous "Turing test." Whereas Ian Goodfellow and his team pioneered generative adversarial networks (GANs), a type of machine learning framework employed for producing images, altering pictures, and crafting deepfakes, and Diederik Kingma and Max Welling introduced variational autoencoders (VAEs) for generating images, videos, and text. Also, Facebook engineered the DeepFace deep learning facial recognition

system, capable of identifying human faces in digital images with accuracy nearly comparable to human capabilities.

- **Year 2016:** DeepMind's AlphaGo secured victory over the esteemed Go player Lee Sedol in Seoul, South Korea, prompting reminiscence of the Kasparov chess match against Deep Blue nearly two decades earlier.Whereas Uber initiated a pilot program for self-driving cars in Pittsburgh, catering to a limited group of users.
- **Year 2018:** The "Project Debater" from IBM debated on complex topics with two master debaters and also performed extremely well.
- Google has demonstrated an AI program, "Duplex," which was a virtual assistant that had taken hairdresser appointments on call, and the lady on the other side didn't notice that she was talking with the machine.
- **Year 2021:** OpenAI unveiled the Dall-E multimodal AI system, capable of producing images based on textual prompts.
- **Year 2022:** In November, OpenAI launched ChatGPT, offering a chat-oriented interface to its GPT-3.5 LLM.

Now AI has developed to a remarkable level. The concept of Deep learning, big data, and data science are now trending like a boom. Nowadays companies like Google, Facebook, IBM, and Amazon are working with AI and creating amazing devices. The future of Artificial Intelligence is inspiring and will come with high intelligence.

KEY MILESTONES

To explore the milestones in AI research that got us here.

Milestone 1: The invention of GANs (2014)

- Generative Adversarial Networks (GANs) were the key technological innovation which would eventually enable realistic-looking AI-generated images and videos.
- GANs use neural networks, a popular machine learning model based on the way neurons in the brain work.

- GANs contain two neural networks: a generator, and an adversary. In a GAN, the generator network is used to generate examples that could easily be mistaken for real data – like a realistic-looking human face.
- The adversary then works out which ones are fake, and this process repeats many times, with the generator gradually learning to make more realistic examples.

Milestone 2: Sophia the robot is activated (2016)

- Sophia the robot, the world's first non-human to be recognised with legal personhood, was activated in February 2016 (and subsequently brought to life by Gigi Goode on Drag Race in 2020).

Fig. Sophia the robot is activated

- Sophia, a humanoid robot developed by Hanson Robotics, is said to combine cutting edge technology in robotics and AI in a way that "personifies our dreams for the future of AI."
- Sophia combines vision algorithms, that process visual inputs from its camera eyes, with speech algorithms that employ natural language processing to process and produce speech, to create a human-like impression.

Milestone 3: The AI audio generator WaveNet is launched (2016)

- Creating natural-sounding computer-generated voices had long been a challenge for computer scientists.
- Most previous attempts relied on cutting up raw voice recordings and mashing them back together, a laborious process that produced robotic and artificial-sounding results. In

2016, this all changed when Google DeepMind launched WaveNet. WaveNet is an AI voice-generator which uses a 'generator' algorithm, similar to that found in a GAN.

- The generator algorithm is trained on an example dataset and can then produce *new*, similar-sounding examples which weren't part of the training data.
- AI-generated voices now have a range of good, bad and ugly real-world applications, from helping those with neurological diseases regain a voice, to being used by criminals to simulate family members in money-extracting scams.

Milestone 4: AlphaGo beats the world's Go champion (2016)

- The ancient boardgame Go was developed in China 3000 years ago, and is so ridiculously complex that the amount of possible moves is a googol greater than in chess.

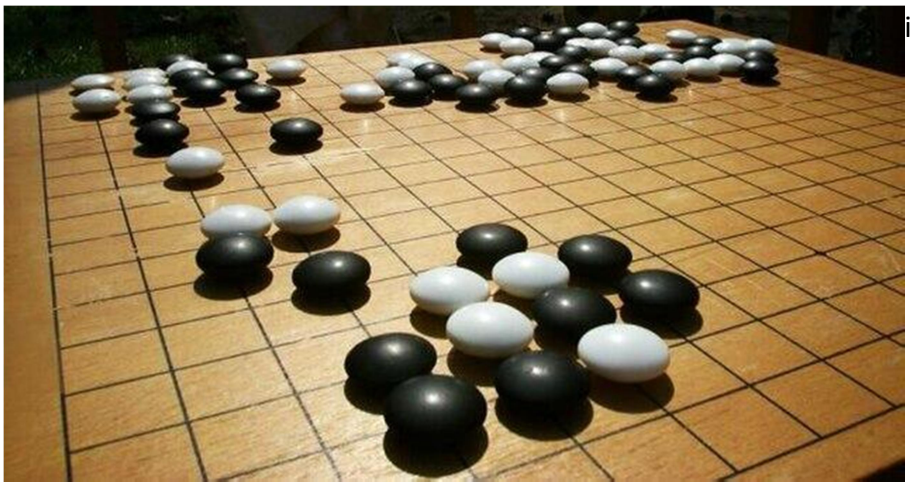


Fig. AlphaGo beats the world's Go champion

- For reference: a googol is a number greater than there are atoms in the universe. And yes, it's also the root of the differently-spelled name of your favourite internet search engine.
- Developing computer programs that can beat humans at logical games, a benchmark for increasingly capable algorithms, had been a goal for AI researchers since a computer first mastered noughts and crosses in 1952. But in 2016, DeepMind's AlphaGo beat the human world champion at Go for the first time.

Milestone 5: The birth of deepfakes (2017)

- The term 'deepfakes' was coined when the Reddit user 'deepfakes' began posting hyper-realistic AI videos online – mostly involving pornographic videos with celebrities' faces super-imposed onto actresses without their consent.

- Computer-generated special effects were nothing new, as anyone else whose childhood was haunted by Harry Potter's dementors will attest to.
- But deepfake videos, with a leg up from GAN technology, allow anyone to easily produce convincingly real videos, and they're only getting better.

Milestone 6: The first 'Transformer' lays the technological foundation for large language models (LLMs) (2017)

- Scientists at Google had been working on a new way to program Google Translate. In 2017, the scientists published the seminal paper 'Attention is all you need', in which they introduced the first Transformer, providing a step-change to machine translation.
- Instead of individually translating each word, Transformers read whole sentences at once, capturing the dependencies between words and extracting meaning based on the context.
- The way Transformers extract and generate meaning from patterns would become central to the technology used in subsequent AI breakthroughs like AlphaFold and large language models.

Milestone 7: OpenAI releases GPT-1 (2018)

- The first Generative Pre-trained Transformer was released by OpenAI in 2018. Employing a Transformer architecture, the large language model GPT-1 was able to answer questions and generate blocks of text.
- It gained these abilities after being trained using two large datasets: one with around 8 million web pages, and one with over 11,000 books.
- Although this language processor was fluent and accurate on an unprecedented scale, it was unable to coherently generate longer blocks of text and was prone to repetition.

Milestone 8: AlphaFold wins protein folding contest (2020)

- After the massive advances for AI in winning games like Go, it was time for a task with real-world implications.
- Predicting protein folding had been the holy grail of biology for 50 years, since these molecular structures determine most biological processes.
- However, characterising specific protein structures traditionally required years of excruciating laboratory tests. In 2020, Google

- DeepMind released the AI algorithm AlphaFold, which, after being trained on a public database of 170,000 protein sequences, reached an accuracy comparable to the lab work at predicting protein structure.
- AlphaFold has revolutionised biological research and is already contributing to novel drug design. And at its algorithmic heart is – you guessed it – a Transformer.

Milestone 9: Generative AI goes mainstream (2022)

- OpenAI introduced ChatGPT to the public in 2022, launching a free preview of GPT-3.5. Just one week after its release, the chatbot interface had surpassed one million users - soon becoming the fastest-growing consumer application in history.

 i

Fig. Generative AI goes mainstream

- This was welcome news to OpenAI, who were using the users' data to improve their product. But not everyone was so enchanted by the new large language model.
- It was blocked in several countries, including China, Iran and Italy. And a legal case was brought against OpenAI with concerns about the use of unauthorised data from artists and writers to train their original model.
- ChatGPT was not the only generative AI to take public interest by storm. In the same year, DALL-E 2, a text-to-image generative AI model, and GitHub Copilot, a code writing assistant, among others, were released to similarly sceptical but enthusiastic receptions.

Milestone 10: The release of Chat GPT-4 (2023)

- The newest GPT to date was launched in March 2023. Although GPT-4 retained many of its predecessors' flaws, it had some key advancements – such as the abilities to take in videos and images, rather than just text, as input prompts, as well as to access the internet in real time.
- Although OpenAI were forerunners in releasing these models to the public, they are not the only companies developing such models: others like DeepMind and Hugging Face are also vying to get ahead in the race to super-human artificial intelligence.

NEURAL NETWORK

Neural Networks are computational models that mimic the complex functions of the human brain. Neural networks extract identifying features from data, lacking pre-programmed understanding. Network components include neurons, connections, weights, biases, propagation functions, and a learning rule. Neurons receive inputs, governed by thresholds and activation functions. Connections involve weights and biases regulating information transfer. Learning, adjusting weights and biases, occurs in three stages: input computation, output generation, and iterative refinement enhancing the network's proficiency in diverse tasks.

These include:

1. The neural network is simulated by a new environment.
2. Then the free parameters of the neural network are changed as a result of this simulation.
3. The neural network then responds in a new way to the environment because of the changes in its free parameters.

Evolution of Neural Networks

Since the 1940s, there have been a number of noteworthy advancements in the field of neural networks:

- **1940s-1950s: Early Concepts**: Neural networks began with the introduction of the first mathematical model of artificial neurons by McCulloch and Pitts. But computational constraints made progress difficult.

- **1960s-1970s: Perceptrons:** This era is defined by the work of Rosenblatt on perceptrons. Perceptrons are single-layer networks whose applicability was limited to issues that could be solved linearly separately.
- **1980s: Backpropagation and Connectionism:** Multi-layer network training was made possible by Rumelhart, Hinton, and Williams' invention of the backpropagation method. With its emphasis on learning through interconnected nodes, connectionism gained appeal.
- **1990s: Boom and Winter:** With applications in image identification, finance, and other fields, neural networks saw a boom. Neural network research did, however, experience a "winter" due to exorbitant computational costs and inflated expectations.
- **2000s: Resurgence and Deep Learning:** Larger datasets, innovative structures, and enhanced processing capability spurred a comeback. Deep learning has shown amazing effectiveness in a number of disciplines by utilizing numerous layers.
- **2010s-Present: Deep Learning Dominance:** Convolutional neural networks (CNNs) and recurrent neural networks (RNNs), two deep learning architectures, dominated machine learning. Their power was demonstrated by innovations in gaming, picture recognition, and natural language processing.

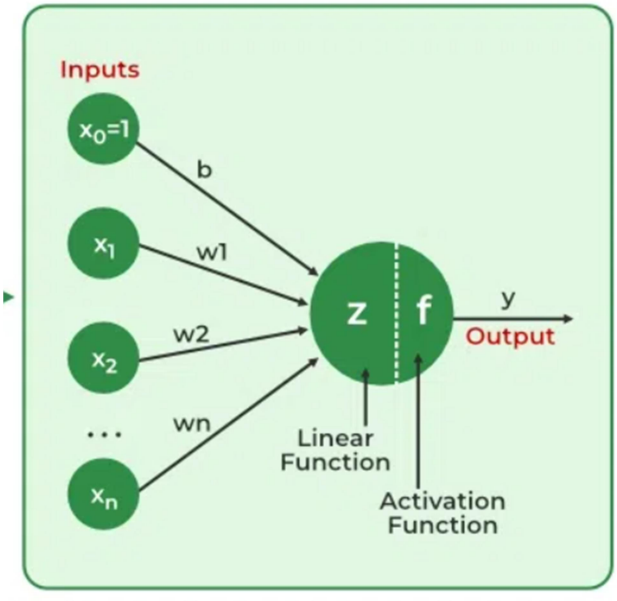


Fig. Neural Networks

Importance of Neural Networks

The ability of neural networks to identify patterns, solve intricate puzzles, and adjust to changing surroundings is essential. Their capacity to learn from data has far-reaching effects, ranging from revolutionizing technology like natural language processing and self-driving automobiles to automating decision-making processes and increasing efficiency in numerous industries. The development of artificial intelligence is largely dependent on neural networks, which also drive innovation and influence the direction of technology.

Consider a neural network for email classification.

- **The input layer** takes features like email content, sender information, and subject. These inputs, multiplied by adjusted weights, pass through hidden layers. The network, through training, learns to recognize patterns indicating whether an email is spam or not.
- The **output layer,** with a binary activation function, predicts whether the email is spam (1) or not (0). As the network iteratively refines its weights through backpropagation, it becomes adept at distinguishing between spam and legitimate emails, showcasing the practicality of neural networks in real-world applications like email filtering.

Working of a Neural Network

Neural networks are complex systems that mimic some features of the functioning of the human brain. It is composed of an input layer, one or more hidden layers, and an output layer made up of layers of artificial neurons that are coupled. The two stages of the basic process are called backpropagation and forward propagation.

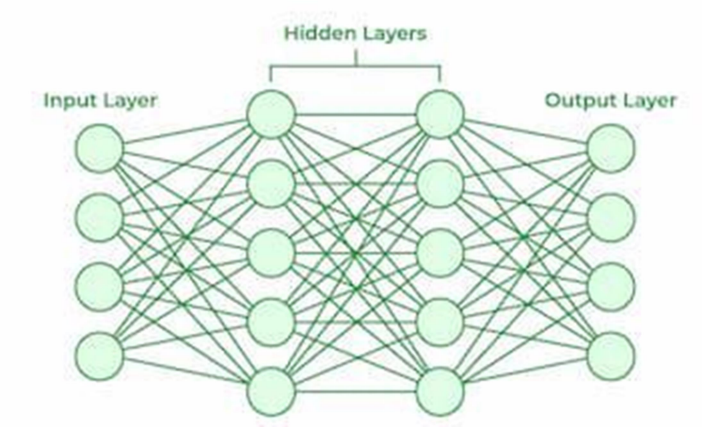


Fig. Working of Neural Network

Forward Propagation

- **Input Layer:** Each feature in the input layer is represented by a node on the network, which receives input data.
- **Weights and Connections:** The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed.
- **Hidden Layers:** Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.
- **Output:** The final result is produced by repeating the process until the output layer is reached.

Backpropagation

- **Loss Calculation:** The network's output is evaluated against the real goal values, and a loss function is used to compute the difference. For a regression problem, the Mean Squared Error (MSE) is commonly used as the cost function.
- **Loss Function:** $MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2$
- **Gradient Descent:** Gradient descent is then used by the network to reduce the loss. To lower the inaccuracy, weights are changed based on the derivative of the loss with respect to each weight.
- **Adjusting weights:** The weights are adjusted at each connection by applying this iterative process, or backpropagation, backward across the network.
- **Training:** During training with different data samples, the entire process of forward propagation, loss calculation, and backpropagation is done iteratively, enabling the network to adapt and learn patterns from the data.
- **Actvation Functions:** Model non-linearity is introduced by activation functions like the rectified linear unit (ReLU) or sigmoid. Their decision on whether to "fire" a neuron is based on the whole weighted input.

Learning of a Neural Network

1. Learning with supervised learning

In supervised learning, the neural network is guided by a teacher who has access to both input-output pairs. The network creates outputs based on inputs without taking into account the surroundings. By comparing these outputs to the teacher-known desired outputs, an error signal is generated. In order to reduce errors, the network's parameters are changed iteratively and stop when performance is at an acceptable level.

2. Learning with Unsupervised learning

Equivalent output variables are absent in unsupervised learning. Its main goal is to comprehend incoming data's (X) underlying structure. No instructor is present to offer advice. Modeling data patterns and relationships is the intended outcome instead. Words like regression and classification are related to supervised learning, whereas unsupervised learning is associated with clustering and association.

3. Learning with Reinforcement Learning

Through interaction with the environment and feedback in the form of rewards or penalties, the network gains knowledge. Finding a policy or strategy that optimizes cumulative rewards over time is the goal for the network. This kind is frequently utilized in gaming and decision-making applications.

Types of Neural Networks

There are *seven* types of neural networks that can be used.

- **Feedforward Neteworks:** A feedforward neural network is a simple artificial neural network architecture in which data moves from input to output in a single direction. It has input, hidden, and output layers; feedback loops are absent. Its straightforward architecture makes it appropriate for a number of applications, such as regression and pattern recognition.
- **Multilayer Perceptron (MLP):** MLP is a type of feedforward neural network with three or more layers, including an input layer, one or more hidden layers, and an output layer. It uses nonlinear activation functions.

- **Convolutional Neural Network (CNN):** A Convolutional Neural Network (CNN) is a specialized artificial neural network designed for image processing. It employs convolutional layers to automatically learn hierarchical features from input images, enabling effective image recognition and classification. CNNs have revolutionized computer vision and are pivotal in tasks like object detection and image analysis.
- **Recurrent Neural Network (RNN):** An artificial neural network type intended for sequential data processing is called a Recurrent Neural Network (RNN). It is appropriate for applications where contextual dependencies are critical, such as time series prediction and natural language processing, since it makes use of feedback loops, which enable information to survive within the network.
- **Long Short-Term Memory (LSTM):** LSTM is a type of RNN that is designed to overcome the vanishing gradient problem in training RNNs. It uses memory cells and gates to selectively read, write, and erase information.

Advantages of Neural Networks

Neural networks are widely used in many different applications because of their many benefits:

- **Adaptability:** Neural networks are useful for activities where the link between inputs and outputs is complex or not well defined because they can adapt to new situations and learn from data.
- **Pattern Recognition:** Their proficiency in pattern recognition renders them efficacious in tasks like as audio and image identification, natural language processing, and other intricate data patterns.
- **Parallel Processing:** Because neural networks are capable of parallel processing by nature, they can process numerous jobs at once, which speeds up and improves the efficiency of computations.
- **Non-Linearity:** Neural networks are able to model and comprehend complicated relationships in data by virtue of the non-linear activation functions found in neurons, which overcome the drawbacks of linear models.

Disadvantages of Neural Networks

Neural networks, while powerful, are not without drawbacks and difficulties:

- **Computational Intensity:** Large neural network training can be a laborious and computationally demanding process that demands a lot of computing power.
- **Black box Nature:** As "black box" models, neural networks pose a problem in important applications since it is difficult to understand how they make decisions.
- **Overfitting:** Overfitting is a phenomenon in which neural networks commit training material to memory rather than identifying patterns in the data. Although regularization approaches help to alleviate this, the problem still exists.
- **Need for Large datasets:** For efficient training, neural networks frequently need sizable, labeled datasets; otherwise, their performance may suffer from incomplete or skewed data.

GENERATIVE MODELS VS AUTOREGRESSIVE MODELS

Generative AI Model

Generative AI models are a type of machine learning model that generates new data similar to the data on which it was trained. These models are called generative because they create something new; for example, images, text, video and audio.

In the realm of artificial intelligence, generative models play a pivotal role in teaching computers to understand the real world and generate novel content that resembles the original data. They are used in unsupervised machine learning (more on this later) to discover underlying patterns and structure in unlabeled training data.

The use of data-driven machine learning (ML) has enabled the possibility of a new paradigm in research and development. ML has proven an effective tool for uncovering the structure-activity relationships in material data. However, the paradigm shift faces challenges due to slow progress in data quality governance and the need for guidance on combining domain knowledge with data-driven analysis. These challenges are three key issues: high dimensionality of feature space vs. small sample, model accuracy vs. usability, and ML results vs. domain knowledge. The key to resolving the above-mentioned issues and enabling accurate

mining of structure-activity relationships lies in embedding domain knowledge into models with generative ability.

The model learns the patterns in the data, adjusting its parameters to match the distribution of the training data. Following the training period, the model can generate new data that resembles the original data.

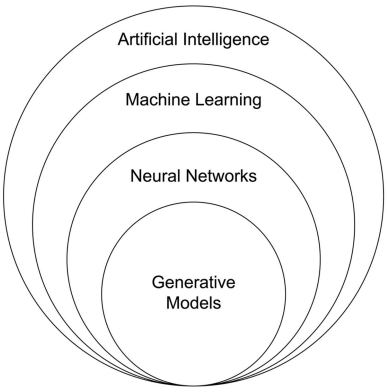


Fig. AI Generative Model

Generative AI models function by scrutinizing patterns and information within extensive datasets, employing this understanding to create fresh content. This process encompasses various stages.

1. Data gathering

When training a generative AI model, the first step is clearly defining the objective. The objective should specify the kind of content that the model is expected to generate. A clear goal, whether images, text, or music, is crucial. The developer can tailor the training process by defining the objective to ensure the model produces the desired output.

2. Preprocessing

To create a quality Generative AI model, collect a diverse dataset that aligns with the objective. Ensure the data is preprocessed and cleaned to remove noise and errors before feeding it into the model.

3. Choose the Right Model Architecture

Choosing the exemplary model architecture is a crucial step in ensuring the success of your generative AI project. Various architectures exist, such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Transformers. Each architecture has unique

advantages and limitations, so it is essential to carefully evaluate the objective and dataset before selecting the appropriate one.

4. Implement the Model

There is a need to create the neural network, define the layers, and establish the connections between them by writing code to implement the chosen model architecture; frameworks and libraries like TensorFlow and PyTorch offer prebuilt components and resources to simplify the implementation process.

5. Train the Model

Train a generative AI model involves sequentially introducing the training data to the model and refining its parameters to reduce the difference between the generated output and the intended result. This training process requires considerable computational resources and time, depending on the model's complexity and the dataset's size. Monitoring the model's progress and adjusting its training parameters, like learning rate and batch size, is crucial to achieving the best results.

6. Evaluate and Optimize

After training a model, it is crucial to assess its performance. This can be done by using appropriate metrics to measure the quality of the generated content and comparing it to the desired output. If the results are unsatisfactory, adjusting the model's architecture, training parameters, or dataset could be necessary to optimize its performance.

7. Fine-tune and Iterate

Developing a generative AI model is a process that requires continuous iteration and improvement. Once the initial results are evaluated, areas for improvement can be identified. By incorporating feedback from users, introducing fresh training data, and refining the training process, it is possible to enhance the model and optimize the results. Therefore, consistent improvements are crucial in developing a high-quality generative AI model.

Types of Generative AI models

Generative AI or foundation models are designed to generate different types of content, such as text and chat, images, code, video, and embeddings. Researchers can modify these models

to fit specific domains and tackle tasks by adjusting the generative AI's learning algorithms or model structures.

This section provides an overview of task-specific and general generative ai research, with a focus on the applications of generative ai models in materials research.

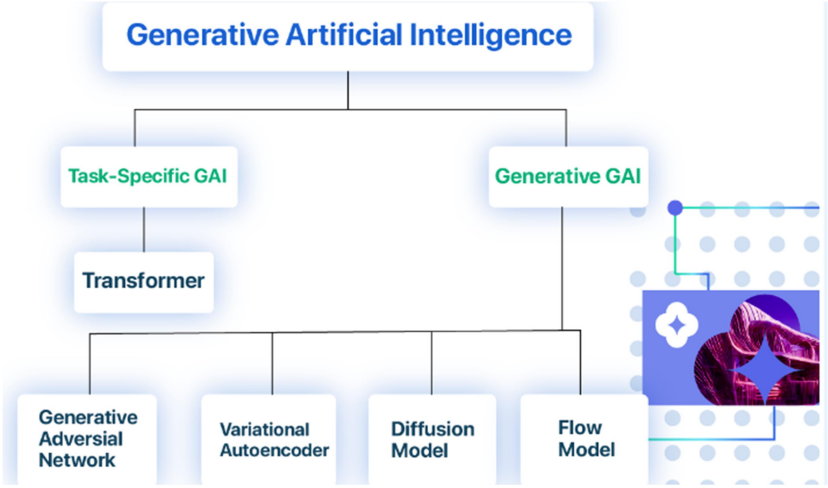


Fig. Types of Generative AI Models

Generative Adversarial Networks (GANs*)*

- GAN, which stands for Generative Adversarial Network, is an advanced deep learning architecture consisting of two essential components: a generator and a discriminator.
- The generator's primary function is to generate synthetic data that closely resembles real data, while the discriminator is responsible for distinguishing between authentic and fabricated data.
- The generator enhances the authenticity of its produced data through adversarial training, while the discriminator effectively determines whether the data is real or synthetic.
- Functioning as a generative model, GAN is commonly employed in deep learning to generate samples to enhance data augmentation and pre-processing techniques.
- Its broad application extends across various fields, such as image processing and biomedicine, where it proves valuable in producing high-quality synthetic data for research and analysis.

2. Diffusion model

- Generative diffusion models can create new data using the data they were trained on. For instance, when trained on an assortment of human faces, a diffusion model can create new and lifelike faces with diverse features and expressions, even if they were not present in the original dataset.
- The fundamental idea behind diffusion models is to transform a simple and easily obtainable distribution into a more complex and meaningful data distribution. This transformation is accomplished through a series of reversible operations.
- Once the model understands the process of transformation, it can generate new samples by starting from a point within the simple distribution and gradually spreading it towards the desired complex data distribution.

Variational Autoencoders (VAEs)

- **VAEs are generative models that combine the capabilities of autoencoders and probabilistic modeling to acquire a compressed representation of data**.
- VAEs encode input data into a lower-dimensional latent space, allowing the generation of new samples by sampling points from the acquired distribution.
- With practical applications spanning image generation, data compression, anomaly detection, and drug discovery, VAEs exhibit versatility across various domains.

Flow model

- **Flow-based models are generative ai model that aims to learn the underlying structure of a given dataset**. These models achieve this by understanding the probability distribution of the different values or events within the dataset.
- Once the model has acquired this probability distribution, it is capable of generating fresh data points that maintain identical statistical properties and characteristics to those of the initial dataset.
- A key feature of flow-based models is that they apply a simple invertible transformation to the input data that can be easily reversed. By starting from a simple initial distribution, such as random noise, and applying the transformation in reverse, the

model can quickly generate new samples without requiring complex optimization. This makes flow-based models computationally efficient and faster than other models.

General GAI (Generative AI)

The development of big data and data representation technologies has enabled the generation of human-readable language from input data patterns and structures, allowing us to achieve objectives across various environments. This goal seeks to go beyond the language generation paradigm, restricted to adapting sample distributions for tasks.

1. The Generative Pre-Trained Transformer (GPT)

Initially showcased its potential for generating task-specific natural language through unsupervised pre-training and fine-tuning for downstream tasks. It utilizes transformer-decoder layers for next-word prediction and coherent text generation. Fine-tuning is used to adapt it to a specific task based on pre-training.

2. GPT-2

It expands on its predecessor's model structure and parameters and trains on various datasets beyond just web text. Despite exhibiting advanced results with zero-shot learning, it still falls under task-specific GAI.

3. GPT-3

It is a language model that employs Prompt to reduce the dependence on large, supervised datasets. It uses the linguistic structure of text probability to make predictions. The model is pre-trained on a vast amount of text, allowing it to perform few-shot or zero-shot learning. By defining a new cue template, it swiftly adjusts to new scenarios, even in situations where there is limited or no labeled data. This methodology proves advantageous for tasks that involve language comprehension and generation, as it minimizes the amount of data needed and enhances overall performance.

Recently, GPT-4, the latest model developed by OpenAI, was trained with an unprecedented scale of computations and data, surprisingly achieved human-like performance across almost all tasks, and significantly outperformed its predecessors. The introduction of GPT-4 represents a significant leap forward in the field of General Artificial Intelligence (GAI).

Building upon the success of the previous GPT models, GPT-4 showcases remarkable advancements in its ability to perceive and generate multimodal data, including text, images, and audio. This groundbreaking development holds great promise for the field of materials science research.. The formidable capabilities of GPT-4 in multimodal generation and conversational interactivity offer a promising outlook for materials science research.

4. LLaMA

Meta, formerly known as Facebook, has recently announced a new LLM in 2023. The LLM is called LLaMA, which stands for Large Language Model for Meta Applications, and comes with 600 billion parameters. LLaMA has been trained on various data sources, including social media posts, web pages, books, news articles, and more. Its purpose is to support various Meta applications such as content moderation, search, recommendation, and personalization. LLaMA claims to be more ethical by incorporating human feedback, fairness, and transparency in its training.

5. PaLM 2

The PaLM model has a new version called PaLM 2, which will be released in 2023 with 400 billion parameters. It is a multimodal LLM that can process and generate text and images. It has been trained with a large-scale dataset that covers 100 languages and 40 visual domains, making it capable of performing cross-modal tasks such as image captioning, visual question answering, text-to-image synthesis, and more. Palm 2 generalizes to new tasks and domains without fine-tuning, thanks to its zero-shot learning capability.

6. BLOOM

BLOOM generates text in 46 natural languages, dialects, and 13 programming languages. It has been trained on enormous data, totaling 1.6 terabytes, equivalent to 320 copies of Shakespeare's works. The model has the capability to process a total of 46 languages, which encompass French, Vietnamese, Mandarin, Indonesian, Catalan, 13 Indic languages (including Hindi), as well as 20 African languages. Although just over 30% of the training data was in English, the system is proficient in all mentioned languages.

7. BERT from Google

One of Google's most influential LLMs released in 2018 is BERT. **BERT is an abbreviation for Bidirectional Encoder Representations from Transformers, which contains 340 million parameter**s. BERT, which is constructed based on the transformer framework, leverages bidirectional self-attention to acquire knowledge from extensive volumes of textual data. With its capabilities, BERT is proficient in executing diverse natural language tasks such as text classification, sentiment analysis, and named entity recognition. Additionally, BERT is widely used as a pre-trained model for fine-tuning specific downstream tasks and domains.

Generative AI Adversarial Networks

1. DALL·E 2

It is an AI system that can take a simple description in natural language and turn it into a realistic image or work of art.

2. StyleGAN 3

An AI system can generate photorealistic images of anything the user can imagine, from human faces to animals and cars. Furthermore, it provides a remarkable degree of personalization by allowing users to manipulate the generated images' style, shape, and pose.

Diffusion Models

1. Stable Diffusion

It is a generative AI model that creates photorealistic images, videos, and animations from text and image prompts. It uses diffusion technology and latent space, which reduces processing requirements and allows it to run on desktops or laptops with GPUs. With transfer learning, developers can fine-tune the model with just five images to meet their needs. It was launched in 2022.

2. DALL-E 2

An innovative language model developed by OpenAI showcases its extraordinary talent for converting textual descriptions into breathtaking images using an advanced diffusion model. The model uses contrastive learning to recognize the differences between similar images and create new ones. It has practical applications in design, advertising, and content creation, making it a groundbreaking example of human-centered AI.

Application of Generative AI

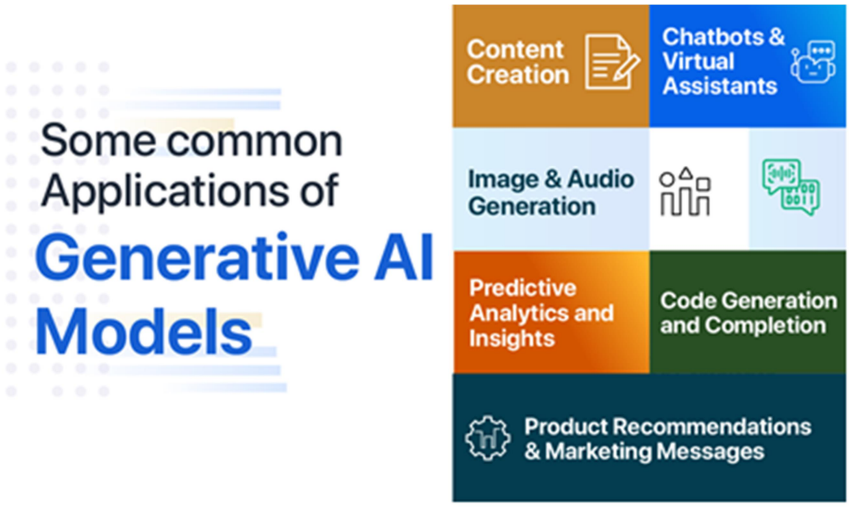


Fig. Application of Generative AI

Evaluation & Monitoring Metrics for Generative AI

Language models such as OpenAI GPT-4 and Llama 2 can cause harmful outcomes if not designed carefully. The evaluation stage helps identify and measure potential harms by establishing clear metrics and completing iterative testing. Mitigation steps, such as prompt engineering and content filters, can then be taken. AI-assisted metrics can be helpful in scenarios without ground truth data, helping to measure the quality and safety of the answer. Below are metrics that help to evaluate the results generated by generative models:

1. The **groundedness** metric assesses how well an AI model's generated answers align with user-defined context. It ensures that claims made in an AI-generated answer are substantiated by the source context, making it essential for applications where factual correctness and contextual accuracy are critical. The input required for this metric includes the question, context, and generated answer, and the score range is Integer [1-5], where one is bad, and five is good.
2. The **relevance** metric is crucial for evaluating an AI system's ability to generate appropriate responses. It measures how well the model's responses relate to the given questions. A high relevance score signifies the AI system's comprehension of the input and its ability to generate coherent and suitable outputs. Conversely, low relevance

scores indicate that the generated responses may deviate from the topic, lack context, or be inadequate.

3. **Coherence** is a metric that measures the ability of a language model to generate output that flows smoothly, reads naturally, and resembles human-like language. It assesses the readability and user-friendliness of the model's generated responses in real-world applications. The input required to calculate this metric is a question and its corresponding generated answer.
4. The **Fluency** score gauges how effectively an AI-generated text conforms to proper grammar, syntax, and the appropriate use of vocabulary. It is an integer score ranging from 1 to 5, with one indicating poor and five indicating good. This metric helps evaluate the linguistic correctness of the AI-generated response. It requires the question and the generated answer as input.
5. The **Similarity** metric rates the similarity between a ground truth sentence and the AI model's generated response on a scale of 1-5. It objectively assesses the performance of AI models in text generation tasks by creating sentence-level embeddings. This metric helps compare the generated text with the desired content. To use the GPT-Similarity metric, input the question, ground truth answer, and generated answer.

Real-World Use Cases of Generative Models

Generative models have penetrated mainstream consumption, revolutionizing the way we interact with technology and experience content, for example:

- **Art creation.** Artists and musicians are using generative models to create new pieces of art or compositions, based on styles they feed into the model. For example, **Midjourney** is a very popular tool that is used to generate artwork.
- **Drug discovery.** Scientists can use generative models to predict molecular structures for new potential drugs.
- **Content creation.** Website owners leverage generative models to speed up the content creation process. For example, **Hubspot's AI content writer** helps marketers generate blog posts, landing page copy and social media posts.

- **Video games.** Game designers use generative models to create diverse and unpredictable game environments or characters.

Benefits of Generative Models

Generative models, with their unique ability to create and innovate, offer a plethora of advantages that extend beyond mere data generation. Here's a deeper dive into the myriad benefits they bring to the table:

- **Data augmentation.** In domains where data is scarce or expensive to obtain, generative models can produce additional data to supplement the original set. For instance, in medical imaging, where obtaining large datasets can be challenging, these models can generate more images to aid in better training of diagnostic tools.
- **Anomaly detection.** By gaining a deep understanding of what constitutes "normal" data, generative models can efficiently identify anomalies or outliers. This is particularly useful in sectors like finance, where spotting fraudulent transactions quickly is paramount.
- **Flexibility.** Generative models are versatile and can be employed in a range of learning scenarios, including **unsupervised**, semi-supervised, and **supervised learning**. This adaptability makes them suitable for a wide array of tasks.
- **Personalization.** These models can be tailored to generate content based on specific user preferences or inputs. For example, in the entertainment industry, generative models can create personalized music playlists or movie recommendations, enhancing user experience.
- **Innovation in design.** In fields like architecture or product design, generative models can propose novel designs or structures, pushing the boundaries of creativity and innovation.
- **Cost efficiency.** By automating the creation of content or solutions, generative models can reduce the costs associated with manual production or research, leading to more efficient processes in industries like manufacturing or entertainment.

Limitations of Generative Models

While generative models are undeniably powerful and transformative, they are not without their challenges. Here's an exploration of some of the constraints and challenges associated with these models:

- **Training complexity.** Generative models, especially sophisticated ones like GANs, require significant computational resources and time. Training them demands powerful hardware and can be resource-intensive.
- **Quality control.** While they can produce vast amounts of data, ensuring the quality and realism of the generated content can be challenging. For instance, a model might generate an image that looks realistic at first glance but has subtle anomalies upon closer inspection.
- **Overfitting.** There's a risk that generative models can become too attuned to the training data, producing outputs that lack diversity or are too closely tied to the input they've seen.
- **Lack of interpretability.** Many generative models, particularly deep learning-based ones, are often seen as "black boxes." This means it can be challenging to understand how they make decisions or why they produce specific outputs, which can be a concern in critical applications like healthcare.
- **Ethical concerns.** The ability of generative models to produce realistic content raises ethical issues, especially in the creation of deep fakes or counterfeit content. Ensuring responsible use is paramount to prevent misuse or deception.
- **Data dependency.** The quality of the generated output is heavily dependent on the quality of the training data. If the training data is biased or unrepresentative, the model's outputs will reflect those biases.
- **Mode collapse.** Particularly in GANs, there's a phenomenon called mode collapse where the generator produces limited varieties of samples, reducing the diversity of the generated outputs.

Comparing Generative Models

As seen above, there are a variety of generative models, and we only touched upon some of the major ones in use today. Let's summarize how the models we highlighted compare:

- GANs generate highly realistic images, but they can be unstable and difficult to train.
- VAEs are easier to train than GANs and great for probabilistic data representation, but can produce lower quality results.
- Autoregressive models are good for predicting the likelihood of time-series events, but can be expensive to train, particularly for long sequences.
- Flow-based models produce high-quality image generation and are computationally efficient, but struggle with long-range dependencies in data.
- Transformer-based models excel in natural language processing tasks and complex sequence generation, but are expensive to train.

With each generative model displaying their own sets of strengths and challenges, organizations will benefit from carefully considering the ones best suited for their needs.

Autoregressive Models

Autoregressive models are a class of generative models that generate data by modeling the conditional distribution of each data point given the previous ones. This approach breaks down the joint probability distribution of the data into a product of conditional probabilities. One of the most well-known autoregressive models is the PixelCNN, which generates images pixel by pixel.

Mechanism

In an autoregressive model, the probability of observing a sequence is decomposed as follows:

This decomposition allows the model to generate each data point sequentially, conditioned on the previously generated data points. The model parameters are typically learned using maximum likelihood estimation, which involves minimizing the negative log-likelihood of the observed data.

Advantages

Exact Likelihood Computation: Autoregressive models allow for exact computation of the likelihood, which facilitates robust training and evaluation.

High-Quality Samples: These models can generate high-quality samples, especially in domains such as natural language processing and image generation.

Flexibility: They can be applied to various types of data, including sequences, images, and audio.

Limitations

Sequential Generation: The sequential nature of generation can be slow, especially for high-dimensional data like images.

Computationally Intensive: Training and sampling from autoregressive models can be computationally expensive.

Limited Parallelism: The sequential dependency limits the ability to parallelize the generation process.

Autoregressive model

An autoregressive (AR) model is a type of statistical model that uses past values of a time series to predict future values. It is based on the assumption that the current value of the time series depends on its past values, with the relationship between the current and past values described by a set of coefficients.

Auto-Regressive Models in Generative AI

Auto-regressive models are a class of statistical models used for predicting future values of a time series based on its past values. In the context of generative AI, auto-regressive models are employed to generate new data points that follow the same distribution as the training data. These models have gained popularity in various applications, such as natural language processing, image synthesis, and time series forecasting.

Overview

Auto-regressive models assume that the value of a variable at a given time step is a linear combination of its past values. This assumption allows the model to learn the underlying structure of the data and generate new data points that follow the same distribution. In generative AI, auto-regressive models are used to generate sequences of data points, such as text, images, or time series, by predicting one data point at a time, conditioning on the previously generated data points.

Auto-Regressive Process

An auto-regressive process is a stochastic process where the value of a variable at a given time step is a linear combination of its past values, plus some noise. The order of an auto-regressive process, denoted as AR(p), indicates the number of past values considered in the linear combination. For example, an AR(1) process considers only the immediately preceding value, while an AR(2) process considers the two previous values.

Conditional Probability

In auto-regressive models, the generation of a new data point is based on the conditional probability of the data point given the previously generated data points. This conditional probability is learned from the training data and used to sample new data points during the generation process.

Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is a statistical method used to estimate the parameters of an auto-regressive model. MLE aims to find the parameter values that maximize the likelihood of the observed data, given the model. In the context of auto-regressive models, MLE is used to learn the coefficients of the linear combination and the noise term.

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are a type of generative model that use auto-regressive models in their architecture. GANs consist of two neural networks, a generator and a discriminator, that are trained simultaneously. The generator learns to generate realistic data points, while the discriminator learns to distinguish between real and generated data points. Auto-regressive models can be used as the generator in GANs to generate sequences of data points.

Applications

Auto-regressive models have been successfully applied in various generative AI tasks, including:

1. Natural Language Processing: Auto-regressive models, such as GPT (Generative Pre-trained Transformer), have been used to generate realistic text by predicting one word at a time, conditioning on the previously generated words.

2. Image Synthesis: PixelRNN and PixelCNN are examples of auto-regressive models used for generating images by predicting one pixel at a time, conditioning on the previously generated pixels.
3. Time Series Forecasting: Auto-regressive models are widely used in time series forecasting, where the goal is to predict future values of a time series based on its past values.

Limitations

Despite their success in various generative AI tasks, auto-regressive models have some limitations:

1. Sequential Generation: Auto-regressive models generate data points one at a time, which can be computationally expensive for long sequences or high-resolution images.
2. Exposure Bias: During training, auto-regressive models are exposed to the true data distribution, while during generation, they are exposed to their own generated data distribution. This discrepancy can lead to compounding errors during the generation process.

Auto-regressive models in generative AI have shown great promise in generating realistic data points across various domains. By understanding their key concepts, applications, and limitations, data scientists can effectively leverage these models for their generative tasks.

APPLICATIONS OF GENERATIVE AI

The most popular examples of generative AI

While generative AI has its drawbacks, its benefits and future potential far outweigh them. Some of the recent generative AI applications have proven how this new-age technology can help with innovation and creativity, indicating usability for both businesses and individuals.

Some of the popular examples of generative AI are the following applications:

- **ChatGPT-** One of the major drivers behind the worldwide popularity of generative AI applications, ChatGPT is a chatbot built by the Microsoft-backed AI research organization, OpenAI. This AI-powered chatbot took the world by storm, thanks to its

human-like responses, starting with OpenAI's GPT 3.5 implementation. Now, GPT-4 has been released, offering a more seamless interface with better AI capabilities for more accurate responses. ChatGPT's massive popularity earned it Microsoft's investment - a significant one - and the tech giant even incorporated GPT into its Bing browser.

- **DALL.E** - One of the first generative AI tools to be widely adopted, DALL.E is another OpenAI creation, built through GPT implementation. DALL.E is a multimodal AI application, trained on a vast amount of images and their text descriptions, that can identify connections across various media like text, audio, and vision. In this case, DALL.E connects words to visual elements, meaning it can generate images from user prompts.
- **Bard-** Google has also been an early leader in facilitating transformer AI models for various types of content, including processing language. However, Google never released a public interface for AI models, until Microsoft used GPT in Bing, which prompted Google to also launch its own chatbot, Google Bard. Bard's initial run was devastating due to overall AI platforms' erratic behavior and inaccurate responses. However, Google has since released a new Bard version, built on PaLM 2 (Google's most advanced LLM) which allows the chatbot to have higher efficiency and offer more visual responses to prompts.
- **Midjourney-** Midjourney is another prominent example of generative AI that generates images from natural language prompts. While Midjourney is one of several machine learning-based image generators to have emerged recently, it has quickly become one of the most preferred generative AI applications alongside DALL.E. One of the biggest reasons behind this is Midjourney's ability to generate high-quality images from simple text prompts, allowing lesser-experienced users to easily access excellent images for digital use.

Key generative AI applications

While chatbots like ChatGPT and Google Bard have quickly risen in popularity, there are other

generative AI use cases that are becoming prominent. Here are some of the most significant applications of generative AI that are being widely implemented today.

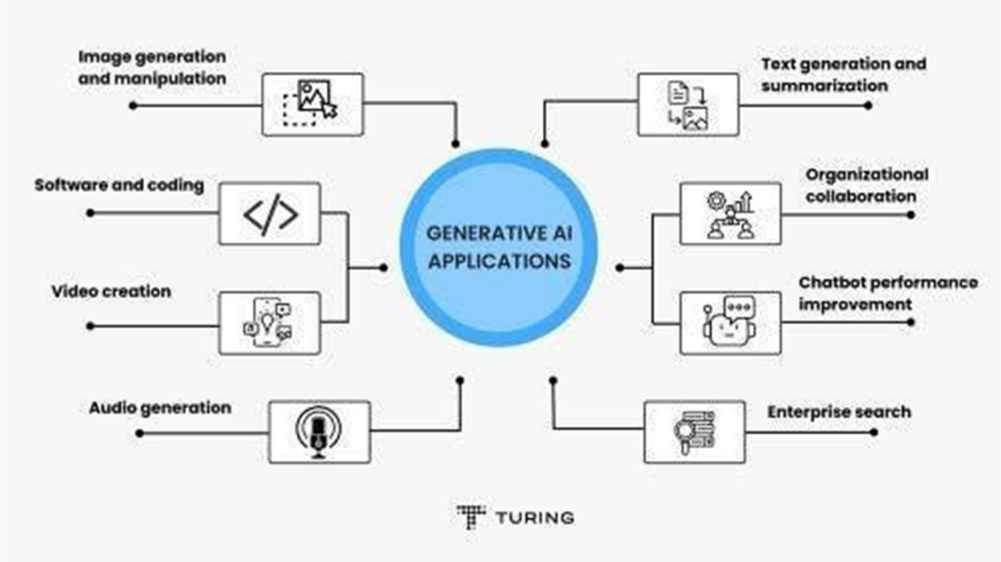


Fig: Applications

1. Image generation and manipulation

One of the most common use cases of generative AI is image generation, which is typically text- to-image conversion. Here, users can enter a textual prompt describing what type of image they want, and the AI tool will process the input to generate realistic images. When using such generative AI applications, users can specify subjects, styles, settings, locations, or objects to generate the exact images as per their requirements.

Apart from text-to-image AI applications that generate realistic images or 3D models, there are tools that facilitate image enhancement and manipulation, letting users modify existing images. Some of the major functions such tools can perform are:

- Semantic image-to-image translation - Creating realistic versions of an image based on semantic photos or sketches.
- Image completion - Generating missing portions of an image, such as filling in backgrounds with objects, people, or other elements. AI tools with this capability can also fix torn photographs or fill in missing pixels.
- Image super-resolution - Enhancing the resolution of images without any pixel-tear or other aspects that can cause loss of detail.

- Image manipulation - Altering or modifying existing images. For example, users can transform an external element of an image, like its color, lighting, form, or style, while maintaining its original elements.

2. Software and coding

Generative AI applications have already begun transforming the software development and coding landscape through innovative solutions that streamline coding. Hence, software and coding have quickly become one of the most prominent use cases of generative AI, as its applications hold the potential to improve code quality, enhance productivity, and even spark new software innovation avenues.

Here's how applications of generative AI are impacting software and coding:

- **Code generation -** One of generative AI's most prominent applications in software development is code generation. This involves training AI models on vast repositories of existing code, allowing them to generate code functions, snippets, or even entire programs based on prompted requirements. Code generation through generative AI applications proves invaluable in accelerating software development by automating repetitive coding tasks, and letting developers focus on problem-solving and higher-level design.
- **Code completion -** Generative AI can also boost coding efficiency by offering intelligent code completion and suggestions. IDEs (integrated development environments) can leverage generative AI models to predict future code lines that developers may write based on the context, expediting the coding process and reducing the possibility of error.
- **Natural language interfaces for coding -** Generative AI also enables natural language interfaces to code, allowing a developer to interact with software systems through human language instead of programming languages. Many organizations implement this through generative AI applications to bridge the gap between domain experts and developers. In turn, this helps to save resources on hiring experts to tackle software systems, by simply letting developers do it.
- **Automated testing -** Generative AI-powered tools can automate test case and

scenario generation which is generally quite time-consuming during a software development lifecycle. Such tools analyze code and its probable execution paths to generate

comprehensive test suits, thus enhancing code coverage and allowing developers to identify potential bottlenecks early on.

3. Video creation

Generative AI applications also simplify video production through highly flexible and efficient features that generate high-quality video content. Using generative AI models, applications can automate tedious tasks like video compositions, and animations, adding special effects, editing video snippets, etc. Like image generation, generative AI tools for video production can create videos from scratch, which can be used for enhancing video resolution, video manipulation, and completion.

Video generation AI tools can also perform:

- Video style transfers - AI video tools with this feature can generate new videos that follow the same style as another reference image or video.
- Video predictions - AI tools with this capability can predict the next frames in a video, using generative AI models. Such tools understand a video's spatial and temporal elements, producing future sequences based on that data.

Besides video generation, generative AI applications are also helpful for 3D shape generation, where they're used to build 3D models and shapes through generative models. AI tools achieve this through techniques like autoregressive models, GANs (generative adversarial networks), and VAEs (variational autoencoders). This is especially helpful when creating highly-detailed shapes which may not be possible when manually creating a 3D image.

4. Audio generation

Another one of the widely implemented generative AI use cases is audio generation, where generative AI is used to expedite the process of creating audio. There are three major use cases under this category, which are:

- **TTS generators -** GAN-based TTS (text-to-speech) generators can generate realistic

speech audio from a user's textual prompts. TTS AI tools use extensive text and speech data to train machine learning models, which can then be tweaked to create high-quality audio from text. Moreover, such tools are often used in applications like speech-based interfaces, speech-enabled devices, and assistive technologies.

- **Creating music -** Making music has proven to be one of the most common generative AI applications today. Generative AI models can easily produce new music pieces and generate complete audio by learning the styles and patterns of the music a user inputs.
- **STS conversions -** STS (speech-to-speech) conversions involve generative AI creating new speech or voices via existing audio files, which is commonly implemented in audio- related AI applications. STS conversions have become massively popular in the gaming and filming industries, where professionals use AI tools with STS conversion capabilities to seamlessly create voiceovers.

5. Text generation and summarization

ChatGPT is one of the best examples of text-generative AI tools that creates and summarizes textual content from user prompts. Such tools utilize generative AI models and are trained on large data sets to generate updated and authentic content. Listed below are some of the most common use cases of generative AI applications used for text generation and summarization:

- **Content creation -** Generative AI models are extremely helpful in creating various types of written content, from blogs to marketing posts and social media copies. Plus, generative AI applications like ChatGPT also speed up the writing process by generating ideas, quotes, content outlines, etc.
- **Language translation -** AI developers can also fine-tune generative AI models for translation tasks, where the models can analyze texts in one language and provide accurate translations in another.
- **Virtual assistants and chatbots -** Generative AI powers virtual assistants and chatbots, letting them generate contextually relevant and natural responses in real-time user conversations. Creating chatbots like ChatGPT has become one of the biggest

generative AI use cases. Such chatbots enhance user engagement and help businesses offer personalized assistance.

- **Content aggregation -** In addition to text creation, generative AI tools can automatically summarize bulk texts like research papers, news articles, blogs, and lengthy emails to help users get a concise overview of the content. This also includes document summarization that helps businesses streamline document-related tasks using generative AI models.
- **Automatic report generation -** In business intelligence and data analysis, generative AI can help summarize complex datasets and generate detailed reports. This simplifies decision-making and allows concerned stakeholders to better understand trends, patterns, and insights.

6. Organizational collaboration

The latest advancements in generative AI applications have also led to businesses achieving better team collaborations. Personal productivity tools like word processing and email can now be augmented via automation to boost the accuracy and efficiency of users, i.e., organization members.

An excellent example of generative AI's collaboration enhancement capabilities is Microsoft implementing GPT-3.5 in Teams Premium, which uses AI to enhance meeting recordings. It automatically divides a recording into sections, generates titles, and adds personalized markers for better reference.

Another notable example is the wildly popular startup, Jasper.ai. This generative AI-powered can be used to automate tedious writing tasks, as its powerful automation capabilities allow it to generate complete texts for various purposes, from job descriptions to marketing copies, and more.

7. Chatbot performance improvement

While chatbots are one of the most prominent generative AI applications, the technology also contributes to enhancing chatbot performance and abilities. In turn, this helps to facilitate more engaging and effective interactions between chatbots and users, which is primarily possible through generative models and NLP (natural language processing).

Here's how generative AI is currently implemented for chatbot performance improvement:

- **NLU enhancement -** Generative AI models help enhance a chatbot's natural language understanding (NLU). Training AI models on vast amounts of text data enables them to learn intricate language patterns, context, and nuances. This allows chatbots to better understand user inputs, accurately extract intent, and determine entities.
- **Human-like response generation -** One of the biggest benefits of generative AI implementation is allowing chatbots to generate human-like text. This has also become one of the most common generative AI applications, where it's used to train a chatbot on a diverse range of conversations to learn how a human expresses themself. In turn, this helps the chatbot generate natural, conversational, and tailored responses.
- **Handling open-ended prompts -** Traditional rule-based chatbots often struggle with unfamiliar topics or open-ended user queries. Generative AI empowers chatbots to better handle such user inputs, even those they aren't specifically programmed for. This elevated flexibility is achieved through training AI models on vast conversational data, enabling a chatbot to generate plausible responses to a wider range of queries.
- **User profiling -** One of the most transformative generative AI applications has been implementing this technology to facilitate chatbots creating user profiles. Using generative AI, chatbots can analyze past conversations to understand user preferences and establish a user profile based on them. This helps chatbots to tailor responses and recommendations to users, offering a highly personalized experience and better user engagement.

8. Enterprise search

Lastly, one of the most recent generative AI use cases has been the enterprise implementation of this technology for streamlined search. Using generative AI, organizations can access information faster, as such AI models can be trained to securely read through all organizational documentation, like contracts, research reports, business trend analysis, and so on. Moreover, developers can train generative AI models to automatically highlight the important sections of a document and allow enterprise members to quickly access the information they need.

Generative AI Use Case by Industries

Generative AI transforms various industries by leaps and bounds. From enhancing efficiency to driving innovation, and personalizing experiences, Gen AI improves processes across sectors such as

1. **Healthcare & Life Sciences**

   Generative AI offers multiple applications in the healthcare and life sciences sector. For example, it can accelerate drug discovery by quickly identifying new drug candidates. In pharmacovigilance, generative AI enhances adverse drug reporting by simplifying medical data analysis and providing timely and accurate insights to enhance patient safety. Additionally, generative AI can be employed for patient assistance in providing personalized support, answering queries, and offering tailored health recommendations, thereby improving patient engagement and outcomes.

   Explore more in our case study how a healthcare service provider faced **outdated data, privacy concerns, and content reliability** challenges. They partnered with Quantiphi to implement our Gen AI platform, baioniq, and achieved enhanced real-time responses, increased data security and privacy, and optimized platform performance.

2. **Banking and Financial Services**

   In banking and financial services, generative A-powered fund navigators assist advisors in quickly extracting essential details, summarizing large, complex documents, and generating content from FAQs, enabling efficient responses to client inquiries. In P&C insurance, Gen AI enhances claim processing by swiftly analyzing policy documents and preparing detailed reports on claim decisions. Additionally, Generative AI is used in customer query redressal with personalized product recommendations, automated insurance quoting, and voice analysis, enhancing decision-making, reducing risks, and boosting customer satisfaction.

3. **Auto & Manufacturing**

   In the auto & manufacturing sector, generative AI can optimize production &

operations, maintenance, supply chains, and energy usage, leading to lower costs, higher productivity, and greater sustainability. For example, Gen AI can provide real-time assistance to manufacturing technicians, improving their efficiency and decision-making on the factory floor. Additionally, Gen AI can monitor manufacturing processes and analyze sensor data using industrial IoT applications to predict potential issues before they occur, thereby reducing downtime and maintenance costs.

4. **Retail & CPG**

In retail, generative AI is used to optimize various processes by analyzing customer data to better understand customer demand, design engaging shopping experiences, and maintain a reliable supply chain. Additionally, Gen AI assists with stock-keeping unit (SKU) segmentation and optimization, enhancing product assortment, and enables accurate demand forecasting and inventory usage pattern analysis to reduce waste. Gen AI-powered personalized product recommendations boost customer engagement and drive sales.

5. **Media & Entertainment**

Generative AI applications can assist in creating new content, personalizing experiences, and optimizing production processes in the media and entertainment industry. It enhances the capabilities of creators, producers, and consumers by providing exceptional levels of creativity, personalization, and efficiency.

For example, Gen AI can generate script ideas, providing fresh concepts for writers and producers. It also enables personalized content recommendations, tailoring media experiences to individual preferences. Additionally, by automating the creation of highlight reels, generative AI streamlines the editing process and delivers engaging content quickly.

6. **Oil & Gas**

Generative AI can transform the oil and gas sector by reimagining asset management and maintenance, streamlining field operations, and exploration and production planning. For asset management, Gen AI predicts equipment failure, enabling proactive

maintenance and minimizing downtime. Moreover, Gen AI-powered assistants offer real-time support to field workers to improve decision-making and safety. Furthermore, generative AI improves production planning by simulating various scenarios, optimizing resource allocation, and ensuring risk-free exploration and production processes.

AI AGENTS

Agents in Artificial Intelligence

An AI system can be defined as the study of the rational agent and its environment. The agents sense the environment through sensors and act on their environment through actuators. An AI agent can have mental properties such as knowledge, belief, intention, etc.

What is an Agent?

An AI agent is a software program designed to interact with its environment, perceive the data it receives, and take actions based on that data to achieve specific goals. AI agents simulate intelligent behavior, and they can be as simple as rule-based systems or as complex as advanced machine learning models. They use predetermined rules or trained models to make decisions and might need external control or supervision.

An agent can be anything that perceive its environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of **perceiving**, **thinking**, and **acting**.

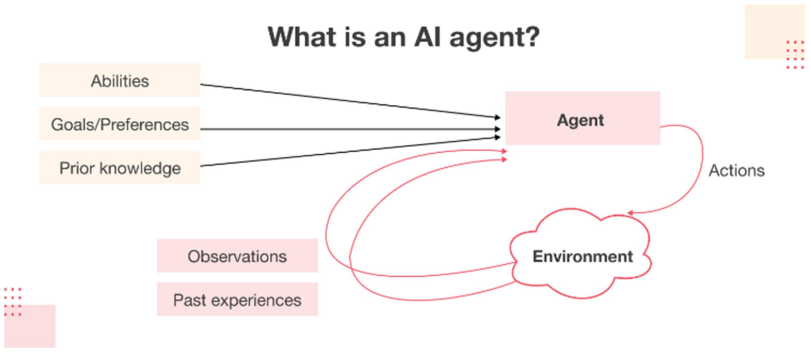


Fig. Agent in AI

An agent can be:

- **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
- **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
- **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.

Hence the world around us is full of agents such as thermostat, cellphone, camera, and even we are also agents.

Intelligent Agents:

An intelligent agent is an autonomous entity which act upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example of an intelligent agent.

Following are the main four rules for an AI agent:

- **Rule 1:** An AI agent must have the ability to perceive the environment.
- **Rule 2:** The observation must be used to make decisions.
- **Rule 3:** Decision should result in an action.
- **Rule 4:** The action taken by an AI agent must be a rational action.

Rational Agent:

A rational agent is an agent which has clear preference, models uncertainty, and acts in a way to maximize its performance measure with all possible actions.

A rational agent is said to perform the right things. AI is about creating rational agents to use for game theory and decision theory for various real-world scenarios.

For an AI agent, the rational action is most important because in AI reinforcement learning algorithm, for each best possible action, agent gets the positive reward and for each wrong action, an agent gets a negative reward.

Note: Rational agents in AI are very similar to intelligent agents.

Rationality:

The rationality of an agent is measured by its performance measure. Rationality can be judged on the basis of following points:

- Performance measure which defines the success criterion.
- Agent prior knowledge of its environment.
- Best possible actions that an agent can perform.
- The sequence of percepts.

Note: Rationality differs from Omniscience because an Omniscient agent knows the actual outcome of its action and act accordingly, which is not possible in reality.

Components of AI agents

AI agents comprise many key components that work together to perceive, reason, and act in their environment. Understanding these components will help you grasp how AI systems function and make decisions.

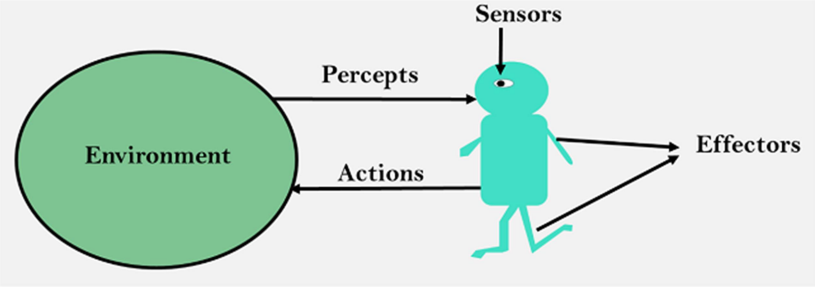


Fig. Interaction of agents with the environment

Sensors: Sensor is a device which detects the change in the environment and sends the information to other electronic devices. An agent observes its environment through sensors.

These allow the models to gather information from its surroundings, much like human senses. For a robot, sensors might include cameras for vision, microphones for hearing, or touch sensors for physical interaction. In software agents, sensors could be data inputs or API connections.

Perception module: This processes the raw sensor data into meaningful information. It involves tasks like image recognition, speech-to-text conversion, or data preprocessing. The perception module essentially interprets the world for the AI.

Actuator: Actuators are the component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, rails, etc. It carries out the chosen actions. In a physical robot, actuators might be motors or mechanical parts. For software agents, actuators could be functions that modify data, send messages, or control other systems.

These components are continuous: sense, perceive, think, decide, and act. Understanding this structure can help you better conceptualize how AI agents operate and interact with their environment.

Effectors: Effectors are the devices which affect the environment. Effectors can be legs, wheels, arms, fingers, wings, fins, and display screen.

Cognitive architecture: It encompasses the AI's knowledge base, reasoning mechanisms, and learning algorithms.

- The knowledge base stores information and rules the AI uses to make decisions.
- Reasoning mechanisms allow the AI to draw conclusions and plan actions based on knowledge and current perceptions.
- Learning algorithms enable the AI to improve its performance over time by adjusting its knowledge and decision-making processes.

Decision-making: It uses processed information and cognitive architecture to determine the best course of action. Depending on the AI's design and purpose, this might involve techniques like search algorithms, planning systems, or neural networks.

Structure/ Architecture of an AI Agent

To understand the structure of Intelligent Agents, we should be familiar with *Architecture* and *Agent* programs. **Architecture** is the machinery that the agent executes on. It is a device with sensors and actuators, for example, a robotic car, a camera, and a

PC. An **agent function** is a map from the percept sequence (history of all that an agent has perceived to date) to an action.

The task of AI is to design an agent program which implements the agent function. The structure of an intelligent agent is a combination of architecture and agent program. It can be viewed as:

Agent = Architecture + Agent program

What are the key components of AI agent architecture?

Agents in artificial intelligence may operate in different environments to accomplish unique purposes. However, all functional agents share these components.

Following are the main three terms involved in the structure of an AI agent:

- program, or a combination. For example, a robotic AI agent consists of actuators, sensors **Architecture:** Architecture is machinery that an AI agent executes on.

  Architecture is the base the agent operates from. The architecture can be a physical structure, a software, motors, and robotic arms. Meanwhile, an architecture that hosts an AI software agent may use a text prompt, API, and databases to enable autonomous operations.
- **Agent function:** Agent function is used to map a percept to an action. f:P* → A

  The agent function describes how the data collected is translated into actions that support the agent's objective. When designing the agent function, developers consider the type of information, AI capabilities, knowledge base, feedback mechanism, and other technologies required.
- **Agent program:** An agent program executes on the physical architecture to produce function f.

  An agent program is the implementation of the agent function. It involves developing, training, and deploying the AI agent on the designated architecture. The agent program aligns the agent's business logic, technical requirements, and performance elements.

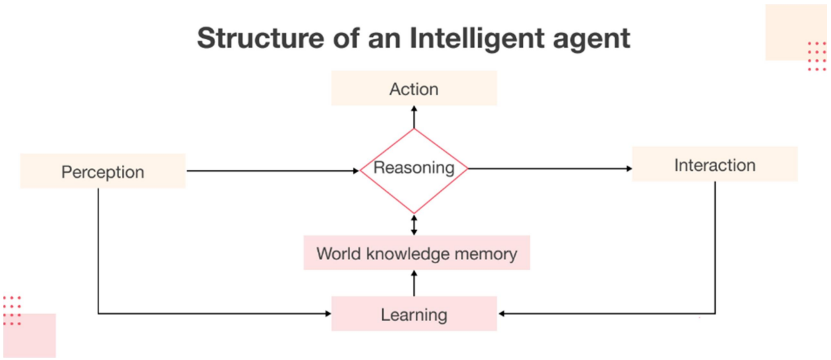


Fig. Structure of Intelligent Agent

At its core, an AI agent is made up of four components: the environment, sensors, actuators, and the decision-making mechanism.

i. **Environment:** The environment refers to the area or domain in which an AI agent operates. It can be a physical space, like a factory floor, or a digital space, like a website.

ii. **Sensors:** Sensors are the tools that an AI agent uses to perceive its environment. These can be cameras, microphones, or any other sensory input that the AI agent can use to understand what is happening around it.

iii. **Actuators:** Actuators are the tools that an AI agent uses to interact with its environment. These can be things like robotic arms, computer screens, or any other device the AI agent can use to change the environment.

iv. **Decision-making mechanism:** A decision-making mechanism is the brain of an AI agent. It processes the information gathered by the sensors and decides what action to take using the actuators. The decision-making mechanism is where the real magic happens.

AI agents use various decision-making mechanisms, such as rule-based systems, expert systems, and neural networks, to make informed choices and perform tasks effectively.

v. **Learning system:** The learning system enables the AI agent to learn from its experiences and interactions with the environment. It uses techniques like reinforcement learning, supervised learning, and unsupervised learning to improve the performance of the AI agent over time.

AI agents operate based on a combination of algorithms and data inputs. They process information using machine learning models to interpret and react to their environment. Key functional components include:

- **Data acquisition:** AI agents acquire data through sensors or data intake mechanisms. This data serves as the foundation for all subsequent operations.
- **Processing and analysis:** Utilizing machine learning and artificial intelligence algorithms, the agent examines and draws insights from the data.
- **Decision making:** They make decisions based on the analysis, which can involve complex algorithms, rule-based logic, or predictive models.
- **Action execution:** Once a decision is made, the agent executes an action, which can be anything from updating a database to controlling a physical robot.

The workflow for an AI agent is often structured as follows:

1. **Receive data:** Obtain new information from the environment or a user.
2. **Analyze data:** Contextualize and interpret the information using AI models.
3. **Decide on action:** Determine the best course of action.
4. **Act:** Implement the decision through a response or a change in the environment.

An example of AI agent functionality in practice might be a customer service chatbot:

- **Intake:** Receives a customer query.
- **Process:** Understands the query using natural language processing.
- **Decide:** Choose an appropriate response based on the query context.
- **Respond:** Replies to the customer with information or further questions.

Through these mechanisms, AI agents are integral to automating complex tasks that require adaptability and learning capability.

Characteristics of an Agent

While AI tools and agents are software programs designed to automate tasks, specific key characteristics differentiate AI agents as more sophisticated AI software.

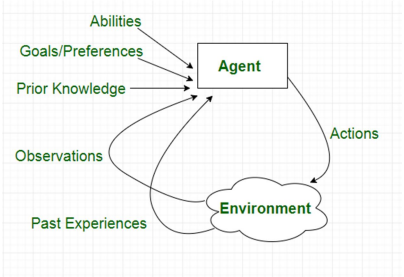


Fig. Characteristics of AI Agent

An AI tool as an AI agent when it has the following characteristics:

- **Autonomy:** An AI virtual agent is capable of performing tasks independently without requiring constant human intervention or input.
- **Perception:** The agent function senses and interprets the environment they operate in through various sensors, such as cameras or microphones.
- **Reactivity:** An AI agent can assess the environment and respond accordingly to achieve its goals.
- **Reasoning and decision-making:** AI agents are intelligent tools that can analyze data and make decisions to achieve goals. They use reasoning techniques and algorithms to process information and take appropriate actions.
- **Learning:** They can learn and enhance their performance through machine, deep, and reinforcement learning elements and techniques.
- **Communication:** AI agents can communicate with other agents or humans using different methods, like understanding and responding to natural language, recognizing speech, and exchanging messages through text.
- **Goal-oriented:** They are designed to achieve specific goals, which can be pre-defined or learned through interactions with the environment.

Types of Agents

Agents can be grouped into five classes based on their degree of perceived intelligence and capability:

- Simple Reflex Agents
- Model-Based Reflex Agents
- Goal-Based Agents
- Utility-Based Agents
- Learning Agent
- Multi-agent systems
- Hierarchical agents

Simple Reflex Agents

Simple reflex agents ignore the rest of the percept history and act only on the basis of the **current percept**. Percept history is the history of all that an agent has perceived to date. The agent function is based on the **condition-action rule**. A condition-action rule is a rule that maps a state i.e., a condition to an action. If the condition is true, then the action is taken, else not. This agent function only succeeds when the environment is fully observable. For simple reflex agents operating in partially observable environments, infinite loops are often unavoidable. It may be possible to escape from infinite loops if the agent can randomize its actions.

Problems with Simple reflex agents are :

- Very limited intelligence.
- No knowledge of non-perceptual parts of the state.
- Usually too big to generate and store.
- If there occurs any change in the environment, then the collection of rules needs to be updated.

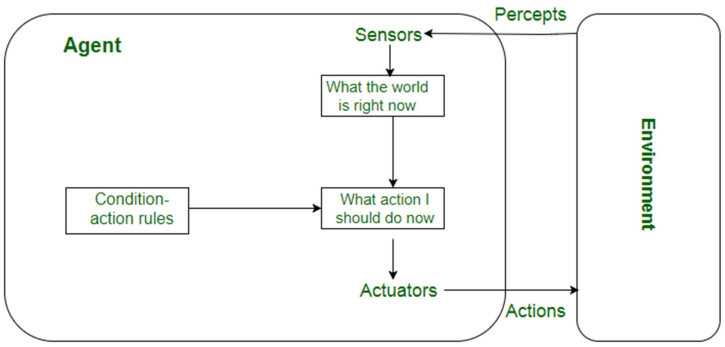


Fig. Simple Reflex Agents

Model-Based Reflex Agents

It works by finding a rule whose condition matches the current situation. A model-based agent can handle **partially observable environments** by the use of a model about the world. The agent has to keep track of the **internal state** which is adjusted by each percept and that depends on the percept history. The current state is stored inside the agent which maintains some kind of structure describing the part of the world which cannot be seen.

Updating the state requires information about:

- How the world evolves independently from the agent?
- How do the agent's actions affect the world?

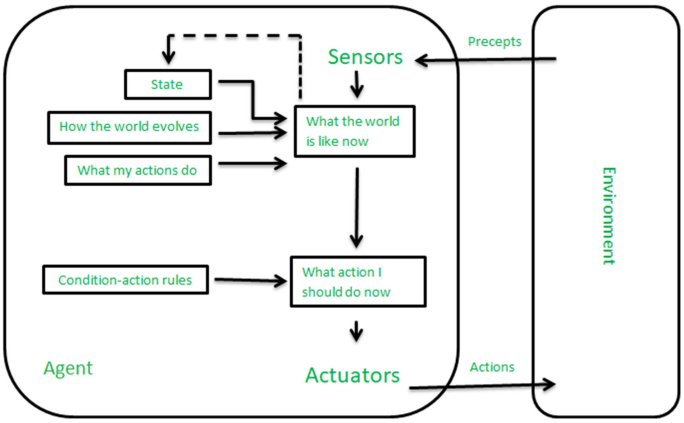


Fig. Model-Based Reflex Agents

Goal-Based Agents

These kinds of agents take decisions based on how far they are currently from their **goal** (description of desirable situations). Their every action is intended to reduce their distance from the goal. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state. The knowledge that supports its decisions is represented explicitly and can be modified, which makes these agents more flexible. They usually require search and planning. The goal-based agent's behavior can easily be changed.

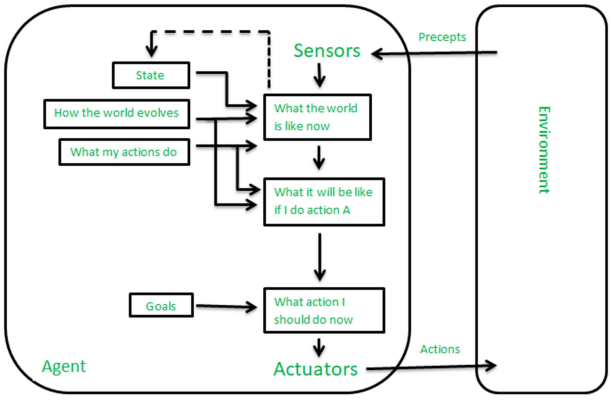


Fig. Goal-Based Agents

Utility-Based Agents

The agents which are developed having their end uses as building blocks are called utility-based agents. When there are multiple possible alternatives, then to decide which one is best, utility-based agents are used. They choose actions based on a **preference (utility)** for each state. Sometimes achieving the desired goal is not enough. We may look for a quicker, safer, cheaper trip to reach a destination. Agent happiness should be taken into consideration. Utility describes how **"happy"** the agent is. Because of the uncertainty in the world, a utility agent chooses the action that maximizes the expected utility. A utility function maps a state onto a real number which describes the associated degree of happiness.

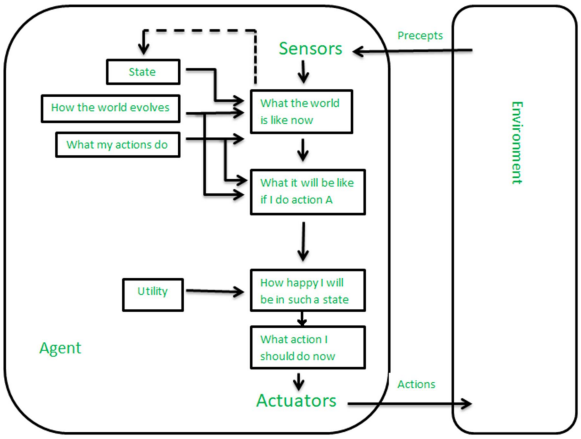


Fig. Utility-Based Agents

Learning Agent

A learning agent in AI is the type of agent that can learn from its past experiences or it has learning capabilities. It starts to act with basic knowledge and then is able to act and adapt automatically through learning. A learning agent has mainly four conceptual components, which are:

1. **Learning element:** It is responsible for making improvements by learning from the environment.
2. **Critic:** The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standard.
3. **Performance element:** It is responsible for selecting external action.

4. **Problem Generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.

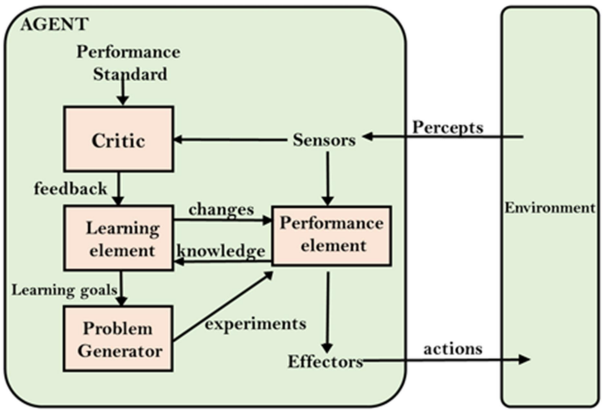


Fig. Learning Agent

Multi-Agent Systems

Multi-agent systems involve multiple agents working together to achieve a common goal. These agents may have to coordinate their actions and communicate with each other to achieve their objectives. Agents are used in a variety of applications, including robotics, gaming, and intelligent systems. They can be implemented using different programming languages and techniques, including machine learning and natural language processing.

- These agents interact with other agents to achieve a common goal. They may have to coordinate their actions and communicate with each other to achieve their objective.
- A multi-agent system (MAS) is a system composed of multiple interacting agents that are designed to work together to achieve a common goal. These agents may be autonomous or semi-autonomous and are capable of perceiving their environment, making decisions, and taking action to achieve the common objective.
- MAS can be used in a variety of applications, including transportation systems, robotics, and social networks. They can help improve efficiency, reduce costs, and increase flexibility in complex systems. MAS can be classified into different types based on their characteristics, such as whether the agents have the same or different goals, whether the agents are cooperative or competitive, and whether the agents are homogeneous or heterogeneous.

- In a homogeneous MAS, all the agents have the same capabilities, goals, and behaviors.
- In contrast, in a heterogeneous MAS, the agents have different capabilities, goals, and behaviors.

This can make coordination more challenging but can also lead to more flexible and robust systems.

- Cooperative MAS involves agents working together to achieve a common goal, while competitive MAS involves agents working against each other to achieve their own goals. In some cases, MAS can also involve both cooperative and competitive behavior, where agents must balance their own interests with the interests of the group.
- MAS can be implemented using different techniques, such as game theory, machine learning, and agent-based modeling. Game theory is used to analyze strategic interactions between agents and predict their behavior. Machine learning is used to train agents to improve their decision-making capabilities over time. Agent-based modeling is used to simulate complex systems and study the interactions between agents.
- Overall, multi-agent systems are a powerful tool in artificial intelligence that can help solve complex problems and improve efficiency in a variety of applications.

Hierarchical Agents

These agents are organized into a hierarchy, with high-level agents overseeing the behavior of lower-level agents. The high-level agents provide goals and constraints, while the low-level agents carry out specific tasks. Hierarchical agents are useful in complex environments with many tasks and sub-tasks.

- Hierarchical agents are agents that are organized into a hierarchy, with high-level agents overseeing the behavior of lower-level agents. The high-level agents provide goals and constraints, while the low-level agents carry out specific tasks. This structure allows for more efficient and organized decision-making in complex environments.

- Hierarchical agents can be implemented in a variety of applications, including robotics, manufacturing, and transportation systems. They are particularly useful in environments where there are many tasks and sub-tasks that need to be coordinated and prioritized.
- In a hierarchical agent system, the high-level agents are responsible for setting goals and constraints for the lower-level agents. These goals and constraints are typically based on the overall objective of the system. For example, in a manufacturing system, the high-level agents might set production targets for the lower-level agents based on customer demand.
- The low-level agents are responsible for carrying out specific tasks to achieve the goals set by the high-level agents. These tasks may be relatively simple or more complex, depending on the specific application. For example, in a transportation system, low-level agents might be responsible for managing traffic flow at specific intersections.
- Hierarchical agents can be organized into different levels, depending on the complexity of the system. In a simple system, there may be only two levels: high-level agents and low-level agents. In a more complex system, there may be multiple levels, with intermediate-level agents responsible for coordinating the activities of lower-level agents.
- One advantage of hierarchical agents is that they allow for more efficient use of resources. By organizing agents into a hierarchy, it is possible to allocate tasks to the agents that are best suited to carry them out, while avoiding duplication of effort. This can lead to faster, more efficient decision-making and better overall performance of the system.

Overall, hierarchical agents are a powerful tool in artificial intelligence that can help solve complex problems and improve efficiency in a variety of applications.

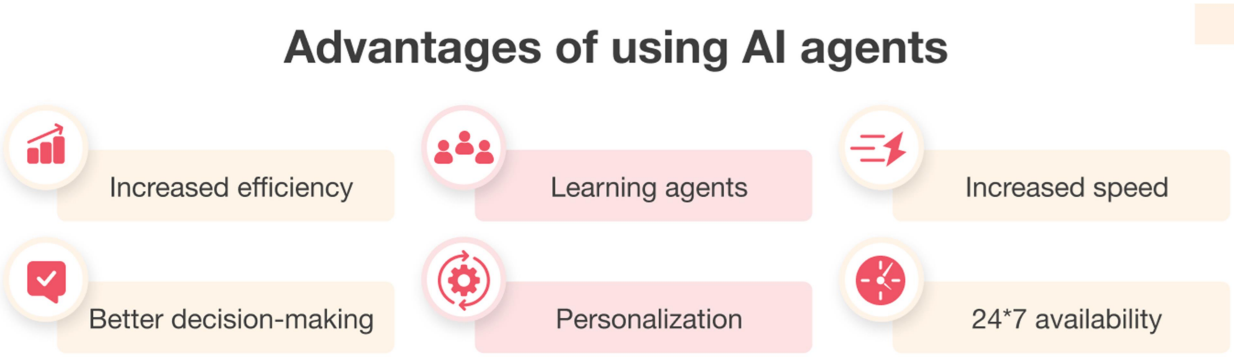
Comparison between the AI Agents

AI agents can be categorized based on their operational sophistication and interaction with the environment. The following table provides a clear classification:

Type	Reactivity	Autonomy	Capability	Example usage
Simple Reflex	Responds to current perceptions only.	Limited; follows pre-programmed rules.	Basic tasks; low adaptability.	Meeting scheduler
Model-based Reflex	Considers internal states for decision making.	Some internal state handling ability.	Can handle partially observable environments.	Identifying security breaches
Goal-based	Actions are taken to achieve specific goals.	Able to evaluate different action paths.	Better planning; deals with complex tasks.	Managing and monitoring projects for set objectives
Utility-based	Decisions are made to maximize utility.	Chooses actions that yield the highest benefit.	Balances task success with cost.	Investment analysis tools
Learning Agent	Learns from experience to improve performance.	Adapts to new situations over time.	Very adaptable; can become highly proficient.	Personalized recommendation systems

Agents are also distinguished by the degree of their learning capabilities, from those that cannot learn, to those that learn from their interactions with the environment, continually improving their performance.

Advantages of using AI agents



AI agents offer businesses the potential to streamline operations, make informed decisions, improve customer experiences, and drive growth and competitiveness in the digital age.

1. Increased efficiency

AI agents can automate repetitive tasks, allowing businesses to complete them faster and more accurately. This efficiency improvement frees employees' time to focus on more business-critical tasks and improves productivity.

2. Better decision-making

AI agents can analyze large amounts of data and provide valuable insights to support decision-making processes. By leveraging advanced algorithms and machine learning, AI agents can identify patterns, trends, and correlations that humans may overlook.

3. Improved customer experience

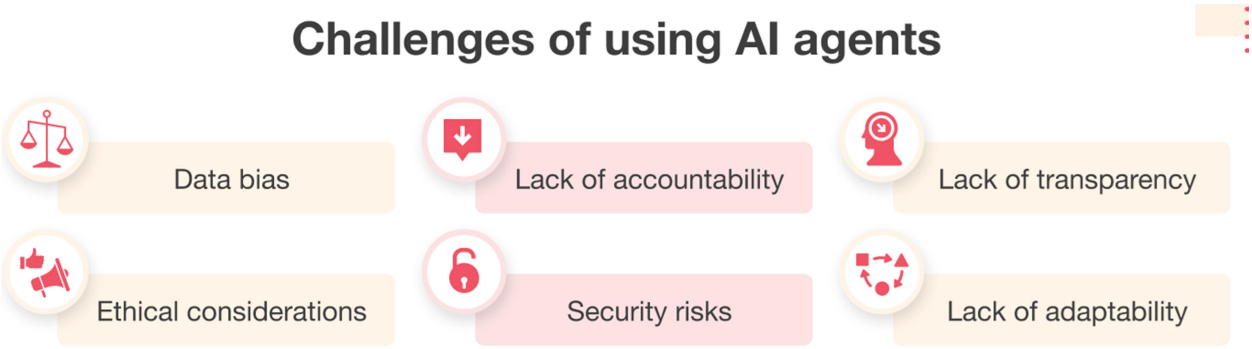
AI agents can provide personalized and timely interactions with customers, enhancing their experience. They can offer instant support, answer queries, and provide recommendations, leading to increased customer satisfaction and loyalty.

4. Cost savings

By automating tasks, AI agents can reduce the need for human resources and manual labor, resulting in cost savings for businesses. They can handle high-volume, repetitive tasks without fatigue or errors.

Most common challenges you may face while using an AI agent

Autonomous AI agents have become increasingly popular in recent years, with many brands implementing them for various purposes. However, several challenges come with using these agents. Some of the most common challenges are:



- **Data bias:** An autonomous artificial intelligence agent program relies heavily on data to make decisions. If the data they use is biased, it can lead to unfair or discriminatory outcomes. **Amazon's AI recruiting tool was biased against women**, leading to a skewed hiring process.
- **Lack of accountability:** Proactive agents can make decisions without human intervention, so holding them accountable for their actions can be difficult. Uber's autonomous vehicle that struck and **killed a pedestrian in 2018** raised questions about who was responsible.
- **Lack of transparency:** The decision-making processes of a learning agent can be complex and opaque, making it difficult to understand how they arrive at certain decisions.
- **Ethical considerations:** A rational agent can make decisions with ethical implications, and it can be challenging to program them to make ethical decisions. Microsoft's chatbot, Tay, was shut down after it started making racist and sexist comments.
- **Security risks:** Software agents can be vulnerable to cyber-attacks, compromising their decision-making processes or leading to data breaches.
- **Lack of adaptability:** Autonomous AI agents act according to their training data, which means they can struggle to adapt to new situations or contexts.

Applications of AI agents

Autonomous AI agents have numerous applications in various industries, including:

- **Healthcare:** Autonomous artificial intelligence agents can assist in diagnosing, treating, and monitoring patients. **IBM Watson Health** is an AI agent that can analyze medical data to identify potential health issues and recommend treatment options.
- **Finance:** They can analyze financial data, detect fraud, and make investment recommendations. Charles Schwab uses an AI agent called **Intelligent Portfolio** to create and manage portfolios based on customers' investment goals.
- **Retail:** Agents can provide personalized recommendations, improve supply chain management, and enhance customer experience. **Amazon's Alexa** is an AI agent that can recommend products, place orders, and track shipments.
- **Manufacturing:** Intelligent agents can optimize production processes, predict maintenance needs, and improve product quality. General Electric uses an AI agent called **Predix** to monitor machines in real-time to predict and prevent equipment failures.
- **Transportation:** Autonomous AI agents can assist in route planning, traffic management, and vehicle safety. **Tesla's Autopilot** helps drive a vehicle autonomously and helps the driver park, lane change, and safe driving.
- **Education:** They can provide personalized learning experiences, automate administrative tasks, and analyze student performance. **Pearson's AI agent, Aida**, can provide feedback to students and suggest personalized learning paths.
- **Agriculture:** AI agents can optimize crop production, monitor soil quality, and predict weather patterns. John Deere uses an AI agent called **See & Spray** to detect and target weeds without affecting crops.
- **Robotics:** Agents can be used to control robots and automate tasks in manufacturing, transportation, and other industries.
- **Smart homes and buildings:** Agents can be used to control heating, lighting, and other systems in smart homes and buildings, optimizing energy use and improving comfort.
- **Games:** Agents can be used to create intelligent opponents in games and simulations, providing a more challenging and realistic experience for players.

- **Natural language processing:** Agents can be used for language translation, question answering, and chatbots that can communicate with users in natural language.
- **Cybersecurity:** Agents can be used for intrusion detection, malware analysis, and network security.
- **Environmental monitoring:** Agents can be used to monitor and manage natural resources, track climate change, and improve environmental sustainability.
- **Social media:** Agents can be used to analyze social media data, identify trends and patterns, and provide personalized recommendations to users.

Examples of agents in artificial intelligence

There are many examples of agents in artificial intelligence. Here are a few:

- **Intelligent personal assistants:** These are agents that are designed to help users with various tasks, such as scheduling appointments, sending messages, and setting reminders. Examples of intelligent personal assistants include Siri, Alexa, and Google Assistant.
- **Autonomous robots:** These are agents that are designed to operate autonomously in the physical world. They can perform tasks such as cleaning, sorting, and delivering goods. Examples of autonomous robots include the Roomba vacuum cleaner and the Amazon delivery robot.
- **Gaming agents:** These are agents that are designed to play games, either against human opponents or other agents. Examples of gaming agents include chess-playing agents and poker-playing agents.
- **Fraud detection agents:** These are agents that are designed to detect fraudulent behavior in financial transactions. They can analyze patterns of behavior to identify suspicious activity and alert authorities. Examples of fraud detection agents include those used by banks and credit card companies.
- **Traffic management agents:** These are agents that are designed to manage traffic flow in cities. They can monitor traffic patterns, adjust traffic lights, and reroute vehicles to minimize congestion. Examples of traffic management agents include those used in smart cities around the world.

- A **software agent** has Keystrokes, file contents, received network packages that act as sensors and displays on the screen, files, and sent network packets acting as actuators.
- A Human-agent has eyes, ears, and other organs which act as sensors, and hands, legs, mouth, and other body parts act as actuators.
- A **Robotic agent** has Cameras and infrared range finders which act as sensors and various motors act as actuators.

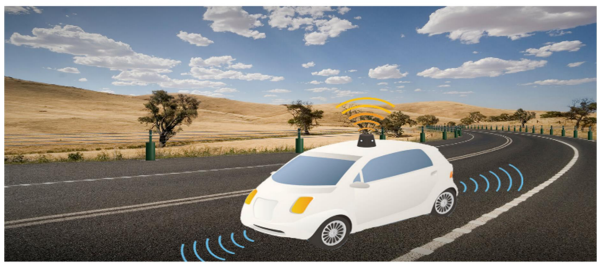
PEAS Representation

PEAS is a type of model on which an AI agent works upon. When we define an AI agent or rational agent, then we can group its properties under PEAS representation model. It is made up of four words:

- **P:** Performance measure
- **E:** Environment
- **A:** Actuators
- **S:** Sensors

Here performance measure is the objective for the success of an agent's behavior.

PEAS for self-driving cars:



Let's suppose a self-driving car then PEAS representation will be:

Performance: Safety, time, legal drive, comfort

Environment: Roads, other vehicles, road signs, pedestrian

Actuators: Steering, accelerator, brake, signal, horn

Sensors: Camera, GPS, speedometer, odometer, accelerometer, sonar.

Agent	Performance measure	Environment	Actuators	Sensors
Medical Diagnose	○ Healthy patient ○ Minimized cost	○ Patient ○ Hospital ○ Staff	○ Tests ○ Treatments	Keyboard (Entry of symptoms)
Vacuum Cleaner	○ Cleanness ○ Efficiency ○ Battery life ○ Security	○ Room ○ Table ○ Wood floor ○ Carpet ○ Various obstacles	○ Wheels ○ Brushes ○ Vacuum Extractor	○ Camera ○ Dirt detection sensor ○ Cliff sensor ○ Bump Sensor ○ Infrared Wall Sensor
Part - picking Robot	○ Percentage of parts in correct bins.	○ Conveyor belt with parts, ○ Bins	○ Jointed Arms ○ Hand	○ Camera ○ Joint angle sensors.

Generative AI Frameworks

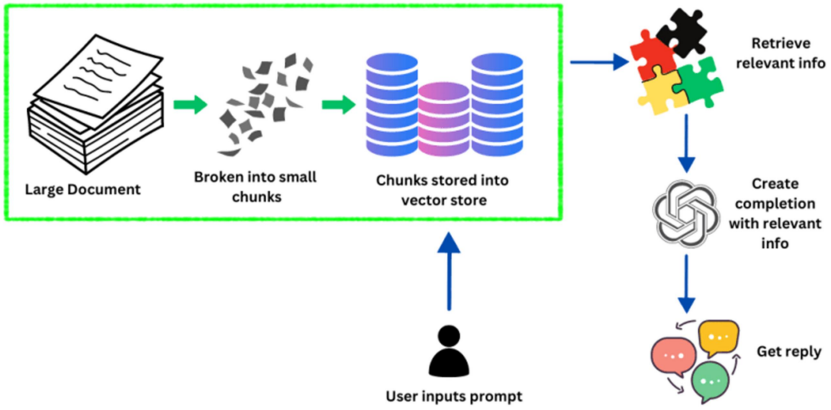
Generative AI Frameworks are the backbone of GenAI, providing the infrastructure enabling machines to create diverse and contextually relevant content. These frameworks act as the guiding principles for AI models, such as LLMs, GANs, VAEs, allowing them to understand patterns within vast datasets. Organizations can harness the power of unsupervised and semi-supervised learning approaches to train AI systems by utilizing

these frameworks. This training forms the foundation for tasks ranging from NLP to image generation, empowering machines to interpret prompts.

LangChain LangChain

Developed by Harrison Chase and debuted in October 2022, LangChain serves as an open-source platform designed for constructing sturdy applications powered by LLMs, such as chatbots like ChatGPT and various tailor-made applications.

LangChain seeks to equip data engineers with an all-encompassing toolkit for utilizing LLMs in diverse use cases, including chatbots, automated question-answering, text summarization, and more.



The above image shows how LangChain handles and processes information to respond to user prompts. Initially, the system starts with a large document containing a vast array of data. This document is then broken down into smaller, more manageable chunks.

These chunks are subsequently embedded into vectors — a process that transforms the data into a format that can be quickly and efficiently retrieved by the system. These vectors are stored in a vector store, essentially a database optimized for handling vectorized data.

When a user inputs a prompt into the system, LangChain queries this vector store to find information that closely matches or is relevant to the user's request. The system employs large LLMs to understand the context and intent of the user's prompt, which guides the retrieval of pertinent information from the vector store.

Once the relevant information is identified, the LLM uses it to generate or complete an answer that accurately addresses the query. This final step culminates in the user receiving a tailored

response, which is the output of the system's data processing and language generation capabilities.

Features of LangChain

LangChain is made up of the following modules that ensure the multiple components needed to make an effective NLP app can run smoothly:

- **Model interaction.** Also called model I/O, this module lets LangChain interact with any language model and perform tasks such as managing inputs to the model and extracting information from its outputs.
- **Data connection and retrieval.** Data that LLMs access can be transformed, stored in databases and retrieved from those databases through queries with this module.
- **Chains.** When using LangChain to build more complex apps, other components or even more than one LLM might be required. This module links multiple LLMs with other components or LLMs. This is referred to as an LLM chain.
- **Agents.** The agent module lets LLMs decide the best steps or actions to take to solve problems. It does so by orchestrating a series of complex commands to LLMs and other tools to get them to respond to specific requests.
- **Memory.** The memory module helps an LLM remember the context of its interactions with users. Both short-term memory and long-term memory can be added to a model, depending on the specific use.

LangChain Key Concepts:

The main properties of LangChain Framework are :

- **Components:** Components are modular building blocks that are ready and easy to use to build powerful applications. Components include LLM Wrappers, Prompt Template and Indexes for relevant information retrieval.
- **Chains:** Chains allow us to combine multiple components together to solve a specific task. Chains make it easy for the implementation of complex applications by making it more modular and simple to debug and maintain.

- **Agents:** Agents allow LLMs to interact with their environment. For example, using an external API to perform a specific action.

Setting up the environment

Installation of langchain is very simple and similar as you install other libraries using the pip command.

```
!pip install langchain
```

There are various LLMs that you can use with LangChain. In this article, I will be using OpenAI. Let us install Openai using the following command:

```
!pip install openai
```

I am also installing the dotenv library to store the API key in an environmental variable. Install it using the command:

```
!pip install python-dotenv
```

You can generate your own API key by signing up to the openai platform. Next, we create a .env file and store our API key in it as follows:

Python

```
OPENAI_KEY='your_api_key'
```

Now, I am creating a new file named 'lang.py' where I will be using the LangChain framework to generate responses. Let us start by importing the required libraries as follows:

```
import os
import openai,langchain
from dotenv import load_dotenv
load_dotenv()


api_key=os.getenv(&amp;quot;OPENAI_KEY&amp;quot;,None)
```

That was the initial setup required to use the LangChain framework with OpenAI LLM.

Creating prompts in LangChain

Prompts serve as input to the LLM that instructs it to return a response, which is often an answer to a query. This response is also referred to as an output. A prompt must be designed and executed correctly to increase the likelihood of a well-written and accurate response from

a language model. That is why prompt engineering is an emerging science that has received more attention in recent years.

Prompts can be generated easily in LangChain implementations using a prompt template, which will be used as instructions for the underlying LLM. Prompt templates can vary in specificity. They can be designed to pose simple questions to a language model. They can also be used to provide a set of explicit instructions to a language model with enough detail and examples to retrieve a high-quality response.

With Python programming, LangChain has a premade prompt template that takes the form of structured text. The following steps are required to use this:

- **Installing Python.** A recent version of Python must be installed. Once the Python shell terminal is open, enter the following command to install just the bare minimum requirements of LangChain for the sake of this example.

```
pip install langchain
```

- **Adding integrations.** LangChain typically requires at least one integration. OpenAI is a prime example. To use OpenAI's LLM application programming interfaces, a developer must create an account on the OpenAI website and retrieve the API access key. Then, using the following code snippet, install OpenAI's Python package and enter the key for access to the APIs.

```
pip install openai
from langchain.llms import OpenAI
llm = OpenAI(openai_api_key="...")
```

- **Importing the prompt template.** Once these basic steps are complete, LangChain's prompt template method must then be imported. The code snippet shown below does this.

```
from langchain import PromptTemplate
prompt_template = PromptTemplate.from_template(
   "Tell me an {adjective} fact about {content}."
)
prompt_template.format(adjective="interesting", content="zebras")
"Tell me an interesting fact about zebras."
```

In this scenario, the language model would be expected to take the two input variables -- the adjective and the content -- and produce a fascinating fact about zebras as its output.

How to develop applications in LangChain

LangChain is built to develop apps powered by language model functionality. There are different ways to do this, but the process typically entails some key steps:

- **Define the application.** An application developer must first define a specific use case for the application. This also means determining its scope, including requirements such as any needed integrations, components and LLMs.
- **Build functionality.** Developers use prompts to build the functionality or logic of the intended app.
- **Customize functionality.** LangChain lets developers modify its code to create customized functionality that meets the needs of the use case and shapes the application's behavior.
- **Fine-tuning LLMs.** It's important to choose the appropriate LLM for the job and also to fine-tune it to adhere to the needs of the use case.
- **Data cleansing.** Using data cleansing techniques ensures clean and accurate data sets. Also, security measures should be implemented to protect sensitive data.
- **Testing.** Regularly testing LangChain apps ensures they continue to run smoothly.