Generative AI and Large Language Models (LLMs)

Content

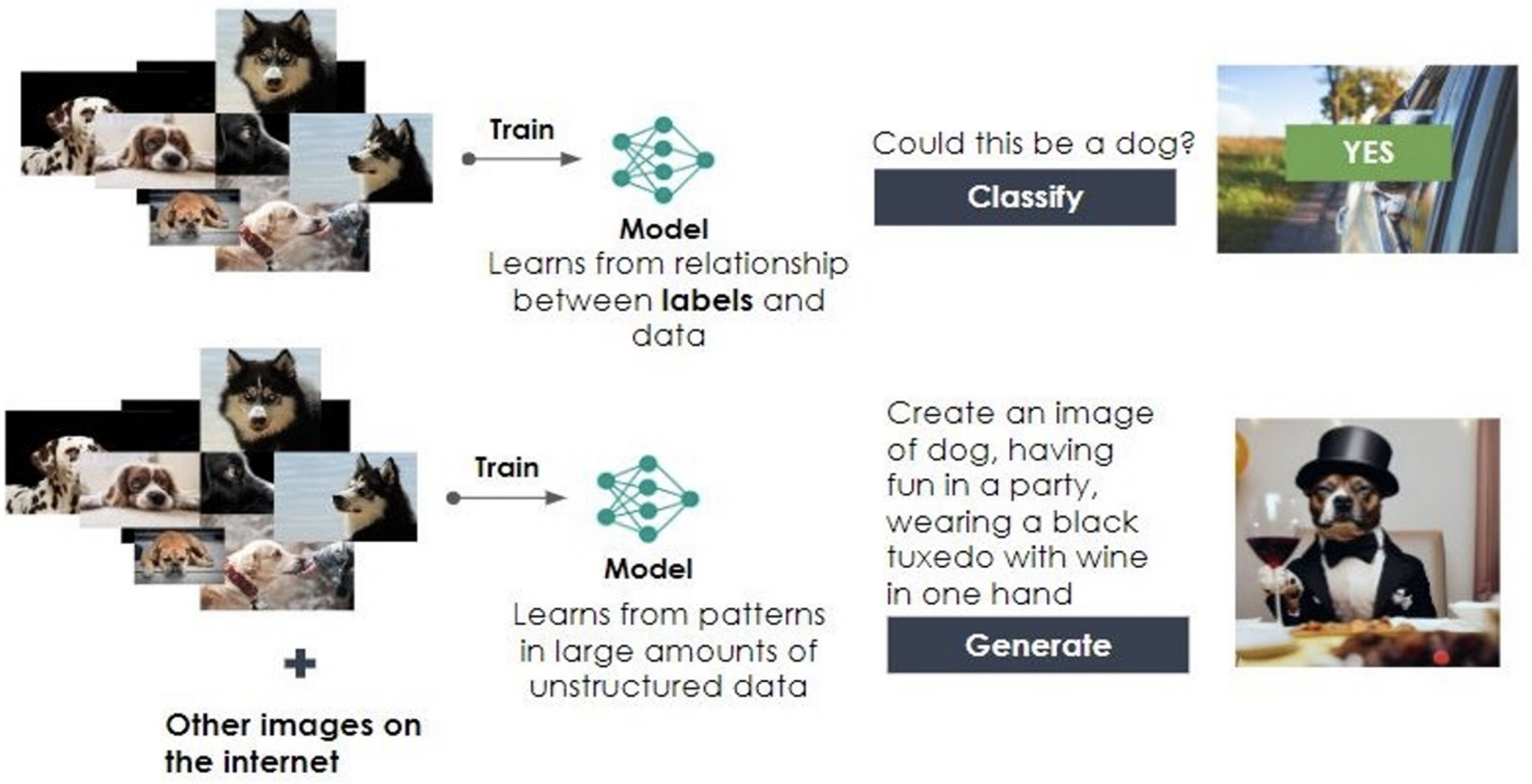
- Introduction to Generative AI
  - Discriminative AI vs. Generative AI
- Introduction to Large Language Models (LLMs)
- Popular LLMs
  - GPT-3
  - FLAN-T5
  - LLaMA-2
  - Falcon
- Leveraging LLMs
  - Prompt Engineering
  - Fine-tuning LLM
  - Building LLM
- LLM Application Development tools : LangChain
- Use cases of Generative AI
- Generative AI Ethical Concerns

Introduction to Generative AI

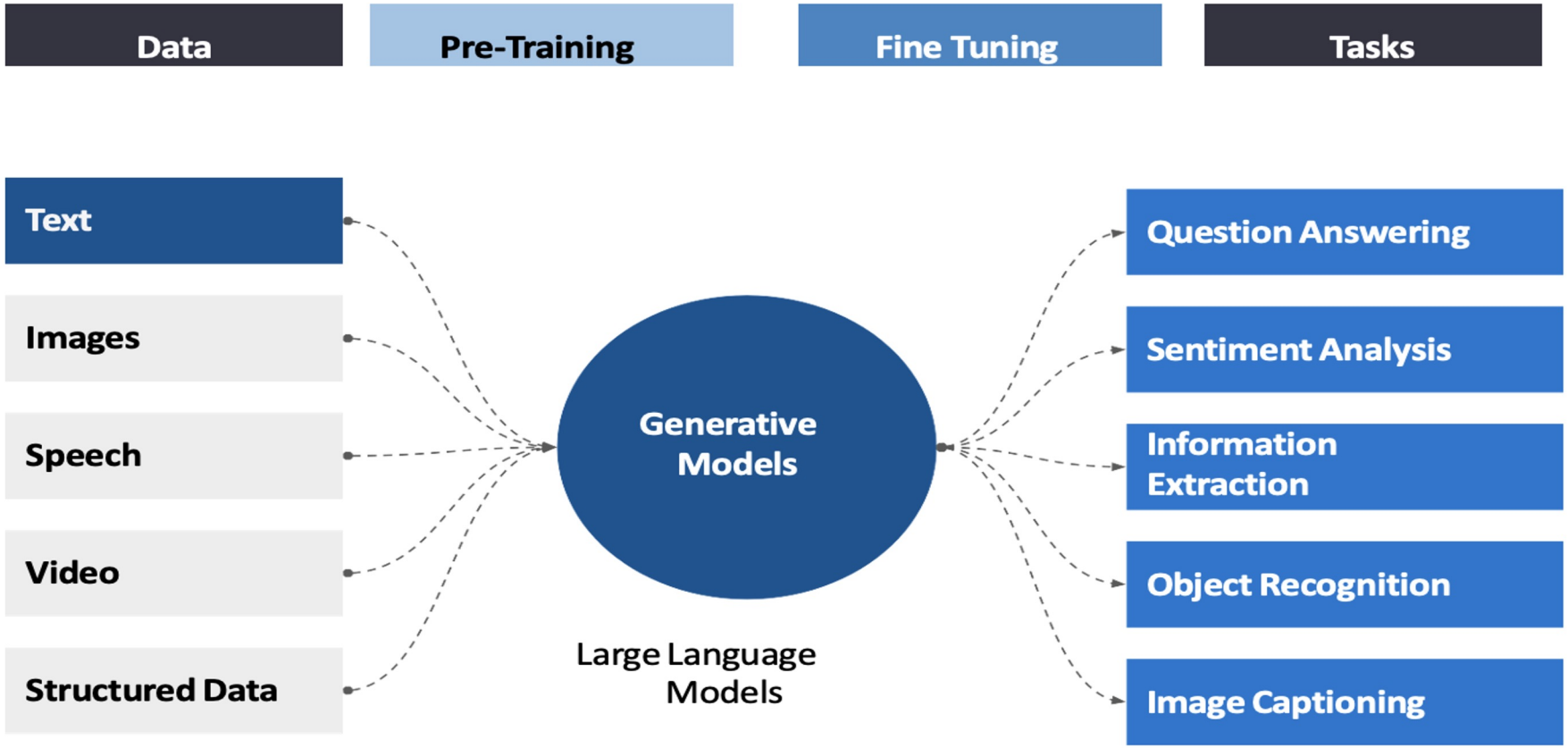
Generative AI

- Generative AI, also termed as creative AI, is a subset of machine learning that generates novel/unique content using artificial intelligence
- Types of content include, Text, Images, Audio, Videos etc.
- It differs from traditional AI models designed to recognize patterns and make predictions based on existing data
- Typically, generative models (in the context of natural language processing) are Large Language Models (LLMs)

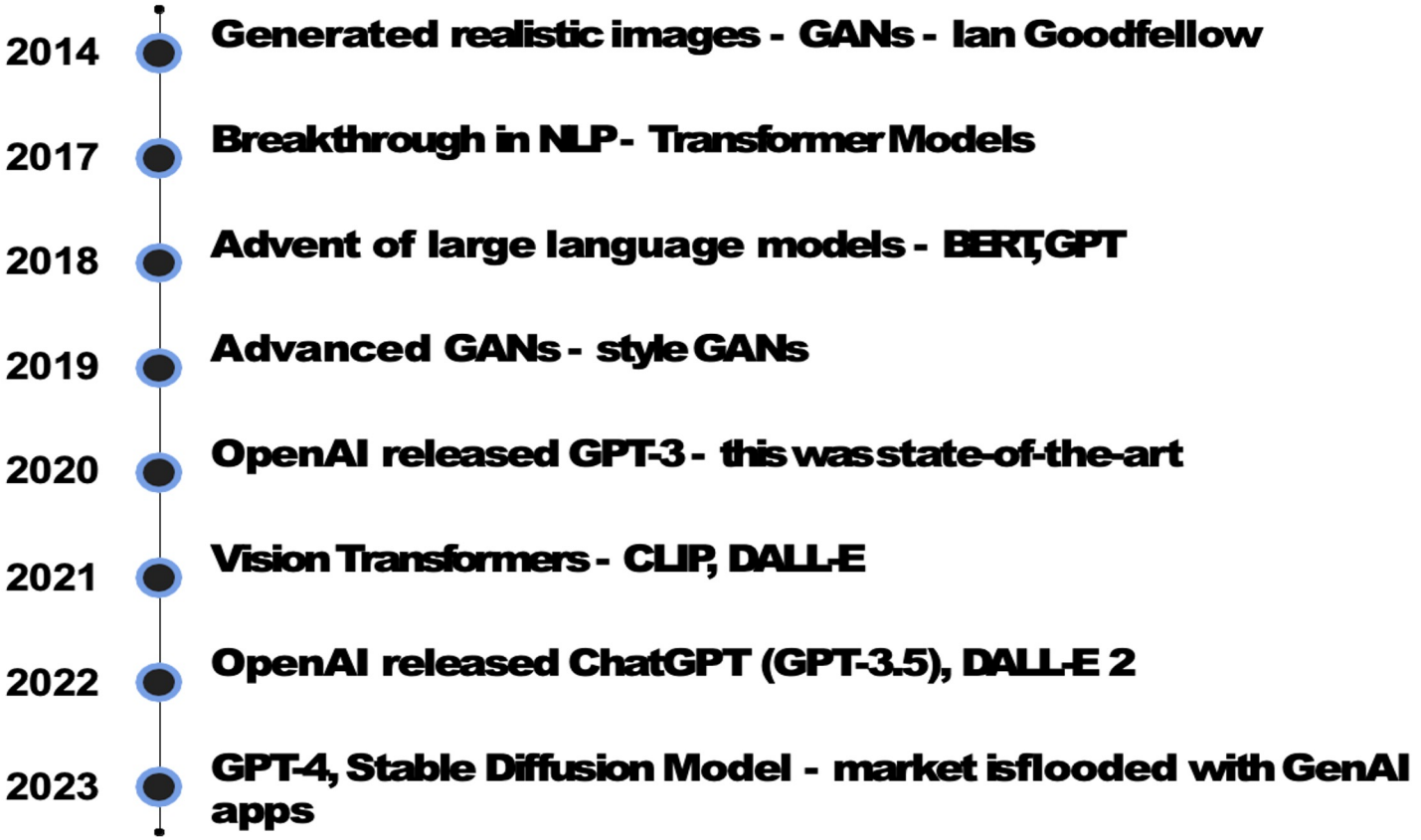
Discriminative AI vs Generative AI



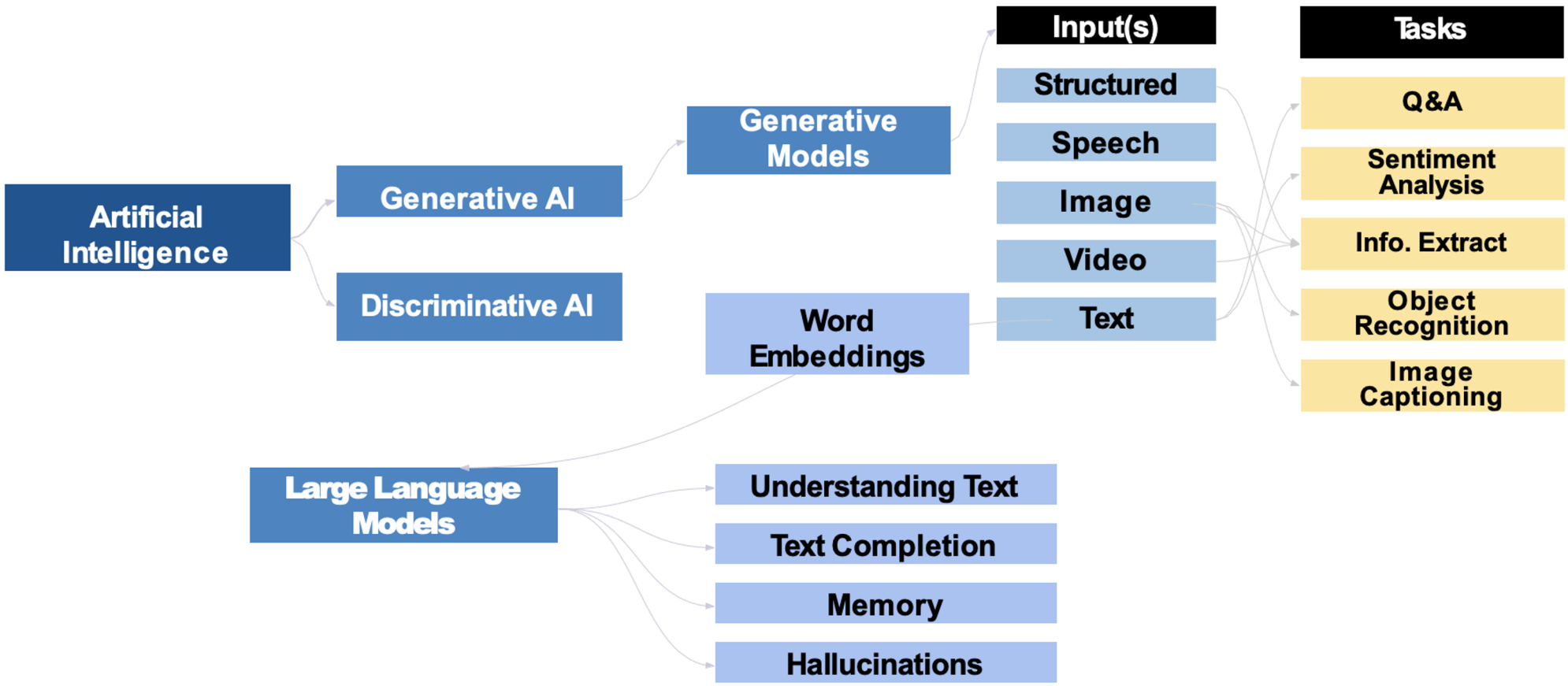
Glimpse into Generative Models



A Brief History of Generative AI



Concept Map

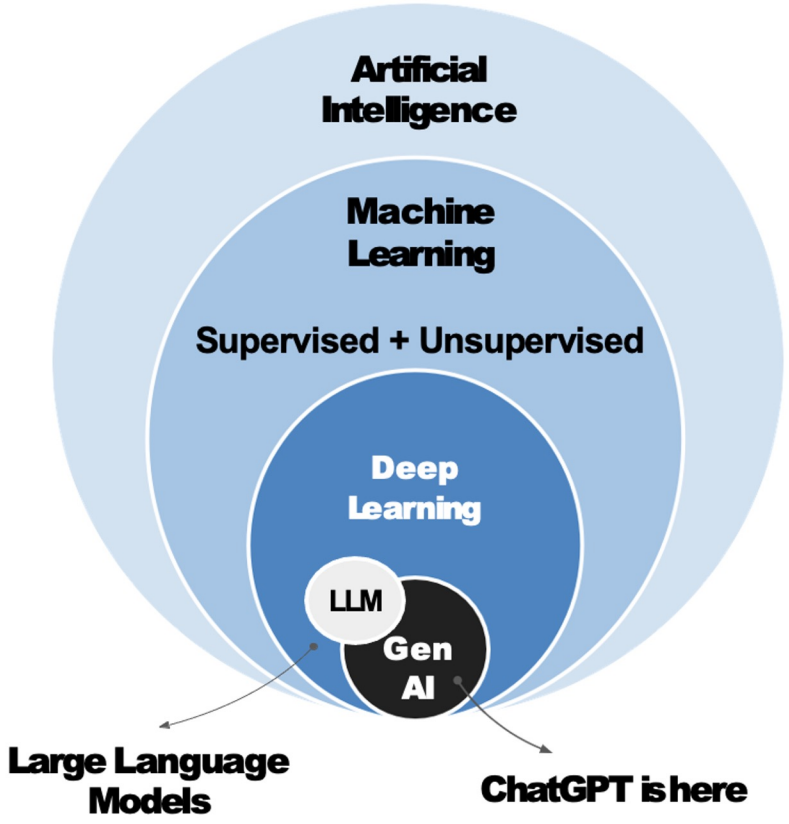


Introduction to Large Language Models (LLMs)

Large Language Models

- Trained on large amounts of raw text in a self-supervised way.
- In self-supervised learning objective/target is automatically computed from the inputs of the model, that is, manual labelling of data is not required
- LLM develops a statistical understanding of the language it has been trained on.
- These models are not very useful for specific practical task.
- For specific practical tasks, LLMs goes through a process called transfer learning. During this process, the model is fine-tuned in supervised way for a particular task/applications). (For example, GPT is LLM and Chat GPT is application)
- To user, LLMs serve as an interface between Deep Learning and Generative AI, enabling human communication with AI models through Natural Language prompts, offering conditional probability-based outputs tailored quests.

LLM & Generative AI



Large Language models (LLMs)

- **Large**, because trained on large data set with billions of trainable parameters
- **Language**, because it deals with text data (takes input in text and generates output in text)
- **Model**, because it predicts the next word/sentence/token
- Hence, LLMs are language models consisting of a neural network with billions of parameters, trained on large quantities of unlabelled text using self-supervised learning.
- LLM exhibits emergent properties such as **zero-shot learning**.

Zero shot learning

- The capability of a (machine learning) model to complete a task, it was **not** explicitly trained on

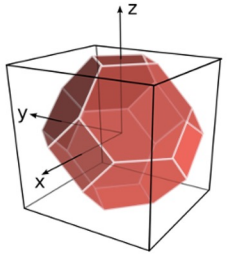
- LLM achieve this because of inherit self-supervised training/learning

How LLMs complete text?

- Leveraging Transformer Architecture

The robot must obey instructions given to _____

Learning from the large set of documents, the model has understood that when the word **robot** is referred to again in a sentence, words like "**it**" / "**its**" are most likely to be the completion word.



It memorized this association by knowing they are close to each other in some vector dimension that associates pronouns with words.

The robot must obey instructions given to it

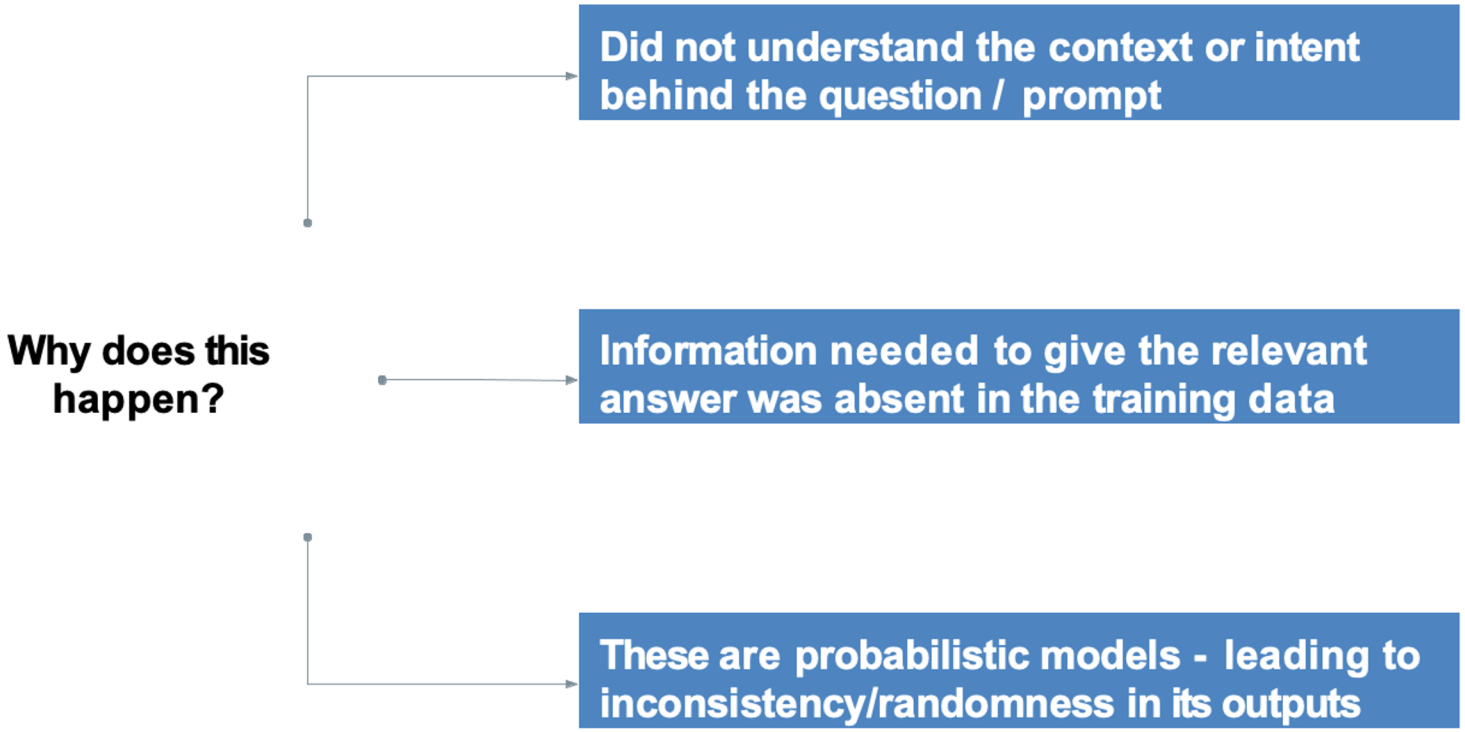
The robot must obey instructions given to its control system or programming

The robot must obey instructions given to its designated operator or user

Limitations/Drawbacks of LLM

- **Hallucinations** : Creation of high-quality text which contains factually incorrect information.
- **High Computation time & cost** : Training an LLM, requires a large amount of data and is costly in terms of time and compute resources, also has environmental impact.
- **Lack of transparency** : LLM's are complete black box and lacks explainability
- **Potential bias** : Data produced by Large Language Models reflects prejudice due to bias or orientation present in the training data
- **Data Security Risk :** The potential for LLMs to handle and produce sensitive information opens the door to exploitation for nefarious activities, such as phishing or the creation of deceptive spam messages etc
- **Data Poisoning** : Language models (LMs) are easily prone to poisoning attacks. It occurs when an attacker manipulates the training data or fine-tuning procedures of an LLM to introduce vulnerabilities, backdoors, or biases that could compromise the model's security, effectiveness, or ethical behaviour.
- **Legality of using LLMs content** : "Who owns creative output of generative AI " is a debatable topic. At present, the Copyright Offices do not consider AI as an author because it is not human. Because there is no author, the raw content of generative AI should be dedicated to the public domain and not claimed by individuals(or groups) as their own.

Why do Language models hallucinate?



Ways to address Hallucination

- Retrieval-augmented generation (RAG)
- Fine-tuning

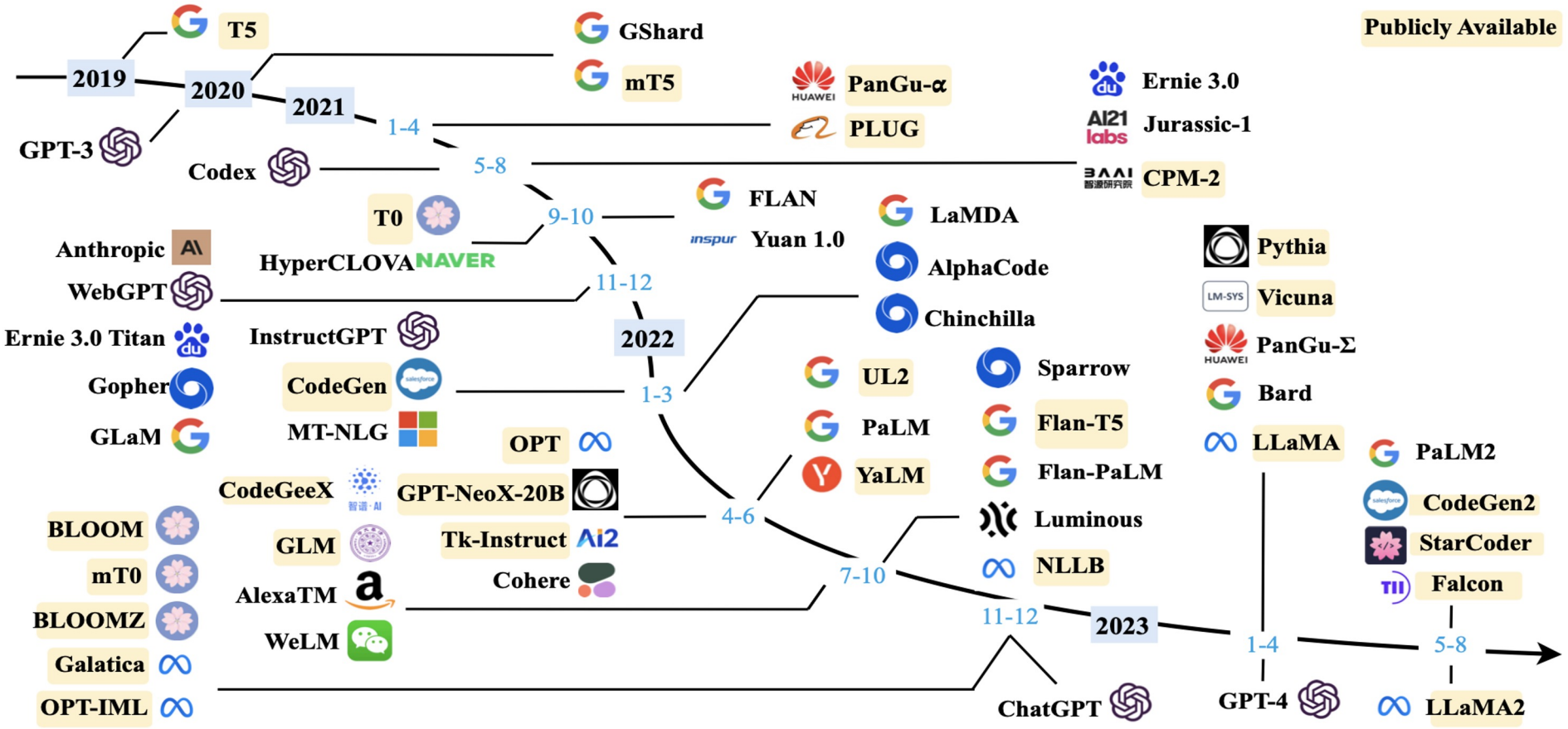
Retrieval-Augmented Generation (RAG)

- Fetch information from a predefined/authenticated source of documents or data, with generative models that can create new content.

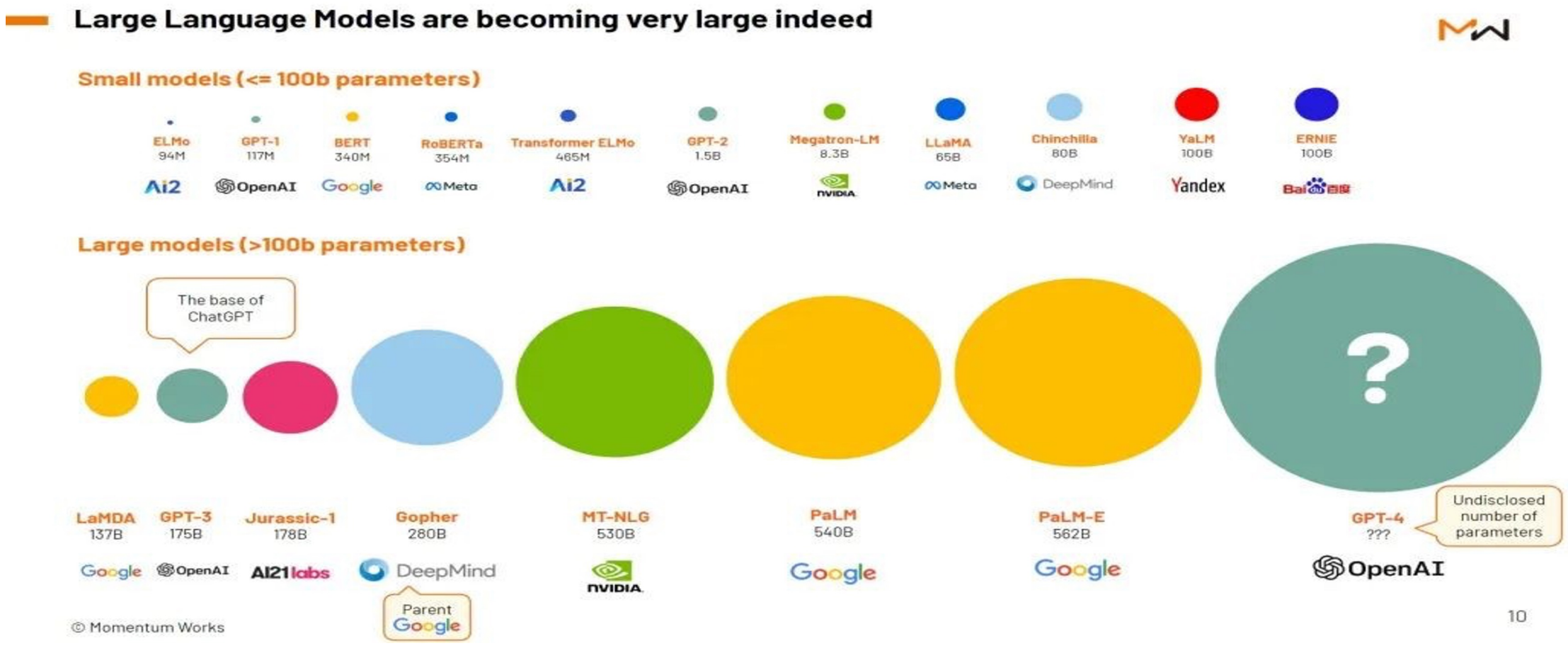
- RAG coupled with real-time data has proven to significantly alleviate Hallucinations.

LLMs : Examples

LLMs Chronology (Size >10b)



LLM Magnitude



A Few Popular LLMs

- GPT-3 (developed by OpenAI) (GPT-3 was trained on 0.5 trillions tokens)
- FLAN-T5 (by Google Research)(opensource)
- Liama-2(by Meta)(opensource) (2T tokens)
- Falcon (3.5 T tokens)

GPT

- Stands for Generative Pre-trained Transformer.
- Large language model created by OpenAI
- GPT models provide text outputs in response to their inputs
- Inputs to GPTs are also referred to as "prompts”
- You can create own application using OpenAI’s GPT API

FLAN-T5

- FLAN-T5, developed by Google Research, is a "Fine-tuned Language Net" (FLAN) with "Text-To-Text Transfer Transformer" (T-5) architecture.
- Basically, it is encoder-decoder model that has been pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format.
- FLAN-T5 excels in various NLP tasks, including translation, classification, and question answering, and it's known for its speed and efficiency.
- During the training phase, FLAN-T5 was fed a large corpus of text data and was trained to predict missing words in an input text via a fill in the blank style objective. This process is repeated multiple times until the model has learned to generate text that is similar to the input data.
- FLAN-T5 can be used to perform a variety of NLP tasks, such as text generation, language translation, sentiment analysis, and text classification.
- Google’s Flan-T5 is available via 5 pre-trained checkpoints:
  - **Flan-T5 small**, **Flan-T5-base**,**Flan-T5-large**, **Flan-T5-XL** ,**Flan-T5 XXL**

LLaMA 2

- Llama 2 is a series of open-source Large Language Models from Meta.

- It comes in three main variants - the 7B, 13B and 70B parameter models - the higher the number of parameters, the more powerful the model. Outside of the base models (which will merely autocomplete text), there are also versions of these models that are fine-tuned for chat - more useful for creating conversational / prompt engineering applications. In the hands-on demonstration for this session, we will be using the Llama 2 13B chat model.

- Llama 2 is considered one of the best open-source alternatives to the GPT series from OpenAI. It was trained on 2 trillion tokens of pre-training data, pushed the state-of-the-art forward on several NLP benchmarks, and even achieves comparable performance to GPT-3.5 on several problem categories. However - to ensure it runs on the free tier of Google Colab (which uses a relatively old Nvidia T4 GPU), we need to quantize the model (compress the values of its weights into 4-bits or 5-bits) to carry out inference. In the faculty notebook, we shall take a look at how this works.

Falcon

- Developed by the UAE's Technology Innovation Institute (TII)
- Is a class of causal decoder-only
- Offers a suite of LLM models, including **Falcon 180B, 40B, 7.5B, and 1.3B,** each tailored to different use cases and requirements.
- Falcon 180B, boasts a staggering 180 billion parameters and has been trained on an extensive dataset of 3.5 trillion tokens.
- Features of Falcon LLM
  - Transparent and Open Source
  - Rich Training Data

Leveraging LLMs

Ways of utilizing LLMs

1. Prompt Engineering
2. Fine tuning the LLM
3. Building fresh/own LLM

Prompt Engineering

- **Prompt Engineering** is the process of **designing and crafting prompts** for Conversational AI interfaces and Generative AI models, in order to get as close as possible to the exact output desired.
- In simpler terms, Prompt Engineering ensures computer/model understands the questions being asked and gives the best answer.
- In Prompt Engineering, LLM is being used without changing any of its model parameters either through,
    1. Web interface : example **Chat GPT** or,
    2. API/Programmatically : example OpenAI API or hugging face libraries
- Drawbacks
    1. Large models size
    2. Good for generic tasks but not so much for domain specific use cases

Prompting Approaches

- **Zero-shot prompting :** language model is tested on unseen task/data (without any fine-tuning or training). The model relies on its pre-trained knowledge to make predictions

- **One-shot prompting** :model is fed a single sample, and its performance is evaluated on a set of similar examples., basically the model is tested on its ability to generalize from a single input and generate a relevant output

- **Few-shot prompting :**language model is fed in with a small number of examples of the task with example input/output pairs. The model uses these examples to understand the structure of the task and prompt and uses it to better adapt to the given input. (Note :Few-shot learning doesn't necessarily involve fine-tuning the model; it's about leveraging the model's pre-trained knowledge to adapt to new tasks with minimal examples.)

Fine-tuning LLMs

- Process of taking a pre-trained model (LLM) and training it further on a specific task or dataset(s), by adjusting/training 1 or more model parameters
- For example, chat GPT which takes a LLM i.e. GPT- 3/3.5 and fine tune it for specific use case(i.e. conversation chat/Virtual assistance)
- Steps to fine tune
    1. Select a base model,
    2. Adjust parameters,
    3. Train the model
- Fine-tuned models are smaller in size, compare to base LLM, and outperforms it for the specific use case
- Finetuning training approaches
    1. Self-Supervised learning
    2. Supervised learning
    3. Reinforcement learning
- Fine-tuning often requires high-processing GPUs. Emerging techniques like Parameter Efficient Fine Tuning tries to minimize resource requirements.

Fine-tuning LLMs Approaches/Techniques

1. **LoRA : Low-Rank Adaptation** reduces the number of parameter updates that need to be done while retaining the knowledge ingested during pretraining. LoRA is one among a family of models that are referred to as Parameter-Efficient Fine Tuning (PEFT) models

1. **QLoRA (Quantized LoRA)**: improves the efficiency of LoRA training further by optimizing few key parameters of the LoRA adapter. Key optimizations executed by QLoRA are:
   - 4-bit NormalFloat quantization that shifts the base Llama2 model from 16-bit to 4-bit
   - Paged optimizers that shift weights to CPU if the GPU RAM is full (instead of throwing up an error)

Building Own LLM : Reasons to build

- Organizations (or individuals) not willing to expose proprietary data to opensource models or aiming for commercial use may prefer building their own LLM.

- Other prominent reasons to build own LLM are ,
  - Customization
  - Reduced dependency
  - Data privacy and security
  - Control over updates and improvement
  - Intellectual property

Building Own LLM : Challenges and Requirements

- **High Resource & Cost and Time requirement:** Building LLMs demands significant financial and temporal investments as well as time. For example, 10 billion parameter model will require approximately 1Lakh GPU hours of training, doing this via renting Nvidia GPU, it would cost around 1.5 crore INR

- **Environmental Impact:** The process of building Large Language Models (LLMs) also has significant environmental implications, as the use of large GPUs during model training contributes to carbon footprints.

Building Own LLM : Phases

1. Data Curation : Getting the data
2. Architect Model : Finalizing the model for training
3. Training at scale : Doing Model-training, using self-supervised learning
4. Evaluation : Validating the model

We will investigate these 4 in detail

Data Curation

- Quality of your model depends upon quality of your data
- From where to fetch data ?
  - From Internet through web scraping : e.g. Wikipedia
  - From Public repositories/datasets : Hugging face Datasets, The Pile, Common crawl
  - From Private data sources : Bloomberg FinPile
  - Data generated from another LLM
- Data preparation checklist
  - Filter : remove low quality data(e.g. racist comments), using either rule base or classifier-model based filtering
  - Eliminating deduplication : avoid keeping identical data in training and validation
  - Privacy redaction: remove sensitive and confidential information (e.g. Aadhaar no)
  - Tokenize : convert text to number using constant technique

Architect Model

- Transformer model (and its variants) Architect is preferred for LLM creation
- Different variants (encoder-decoder, encoder only or decoder only) are used, based on the specific task requirement.
- Based on requirement Model specifications are chosen
  - Model Size and architecture : e.g., context length, number of parameter, data size
  - Training objective : e.g., next token prediction, fill-in-the –middle
  - Attention mechanism : self-attention, (no of) multi-head
  - Embeddings : rotatory, positioning ,etc,.

Training at scale

- Training an LLM requires significant computational resources, often involving distributed computing systems or cloud-based infrastructure. GPUs, TPUs, or cloud-based accelerators are commonly used to provide the necessary processing power.
- Few Tuning/optimization techniques used to for faster training of large datasets,
  - Mixed precision Training
  - 3D Parallelism
  - Zero Redundancy Optimizer (ZeRO)

Training at Scale :Mixed precision Training

- Mixed precision training is a technique used to train deep learning models more efficiently. It involves using both single-precision and half-precision floating-point numbers to represent model parameters and intermediate results. This can significantly reduce the memory footprint and computational cost of training without sacrificing accuracy.
- In single-precision floating-point format, each number is represented using 32 bits. This allows for a wide range of values to be represented, but it also comes at a higher memory cost and computational complexity.
- Half-precision floating-point format, on the other hand, uses only 16 bits per number. This reduces the range of values that can be represented, but it also significantly reduces the memory footprint and computational cost.
- Benefits of using mixed precision training
    1. Reduces memory footprint
    2. Reduces computational cost
    3. Overall performance gets improved

Training at Scale : 3D Parallelism

3D parallelism, also called as hybrid parallelism, improves speed and scalability during training of deep learning models, specially large language models (LLMs), efficiently by combining below three parallelism techniques.

1. **Pipeline parallelism** divides the model into stages and executes each stage on a separate GPU or accelerator. This allows for pipelined processing, where each stage processes a portion of the input data before passing it on to the next stage. This can significantly improve throughput, as the model can process more data in parallel.

1. **Model parallelism** distributes the model's parameters across multiple GPUs or accelerators. This allows for larger models to be trained, as the memory requirements are shared across multiple devices. Additionally, model parallelism can improve training speed, as the computations for each parameter update can be performed in parallel.

1. **Data parallelism** replicates the model and trains it on different batches of data in parallel. This allows for more data to be processed in parallel, which can improve training speed and reduce the time required to train the model.

Training at Scale : Zero Redundancy Optimizer (ZeRO)

- Zero Redundancy Optimizer (ZeRO) is a memory optimization technique, it reduces data redundancy by partitioning the optimizer state, gradients, and parameters across multiple GPUs or accelerators. Thus, significantly reduce the memory footprint of the training process, allowing for larger models to be trained on smaller hardware configurations.

- Optimizer State Partitioning: The optimizer state, which includes information such as momentum and variance estimates, is typically replicated across all GPUs or accelerators. ZeRO partitions this state across multiple devices, allowing each device to store only a portion of the state. This can reduce the memory footprint by up to 4x, without impacting training accuracy.

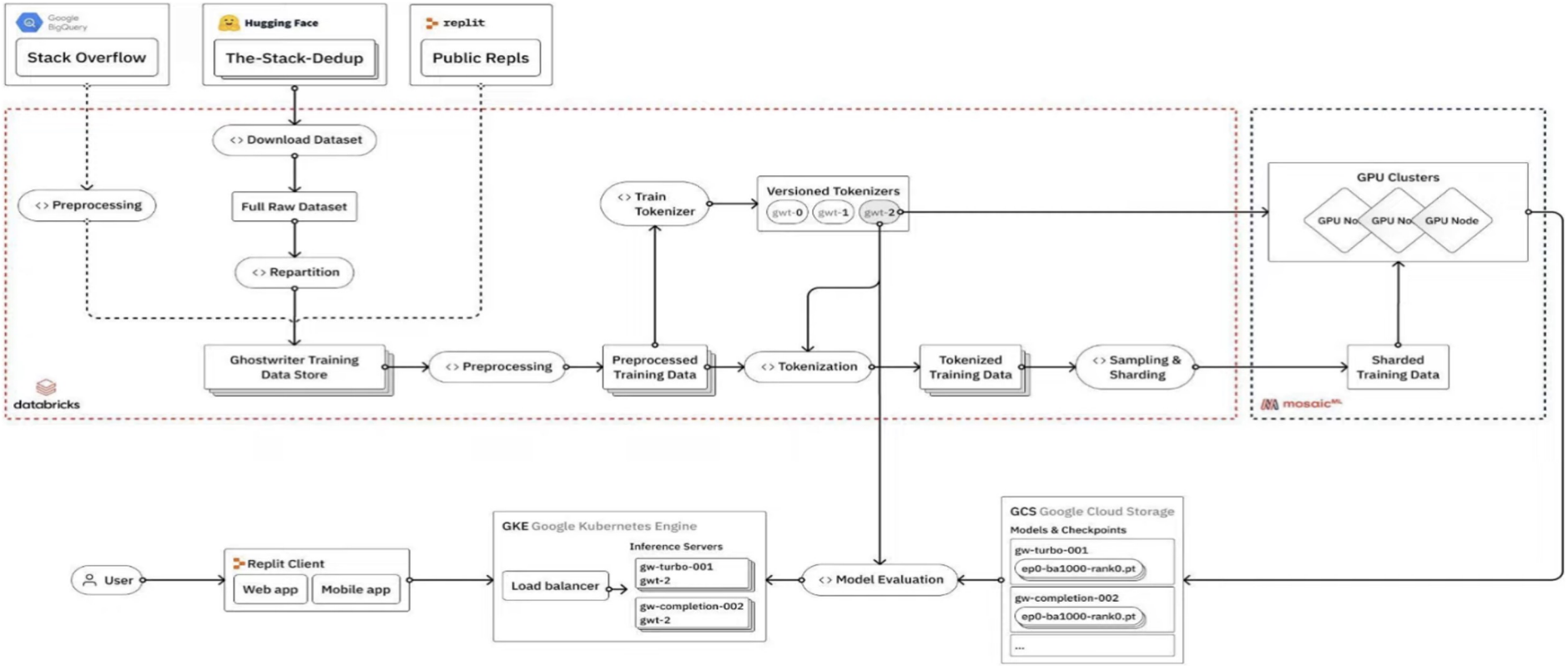
- Gradient Partitioning:Gradients, which represent the error of the model's predictions, are also typically replicated across all GPUs or accelerators. ZeRO partitions gradients across multiple devices, allowing each device to store only a portion of the gradients. This can reduce the memory footprint by up to 8x, without impacting training accuracy.

- Parameter Partitioning: Model parameters, which represent the weights and biases of the model, are typically replicated across all GPUs or accelerators. ZeRO partitions parameters across multiple devices, allowing each device to store only a portion of the parameters. This can reduce the memory footprint by up to 16x, without impacting training accuracy.

Model Evaluation

- Difficult to do correctly (even for model like GPT -4); can be more art than science
- Evaluate against Benchmark dataset (example Open LLM dataset)
- **Open LLM Leaderboard :** Public LLM benchmark, by Hugging Face, gets updated regularly
  - Four benchmark datasets are there :
    1. ARC,
    2. HellaSwag,
    3. MMLU,
    4. TruthfulQA
- High scoring metrics does not necessarily mean a good model
- Open ended task :
  - Human Evaluations,
  - NLP Metrics (BLEU/ROUGH score),
  - Auxiliary Fine-tuned LLM

Building LLM : Sample LLM design architecture



Source :Databricks

LLMs Service providers

- LangChain : An open-source hub for the development of LLM-based applications.
- Fixie: An enterprise-level platform dedicated to the creation, deployment, and management of AI agents.
- Cloud LLM providers: Microsoft's Semantic Kernel (Copilot leverage this) , Google Cloud's Vertex AI platform

LLM Application Development Tools: LangChain

LangChain

LangChain is a versatile framework, developed by Harrison Chase, designed for harnessing the capabilities of Large Language Models (LLMs). Its features include,

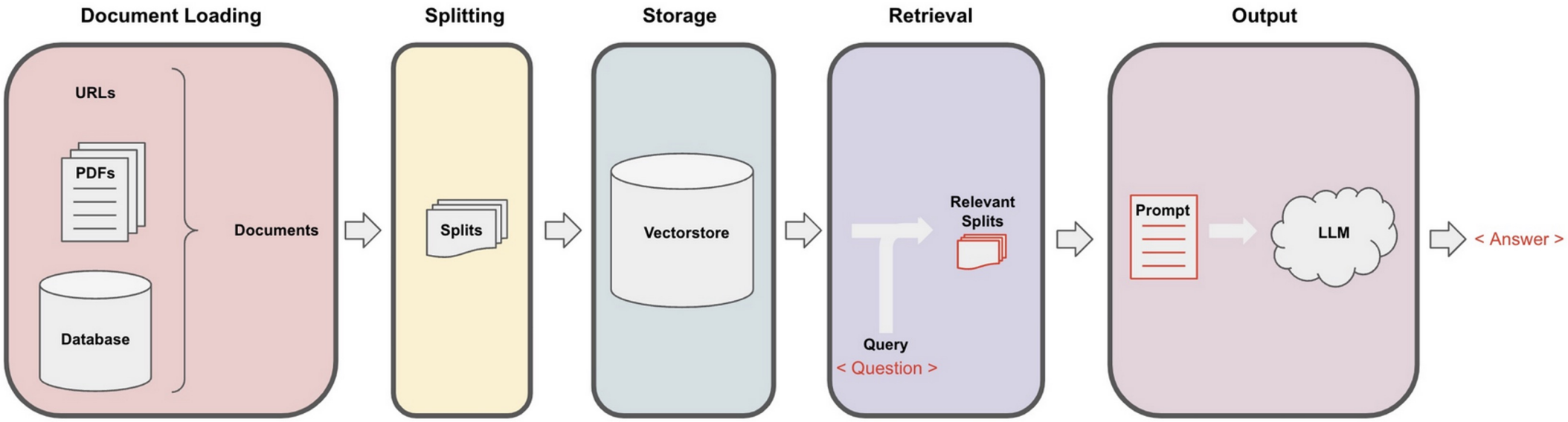
- Robust toolkit for tasks like chatbots, question-answering, and more
- Unique ability to "chain" together various components for sophisticated use cases with LLMs
- Introduces the concept of prompt templates for structured inputs to LLMs for tailored results
- Proficient in handling multiple questions
- Generates text responses for complex queries
- Can incorporate more potent LLMs for even more impressive results
- Offers a plethora of additional features and capabilities
- Indispensable tool for harnessing the power of Large Language Models for a multitude of tasks

The LangChain Pipeline

Pipeline for converting raw unstructured data into a QA chain has following steps:

1. Loading:
2. Splitting
3. Storage:
4. Retrieval:
5. Generation:
6. Conversation (Extension):

The LangChain Pipeline illustration



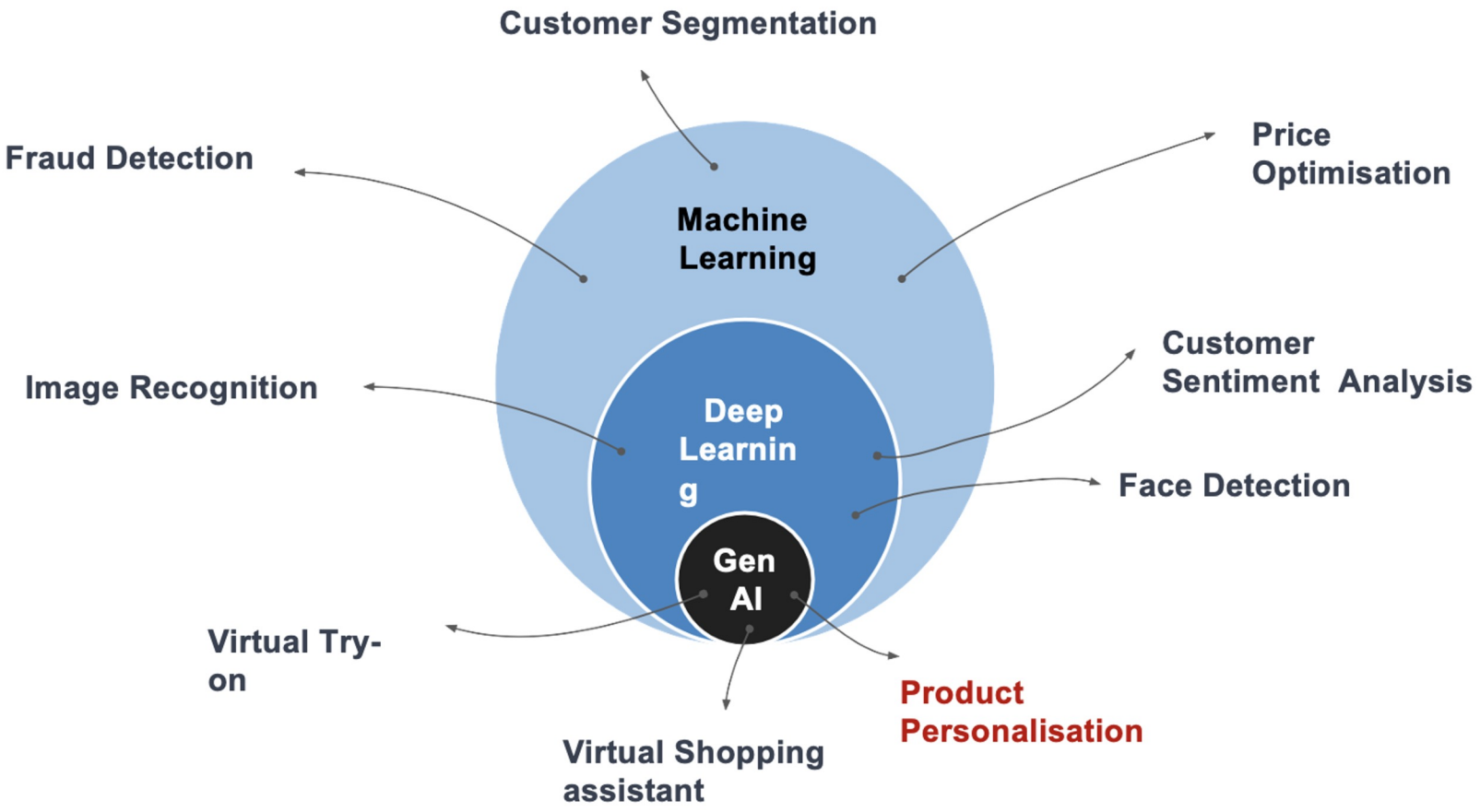
Source : https://js.langchain.com/docs/use_cases/question_answering/

Use Cases of Generative AI

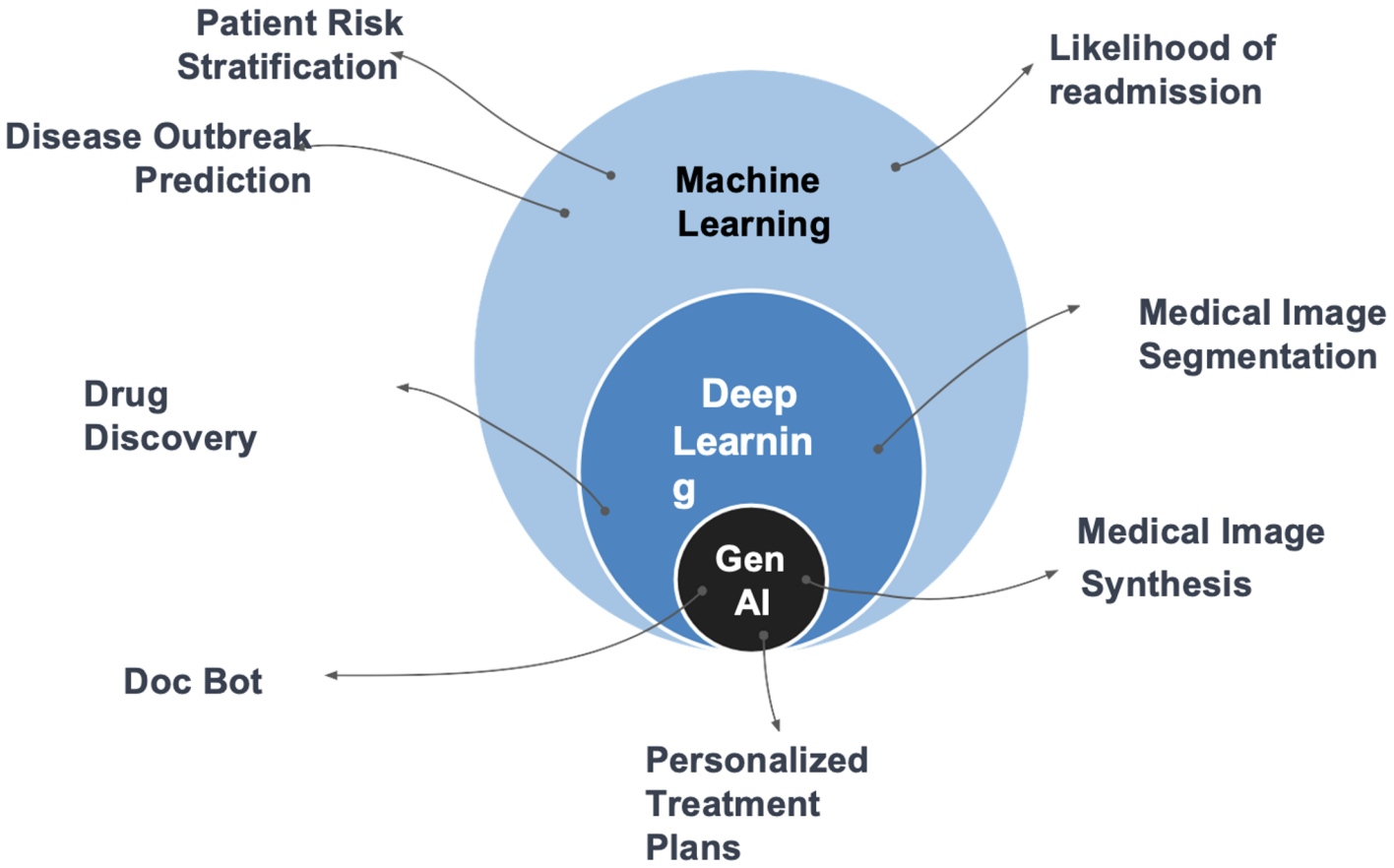
Use cases of Generative AI

- **Helpful for creative industries** of art, music, and fashion, content generation, music compositions etc.
- **Medical field** : Drug discovery and medical imaging, leading to better patient outcomes. New insights into complex diseases and develop innovative treatments that were previously out of reach. Analyzing X-rays or MRI scans, generative AI can assist doctors in identifying potential health issues more quickly and accurately, enabling timely interventions that can ultimately save lives.
- **Personalized Marketing**: Application: Personalized content creation for targeted marketing campaigns.
- **Architectural Rendering**: Application: Assists in architectural design, urban planning, and landscape generation.
- **Scientific Discovery**: Application: Used in simulations and data generation for scientific research.
- **Recommender Systems**: Application: Powers personalized content recommendations and product suggestions.

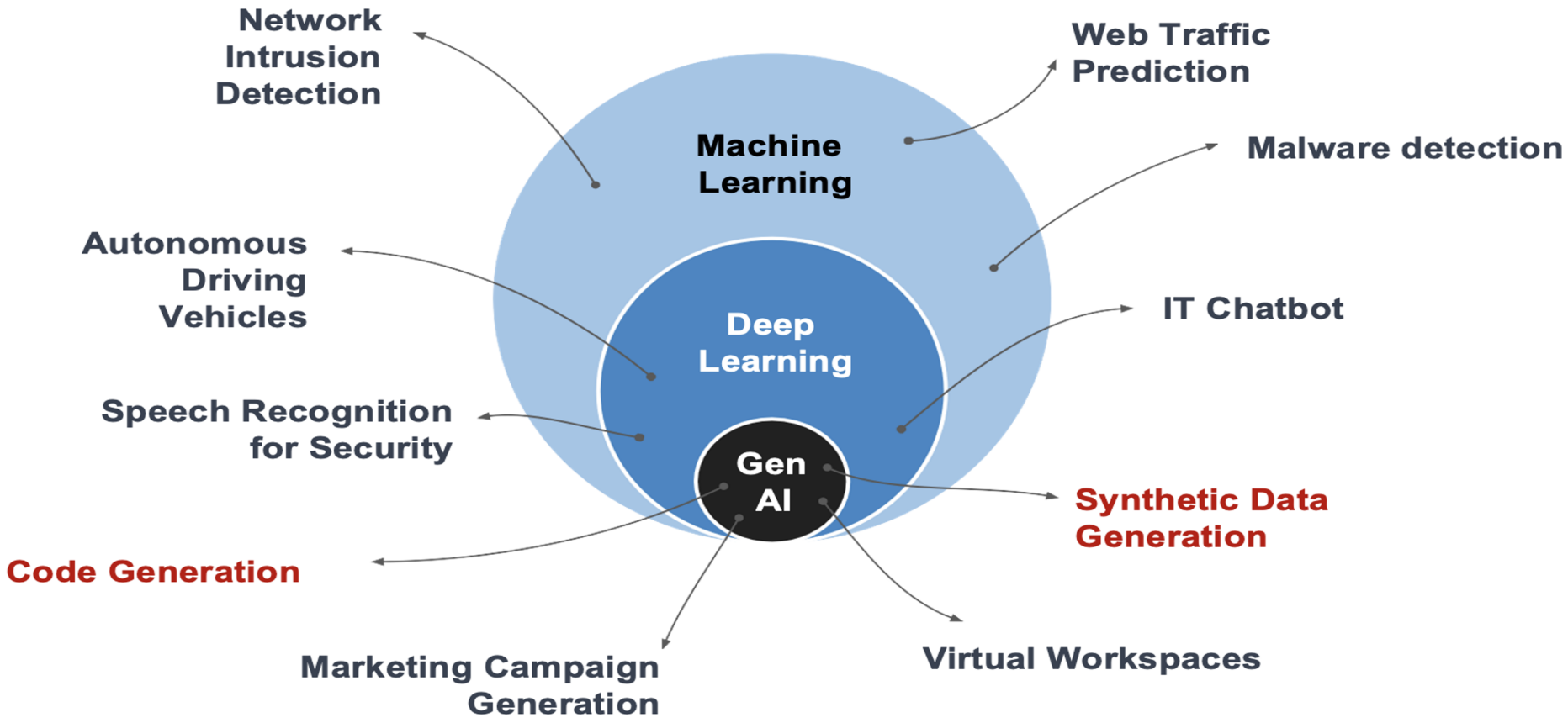
Business Problems solved by GenAI - Retail



Business Problems solved by GenAI - Healthcare



Business Problems solved by GenAI - Tech



Generative AI Ethical Concerns

Generative AI Concerns: a threat or opportunity

- GenAI has the potential to be misused to create harmful or offensive content, such as fake news, propaganda, or deepfakes.
- GenAI can perpetuate biases and discrimination, as it is trained on data that may contain these biases.
- GenAI can raise privacy concerns, as it can be used to generate personalized content that could be used to identify or track individuals.
- GenAI can lead to job displacement, as it can automate tasks that are currently performed by humans.
- GenAI innovations may also have unintended consequences, impacting society in ways not initially anticipated.

Mitigating Generative AI Ethical Concerns

- Develop guidelines and best practices for the ethical development and use of generative AI.
- Educate users about the potential risks and biases of generative AI.
- Implement mechanisms for detecting and removing harmful or offensive content generated by generative AI.
- Invest in research to develop methods for mitigating the biases and discrimination of generative AI.
- Encourage collaboration between the AI community, policymakers, and the public to address the ethical concerns of generative AI.