Decentralizing Intelligence:

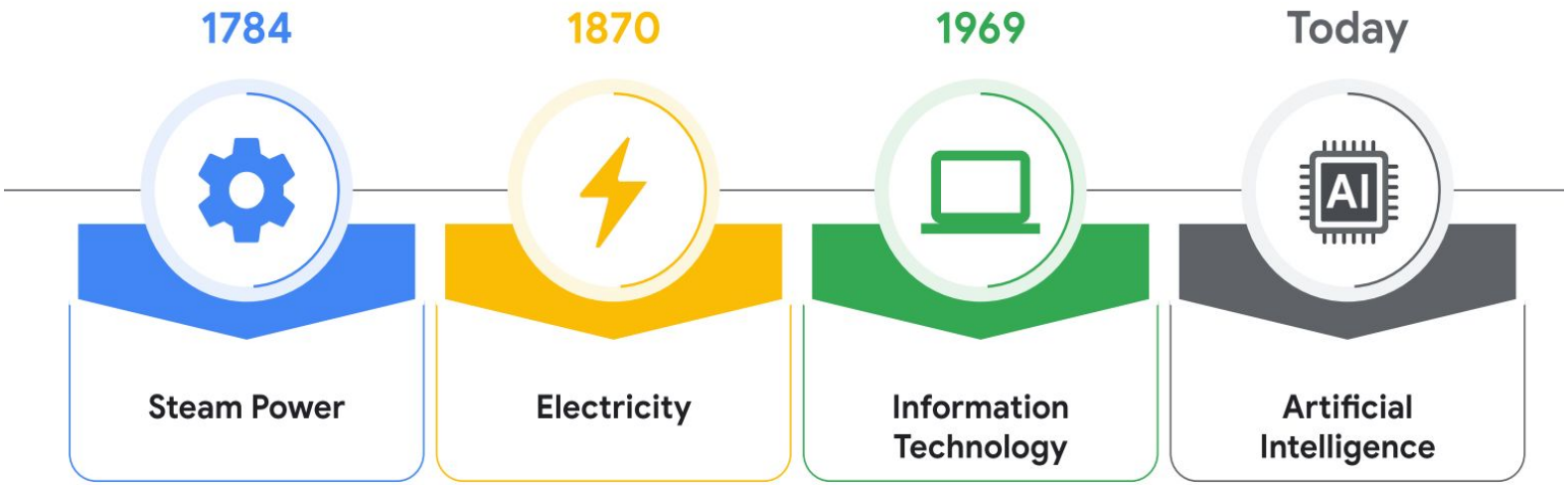
How Edge Computing and Federated Learning Empower Generative AI

Prof. Annappa B
Professor, Dept of CSE.
National Institute of Technology Karnataka,
Surathkal

Outline

- Recap - AI,ML and Gen AI
- Introduction to Decentralized AI
- Edge Computing Overview
- Federated Learning Overview
- Generative AI in a Decentralized Framework
- Research Problems

AL and ML : Recap

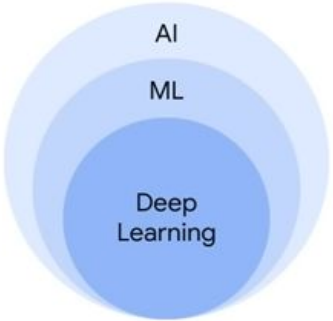


Source: AI: Recent Trends and Applications, Emerging Communication and Computing

AL and ML : Recap

Artificial Intelligence

is a discipline



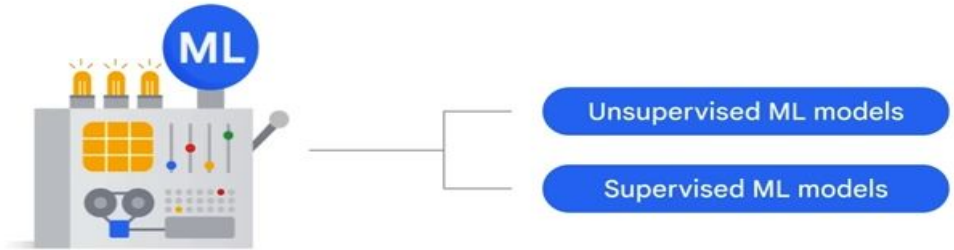
Machine Learning

is a subfield

AL and ML : Recap

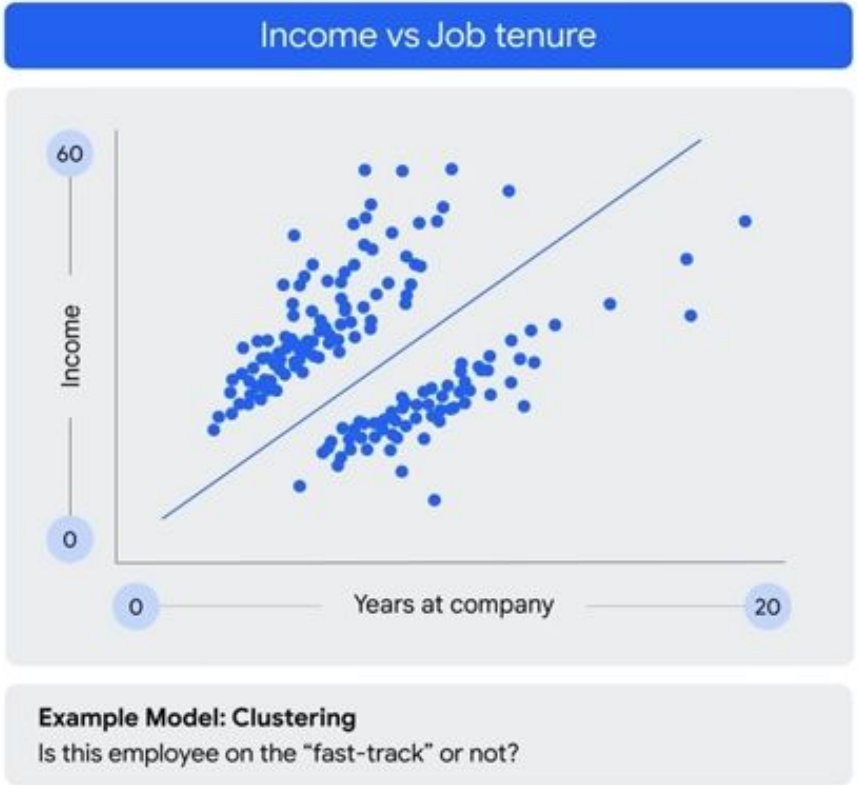
ML gives computers the ability to
learn without explicit programming.

AL and ML : Recap



Unsupervised learning implies the data is not labeled

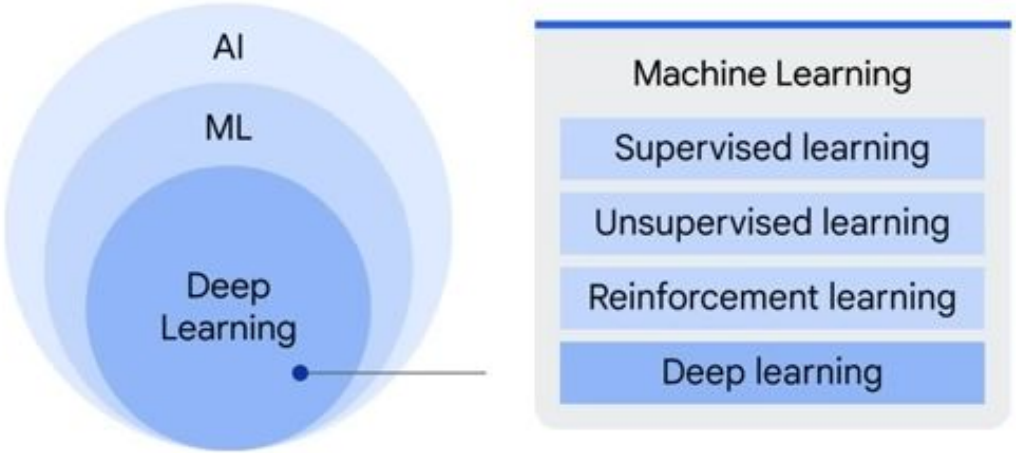
Unsupervised problems are all about looking at the raw data, and seeing if it naturally falls into groups



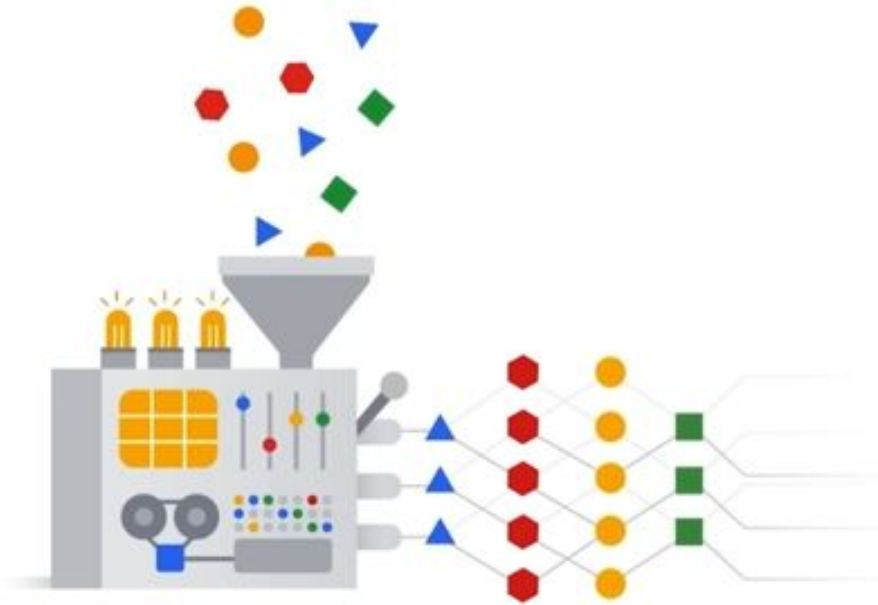
Supervised learning implies the data is already labeled

In supervised learning we are learning from past examples to predict future values.

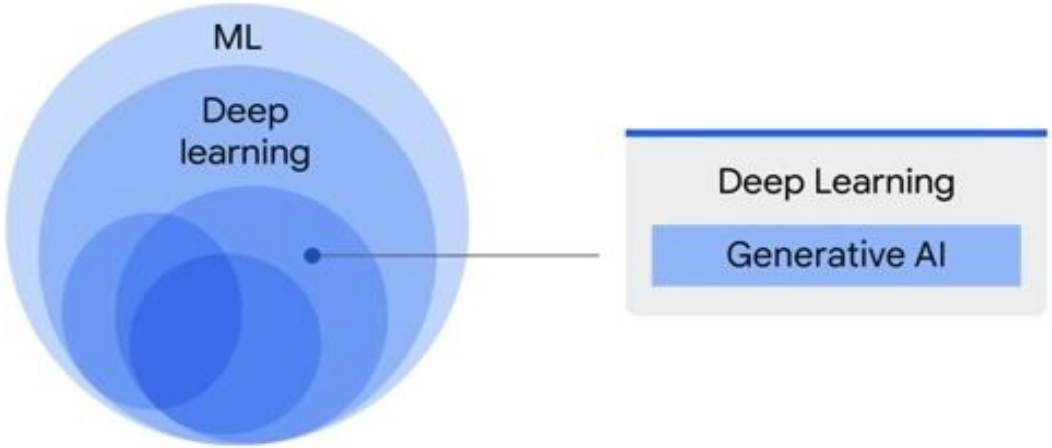
Deep learning is a subset of ML



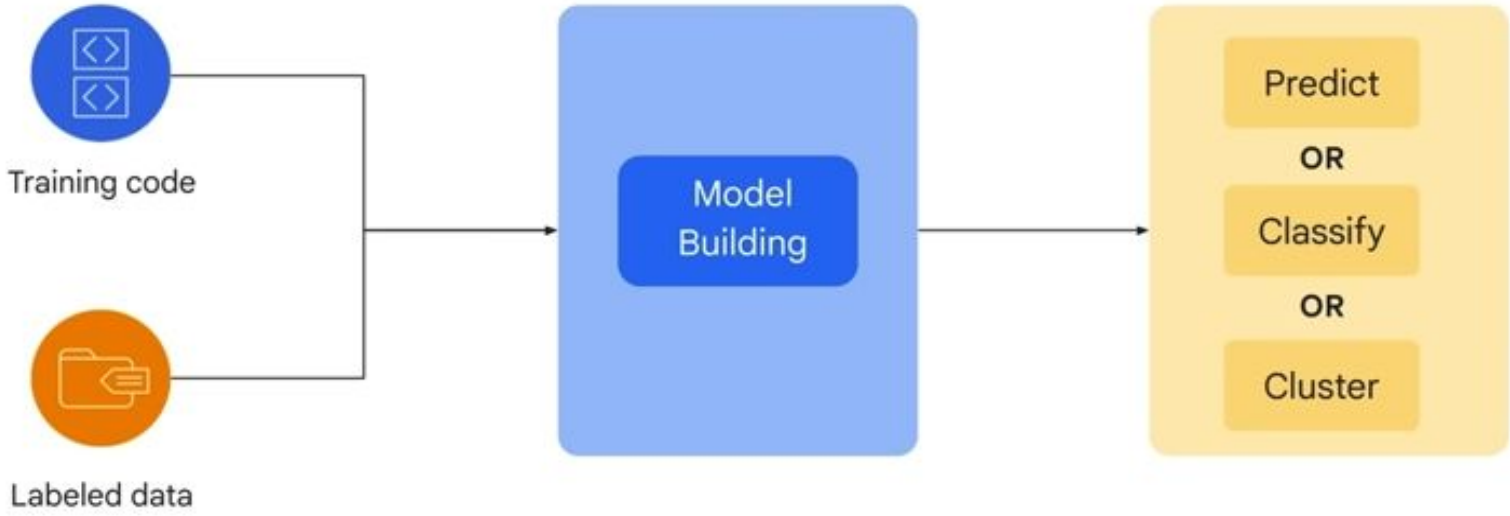
Deep learning uses Artificial Neural Networks - allowing them to process more complex patterns than traditional machine learning.



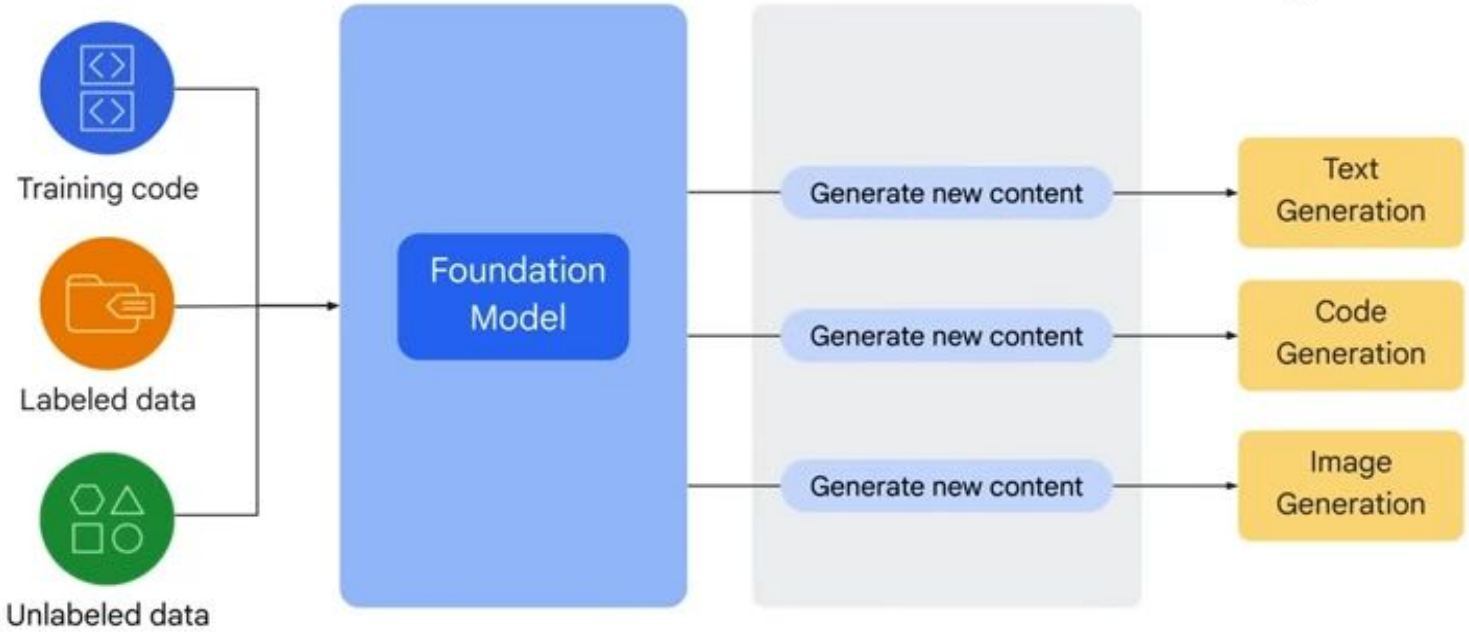
Generative AI is a subset of Deep Learning



Classical Supervised & Unsupervised Learning



Gen AI Supervised, Semi-Supervised & Unsupervised Learning



What is Generative AI?

- GenAI is a type of Artificial Intelligence that creates new content based on what it has learned from existing content.
- The process of learning from existing content is called training and results in the creation of a statistical model.
- When given a prompt, GenAI uses this statistical model to predict what an expected response might be–and this generates new content.

Generative Models

Generative language models

Generative language models learn about patterns in language through training data.

Then, given some text, they predict what comes next.

Generative image models

Generative image models produce new images using techniques like diffusion.

Then, given a prompt or related imagery, they transform random noise into images or generate images from prompts.

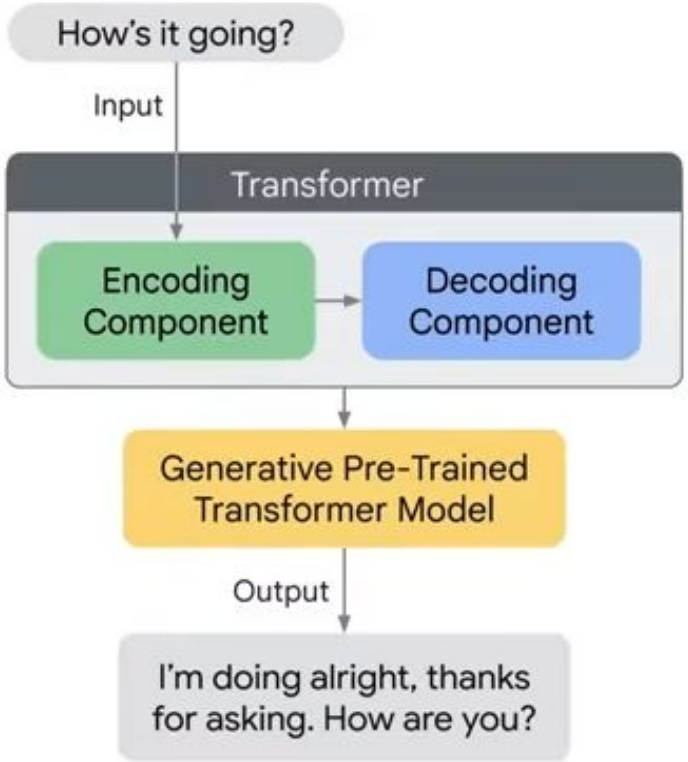
Generative language models learn about patterns in language through training data.

Then, given some text, they predict what comes next.

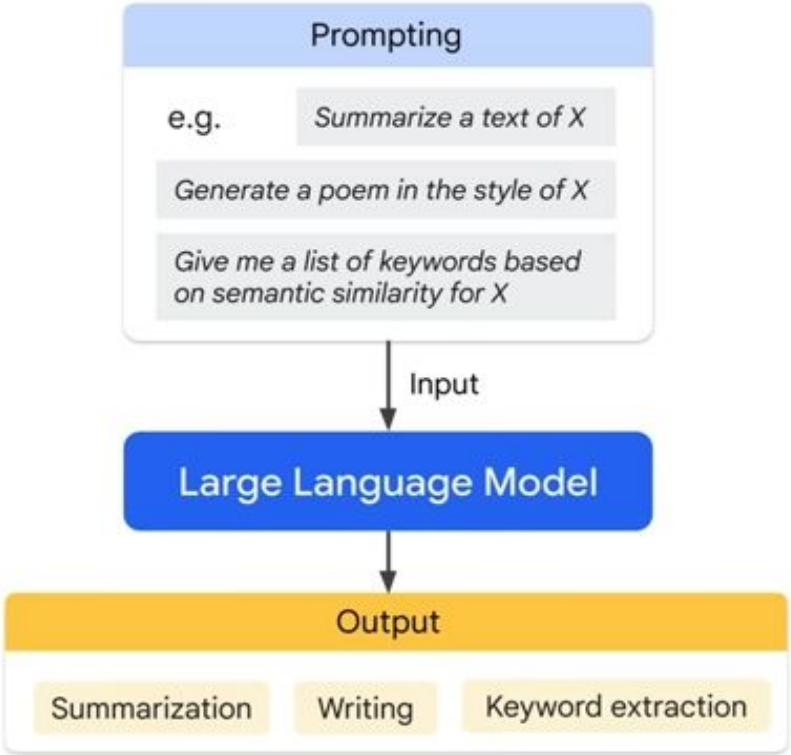
How it Works

Pre-Training:

- Large amount of Data
- Billions of parameters
- Unsupervised learning



Prompt Design:
the quality of the input determines the quality of the output.



Forming a Database

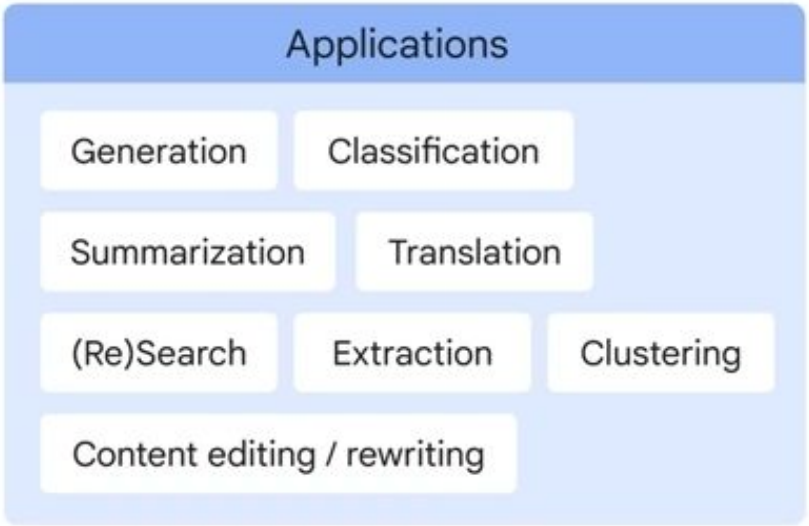
Inputting a Prompt

Generating content

Model Types

text-to-text

Text-to-text models take a natural language input and produce text output. These models are trained to learn the mapping between a pair of texts (e.g. translation from one language to another).



Model Types

text-to-image

Text-to-image models are relatively new and are trained on a large set of images, each captioned with a short text description. Diffusion is one method used to achieve this.

Applications

- Image generation
- Image editing

Model Types

text-to-video

text-to-3D

Text-to-video models aim to generate a video representation from text input. The input text can be anything from a single sentence to a full script, and the output is a video that corresponds to the input text. Similarly Text-to-3D models generate three-dimensional objects that correspond to a user's text description (for use in games or other 3D worlds).

Applications

- Video generation
- Video editing
- Game assets

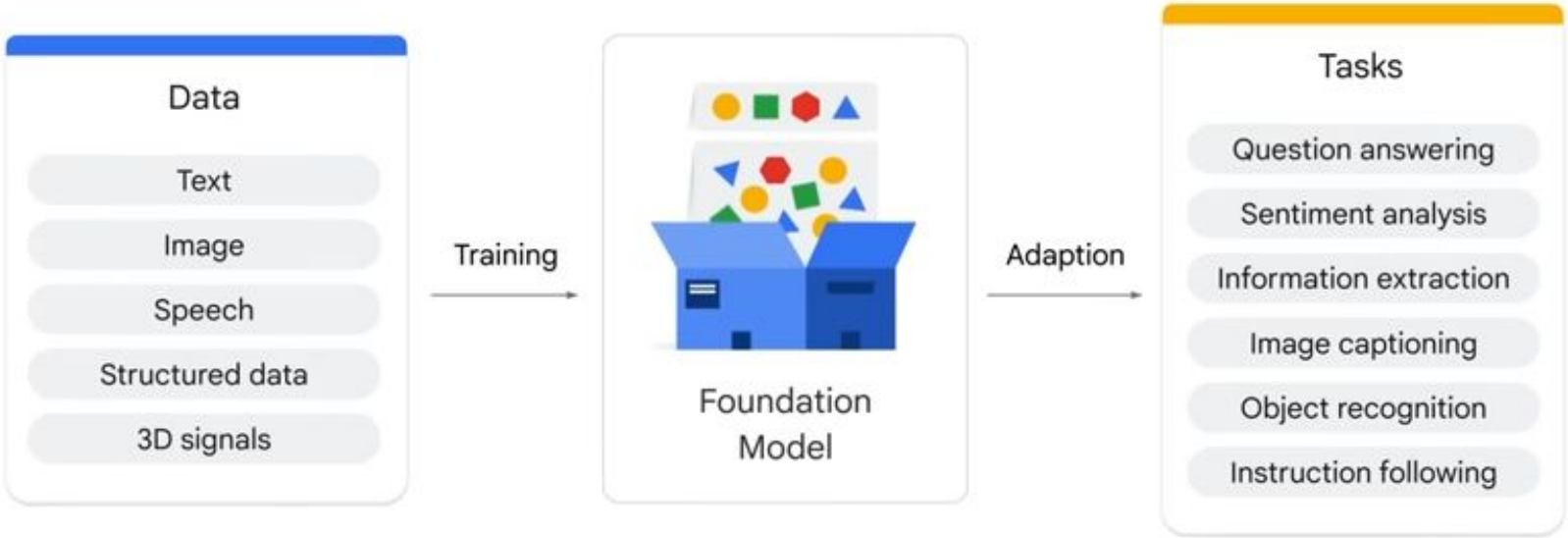
Model Types

text-to-task

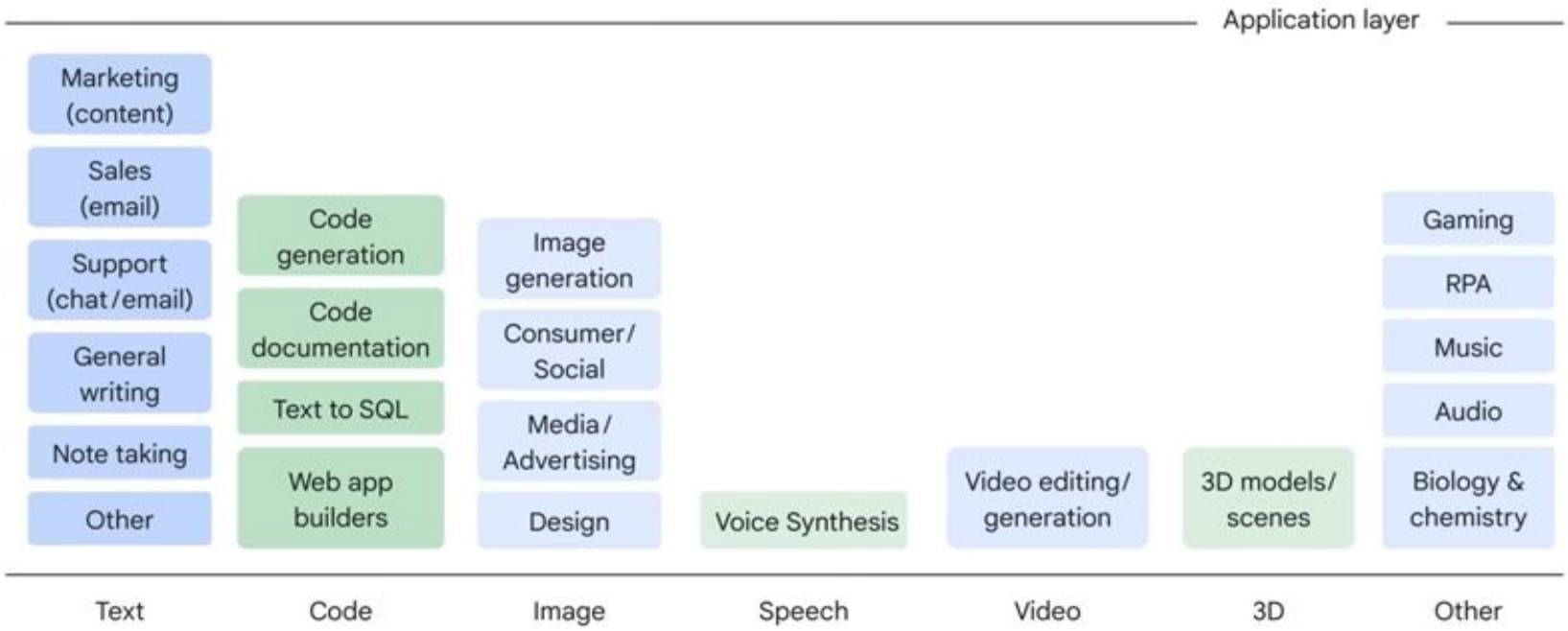
Text-to-task models are trained to perform a specific task or action based on text input. This task can be a wide range of actions such as answering a question, performing a search, making a prediction, or taking some sort of action. For example, a text-to-task model could be trained to navigate web UI or make changes to a doc through the GUI.

Applications

- Software agents
- Virtual assistants
- Automation


Data
Text
Image
Speech
Structured data
3D signals
Training
Foundation Model
Adaption
Tasks
Question answering
Sentiment analysis
Information extraction
Image captioning
Object recognition
Instruction following

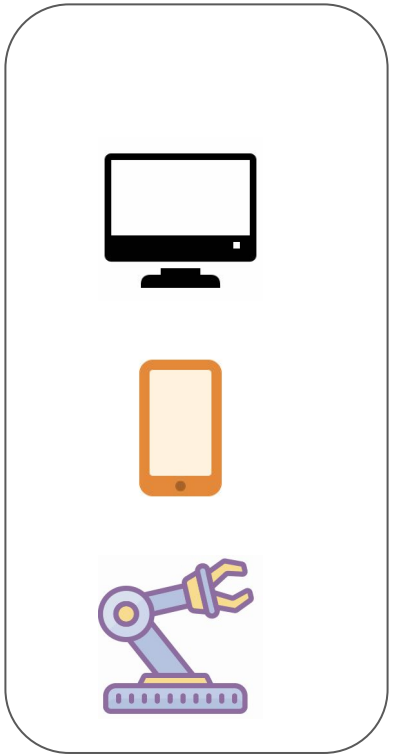
The generative AI Application Landscape



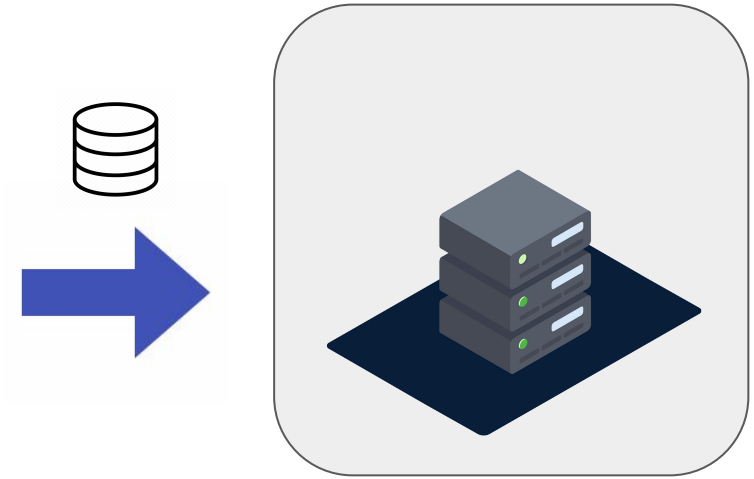
Decentralized AI

The future of AI development, the power and necessity

How Traditional AI works?



Edge Devices



Central Server

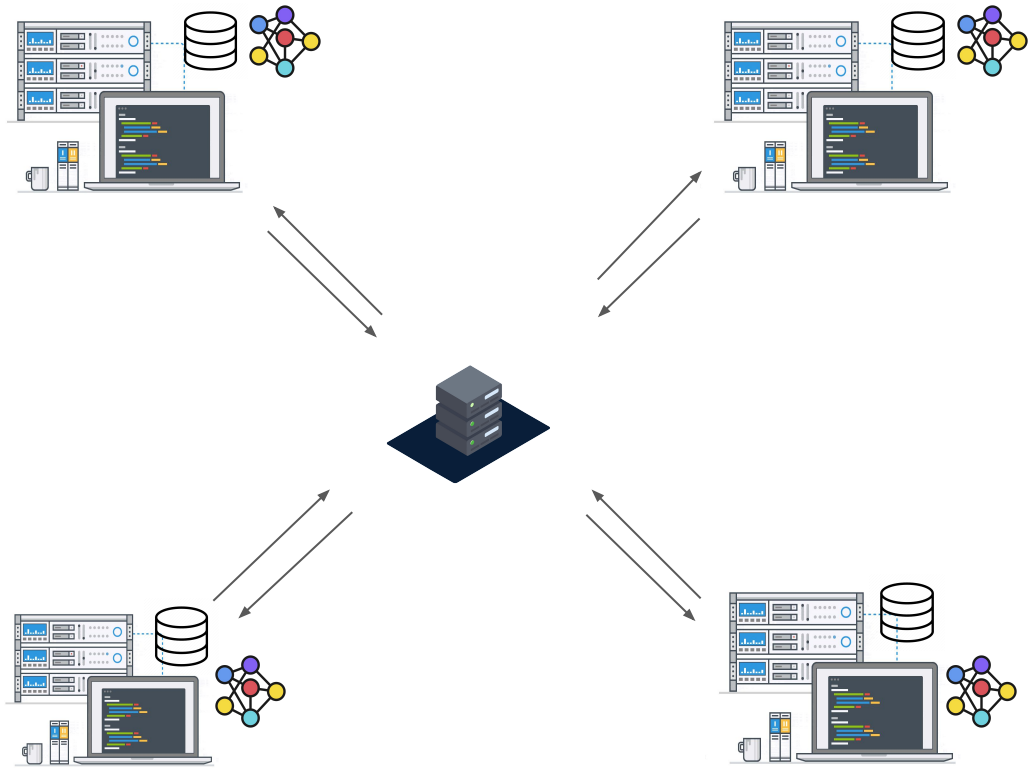
- Need to send all data to central server for building AI model
- Then what about data privacy?

Limitations of Centralized AI

Centralization Issues:

1. **Privacy Concerns**: Sensitive data is stored on central servers, increasing vulnerability to breaches.
2. **Latency Issues**: Delays due to data transmission from devices to a central server, especially in time-critical applications.
3. **Bandwidth Constraints**: Continuous data uploads strain network bandwidth.
4. **Resource Limitations**: Heavy reliance on centralized servers leads to bottlenecks, making scaling difficult.

What is Distributed AI ?



- Distributed AI allow direct raw data communication.
- Utilises a single server or a cluster in a single region, which belongs to a single organisation.

What is Decentralization in AI?

Decentralized AI refers to the distribution of data processing and model training across multiple devices or nodes instead of relying on a single central server.

Key Characteristics:

- Data remains at the source (e.g., on devices or edge servers).
- Model updates are shared, not the data itself.
- Emphasis on privacy, autonomy, and local processing.

Why Decentralization?

Core Benefits:

1. **Privacy Preservation**: Data stays with the user, protecting sensitive information.
2. **Reduced Latency**: Processing close to the data source minimizes delays, essential for real-time applications.
3. **Scalability**: Distributes computational load across devices, reducing dependence on centralized resources.
4. **Energy Efficiency**: Minimizes data transfer costs and uses local device processing.

Decentralized AI v/s Centralized AI

Understanding the Core Differences

Decentralized AI

V/S

Centralized AI

Centralized vs. Decentralized AI - A Comparison

Aspect	Centralized AI	Decentralized (Distributed) AI
Data Storage	Data stored and processed on a central server.	Data remains on local devices or is distributed across nodes.
Data Privacy	Moderate to low; relies on server-based security.	High; data stays locally, reducing exposure risks.
Latency	Higher due to data transfer to and from the central server.	Lower; data processed closer to the source.
Bandwidth Usage	High; continuous data transfer required.	Lower; only model updates or insights shared.
Scalability	Limited by server capacity and bandwidth.	Highly scalable; scales with number of nodes/devices.

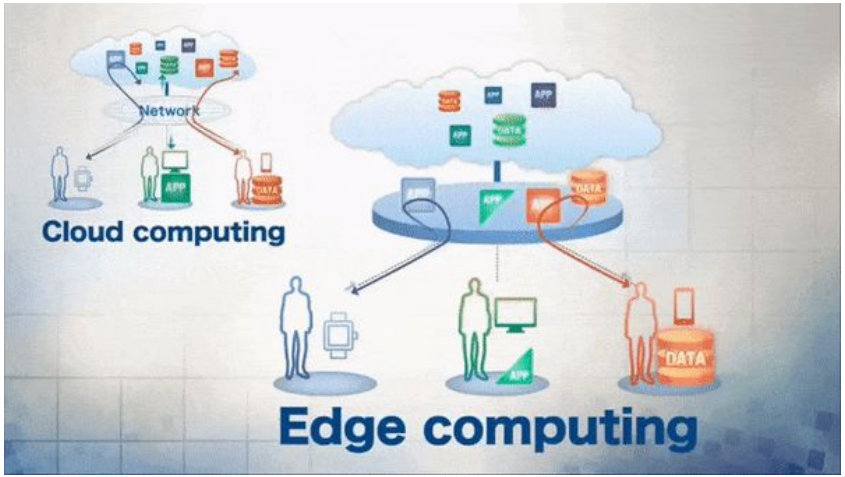
Centralized vs. Decentralized AI - A Comparison

Aspect	Centralized AI	Decentralized (Distributed) AI
Fault Tolerance	Low; single point of failure at central server.	High; failure of one node does not impact the entire system.
Control and Autonomy	Centralized control; server has full authority.	Distributed control; devices or nodes have local autonomy.
Security Risks	Vulnerable to breaches at a single point.	Enhanced security; data is distributed, reducing attack vectors.
Energy Efficiency	Higher energy consumption in centralized setup.	More efficient; energy distributed across devices.
Real-Time Processing	Limited by need for data transmission.	High; enables real-time processing on local devices.

Key Components of Decentralized AI

Core Technologies:

- **Edge Computing**: Performs data processing at or near data source.
- **Federated Learning**: Allows collaborative model training on decentralized data without direct data sharing.
- **Blockchain (optional)**: Secures decentralized interactions and supports peer-to-peer trust.



Real-World Applications of Decentralized AI

Example Use Cases:

1. **Healthcare**: Collaborative training on patient data across hospitals while maintaining privacy (e.g., diagnostics, imaging).
2. **Smart Cities**: Traffic pattern analysis and public safety insights with data processed on edge devices.
3. **IoT Devices**: Personalization and data-driven insights on devices (e.g., smart homes, wearables).
4. **Autonomous Vehicles**: Real-time data processing for decision-making with reduced reliance on cloud.

Summary and Next Steps

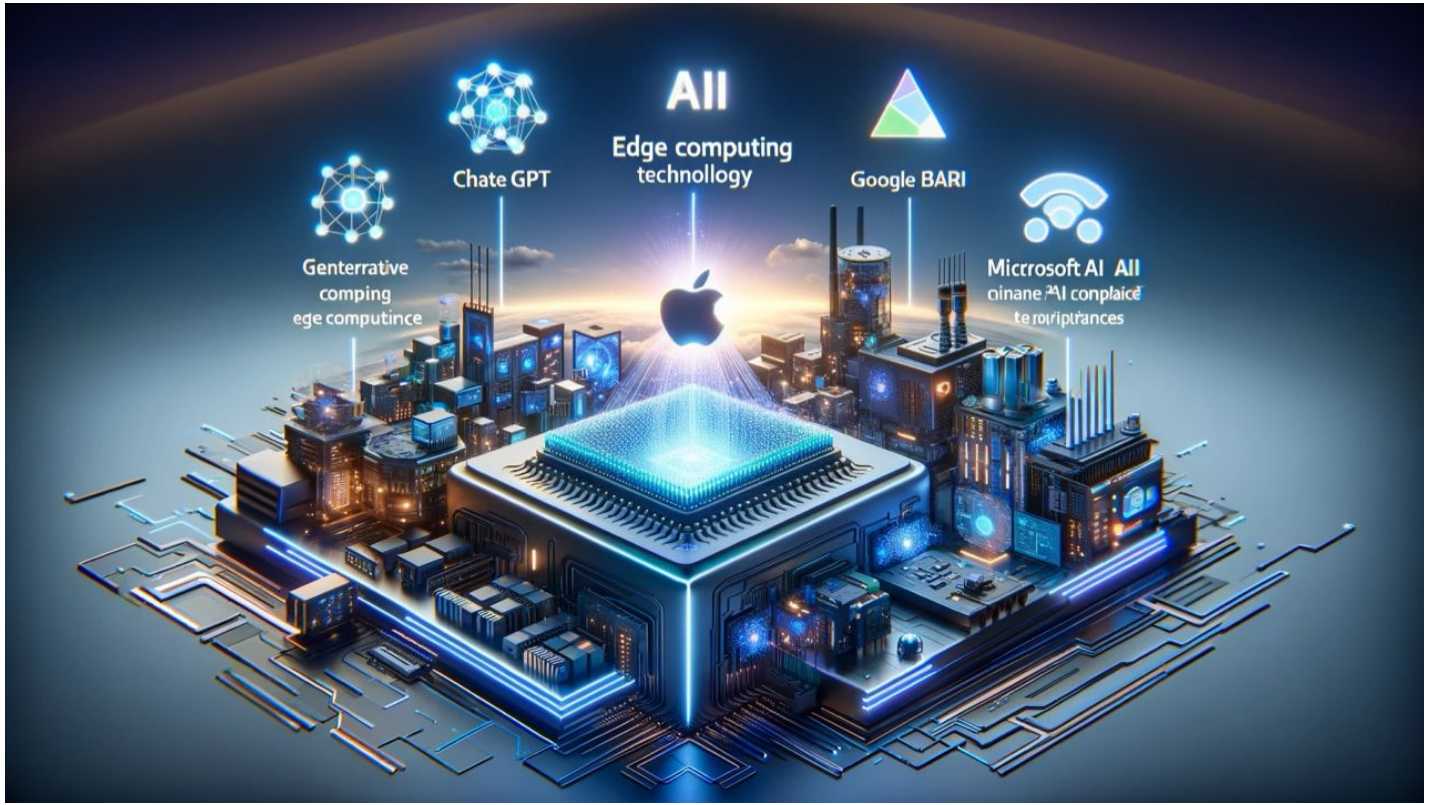
Key Takeaways:

- Decentralized AI mitigates privacy, latency, and scalability issues of centralized models.
- Edge computing and federated learning are foundational to decentralized AI applications.
- Decentralized models support emerging needs in real-time, privacy-conscious applications.

Next Topic:

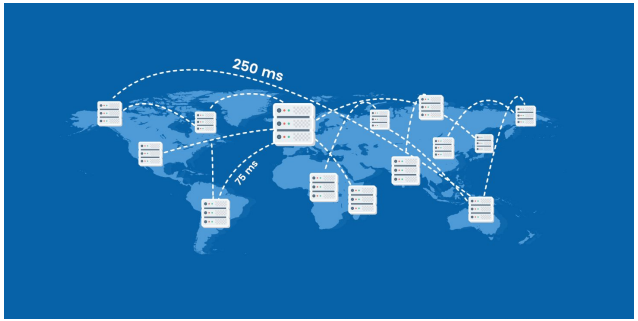
- Deep dive into Edge Computing as a cornerstone of Decentralized AI.

Edge Computing: GenAI Perspective



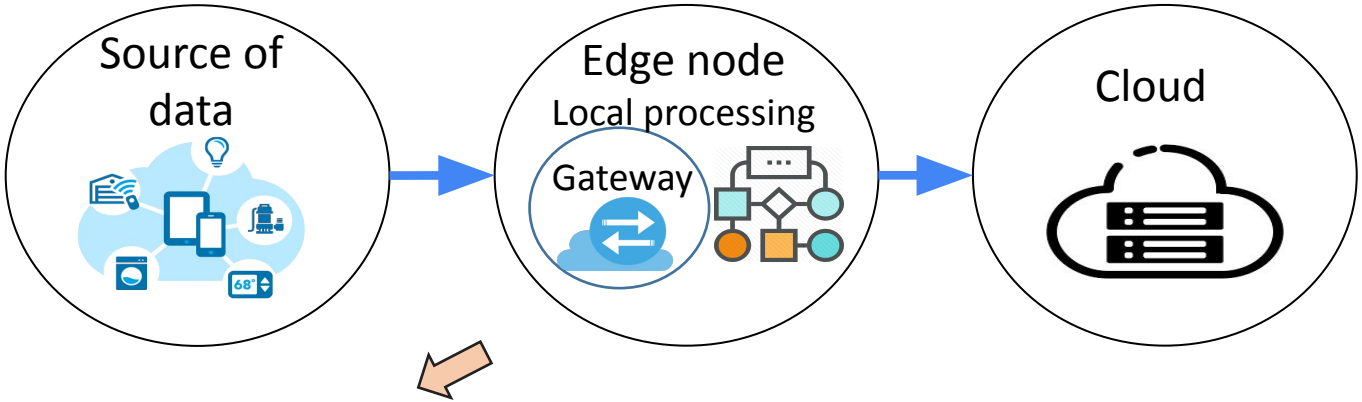
Why Edge Computing ?

- Application That Require Autonomy
- Latency to Cloud Servers
- Bandwidth load
- Slow response speed



Edge computing

Allows data to be processed at the edge node before it's sent to the cloud

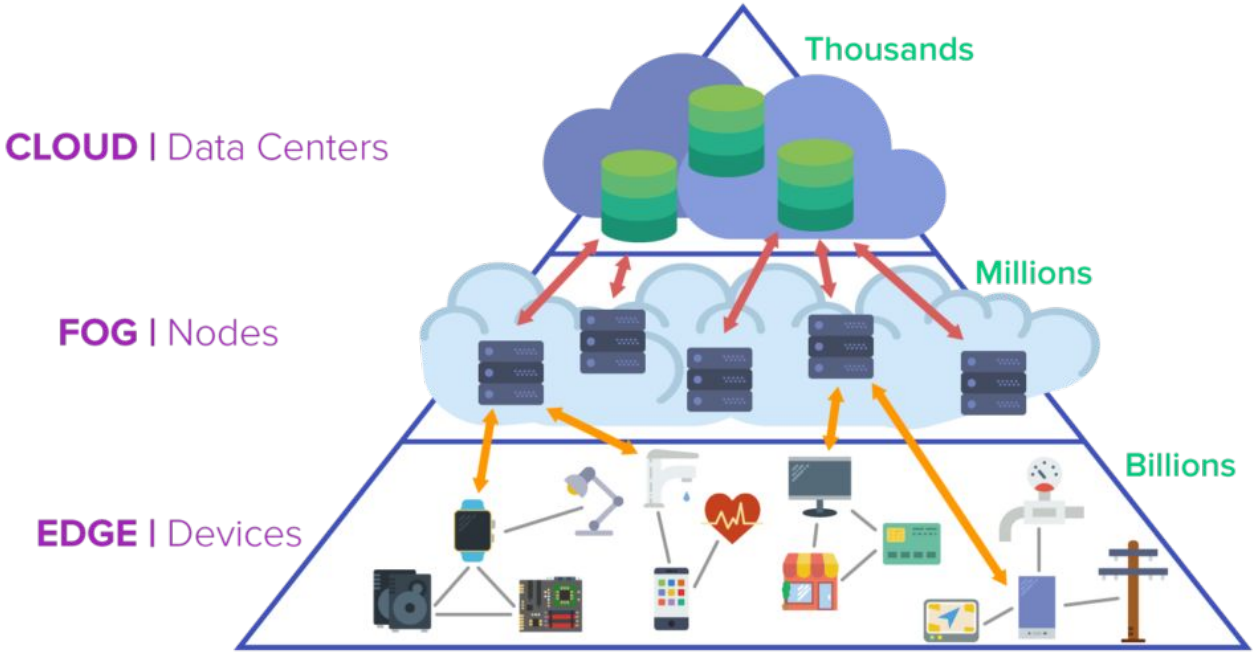


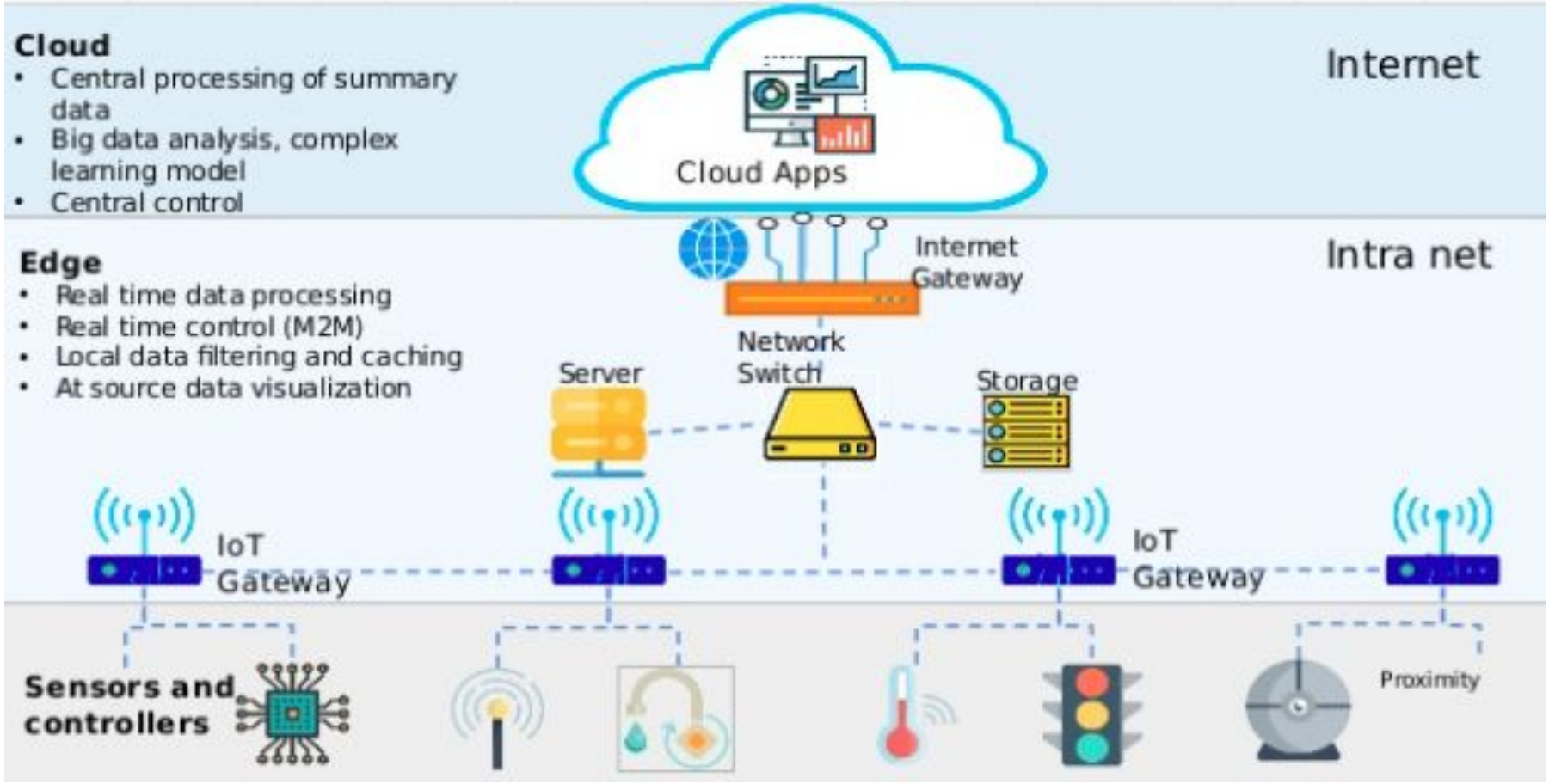
Opportunities:

- Providing more computing resources
- Saving bandwidth

Fog and edge computing are **both extensions of cloud networks**, which are a collection of servers comprising a distributed network. ... The increased distribution of data processing and storage made possible by these systems reduces network traffic, thus improving operational efficiency

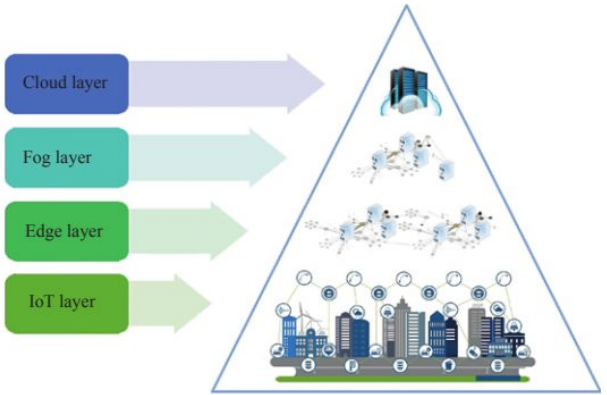
Cloud vs Fog vs Edge



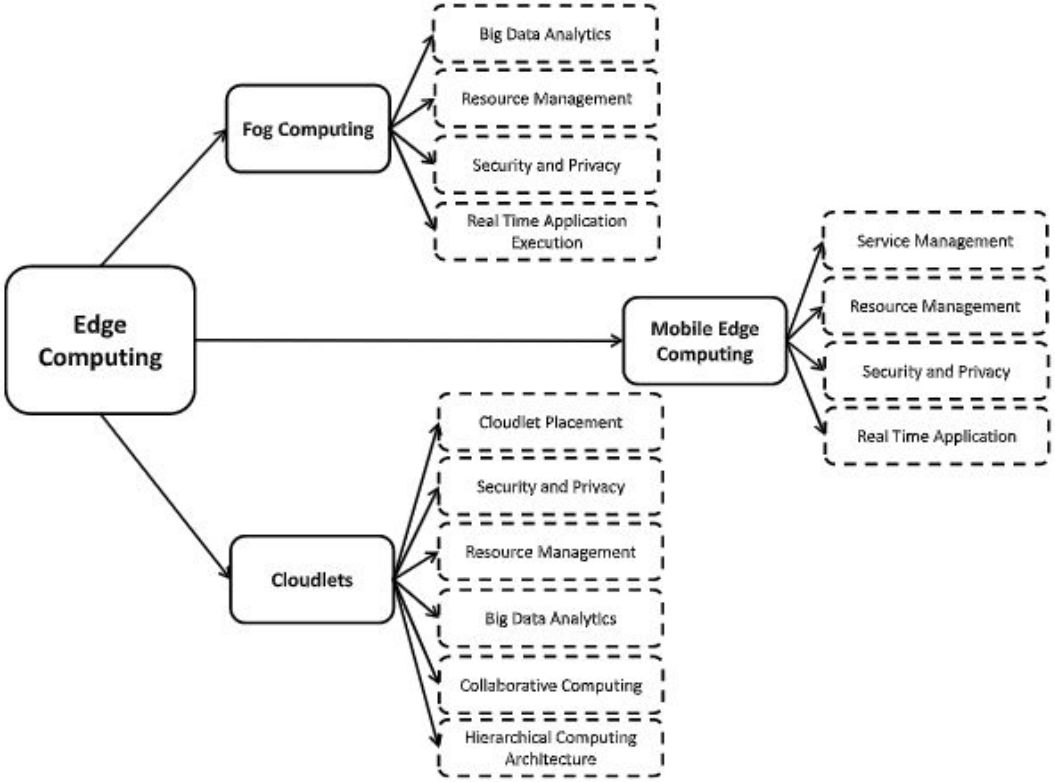

Cloud
• Central processing of summary data
• Big data analysis, complex learning model
• Central control
Internet
Cloud Apps
Edge
• Real time data processing
• Real time control (M2M)
• Local data filtering and caching
• At source data visualization
Intra net
Internet Gateway
Network Switch
Server
Storage
IoT Gateway
IoT Gateway
Sensors and controllers
Proximity

Characteristics and functionality within Edge Computing Architectures

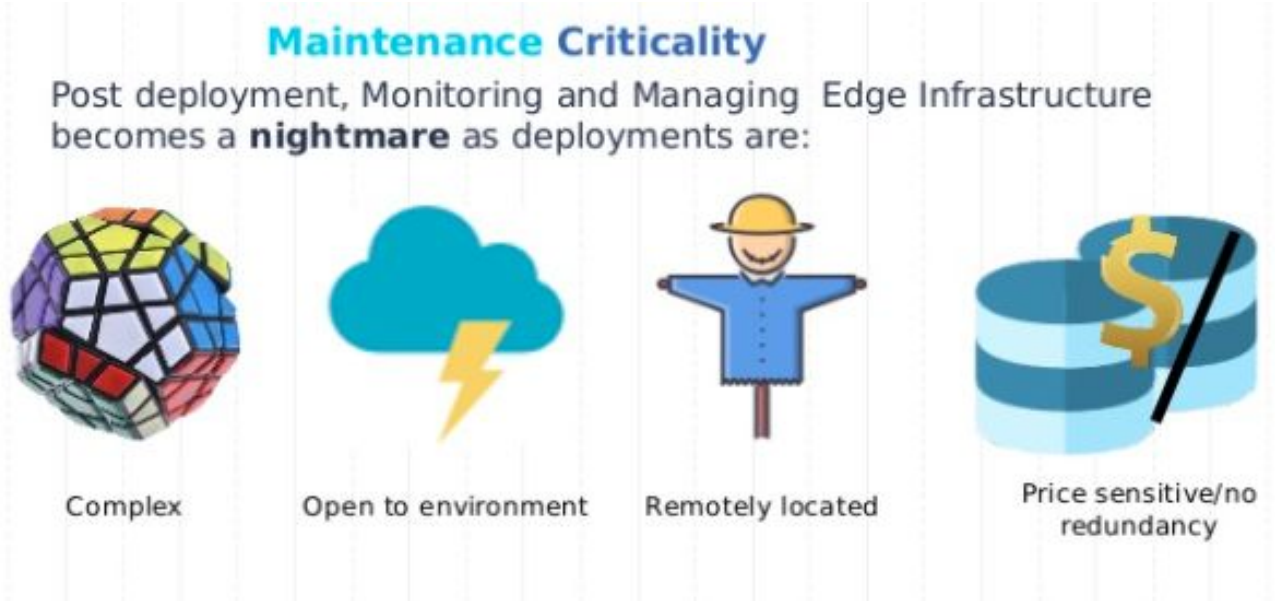
Characteristics	Edge Computing Architecture Layer			
	IoT	**Edge**	**Fog**	**Cloud**
Deployment	Distributed	Distributed	Distributed	Centralised
Components	Physical devices	Edge Nodes	Fog Nodes	Virtual resources
Location awareness	Aware	Aware	Aware	Aware
Computational Limits	Limited	Limited	Limited	Unlimited
Storage Limits	Very limited	Limited	Limited	Unlimited
Data	Source	Process	Process	Process
Distance to data source	The source	The nearest	Near	Far
Response time	No response time	The fastest	Fast	Slow
Nodes count	The largest	Very large	Large	Small



Classifications of Edge computing



Challenges of Edge Deployments



Challenges of Edge Deployments

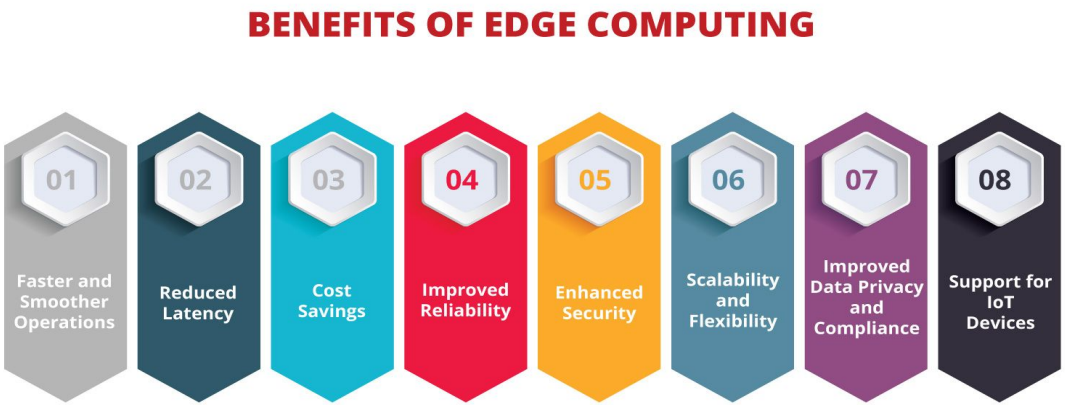
Challenges of Edge Deployments

Open challenges and guidelines

Challenges	Causes	Guidelines
User's trust on Edge computing systems	(a) Lack of security and privacy-preserving mechanisms	The influential factors of Consumer trust e.g., security and privacy, can be adopted to deal with challenges in stimulating the consumer's trust upon Edge computing systems.
Dynamic and agile pricing models	(a) High QoS requirements, (b) Inappropriate pricing models, (c) Service provider's high cost.	Considering the three important factors e.g., resource availability, frequency and duration of resource usage by consumers can help in developing dynamic pricing models.
Service discovery, service delivery and mobility	(a) Intermitted connectivity due to mobility, (b) Non-accessibility of local resources, (c) Immature security policies.	The mobility management for wireless networks and Service discovery for peer to peer networks can be used as guidelines.
Collaborations between heterogeneous Edge computing Systems	(a) Heterogeneous architectures, (b) Interoperability problems, (c) Data privacy issues, (e) Deficiencies in terms of load balancing.	The interoperability and collaborations among ubiquitous systems can be used as guidelines.
Low-cost fault tolerant deployment models	(a) High availability, (b) Data integrity, (c) disaster recovery.	Anomaly detection and predictive maintenance through machine Learning can help in providing low-cost fault tolerance.
Security	(a) Involvement of distributed data processing, (b) Indispensable requirements of Edge computing, *e.g.*, distributed architecture, immense data processing capabilities, location-awareness, and mobility support.	Salient features of blockchain technology, *e.g.*, tamper-proof, redundant, and self-healing, can help to mitigate important security threats. Additionally, quantum cryptography based solutions can also be helpful.

What problems does edge computing solve?

“By processing data closer to the point of generation, it is possible to avoid unnecessary communication and storage costs while simultaneously applying machine learning and AI to identify data patterns that have an impact to the business.”



AI on the edge reduces response times

A few examples of emerging edge AI applications

Edge AI use case

In-home smart cameras can recognize that a person(s) has entered an area

nest

aws

On-device facial recognition and object recognition, where user data doesn`t leave the device

HUAWEI

On-board AI making instantaneous driving decisions

Vision for baby monitors, drones, robots, and other devices that can respond to situations without internet connection

intel

Cloud stores large datasets, trains algorithms, collects edge data, pushes AI model updates

Generative AI at the Edge

Generative AI models can be deployed on edge devices to enable real-time content generation, personalized experiences, and data synthesis without cloud dependency.

- **Generative AI**: AI models capable of creating content, such as images, text, or code.
- **Edge Computing**: Brings data processing and model inference closer to the source (e.g., IoT devices, local servers).
- **Combined Power**: Generative AI at the edge enables faster, context-aware content generation while enhancing privacy and reducing latency.

Why Generative AI Needs Edge Computing

- **Reduced Latency**: Edge computing enables real-time generation and interaction, essential for applications like augmented reality or real-time personalization.
- **Privacy Preservation**: Sensitive user data can remain on local devices, reducing privacy risks by avoiding cloud transmission.
- **Bandwidth Efficiency**: Generative AI models often work with large datasets (e.g., image or video content). Edge processing limits the need for extensive cloud data transfers.

Example:

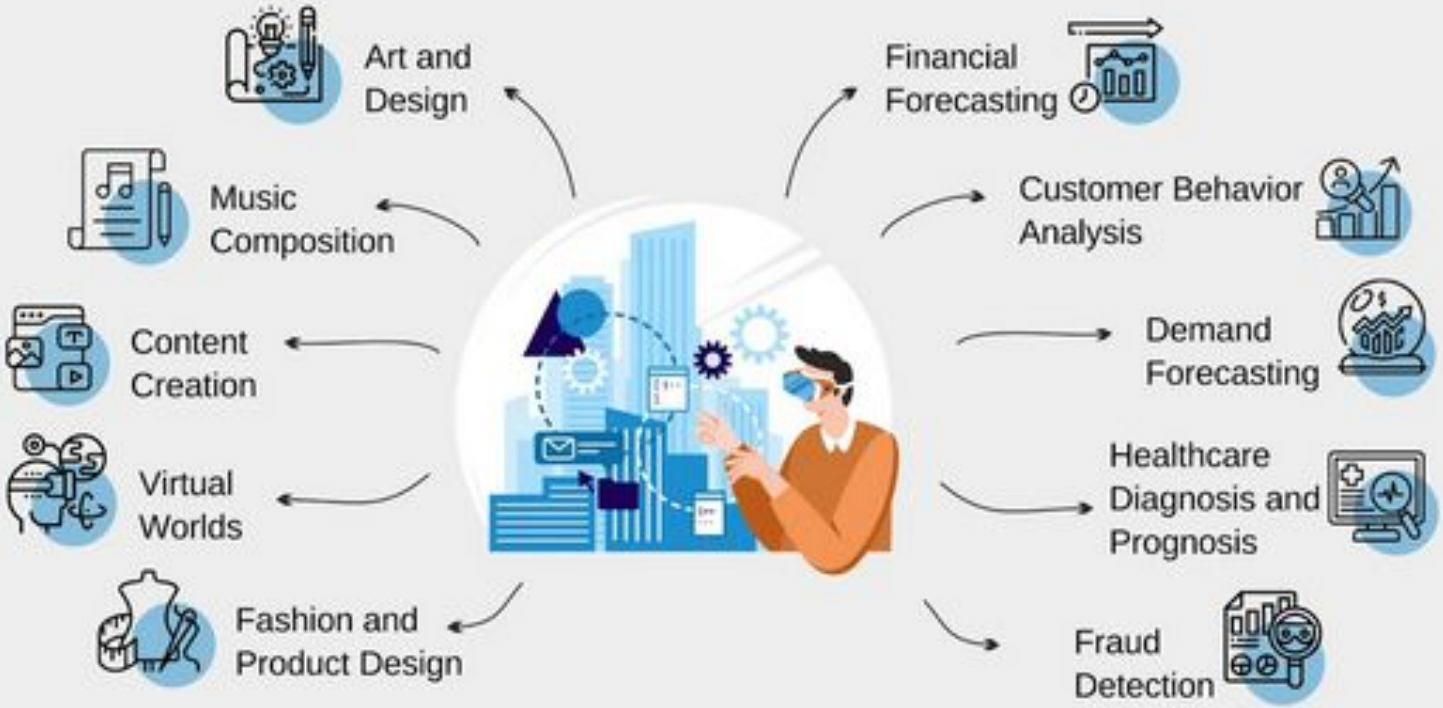
- “Imagine a generative AI model deployed on a smart home assistant that generates personalized responses locally—privacy is preserved, and the interaction is seamless and responsive.”

Key Applications of Generative AI at the Edge

- **Augmented Reality (AR) and Virtual Reality (VR)**: Edge-deployed generative models create immersive environments in real time, enhancing user experiences with minimal latency.
- **Smart Retail and Advertising**: On-device AI generates targeted content based on real-time data like customer behavior and preferences.
- **Healthcare**: Local generative models produce personalized treatment recommendations and synthetic data for diagnostics without compromising patient data privacy.
- **Industrial IoT**: Real-time anomaly detection and content generation for predictive maintenance, tailored to each unique device’s operational context.

Generative AI Applications

Neebal Technologies
www.neebal.com



Generative AI Models Suitable for Edge Deployment

- **Lightweight Variants**: Models like MobileGAN and TinyML can perform generative tasks with reduced computational requirements, making them ideal for edge deployment.
- **Edge-Tailored Models**: Techniques such as model pruning, quantization, and distillation reduce model size and complexity without sacrificing quality.
- **On-Device Inference**: Running models directly on devices (e.g., smartphones, IoT) ensures faster inference times and independence from cloud resources.

Challenges of Generative AI on Edge Devices

- **Computational Constraints**: Edge devices have limited processing power, which can limit the complexity of generative models.
- **Data Heterogeneity**: Edge devices may encounter diverse, inconsistent data, complicating model training and fine-tuning.
- **Energy Efficiency**: Continuous model inference can drain device battery life, especially in applications requiring real-time interactions.
- **Security Risks**: Sensitive AI models on edge devices are vulnerable to tampering or unauthorized access.

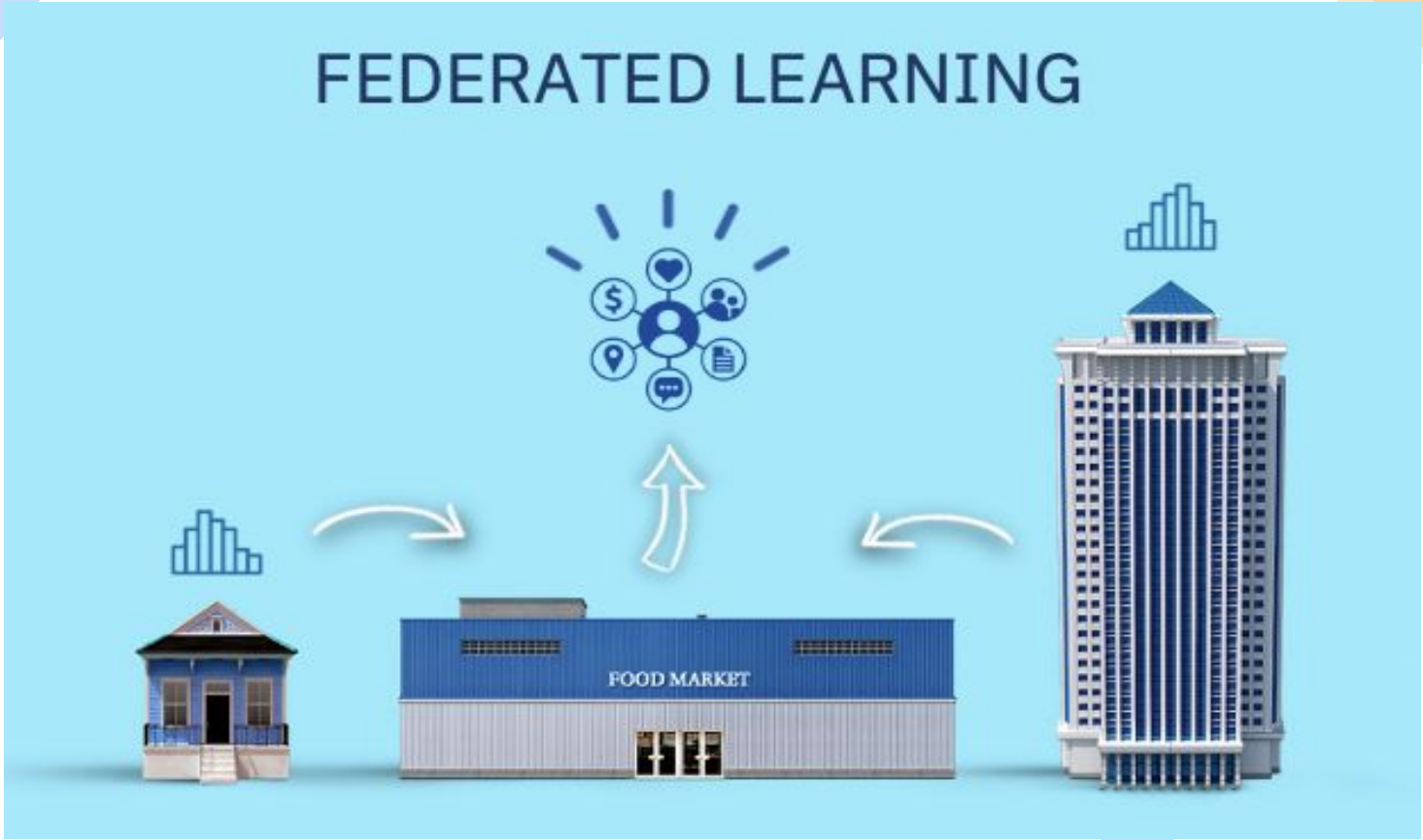
Future Directions for Generative AI at the Edge

- **Federated Learning**: Decentralized model training enables continuous model improvement on edge devices without central data aggregation.
- **Model Compression Advances**: Emerging techniques, like neural architecture search, identify model structures best suited for edge devices.
- **5G and Network Improvements**: Higher-speed networks will enhance edge capabilities, supporting more complex generative tasks with minimal latency.
- **Edge AI Hardware Innovations**: Hardware accelerators (e.g., Edge TPUs) enable efficient on-device inference, improving model performance and energy efficiency.

Summary and Key Takeaways: Edge AI

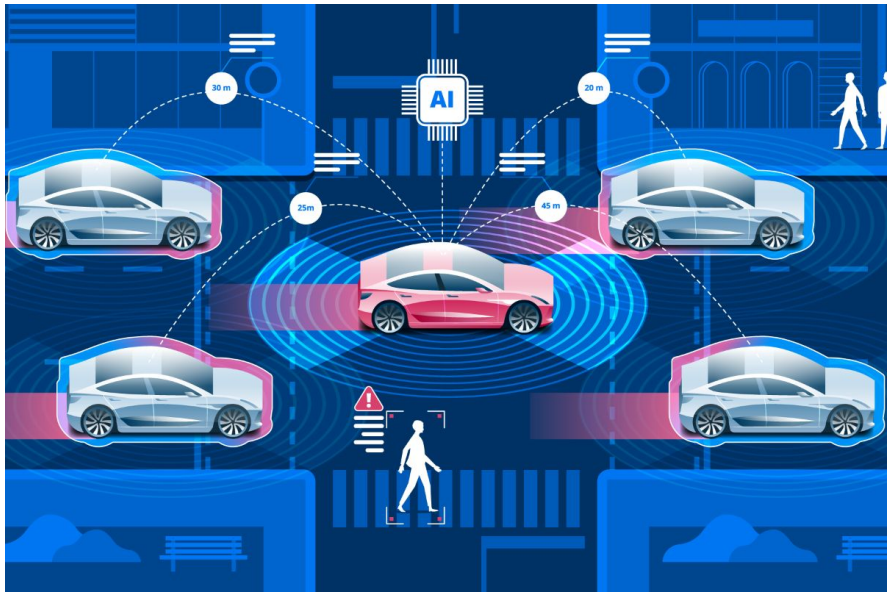
Main Points Recap:

- Edge computing enables generative AI to operate in real-time, providing personalized, low-latency interactions.
- Edge-specific generative AI applications span industries, from AR/VR and healthcare to industrial IoT.
- Ongoing research in model compression, federated learning, and edge hardware continues to unlock generative AI’s potential on the edge.


FEDERATED LEARNING
FOOD MARKET

Data is born at the edge

- Billions of phones & IoT devices constantly generate data
- Data enables better products and smarter models



Can data live at the edge?

Data processing is moving on device:

- Improved latency
- Works offline
- Better battery life
- Privacy advantages

E.g., on-device inference for mobile keyboards and cameras.

What about analytics?
What about learning?

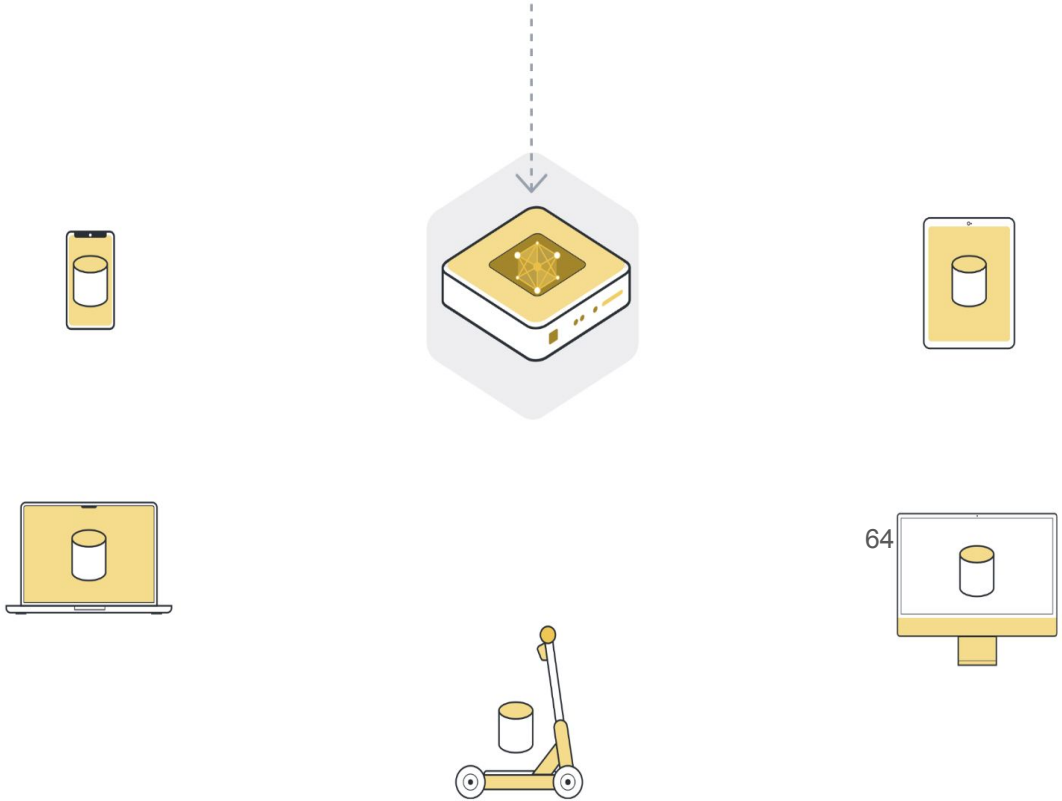
Federated learning

- Federated learning is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider.
- Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective.

- Definition proposed in Advances and Open Problems in Federated Learning (arxiv/1912.04977)

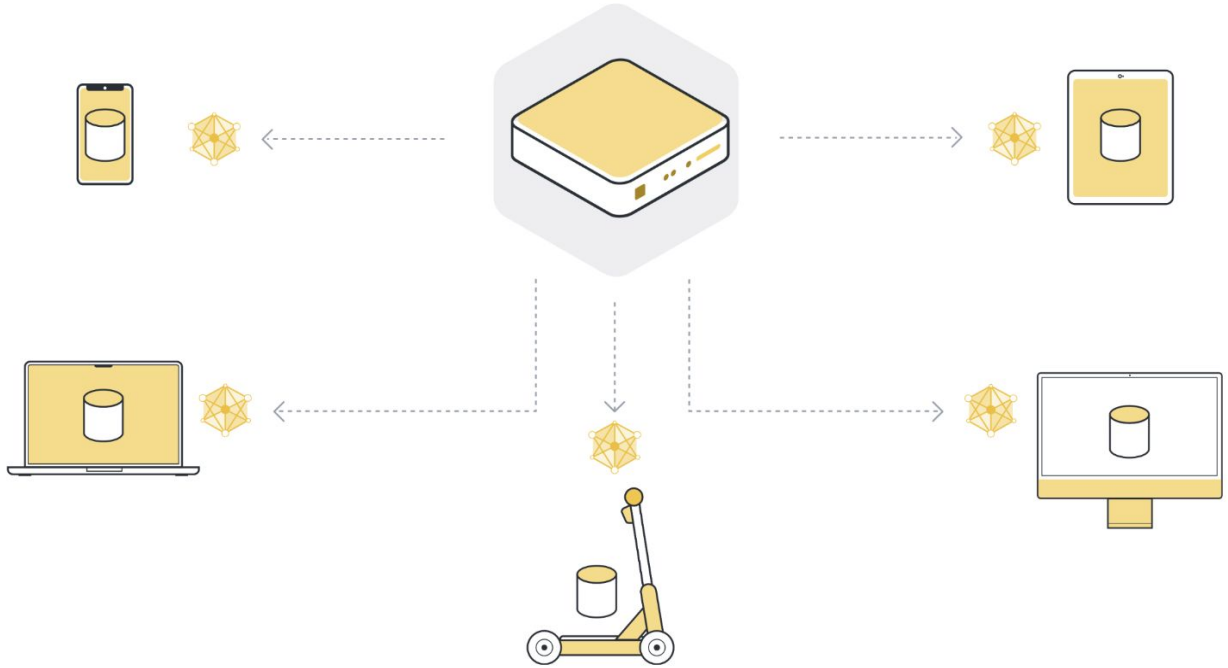
Federated learning in five steps

- Step 0: Initialize global model



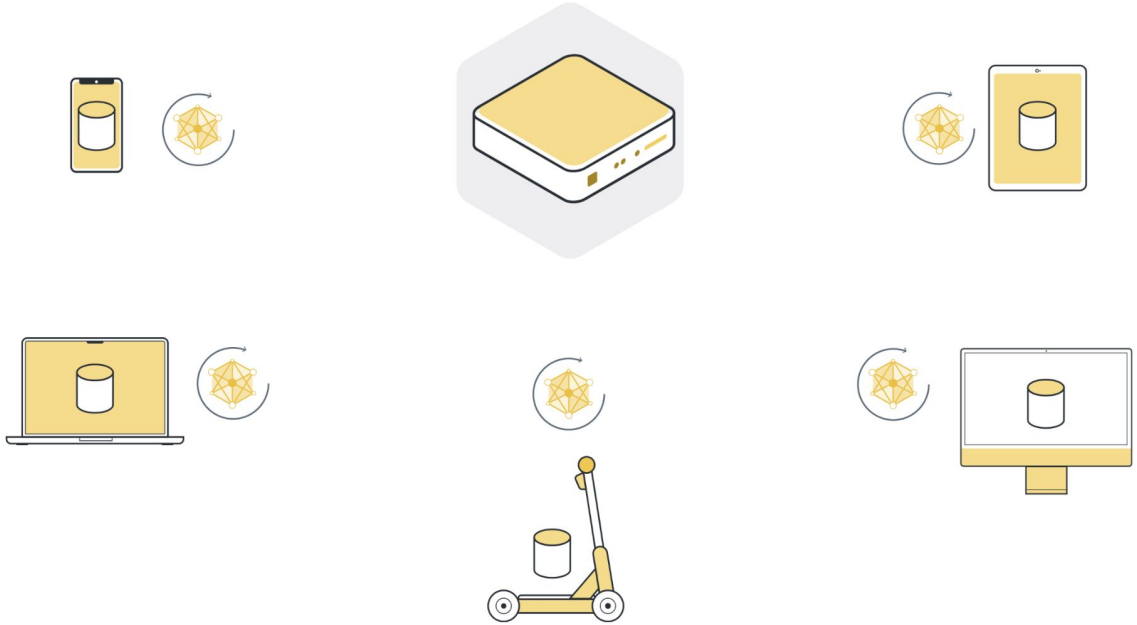
Federated learning in five steps

- Step 1: Send model to a number of connected organizations/devices (client nodes)



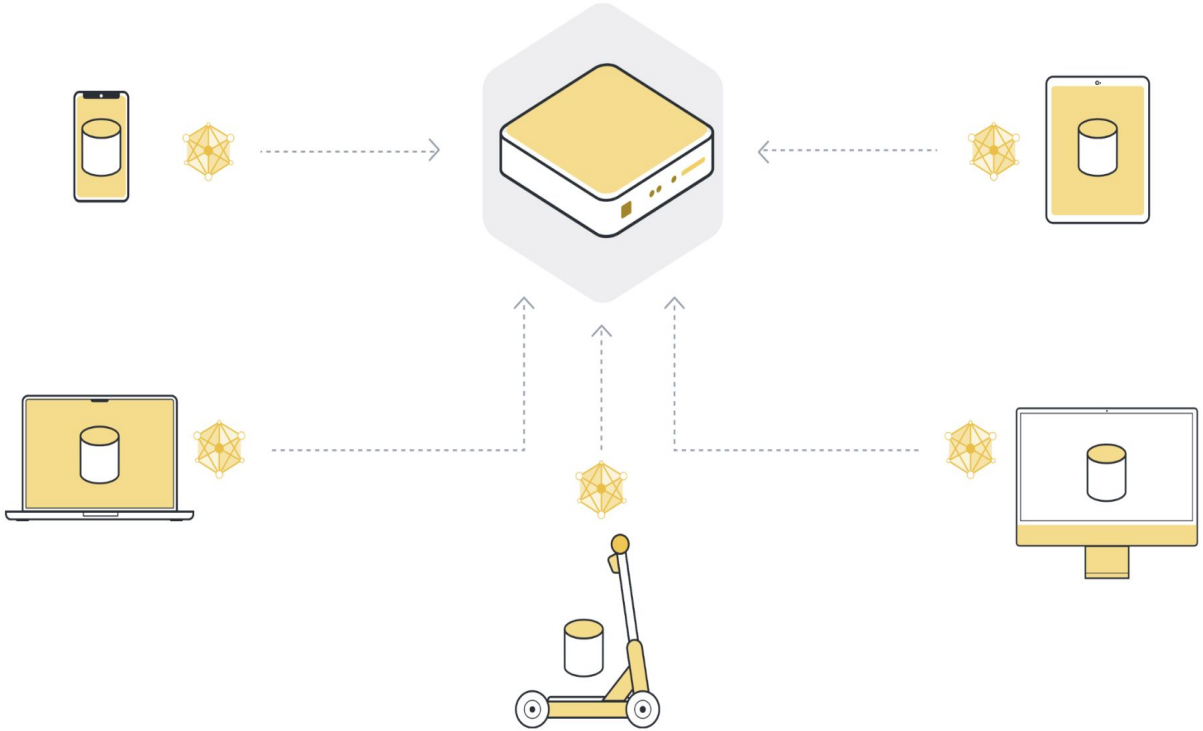
Federated learning in five steps

- Step 2: Train model locally on the data of each organization/device (client node)



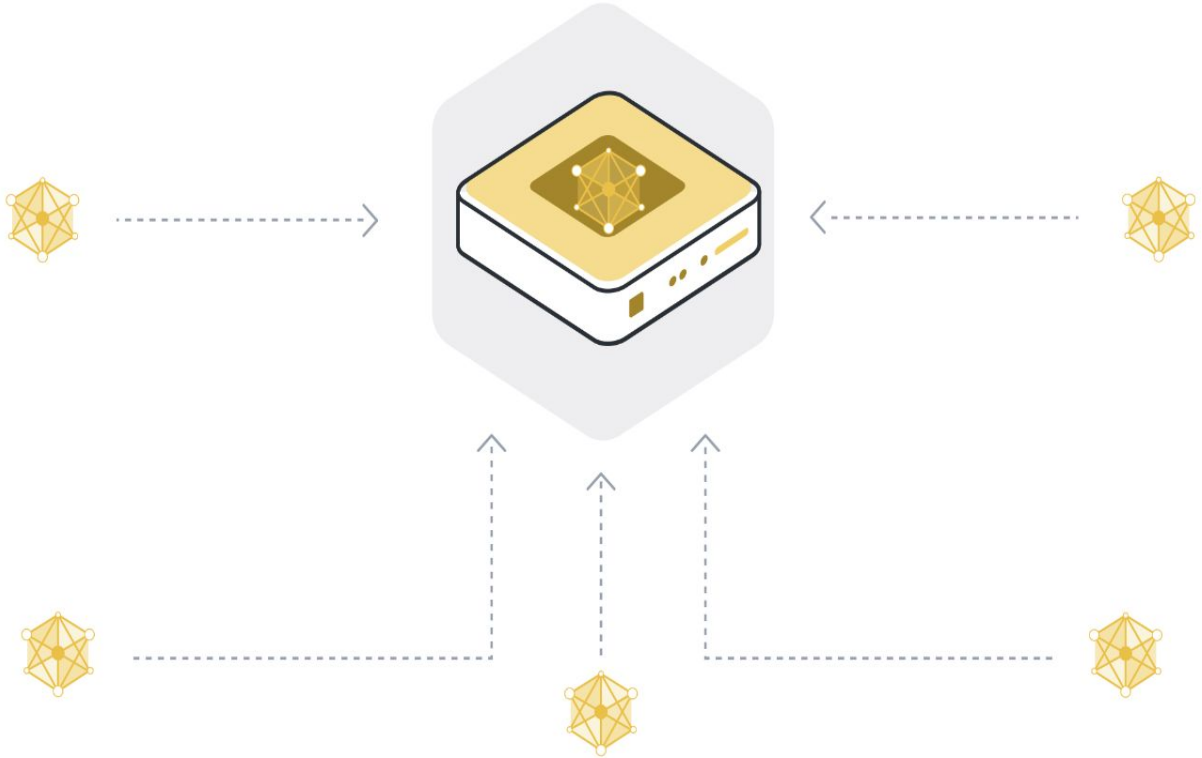
Federated learning in five steps

- Step 3: Return model updates back to the server



Federated learning in five steps

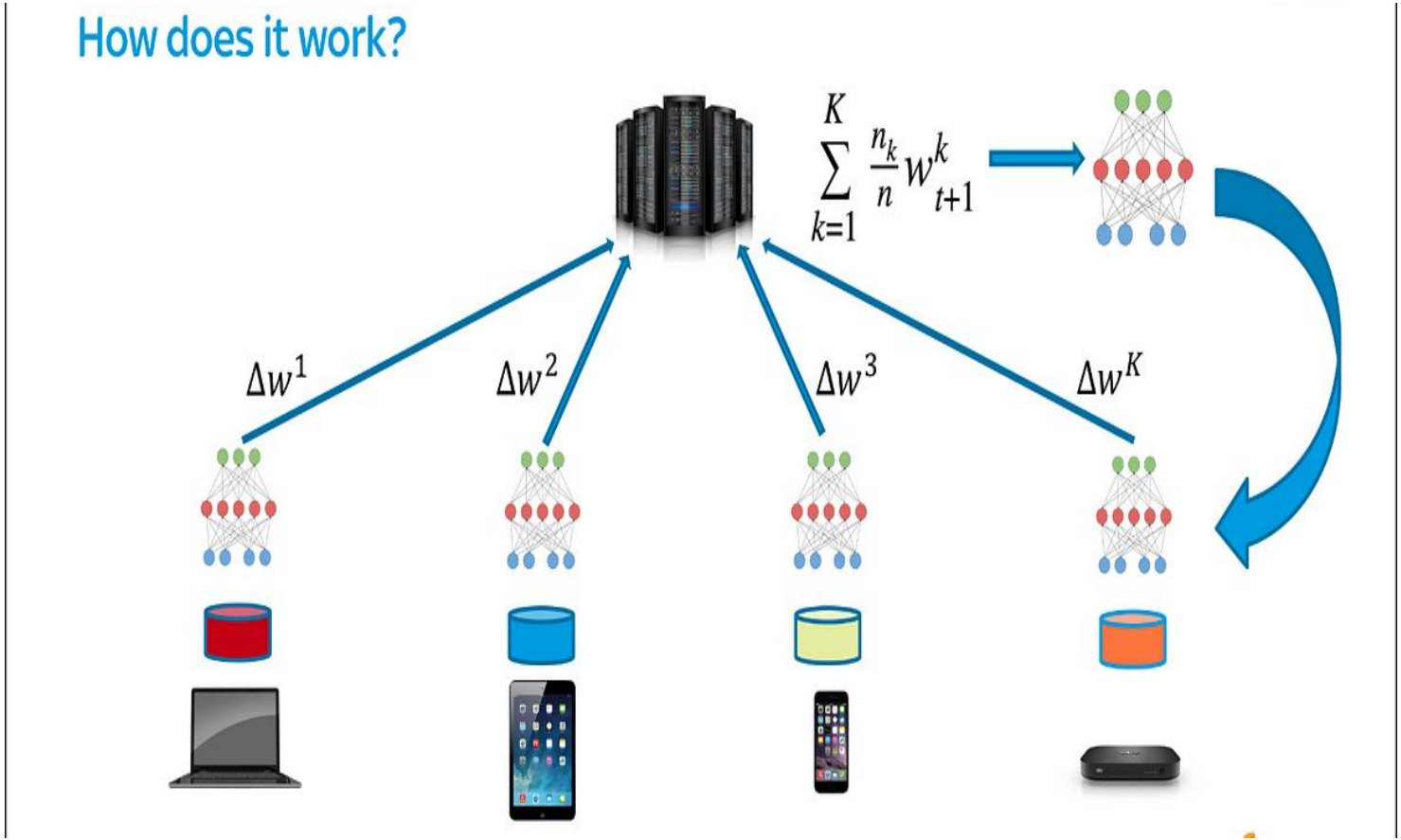
- Aggregate model updates into a new global model



Federated learning in five steps

- Step 5: Repeat steps 1 to 4 until the model converges
  - Steps 1 to 4 are what we call a single round of federated learning.
  - The global model parameters get sent to the participating client nodes (step 1),
  - the client nodes train on their local data (step 2),
  - they send their updated models to the server (step 3),
  - and the server then aggregates the model updates to get a new version of the global model (step 4

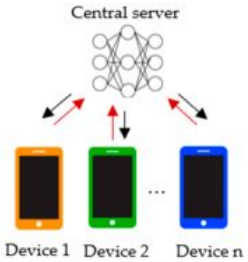
Federated learning in five steps



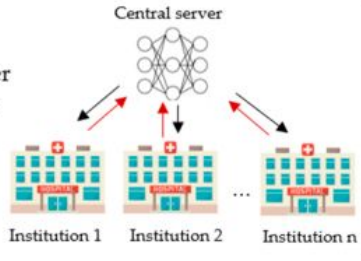
Types of Federated Learning

Types of participants

Cross-device Federated Learning: usually requires a million devices with small amount of data, such as smartphones, wearables, and edge devices

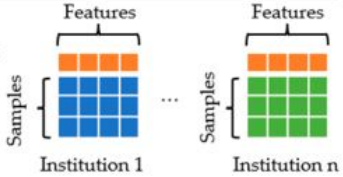


Cross-silo Federated Learning: usually requires a smaller number of collaborative participants with large sample sizes, such as hospitals or banks

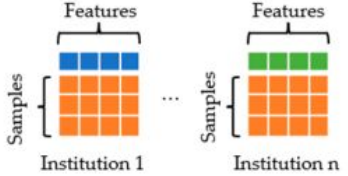


Types of data

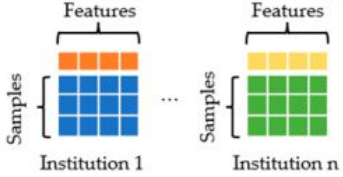
Horizontal Federated Learning: similar data features but from different samples



Vertical Federated Learning: shared or overlapped samples but different data features

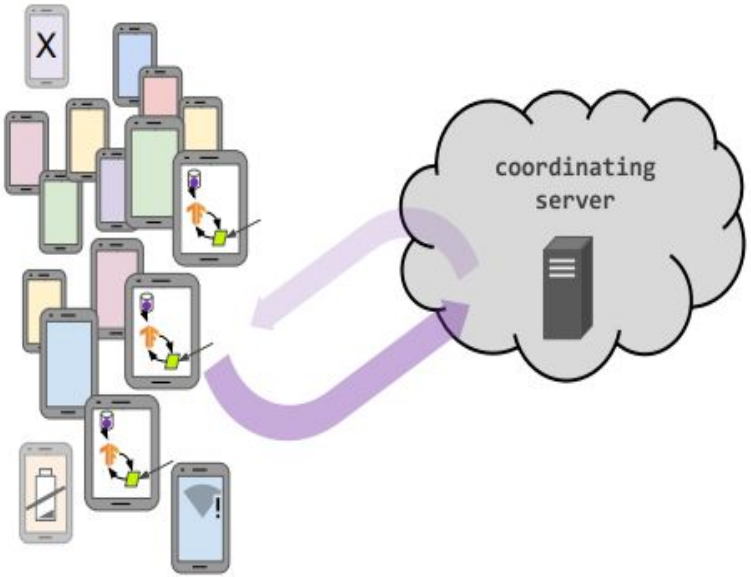


Federated Transfer Learning: different in both samples and features



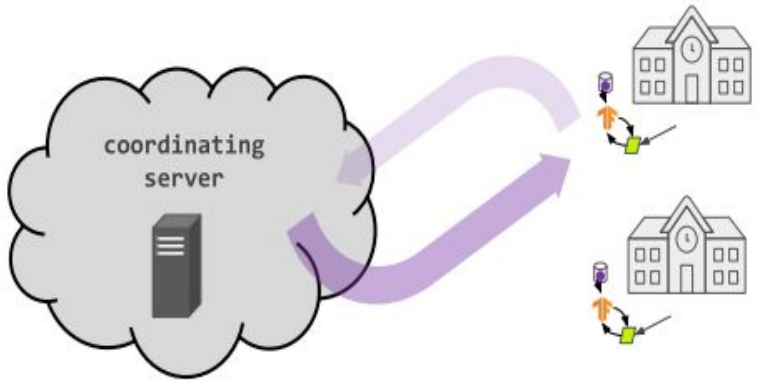
Cross-device federated learning

millions of intermittently
available client devices



Cross-silo federated learning

small number of clients
(institutions, data silos),
high availability



Applications of cross-device federating learning

What makes a good application?

- On-device data is more relevant than server-side proxy data
- On-device data is privacy sensitive or large
- Labels can be inferred naturally from user interaction

Example applications

- Language modeling for mobile keyboards and voice recognition
- Image classification for predicting which photos people will share
- ...

Characteristics of the federated learning setting

	Datacenter distributed learning	Cross-silo federated learning	Cross-device federated learning
Setting	Training a model on a large but "flat" dataset. Clients are compute nodes in a single cluster or datacenter.	Training a model on siloed data. Clients are different organizations (e.g., medical or financial) or datacenters in different geographical regions.	The clients are a very large number of mobile or IoT devices.
Data distribution	Data is centrally stored, so it can be shuffled and balanced across clients. Any client can read any part of the dataset.	**Data is generated locally and remains decentralized.** Each client stores its own data and cannot read the data of other clients. Data is not independently or identically distributed.	
Orchestration	Centrally orchestrated.	A central orchestration server/service organizes the training, but never sees raw data.	
Wide-area communication	None (fully connected clients in one datacenter/cluster).	Typically hub-and-spoke topology, with the hub representing a coordinating service provider (typically without data) and the spokes connecting to clients.	
Data availability	All clients are almost always available.		Only a fraction of clients are available at any one time, often with diurnal and other variations.
Distribution scale	Typically 1 - 1000 clients.	Typically 2 - 100 clients.	Massively parallel, up to 10^{10} clients.

Adapted from Table 1 in Advances and Open Problems in Federated Learning (arxiv/1912.04977)

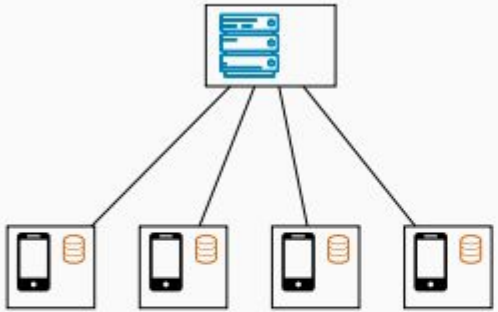
Characteristics of the federated learning setting

	Datacenter distributed learning	Cross-silo federated learning	Cross-device federated learning
Addressability	Each client has an identity or name that allows the system to access it specifically.		Clients cannot be indexed directly (i.e., no use of client identifiers)
Client statefulness	Stateful --- each client may participate in each round of the computation, carrying state from round to round.		Generally stateless --- each client will likely participate only once in a task, so generally we assume a fresh sample of never before seen clients in each round of computation.
Primary bottleneck	Computation is more often the bottleneck in the datacenter, where very fast networks can be assumed.	Might be computation or communication.	Communication is often the primary bottleneck, though it depends on the task. Generally, federated computations uses wi-fi or slower connections.
Reliability of clients	Relatively few failures.		Highly unreliable --- 5% or more of the clients participating in a round of computation are expected to fail or drop out (e.g., because the device becomes ineligible when battery, network, or idleness requirements for training/computation are violated).
Data partition axis	Data can be partitioned / re-partitioned arbitrarily across clients.	Partition is fixed. Could be example-partitioned (horizontal) or feature-partitioned (vertical).	Fixed partitioning by example (horizontal).

Adapted from Table 1 in Advances and Open Problems in Federated Learning (arxiv/1912.04977)

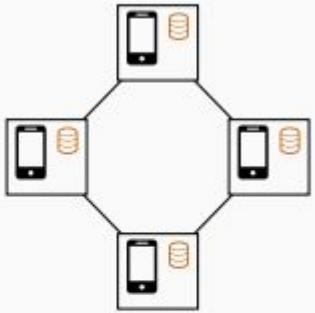
Server VS. Fully Decentralized FL

Server-orchestrated FL



- Server-client communication
- Global coordination, global aggregation
- Server is a single point of failure and may become a bottleneck

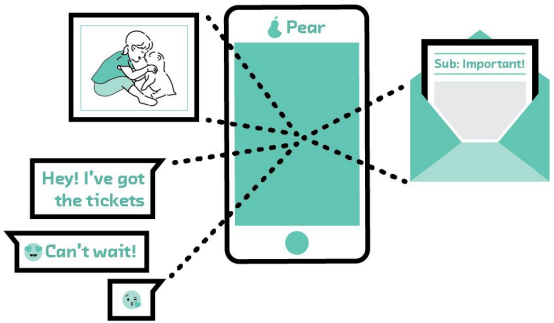
Fully decentralized FL



- Device-to-device communication
- No global coordination, local aggregation
- Naturally scales to a large number of devices

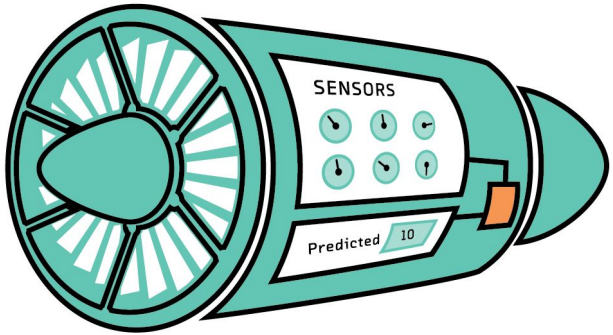
Use Cases

- **Pear** - a company that makes a popular smartphone. Users love it. They use it to take pictures of their kids, email their colleagues, and write quick text messages to friends. All of this activity on millions of Pear phones generates data that, if it were combined, would allow Pear to train models to make its phones even better.



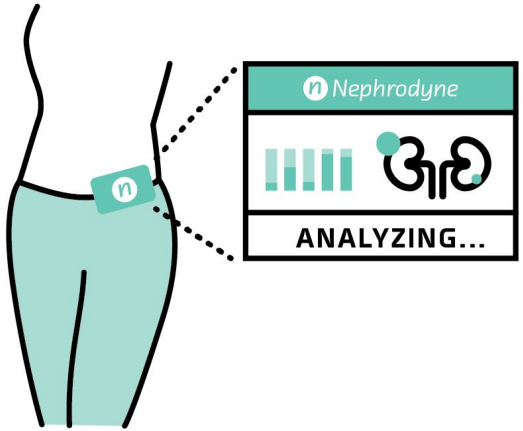
Use Cases

- **TurbineCorp** sells turbines for installation in power stations. These machines are profitable to run but expensive to maintain, and very expensive to repair. TurbineCorp wants to differentiate itself from its competitors by offering customers access to a predictive model of turbine failure.



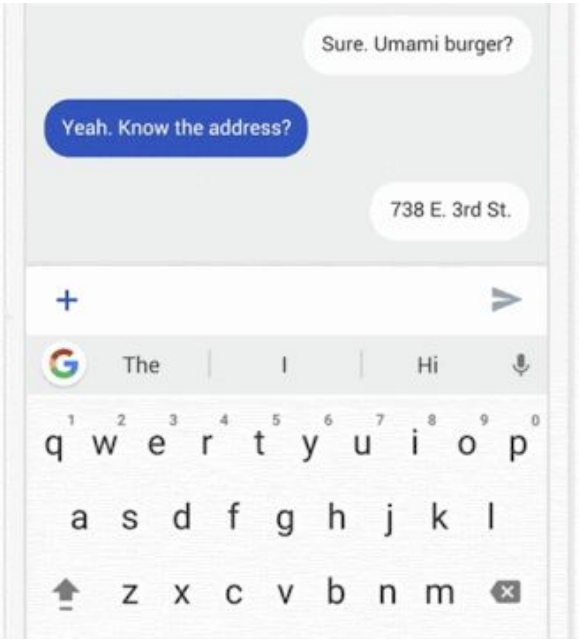
Use Cases

- Nephrodyne, a medical device company, wants to offer a wearable device that detects kidney problems in users before they become acutely serious.



Federated Learning Applications

- Gboard in Android



Federated Learning Applications

- Siri in Apple

MIT Technology Review

Sign in Subscribe

Artificial intelligence / Machine learning

How Apple personalizes Siri without hoovering up your data

The tech giant is using privacy-preserving machine learning to improve its voice assistant while keeping your data on your phone.

by **Karen Hao** December 11, 2019

https://www.technologyreview.com/2019/12/11/131629/%20apple-ai-personalizes-siri-federated-learning/

Federated Learning Applications

- Recommendations in Brave Browser

Using Federated Learning to Improve Brave's On-Device Recommendations While Protecting Your Privacy

Blog › Developers & community

Last updated Jun 8, 2021

Authors: Lorenzo Minto (Machine Learning) and Moritz Haller (Data Scientist) of Brave Software

Recommender systems have become ubiquitous in today's web. Content streaming platforms, e-commerce, and social networks all leverage some form of recommendation to personalize their services for the customer. But while recommendation can be of great benefit to the end-user—from discovering your new favorite artist to being informed of other news outlets relevant to your interests—modern recommendation engines are built by centrally collecting all kinds of user data (e.g. clicks, views, mouse scrolls) at the great expense of user privacy.

Recently, Brave introduced Brave News, a privacy-preserving news aggregator integrated into the browser. This service allows users to anonymously subscribe to RSS feeds of their favorite news outlets and stay up to date with all the latest news in a single place. The user can choose from 15 different categories of curated sources, and can easily customize their stream by adding or removing sources.

https://brave.com/federated-learning/

Federated Learning Applications

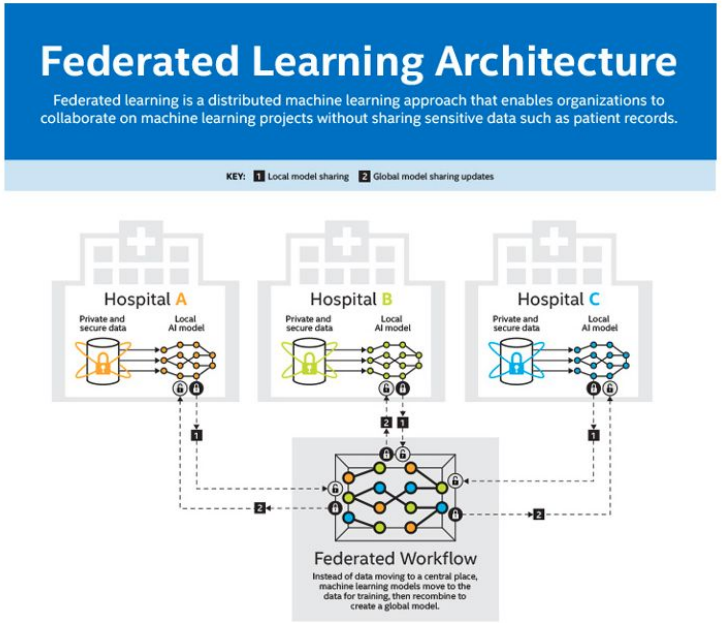
- Intel and Hospitals

News Byte

May 11, 2020

Contact Intel PR

Intel Works with University of Pennsylvania in Using Privacy-Preserving AI to Identify Brain Tumors



Some Awesome Tools to Build FL Applications

Generative AI and Types of Generative Models

Definition: Generative AI refers to AI models that create new data instances resembling training data, such as images, text, or sounds.

- **Types of Generative Models**:
  - **Generative Adversarial Networks (GANs)**: Pairs two networks (generator and discriminator) to create realistic data.
  - **Variational Autoencoders (VAEs)**: Encodes data into a continuous latent space, enabling controlled data generation.
  - **Autoregressive Models**: Models like GPT use sequences of previous data to generate the next item in a sequence.
- **Applications**:
  - **Personalization**: Tailoring content to individual users.
  - **Data Synthesis**: Creating synthetic datasets for training without exposing sensitive data.
  - **Content Creation**: Generating media assets like images, text, and videos for entertainment, marketing, and more.

How Edge Computing Enhances Generative AI

- **On-Device Processing**: Allows generative AI models to run on local devices, enabling immediate responses and reducing dependency on cloud servers.
- **Privacy Benefits**: Sensitive data stays on the device, minimizing the risk of data breaches and supporting regulatory compliance.
- **Examples**:
  - **Smart Assistants**: Generate personalized responses directly on the device.
  - **AR/VR Applications**: Real-time content generation for immersive experiences, adapting based on user actions.

How Federated Learning Enhances Generative AI

Collaborative Model Training: Federated learning allows models to learn from data across multiple devices without centralizing it.

Data Privacy: Ensures sensitive data remains local while contributing to model improvement.

Examples:

- **Healthcare Devices**: Local generative models for synthetic data generation used in diagnostics or treatment recommendations.
- **Personalized Applications**: Individual user preferences contribute to a global model without sharing actual data.

Challenges in Decentralized Generative AI

Key Challenges:

- **Computational Limitations**: Edge devices have limited processing power, affecting the model complexity they can support.
- **Data Heterogeneity**: Data from different devices may vary in quality and format, complicating model training.
- **Security Risks**: Decentralized models on local devices may face threats like tampering or unauthorized access.

Considerations:

- **Model Compression**: Techniques like pruning and quantization to make models fit edge constraints.
- **On-Device Optimization**: Designing lightweight architectures or using accelerators to enhance efficiency.
- **Data Consistency**: Developing strategies to standardize or pre-process data on each device.

Case Study - Generative AI for Smart Home Devices

Example: Federated learning and edge computing used in smart home devices to generate responses that adapt to user preferences over time.

Process:

- **Data Locality**: Smart devices collect data locally to personalize user interactions.
- **Federated Updates**: Devices send model updates, not raw data, to a central server for aggregation.
- **Real-Time Personalization**: Responses or content generated adapt dynamically to user behavior, preferences, and contexts.

Research Problems in Decentralized Generative AI

- **Model Efficiency and Compression**
  - Problem: Generative AI models are often large and computationally heavy, which limits their deployment on resource-constrained edge devices.
  - Research Focus: Developing lightweight versions of generative models through techniques like model pruning, quantization, and knowledge distillation.
- **Data Heterogeneity and Consistency**
  - Problem: Data collected from different devices is often heterogeneous, leading to challenges in training and ensuring model consistency across devices.
  - Research Focus: Designing methods for handling diverse data types and qualities, as well as techniques for standardized on-device pre-processing.

Research Problems in Decentralized Generative AI

- **Privacy and Security**
  - Problem: Decentralized setups increase security risks, as models and data on edge devices may be vulnerable to tampering and unauthorized access.
  - Research Focus: Enhancing privacy-preserving techniques, including differential privacy, secure multi-party computation, and federated learning with encryption.
- **Energy Efficiency and Battery Life**
  - Problem: Continuous on-device processing can be power-intensive, which limits the feasibility of generative AI on battery-powered devices.
  - Research Focus: Developing energy-efficient algorithms and hardware accelerators tailored to run generative models within power constraints.
- **5Model Convergence and Communication Overhead**
  - Problem: Synchronizing model updates across devices in federated learning can be slow, leading to delays in model convergence.
  - Research Focus: Optimizing communication strategies, like asynchronous updates or selective model sharing, to reduce bandwidth use and speed up convergence.

Future Directions in Decentralized Generative AI

1. Advanced Model Compression Techniques

- Direction: Utilizing emerging techniques like neural architecture search (NAS) to automatically design and compress generative models for edge devices.
- Expected Outcome: Higher performance models with minimal loss in output quality, enabling sophisticated generative tasks on constrained hardware.

2. Privacy-First Frameworks

- Direction: Building privacy-centric frameworks for decentralized generative AI, incorporating advanced encryption and on-device differential privacy.
- Expected Outcome: Scalable systems where sensitive data remains secure across devices, allowing decentralized AI to meet privacy regulations in sectors like healthcare and finance.

3. Federated Generative Model Training

- Direction: Expanding federated learning to support generative models, allowing for collaborative model training across devices without centralizing data.
- Expected Outcome: Improved model quality through diverse data while preserving privacy, supporting personalized and context-aware content generation.

Future Directions in Decentralized Generative AI

4. Real-Time Personalization with Edge AI

- Direction: Combining generative AI with real-time, on-device adaptation to create dynamic, personalized content and experiences (e.g., smart home assistants, AR/VR).
- Expected Outcome: Increased user engagement and relevance in applications, as models adapt in real time based on individual behaviors and preferences.

5. Edge AI Hardware and Accelerators

- Direction: Development of specialized hardware, like edge TPUs and AI accelerators, to support generative AI workloads on mobile and IoT devices.
- Expected Outcome: Faster and more energy-efficient processing of generative models at the edge, unlocking more complex use cases across industries.

6. Interoperable Decentralized Ecosystems

- Direction: Establishing standardized protocols and ecosystems where different devices and platforms can contribute to a shared generative AI model pool.
- Expected Outcome: Collaborative intelligence networks that leverage decentralized generative AI, facilitating cross-device and cross-platform AI growth.

Thank you!

annappa@nitk.edu.in
www.annappa.in