AI FOR EDGE COMPUTING

UNIT I – Introduction to Edge Computing and AI

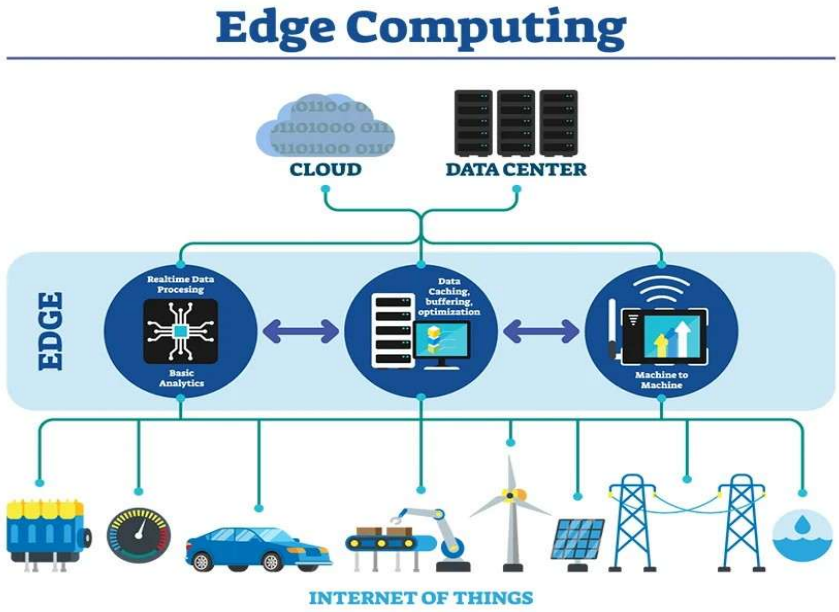
Evolution of Computing Paradigms: Cloud, Fog, and Edge

Building Blocks of Edge Computing.

Recapitulation

- Cloud computing provides centralized processing but has latency limitations.
- Fog computing acts as a bridge between cloud and edge.
- Edge computing enables real-time, low-latency processing at the data source.
- Edge AI brings intelligence closer to devices.
- Future systems rely on **integrated Cloud–Fog–Edge architectures**.

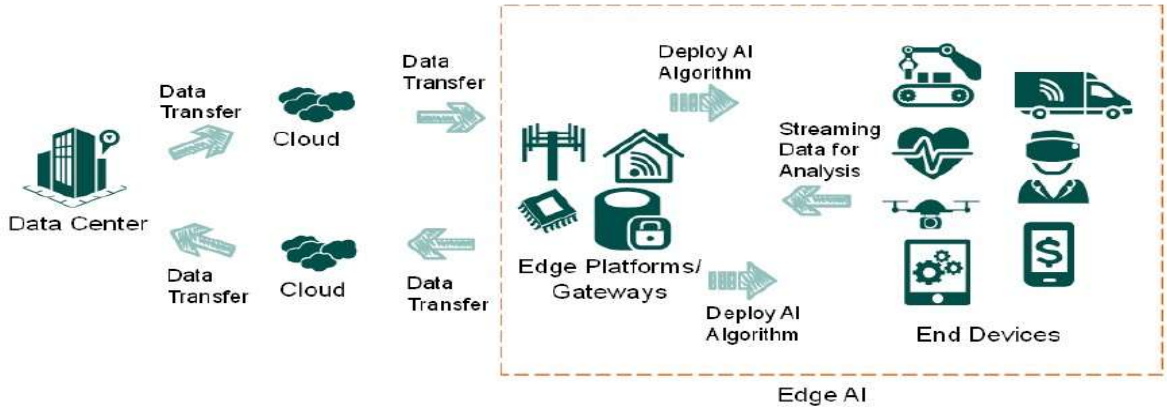
Introduction to Edge Computing and AI



- **Edge Computing** is a distributed computing approach where data is processed **near the source of data generation** instead of a centralized data center.
- It reduces **latency**, **bandwidth consumption**, and **response time**.
- **Edge AI** refers to executing **Artificial Intelligence (AI) and Machine Learning (ML) models directly on edge devices**.
- Edge computing supports **real-time analytics and decision making**.
- Common applications include **IoT systems, smart cities, healthcare monitoring, autonomous vehicles, and industrial automation**.

Evolution of Computing Paradigms

Figure 2: Edge AI Processing Flow
(Source: ABI Research)



1. Cloud Computing

- Centralized computing model using remote data centers.
- All data is sent to the cloud for processing and storage.
- Provides high scalability and computing power.
- Suffers from **high latency and network dependency**.
- Not ideal for time-critical applications.

2. Fog Computing

- Intermediate layer between cloud and edge.
- Data is processed at **gateways, routers, or local servers**.
- Reduces latency compared to cloud computing.
- Improves efficiency by filtering data before sending to the cloud.
- Suitable for **location-based services**.

3. Edge Computing

- Computation is performed **directly on end devices**.
- Very low latency and fast response.
- Minimizes data transfer to cloud.
- Operates even with **limited internet connectivity**.
- Ideal for **real-time and mission-critical applications**.

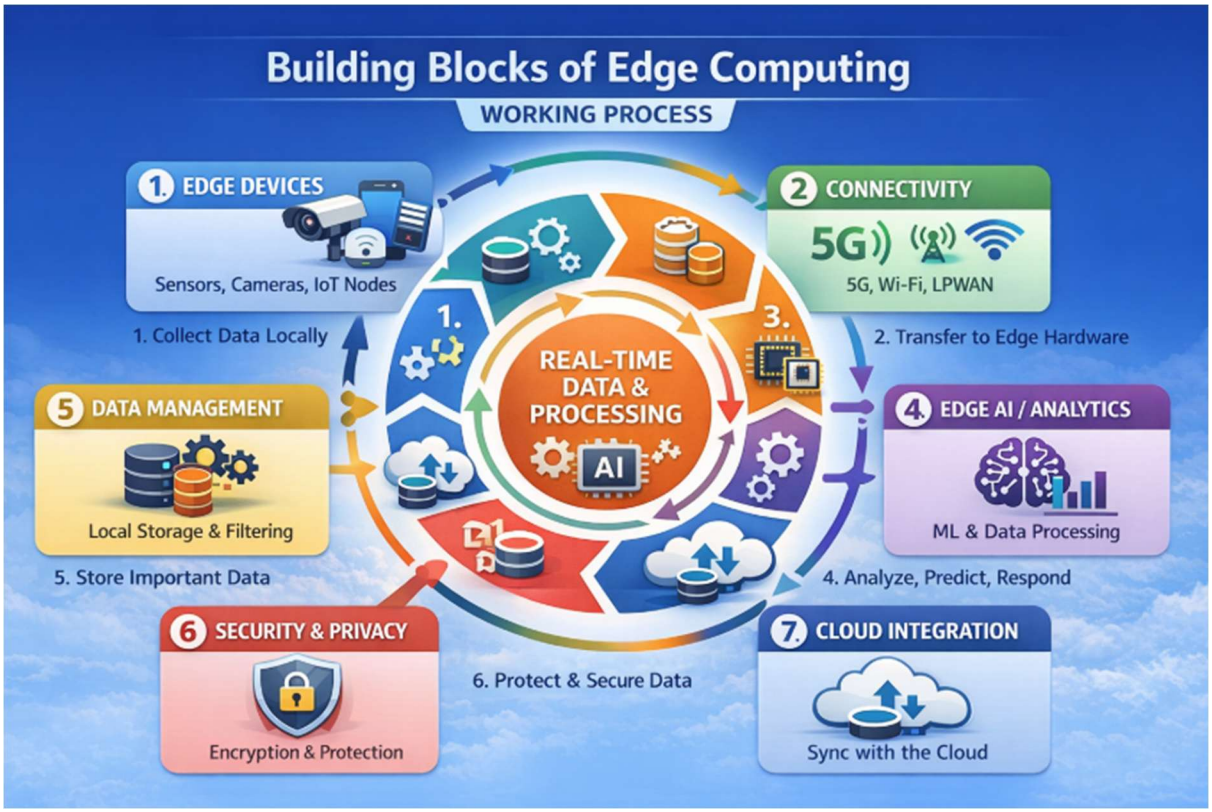
Comparison of Cloud, Fog, and Edge Computing

- **Cloud**: Centralized, high latency, high bandwidth usage.
- **Fog**: Semi-distributed, moderate latency, optimized bandwidth usage.
- **Edge**: Fully distributed, minimal latency, low bandwidth usage.

Role of AI in Edge Computing

- AI models are trained in the cloud and deployed at the edge.
- Enables **real-time inference** on local data.
- Enhances **privacy** by keeping sensitive data local.
- Reduces communication overhead with the cloud.
- Enables autonomous and intelligent systems.

Building Blocks of Edge Computing (7 Topics)



1. Edge Devices

- Sensors, actuators, cameras, and smart devices.
- Generate and process data locally.

2. Connectivity

- Communication technologies like 5G, Wi-Fi, Ethernet, and LPWAN.
- Ensures fast and reliable data transfer.

3. Edge Computing Hardware

- Embedded systems, microcontrollers, GPUs, NPUs.
- Designed for low power and high performance.

4. Edge AI and Analytics

- Machine learning and deep learning algorithms.

- Used for pattern recognition and prediction.

5. Data Management

- Local data filtering, preprocessing, and aggregation.
- Reduces data sent to the cloud.

6. Security and Privacy

- Authentication, encryption, and secure boot mechanisms.
- Protects data and devices from attacks.

7. Cloud Integration

- Cloud supports training, updates, and long-term storage.

Works in coordination with edge systems.

Introduction to Edge AI – Concepts and Motivation

1. Edge AI (Concept)

Edge AI **running AI directly on devices near the data source** instead of using cloud servers.

Key points:

- Data is processed locally
- Faster response (real-time)
- More privacy and security

Example:
A smart camera detecting intruders on the device itself.

2. Motivation for Edge AI

Edge AI is needed because cloud-based AI has limitations.

Main reasons:

- **Low latency:** Faster decisions
- **Privacy:** Data stays on device
- **Less bandwidth:** No need to send raw data
- **Reliable:** Works without internet
- **Energy efficient:** Saves power

3. Embedded Systems (EBS)

Embedded systems are **small computers designed for a specific task**.

Role in Edge AI:

- Run AI models

- Connect sensors and actuators
- Perform inference on-device

Features:
Low power, limited memory, real-time operation.

4. Digital Signal Processing (DSP)

DSP deals with **processing real-world signals** like audio, images, and sensor data.

Why important in Edge AI:

- Fast mathematical operations
- Real-time signal processing
- Helps in feature extraction before ML

Example:
Audio filtering before speech recognition.

5. TinyML

TinyML is Edge AI on **very small, low-power microcontrollers**.

Key features:

- Very small ML models
- Ultra-low power usage
- Runs on ARM Cortex-M MCUs

Applications:
Wake-word detection, gesture recognition, sensor monitoring.

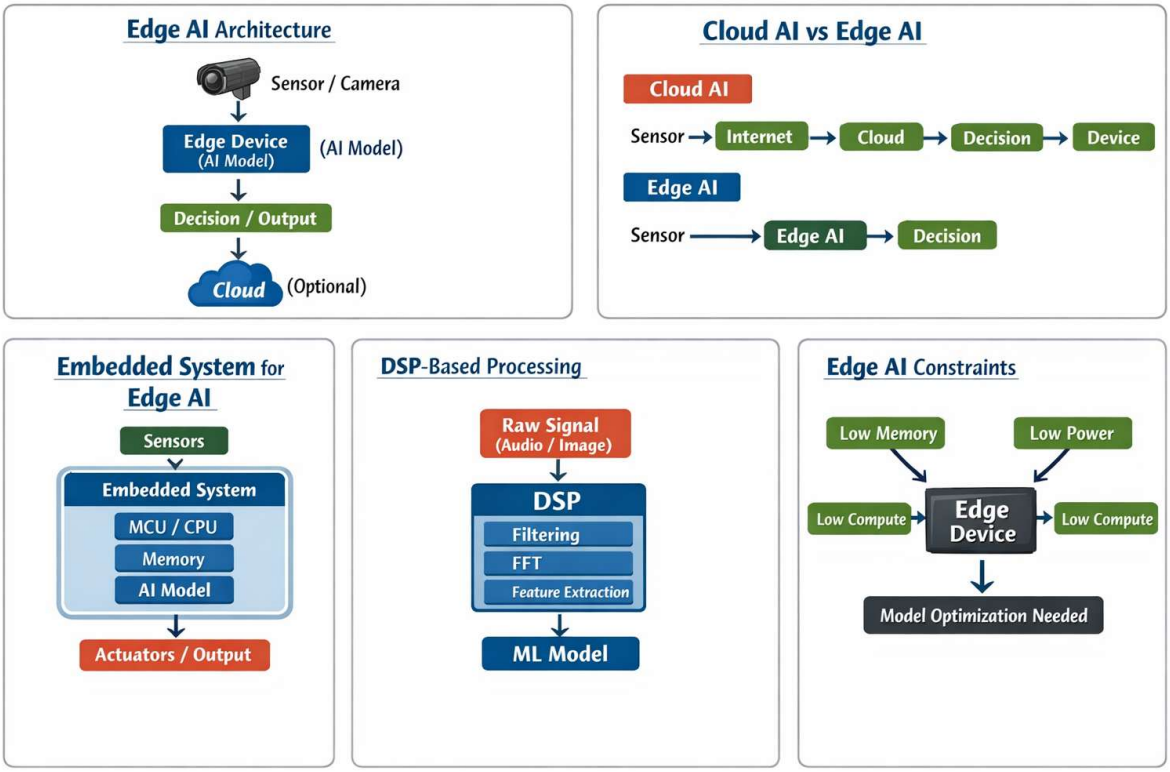
6. Challenges in Edge AI & TinyML

Running AI on small devices is difficult.

Major challenges:

- Limited memory and processing power
- Power constraints
- Model optimization needed
- Deployment complexity
- Accuracy vs efficiency trade-off

Diagrams:



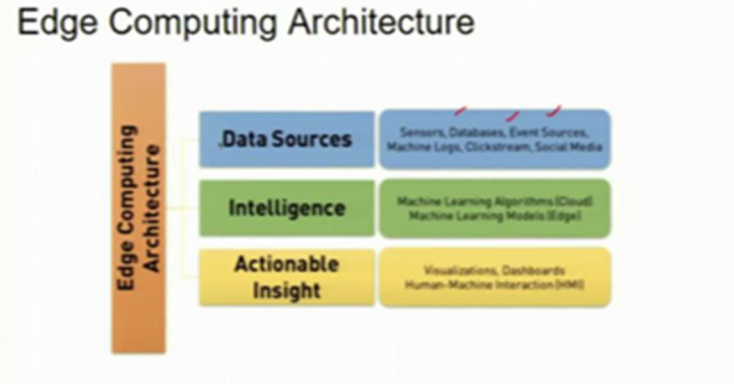
Architecture of Edge Computing Systems

Edge computing architecture three tire

1. Now, to understand this edge computing in more detail, that is in a systematic manner,

2. we have now divided into three different layers of an edge-computing architecture,

3. these layers may vary, but for our understanding of our lecture, let us understand let us assume that the edge computing architecture we can divide into three different layers.



•**The First layer is data source** : (data source are the origins of data that generates information at edge)

That layer data source comprises of sensors, database events, sources, machine logs, clickstream, social media, they are all nothing but the data sources with these data sources in place into the edge computing that means edge computing supports the data ingestion from these different sources.

•**The Second one is intelligent tier** : (smart decisions are made using AI, ML)

This intelligent tier cuts across the cloud and edges so there is a very well-defined boundary between edge and cloud where the training takes place on the cloud and the inferencing is run on the edge. But collectively, this overlap between the cloud and the edge is this intelligence layer.

•**The Third layer is actionable insight** **:** (here decisions are taken quickly)

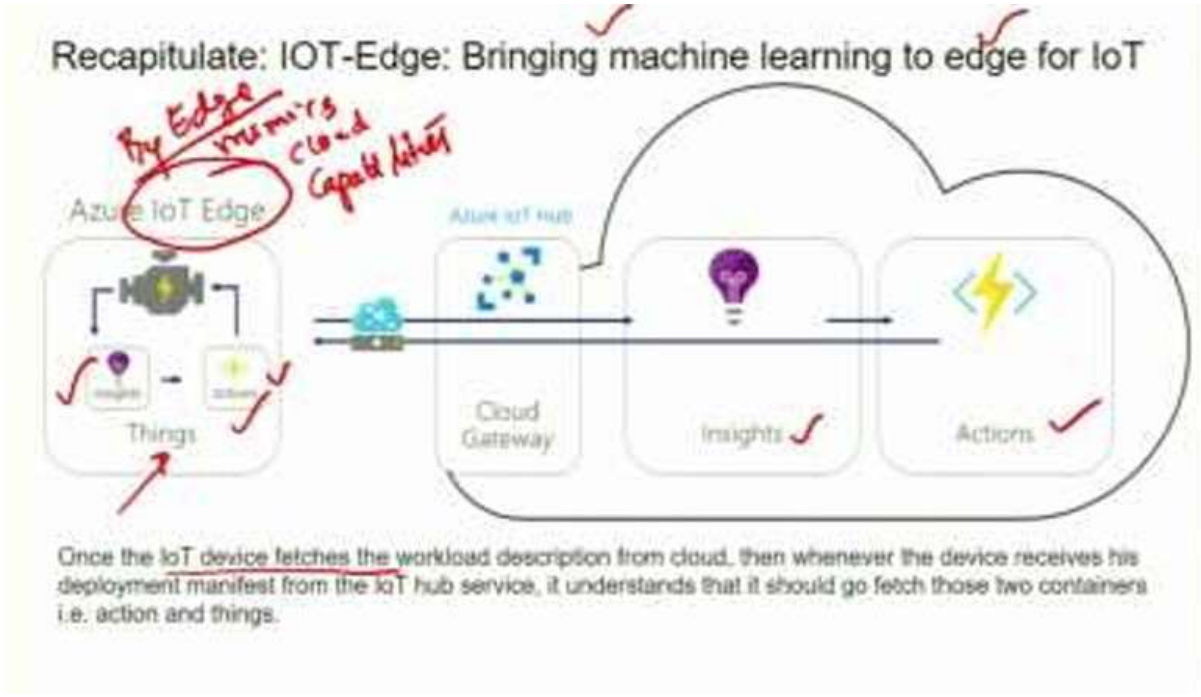
Actionable insight layer responsible for sending an alert to the relevant stakeholder the dashboards and showing some visualizations even the edge taking an action immediately shut down a controlling activator. And again, actionable insight takes place on the edge. So, therefore, you can see that these visualizations dashboard, and human-machine interaction, they are all supported at the edge itself.

Differences between Edge AI and Cloud AI

Ml on cloud vs ml on edge

Ml on cloud AI	Ml on edge AI
• Remote monitoring and control • Merging remote data across multiple IOT devices • Near infinite storage to train machine learning and other advanced ml models	• Low accuracy sight control loops require near real-time response • Pre-process data on promise • Intelligence on edge • Office operations • Data privacy and IP section

Brining machine learning to edge for IOT:



Enabling Intelligence at Edge layer for IOT:

To manage the increasing amount of data that is generated by the devices , sensors , most of the business logic is now applied at the edge instead of the cloud to achieve low-latency and have faster response time for IOT device using machine learning at edge.

Edge layer is delivering three essential capabilities

1.local data processing

2.filitered data transfer to the cloud

3.faster decision-making

1.**local data processing:**

- ❖ In order deal with increasing amount of dat generated by sensor,most of the business logic is now deployed at the edge layer instead of cloud to ensure low-latency and faster response time.

2.**Filitered data transfer to cloud**:

- ❖ This edge computing approach significantly saves the bandwidth and cloud storage.

3.**Faster decision-making:**

- ❖ Ai has enabled new capabilities for edge computing.
- ❖ Most of the decision-making is now taking advantage of artificial intelligence.

The edge layer is becoming the perfect destination for deploying machine-learning model trained in the cloud.

Use Cases of Edge AI – Smart Cities, Healthcare, IoT, and Industry 4.0

INTRODUCTION:

Edge AI is transforming smart cities, healthcare ,IOT, and industry 4.0 by enabling real-time decision-making, reducing latency,and improving efficiency.

KEY USE CASES:-

1. Smart Cities:-

Traffic Management: AI at the edge processes live video feeds from cameras to optimize traffic lights, and detect accidents instantly.

Energy Optimization: Smart grids use edge AI to balance demand and supply.

Waste Management: Sensors with edge AI predict waste bin fill levels.

2. Health care:-

Smart hospitals: ICU devices stream patient data to edge AI system for real-time on critical conditions.

Medical imaging: Diagnostics by processing scans (X-rays, MRI's).

3. IOT (Internet of things):-

Efficiency Energy: Smart meters and HVAC systems optimize usage locally.

Smart homes: Edge AI enables voice assistants, security cameras.

Predictive maintenance: IOT sensors in machinery detect wear and tear early.

4. Industry 4.0:-

Smart manufacturing: Edge AI analyze production line data in real time to detect and optimize.

Robotics: Collaborative robots use edge AI for decision-making in assembly lines.

Worker safety: Wearable sensors detect hazardous conditions and alert workers instantly.

Hardware for Edge AI – Edge GPUs, TPUs, FPGAs

(1).Edge GPUs (Graphics Processing Units)

- GPUs are the most common hardware for Edge AI due to their mature software ecosystems and powerful parallel processing capabilities.

ROLE: Excellent for complex deep learning models, high-speed graphics rendering, and real-time video analytics.

Examples: NVIDIA Jetson series (e.g., Orin, Nano).

2. Edge TPUs (Tensor Processing Units)

- TPUs are Application-Specific Integrated Circuits (ASICs) designed from the ground up for accelerating neural network computations.

Role: Optimized for high-speed, low-power inference, particularly for matrix multiplications and convolutions found in deep learning.

Examples:

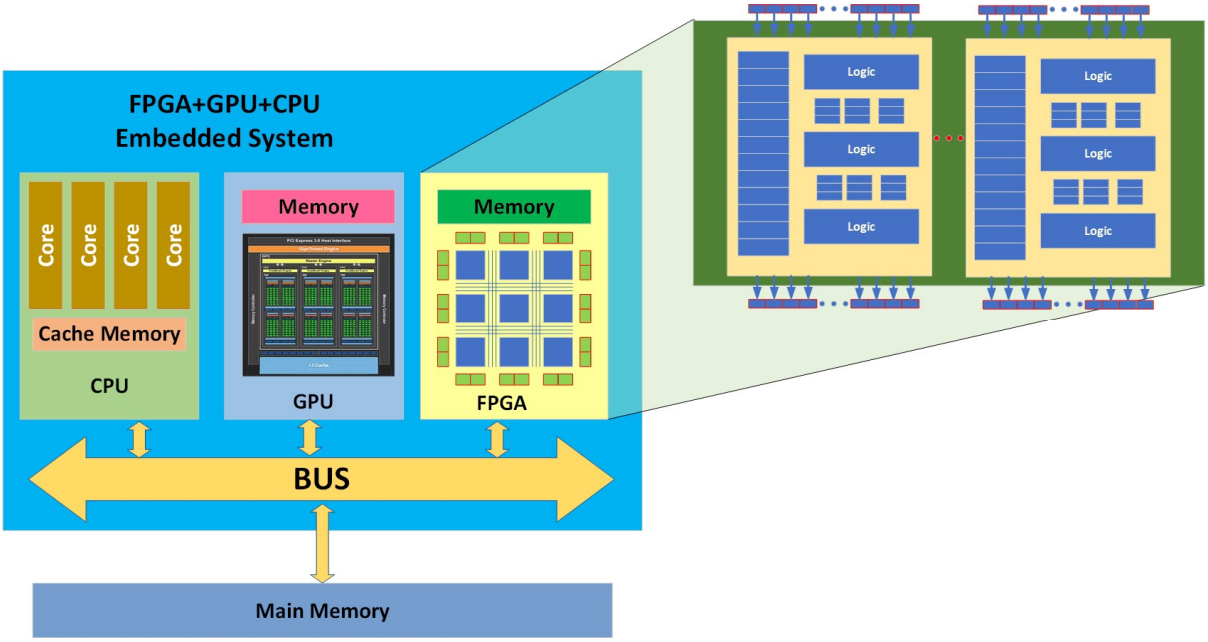
1. Google Coral Edge TPU,
2. Google Trillium TPU

3. Edge FPGAs (Field-Programmable Gate Arrays)

- FPGAs are unique because they can be reprogrammed at the gate level after manufacturing, allowing for custom hardware architectures tailored to specific algorithms.

Role: Used for specialized AI tasks where low latency and deterministic performance are critical, such as 5G networks and industrial robotics

Examples: AMD (Xilinx) Versal and Zynq series, Intel Agilex FPGAs.



Types of Edge Devices – Raspberry Pi, Jetson Nano, Cora

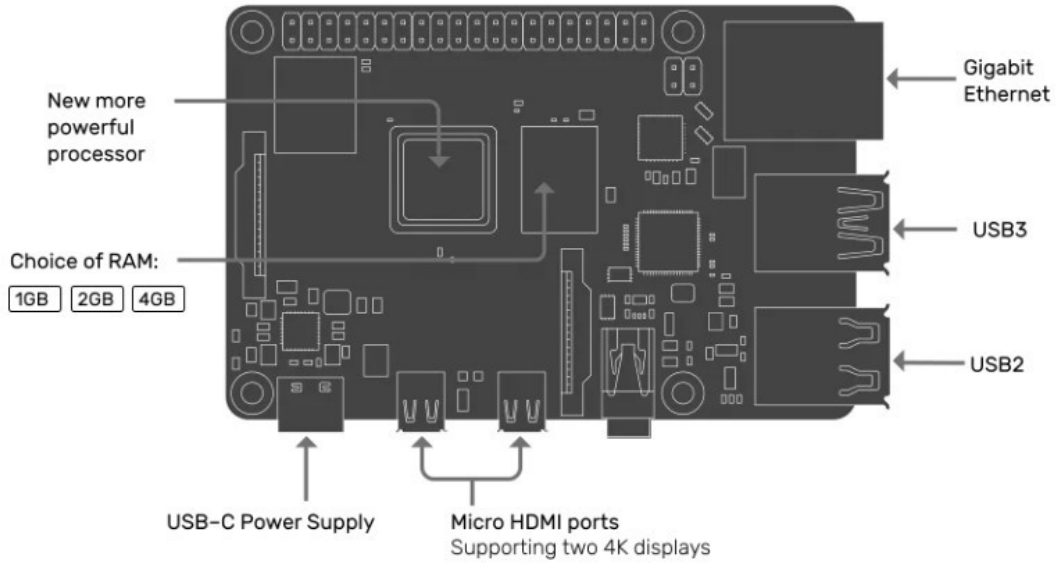
Edge devices in edge computing range from small **IoT sensors, smart cameras, and wearables** processing data locally to more powerful **edge servers.**

RASPBERRY PI :

A Raspberry Pi acts as a versatile, low-cost, credit-card-sized computer for edge computing, enabling data processing and analysis directly at the data source (the "edge") rather than sending everything to the cloud, making it ideal for IoT, robotics, and AI applications by offering local intelligence, reduced latency, enhanced security, and offline functionality.

Examples of Edge Projects:

1. Smart Cameras
2. Predictive Maintenancell
3. Environmental Monitoring
4. Retail Analytics

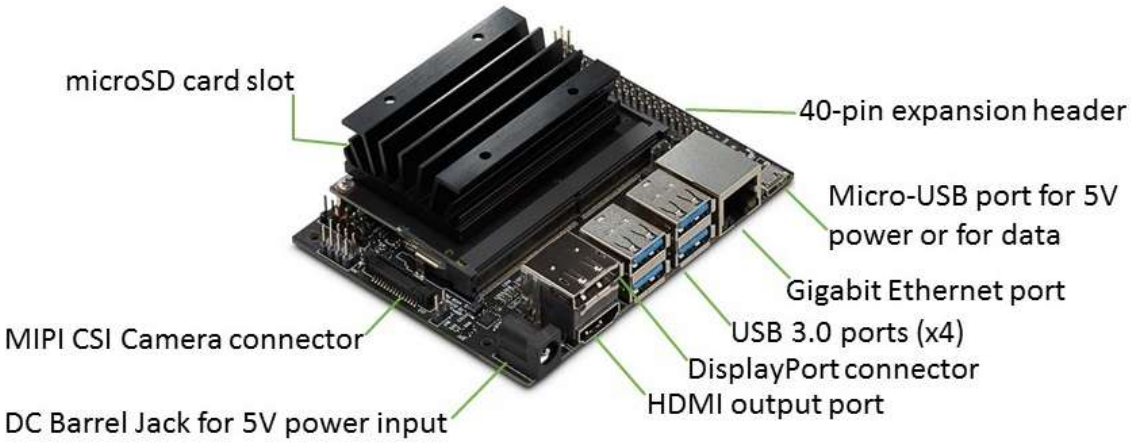


JETSON NANO:

NVIDIAJetson Nano is small, powerful, low-power single-board computer designed for **edge computing** and **AI (Artificial Intelligence)**enabling developers, students, and enthusiasts to run deep learning models and AI inference directly on devices, rather than relying on the cloud

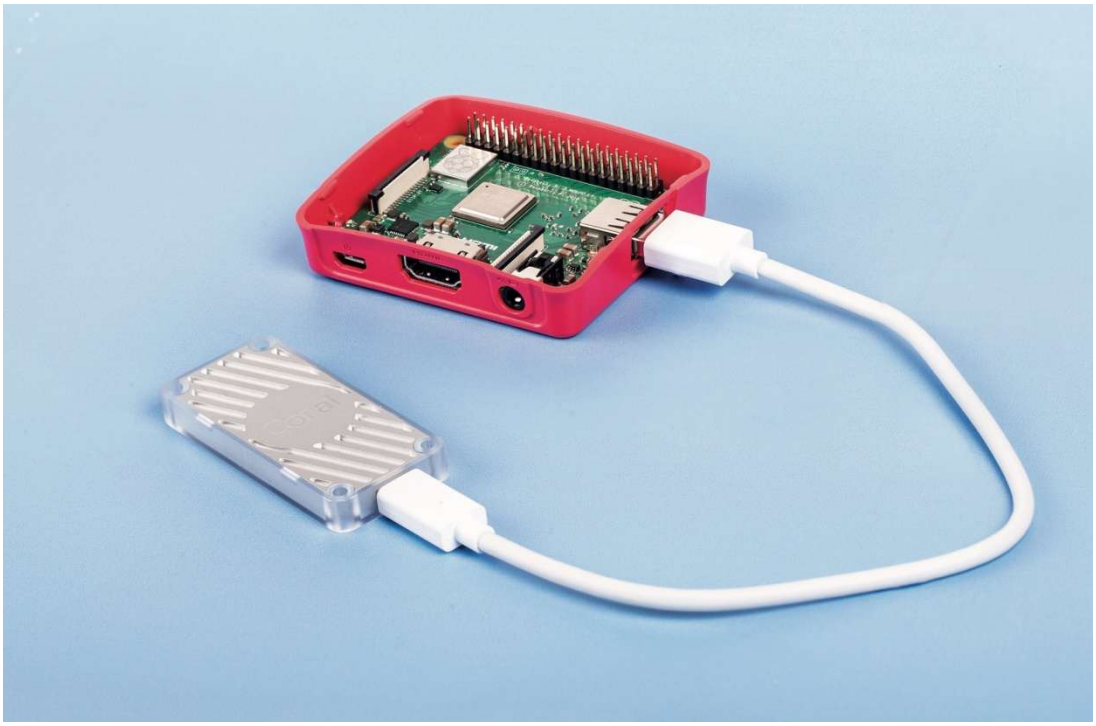
Key Features for Edge AI:

1. GPU-Accelerated Performance
2. Power Efficiency
3. Compact Form Factor



CORAL:

Coral Edge Computing refers to Google's hardware and software platform, centered around the **Edge TPU (Tensor Processing Unit)**, designed for efficient, low-power Machine Learning (ML) inference directly on edge devices (like IoT gadgets, wearables, cameras) for fast, private, and offline AI, running TensorFlow Lite models without relying on the cloud for every task.



Challenges in Deploying AI on Edge

- **Limited memory/compute**
- **Power constraints**
- **Model size too large**
- **Hardware diversity**
- **Security issues**

UNIT II – AI Models and Edge Inference

1. Types of AI Models Suitable for Edge Deployment

Edge devices (Raspberry Pi, Jetson Nano, smartphones, microcontrollers) have **limited CPU, GPU, RAM, power**, so models must be lightweight and efficient.

a) CNN-based Models

- Used for vision tasks.
- Examples:
    - **MobileNet**
    - **SqueezeNet**
    - **ShuffleNet**
    - **EfficientNet-Lite**

b) RNN-based Models

- Used for speech, time-series data.
- Examples:
    - **LSTM**, **GRU** small versions.

c) Transformer-based Tiny Models

- Optimized NLP transformers for edge.
- Examples:
    - **DistilBERT**
    - **MobileBERT**
    - **TinyBERT**

d) TinyML Models

- Extremely small models (<100 KB) for microcontrollers (Arduino, ESP32).
- Frameworks like TensorFlow Lite Micro.

---

2. Model Optimization Techniques

(How to make ML models smaller, faster, and suitable for edge devices)

Modern ML models are often **too large and slow** to run on mobile phones, Raspberry Pi, IoT devices, or embedded systems.
Model optimization helps us **reduce size, improve speed, and lower power consumption** with minimal loss of accuracy.

a) Quantization

- **What is Quantization?**

Quantization means **reducing the numerical precision** used to store model weights and activations.

- **Example**

Instead of using **32-bit floating point (FP32)** numbers, we use **lower precision** numbers:

FP32 → FP16 → INT8 → INT4

- **Why this helps**
  - Smaller numbers need **less memory**
  - Calculations become **faster**
  - Less power is consumed (important for battery devices)
- **Benefits**

✓ Model size reduces by **2× to 4×**
✓ Inference speed increases (especially on **ARM CPUs**)
✓ Lower energy consumption

- **Simple analogy**

Think of money:

- ₹100.5678 (FP32)
- ₹100.56 (FP16)
- ₹101 (INT8)

For most decisions, **₹101 is good enough**, and it's much easier to handle.

- **Tools**
  - **TensorFlow Lite (TFLite)**
  - **ONNX Quantization Toolkit**
  - **PyTorch Quantization**

🔹 **Where used**

- Mobile apps
- Edge AI (Raspberry Pi, Jetson Nano)
- Real-time inference systems

b) Pruning

◇ What is Pruning?

Pruning means **removing unnecessary parts of a neural network** that contribute very little to the final output.

◇ Key idea

Not all neurons or connections are equally important.

🔹 **Types of Pruning**

1️⃣ **Unstructured Pruning**

- Removes **individual weights**
- Creates sparse matrices
- Less hardware-friendly

Example:

Weight = 0.00001 → removed

2️⃣ **Structured Pruning (Better for hardware)**

- Removes **entire neurons, filters, or channels**
- Works well on GPUs and CPUs

Example:

Remove one full CNN filter

🔹 **Benefits**

✓ Model size reduced by **30% – 90%**
✓ Less computation
✓ Faster inference

🔹 **Simple analogy**

Imagine a classroom:

- Some students never participate

- Removing them does not affect learning outcome

c) Knowledge Distillation

🔹 **What is Knowledge Distillation?**

A **large, powerful model (Teacher)** trains a **smaller model (Student)**.

The student learns:

- Final predictions
- Probability distributions
- Decision patterns

🔹 **Why use it?**

Instead of training a small model from scratch, we **learn from a smarter model**.

🔹 **Benefits**

✓ Student model is:

- Smaller
- Faster
- Almost as accurate

🔹 **Real examples**

Teacher	Student
BERT	DistilBERT
ResNet	MobileNet
Large CNN	Tiny CNN

🔹 **Simple analogy**

- Teacher explains concepts deeply
- Student learns shortcuts
- Student answers quickly but correctly

d) Transfer Learning

🔹 **What is Transfer Learning?**

Reuse a **pretrained model** and **train only the last layers** for a new task.

🔹 Why it works

Early layers learn **general features**:

- Edges
- Shapes
- Colors

Only final layers need retraining.

🔹 Benefits

✓ Much faster training
✓ Requires less data
✓ Works very well on edge devices

🔹 Common applications

✓ Face recognition
✓ Crop disease detection
✓ Vehicle detection
✓ Medical image classification

🔹 Example

ImageNet-trained CNN

↓

Replace last layer

↓

Train on crop disease dataset

Summary Table (Great for exams)

Technique	Main Goal	Benefit
Quantization	Reduce precision	Smaller, faster, low power
Pruning	Remove unnecessary weights	Less computation

Technique	Main Goal	Benefit
Knowledge Distillation	Teach small model	Fast + accurate
Transfer Learning	Reuse pretrained model	Saves training time

---

3. Inference Acceleration Using Edge Hardware

a) Edge TPUs

- Google Coral Edge TPU
- Optimized for **INT8** models
- Extremely fast inference

b) GPU Accelerators

- Nvidia Jetson Nano / Xavier NX
- Supports TensorRT optimization

c) NPUs / AI Accelerators

- Smartphone AI engines (Qualcomm Hexagon DSP, Apple Neural Engine)

d) FPGA-based Acceleration

- Low-power programmable hardware for custom edge AI applications.

---

4. Lightweight Models for Edge AI

1️⃣ MobileNet – Practical View

What is MobileNet?

MobileNet is a CNN designed for mobile & edge devices.

Key idea (simple)

👉 **Replace heavy convolution with Depthwise Separable Convolution Instead of:**

Normal Convolution = Slow + Heavy

MobileNet uses:

Depthwise Convolution + Pointwise (1×1) Convolution

Why this helps?

- 8–9× fewer calculations
- Smaller model
- Faster inference

Where MobileNet is used (REAL examples)

- Mobile camera face detection
- Object detection on Jetson Nano
- Smart cameras
- Android apps

Practical pipeline

```
Camera Image
    ↓
Resize (224×224)
    ↓
MobileNet
    ↓
Class / Bounding box
```

Performance

Device	FPS
Mobile phone	30–60 FPS
Jetson Nano	20–30 FPS

2️⃣ SqueezeNet – Practical View

What is SqueezeNet?

SqueezeNet is a very small CNN model that gives AlexNet-level accuracy with much smaller size**.**

Key idea

👉 Use Fire Modules

Fire Module =

Squeeze layer (1×1 conv)

Expand layer (1×1 + 3×3 conv)

This reduces:

- Parameters
- Model size

Why SqueezeNet?

- Model size ≈ 5 MB
- Good accuracy
- Faster than older CNNs

Where SqueezeNet is used

- Embedded vision systems
- Industrial inspection
- Older edge devices

Practical pipeline

```
Image
  ↓
SqueezeNet
  ↓
Classification
```

---

3 TinyML – Practical View (Very Important)

What is TinyML?

👉 **Running ML models on microcontrollers** (ESP32, STM32, Arduino)

TinyML is **not one model**, it is a **method**.

Typical TinyML models

- Small CNNs
- Decision trees
- Linear models

Model size

- 10 KB – 300 KB
- Stored in Flash

Real TinyML example (ESP32)

Use case: Voice keyword detection

Microphone

↓

MFCC feature extraction

↓

Tiny CNN

↓

ON / OFF command

Why TinyML works

No GPU needed
Low power (battery)
Real-time inference

TinyML limitations

❌ Cannot run MobileNet fully

❌ Very limited memory

❌ Simple tasks only

MobileNet vs SqueezeNet vs TinyML

Feature	MobileNet	SqueezeNet	TinyML
Runs on	Mobile / Edge	Edge	Microcontrollers
Model size	5–20 MB	~5 MB	<300 KB
Power	Medium	Medium	Very Low
Accuracy	High	Medium	Low–Medium

Feature	MobileNet	SqueezeNet	TinyML
Use cases	Vision	Vision	Sensor / Audio

---

5. Frameworks for Edge Deployment

a) TensorFlow Lite

- Convert full TensorFlow models to light versions.
- Supports:
    - Quantization
    - Edge TPU
    - TFLite Micro (for microcontrollers)

b) ONNX Runtime

- Universal format
- Runs on many devices (GPU, CPU, FPGA, NN accelerators)

c) OpenVINO (Intel)

- Optimized for Intel CPUs, VPUs (Myriad-X), integrated GPUs
- Good for computer vision

d) PyTorch Mobile / ExecuTorch

- PyTorch edge inference toolkit
- Works for Android, iOS, embedded devices.

---

6. Compilation Tools for Edge AI

a) Apache TVM

- Converts models to optimized hardware-specific binaries.
- Supports:
    - ARM

    - CUDA
    - OpenCL
    - CPU/GPU/FPGA

b) Glow (Facebook)

- Neural network compiler
- Generates accelerator-ready code

c) XLA (Accelerated Linear Algebra)

- Google compiler used for TensorFlow

d) TensorRT

- Nvidia inference accelerator
- Converts FP32 → FP16 / INT8 engines

---

7. Energy and Latency-aware Inference

Important for battery-powered and real-time edge systems.

Techniques

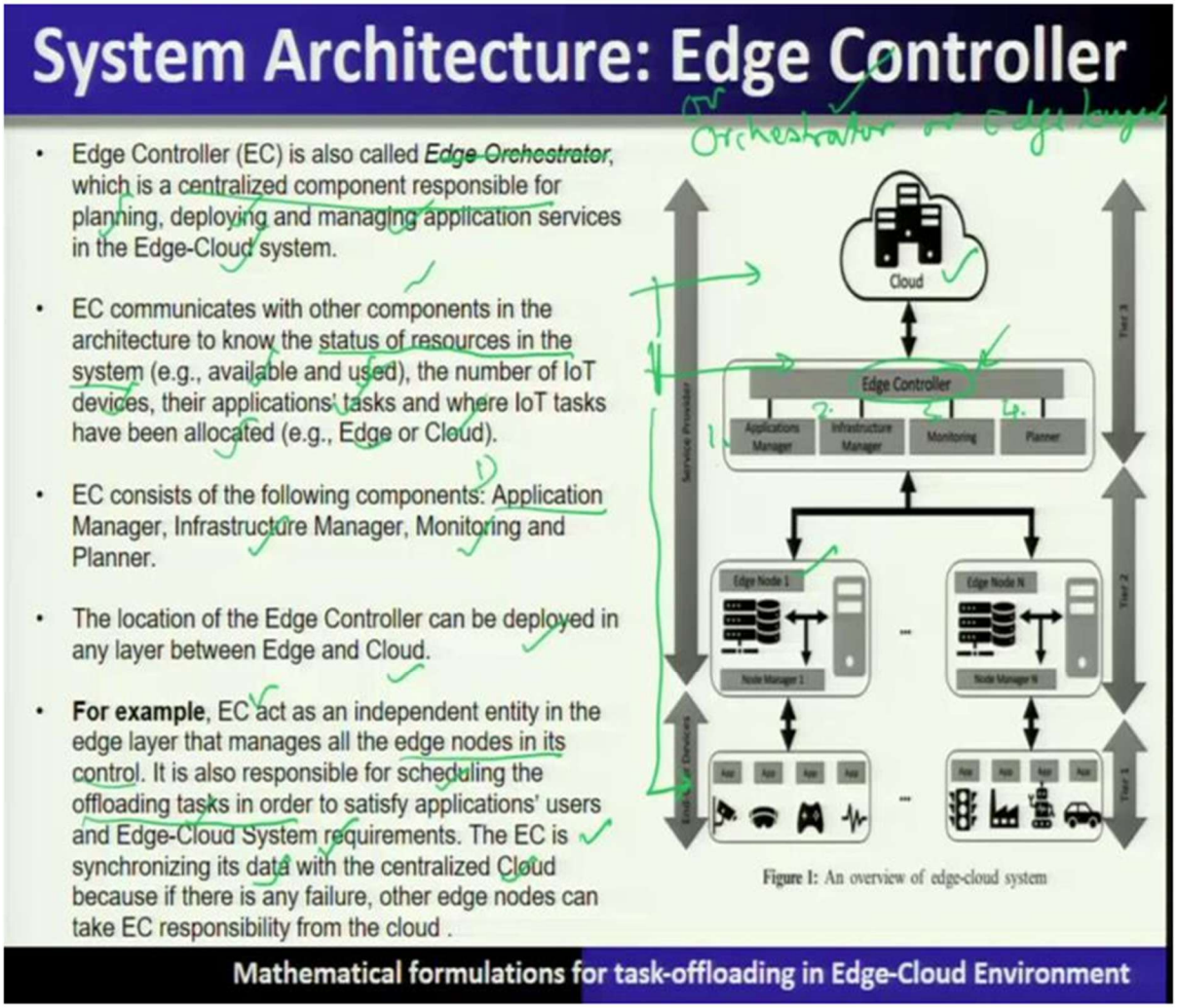
- Use **quantized models** (INT8 → lowest power)
- Use **smaller batch sizes** (real-time applications)
- Use **hardware accelerators** (GPU/NPU/TPU)
- Use **model caching**
- Perform **on-device postprocessing** to minimize cloud communication.

Measurements

- **Latency**: time per inference (ms)
- **Throughput**: inferences per second (IPS)
- **Energy per inference** (Joules)

UNIT III – Edge-Centric Architectures and Data Management

Distributed AI Architectures: Edge, Fog, and Cloud



Latency sensitive application arch and application:

- Edge controller (EC) is also called Edge orchestrator ,which is a centralized component responsible for planning, deploying and managing application services in the edge-cloud system.
- Ec consists of the following components: application manager, infrastructure manager, monitoring and planner.
- The location of the edge controller can be deployed in any layer between edge and cloud

The application manager:

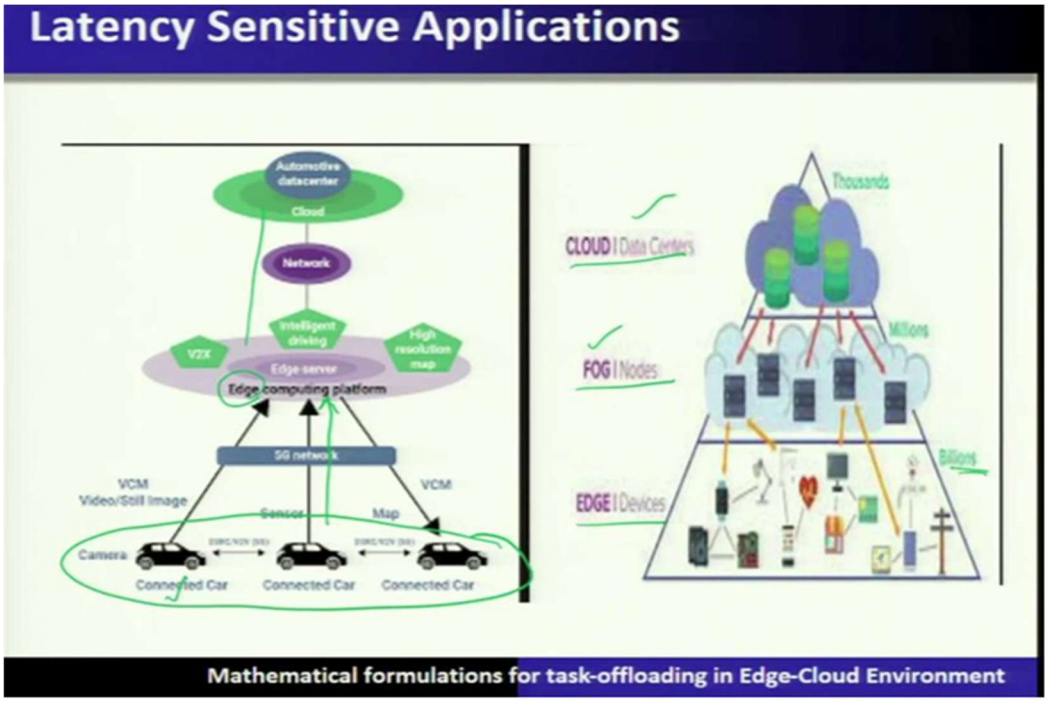
- It is responsible for managing application running in the edge-cloud system.
- Mange application running in edge-cloud system
- Application need resources such as cpu on among etc
- Support from edge cloud

The infrastructure manager:

- the role of the infrastructure manager is to be in charge of the physical resources in the entire edge-cloud system. for instance, processors, networking and the connected iot devices for all edge nodes
- the main responsibility of this components is to monitoring application tasks(eg., computational delay and communication delay) and computational resources

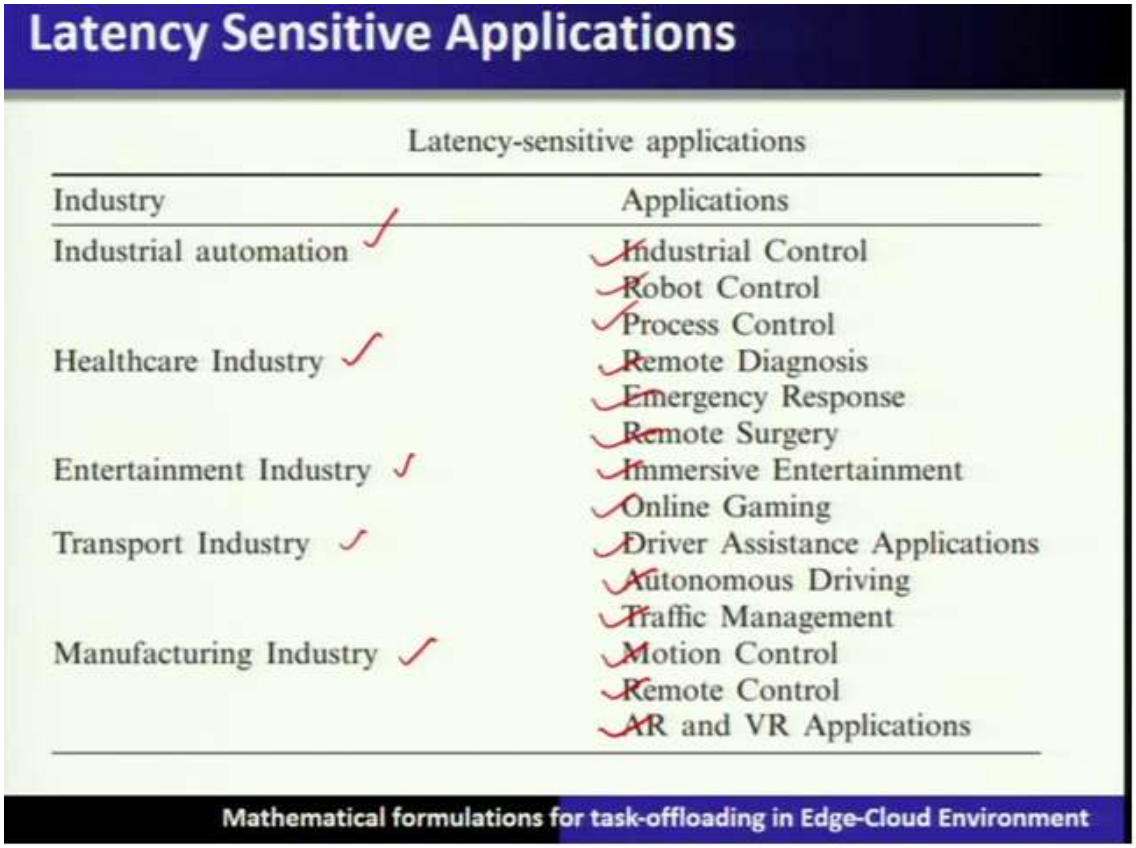
planner:

the main role of this components is to propose scheduling policy of the offloading tasks in the edge-cloud system and the location where they where they will be placed(eg., local edge ,other edges or the cloud)



- latency-sensitive application have high sensitivity to any delays accrue in communication or computation during the interaction with the edge-cloud system
- first, critical application ,which must be processed in the cars computational resources, for instance, autonomous driving and road safety application

- second, high-priority application, which can be offloaded but with minimum latency, such as image aided navigation, parking navigation system and traffic control.
- Eg., Entertainment , multimedia, and speech processing
-



Collaborative Intelligence – Edge-Cloud Offloading Strategies

Vertical And Horizontal Offloading For Cloud-Edge

Introduction:

1.Edge computing is a paradigm that enables virtualized computational and communication resources to be deployed near the source of service workloads
Instead of relying on massive data centers

2.This allows for a reduction in end-to-end delay for accessing this resources and makes it more suitable for real time services

3.Additionally,edge computing enables virtualized resources to be geographically distributed which can address the requirements of mobility and geo-distribution of mobile and iot services

Cloud-Edge Computing:

1.Cloud edge computing can efficiently accommodate different types of services
With end devices and network edges suitable for real time services

2.Integration of cloud and edge computing is proposed to take advantage of the benefits both technologies offer

3.Cloud-edge computing should consider both vertical and horizontal offloading between services nodes

EX :

- A smart home system that utilizes edge computing cloud provide a more secure,efficient and cost effective solution for controlling and monitoring devices such as lights,cameras,doorlocks

- The system would have a gateway device,such as a router,that would provide a local connection for each device

Vertical offloading :

*Vertical offloading refers to the process of transferring tasks or services from cloud or data centers to edge nodes in orders to reduces latency.

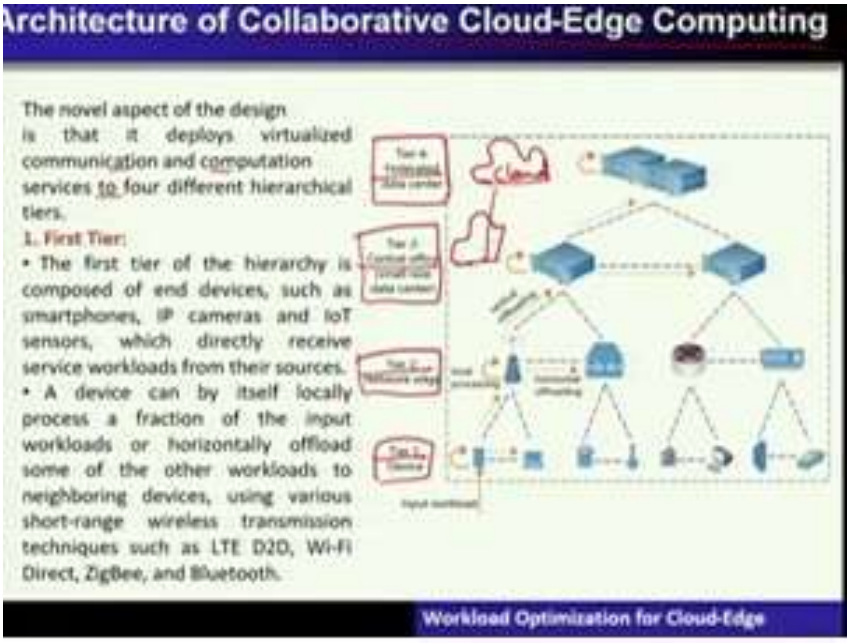
*It is also known as cloud-edge computing and it is used to reduce burden of the cloud.

Horizontal offloading :

*Horizontal offloading on the other hand, is the process of transferring tasks of services between edge nodes in order to reduce latency.

*It is used to improve the capacity of edge nodes and can also be used to reduce the load on the cloud

Architecture of Collaborative Cloud-Edge Computing:



First tier :

*The first tier of the hierarchy is a composed of end devices, such as smartphones,ip cameras and iot sensors.

Second tier :

*The second tier comprises access network technologies such as internet , wi-fi and 4g/5g.

*The edge nodes are capable of processing part of workloads.

Third tier :

*The third tier consists of horizontal and vertical offloading from the edge nodes to the central offices.

Fourth tier :

*The fourth tier consisting of horizontal offloading from the central offices to neighbouring central offices and vertical offloading remote data centre.

*The data center is the top-most tier of the cloud-edge computing hierarchy and is responsible for processing the remaining workloads.

As for 1st MID