The City College of New York

CSC 36000: Modern Distributed Computing *with AI Agents*

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at City University of New York

November 12, 2025 CSC 36000

Today's Lecture

Distributed Multimodal AI Agents

Addressing Distributed Challenges in Multimodality

Class Research Project Guidance

Dealing with Distributed Multimodal AI

- Use ffmpeg library to generate image frames/textual instance from multimodal inputs
- You can parallelize over large videos using pre-processing followed by multi-processing
- Download large datasets for your project with wget command!
- Do not wait for the data and model downloads till the end of the semester!
- You can code on Colab (alternatively on VSCode)
- Run smaller AI models parallely to showcase the success of a Distributed Approach

Multimodal AI Agents

What is Multimodality?

So far, many of the problems we've looked at have focused on a single particular modality (data type).

This could be text in LLMs, images in a object detection model, or sensors in a home IoT device such as a thermostat.

Multimodality is a property given to models that deal with more tha one type of data. For example:

- Autonomous Vehicles have video, sound, LiDAR, radar, etc.
- Smart Glasses have video, images, sound, etc.

Examples of Modalities

There are many, many kinds of data that AI models can use, including but not limited to:

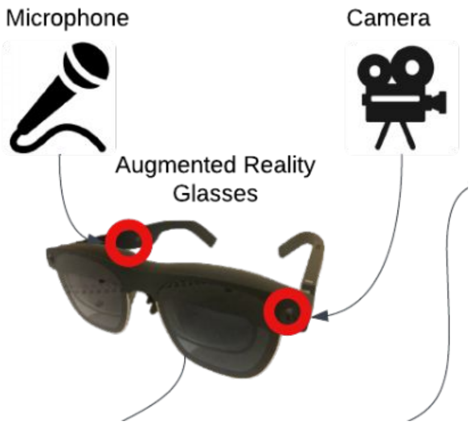
- **Language:** Written text and/or spoken audio commands
- **Visual:** Images and/or dynamic video feeds
- **Audio:** Speech, environmental noises, music, etc.
- **Sensors:** LiDAR, radar, IMU (inertial measurement units), etc.

By combining two or more of these modalities, we can move AI towards a more human-like perception of the world.

Real-World Use Case: Personal AI Assistants

Personal AI Assistants often exist on a user's personal device ("edge device") such as a smartphone, smart glasses, smart watch, etc.

A prominent example is Google Deepmind's Astra Augmented Reality Glasses or Meta's Smart Ray-Ban Glasses



Real-World Use Case: Autonomous Vehicles

Self-driving cars are highly multimodal. In a Waymo car we have:

- High-resolution cameras
- LiDAR (Light Detection and Ranging) Sensors
- Radar
- Microphones

Tesla cars are known as “vision-only” but even they combine cameras microphones, map data, and sensors in their end-to-end decision-making.

Connection to Distributed Computing

While Personal AI Assistants and Autonomous Vehicles are both Multimodal AI Assistants, they occupy different ends of the failure tolerance and reliability spectrum:

- Personal AI Assistants are optimized for energy efficiency and low-latency; it’s okay for them to have eventual consistency if that means less computation, and failures might cause a bad user experience.
- Autonomous Vehicles need to have high availability and Byzantine fault tolerance or they will be unsafe to use.

Addressing Challenges in Distributed Multimodality

Challenge: Data Fusion and Synchronization

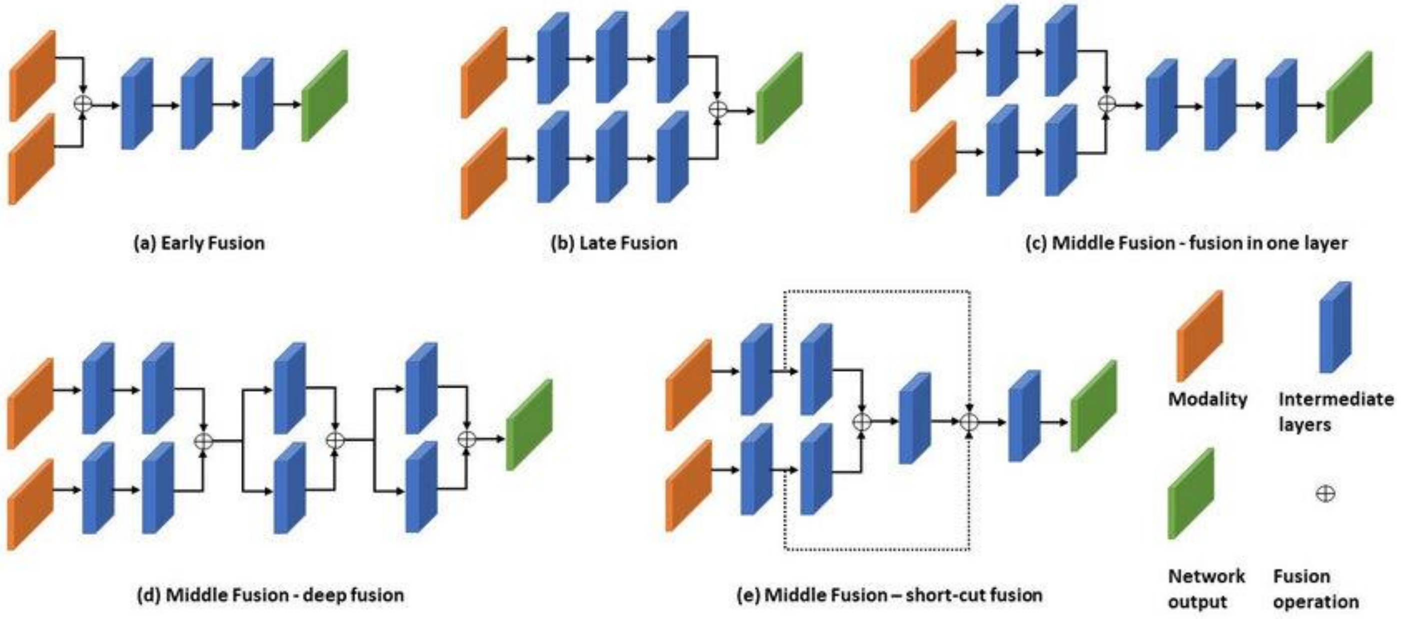
Problem: Multimodal AI Agents get data from different sensors at different rates and in different formats. How do we fuse them together?

Example: Consider a self-driving car. The AI Agent needs to decide “Is an emergency vehicle approaching?" using three inputs:

- **4K Camera (Visual):** 30fps (one frame every 33.3ms)
- **LiDAR (Sensory):** 10fps (one scan every 100ms)
- **Microphone (Audio):** 44.1kHz stream, processed in 10ms chunks

If the microphone detects a siren, should the car move aside? What if the video frame doesn’t show an emergency vehicle?

Types of Multimodal Fusion

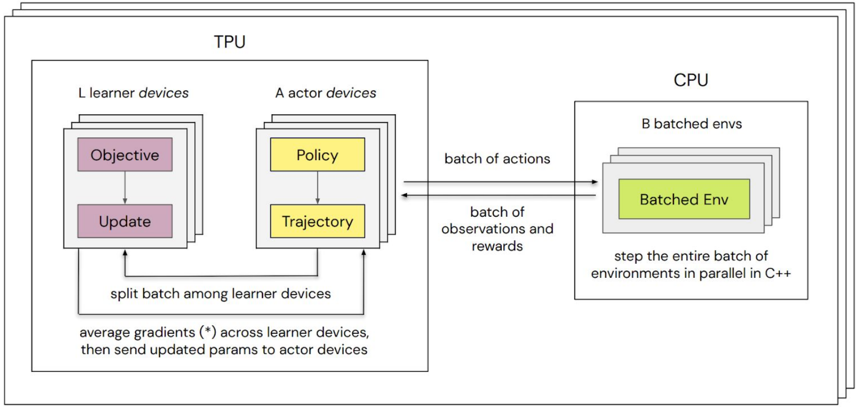


(a) Early Fusion
(b) Late Fusion
(c) Middle Fusion - fusion in one layer
(d) Middle Fusion - deep fusion
(e) Middle Fusion – short-cut fusion
Modality
Intermediate layers
Network output
Fusion operation

Challenge: Heterogeneous Compute Scheduling

Modern AI Agents are hardly ever only on a CPU. They integrate different processors, and different modalities can be on different processors:

- **CPU:** Controlled logic (e.g. coordinating data streams)
- **GPU:** Highly parallel tasks (e.g. video streams)
- **NPU/TPU:** Neural network tasks (e.g. text-to-speech, multimodal fusion)



TPU
L learner *devices*
A actor *devices*
Objective
Update
Policy
Trajectory
split batch among learner devices
average gradients (*) across learner devices, then send updated params to actor devices
batch of actions
batch of observations and rewards
CPU
B batched envs
Batched Env
step the entire batch of environments in parallel in C++

This entire computation is replicated across S slices of a TPU Pod, in which case gradients in (*) are averaged across all learner devices of all slices

Adaptive Resource Utilization

Teach the system to offload computational tasks adaptively depending on what resources are available.

Example: Let's say you have Intelligent AR Glasses that have Cloud Connectivity to powerful GPUs and less powerful on-device resources:

- If the network connection is strong (e.g., fast Wi-Fi), the framework may offload the entire task to the powerful cloud.
- If the network connection is weak (e.g., 4G), it may use a "split inference" model with a nearby edge server.
- If the network is non-existent, it will fall back to a smaller, less-powerful model running entirely on-device.

Challenge: Bandwidth and Latency

Raw multimodal data is massive: A single autonomous vehicle can generate 1TB of data per hour.

Transmitting high-resolution video, 3D point clouds, and real-time tactile data streams over a network is often infeasible due to bandwidth limitations and latency requirements.

This presents an opportunity for distributed learning!

Distributed Split Inference

Similarly to how adaptive resource usage allocates computational work to either the edge or the cloud, we can distributethe AI model itself across cloud and edge resources:

Split the model into two sections:

- **"Head" (Section 1):** The first few layers of the model run on the edge device (e.g., the camera or car). This part typically performs initial processing and feature extraction.
- **"Tail" (Section 2):** The remaining layers of the model run on a more powerful edge server or in the cloud. This part performs the final, complex reasoning and decision-making.

This allows us to perform complex computation without transmitting vast amounts of data! However, more is needed to enable these multimodal models to run in the edge at all.

Questions?