From the Edge to the Cloud: Exploring AI Inference Across the Computing Continuum

(yes, including Generative AI)

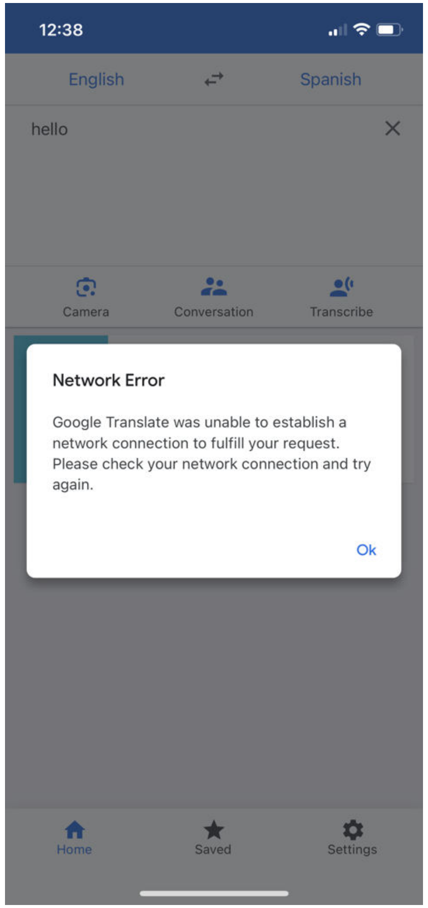
Roberto Morabito
Assistant Professor @ EURECOM
https://www.linkedin.com/in/robertomorabito

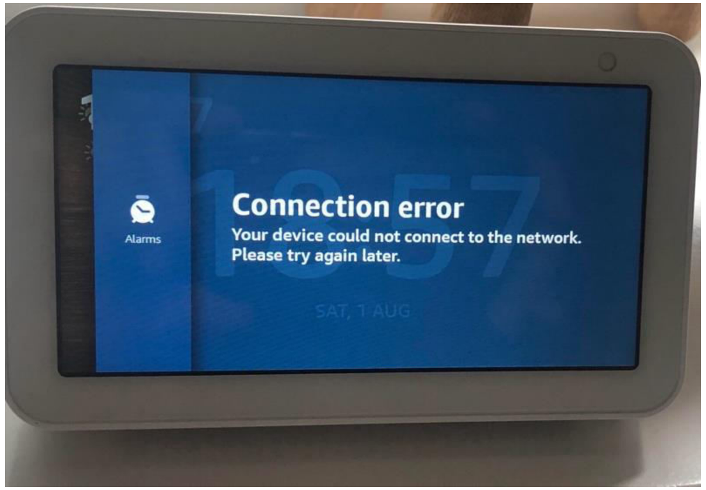
From the Edge to the Cloud: Exploring AI Inference Across the Computing Continuum

(yes, including Generative AI)

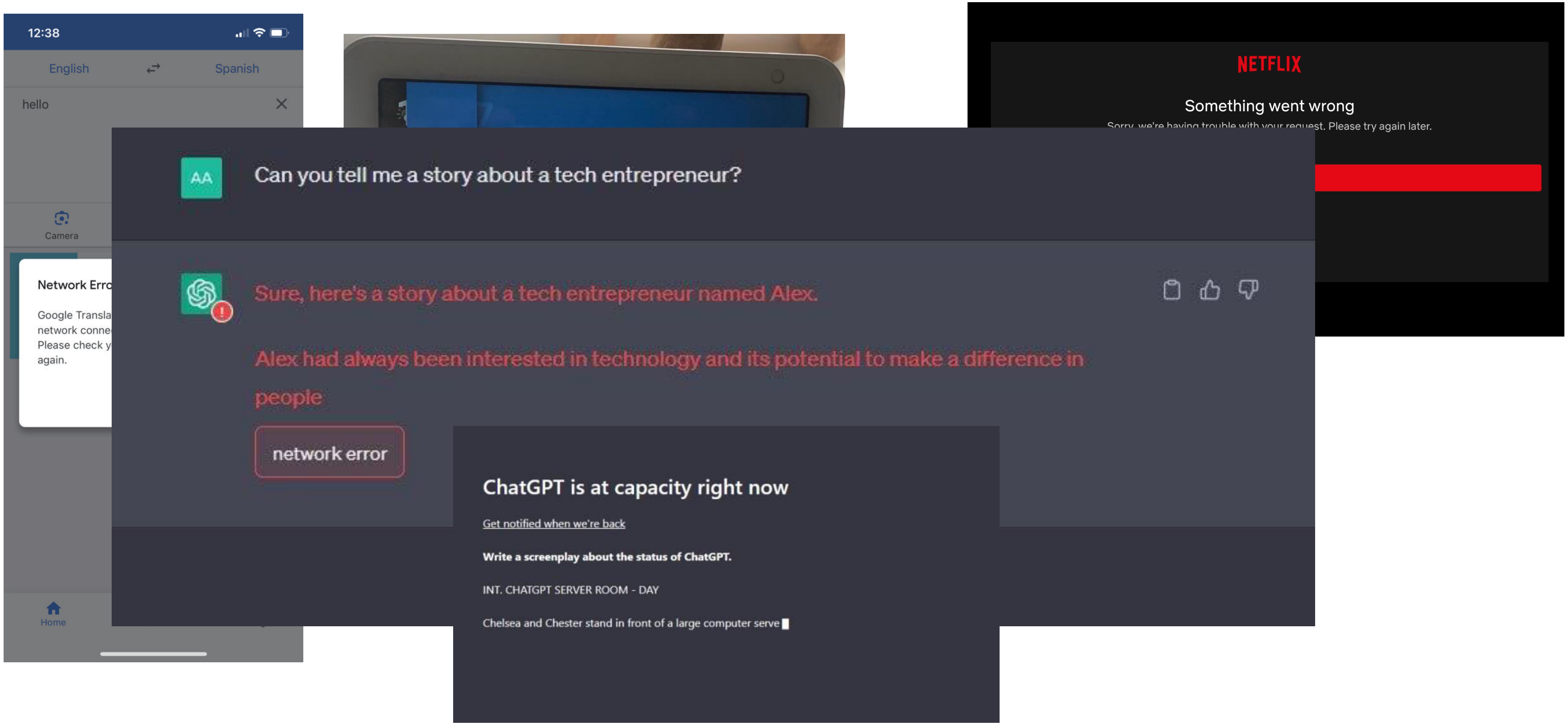
Let's start from the edge

Cloud Computing Is Great But…

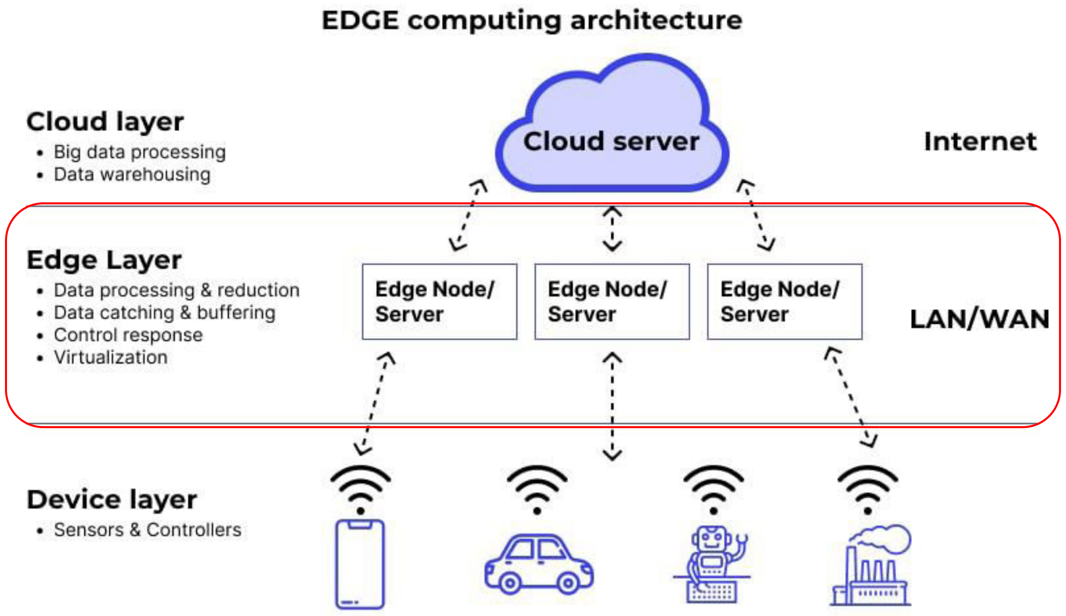




Cloud Computing Is Great But…



Edge Computing



- Idea is to **push applications, data and computing power to the edge of the Internet,** near mobile devices, sensors, and end users

Main Drivers

Latency

- Data processing close to where it originates avoids round-trip time to the cloud

Bandwidth

- Optimization of communication to and from the cloud

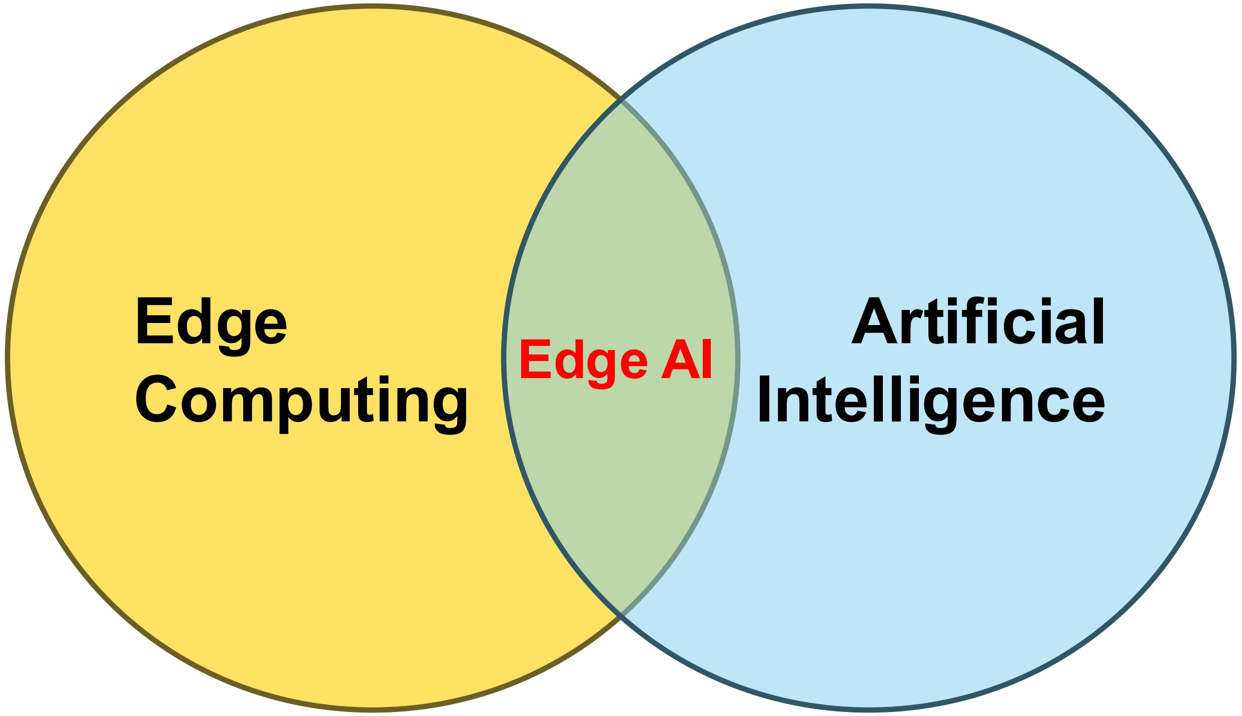
Privacy / Security

- Sensitive data stays local

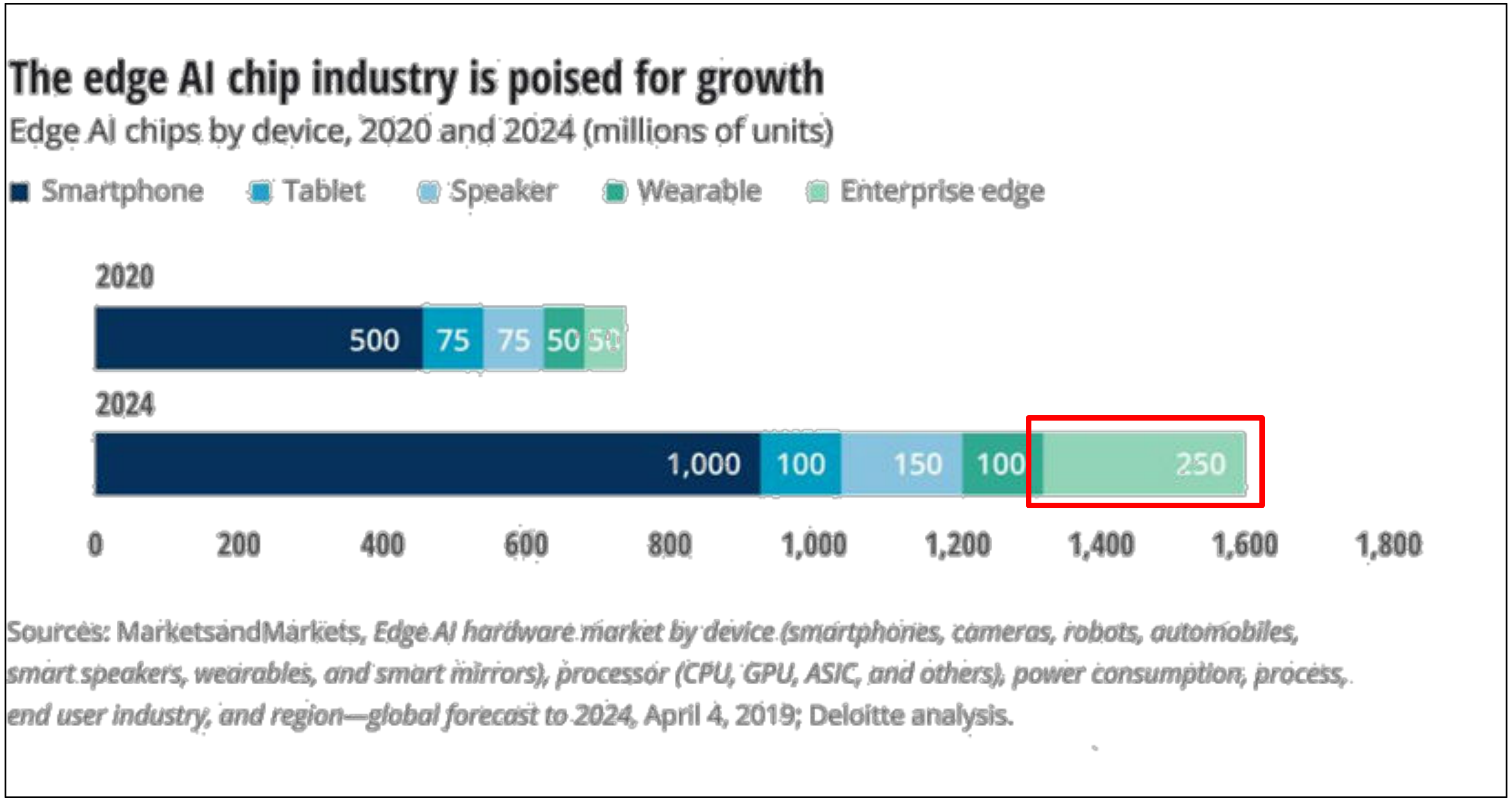
Connectivity

- Continued processing (in some cases) despite lack of connectivity to the cloud

Edge AI

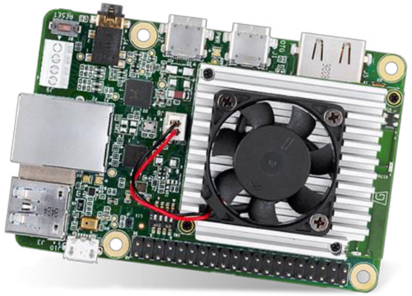


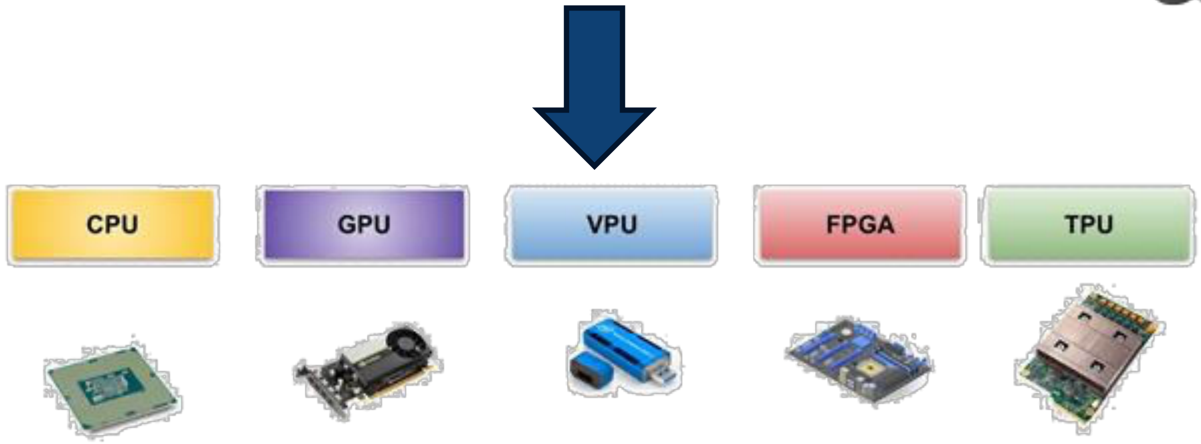
Edge AI Chips MARKET



"Bringing AI to the device: Edge AI chips come into their own " Source: https://bit.ly/3r31lJv

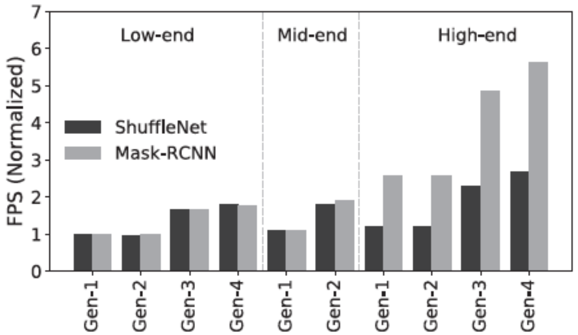
Edge AI Chips – AI Acceleration





Source: https://www.thinkautonomous.ai/blog/vision-processing-units-vpus/

AI Inference At The Edge 2019 Vs. 2024



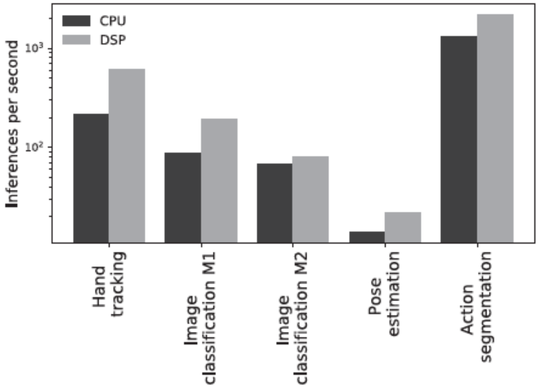


Figure 8: Inference time performance comparison between CPU and DSP.

(Source: Wu, C.J., Brooks, D., Chen, K., Chen, D., Choudhury, S., Dukhan, M., Hazelwood, K., Isaac, E., Jia, Y., Jia, B. and Leyvand, T., 2019, February. Machine learning at facebook: Understanding inference at the edge. In 2019 IEEE international symposium on high performance computer architecture (HPCA) (pp. 331-344). IEEE.)

Samsung bets heavily on AI tricks to boost Galaxy S24 appeal

South Korean firm will hope generative AI text, voice, image and video tools can help it regain top spot in phone market

News

PREVIOUS | NEXT

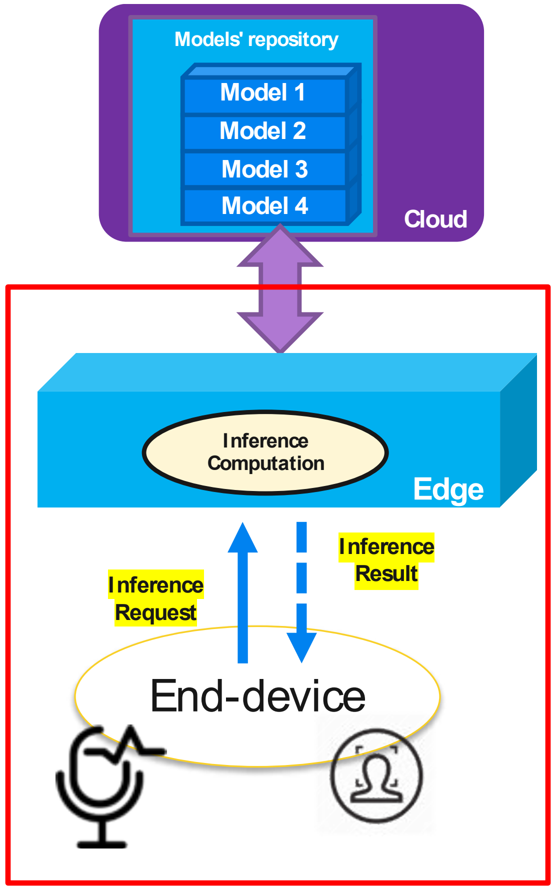
n San Jose

Samsung unveils Galaxy S24 series with AI-powered Camera, Translation, and Editing tools

Samsung says users can decide how much data they want to use for AI features, and opt out of online processing if they want to

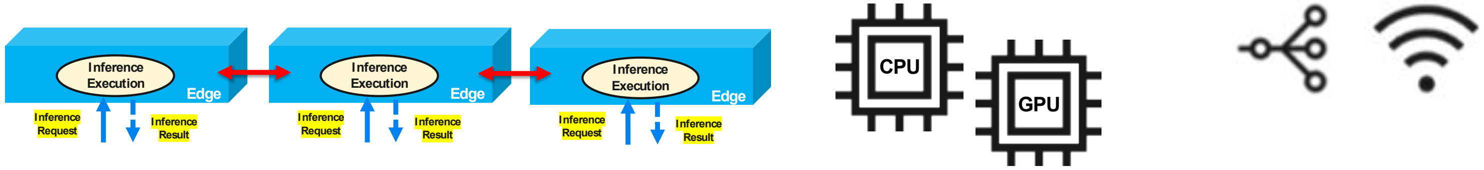
(Source: https://www.theguardian.com/technology/2024/jan/18/samsung-bets-heavily-on-ai-tricks-to-boost-galaxy-s24-appeal and https://mobilesyrup.com/2024/01/17/samsung-s24-series-ai-features/)

AI Inference In Distributed Edge Computing Systems

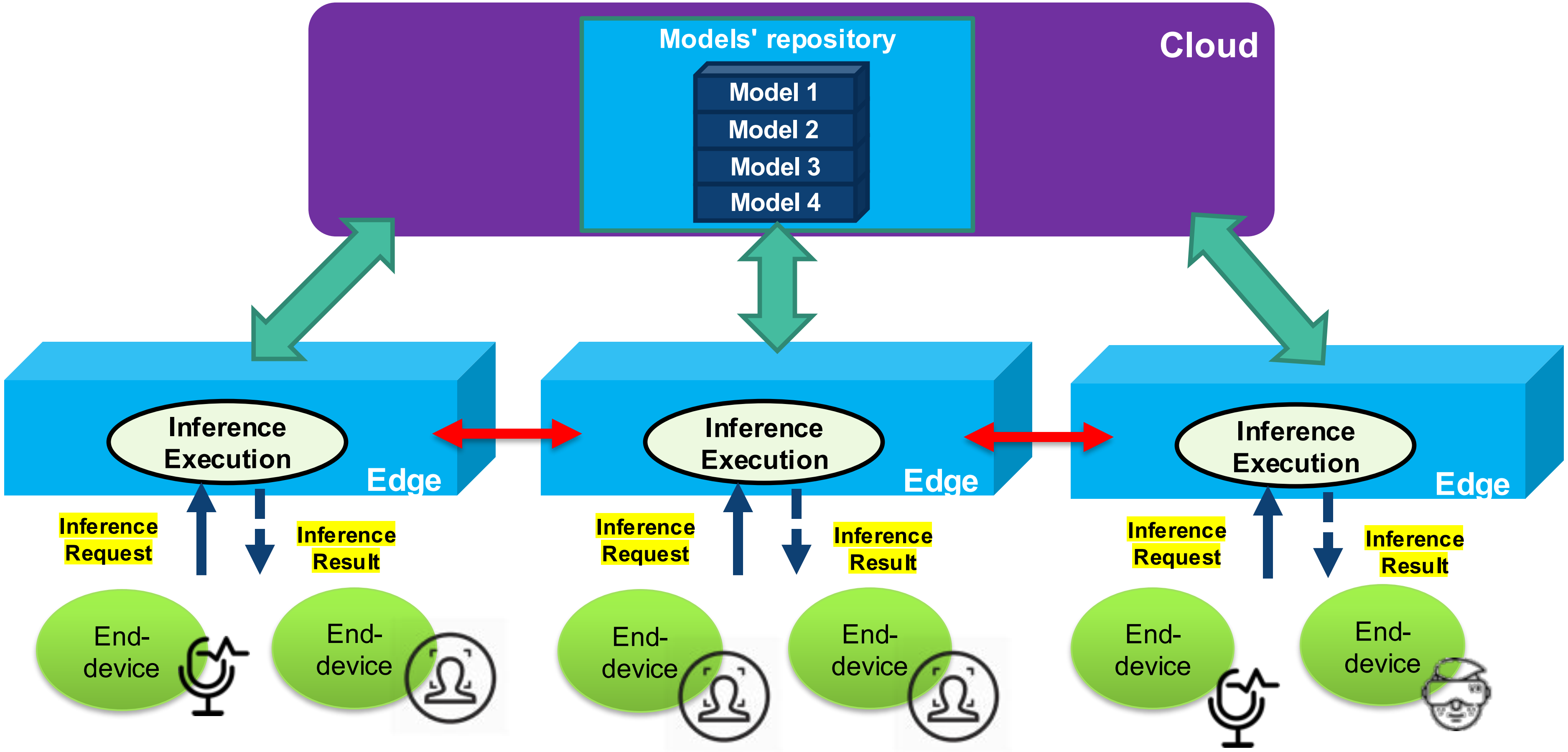


From the perspective of distributed edge AI systems, related studies had primarily focused on theoretical models and simple scenarios involving interactions between a single device or edge and the cloud.

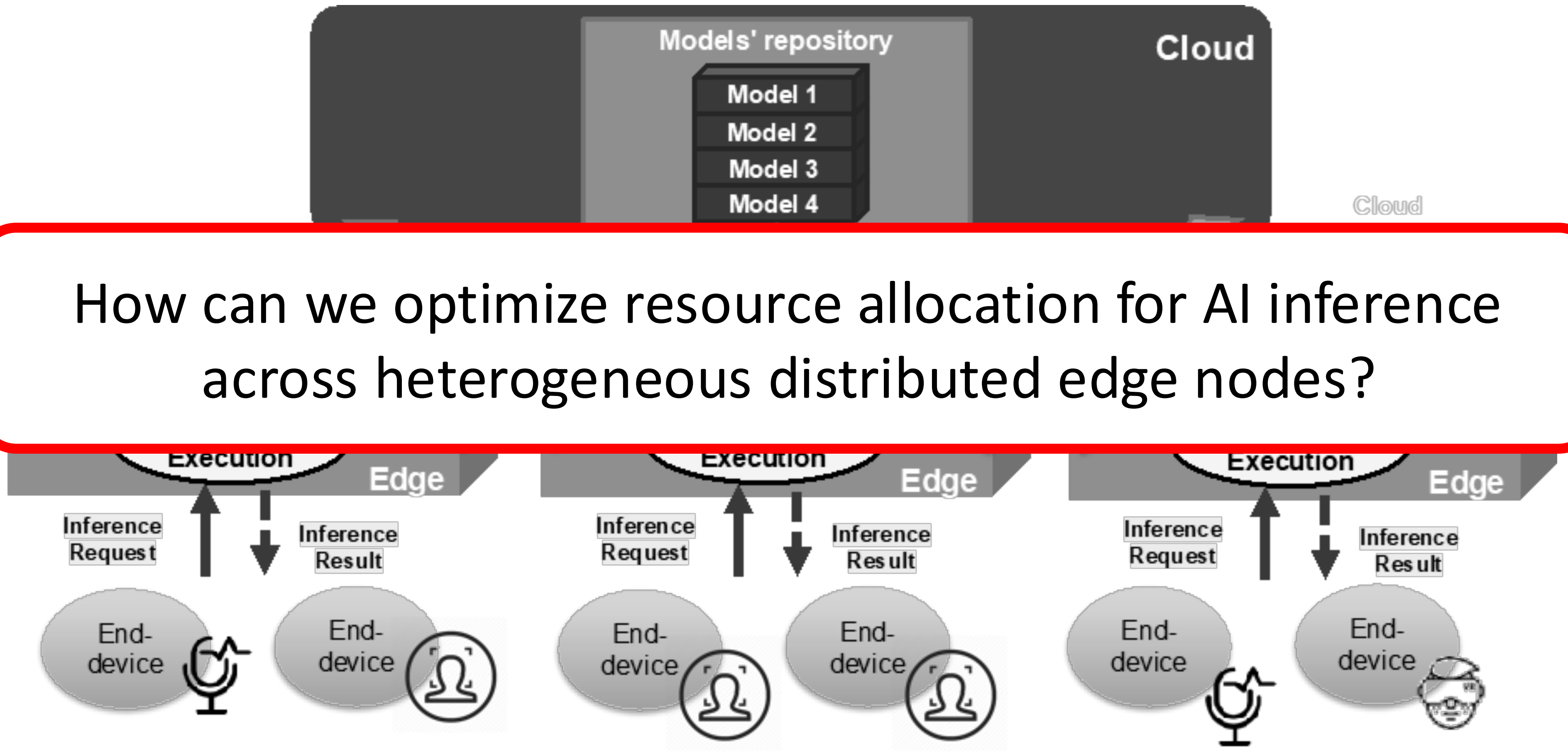
- Not many scenarios where **multiple edge nodes** are involved
- **Hardware heterogeneity** and **networking aspects** are often not considered



AI Inference In Distributed Edge Computing Systems



AI Inference In Distributed Edge Computing Systems



How can we optimize resource allocation for AI inference across heterogeneous distributed edge nodes?

Challenges And Requirements: Latency

Latency Components

- Composed of communication latency (data exchange) and computing latency (model training/inference execution).

Latency Significance

- Crucial for Inference: Requires **near real-time execution for prompt responses**.

Examples:

- Voice assistants need predictions within **200ms**.
- Tactile Internet and autonomous driving operations demand below **10ms** latency.

Source: Campolo, C., Iera, A. and Molinaro, A., 2023. Network for Distributed Intelligence: a Survey and Future Perspectives. IEEE Access.

Challenges And Requirements: Reliability

Reliability Components

- The ability of the network to consistently perform its intended function accurately and dependably.
- Key Aspects: Includes error rates, **uptime**, and **fault tolerance**.

Reliability Significance

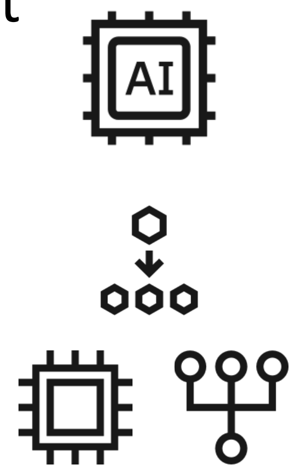
- Crucial for Consistent AI Performance: Ensures that **AI systems can function correctly** and deliver accurate results over time, **regardless of network conditions**.
- Impact on AI Applications: **High reliability** is essential for **mission-critical AI applications where errors or downtime** can have severe consequences.

Source: Campolo, C., Iera, A. and Molinaro, A., 2023. Network for Distributed Intelligence: a Survey and Future Perspectives. IEEE Access.

Challenges And Requirements: Practical Development

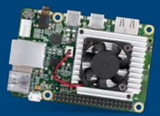
Realistic and more complex Edge AI/IoT deployment scenarios demand additional **requirements** to be fulfilled.

- ***Plug and Play* interoperability** among edge devices embedding different AI accelerators (e.g., **GPU, VPU, TPU**).
- **Agnostic AI inference services discovery** and **provisioning**.
- Combined **computing-** and **networking- aware orchestration mechanisms**, suitable for satisfying the Quality of Service (**QoS**) requirements
  of AI-enabled applications



EURECOM
Sophia Antipolis

(Resource Constrained) Heterogeneous Edge AI Nodes

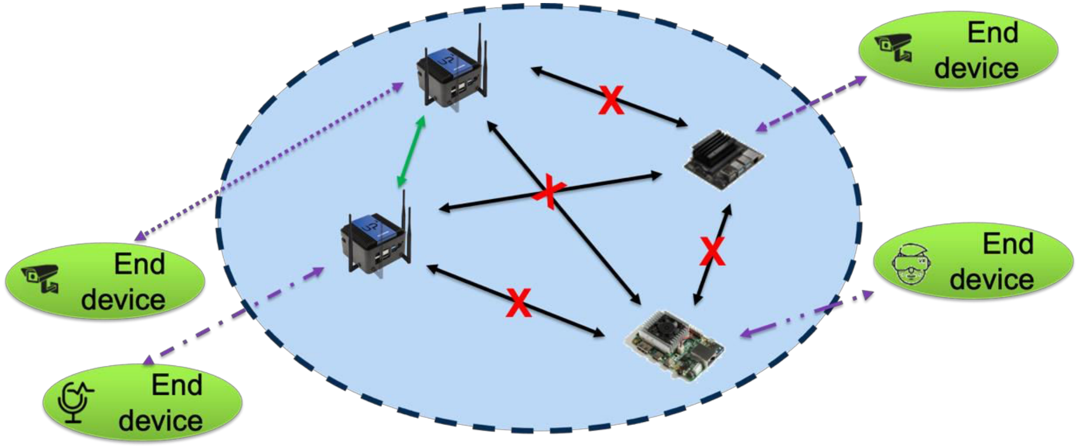
Feature	Coral Dev Board 	Jetson Nano 	Up Squared AI Edge X 
CPU Chipset	NXP i.MX 8M SoC (quad Cortex-A53, Cortex-M4F)	Quad-core ARM Cortex-A57 MPCore processor	Intel® Apollo Lake SoC ATOM x7-E3950
AI Accelerator Chipset	Google Edge TPU coprocessor: 4 TOPS (int8)	GPU NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores	Movidius Myriad X VPU integrated
Memory RAM	1GB LPDDR4	4GB LPDDR4	8GB LPDDR4
Storage	8GB eMMC MicroSD card slot	MicroSD card slot	64GB eMMC
Connectivity	Ethernet (1 x GbLAN) WiFi Bluetooth	Ethernet (1 x GbLAN) WiFi	WiFi 802.11 AC 2T2R + Bluetooth 4.2 (BLE)+ *LTE + Gigabit Ethernet

Hardware And Software Heterogeneity Major Implications

Impossible full *'out of the box'* devices' interoperability

AI Inference Latency may vary from board to board

Lack of seamless AI Inference provisioning offloading



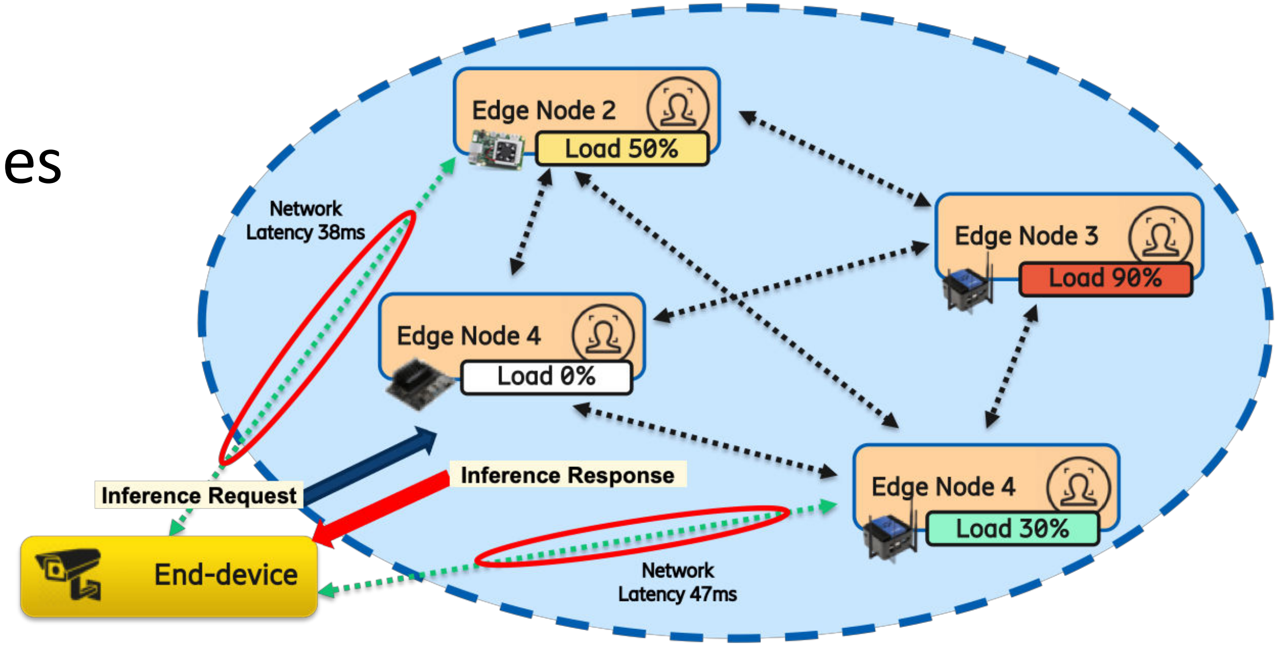
AI Inference Provisioning In Distributed Edge Systems

Edge Nodes Resources

- Computation Capabilities

Network Performance

- Latency
- Bandwidth



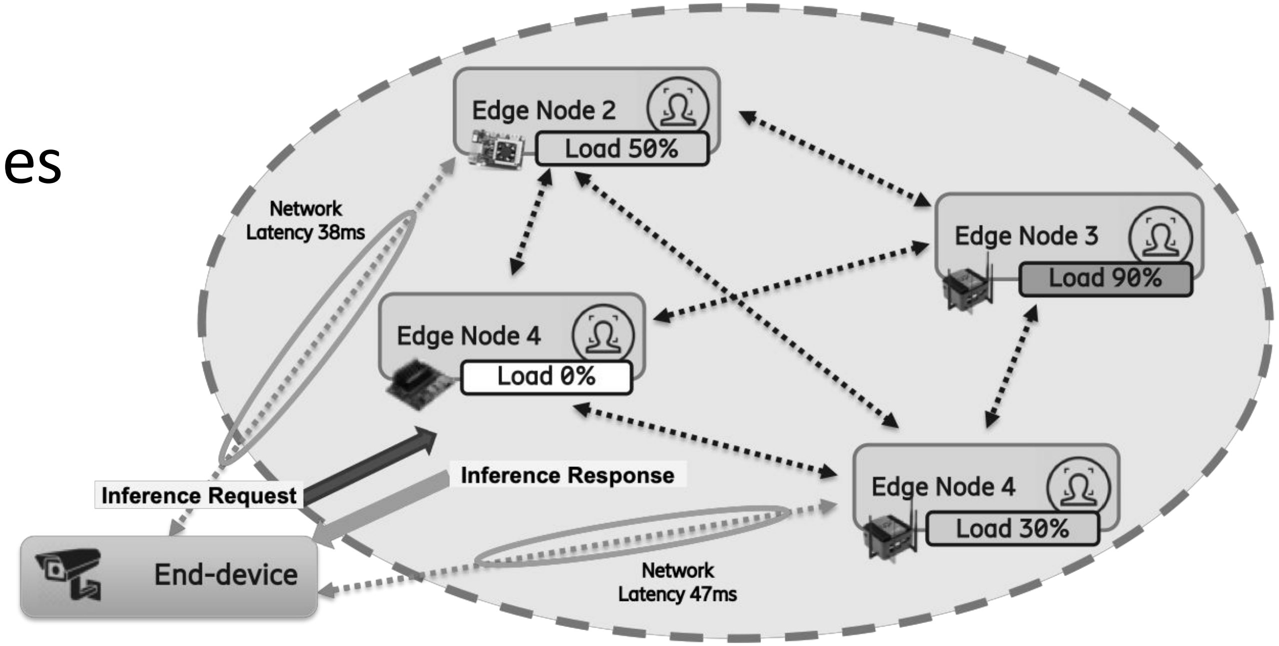
AI Inference Provisioning In Distributed Edge Systems

Edge Nodes Resources

- Computation Capabilities

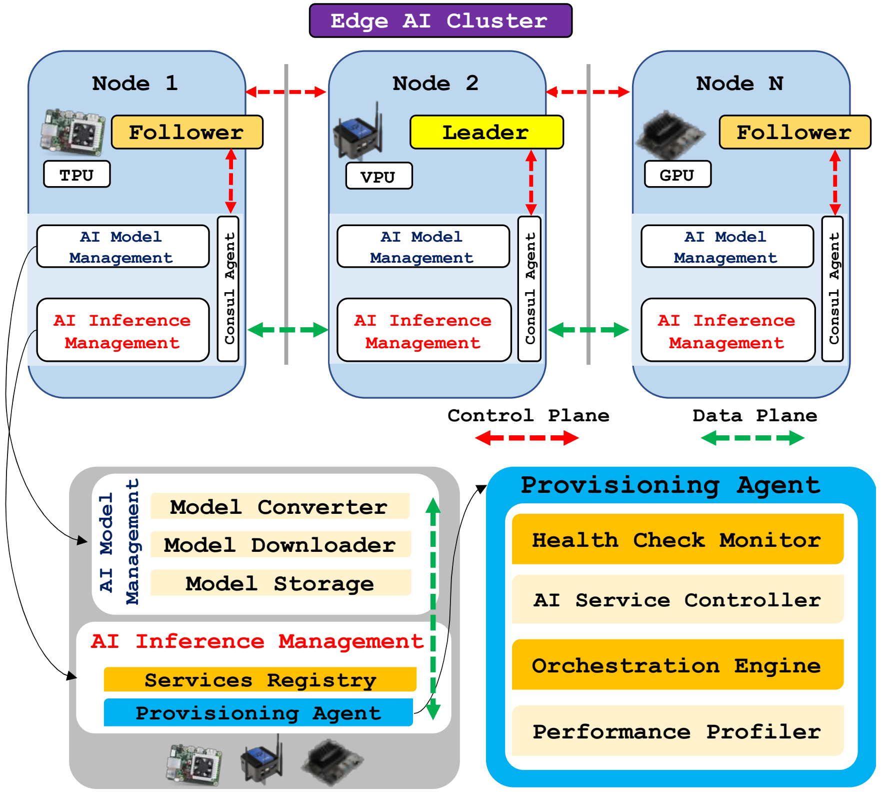
Network Performance

- Latency
- Bandwidth



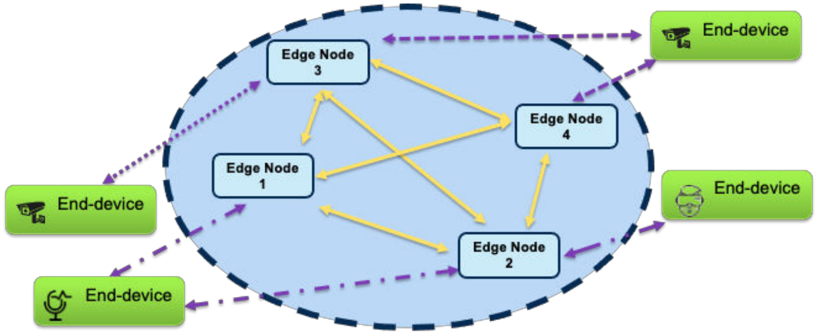
Which Edge AI node can best provide a specific AI inference service while meeting the requesting device's QoS requirements?

AI Inference Provisioning In Distributed Edge Systems



It ensure an abstraction layer that:

- Enables **interoperability** between different AI-enabled devices
- Allows **platform-agnostic service discovery** and provisioning of AI inference services
- Supports **seamless service orchestration** and execution migration capabilities



EURECOM
Sophia Antipolis

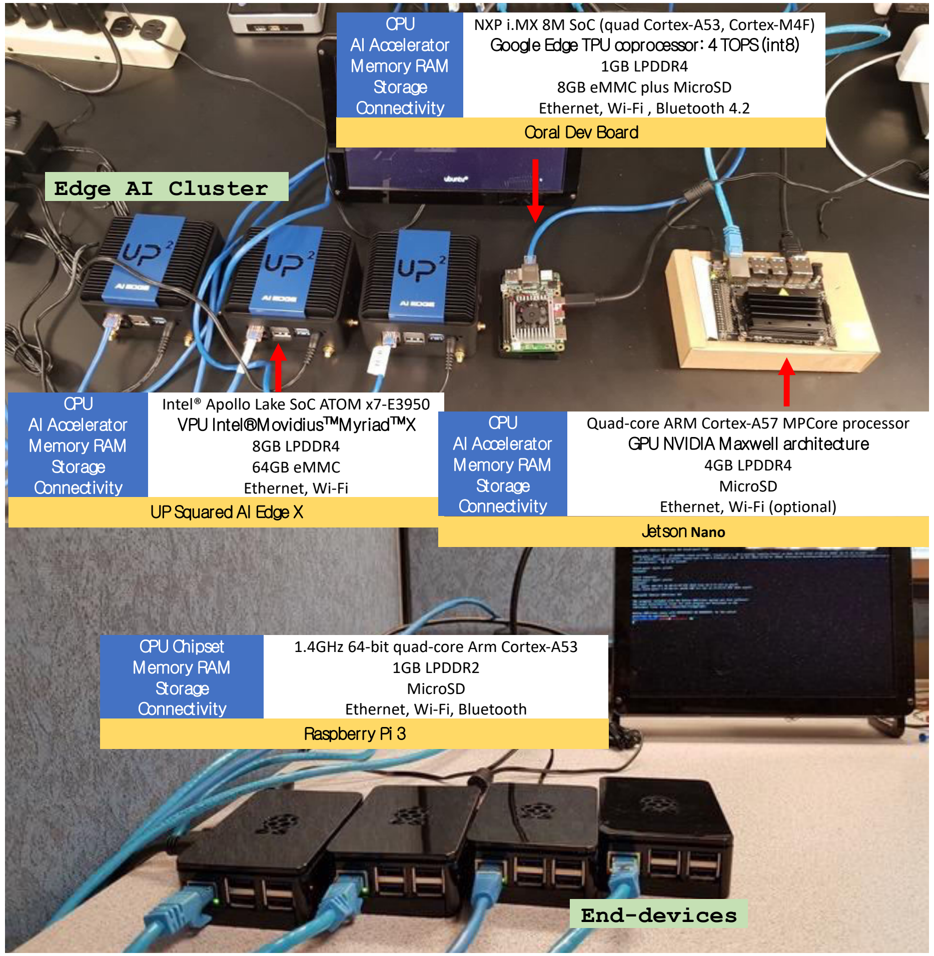
Edge AI Testbed

Edge AI Cluster:
- Intel Movidius Myriad X VPU (UP Squared AI Edge X)
- Google Edge TPU (Coral Dev Board)
- NVIDIA 128-core Maxwell GPU (Jetson Nano)

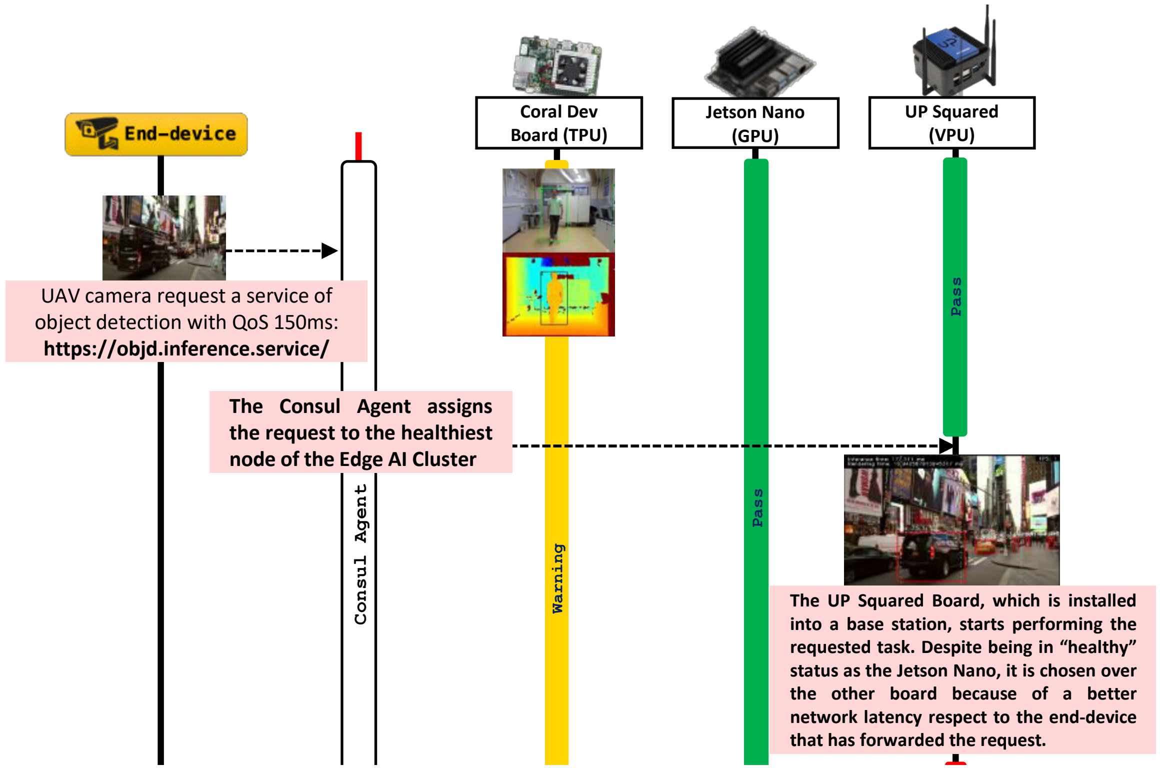
End-Devices:
- Raspberry Pi 3 Model B (x4)

Network Setup:
- Controlled wireless network for device communication
- Emulation of realistic edge system deployment

Latency Measurement & Emulation:
- Analysis based on 5G to edge server latency
- Best-fit distribution: Stable (Shape: 1.6878, Scale: 0.0980)
- Average network latency: 13.405 ms
- Standard deviation: 16.065 ms, showing high variability



AI Inference Provisioning: Edge AI Node Selection



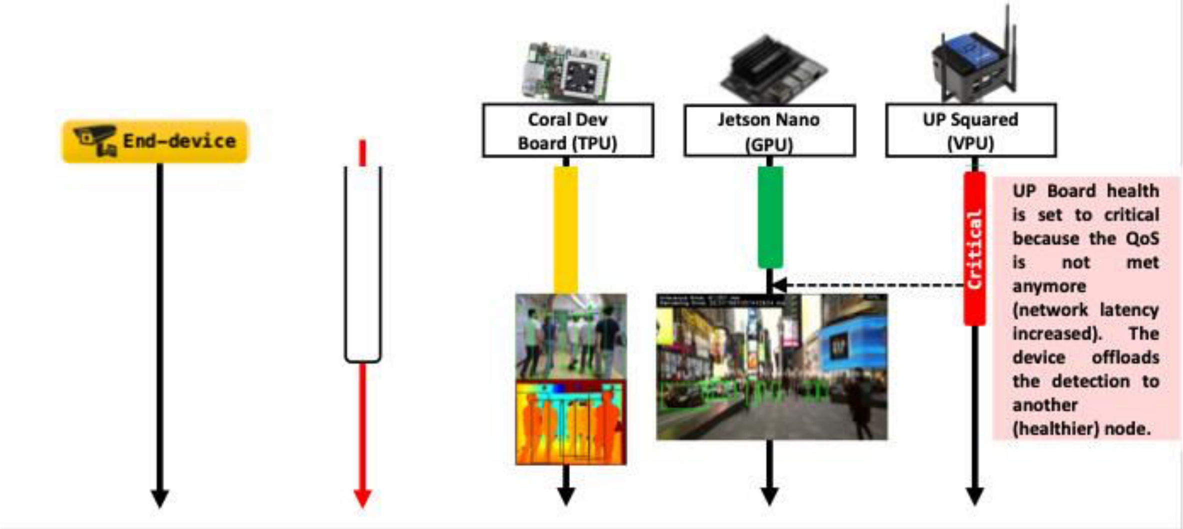
Algorithm 1 Select node for AI Inference provisioning and AI Inference execution offload

Require: Set of edge nodes (*edgeNodes*), end-device (*endDevice*), NLM (*NetworkLatencyMatrix*), QoS requirements

Ensure: Selected healthy edge node (*selectedNode*)

```
Preload AI models in edgeNodes
Initialize selectedNode to null
function ASSIGNNODE(edgeNodes, endDevice, latencyMatrix)
    Initialize minLatency to max(NetworkLatencyMatrix)
    Initialize healthyNode to null
    for each edgeNode in edgeNodes do
        if edgeNode is healthy and the latency between edgeNode and endDevice is less than minLatency then
            Update minLatency with the latency between edgeNode and endDevice
            Update healthyNode with the current edgeNode
        end if
    end for
    Assign the AI inference task to edgeNode
end function
selectedNode ← ASSIGNNODE(edgeNodes, endDevice, NetworkLatencyMatrix)
```

AI Inference Provisioning: Service Provisioning Offloading



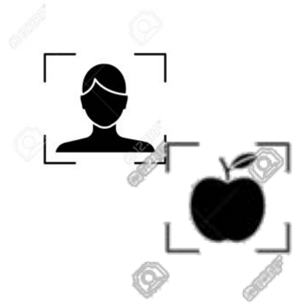
Algorithm 1 Select node for AI Inference provisioning and AI Inference execution offload

```
while true do
    Continuously monitor the health status and QoS requirements for selectedNode
    if selectedNode cannot guarantee required QoS for handled applications then
        Temporarily make selectedNode unreachable for new requests
        Find another healthy node as described in the ASSIGNNODE function
        if a new healthy node is found then
            Perform a seamless migration of the affected AI inference task to the new healthy node
        end if
    end if
end while
return selectedNode
```

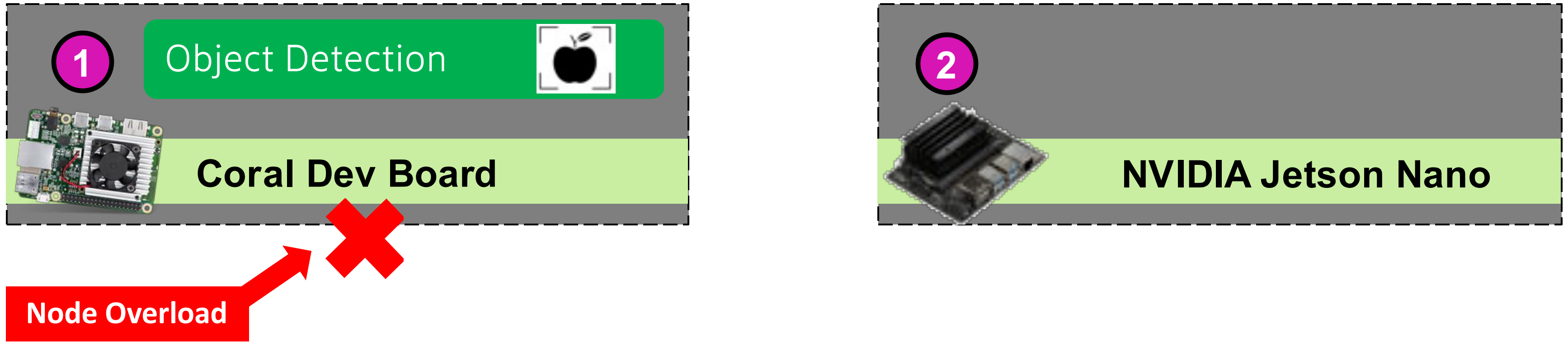
Additional details about the Health Checks definition can be found in the references

AI Inference Service Execution Orchestration

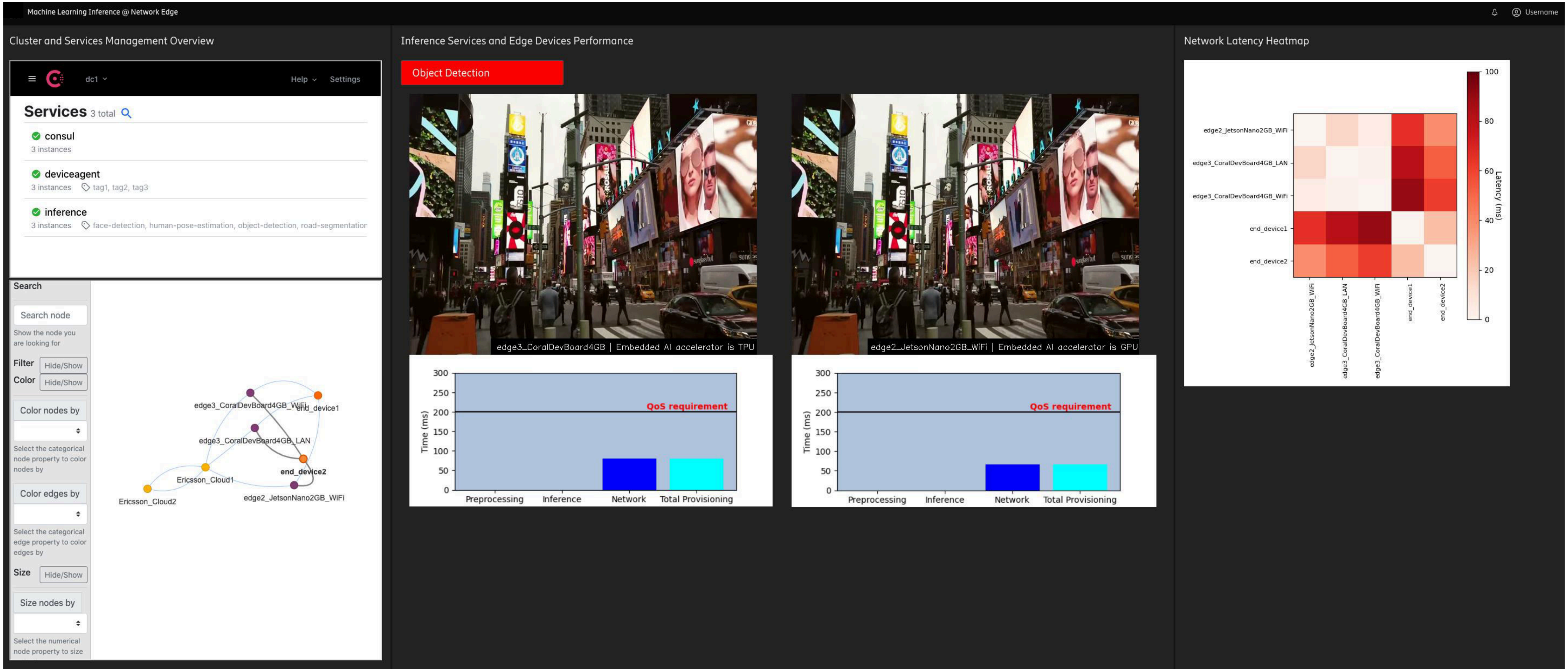
Object Detection over real-time video streaming



- The application uses an object detection model (MobileNet-SSDv1).
- The ML inference execution is migrated from an edge node to another one as soon as the device gets too overloaded, or network latency increases.

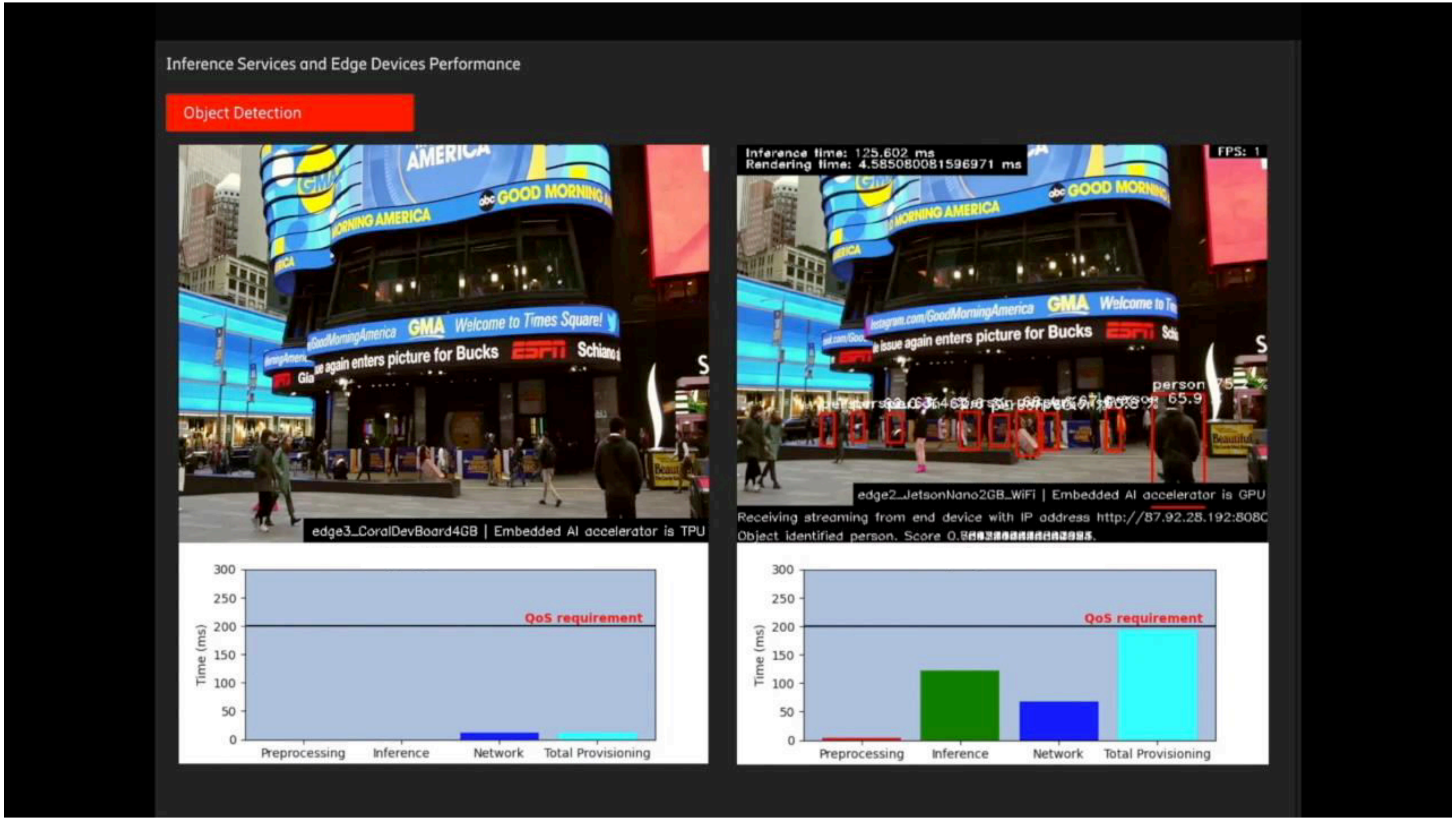


Demo

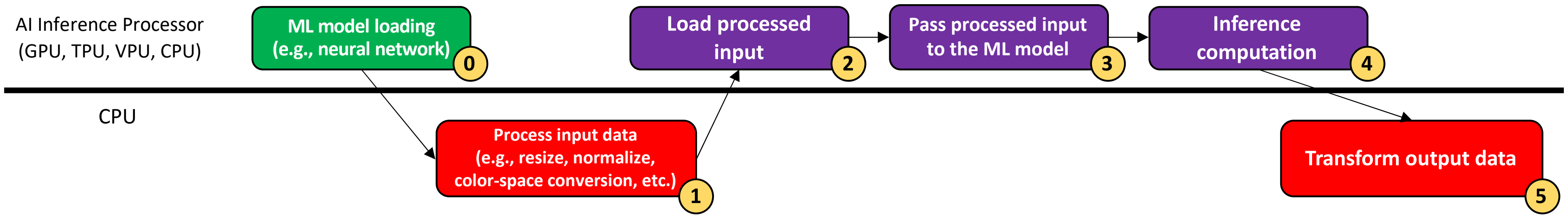


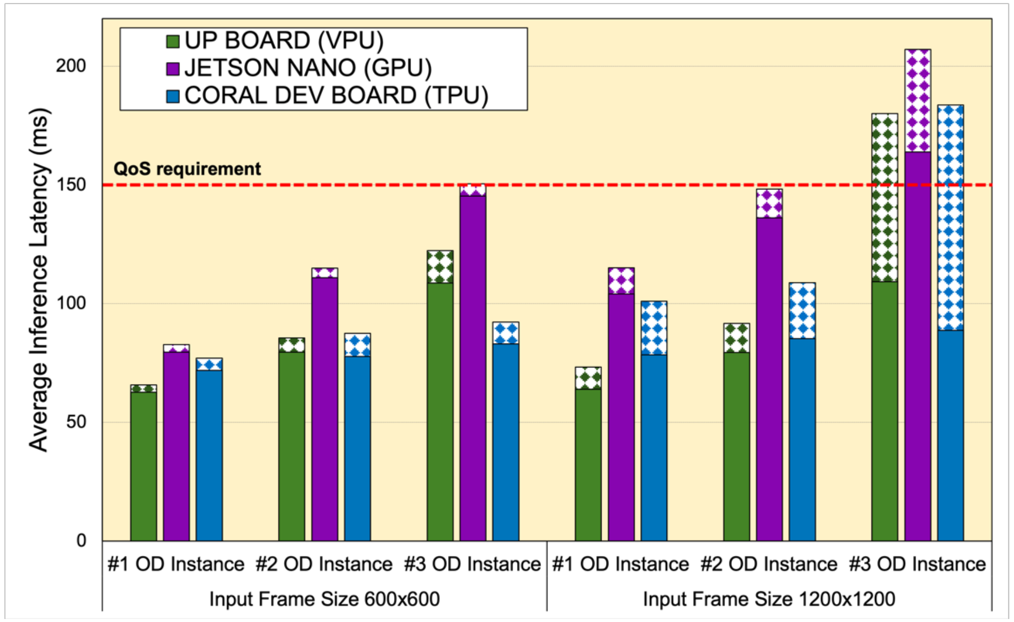
Machine Learning Inference @ Network Edge
Username
Cluster and Services Management Overview
Inference Services and Edge Devices Performance
Network Latency Heatmap
dc1
Help
Settings
Services 3 total
consul
3 instances
deviceagent
3 instances
tag1, tag2, tag3
inference
3 instances
face-detection, human-pose-estimation, object-detection, road-segmentation
Search
Search node
Show the node you are looking for
Filter
Hide/Show
Color
Hide/Show
Color nodes by
Select the categorical node property to color nodes by
Color edges by
Select the categorical edge property to color edges by
Size
Hide/Show
Size nodes by
Select the numerical node property to size
edge3_CoralDevBoard4GB_WiFi
end_device1
edge3_CoralDevBoard4GB_LAN
end_device2
Ericsson_Cloud1
Ericsson_Cloud2
edge2_JetsonNano2GB_WiFi
Object Detection
edge3_CoralDevBoard4GB | Embedded AI accelerator is TPU
edge2_JetsonNano2GB_WiFi | Embedded AI accelerator is GPU
QoS requirement
Time (ms)
Preprocessing
Inference
Network
Total Provisioning
Latency (ms)
edge2_JetsonNano2GB_WiFi
edge3_CoralDevBoard4GB_LAN
edge3_CoralDevBoard4GB_WiFi
end_device1
end_device2

Inference Execution Migration



Hardware And Software Heterogeneity Impact On The AI Inference Performance



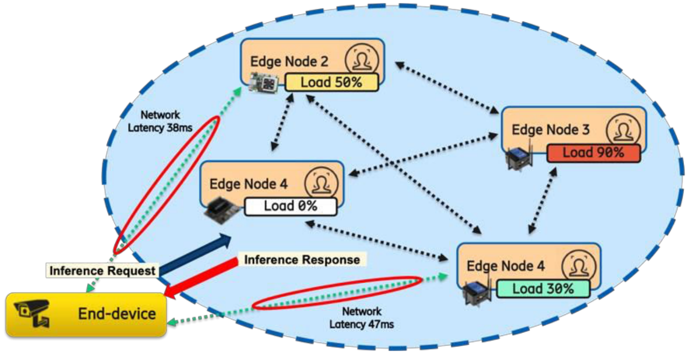


Insights:

- CPU more affected by frame size increase than AI Accelerator
- Overall system latency inflates by 45-60% with larger frame size
- **AI application parameters significantly affect inference latency, with different devices showing varied responses to the same workload**
- **Device responsiveness to computational demands varies**

Need For Newer Allocation Strategies

Balancing Load and Network Latency in AI-Enabled Heterogeneous Edge Networks: Proposes allocation based on device-specific CPU and AI accelerator performance, with a Performance Profiler assigning weights to reflect each node's capabilities for task distribution.

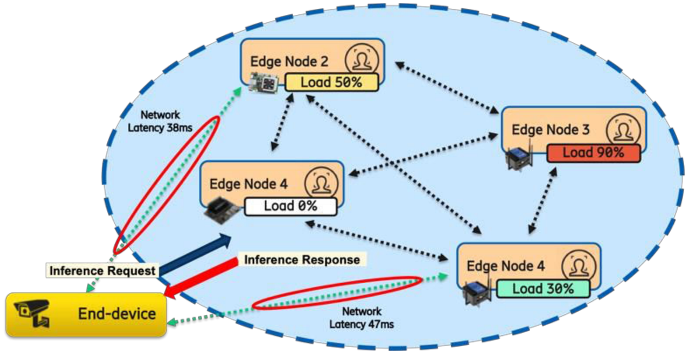


$$W_{\text{combined}} = \alpha \times W_{\text{cpu}} + \beta \times W_{\text{ai}} + \gamma \times W_{\text{nl}}$$

Dynamic Adaptation to Workload: Framework allocates tasks by assessing workload features and dynamically updates node weights to optimize task execution in response to changing system demands and resource availability.

Need For Newer Allocation Strategies

Balancing Load and Network Latency in AI-Enabled Heterogeneous Edge Networks: Proposes allocation based on device-specific CPU and AI accelerator performance, with a Performance Profiler assigning weights to reflect each node's capabilities for task distribution.



$$W_{\text{combined}} = \alpha \times W_{\text{cpu}} + \beta \times W_{\text{ai}} + \gamma \times W_{\text{nl}}$$

Dynamic Adaptation to Workload: Framework allocates tasks by assessing workload features and dynamically updates node weights to optimize task execution in response to changing system demands and resource availability.

FL angle? What if α, β, γ were learned in a federated way across nodes to reflect evolving conditions, while preserving privacy?

References

Sources:

- Morabito R., and Chiang M., 2021, July. "Discover, Provision, and Orchestration of Machine Learning Inference Services in Heterogeneous Edge: A Demonstration". In 2021 41st IEEE International Conference on Distributed Computing Systems (ICDCS 2021). IEEE. (**Best Demo Award**)
- Morabito R., Tatipamula M., Tarkoma S., and Chiang M., 2023. "Edge AI Inference in Heterogeneous Constrained Computing: Feasibility and Opportunities". In 2023 IEEE International Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks (IEEE CAMAD).
- Morabito, R. and Chiang, M., 2024. Exploring Edge AI Inference in Heterogeneous Environments: Requirements, Challenges, and Solutions. In IoT Edge Intelligence (pp. 37-66). Cham: Springer Nature Switzerland.

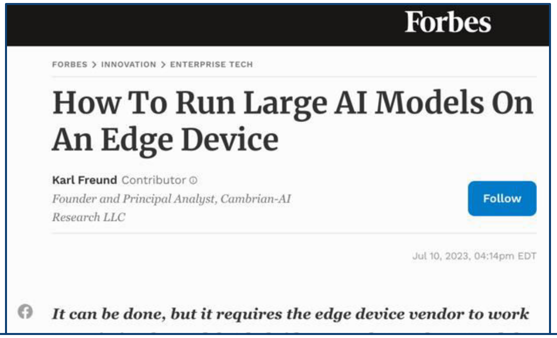
From the Edge to the Cloud: Exploring AI Inference Across the Computing Continuum

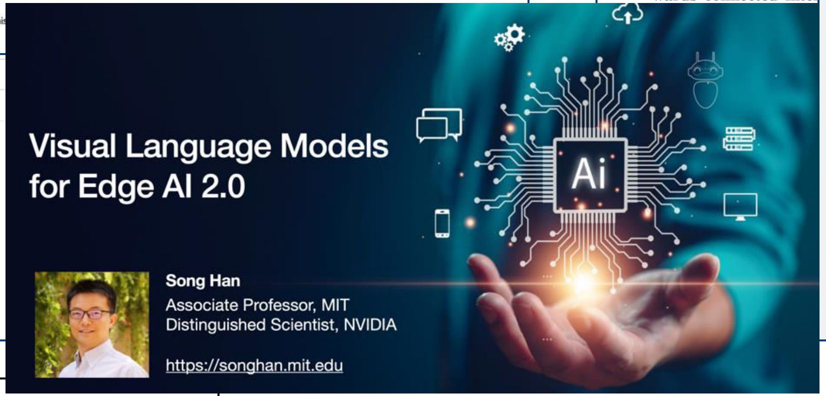
(yes, including Generative AI)

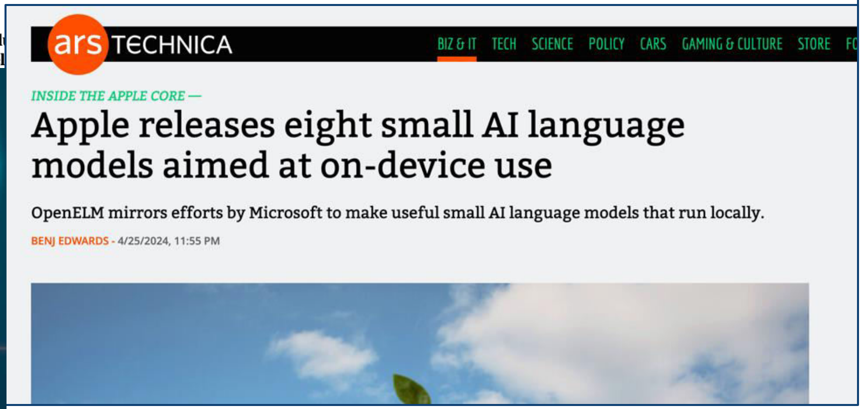
The Generative AI Wave: Is There Any Opportunity for the Edge?

The Generative AI Wave

A Great Opportunity for the Edge



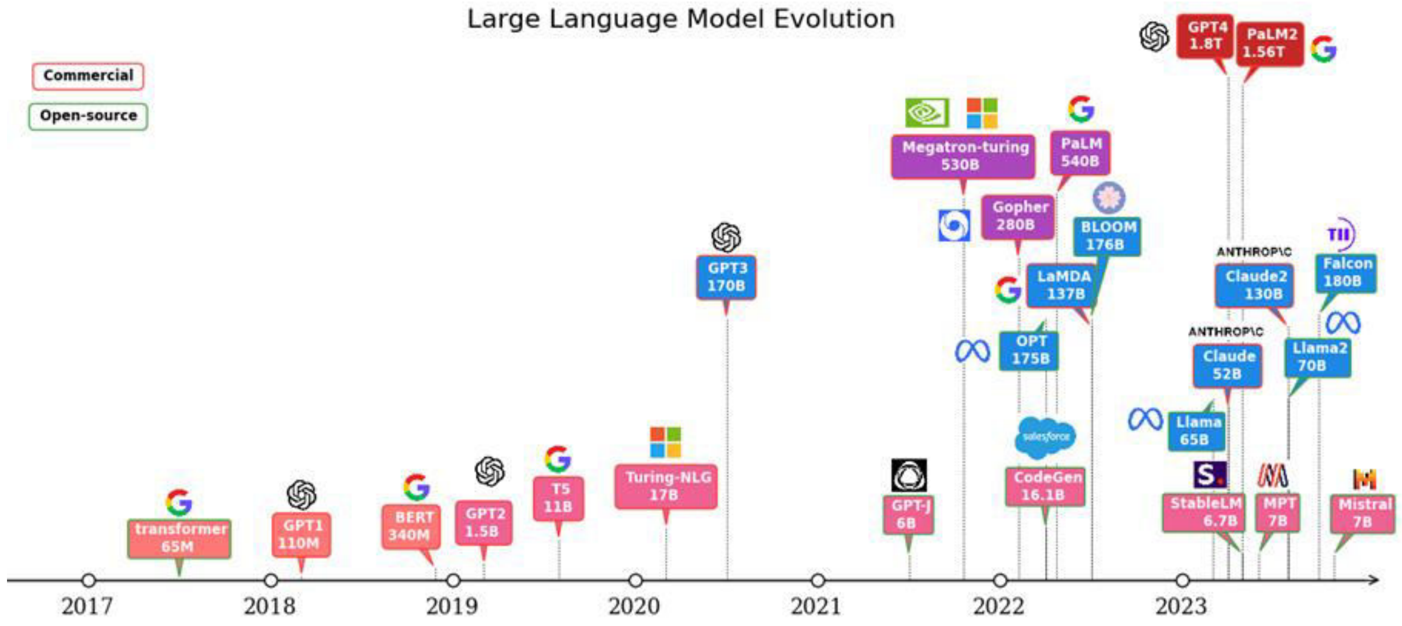




Sources:
https://www.forbes.com/sites/karlfreund/2023/07/10/how-to-run-large-ai-models-on-an-edge-device/
https://arxiv.org/pdf/2307.02779
https://thenewstack.io/the-rise-of-small-language-models/
https://arstechnica.com/information-technology/2024/04/apple-releases-eight-small-ai-language-models-aimed-at-on-device-use/
https://www.youtube.com/watch?v=1DeQq1ZuG9o&t=155s&ab_channel=tinyMLFoundation
https://ieeexplore.ieee.org/document/10384606/

The Generative AI Wave

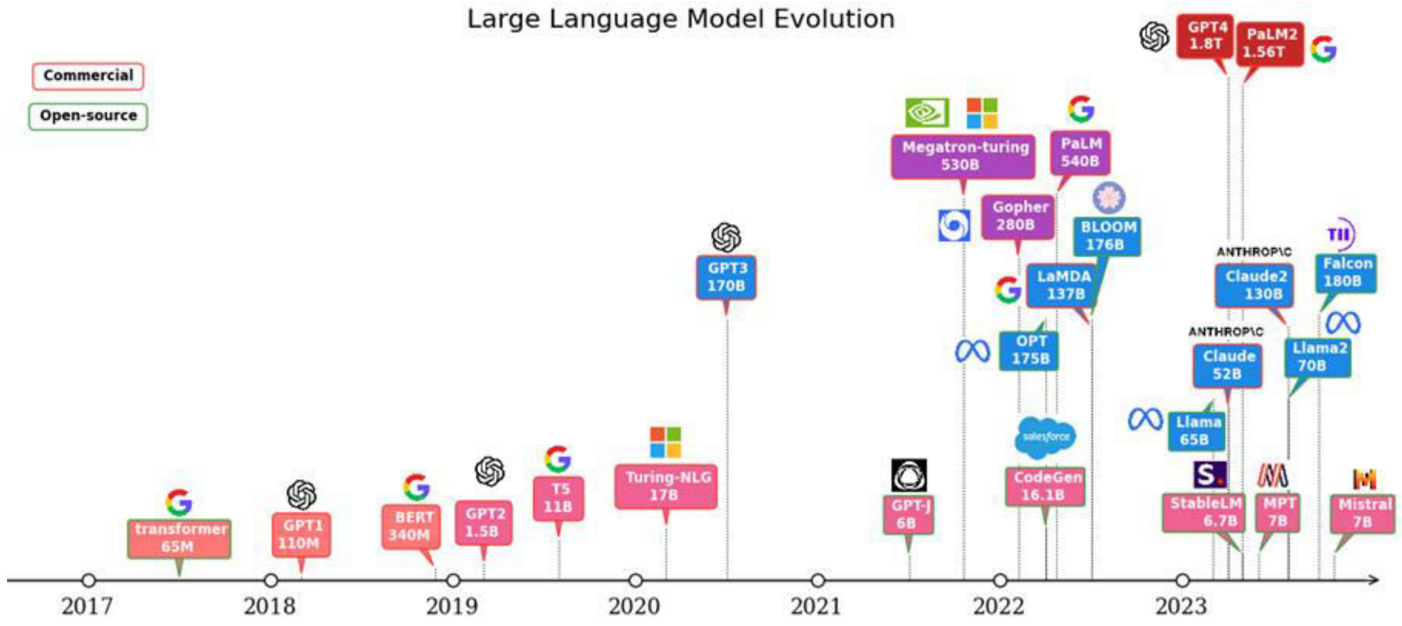
A Great Opportunity for the Edge



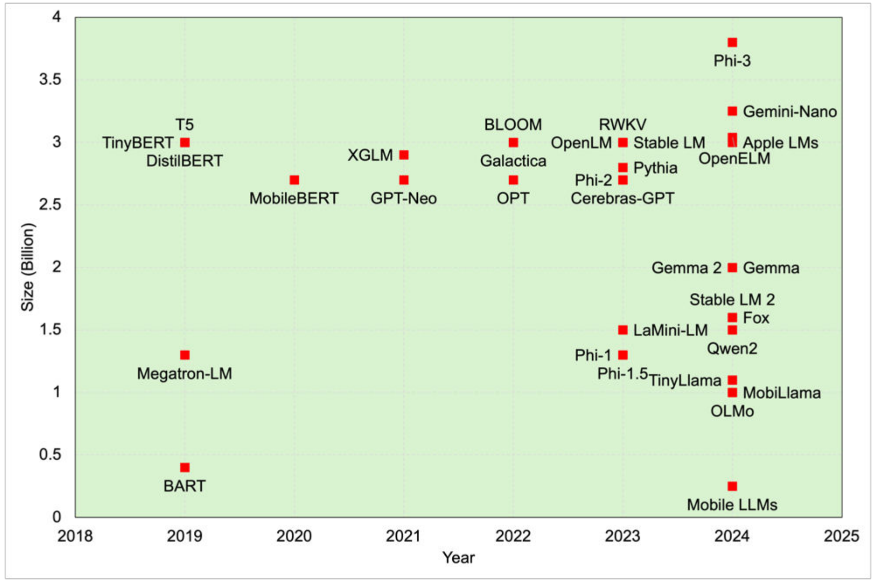
Source:
https://infohub.delltechnologies.com/en-us/p/investigating-the-memory-access-bottlenecks-of-running-llms/

The Generative AI Wave

A Great Opportunity for the Edge



Small Language Models (SLMs) Explosion

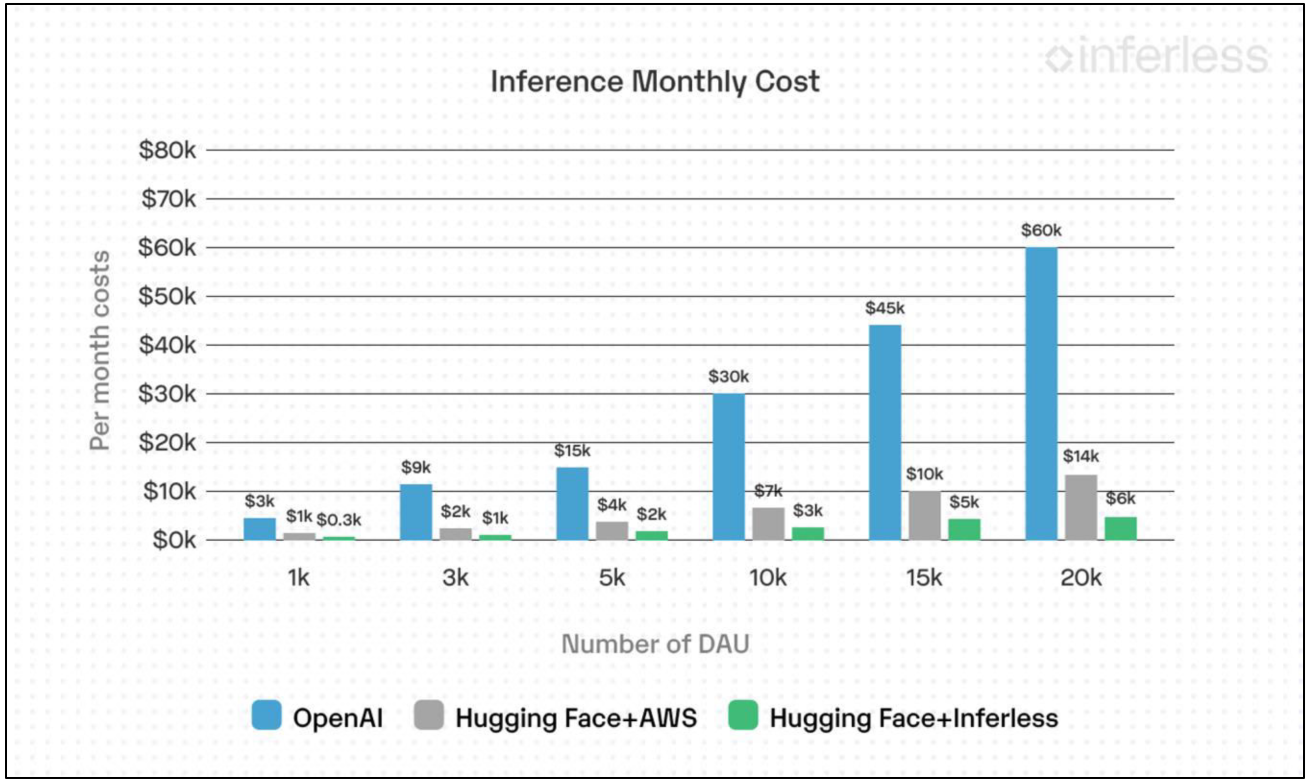


Source:
https://infohub.delltechnologies.com/en-us/p/investigating-the-memory-access-bottlenecks-of-running-llms/

Large Language Models Inference Cost

Costs of inference as an application scales from 1k daily active users (DAUs) to 20k DAUs

- Each user sends an average of 15 requests per day

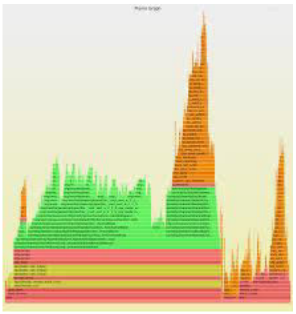


(Source: https://www.inferless.com/learn/unraveling-gpu-inference-costs-for-llms-openai-aws-and-inferless)

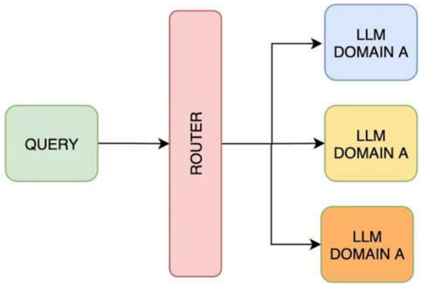
EURECOM
Sophia Antipolis

Current Activities In This Area

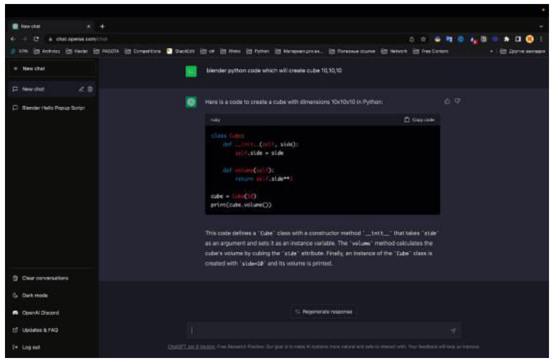
Benchmarking SLMs in Constrained Devices



SLM/LLM Query Routing via Edge Collaboration



Streamlining TinyML Lifecycle with Large Language Models

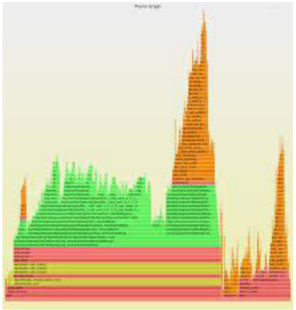


Current Activities In This Area

Benchmarking SLMs in Constrained Devices

SLM/LLM Query Routing via Edge Collaboration

Streamlining TinyML Lifecycle with Large Language Models



EURECOM
Sophia Antipolis

Benchmarking SLMs In Constrained Devices

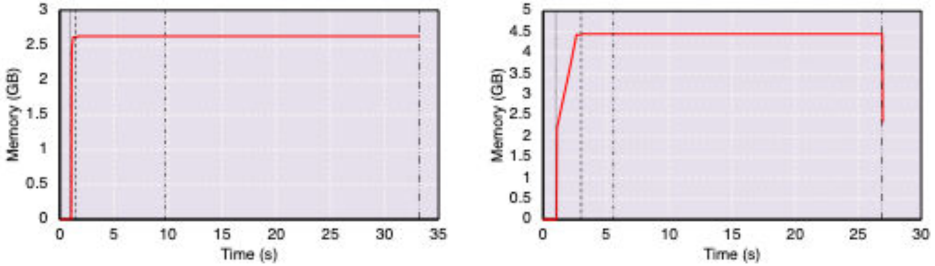
Small Language Models (SLMs) for resource constrained devices:

- Recent work shows the possibility to adopt LLMs at the constrained edge
- Still some way to go regarding the resource consumption when executed on MCU devices
- Potential to run on more capable but still lightweight edge devices (e.g., Single-Board Computers)

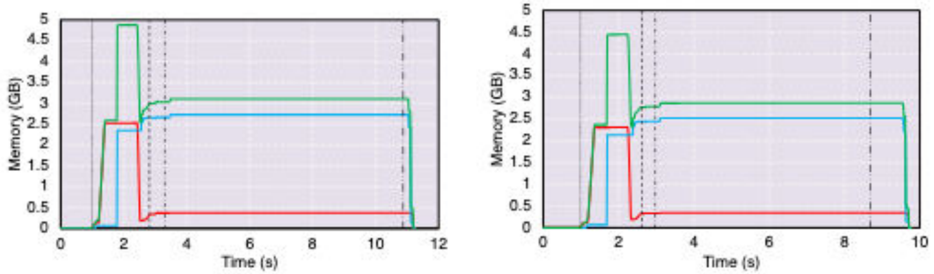
Source: https://github.com/maxbbraun/llama4micro/

Benchmarking SLMs In Constrained Devices

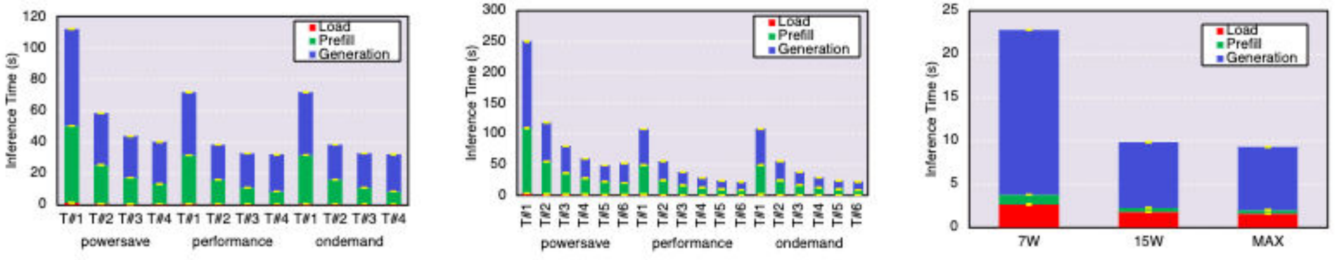
Current Focus: developing a benchmark suite for evaluating the capabilities of LLMs on edge to **constrained edge devices**.



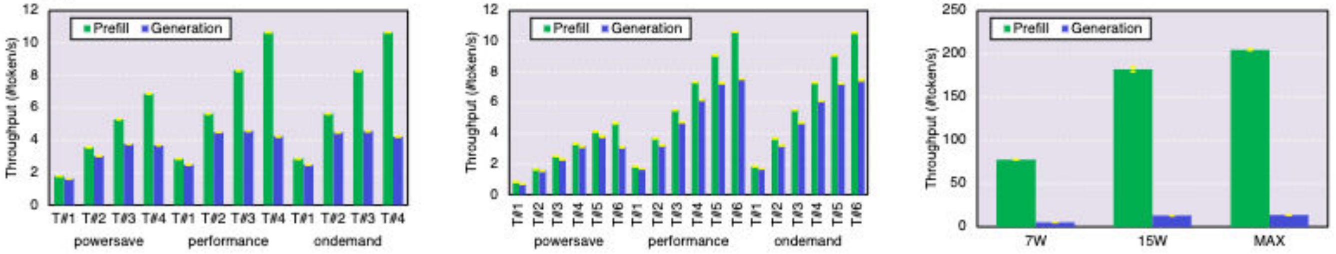
(a) RPi 5, Q4_K_M (b) RPi 5, Q4_0



(c) Orin GPU, Q4_K_M (d) Orin GPU, Q4_0



(a) Latency RPi 5, *Q4_K_M* (b) Latency Orin CPU, *Q4_K_M* (c) Latency Orin GPU, *Q4_K_M*



(d) Throughput RPi 5, *Q4_K_M* (e) Throughput Orin CPU, *Q4_K_M* (f) Throughput Orin GPU, *Q4_K_M*

EURECOM
Sophia Antipolis

Read the Paper about this Benchmarking Work!

Sometimes Painful but Certainly Promising: Feasibility and Trade-offs of Language Model Inference at the Edge

MAXIMILIAN ABSTREITER, University of Helsinki, Finland
SASU TARKOMA, University of Helsinki, Finland
ROBERTO MORABITO, EURECOM, France and University of Helsinki, Finland

The rapid rise of Language Models (LMs) has expanded the capabilities of natural language processing, powering applications from text generation to complex decision-making. While state-of-the-art LMs often boast hundreds of billions of parameters and are primarily deployed in data centers, recent trends show a growing focus on compact models—typically under 10 billion parameters—enabled by techniques such as quantization and other model compres-

https://arxiv.org/pdf/2503.09114

Read the Paper about this Benchmarking Work!

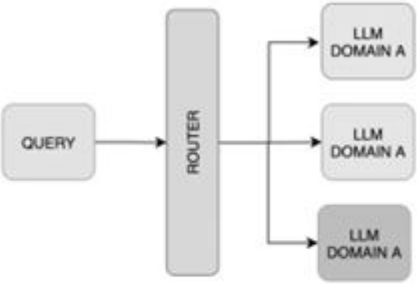
Sometimes Painful but Certainly Promising: Feasibility and Trade-offs of Language Model Inference at the Edge

- **(C1)** Addressing **(RQ1-3)**, we benchmark 11 generative LMs on two widely used SBCs, analyzing key performance indicators such as memory usage, inference speed, and energy efficiency to quantify the feasibility of edge inference.
- **(C2)** Answering **(RQ2)**, we evaluate the effect of quantization and model scaling on inference efficiency, resource utilization, and model performance, highlighting the trade-offs between accuracy and computational cost.
- **(C3)** Directly addressing **(RQ3)**, we compare CPU-based inference against GPU acceleration, investigating their impact on execution speed, energy consumption, and practical deployment feasibility on edge devices.
- **(C4)** Responding to **(RQ4)**, we analyze the impact of power modes, threading configurations, and system settings, while also evaluating micro-architectural metrics, such as cache misses and context switches, to uncover bottlenecks and efficiency gaps in edge-based LM execution.
- **(C5)** Informed by **(RQ5)**, we assess the practical challenges of running LMs at the edge, including inference cost analysis, qualitative benchmarking of the models, and aspects such as usability related to real-world applicability, providing a broader perspective beyond raw performance metrics.

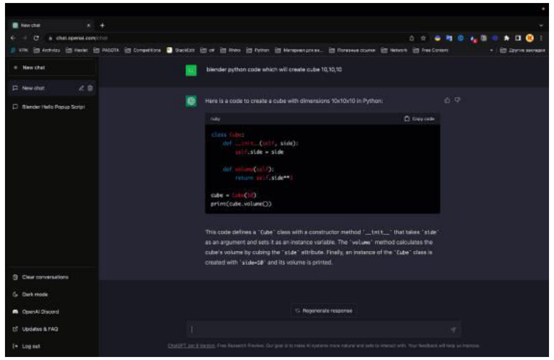
Current Activities In This Area

Benchmarking SLMs in Constrained Devices

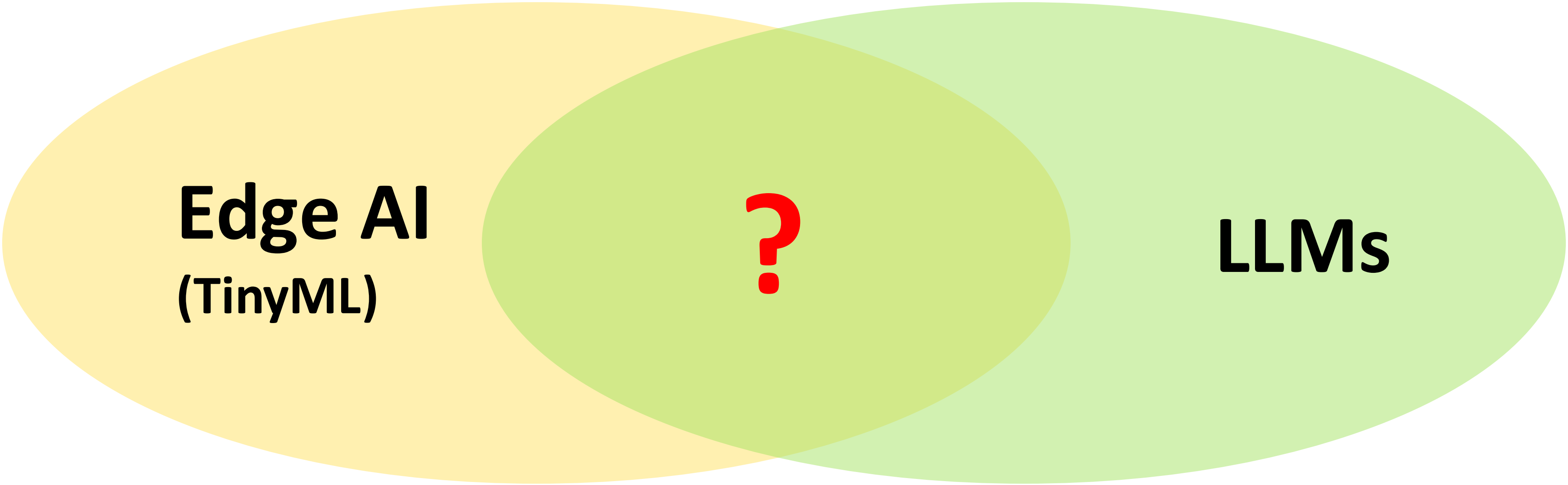
SLM/LLM Query Routing via Edge Collaboration



Streamlining TinyML Lifecycle with Large Language Models



LLMs for Edge AI: Any Possibility?

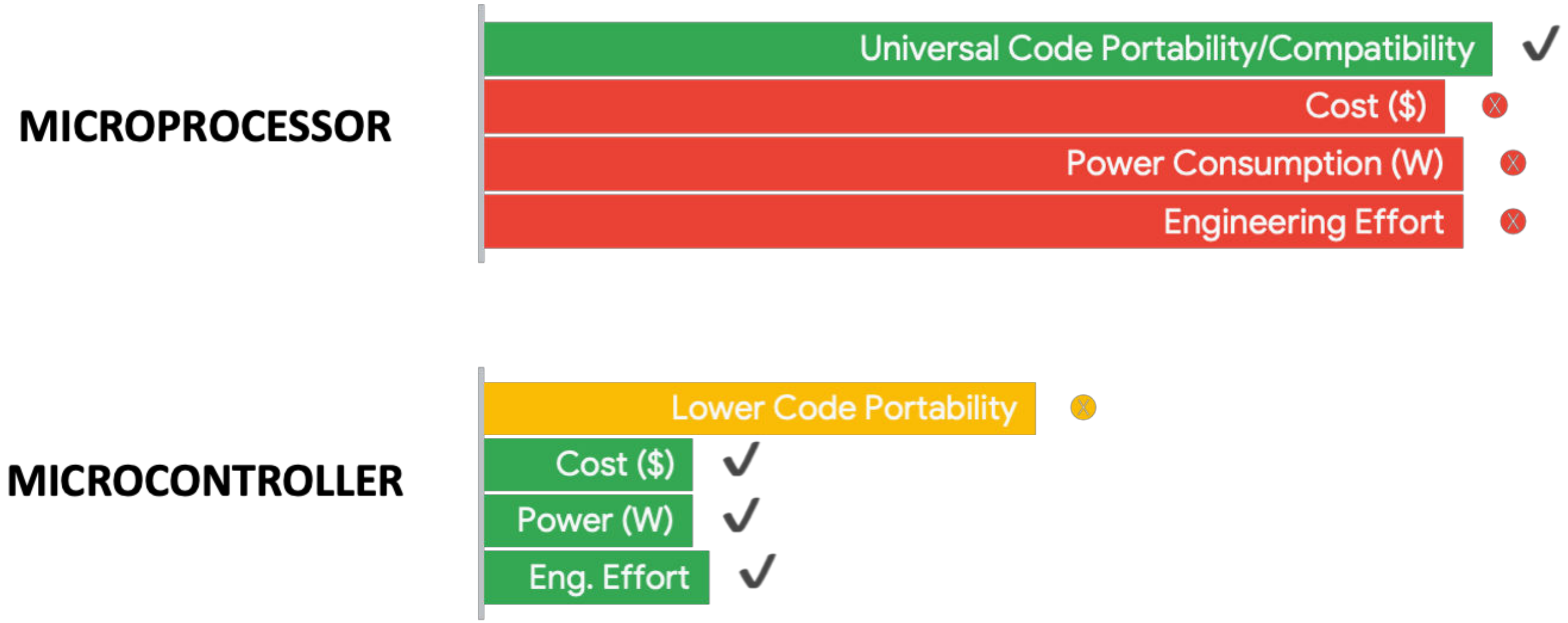


Hardware – Key Differences

	Micro**processor**	>	Micro**controller**
Platform			
Compute	1GHz–4GHz	**~10X**	1MHz–400MHz
Memory	512MB–64GB	**~10000X**	2KB–512KB
Storage	64GB–4TB	**~100000X**	32KB–2MB
Power	30W–100W	**~1000X**	150µW–23.5mW

Source: Content on these slides is sourced from https://github.com/edgeimpulse/courseware-embedded-machine-learning and https://github.com/tinyMLx/courseware/tree/master/edX

Portability Trade-offs



Source: Content on these slides is sourced from https://github.com/edgeimpulse/courseware-embedded-machine-learning and https://github.com/tinyMLx/courseware/tree/master/edX

ML Software

Software

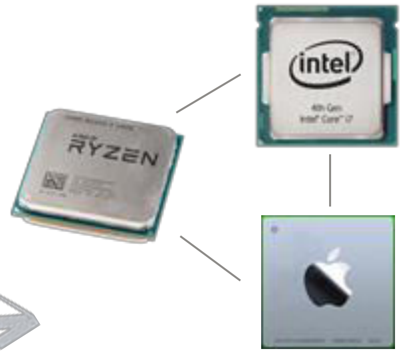
Applications

Libraries

Operating System

Hardware

```
import numpy as np

for x in range(10):
    np.SaveTheWorld()
```



Able to execute the same code on different microprocessor hardware and architectures.

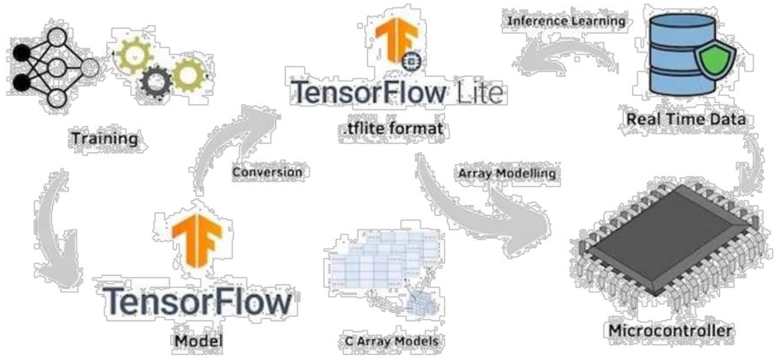
Source: Content on these slides is sourced from https://github.com/edgeimpulse/courseware-embedded-machine-learning and https://github.com/tinyMLx/courseware/tree/master/edX

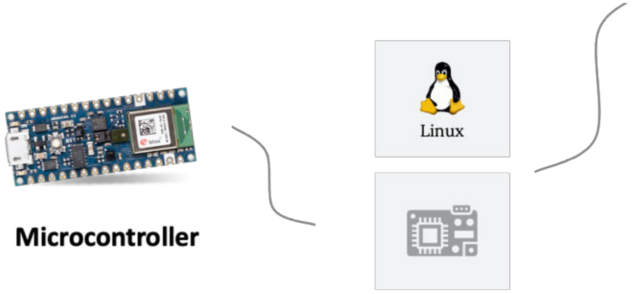
ML Software For Constrained Devices

Machine Learning Models

Machine Learning Runtimes

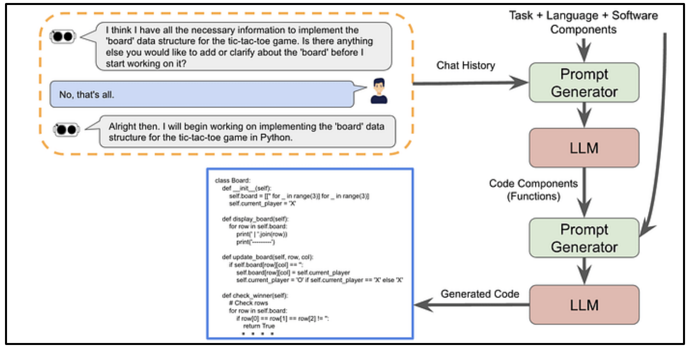
Machine Learning Hardware

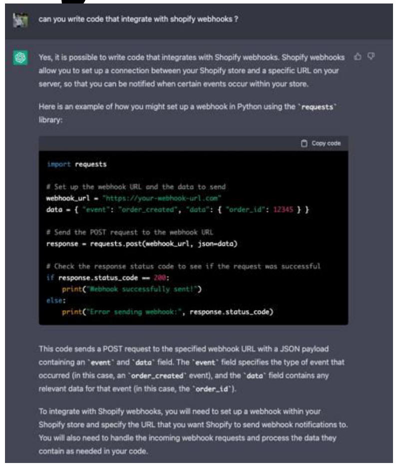




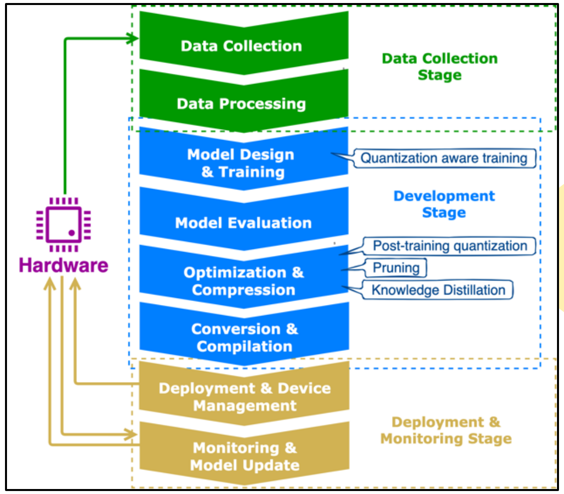
Source: Content on these slides is sourced from https://github.com/edgeimpulse/courseware-embedded-machine-learning and https://github.com/tinyMLx/courseware/tree/master/edX

LLMs for Edge AI: Any Possibility?

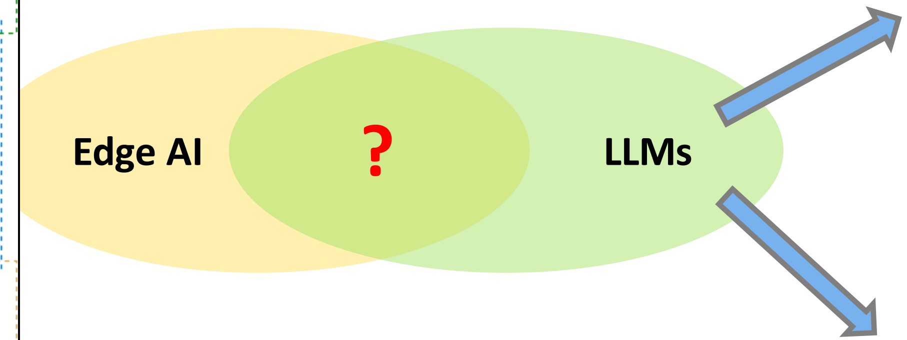




Code Generation Capabilities



Edge AI Lifecycle

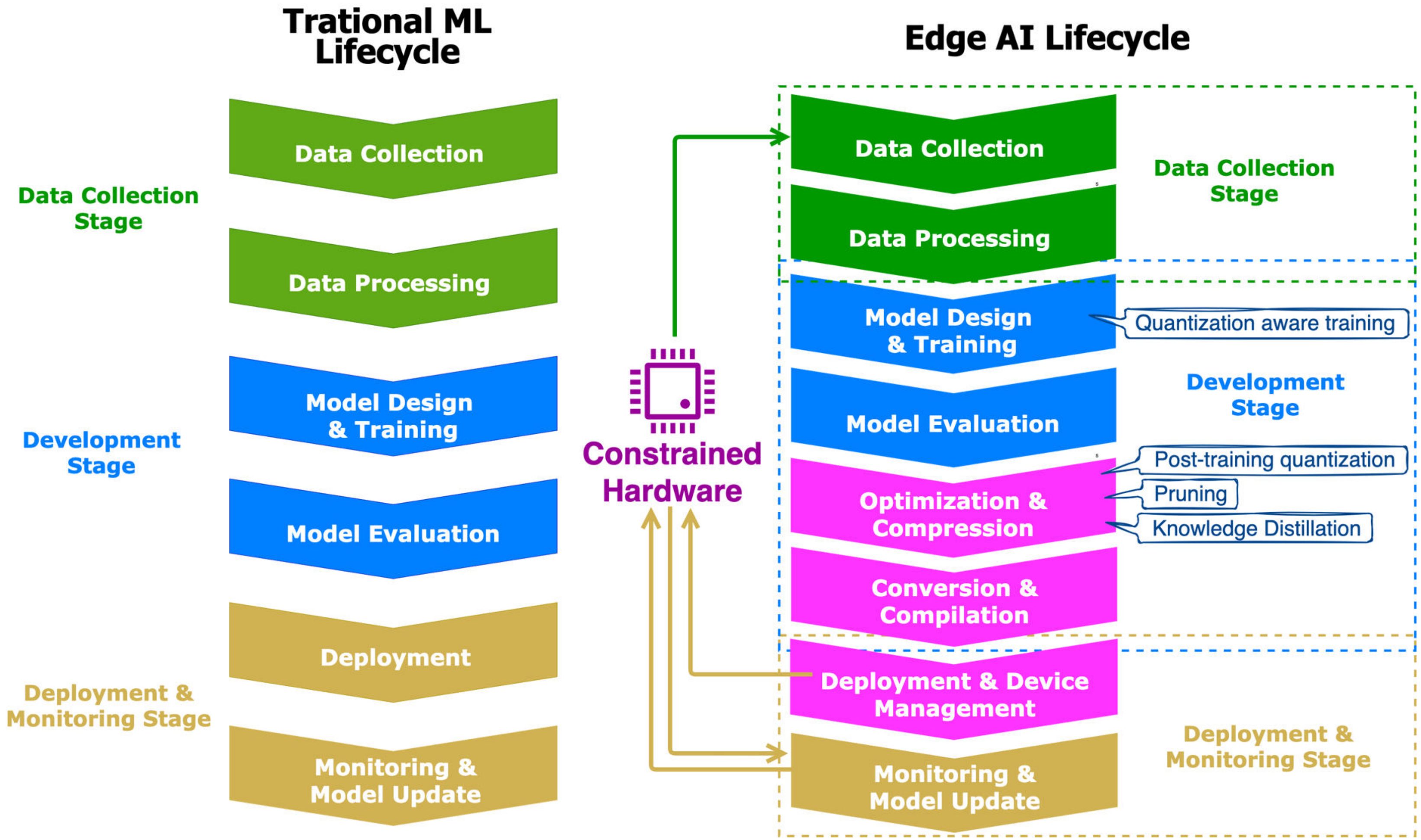


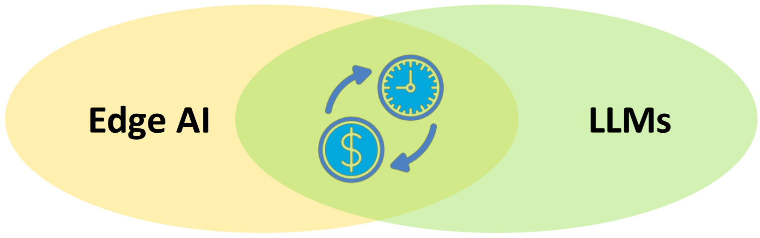
Model	Pricing	Pricing with Batch API*
gpt-4o	$2.50 / 1M input tokens	$1.25 / 1M input tokens
	$1.25 / 1M cached** input tokens	
	$10.00 / 1M output tokens	$5.00 / 1M output tokens
gpt-4o-2024-08-06	$2.50 / 1M input tokens	$1.25 / 1M input tokens
	$1.25 / 1M cached** input tokens	
	$10.00 / 1M output tokens	$5.00 / 1M output tokens
gpt-4o-2024-05-13	$5.00 / 1M input tokens	$2.50 / 1M input tokens
	$15.00 / 1M output tokens	$7.50 / 1M output tokens

API Cost

Sources:
https://medium.com/@saeedshamshiri_94060/looking-inside-gpt-synthesizer-and-the-idea-of-llm-based-code-generation-ff776b9e902f
https://medium.com/@diegodursel/coding-with-chat-gpt-real-intelligence-b5e6e6f129b4
https://openai.com/api/pricing/

LLMs for Edge AI Lifecycle Automation

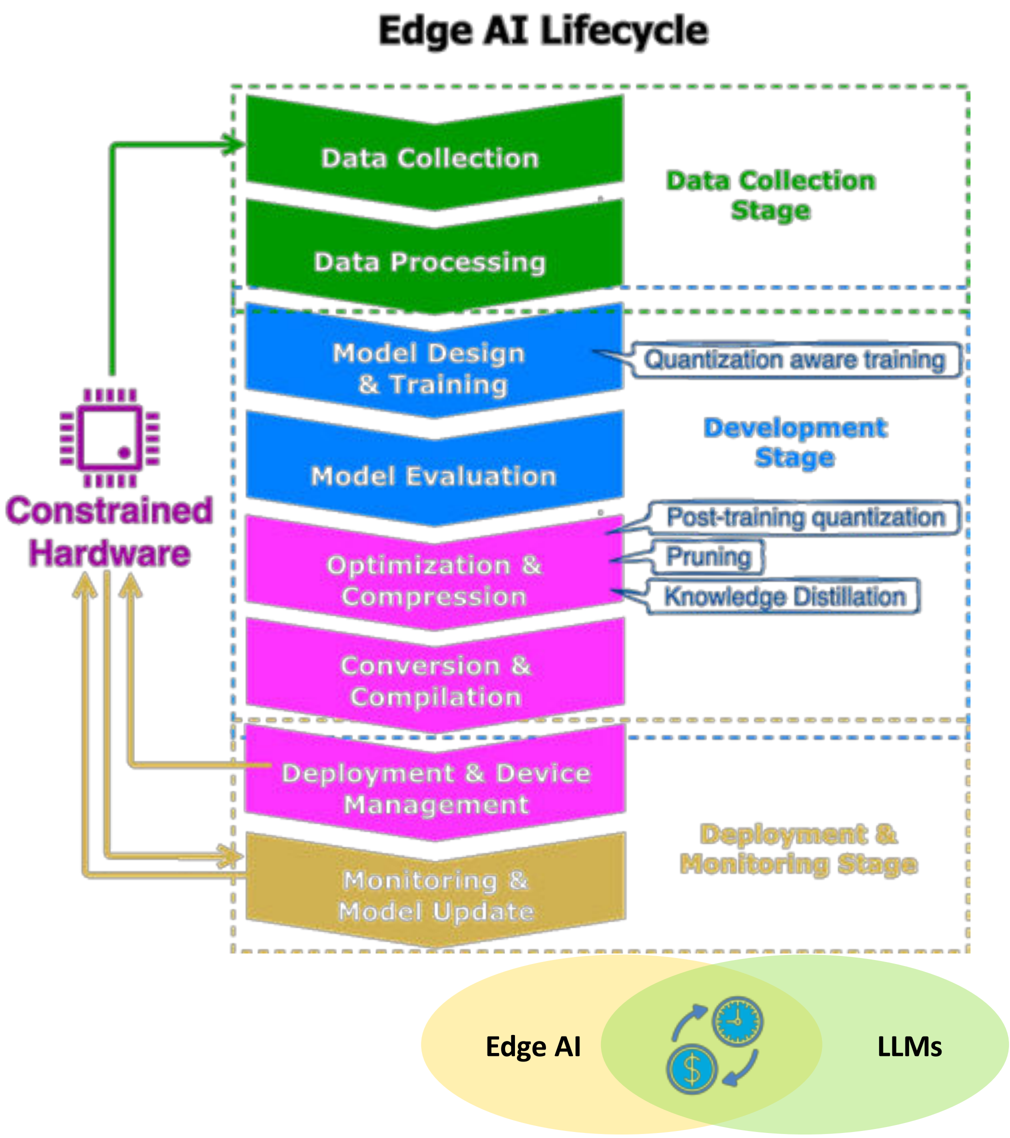




What Is Then This Work About?

Three main questions we would like to answer:

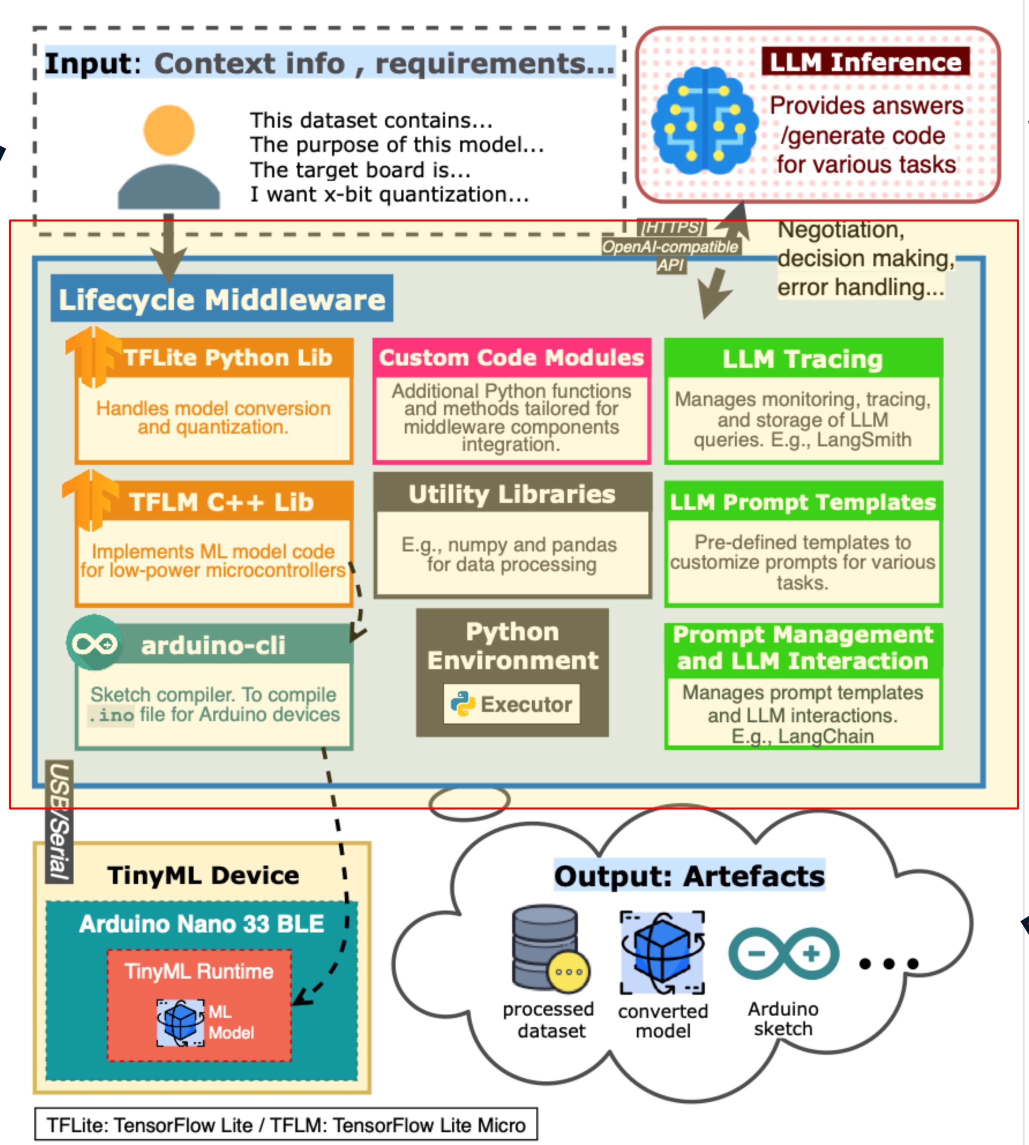
1. What aspects of the **Edge AI lifecycle** can be processed and automated using LLMs?
2. How can **LLMs** be effectively **tailored** to optimize **Edge AI lifecycle stages**?
3. What are the **trade-offs**, **challenges**, and real-world considerations when integrating LLMs with *EdgeAIOps*?



The Overall View of Our Framework

1 The human in the loop

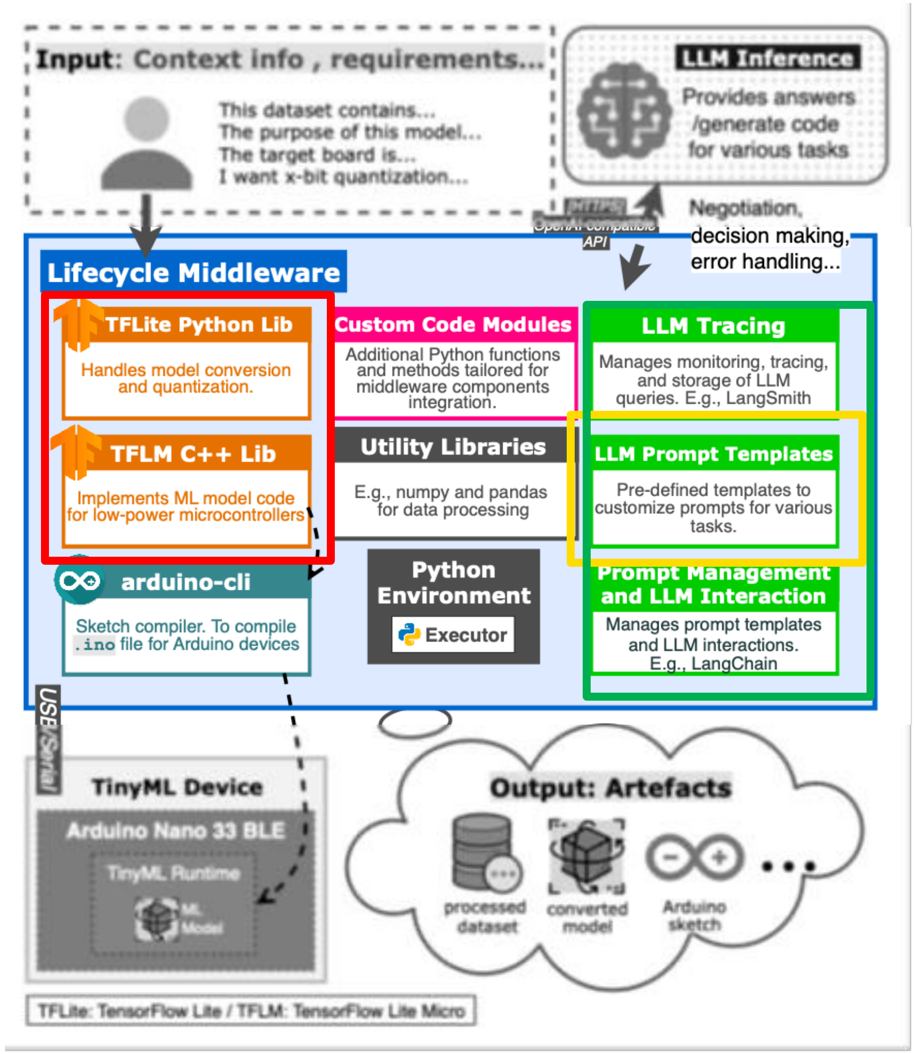
2 The Large Language Model API

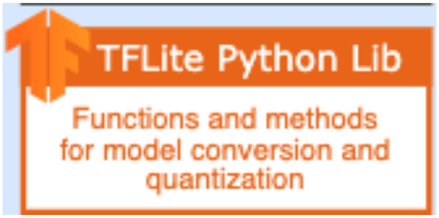


3 End-Device

4 Output

The Core of Our Framework





ML Software Libraries

LangChain is a framework designed to build applications powered by LLMs that can connect to external data sources and perform dynamic decision-making based on those interactions.

LangSmith is a tool for debugging, testing, and monitoring LLM applications.

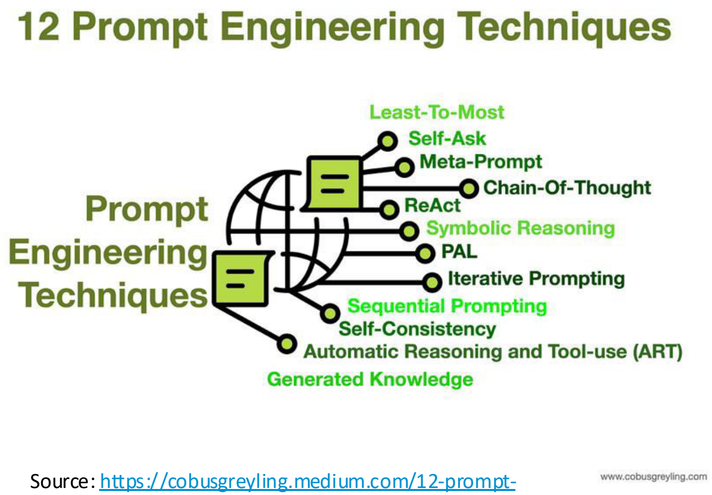
LLMs Supporting Tools

Predefined structure or format used to guide the LLMs' responses by embedding variables or placeholders within a fixed text – It helps ensuring consistent and targeted outputs for specific tasks.

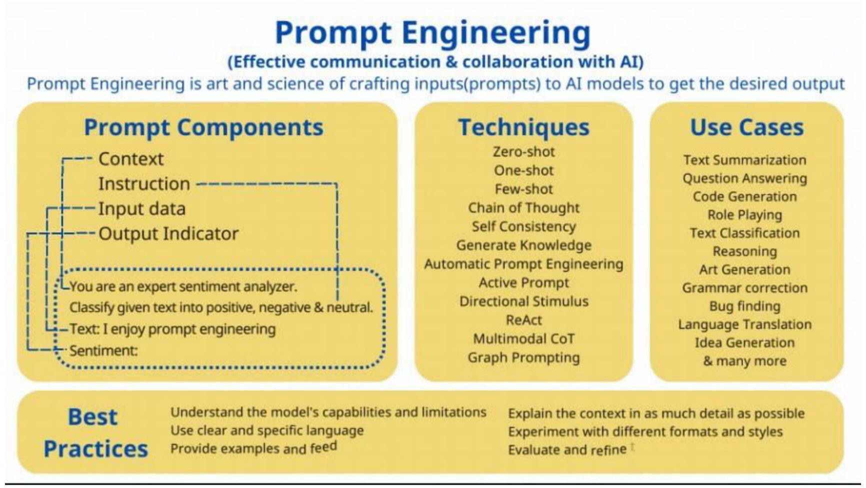
Prompt Templates

What About Prompts Templates?

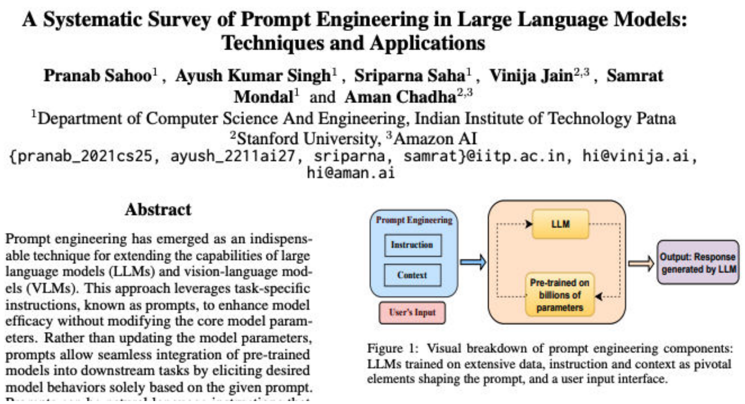
Predefined structure or format used to guide the LLMs' responses by embedding variables or placeholders within a fixed text – It helps ensuring consistent and targeted outputs for specific tasks.



Source: https://cobusgreyling.medium.com/12-prompt-engineering-techniques-644481c857aa



Source: https://www.linkedin.com/pulse/importance-prompt-engineering-natural-language-c-cardoso-r-



A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications

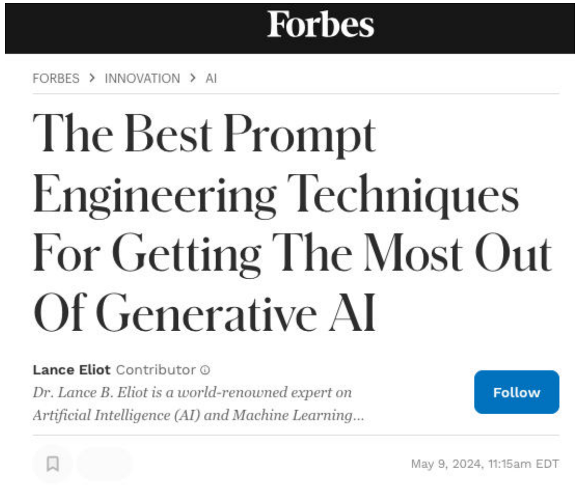
Pranab Sahoo[1], Ayush Kumar Singh[1], Sriparna Saha[1], Vinija Jain[2,3], Samrat Mondal[1] and Aman Chadha[2,3]
[1]Department of Computer Science And Engineering, Indian Institute of Technology Patna
[2]Stanford University, [3]Amazon AI
{pranab_2021cs25, ayush_2211ai27, sriparna, samrat}@iitp.ac.in, hi@vinija.ai, hi@aman.ai

Abstract

Prompt engineering has emerged as an indispensable technique for extending the capabilities of large language models (LLMs) and vision-language models (VLMs). This approach leverages task-specific instructions, known as prompts, to enhance model efficacy without modifying the core model parameters. Rather than updating the model parameters, prompts allow seamless integration of pre-trained models into downstream tasks by eliciting desired model behaviors solely based on the given prompt.

Figure 1: Visual breakdown of prompt engineering components: LLMs trained on extensive data, instruction and context as pivotal elements shaping the prompt, and a user input interface.

Source: https://arxiv.org/pdf/2402.07927



Forbes

FORBES > INNOVATION > AI

The Best Prompt Engineering Techniques For Getting The Most Out Of Generative AI

Lance Eliot Contributor
Dr. Lance B. Eliot is a world-renowned expert on Artificial Intelligence (AI) and Machine Learning...

Follow

May 9, 2024, 11:15am EDT

Source: https://www.forbes.com/sites/lanceeliot/2024/05/09/the-best-prompt-engineering-techniques-for-getting-the-most-out-of-generative-ai/

What About Prompts Templates?

Predefined structure or format used to guide the LLMs' responses by embedding variables or placeholders within a fixed text – It helps ensuring consistent and targeted outputs for specific tasks.

Prompt Templates

```
full_pro_tem = """
### CONTEXT ###\n{context_prompt}
\n### OBJECTIVE ###\n{task_prompt}
"""

context_pro_tem = """You are an expert in Tiny Machine Learning (TinyML), \
    highly skilled in the workflow, tools, techniques, \and best practices of TinyML operations. \
    Your expertise extends to hardware, including microcontrollers. You will be asked questions \
    regarding various tasks of TinyML, for example, data engineering, model designing, \
    model evaluation, model conversion, deployment sketch developing, etc, of TinyML and may need \
        to generate code to execute corresponding tasks, for example, data cleaning, model training code, etc."""
```

Modular prompt templates developed for different TinyML lifecycle stages include:

1. ***Context***: Define LLM's role and expertise.
2. ***Task-Specific Instructions***: Tailored for Edge AI tasks.
3. ***Error Handling***: Help correct execution errors.
4. ***Specification***: Provides a template for application specifications, such as hardware, sensors, software, and their configurations.
5. ***Sketch Guideline Templates***: Provides code generation guidelines.

What About Prompts Templates?

Predefined stru
responses by en
fixed text – It
outputs for spec

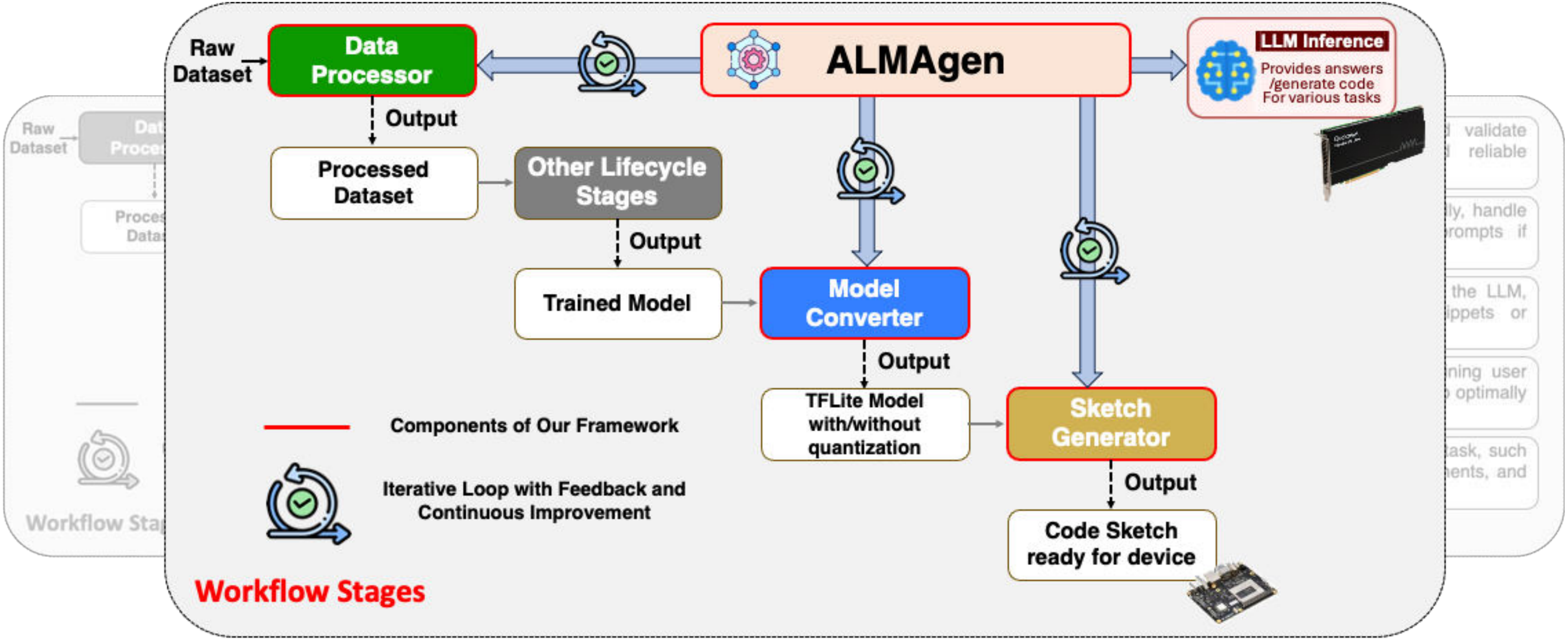
```
# Context template for TinyML expert role
context_pro_tem: str = (
    """You are an expert in Edge AI, highly skilled in the workflow, tools, \
        techniques, and best practices of Edge AI operations. Your expertise extends to hardware, including microcontrollers. \
        You will be asked questions regarding various tasks of Edge AI, for example, data engineering, model designing, \
        model evaluation, model conversion, deployment sketch developing, etc, of Edge AI and may need \
        to generate code to execute corresponding tasks, for example, data cleaning, model training code, etc."""
)

# Format template for specification filling responses
format_spec_filling_pro_tem: str = r"""
### RESPONSE FORMAT ###
- Output only one code block
- In the code block, only put the updated \"application_specifications\":{{}}, remove everything outside of it.
- The response should be clear, accurate and strictly following the target_goal. Instead of assuming things, skip anything you are unsure about the detail.
"""

# Template for filling application specifications
# placeholders: board_fullname, dataset_summary, app_spec_pro_tem
# count_placeholders: 3
spec_filling_pro_tem: str = r"""
### OBJECTIVE ###
- **TASK**: Fill in the requested fields in application_specifications.
- **TASK INSTRUCTIONS**:
Read the value of \"board_fullname\" under \"hardware\" in \"application_specifications\", \
based on the board info, application description, and sensors that will be used in the application, \
fill in proper parameters into the placeholder fields decide_when_generating_code_based_on_given_board_and_application_description \
and decide_when_generating_code_based_on_given_data_sample_and_application_description. Keep \"guideline\" as it is originally. \
Make sure the libraries imported are all compatible with the board {board_fullname}, instead of using the library \"Qualcomm_TensorFlowLite.h\", \
directly use the library \"TensorFlowLite.h\"."
```

Context

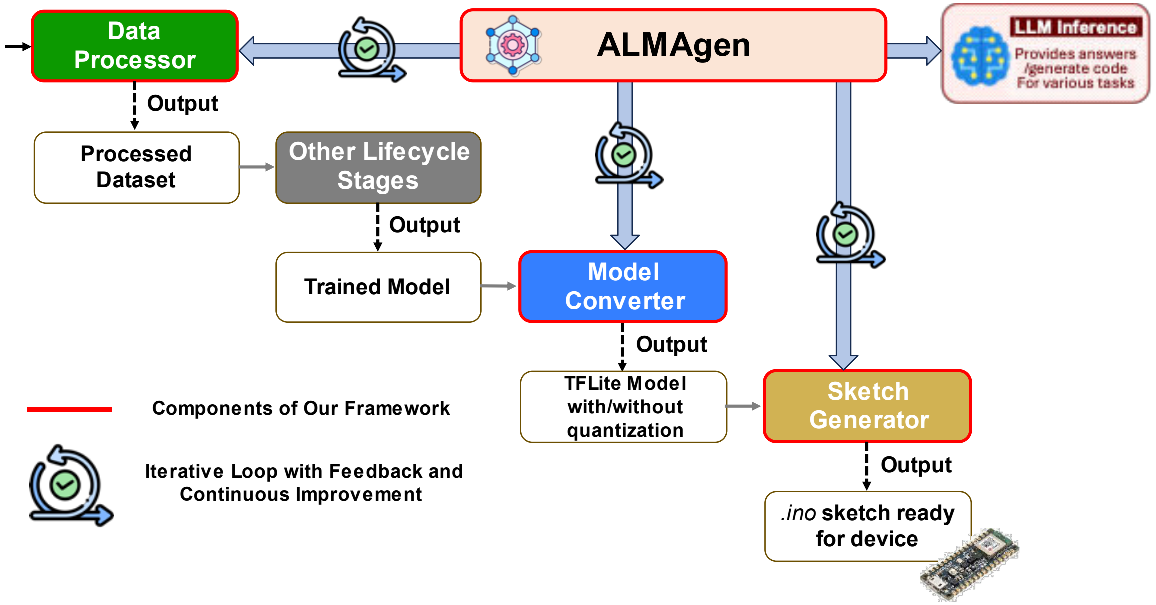
Task

5. *Sketch Guideline Templates*: Provides code generation guidelines.

Framework Workflow



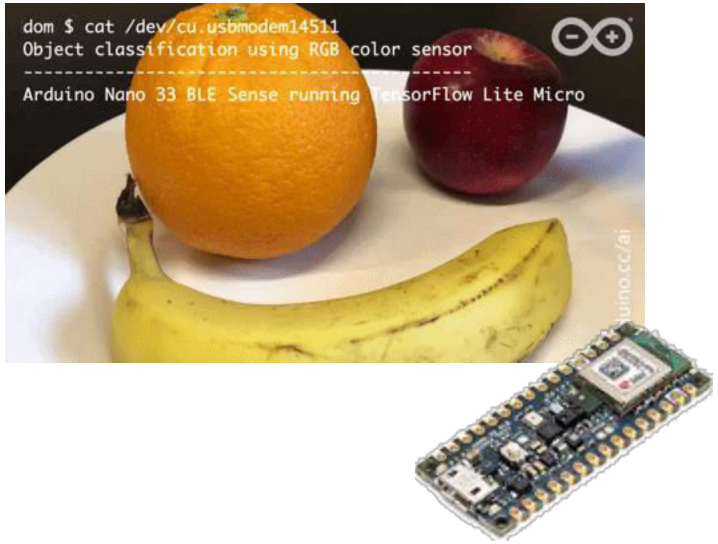
Test Case – CNN-based vision model



Fruit identification using Arduino and TensorFlow

Posted by: ARDUINO TEAM — November 7th, 2019

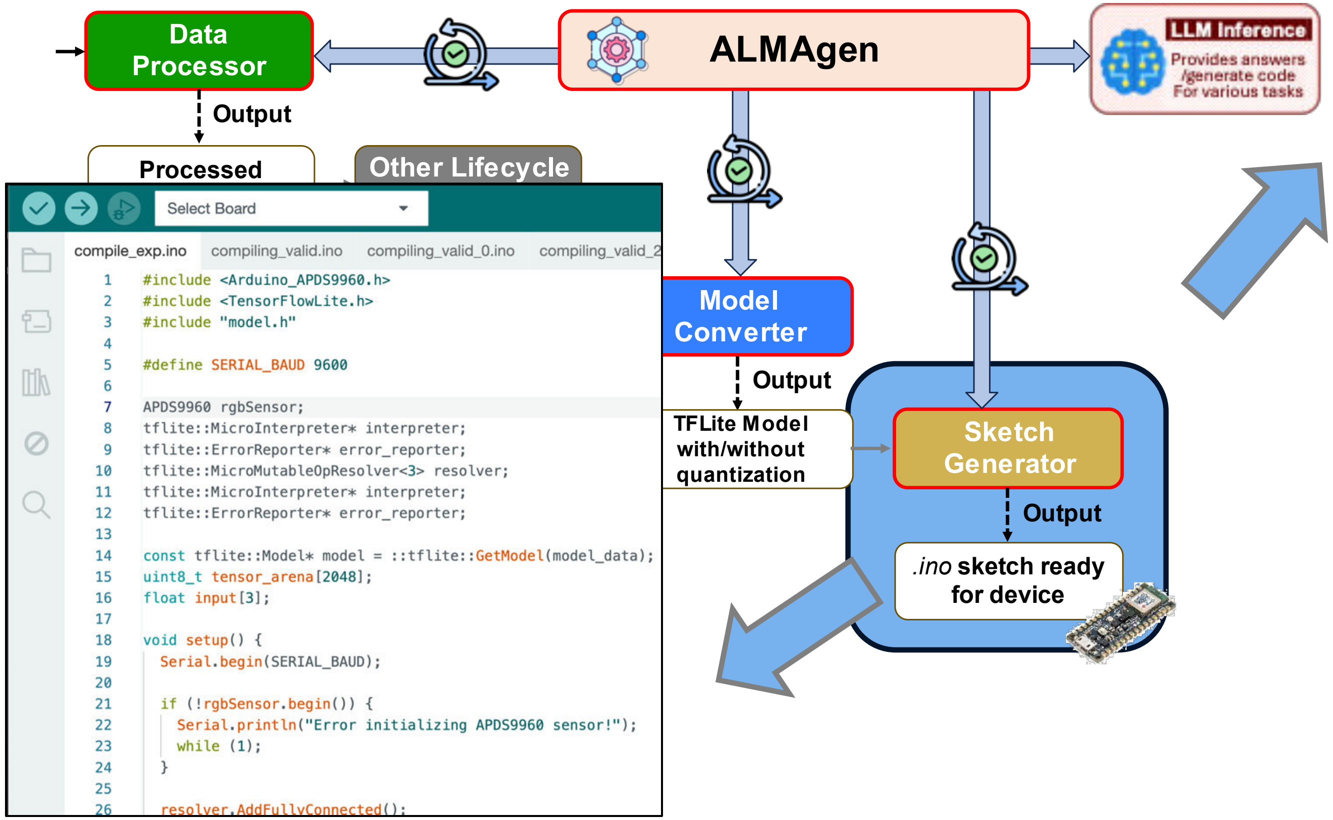
By Dominic Pajak and Sandeep Mistry

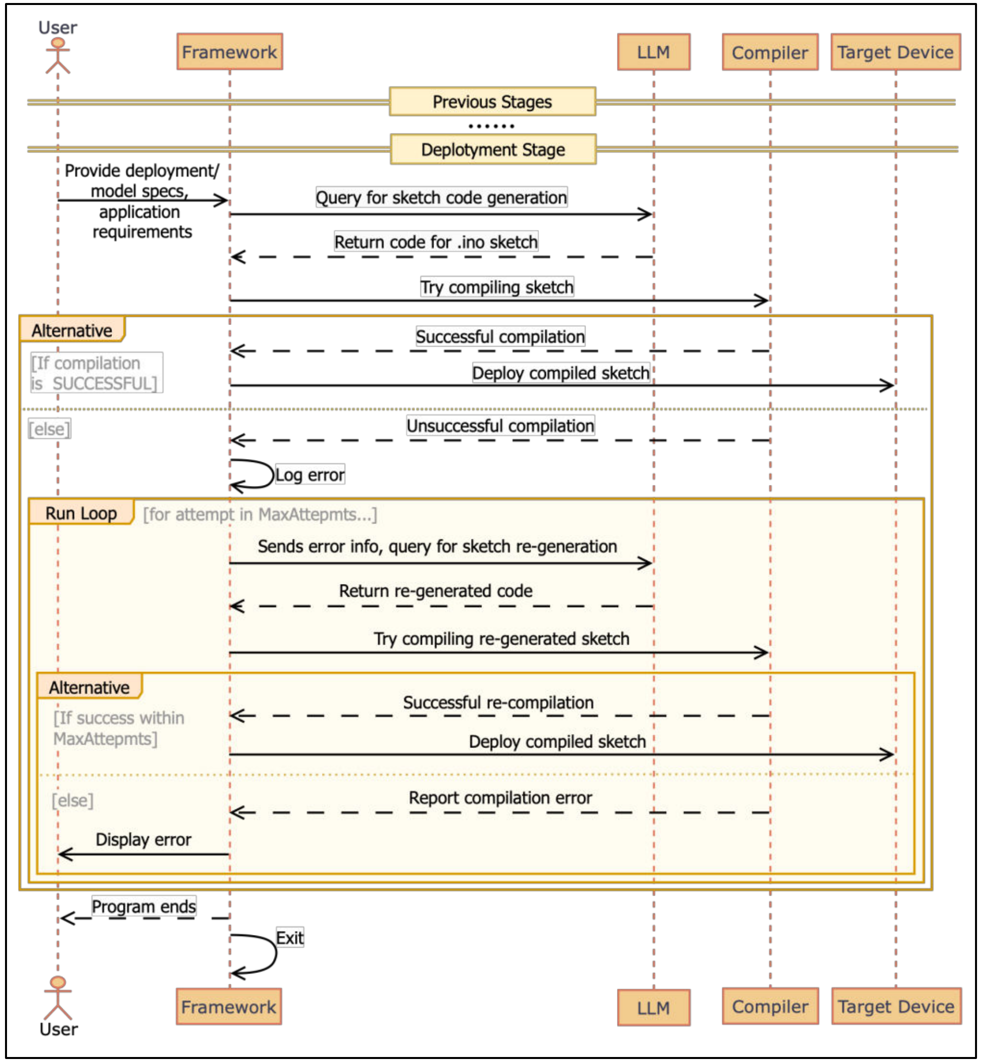


(Source: https://blog.arduino.cc/2019/11/07/fruit-identification-using-arduino-and-tensorflow/)

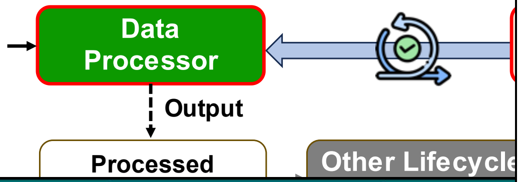
Test Case – CNN-based vision model

Sketch Generation Lifecycle

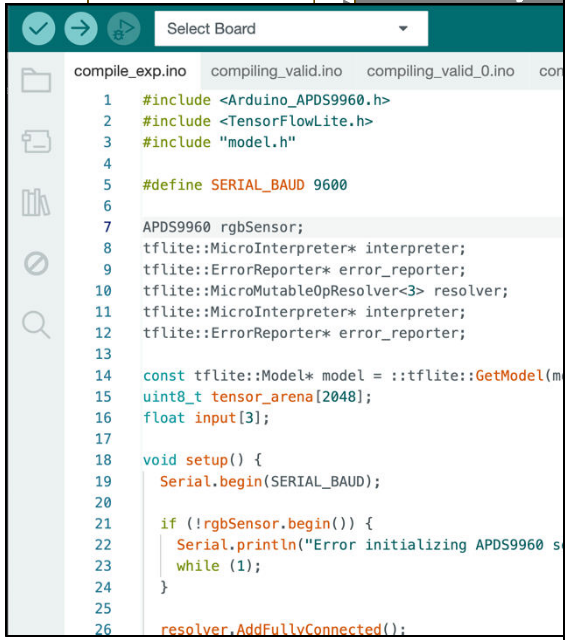




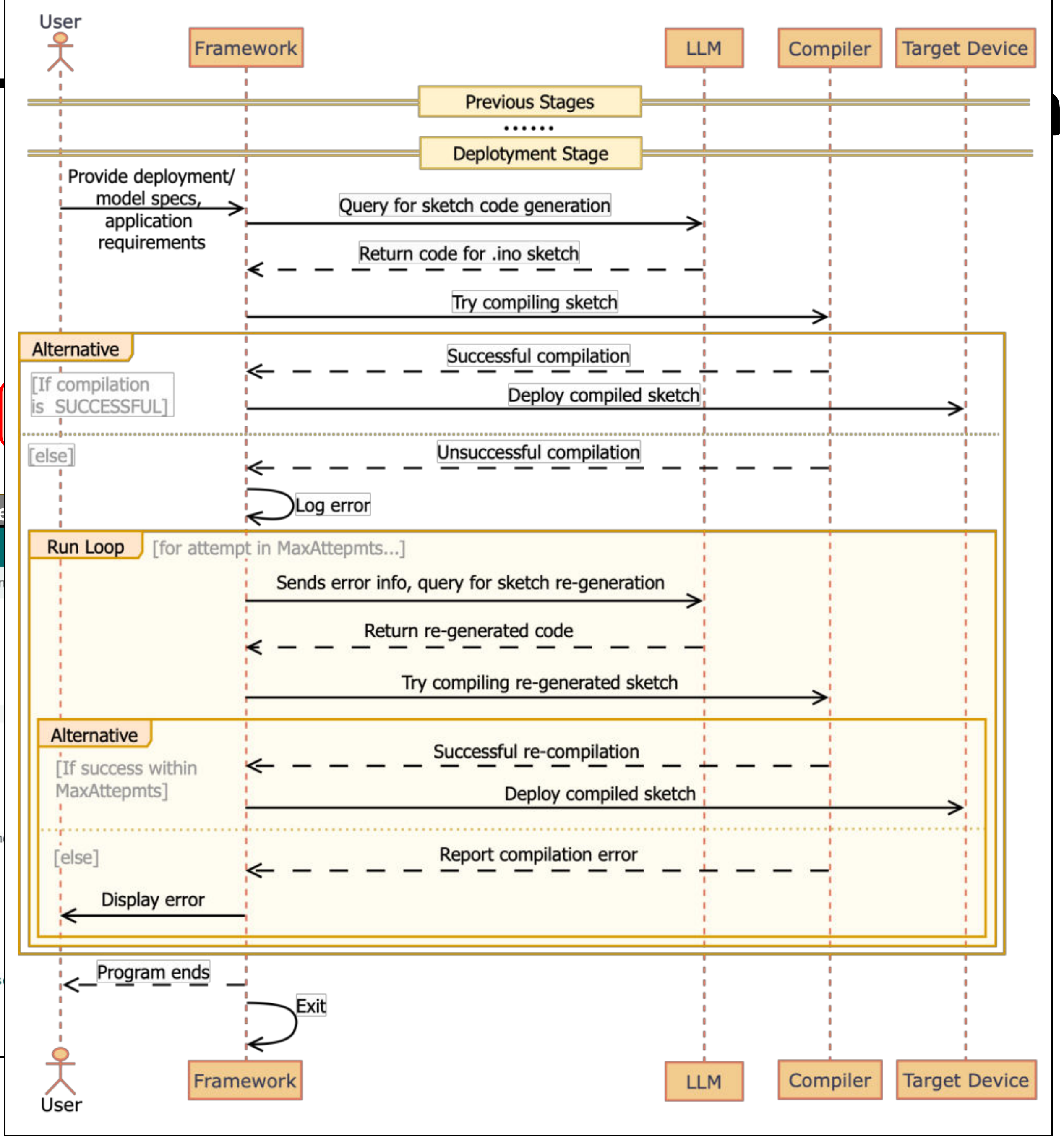
model



Data Processor
Output
Processed
Other Lifecycle

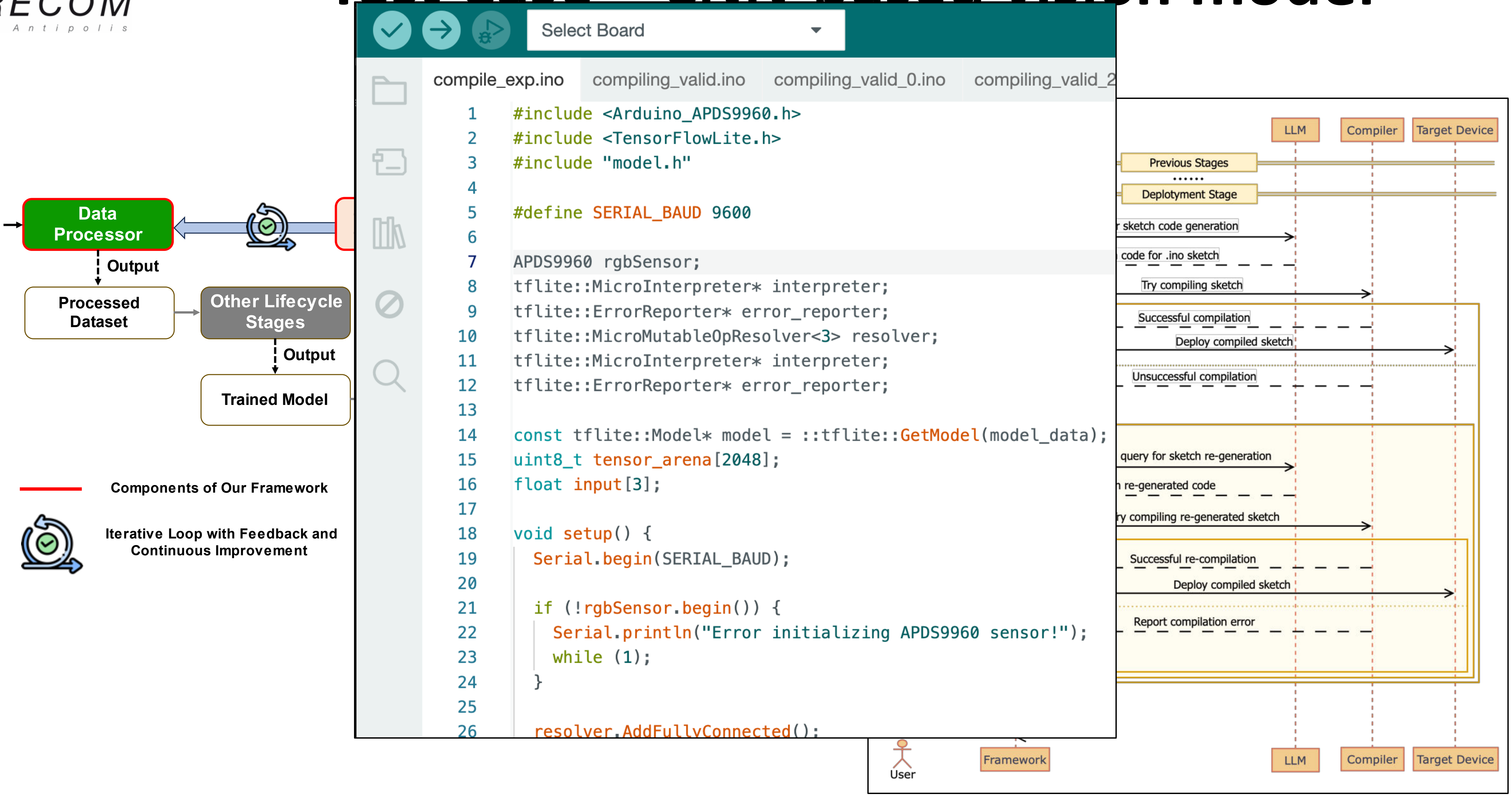


Select Board
compile_exp.ino
compiling_valid.ino
compiling_valid_0.ino
#include <Arduino_APDS9960.h>
#include <TensorFlowLite.h>
#include "model.h"
#define SERIAL_BAUD 9600
APDS9960 rgbSensor;
tflite::MicroInterpreter* interpreter;
tflite::ErrorReporter* error_reporter;
tflite::MicroMutableOpResolver<3> resolver;
tflite::MicroInterpreter* interpreter;
tflite::ErrorReporter* error_reporter;
const tflite::Model* model = ::tflite::GetModel(
uint8_t tensor_arena[2048];
float input[3];
void setup() {
Serial.begin(SERIAL_BAUD);
if (!rgbSensor.begin()) {
Serial.println("Error initializing APDS9960 s
while (1);
}
resolver.AddFullyConnected();

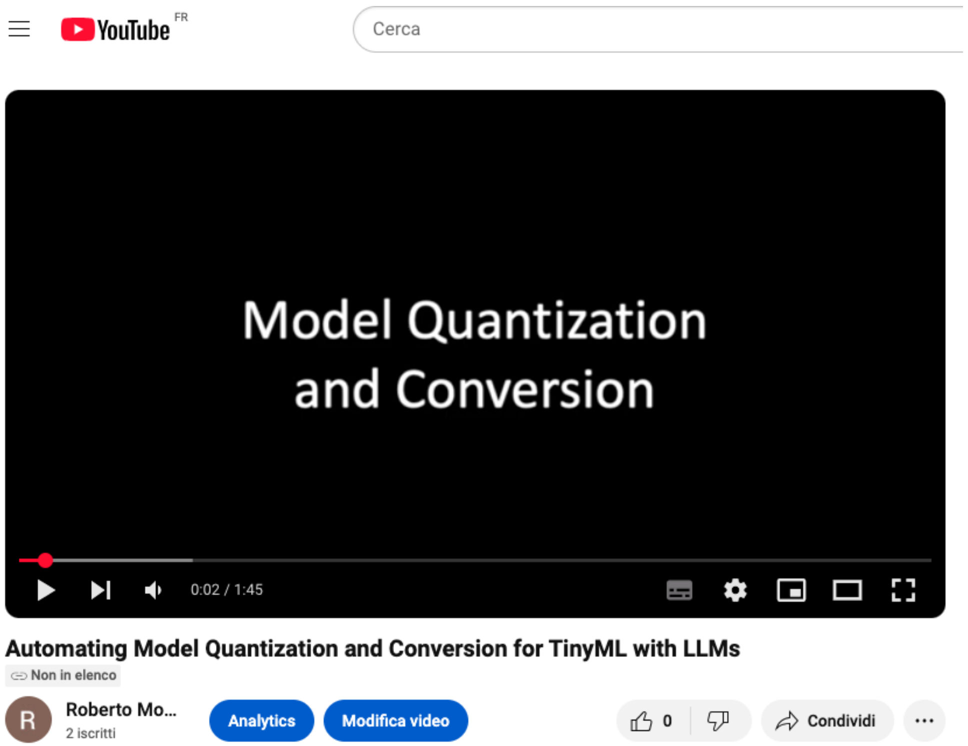


User
Framework
LLM
Compiler
Target Device
Previous Stages
....
Deplotyment Stage
Provide deployment/ model specs, application requirements
Query for sketch code generation
Return code for .ino sketch
Try compiling sketch
Alternative
[If compilation is SUCCESSFUL]
Successful compilation
Deploy compiled sketch
[else]
Unsuccessful compilation
Log error
Run Loop
[for attempt in MaxAttepmts...]
Sends error info, query for sketch re-generation
Return re-generated code
Try compiling re-generated sketch
Alternative
[If success within MaxAttepmts]
Successful re-compilation
Deploy compiled sketch
[else]
Report compilation error
Display error
Program ends
Exit
User
Framework
LLM
Compiler
Target Device

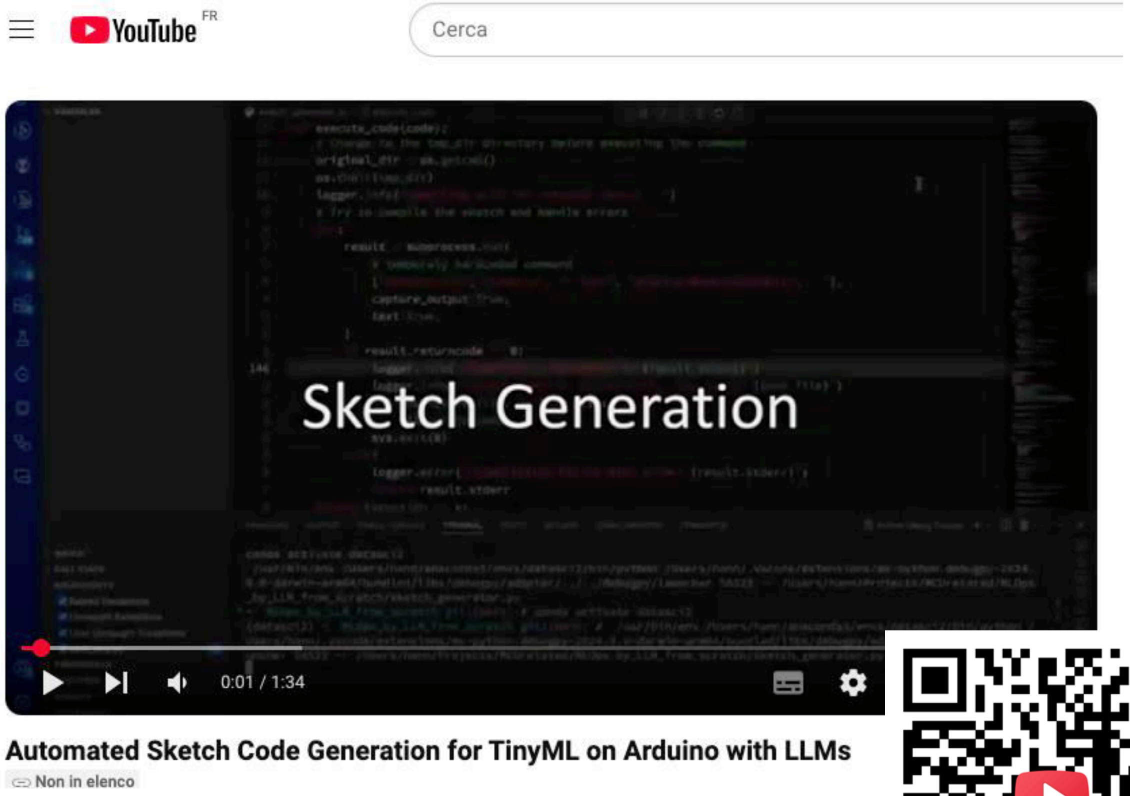
Test Case – CNN-based vision model



Demos

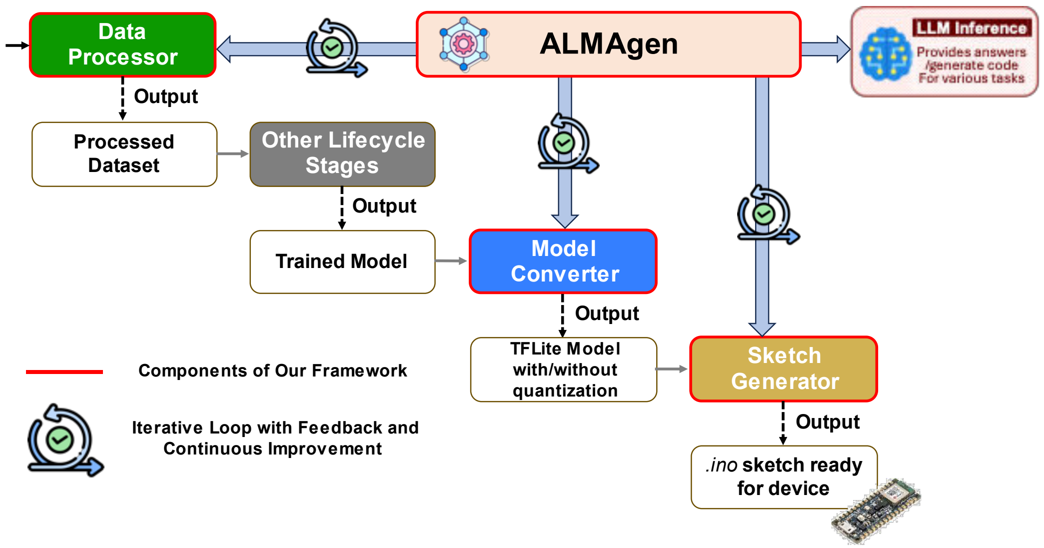


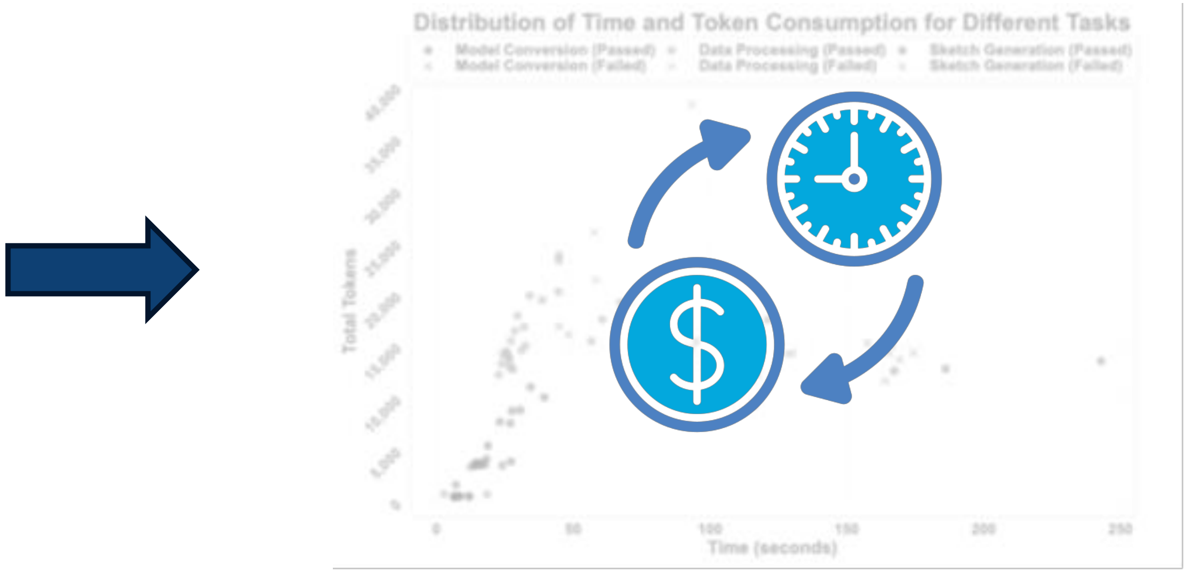
https://www.youtube.com/watch?v=KnJ5m78x_X8



https://www.youtube.com/watch?v=Ojpsb5WnnI8

What is The Cost of This?





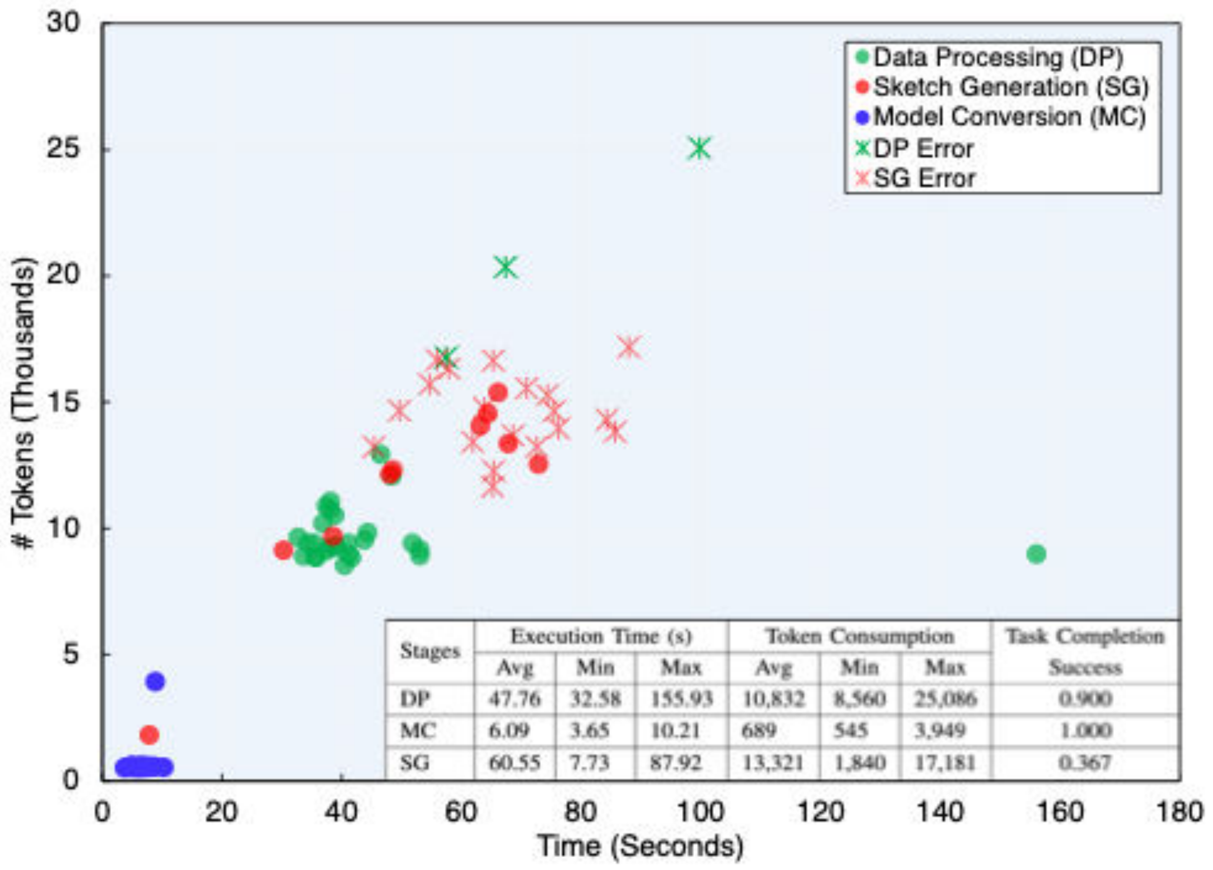
The data presented in this empirical evaluation is subject to change as OpenAI models are frequently updated, which may impact results over time.

What is The Cost of This?
Tokens and Time Perspective

- A ***token*** is a unit of text (e.g., word, subword, or character) that the model processes.
- **Input tokens** are the tokens derived from the text provided to the model for analysis or generation.
- **Output tokens** are the tokens generated by the model in response to the input, forming the predicted or generated text.

Both input and output tokens impact processing time and computational resources.

What is The Cost of This? Tokens and Time Perspective



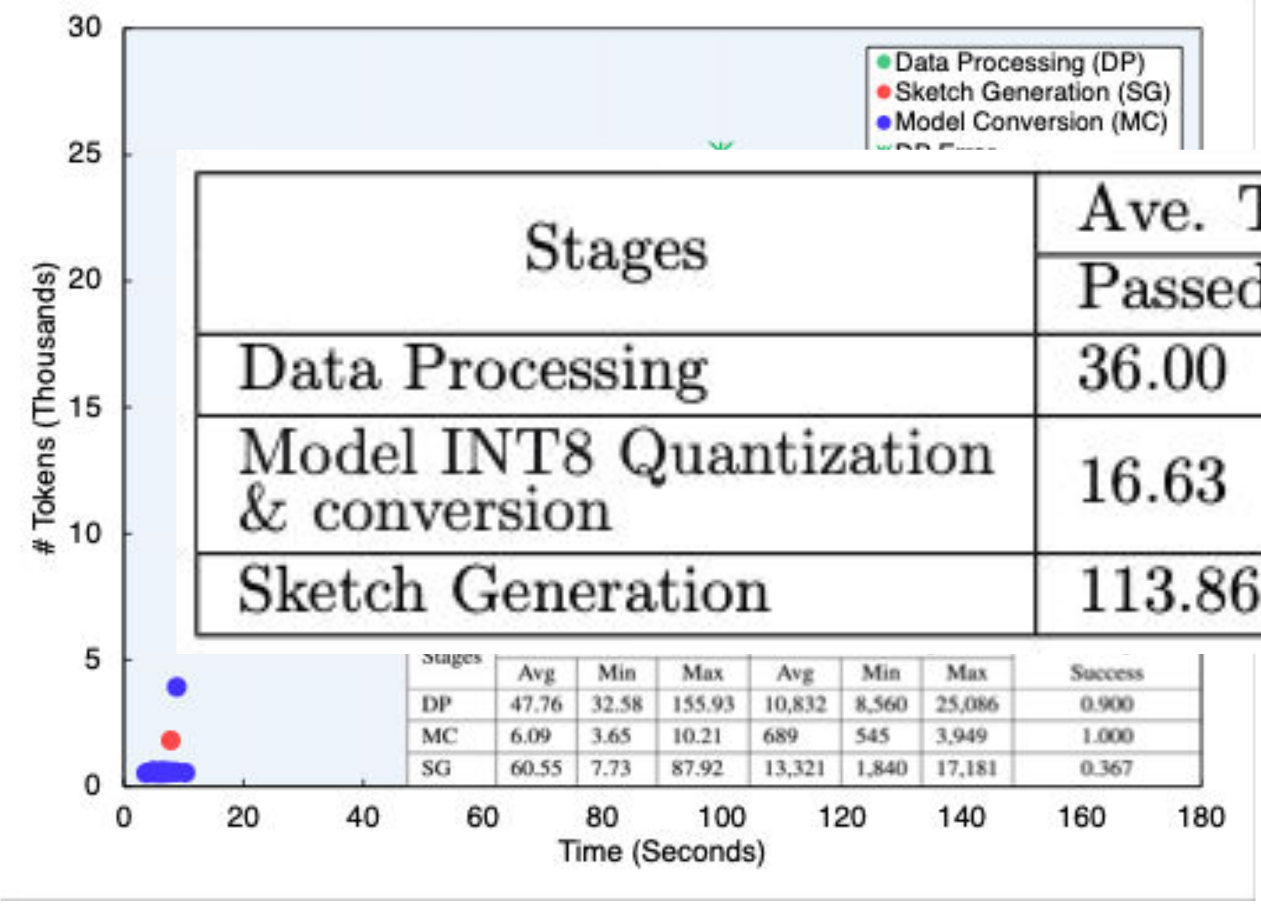
Stages	Execution Time (s)			Token Consumption			Task Completion
	Avg	Min	Max	Avg	Min	Max	Success
DP	47.76	32.58	155.93	10,832	8,560	25,086	0.900
MC	6.09	3.65	10.21	689	545	3,949	1.000
SG	60.55	7.73	87.92	13,321	1,840	17,181	0.367

Stages	Success Rate
Data Processing	0.900
Model INT8 Quantization & conversion	0.933
Sketch Generation	0.300

Stages	Ave. Time (s)		Ave. Total Tokens	
	Passed	Failed	Passed	Failed
Data Processing	36.00	69.79	15284.30	28654.00
Model INT8 Quantization & conversion	16.63	46.58	3849.43	16622.00
Sketch Generation	113.86	119.58	10094.89	14286.14

EURECOM
Sophia Antipolis

What is The Cost of This? Tokens and Time Perspective

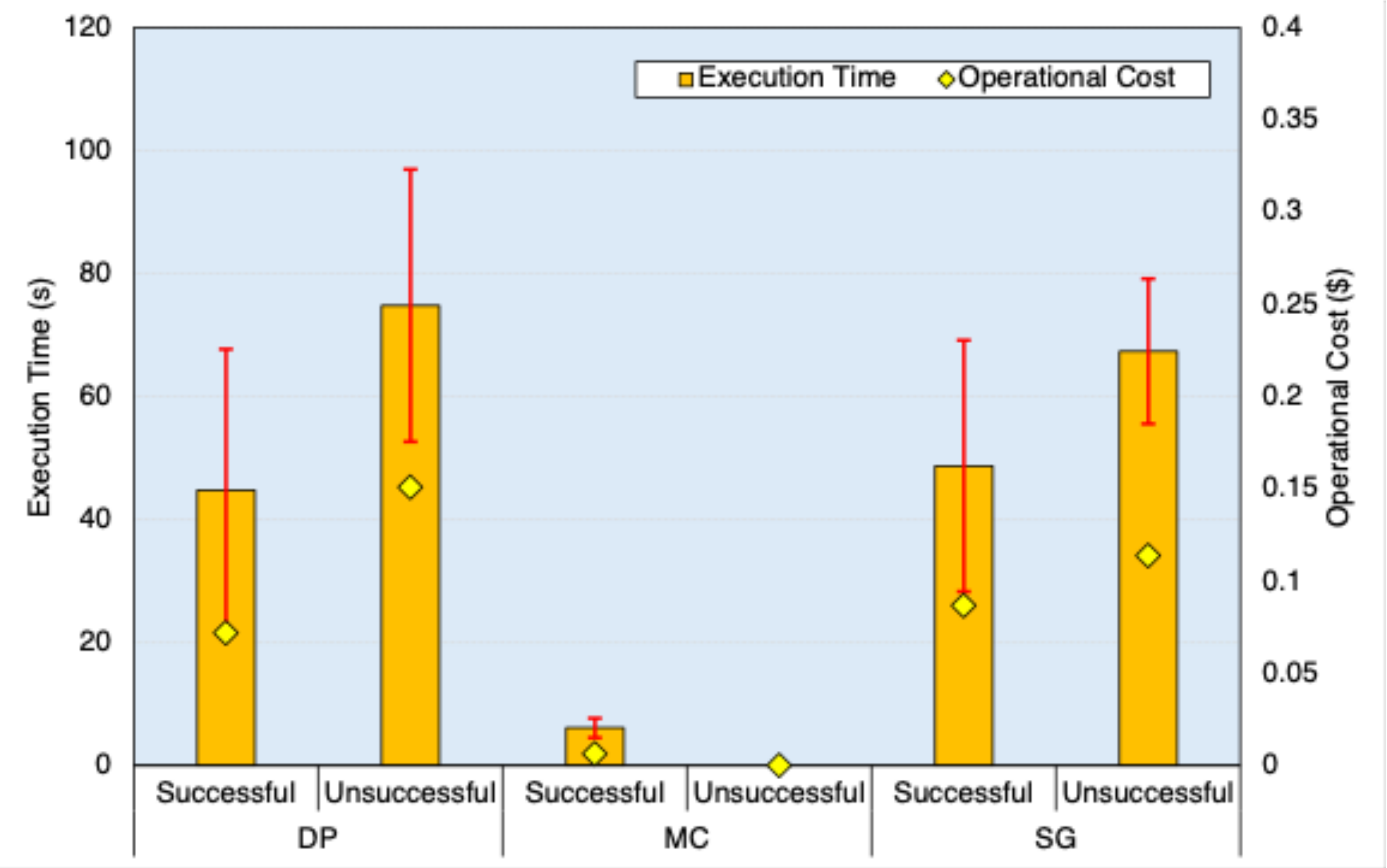


Stages	Avg	Min	Max	Avg	Min	Max	Success
DP	47.76	32.58	155.93	10,832	8,560	25,086	0.900
MC	6.09	3.65	10.21	689	545	3,949	1.000
SG	60.55	7.73	87.92	13,321	1,840	17,181	0.367

Stages	Ave. Time (s)		Ave. Total Tokens	
	Passed	Failed	Passed	Failed
Data Processing	36.00	69.79	15284.30	28654.00
Model INT8 Quantization & conversion	16.63	46.58	3849.43	16622.00
Sketch Generation	113.86	119.58	10094.89	14286.14

Stages	Success Rate

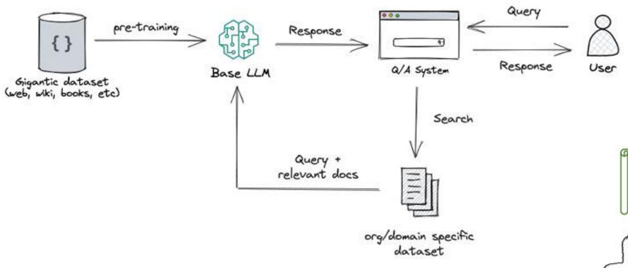
What is The Cost of This? Monetary Cost Perspective



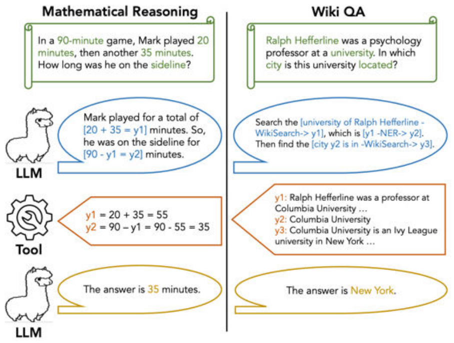
Reality, Illusion, or Opportunity?

Potential for expanded automation in the TinyML lifecycle.

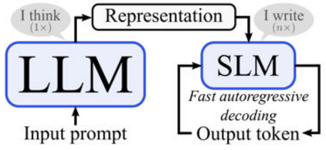
- **LLM fine-tuning** can improve code generation reliability we can enhance versatility.
- **Integrating LLMs with external tools** for enhanced reasoning can unlock additional level of reasoning (and so improvements).
- **Involving the end-device** may enable abstraction of device-specific info and real-time optimization for more efficient lifecycle management.



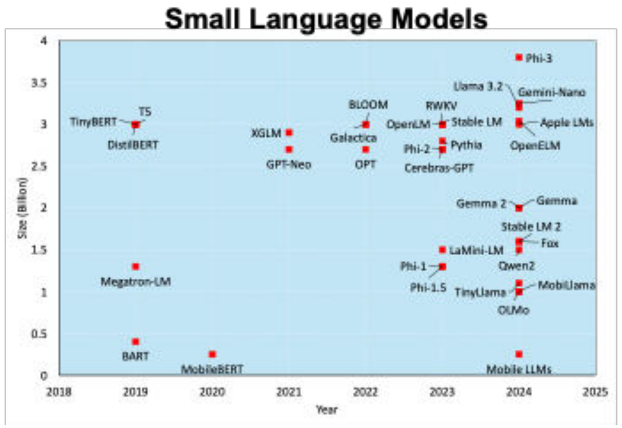
Source: https://towardsdatascience.com/rag-vs-finetuning-which-is-the-best-tool-to-boost-your-llm-application-94654b1eaba7



Source: https://arxiv.org/pdf/2401.17464



Source: https://arxiv.org/pdf/2402.16844



Read the Paper about this Work!

Consolidating TinyML Lifecycle with Large Language Models: Reality, Illusion, or Opportunity?

Guanghan Wu[‡], Sasu Tarkoma[‡], Roberto Morabito[*]
[*]Department of Communication Systems, EURECOM, France.
[‡]Department of Computer Science, University of Helsinki, Finland.

https://arxiv.org/pdf/2501.12420

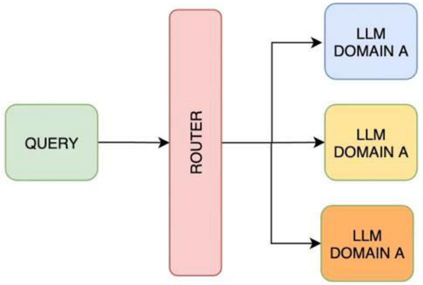
To appear in IEEE IoT Magazine!

Current Activities In This Area

Benchmarking SLMs in Constrained Devices

SLM/LLM Query Routing via Edge Collaboration

Streamlining TinyML Lifecycle with Large Language Models



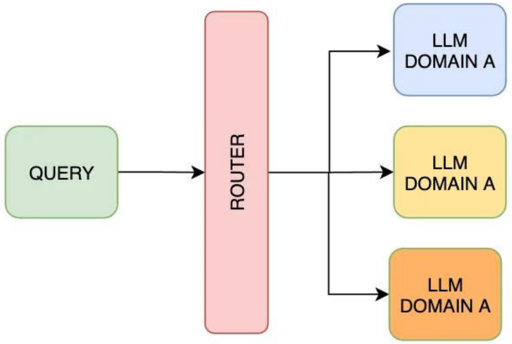
Towards An Efficient LLM Query Routing Via Edge Collaboration

Problem: Mobile or edge devices cannot support large LLMs due to their resource limitations, while cloud-based LLMs are expensive and raise privacy concerns.

Research Question: How can we improve user query responses by collaboratively utilizing smaller local LMs and larger cloud-based LLMs?

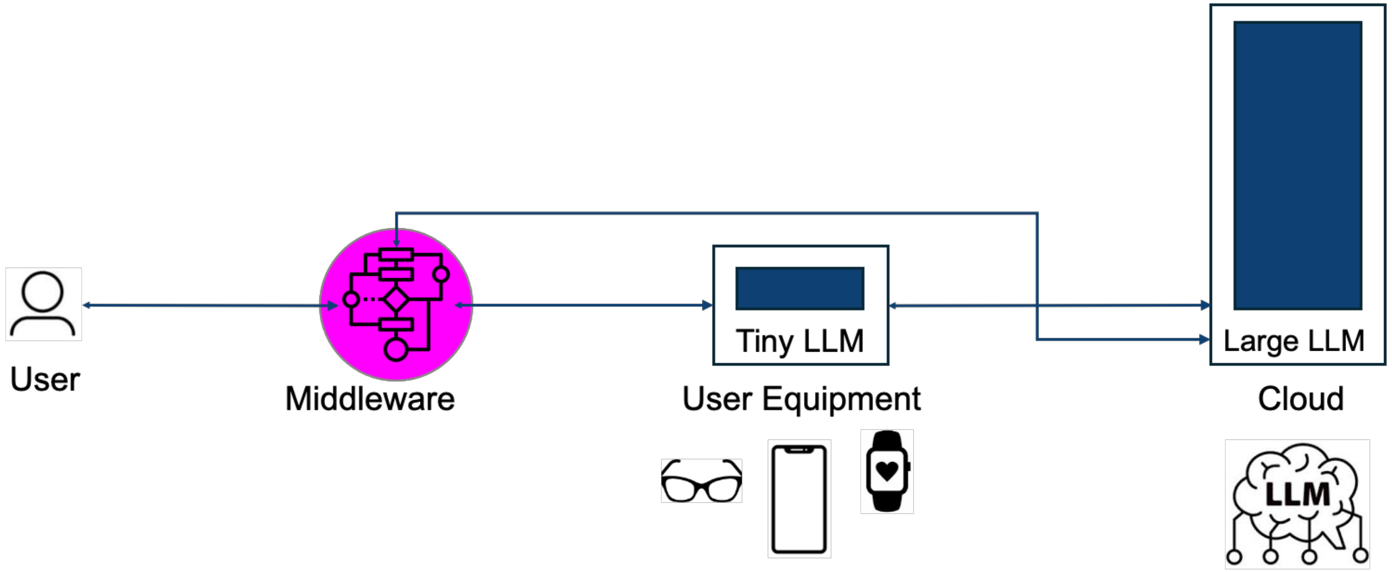
Challenge

- How do we determine which queries should be processed locally and which should be sent to the cloud?
- How can we balance the trade-offs between cost, performance, and privacy?



Work done in collaboration with NOKIA BELL LABS Cambridge (Pervasive Systems group)

Towards An Efficient LLM Query Routing Via Edge Collaboration

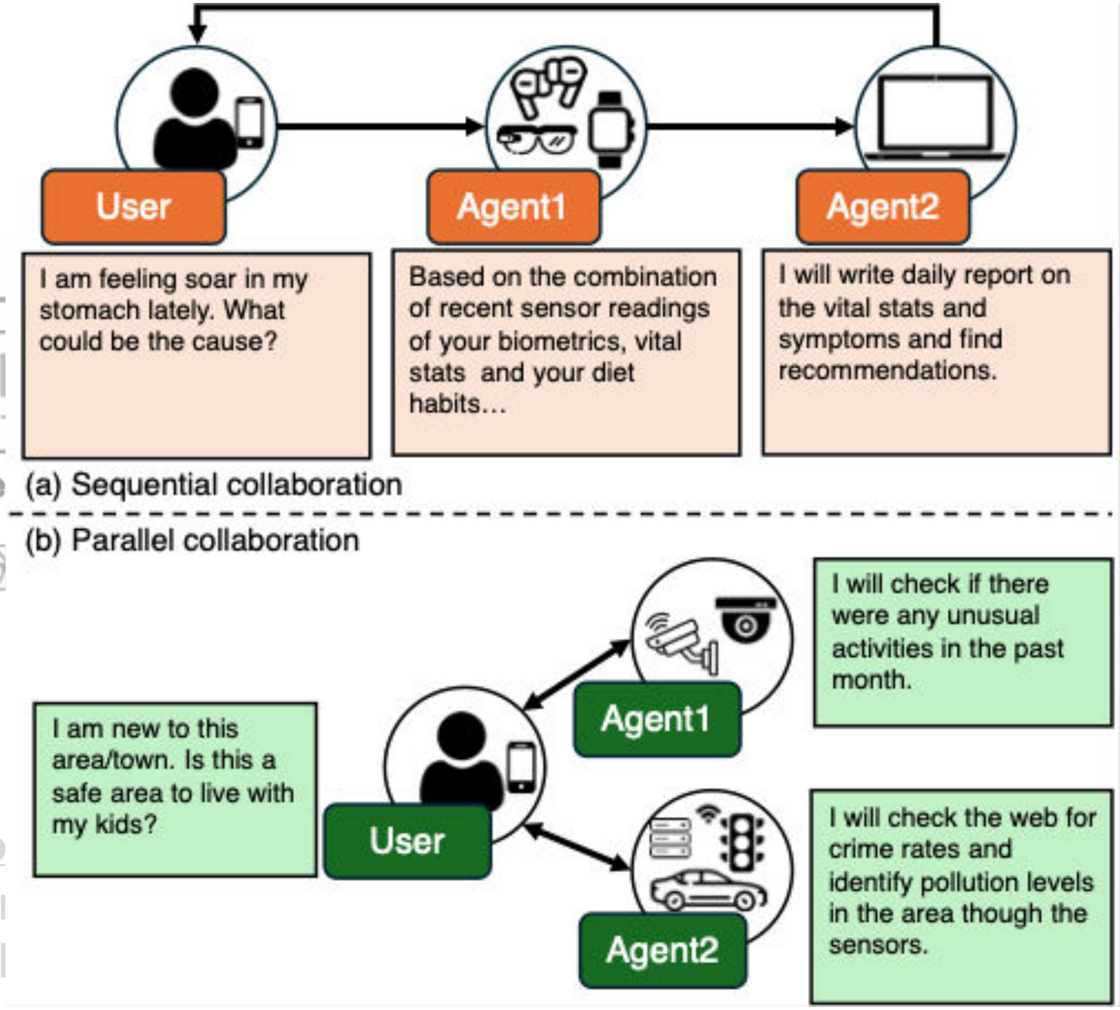


The key concept is 'query split and routing.'

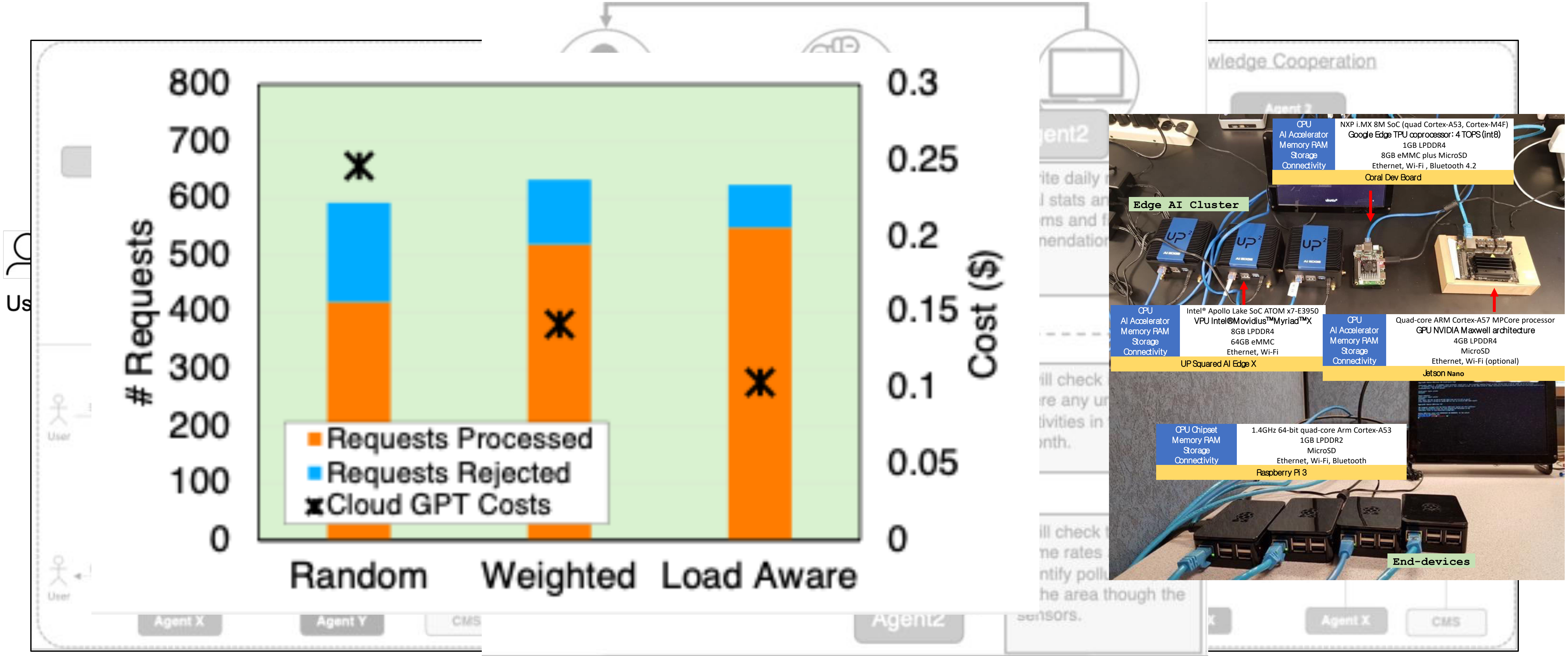
- If the query complexity is high, it is logical to route it to the back-end cloud model.
- If the query complexity is low, it makes sense to process it with the front-end local model.
- However, if the query falls into a gray area between local and back-end processing...

...a **collaborative approach** is ideal. In this case, the local model can handle preliminary processing, filtering, or understanding the context, while the cloud-based LLM provides deeper analysis or complex reasoning.

Towards An Efficient LLM Query Routing Via Edge Collaboration



Towards An Efficient LLM Query Routing Via Edge Collaboration



EURECOM
Sophia Antipolis

Federated Learning Directions

Federated Prompt Fine-Tuning

What if prompt templates or instruction tuning for LLM-based Edge AI workflows were collaboratively adapted using FL?

- Especially relevant when privacy prevents uploading prompt examples to a central server.

FL for Model Routing Policies

Could routing policies for SLM/LLM (i.e., when to run locally or send to the cloud) be learned in a federated fashion, adapting to each device's workload and usage patterns?

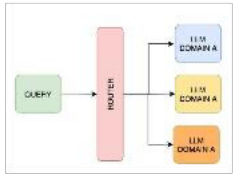
- Leverages FL to train routing classifiers without leaking device usage data.

FL for Lifecycle Feedback Loops

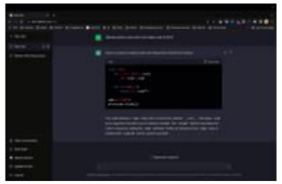
Could lifecycle automation (e.g., code sketch generation) benefit from federated feedback loops, where each device refines LLM usage strategies based on local success/failure logs?

- Could be posed as a federated reinforcement learning or federated policy optimization challenge.

SLM/LLM Query Routing via Edge Collaboration



Streamlining TinyML Lifecycle with Large Language Models



From the Edge to the Cloud: Exploring AI Inference Across the Computing Continuum

(yes, including Generative AI)

Roberto Morabito
Assistant Professor @ EURECOM
https://www.linkedin.com/in/robertomorabito

Kudos to Maximilian Abstreiter, Guanghan Wu (University of Helsinki), and SiYoung Jang (Nokia Bell Labs)