North Carolina Agricultural and Technical State University

MCUNet: An Introduction

AGGIES DO

Machine Learning

Machine Learning allows us to create solutions to problems using techniques that simulate human behavior.

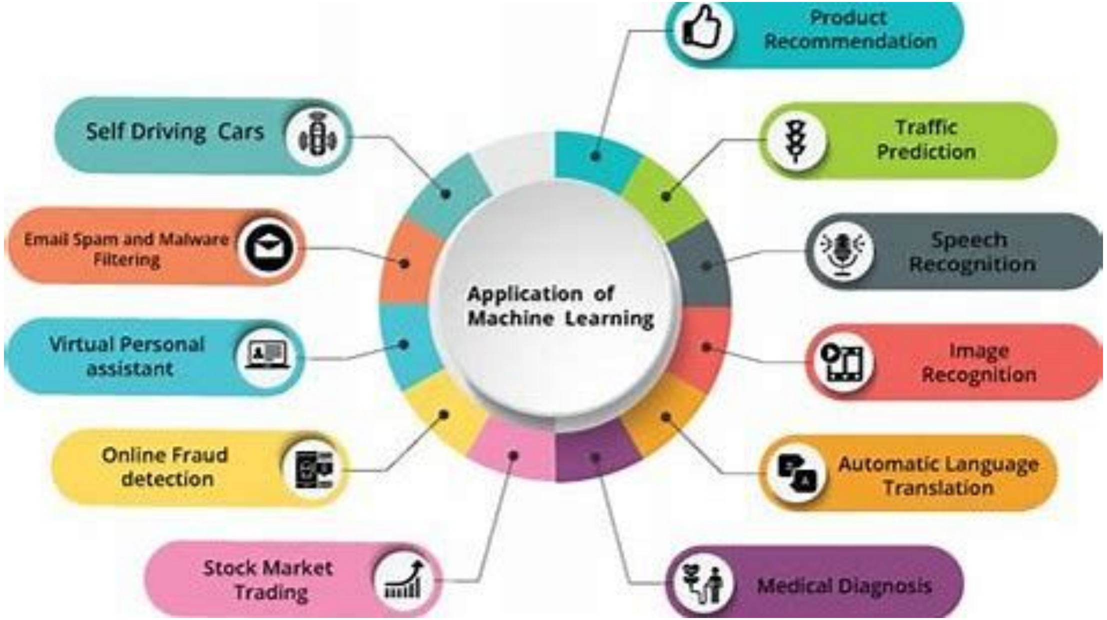
Cloud AI:

- Edge devices send data to a cloud center for data processing, sends back result.

Mobile AI:

- Using mobile phones in Machine Learning solutions, either by sending data to the cloud, or doing the inference on the edge

Machine Learning Applications



TinyML - Machine Learning on Edge Devices

TinyML - Combining artificial intelligence and edge devices to create ML solutions that can fit on a Microcontroller

Advantages of microcontrollers:

- Low-cost
- Low-power
- Compact size

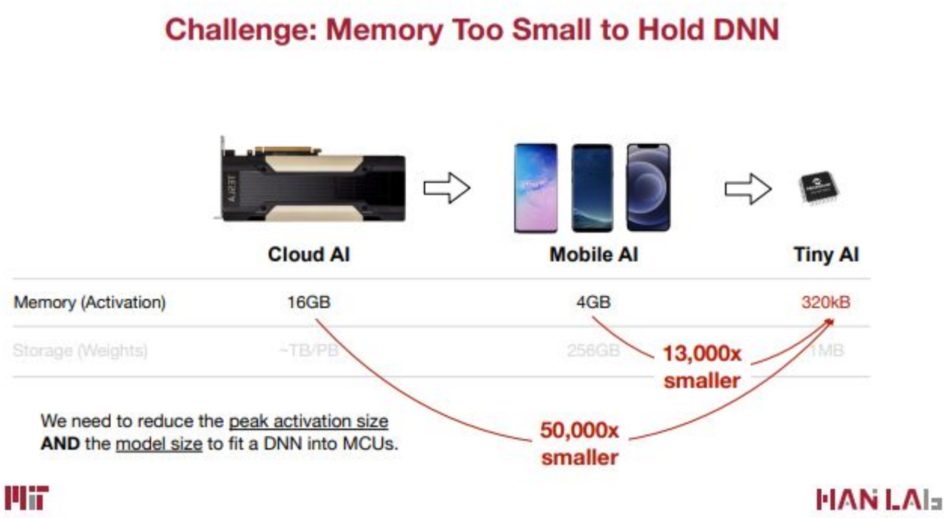
Applications:

- Smart home applications
- Smart retail
- Autonomous Driving

Disadvantages of TinyML

- Small Memory/storage capacity
- High peak activation size

How do we lower the size(storage/memory) of a neural network, while still keeping a high accuracy?



Challenge: Memory Too Small to Hold DNN

	Cloud AI	Mobile AI	Tiny AI
Memory (Activation)	16GB	4GB	320kB
Storage (Weights)	~TB/PB	256GB	1MB

13,000x smaller

50,000x smaller

We need to reduce the peak activation size AND the model size to fit a DNN into MCUs.

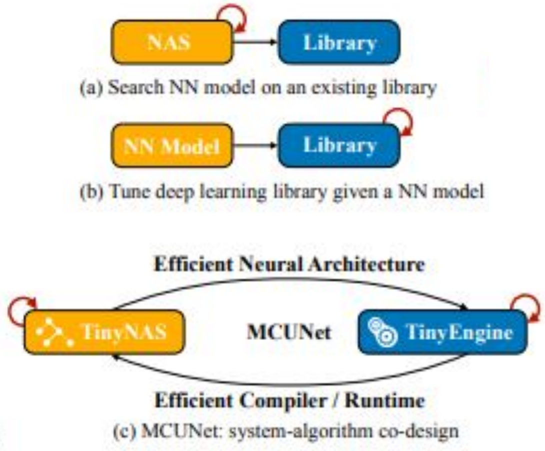
MCUNet

MCUNet consists of 2 co-designed systems:

TinyNAS - Neural Architectural Search engine that finds an optimal neural network model from an already existing library.

TinyEngine - Optimizes memory overhead while maintaining inference efficiency

Combined, this tools allows for deep-learning on off-the-shelf microcontrollers

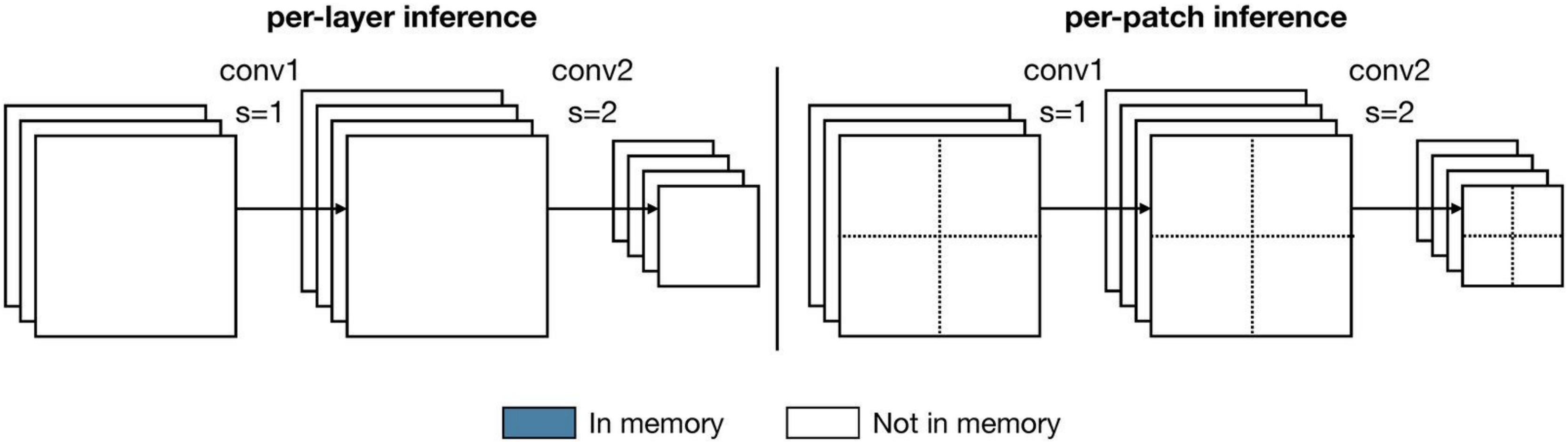


(a) Search NN model on an existing library

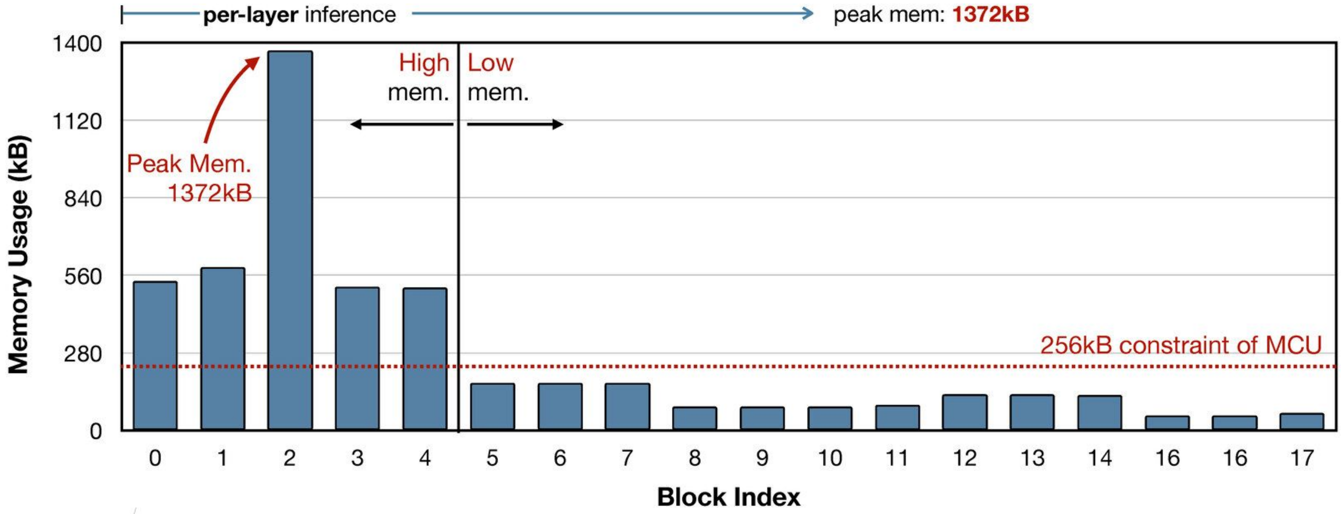
(b) Tune deep learning library given a NN model

(c) MCUNet: system-algorithm co-design

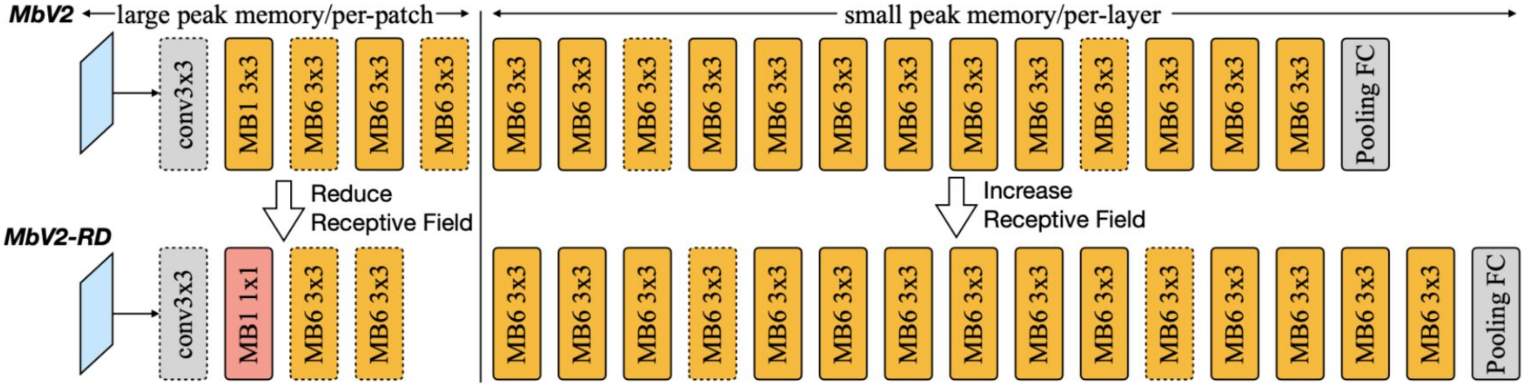
MCUNet - Patch Based Inferencing (TinyEngine)



MCUNet - Patch Based Inferencing (TinyEngine)



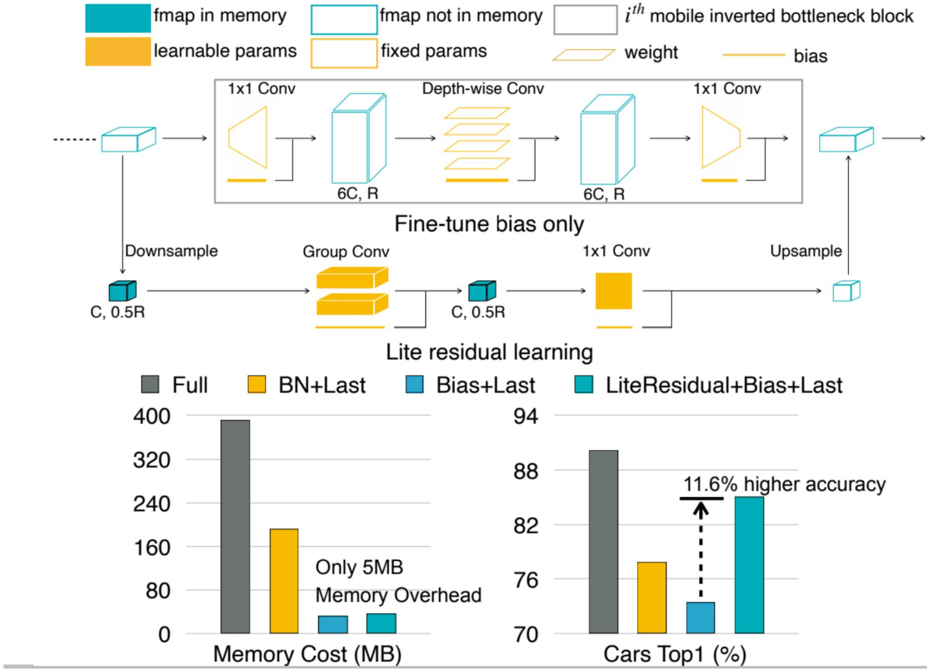
MCUNet - Patch Based Inferencing (TinyEngine)



10% overhead

3% overhead

MCUNet - Training Via Fine-Tuning and Residual Learning



Research Goal

Understand impact of physical faults on TinyML and FL

Phase 1: Reproduce MCUNet Demo (Summer 2023)

- Wake word tutorial from Song Han's TinyML Lab
- MIT 6.5940: TinyML and Efficient Deep Learning Computing

Phase 2: Reimplement MCUNet on RPi Pico (Fall 2023)

- ARM Cortex M0 has more resources (RTL, Debug) than Cortex M7

Phase 3: Inject faults in RPi Pico running MCUNet (Spring 2024 - Spring 2025)

- Larger effort by the ADEPT lab
- Connects to FL studies by ATHENA

Implementation of MCUNet

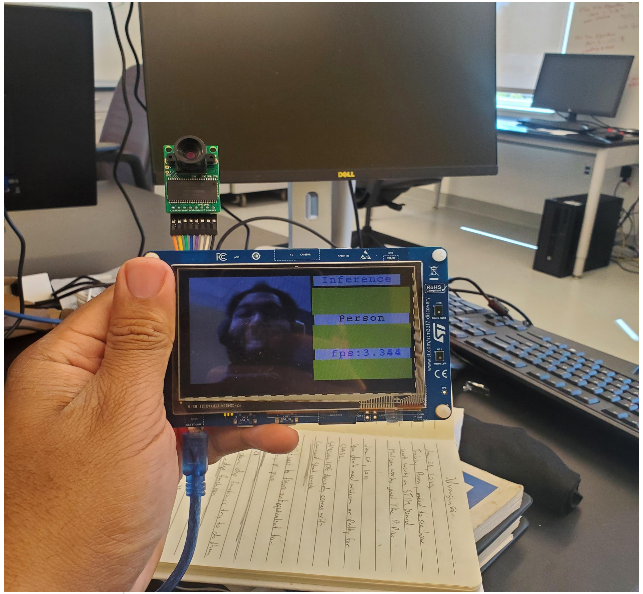
Tools used:

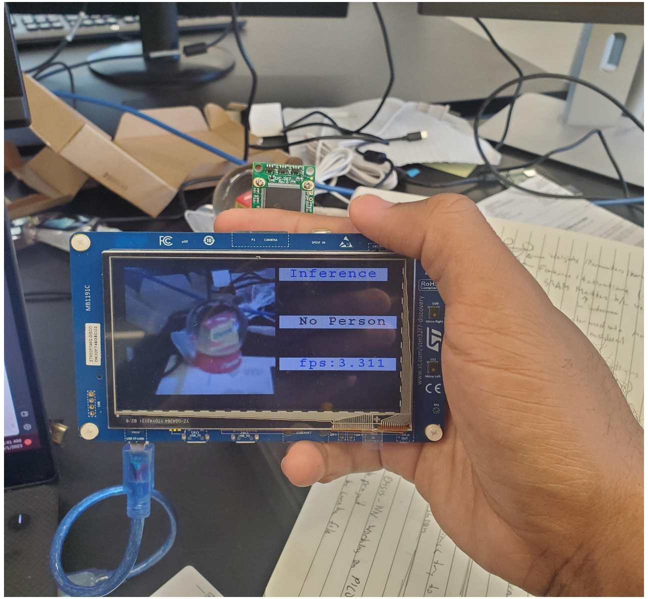
- STM32F746G-DISCO discovery board
- STM32Cube IDE
- MCUNet Github Page(Tinyenine’s inference and training tutorial)

Result:

- Working examples on STM32 Microcontroller for inference and training

MCUNet Implementation Pictures





Benefits of MCUNet on Raspberry Pi Pico

Available RTL

- DesignStart
- Can simulate and emulate physical faults
- Can redesign chip in future studies

Bare-Bones solution

- If a solution is possible for the Arm Cortex M0+, it should be possible for all arm processors

Smaller form factor

- Same functionality in a smaller package

Board Differences

STM32 Microcontroller (STM32F746G-DISCO)

- 1Mb Flash Memory
- 340 Kb RAM
- Arm Cortex M7 processor

Raspberry Pi Pico -

- 16Mb flash memory
- 264 Kb internal RAM
- Arm Cortex-M0+ (No DSP)

Next Steps

- Phase 2:
  - Rewriting DSP algorithms
  - CMSIS for ARM Cortex M0
  - Fitting MCUNet into reduced internal RAM
- Phase 3:
  - Running MCUNet on RPI Pico with GDB
  - Running labmates’ GDB-based fault injector
  - Analysis of fault vulnerabilities in FL testbed