TinyTTA: Efficient Test-time Adaptation via Early-exit Ensembles on Edge Devices

Hong Jia, Young D. Kwon, Alessio Orsino, Ting Dang, Domenico Talia and Cecilia Mascolo

UNIVERSITY OF CAMBRIDGE

SAMSUNG Research

THE UNIVERSITY OF MELBOURNE

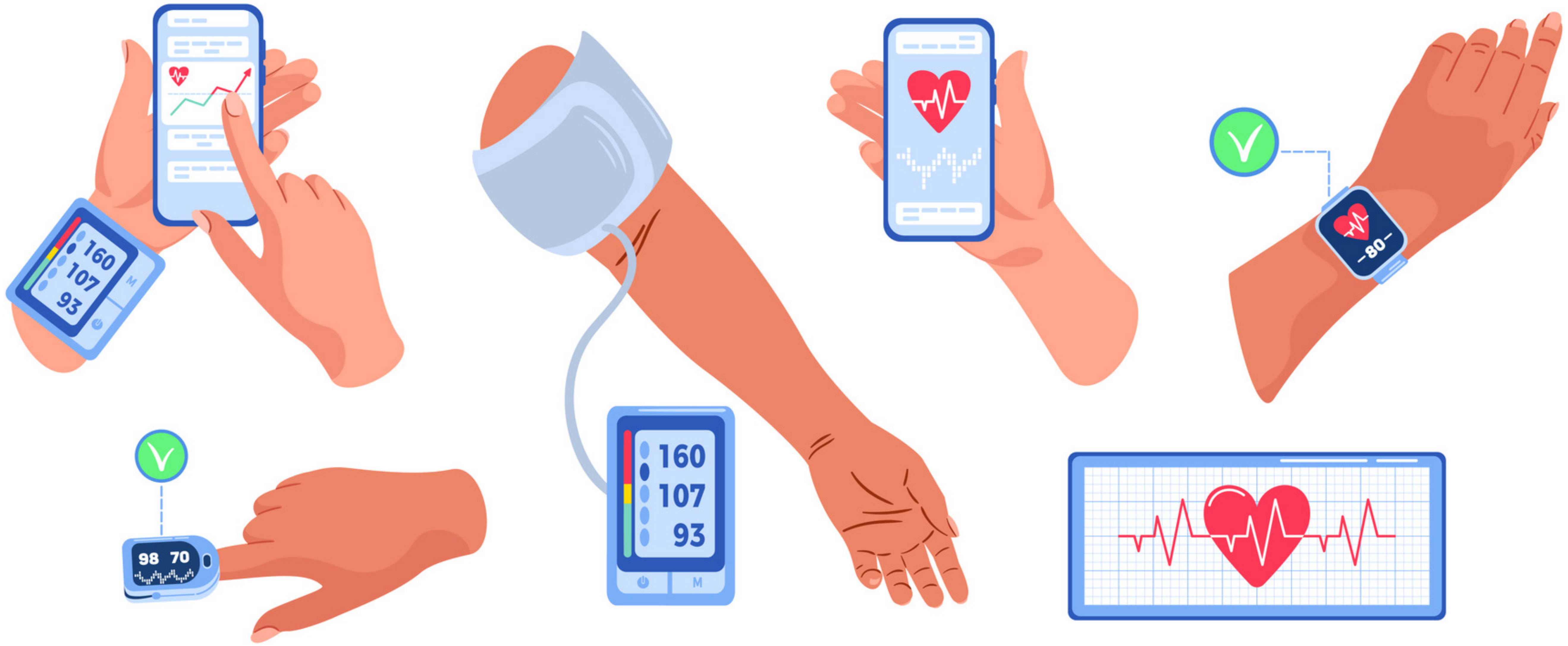
UNIVERSITÀ DELLA CALABRIA

AI/Deep Learning on Edge Devices

- Deploy ML on edge devices becomes popular:

  real-time data analysis and low-latency responses

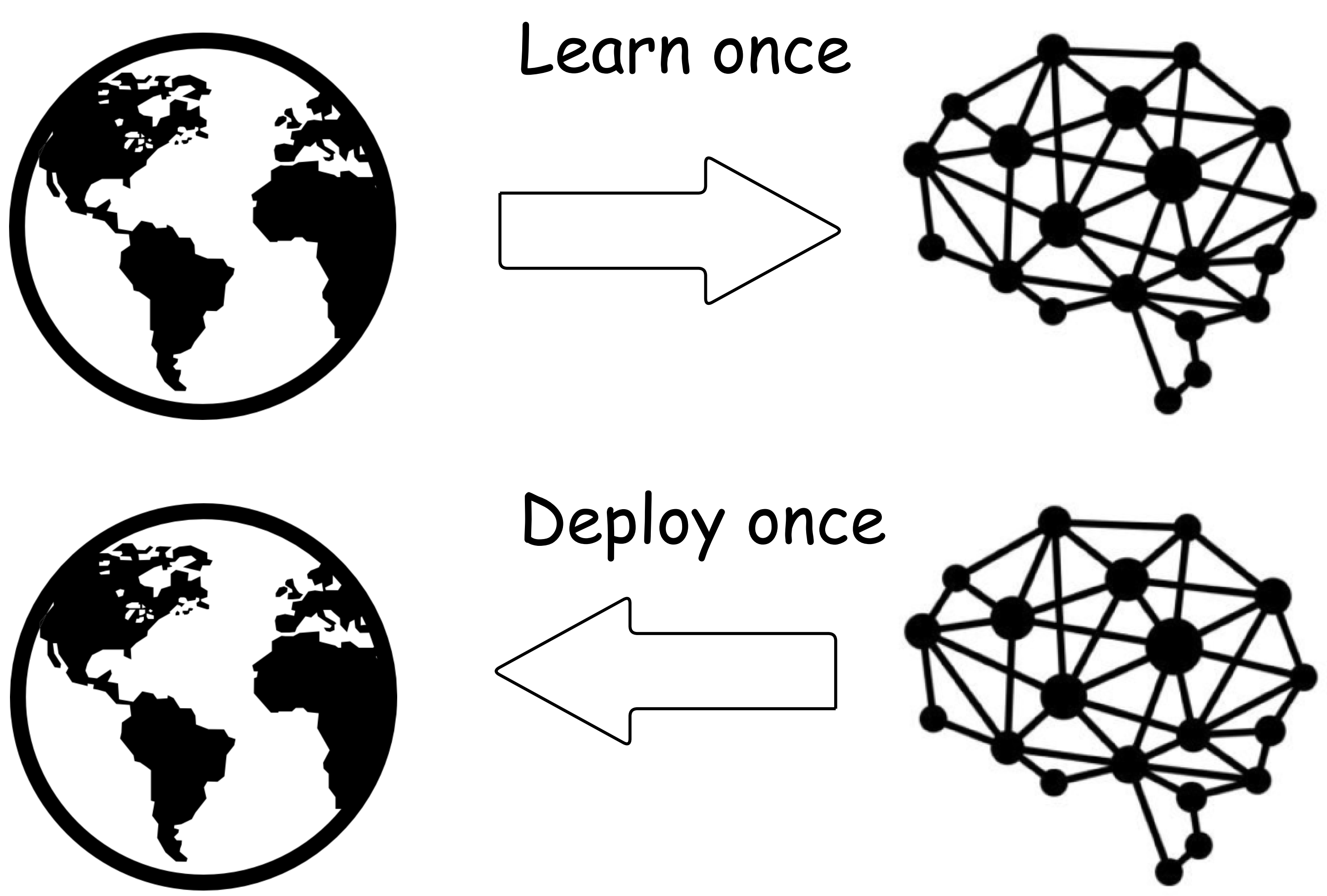
  e.g., Real-time human health monitoring and robotics



Realistic Scenarios

- Adaptive ML is essential
- Test-time adaptation (TTA) is a practical solution but challenging

Static ML



Test-Time Adaptation

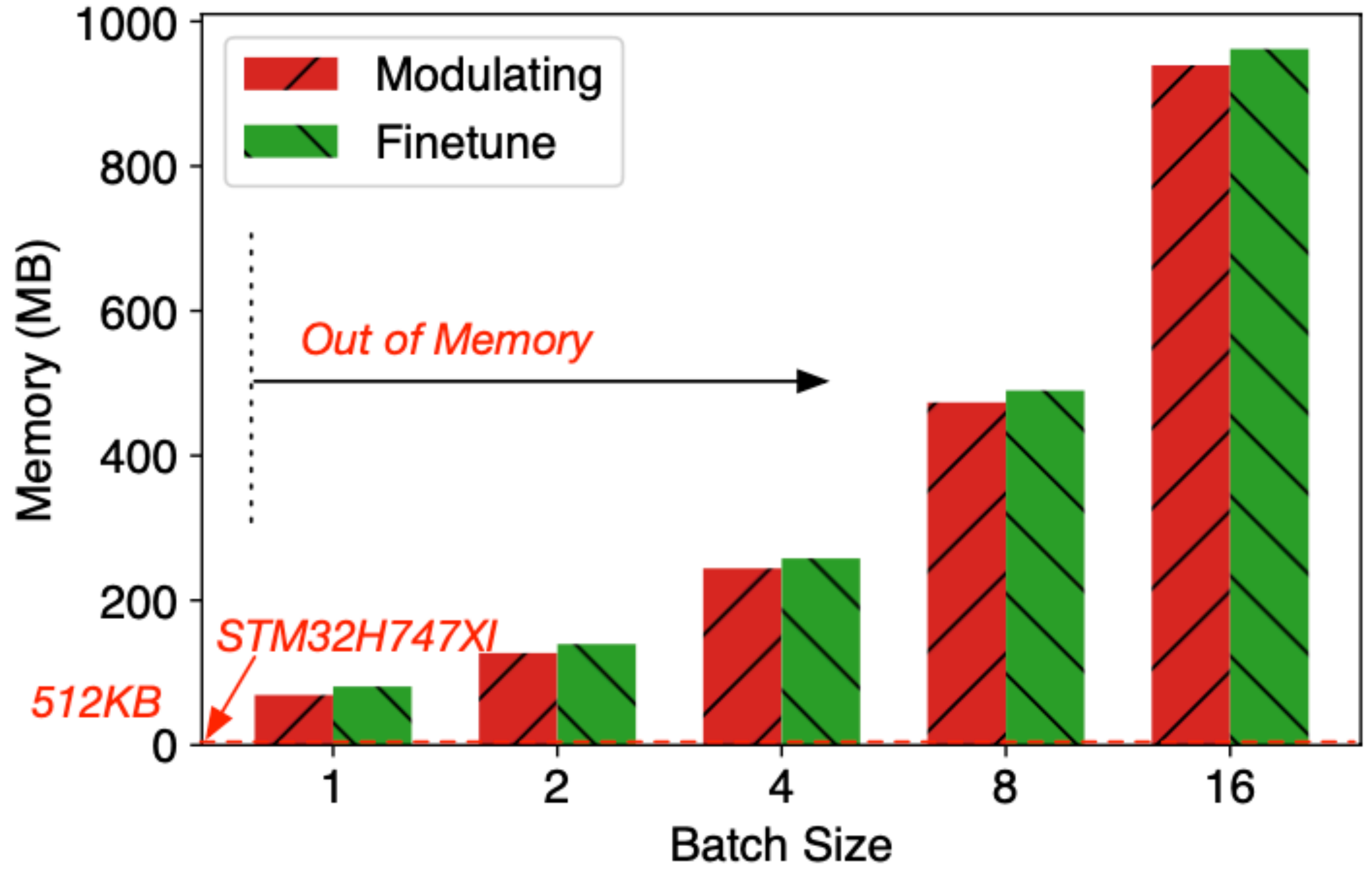
Unique Challenges of TTA on Edge Devices

1. No batch normalization layers are supported on MCUs

Unique Challenges of TTA on Edge Devices

1. No batch normalization layers are supported on MCUs

2. Adjust model parameters is expensive in terms of memory and computation



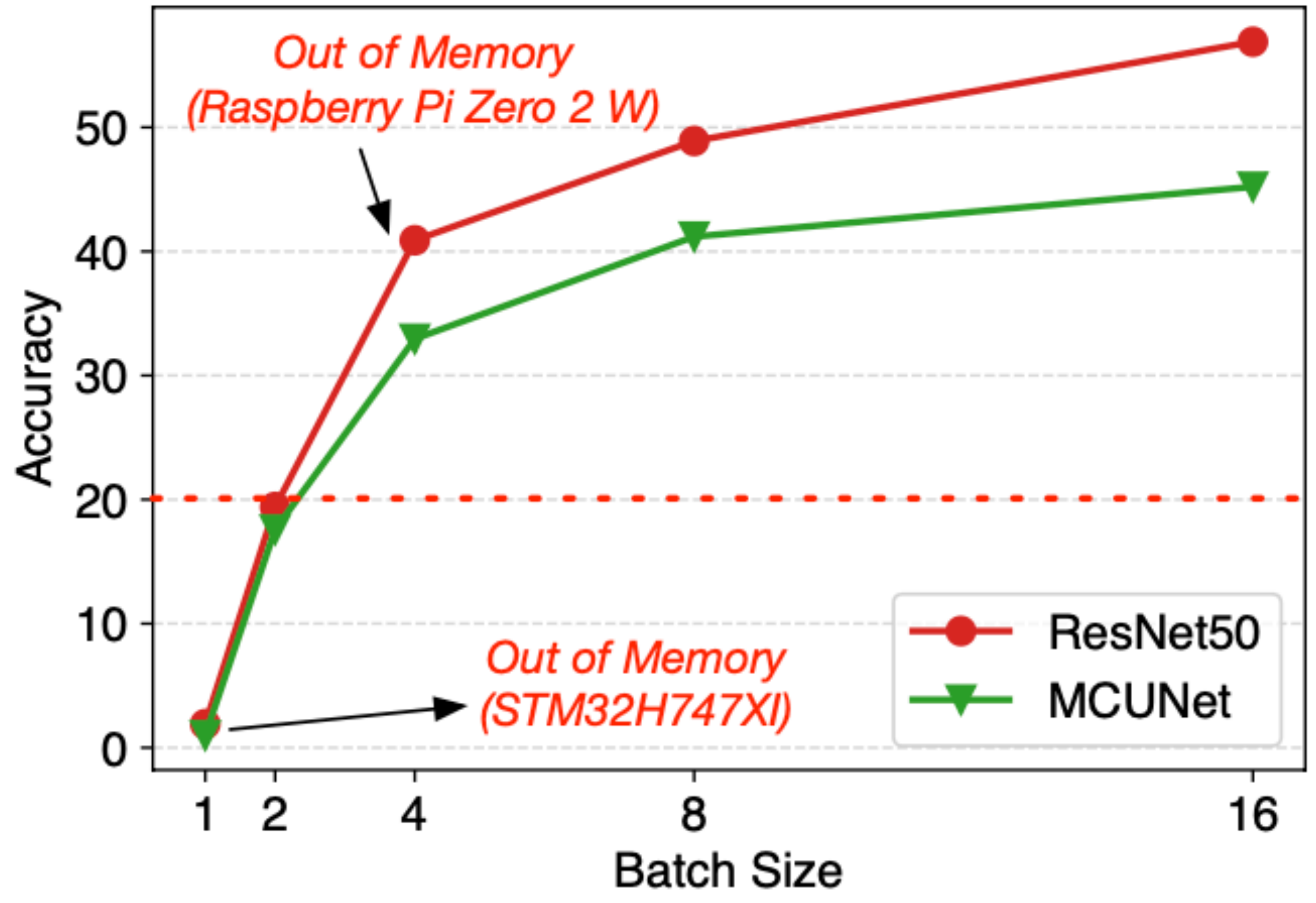
MCUNet

Unique Challenges of TTA on Edge Devices

1. No batch normalization layers are supported on MCUs

2. Adjust model parameters is expensive in terms of memory and computation

3. Poor performance with small batch size when computational resources are limited



Prior Works & Limitations

Finetune-based

- Update entire model

- Suffer from intensive memory usage

Prior Works & Limitations

Finetune-based	Modulating-based
Update entire model	Update normalization layers only and freeze other layers
Suffer from intensive memory usage	Suffer from intensive memory usage

Prior Works & Limitations

Finetune-based	Modulating-based	Memory-efficient TTA
• Update entire model	• Update normalization layers only and freeze other layers	• Update enabled with low memory on GPUs
• Suffer from intensive memory usage	• Suffer from intensive memory usage	• Remain memory intensive on CPUs

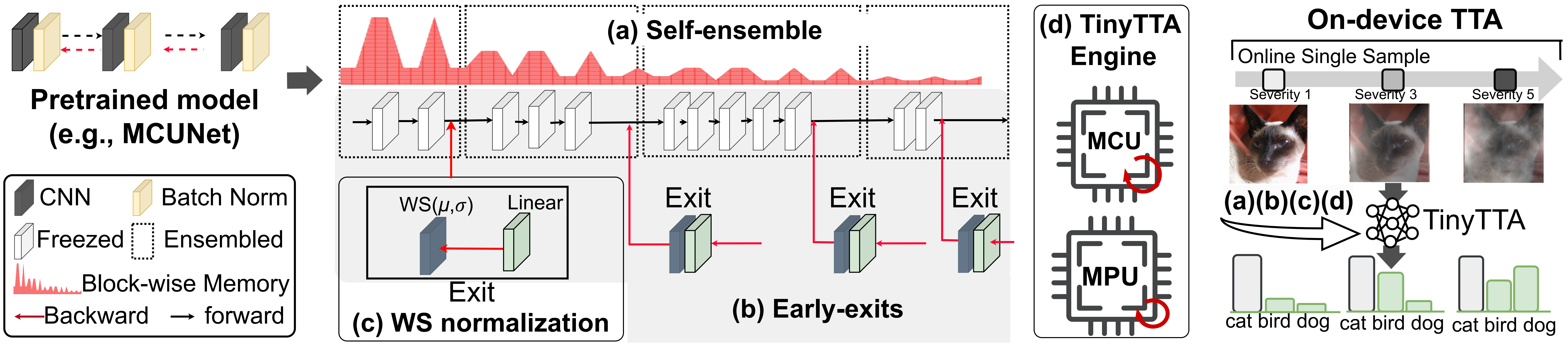
Prior Works & Limitations

Finetune-based	Modulating-based	Memory-efficient TTA
• Update entire model	• Update normalization layers only and freeze other layers	• Update enabled with low memory on GPUs
• Suffer from intensive memory usage	• Suffer from intensive memory usage	• Remain memory intensive on CPUs

- Model collapse with batch size of one
- Normalization layers are unavailable on MCUs

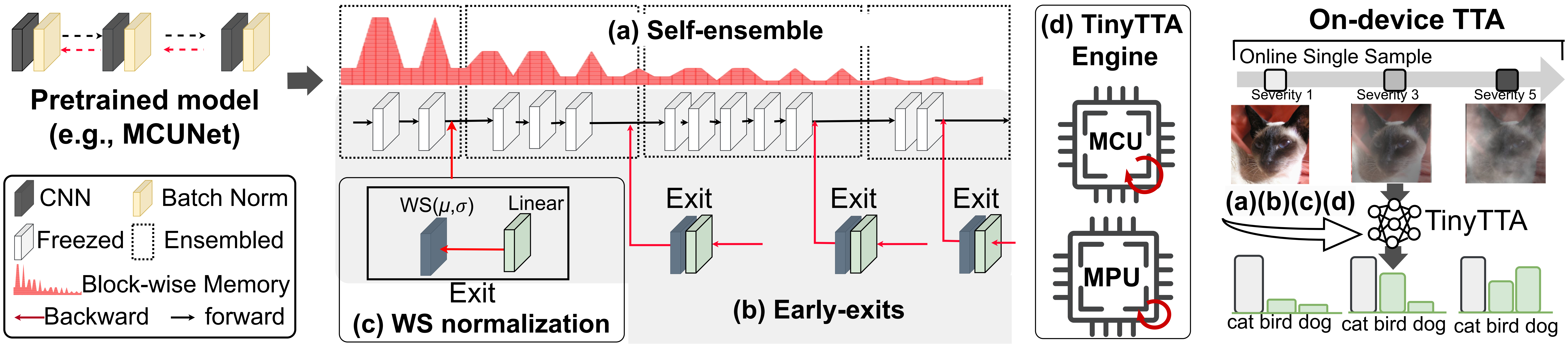
TinyTTA

- Efficient, batch-agnostic, and robust TTA on edge devices



TinyTTA

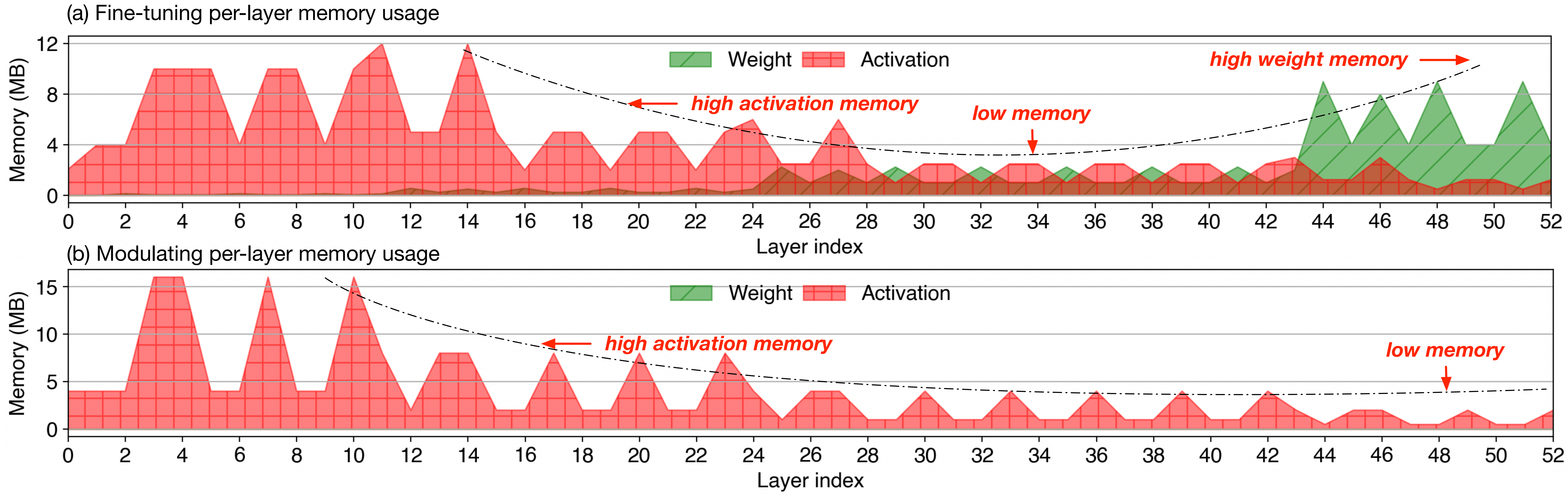
- Efficient, batch-agnostic, and robust TTA on edge devices



- Early-exit ensemble to co-optimize memory footprint and accuracy
- TinyTTA Engine to enable TTA on MCUs

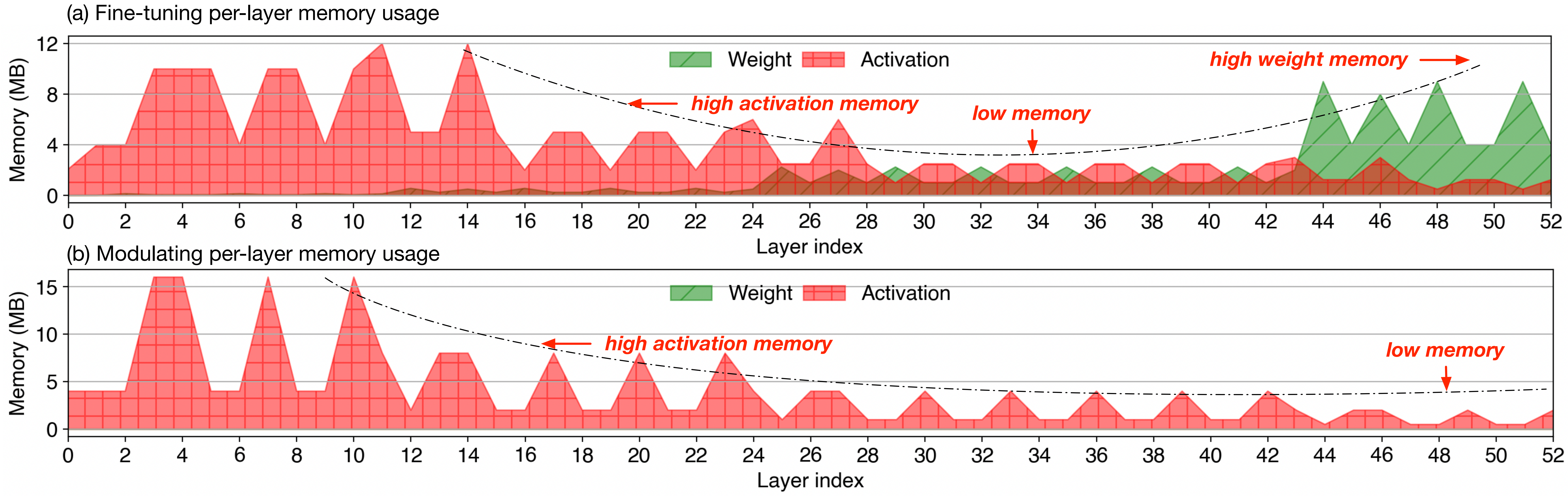
Batch-Agnostic Early-exit Ensembles

- Co-optimizes memory footprint and accuracy



Batch-Agnostic Early-exit Ensembles

- Co-optimizes memory footprint and accuracy



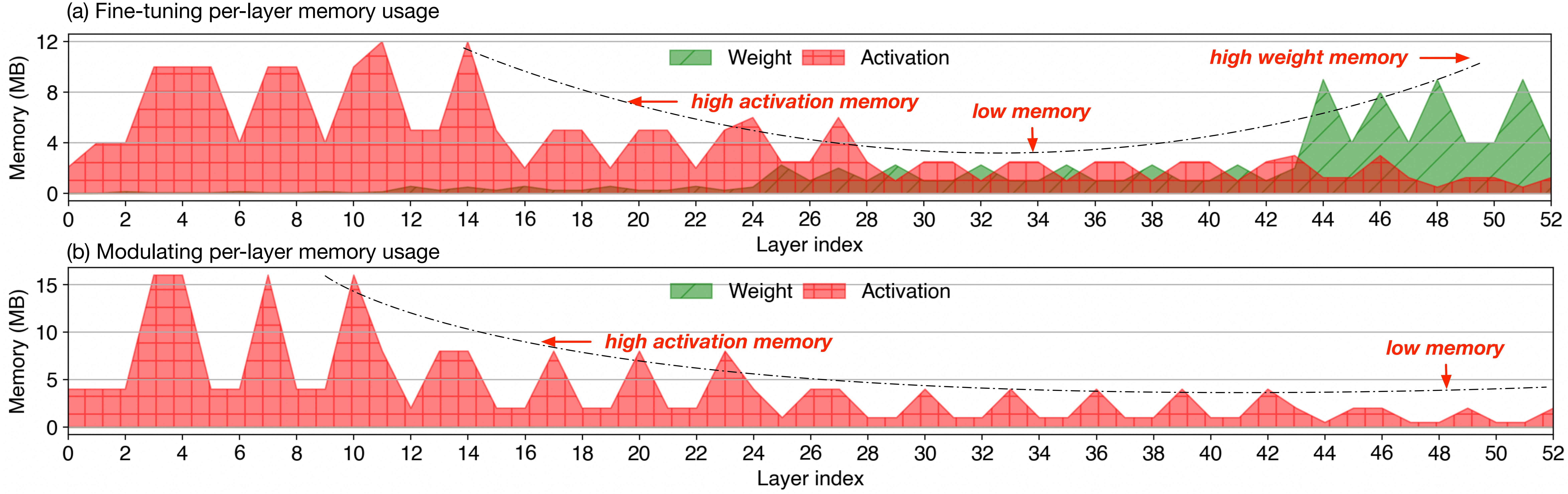
latent representation of submodules

submodule output

$$\boldsymbol{p}_i^k = \frac{\exp\left(\boldsymbol{z}_i^k\right)}{\sum_{j=1}^{C} \exp\left(\boldsymbol{z}_j^k\right)}$$

Batch-Agnostic Early-exit Ensembles

- Co-optimizes memory footprint and accuracy



submodule output

latent representation of submodules

$$\boldsymbol{p}_i^k = \frac{\exp\left(\boldsymbol{z}_i^k\right)}{\sum_{j=1}^{C} \exp\left(\boldsymbol{z}_j^k\right)}$$

Align submodule output

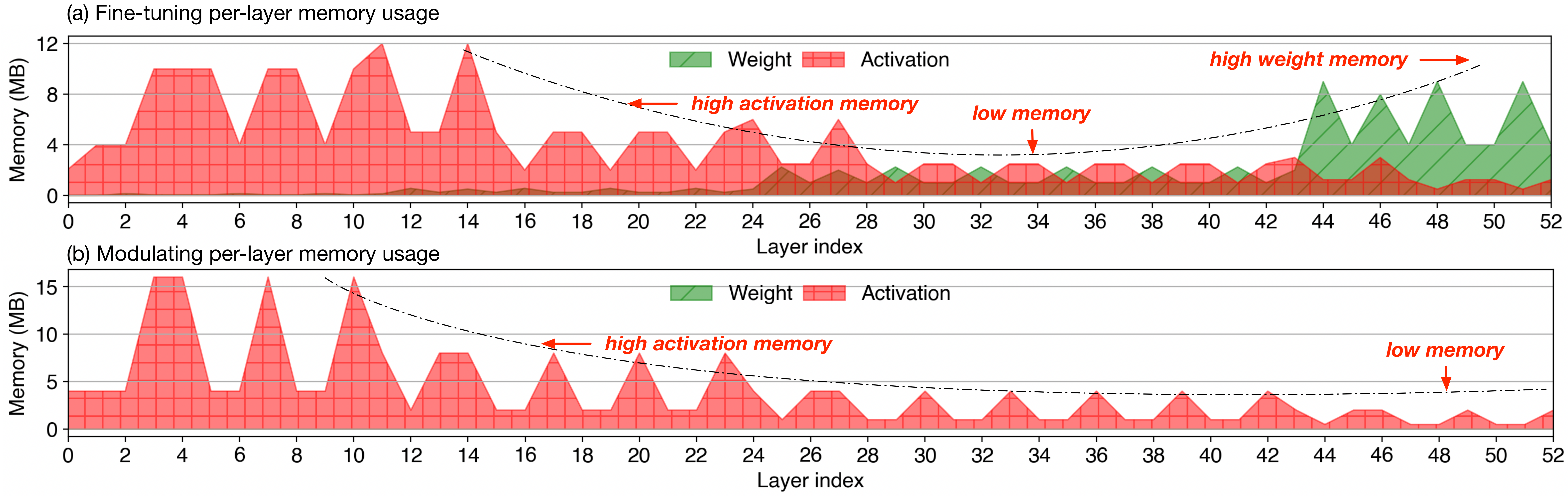
$$\mathcal{L}_1 = \sum_{i=1}^{C} CE\left(\boldsymbol{p}_i, y\right)$$

Align latent representations

$$\mathcal{L}_2 = \left\| \tilde{\boldsymbol{z}}_k - \boldsymbol{z}_k \right\|_1$$

Batch-Agnostic Early-exit Ensembles

- Co-optimizes memory footprint and accuracy



submodule output

latent representation of submodules

$$\boldsymbol{p}_i^k = \frac{\exp\left(\boldsymbol{z}_i^k\right)}{\sum_{j=1}^{C} \exp\left(\boldsymbol{z}_j^k\right)}$$

Align submodule output

$$\mathcal{L}_1 = \sum_{i=1}^{C} CE\left(\boldsymbol{p}_i, y\right)$$

Align latent representations

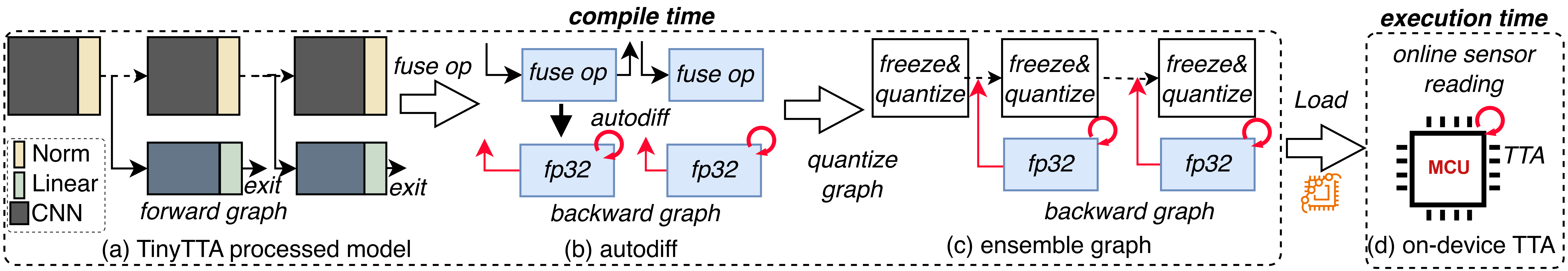
$$\mathcal{L}_2 = \left\| \tilde{\boldsymbol{z}}_k - \boldsymbol{z}_k \right\|_1$$

Weight standar-dization exits

$$\widetilde{\boldsymbol{W}} = \frac{\boldsymbol{W} - \boldsymbol{\mu}_w}{\boldsymbol{\sigma}_w + \epsilon}$$

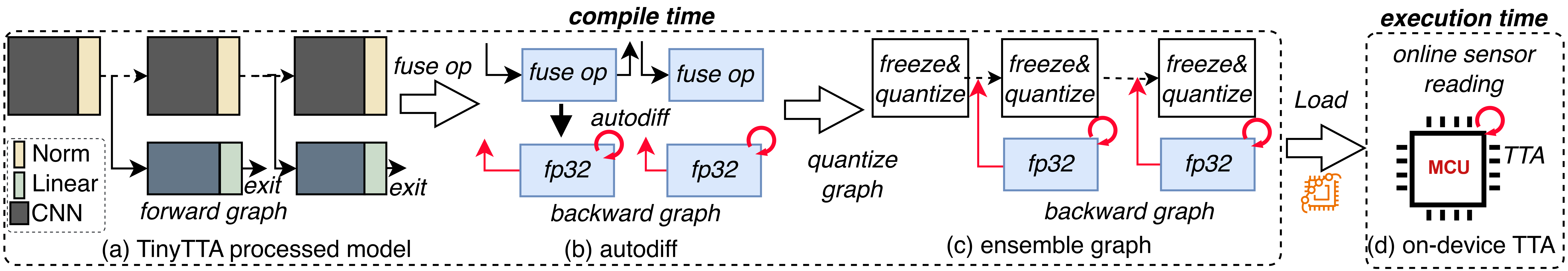
TinyTTA Engine

- First-of-its-kind TTA engine on MCUs
- Optimized to mitigate resource limitations during TTA



TinyTTA Engine

- First-of-its-kind TTA engine on MCUs
- Optimized to mitigate resource limitations during TTA



- BP operators support for Tensorflow Lite Micro
- Layer-wise update strategy to optimize memory efficiency

Experimental Setup

- **Datasets**

(1) CIFAR10C
(2) CIFAR100C
(3) OfficeHome
(4) PACS

- **Baselines**

(1) Tent (Modulating)
(2) Tent (Finetune)
(3) EATA
(4) CoTTA
(5) EcoTTA

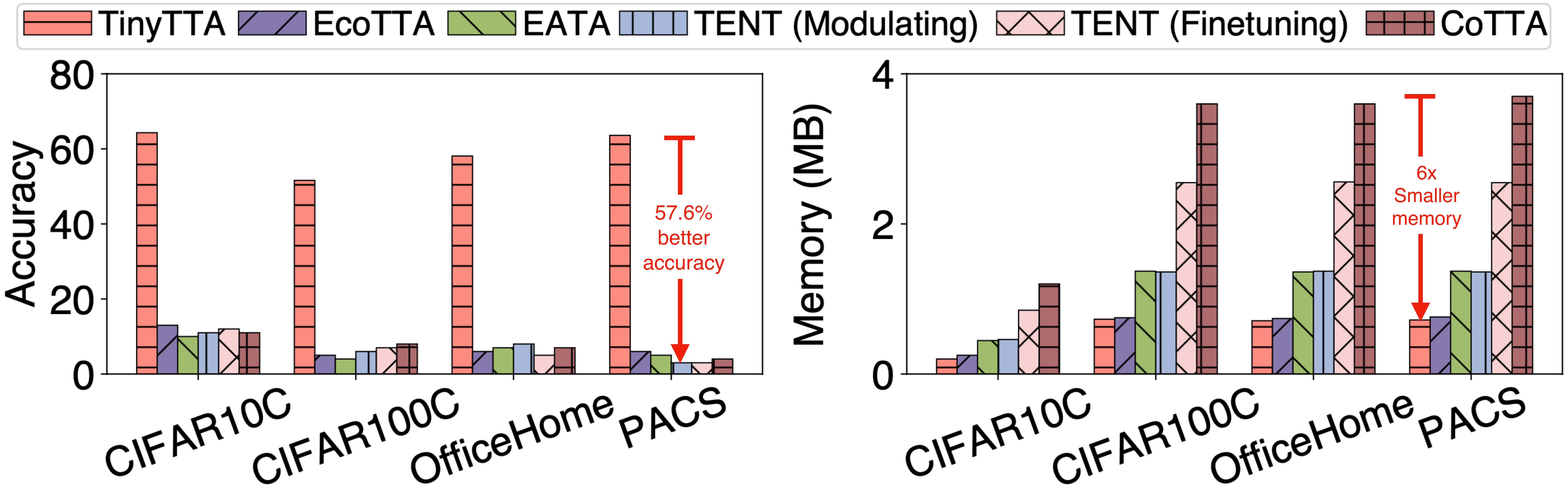
- **Architectures**

(1) MCUNet
(2) MobileNetV2_×05
(3) EfficientNet_b1
(4) RegNet-200m

- **Hardwares**

(1) MCU: STM32H747
(2) MPU: RaspberryPi Zero 2 W

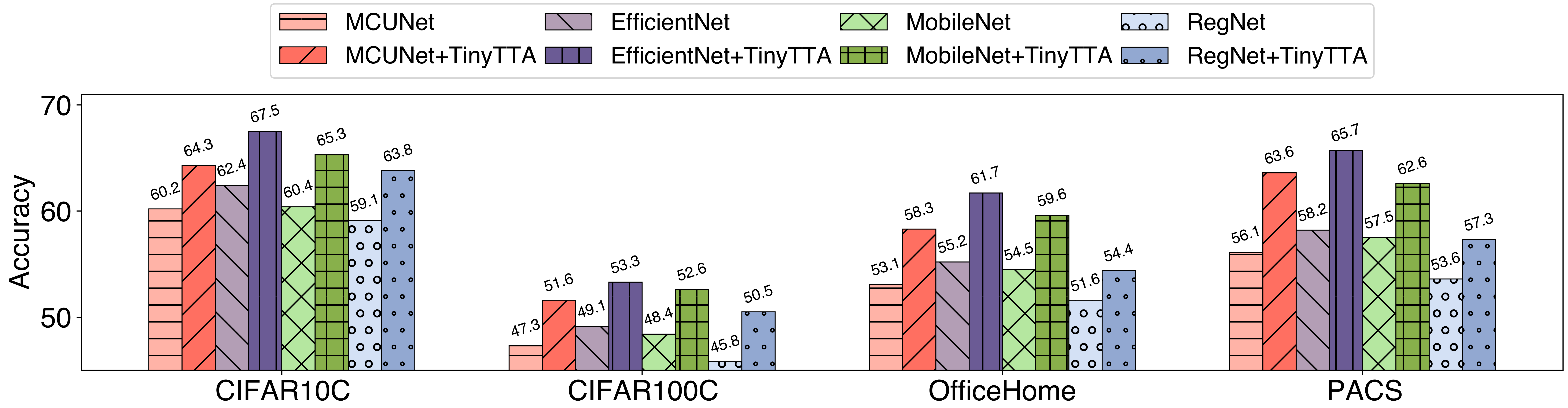
Results



TinyTTA achieves up to 57.6% higher accuracy compared to TENT (Modulating) with a batch size of one

TinyTTA achieves up to 6x lower memory usage compared to CoTTA with a batch size of one

Results



TinyTTA achieves an average of 4.3% higher accuracy compared to a model without update with a batch size of one

TinyTTA is the only framework capable of performing TTA under an MCU's 512 KB memory constraint

Table 2: MCU deployment of the baseline and TinyTTA on STM32H747 using MCUNet and CIFAR10C.

System	Accuracy	SRAM	Flash	Latency	Energy
Inference Only	60.2%	**82.8KB**	**290KB**	55.8ms	12.7mJ
TinyTTA (update)	**64.3%**	123KB	375KB	**50.7ms**	**11.5mJ**

Summary & Take-away Messages

S1. TinyTTA enables efficient, batch-agnostic and robust on-device TTA for the first time

T1. Self-ensemble framework and early-exit policy is effective in ensuring high TTA accuracy

T2. TinyTTA Engine enables TTA for diverse MCU applications

Thank you!

Any questions?
You can find me at:
hong.jia@unimelb.edu.au
h-jia.github.io

UNIVERSITY OF CAMBRIDGE

SAMSUNG Research

THE UNIVERSITY OF MELBOURNE

UNIVERSITÀ DELLA CALABRIA