MIT HAN LAB

Lecture 11
MCUNet: Tiny Neural Network Design for Microcontrollers

Song Han
songhan@mit.edu

Lecture Plan

Today we will introduce the following:

1. What is tinyML?
2. Understanding the challenges of tinyML
3. Tiny neural network design
4. Applications
    1. Tiny vision
    2. Tiny audio
    3. Tiny time series/anomaly detection

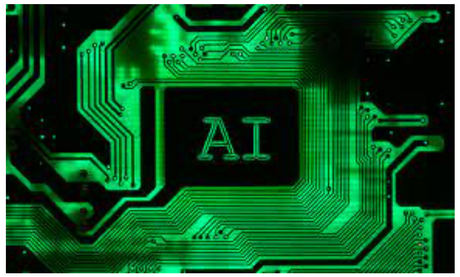
Lecture Plan

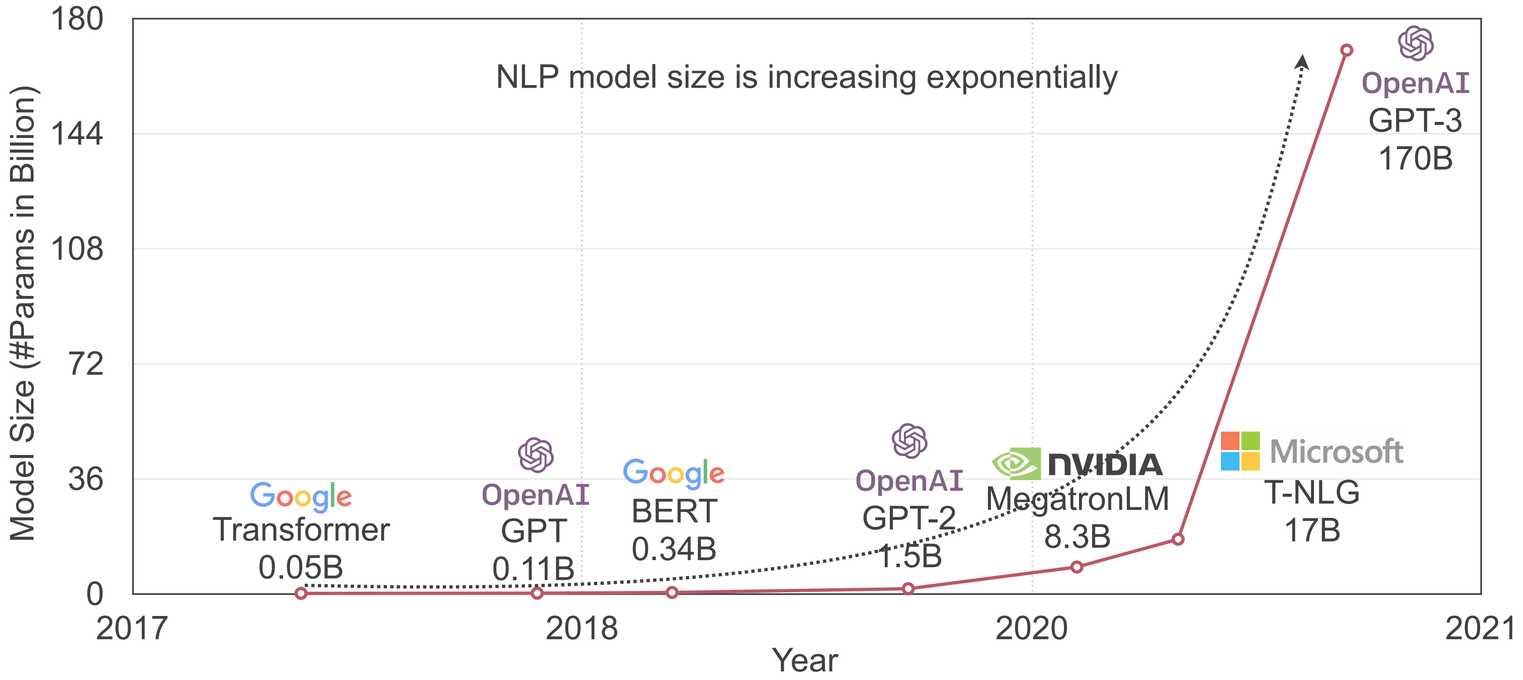
Today we will introduce the following:

1. **What is tinyML?**
2. Understanding the challenges of tinyML
3. Tiny neural network design
4. Applications
    1. Tiny vision
    2. Tiny audio
    3. Tiny time series/anomaly detection

Today's AI is too Big

We need new algorithms and hardware for TinyML and Green AI
Low Energy, Low Latency, Low Cost, Better Privacy



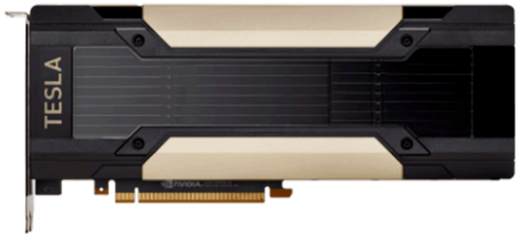


GPT-3: 175 billion parameters, 355 GPU years to train and cost $4.6M

AlphaGo: 1920 CPUs and 280 GPUs, $3000 per game for electric bill

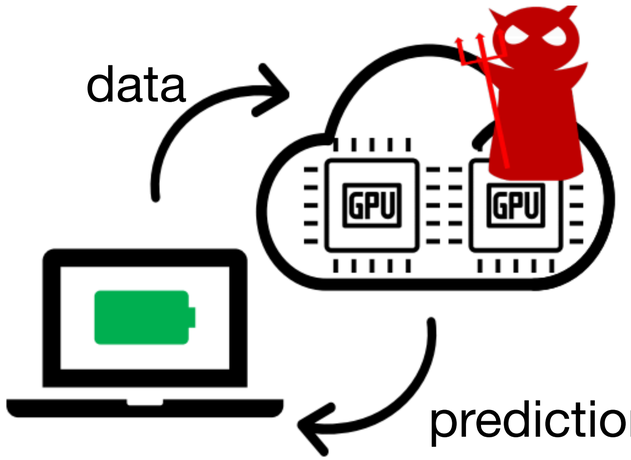
Deep Learning Going “Tiny”

Cloud → Mobile → Tiny



Cloud AI

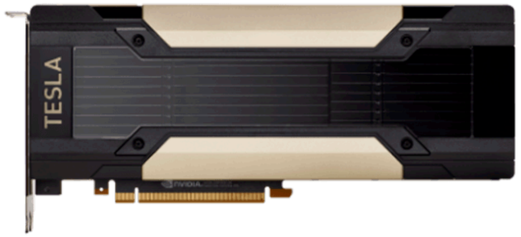
GPUs/TPUs

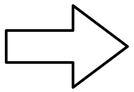


- Data uploaded to the cloud for inference

Deep Learning Going “Tiny”

Cloud → Mobile → Tiny





Cloud AI

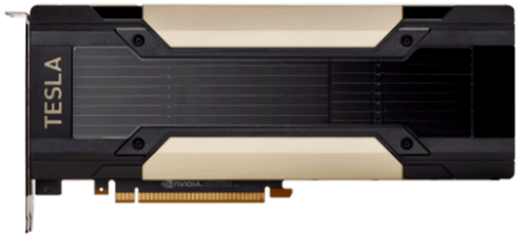
GPUs/TPUs

Mobile AI

Smartphones

Deep Learning Going “Tiny”

Cloud → Mobile → Tiny



?

Cloud AI

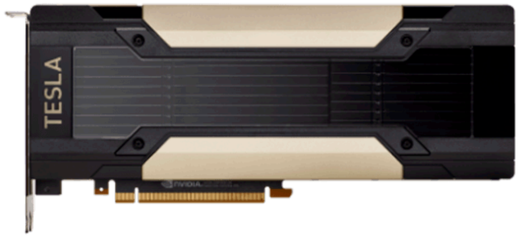
GPUs/TPUs

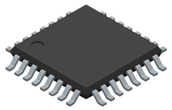
Mobile AI

Smartphones

Deep Learning Going “Tiny”

Cloud → Mobile → Tiny





Cloud AI

GPUs/TPUs

Mobile AI

Smartphones

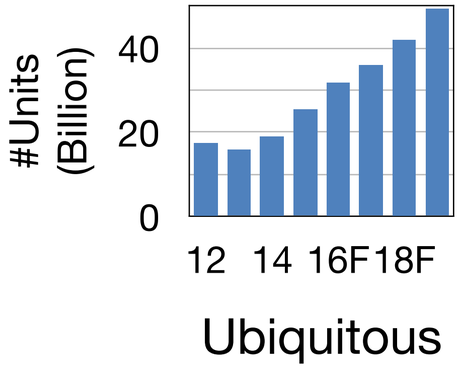
Tiny AI

IoT/Microcontrollers

Deep Learning Going “Tiny”

Squeezing deep learning into IoT devices

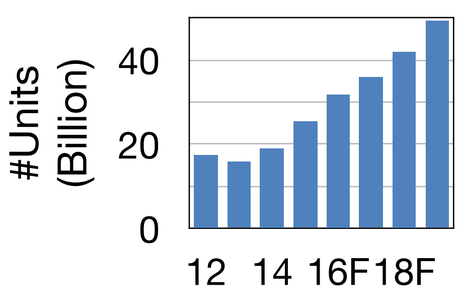
- Billions of IoT devices around the world based on **microcontrollers**



Deep Learning Going “Tiny”

Squeezing deep learning into IoT devices

- Billions of IoT devices around the world based on **microcontrollers**
- **Low-cost**: low-income people can afford access. Democratize AI.



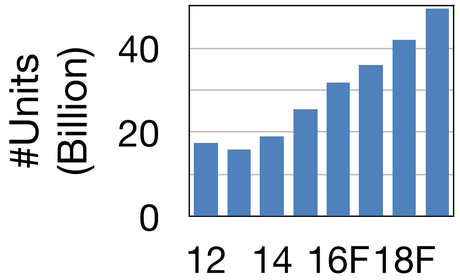
Ubiquitous

Low-cost
($0.1 - $10)

Deep Learning Going "Tiny"

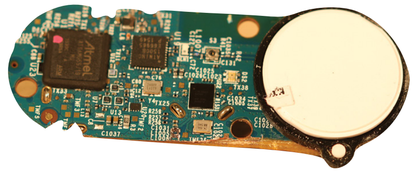
Squeezing deep learning into IoT devices

- Billions of IoT devices around the world based on **microcontrollers**
- **Low-cost**: low-income people can afford access. Democratize AI.
- **Low-power**: green AI, reduce carbon



Ubiquitous

Low-cost
($0.1 - $10)



Low-power
(mW)

Deep Learning Going "Tiny"

Squeezing deep learning into IoT devices

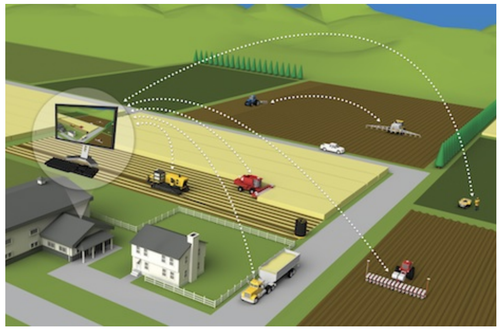
- Billions of IoT devices around the world based on **microcontrollers**
- **Low-cost**: low-income people can afford access. Democratize AI.
- **Low-power**: green AI, reduce carbon
- Various applications

Smart Home

Smart Manufacturing

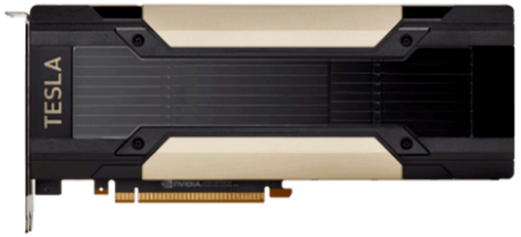
Personalized Healthcare

Precision Agriculture



TinyML is Challenging

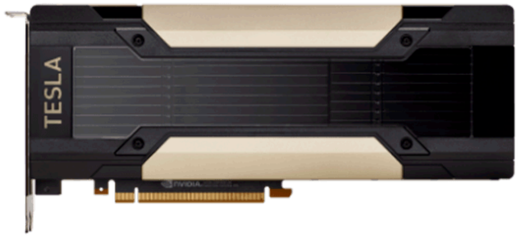
Memory size is too small to hold DNNs



	Cloud AI	Mobile AI
Memory (Activation)	32GB	4GB
Storage (Weights)	~TB/PB	256GB

TinyML is Challenging

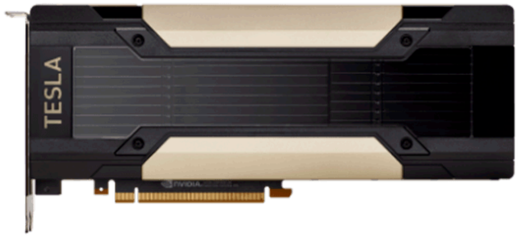
Memory size is too small to hold DNNs



	Cloud AI	Mobile AI	Tiny AI
Memory (Activation)	32GB	4GB	320kB
Storage (Weights)	~TB/PB	256GB	1MB

TinyML is Challenging

Memory size is too small to hold DNNs



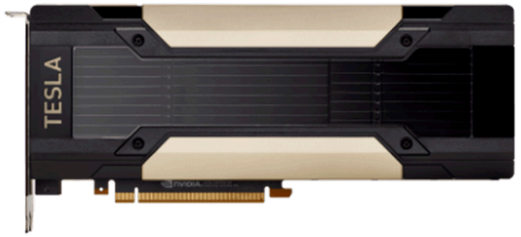
	Cloud AI	Mobile AI	Tiny AI
Memory (Activation)	32GB	4GB	320kB
Storage (Weights)	~TB/PB	256GB	1MB

13,000x smaller

100,000x smaller

TinyML is Challenging

Memory size is too small to hold DNNs



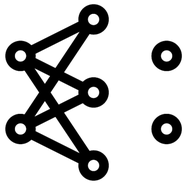
	Cloud AI	Mobile AI	Tiny AI
Memory (Activation)	32GB	4GB	320kB
Storage (Weights)	~TB/PB	256GB	1MB

- We need to reduce both **weights** and **activation** to fit DNNs for tinyML

TinyML is Challenging

Memory size is too small to hold DNNs

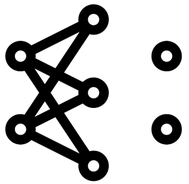
+



Latency Constraint

Energy Constraint

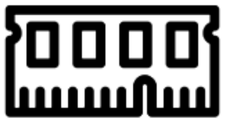
+



Latency Constraint

Energy Constraint

Memory Constraint



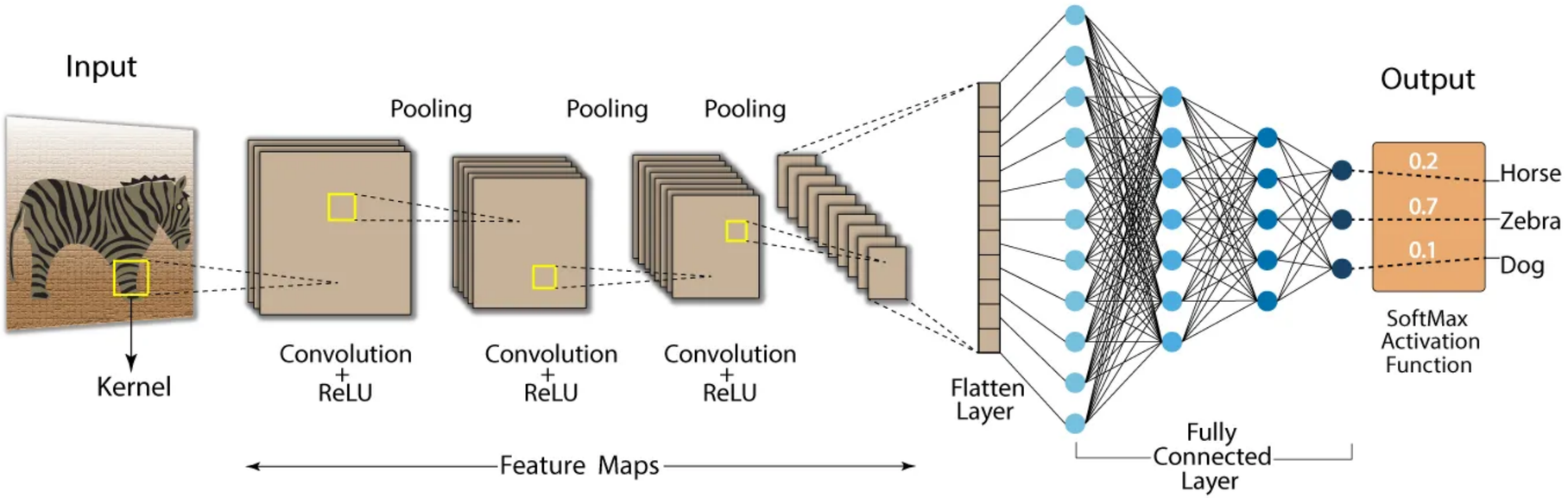
Different from Mobile AI

Lecture Plan

Today we will introduce the following:

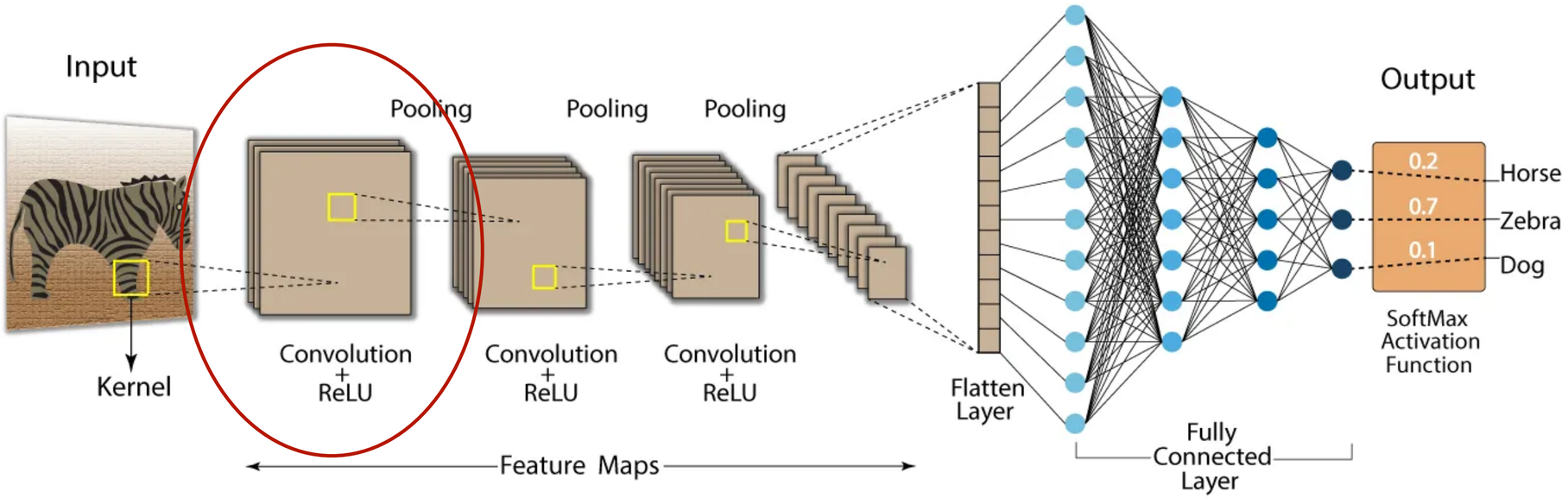
1. What is tinyML?
2. **Understanding the challenges of tinyML**
3. Tiny neural network design
4. Applications
    1. Tiny vision
    2. Tiny audio
    3. Tiny time series/anomaly detection

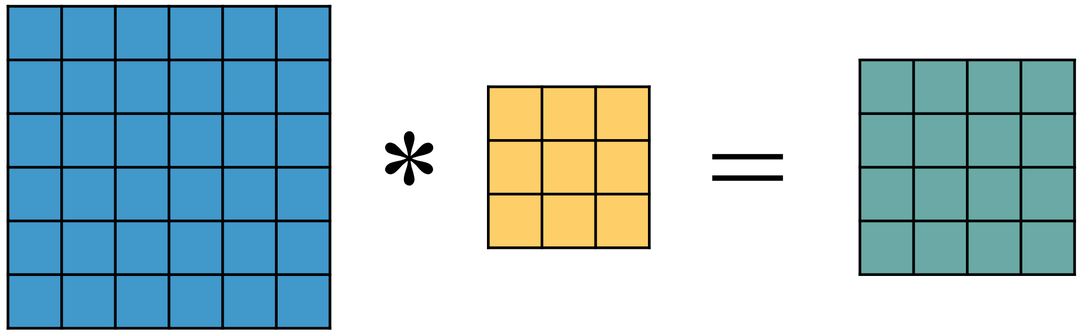
Running CNNs on Microcontrollers



* Image source: https://datamahadev.com/comprehensive-understanding-of-convolutional-neural-networks/

Running CNNs on Microcontrollers

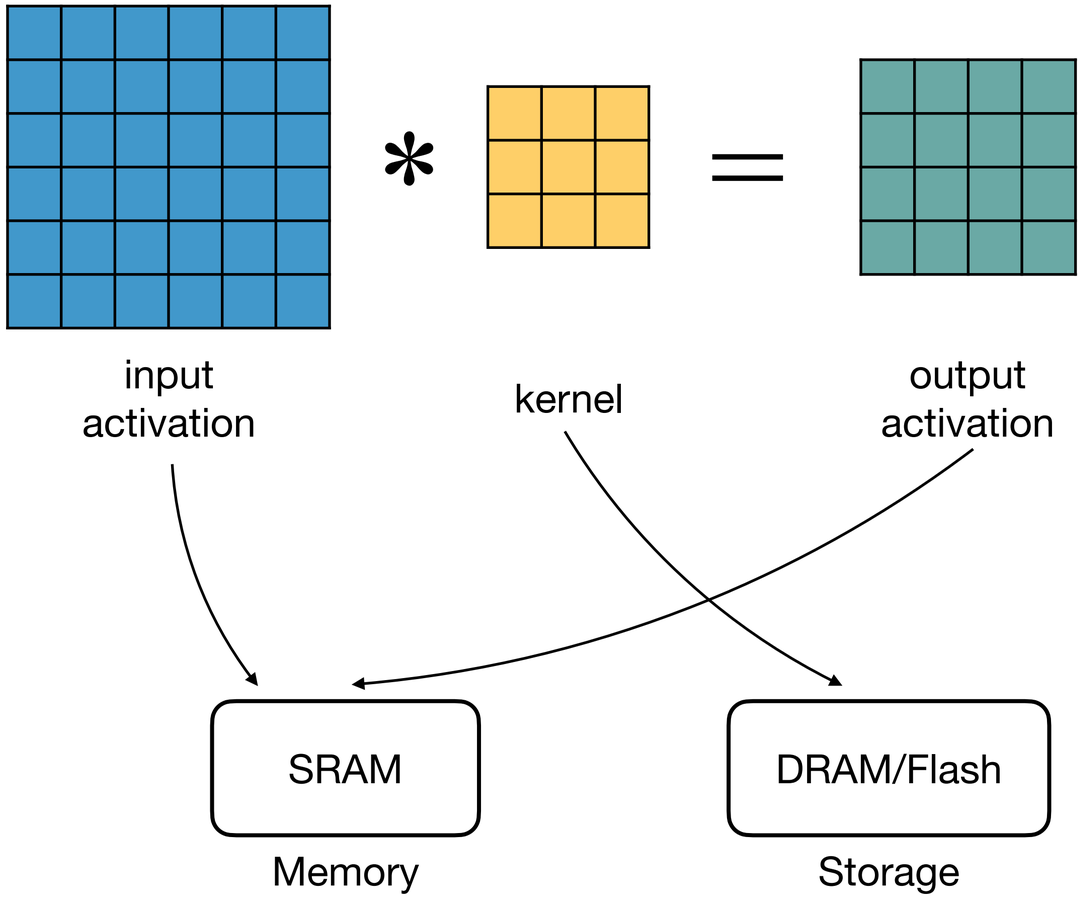




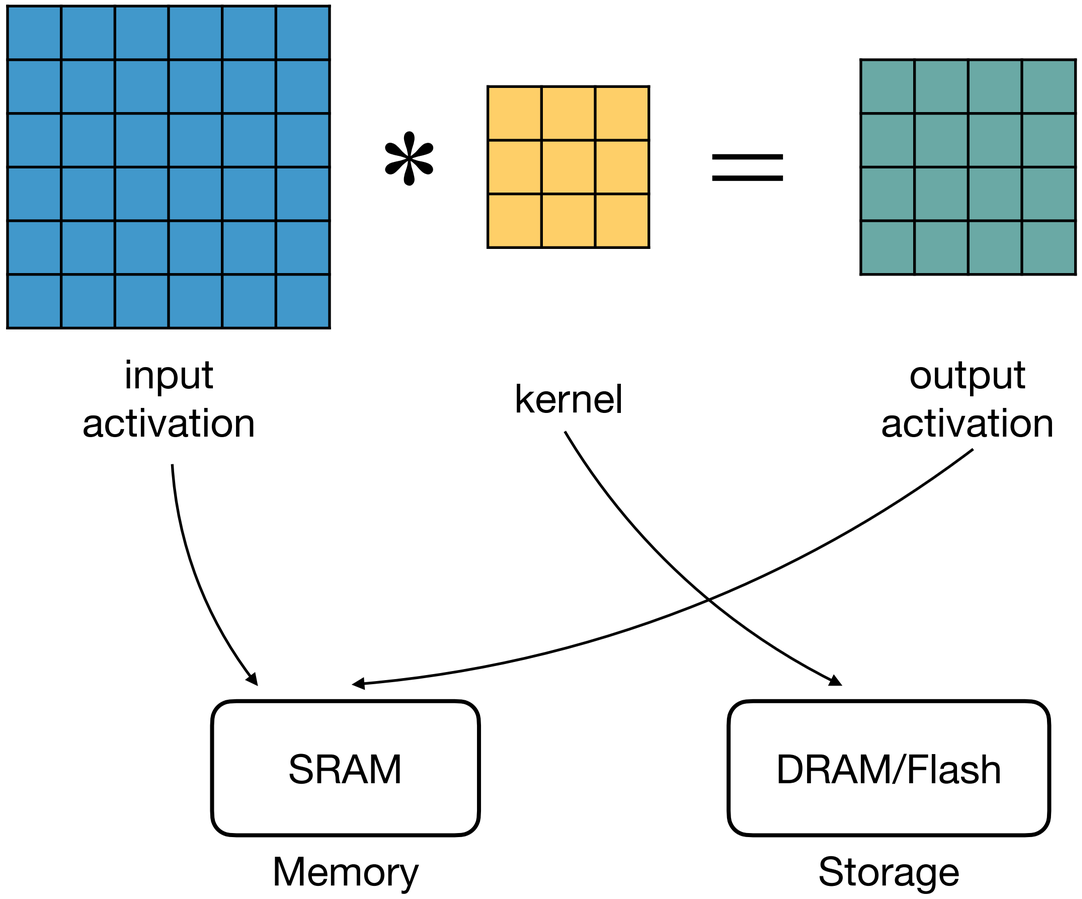
input activation | kernel | output activation

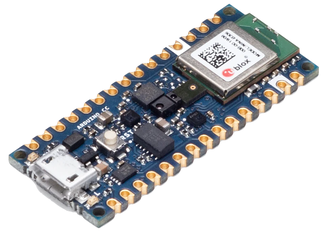
* Image source: https://datamahadev.com/comprehensive-understanding-of-convolutional-neural-networks/

Running CNNs on Microcontrollers

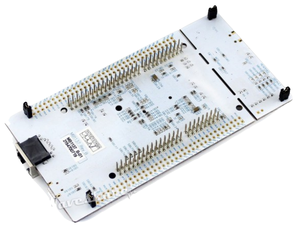


Running CNNs on Microcontrollers





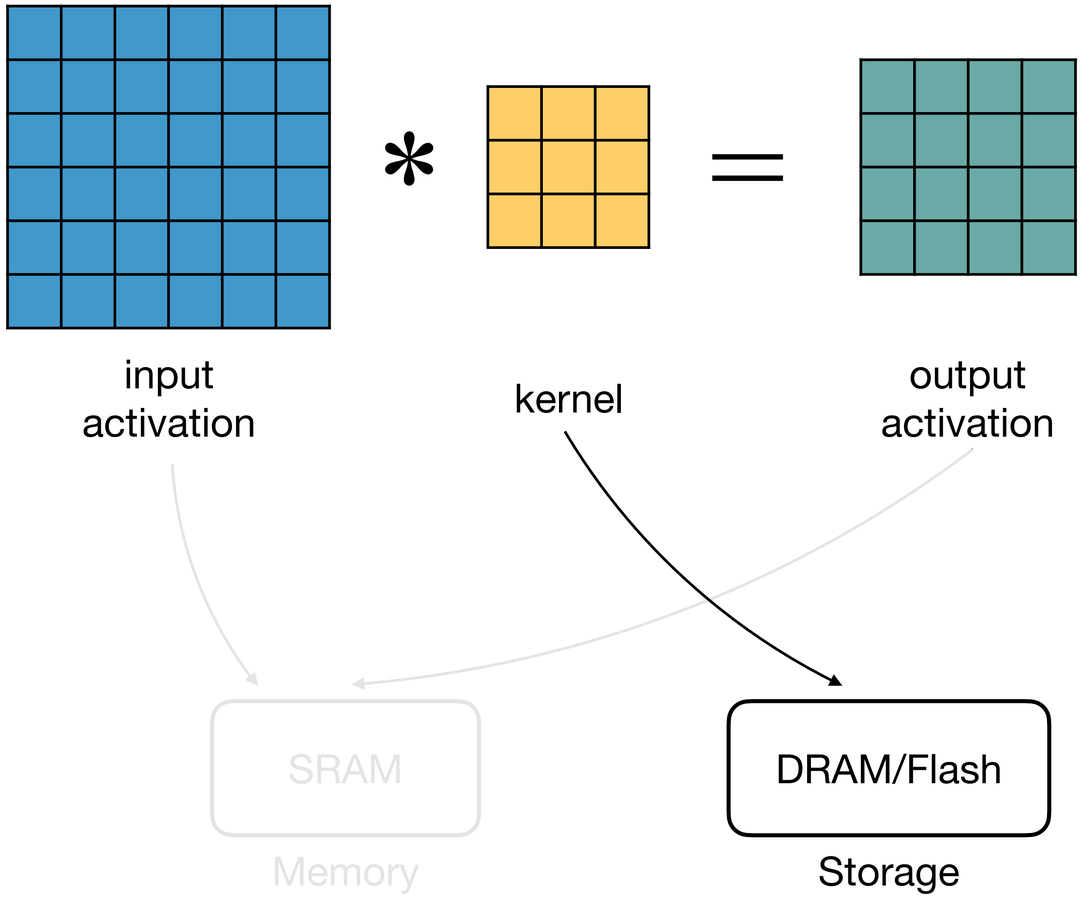
Arduino Nano 33 BLE Sense
- SRAM: 256KB
- Flash: 1MB



STM32 F746ZG
- SRAM: 320KB
- Flash: 1MB

Running CNNs on Microcontrollers

A simplified* method to estimate memory usage

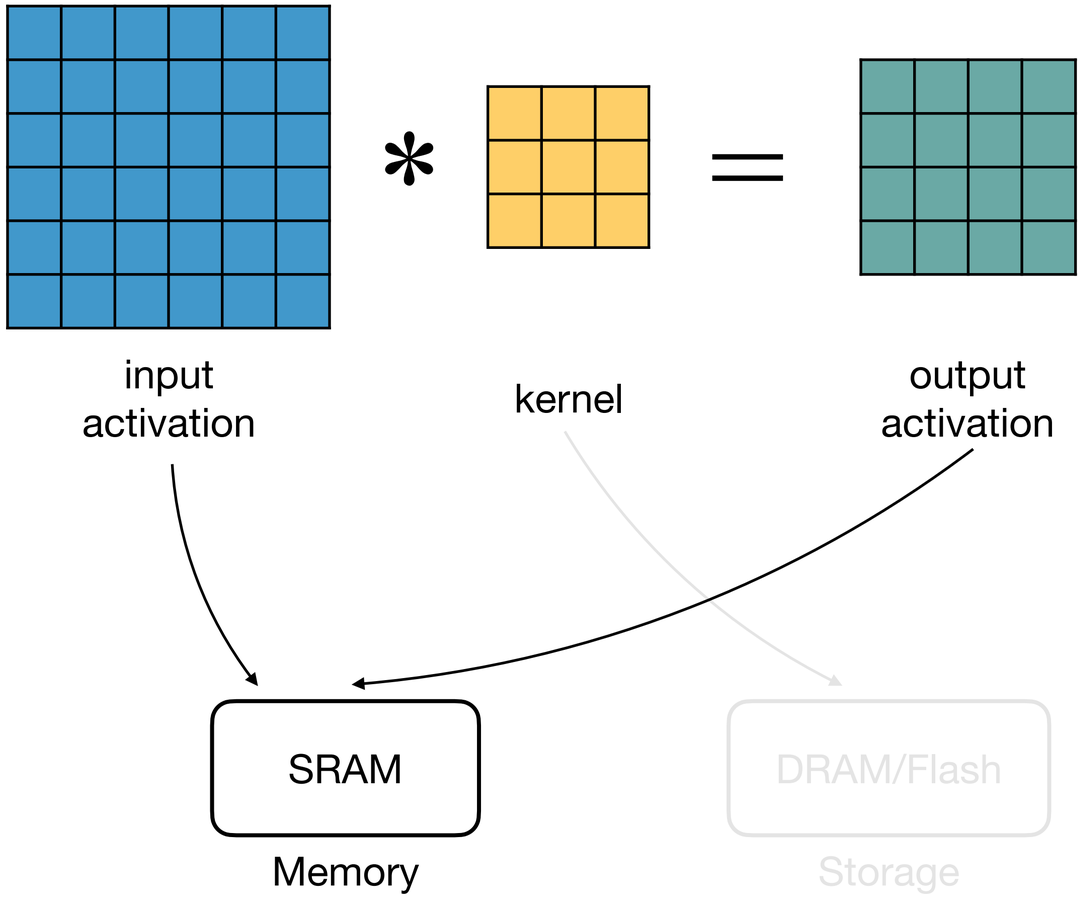


- **Flash** Usage
  - **= model size**
  - **Static**, need to hold the entire model

* Does not consider temporary buffers for SRAM, binary code size, etc. for now. Will discuss later.

Running CNNs on Microcontrollers

A simplified* method to estimate memory usage

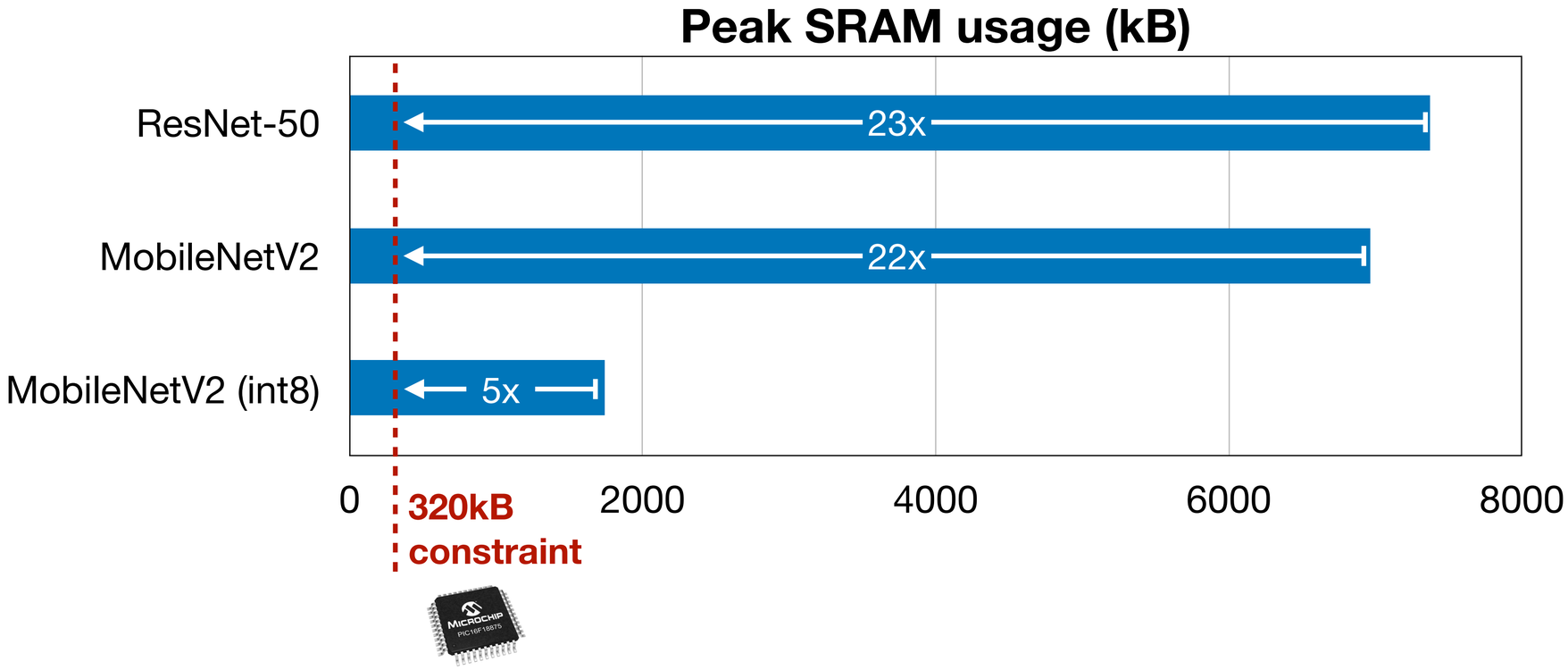


- **Flash** Usage
  - **= model size**
  - **Static**, need to hold the entire model
- **SRAM** usage
  - **= Input activation + output activation**
  - Dynamic, different for each layer
  - We care about peak SRAM
  - (Weights are not counted since they can be partially fetched)

* Does not consider temporary buffers for SRAM, binary code size, etc. for now. Will discuss later.

Today's CNNs are Too Big for TinyML

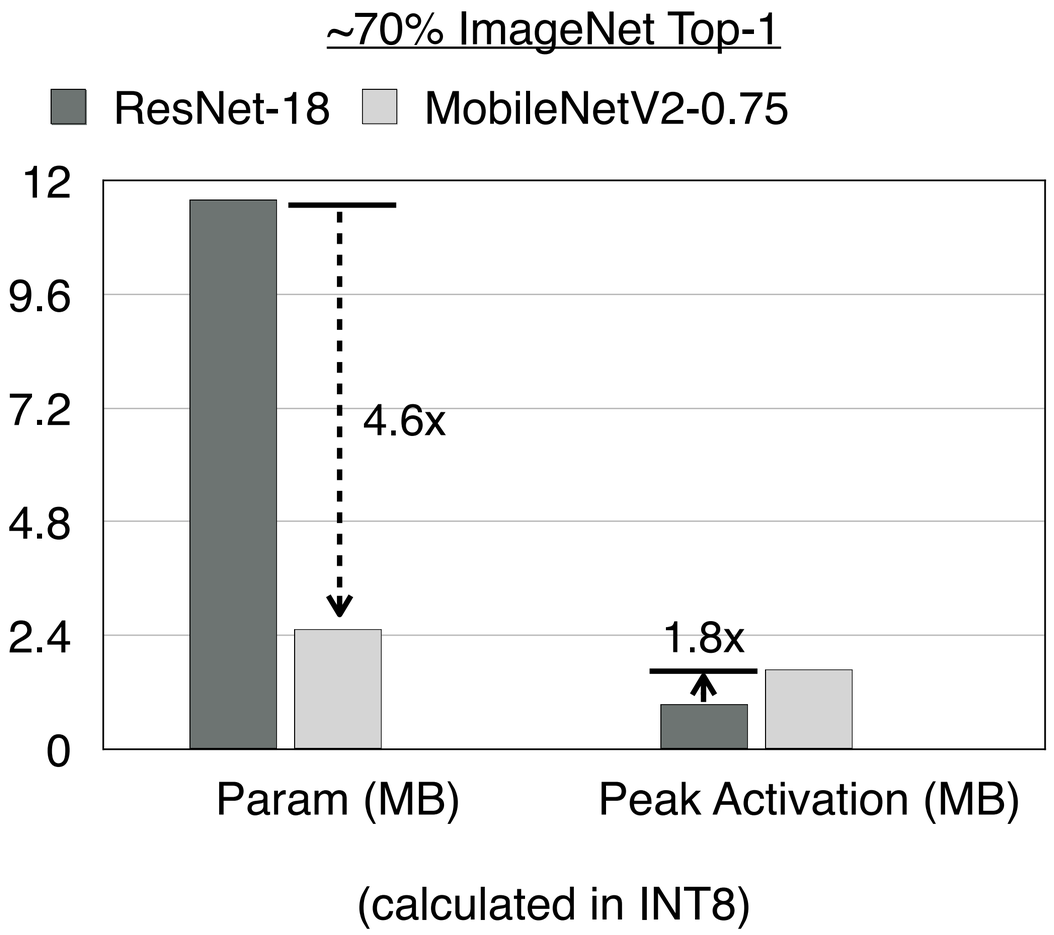
Cloud/Mobile CNNs cannot fit tinyML



Today's CNNs are Too Big for TinyML

We need to reduce not only model size, but also activation size

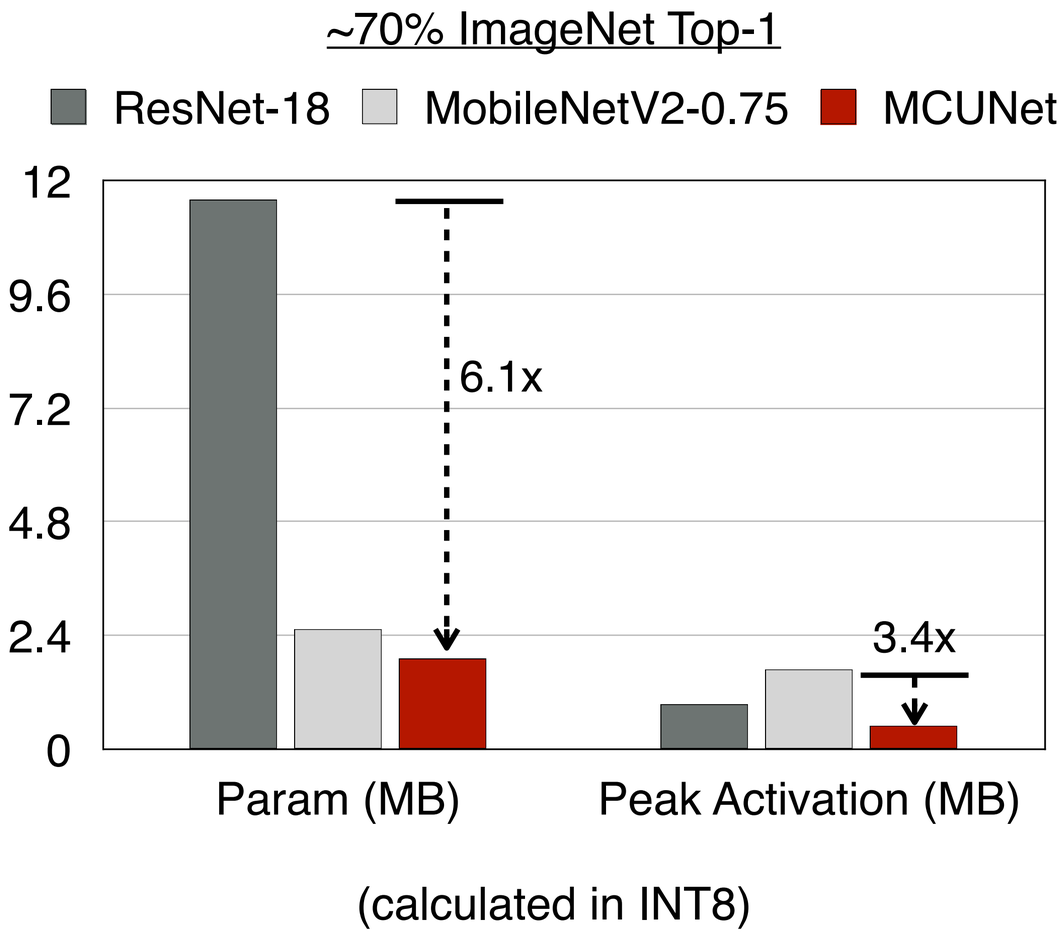
- MobileNetV2 reduces only model size but **not** peak activation size



Today's CNNs are Too Big for TinyML

We need to reduce not only model size, but also activation size

- MobileNetV2 reduces only model size but **not** peak activation size
- MCUNet reduces not only model size but also activation size



Lecture Plan

Today we will introduce the following:

1. What is tinyML?
2. Understanding the challenges of tinyML
3. **Tiny neural network design**
4. Applications
    1. Tiny vision
    2. Tiny audio
    3. Tiny time series/anomaly detection

MCUNet: System-Algorithm Co-design

MIT News
ON CAMPUS AND AROUND THE WORLD

SUBSCRIBE | SEARCH NEWS

System brings deep learning to "internet of things" devices

Advance could enable artificial intelligence on household appliances while enhancing data security and energy efficiency.

Watch Video

Daniel Ackerman | MIT News Office
November 13, 2020

PRESS INQUIRIES

MIT researchers have developed a system, called MCUNet, that brings machine learning to microcontrollers. The advance could enhance the function and security of devices connected to the Internet of Things (IoT).

MIT News
ON CAMPUS AND AROUND THE WORLD

SUBSCRIBE | SEARCH NEWS

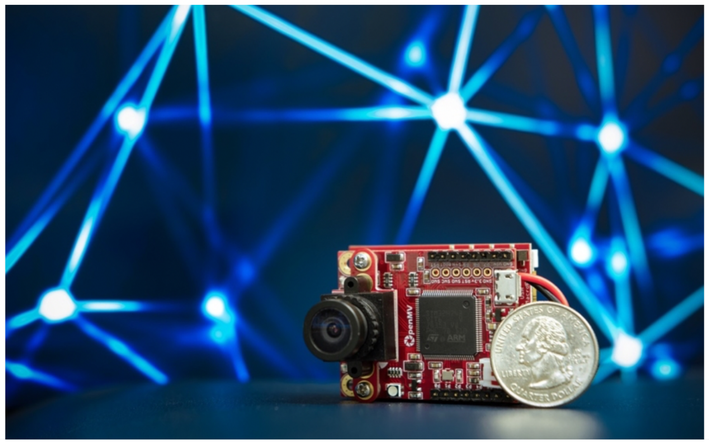
Tiny machine learning design alleviates a bottleneck in memory usage on internet-of-things devices

New technique applied to small computer chips enables efficient vision and detection algorithms without internet connectivity.

Watch Video

Lauren Hinkel | MIT-IBM Watson AI Lab
December 8, 2021

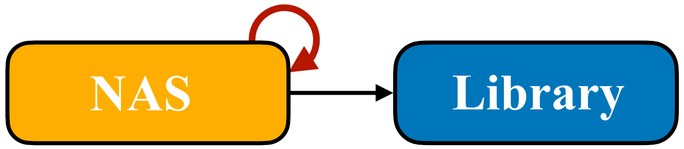
PRESS INQUIRIES



An MIT team's tinyML vision system outperforms other models in many image classification and detection tasks.

Photo courtesy of the researchers.

MCUNet: System-Algorithm Co-design

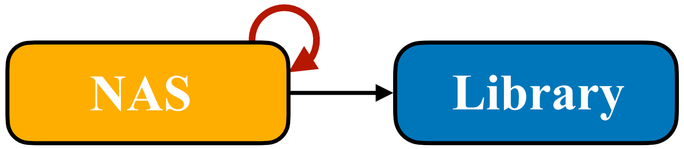


(a) Search NN model on an existing library

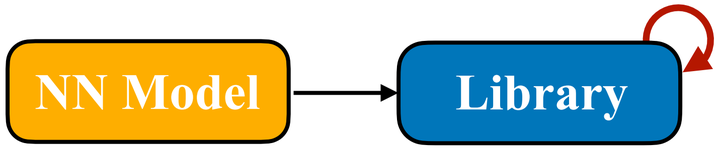
e.g., *ProxylessNAS, MnasNet*

MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2019]
ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware [Cai *et al.*, ICLR 2019]
MnasNet: Platform-Aware Neural Architecture Search for Mobile [Tan *et al.*, CVPR 2019)

MCUNet: System-Algorithm Co-design



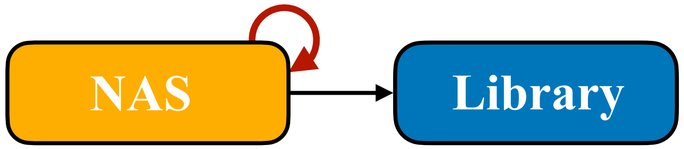
(a) Search NN model on an existing library
e.g., *ProxylessNAS, MnasNet*



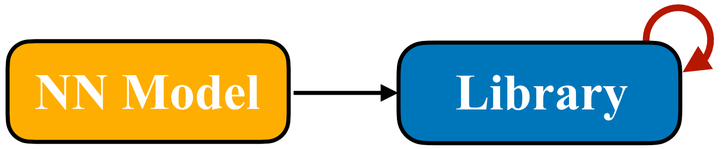
(b) Tune deep learning library given a NN model
e.g., *TVM*

TVM: An Automated End-to-End Optimizing Compiler for Deep Learning [Chen *et al.*, OSDI 2018]

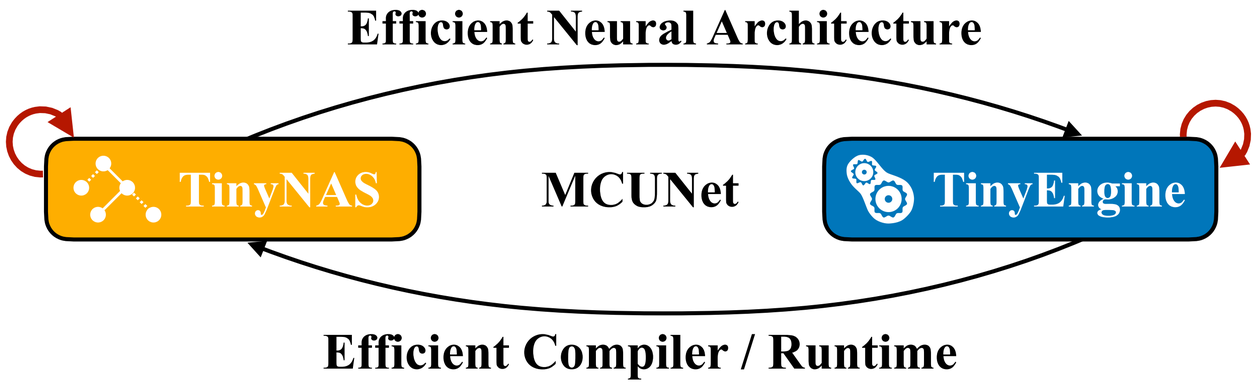
MCUNet: System-Algorithm Co-design



(a) Search NN model on an existing library
e.g., *ProxylessNAS, MnasNet*

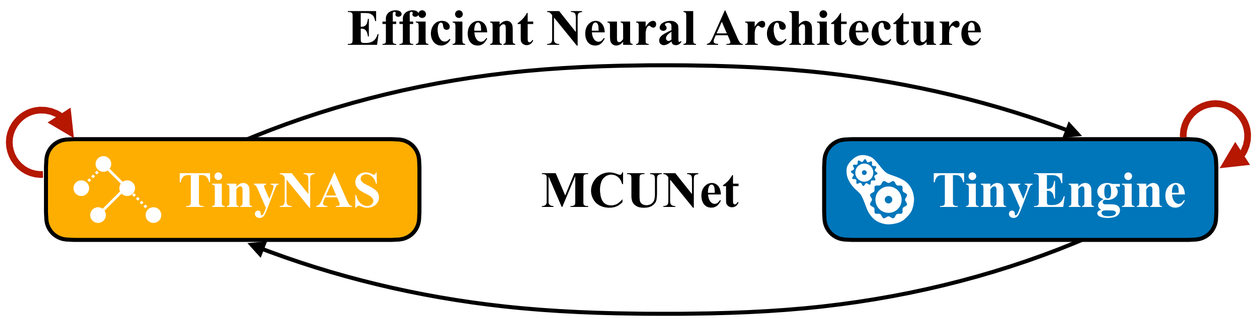


(b) Tune deep learning library given a NN model
e.g., *TVM*

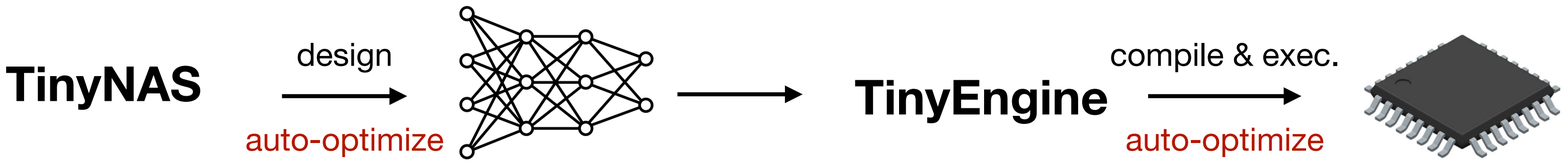


(c) *MCUNet*: system-algorithm co-design

MCUNet: System-Algorithm Co-design

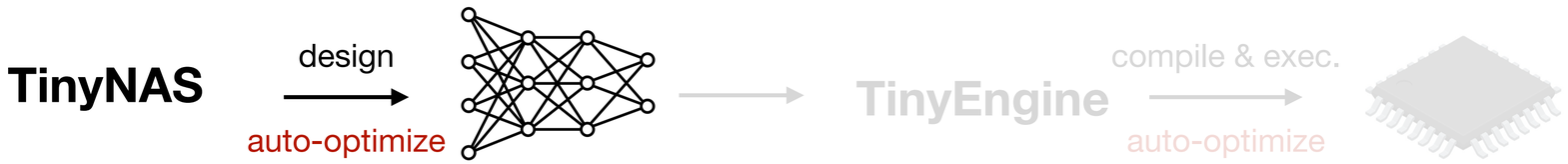


(c) *MCUNet*: system-algorithm co-design



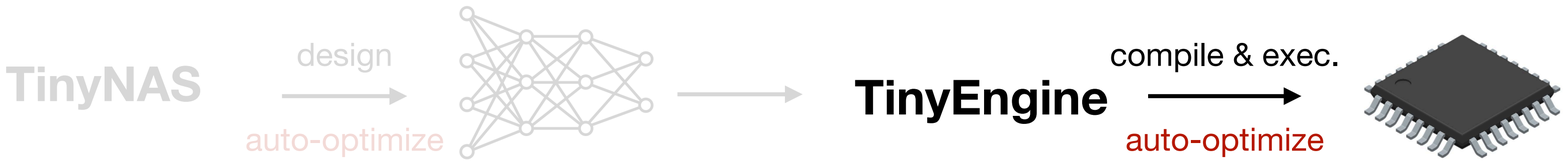
MCUNet: System-Algorithm Co-design

(c) *MCUNet*: system-algorithm co-design



MCUNet: System-Algorithm Co-design

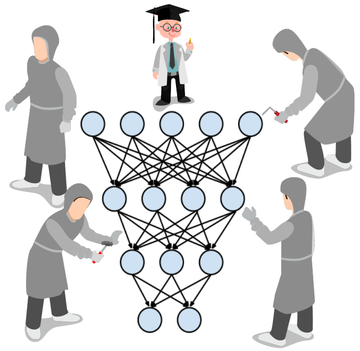
(c) *MCUNet*: system-algorithm co-design



Will introduce in lecture 17

TinyNAS: Two-Stage NAS for Tiny Memory Constraints

Recap: design automation for neural nets



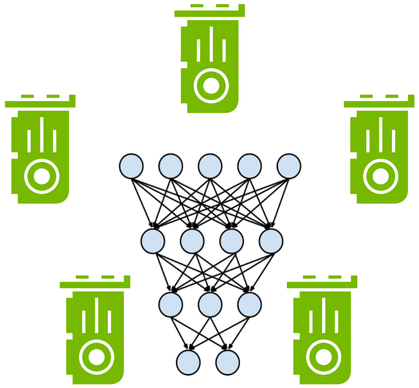
Use Human Expertise

Manual Architecture Design

Computational Resources

Use Machine Learning

Automatic Architecture Search

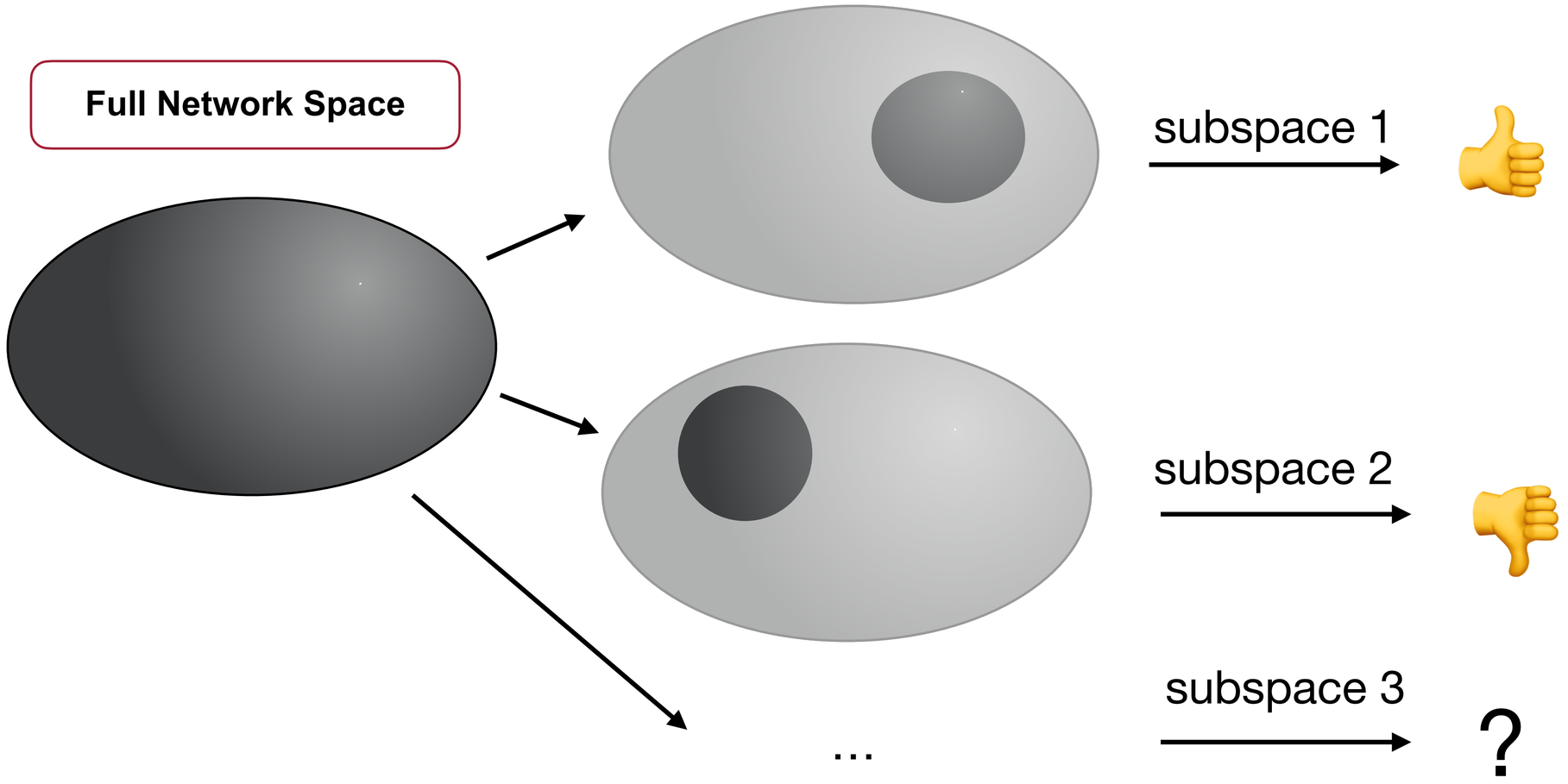


Neural Architecture Search (NAS)

TinyNAS: Two-Stage NAS for Tiny Memory Constraints

Problem: what is the right search space?

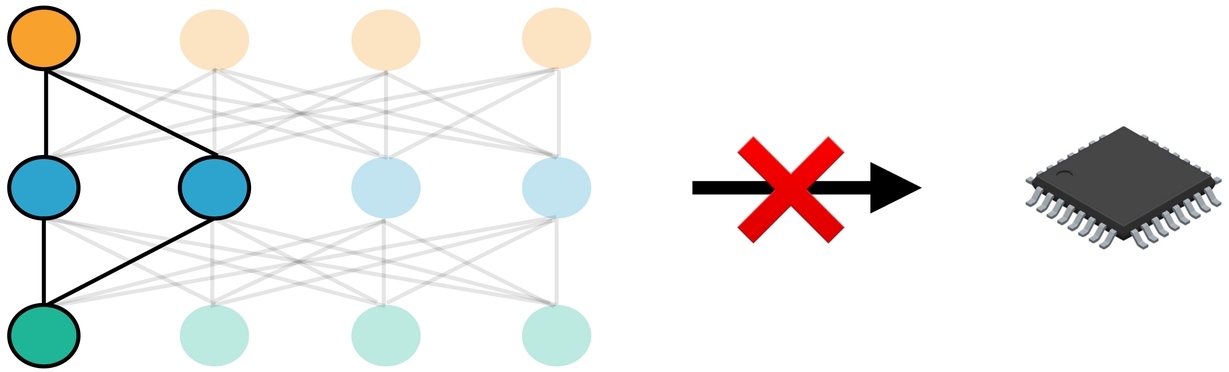
- The quality of search space largely determines the performance of the searched model



TinyNAS: Two-Stage NAS for Tiny Memory Constraints

Problem: what is the right search space?

- The quality of search space largely determines the performance of the searched model
- **Option 1**: reuse the carefully designed mobile search space (e.g., MnasNet space) for tinyML
  - **Problem**: the smallest sub-network in the search space **cannot** fit the hardware due to the huge gap (recap: 320KB vs. 4GB memory)



TinyNAS: Two-Stage NAS for Tiny Memory Constraints

Problem: what is the right search space?

- The quality of search space largely determines the performance of the searched model
- **Option 1**: reuse the carefully designed mobile search space (e.g., MnasNet space) for tinyML
  - **Problem**: the smallest sub-network in the search space **cannot** fit the hardware due to the huge gap (recap: 320KB vs. 4GB memory)
- **Option 2**: (smartly) pick a search space for the IoT device
  - **Problem**: how?

TinyNAS: Two-Stage NAS for Tiny Memory Constraints

Example: how to scale the MBNet-alike search space?

- We can scale the MBNet-alike search space by using different resolution R and width multiplier W
  - A specific R* and W* lead to a sub-search space
- For example
  - The original search space (R=224, W=1.0) is good for smartphones

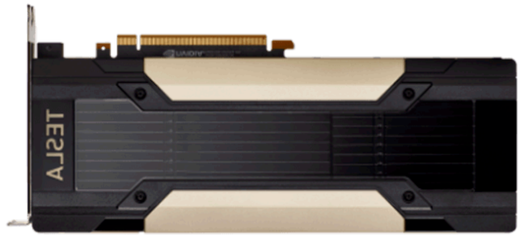
R=224, W=1.0

TinyNAS: Two-Stage NAS for Tiny Memory Constraints

Example: how to scale the MBNet-alike search space?

- We can scale the MBNet-alike search space by using different resolution R and width multiplier W
  - A specific R* and W* lead to a sub-search space
- For example
  - The original search space (R=224, W=1.0) is good for smartphones
  - A scaled-up space is good for GPUs

R=260, W=1.4



R=224, W=1.0

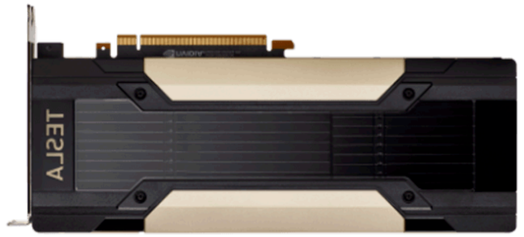
Once-for-All: Train One Network and Specialize it for Efficient Deployment [Cai *et al.*, ICLR 2020]

TinyNAS: Two-Stage NAS for Tiny Memory Constraints

Example: how to scale the MBNet-alike search space?

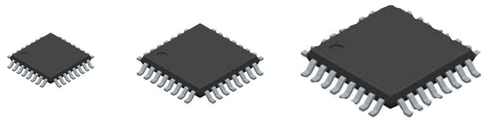
- We can scale the MBNet-alike search space by using different resolution R and width multiplier W
  - A specific R* and W* lead to a sub-search space
- For example
  - The original search space (R=224, W=1.0) is good for smartphones
  - A scaled-up space is good for GPUs
  - How can we choose different R* and W* for different microcontrollers?

R=260, W=1.4



R=224, W=1.0

R=?, W=?

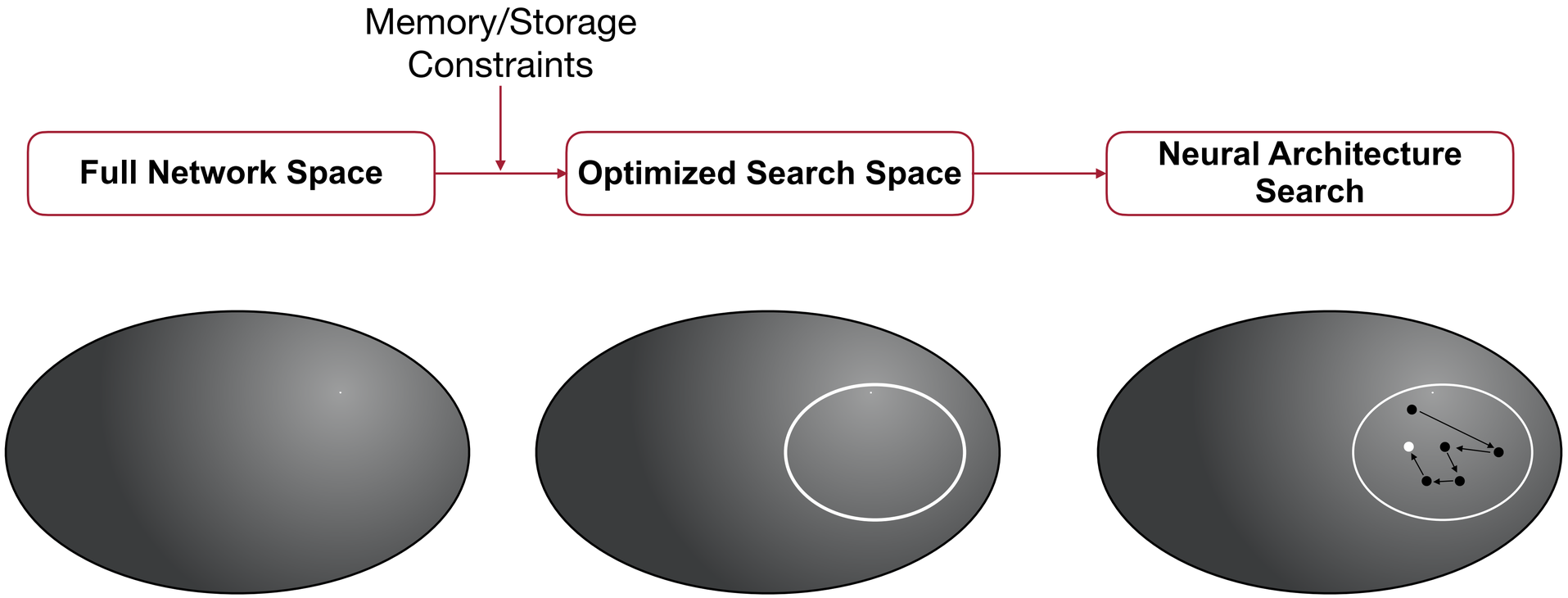


F412/F743/H746/…
256kB/320kB/512kB/…

Once-for-All: Train One Network and Specialize it for Efficient Deployment [Cai *et al.*, ICLR 2020]

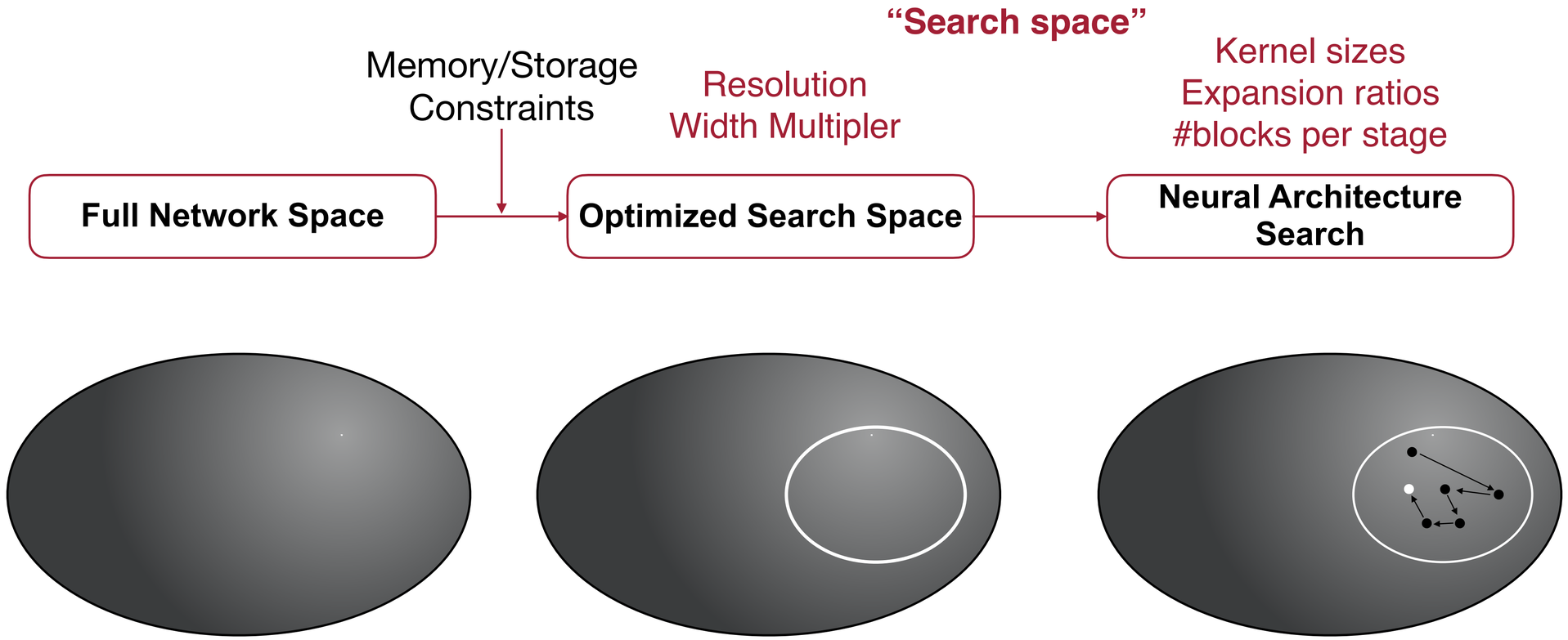
TinyNAS: Two-Stage NAS for Tiny Memory Constraints

First design the design space, then search the sub-network



TinyNAS: Two-Stage NAS for Tiny Memory Constraints

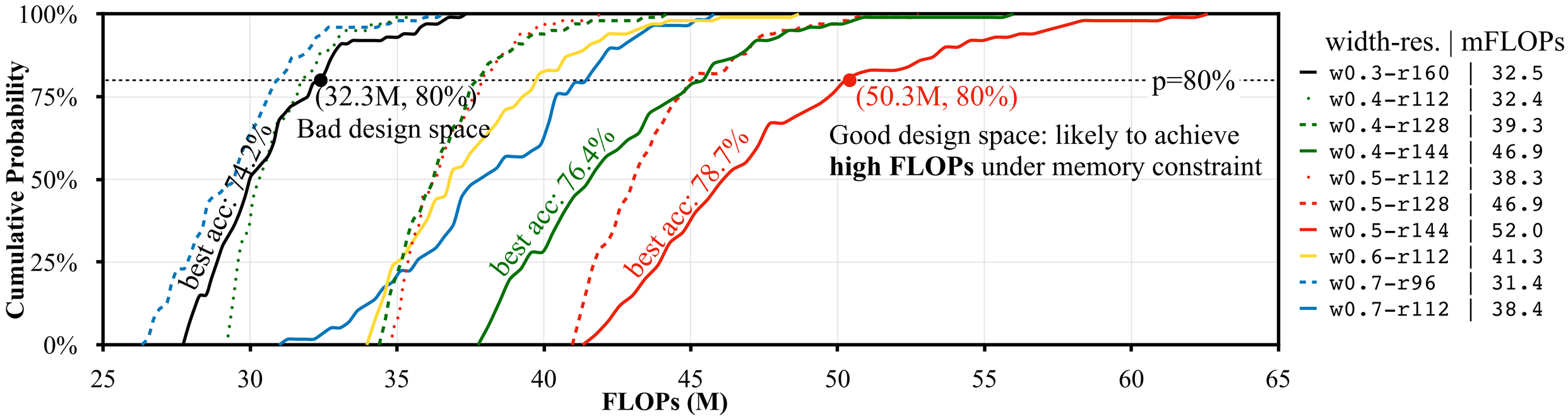
First design the design space, then search the sub-network



TinyNAS: Two-Stage NAS for Tiny Memory Constraints

1. Automated search space optimization

Analyzing **FLOPs distribution** of satisfying models in each search space:
Larger FLOPs -> Larger model capacity -> More likely to give higher accuracy

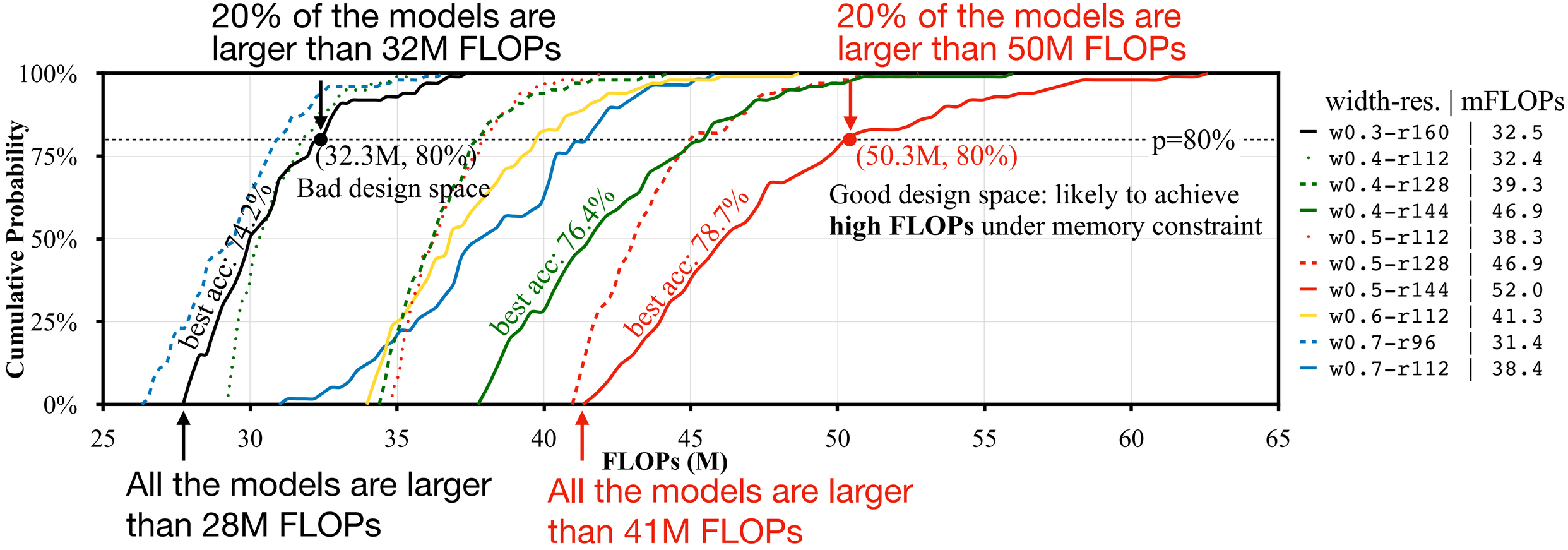


TinyNAS: Two-Stage NAS for Tiny Memory Constraints

1. Automated search space optimization

Analyzing **FLOPs distribution** of satisfying models in each search space:

Larger FLOPs -> Larger model capacity -> More likely to give higher accuracy



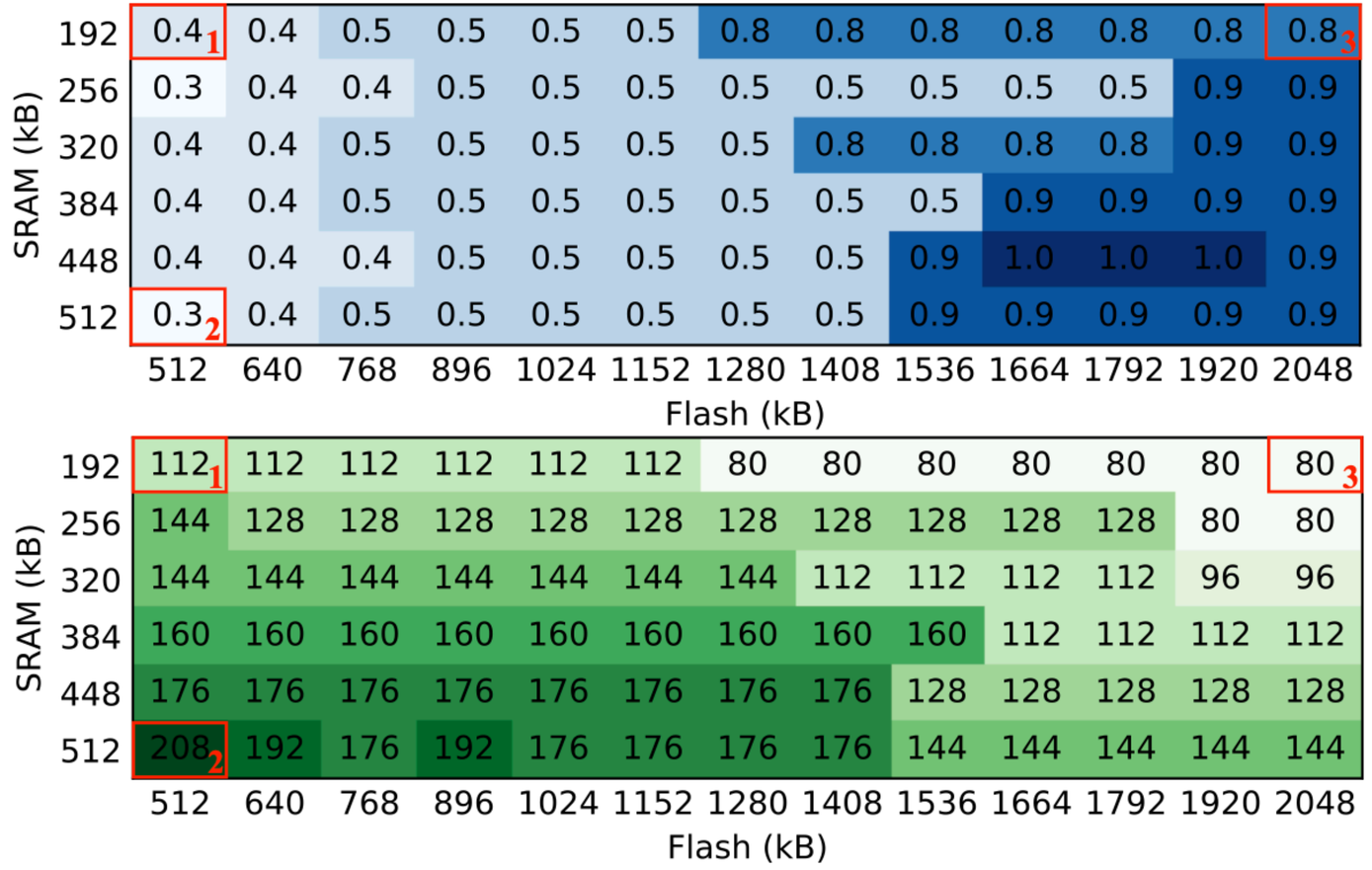
TinyNAS: Two-Stage NAS for Tiny Memory Constraints

1. Automated search space optimization

- Best configurations (width multiplier, resolutions) found for different SRAM, Flash combinations

Flash increases, same SRAM: what would happen?

SRAM increases, same Flash: what would happen?



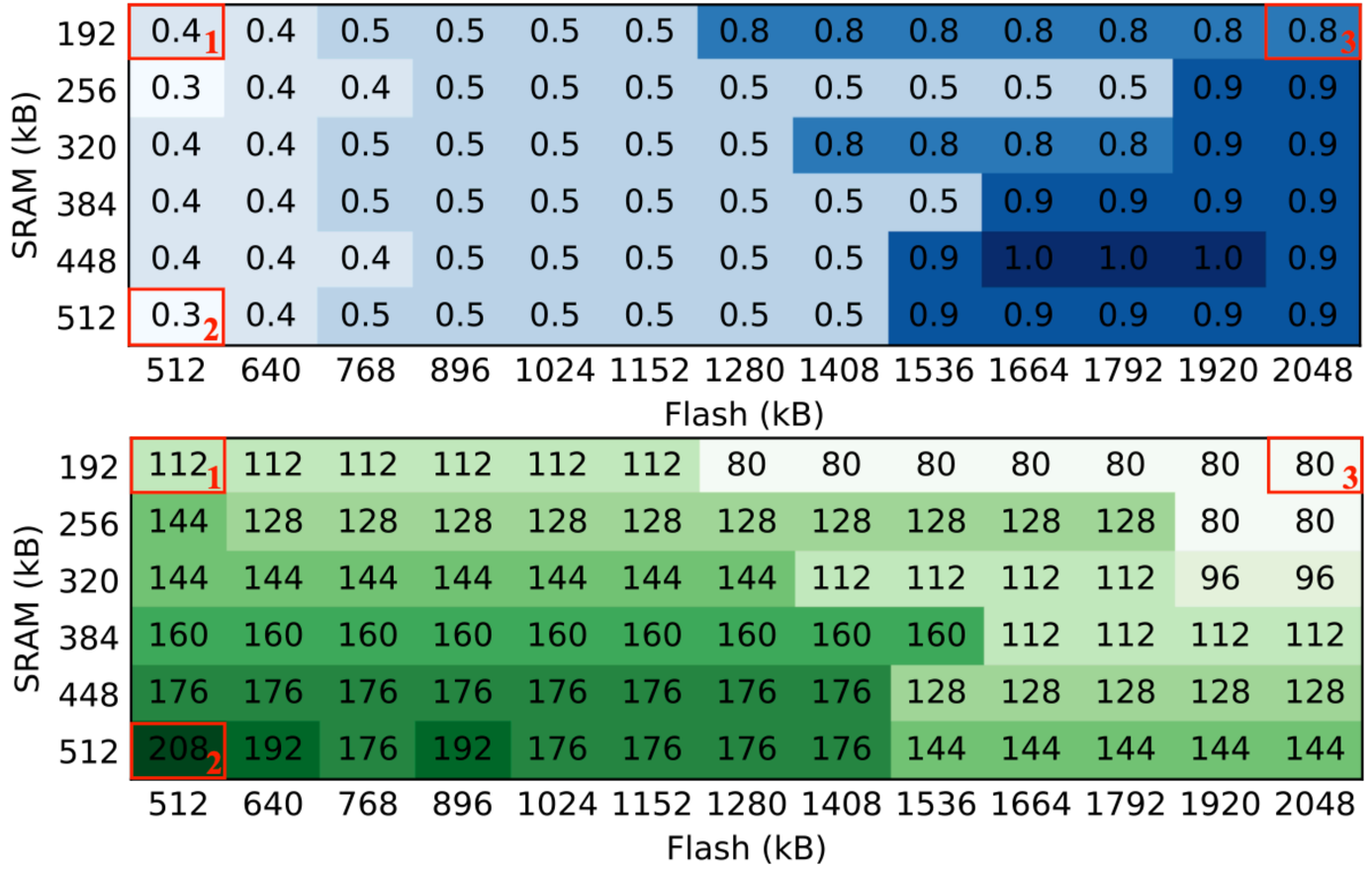
TinyNAS: Two-Stage NAS for Tiny Memory Constraints

1. Automated search space optimization

- Best configurations (width multiplier, resolutions) found for different SRAM, Flash combinations

Flash increases, same SRAM: larger model, smaller resolution

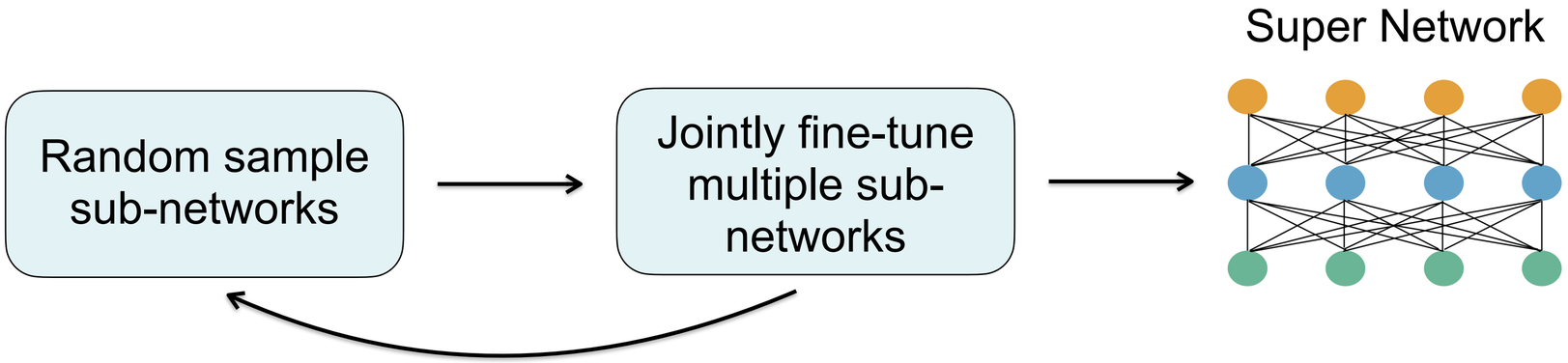
SRAM increases, same Flash: larger resolution, smaller model



TinyNAS: Two-Stage NAS for Tiny Memory Constraints

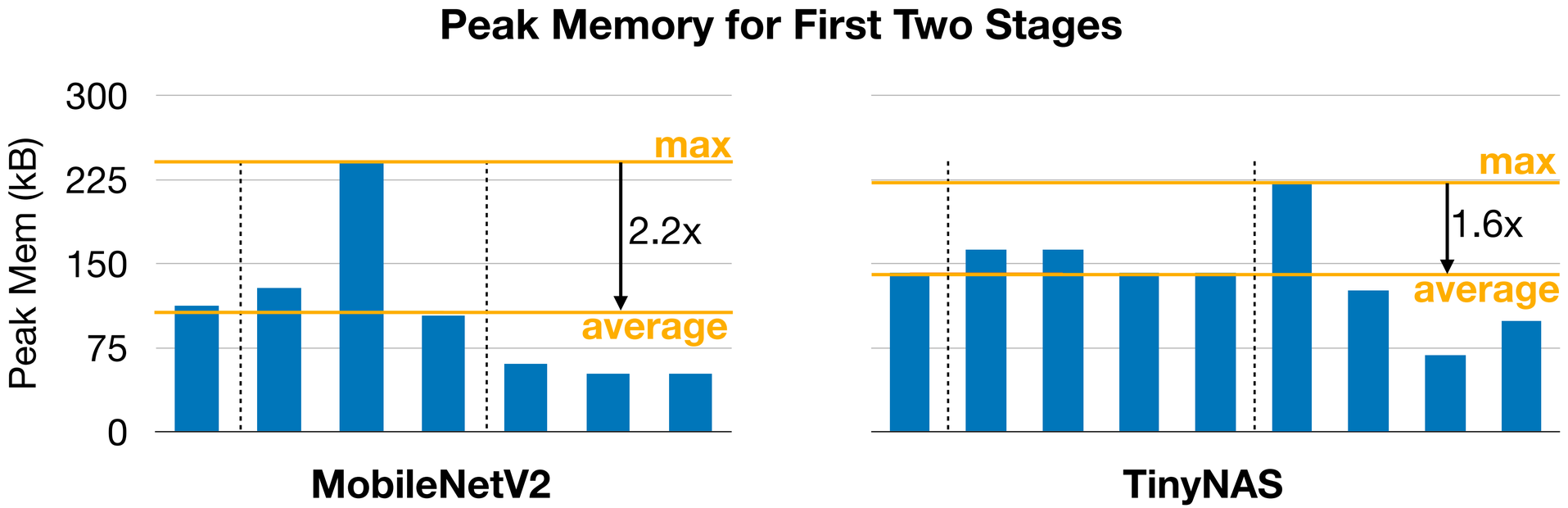
2. Resource-constrained model specialization

- One-shot NAS through weight sharing



- Small child networks are nested in large ones.

TinyNAS Better Utilizes the Memory

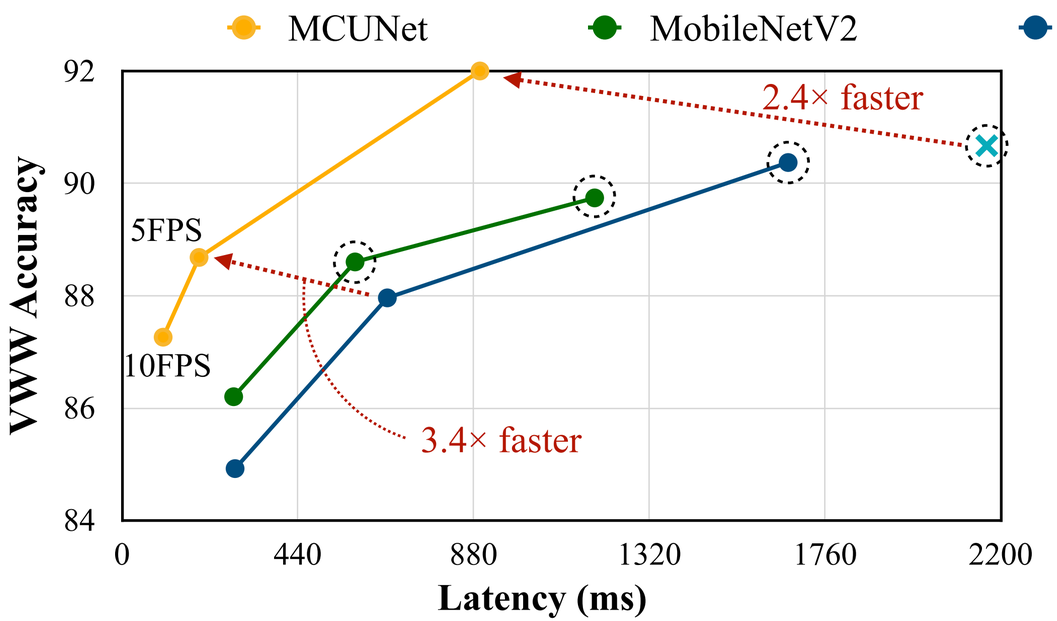


- TinyNAS designs networks with more uniform peak memory for each block, allowing us to fit a larger model at the same amount of memory

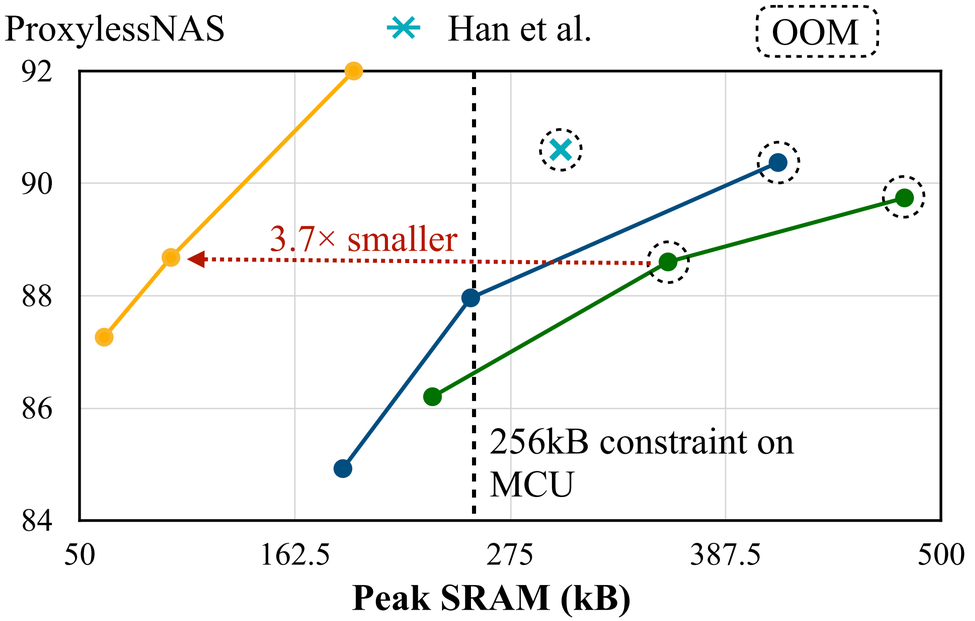
Outperforming Manual&NAS Models

MCUNet achieves higher accuracy at lower memory

- Higher accuracy at 3x faster inference speed, 4x smaller memory cost on the Visual Wake Words (VWW) dataset



(a) Trade-off: accuracy *vs.* measured latency



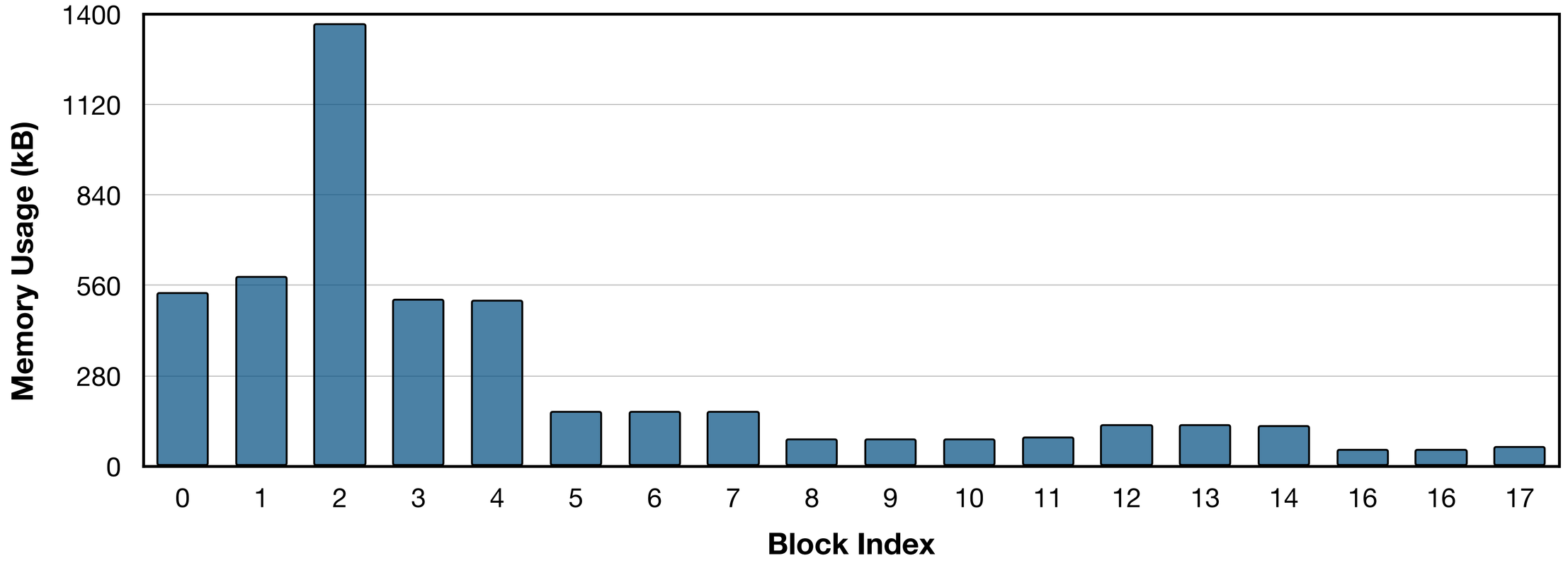
(b) Trade-off: accuracy *vs.* peak memory

Visual wake words dataset. [Chowdhery *et al.*, arXiv 2019]

MCUNetV2: Patch-based Inference

Further reducing SRAM by breaking the memory bottleneck

- Recap: SRAM usage is dynamic; we care about **peak** SRAM.
- Let's take a loop at the SRAM usage of each block in MobileNetV2

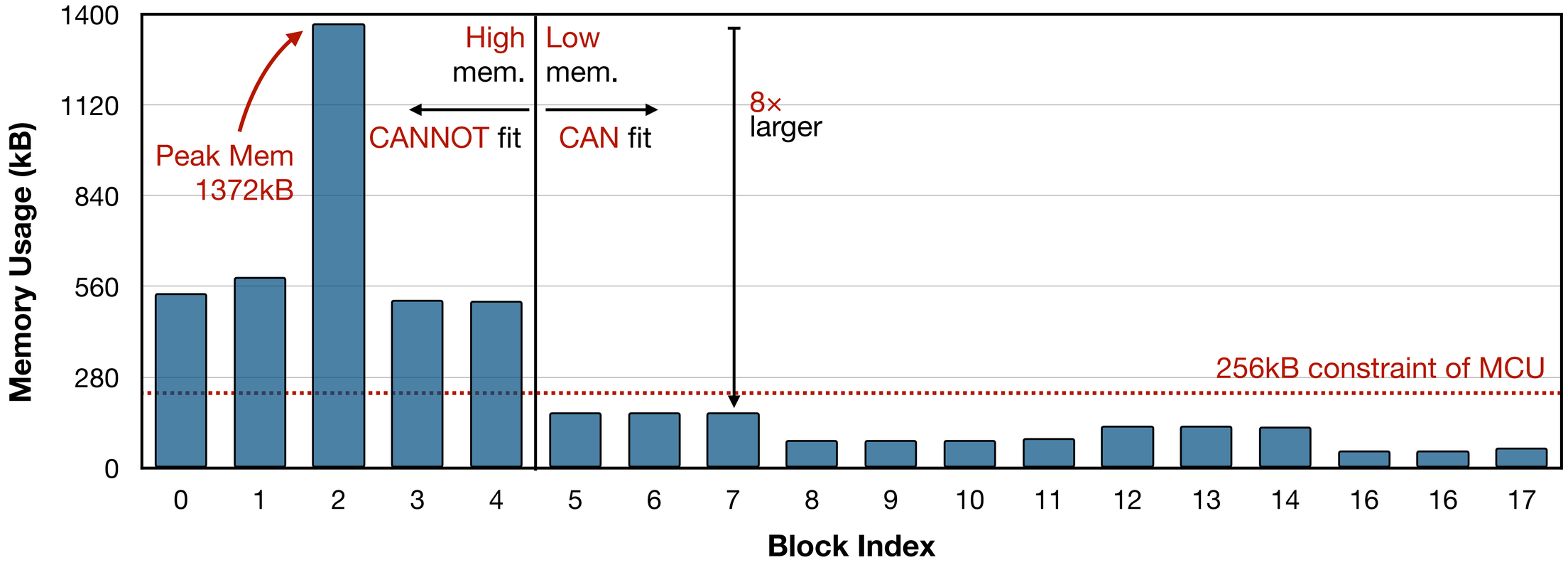


MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

MCUNetV2: Patch-based Inference

Imbalanced Memory Distribution of CNNs

- Recap: SRAM usage is dynamic; we care about **peak** SRAM.
- Let's take a loop at the SRAM usage of each layer in MobileNetV2

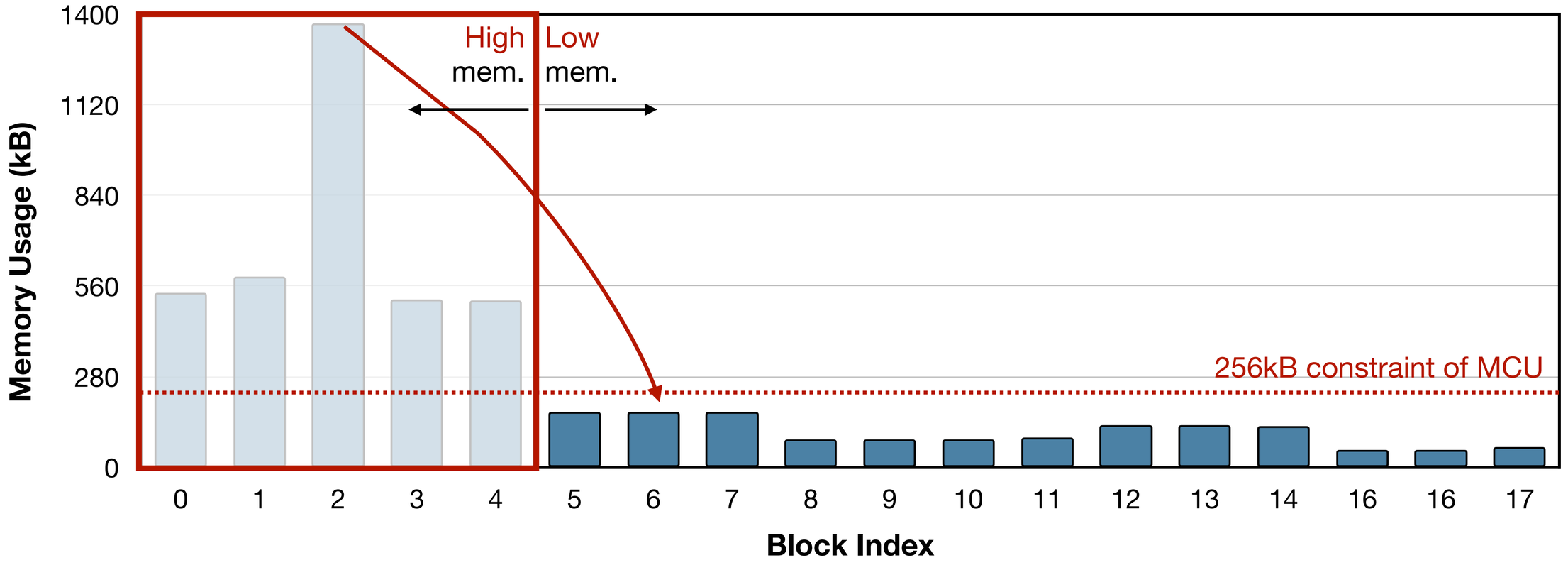


MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

MCUNetV2: Patch-based Inference

Imbalanced Memory Distribution of CNNs

- If we can reduce the memory usage of the **initial** stage, we can reduce the **overall** memory

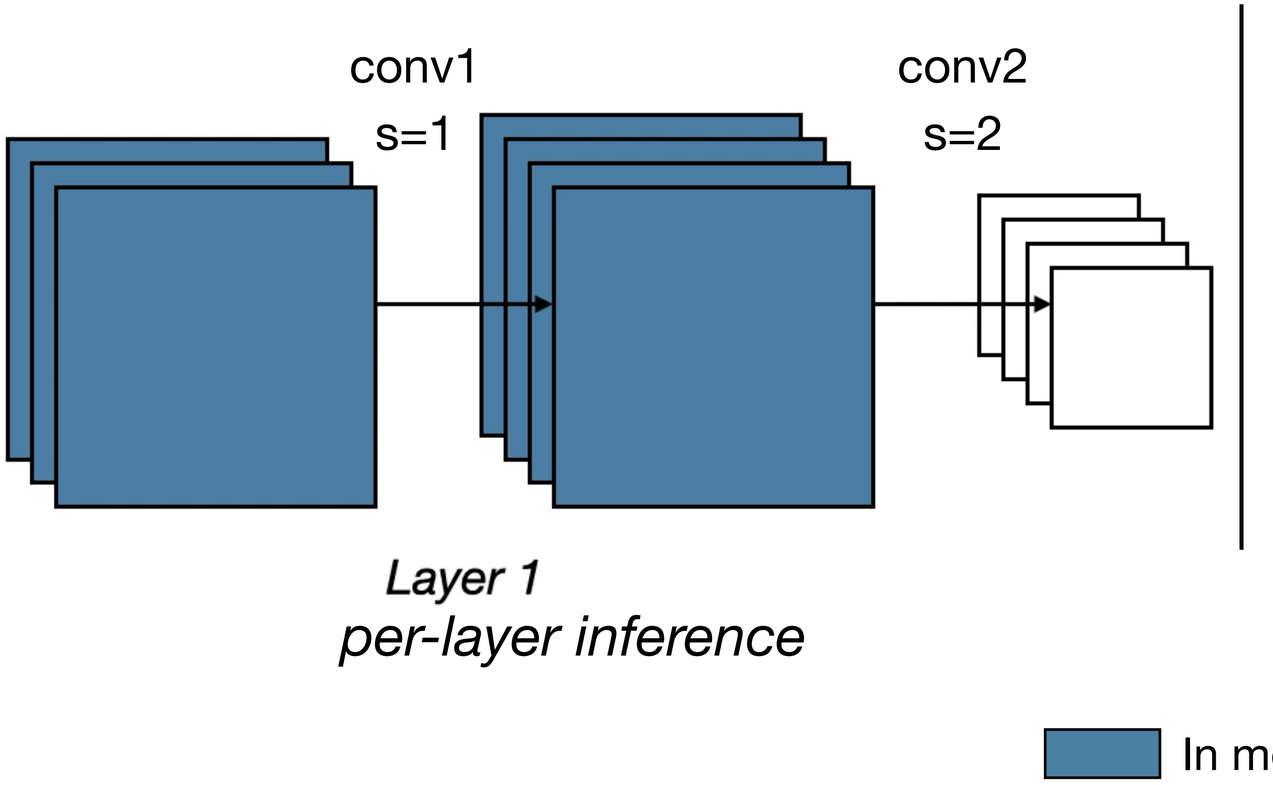


MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

MCUNetV2: Patch-based Inference

1. Saving Memory with Patch-based Inference

- Break the memory bottleneck with patch-based inference
  - a practical 2-layer example

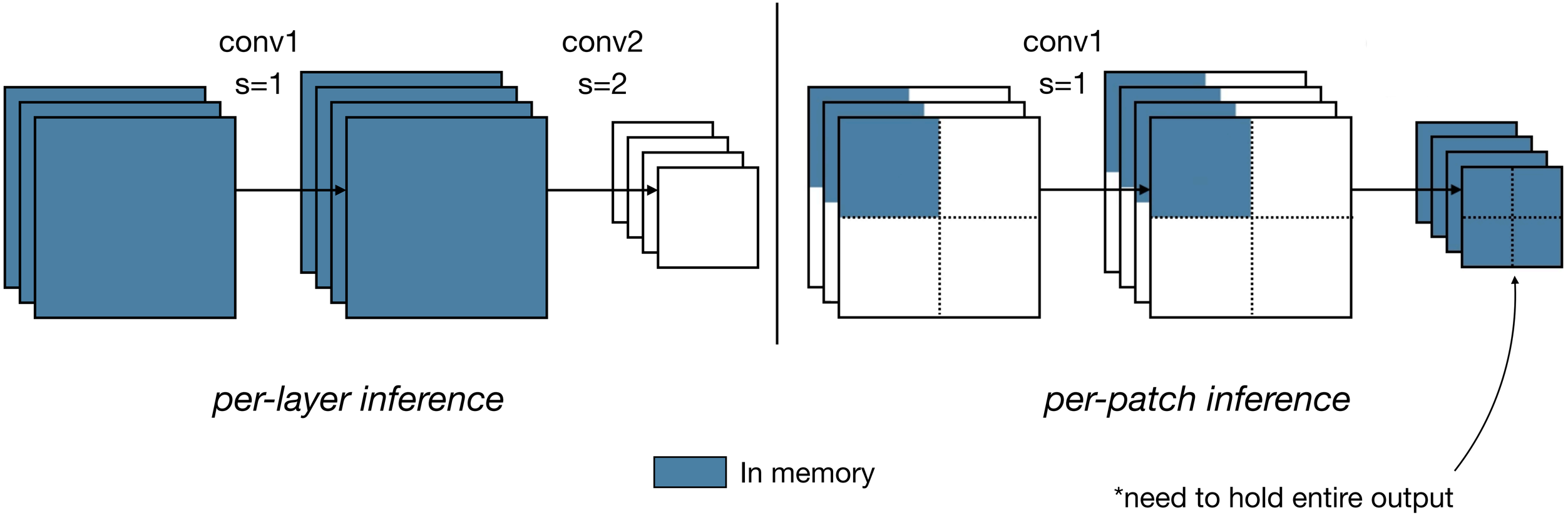


MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

MCUNetV2: Patch-based Inference

1. Saving Memory with Patch-based Inference

- Break the memory bottleneck with patch-based inference
  - a practical 2-layer example

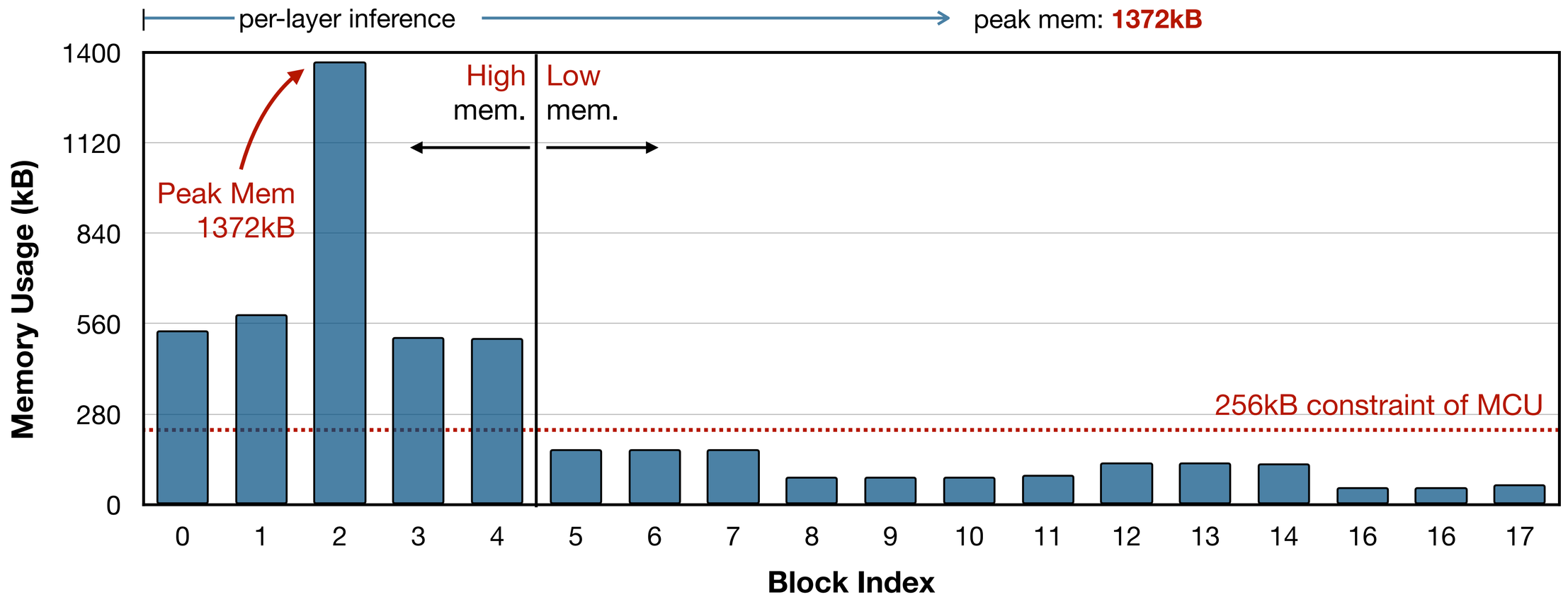


MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

MCUNetV2: Patch-based Inference

1. Saving Memory with Patch-based Inference

- Memory saving for MobileNetV2

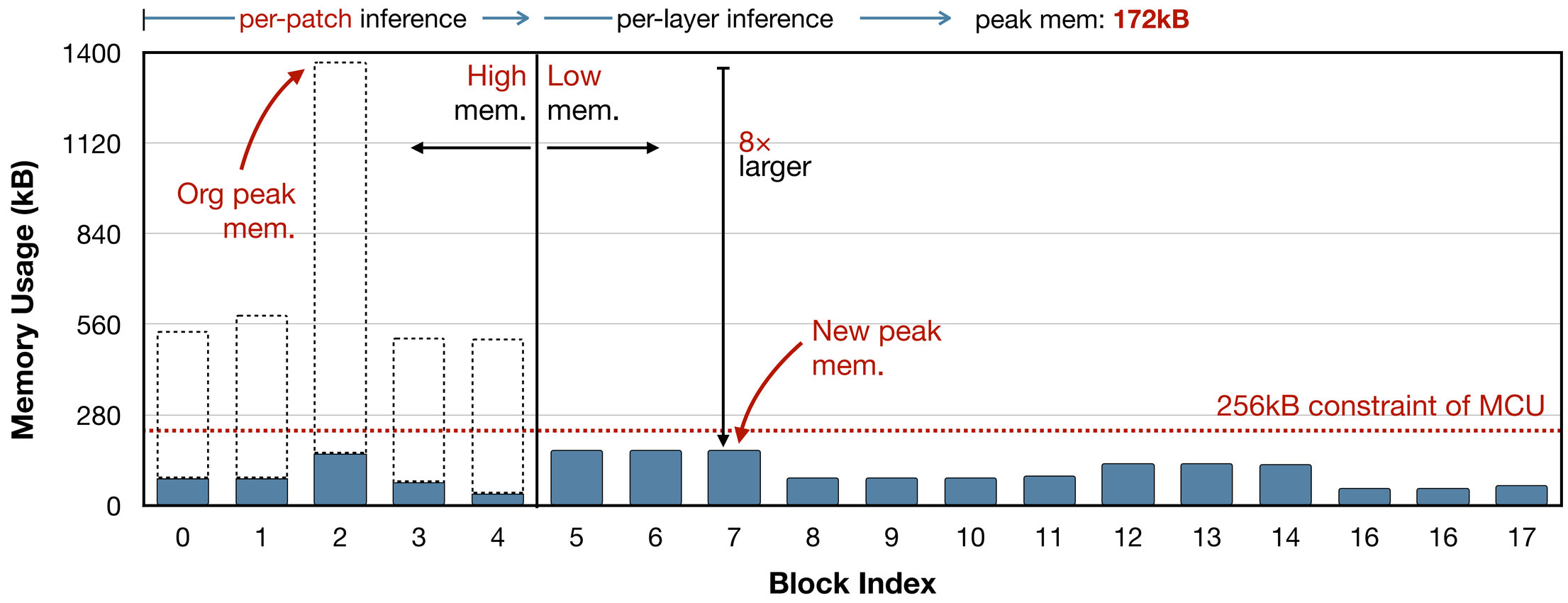


MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

MCUNetV2: Patch-based Inference

1. Saving Memory with Patch-based Inference

- Memory saving for MobileNetV2

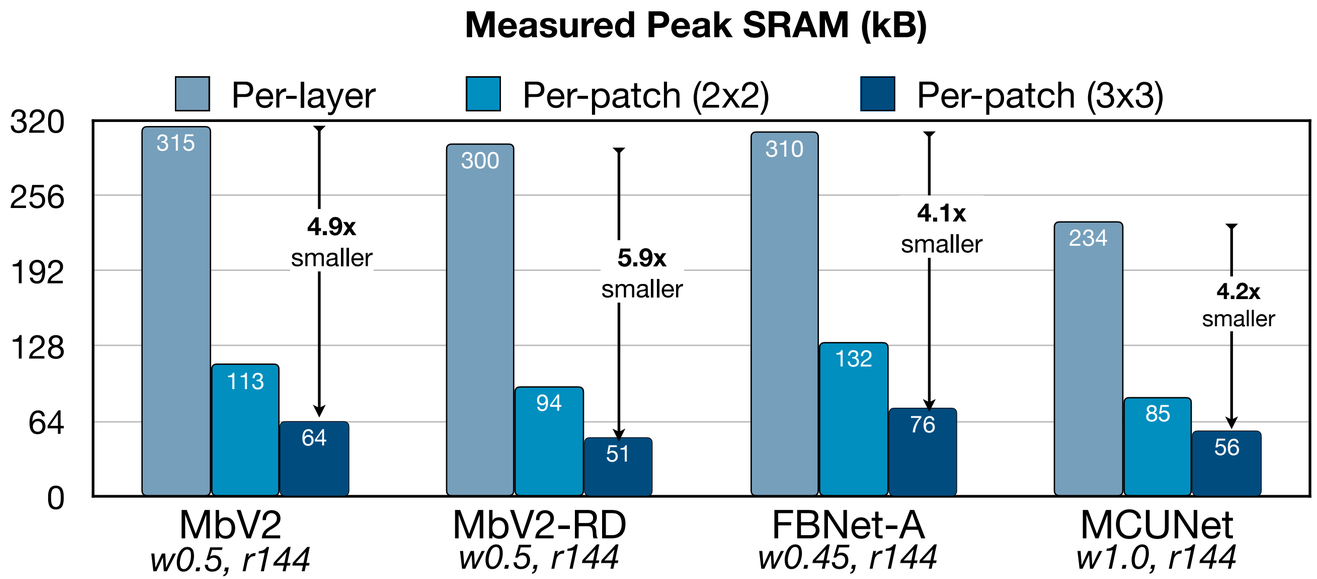


MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

MCUNetV2: Patch-based Inference

1. Saving Memory with Patch-based Inference

- Memory saving for other models
  - Baseline: TinyEngine. Measured on STM32F746

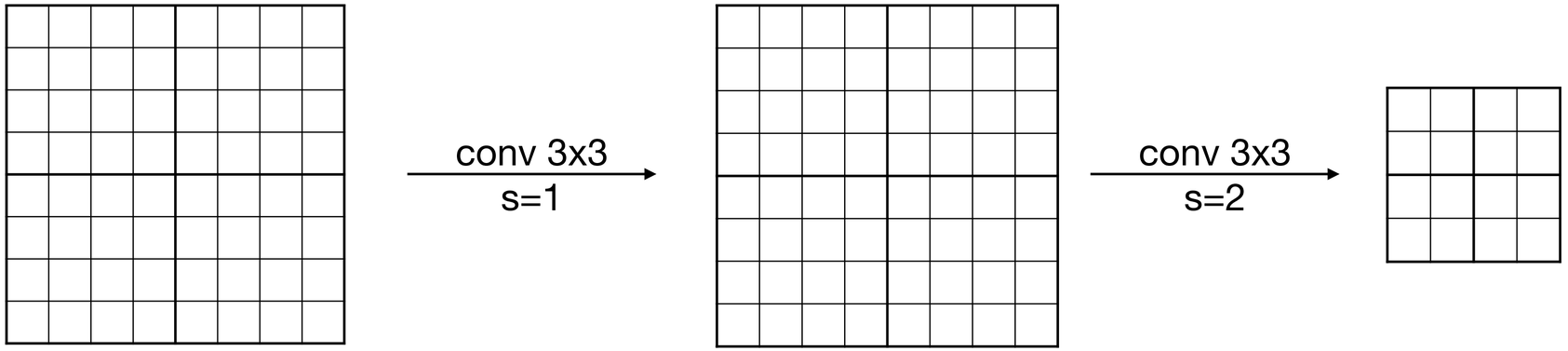


MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

MCUNetV2: Patch-based Inference

Problem: halo leads to repeated computation

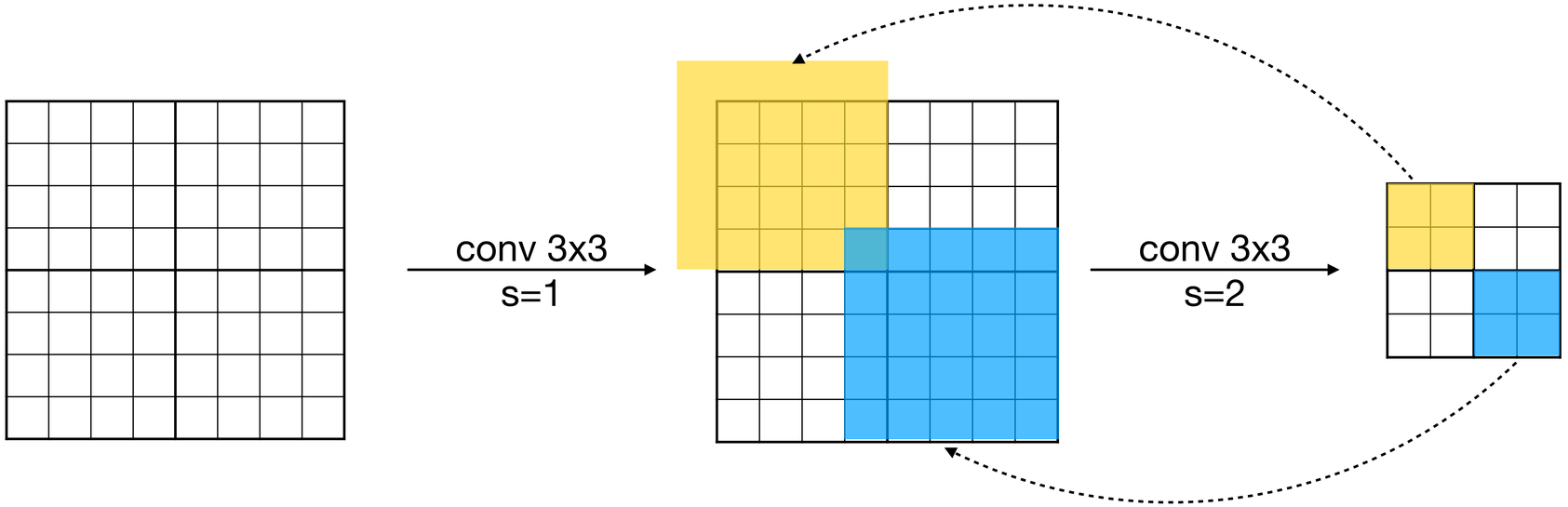
- Example: when using 2x2 patches



MCUNetV2: Patch-based Inference

Problem: halo leads to repeated computation

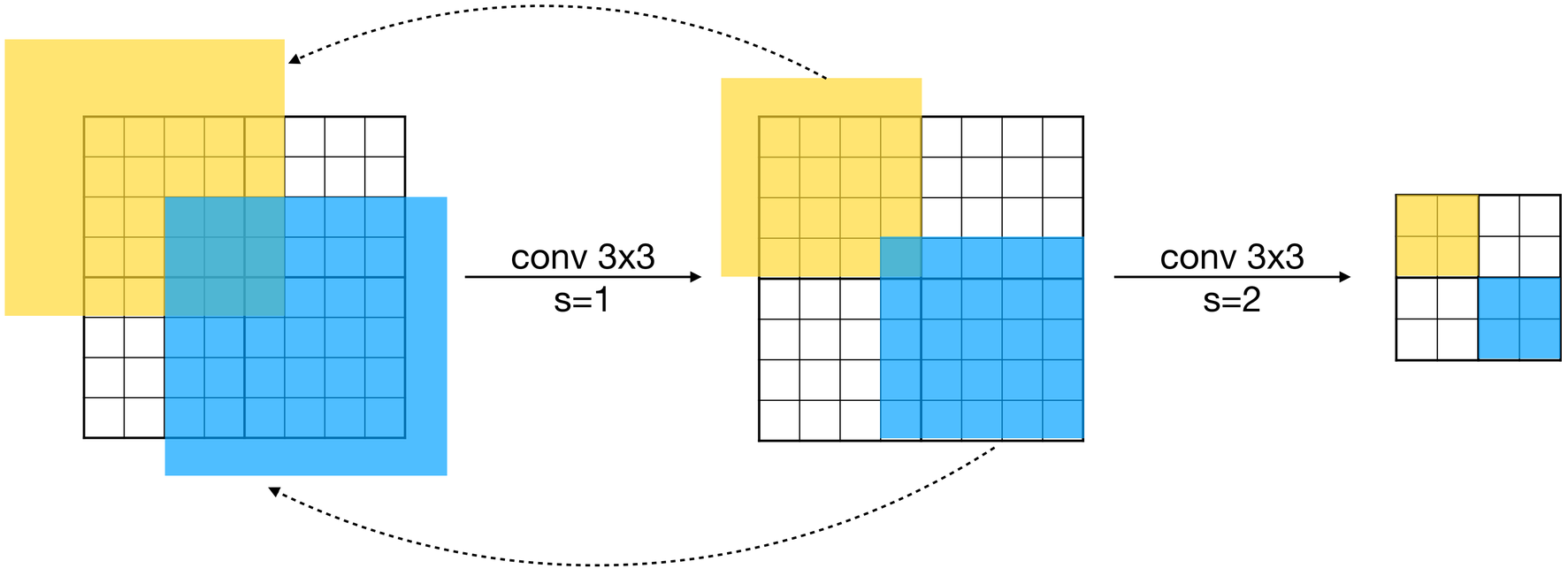
- Example: when using 2x2 patches



MCUNetV2: Patch-based Inference

Problem: halo leads to repeated computation

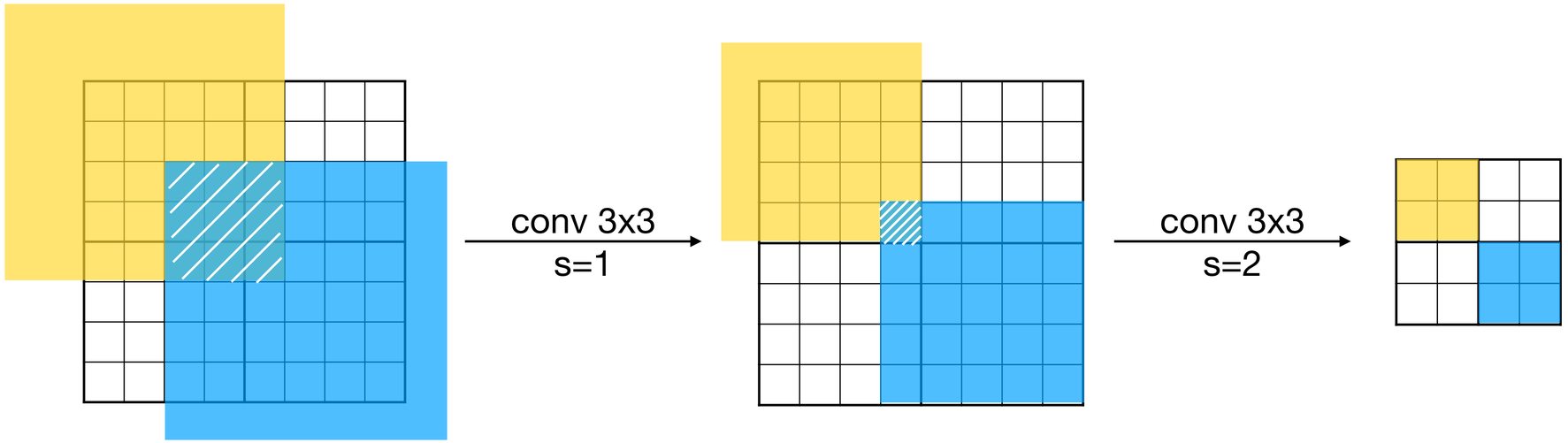
- Example: when using 2x2 patches



MCUNetV2: Patch-based Inference

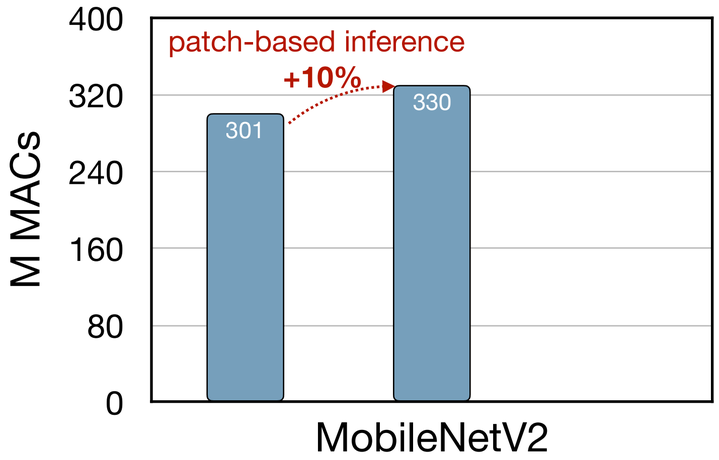
Problem: halo leads to repeated computation

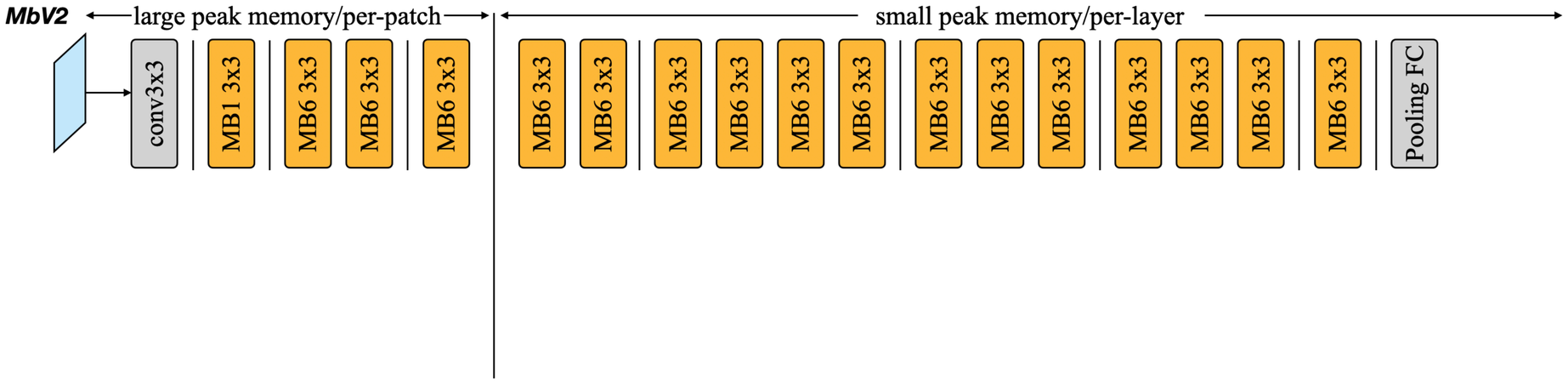
- Spatial overlapping gets larger as the **receptive field** grows! Leading to larger computation overhead



MCUNetV2: Patch-based Inference

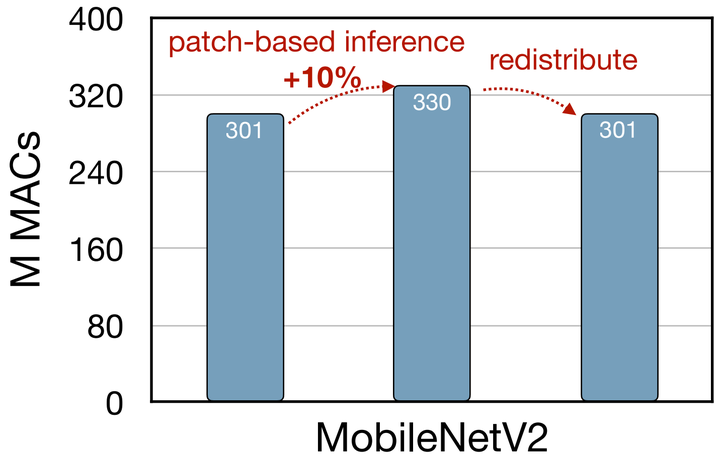
2. Network Redistribution to Reduce Overhead





MCUNetV2: Patch-based Inference

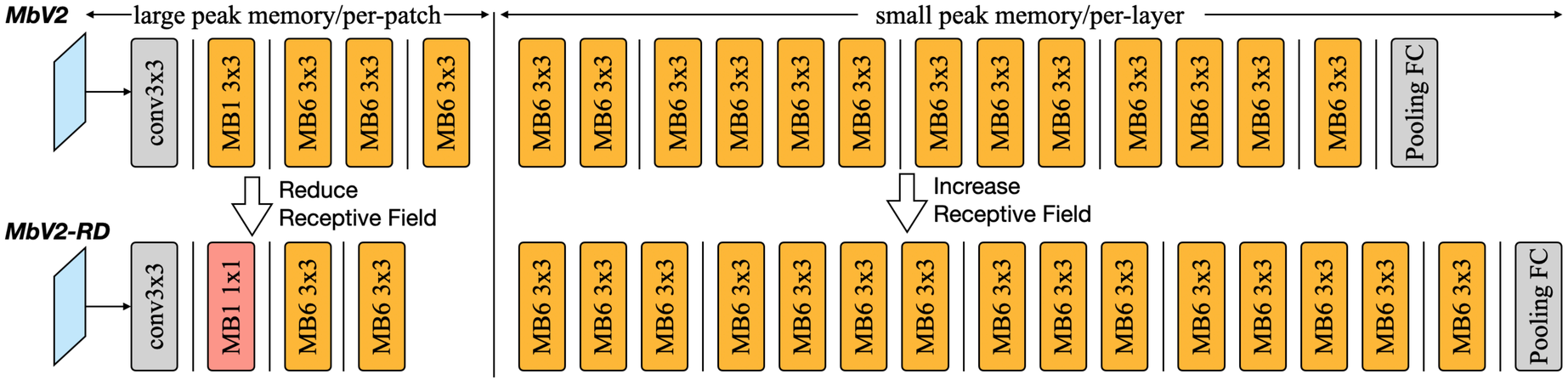
2. Network Redistribution to Reduce Overhead



Same performance on:

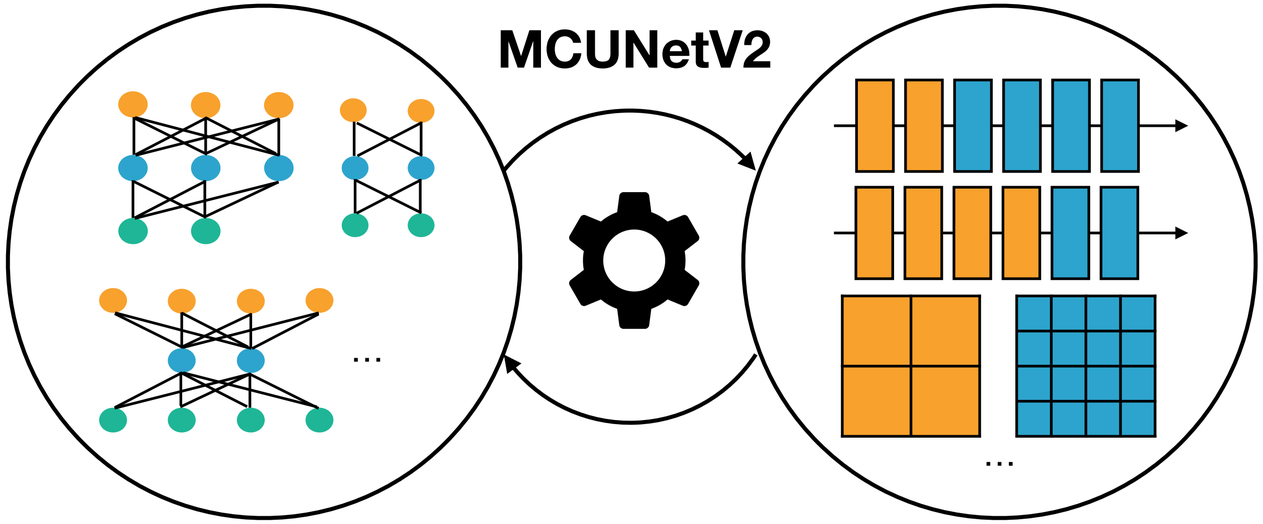
- Image classification
- Object detection
- ….

Negligible overhead



MCUNetV2: Patch-based Inference

3. Joint Automated Search for Optimization



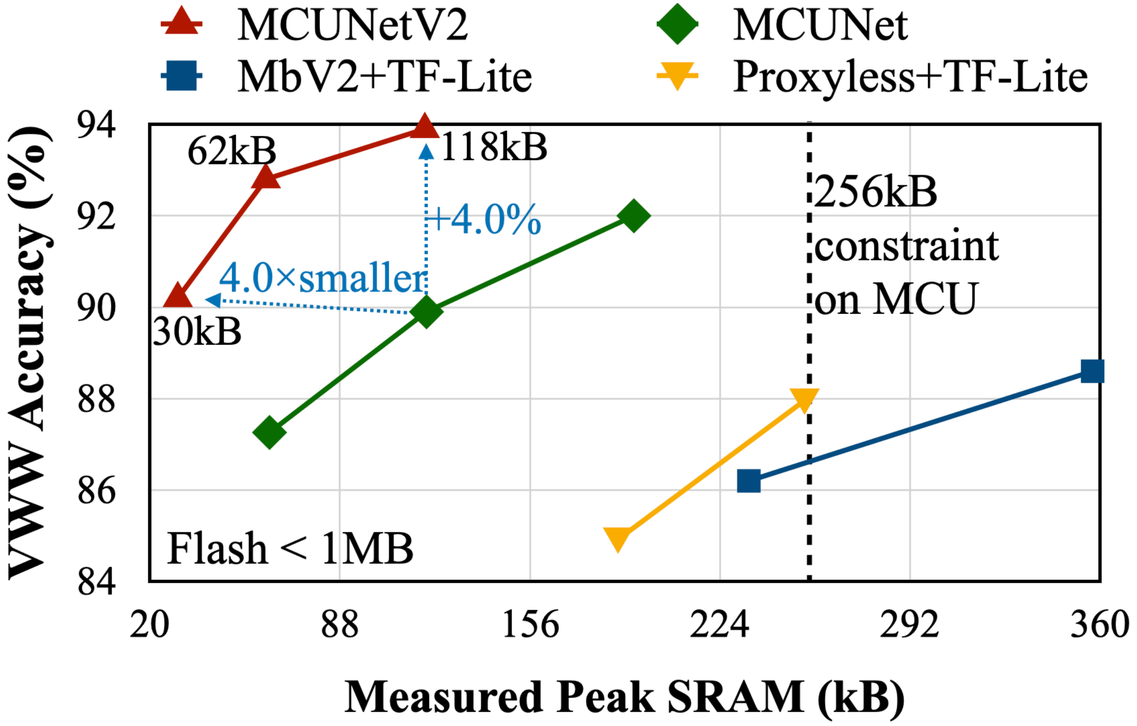
Neural architecture
#layers
#channels
kernel size
…

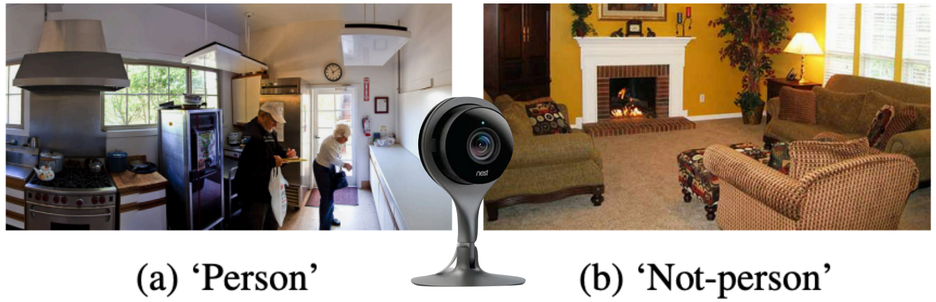
Inference scheduling
#patches
#layers for patch-based
other knobs from TinyEngine
…

MCUNetV2: Patch-based Inference

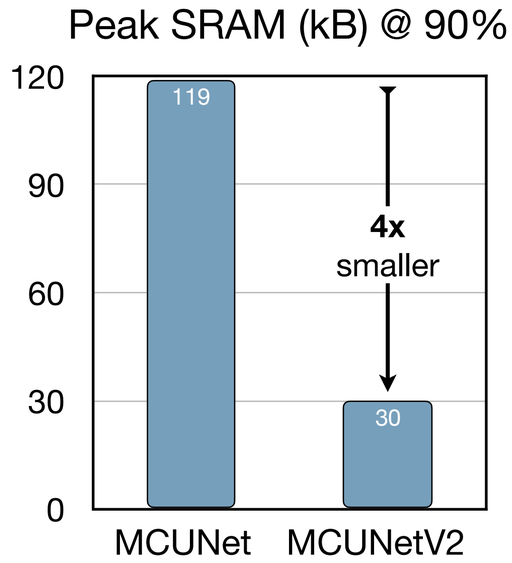
Visual Wake Words under 32KB memory

- Higher accuracy, 4x lower SRAM





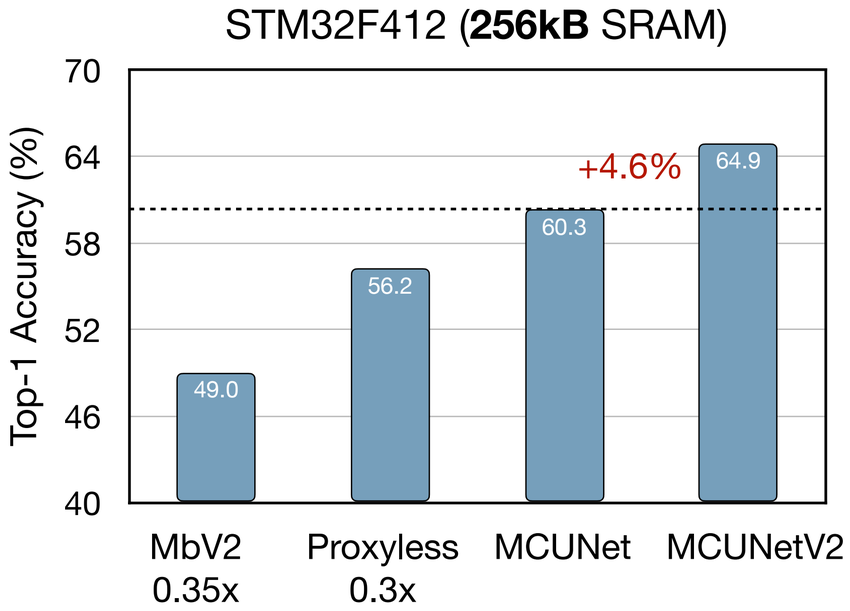
(a) 'Person' (b) 'Not-person'

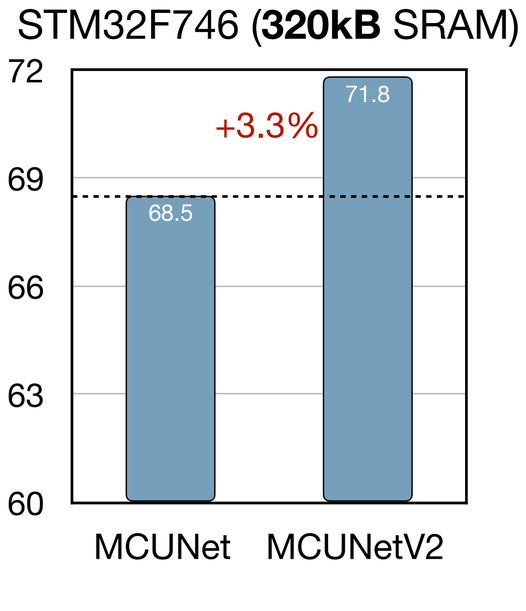


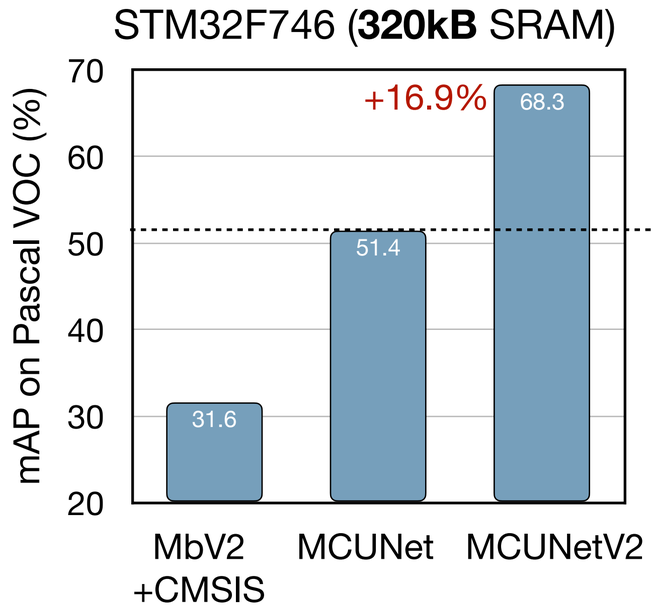
MCUNetV2: Patch-based Inference

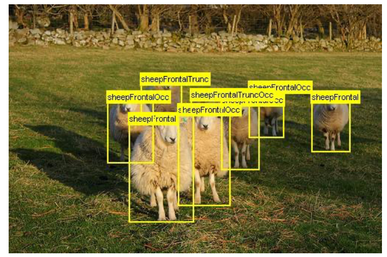
Advancing classification and detection

- ImageNet classification and Pascal VOC object detection
- **int8** quantization



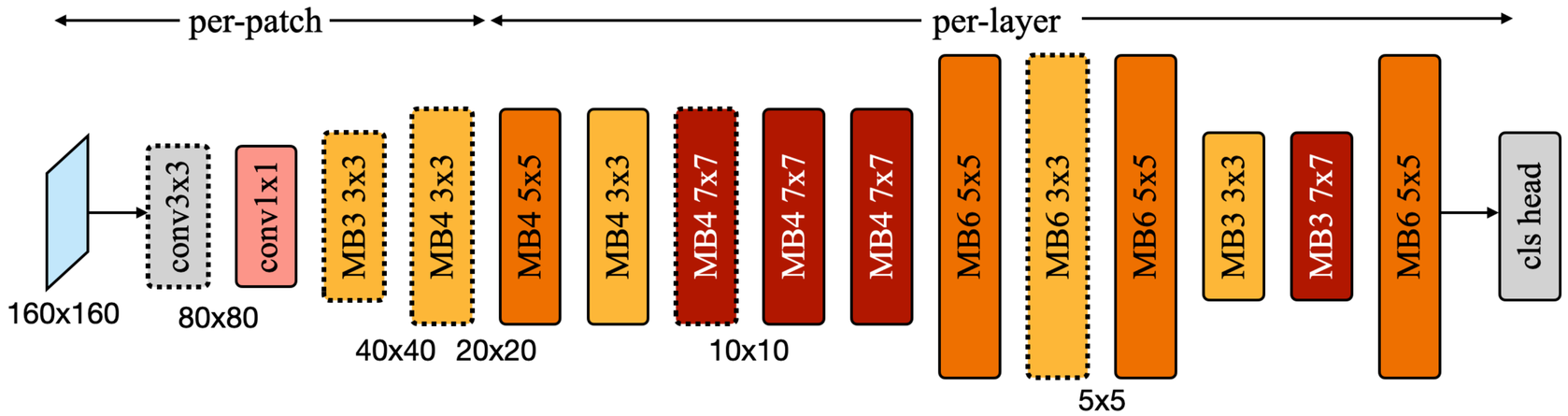






MCUNetV2: Patch-based Inference

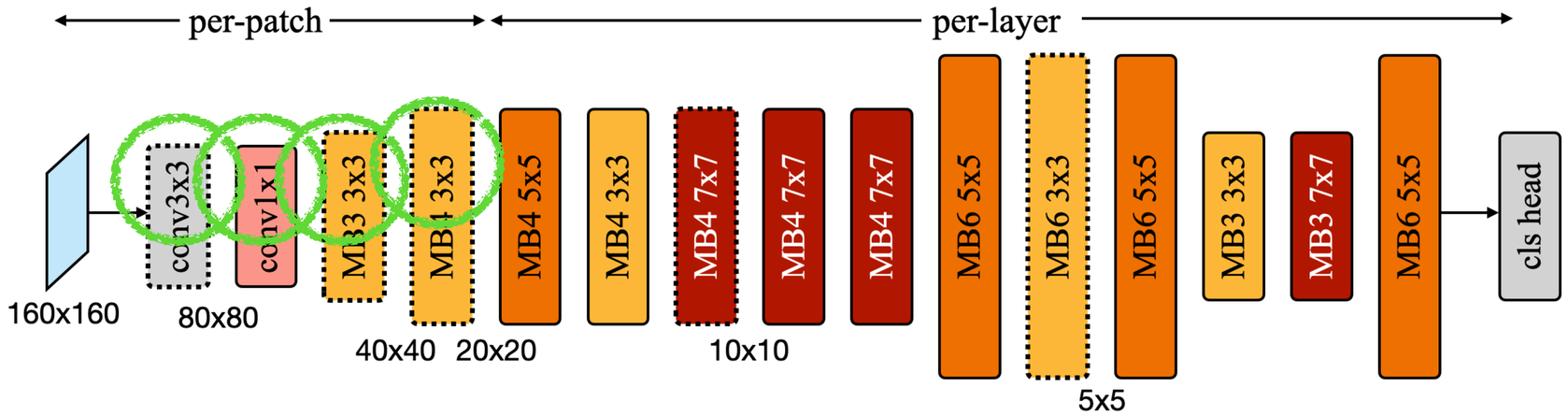
Dissecting MCUNetV2 architecture



Sample arch from VWW. ***Legend:*** `MB{expansion}_{k_size}x{k_size}`

MCUNetV2: Patch-based Inference

Dissecting MCUNetV2 architecture

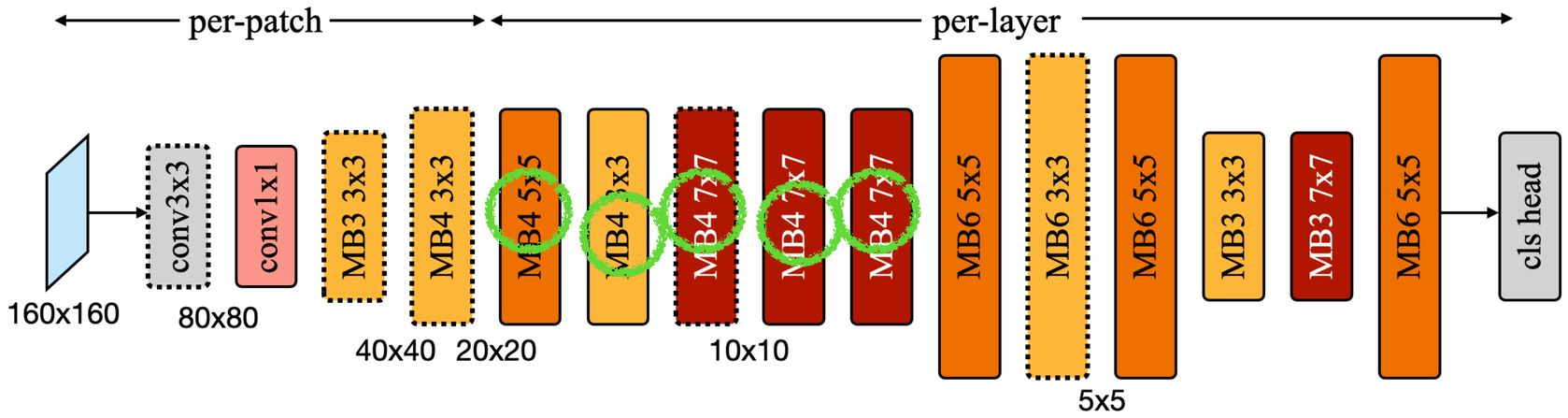


Sample arch from VWW. ***Legend:*** `MB{expansion}_{k_size}x{k_size}`

- Kernel size in per-patch stage is small to reduce spatial overlapping

MCUNetV2: Patch-based Inference

Dissecting MCUNetV2 architecture

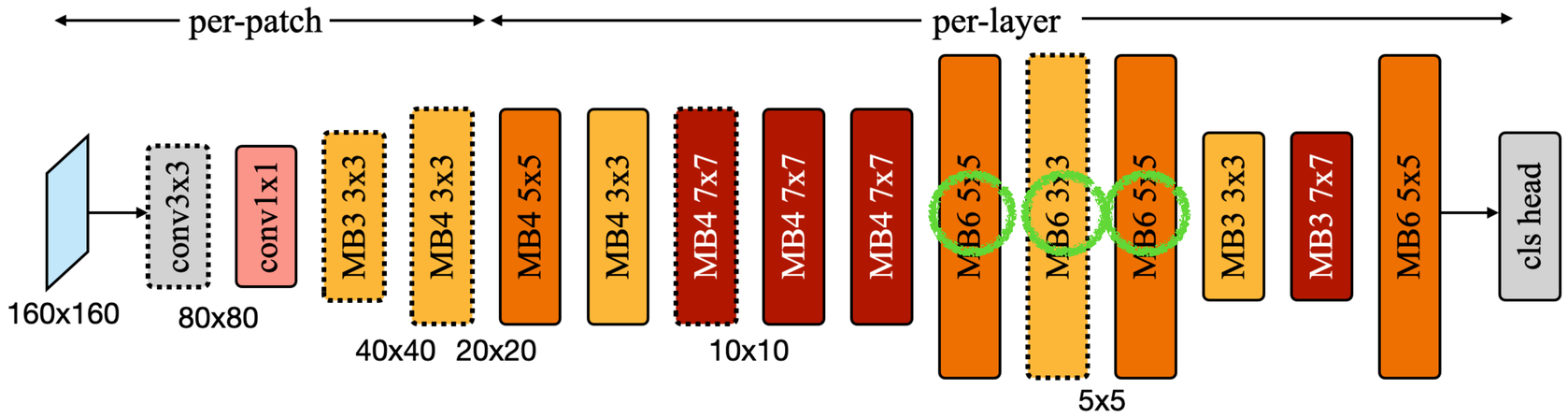


Sample arch from VWW. ***Legend:*** `MB{expansion}_{k_size}x{k_size}`

- Kernel size in per-patch stage is small to reduce spatial overlapping
- Expansion ratio in middle stage is small to reduce peak memory

MCUNetV2: Patch-based Inference

Dissecting MCUNetV2 architecture

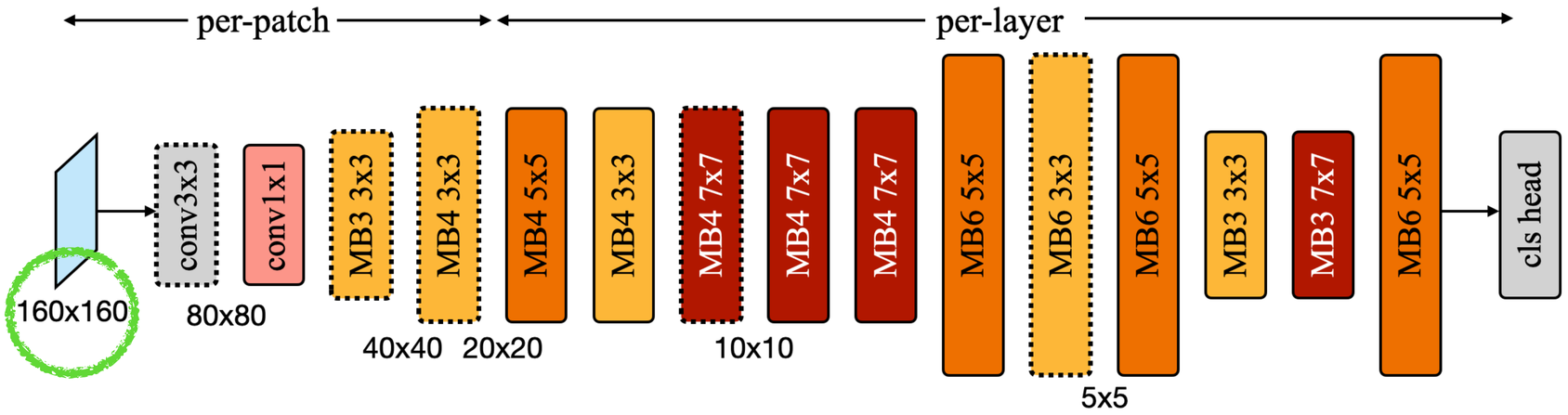


Sample arch from VWW. ***Legend:*** `MB{expansion}_{k_size}x{k_size}`

- Kernel size in per-patch stage is small to reduce spatial overlapping
- Expansion ratio in middle stage is small to reduce peak memory; large in later stage to boost performance.

MCUNetV2: Patch-based Inference

Dissecting MCUNetV2 architecture



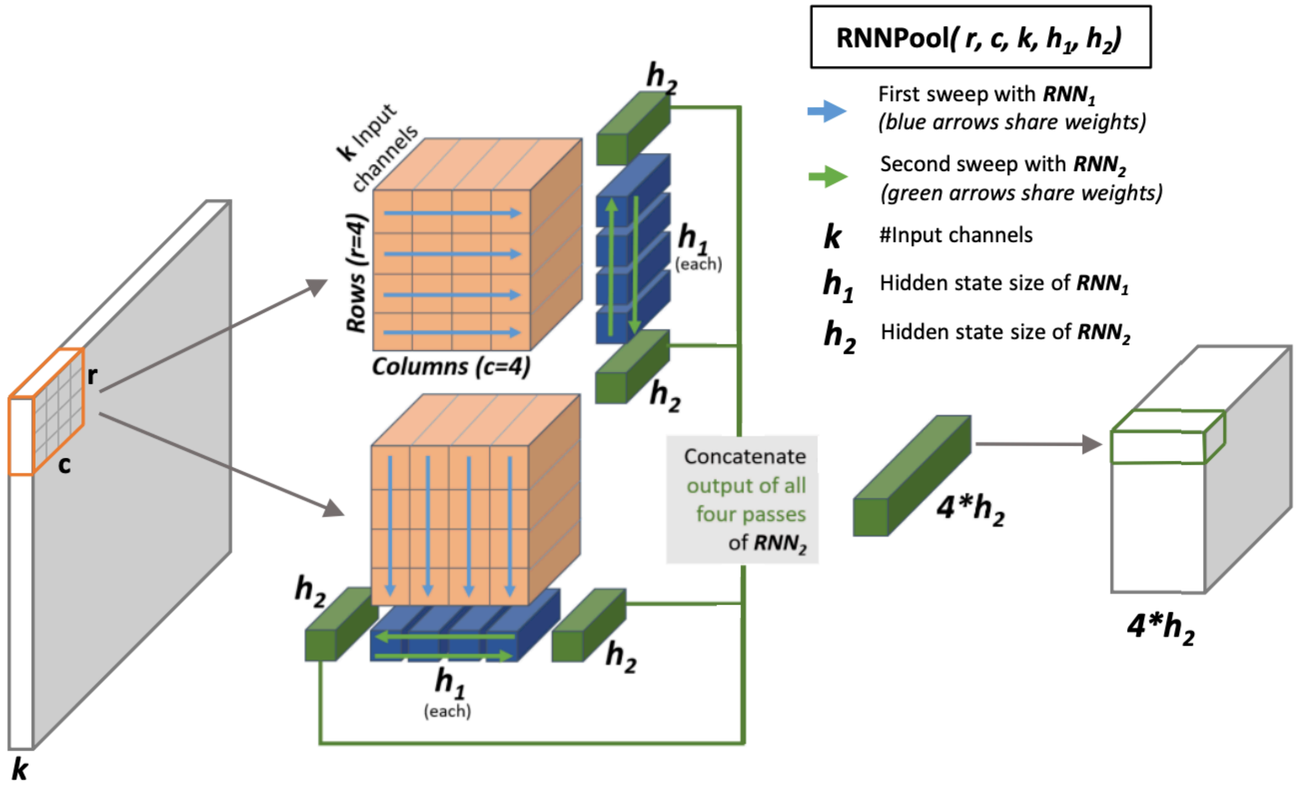
Sample arch from VWW. ***Legend:*** `MB{expansion}_{k_size}x{k_size}`

- Kernel size in per-patch stage is small to reduce spatial overlapping
- Expansion ratio in middle stage is small to reduce peak memory; large in later stage to boost performance.
- Larger input resolution for resolution-sensitive datasets like VWW (MCUNet: 128x128)

RNNPool

Non-linear Pooling for RAM Constrained Inference

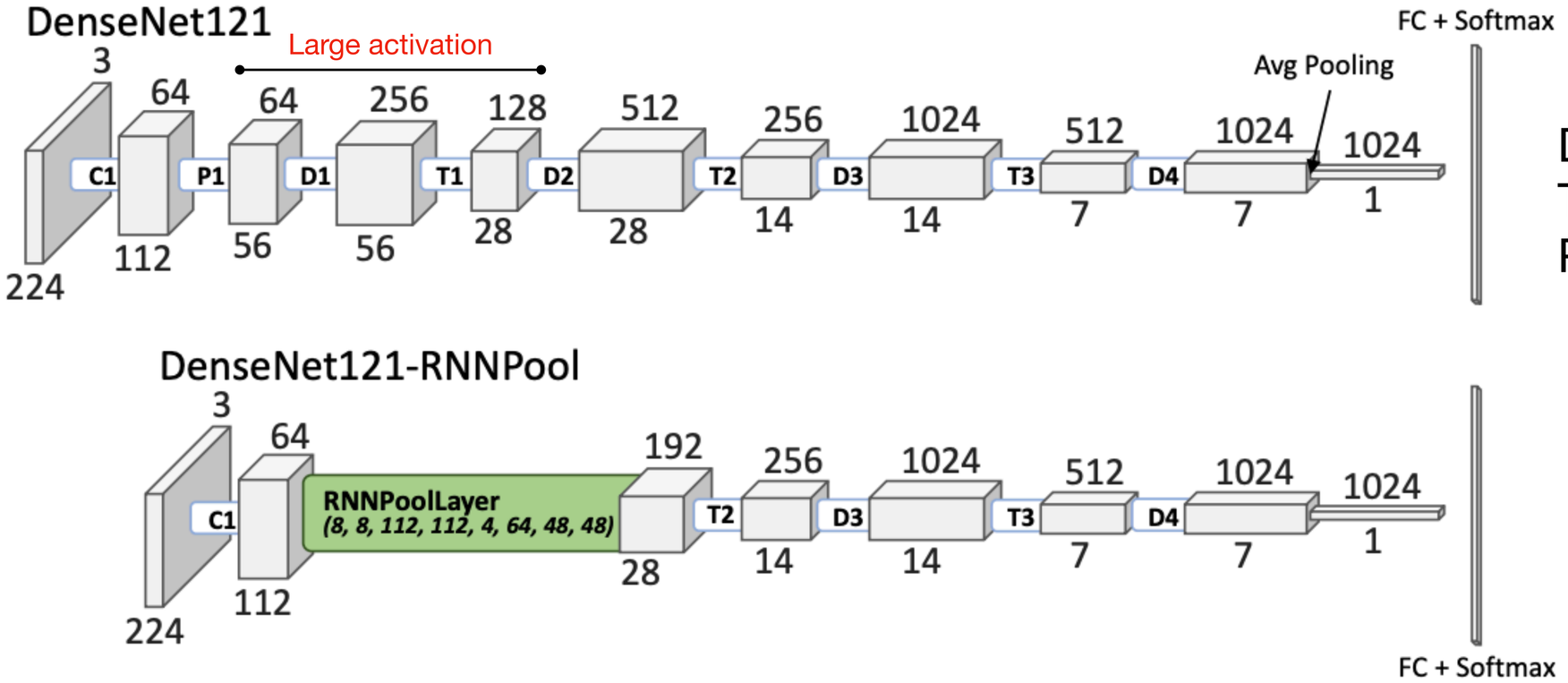
- Typical CNNs have large intermediate activation maps
  - Linear pooling operators or stride convolution are limited to small receptive fields, e.g., < 3x3, due to significant loss of accuracy
- RNN Pool: A more refined aggregation over a large receptive field of the activation map



RNNPool: Efficient Non-linear Pooling for RAM Constrained Inference [Saha *et al.*, NeurIPS 2020]

RNNPool

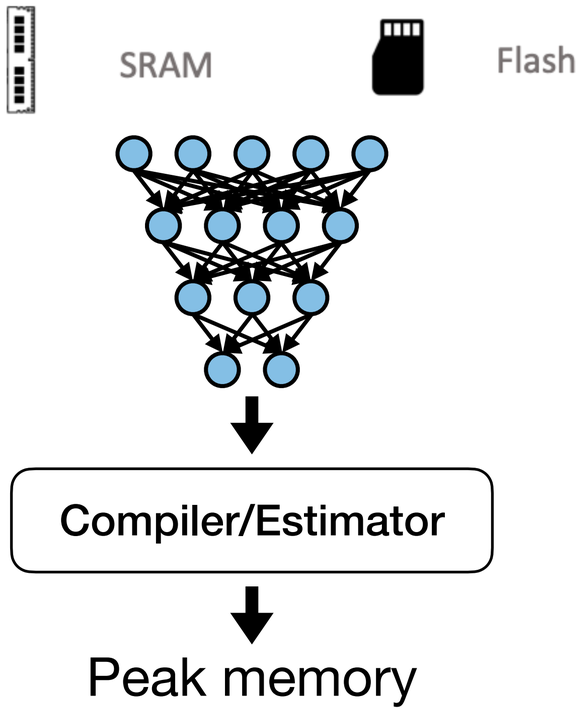
- Replace a sequence of blocks with RNNPool



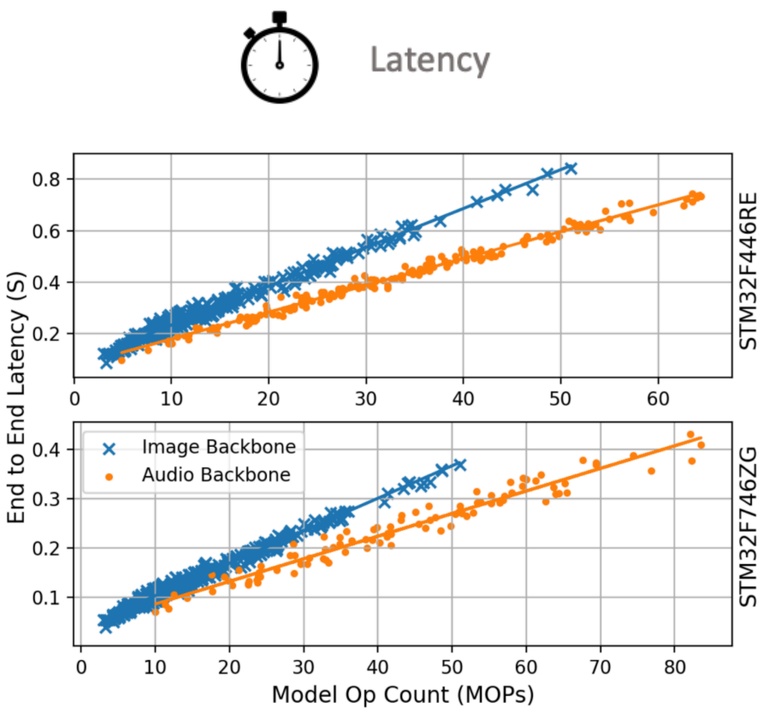
RNNPool: Efficient Non-linear Pooling for RAM Constrained Inference
Source: https://arxiv.org/pdf/2002.11921.pdf

MicroNets

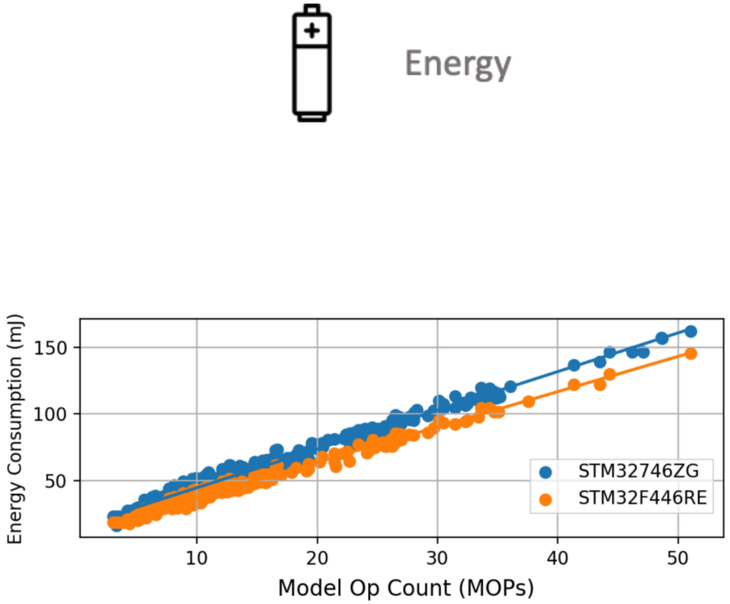
Key: how to estimate the hardware costs of the neural network?



- SRAM and Flash usage can be easily calculated offline for a given model



- OP count is a viable proxy for latency when sampling from the **same superset**



- Energy consumption is only a function of OP count for a given MCU

MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers [Banbury et al., MLSys 2021]

Lecture Plan

Today we will introduce the following:

1. What is tinyML?
2. Understanding the challenges of tinyML
3. Tiny neural network design
4. **Applications**
    1. **Tiny vision**
    2. Tiny audio
    3. Tiny time series/anomaly detection

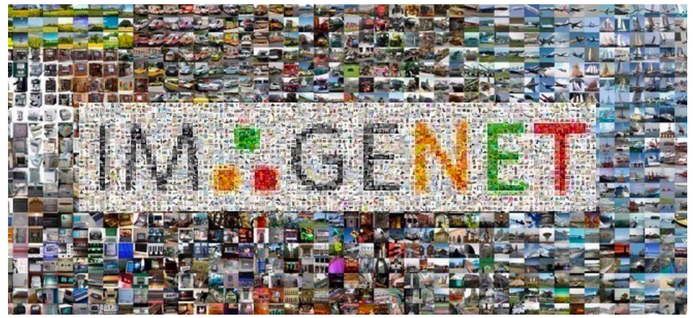
Tiny Image Classification

ImageNet-level image classification

- With techniques like MCUNet, we are able to achieve ImageNet-level image classification performance on microcontrollers

Toy applications

MCUNet



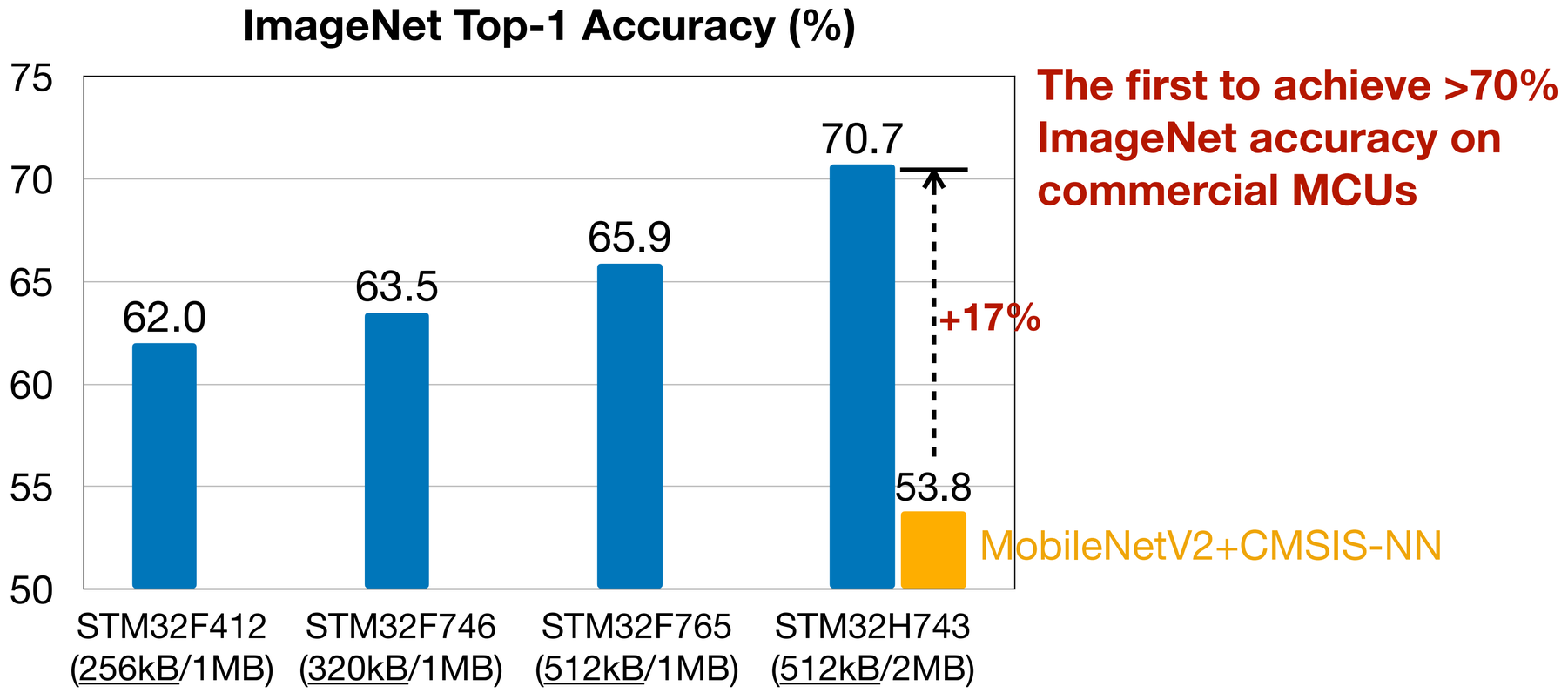
Real-life applications

MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2019]

Tiny Image Classification

ImageNet-level image classification

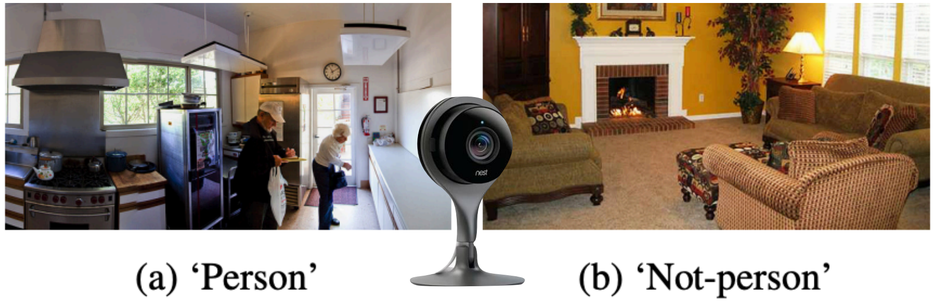
- With techniques like MCUNet, we are able to achieve ImageNet-level image classification performance on microcontrollers (**int4** quantization)



MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2019]

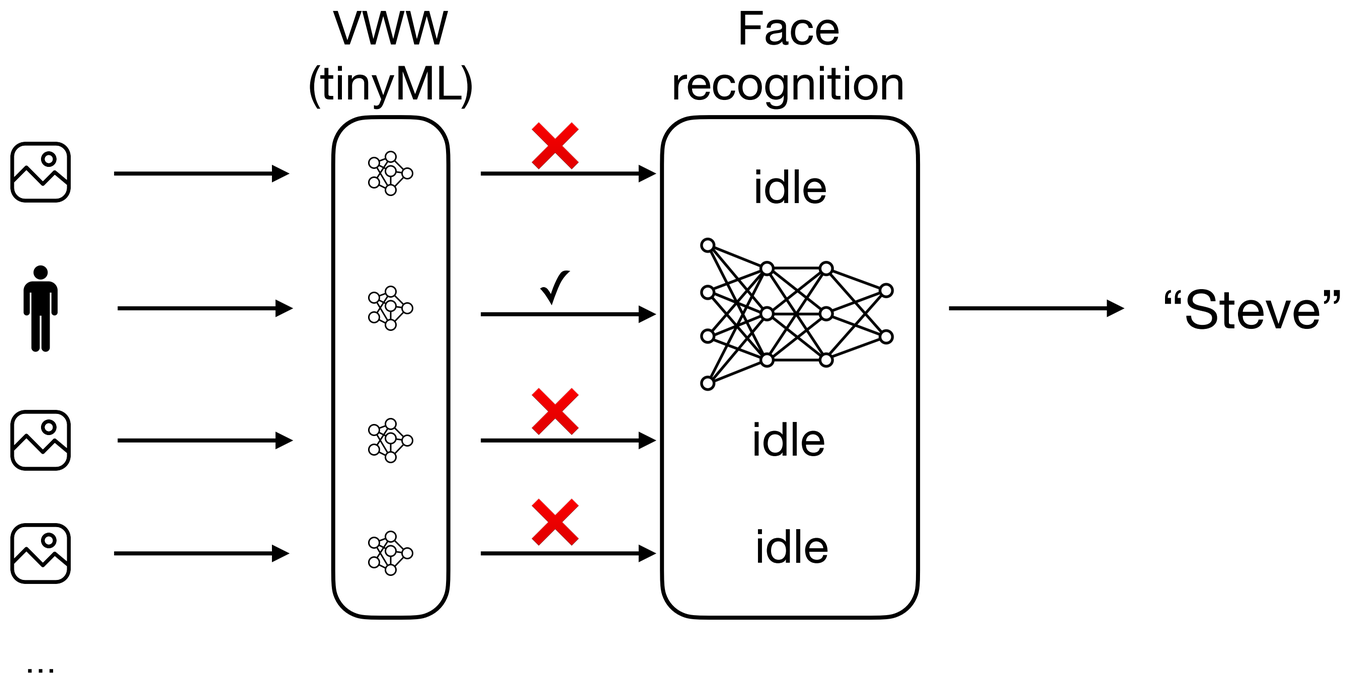
Tiny Image Classification

Tiny vision application: visual wake words



(a) 'Person' (b) 'Not-person'

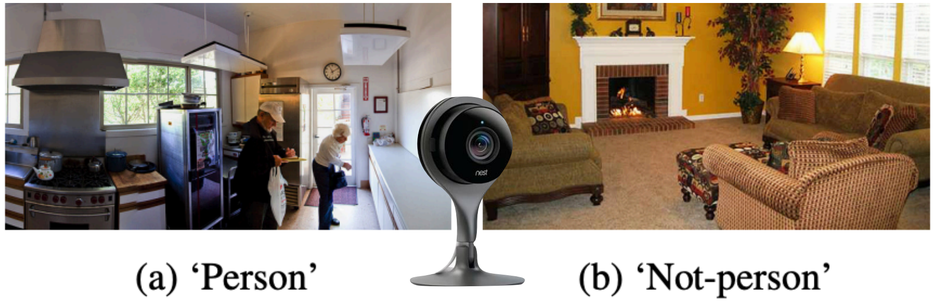
- It can be used to activate the later ML pipeline
  - Vision counterpart of "Hey Siri"/"OK Google"
  - For example, only enable face recognition when a person is in front of the camera, which saves a lot of energy (the face recognition model is much larger than the VWW model)



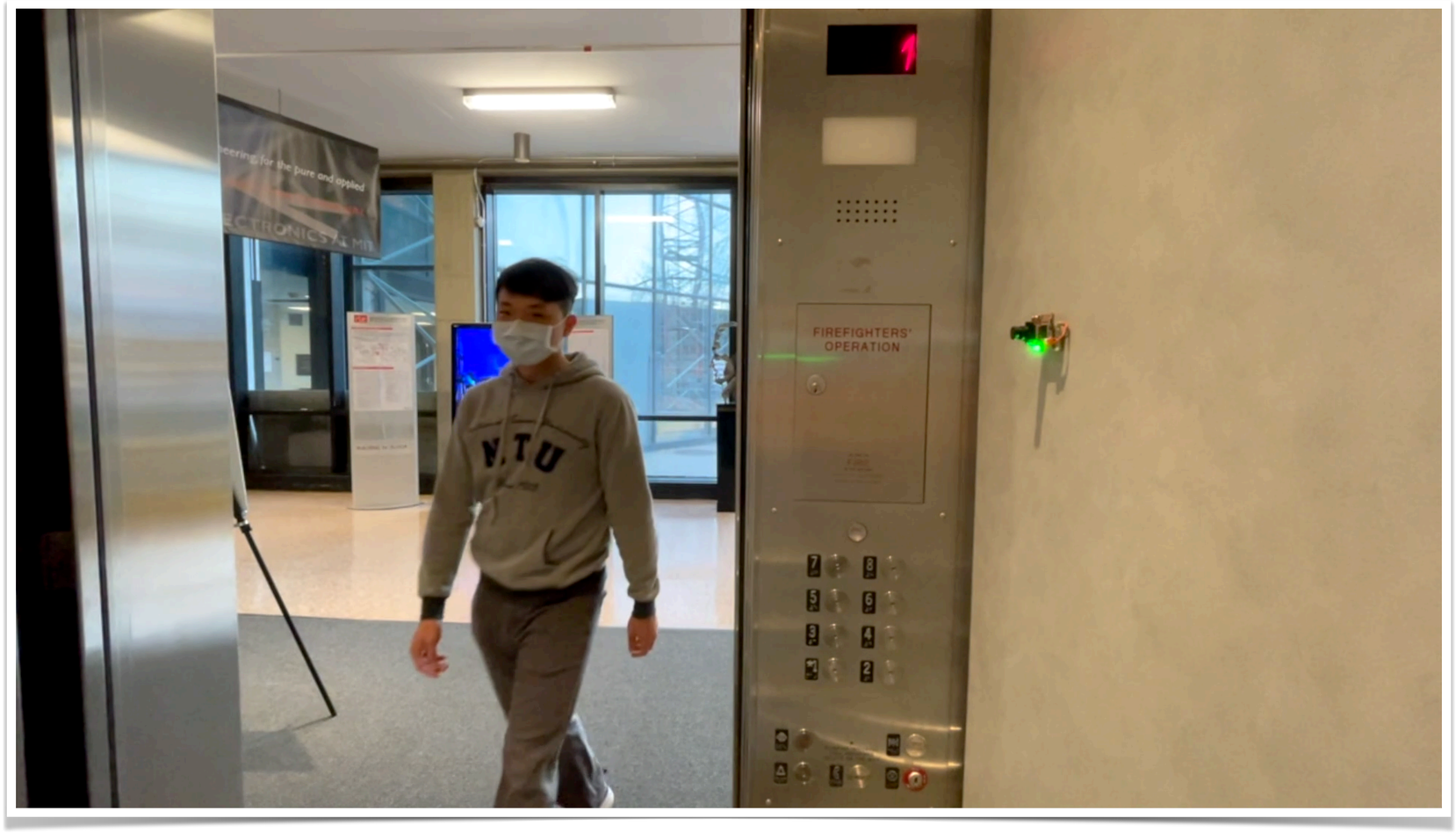
Visual wake words dataset. [Chowdhery *et al.*, arXiv 2019]

Tiny Image Classification

Tiny vision application: visual wake words



(a) 'Person' (b) 'Not-person'

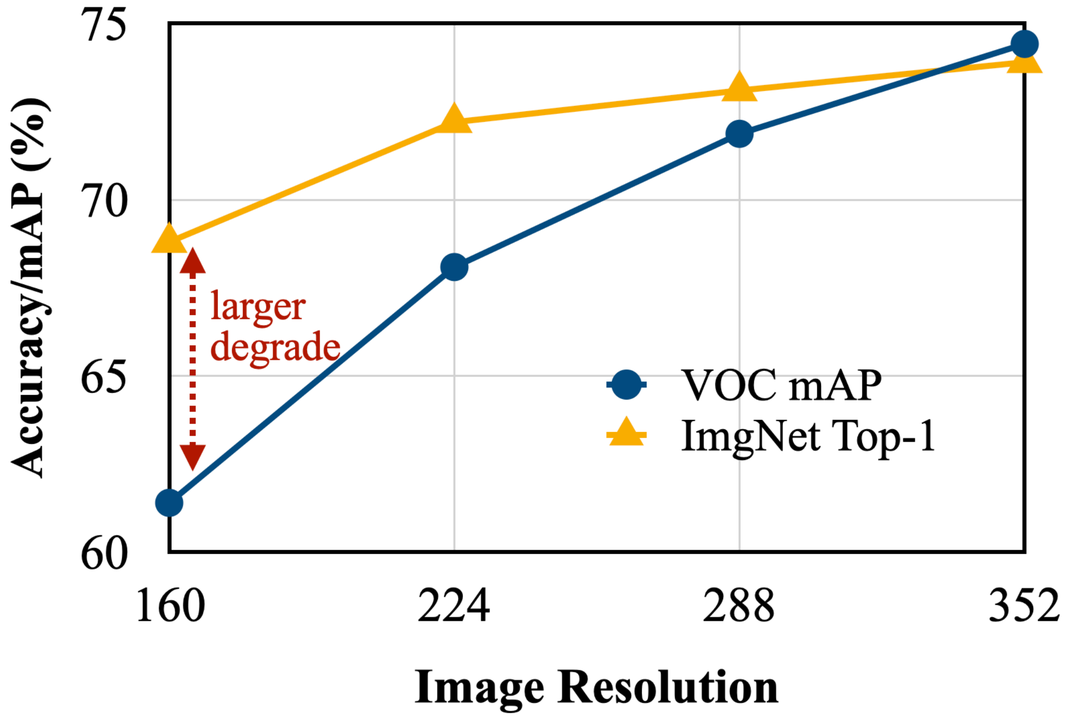


Visual wake words dataset. [Chowdhery *et al.*, arXiv 2019]

Tiny Object Detection

Patch-based method allows for a larger input resolution

- Object detection is more sensitive to input resolution since we need to make a denser prediction.
- Patched-based inference can fit a larger resolution by breaking the memory bottleneck.



MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

Tiny Object Detection

Patch-based method allows for a larger input resolution

- We can further enable object detection on microcontrollers

Face/mask detection

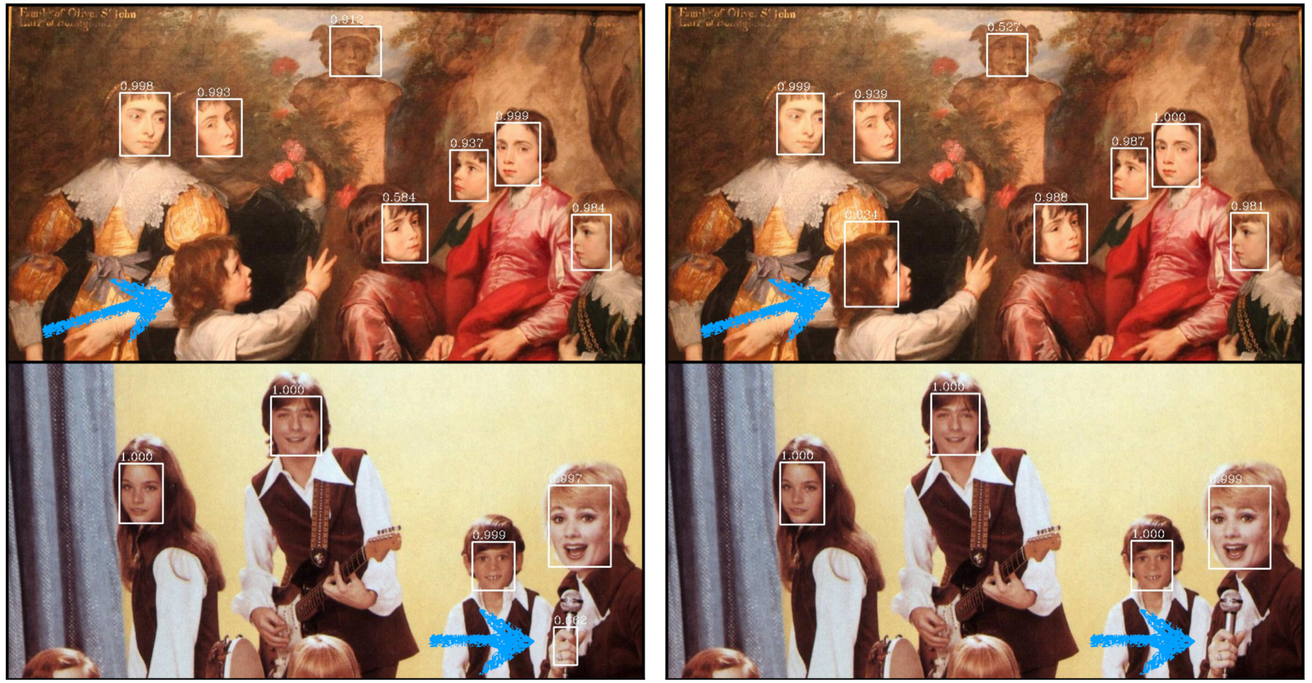
Person detection

MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

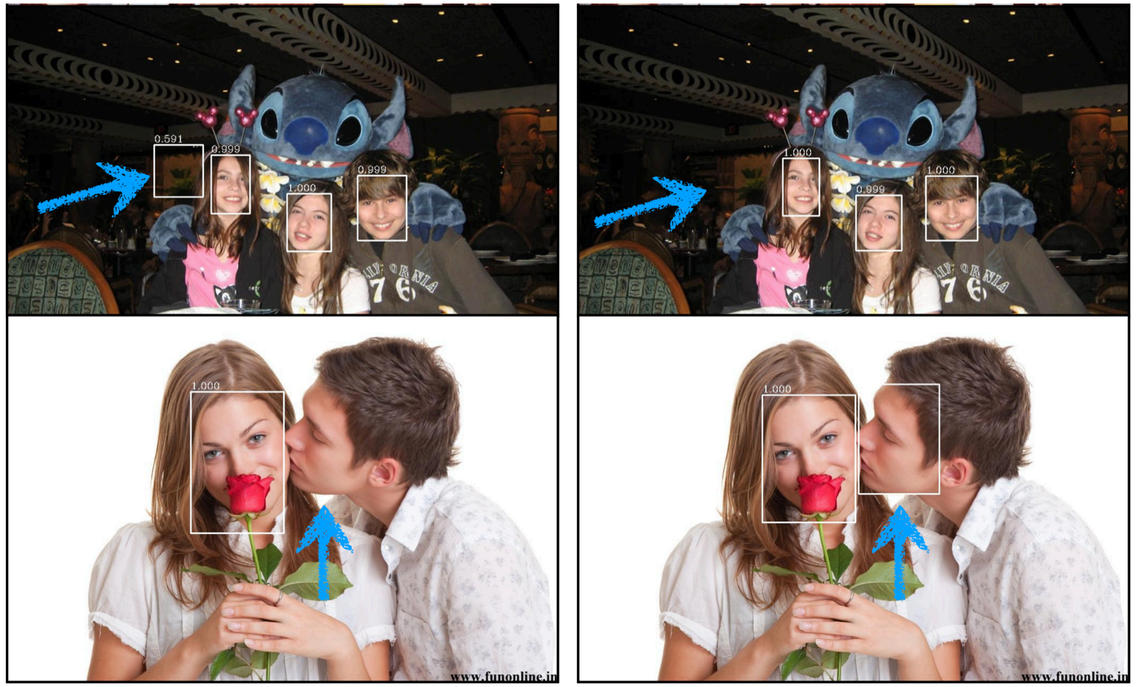
Tiny Object Detection

Patch-based method allows for a larger input resolution

- Patch-based inference obtains better object detection results compared to existing work



(a) RNNPool-Face-Quant (b) MCUNetV2



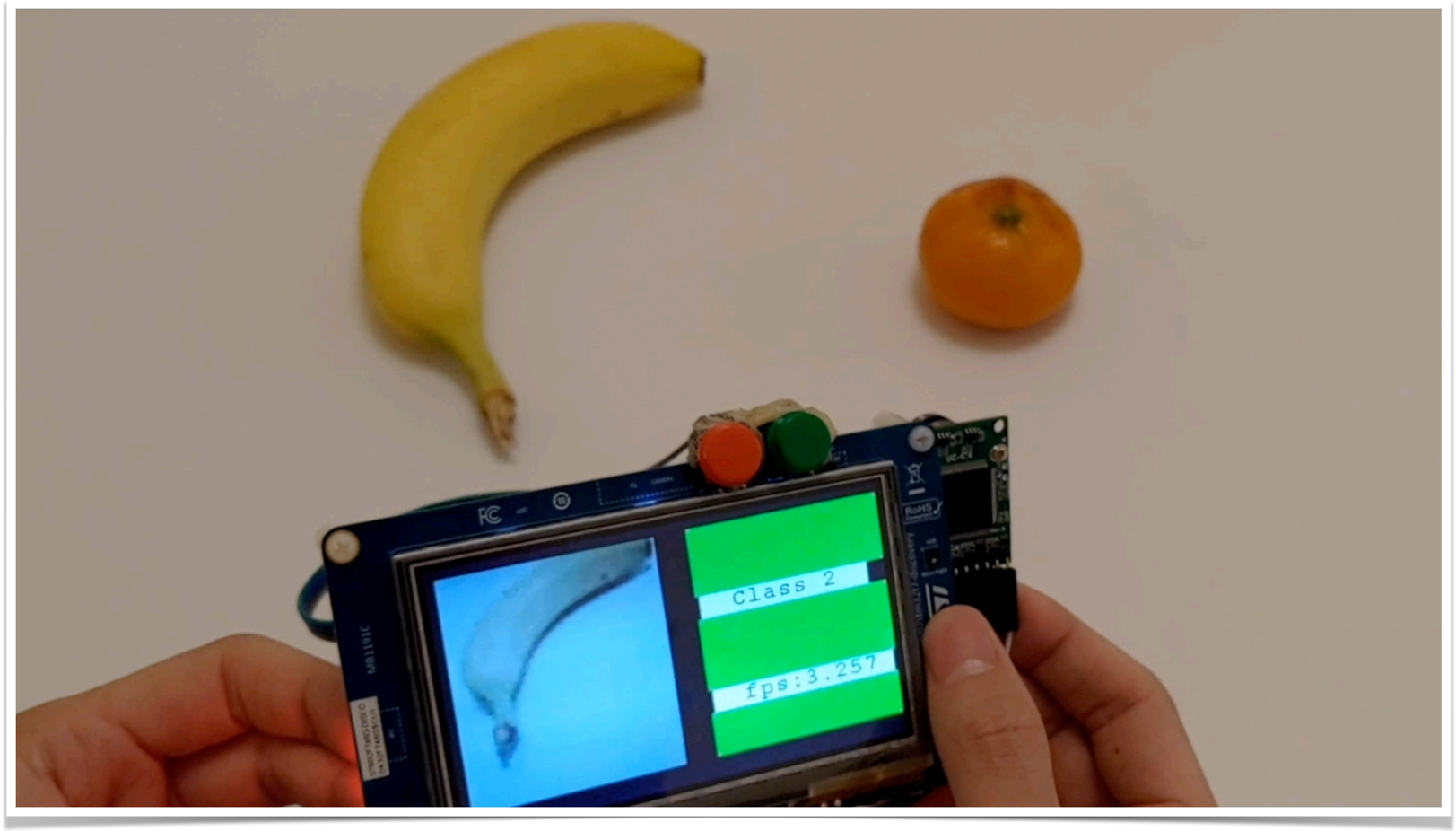
(a) RNNPool-Face-Quant (b) MCUNetV2

MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]

Tiny On-device Training

We can even learn new tasks on a MCU!

- We will leave it to the future lectures



On-Device Training Under 256KB Memory [Lin et al., NeurIPS 2022]

Lecture Plan

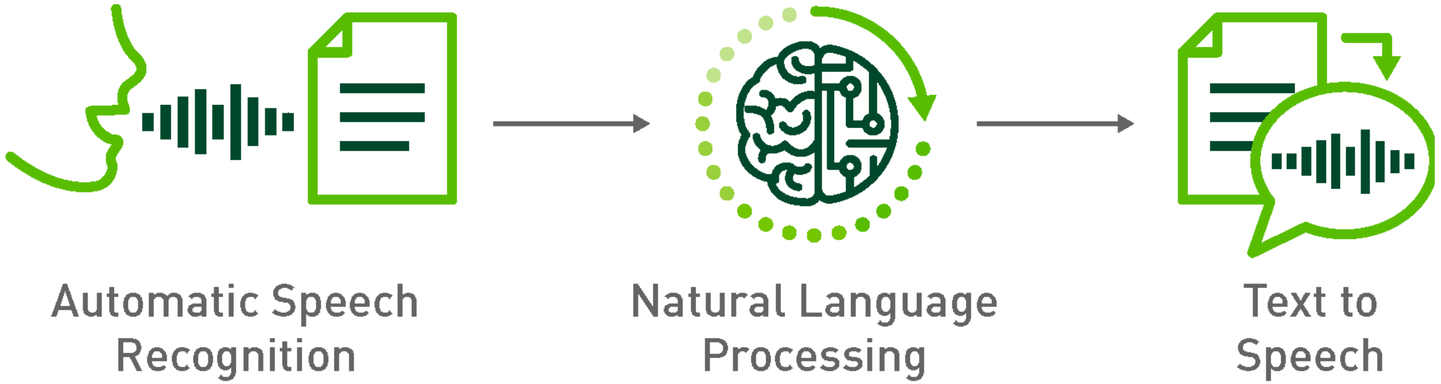
Today we will introduce the following:

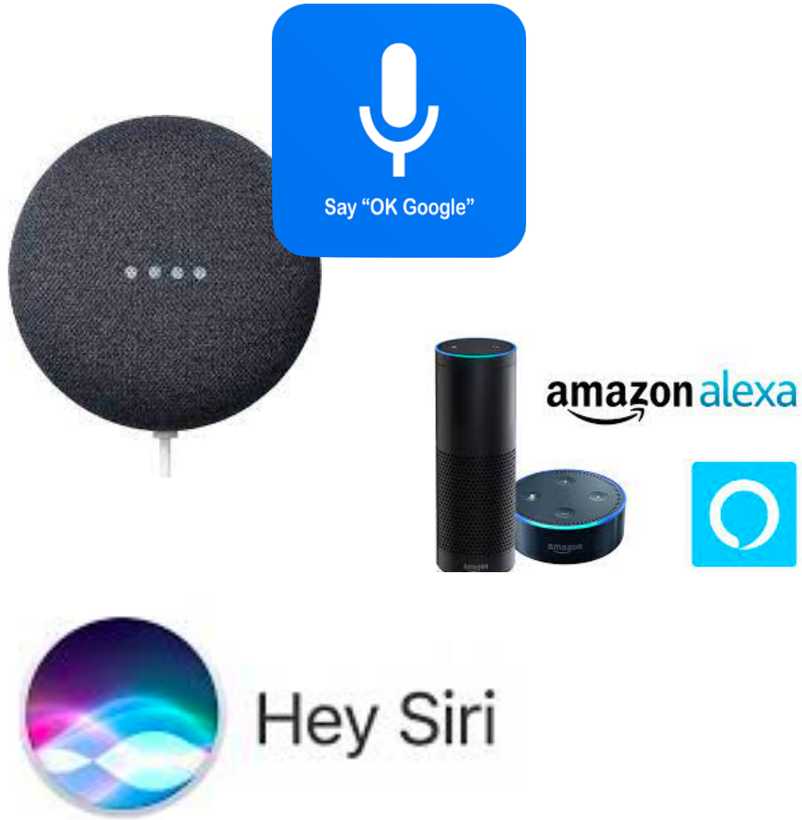
1. What is tinyML?
2. Understanding the challenges of tinyML
3. Tiny neural network design
4. **Applications**
    1. Tiny vision
    2. **Tiny audio**
    3. Tiny time series/anomaly detection

Audio Deep Learning Applications on Tiny Devices

Common applications in our daily life

- Audio keywords/Keyword spotting
- Speech recognition
- Noise canceling
- …



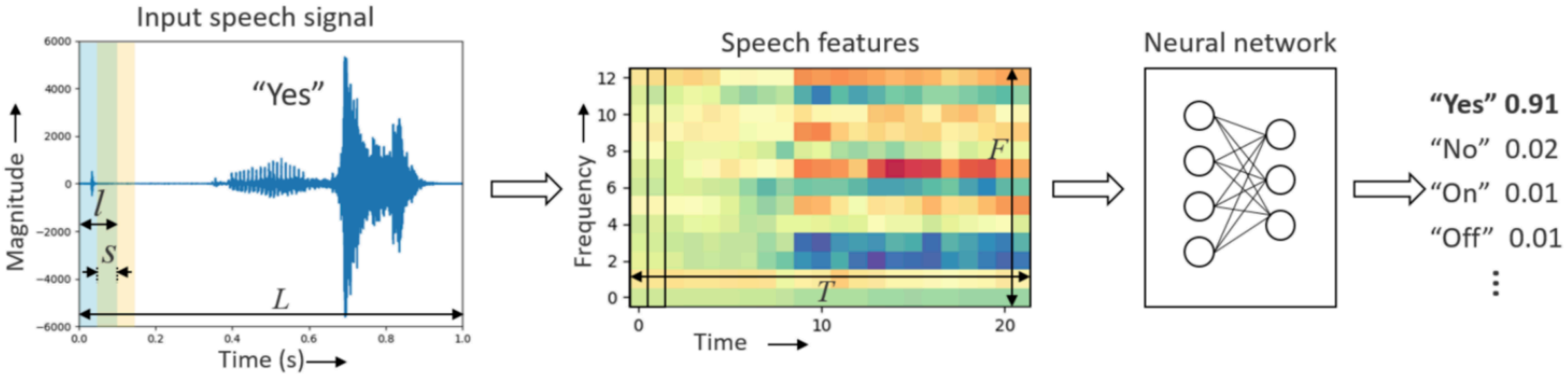


Convolutional Neural Networks for Small-footprint Keyword Spotting, [Sainath et al., Google Research 2017]

Case Study: Keyword Spotting

General pipeline of keyword spotting

- Input: Overlapping frames of length l with a stride s, giving a total of $T = (L - l)/s + 1$ frames.
- Human-engineered features: Translating time-domain signal into a set of frequency-domain spectral coefficients.
- Neural network: Generating the probabilities for the output classes.

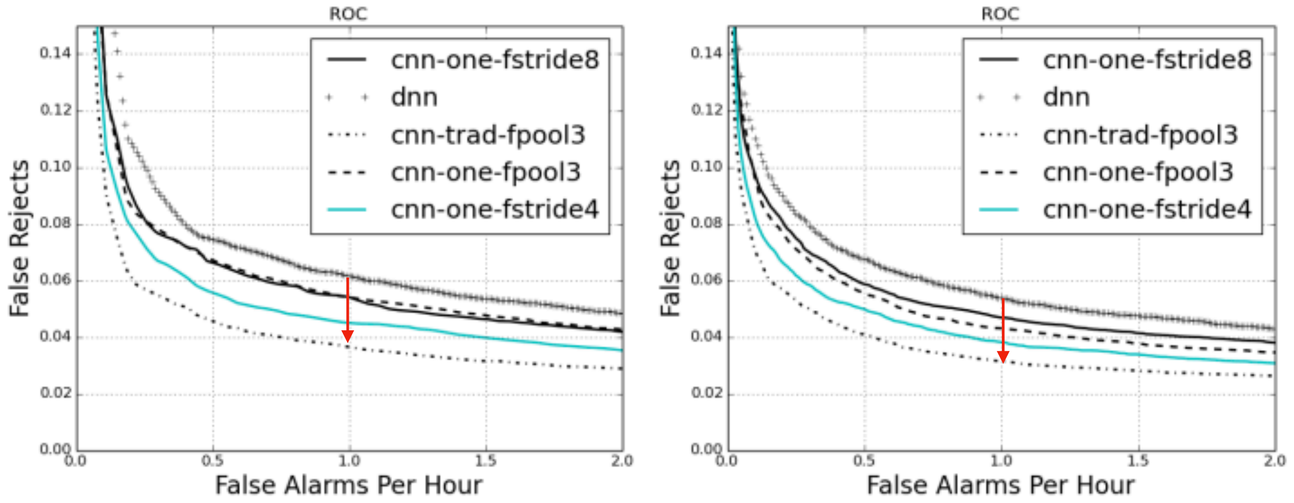


Hello Edge: Keyword Spotting on Microcontrollers, [Zhang1 et al., arXiv 2018]

CNN for Small-footprint Keyword Spotting

CNN architectures can outperform DNNs

- Spectral representations of speech have strong correlations in time and frequency, which is ignored by DNNs
- DNNs are not explicitly designed to model translational variance within speech signals, which can exist due to different speaking styles



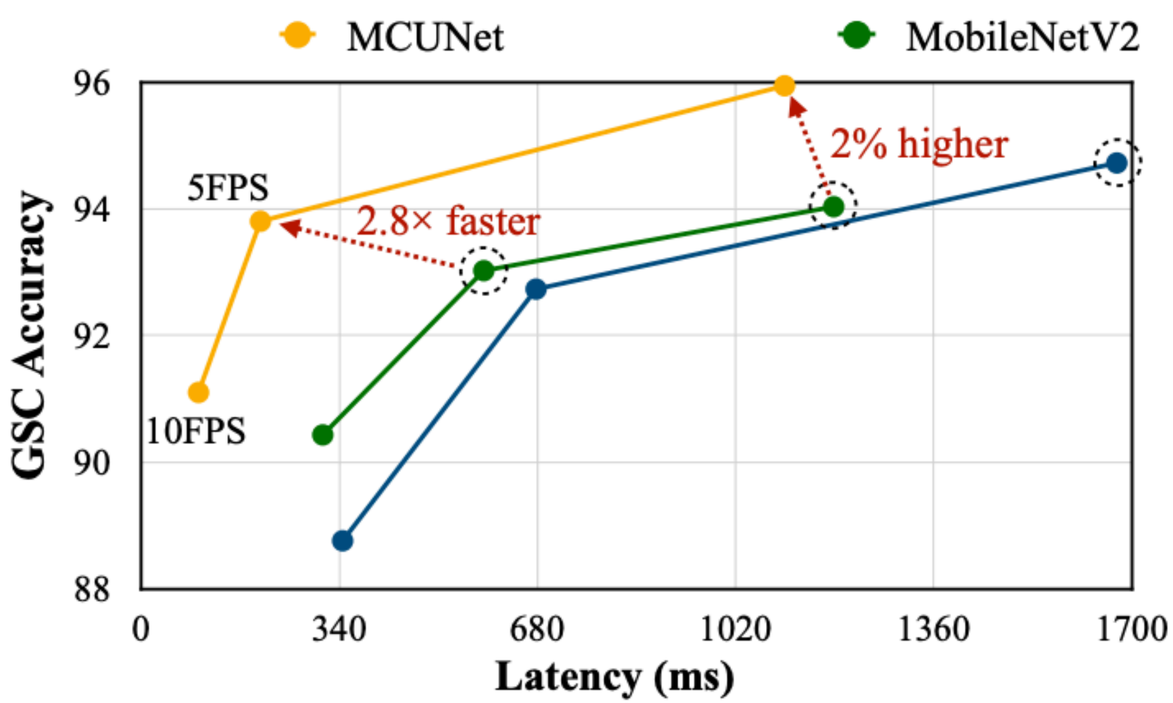
(a) Results on Clean (b) Results on Noisy

DNN vs. CNN, Matching Multiplies

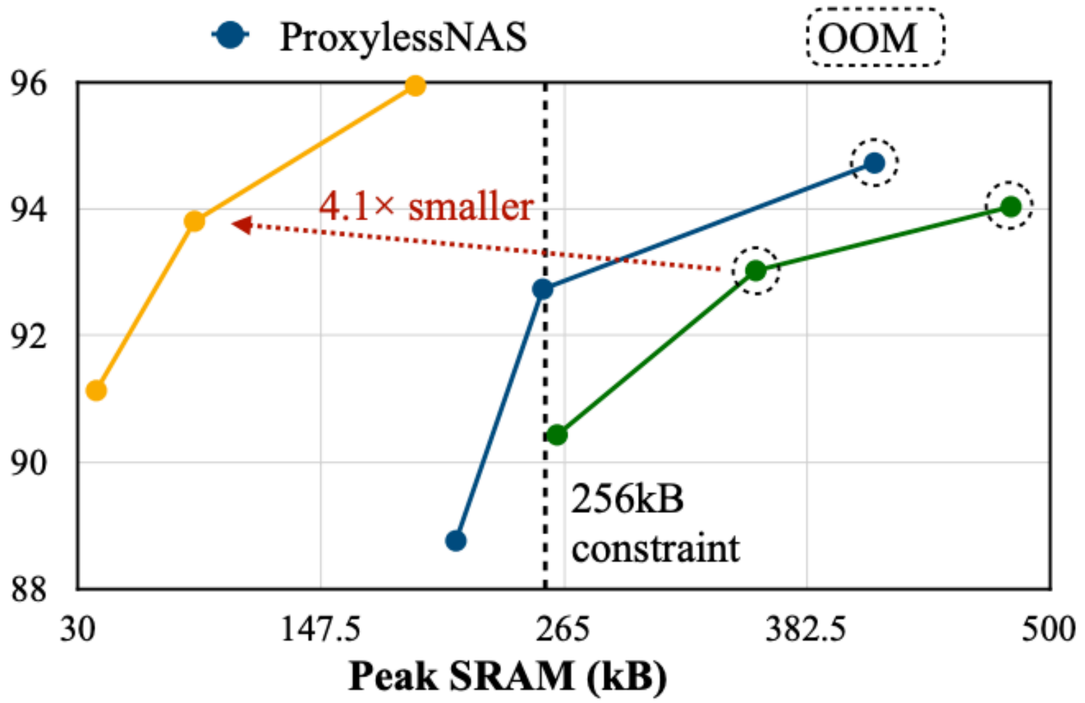
Convolutional Neural Networks for Small-footprint Keyword Spotting, [Sainath et al., Google Research 2017]

MCUNet: Introduce Hardware Characterization

Improve efficiency with software-hardware co-design on speech commands



(a) Trade-off: accuracy *vs.* measured latency



(b) Trade-off: accuracy *vs.* peak memory

MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2019]

Lecture Plan

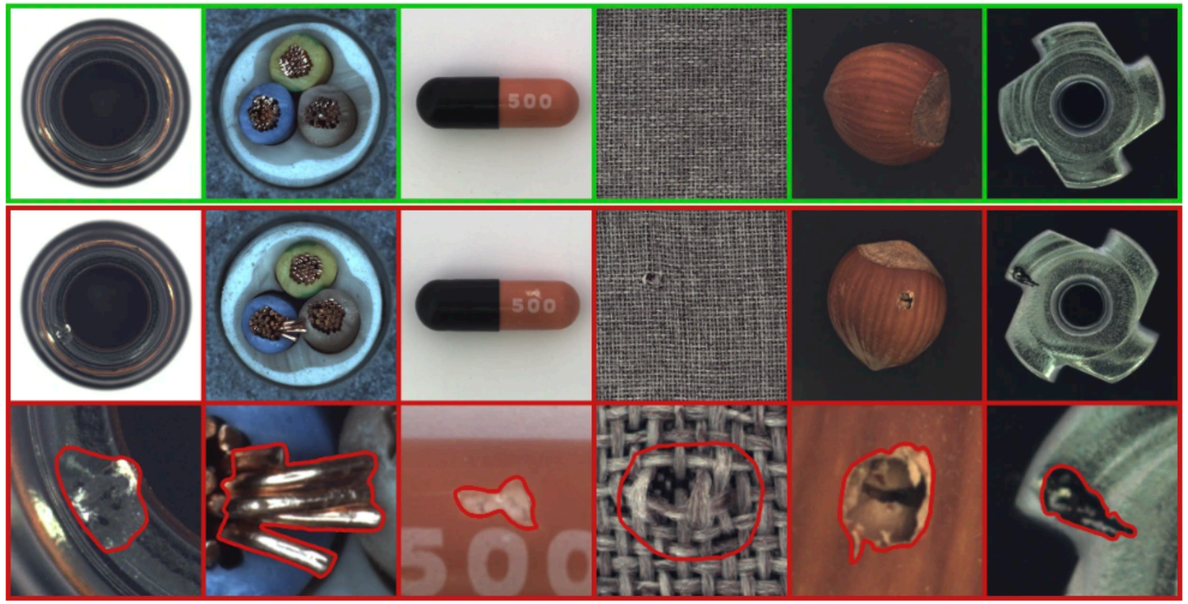
Today we will introduce the following:

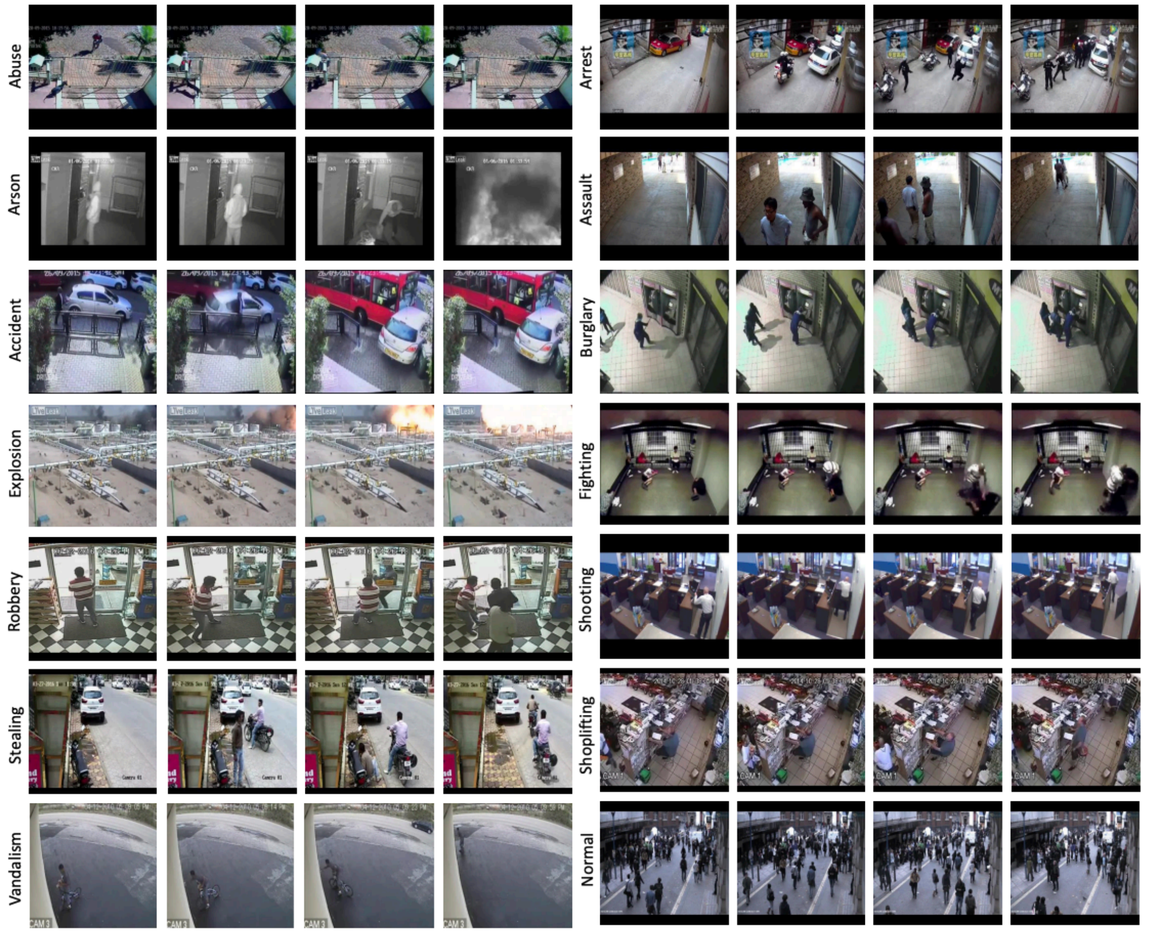
1. What is tinyML?
2. Understanding the challenges of tinyML
3. Tiny neural network design
4. **Applications**
    1. Tiny vision
    2. Tiny audio
    3. **Tiny time series/anomaly detection**

Anomaly Detection on Tiny Devices

Application in many domains

- Video surveillance
- Health care
- Prevent fraud, adversary attacks
- …



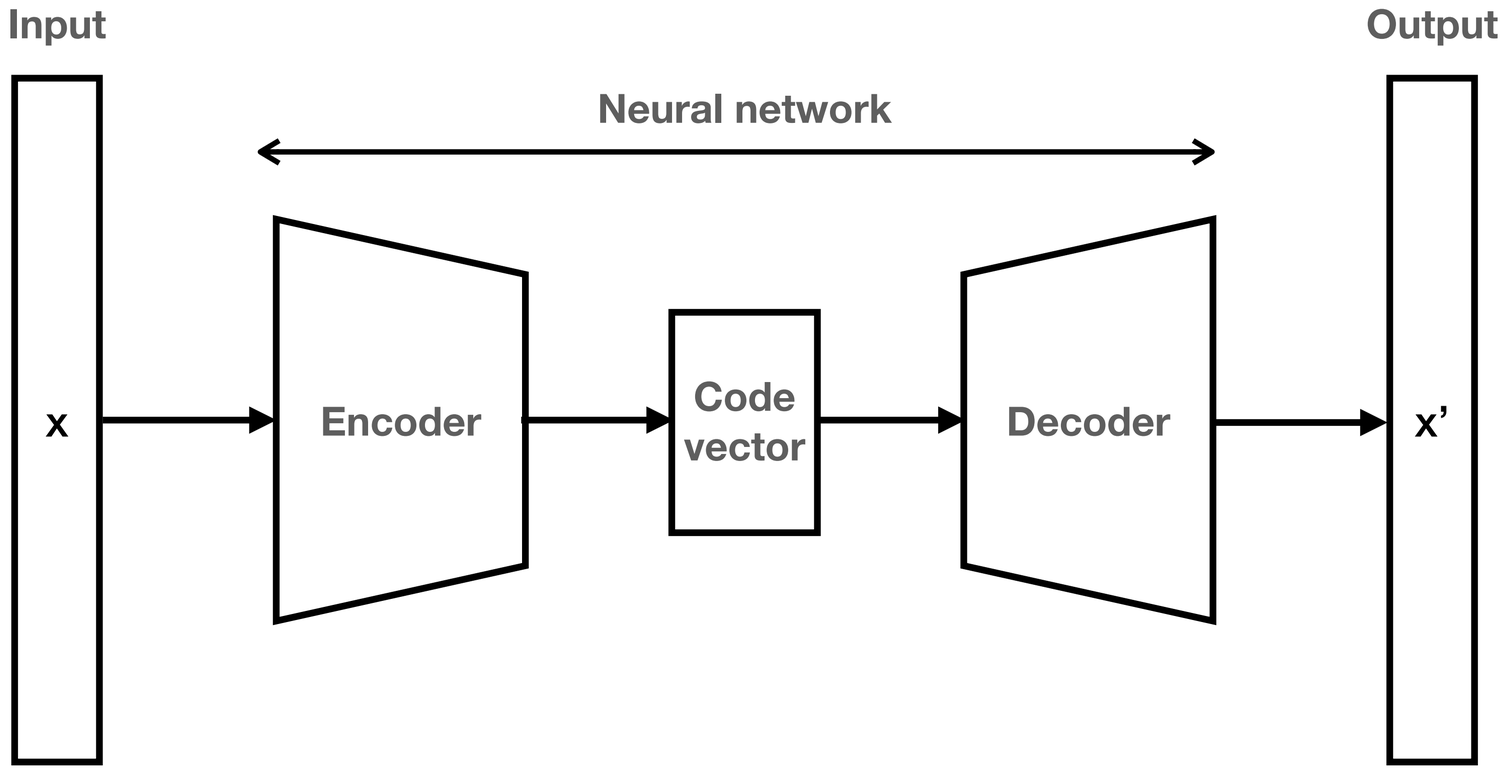


MVTec dataset: https://www.mvtec.com/company/research/datasets/mvtec-ad
Real-world Anomaly Detection in Surveillance Videos: https://arxiv.org/pdf/1801.04264.pdf

Detect Anomaly with Autoencoders

Introducing autoencoders

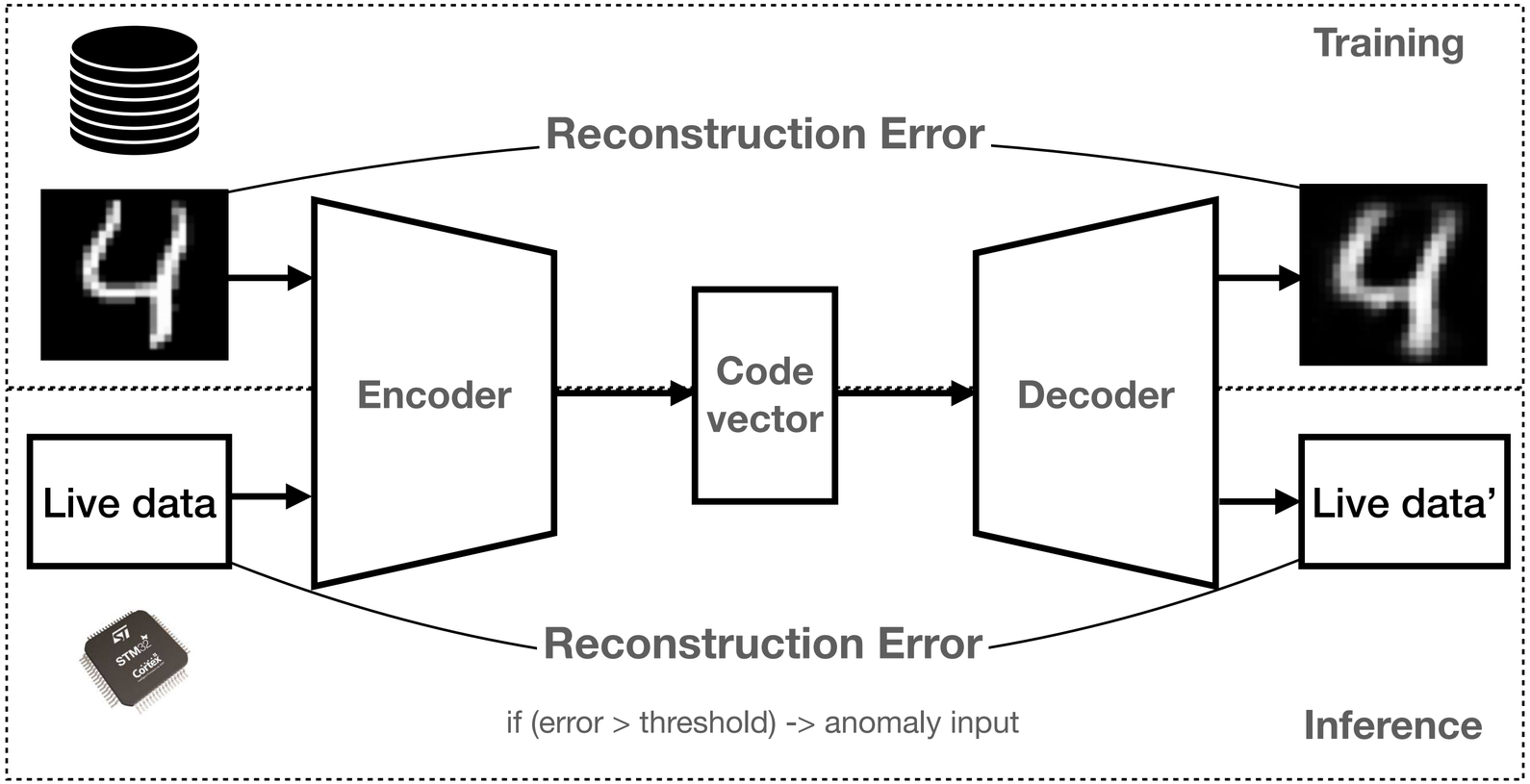
- An autoencoder is a neural network that predicts its **input** (ideally x' = x)
  - Encoder: Compress the input into a lower-dimensional code vector
  - Code vector: Abstraction of the input
  - Decoder: Reconstruct the output from the code vector



Detect Anomaly with Autoencoders

Introducing autoencoders

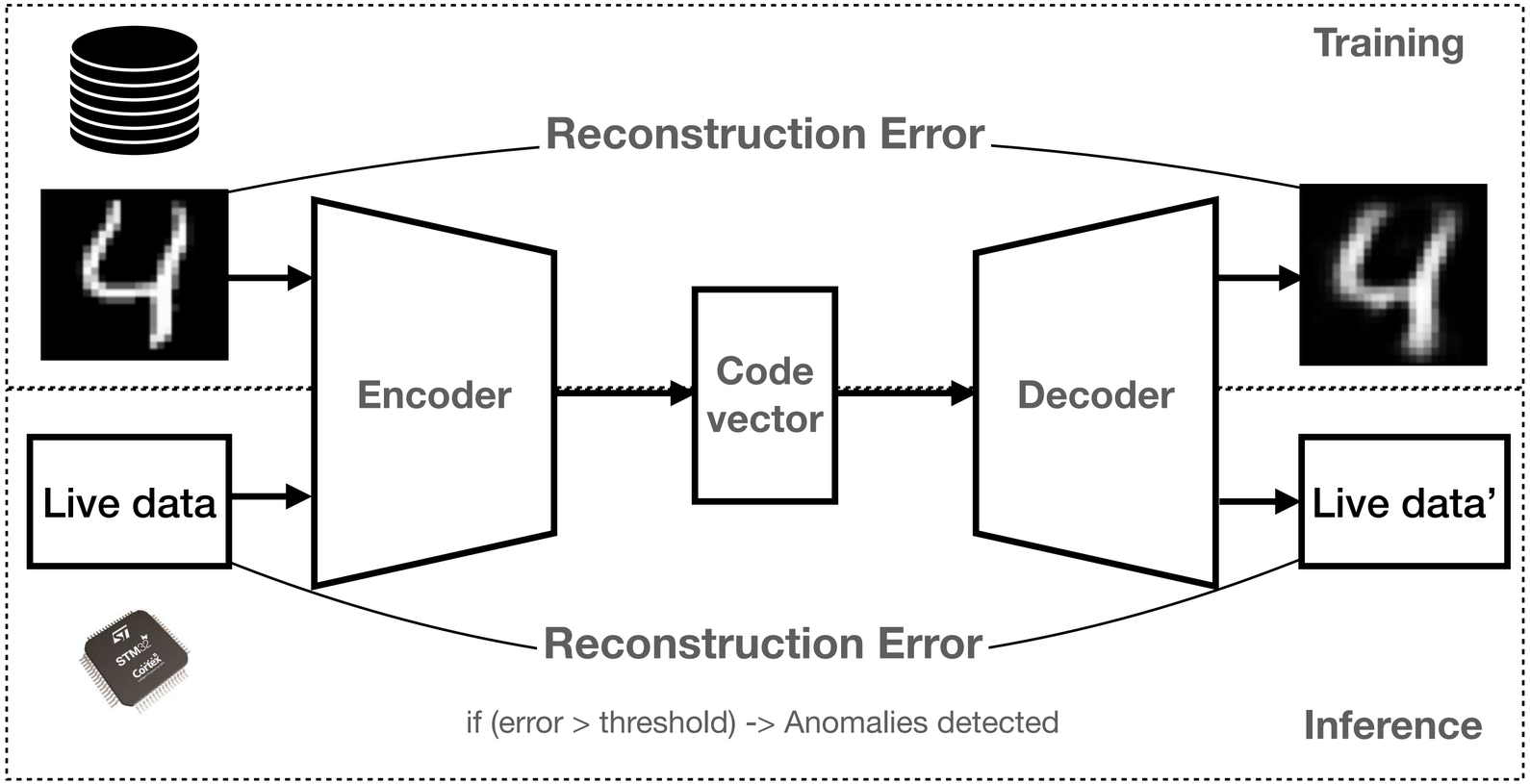
- Training : Minimizing the reconstruction error
- Inference: Detect anomaly inputs with live data



Detect Anomaly with Autoencoders

Properties of autoencoders

- Unsupervised: We don't need labels for training.
- Data-specific: They can only meaningfully compress data similar to the training dataset.
- Lossy: The output will not be the same as the input.



Example: Fan Anomaly Detection

Detect fan anomaly with Arduino Nano 33 BLE Sense

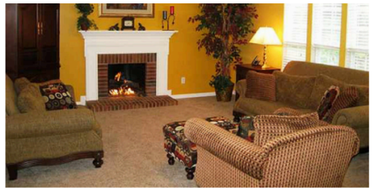
- Hardware constraints
  - SRAM: 256KB
  - FLASH: 1MB
  - CPU: Cortex-M4@64MHz

- Common anomaly detection approaches include
  - K-means
  - Autoencoders
  - GMMs

Summary of Today's Lecture

- In this lecture, we introduce:
  - What is TinyML;
  - Challenges of TinyML;
  - Neural networks design for tiny devices;
  - Applications in our daily life;
    - Vision, audio, time-series/anomaly detection
- We'll have paper presentation in the next lecture
- From next week, we will switch gear from inference to training, and introduce techniques to make training faster.

(a) 'Person'



(b) 'Not-person'