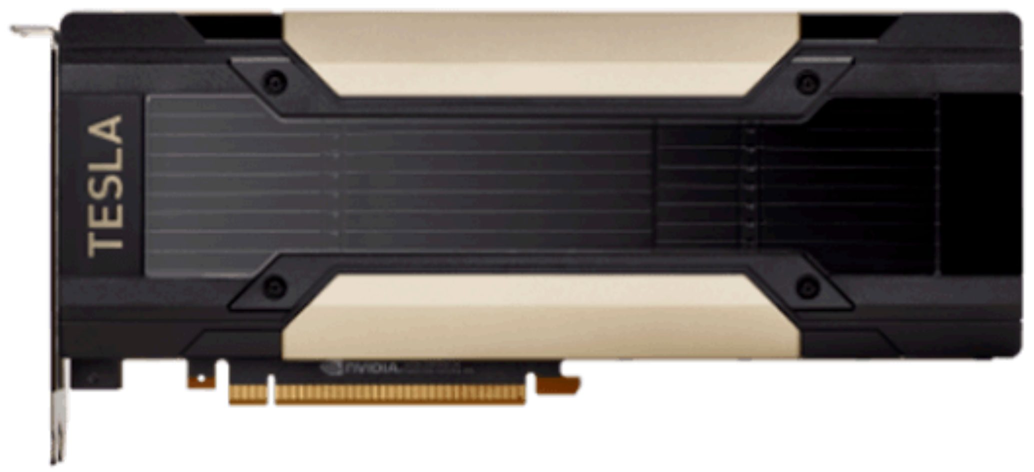
MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning

Ji Lin[1], Wei-Ming Chen[1], Han Cai[1], Chuang Gan[2], Song Han[1]

[1] Massachusetts Institute of Technology
[2] MIT-IBM Watson AI Lab

Deep Learning Going “Tiny”

Deep Learning Going "Tiny"



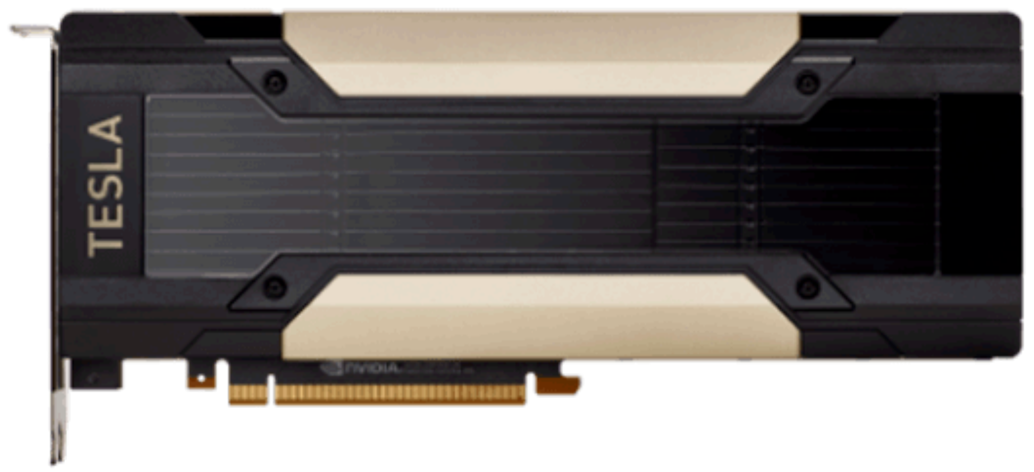
Cloud AI

Data centers
Expensive
Privacy issue

Mobile AI

Smartphones
Accessible
Process locally

Deep Learning Going “Tiny”



→

→

?

Cloud AI

Data centers
Expensive
Privacy issue

Mobile AI

Smartphones
Accessible
Process locally

Can we go even smaller?

Can we go even smaller?

- The future belongs to Tiny AI.

Can we go even smaller?

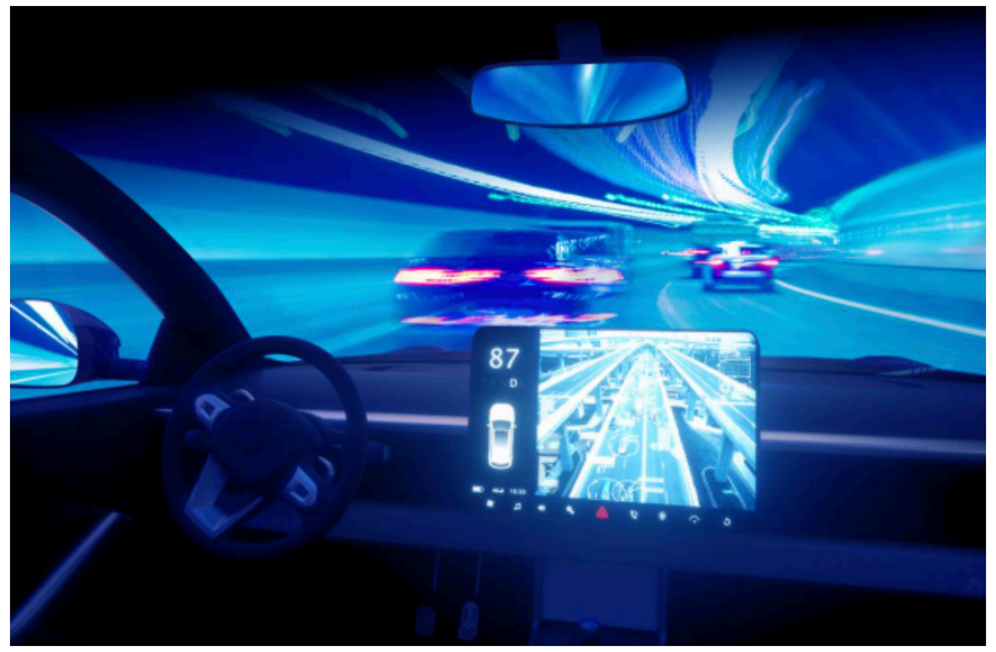
- The future belongs to Tiny AI.
- Billions of <u>IoT devices</u> around the world based on <u>microcontrollers</u>

Smart Home

Smart Manufacturing

Personalized Healthcare

Driving Assist



Can we go even smaller?

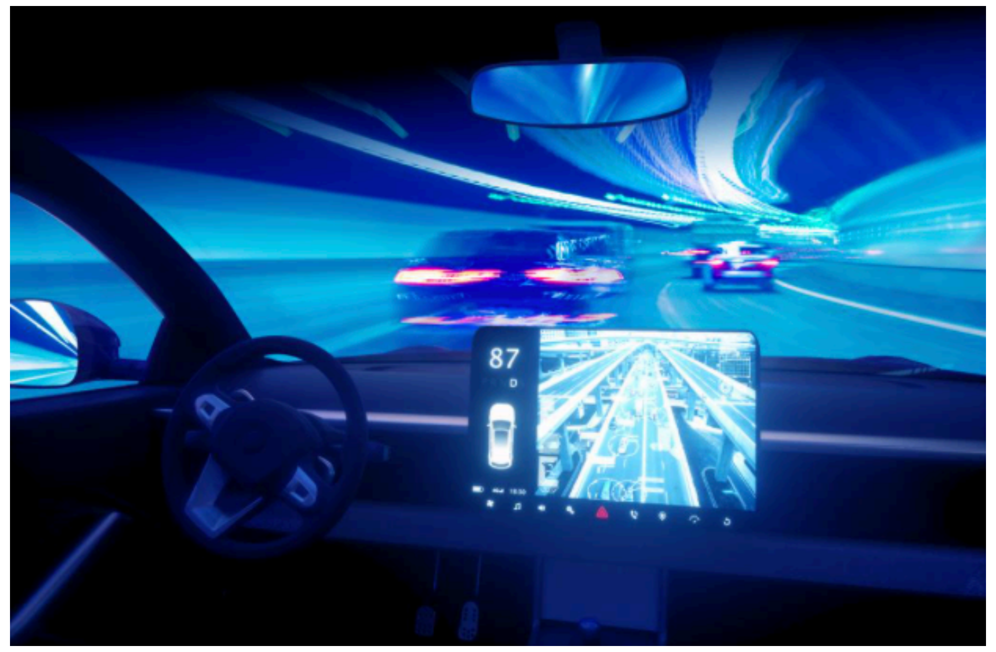
- The future belongs to Tiny AI.
- Billions of <u>IoT devices</u> around the world based on <u>microcontrollers</u>
- **Low-cost**: low-income people can have access. Democratize AI.

Smart Home

Smart Manufacturing

Personalized Healthcare

Driving Assist



Can we go even smaller?

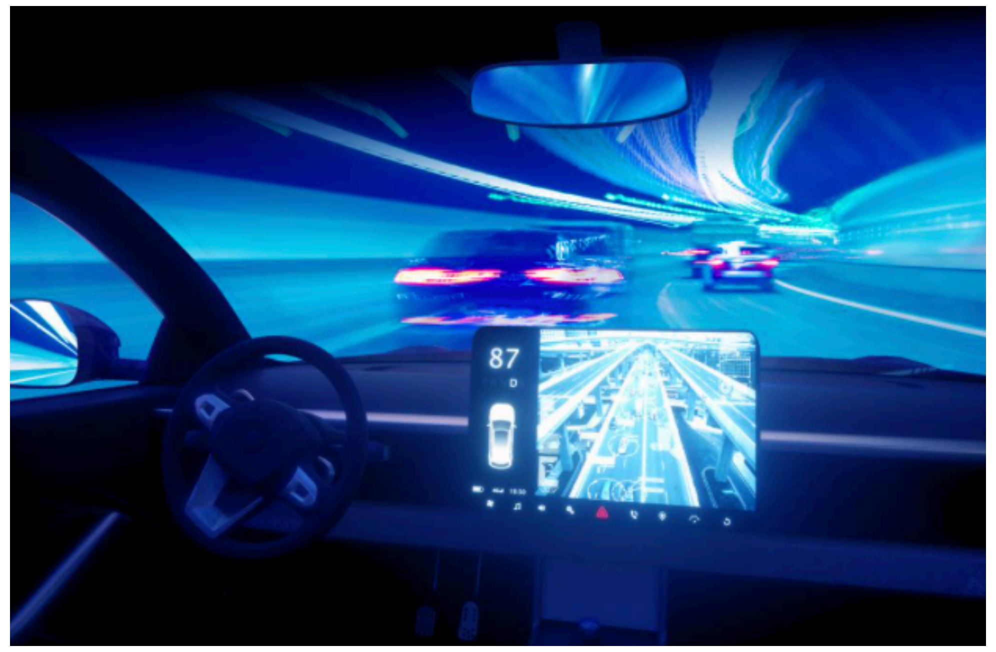
- The future belongs to Tiny AI.
- Billions of <u>IoT devices</u> around the world based on <u>microcontrollers</u>
- **Low-cost**: low-income people can have access. Democratize AI.
- **Low-power**: reduce carbon. Green AI.

Smart Home

Smart Manufacturing

Personalized Healthcare

Driving Assist

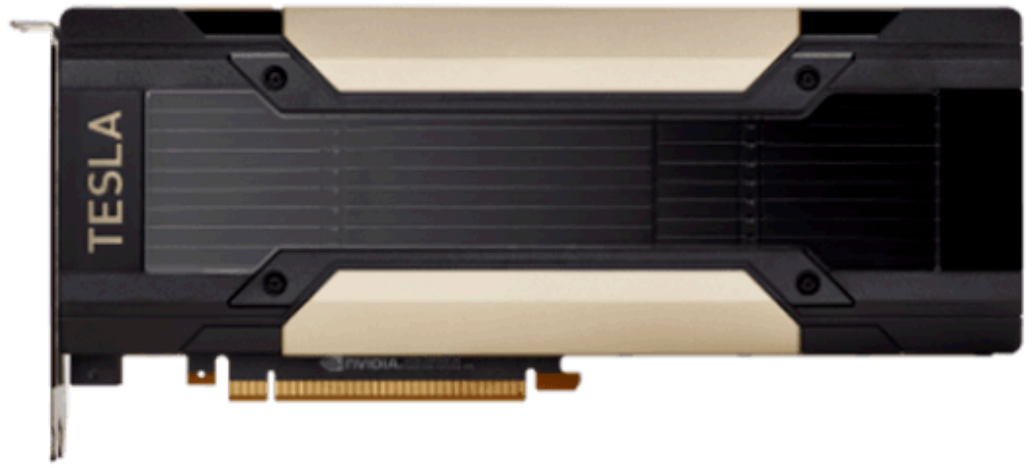


But Tiny AI is Difficult

- Tiny model design is fundamentally different from mobile AI, due to limited memory.

But Tiny AI is Difficult

- Tiny model design is <u>fundamentally different</u> from mobile AI, due to <u>limited memory</u>.



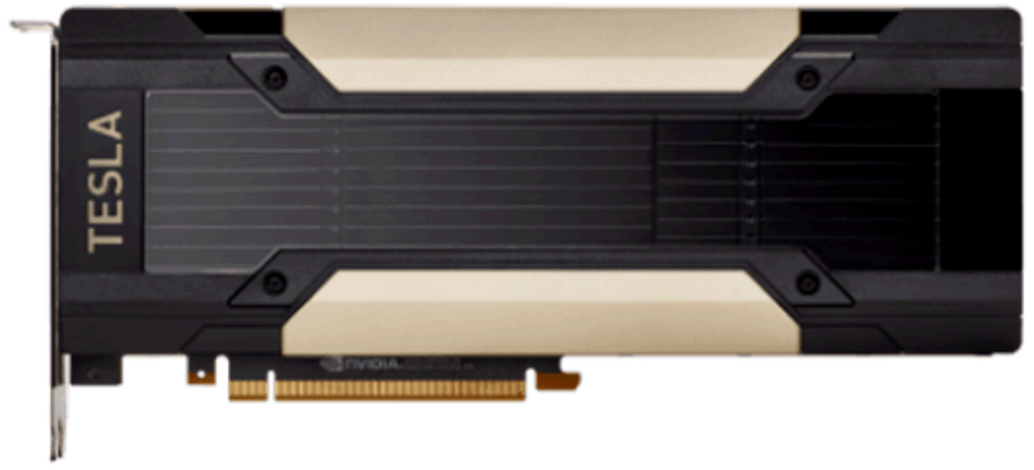
	Cloud AI	Mobile AI	Tiny AI
Memory	32GB	4GB	256kB

16,000x smaller

100,000x smaller

But Tiny AI is Difficult

- Tiny model design is <u>fundamentally different</u> from mobile AI, due to <u>limited memory</u>.
- Existing work optimize for #parameters/#FLOPs, but <u>#activation</u> is the real bottleneck.
- CANNOT directly scale.



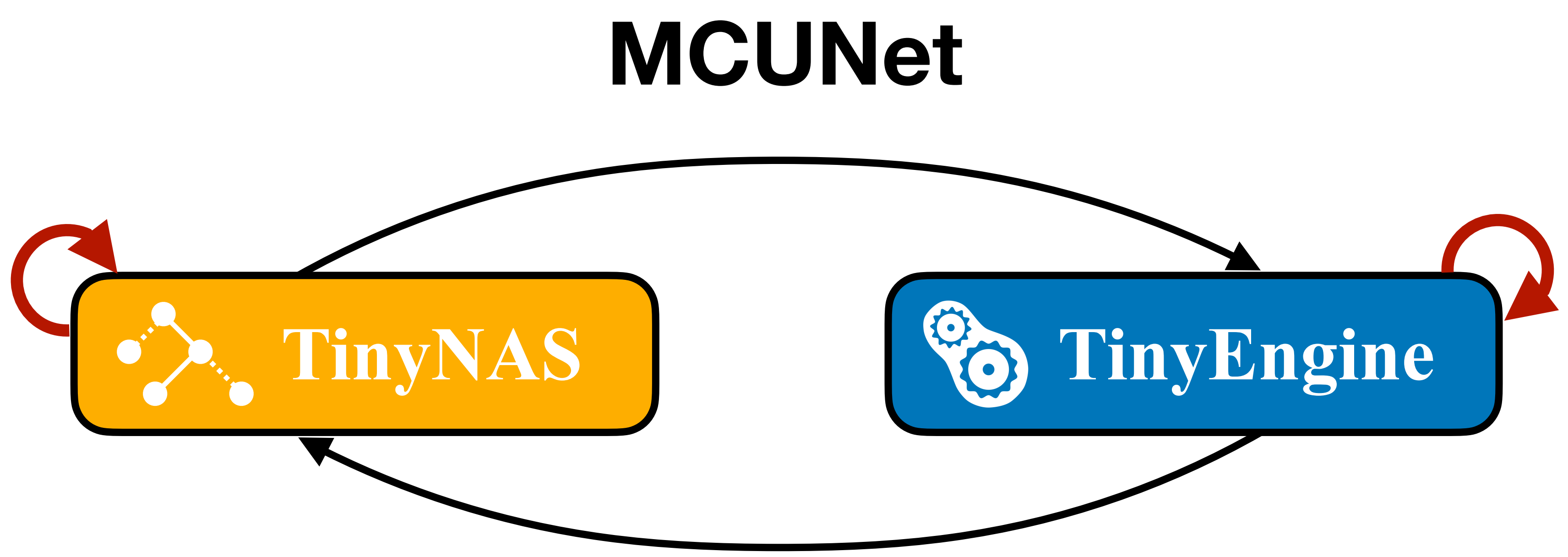
	Cloud AI	**Mobile AI**	**Tiny AI**
Memory	32GB	4GB	256kB

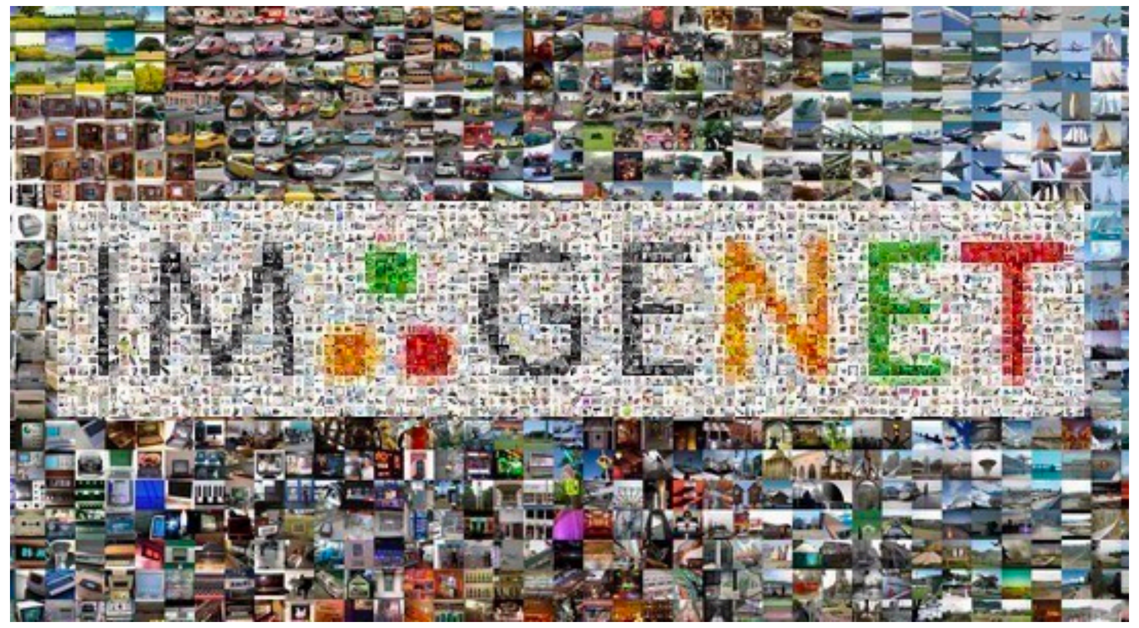
16,000x smaller

100,000x smaller

Breaking the Memory Bottleneck of TinyML

Toy applications



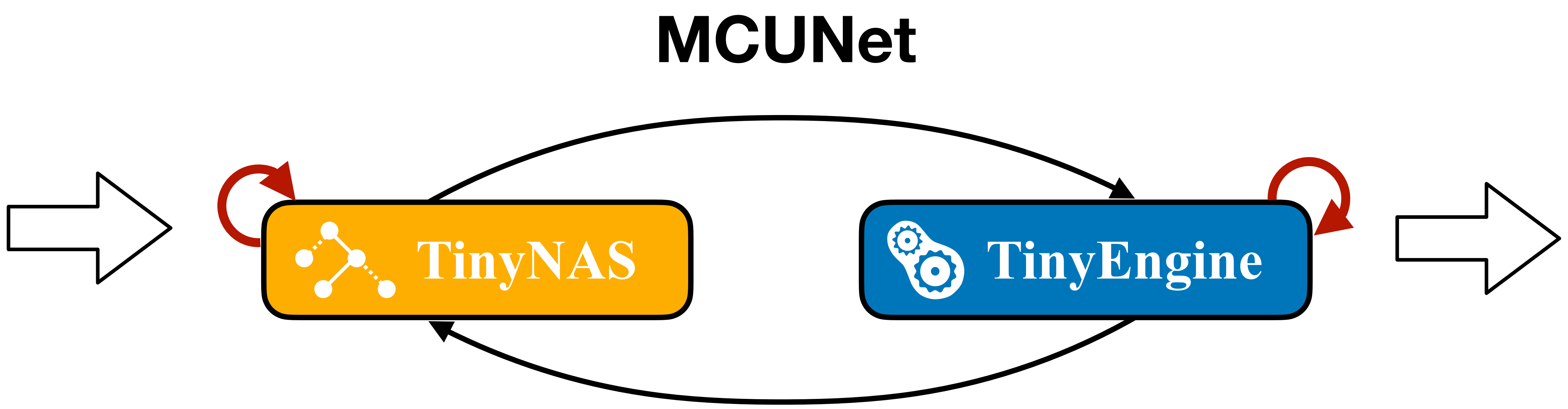


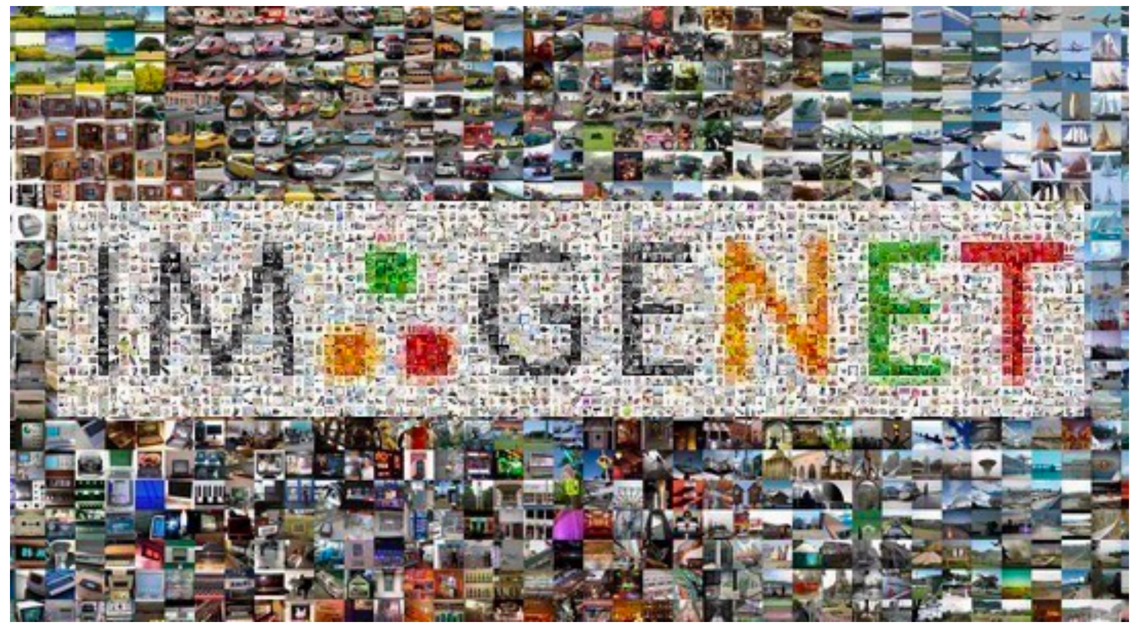
Real-life applications

* Lin *et al.*, MCUNet: Tiny Deep Learning on IoT Devices

Breaking the Memory Bottleneck of TinyML

Toy applications





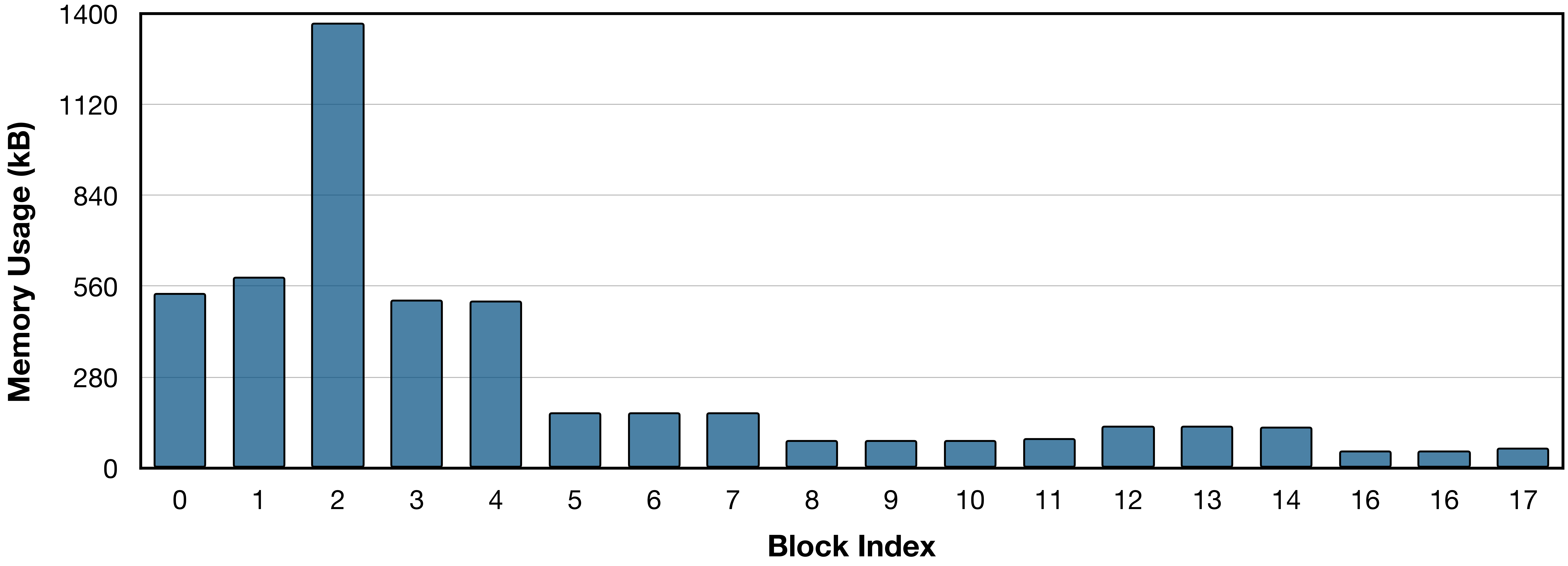
Real-life applications

- Problems:
  - Insufficient/imbalanced memory utilization across blocks
  - Poor performance on applications beyond classification (e.g., detection)

* Lin *et al.*, MCUNet: Tiny Deep Learning on IoT Devices

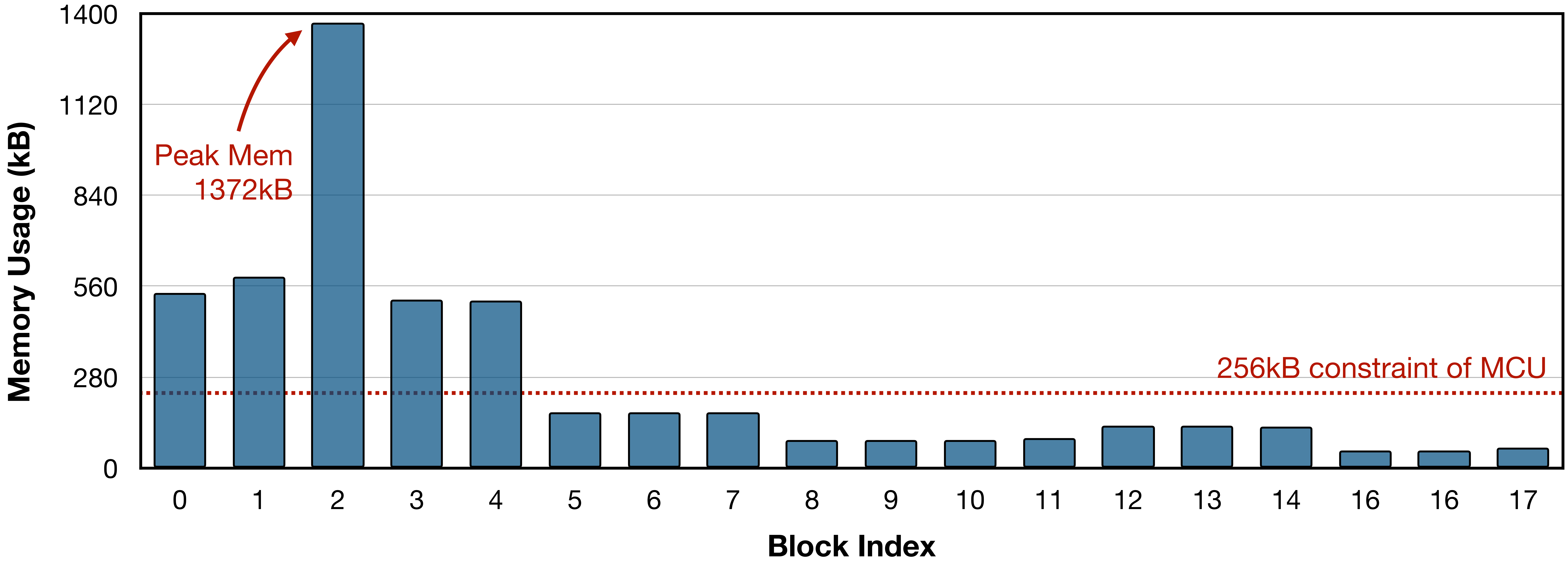
Imbalanced Memory Distribution of CNNs

- Per-block memory usage of MobileNetV2



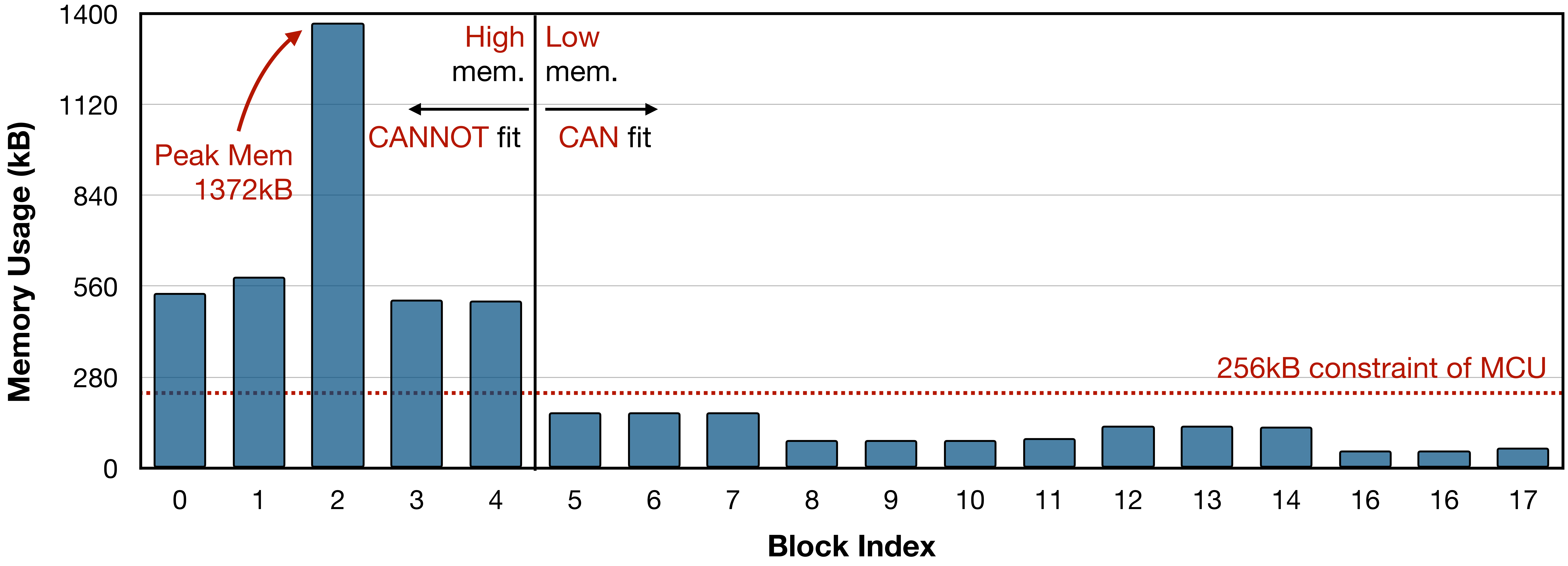
Imbalanced Memory Distribution of CNNs

- Per-block memory usage of MobileNetV2



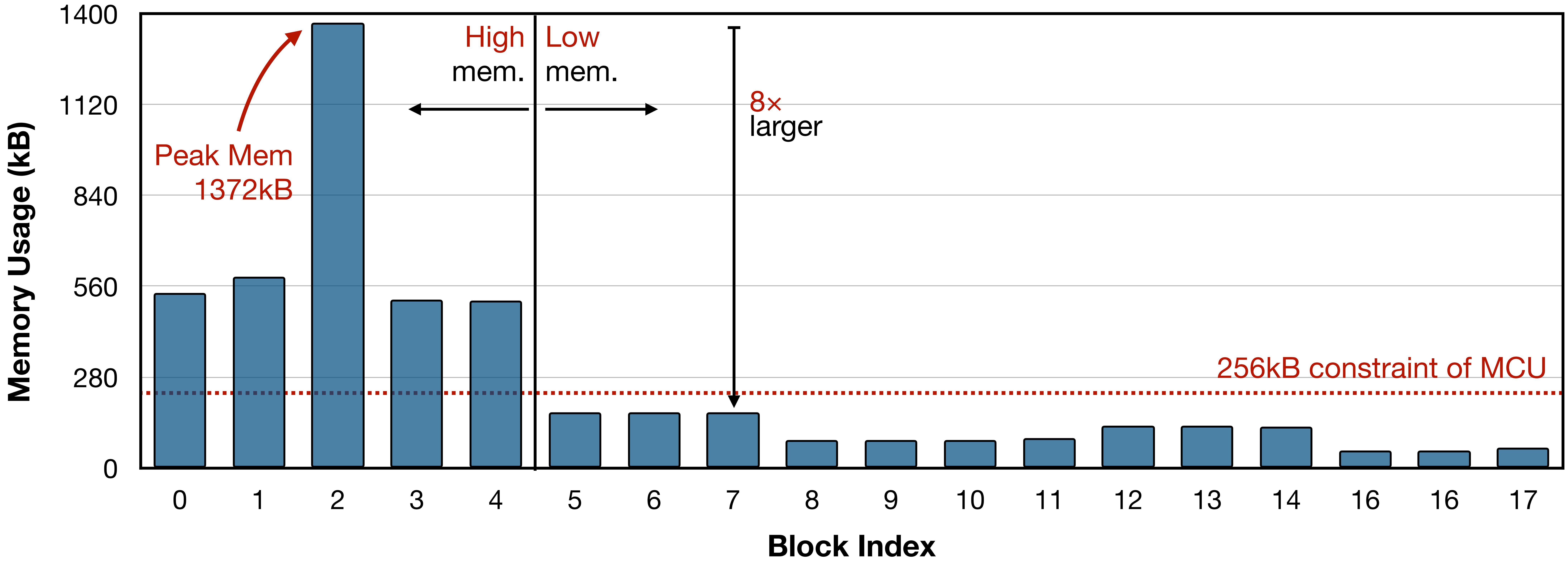
Imbalanced Memory Distribution of CNNs

- Per-block memory usage of MobileNetV2



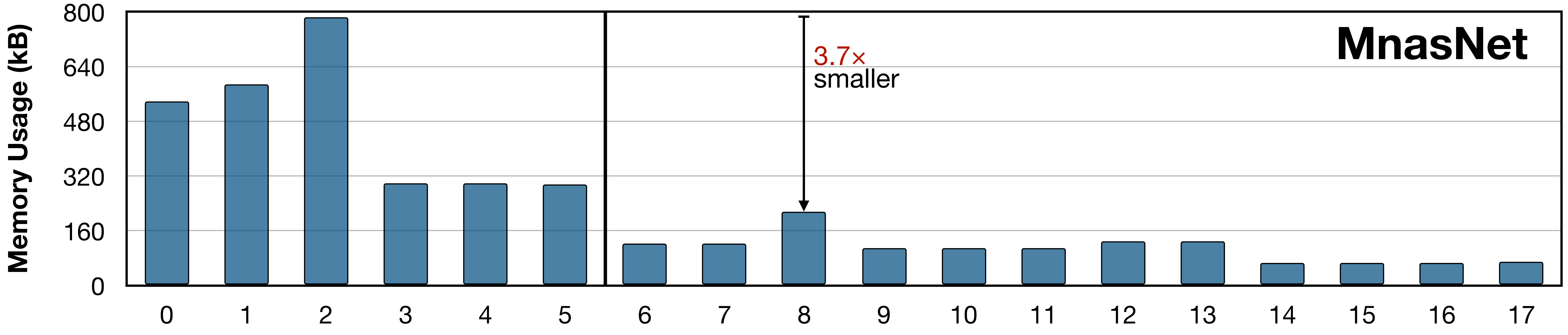
Imbalanced Memory Distribution of CNNs

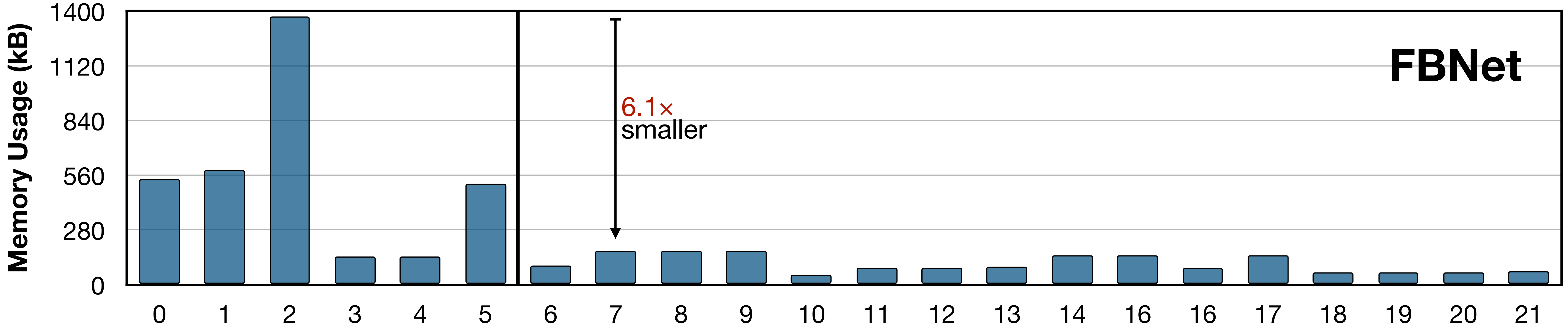
- Per-block memory usage of MobileNetV2



Imbalanced Memory Distribution of CNNs

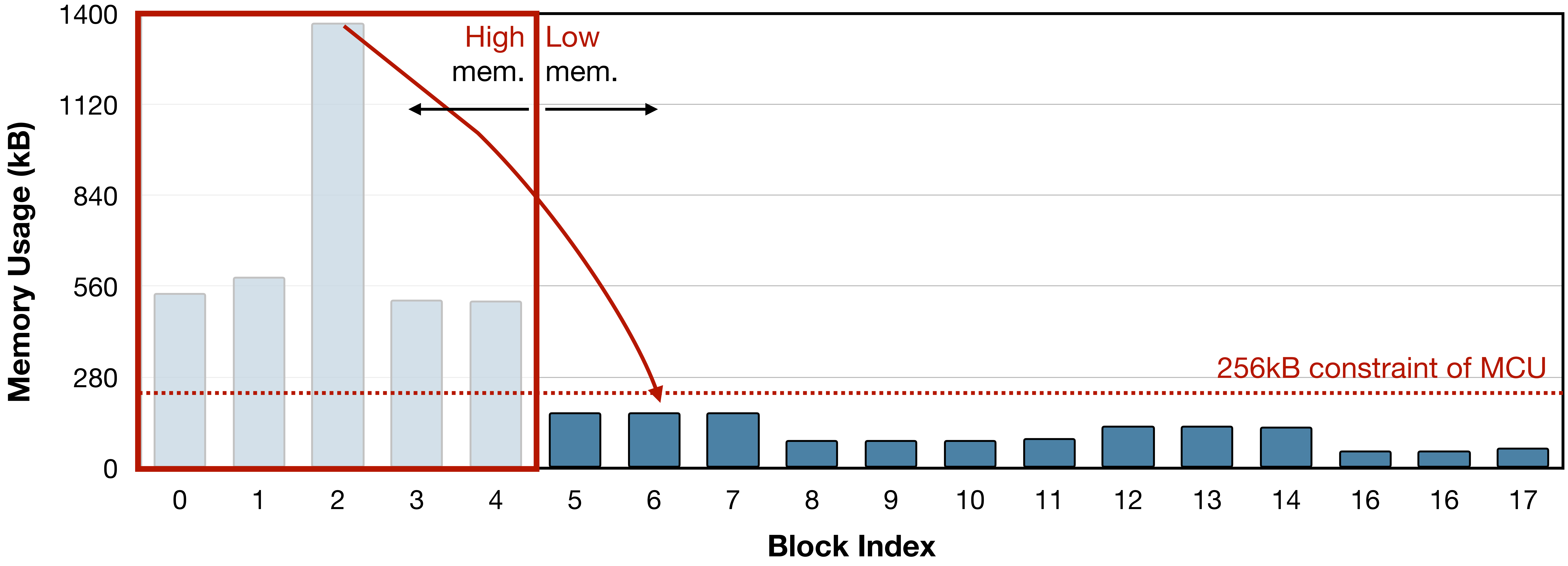
- Common case in efficient CNN design



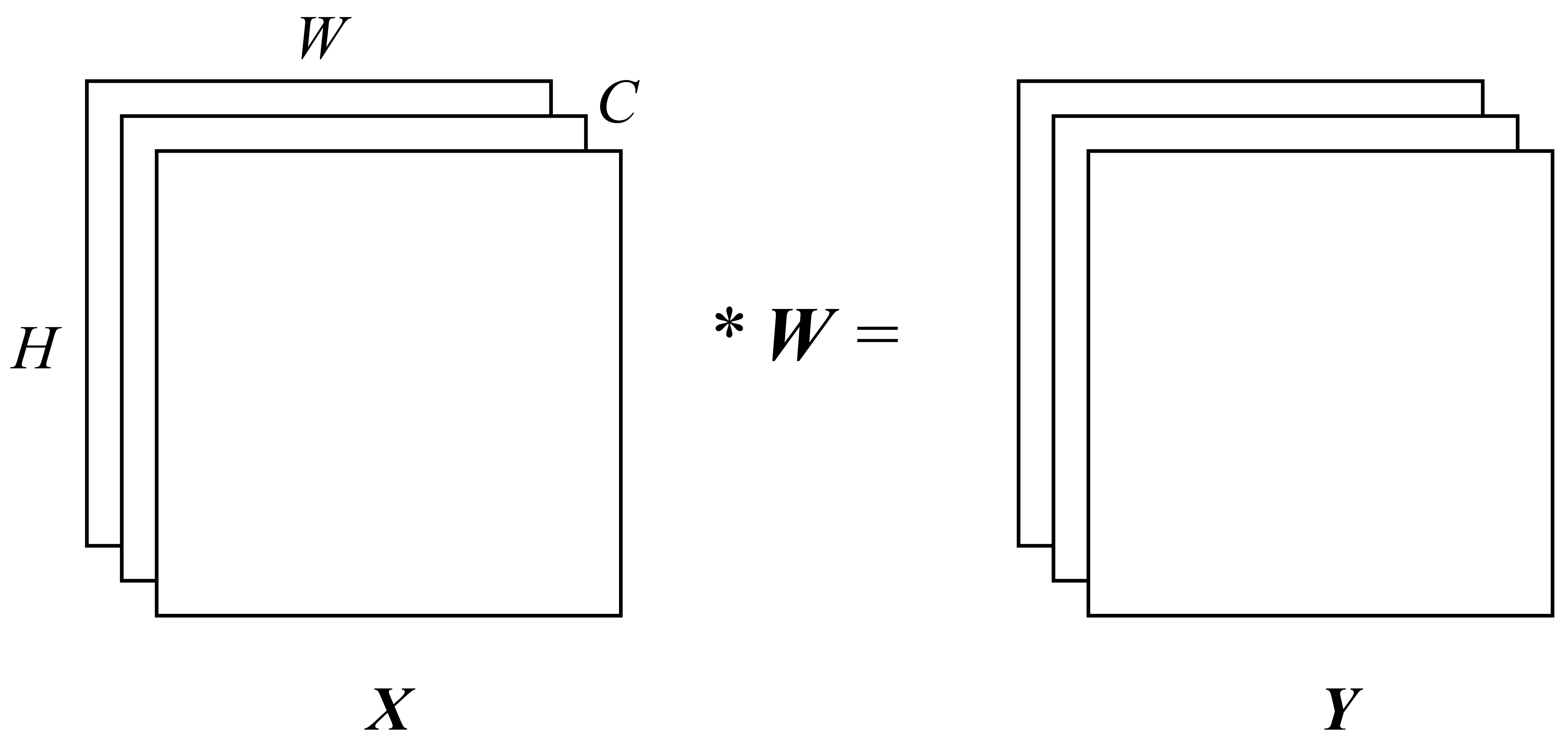


Imbalanced Memory Distribution of CNNs

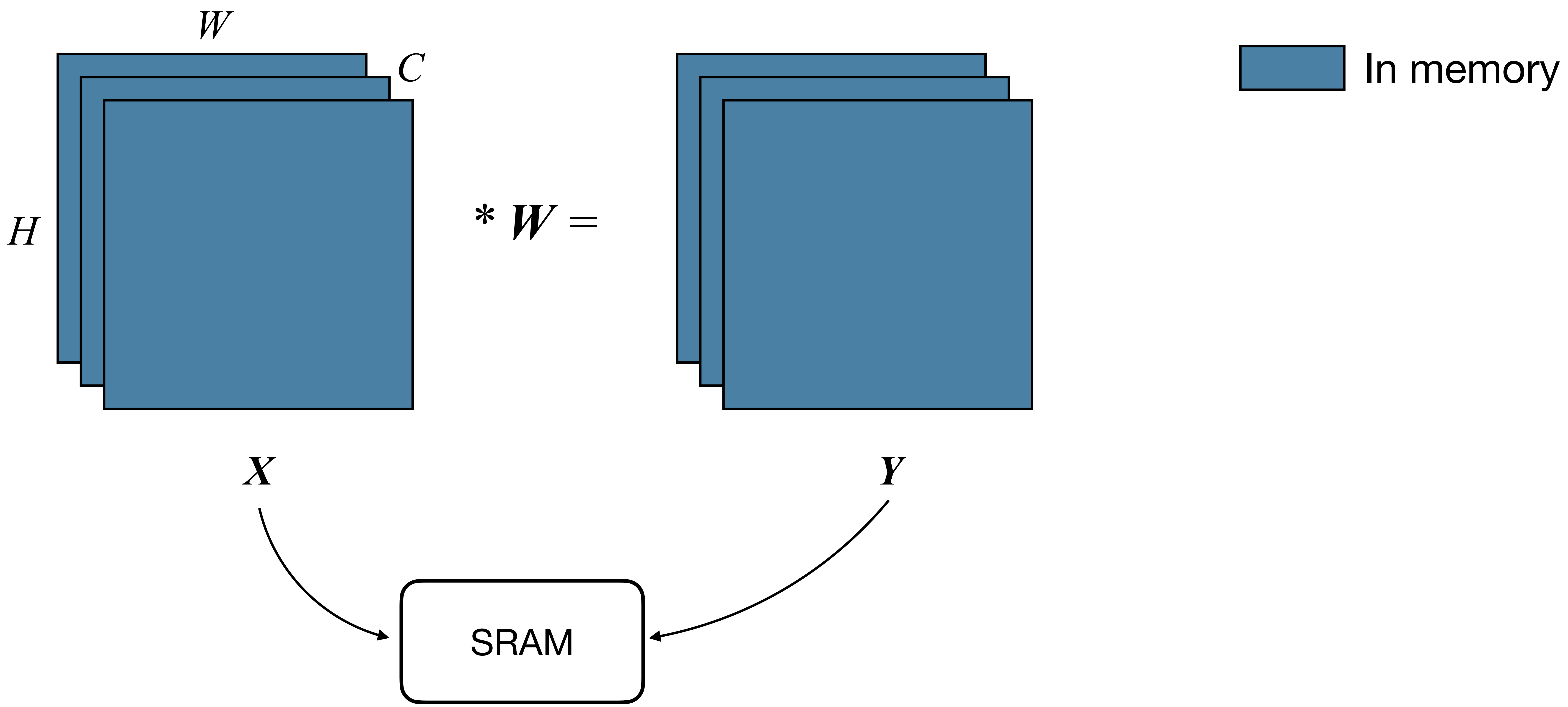
Reduce memory usage of the initial stage
-> Reduce the overall memory usage



1. Saving Memory with Patch-based Inference



1. Saving Memory with Patch-based Inference

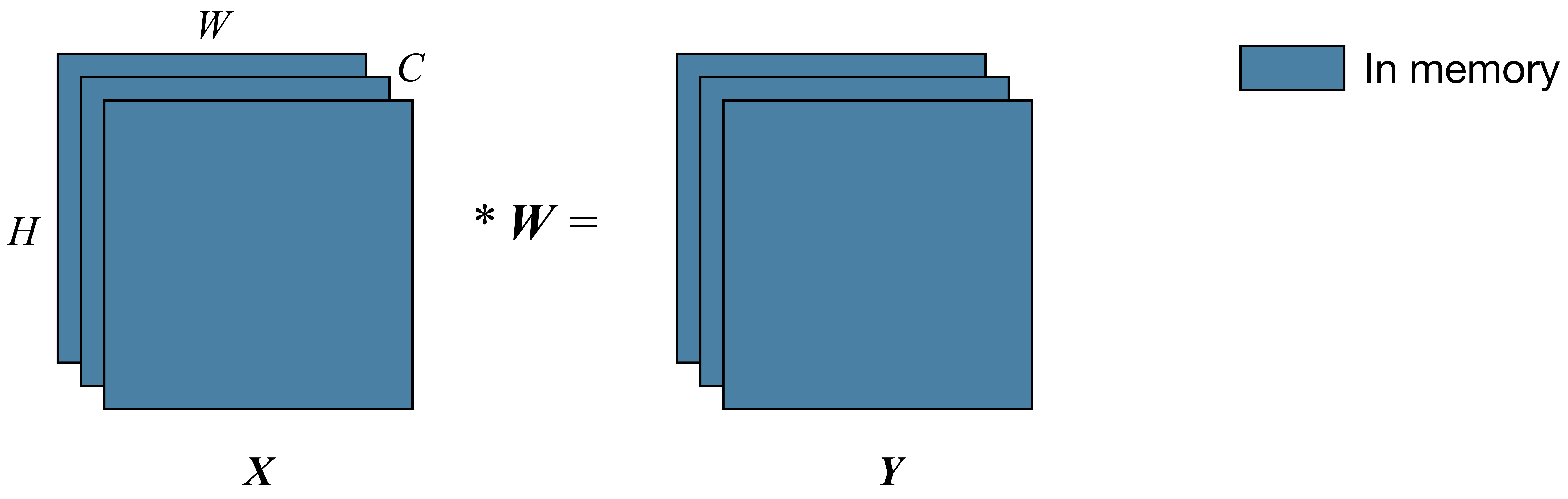


* weights are usually partially fetched from Flash

1. Saving Memory with Patch-based Inference

1. Per-layer inference

Peak Mem $= 2\ WHC$



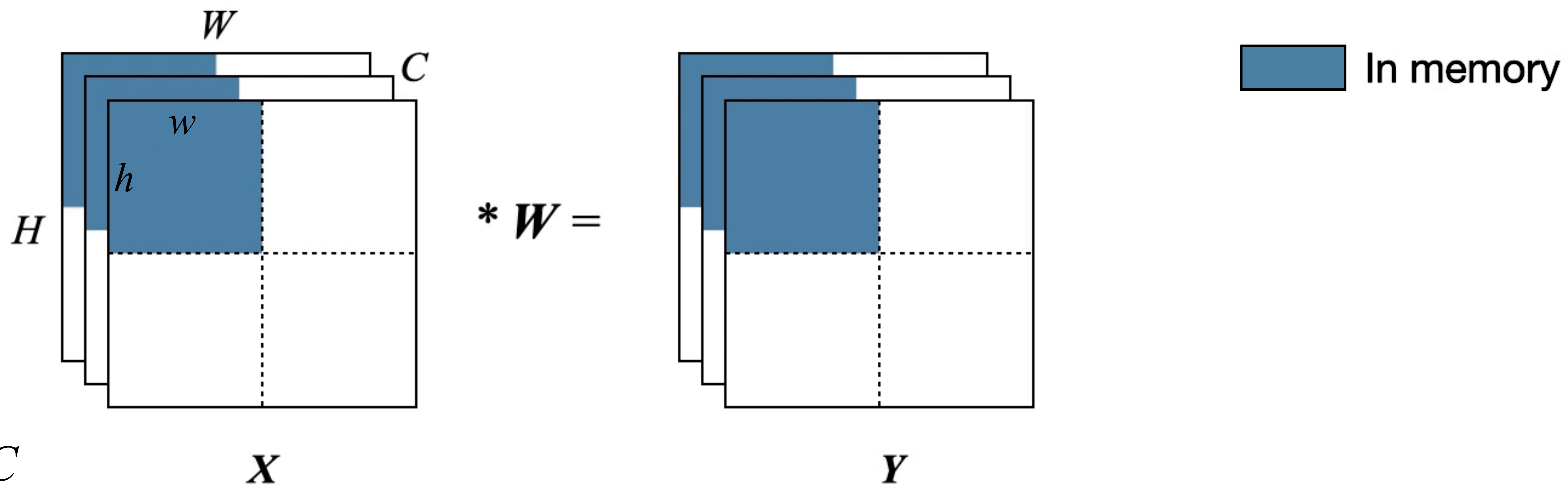
1. Saving Memory with Patch-based Inference

1. Per-layer inference

Peak Mem = $2\ WHC$

2. Per-patch inference

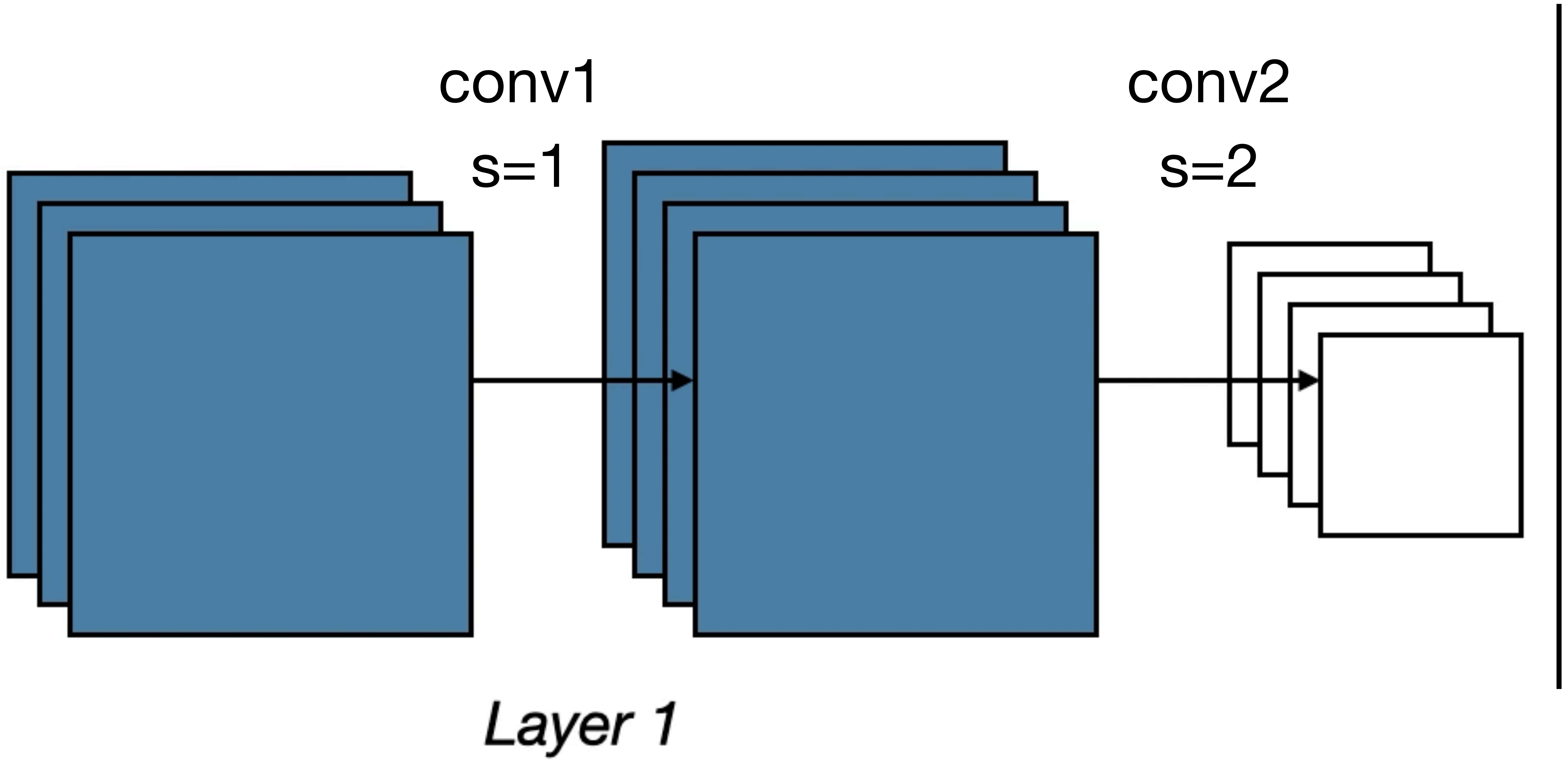
Peak Mem = $2\ whC << 2WHC$



* *can use more than 2x2 patches*

1. Saving Memory with Patch-based Inference

- a practical 2-layer example



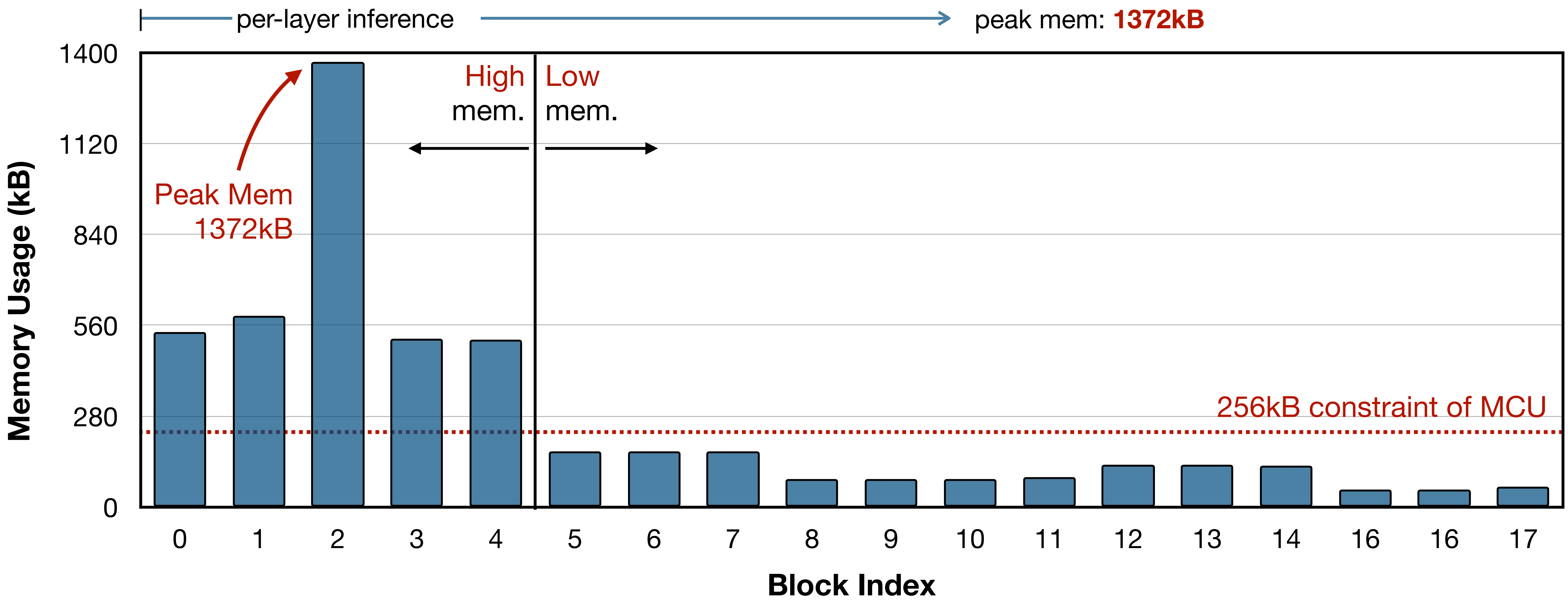
per-layer inference

1. Saving Memory with Patch-based Inference

- a practical 2-layer example

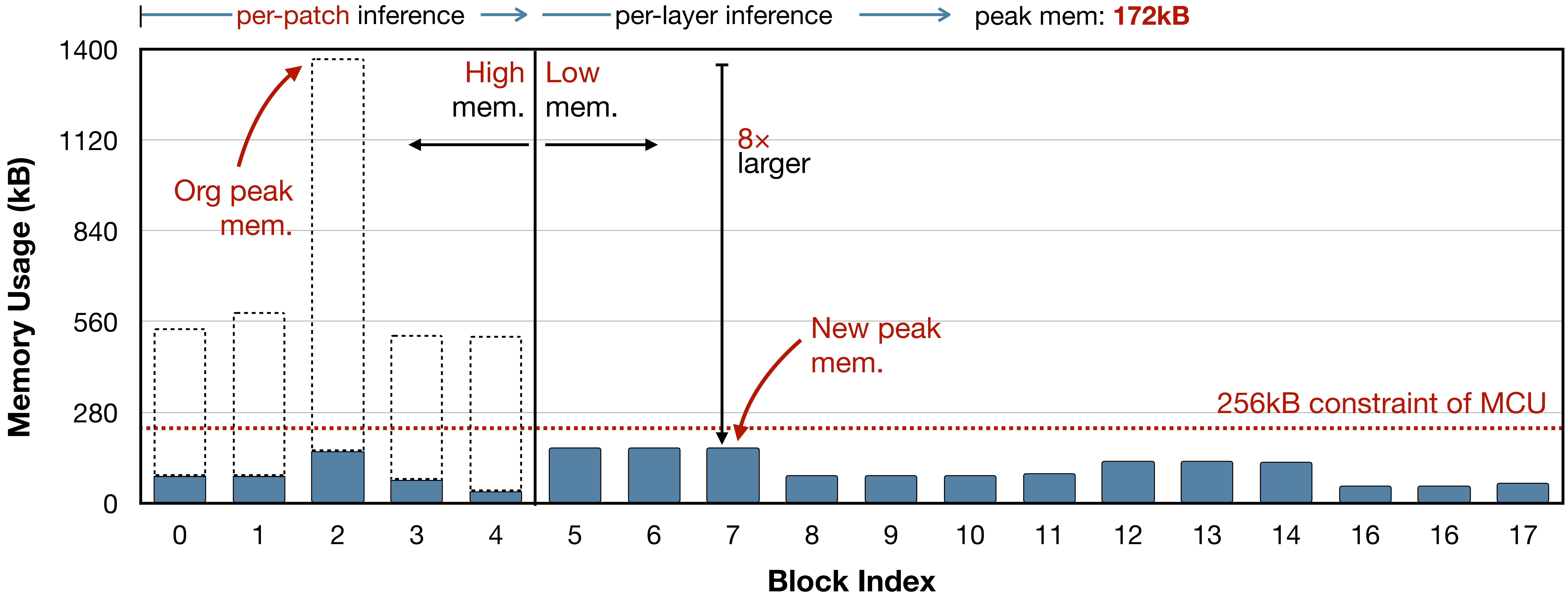
1. Saving Memory with Patch-based Inference

- Applying to MobileNetV2



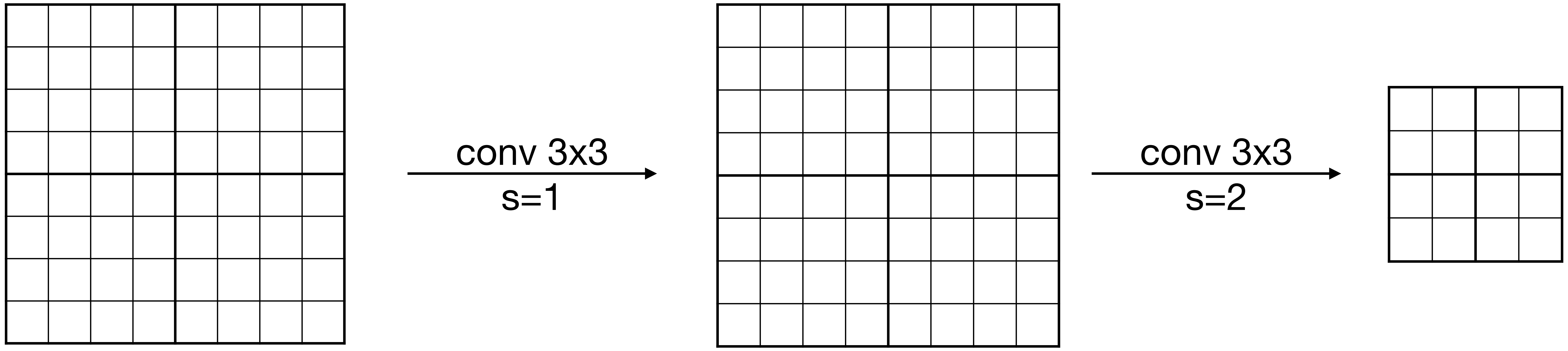
1. Saving Memory with Patch-based Inference

- Applying to MobileNetV2



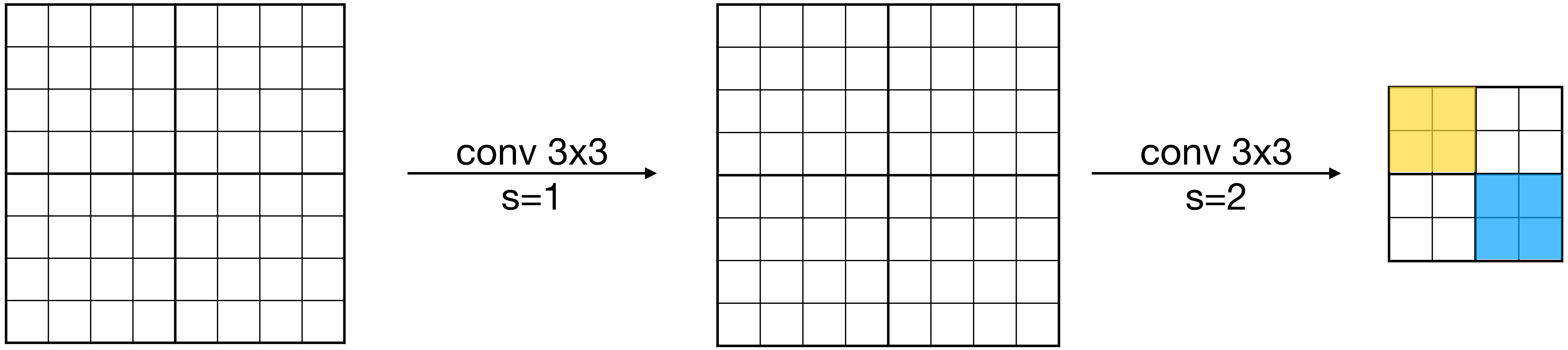
Problem: Computation Overhead from Overlapping

- Using 2x2 patches



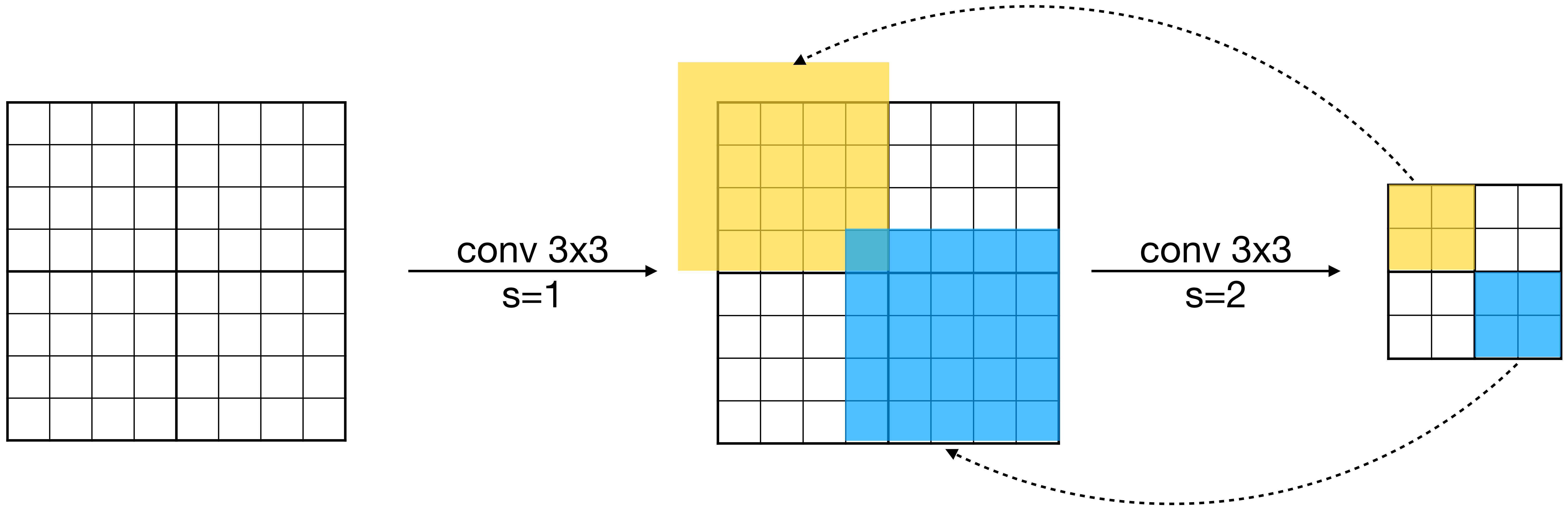
Problem: Computation Overhead from Overlapping

- Using 2x2 patches



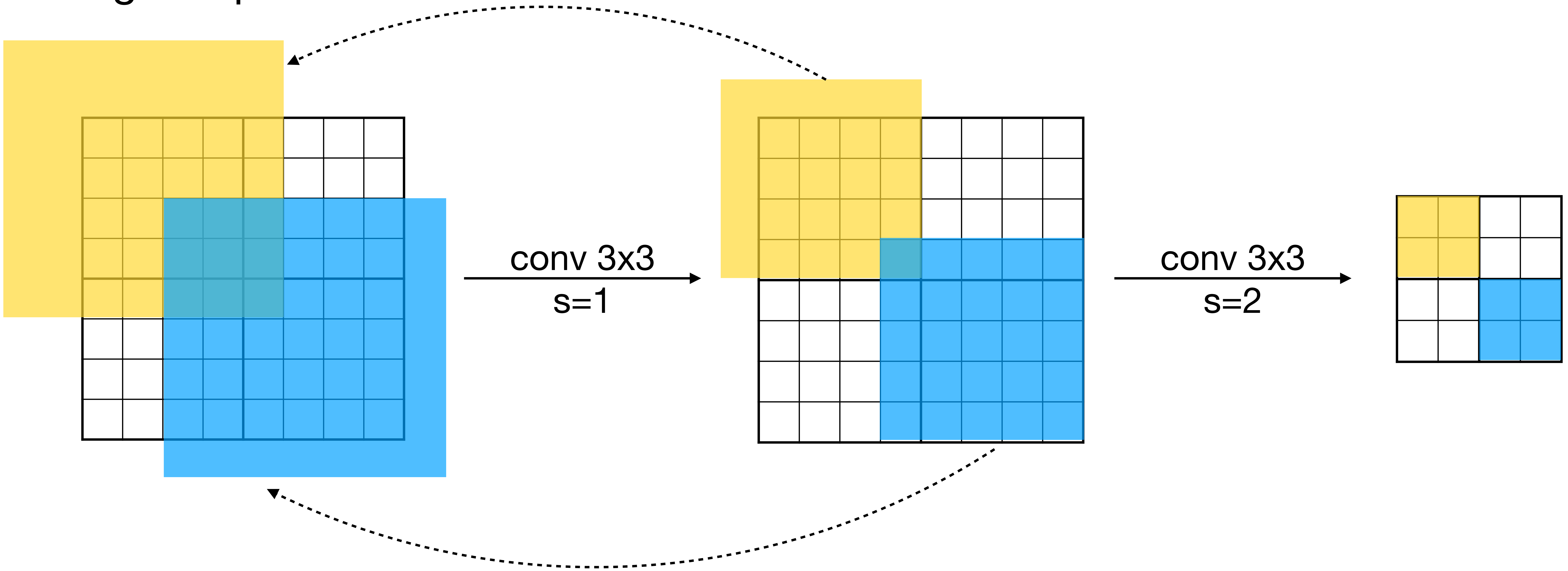
Problem: Computation Overhead from Overlapping

- Using 2x2 patches



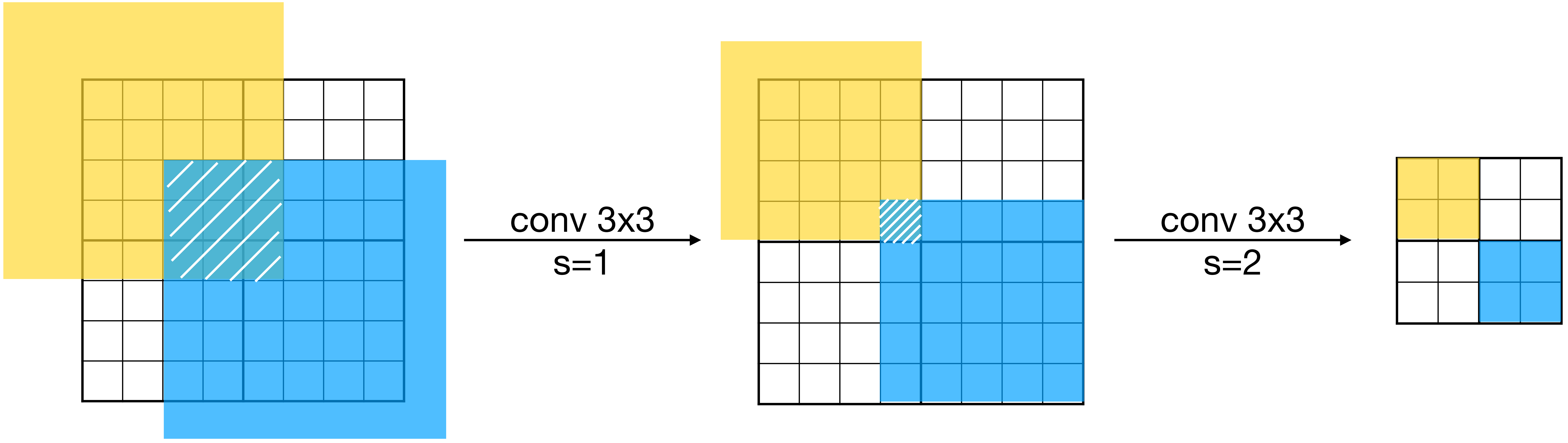
Problem: Computation Overhead from Overlapping

- Using 2x2 patches



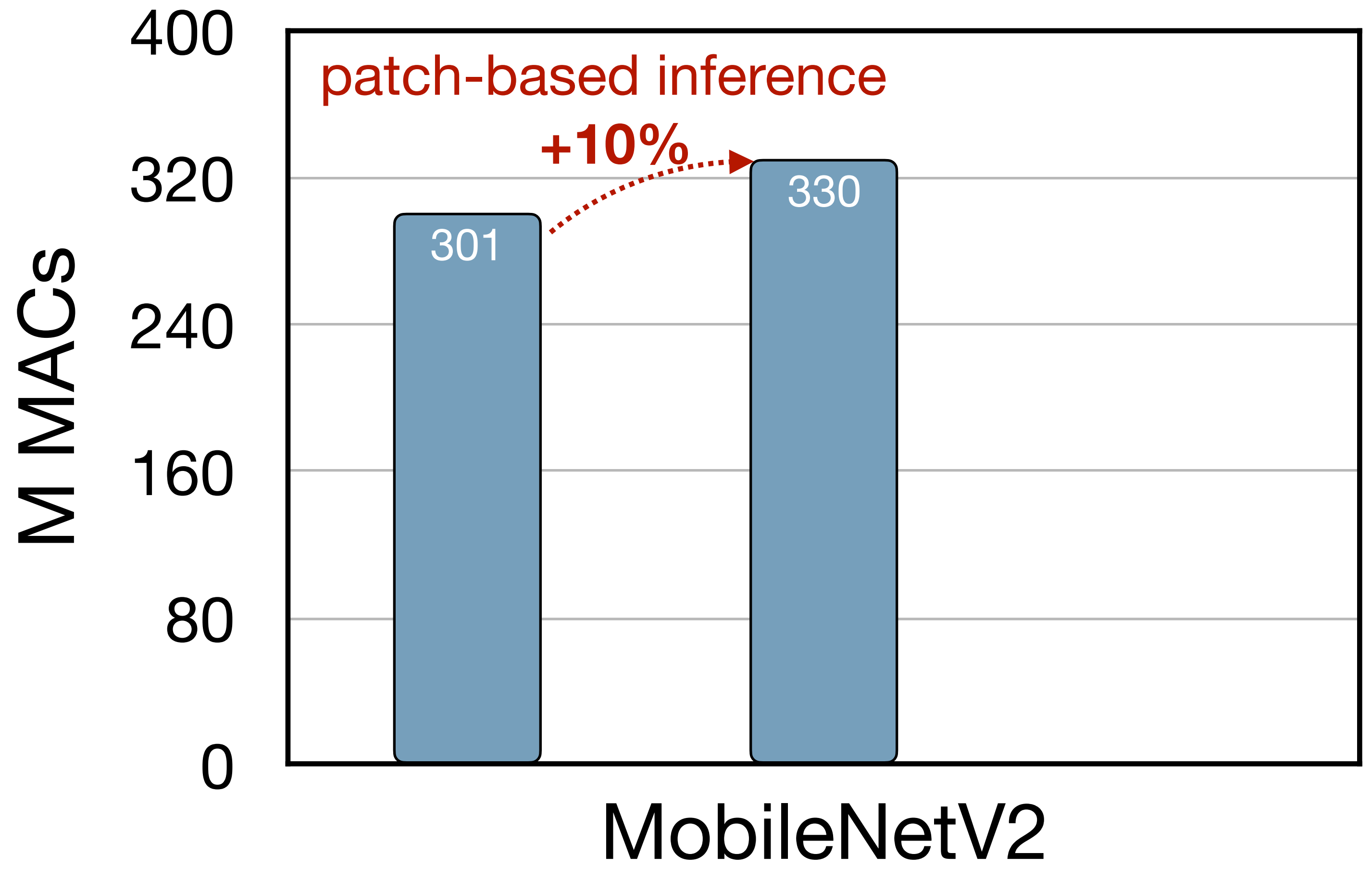
Problem: Computation Overhead from Overlapping

- Using 2x2 patches

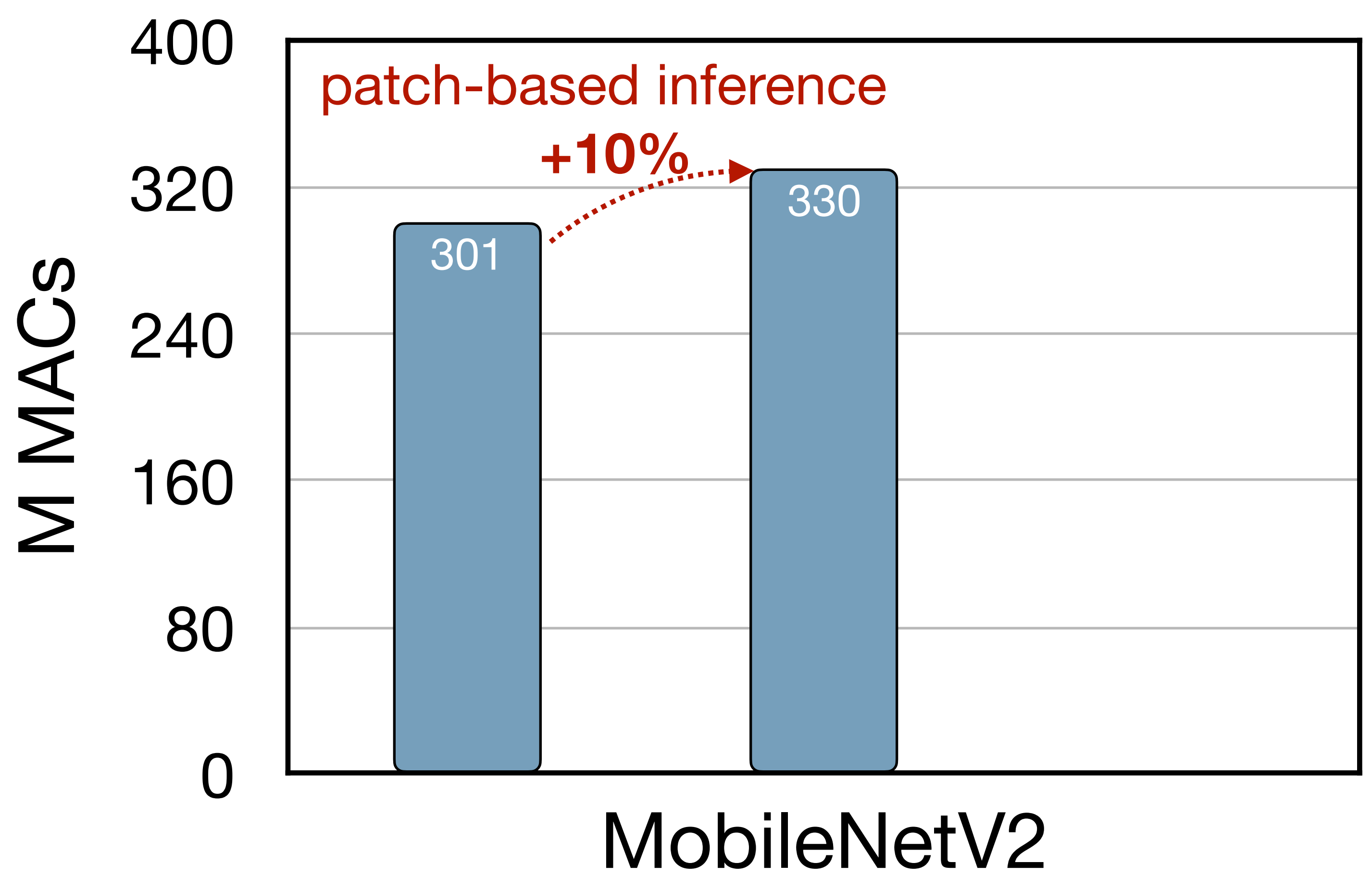


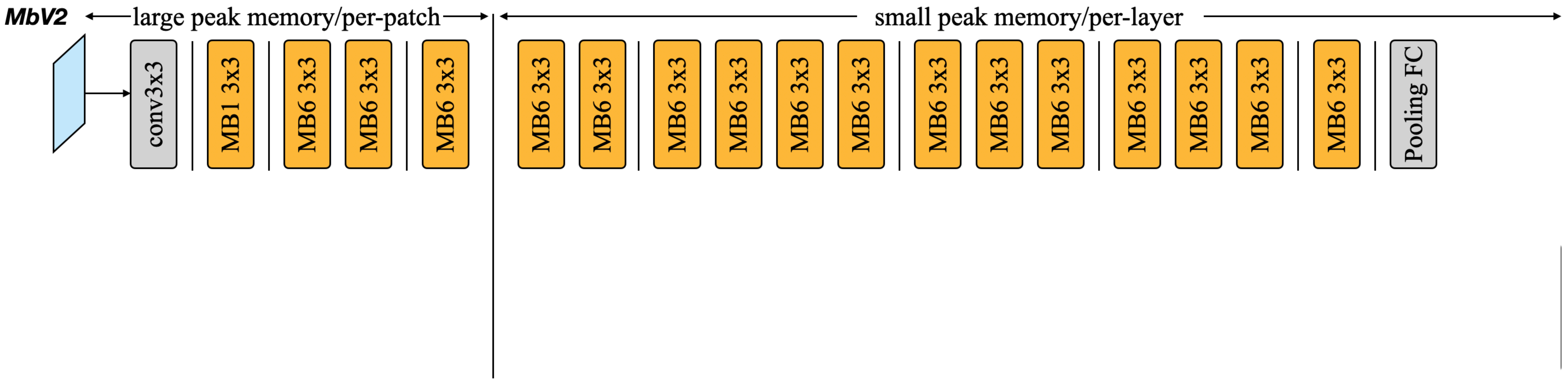
Spatial overlapping gets larger as **receptive field** grows!

Problem: Computation Overhead from Overlapping

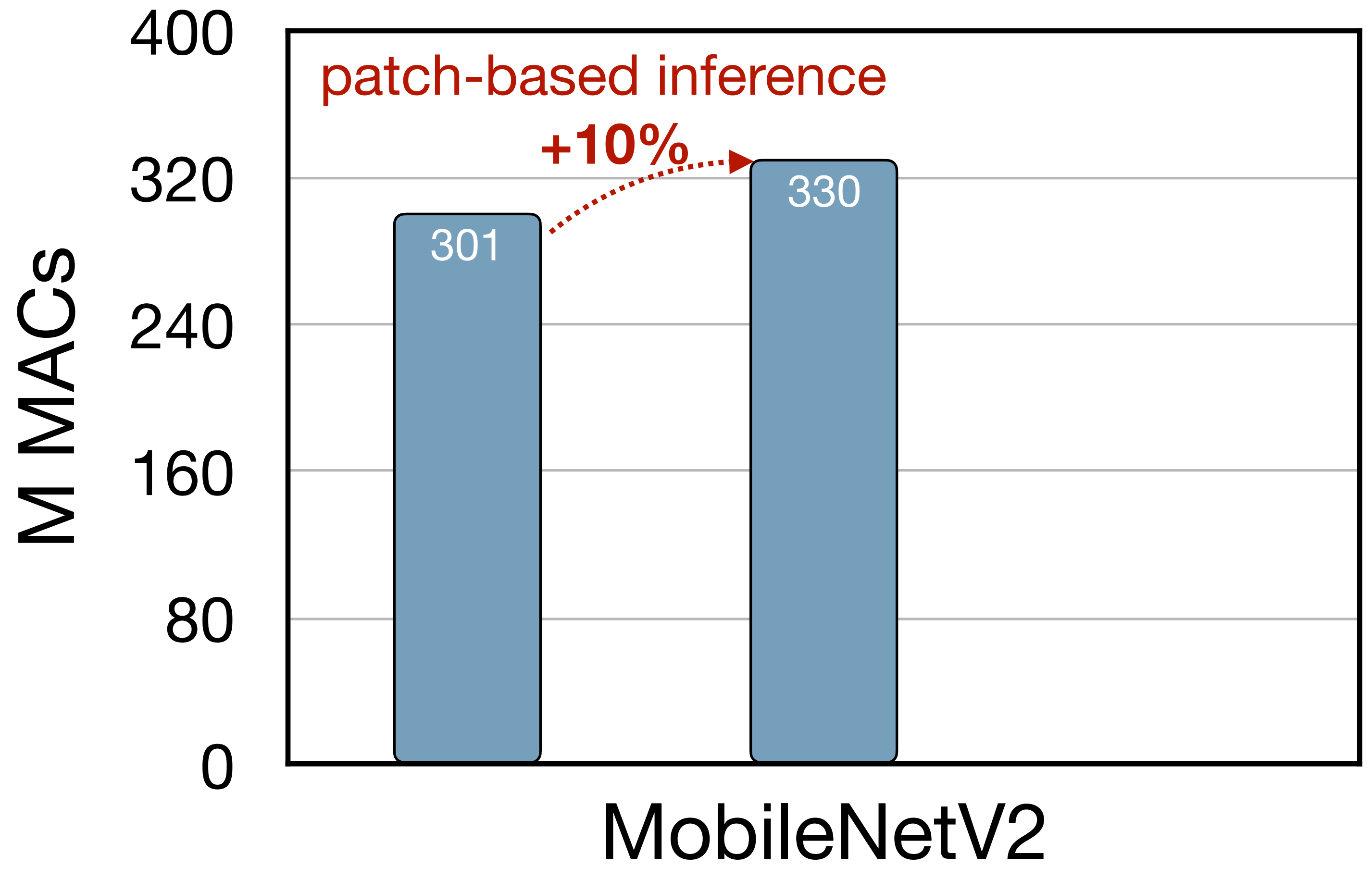


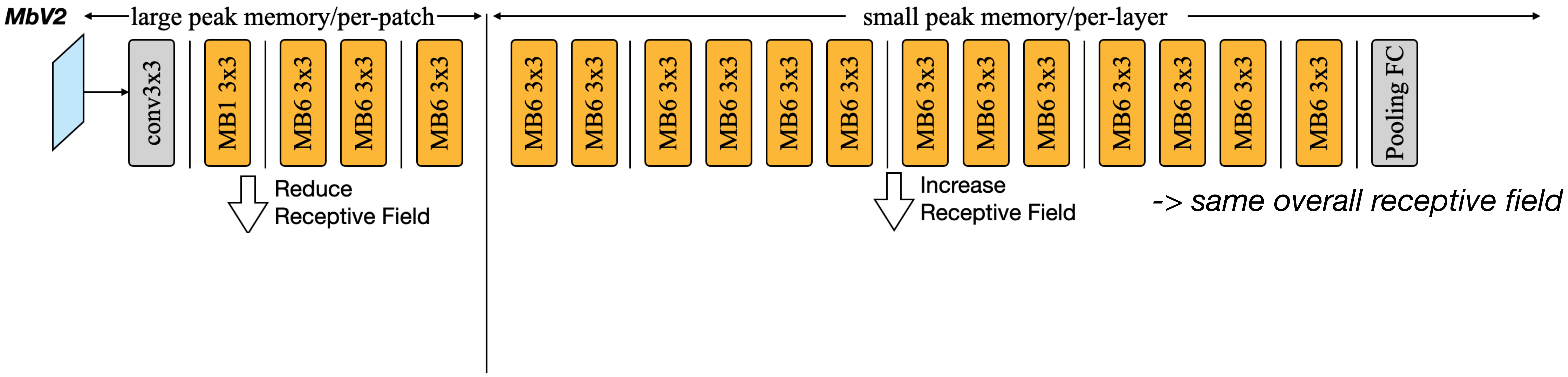
2. Network Redistribution to Reduce Overhead



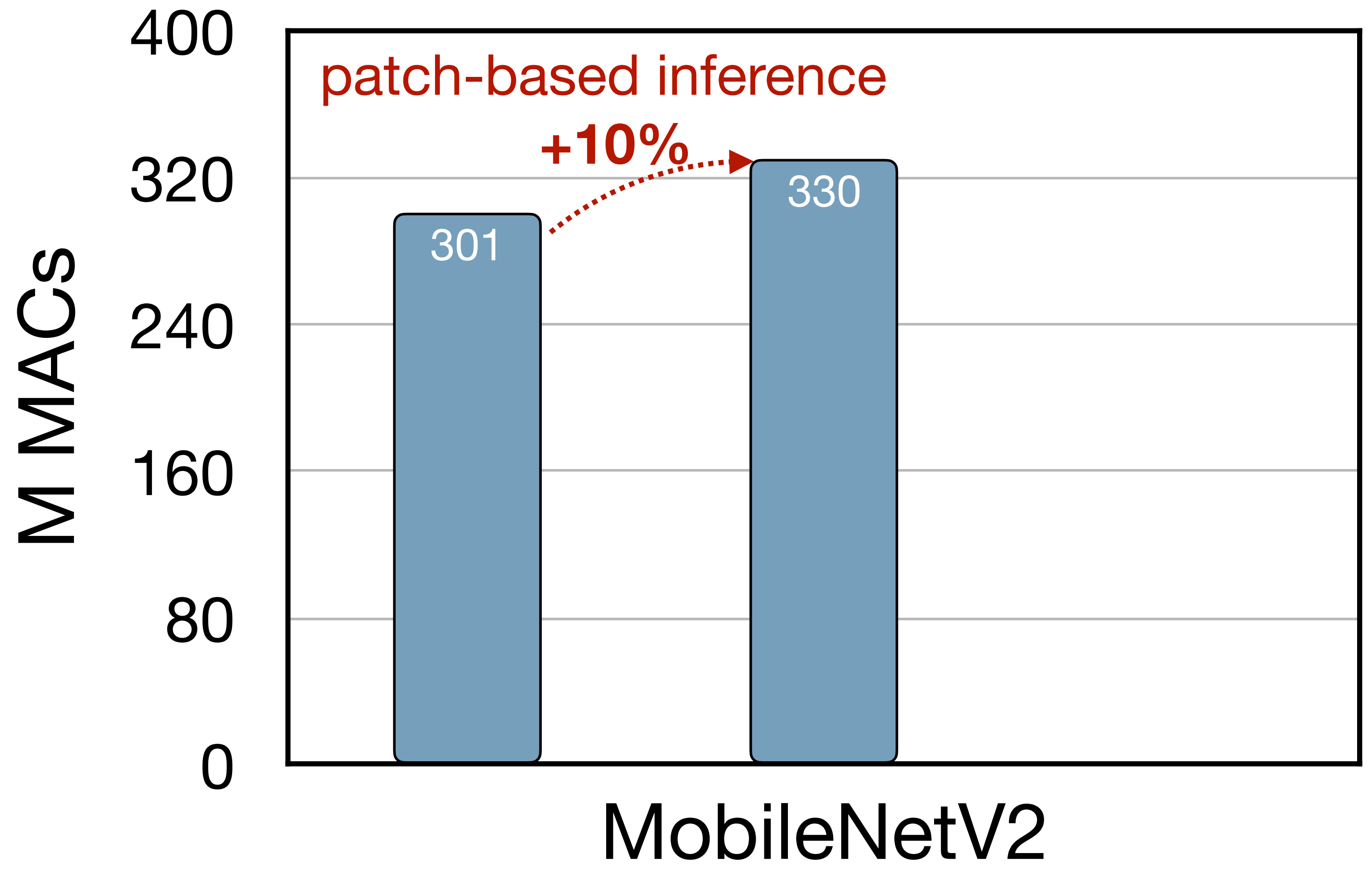


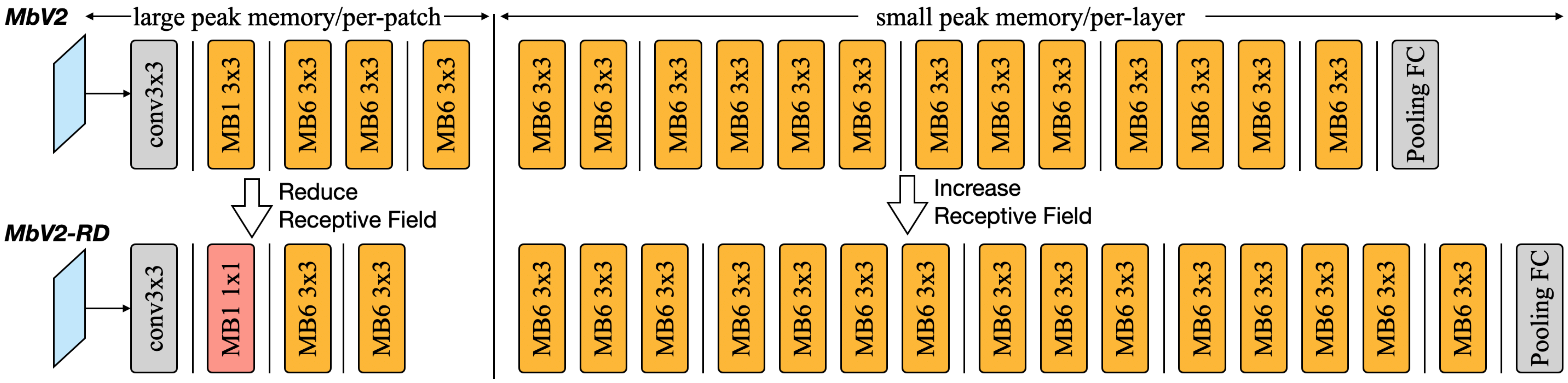
2. Network Redistribution to Reduce Overhead



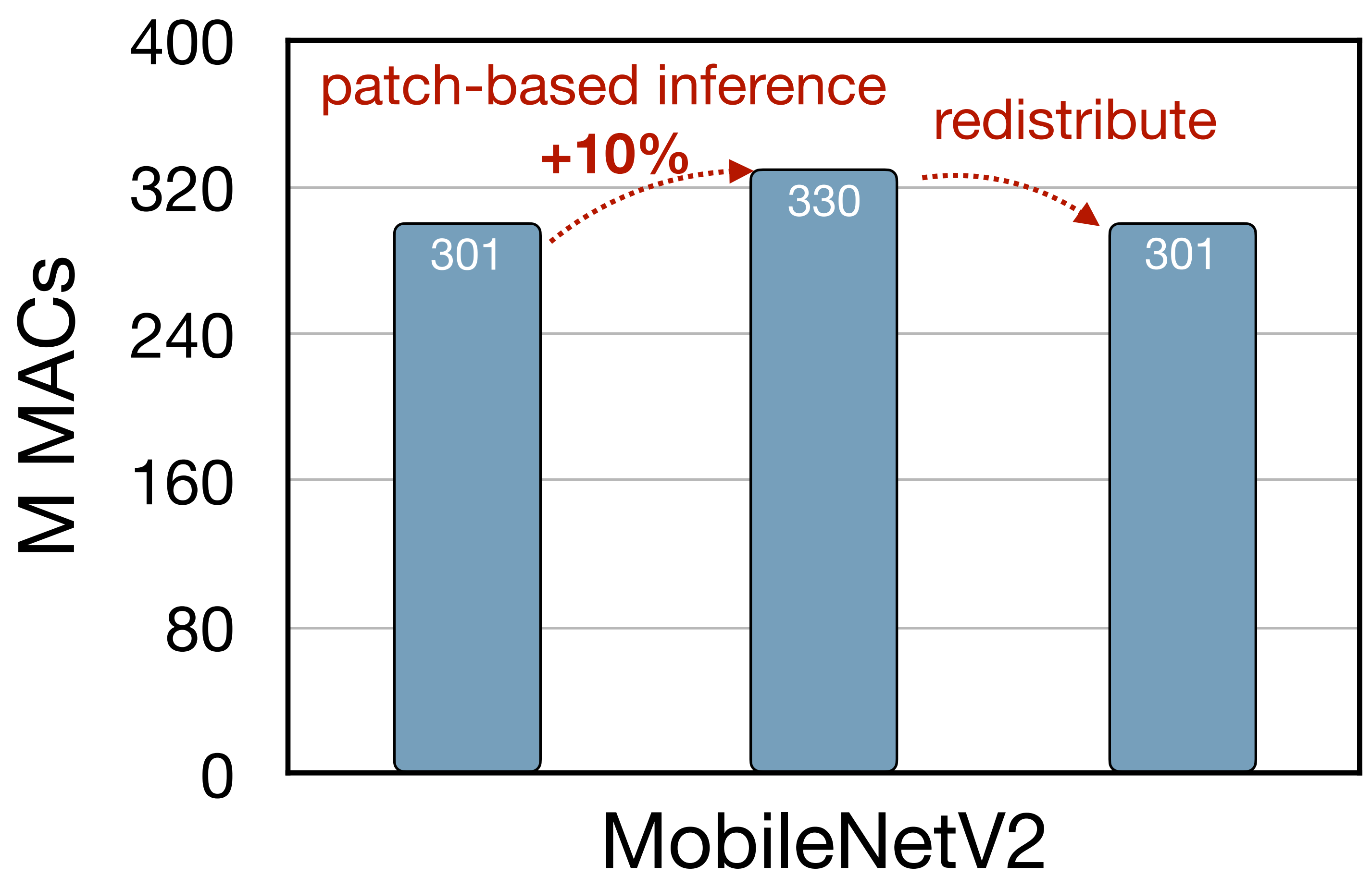


2. Network Redistribution to Reduce Overhead





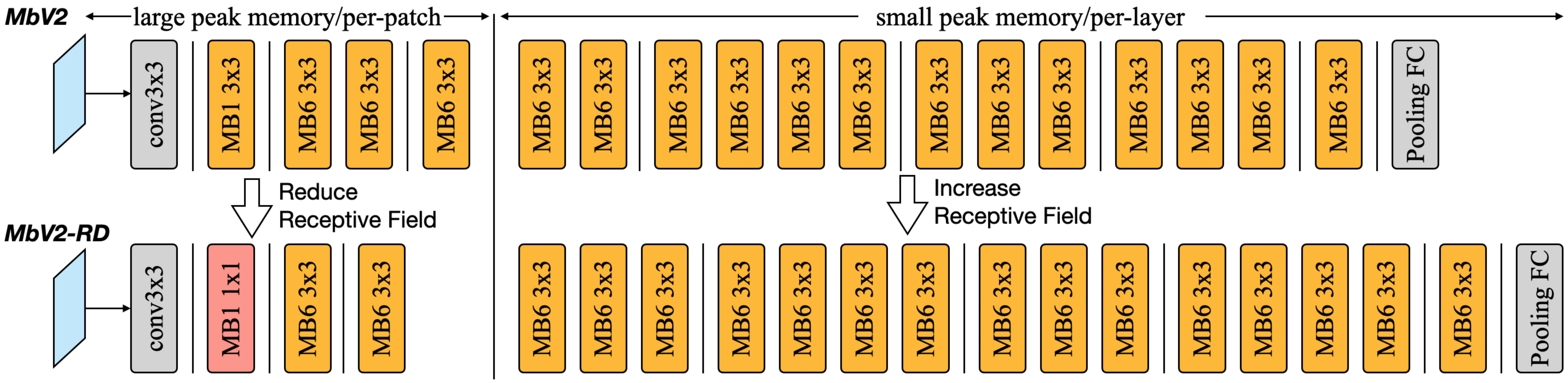
2. Network Redistribution to Reduce Overhead



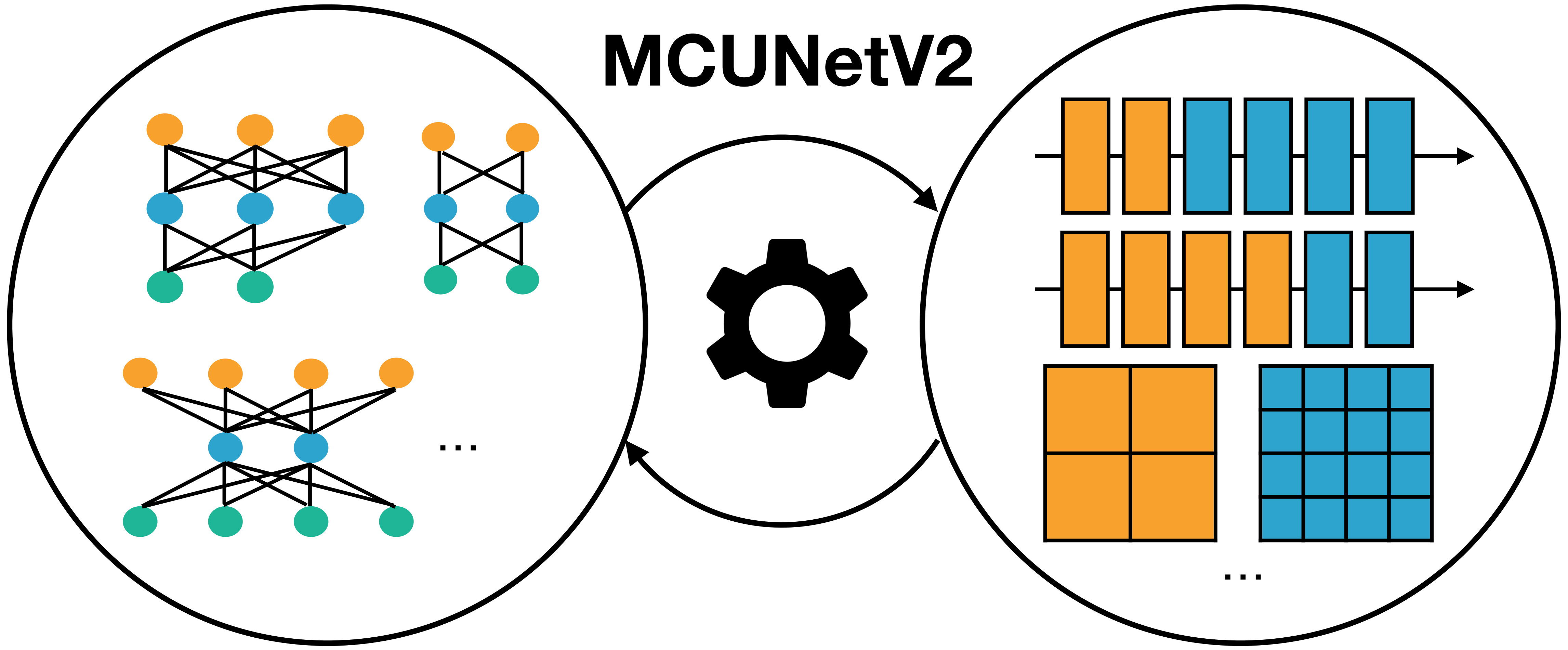
Same performance on:

- Image classification
- Object detection
- ….

Negligible overhead



3. Joint Automated Search for Optimization



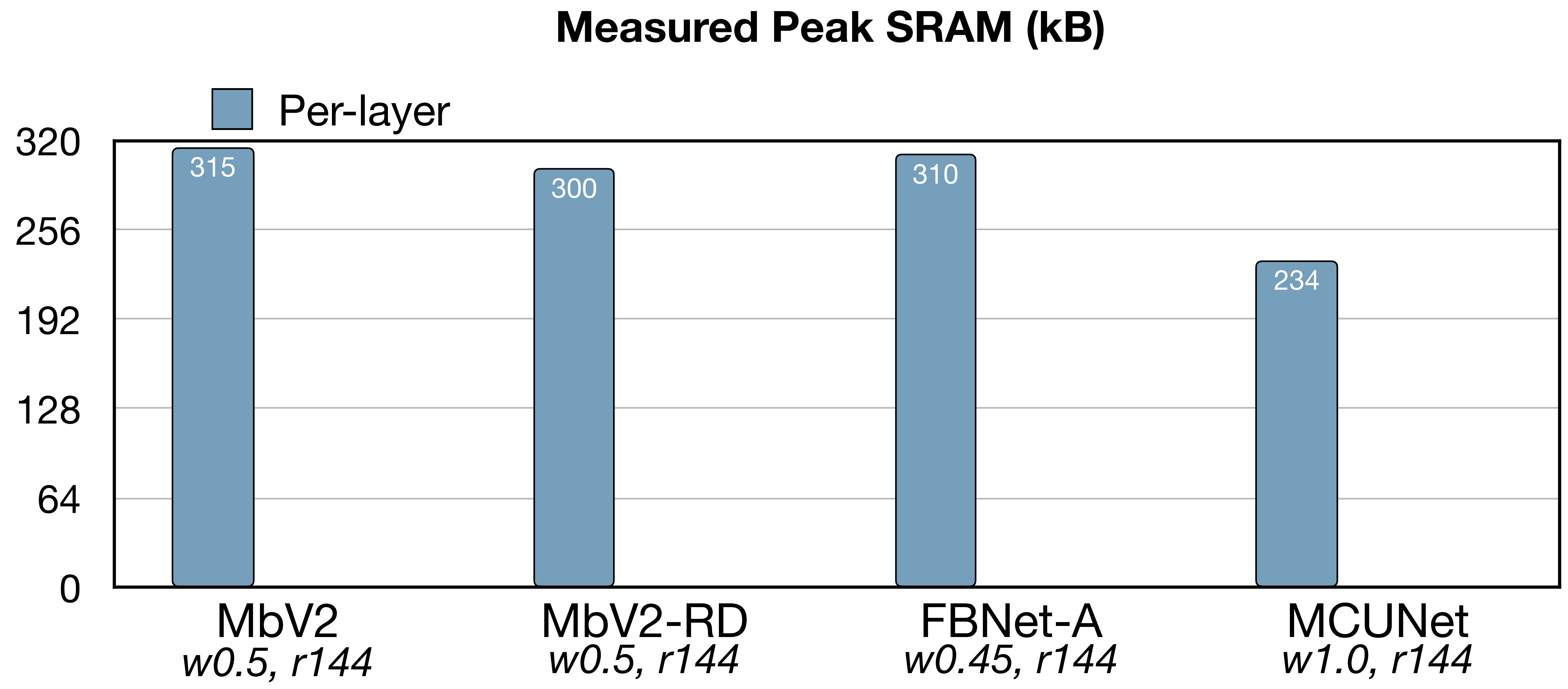
Neural architecture
#layers
#channels
kernel size
...

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

* Lin *et al.*, MCUNet: Tiny Deep Learning on IoT Devices

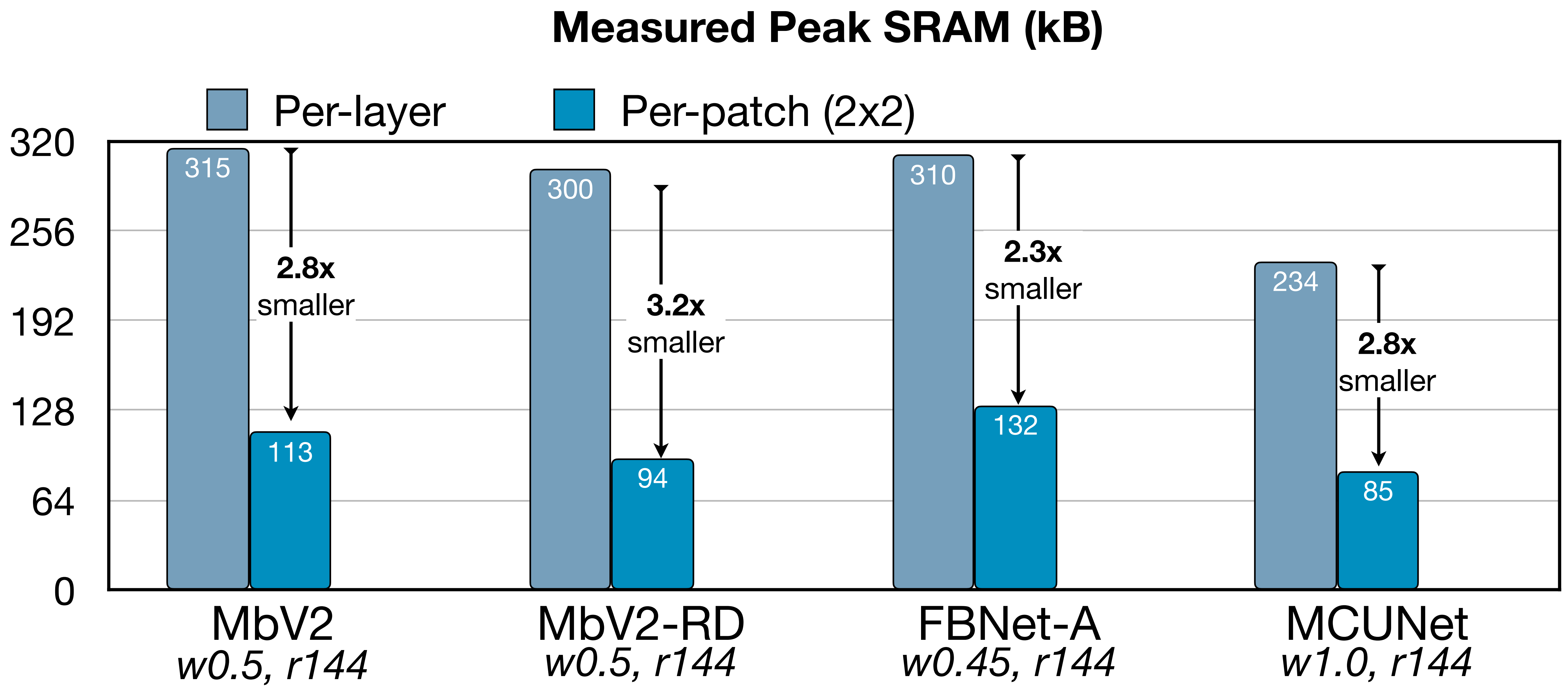
Reducing the Peak Memory of CNNs

- Baseline: TinyEngine, the SOTA system stack for tinyML
- Measured on STM32F746 MCU



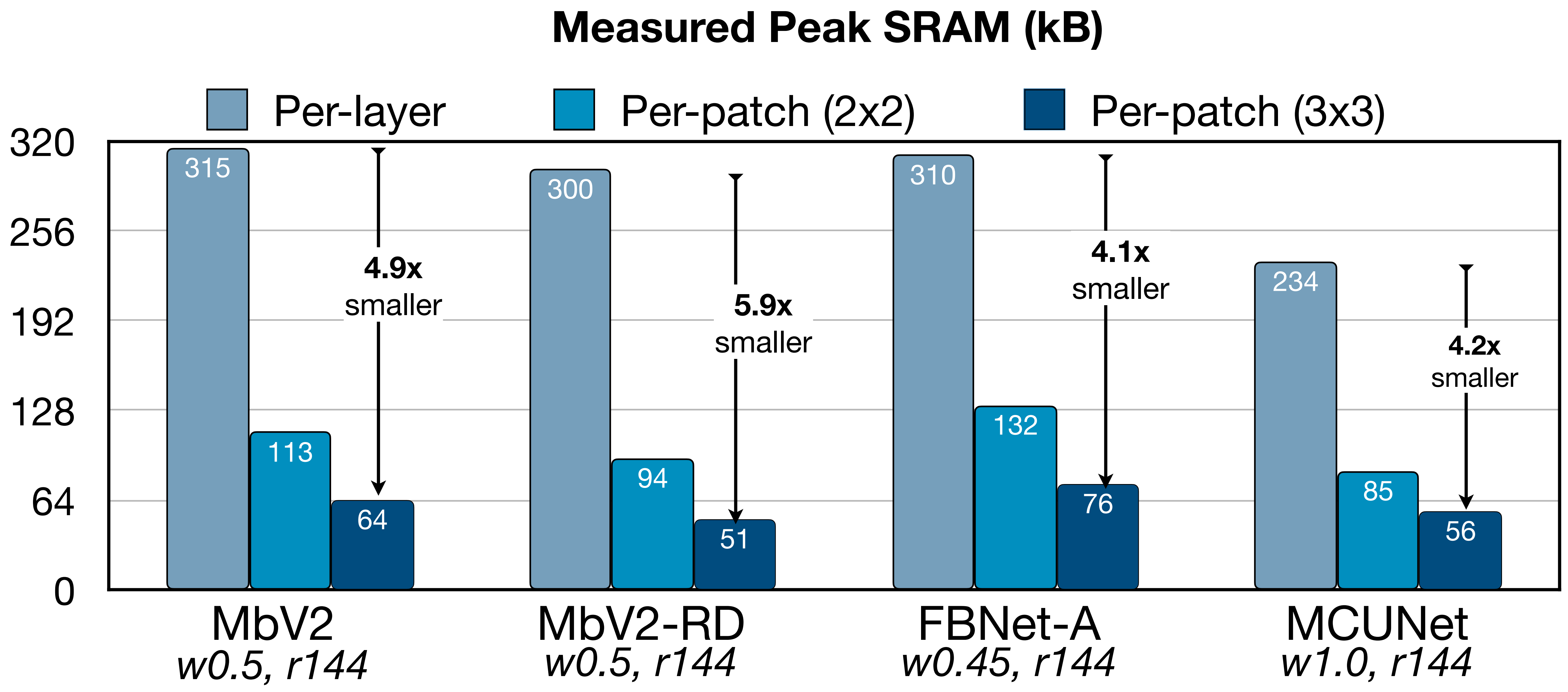
Reducing the Peak Memory of CNNs

- Baseline: TinyEngine, the SOTA system stack for tinyML
- Measured on STM32F746 MCU



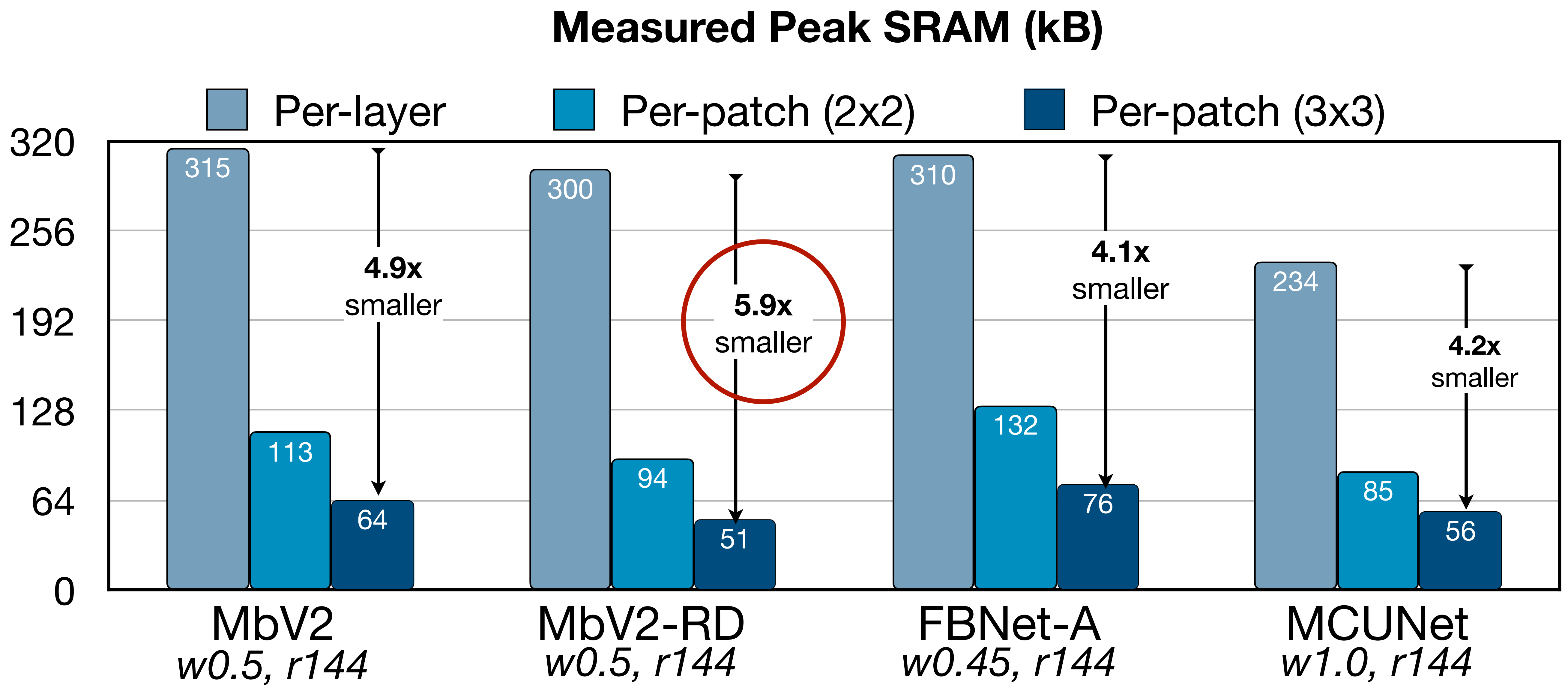
Reducing the Peak Memory of CNNs

- Baseline: TinyEngine, the SOTA system stack for tinyML
- Measured on STM32F746 MCU



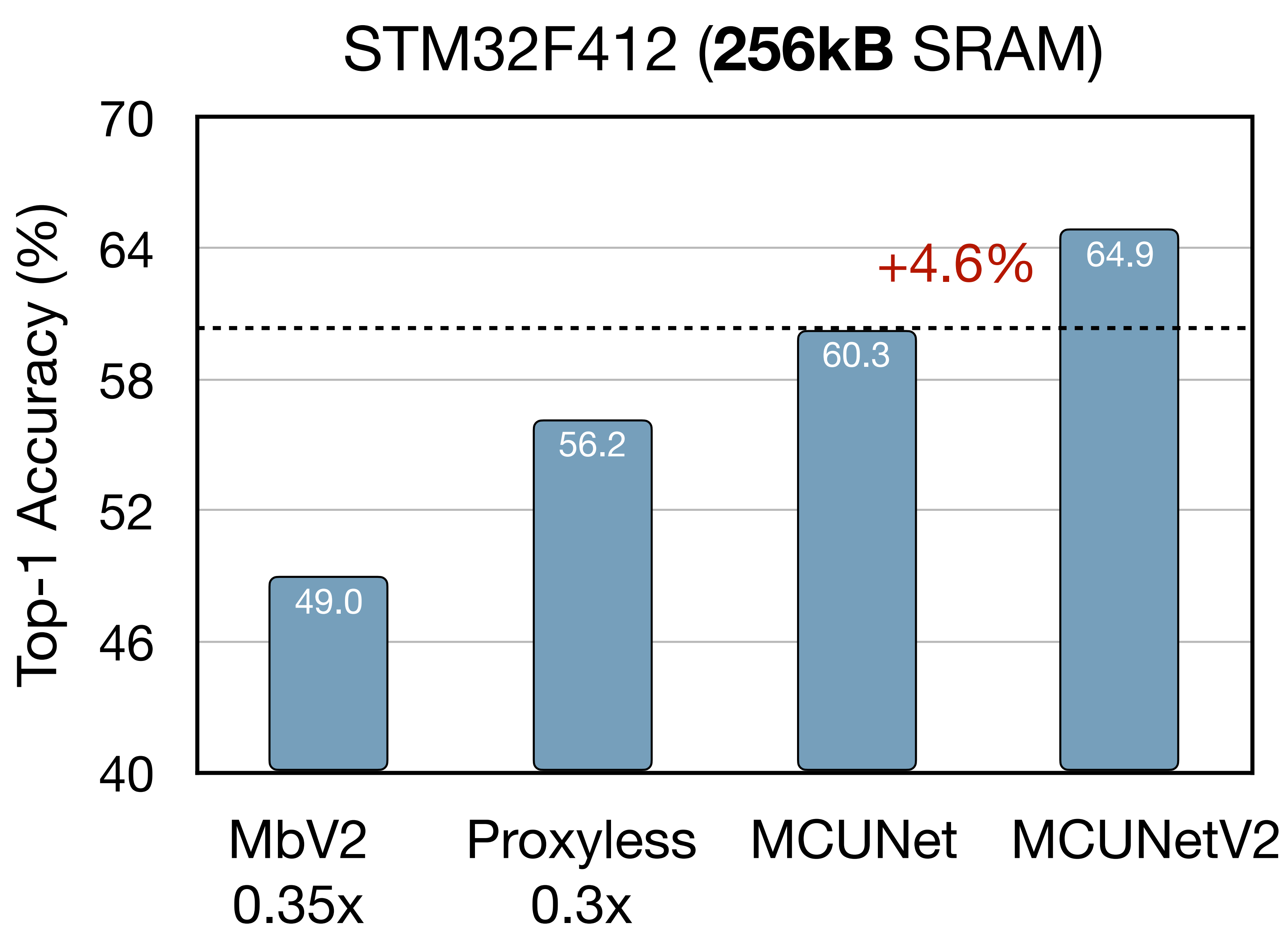
Reducing the Peak Memory of CNNs

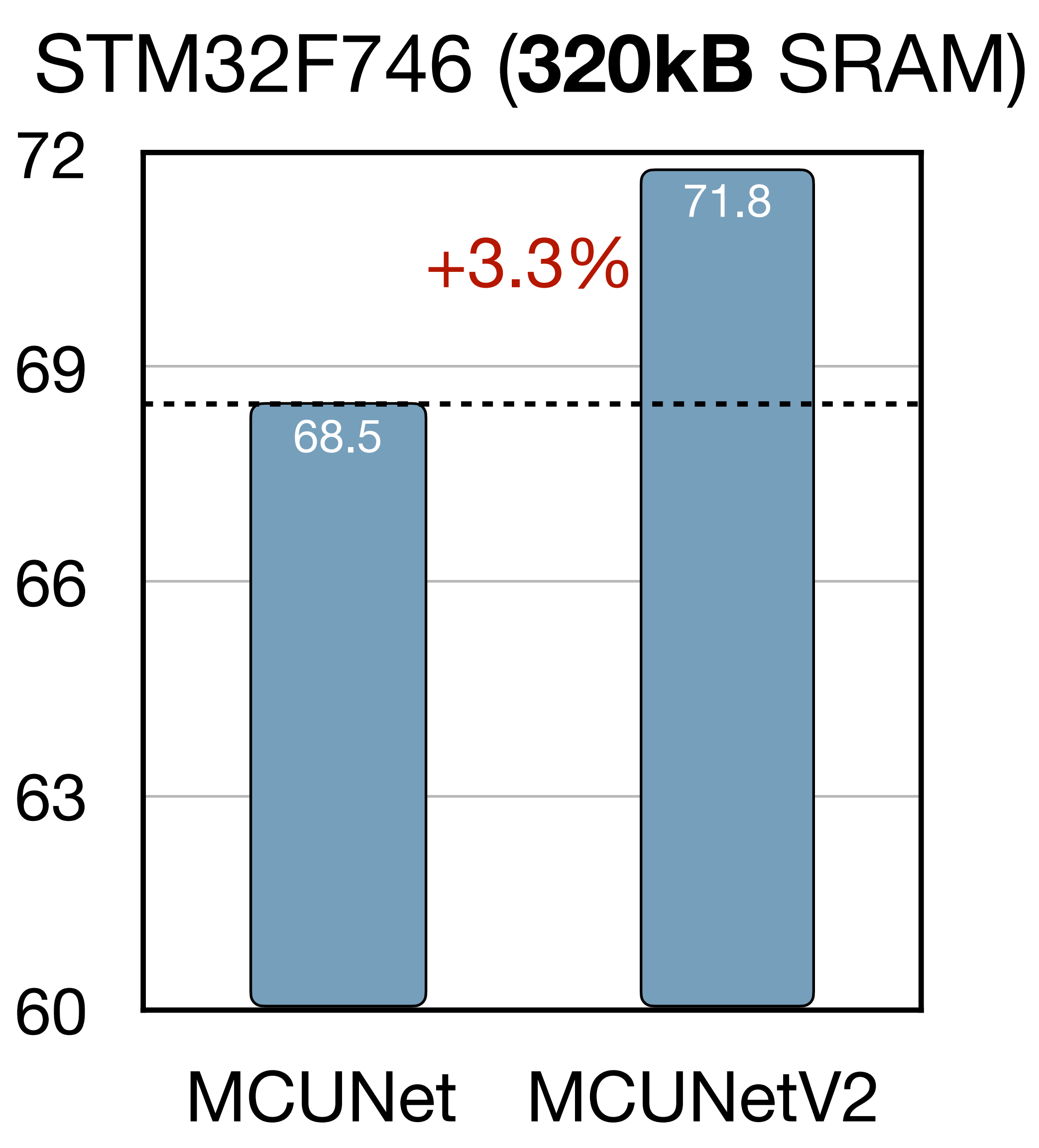
- Baseline: TinyEngine, the SOTA system stack for tinyML
- Measured on STM32F746 MCU



MCUNetV2 for Tiny Image Classification

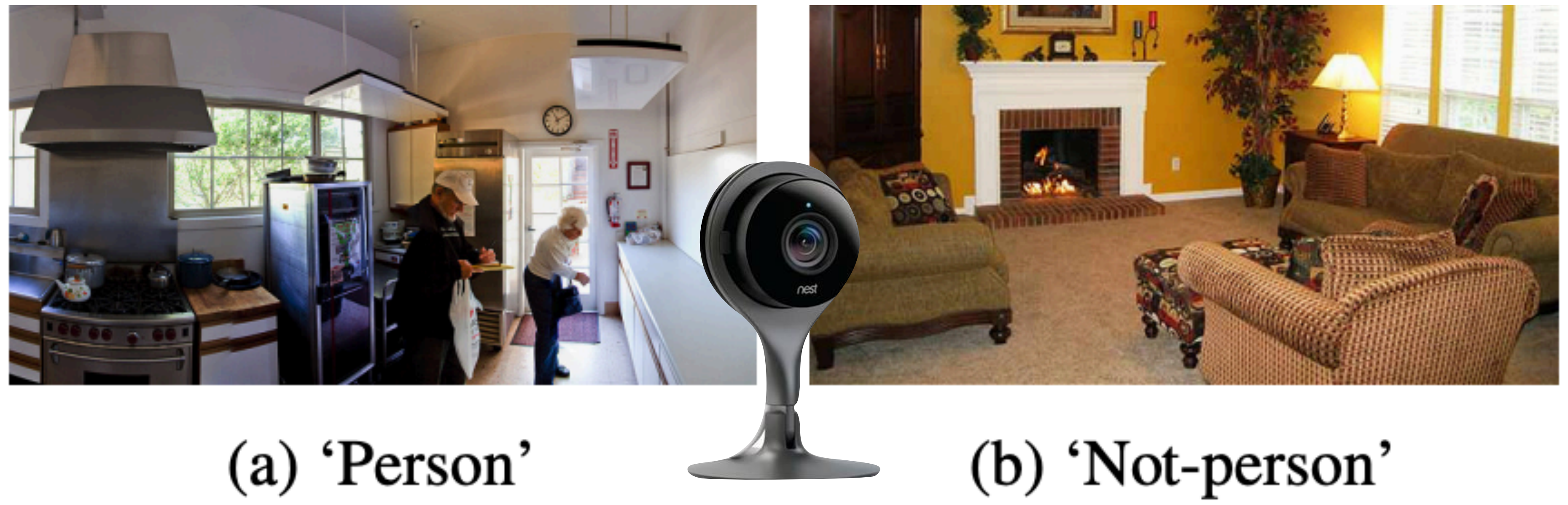
- Large-scale **ImageNet** classification
- Models are quantized to int8
- Serving using TinyEngine.



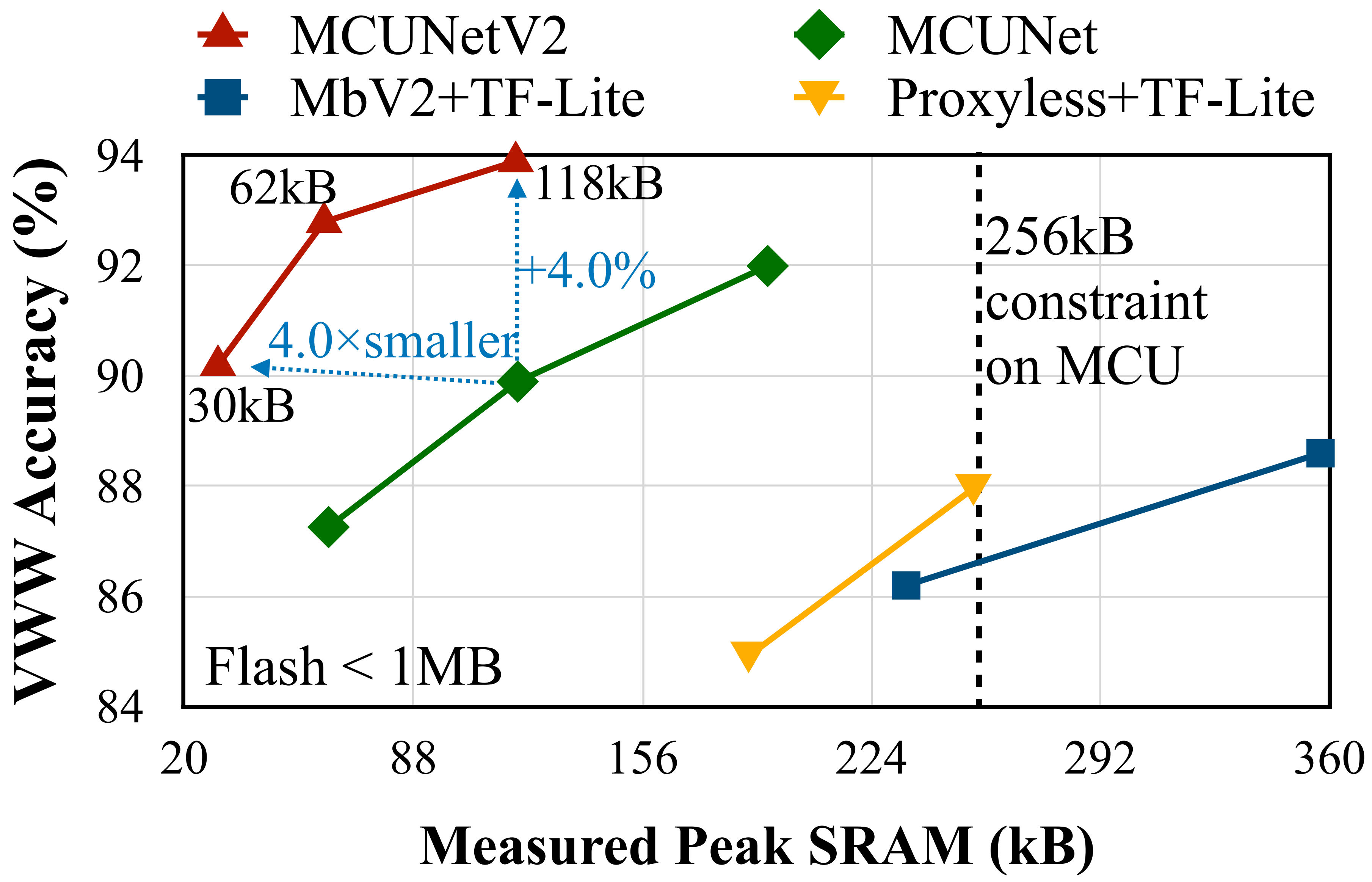


MCUNetV2 for Tiny Image Classification

- TinyML application: **Visual Wake Words (VWW)**
- Higher accuracy, lower SRAM

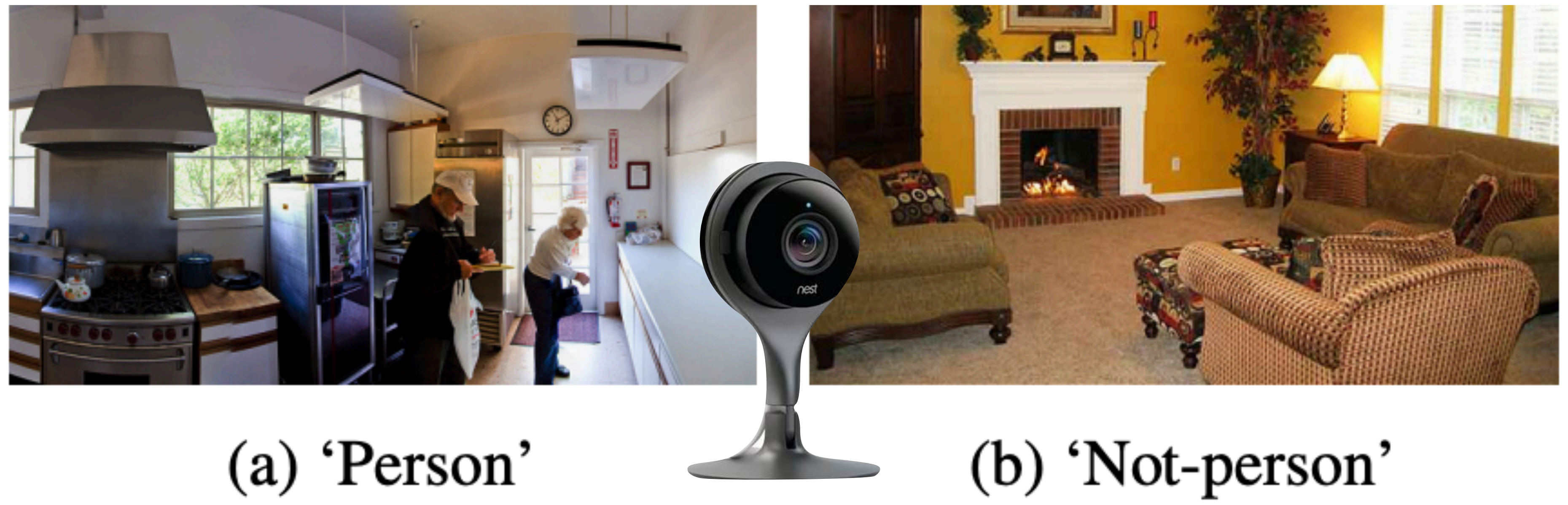


(a) 'Person' (b) 'Not-person'

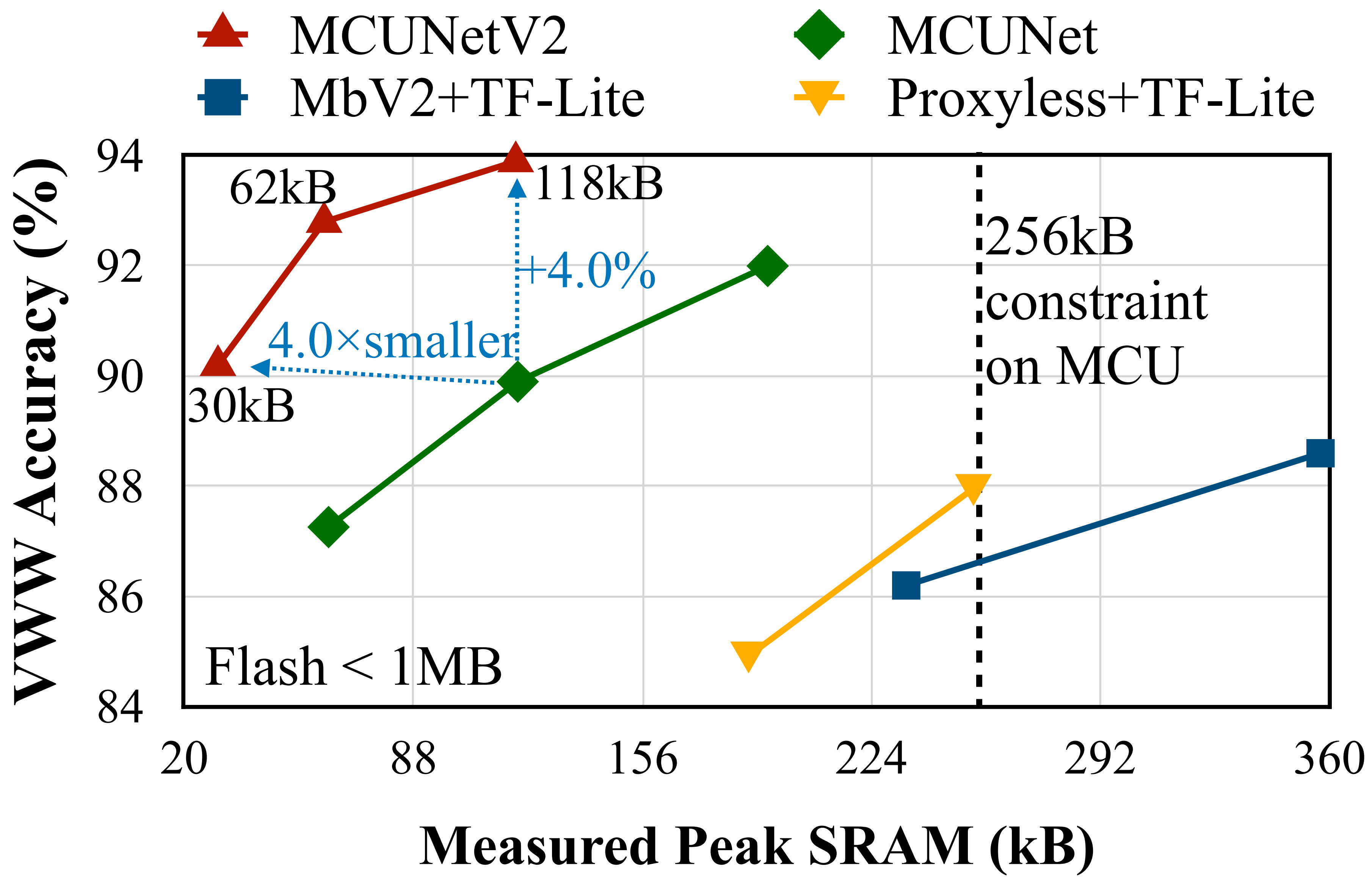


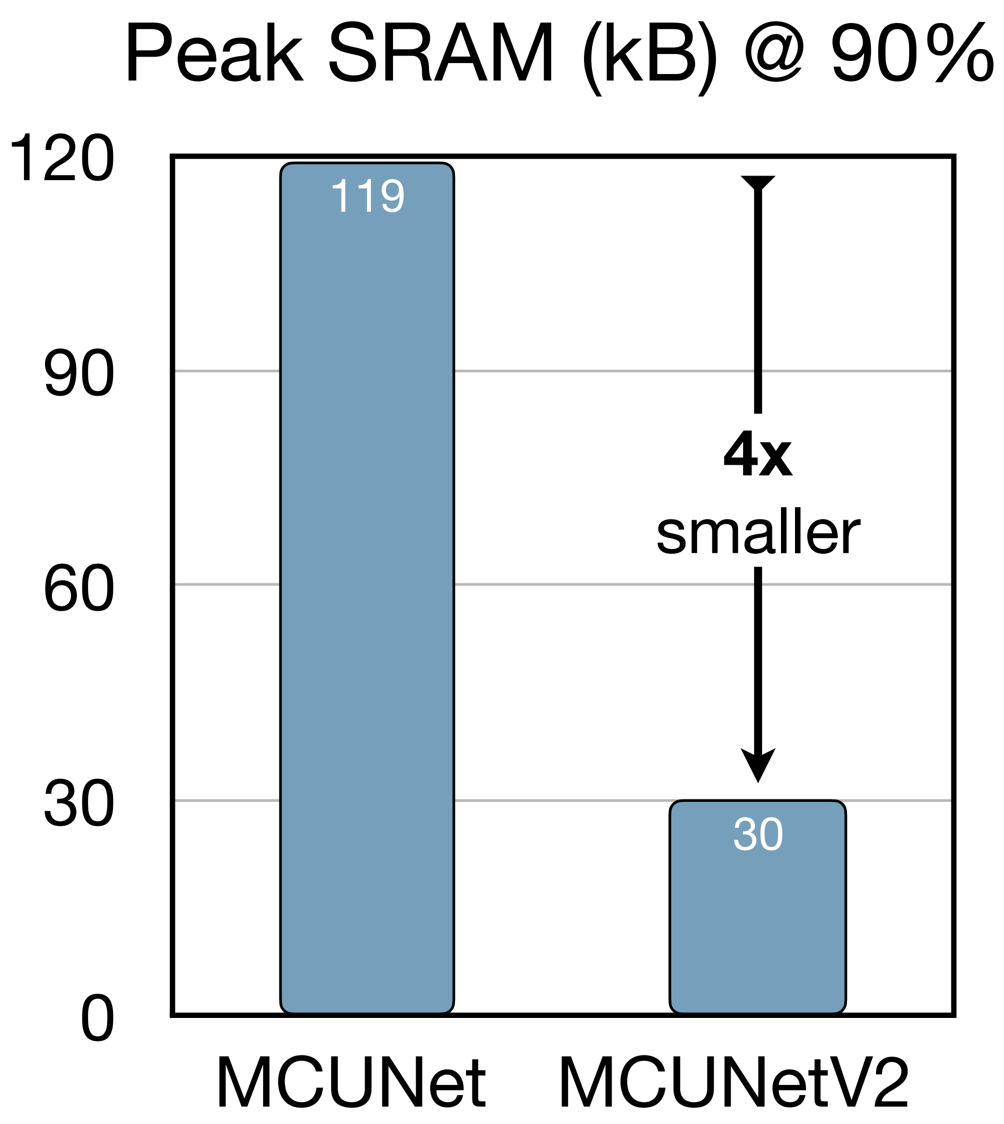
MCUNetV2 for Tiny Image Classification

- TinyML application: **Visual Wake Words (VWW)**
- Higher accuracy, lower SRAM



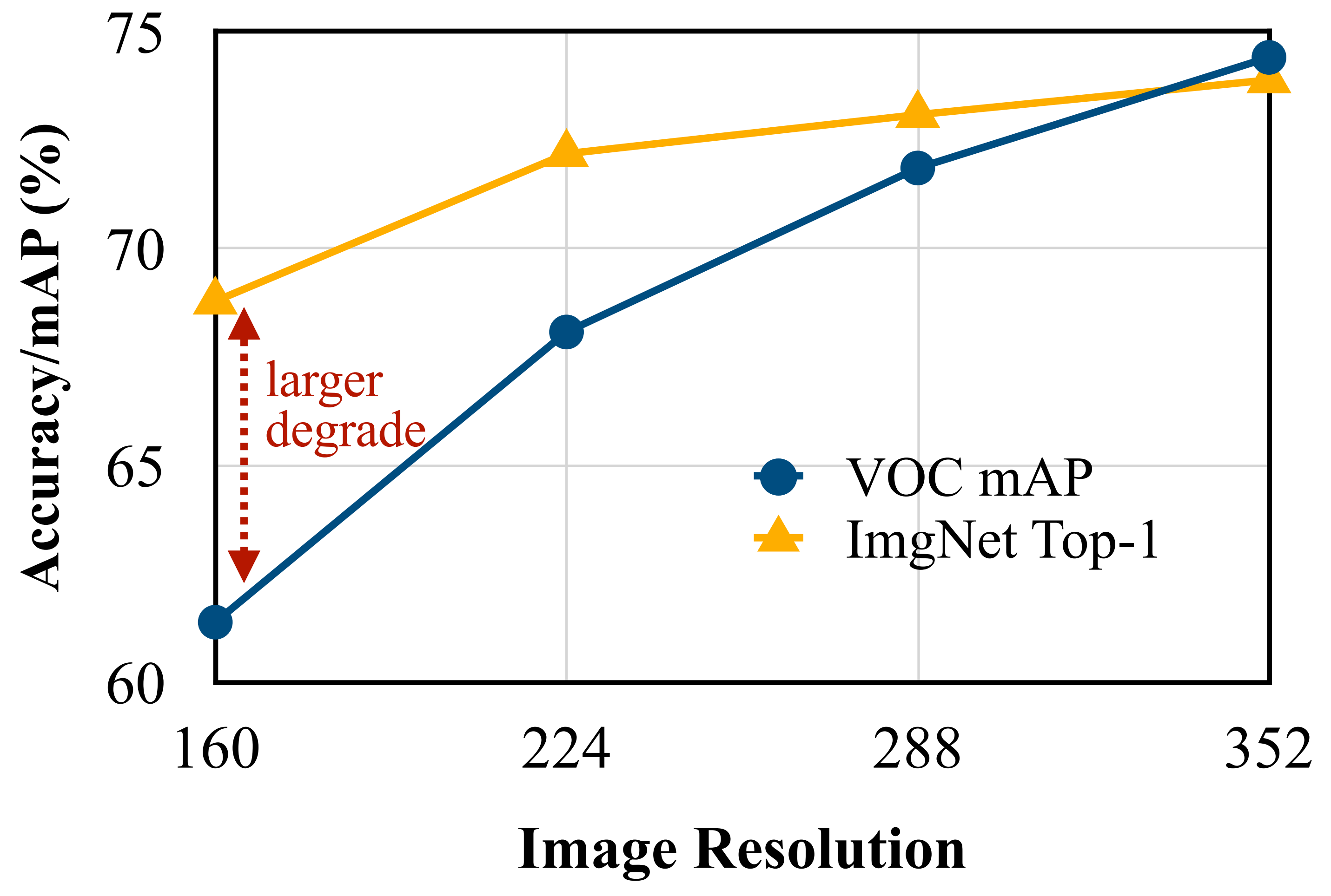
(a) 'Person' (b) 'Not-person'





MCUNetV2 for Tiny Object Detection

- Object detection is more sensitive to input resolution

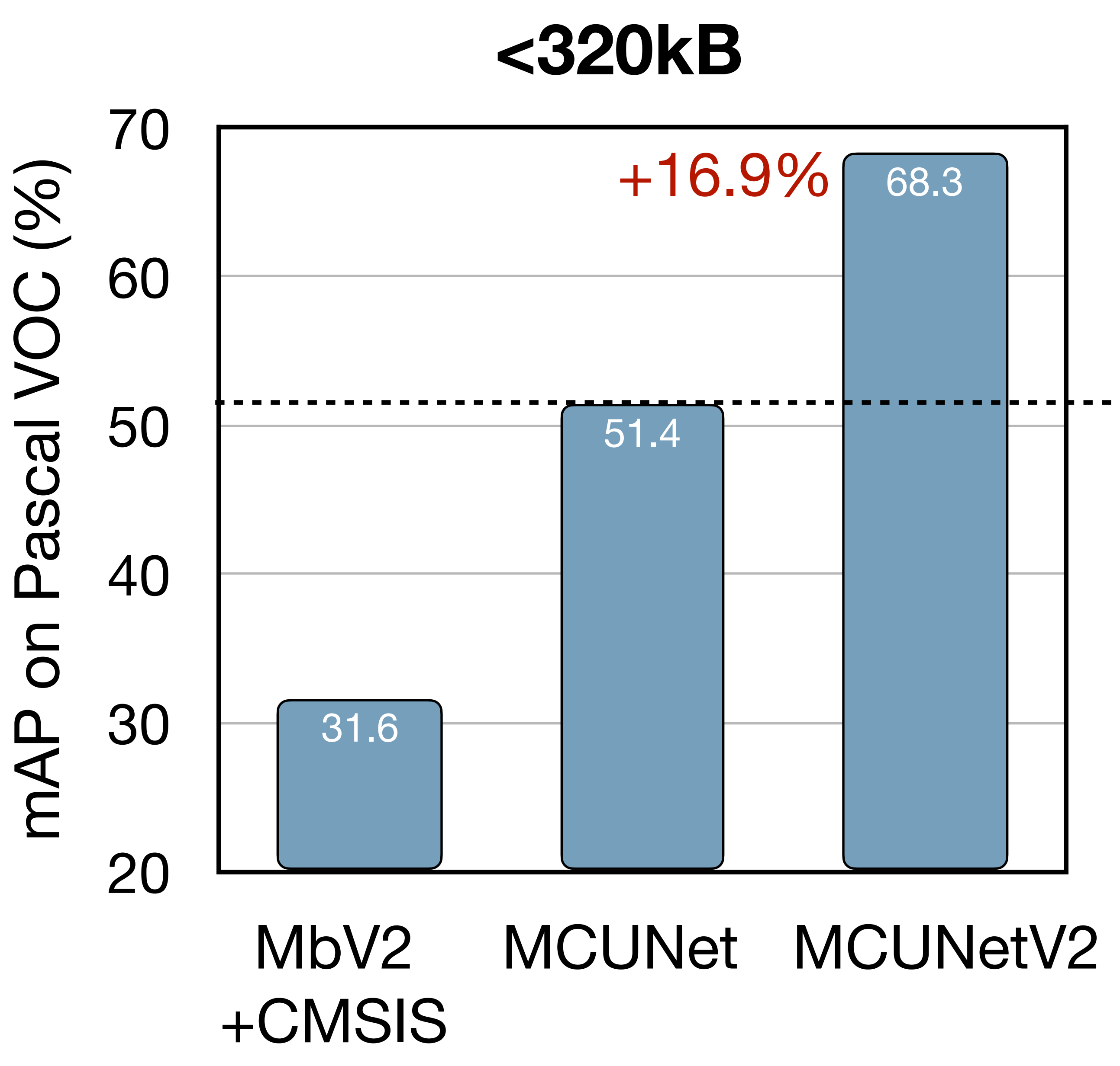


MCUNetV2 for Tiny Object Detection

- Object detection is more sensitive to input resolution
- Patch-based inference allows for a larger resolution, improving detection performance

MCUNetV2 for Tiny Object Detection

- Object detection is more sensitive to input resolution
- Patch-based inference allows for a larger resolution, improving detection performance

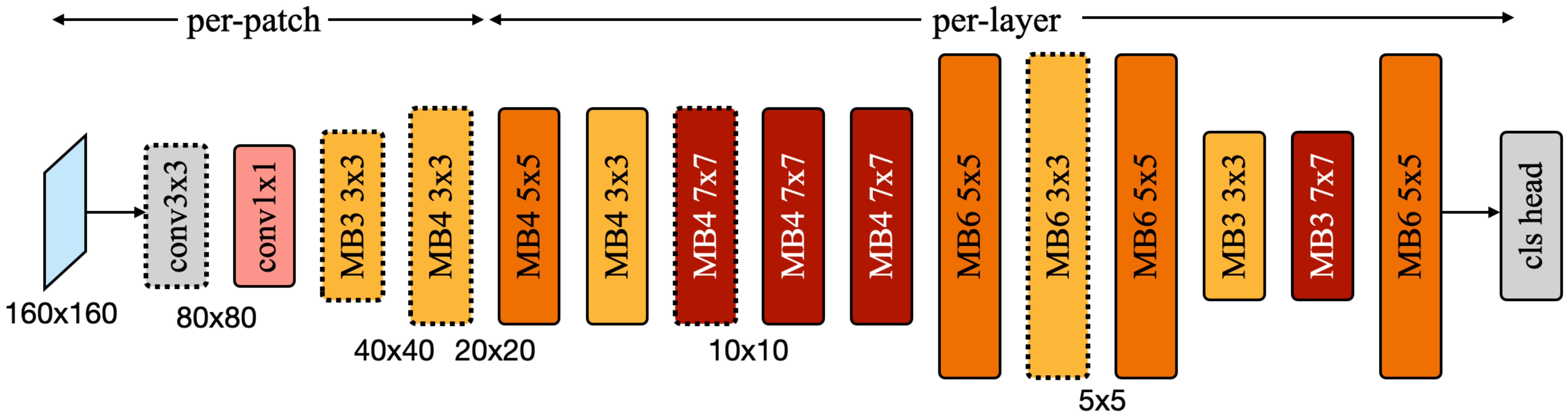


MCUNetV2 for Tiny Object Detection

- Face detection on WIDER Face
- More robust results at a smaller peak memory

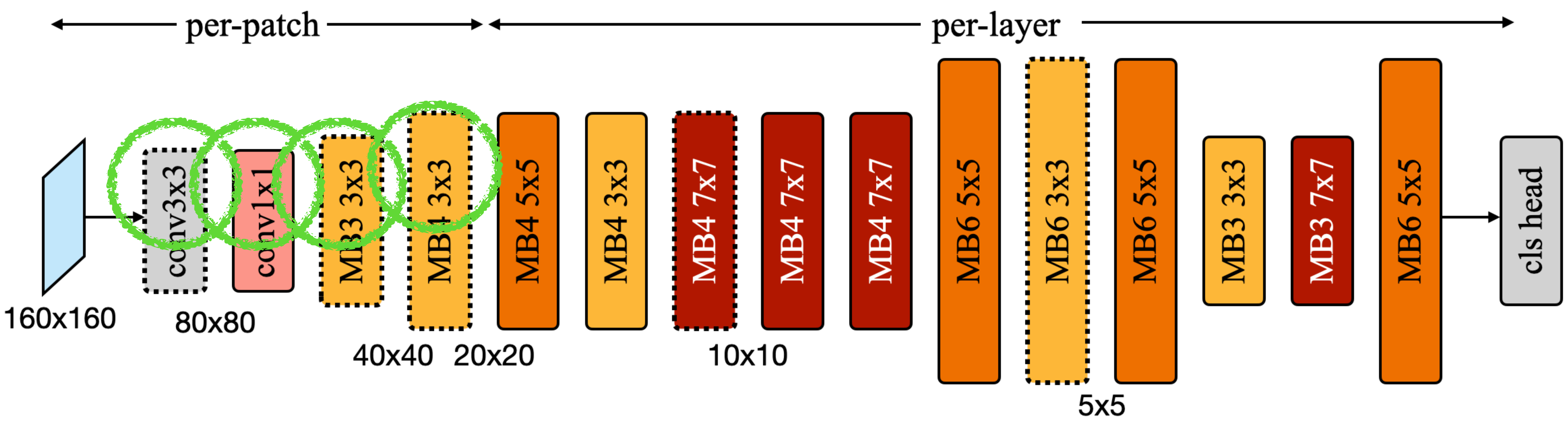
(a) RNNPool-Face-Quant (b) MCUNetV2

Dissecting MCUNetV2 Architecture



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

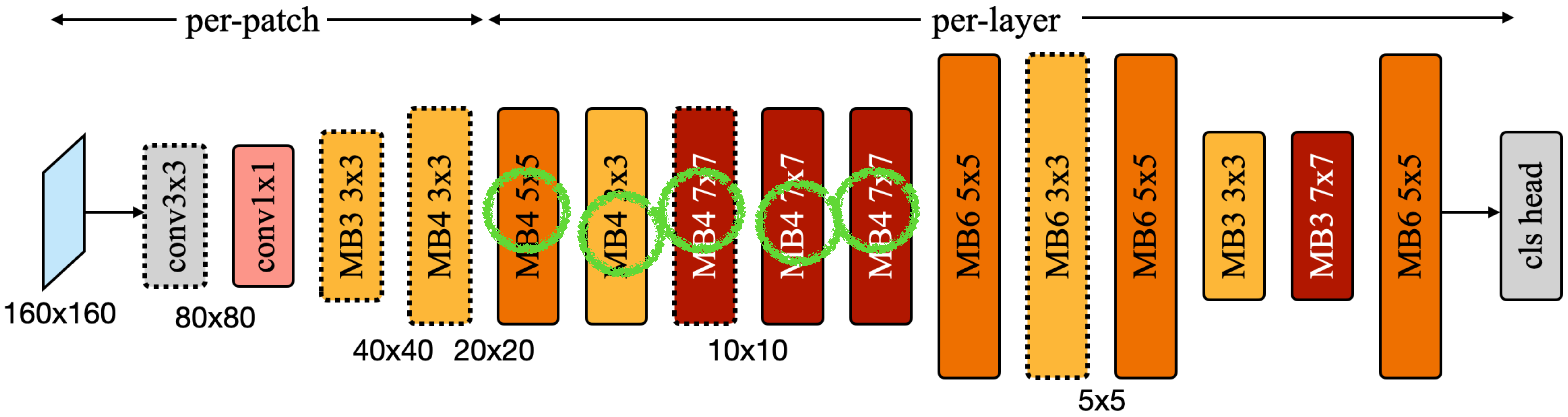
Dissecting MCUNetV2 Architecture



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

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

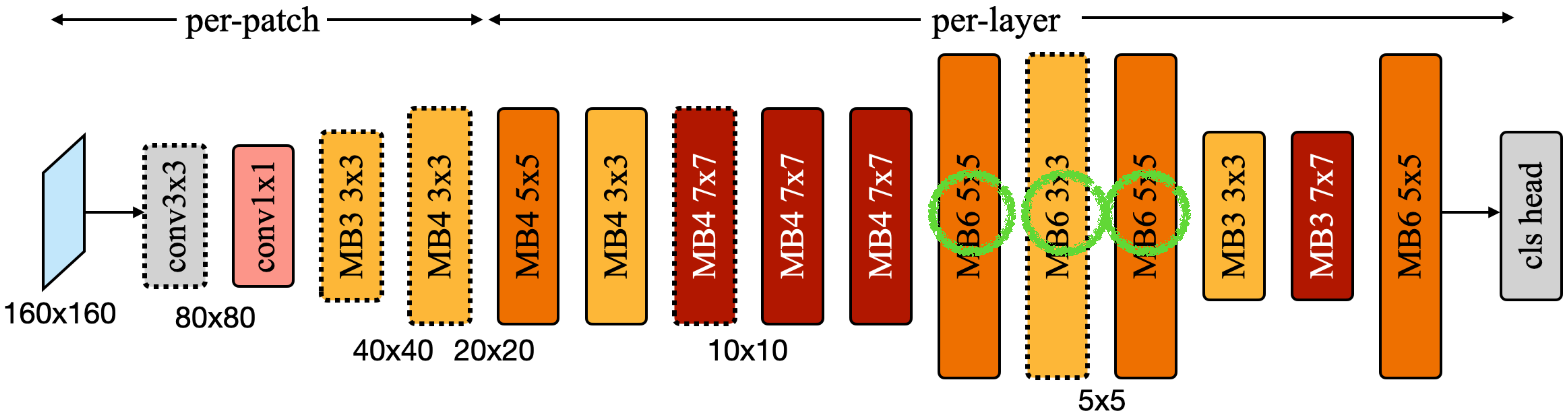
Dissecting MCUNetV2 Architecture



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

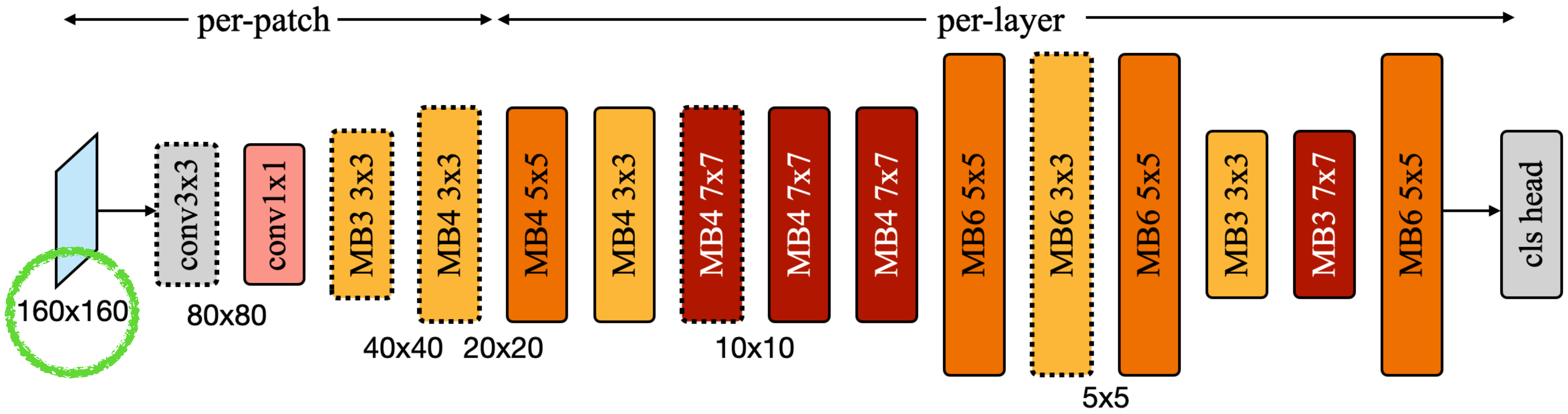
Dissecting MCUNetV2 Architecture



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.

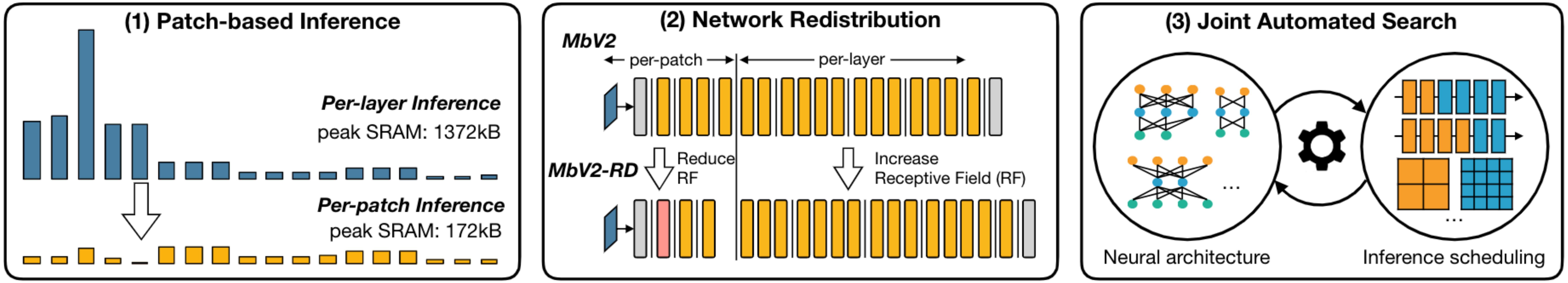
Dissecting MCUNetV2 Architecture



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)

Thanks for listening!



MCUNetV2