Putting AI on a Diet:

TinyML and Efficient Deep Learning

Song Han
Assistant Professor
Massachusetts Institute of Technology

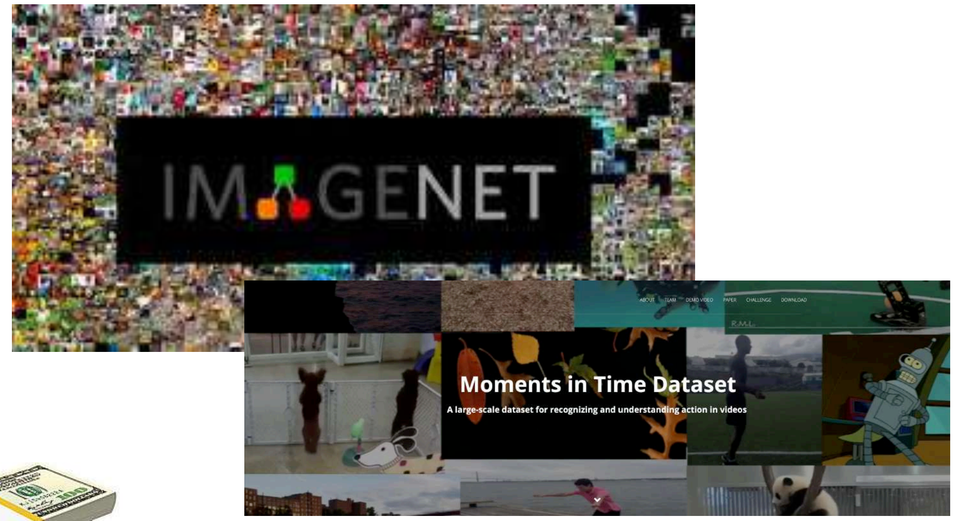
https://songhan.mit.edu

Challenges for Deep Learning:

computation-hungry, data-hungry

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

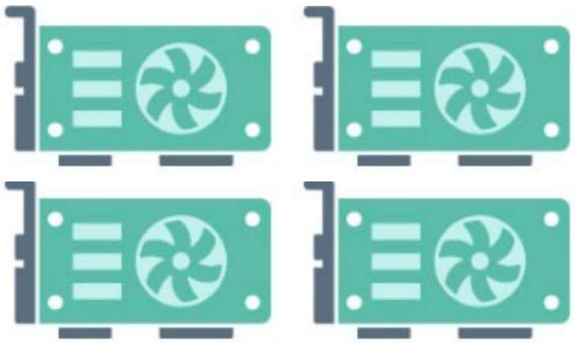
ImageNet / Moments in Time
>1 million images / videos



Cloud AI: increases TCO & carbon
Mobile AI: drains battery
Tiny AI: can't fit memory

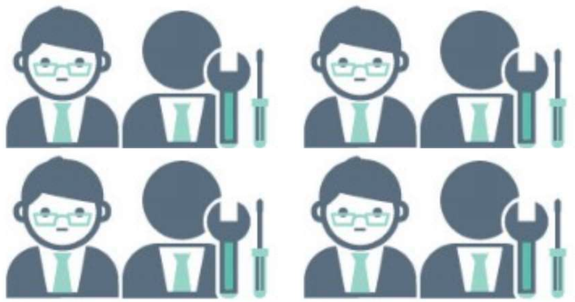
Lots of efforts to collect/label

TinyML and Efficient Deep Learning

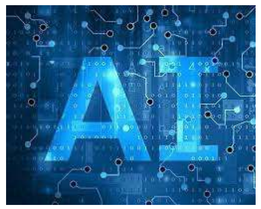


A lot of computation

A lot of carbon



Many engineers



A lot of data

TinyML and Efficient Deep Learning

Three aspects

less computation
less carbon

TinyML

fewer engineers

less data

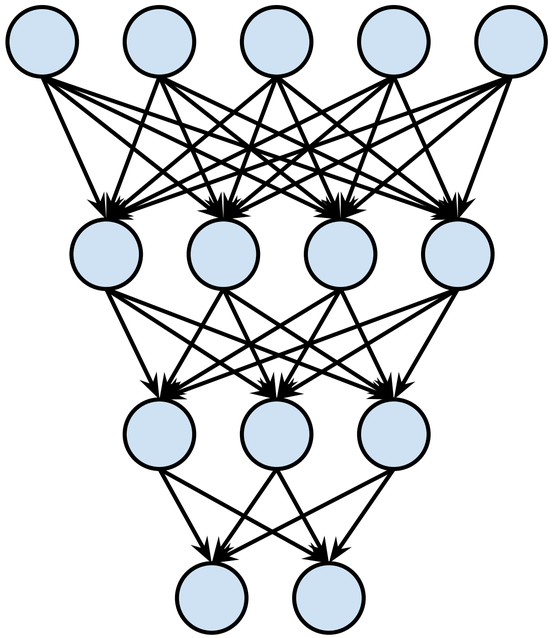
Deep Compression

Make AI run Fast and Efficiently
with Limited Hardware Resource

Han, Mao, Dally, Deep Compression, ICLR'16, best paper award

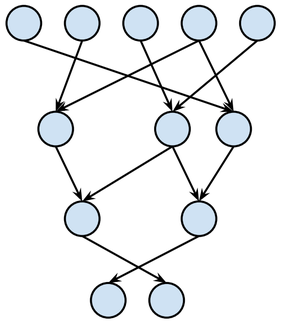
Deep Compression

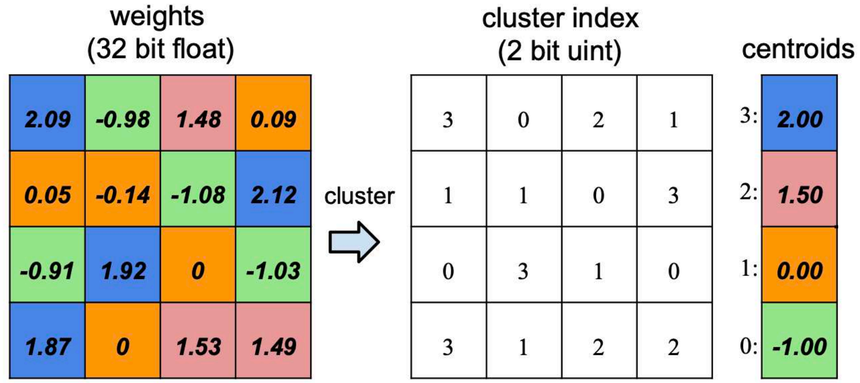
Make AI run Fast and Efficiently
with Limited Hardware Resource

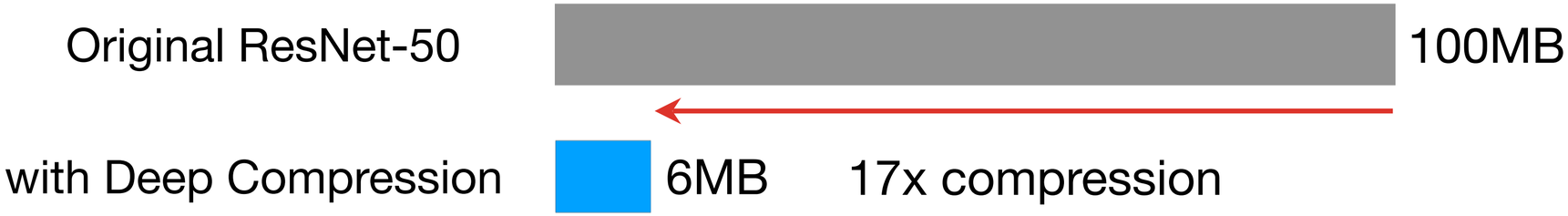


Pruning

Deep Compression

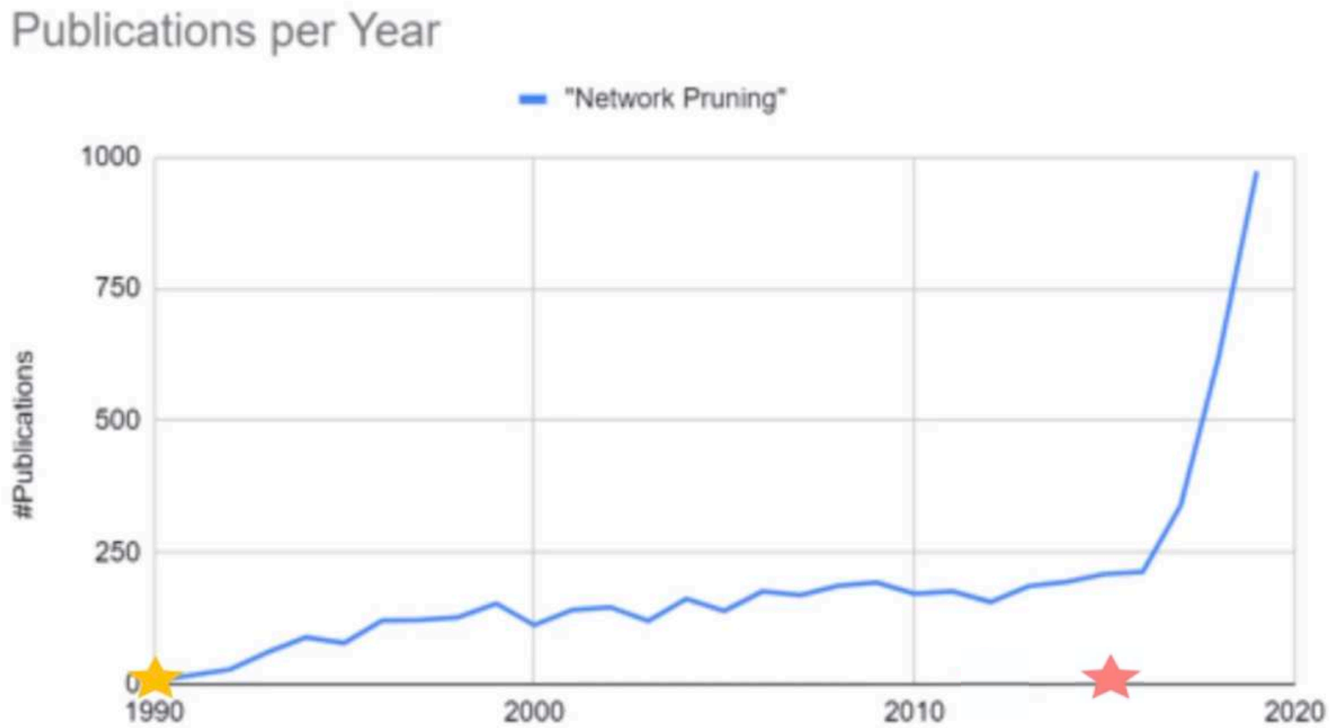


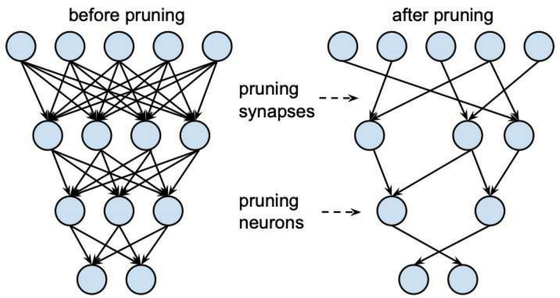




Pruning & Sparsity

Increased attention since 2015





Han et al., NIPS'15

Optimal Brain Damage

Yann Le Cun, John S. Denker and Sara A. Solla
AT&T Bell Laboratories, Holmdel, N. J. 07733

Learning both Weights and Connections for Efficient Neural Networks

Song Han
Stanford University
songhan@stanford.edu

Jeff Pool
NVIDIA
jpool@nvidia.com

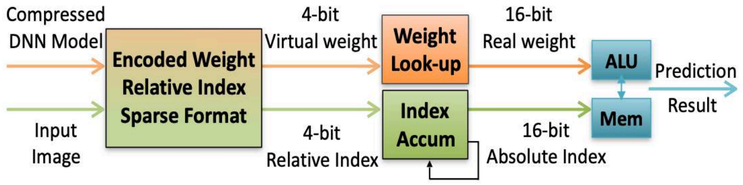
John Tran
NVIDIA
johntran@nvidia.com

William J. Dally
Stanford University
NVIDIA
dally@stanford.edu

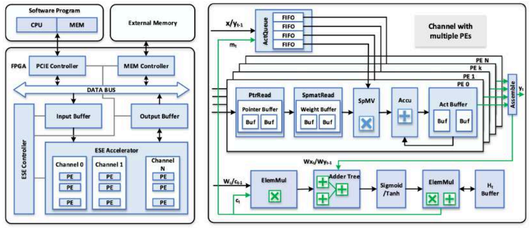
the curve credits to NVIDIA

Pruning & Sparsity

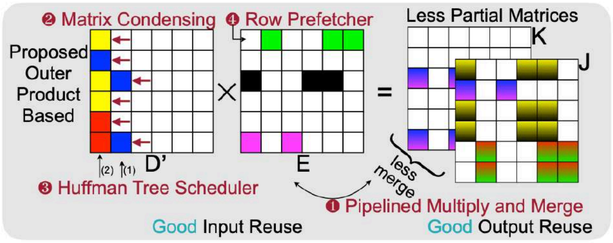
applied by industry:



EIE, Han et al, ISCA'16

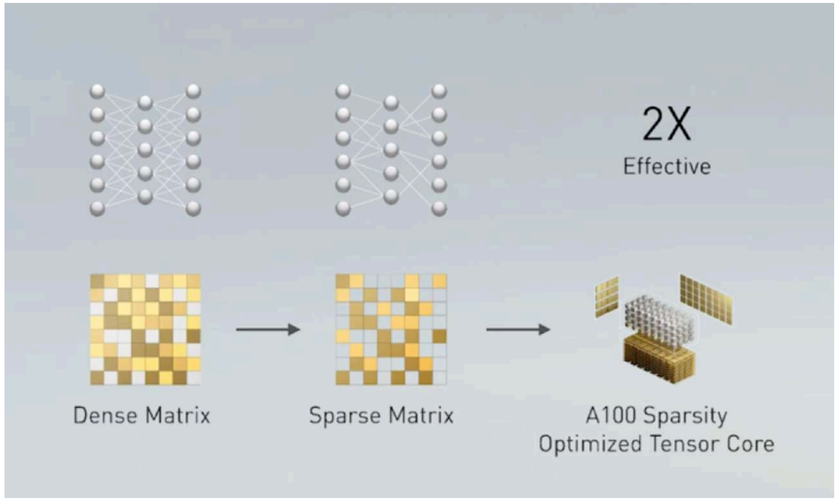


ESE, Han et al, FPGA'17



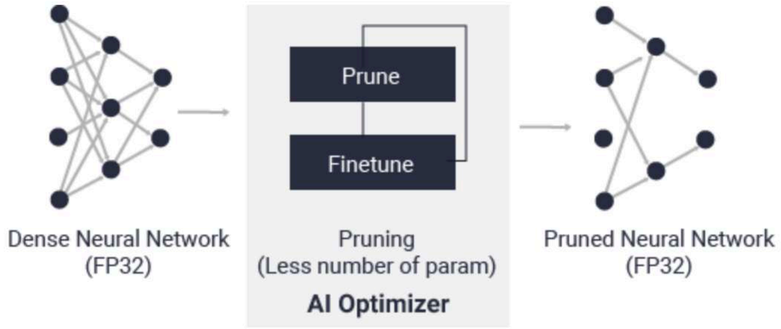
SpArch, Zhang et al, HPCA'20
SpAtten, Wang et al, HPCA'21

NVIDIA



2:4 sparsity in A100 GPU
2X peak performance,
1.5X measured BERT speedup

XILINX VITIS™



Reduce model complexity by 5x to 50x with minimal accuracy impact. Deep Compression takes the performance of your AI inference to the next level.

TinyML and Efficient Deep Learning

- **AutoML and NAS**
  - Once-for-all Network [ICLR'19]

- **Less Computation**
  - Inference: MCUNet for IoT Devices [NeurIPS'20, spotlight]
  - Training: Tiny On-Device Transfer Learning (TinyTL) [NeurIPS'20]

- **Less Training Data**
  - Differentiable Augmentation for Data-Efficient GAN Training [NeurIPS'20]

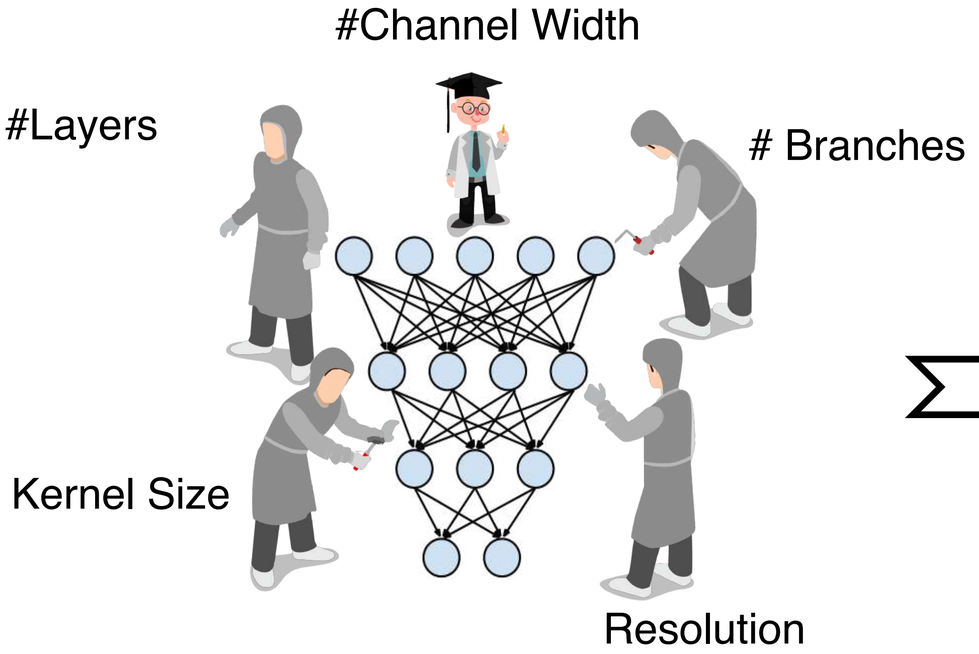
TinyML and Efficient Deep Learning

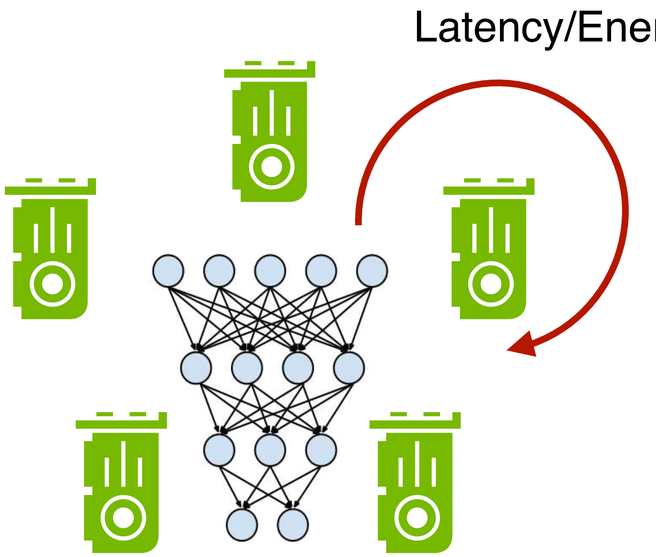
- **AutoML and NAS**
  - Once-for-all Network [ICLR'19]

- **Less Computation**
  - Inference: MCUNet for IoT Devices [NeurIPS'20, spotlight]
  - Training: Tiny On-Device Transfer Learning (TinyTL) [NeurIPS'20]

- **Less Training Data**
  - Differentiable Augmentation for Data-Efficient GAN Training [NeurIPS'20]

Hardware-aware Neural Architecture Search





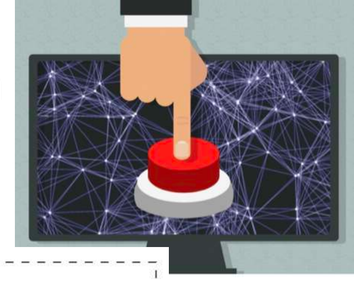
Manual Design:
black magic, nutoriously hard to tune

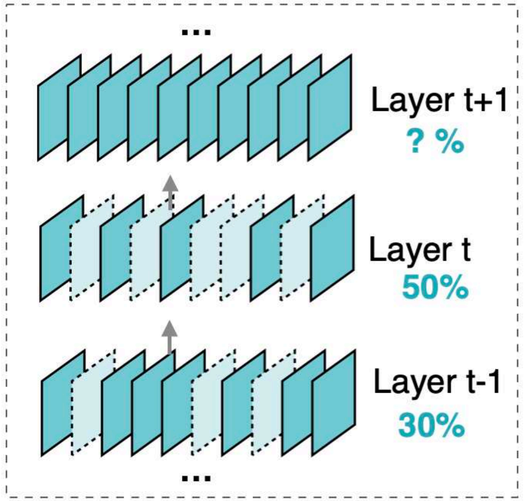
Automatic Design:
Synthesize NNs to fit latency/accuracy/memory constraints

EDA tool : Circuit = NAS : Neural net

AutoML and Neural Architecture Search

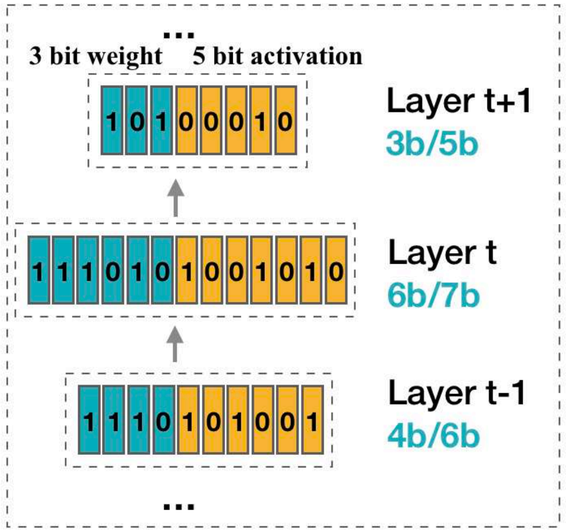
push-button solultion
for efficient NN design





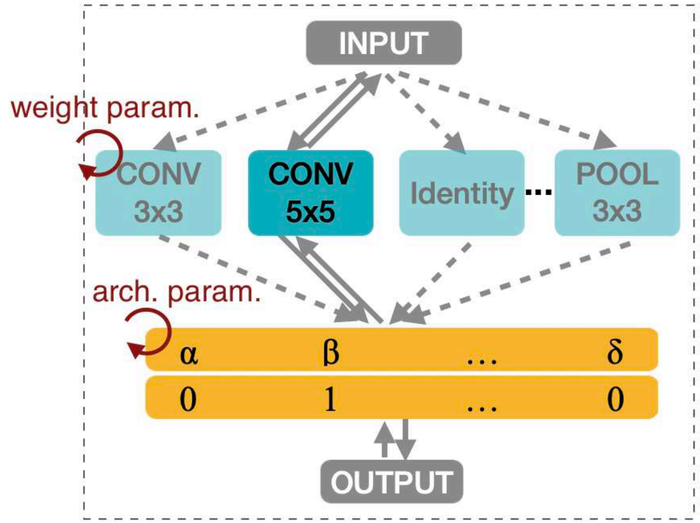
AMC: AutoML for Model Compression

[ECCV 2018]



HAQ: Hardware-aware Automated Quantization

[CVPR 2019], oral
integrated by Intel OpenVINO

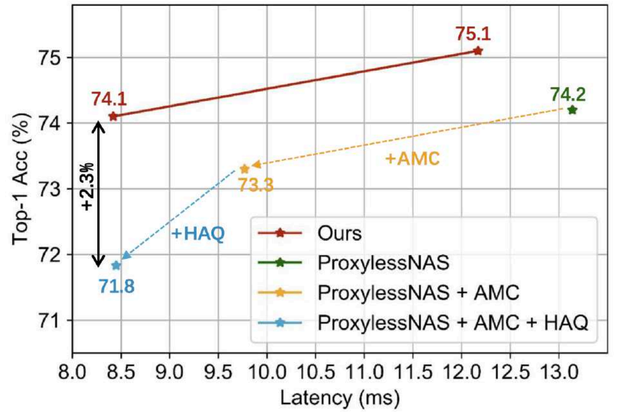


Proxyless Neural Architecture Search

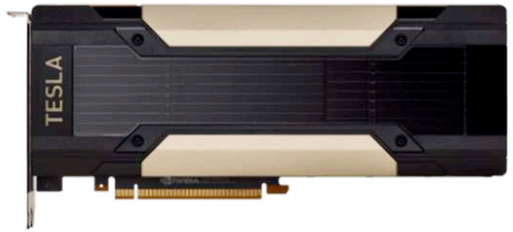
[ICLR 2019]
integrated by Facebook PyTorch and Amazon AutoGluon

Combine them together: APQ: Joint Search for Network Architecture, Pruning and Quantization Policy [CVPR'20]

1st Place of Visual Wakeup Words (VWW) Challenge, host by Google @CVPR2019



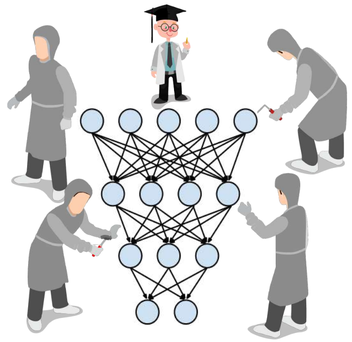
How to design efficient NN models for diverse hardware platforms?



GPU, 16GB

Mobile phone, 4GB

Cortex M4
(256kB/1MB)



Lots of hand tuning for different devices!
layers, # channels, resolution, kernel size…

Hardware-aware Neural Architecture Search

2019

Design Cost (GPU hours)

200

For training iterations:
    forward-backward();

The design cost is calculated under the assumption of using MobileNet-v2.

Hardware-aware Neural Architecture Search

2019

Design Cost (GPU hours)

40K

For search episodes: // meta controller

  For training iterations:

    forward-backward(); **Expensive**

  If good_model: **break;**

For post-search training iterations:

  forward-backward(); **Expensive**

The design cost is calculated under the assumption of using MnasNet.

[1] Tan, Mingxing, et al. "Mnasnet: Platform-aware neural architecture search for mobile." *CVPR*. 2019.

Hardware-aware Neural Architecture Search

Diverse Hardware Platforms

2019 2017 2014 2010

For devices:

For search episodes: // meta controller

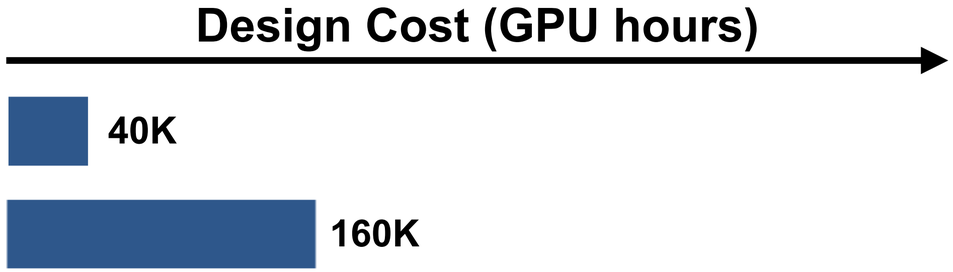
For training iterations:

forward-backward(); **Expensive!**

If good_model: **break;**

For post-search training iterations:

forward-backward(); **Expensive!**

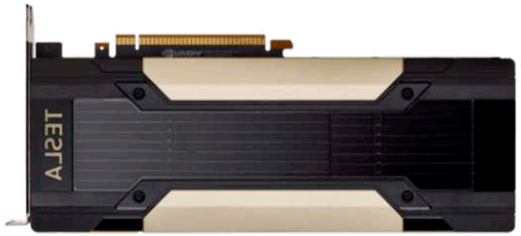


The design cost is calculated under the assumption of using MnasNet.

[1] Tan, Mingxing, et al. "Mnasnet: Platform-aware neural architecture search for mobile." *CVPR*. 2019.

Hardware-aware Neural Architecture Search

Diverse Hardware Platforms



...

Cloud AI (10^{12} FLOPS)

Mobile AI (10^{9} FLOPS)

Tiny AI (10^{6} FLOPS)

For many devices:

  For search episodes: // meta controller

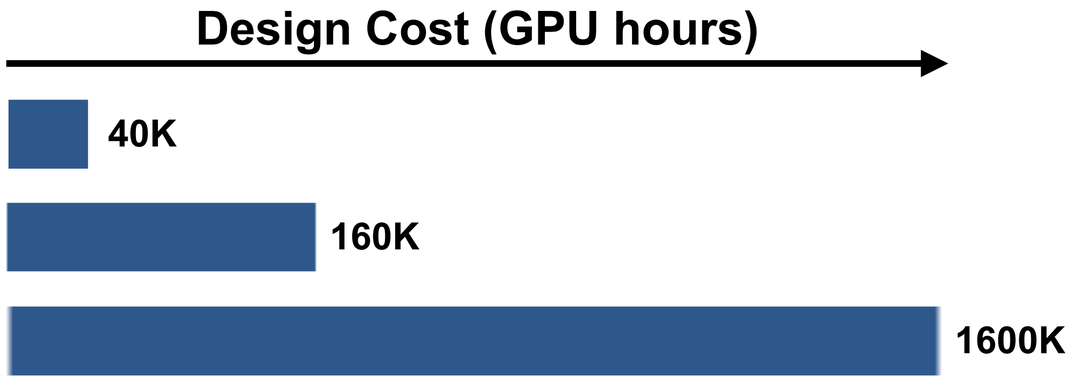
    For training iterations:

      forward-backward(); **Expensive!!**

    If good_model: **break;**

For post-search training iterations:

  forward-backward(); **Expensive!!**

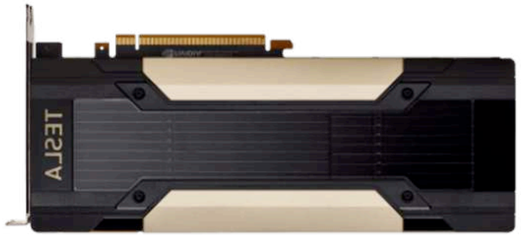


The design cost is calculated under the assumption of using MnasNet.

[1] Tan, Mingxing, et al. "Mnasnet: Platform-aware neural architecture search for mobile." *CVPR*. 2019.

Hardware-aware Neural Architecture Search

Diverse Hardware Platforms



...

Cloud AI (10^{12} FLOPS)

Mobile AI (10^{9} FLOPS)

Tiny AI (10^{6} FLOPS)

For many devices:

  For search episodes: // meta controller

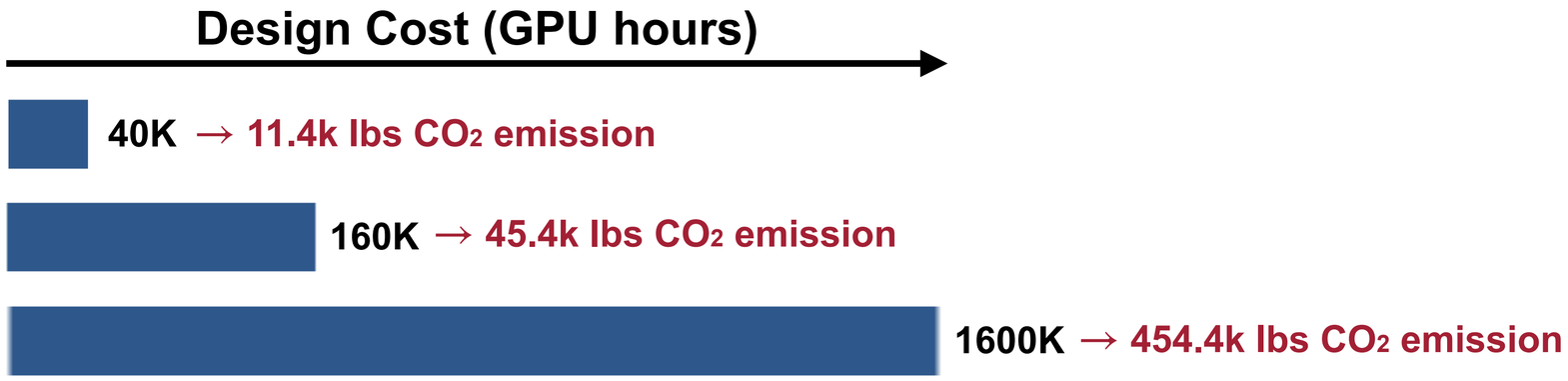
    For training iterations:

      forward-backward(); **Expensive!!**

    If good_model: **break;**

For post-search training iterations:

  forward-backward(); **Expensive!!**



1 GPU hour translates to 0.284 lbs CO_2 emission according to Strubell, Emma, et al. "Energy and policy considerations for deep learning in NLP." *ACL*. 2019.

Today's NAS is too expensive

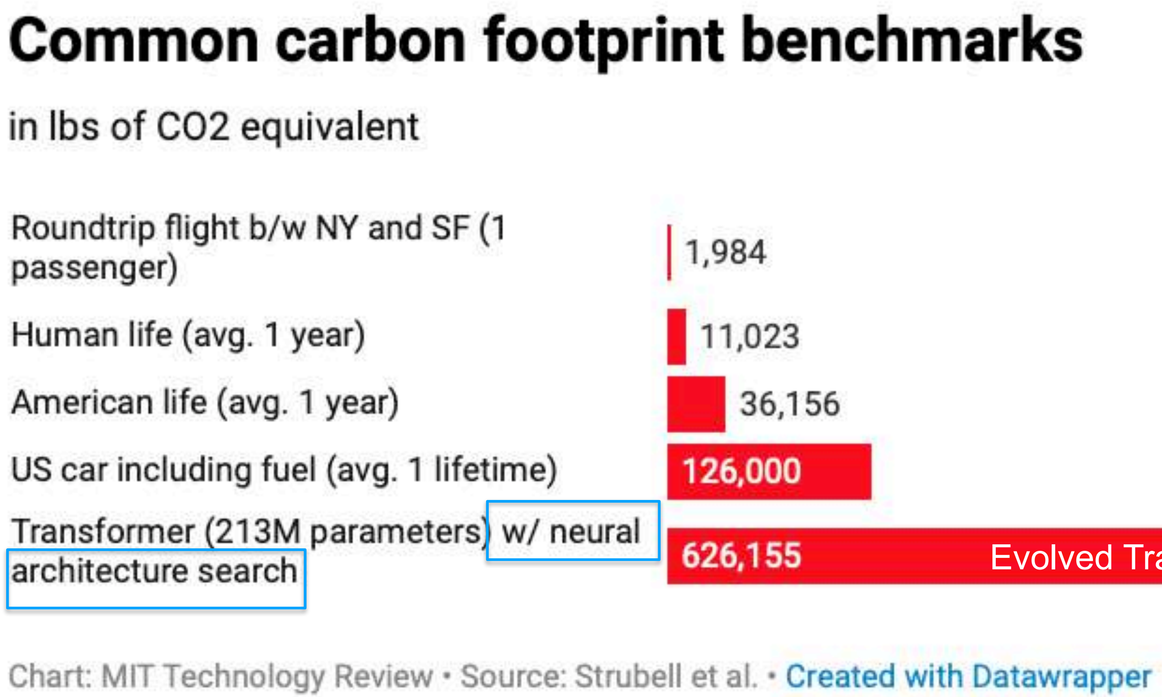
We need Green AI

Artificial intelligence / Machine learning

Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

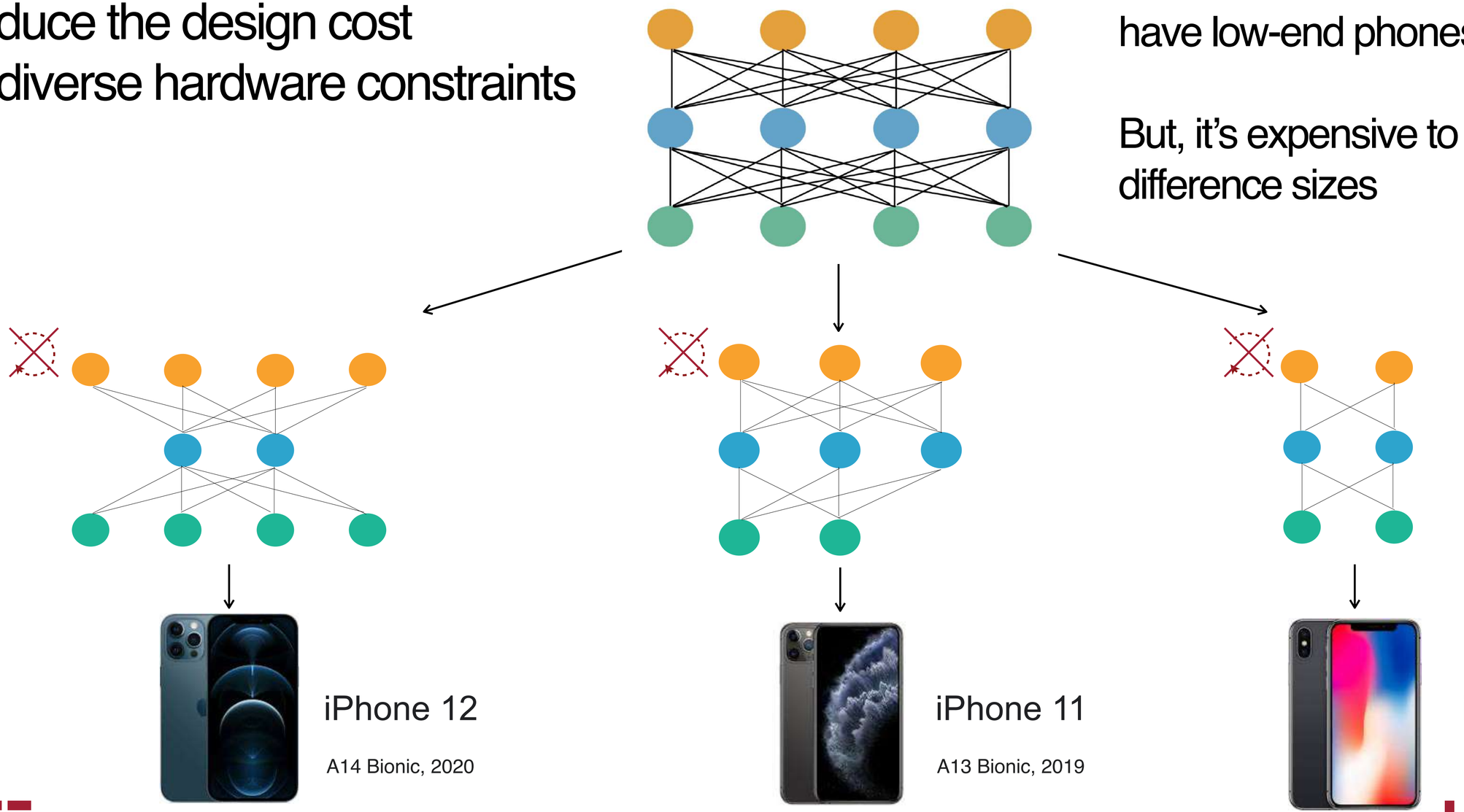
by **Karen Hao** June 6, 2019



ICML'19, ACL'19

Once-for-All Network

Train once, get many
Reduce the design cost
Fit diverse hardware constraints



iPhone 12

A14 Bionic, 2020

iPhone 11

A13 Bionic, 2019

iPhone XS

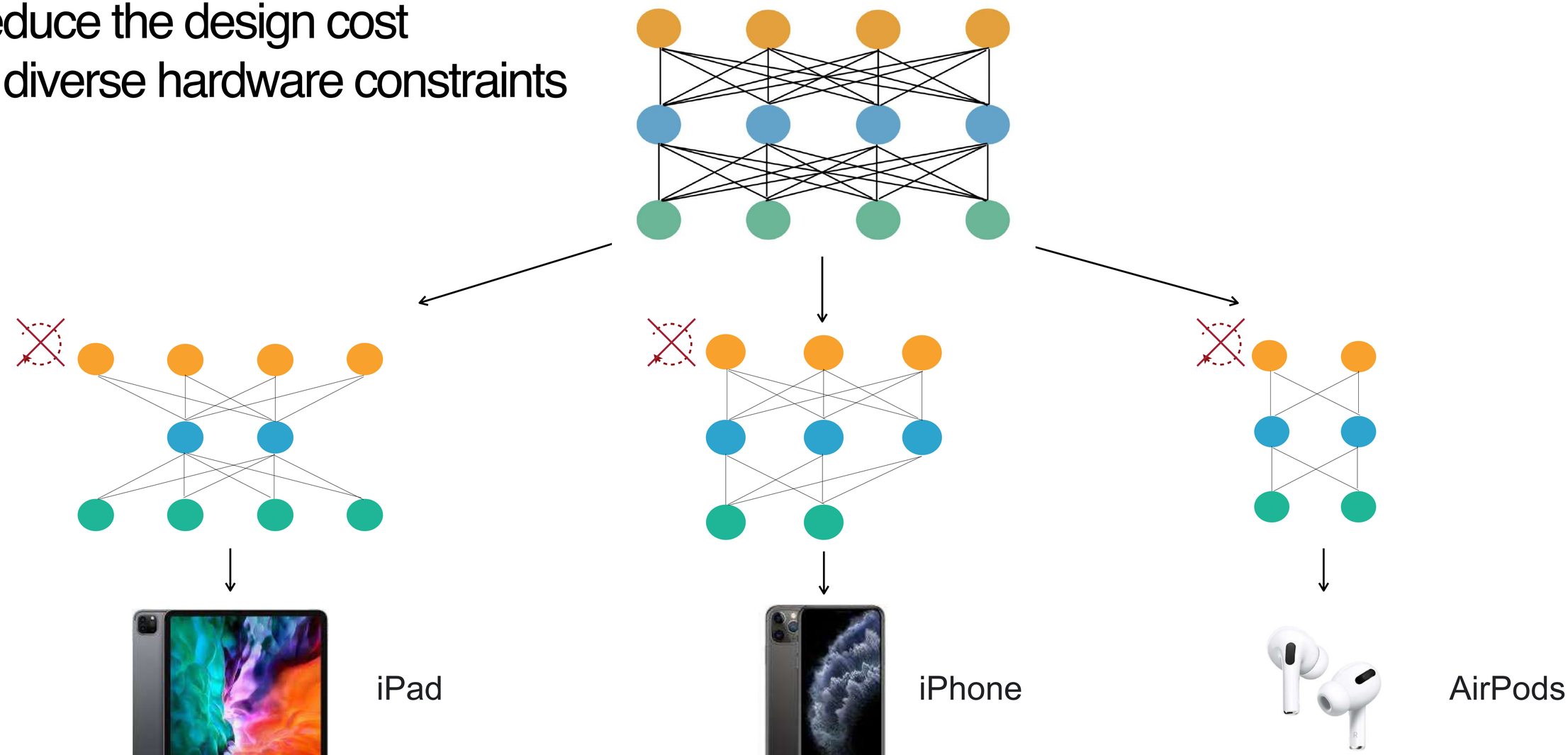
A12 Bionic, 2018

Users may have the high-end phones and also low-end phones;
We want to be inclusive for users who have low-end phones.

But, it's expensive to design NN of difference sizes

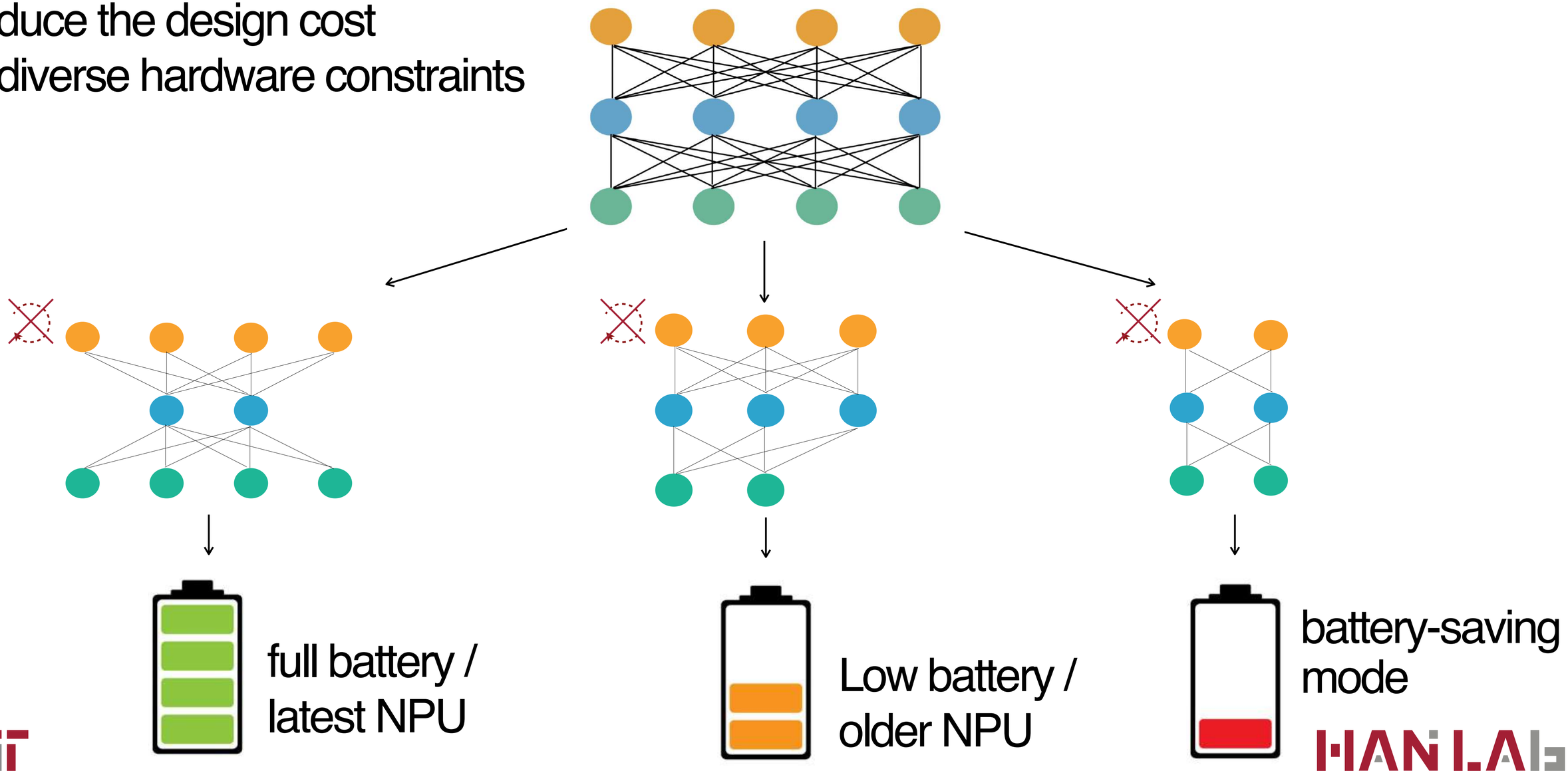
Once-for-All Network

Train once, get many
Reduce the design cost
Fit diverse hardware constraints



Once-for-All Network

Train once, get many
Reduce the design cost
Fit diverse hardware constraints



Once-for-All Network

Weight-sharing,
Decouple Training and Search

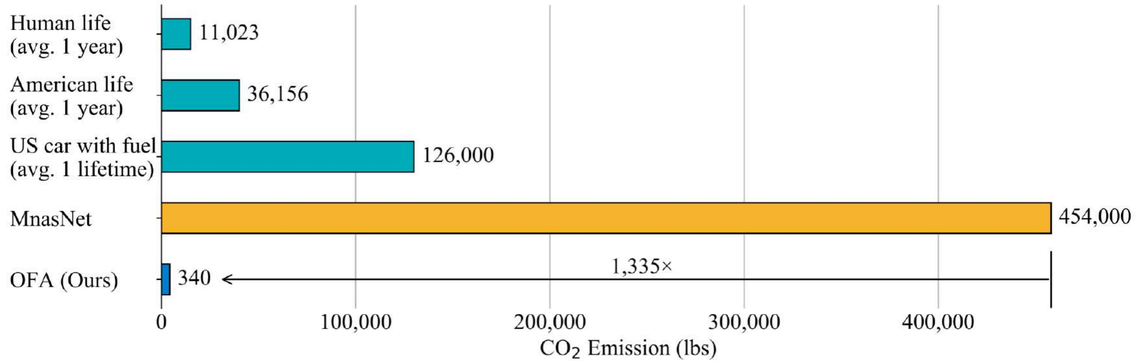
Conventional NAS

```
For devices:
    For search episodes: // meta controller
        For training iterations:
            forward-backward();        Expensive
        If good_model: break;
```

=>

Once-for-All:

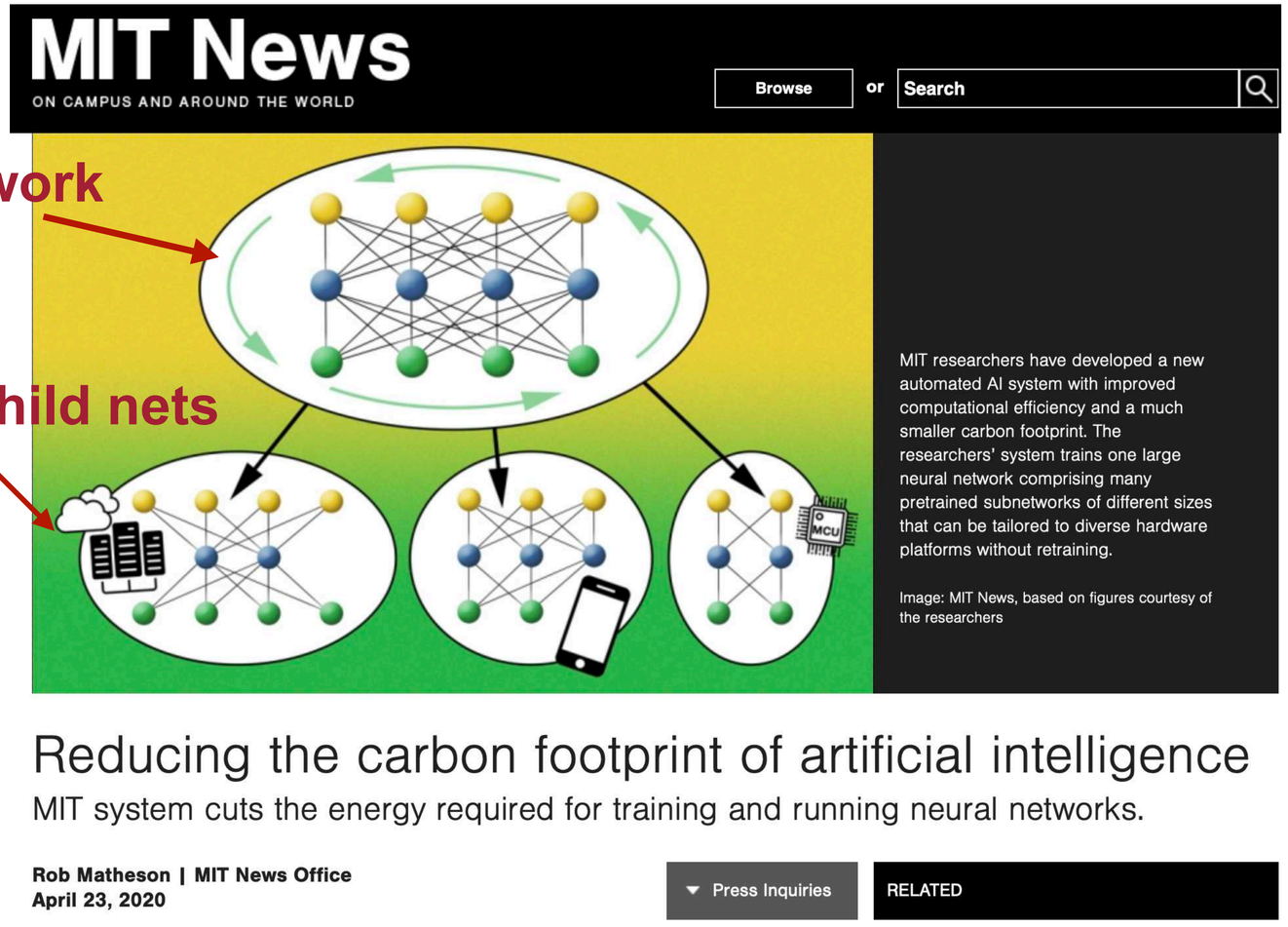
```
For OFA training iterations:                    Expensive
    forward-backward();                         Training
                          Decouple
For devices:                                    Search
    For search episodes:
        sample from OFA;                        Light-Weight
        If good_model: break;
```



Once for All Network: Train 10^{19} networks at the same time

Once-for-All Network

Get many (10^{19}) child nets for free



Human brain activates sparsely

OFA network containts many child networks that are sparsely activated

Child networks share the weights with the Once-for-All network, trained joinly

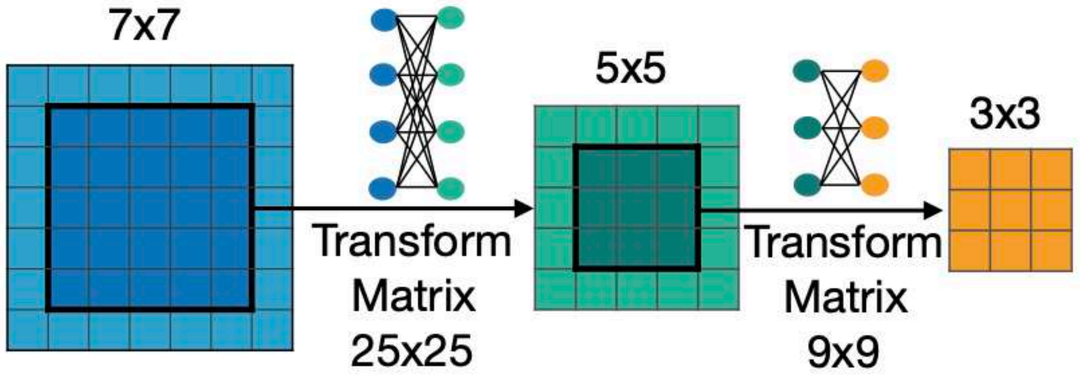
Once-for-All Network

Make the OFA network elastic:

How to prevent large & small sub networks from interfering with each other?

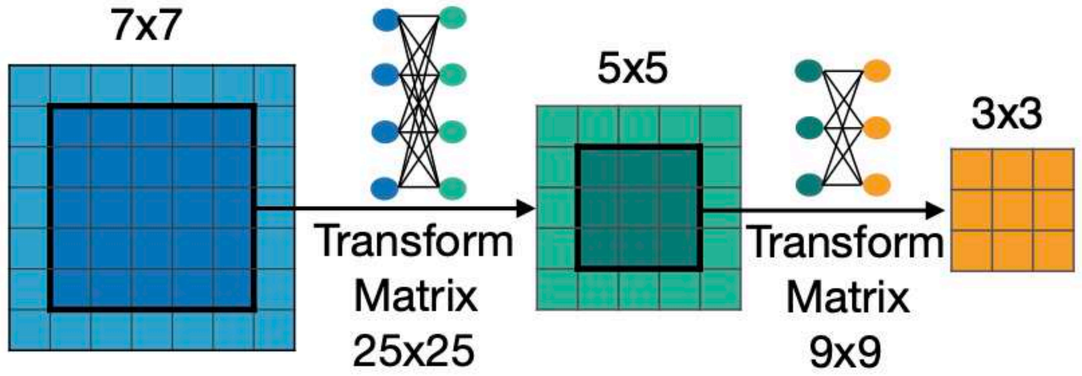
Once-for-All Network

progressive shrinking: kernel size

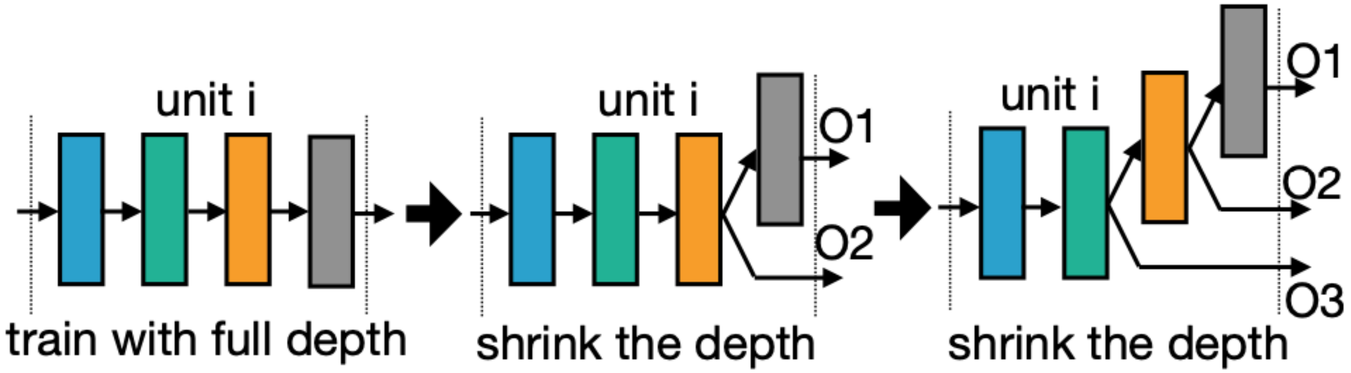


Once-for-All Network

progressive shrinking:
kernel size

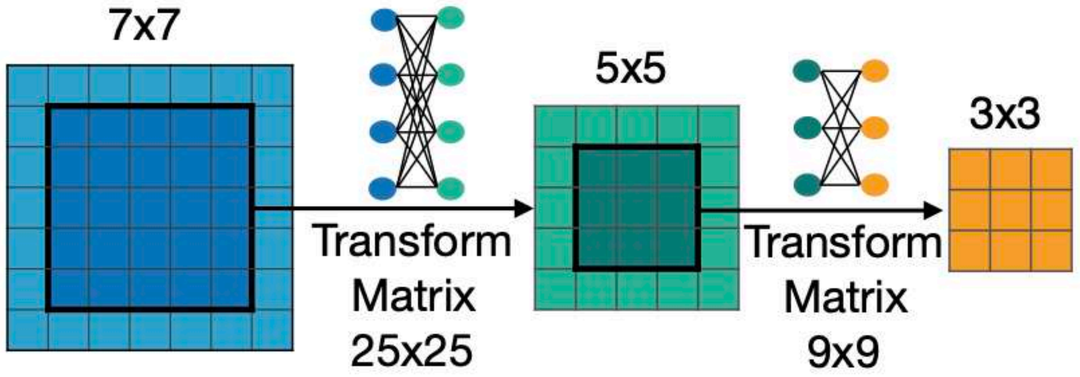


progressive shrinking:
layers

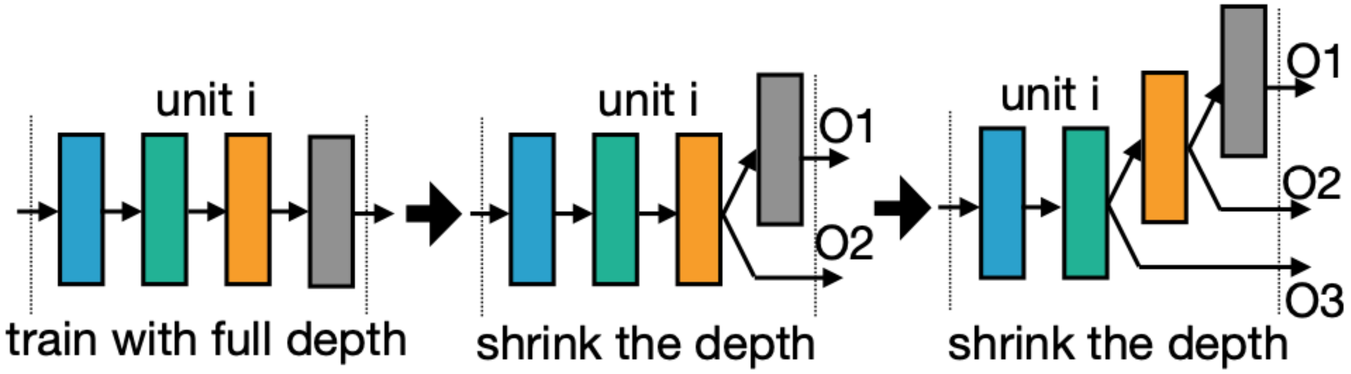


Once-for-All Network

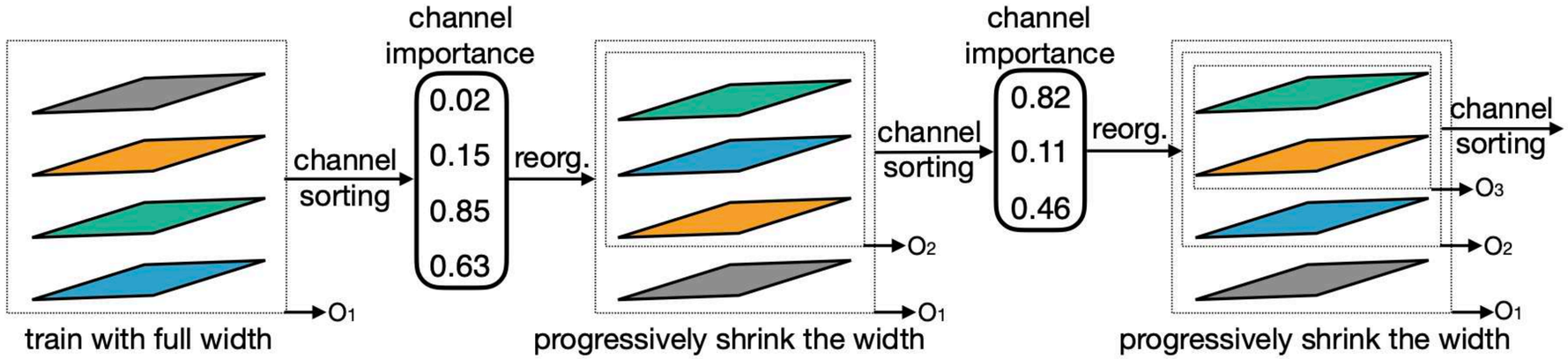
progressive shrinking:
kernel size



progressive shrinking:
layers

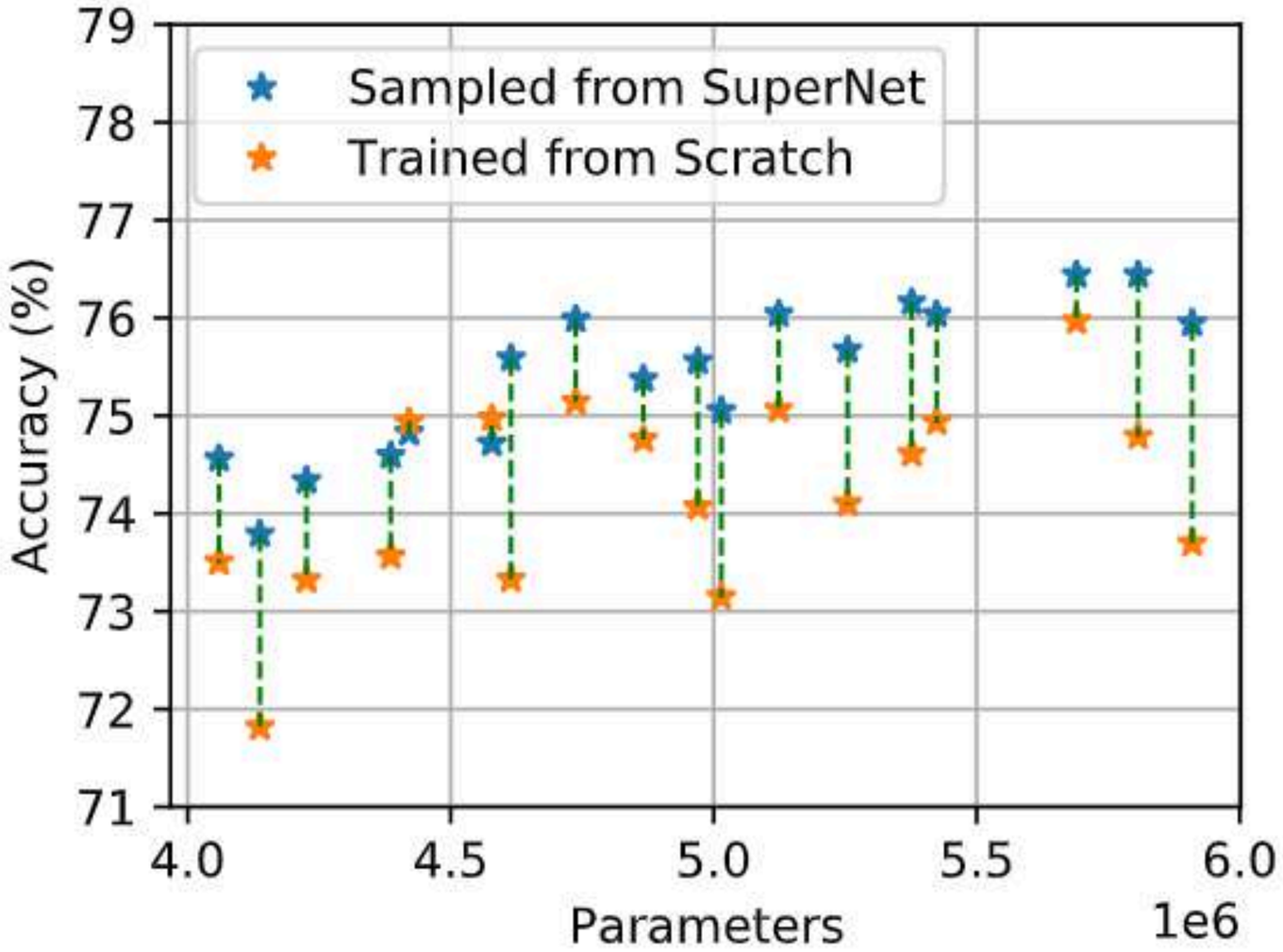


progressive shrinking:
channels



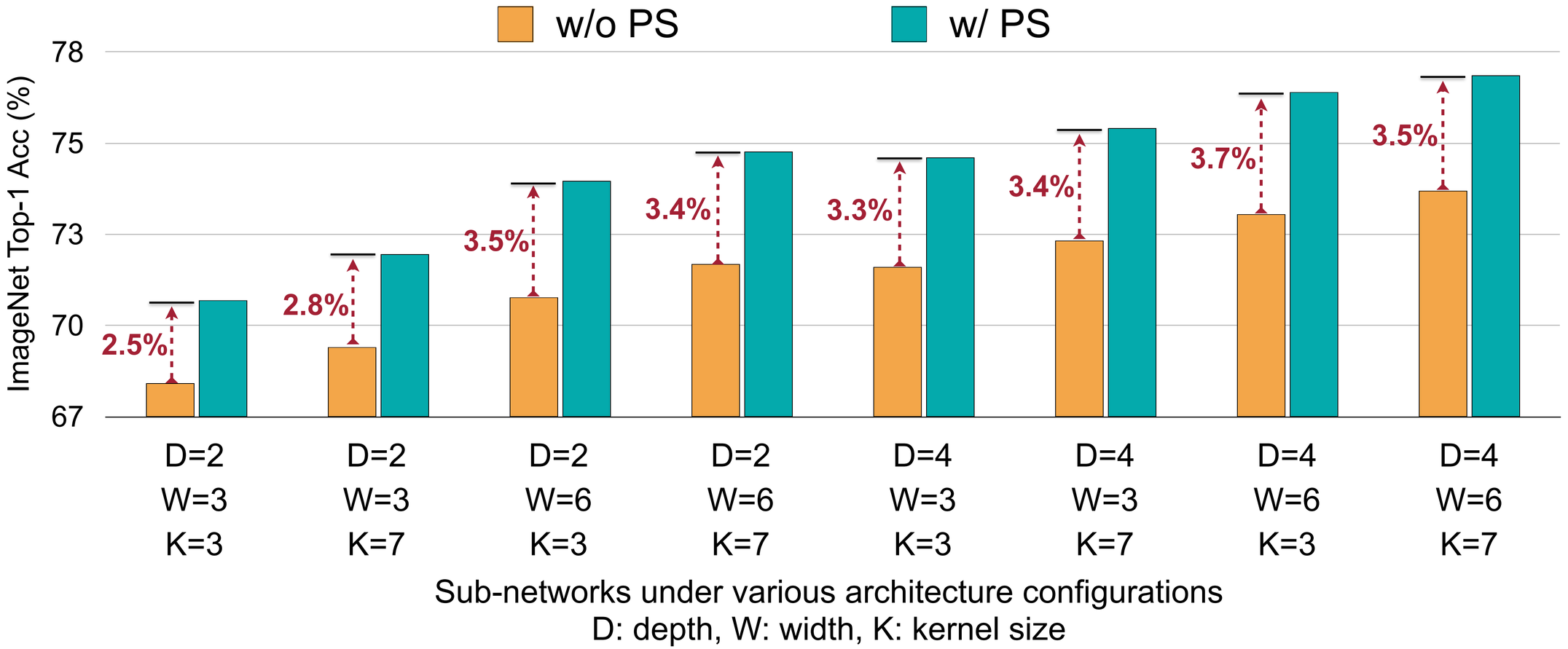
Once-for-All Network

Subnets sampled from OFA SuperNet outperforms training from scratch!



upto +2.8% improvement of top-1 accuracy

Performances of Sub-networks on ImageNet



- Progressive shrinking consistently improves accuracy of sub-networks on ImageNet.

Once-for-All Network

Evolutionary Architecture Search

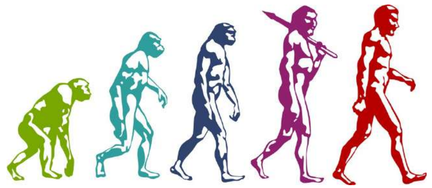
OFA Network

Acc Dataset
[Architecture, Accuracy]

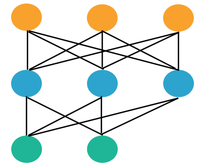
Accuracy Predictor

Latency Predictor

Evolutionary
Architecture Search



Specialized
Sub-Network



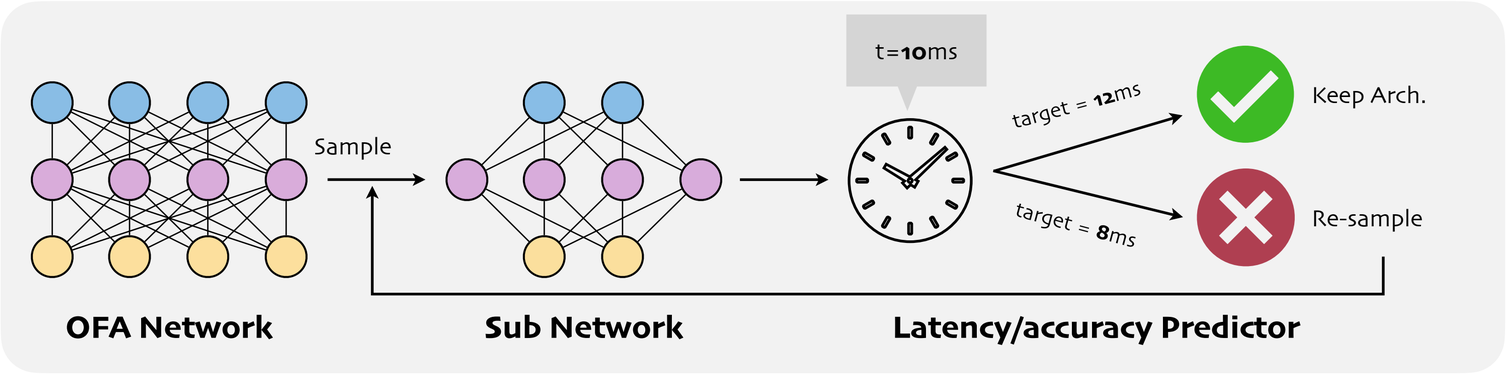
mutation, crossover,
select best fit

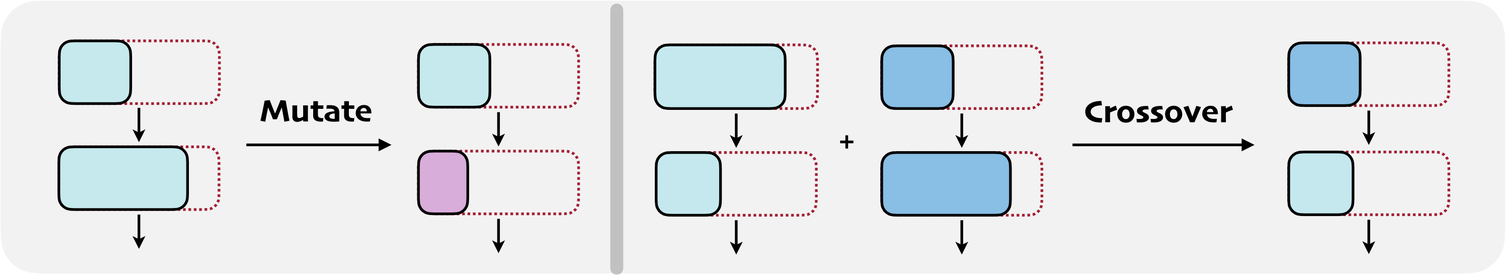
Latency measurement

Latency Dataset
[Architecture, Latency]

Once-for-All Network

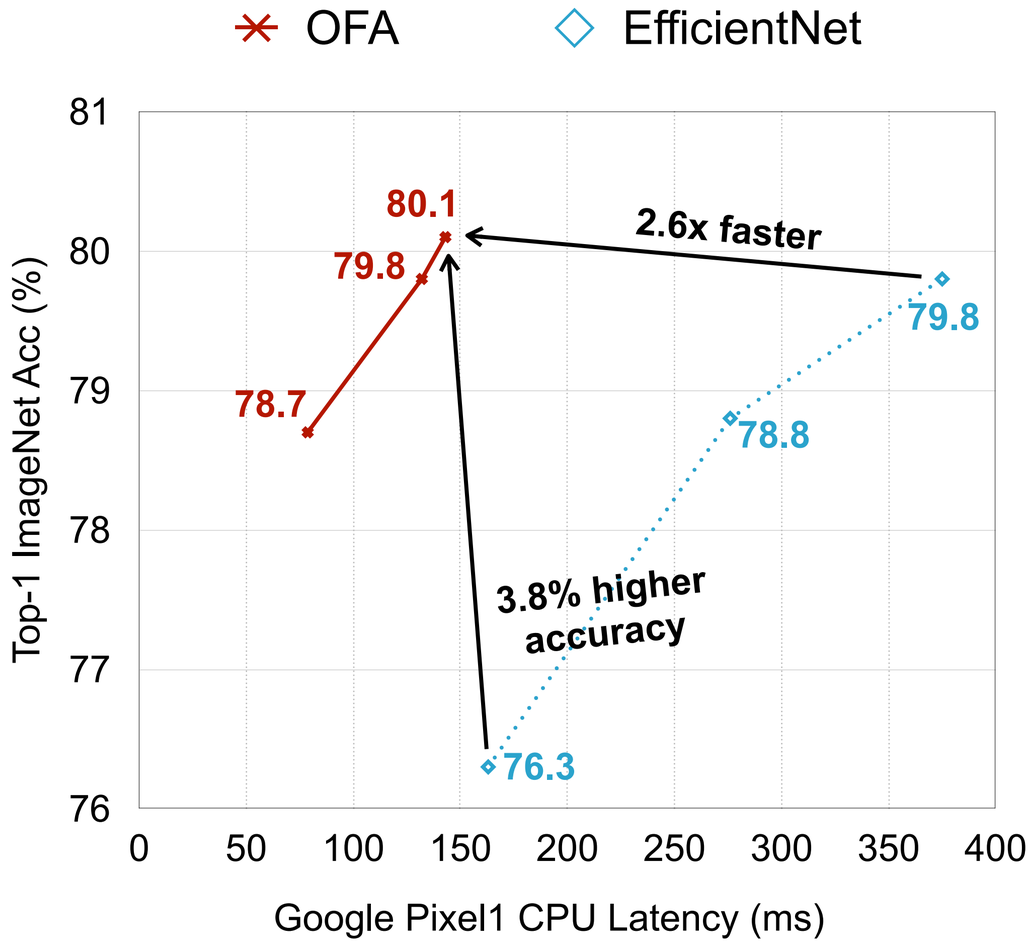
Evolutionary Architecture Search





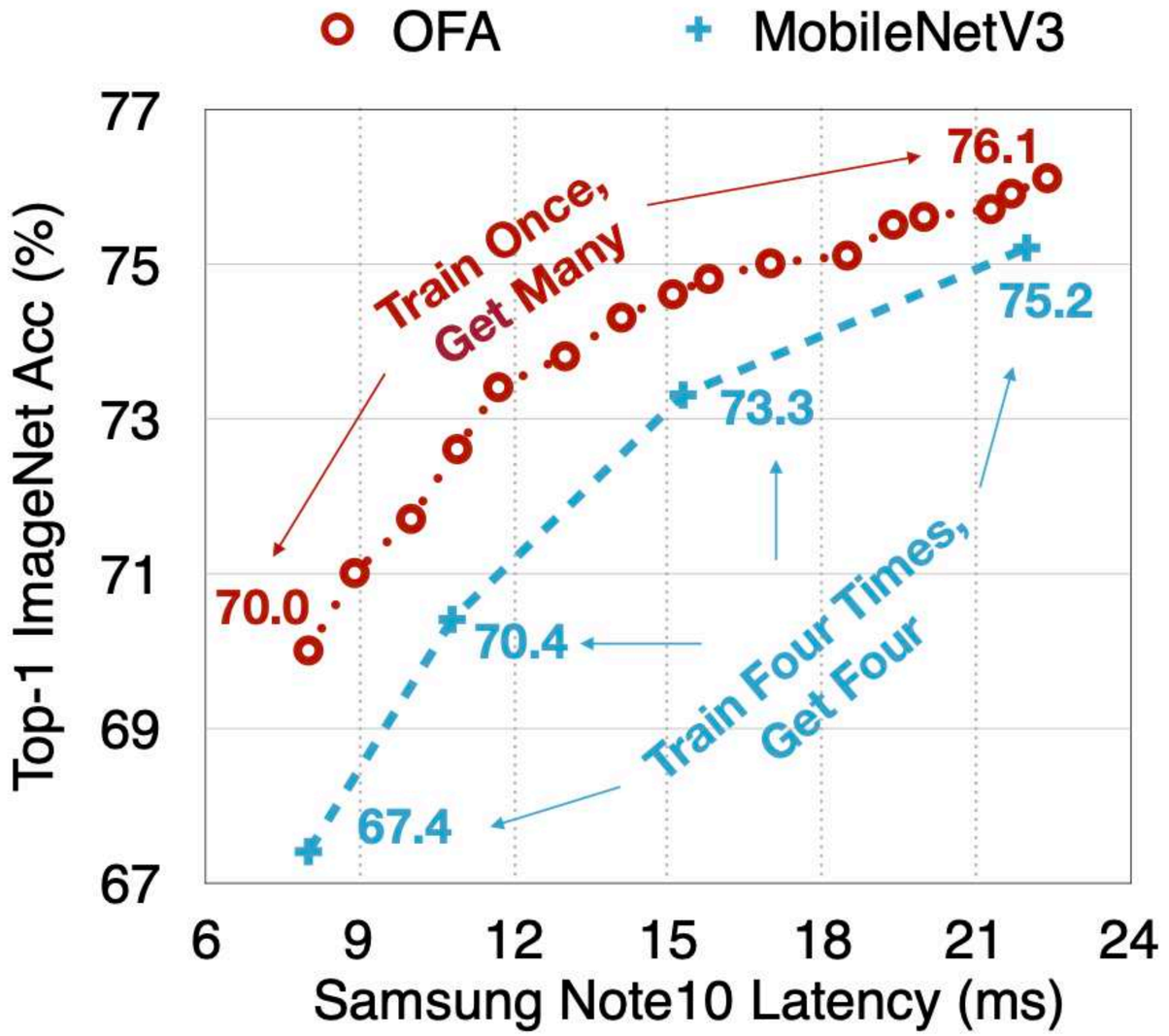
Once-for-All Network

Train only once, handle diverse hardware constraints



Once-for-All Network

Train only once, generate the entire Pareto curve



Once-for-All, ICLR'20

Automatically Synthesize Neural Nets

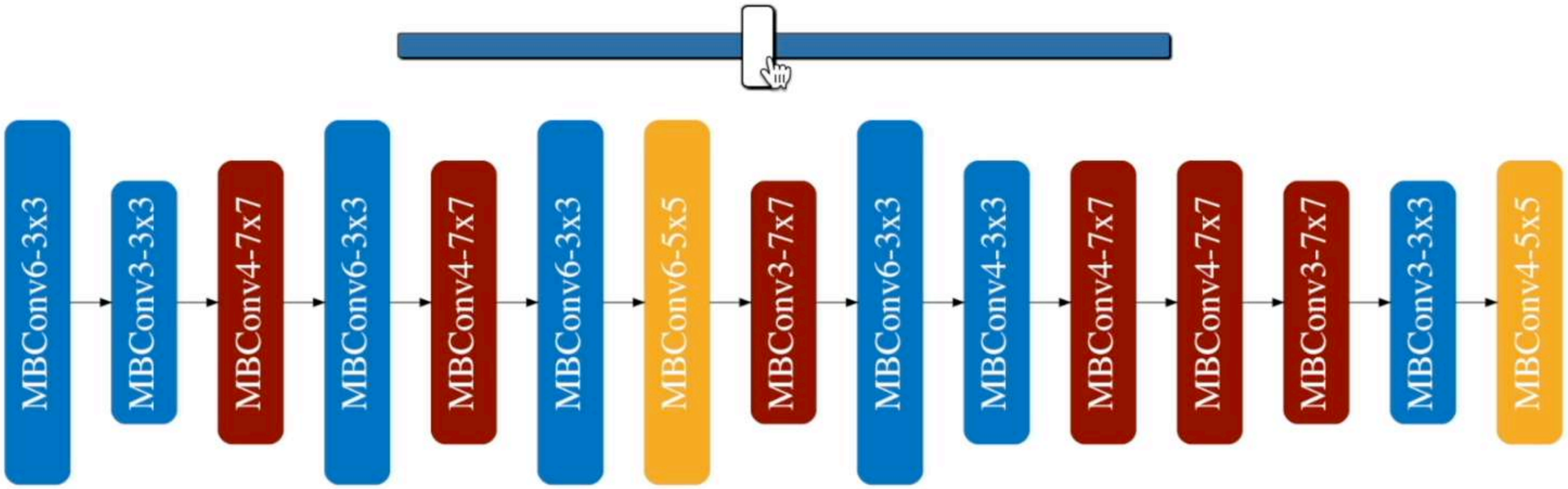
given different latency/accuracy constraints

Drag the bar to target different latency.

← Slide left for faster and less accurate models

→ Slide right for slower but more accurate models

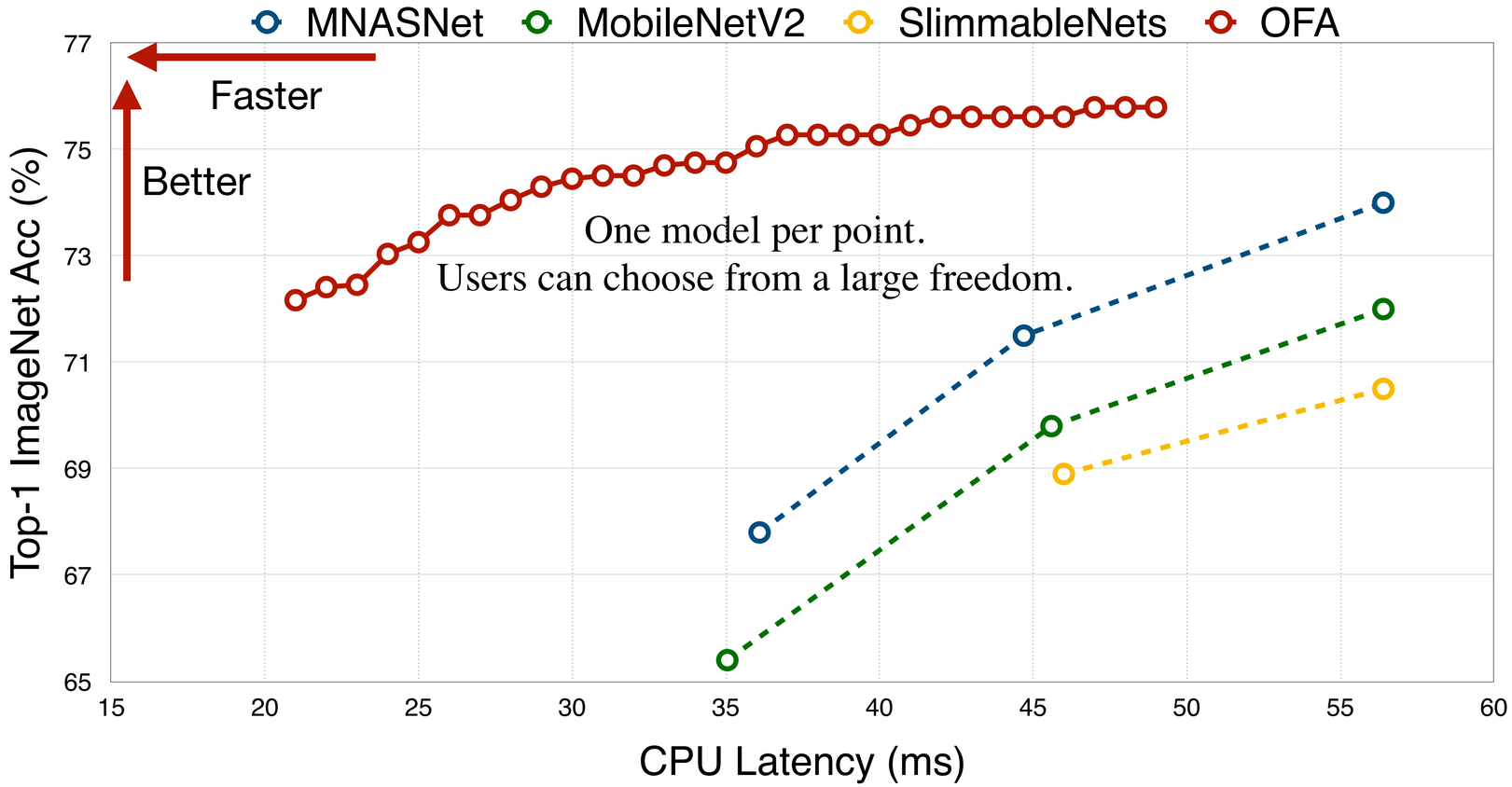
Specialize for 35(ms) on Note10 Device, top1 78.47(%)



https://hanlab.mit.edu/projects/ofa/demo/

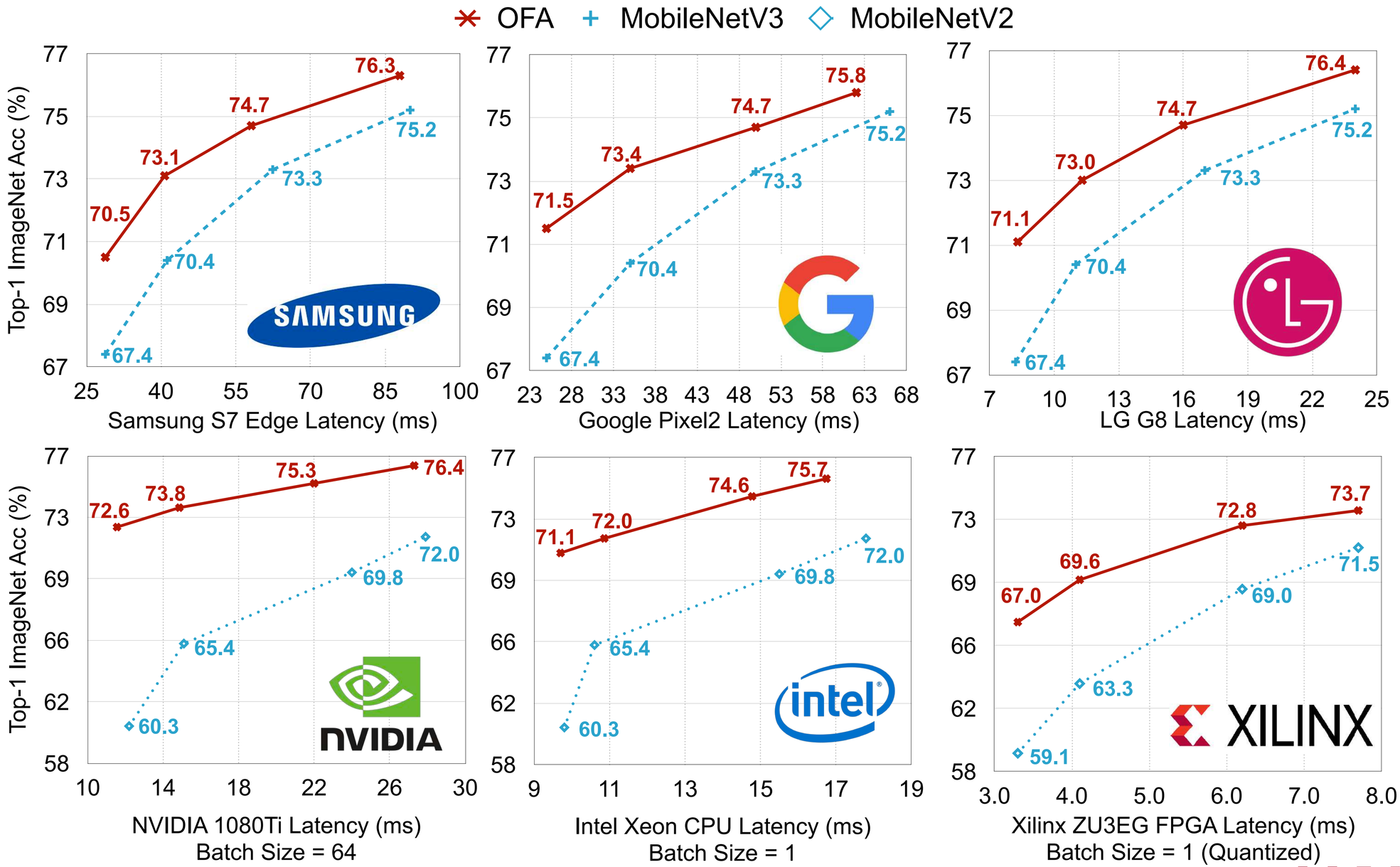
Automatically Synthesize Neural Nets

given different latency/accuracy constraints



Platform: Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz (28 cores), HT ON, turbo ON,
Total Memory: 192GB (12 slots / 16 GB / 2933 MHz)
Framework: PyTorch v.1.5.0 with Intel®MKL-DNN 0.14 enabled, data type FLOAT 32
System: BIOS SE5C620.86B.02.01.0008.031920191559 / Ubuntu 18.04.4 LTS, Kernel 4.15.0-88-generic

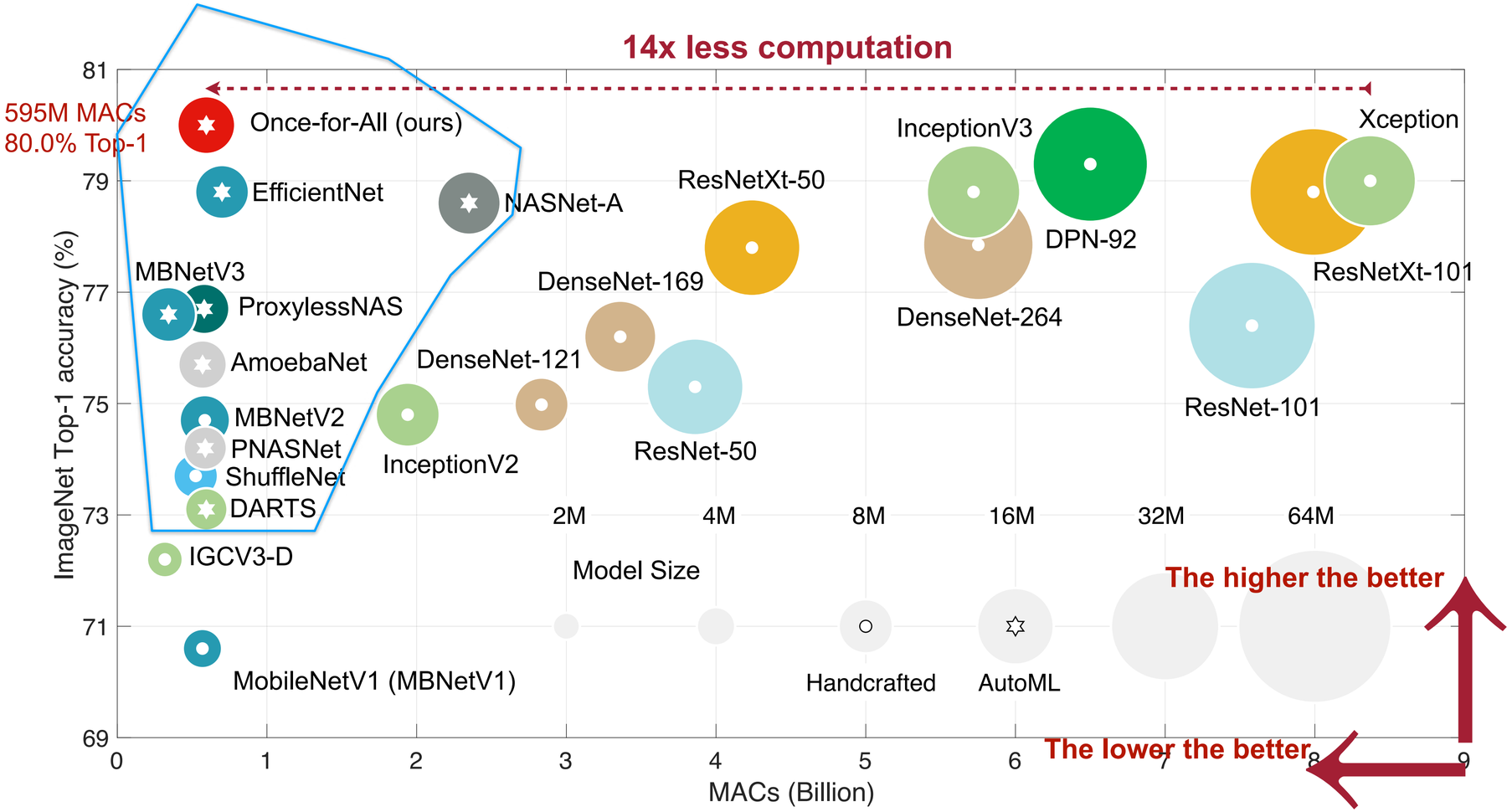
Once-for-All Network



Once-for-All Network

Consistently outperforms human baselines, world-record on MLPerf

Turn-key solution for co-design



- OFA sets a world-record in the open division of [MLPerf Inference Benchmark]: 1.078M images per second on eight A100 GPUs

Award Winning Technology

CPU detection
FPGA detection

5th Low-Power Computer Vision Challenge

CPU classification CPU detection

4th Low-Power Computer Vision Challenge

DSP Recognition

3th Low-Power Computer Vision Challenge

Visual Wake Words
on TF-lite

Visual Wake Words Challenge @CVPR 2019

3D Semantic
Segmentation

AI Driving Olympics @ICRA 2021

NLP track
Language Model

MicroNet Challenge @NeurIPS 2019

Industry Adoption

Once-for-All (OFA) Network adopbed by Alibaba received a world-record in the open division of MLPerf Inference Benchmark, achieving 1.078M images per econd on eight A100 GPUs

Once-for-All (OFA) Network adopbed by Maxim Integrated provides 6% accuracy increase in image recognition and 2% accuracy increase in speech command recognition, with >100x energy efficiency compared to Cortex-M4.

Proxyless Neural Architecture Search, an efficient neural architecture search algorithm with light-weight model for mobile AI is integrated by AWS AutoGluon and Facebook PyTorch.

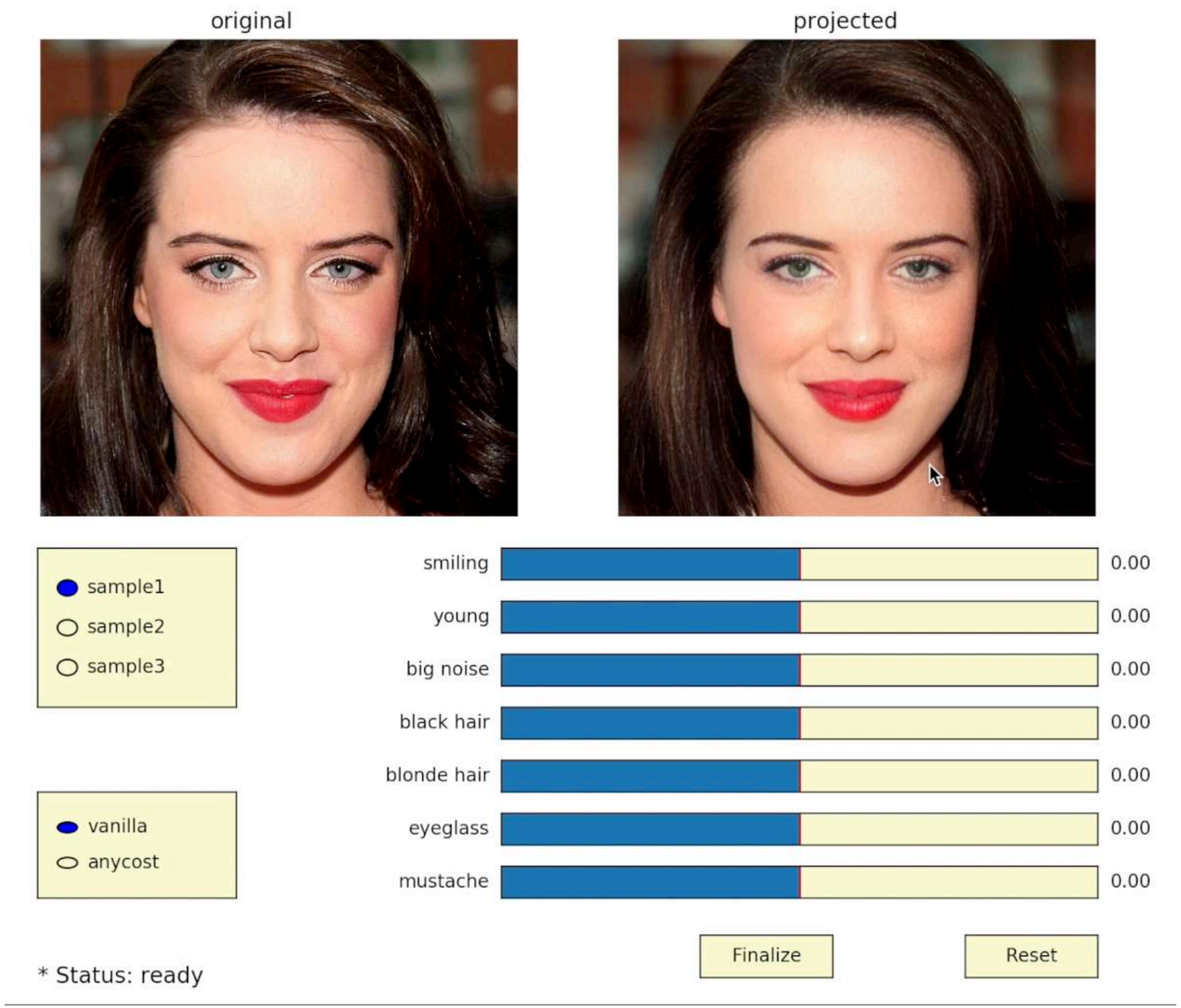
HAQ: Hardware-Aware Automated Quantization with Mixed Precision is integrated by Intel OpenVINO Toolkit. Efficiently search over the bitwidth space for mixed-precision machine learning inference (2, 4, 8 bits)

TSM: Temporal Shift Module for Efficient Video Understanding is integrated by NVIDIA for video classification.

Anycost GAN

AnyCost GAN, CVPR'21

Demo:



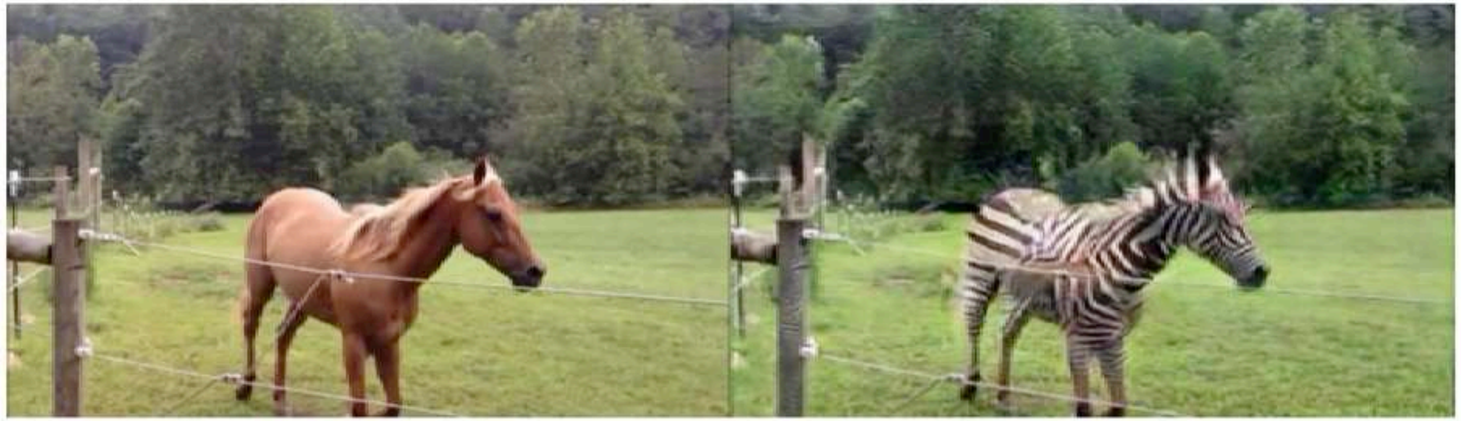
Anycost GAN

Smaller, faster child networks are nested in larger ones

GAN Compression

Accelerating Horse2zebra by GAN Compression

Demo:



Original CycleGAN; FLOPs: 56.8G; FPS: 12.1; FID: 61.5

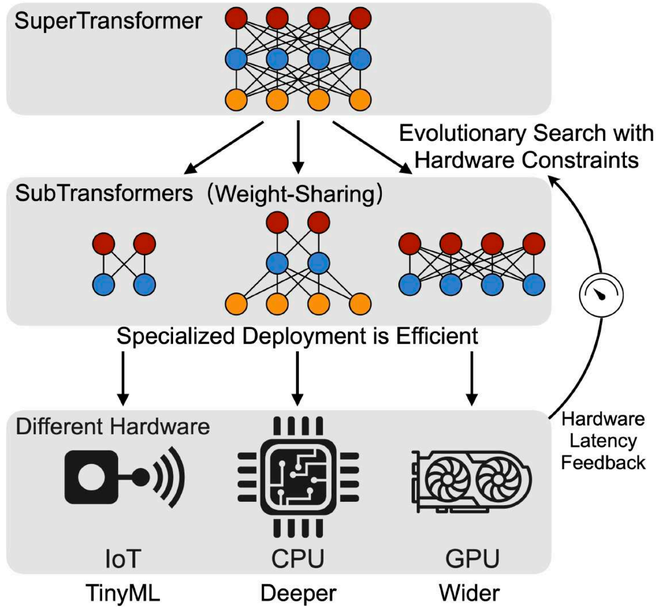
GAN Compression; FLOPs: 3.50G (16.2x); FPS: 40.0 (3.3x); FID: 53.6

Efficient NLP

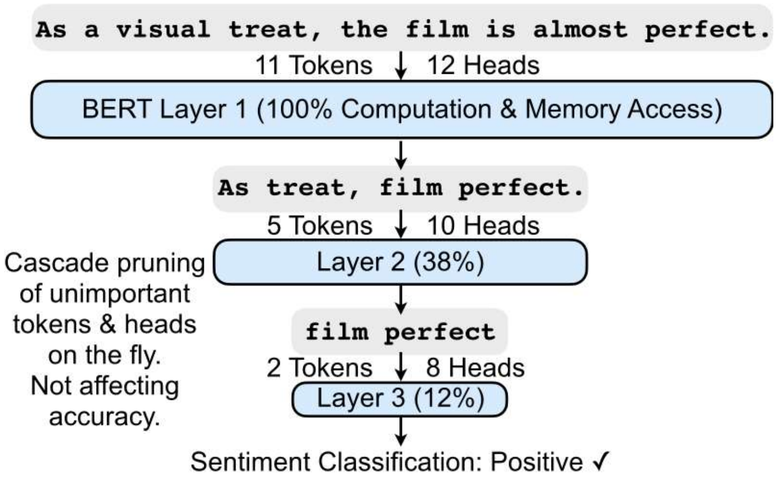
HAT, ACL'20

SpAtten, HPCA'21

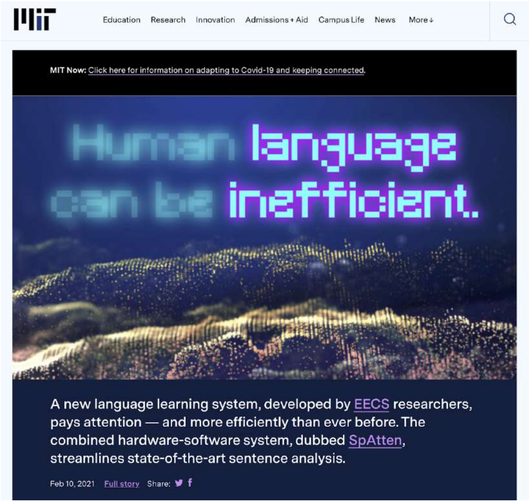
Once-for-all Transformer

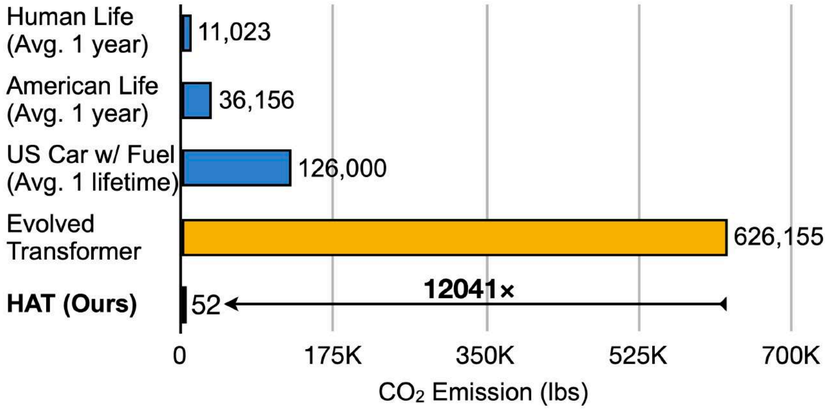


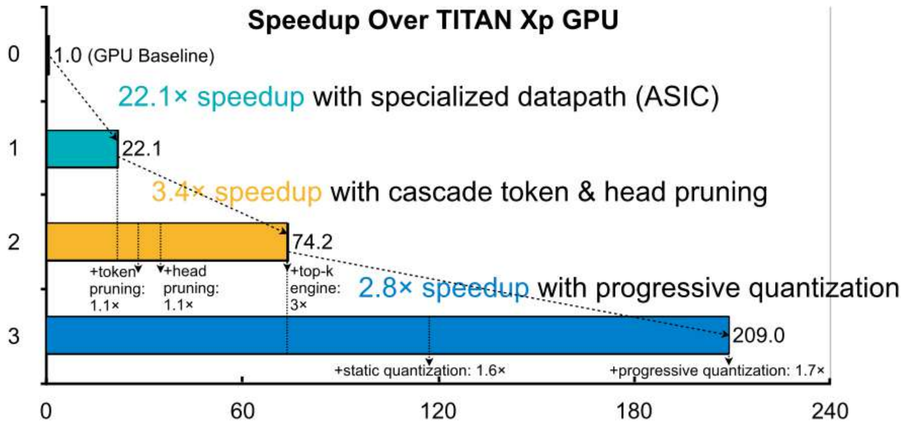
Cascade token pruning



Spotlight on MIT homepage





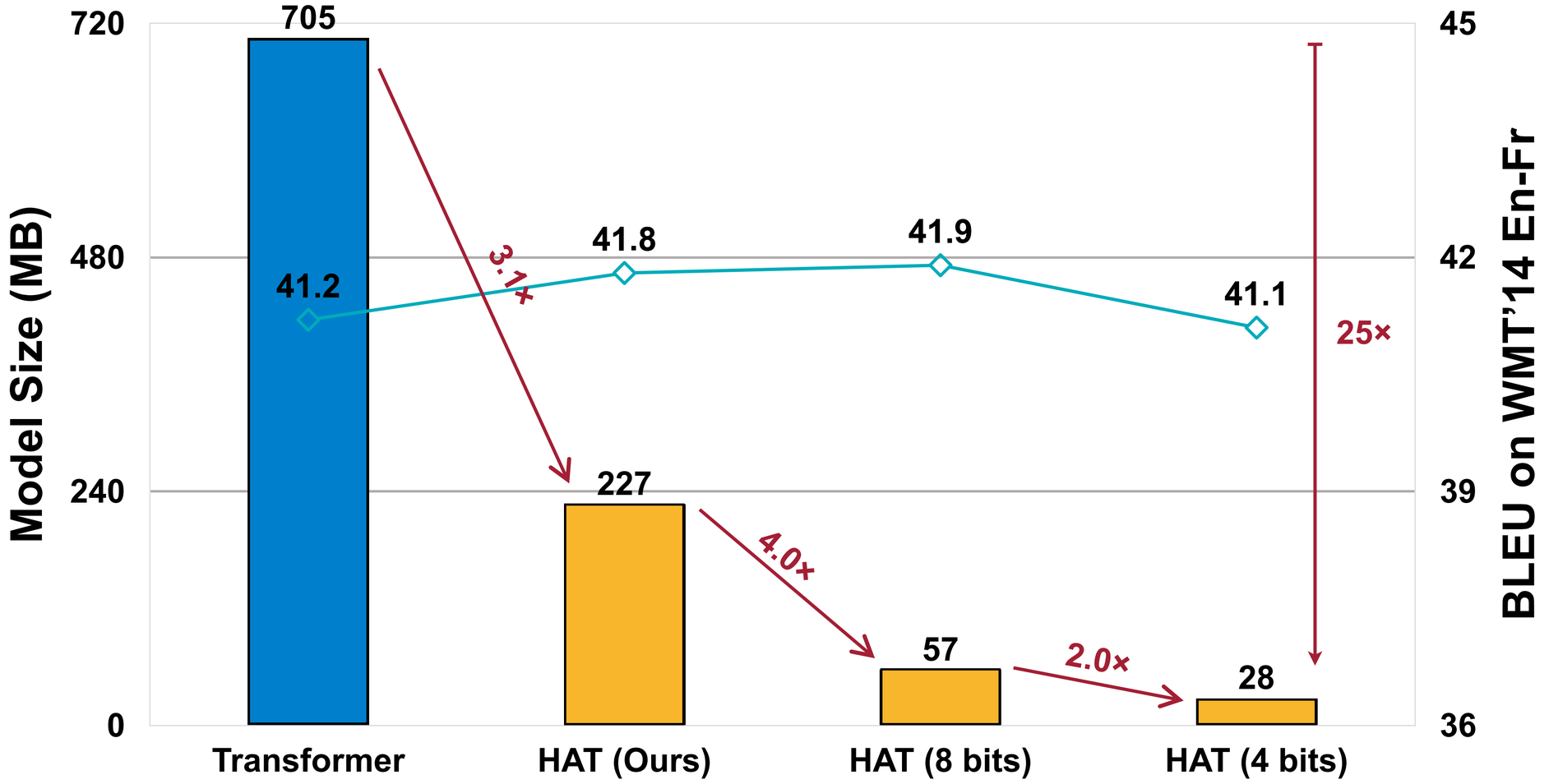


[HPCA'21] Hanrui Wang, Zhekai Zhang, Song Han; "SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning"

Efficient NLP

HAT, ACL'20
SpAtten, HPCA'21

On WMT'14 En-Fr Task

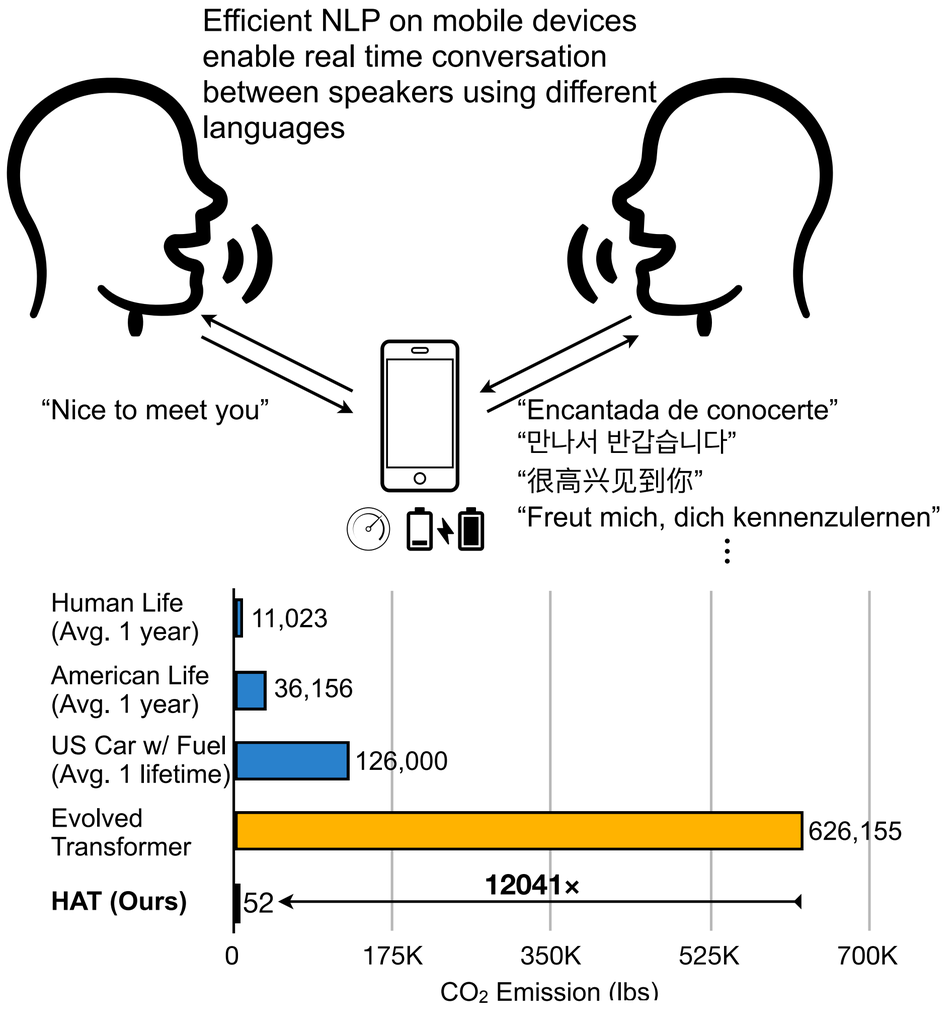


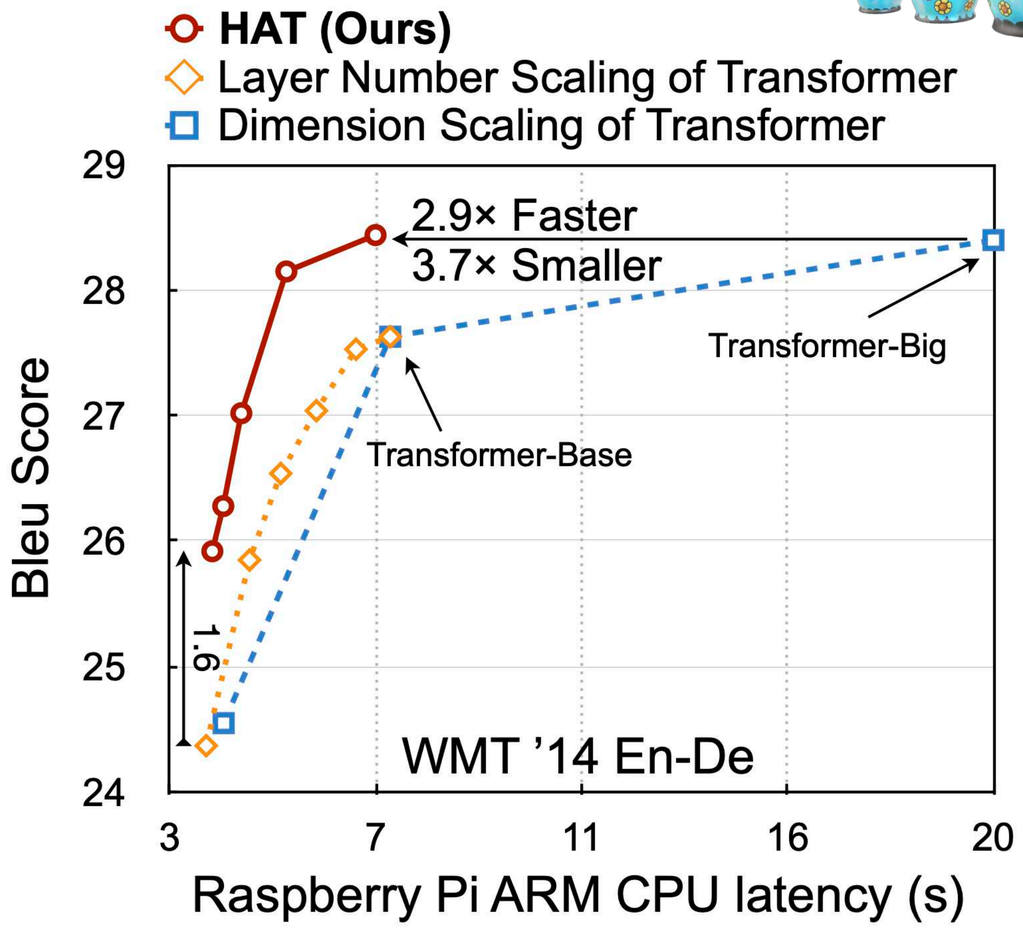
- HAT is orthogonal to general model compression techniques

Efficient NLP

HAT, ACL'20

SpAtten, HPCA'21



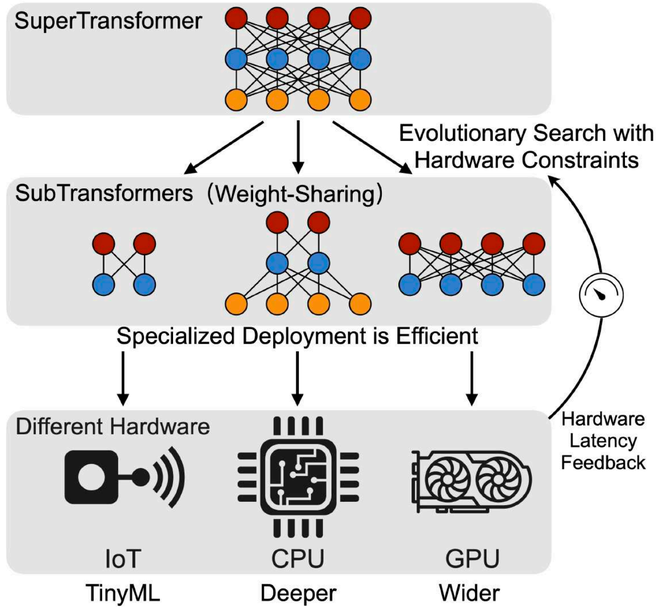


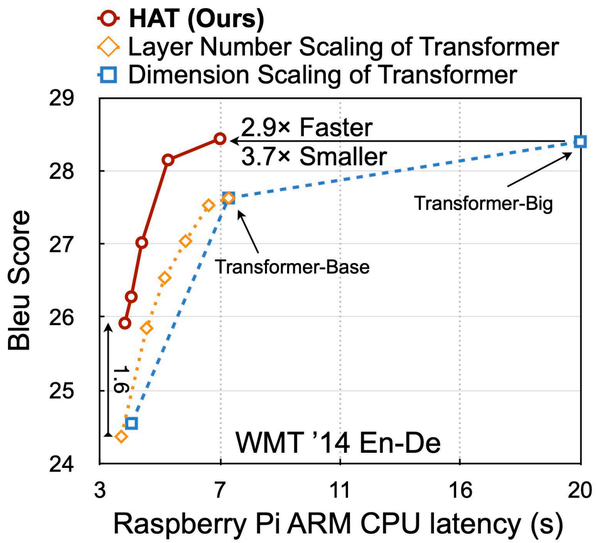
OFA for NLP

HAT, ACL'20

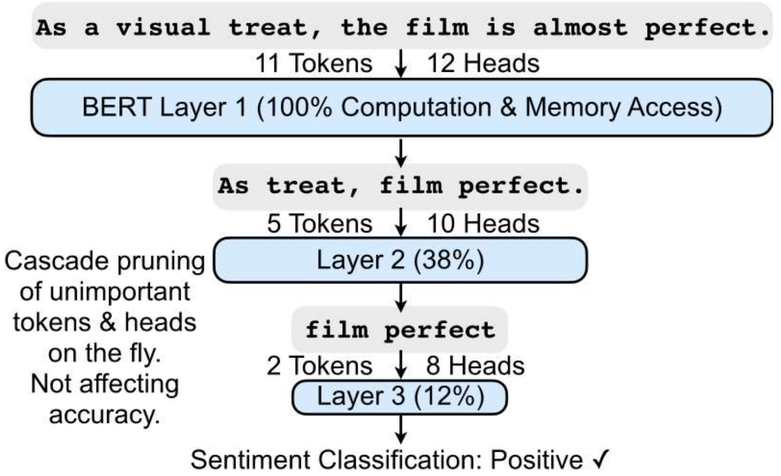
SpAtten, HPCA'21

Once-for-all Transformer

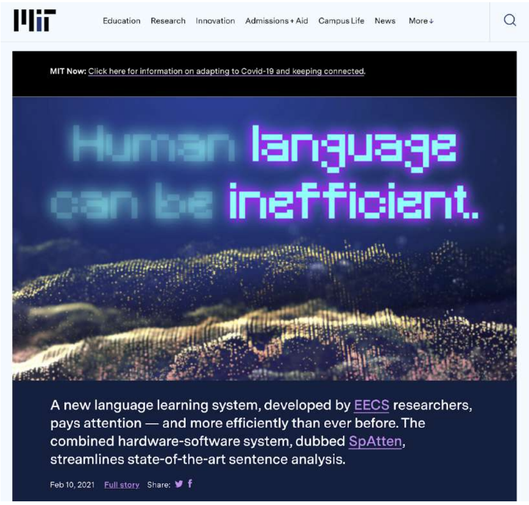


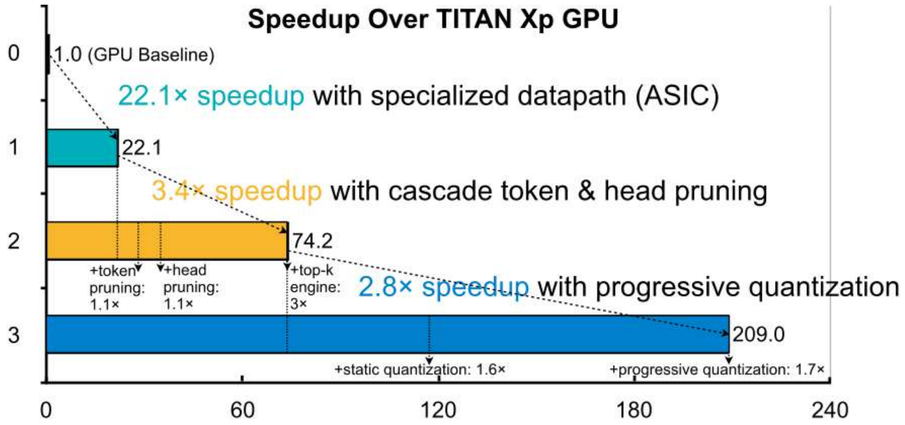


Cascade token pruning

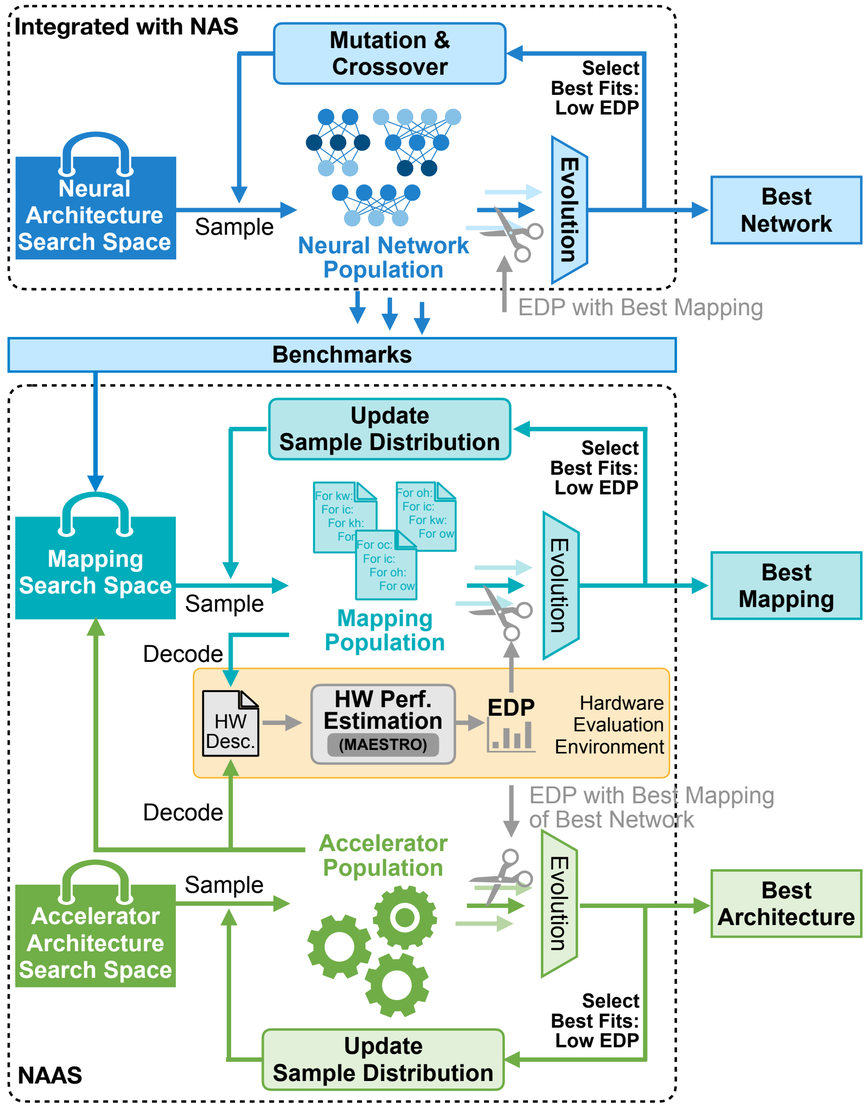


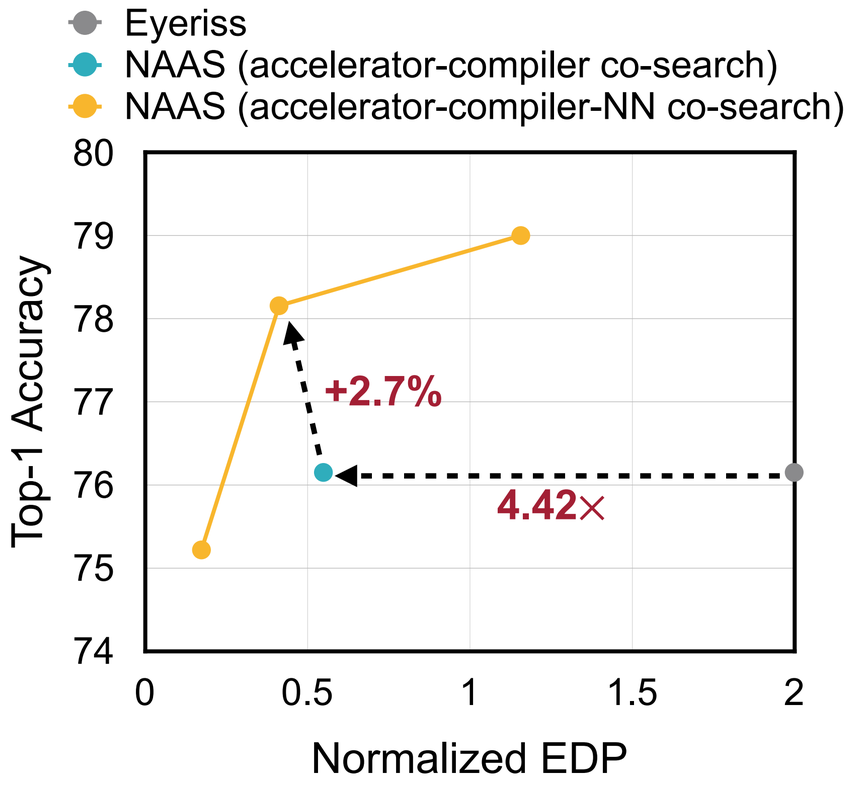
Spotlight on MIT homepage





NAAS: Neural Accelerator Architecture Search





Can we go even smaller?

TinyML and Efficient Deep Learning

- **AutoML and NAS**
  - Once-for-all Network [ICLR'19]

- **TinyML**
  - Inference: MCUNet for IoT Devices [NeurIPS'20, spotlight]
  - Training: Tiny On-Device Transfer Learning (TinyTL) [NeurIPS'20]

- **Data-Efficiency**
  - Differentiable Augmentation for Data-Efficient GAN Training [NeurIPS'20]

MCUNet: Tiny Deep Learning on IoT Devices

Ji Lin[1] Wei-Ming Chen[1,2] Yujun Lin[1] John Cohn[3] Chuang Gan[3] Song Han[1]

[1]MIT [2]National Taiwan University [3]MIT-IBM Watson AI Lab

NeurIPS 2020 (spotlight)

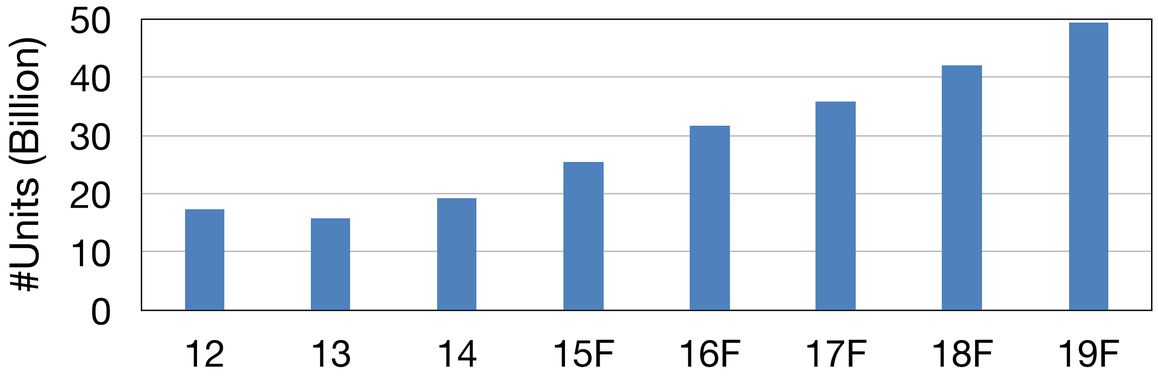
Background: The Era of AIoT on Microcontrollers (MCUs)

- **Low-cost, low-power**

Background: The Era of AIoT on Microcontrollers (MCUs)

- **Low-cost, low-power**

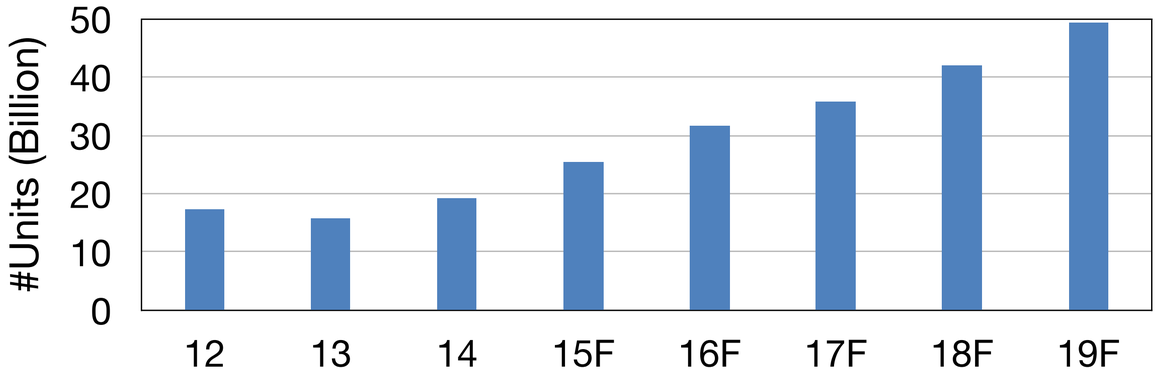
- **Rapid growth**



Background: The Era of AIoT on Microcontrollers (MCUs)

- **Low-cost, low-power MCU**

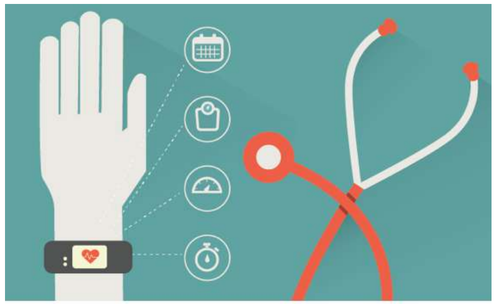
- **Rapid growth**



- **Wide applications**

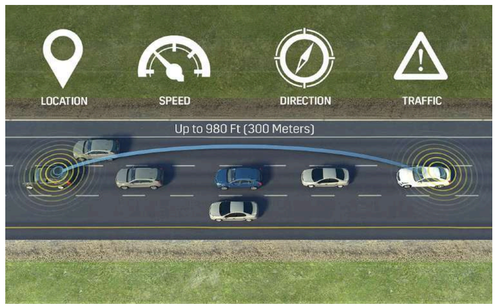
Smart Retail

Personalized Healthcare



Smart Manufacturing

Autonomous Driving



...

Deep Learning Going “Tiny”

Cloud AI
=> ResNet

Data centers, connection required, privacy issue

Mobile AI
=> MobileNet

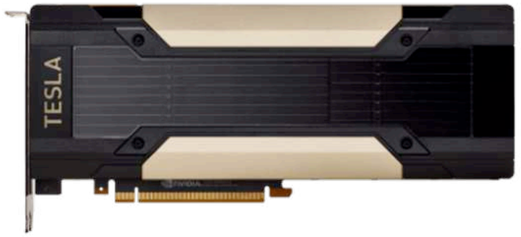
Smartphones process locally

Tiny AI
=> MCUNet

IoT devices, cheap, small, low-power, rapid growth

- The future belongs to Tiny AI.
- There are billions of IoT devices around the world based on microcontrollers
- Much cheaper, much smaller, almost everywhere in our lives.
- If we can enable powerful AI algorithms on those IoT devices, we can greatly democratize AI and extend the applications of deep learning.

Challenge: Memory Too Small to Hold DNN



Cloud AI | **Mobile AI** | **Tiny AI**

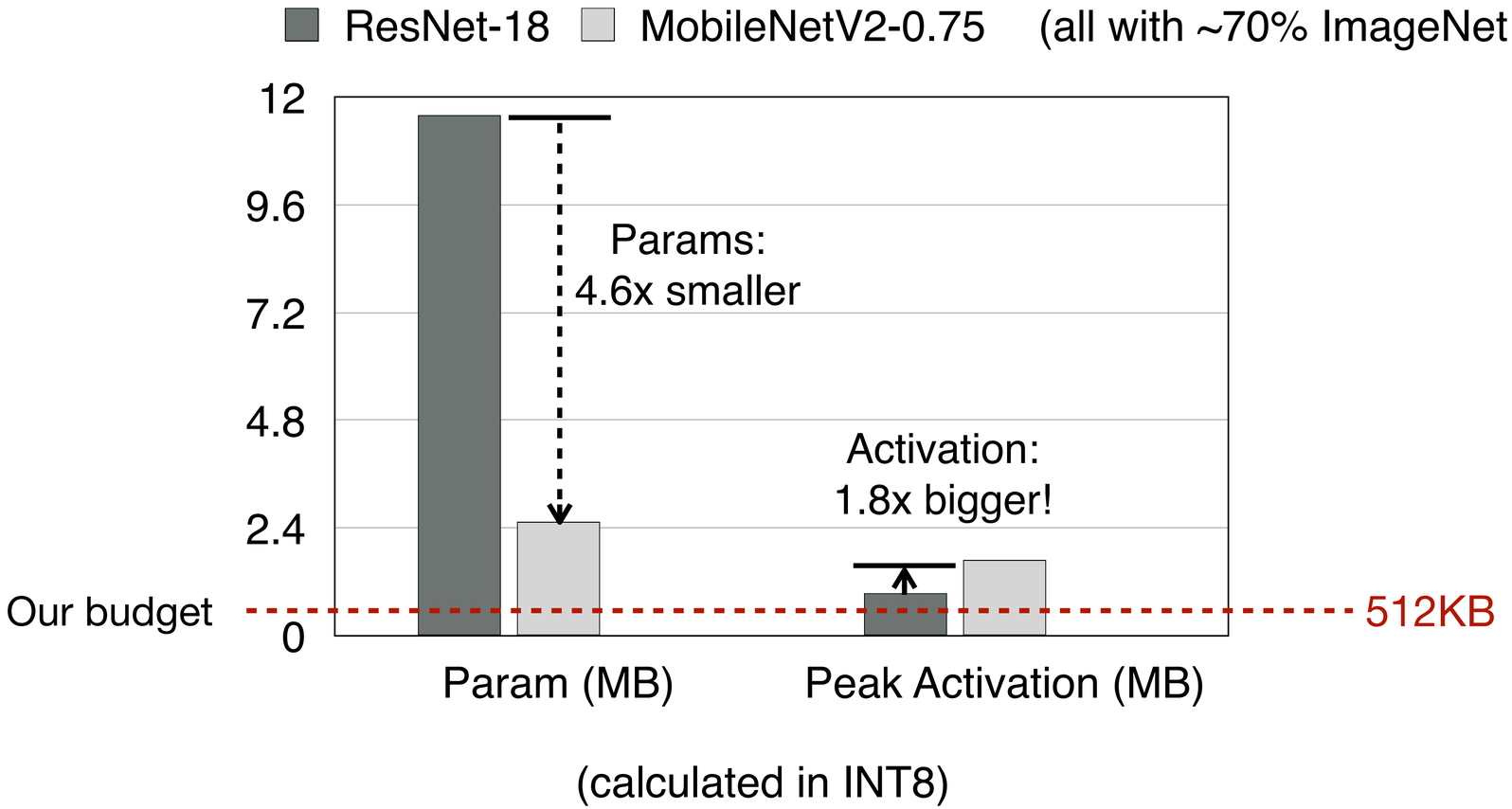
	Cloud AI	Mobile AI	Tiny AI
Memory (Activation)	16GB	4GB	320kB
Storage (Weights)	~TB/PB	256GB	1MB

13,000x smaller

50,000x smaller

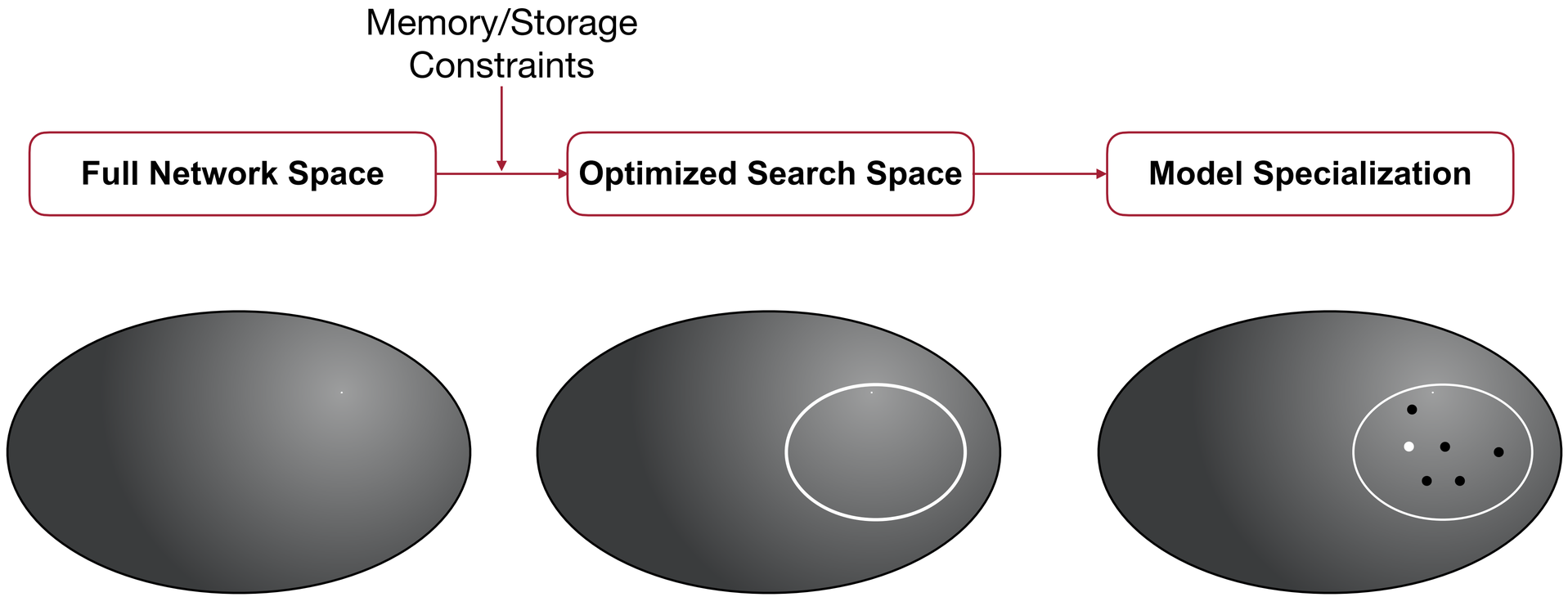
- Tiny model design is fundamentally different.
- No DRAM. No operating system (no virtual memory).
- Can't directly scale. (non-proportional activation vs. params)

Activation is the bottleneck, not parameters

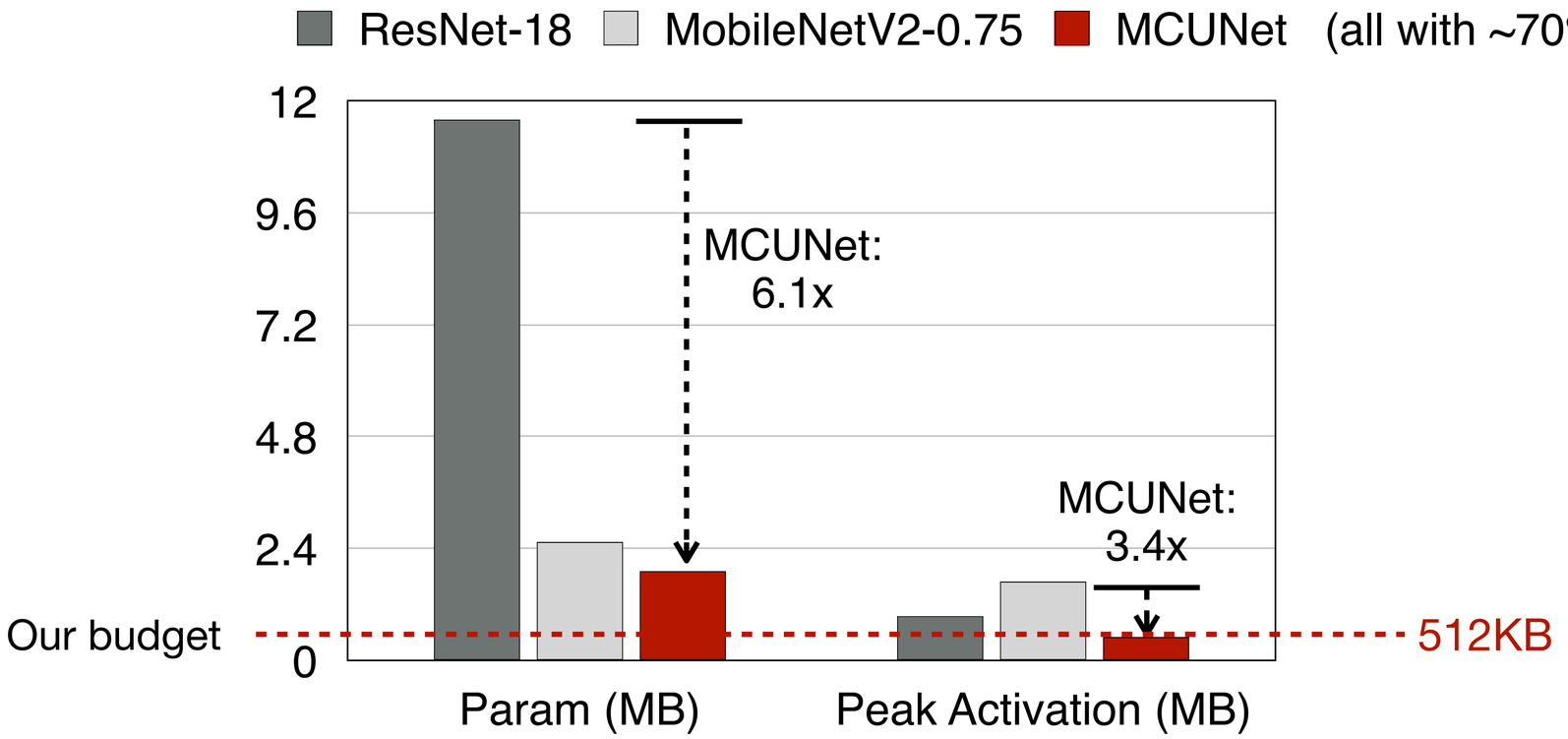


MCUNet: TinyNAS + TinyEngine

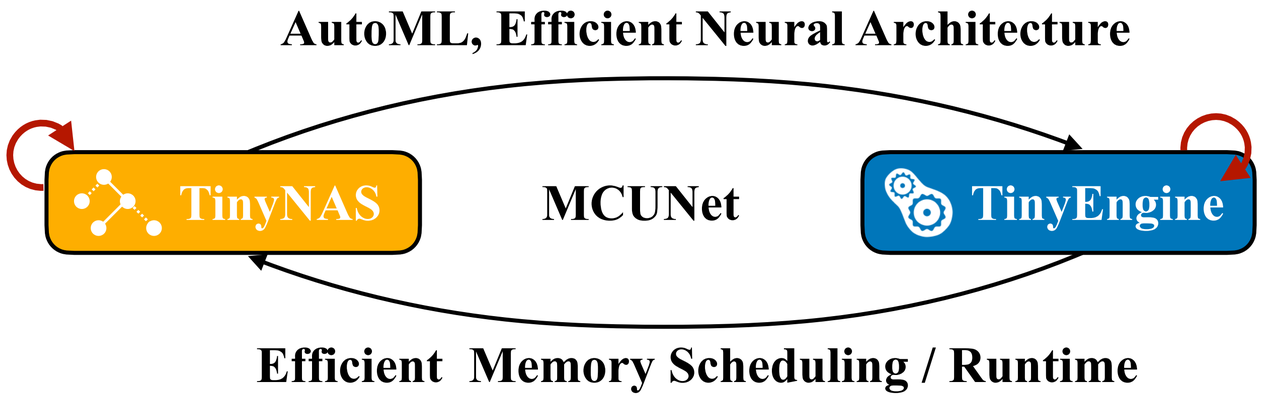
Search space design is crucial for NAS performance
There is no prior expertise on MCU model design



Reduce Both Model Size and Activation Size



MCUNet: TinyNAS+TinyEngine Co-design

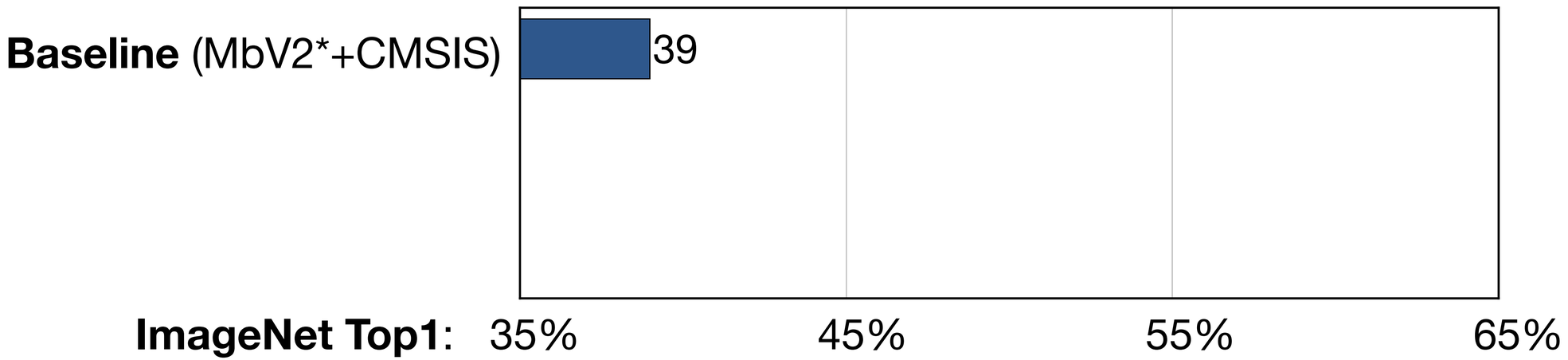


- TinyNAS:
  - Re-design the design space
  - Latency-aware
  - Energy-aware
  - Once-for-all Network: train once, get many

- TinyEngine:
  - Co-design, specialization
  - Run time => Compile time
  - Graph optimizations
  - Memory-aware scheduling
  - Low-precision
  - Assembly-level optimizations

MCUNet: TinyNAS+TinyEngine

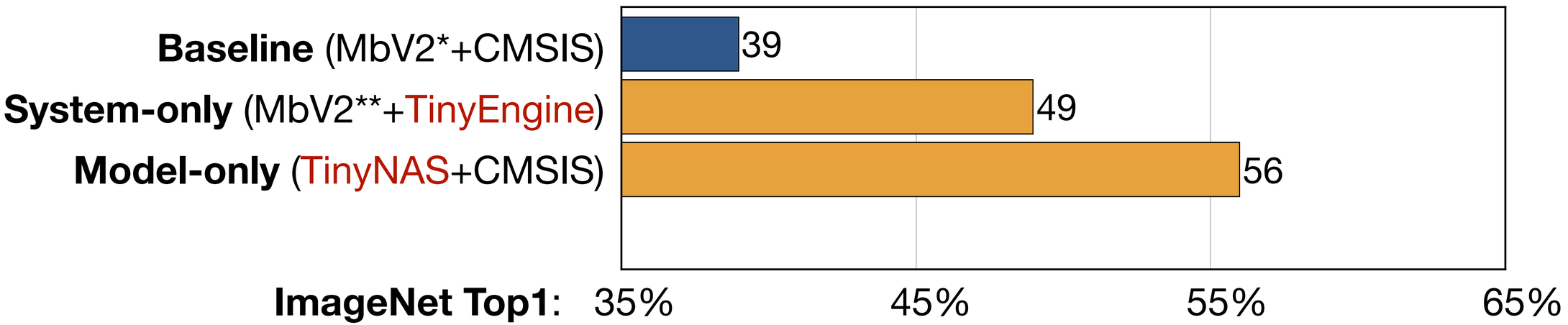
- ImageNet classification on STM32F746 MCU (**320kB SRAM**, **1MB Flash**)



* scaled down version: width multiplier 0.3, input resolution 80

MCUNet: TinyNAS+TinyEngine

- ImageNet classification on STM32F746 MCU (**320kB SRAM**, **1MB Flash**)

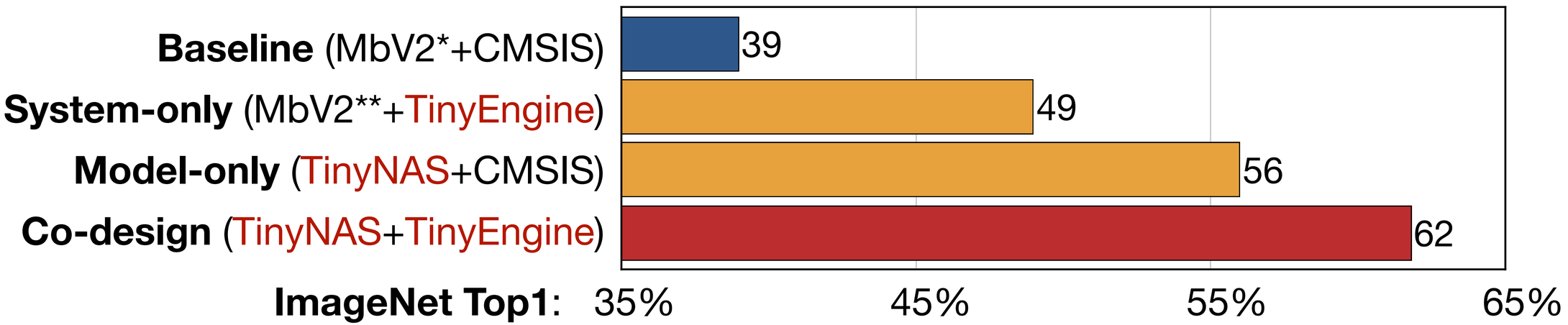


* scaled down version: width multiplier 0.3, input resolution 80

** scaled down version: width multiplier 0.35, input resolution 144

MCUNet: TinyNAS+TinyEngine

- ImageNet classification on STM32F746 MCU (**320kB SRAM**, **1MB Flash**)



* scaled down version: width multiplier 0.3, input resolution 80

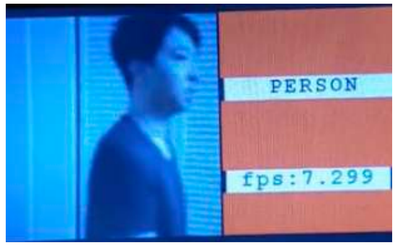
** scaled down version: width multiplier 0.35, input resolution 144

Bring AI to IoT Devices

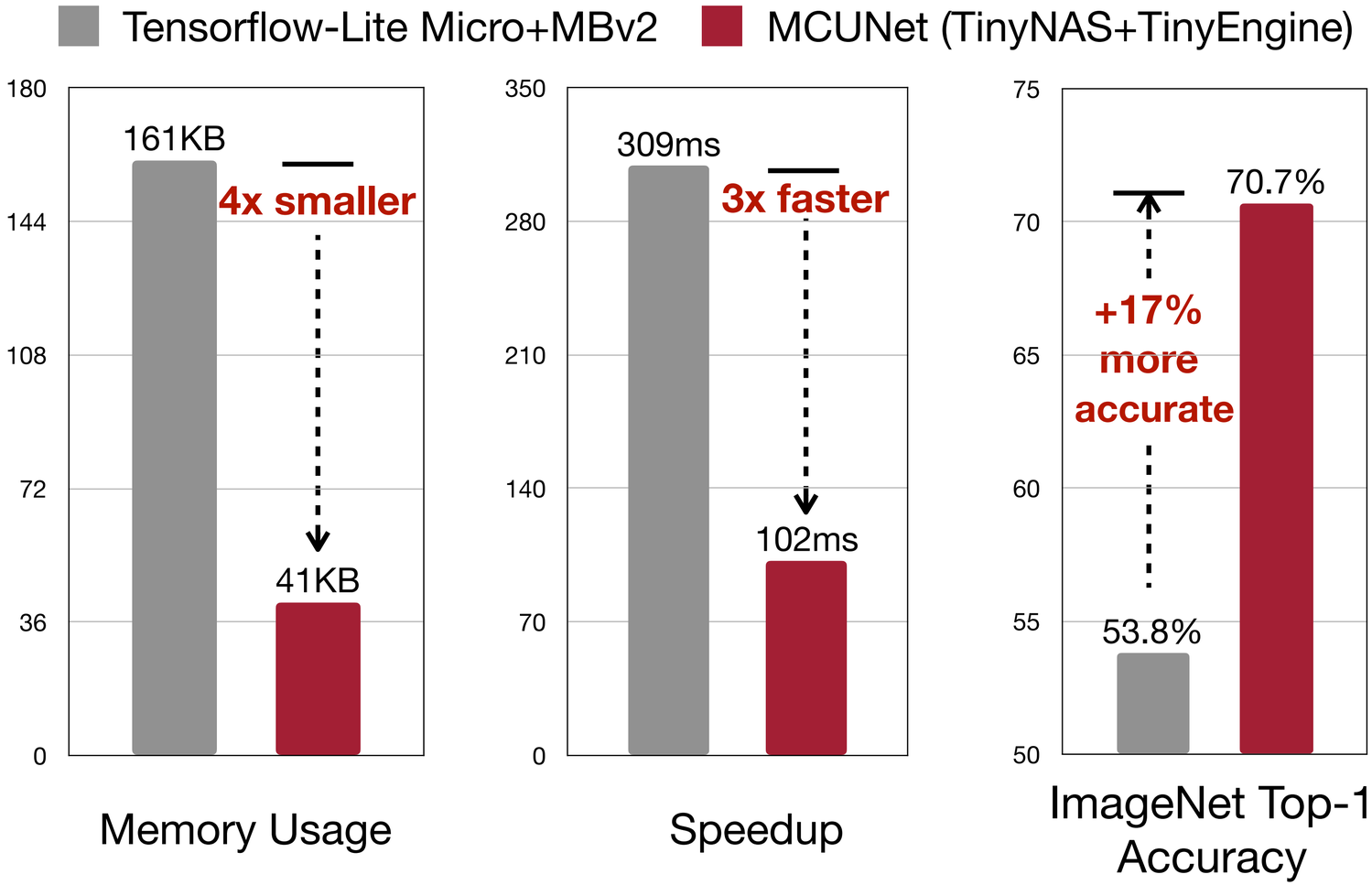
toy IoT applications

↓ MCUNet

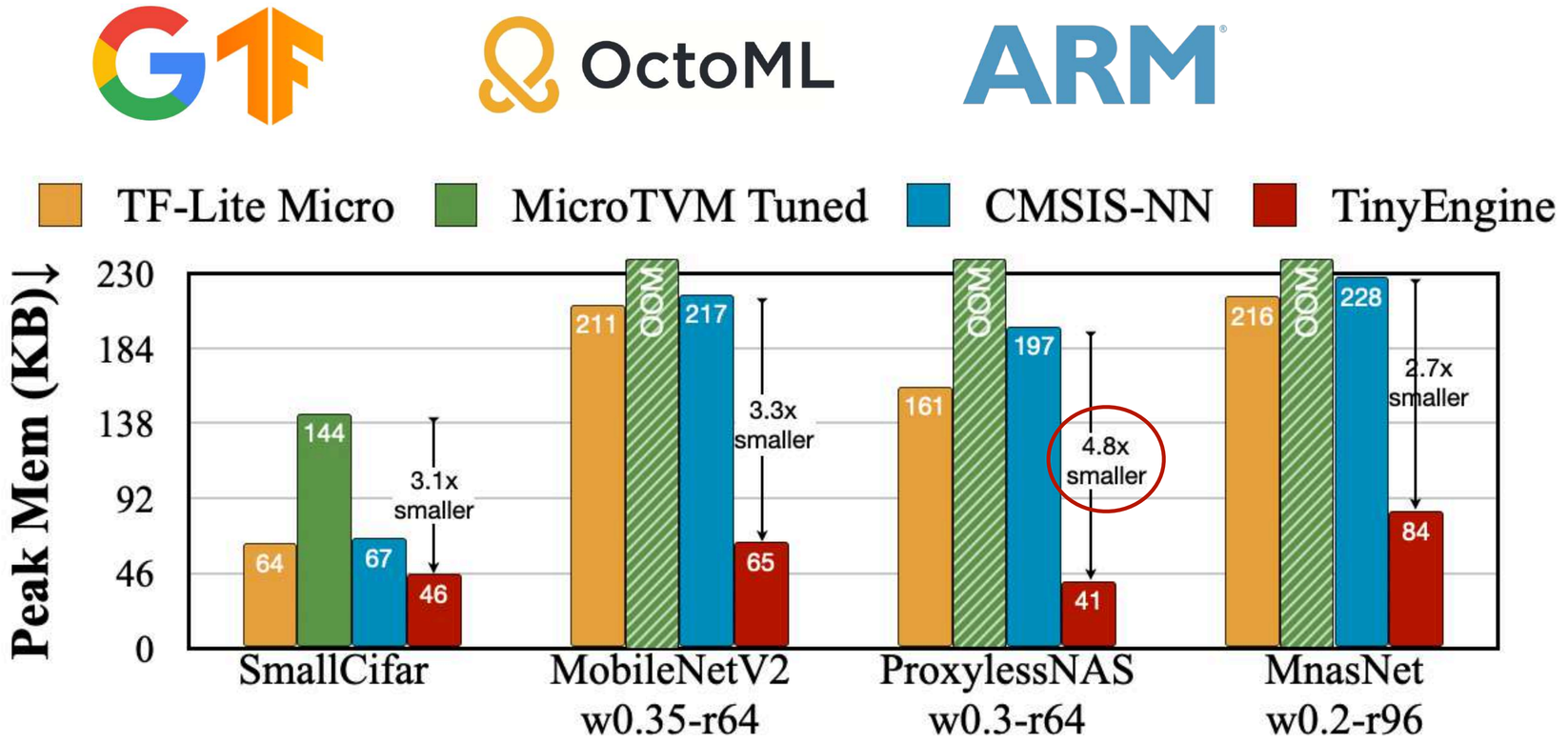
large scale, real-world applications



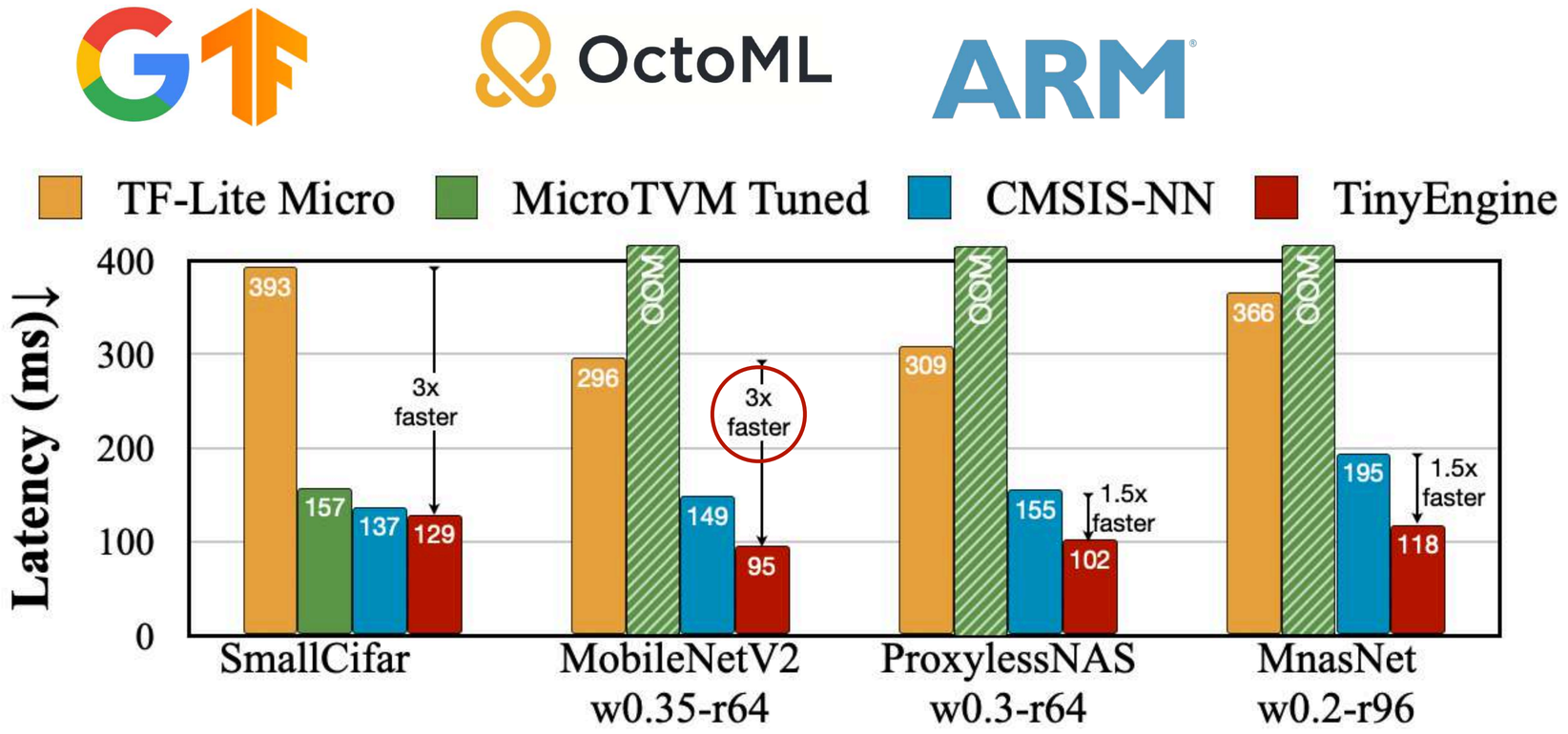
face/mask detection, person detection,
VWW on STM32F746 (1MB Flash)



TinyEngine: Memory Saving

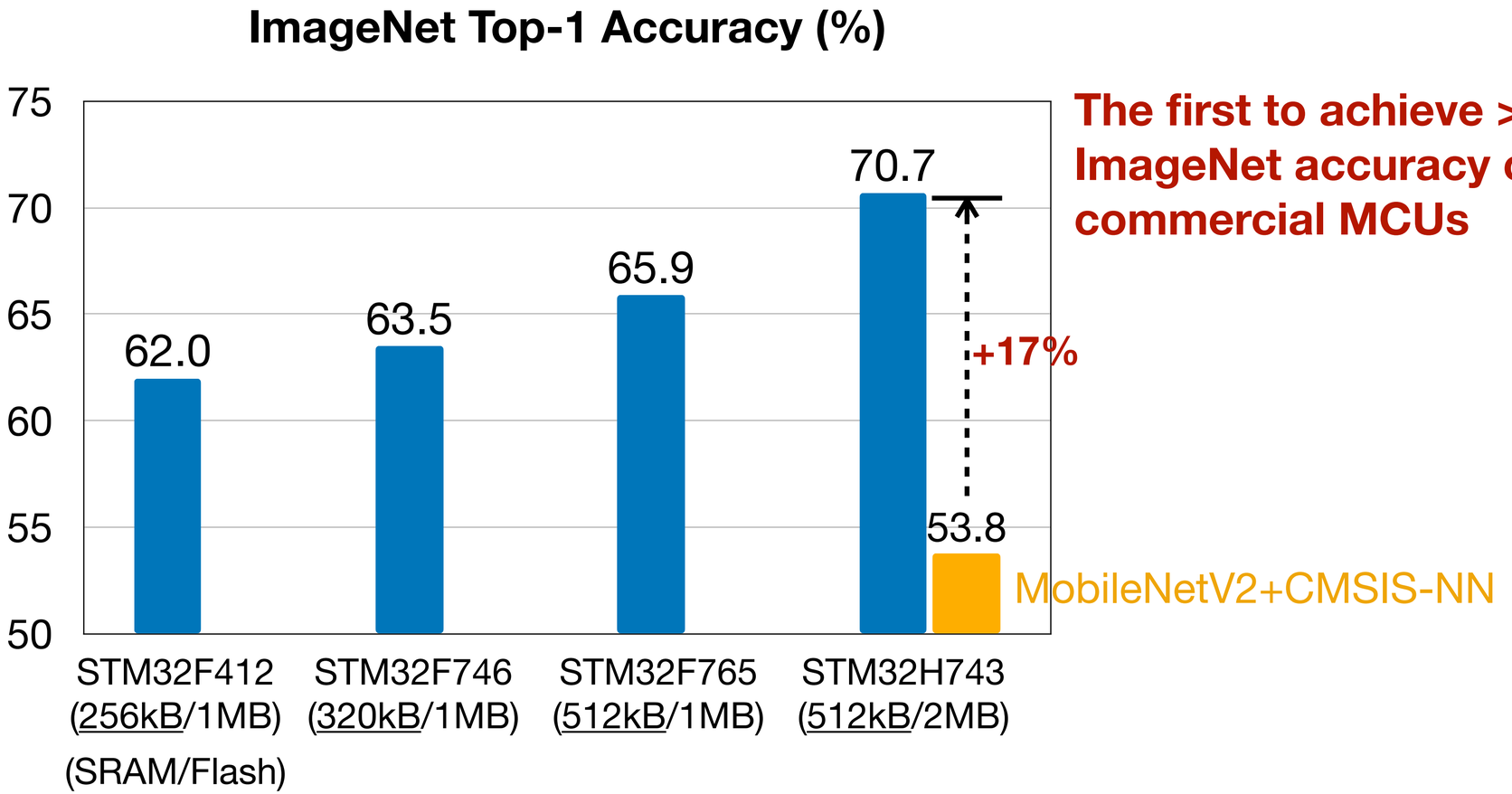


TinyEngine: Speedup



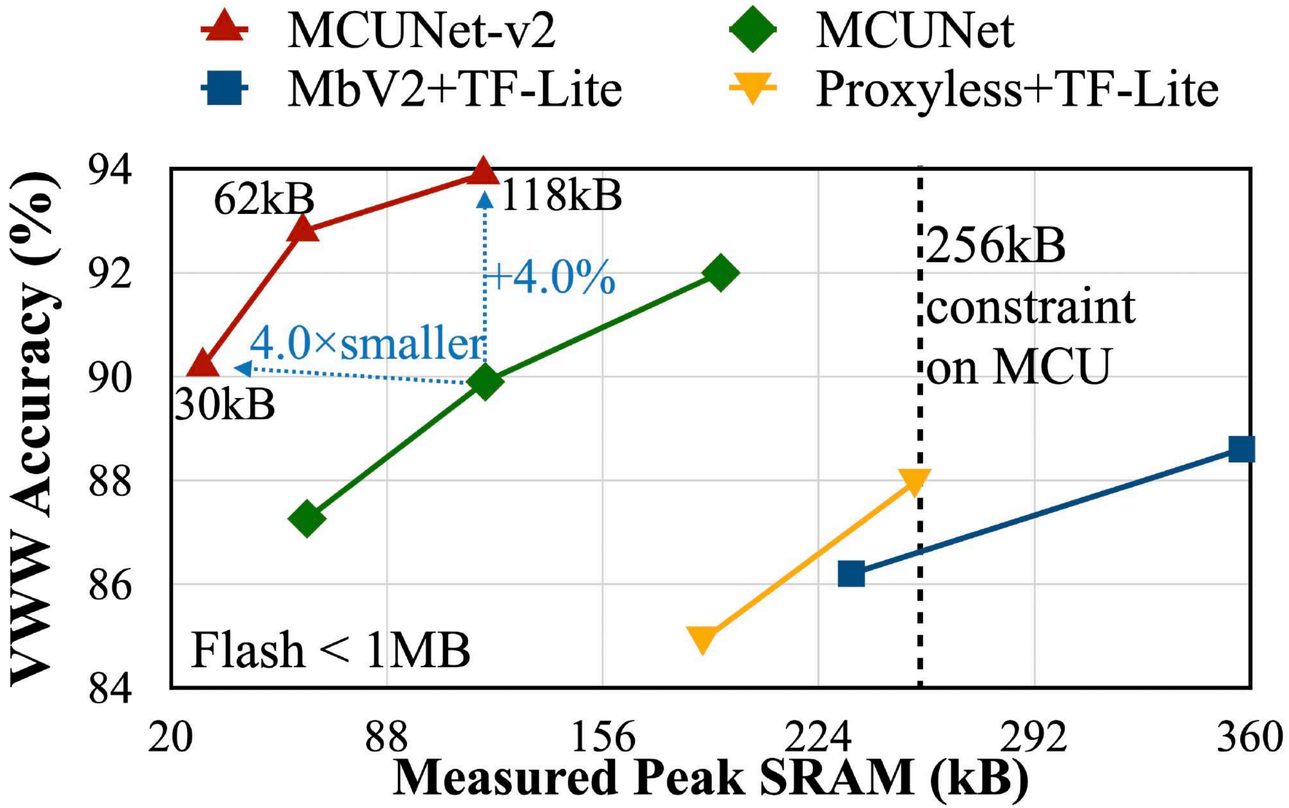
Once-for-All Network + MCUNet

- Design specialized models for different MCUs

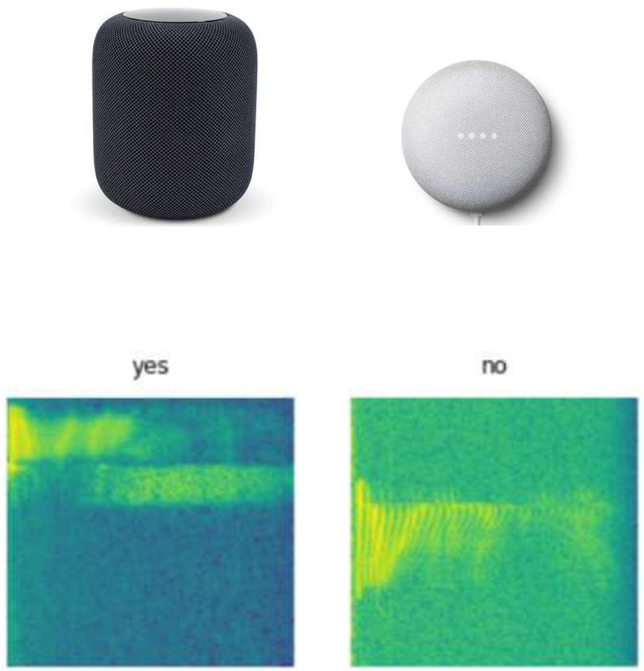


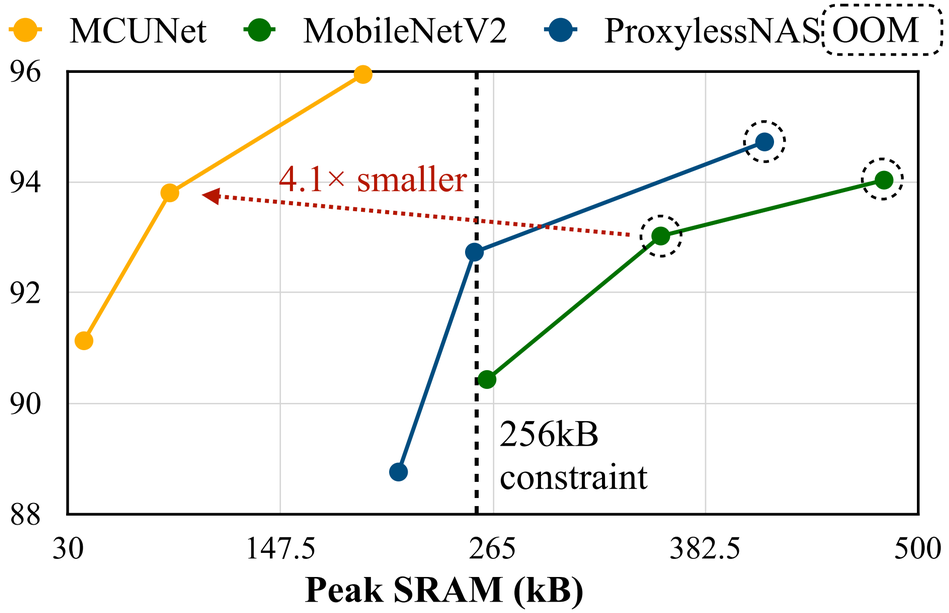
The first to achieve >70% ImageNet accuracy on commercial MCUs

Visual Wake Words (VWW)



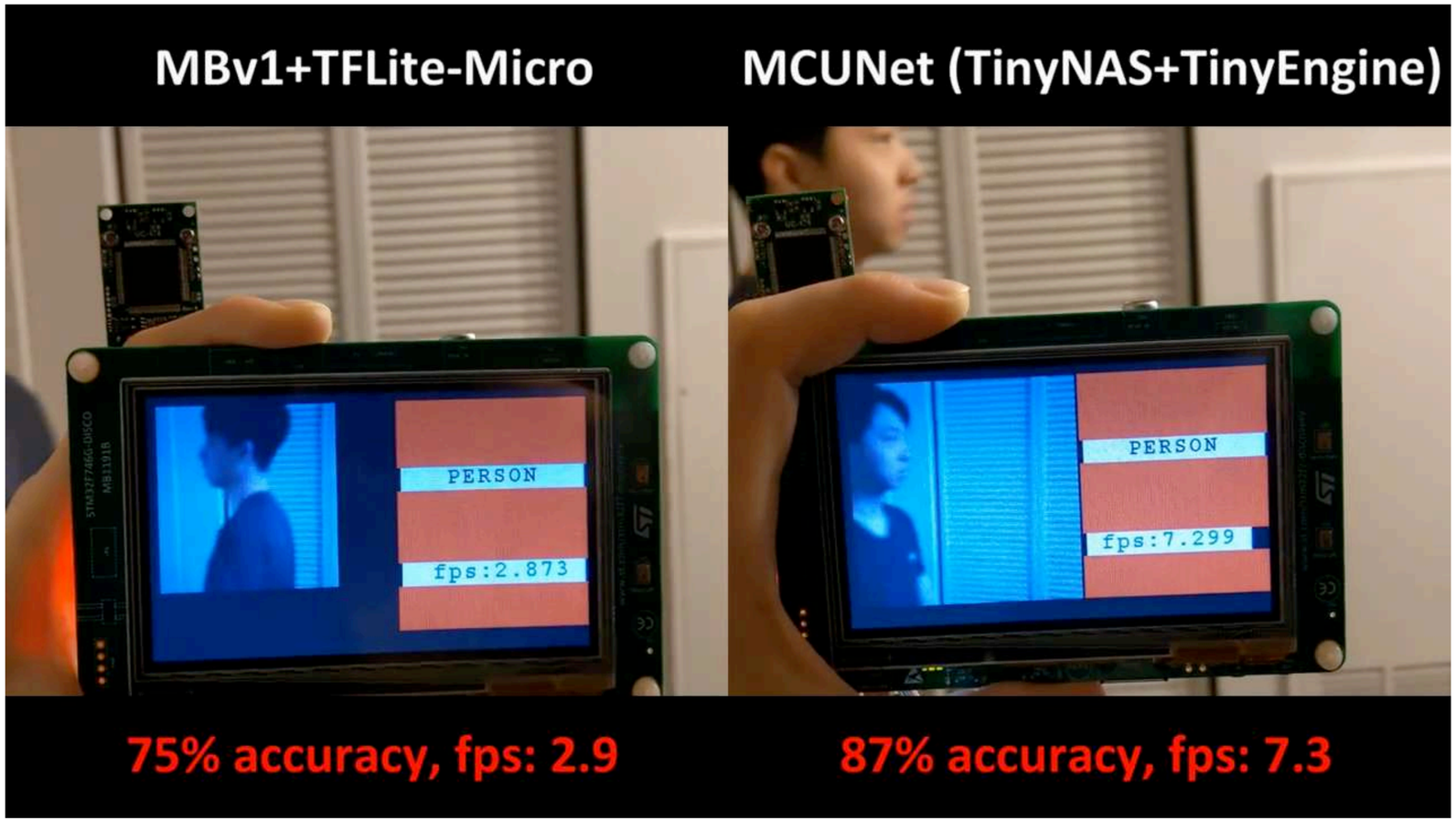
Audio Wake Words (Speech Commands)





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

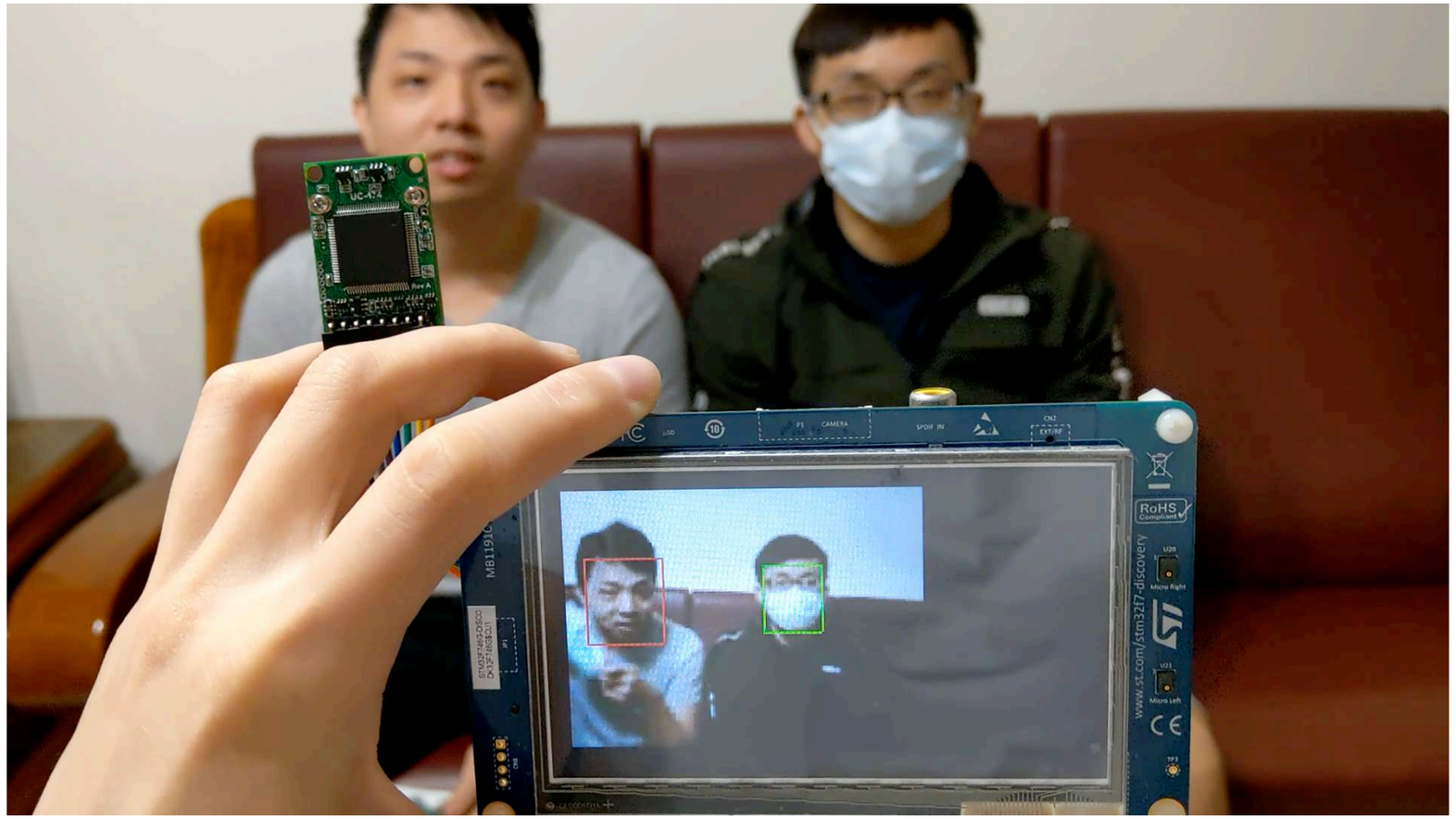
Demo: Visual Wake Words on MCU



- Detecting if there is person

- STM32F746
- 320KB SRAM
- 1MB Flash
- ARM Cortex-M7 @216MHz

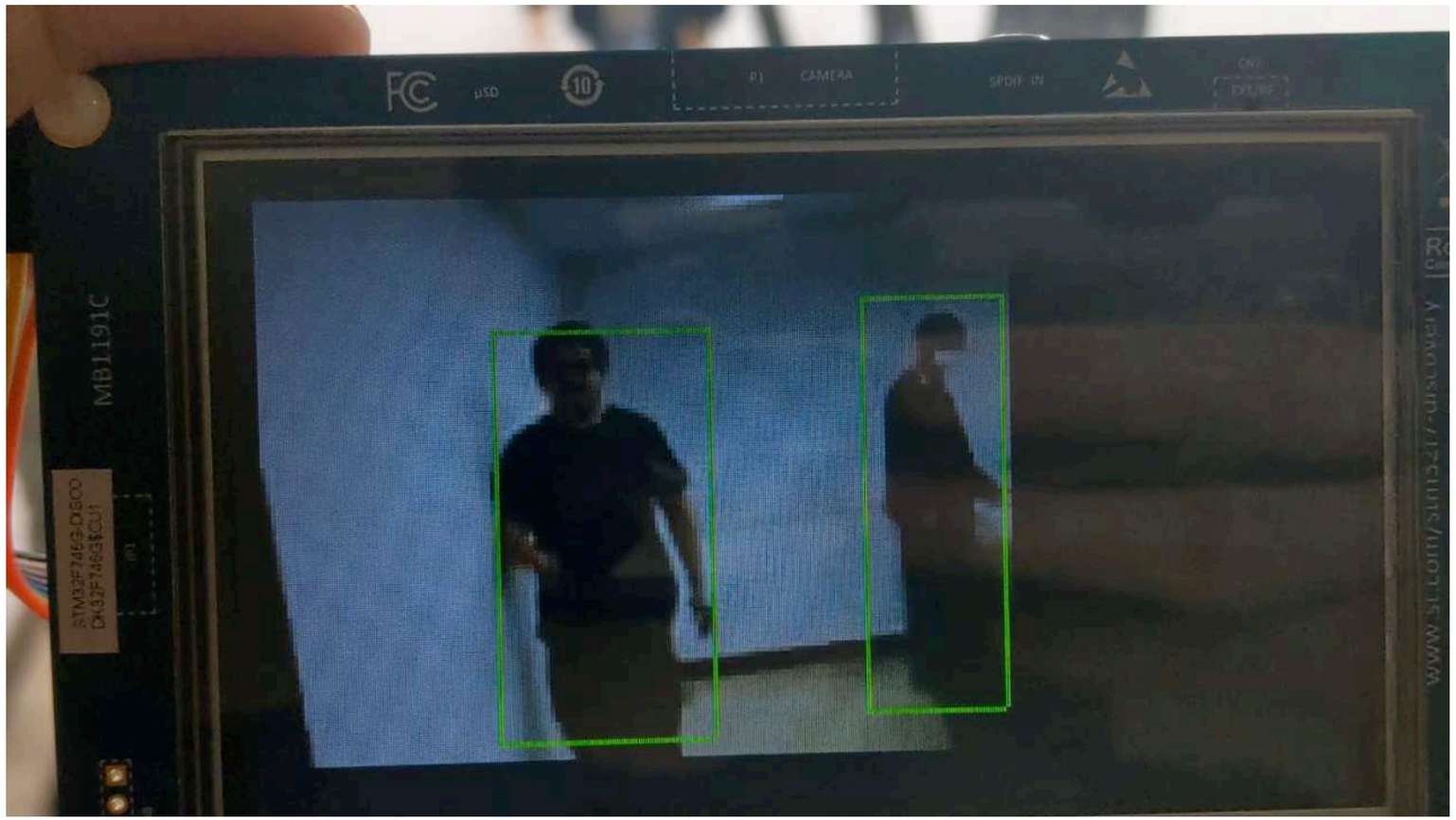
Demo: Face Mask Detection on MCU



- Detecting faces & masks

- STM32F746
- 320KB SRAM
- 1MB Flash
- ARM Cortex-M7 @216MHz

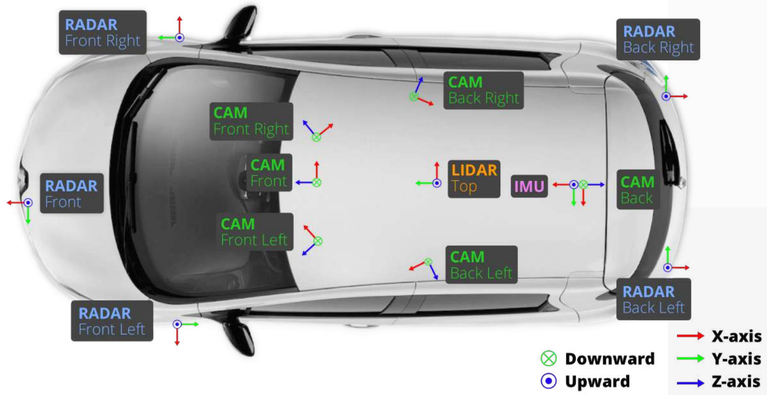
Demo: Person Detection on MCU

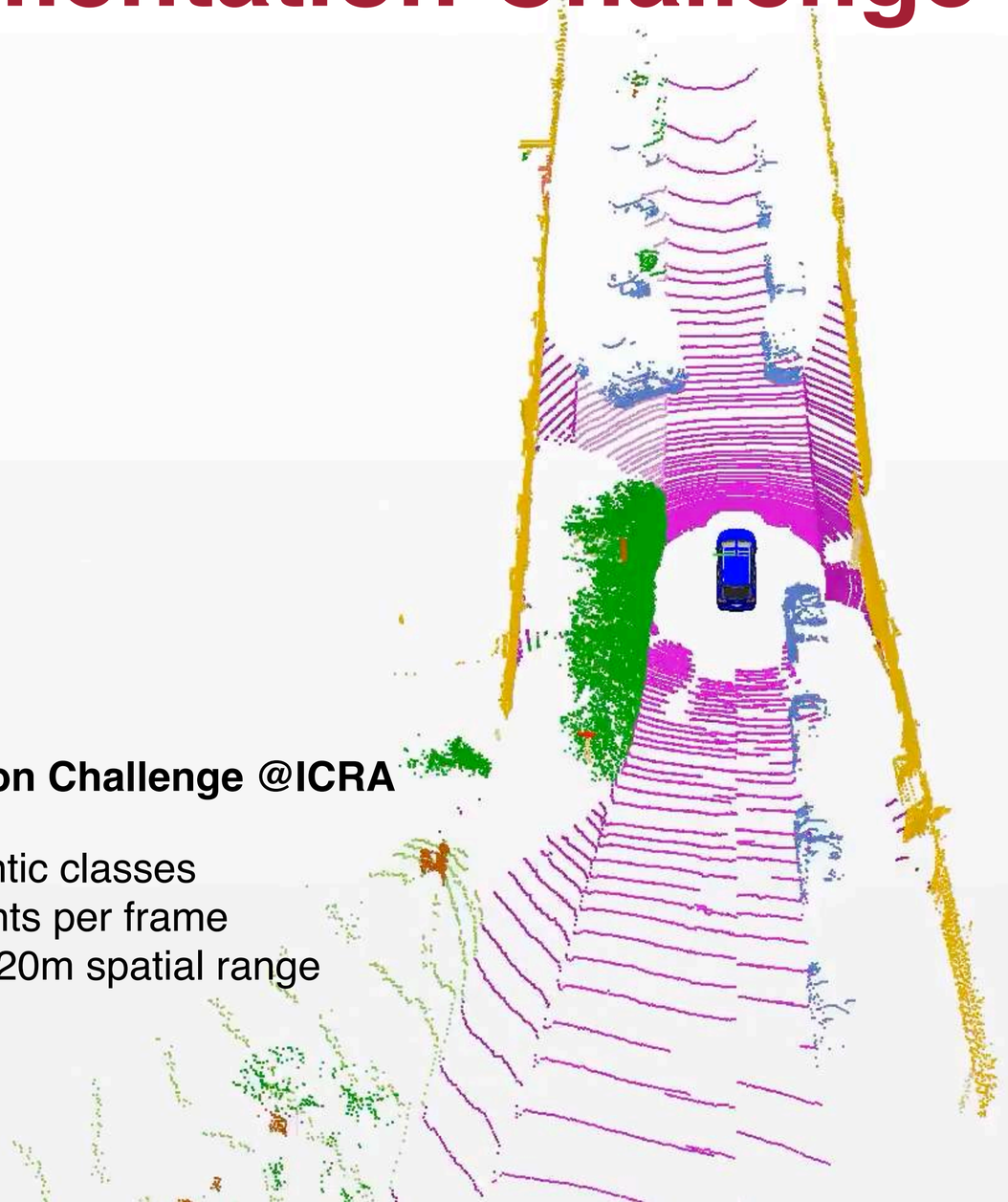


- Detecting persons

- STM32F746
- 320KB SRAM
- 1MB Flash
- ARM Cortex-M7 @216MHz

NuScenes LiDAR Segmentation Challenge





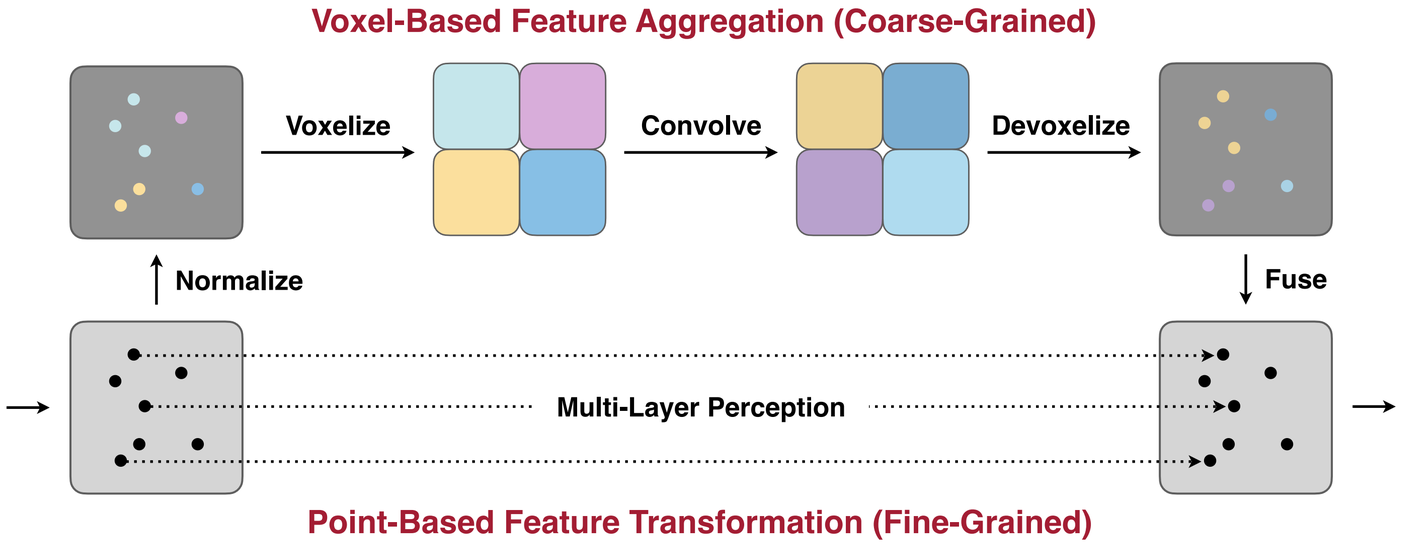
nuScenes Dataset

1 Spinning LiDAR (20 Hz, 32 channels)
40,000 annotated frames
1,000 driving sequences

LiDAR Segmentation Challenge @ICRA

16 semantic classes
35,000 points per frame
100m x 100m x 20m spatial range

Point-Voxel Convolution (PVConv)



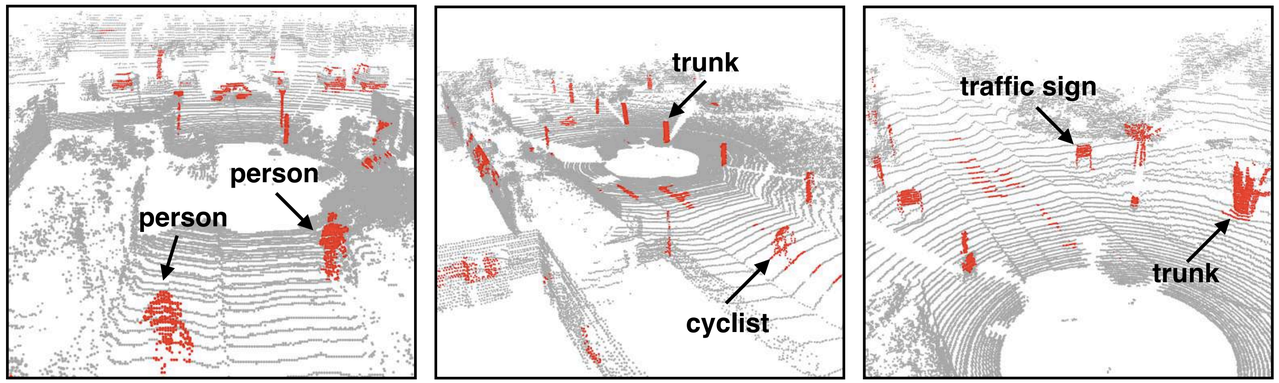
Comparisons with Point-Based Models

Voxel-based branch conducts convolution over a regular grid representation, which resolves the issue of random memory access.

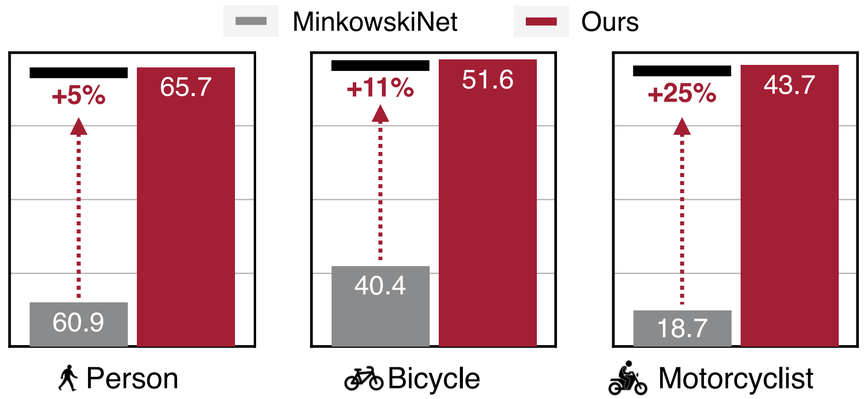
Comparisons with Voxel-Based Models

Point-based branch captures high-resolution information efficiently, which resolves the issue of large memory footprint.

Low-Cost Point-Based Branch Provides Fine Details of the Scene



Huge Improvements on Safety-Critical Small Objects



Challenge Results

Rank 1: SPVCNN++

Authors: Zhijian Liu, Haotian Tang, Kevin Shao, Song Han

Affiliation: MIT HAN Lab

Method: Use point cloud segmentation network composed of multiple Sparse Point-Voxel Convolutions. More computationally heavy variant of SPVCNN, longer training, Lovasz-Softmax-loss with standard cross entropy loss. Lidar-only. 13 FPS.

Link: https://spvnas.mit.edu, "Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution"

Mean IOU: 81.1% +0.8%

Award: Best lidarseg submission

Previous awards: 4th ranked lidarseg submission in NeurIPS 2020

NUSCENES by Motional — Home | nuScenes | nuImages | Resources | Tasks | About | Sign Up

	Method					Metrics			
Date	Name	Modalities		Map data	External data	mIOU	fwIOU	FPS (Hz)	Stats
		Any		All	All				
2021-05-27	SPVCNN++	**Ours (2021)**	Lidar	no	no	0.811	0.910	n/a	
2021-05-26	GU-Net	UISEE	Lidar	no	no	0.803	0.910	n/a	
2021-05-26	2D3DNet	Google	Camera, Lidar	no	yes	0.800	0.901	n/a	
2020-12-09	AF2S3Net	Huawei	Lidar	no	no	0.783 (+2.8)	0.885 (+2.5)	3.7 (x3.5)	
2020-12-09	Cylinder3D++	CUHK	Lidar	no	no	0.779	0.899	10.6	
2020-12-08	CPFusion	Horizon Robotics	Camera, Lidar, Radar	no	no	0.777	0.892	n/a	
2020-12-09	SPVNAS	**Ours (2020)**	Lidar	no	no	0.774	0.897	15.9	

Industry Submissions

Ranks 1st on the NuScenes Leaderboard

Best LiDARSeg Submission @ ICRA 2021

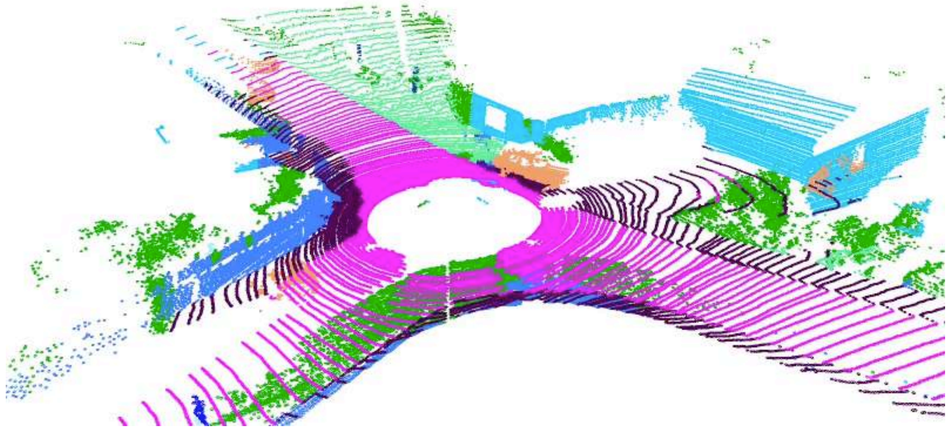
Efficient Point Cloud

SPVNAS, ECCV'20

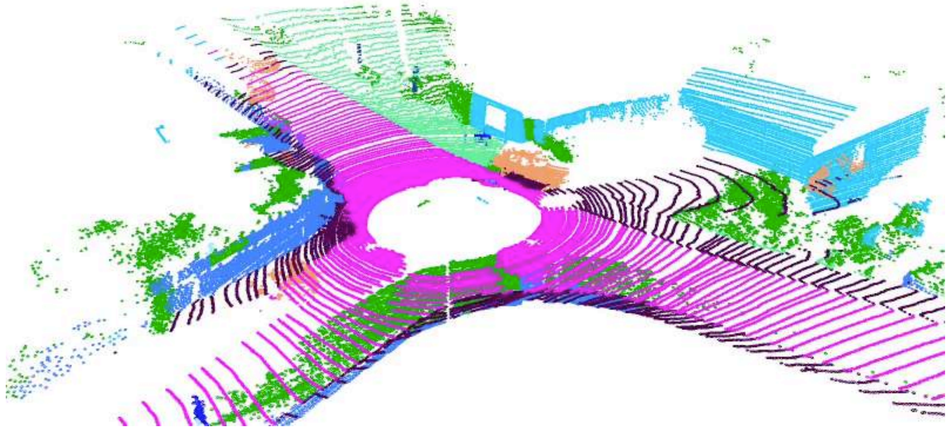
AR/VR: a whole backpack of computer

Self-driving: a whole trunk of GPU

Mobile phone: limited battery



MinkowskiNet: **3.4** FPS



SPVNAS (Ours): **9.1** FPS

Efficient Point Cloud

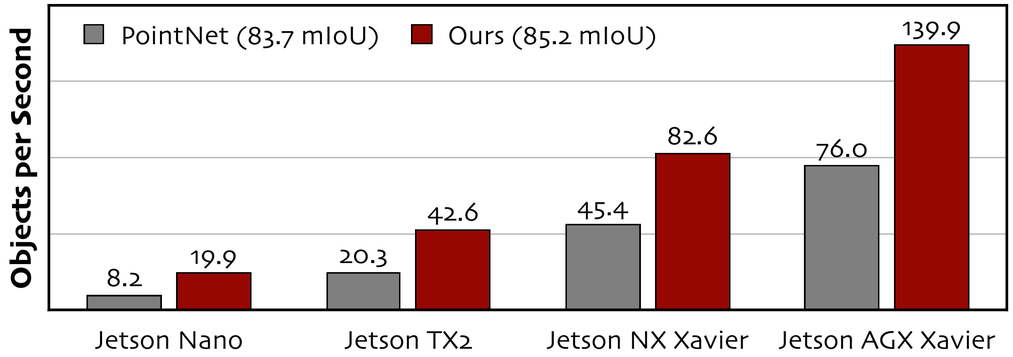
PVCNN, NeurIPS'19
SPVNAS, ECCV'20
FastLidarNet, ICRA'21

State-of-the-Art Accuracy

Approach	Paper	Code	mIoU	Classes (IoU)
SPVNAS			67.0	
TORNADONet			63.1	
KPRNet			63.1	
Cylinder3D			61.8	

First Place on SemanticKITTI Leaderboard (as of Fall 2020)

Real-Time Inference on Edge Devices

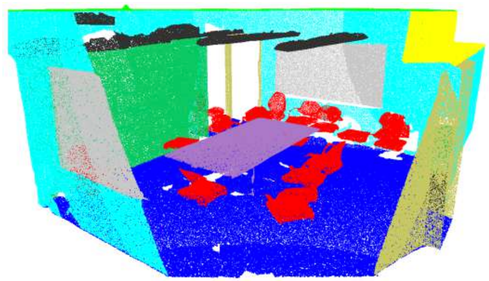


"**Project of the Month**" by NVIDIA Jetson Community

Object Part Segmentation

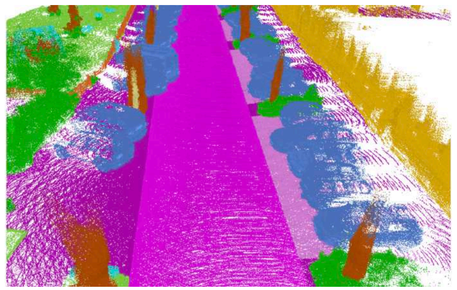
2.7x measured speedup
1.5x memory reduction

Indoor Scene Semantic Segmentation



6.9x measured speedup
5.7x memory reduction

Outdoor Scene Semantic Segmentation

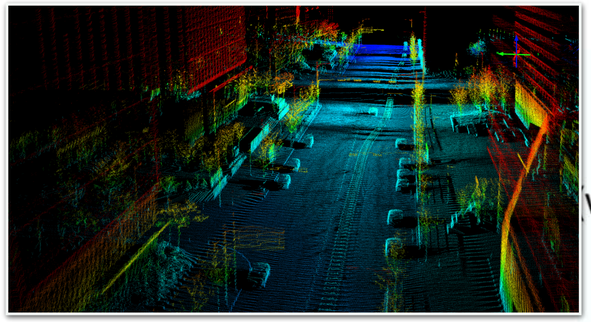


2.7x measured speedup
7.6x computation reduction

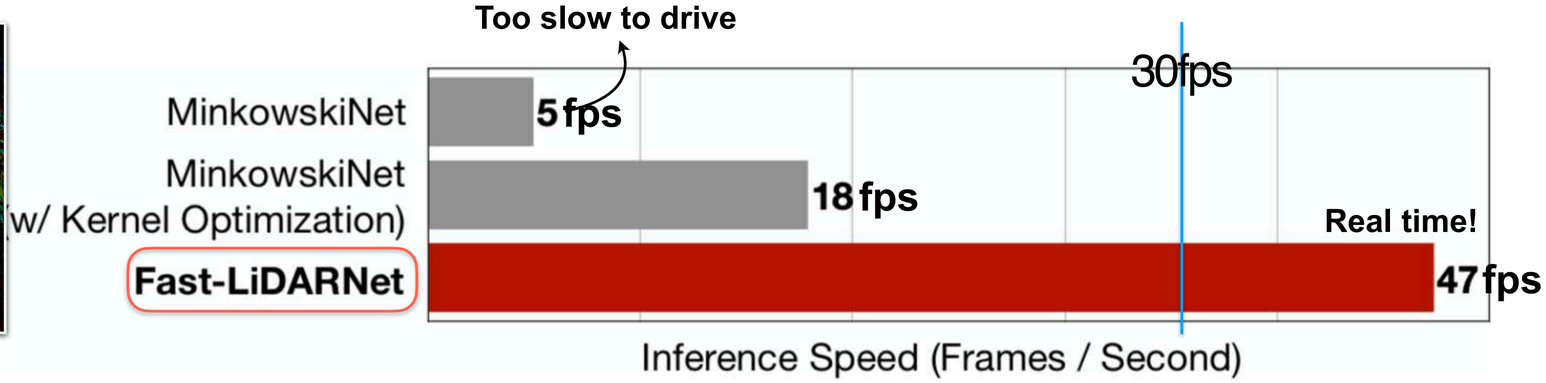
Efficient Point Cloud

[Liu et al. ICRA'21]
In collaboration with Daniela Rus

3D LiDAR Sensor



3D Point Cloud: 2M points/s

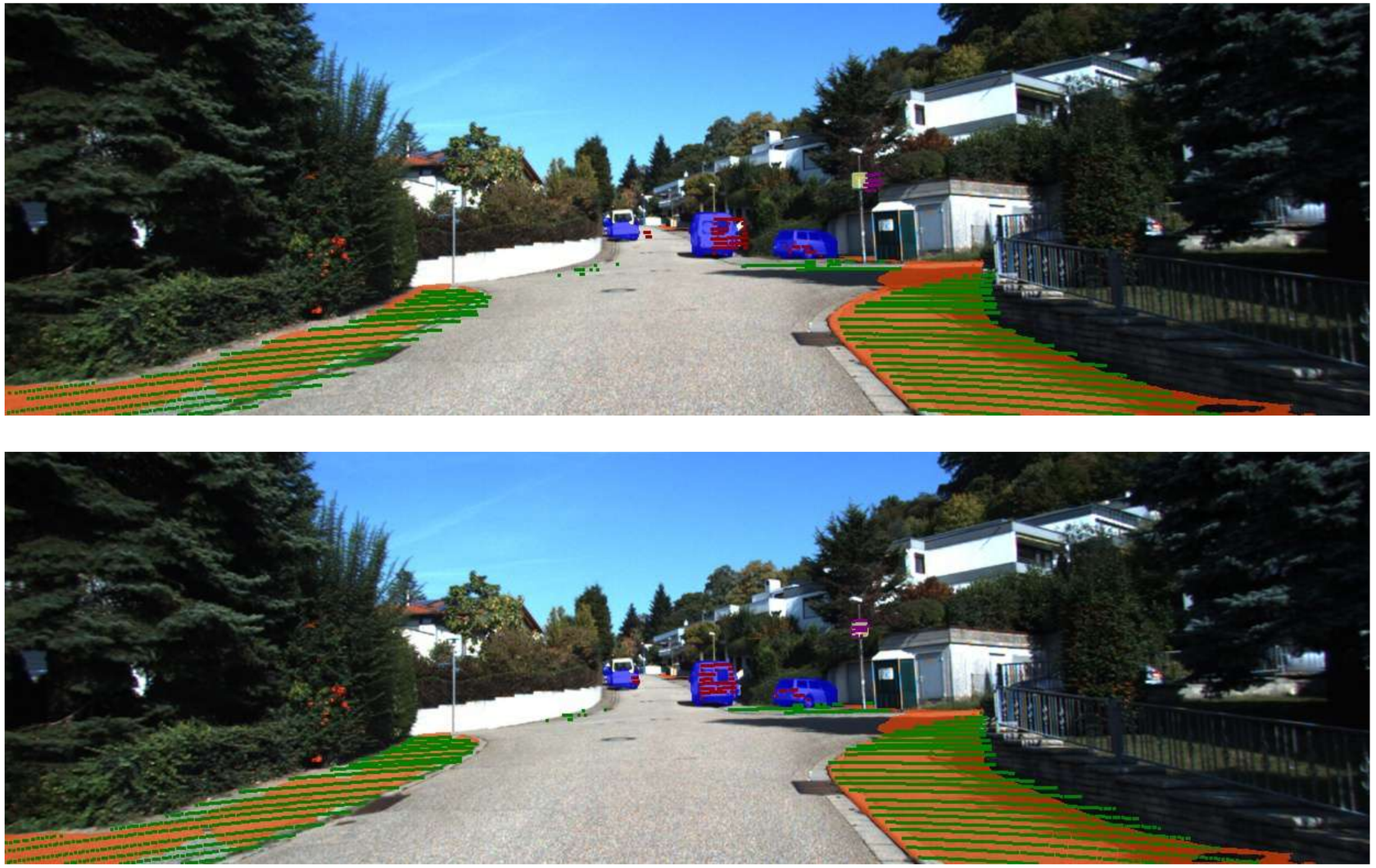


Demo:

SemAlign: Annotation-Free Camera-LiDAR Calibration with Semantic Alignment Loss

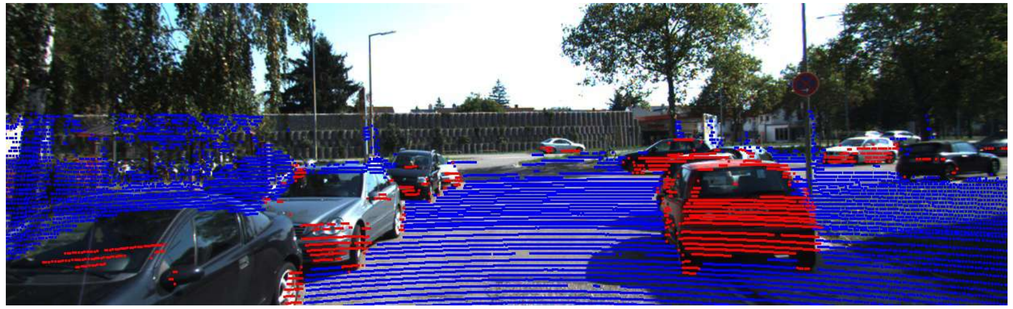
[IROS'21]

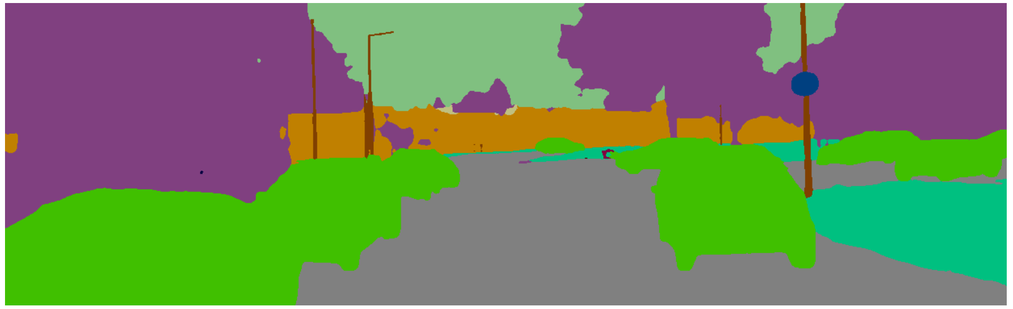
After **10** seconds of optimization by SemAlign:



- Car in 3D
- Car in 2D
- Sidewalk in 3D
- Sidewalk in 2D
- Traffic Sign in 3D
- Traffic Sign in 2D

SemAlign: Annotation-Free Camera-LiDAR Calibration with Semantic Alignment Loss





N points * 3 dimension

nn.Parameter(Calibration Matrix)

M points * 2 dimension

N points * 2 dimension

3D to 2D projection

N points * M copies * 2 dimension

torch.sum(torch.pow(left-right, 2))

N copies * M points * 2 dimension

distance matrix between each point

N * M matrix

torch.min()

distance matrix of each point to its nearest neighbor

N * 1 matrix

torch.mean()

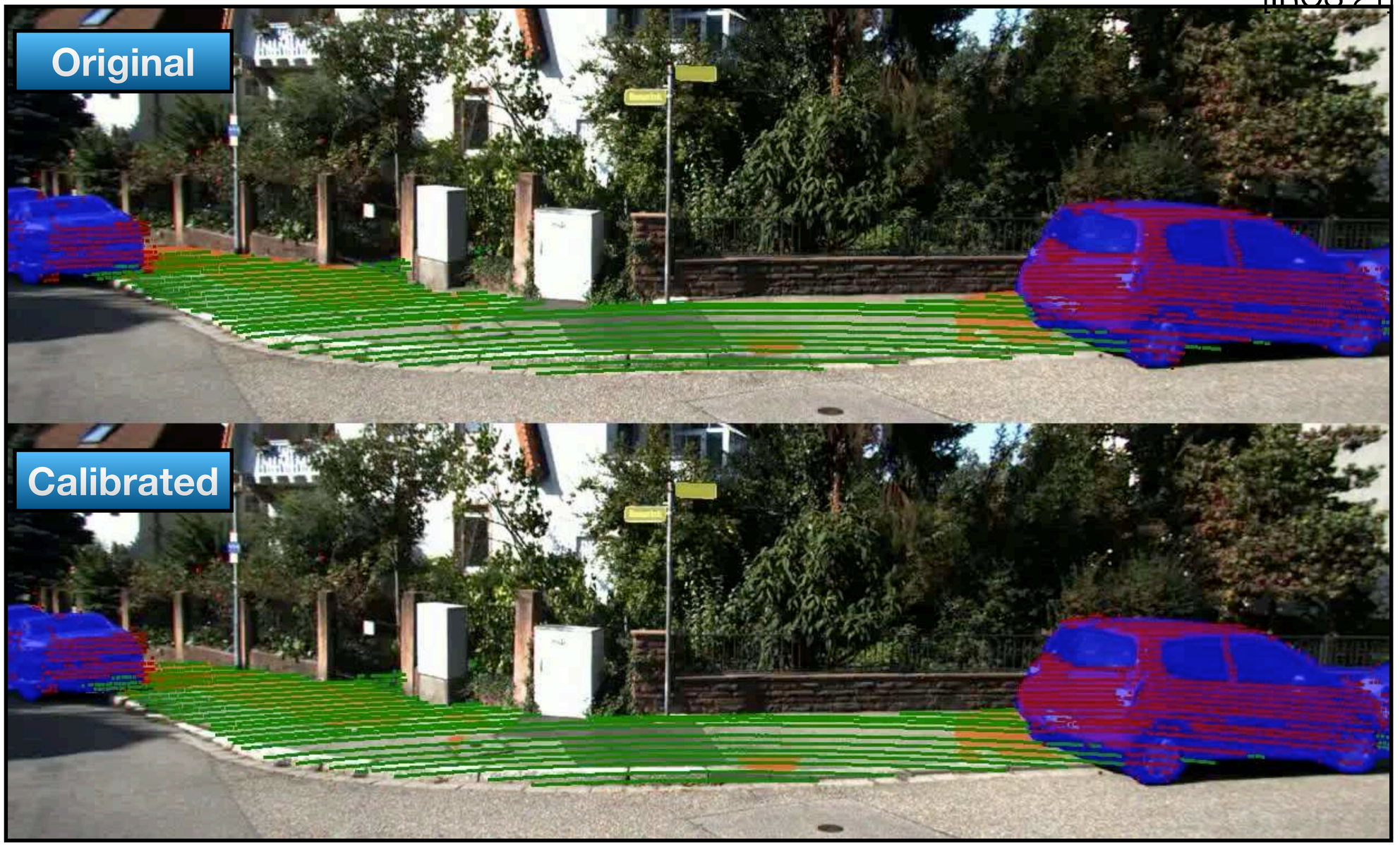
Loss

SemAlign: Annotation-Free Camera-LiDAR Calibration with Semantic Alignment Loss

[IROS'21]

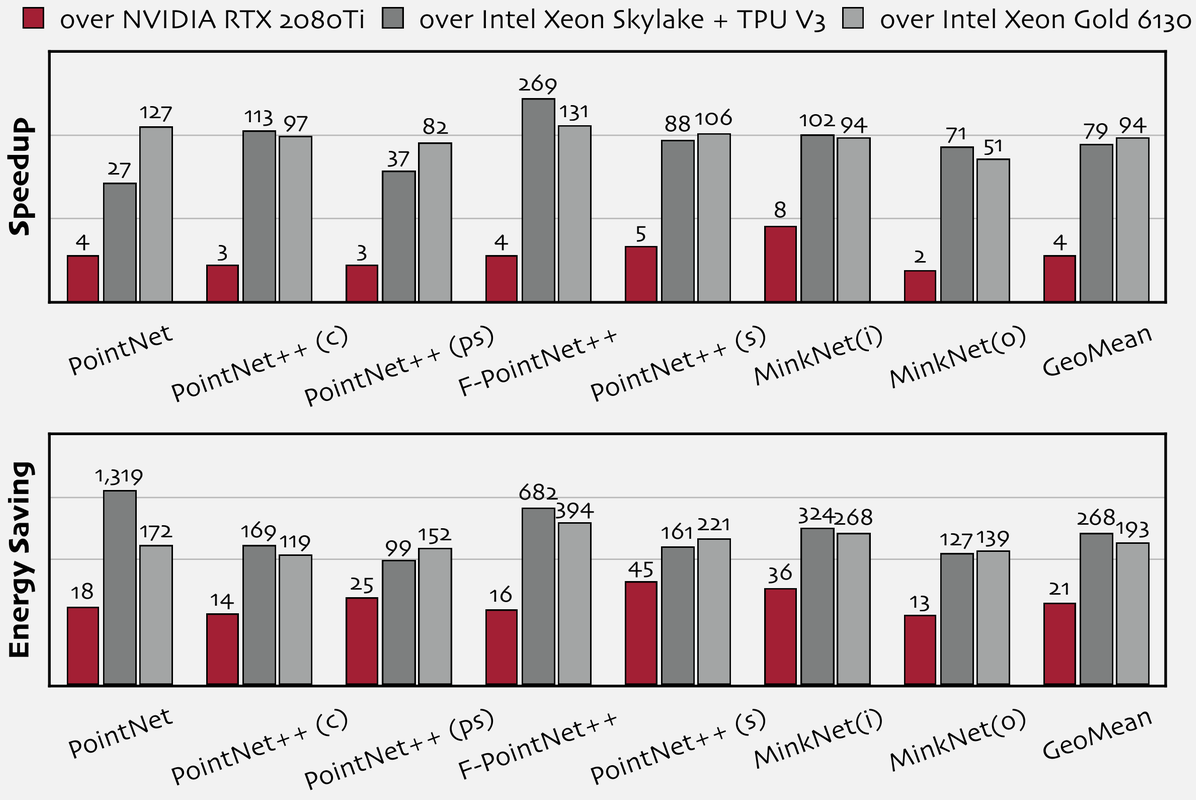
SemAlign: Annotation-Free Camera-LiDAR Calibration with Semantic Alignment Loss

[IROS'21]



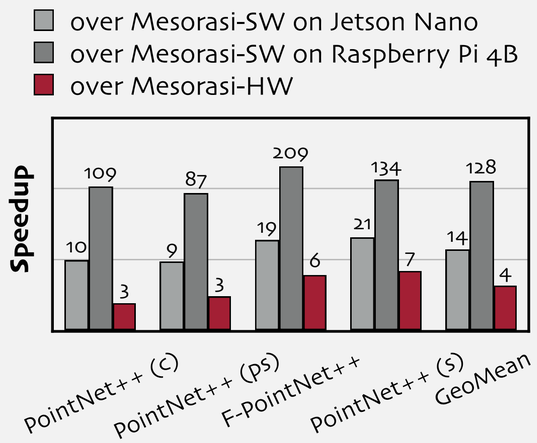
Specialized Accelerator Design: PointAcc [Micro'21]

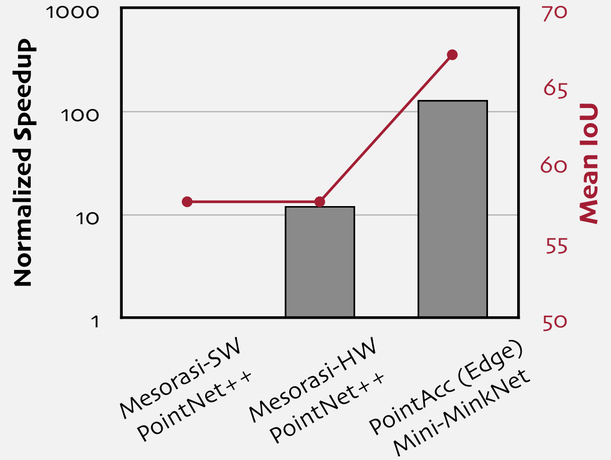
Significant Speedups and Energy Savings over CPU, GPU and TPU



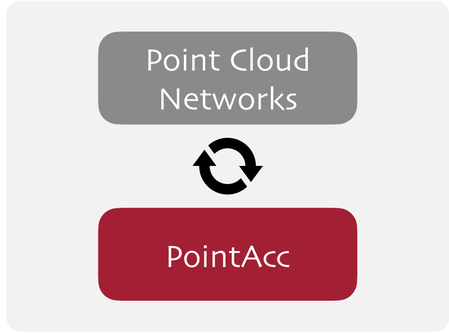
- **Assorted Applications**: classification, semantic segmentation, detection
- **Various Networks**: PointNet, PointNet++, MinkowskiNet
- **Diverse Inputs**: single objects, indoor scenes, outdoor scenes

Lower Latency and Higher Accuracy with PointAcc (Edge)

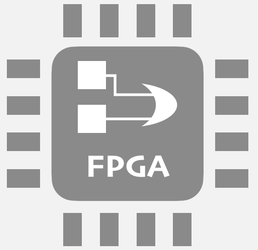




Future Plan



Co-Design with Point Cloud Networks



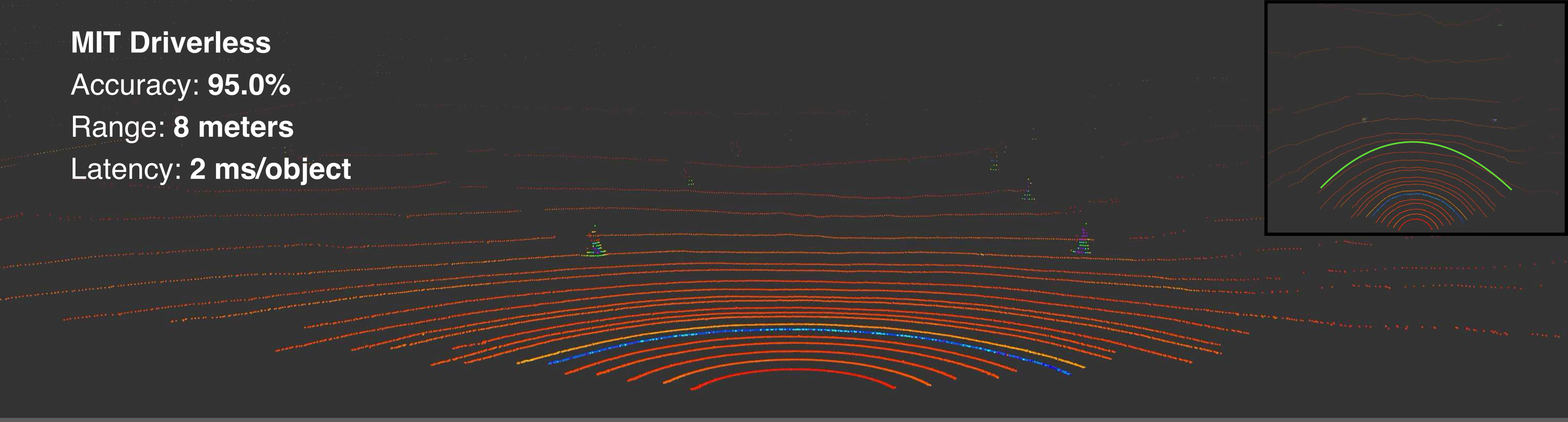
FPGA prototype

MIT Driverless

Accuracy: **95.0%**

Range: **8 meters**

Latency: **2 ms/object**

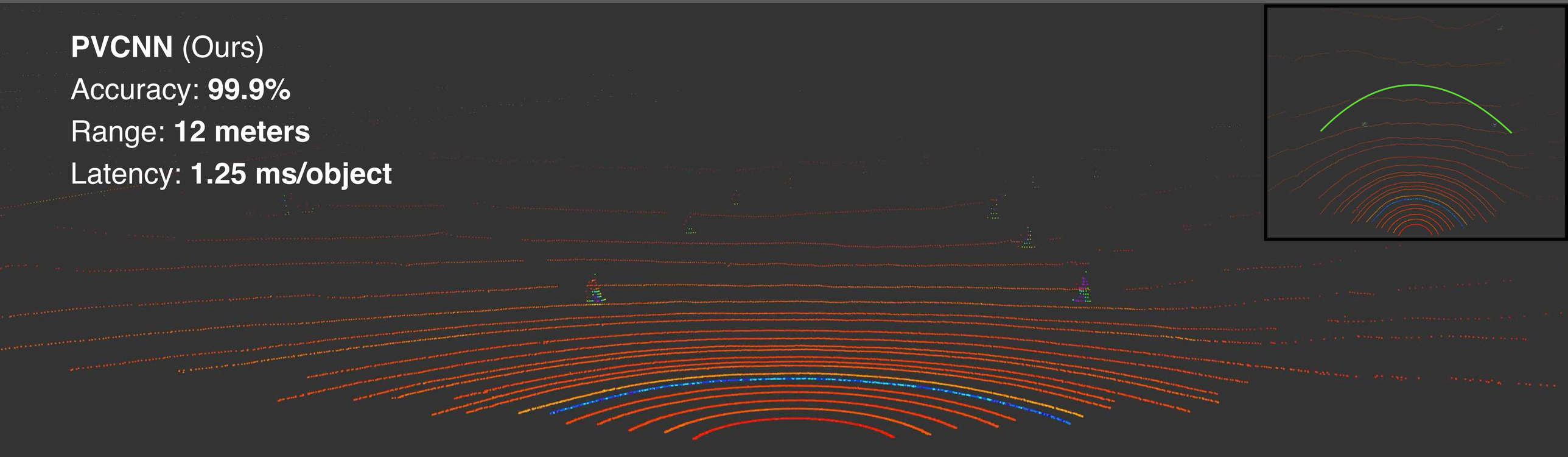


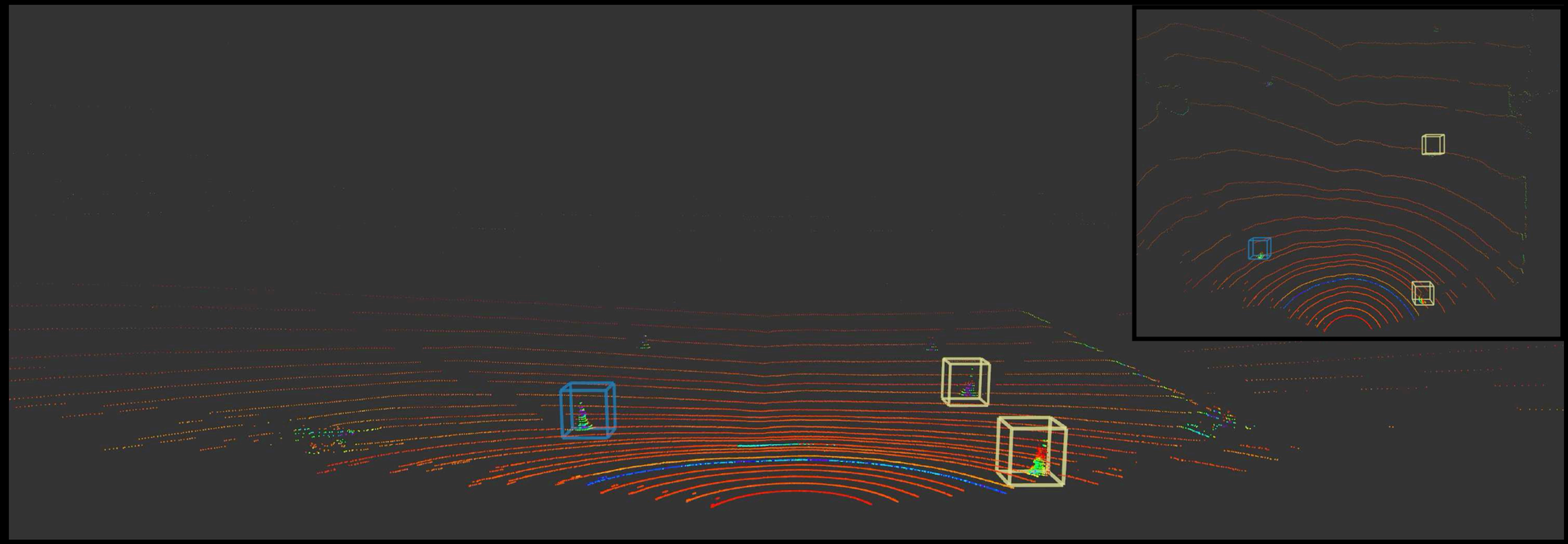
PVCNN (Ours)

Accuracy: **99.9%**

Range: **12 meters**

Latency: **1.25 ms/object**





Challenge Results

Rank 1: SPVCNN++

Authors: Zhijian Liu, Haotian Tang, Kevin Shao, Song Han

Affiliation: MIT HAN Lab

Method: Use point cloud segmentation network composed of multiple Sparse Point-Voxel Convolutions. More computationally heavy variant of SPVCNN, longer training, Lovasz-Softmax-loss with standard cross entropy loss. Lidar-only. 13 FPS.

Link: https://spvnas.mit.edu, "Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution"

Mean IOU: 81.1% +0.8%

Award: Best lidarseg submission

Previous awards: 4th ranked lidarseg submission in NeurIPS 2020

NUSCENES by Motional — Home | nuScenes | nuImages | Resources | Tasks | About | Sign Up

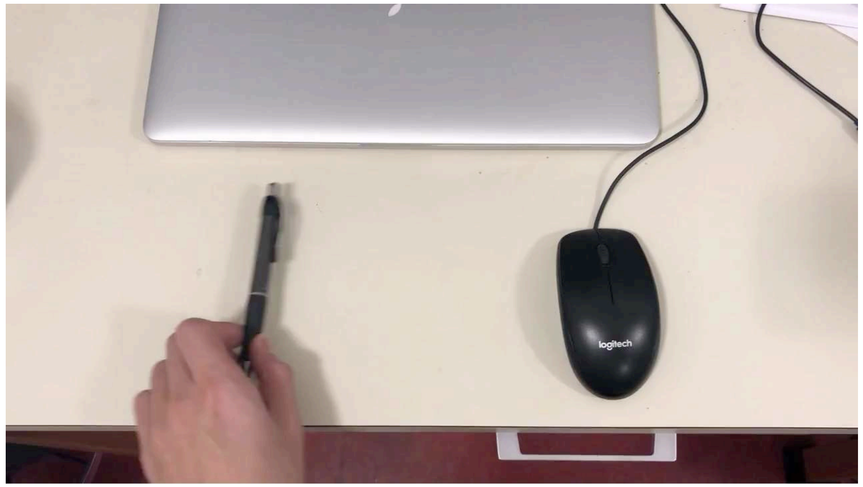
	Method					Metrics			
	Date	Name	Modalities	Map data	External data	mIOU	fwIOU	FPS (Hz)	Stats
			Any	All	All				
>	2021-05-27	SPVCNN++	**Ours (2021)** Lidar	no	no	0.811	0.910	n/a	
>	2021-05-26	GU-Net	**Industry Submissions** Lidar	no	no	0.803	0.910	n/a	
>	2021-05-26	2D3DNet	Camera, Lidar	no	yes	0.800	0.901	n/a	
>	2020-12-09	AF2S3Net	**Winner (2020)** Lidar	no	no	0.783 (+2.8)	0.885 (+2.5)	3.7 (x3.5)	
>	2020-12-09	Cylinder3D++	Lidar	no	no	0.779	0.899	10.6	
>	2020-12-08	CPFusion	Camera, Lidar, Radar	no	no	0.777	0.892	n/a	
>	2020-12-09	SPVNAS	**Ours (2020)** Lidar	no	no	0.774	0.897	15.9	

Ranks 1st on the NuScenes Leaderboard

Best LiDARSeg Submission @ ICRA 2021

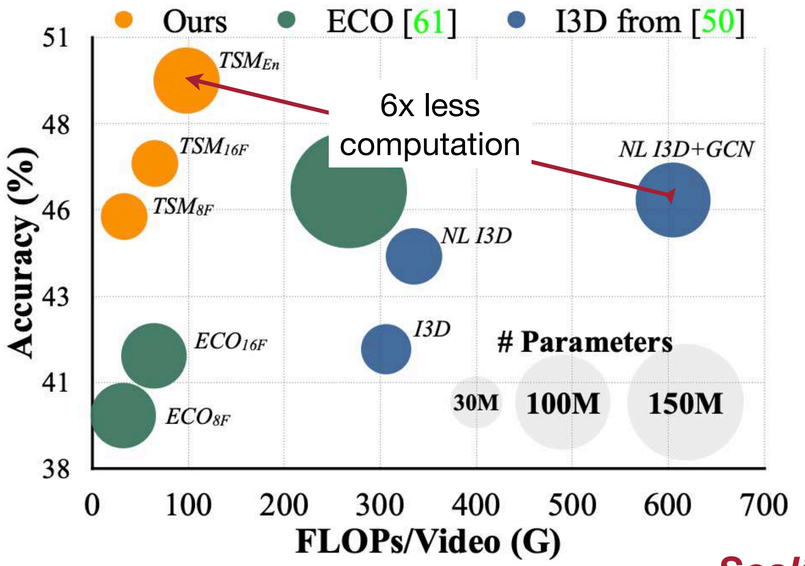
Efficient Video Recognition

TSM, ICCV 2019

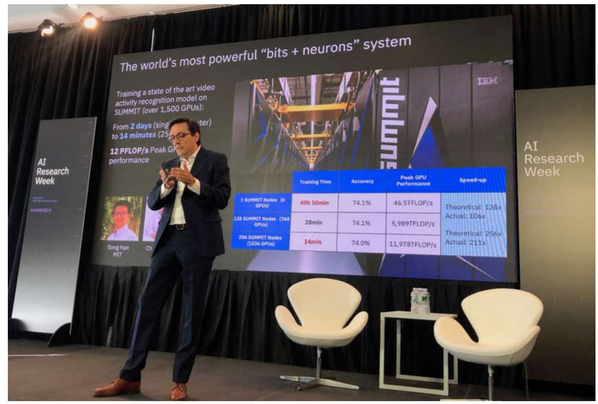


Prediction: Moving something closer to something

Compress the model by 6x, higher accuracy



Spotlight by IBM director Dario Gil
@MIT-IBM Watson AI Lab's AI Research Week



Scaling Down Inference:
Inference on Low-power Edge Devices

Devices	Jetson Nano		Jetson TX2		Rasp.	Note8	Pixel1
	CPU	GPU	CPU	GPU			
FPS	20.9	74.6	27.5	117.6	14.4	29.0	21.1
Power (watt)	4.8	4.5	5.6	5.8	3.8	- **LED Bulb Level!**	

Scaling Up Training:
Large-Scale Distributed Training with 1536 GPUs

	Training Time	Accuracy	Peak GPU Performance	Speed-up
1 Nodes (6 GPUs)	49h 50min	74.1%	46.5 TFLOP/s	1x
256 Nodes (1536 GPUs)	14min	74.0%	11,978 TFLOP/s	**211x**

TinyML and Efficient Deep Learning

- **AutoML and NAS**
  - Once-for-all Network [ICLR'19]

- **TinyML**
  - Inference: MCUNet for IoT Devices [NeurIPS'20, spotlight]
  - Training: Tiny On-Device Transfer Learning (TinyTL) [NeurIPS'20]

- **Data-Efficiency**
  - Differentiable Augmentation for Data-Efficient GAN Training [NeurIPS'20]

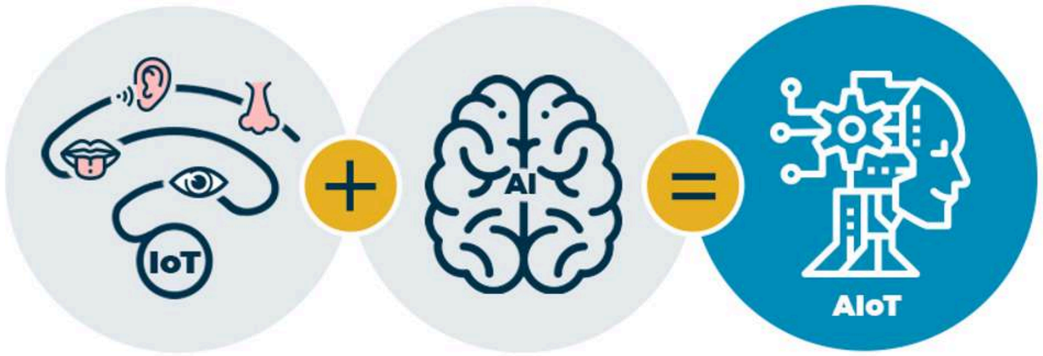
TinyTL: Reduce Memory, not Parameters for Efficient On-Device Learning

Han Cai[1] Chuang Gan[2] Ligeng Zhu[1] Song Han[1]

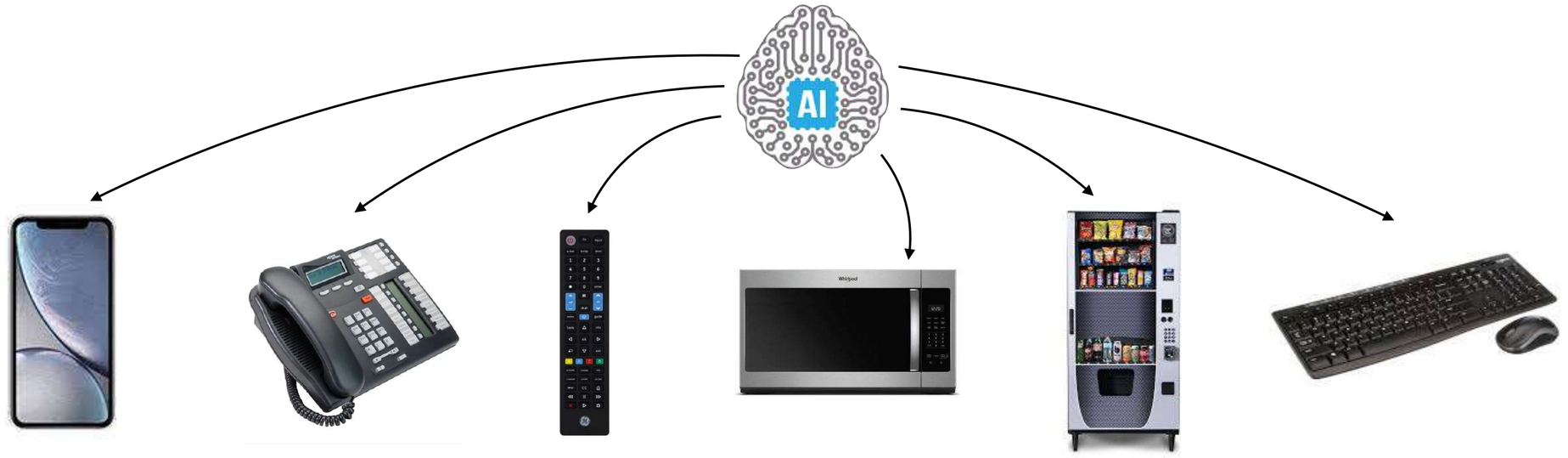
[1]MIT [2]MIT-IBM Watson AI Lab

NeurIPS 2020

The Rise of AIoT



IoT + AI = AIoT



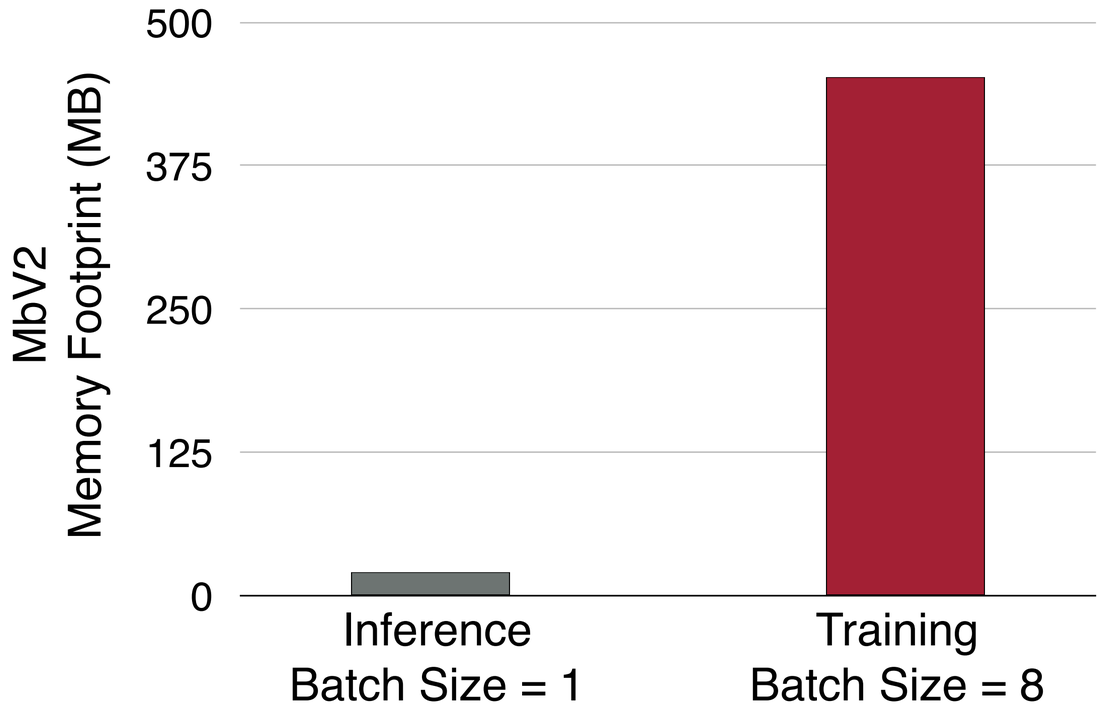
Tiny Transfer Learning

- Customization: AI systems need to continually adapt to new data collected from the sensors.

Tiny Transfer Learning

- Customization: AI systems need to continually adapt to new data collected from the sensors.
- Security: Data cannot leave devices because of security and regularization.
- TinyTL reduces the training memory **from 300MB to 16MB**

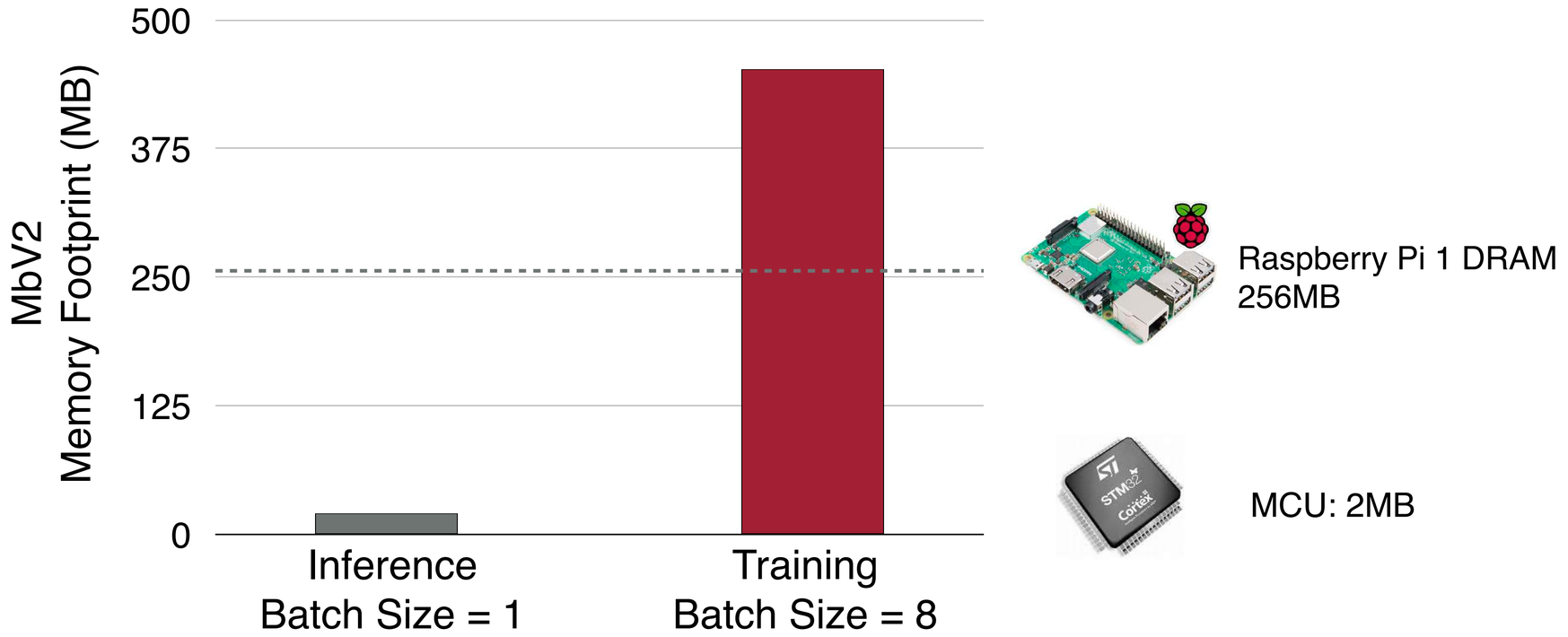
Training Memory is much Larger than Inference



- Edge devices have tight memory constraints. The training memory footprint of neural networks can easily exceed the limit.

- Edge devices are energy-constrained. Failing to fit the training process into the energy-efficient on-chip SRAM will significantly increase the energy cost.

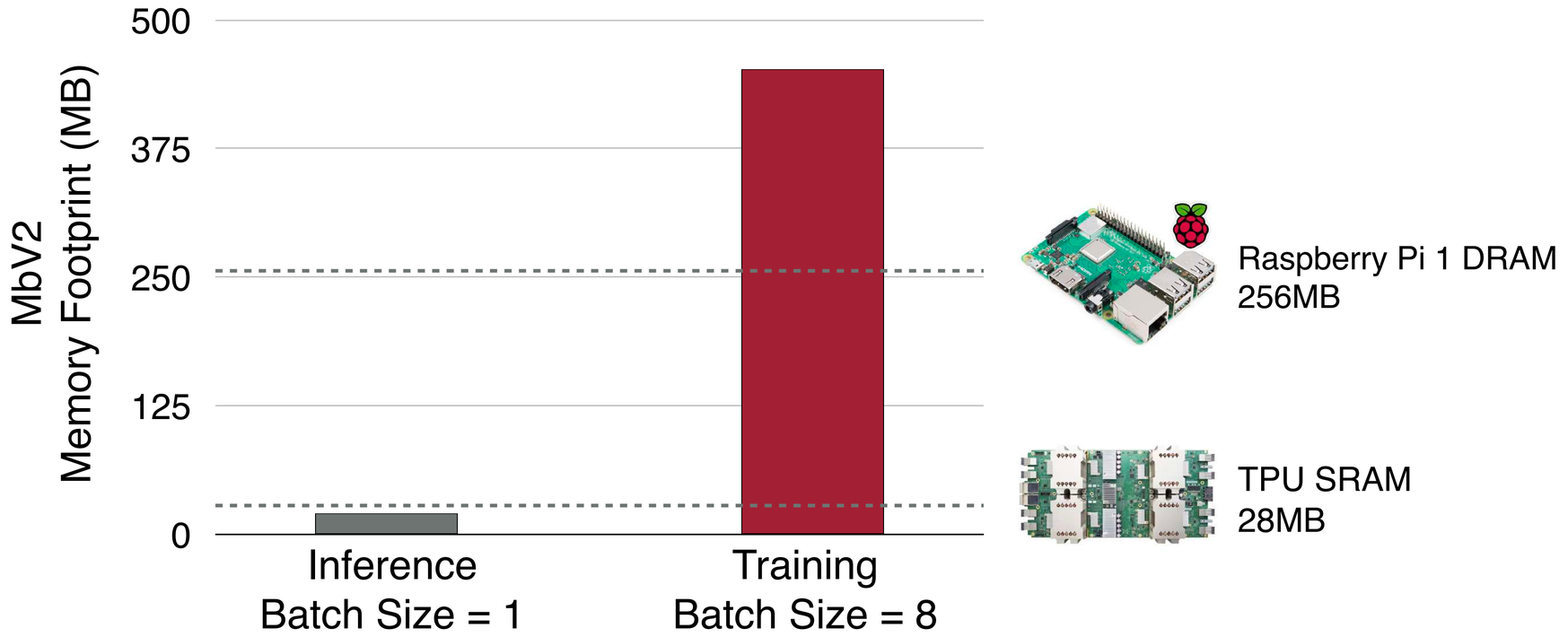
Training Memory is much Larger than Inference



- Edge devices have tight memory constraints. The training memory footprint of neural networks can easily exceed the limit.

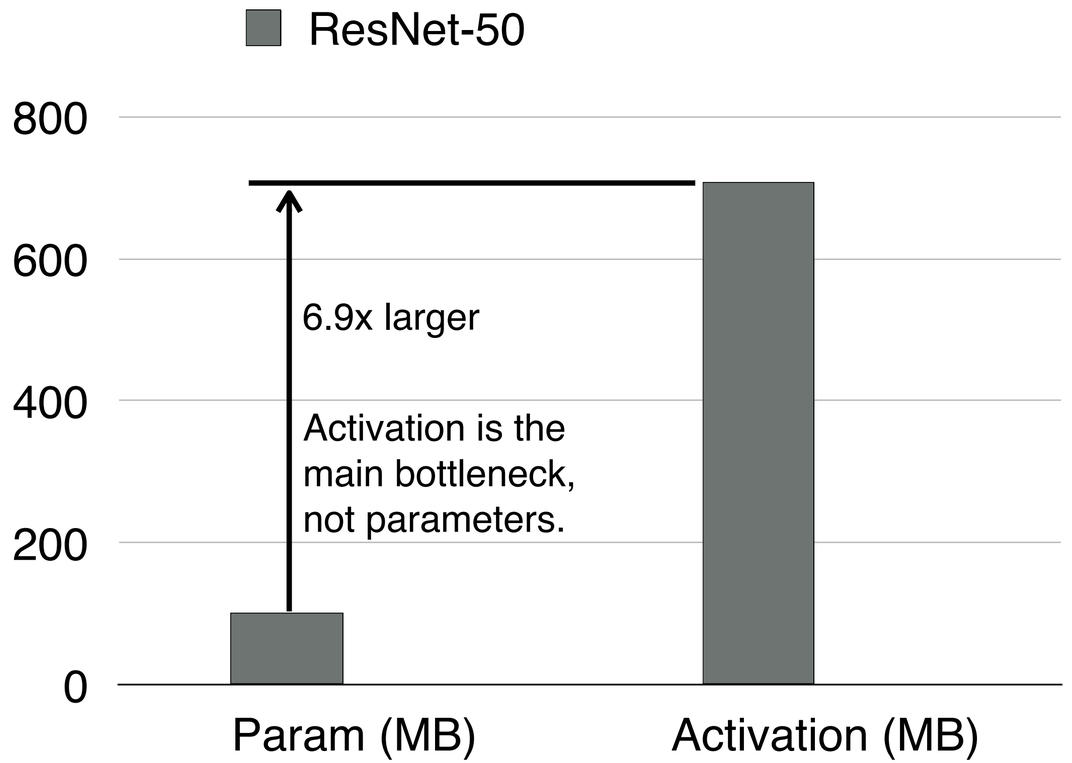
- Edge devices are energy-constrained. Failing to fit the training process into the energy-efficient on-chip SRAM will significantly increase the energy cost.

Efficient On-device Learning Requires Small Training Memory



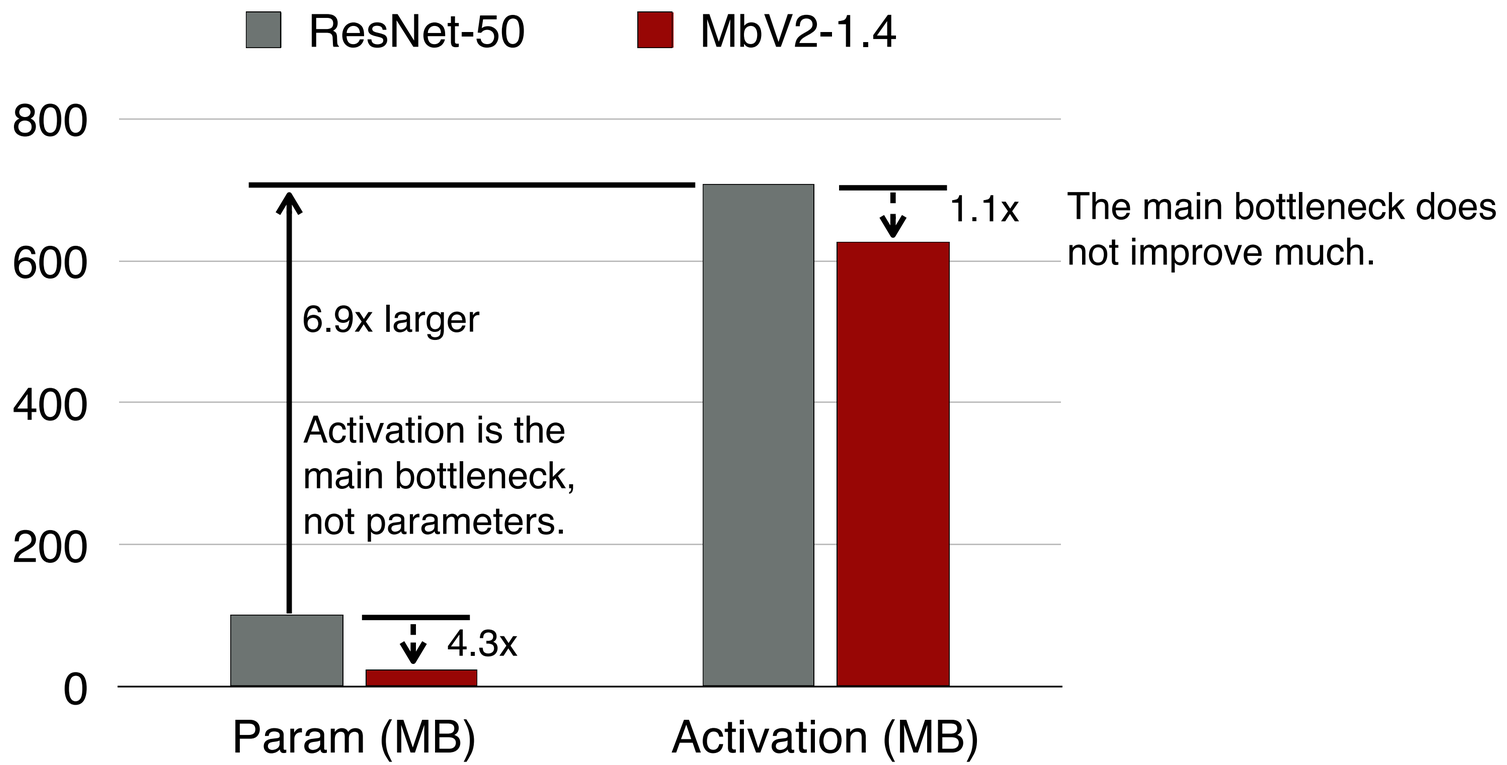
- Edge devices have tight memory constraints. The training memory footprint of neural networks can easily exceed the limit.

Activation is the Memory Bottleneck, not Parameters



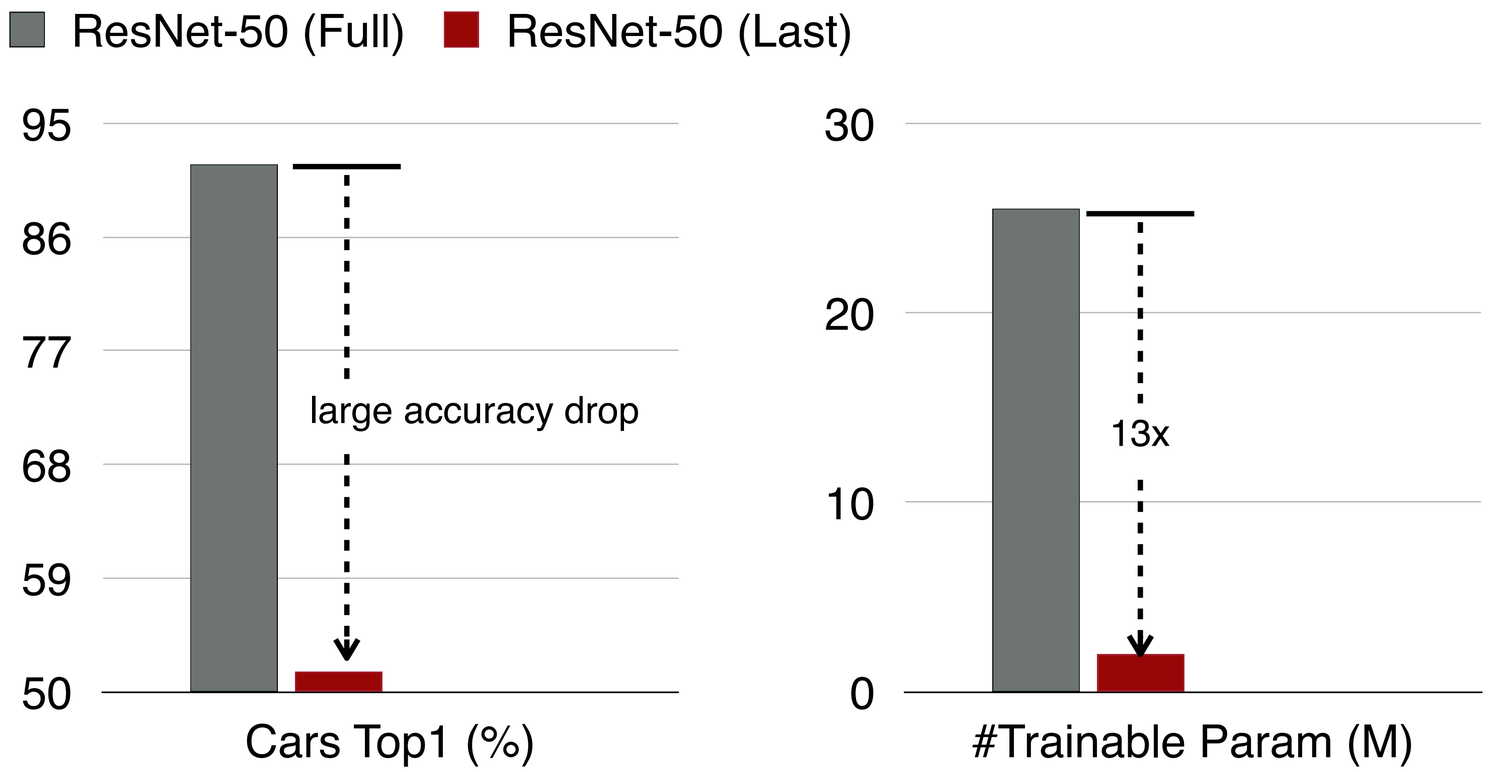
- Activation is the main bottleneck for on-device learning, not parameters.

Activation is the Memory Bottleneck, not Parameters



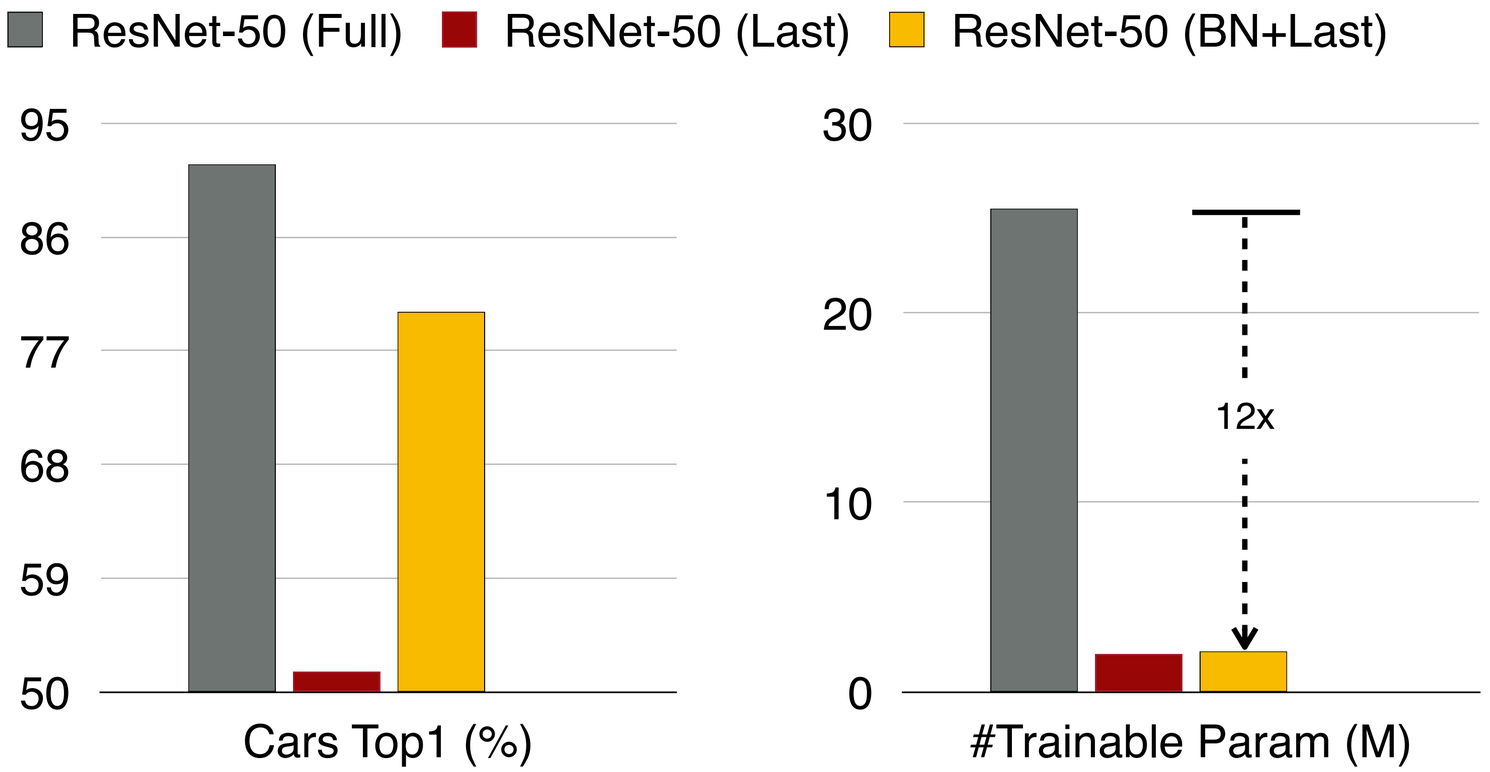
- Activation is the main bottleneck for on-device learning, not parameters.
- Previous methods focus on reducing the number of parameters or FLOPs, while the main bottleneck does not improve much.

Related Work: Parameter-Efficient Transfer Learning



- Full: Fine-tune the full network. Better accuracy but highly inefficient.
- Last: Only fine-tune the last classifier head. Efficient but the capacity is limited.

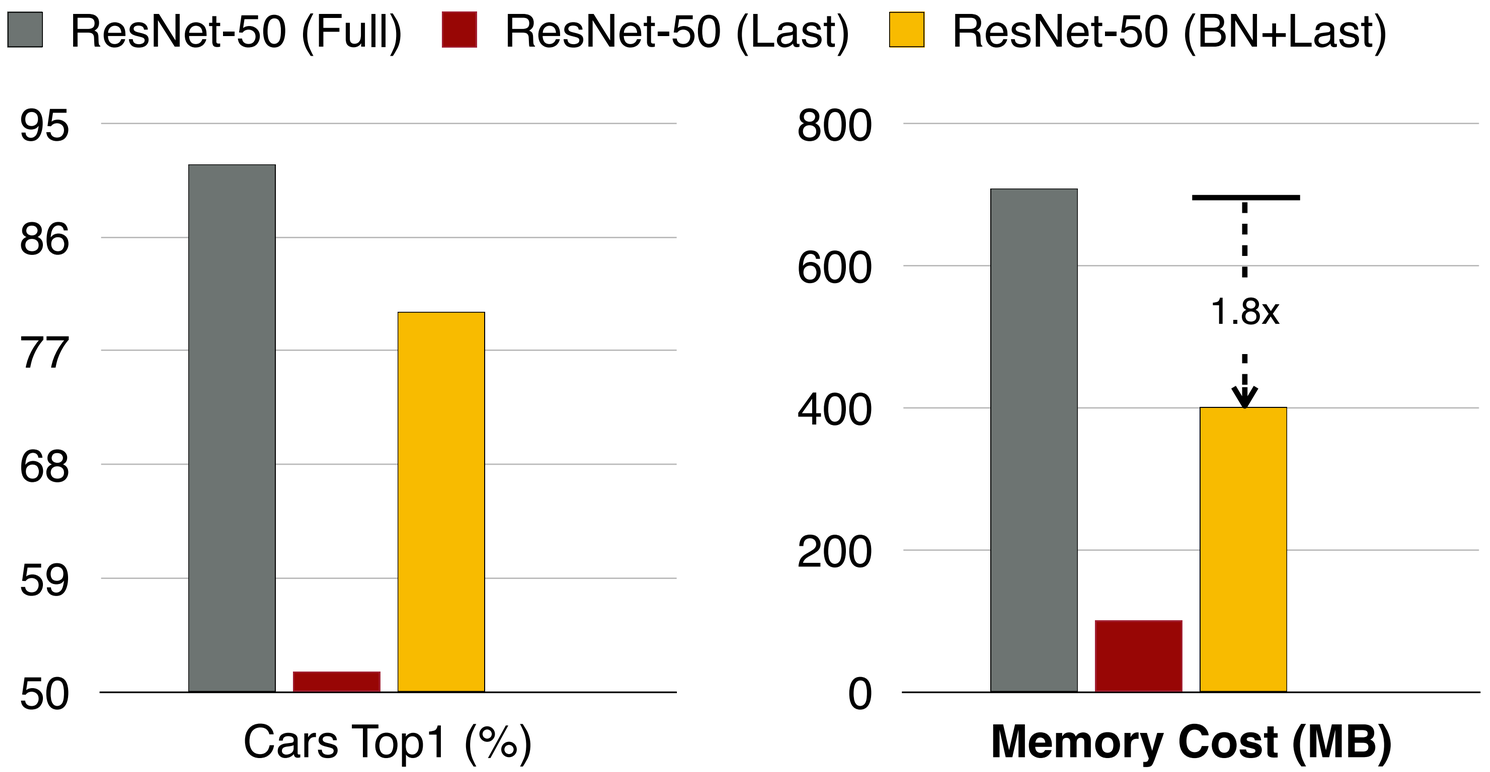
Related Work: Parameter-Efficient Transfer Learning



- Full: Fine-tune the full network. Better accuracy but highly inefficient.
- Last: Only fine-tune the last classifier head. Efficient but the capacity is limited.
- BN+Last: Fine-tune the BN layers and the last layer. Parameter-efficient.

- Mudrakarta, Pramod Kaushik, et al. "K for the Price of 1: Parameter-efficient Multi-task and Transfer Learning." *ICLR 2019.*

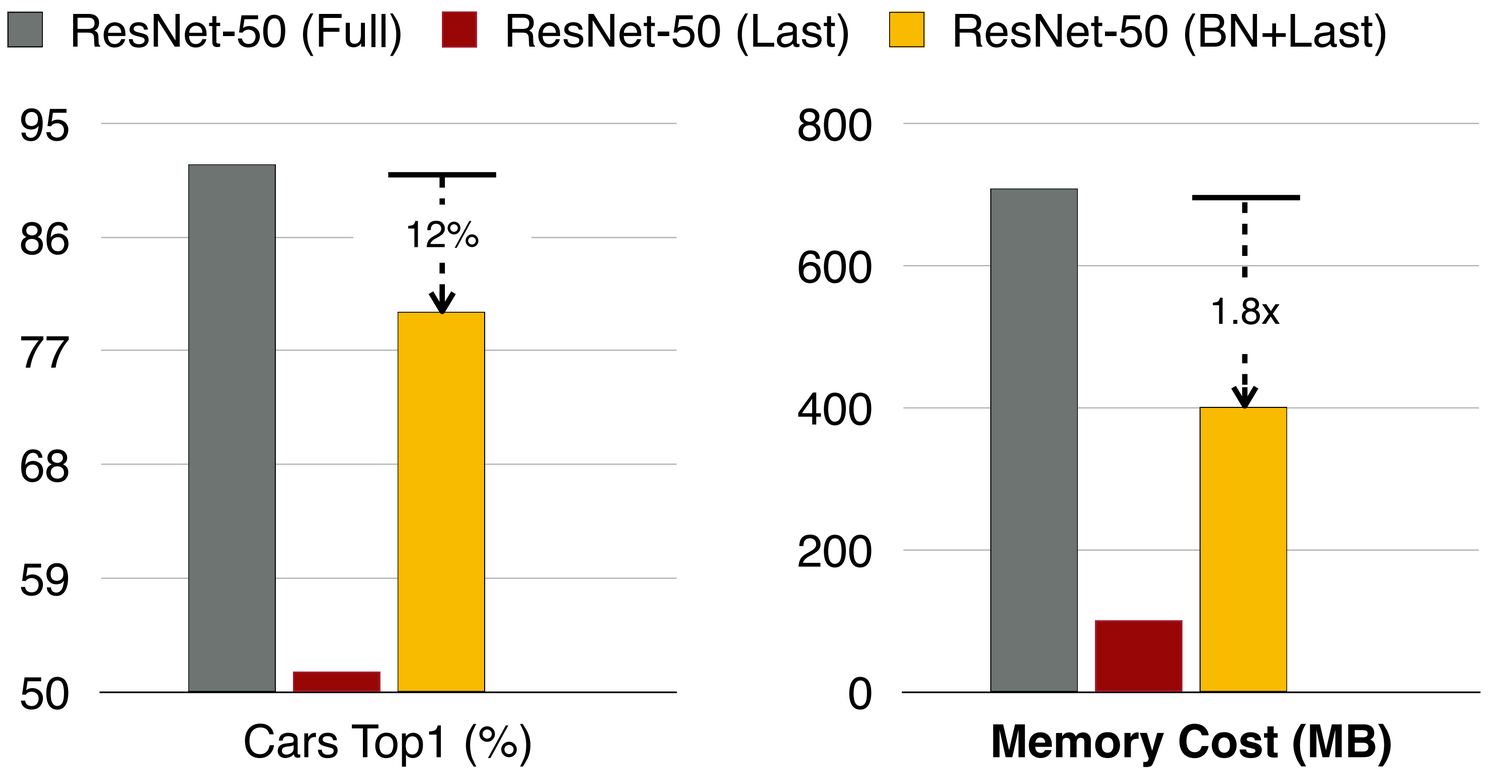
Related Work: Parameter-Efficient Transfer Learning



Parameter-efficiency does not directly translate to memory-efficiency

- Full: Fine-tune the full network. Better accuracy but highly inefficient.
- Last: Only fine-tune the last classifier head. Efficient but the capacity is limited.
- BN+Last: Fine-tune the BN layers and the last layer. Parameter-efficient, **but the memory saving is limited.**

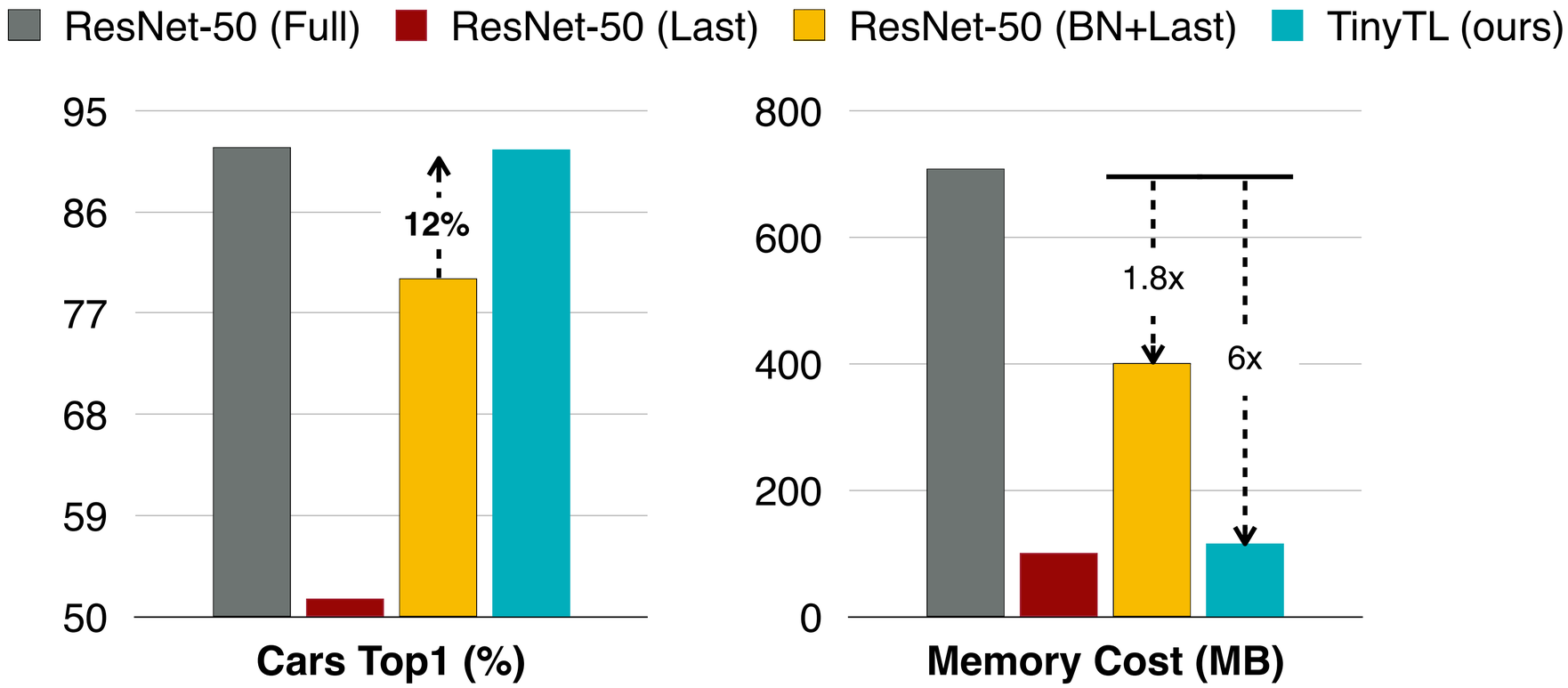
Related Work: Parameter-Efficient Transfer Learning



Parameter-efficiency does not directly translate to memory-efficiency

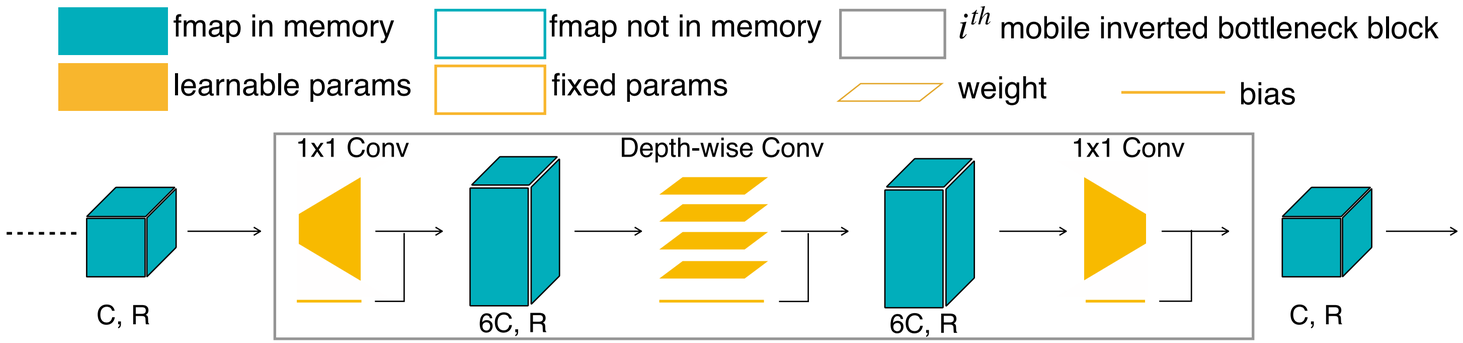
- Full: Fine-tune the full network. Better accuracy but highly inefficient.
- Last: Only fine-tune the last classifier head. Efficient but the capacity is limited.
- BN+Last: Fine-tune the BN layers and the last layer. Parameter-efficient, **but the memory saving is limited. Significant accuracy loss.**

TinyTL: Memory-Efficient Transfer Learning



- Full: Fine-tune the full network. Better accuracy but highly inefficient.
- Last: Only fine-tune the last classifier head. Efficient but the capacity is limited.
- BN+Last: Fine-tune the BN layers and the last layer. Parameter-efficient, **but the memory saving is limited. Significant accuracy loss.**
- TinyTL: fine-tune bias only + lite residual learning: high accuracy, large memory saving

Weight update is Memory-expensive; Bias update is Memory-efficient



Fine-tune the full network (Conventional)

Forward: $\mathbf{a}_{i+1} = \mathbf{a}_i \mathbf{W}_i + \mathbf{b}_i$

Backward: $\frac{\partial L}{\partial \mathbf{W}_i} = \mathbf{a}_i^T \frac{\partial L}{\partial \mathbf{a}_{i+1}}, \qquad \frac{\partial L}{\partial \mathbf{b}_i} = \frac{\partial L}{\partial \mathbf{a}_{i+1}} = \frac{\partial L}{\partial \mathbf{a}_{i+2}} \mathbf{W}_{i+1}^T$

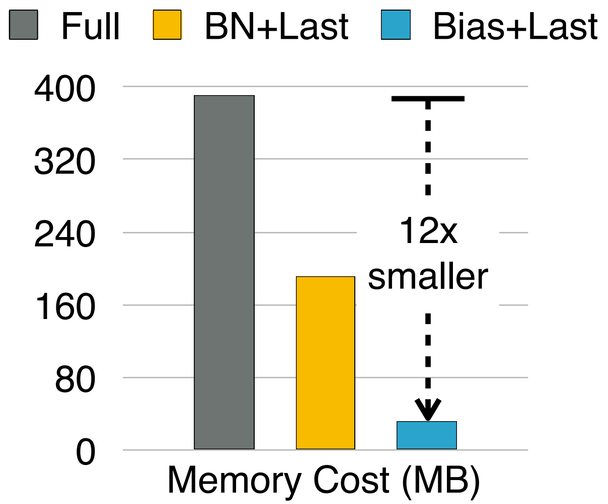
- Updating weights requires storing intermediate activations
- Updating biases does not

TinyTL Idea 1: Fine-tune Bias Only

y=Wa+b

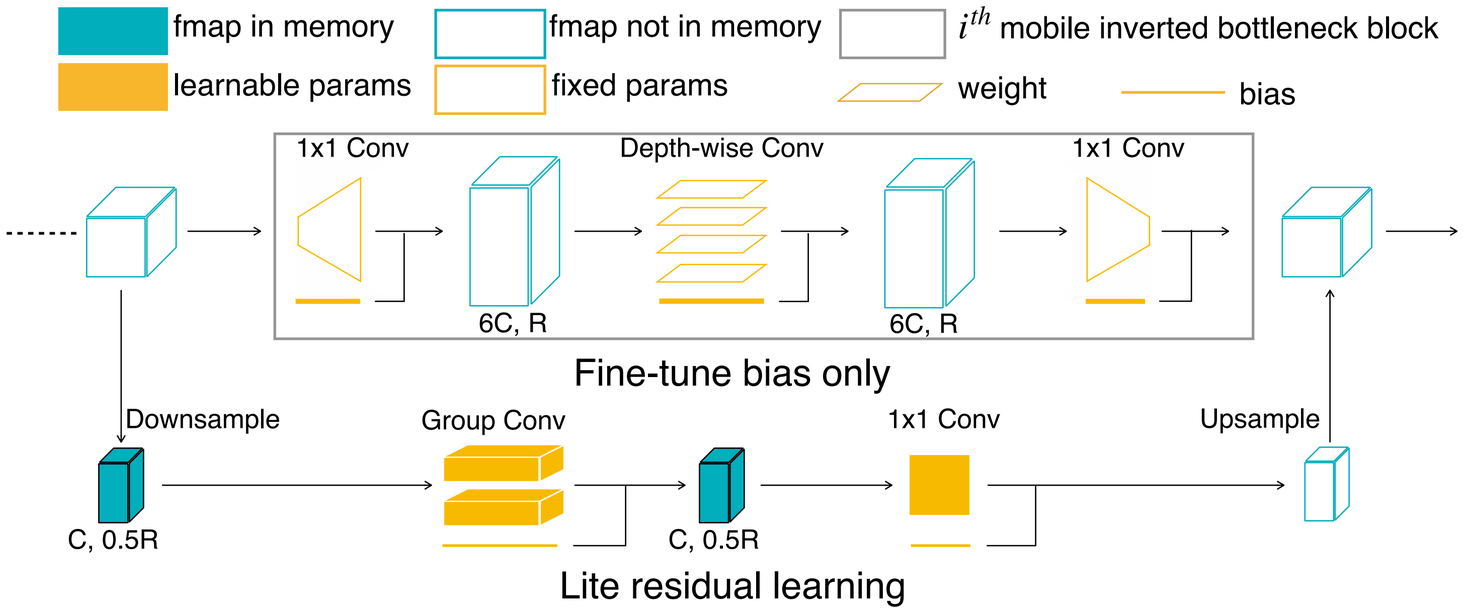
Backward: $\frac{\partial L}{\partial \mathbf{W}_i} = \mathbf{a}_i^T \frac{\partial L}{\partial \mathbf{a}_{i+1}}, \qquad \frac{\partial L}{\partial \mathbf{b}_i} = \frac{\partial L}{\partial \mathbf{a}_{i+1}} = \frac{\partial L}{\partial \mathbf{a}_{i+2}} \mathbf{W}_{i+1}^T$

- Updating weights requires storing intermediate activations
- Updating biases does not



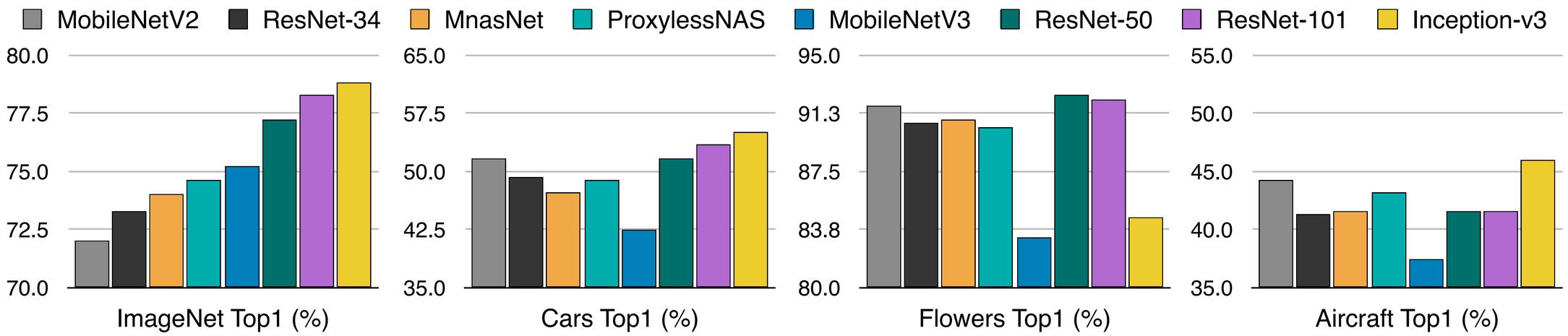
Freeze weights, only fine-tune biases => save 12x memory

TinyTL Idea 2: Lite Residual Learning



- Add lite residual modules (small memory overhead) to increase model capacity
  - (1/6 channel, 1/2 resolution, 2/3 depth)

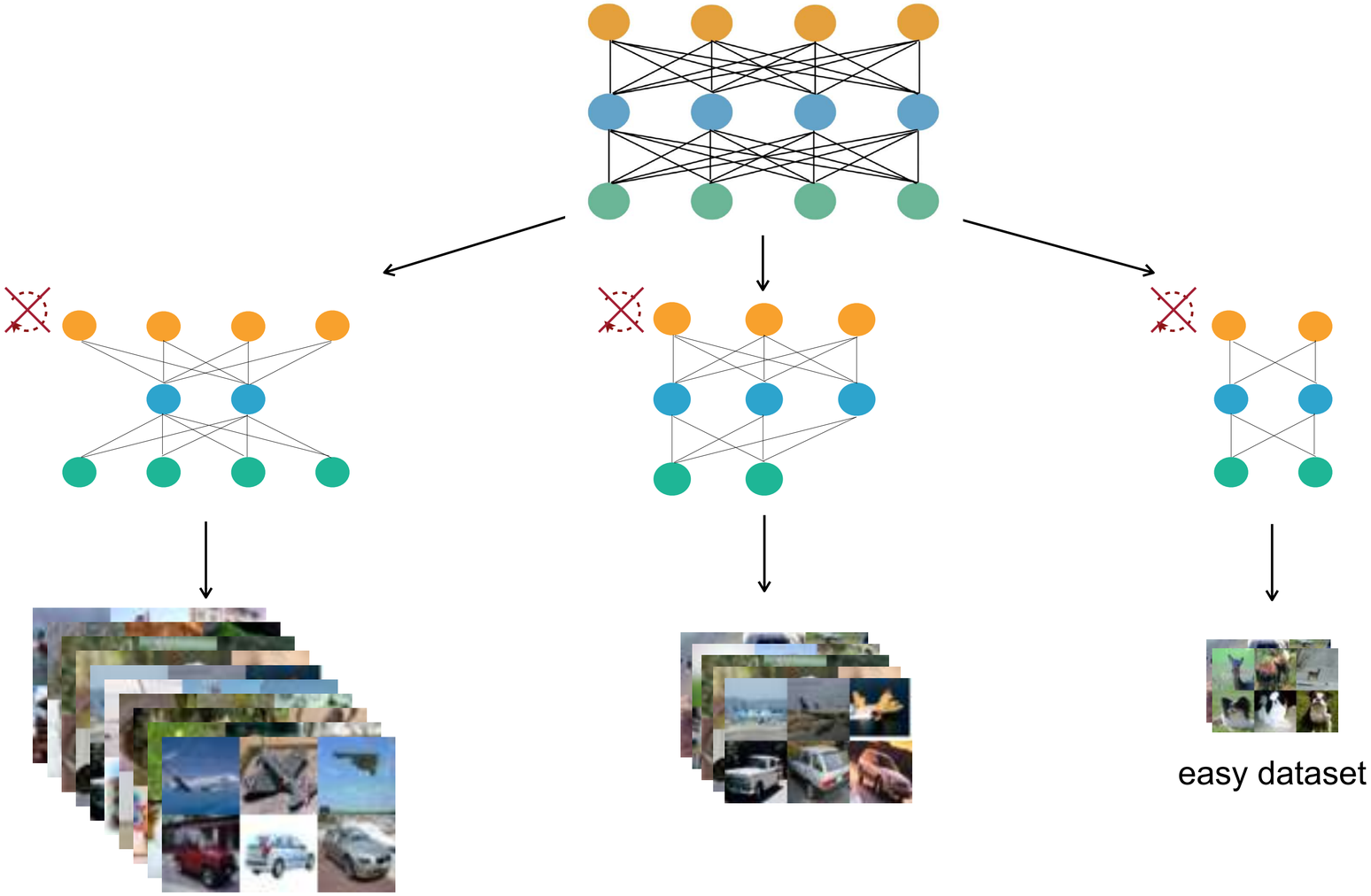
TinyTL Idea3: Specialized Models for Different Tasks



The relative accuracy order between different pre-trained models changes **significantly** among ImageNet and the transfer learning datasets, which motivates **personalized and specialized NN architecture for different downstream tasks**.

TinyTL + Once-for-All Network

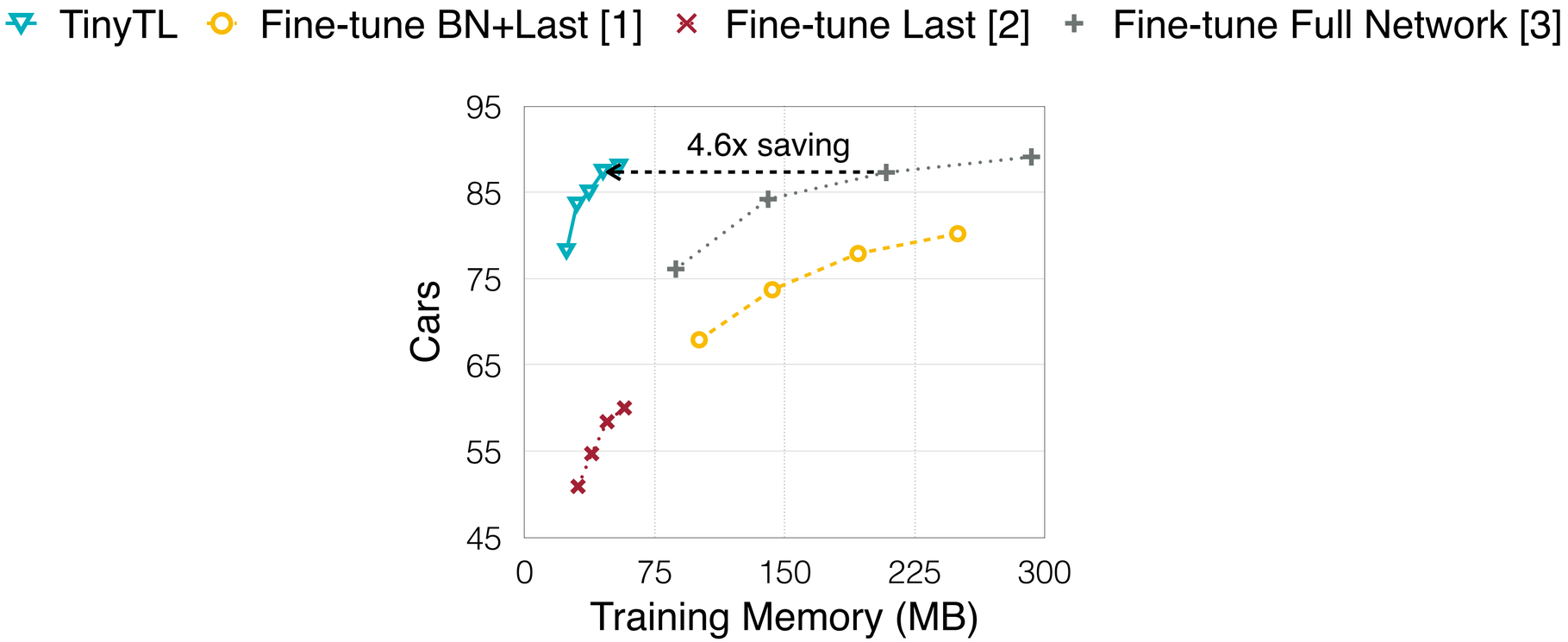
personalized model for different hardware and different tasks



difficult dataset

easy dataset

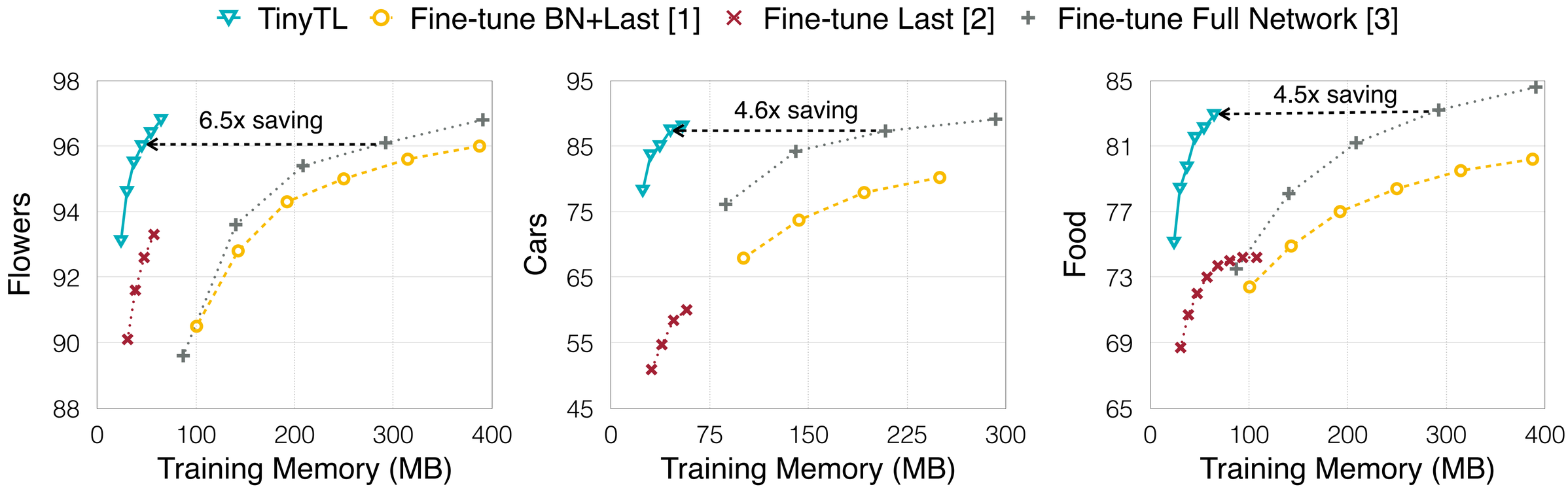
Memory Saving



- TinyTL provides **4.6x** memory saving **without accuracy loss**.

- [1] Chatfield, Ken, et al. "Return of the devil in the details: Delving deep into convolutional nets." *BMVC 2014*.
- [2] Mudrakarta, Pramod Kaushik, et al. "K for the Price of 1: Parameter-efficient Multi-task and Transfer Learning." *ICLR 2019.*
- [3] Kornblith, Simon, Jonathon Shlens, and Quoc V. Le. "Do better imagenet models transfer better?." *CVPR* 2019.

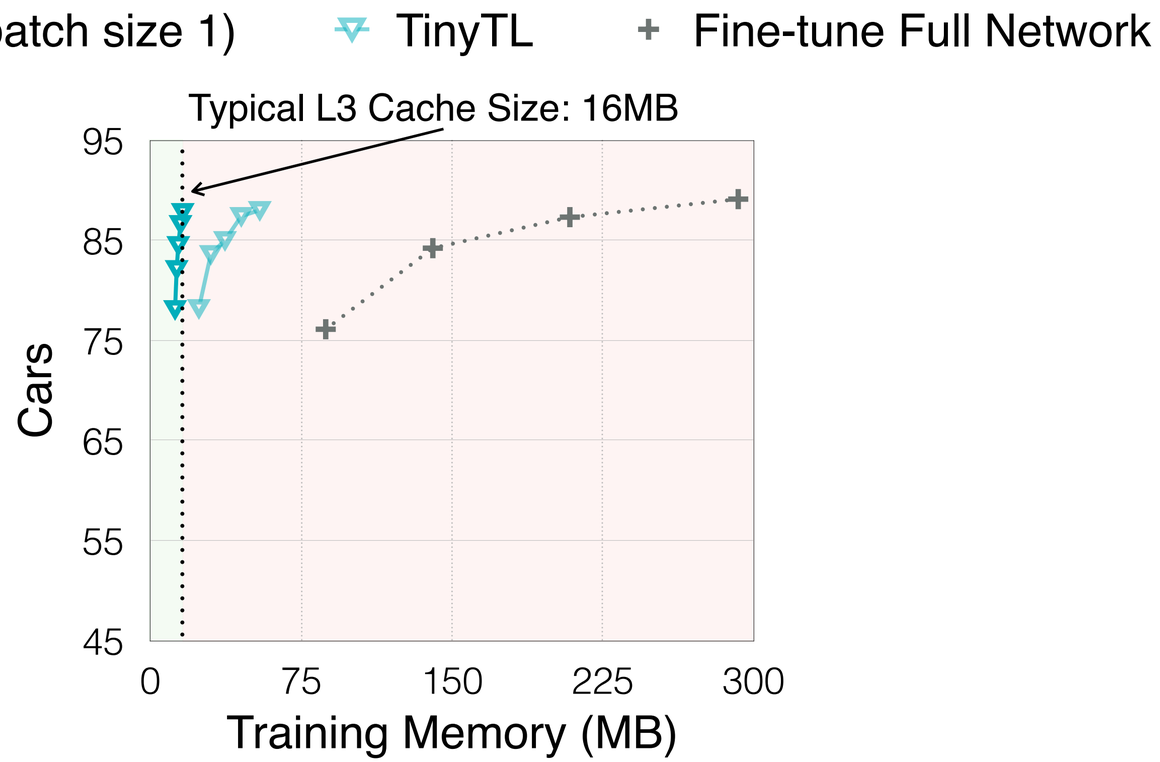
Memory Saving



- On different datasets, TinyTL provides up to **6.5x** memory saving **without accuracy loss**.

- [1] Chatfield, Ken, et al. "Return of the devil in the details: Delving deep into convolutional nets." *BMVC 2014*.
- [2] Mudrakarta, Pramod Kaushik, et al. "K for the Price of 1: Parameter-efficient Multi-task and Transfer Learning." *ICLR 2019.*
- [3] Kornblith, Simon, Jonathon Shlens, and Quoc V. Le. "Do better imagenet models transfer better?." *CVPR* 2019.

TinyTL enables in-memory training



- TinyTL (tiny transfer learning) supports batch 1 training by **group normalization**.
- Together with the lite residual model, it further reduces the training memory cost to 16MB (fits L3 cache), enabling fitting the training process into cache, which is much more energy-efficient than training on DRAM.

TinyML and Efficient Deep Learning

- **AutoML and NAS**
  - Once-for-all Network [ICLR'19]

- **TinyML**
  - Inference: MCUNet for IoT Devices [NeurIPS'20, spotlight]
  - Training: Tiny On-Device Transfer Learning (TinyTL) [NeurIPS'20]

- **Data-Efficiency**
  - Differentiable Augmentation for Data-Efficient GAN Training [NeurIPS'20]

Data Is Expensive

FFHQ dataset: **70,000** selective post-processed human faces

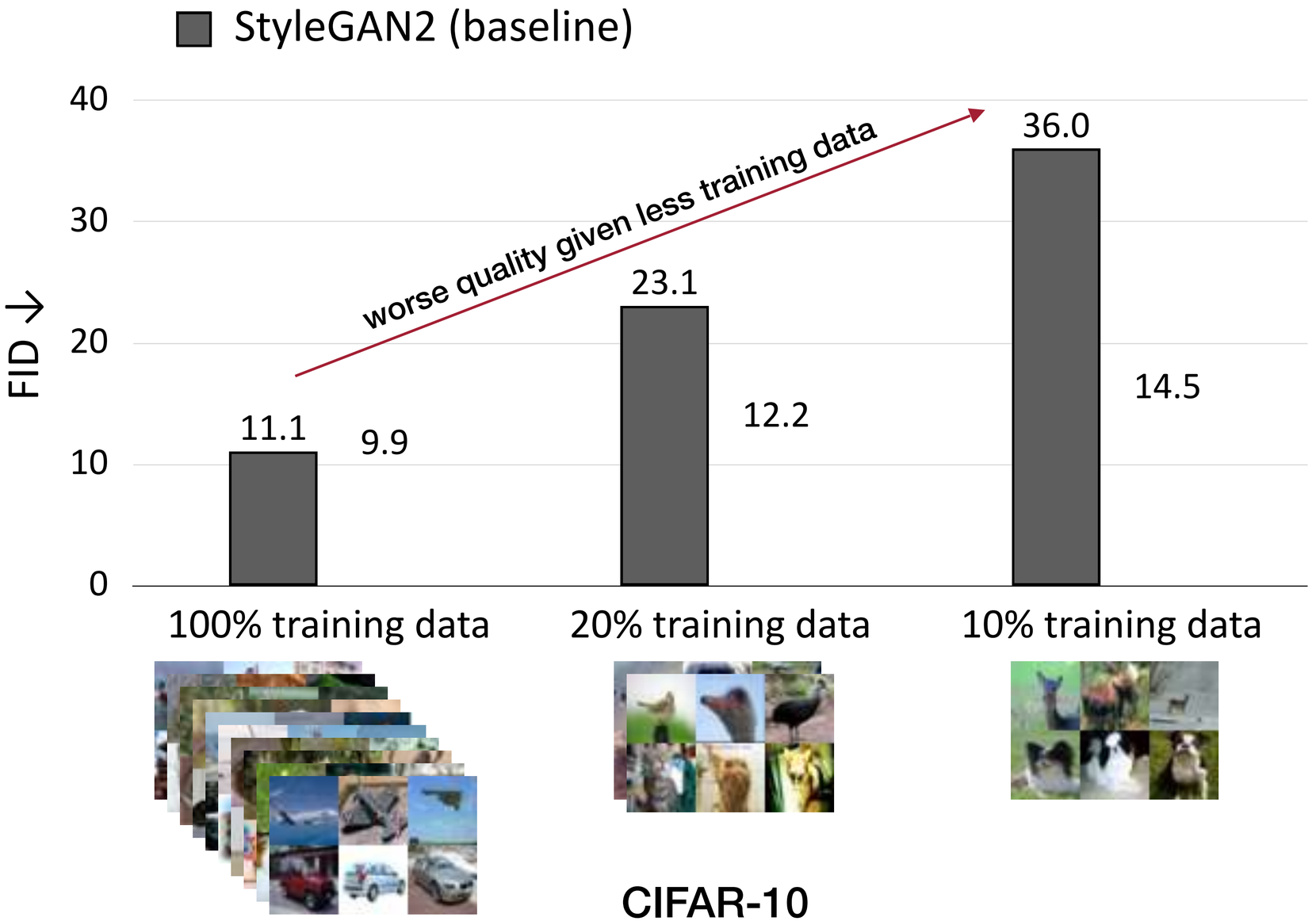
ImageNet dataset: **millions** of images from diverse categories

Months or even years to collect the data,
along with *prohibitive* annotation costs.

GANs Heavily Deteriorate Given Limited Data

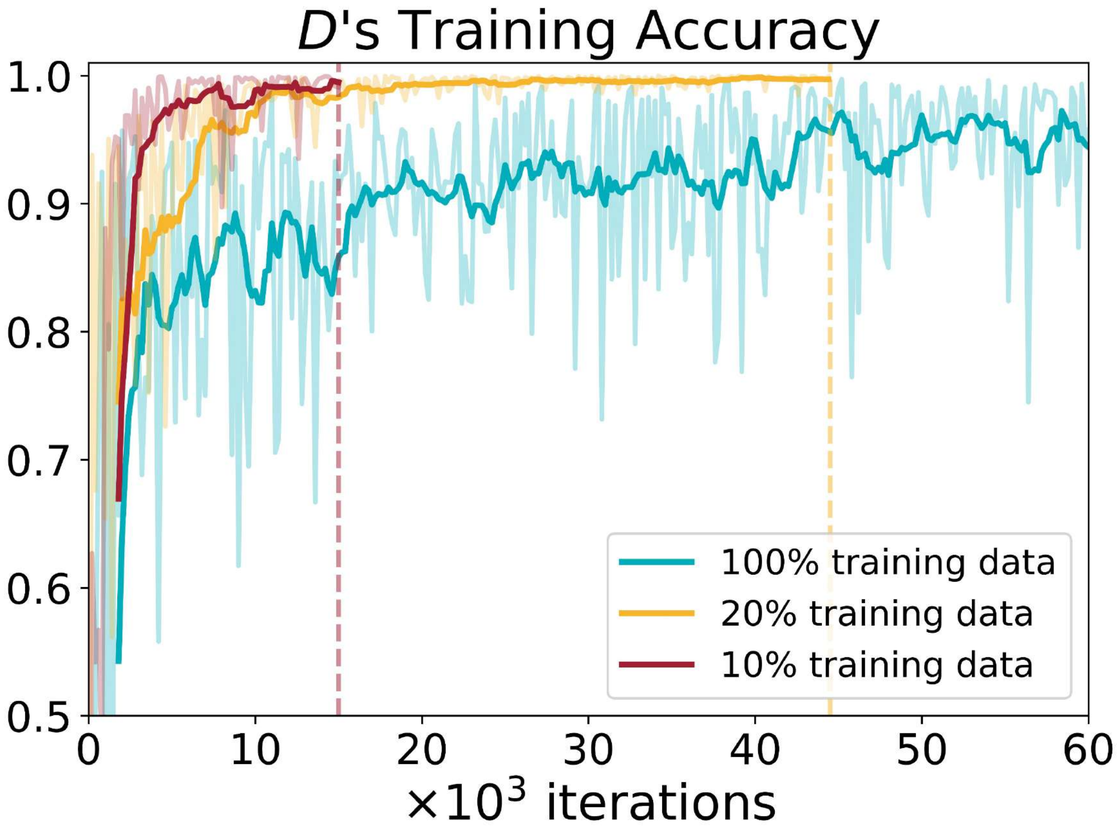
Generated samples of StyleGAN2 (Karras et al.)
using only hundreds of images

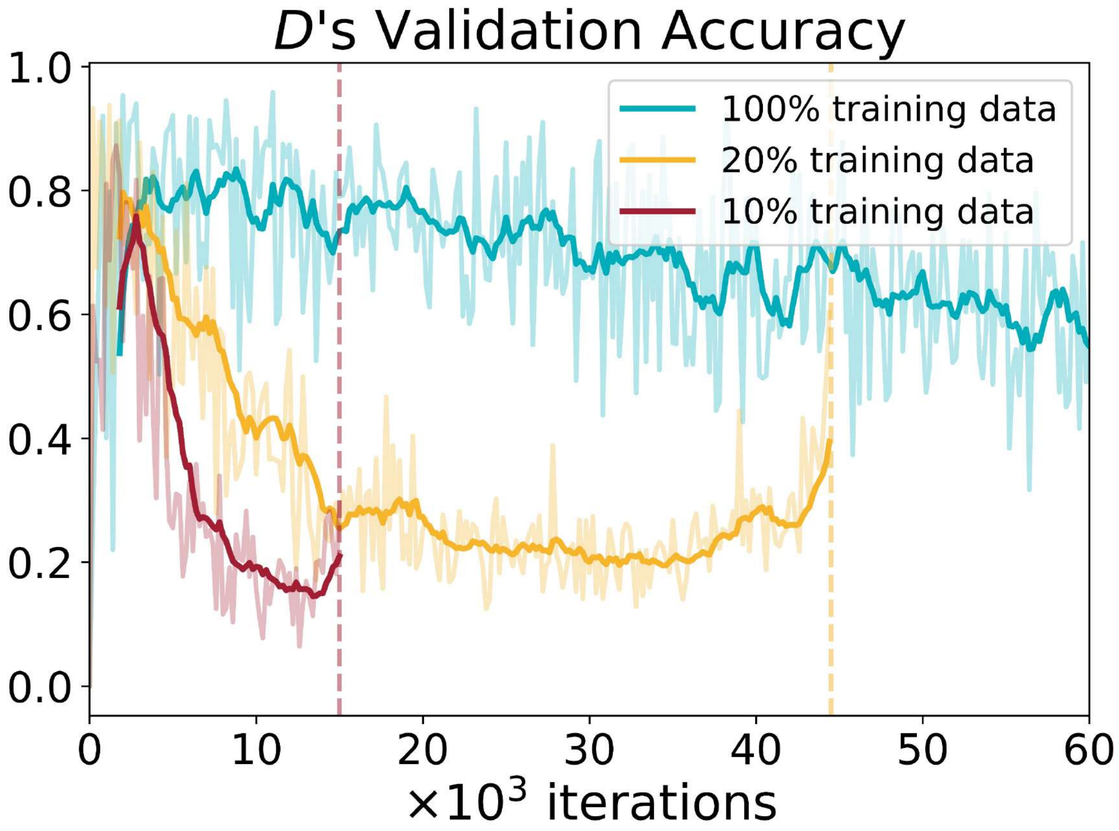
GANs Heavily Deteriorate Given Limited Data



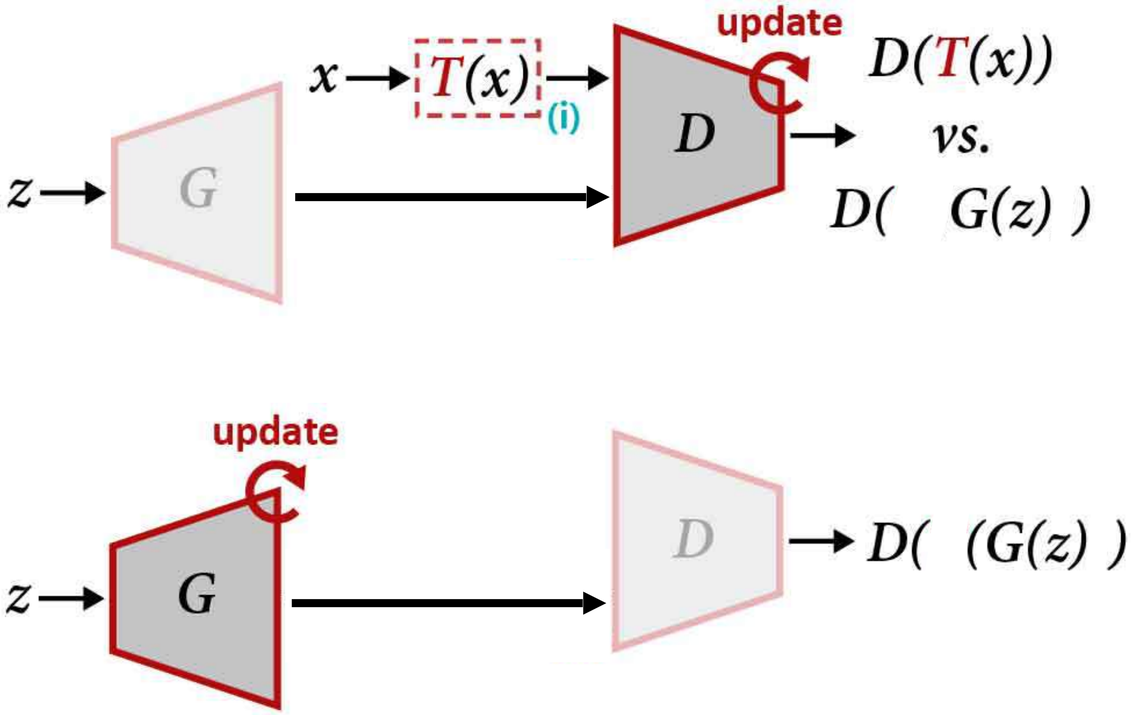
CIFAR-10

Discriminator Overfitting

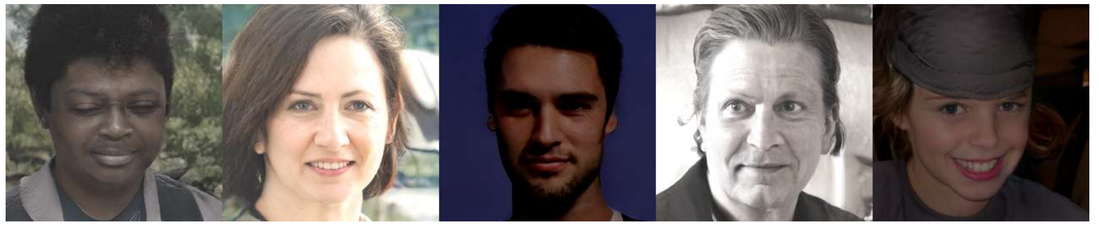




#1 Approach: Augment reals only



Generated images



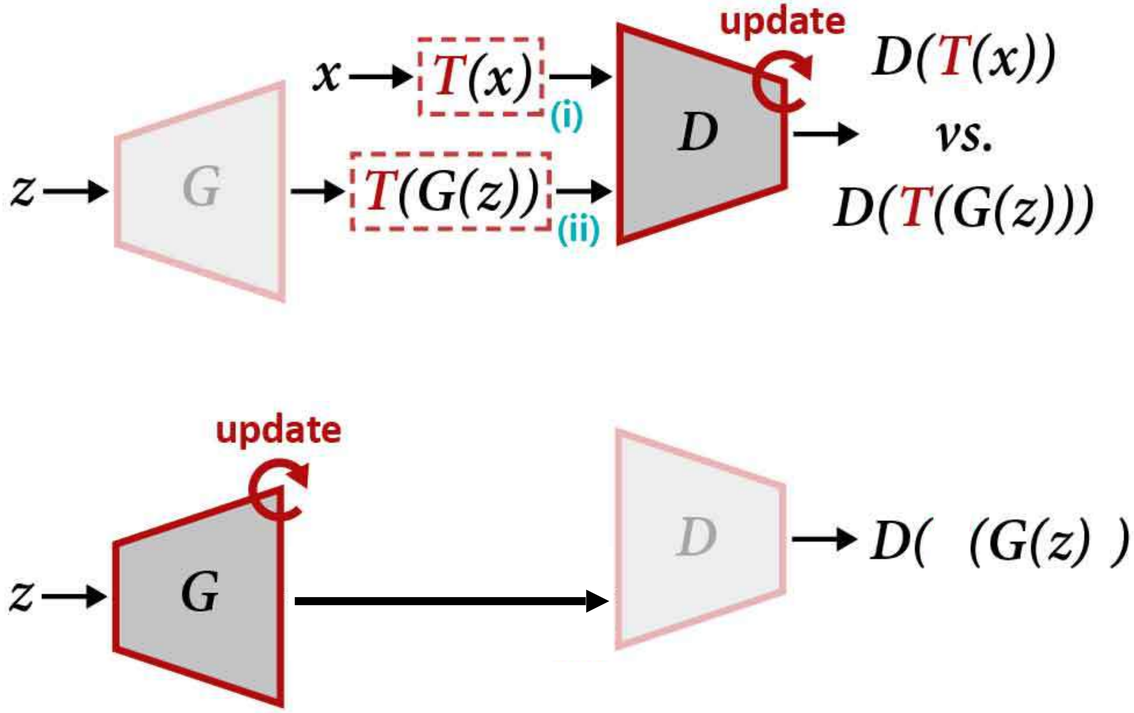
Artifacts from Color jittering

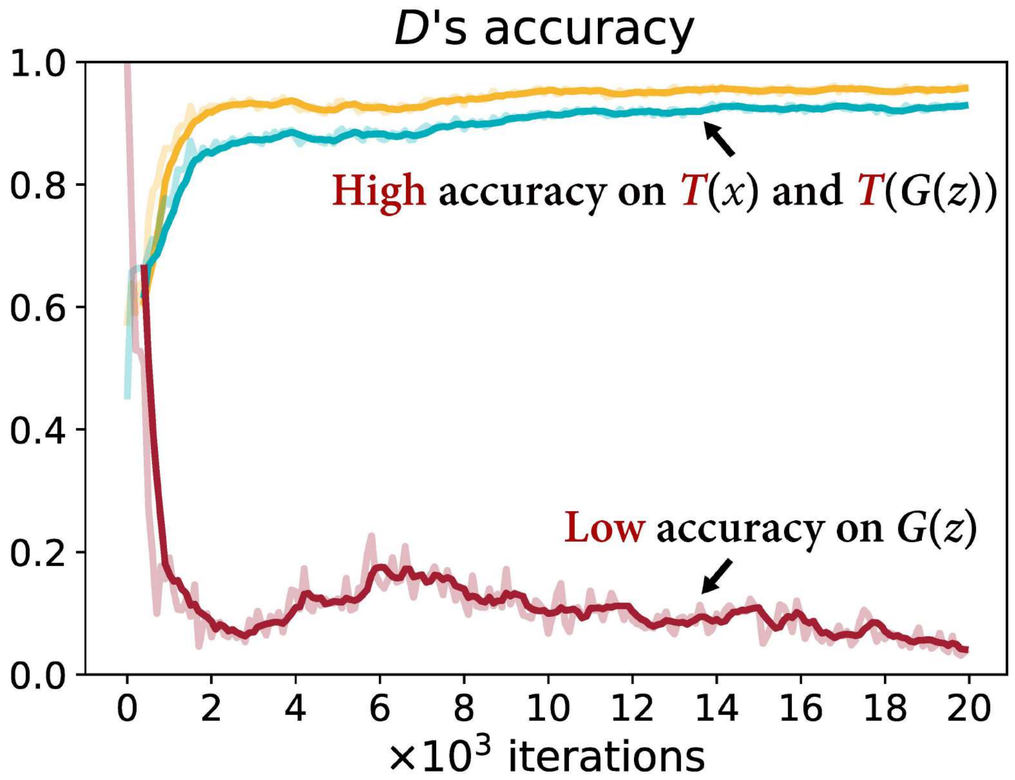
Artifacts from Translation

Artifacts from Cutout (DeVries et al.)

Augment reals only: the same artifacts appear on the generated images.

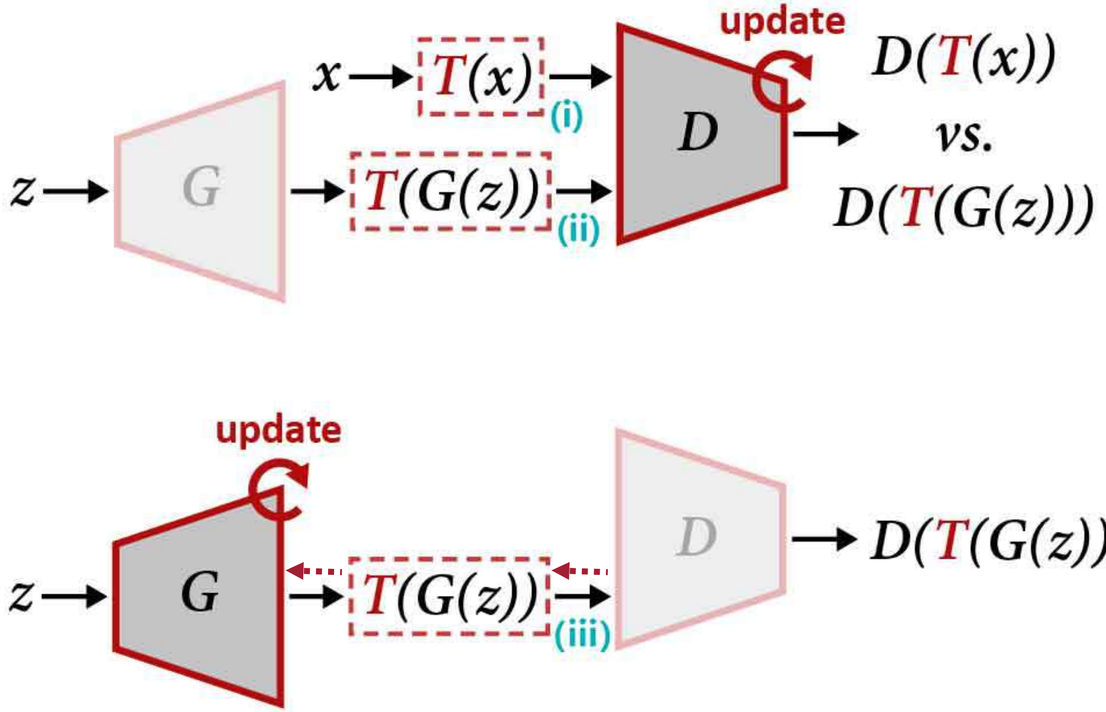
#2 Approach: Augment reals & fakes for D only

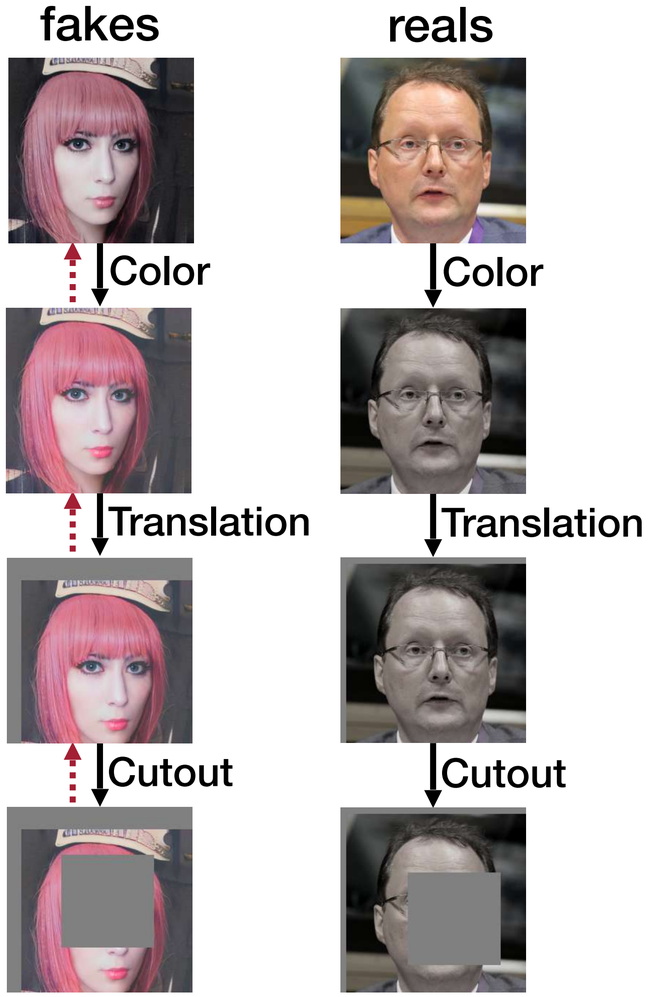




Augment D only: the unbalanced optimization cripples training.

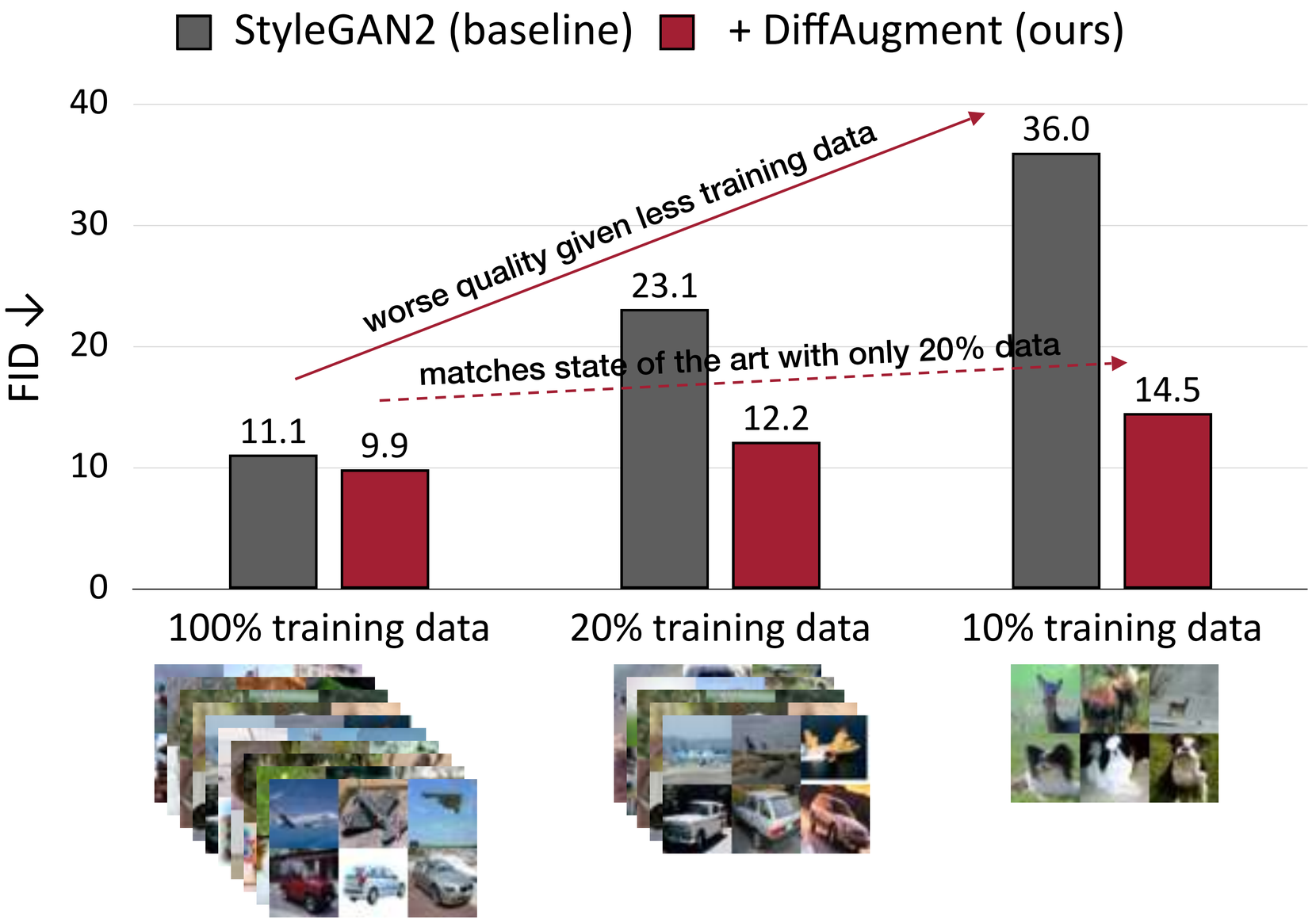
#3 Approach: Differentiable Augmentation (Ours)





Our approach (DiffAugment): Augment reals + fakes for both D and G

Our Results


StyleGAN2 (baseline)
+ DiffAugment (ours)
FID ↓
40
30
20
10
0
worse quality given less training data
matches state of the art with only 20% data
11.1
9.9
23.1
12.2
36.0
14.5
100% training data
20% training data
10% training data

Train GAN with only 100 Images

Without our technique:

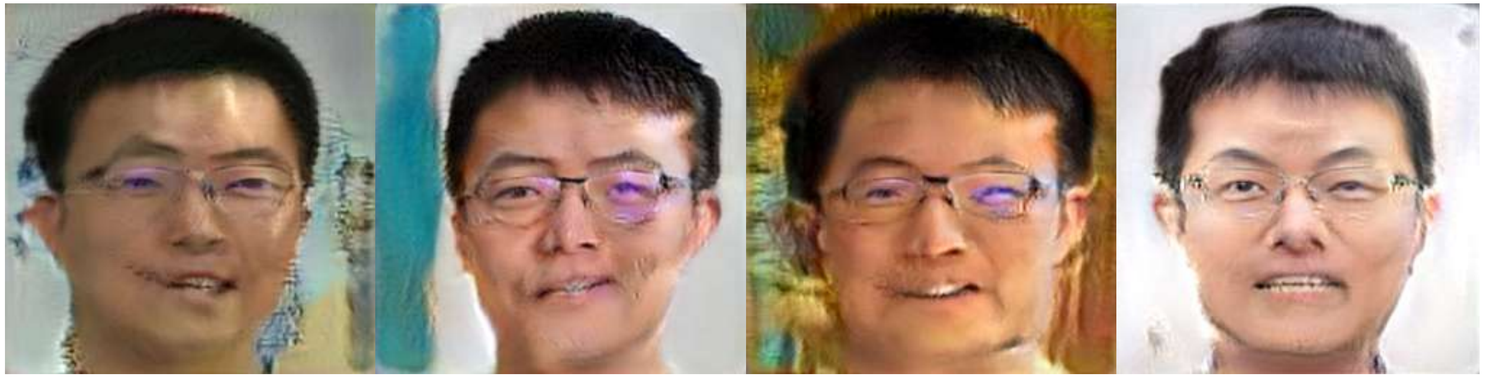
Generated samples of StyleGAN2 **(baseline)**

With our technique:

Generated samples of StyleGAN2 + DiffAugment **(ours)**

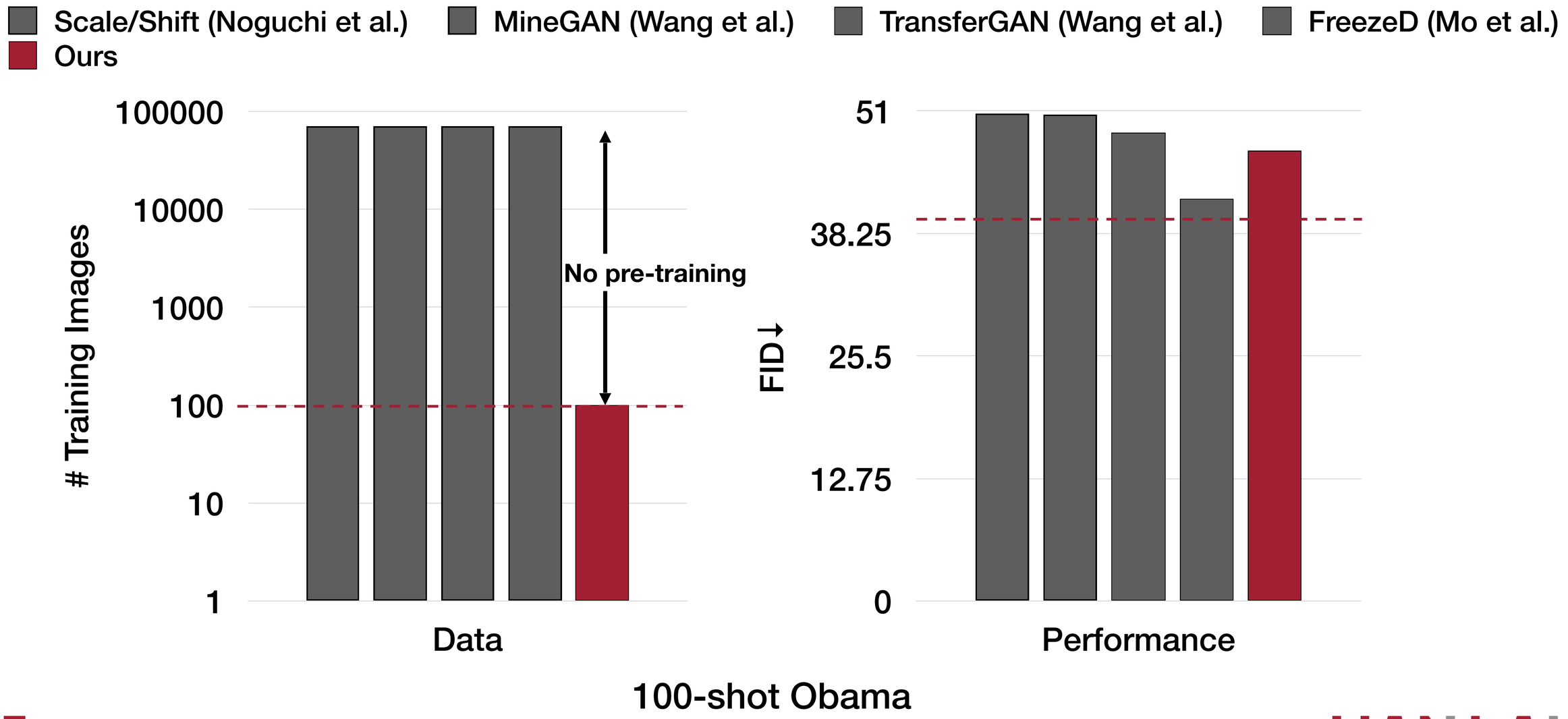
Train GAN with only 100 Images

Without our technique:



With our technique:

Fine-Tuning vs. Ours



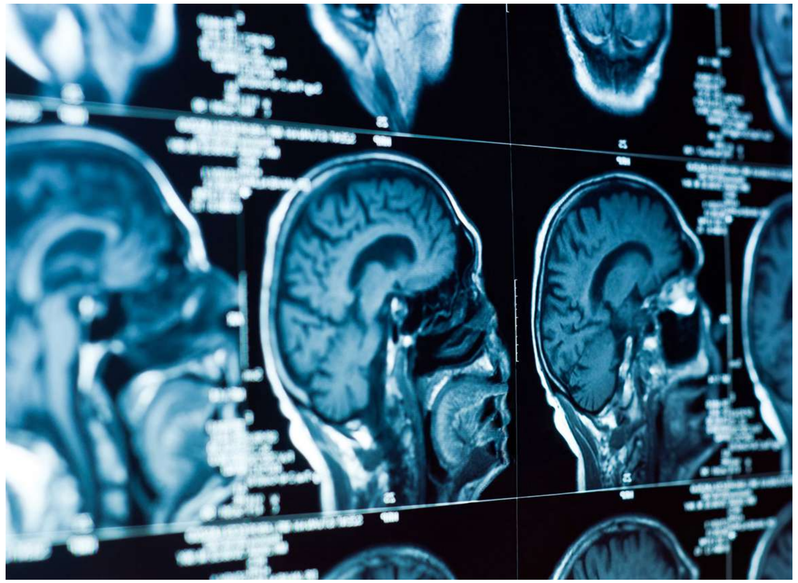
Train GAN with only 100 Images

Smooth interpolation, generalize well

https://github.com/mit-han-lab/data-efficient-gans

Data-Efficient Deep Learning

Rare incidents



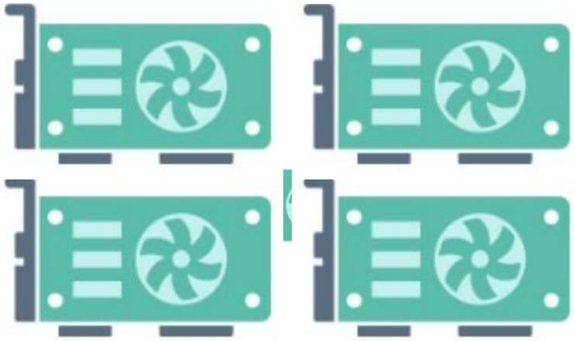
Privacy concerns

Under-represented subpopulations

Various factual and ethical reasons could cause limited data available.
This research will help alleviate these limitations.

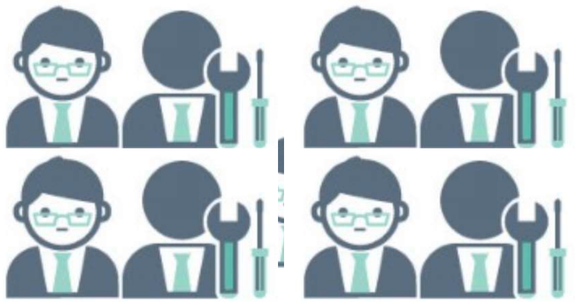
TinyML and Efficient Deep Learning

Three aspects: computation, engineers, data

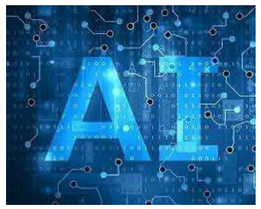


A lot of computation

A lot of carbon



Many engineers



A lot of data

TinyML and Efficient Deep Learning

- **AutoML and NAS**
  - Once-for-all Network [ICLR'19]

- **TinyML**
  - Inference: MCUNet for IoT Devices [NeurIPS'20, spotlight]
  - Training: Tiny On-Device Transfer Learning (TinyTL) [NeurIPS'20]

- **Data-Efficient**
  - Differentiable Augmentation for Data-Efficient GAN Training [NeurIPS'20]

TinyML and Efficient AI

MIT HAN Lab
Hardware, AI and Neural-nets

songhan.mit.edu
tinyml.mit.edu

youtube.com/c/MITHANLab