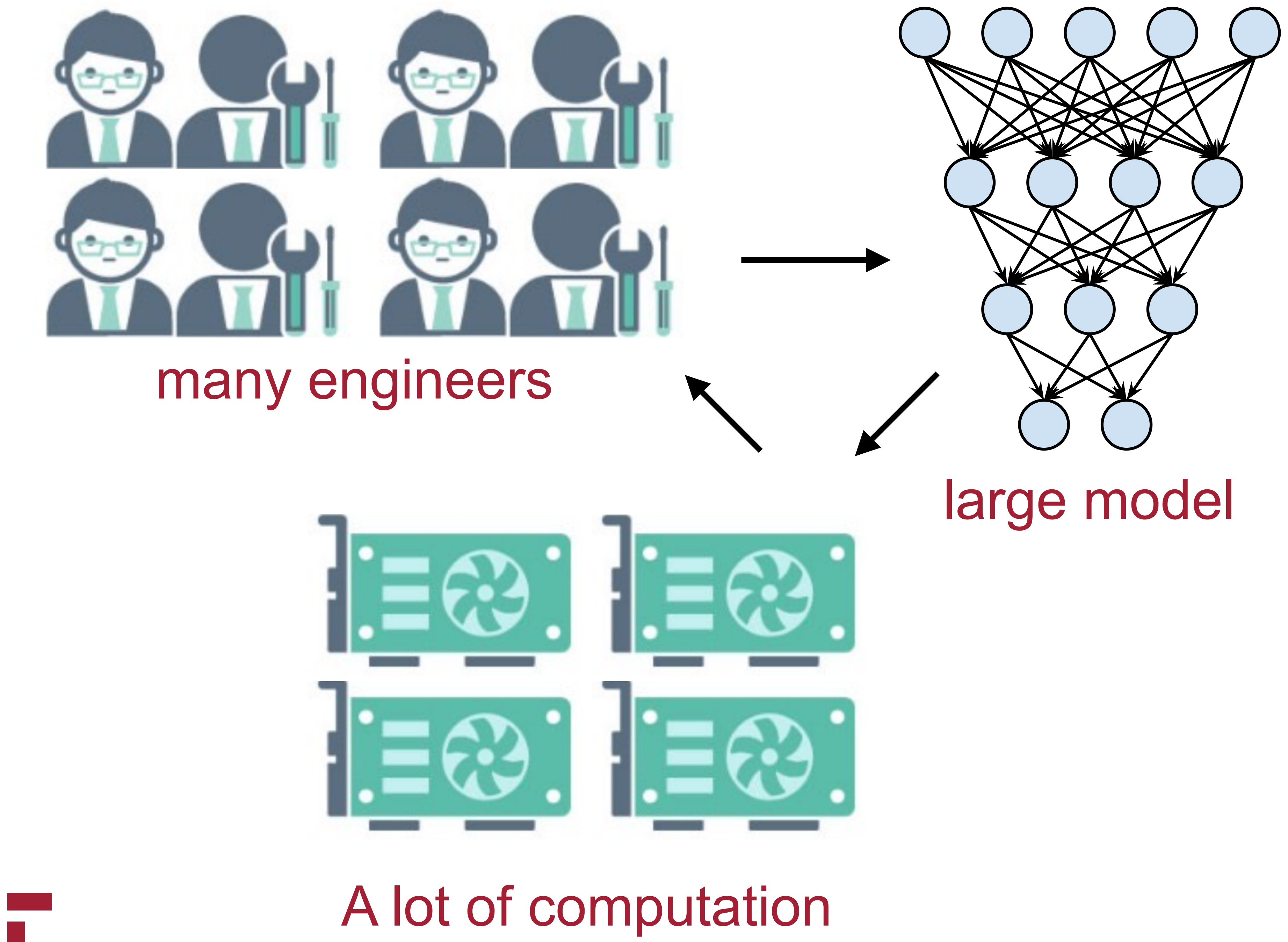
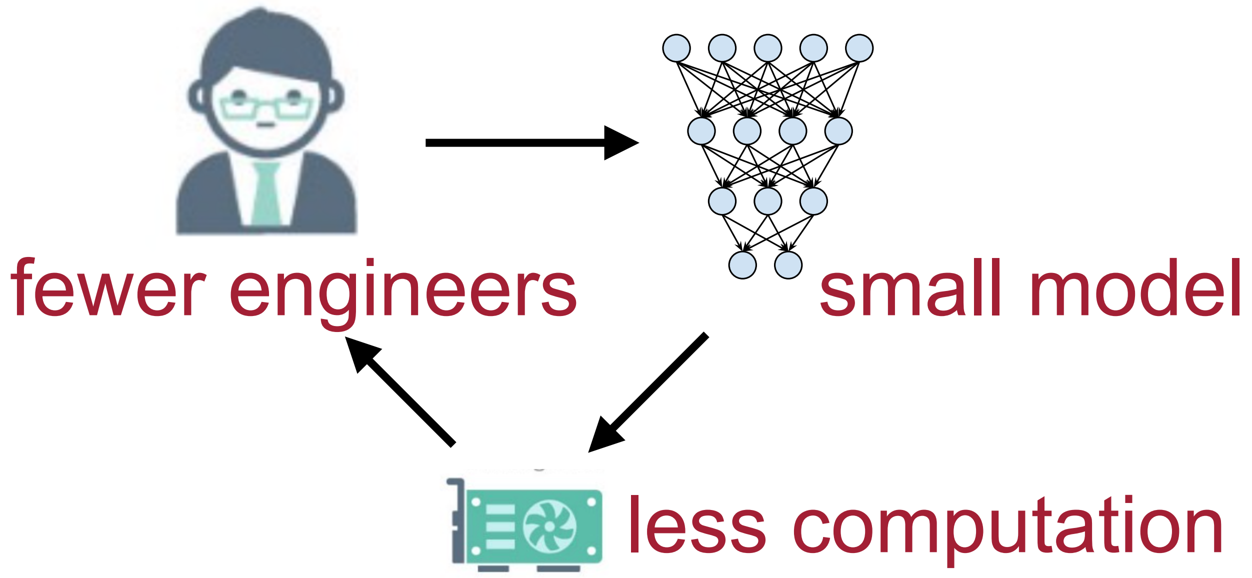
AutoML for TinyML with Once-for-All Network

Song Han
Massachusetts Institute of Technology

AutoML for TinyML
with Once-for-All Network

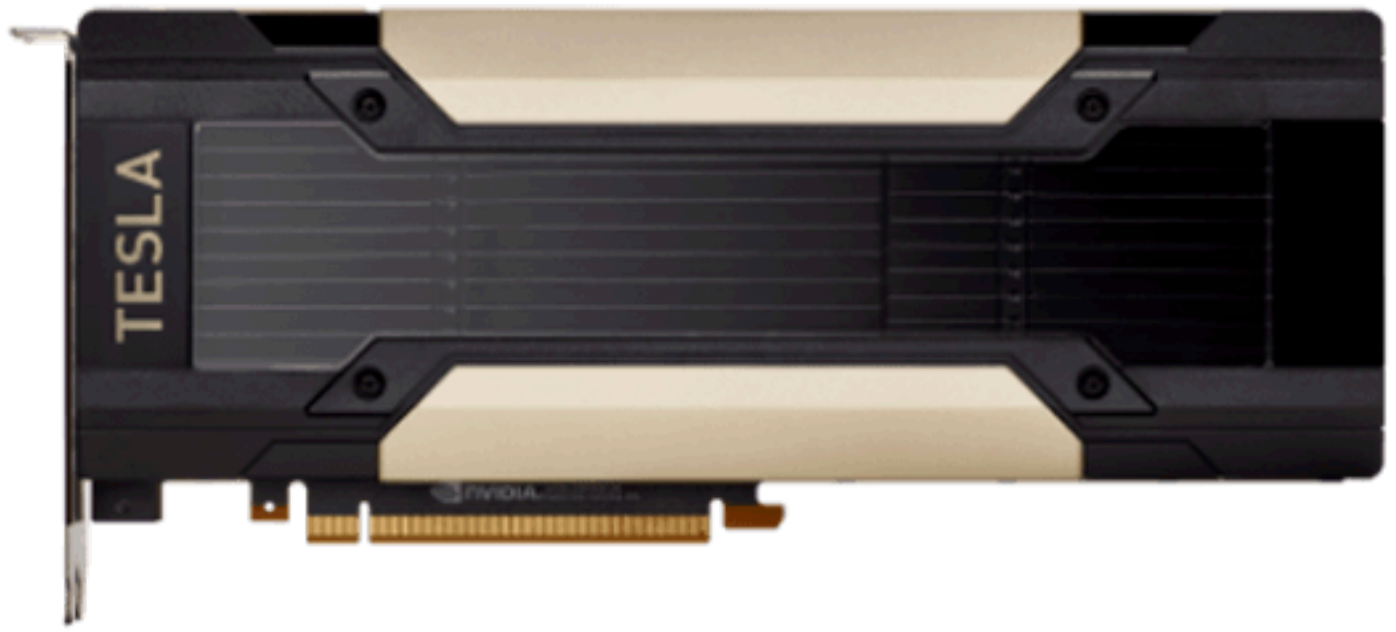




Less Engineer Resources: AutoML
Less Computational Resources: TinyML

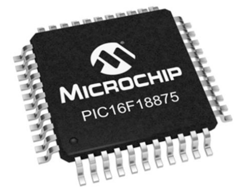
Challenge: Efficient Inference on Diverse Hardware Platforms

Cloud AI		Mobile AI		Tiny AI (AIoT)



less → resource

less → resource



Cloud AI
- Memory: 32GB
- Computation: TFLOPS/s

Mobile AI
- Memory: 4GB
- Computation: GFLOPS/s

Tiny AI (AIoT)
- **Memory: <100 KB**
- **Computation: <MFLOPS/s**

- Different hardware platforms have different resource constraints. We need to **customize** our models **for each platform** to achieve the best accuracy-efficiency trade-off, **especially on** resource-constrained **edge devices**.

Challenge: Efficient Inference on Diverse Hardware Platforms


```
for training iterations:
     forward-backward();
```

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

Challenge: Efficient Inference on Diverse Hardware Platforms

(1) for search episodes:

```
    for training iterations:
        forward-backward();
    if good_model: break;
for post-search training iterations:
        forward-backward();
```

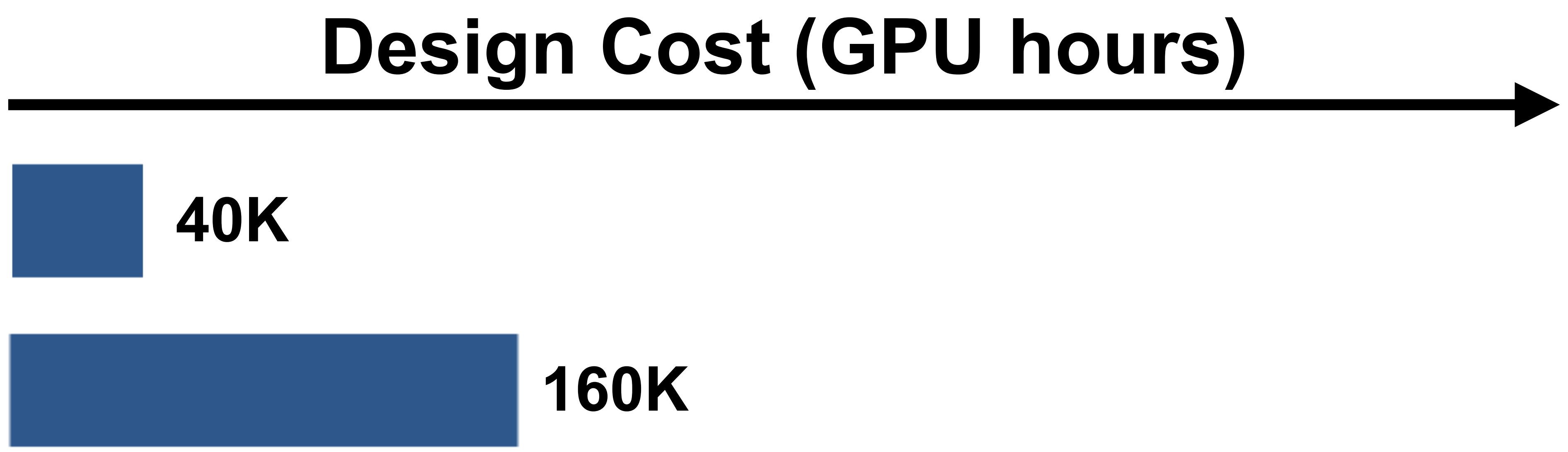
The design cost is calculated under the assumption of using MnasNet.

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

Challenge: Efficient Inference on Diverse Hardware Platforms

Diverse Hardware Platforms


```
(2) for devices:
  (1) for search episodes:
        for training iterations:
            forward-backward();
        if good_model: break;
      for post-search training iterations:
            forward-backward();
```

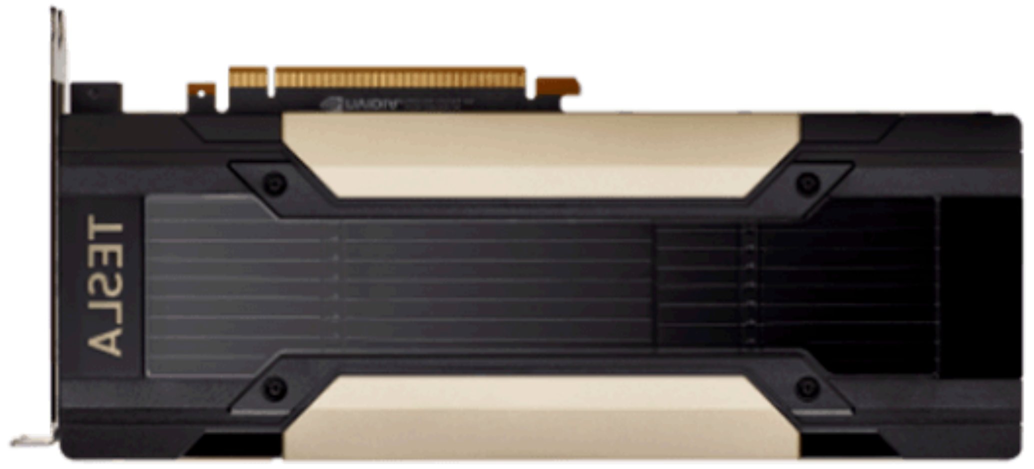


The design cost is calculated under the assumption of using MnasNet.

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

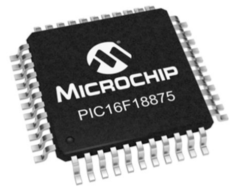
Challenge: Efficient Inference on Diverse Hardware Platforms

Diverse Hardware Platforms



Cloud AI (10^{12} FLOPS)

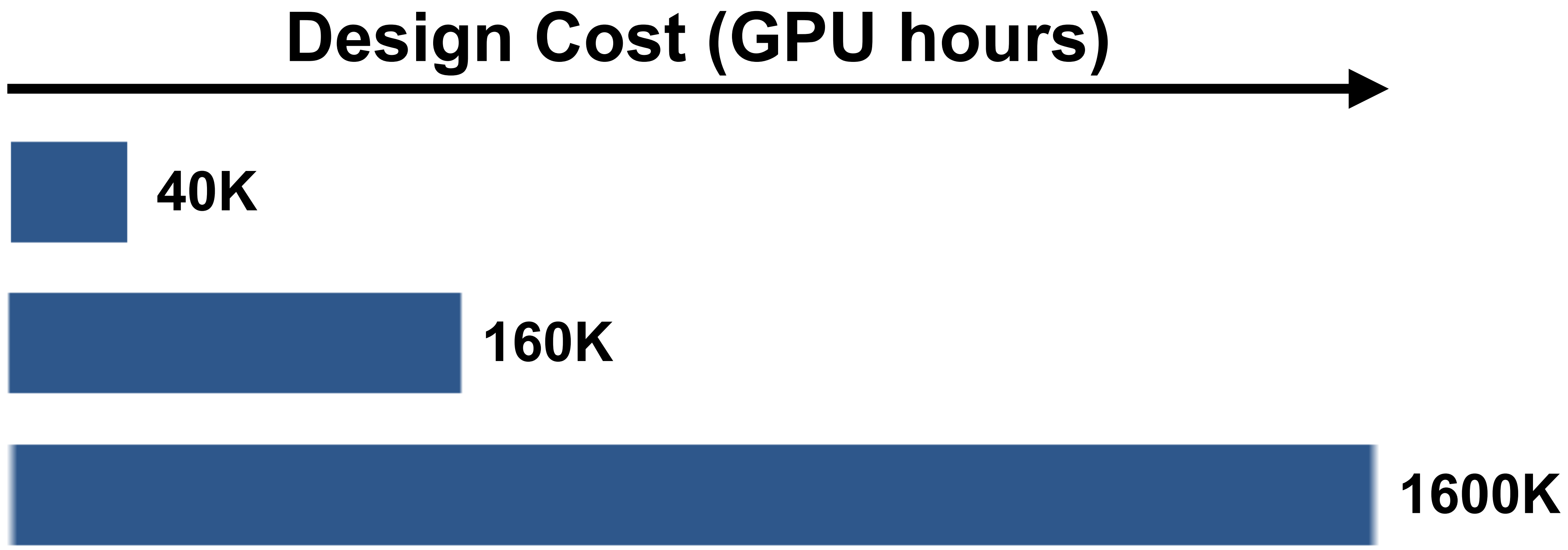
Mobile AI (10^{9} FLOPS)



Tiny AI (10^{6} FLOPS)

…

```
(2) for many devices:
    (1) for search episodes:
            for training iterations:
                forward-backward();
            if good_model: break;
        for post-search training iterations:
            forward-backward();
```

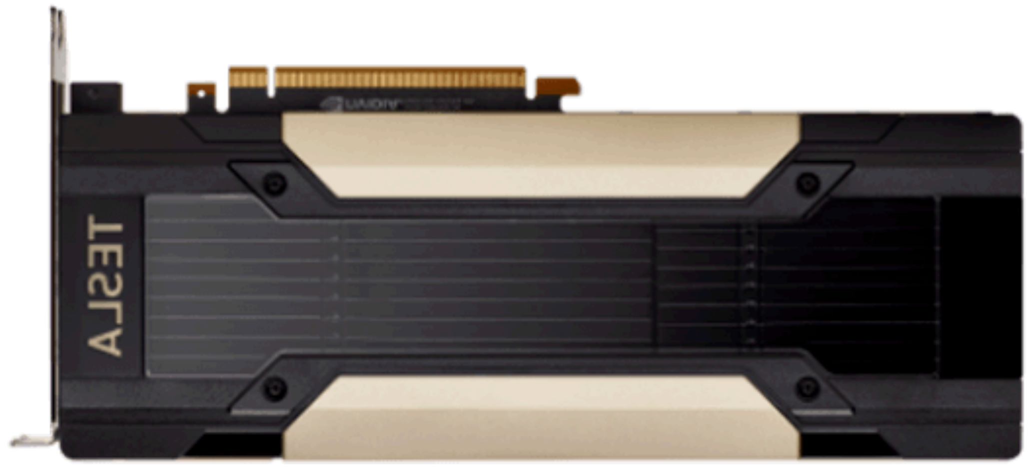


The design cost is calculated under the assumption of using MnasNet.

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

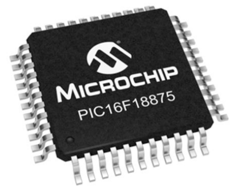
Challenge: Efficient Inference on Diverse Hardware Platforms

Diverse Hardware Platforms



Cloud AI (10^{12} FLOPS)

Mobile AI (10^{9} FLOPS)



Tiny AI (10^{6} FLOPS)

...

(2) for many devices:

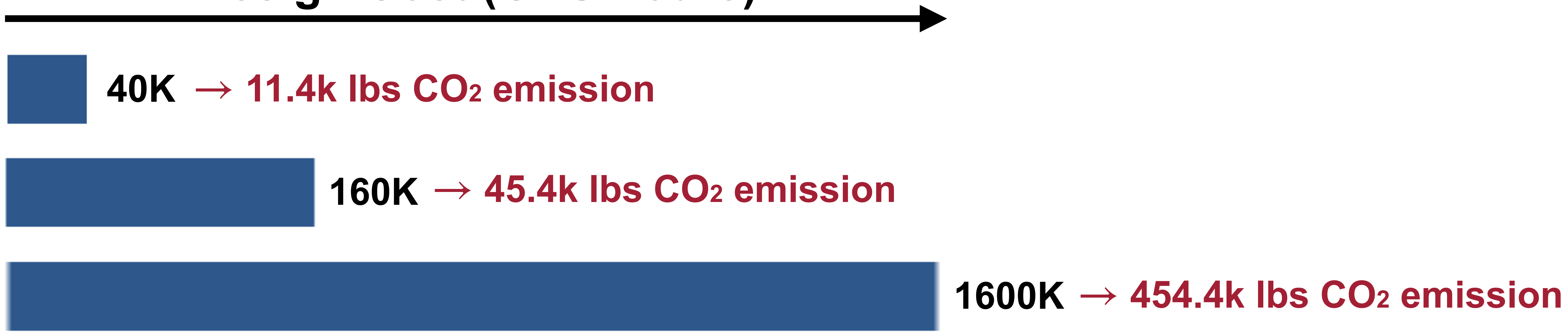
(1) for search episodes:

for training iterations:
forward-backward();

if good_model: break;

for post-search training iterations:
forward-backward();

Design Cost (GPU hours)



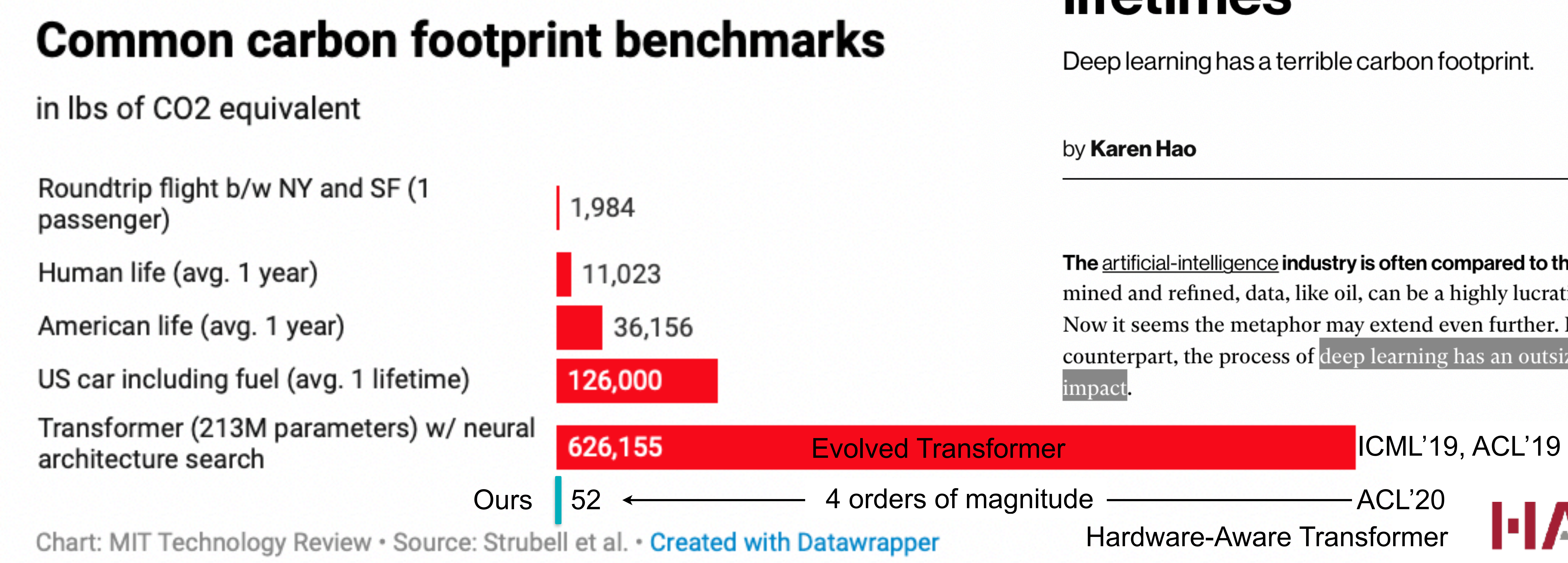
1 GPU hour translates to 0.284 lbs CO_2 emission according to
Strubell, Emma, et al. "Energy and policy considerations for deep learning in NLP." *ACL.* 2019.

Problem:

<u>TinyML</u> comes at the cost of <u>BigML</u>

(inference) (training/search)

We need Green AI:
Solve the Environmental Problem of NAS



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

The <u>artificial-intelligence</u> industry is often compared to the oil industry: once mined and refined, data, like oil, can be a highly lucrative commodity. Now it seems the metaphor may extend even further. Like its fossil-fuel counterpart, the process of deep learning has an outsize environmental impact.

OFA: Decouple Training and Search

Conventional NAS

```
(2) for devices:
  (1) for search episodes:
        for training iterations:
            forward-backward();
        if good_model: break;
      for post-search training iterations:
            forward-backward();
```

=>

Once-for-All:

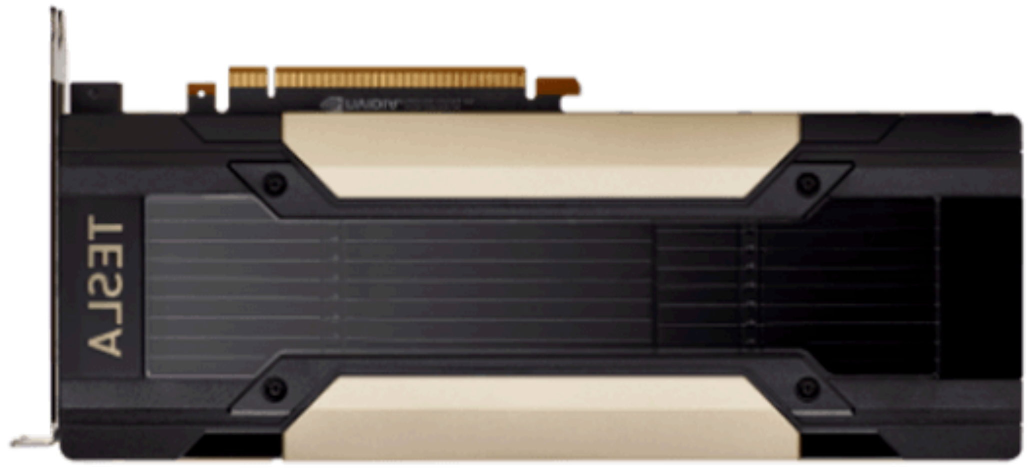
```
for OFA training iterations:
    forward-backward();
------------------------------- decouple
for devices:
    for search episodes:
        sample from OFA;
        if good_model: break;
    direct deploy without training;
```

training

search

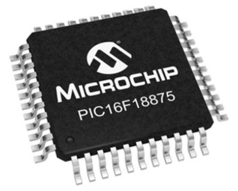
Challenge: Efficient Inference on Diverse Hardware Platforms

Diverse Hardware Platforms



Cloud AI (10^{12} FLOPS)

Mobile AI (10^{9} FLOPS)



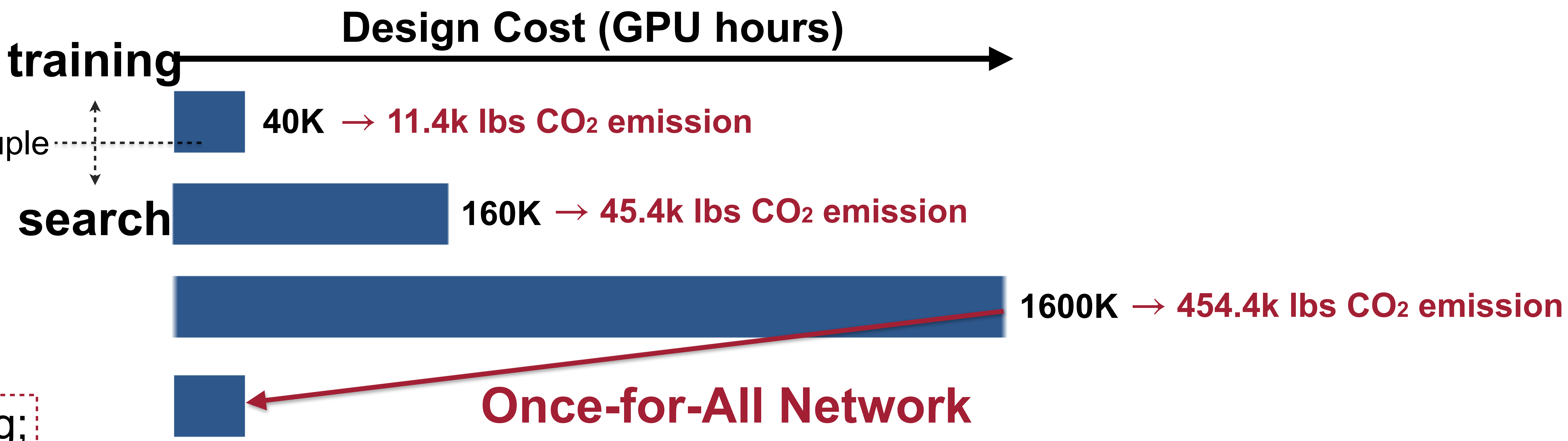
Tiny AI (10^{6} FLOPS)

...

```
for OFA training iterations:
    forward-backward();
```

decouple

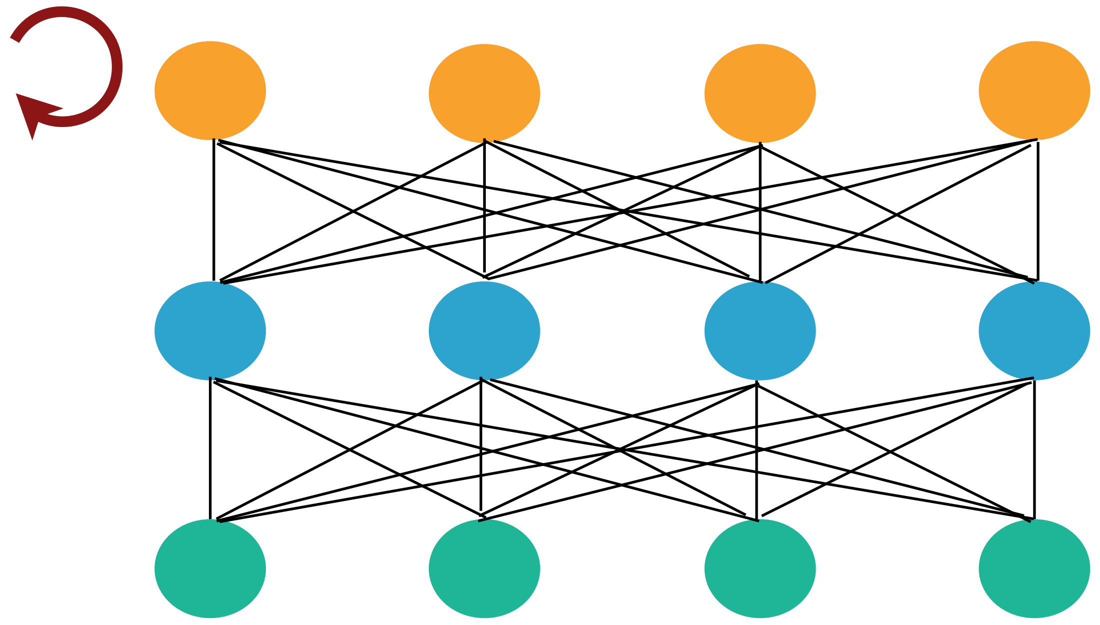
```
for devices:
    for search episodes:
        sample from OFA;
        if good_model: break;
    direct deploy without training;
```

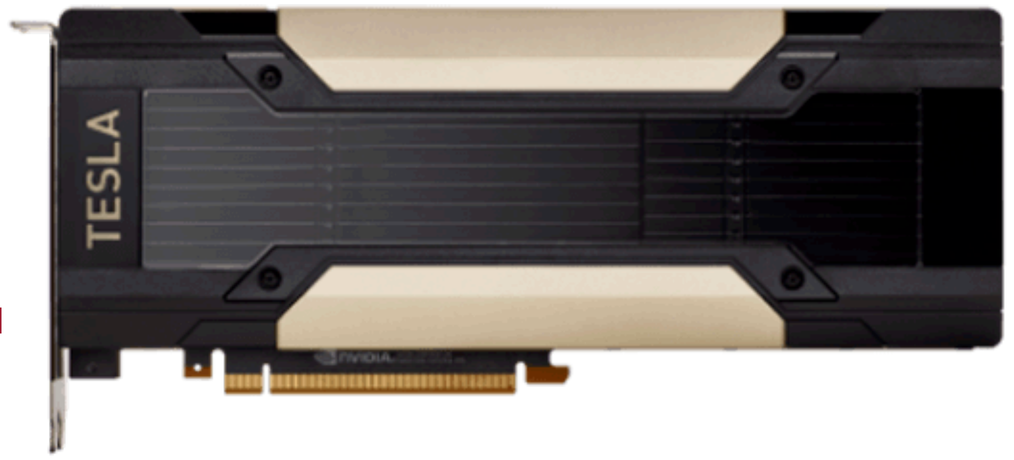


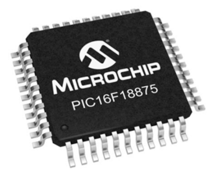
1 GPU hour translates to 0.284 lbs CO_2 emission according to
Strubell, Emma, et al. "Energy and policy considerations for deep learning in NLP." *ACL.* 2019.

Once-for-All Network:
Decouple Model Training and Architecture Design

once-for-all network



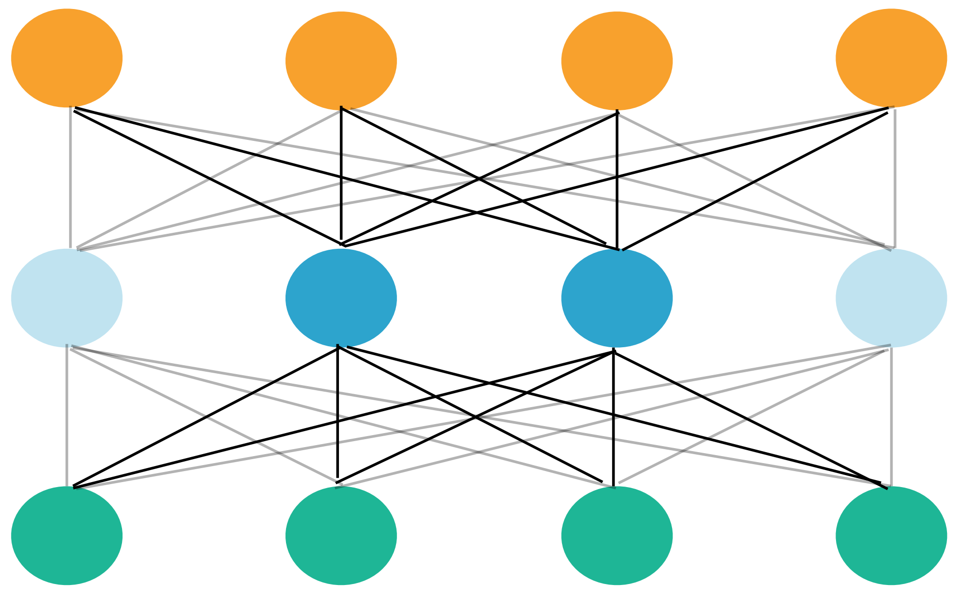


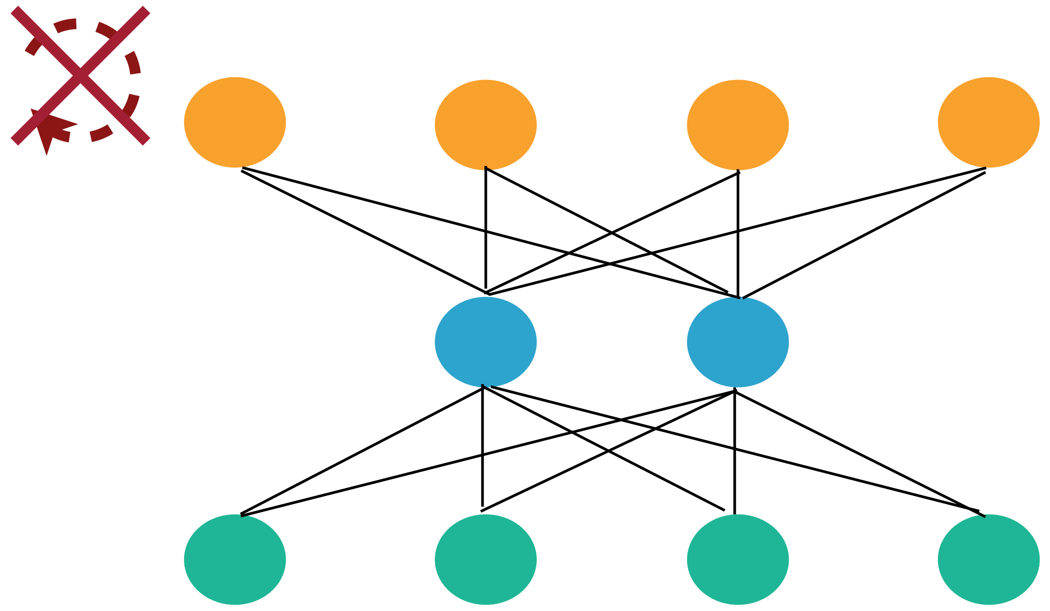


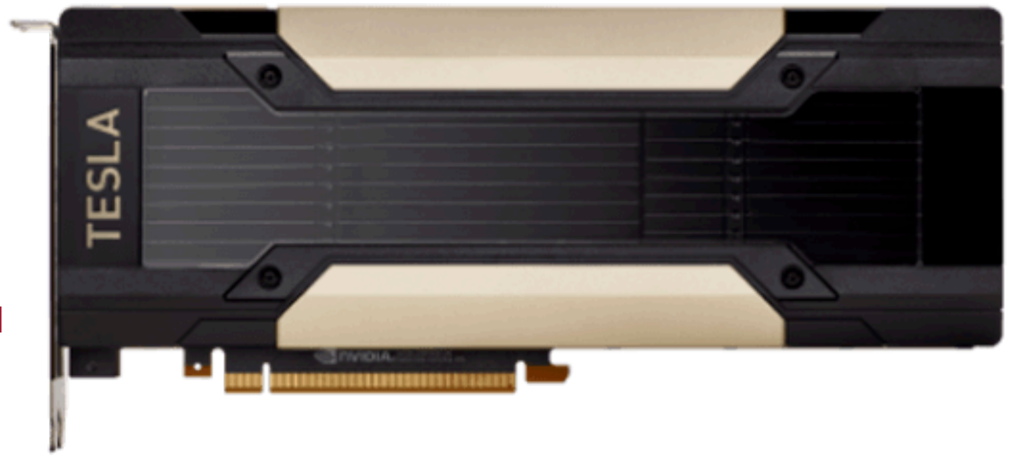
[Once-for-All](), ICLR'20

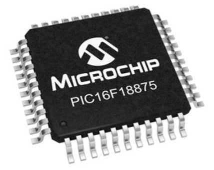
Once-for-All Network:
Decouple Model Training and Architecture Design

once-for-all network





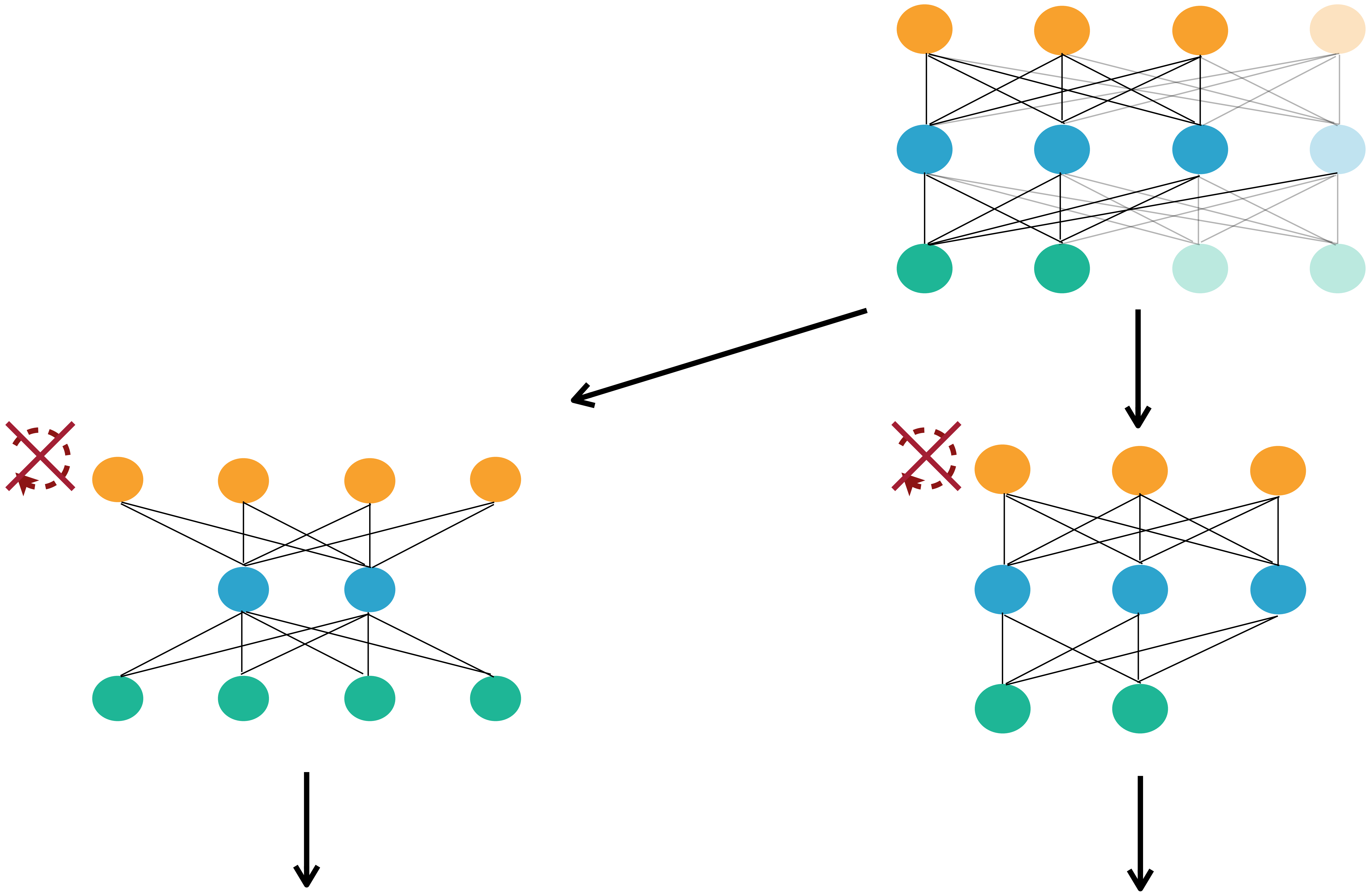


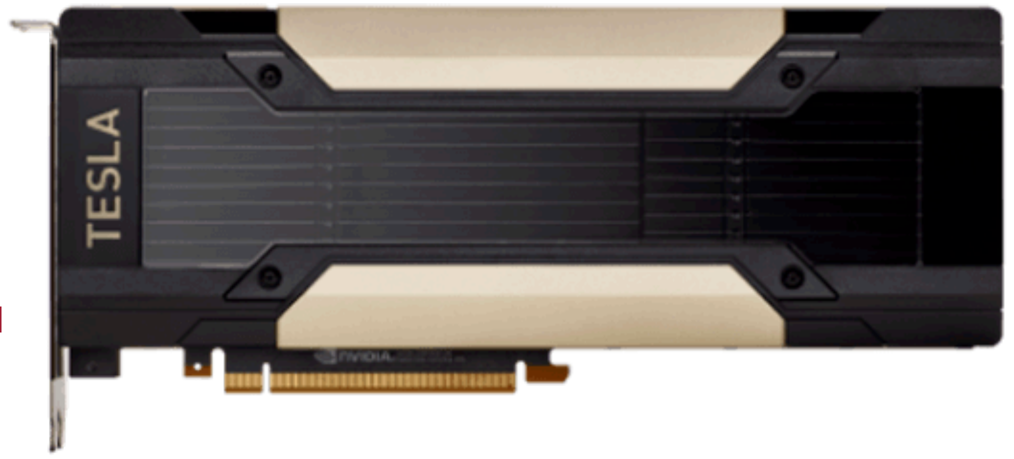


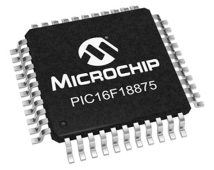
Once-for-All, ICLR'20

Once-for-All Network: Decouple Model Training and Architecture Design

once-for-all network

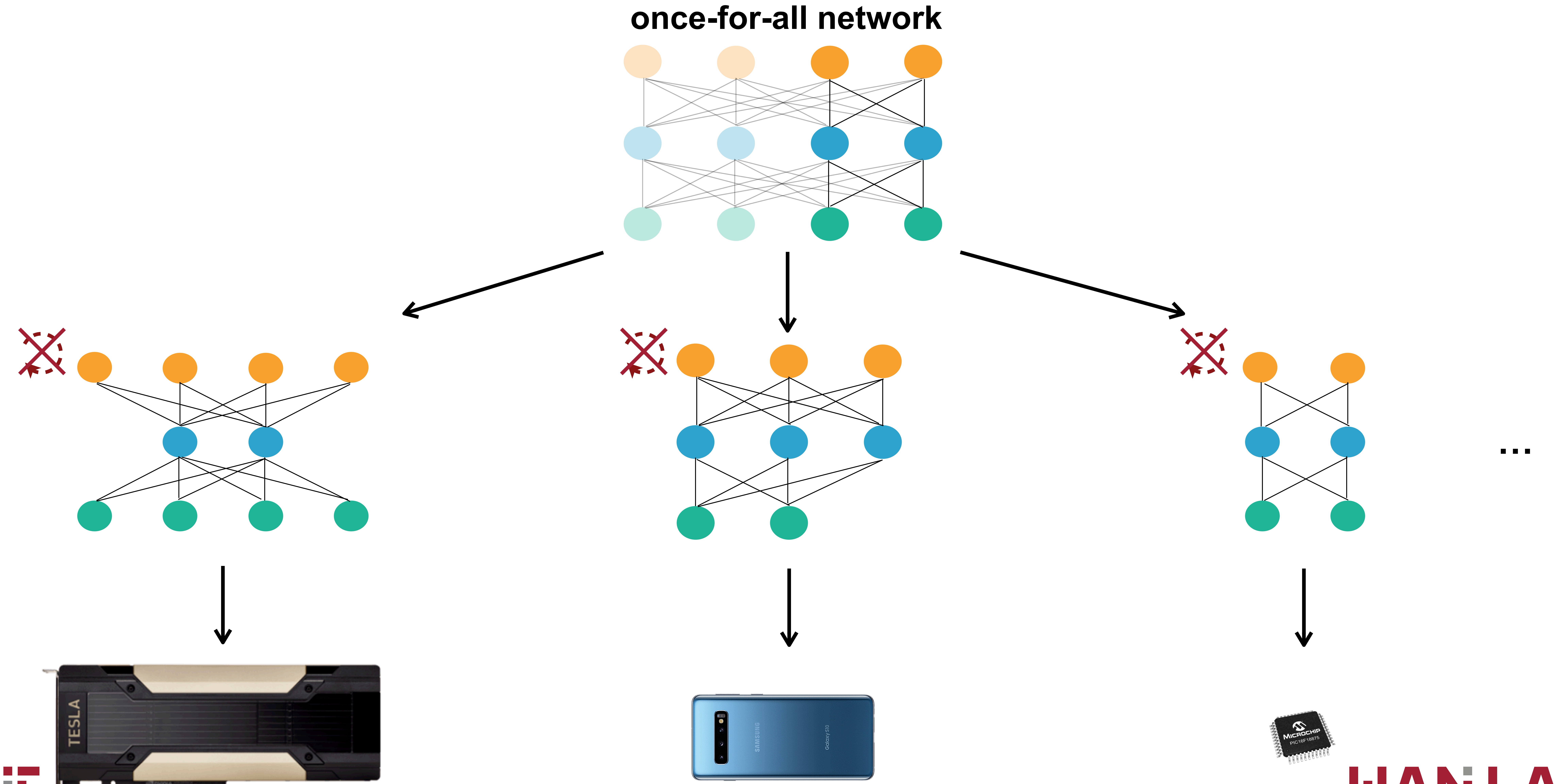






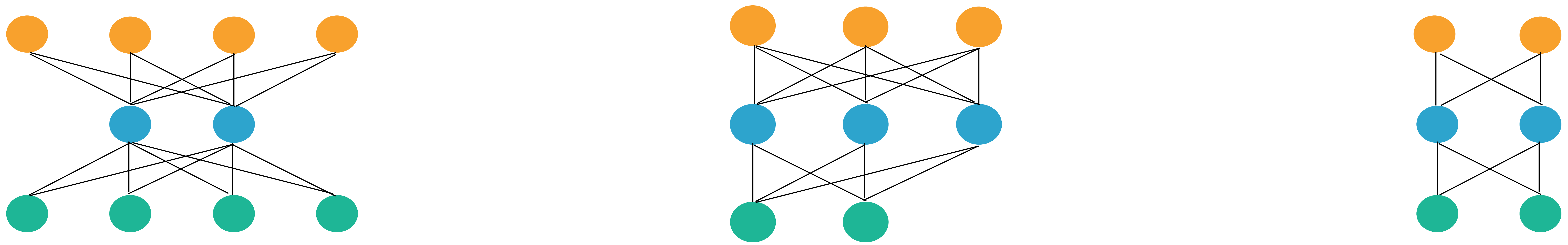
Once-for-All, ICLR'20

Once-for-All Network: Decouple Model Training and Architecture Design



[Once-for-All](), ICLR'20

Challenge: how to prevent different subnetworks from interfering with each other?



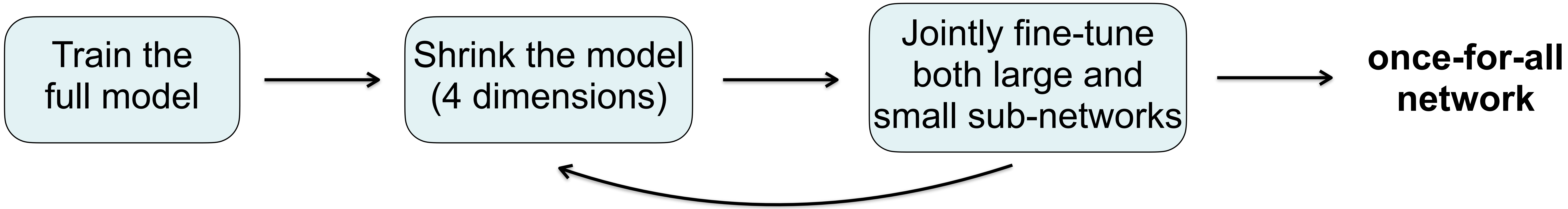
Solution: Progressive Shrinking

- More than 10^{19} **different sub-networks** in a single once-for-all network, covering 4 different dimensions: **resolution**, **kernel size**, **depth**, **width**.
- Directly optimizing the once-for-all network from scratch is much more challenging than training a normal neural network given so many sub-networks to support.

Solution: Progressive Shrinking

- More than 10^{19} **different sub-networks** in a single once-for-all network, covering 4 different dimensions: **resolution**, **kernel size**, **depth**, **width**.
- Directly optimizing the once-for-all network from scratch is much more challenging than training a normal neural network given so many sub-networks to support.

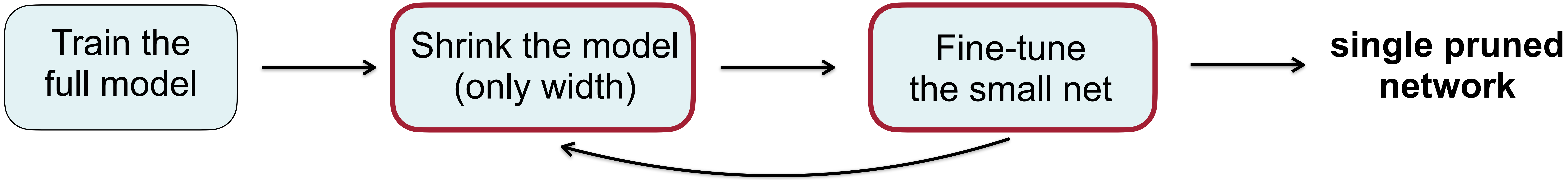
Progressive Shrinking



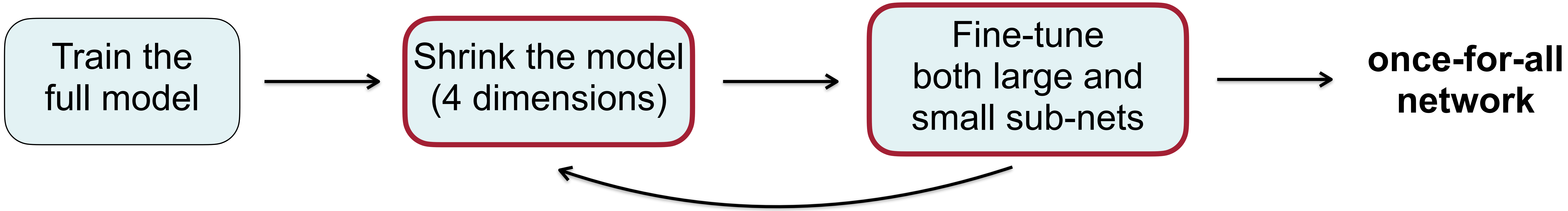
- Small sub-networks are nested in large sub-networks.
- Cast the training process of the once-for-all network as a progressive shrinking and joint fine-tuning process.

Connection to Network Pruning

Network Pruning

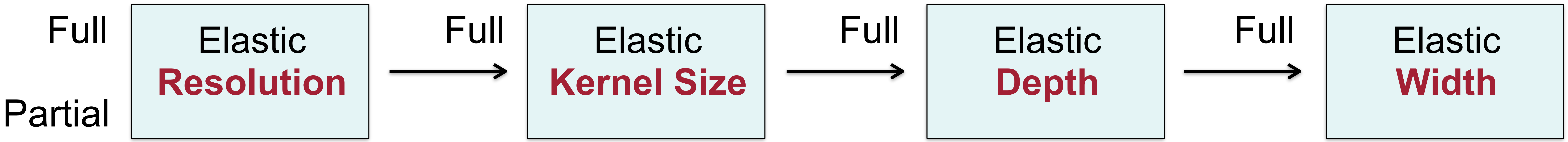


Progressive Shrinking

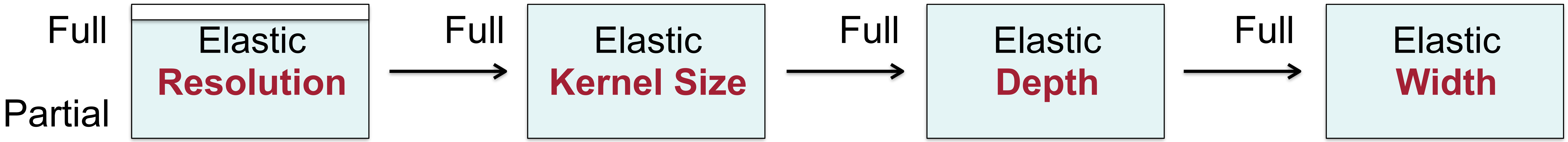


- Progressive shrinking can be viewed as a generalized network pruning with much higher flexibility across 4 dimensions.

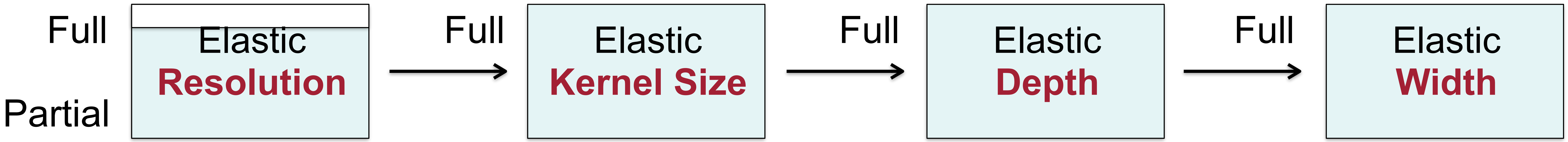
Progressive Shrinking



Progressive Shrinking



Progressive Shrinking



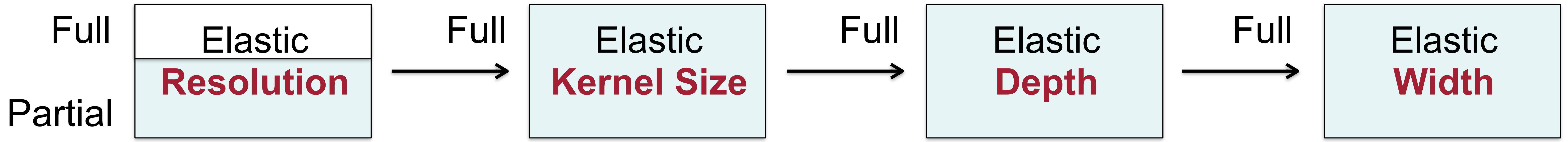
Progressive Shrinking

Full / Partial	**Elastic Resolution**	Full →	**Elastic Kernel Size**	Full →	**Elastic Depth**	Full →	**Elastic Width**

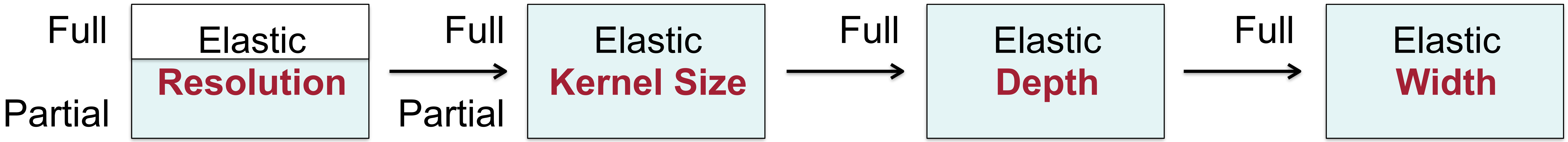
Progressive Shrinking

Full / Partial: Elastic **Resolution** → Full: Elastic **Kernel Size** → Full: Elastic **Depth** → Full: Elastic **Width**

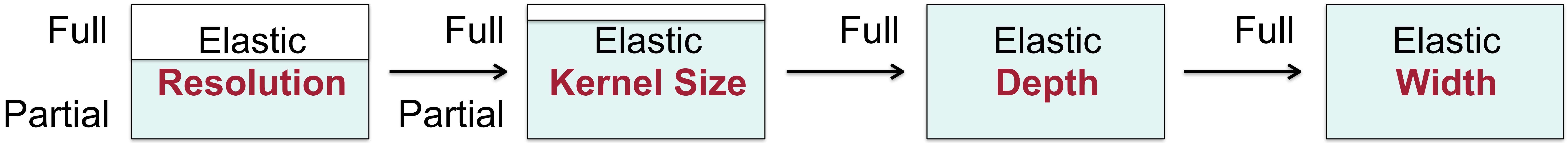
Progressive Shrinking



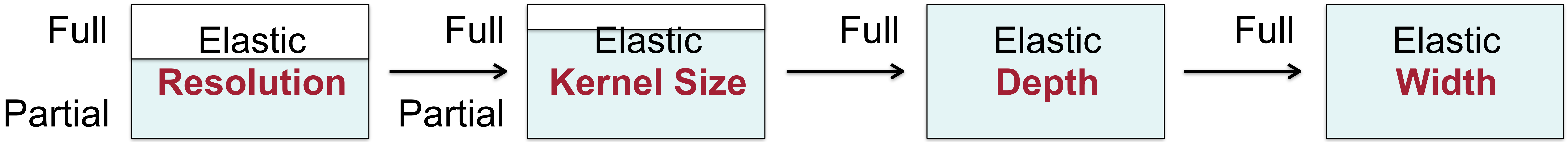
Progressive Shrinking



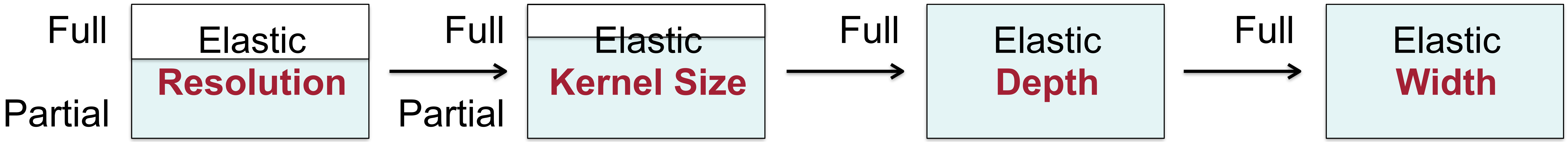
Progressive Shrinking



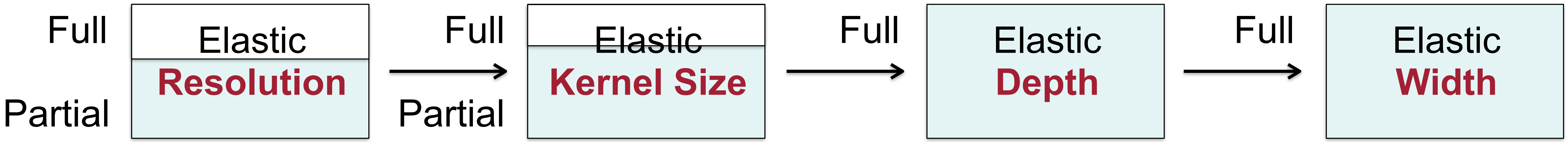
Progressive Shrinking



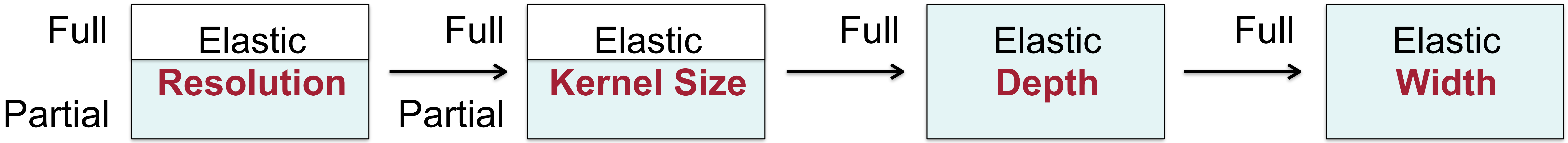
Progressive Shrinking



Progressive Shrinking



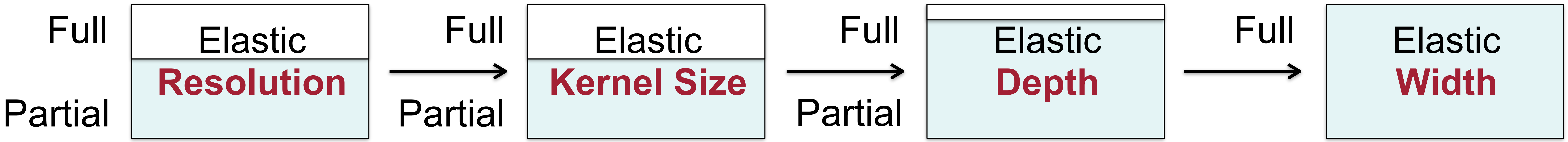
Progressive Shrinking



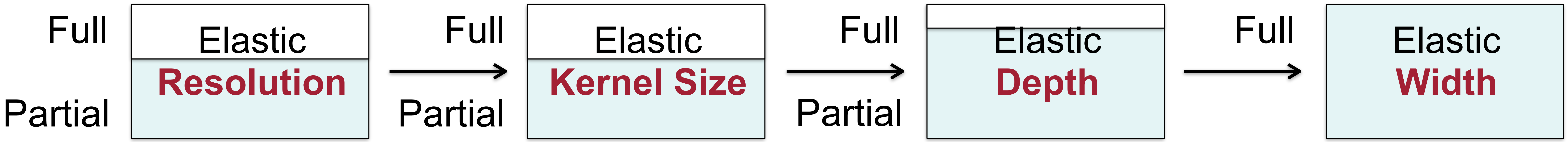
Progressive Shrinking

Full / Partial → **Elastic Resolution** → Full / Partial → **Elastic Kernel Size** → Full / Partial → **Elastic Depth** → Full → **Elastic Width**

Progressive Shrinking



Progressive Shrinking



Progressive Shrinking

Full / Partial	Elastic **Resolution**	→ Full / Partial	Elastic **Kernel Size**	→ Full / Partial	Elastic **Depth**	→ Full	Elastic **Width**

Progressive Shrinking

Full / Partial → **Elastic Resolution** → Full / Partial → **Elastic Kernel Size** → Full / Partial → **Elastic Depth** → Full → **Elastic Width**

Progressive Shrinking

Full

Partial

Elastic **Resolution**

Full

Partial

Elastic **Kernel Size**

Full

Partial

Elastic **Depth**

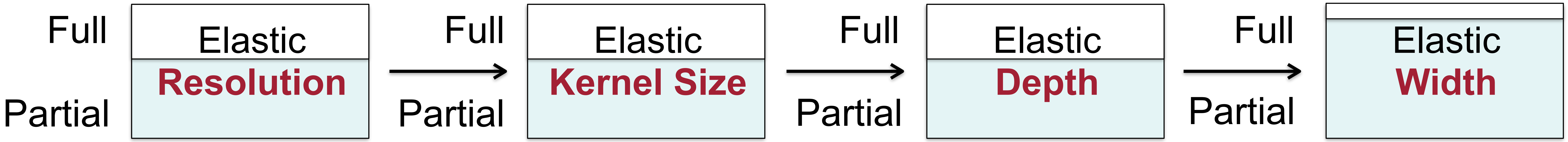
Full

Elastic **Width**

Progressive Shrinking

Full / Partial: Elastic **Resolution** → Full / Partial: Elastic **Kernel Size** → Full / Partial: Elastic **Depth** → Full / Partial: Elastic **Width**

Progressive Shrinking



Progressive Shrinking

Full / Partial → Elastic **Resolution** → Full / Partial → Elastic **Kernel Size** → Full / Partial → Elastic **Depth** → Full / Partial → Elastic **Width**

[Once-for-All](), ICLR'20

Progressive Shrinking

Full → Partial: Elastic **Resolution** → Full → Partial: Elastic **Kernel Size** → Full → Partial: Elastic **Depth** → Full → Partial: Elastic **Width**

Progressive Shrinking

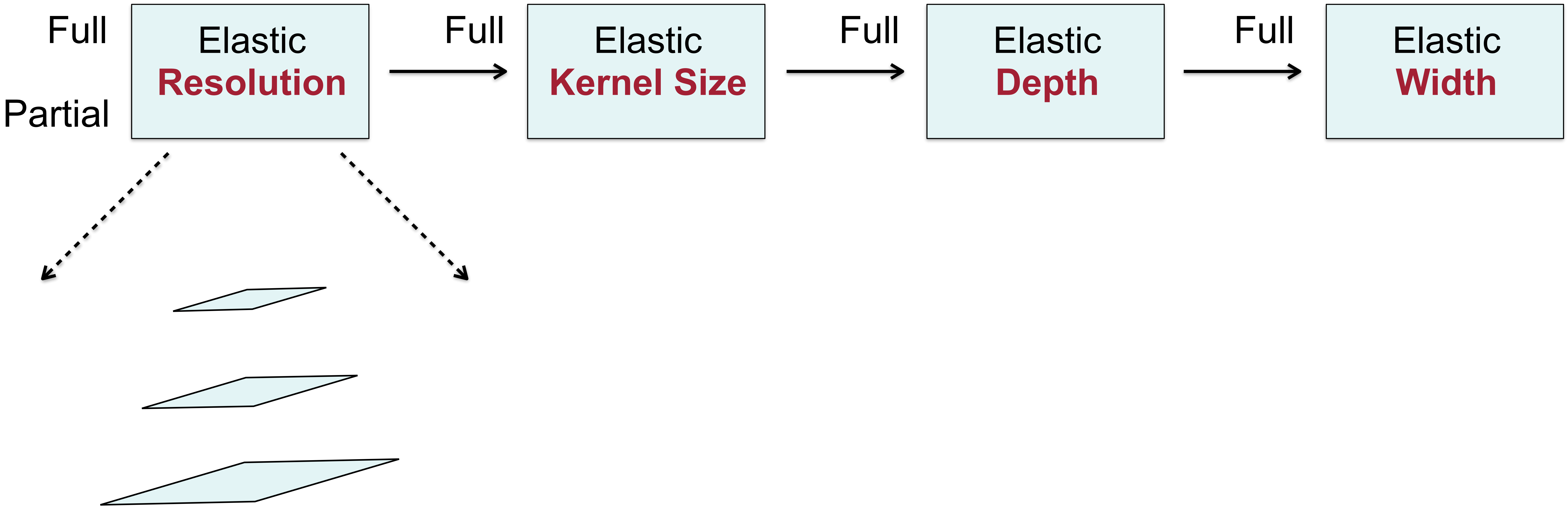
Full → Partial: Elastic **Resolution** → Elastic **Kernel Size** → Elastic **Depth** → Elastic **Width**

Full / Partial	Elastic	Full / Partial	Elastic	Full / Partial	Elastic	Full / Partial	Elastic
Full → Partial	**Resolution**	Full → Partial	**Kernel Size**	Full → Partial	**Depth**	Full → Partial	**Width**

Progressive Shrinking

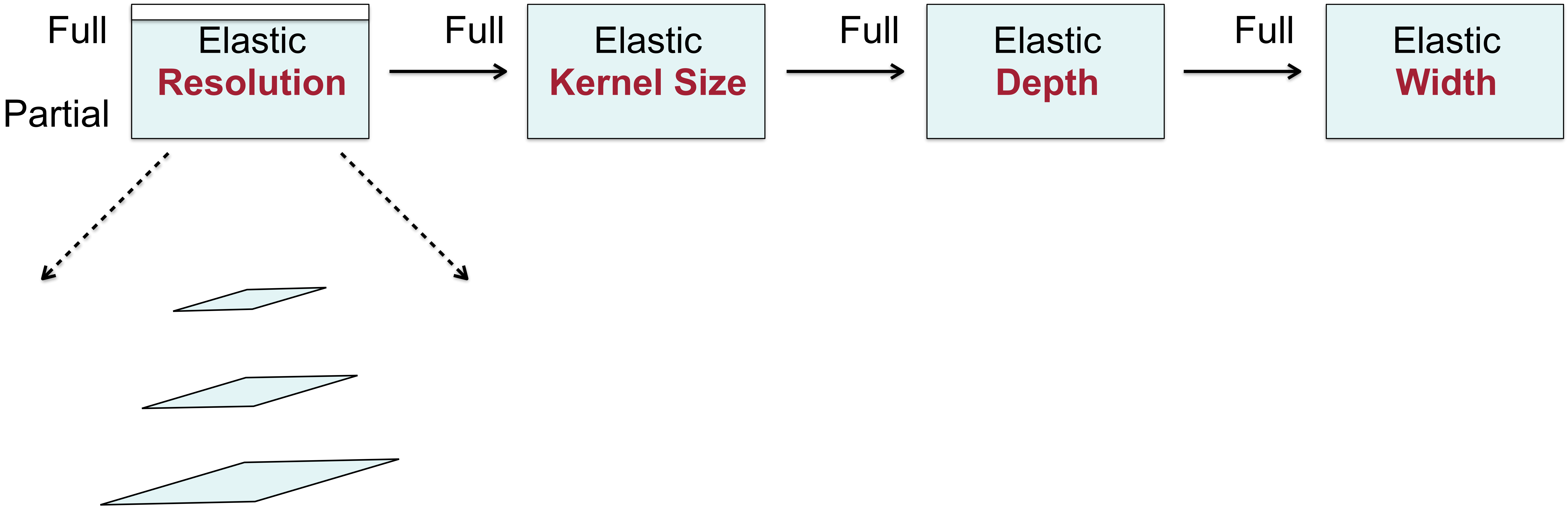
Full / Partial → **Elastic Resolution** → Full / Partial → **Elastic Kernel Size** → Full / Partial → **Elastic Depth** → Full / Partial → **Elastic Width**

Progressive Shrinking



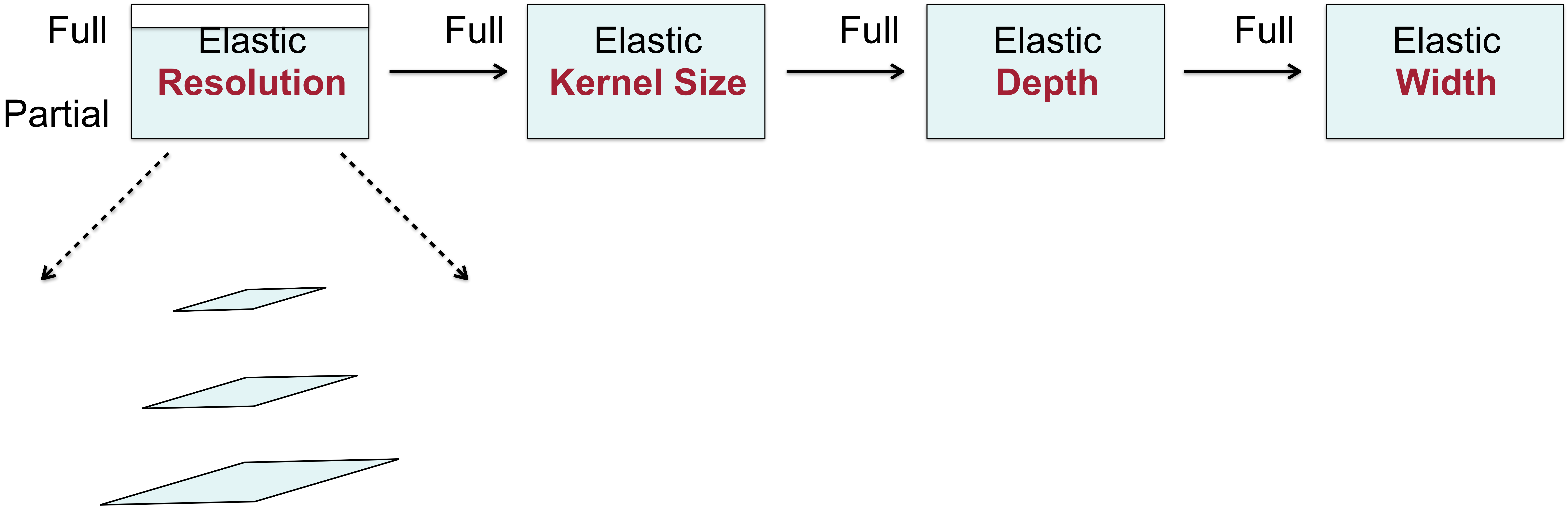
Randomly sample input image size for each batch

Progressive Shrinking



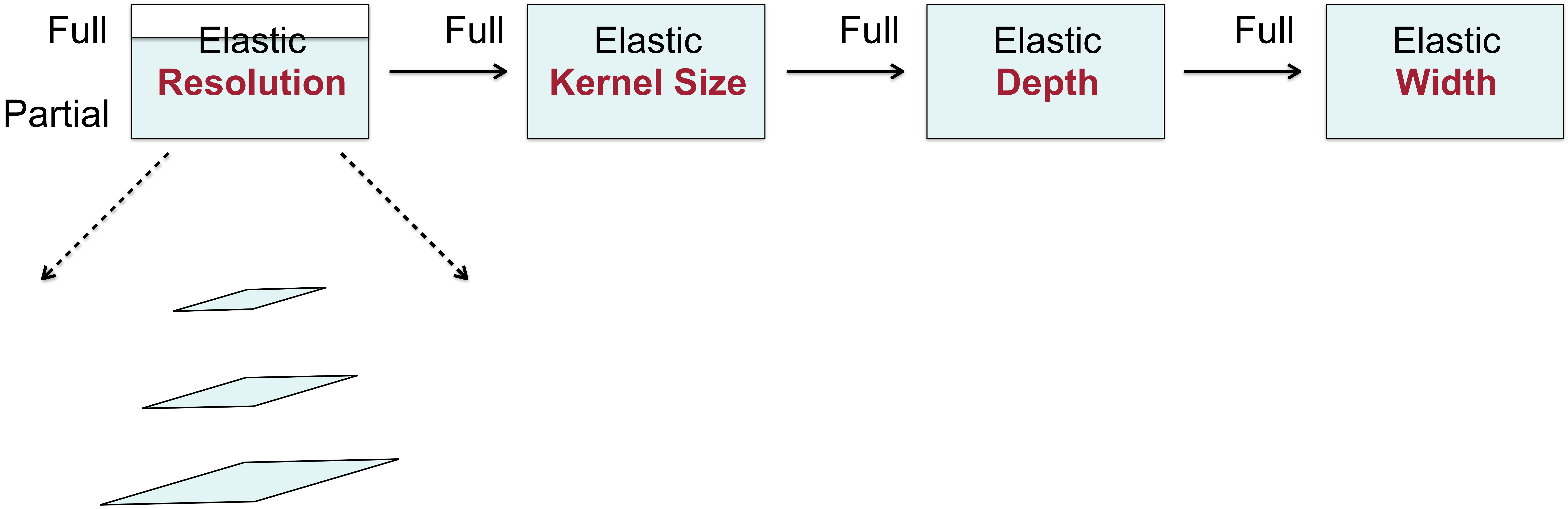
Randomly sample input image size for each batch

Progressive Shrinking



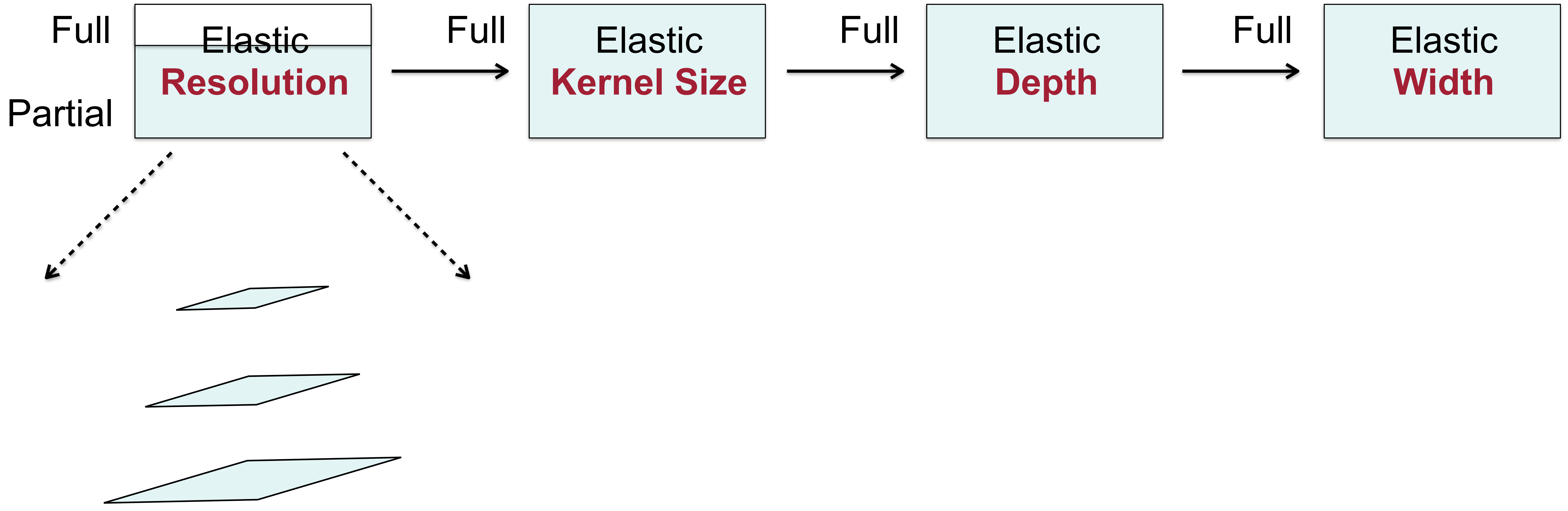
Randomly sample input image size for each batch

Progressive Shrinking



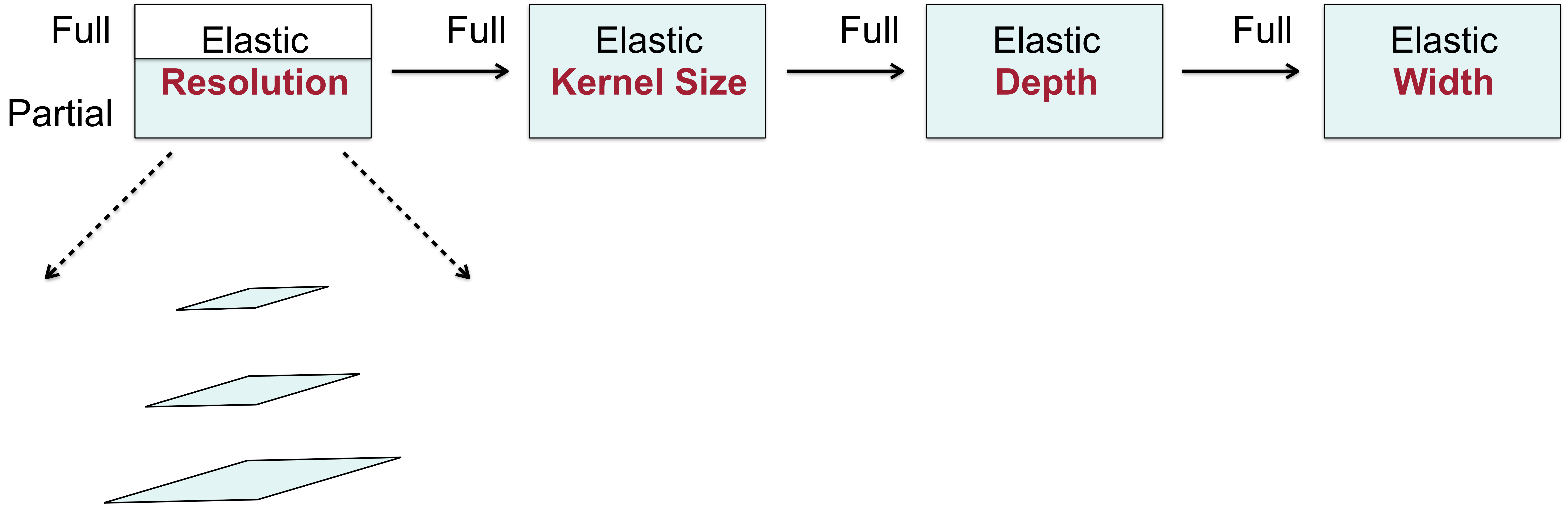
Randomly sample input image size for each batch

Progressive Shrinking



Randomly sample input image
size for each batch

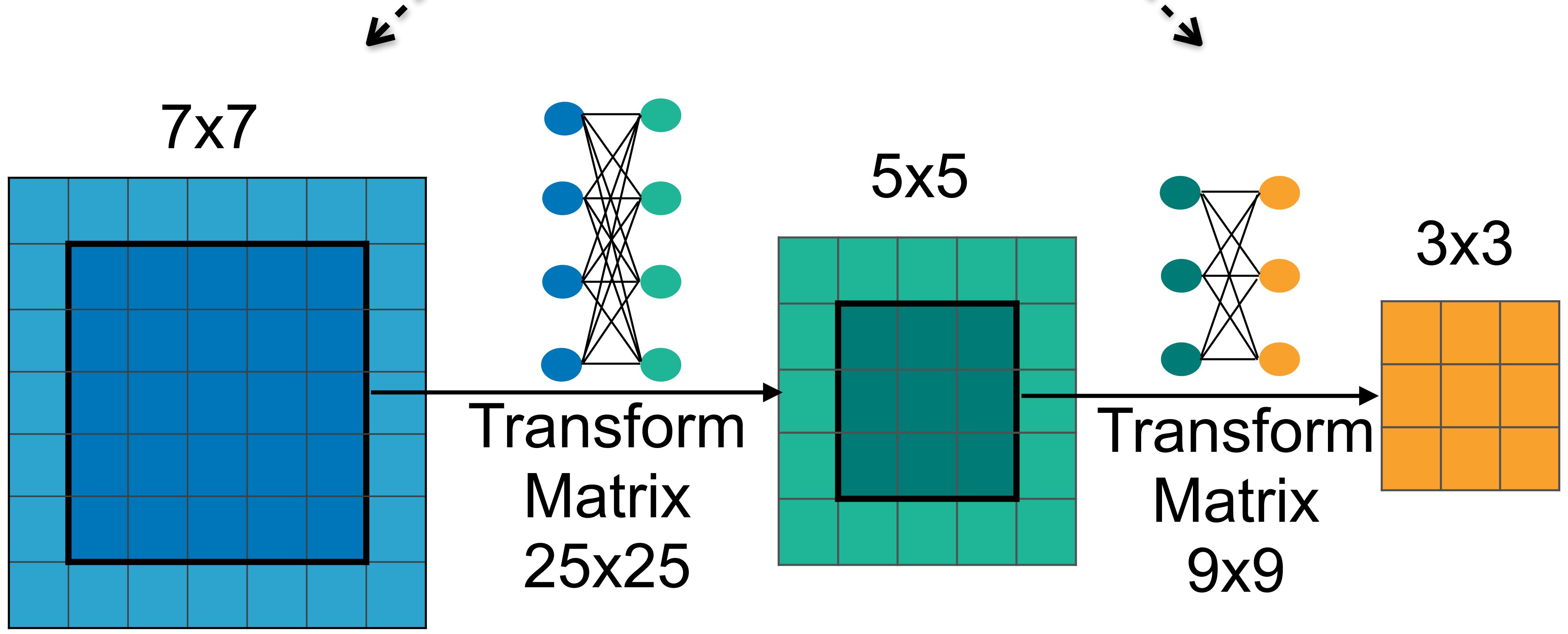
Progressive Shrinking



Randomly sample input image size for each batch

Progressive Shrinking

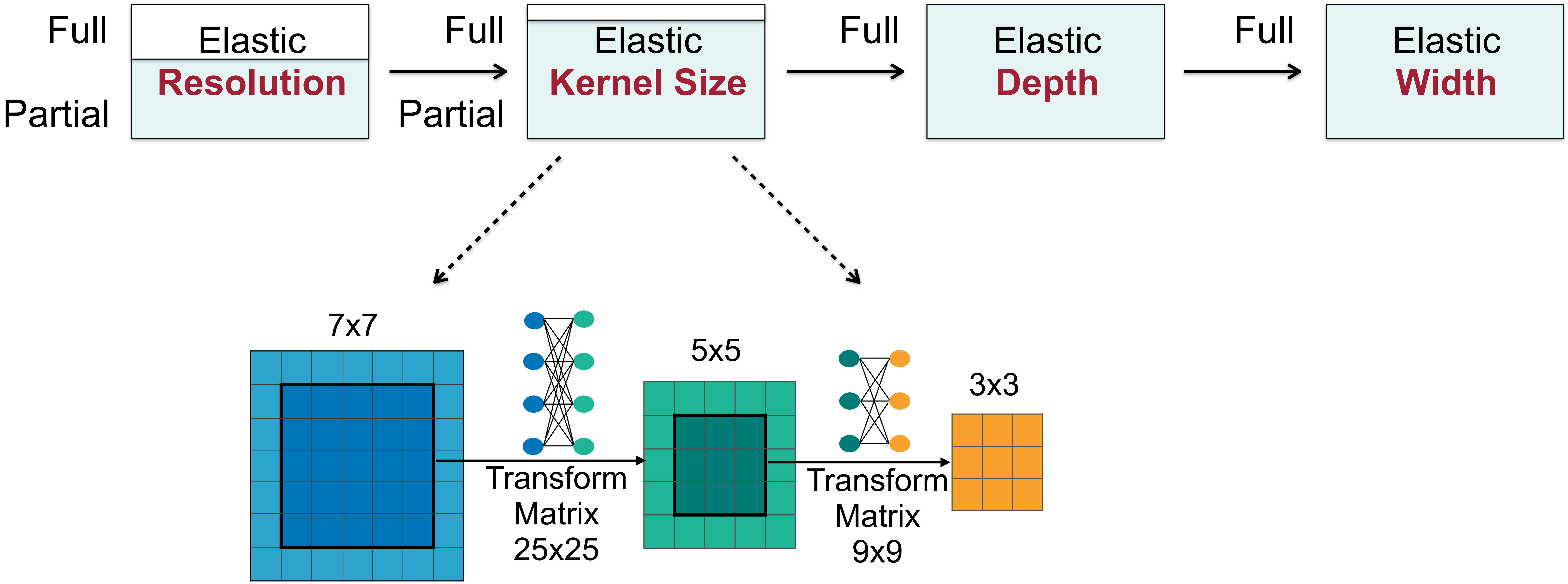
Full / Partial → **Elastic Resolution** → Full / Partial → **Elastic Kernel Size** → Full → **Elastic Depth** → Full → **Elastic Width**



Start with full kernel size

Smaller kernel takes centered weights via a transformation matrix

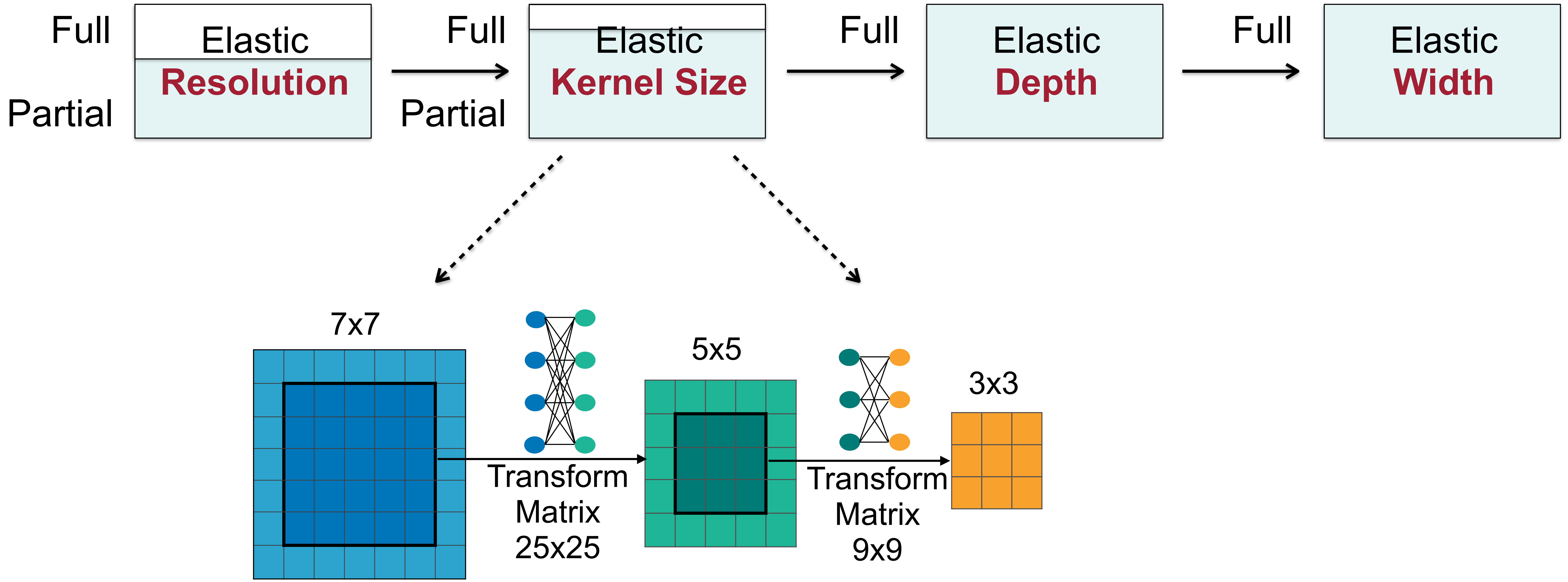
Progressive Shrinking



Start with full kernel size

Smaller kernel takes centered weights via a transformation matrix

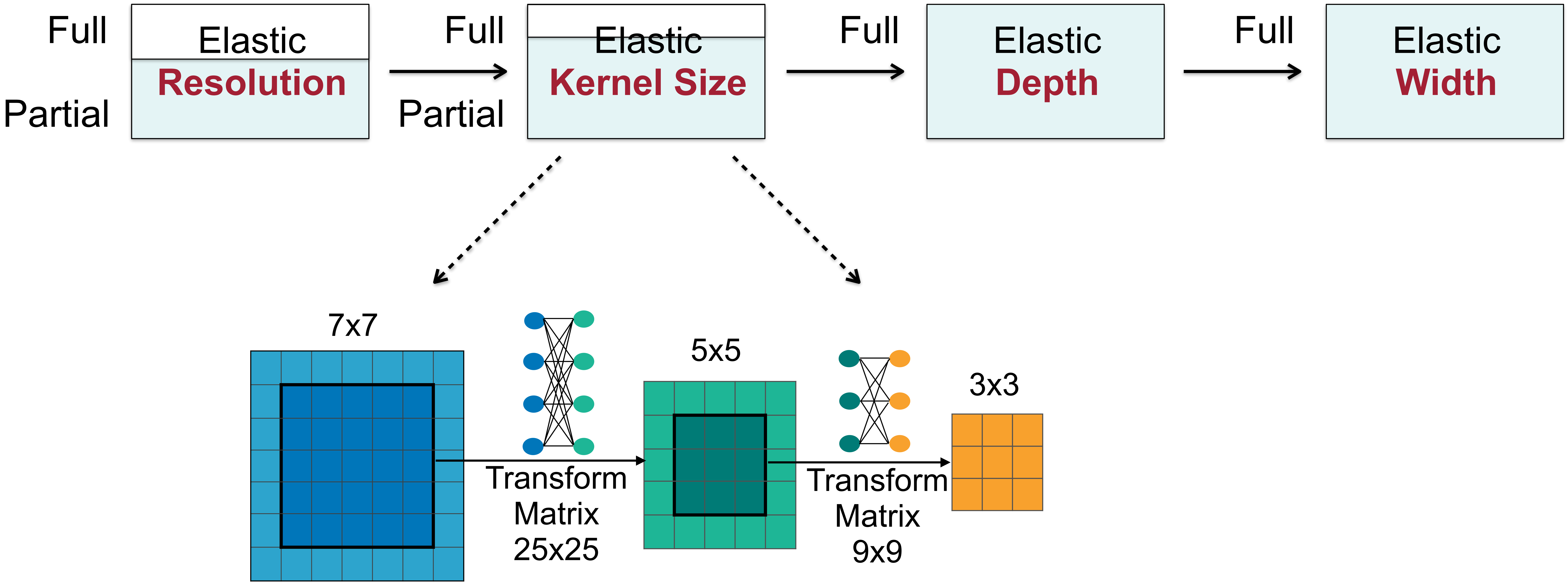
Progressive Shrinking



Start with full kernel size

Smaller kernel takes centered weights via a transformation matrix

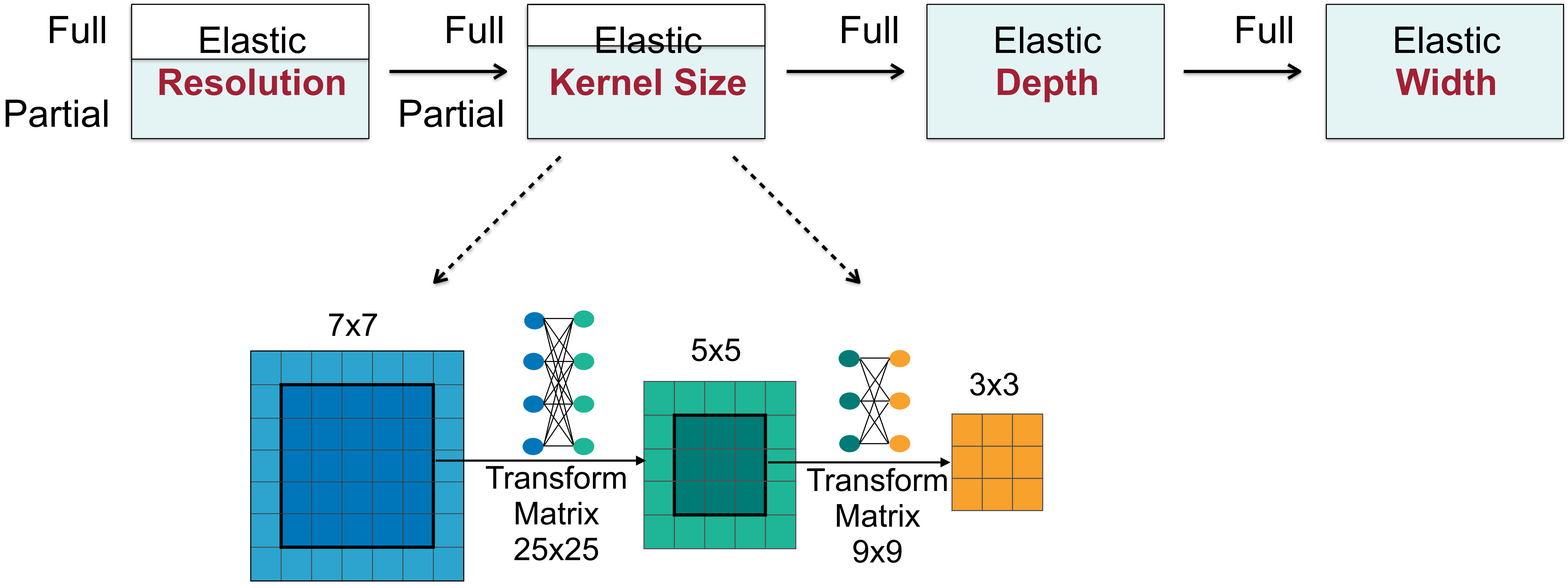
Progressive Shrinking



Start with full kernel size

Smaller kernel takes centered weights via a transformation matrix

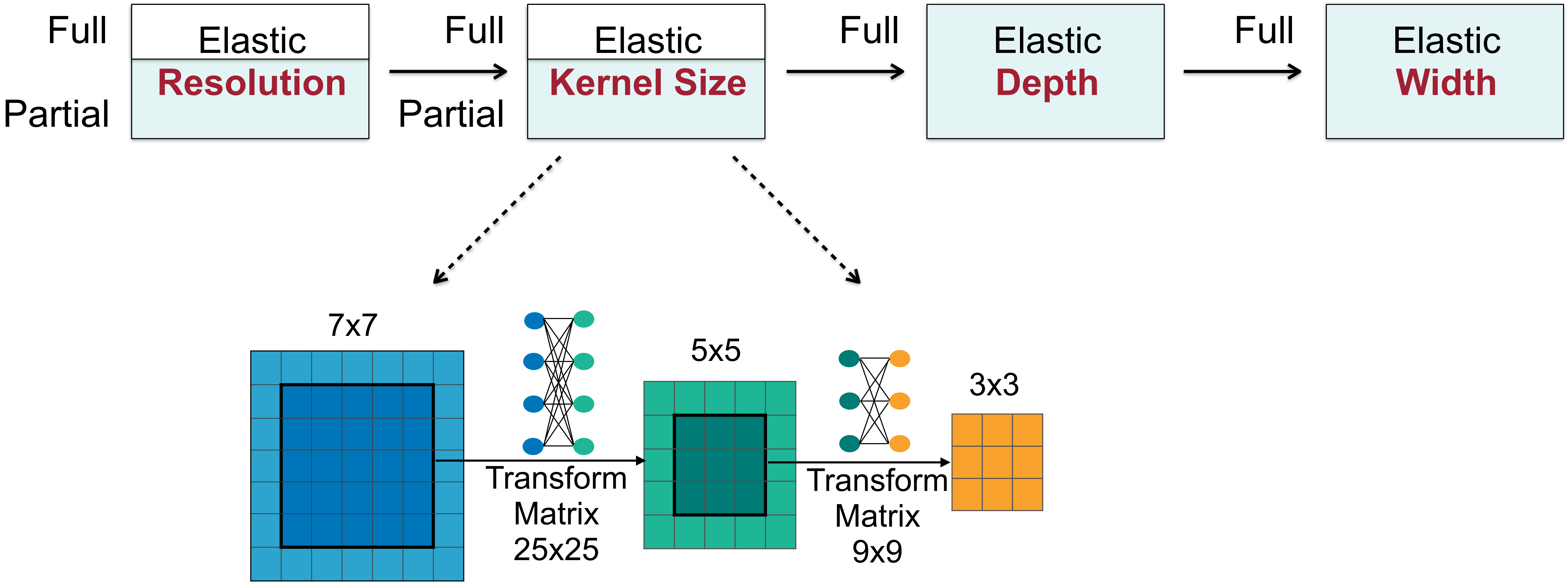
Progressive Shrinking



Start with full kernel size

Smaller kernel takes centered weights via a transformation matrix

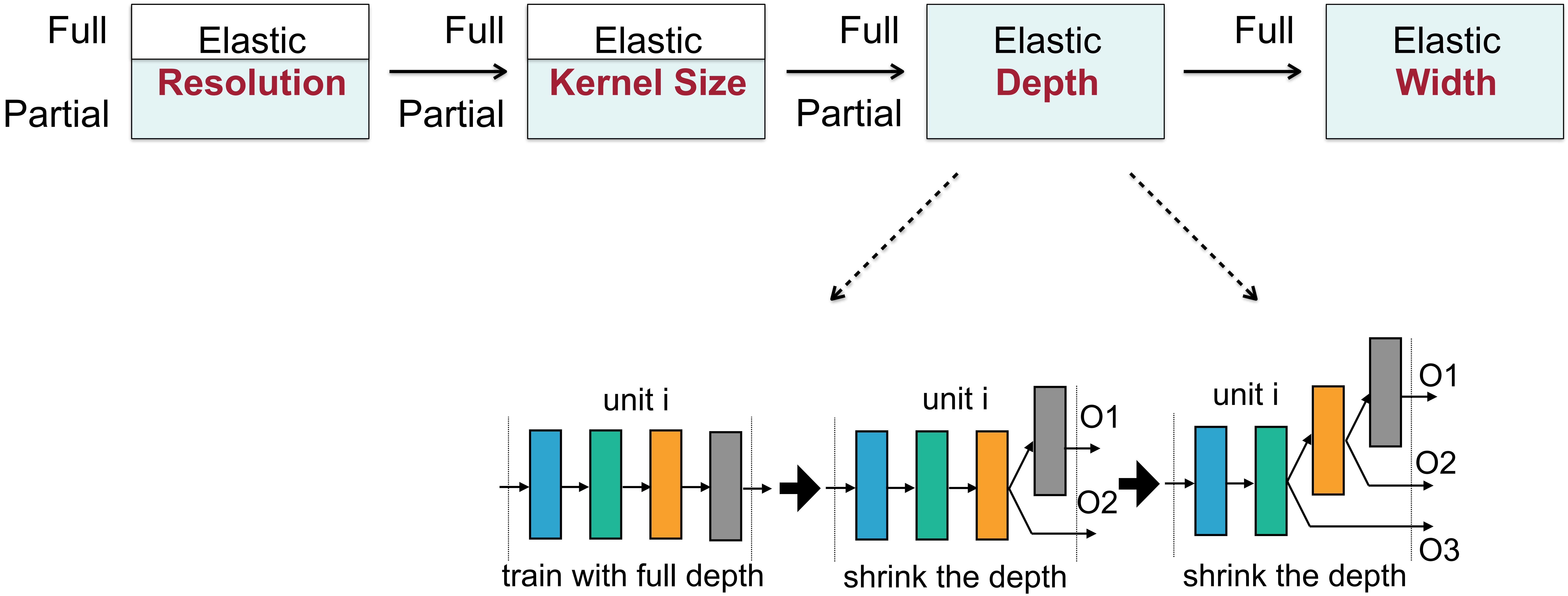
Progressive Shrinking



Start with full kernel size

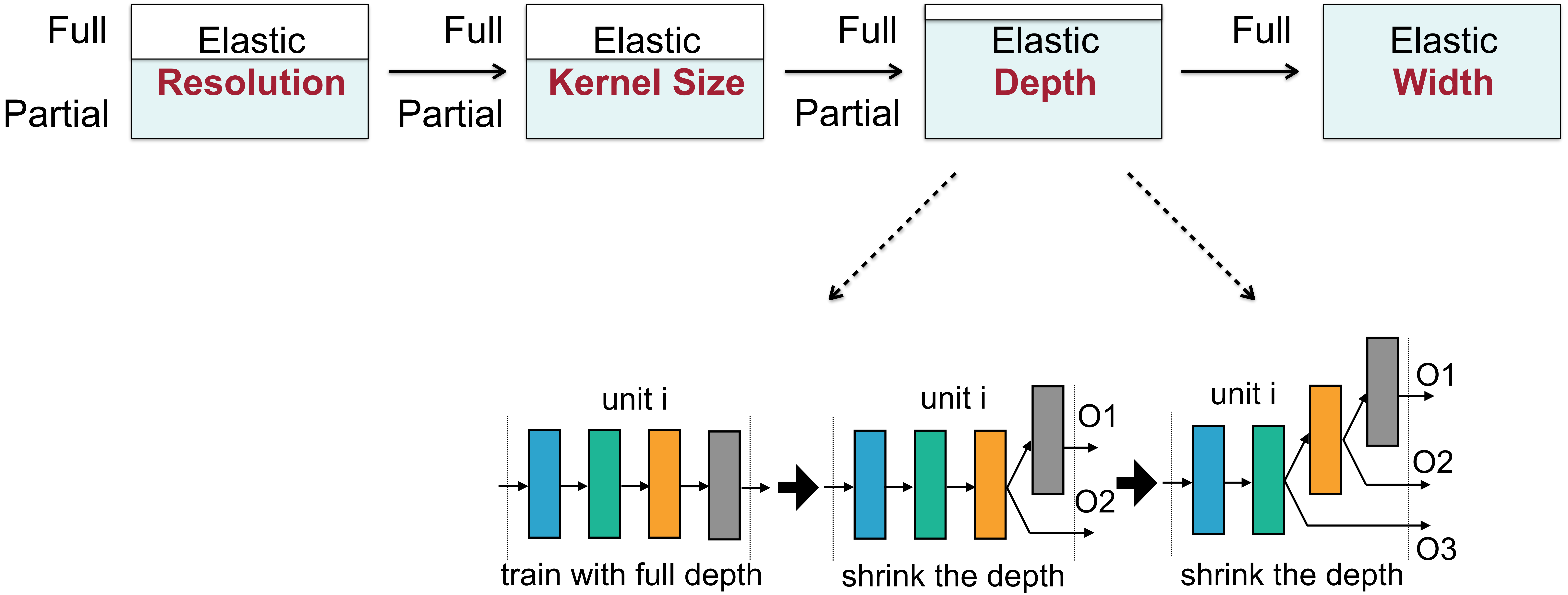
Smaller kernel takes centered weights via a transformation matrix

Progressive Shrinking



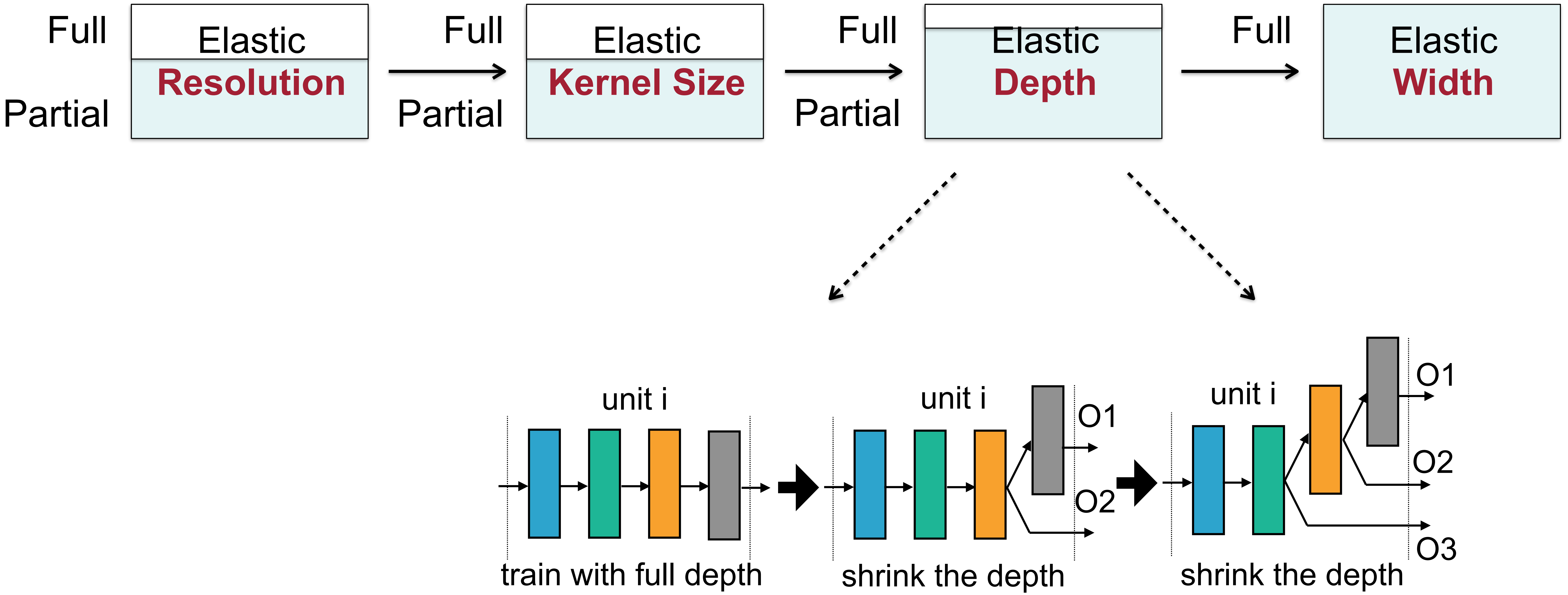
Gradually allow later layers in each unit
to be skipped to reduce the depth

Progressive Shrinking



Gradually allow later layers in each unit
to be skipped to reduce the depth

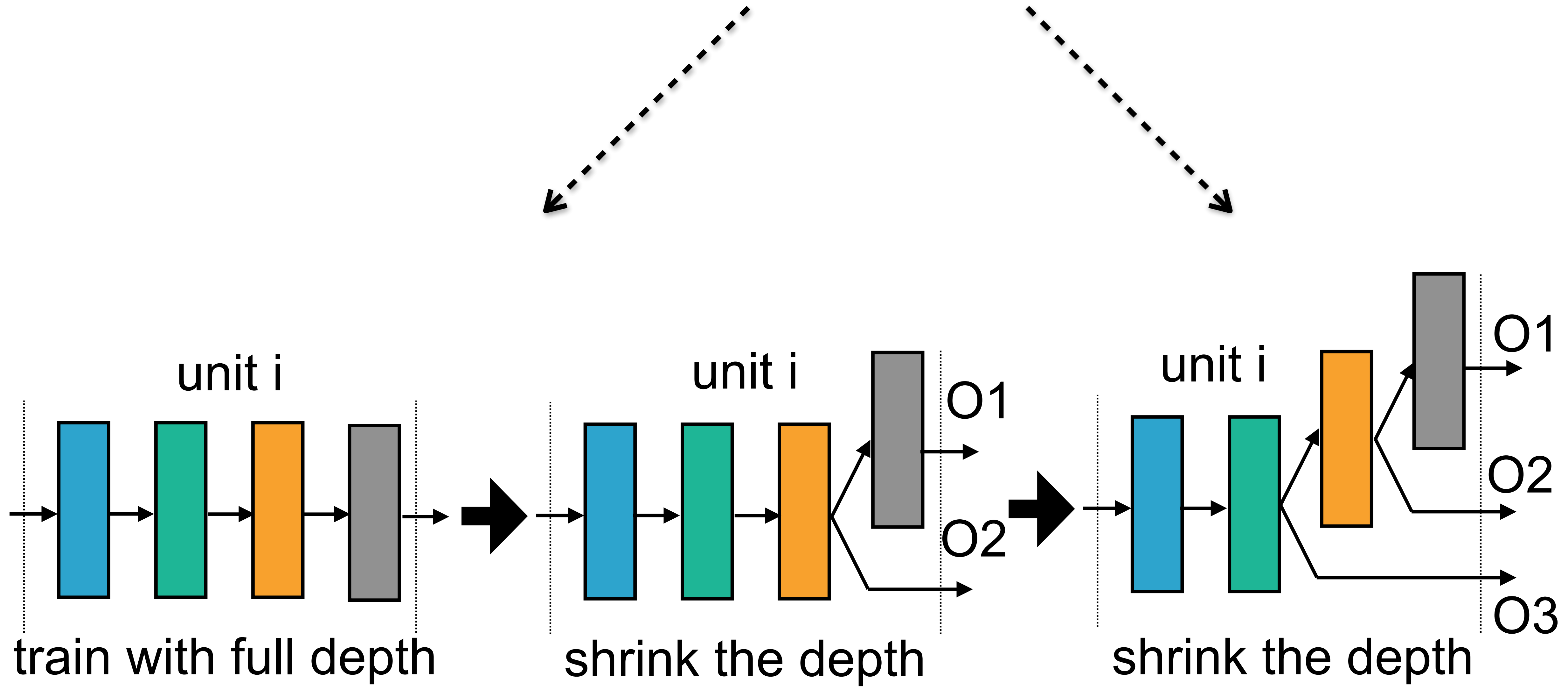
Progressive Shrinking



Gradually allow later layers in each unit
to be skipped to reduce the depth

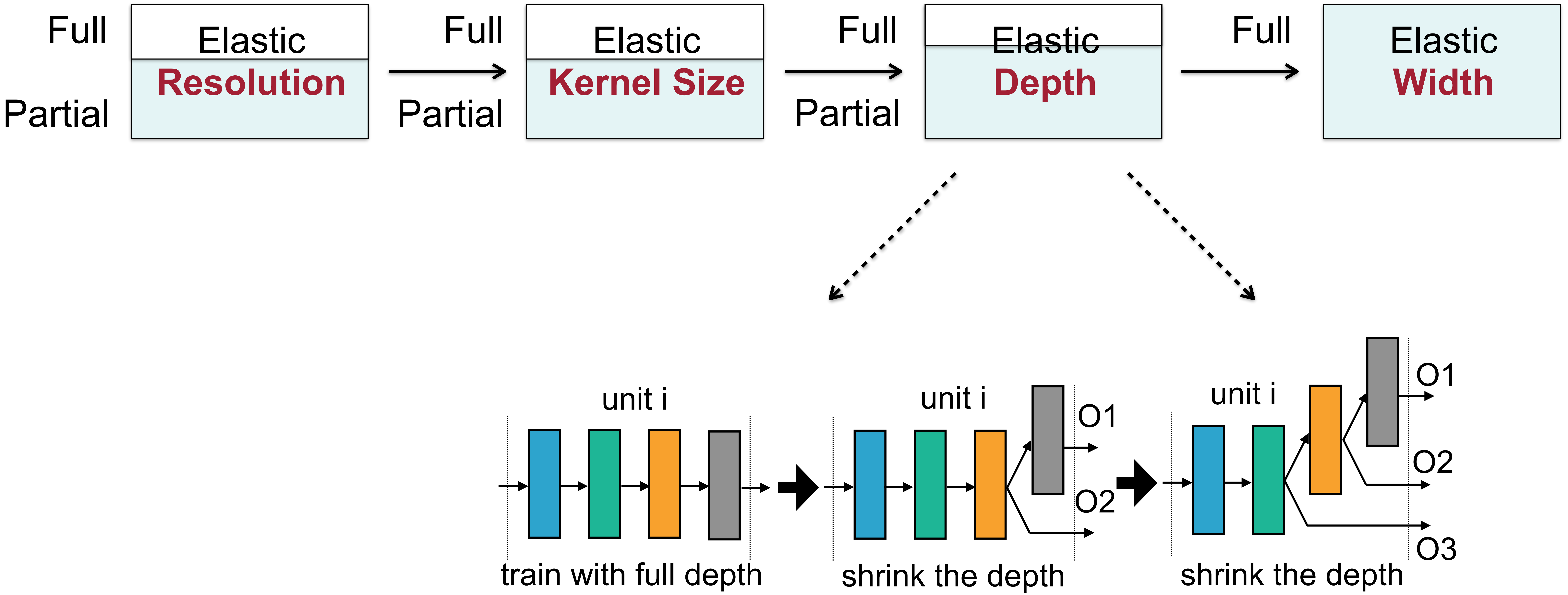
Progressive Shrinking

Full / Partial → **Elastic Resolution** → Full / Partial → **Elastic Kernel Size** → Full / Partial → **Elastic Depth** → Full → **Elastic Width**



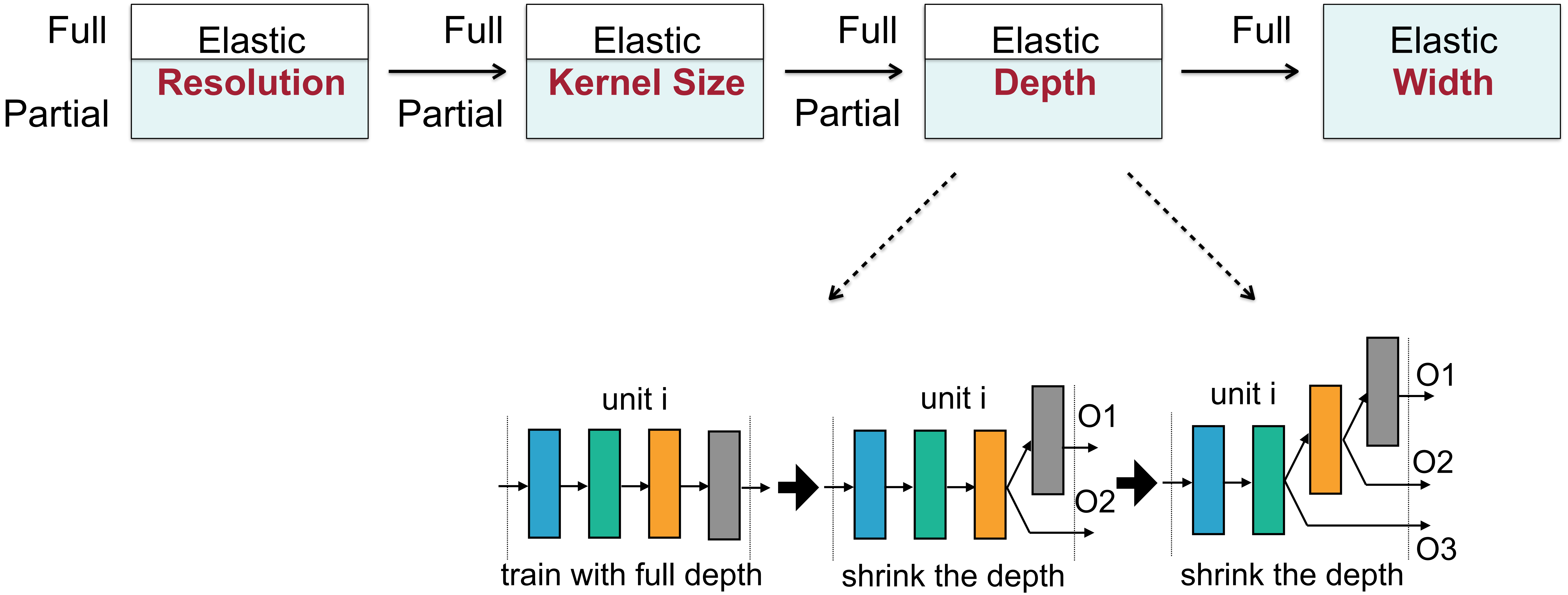
Gradually allow later layers in each unit
to be skipped to reduce the depth

Progressive Shrinking



Gradually allow later layers in each unit
to be skipped to reduce the depth

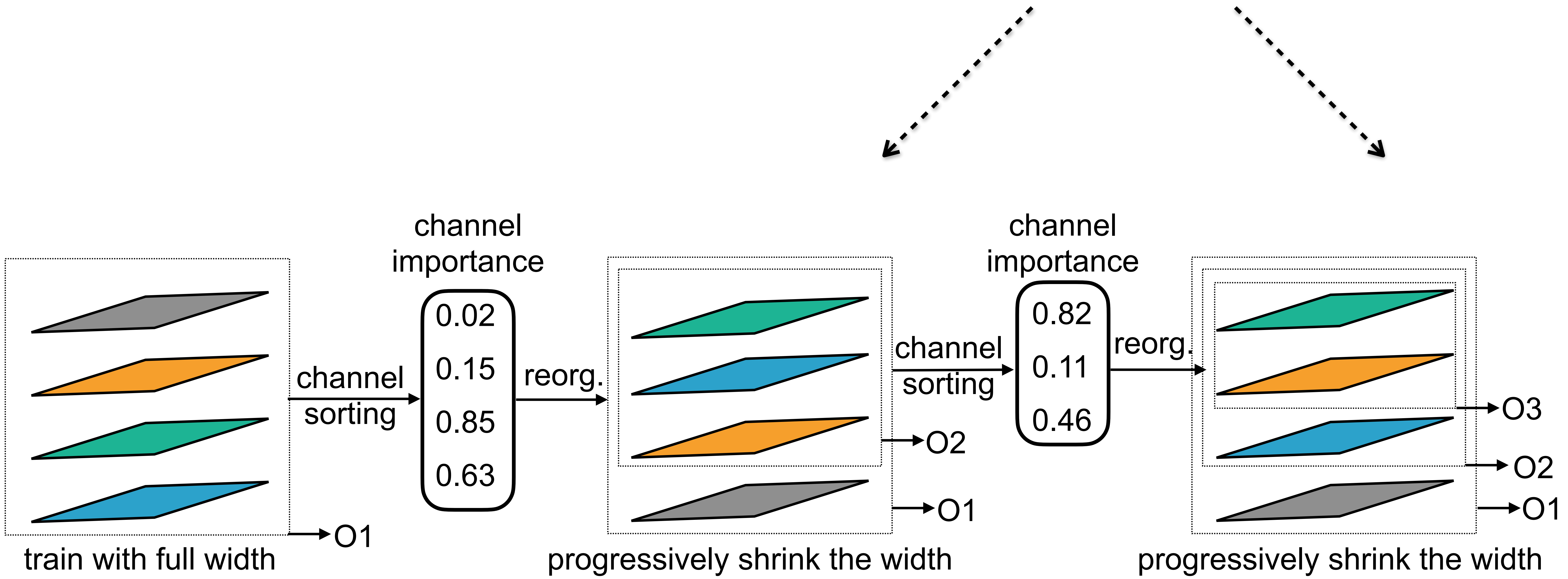
Progressive Shrinking



Gradually allow later layers in each unit
to be skipped to reduce the depth

Progressive Shrinking

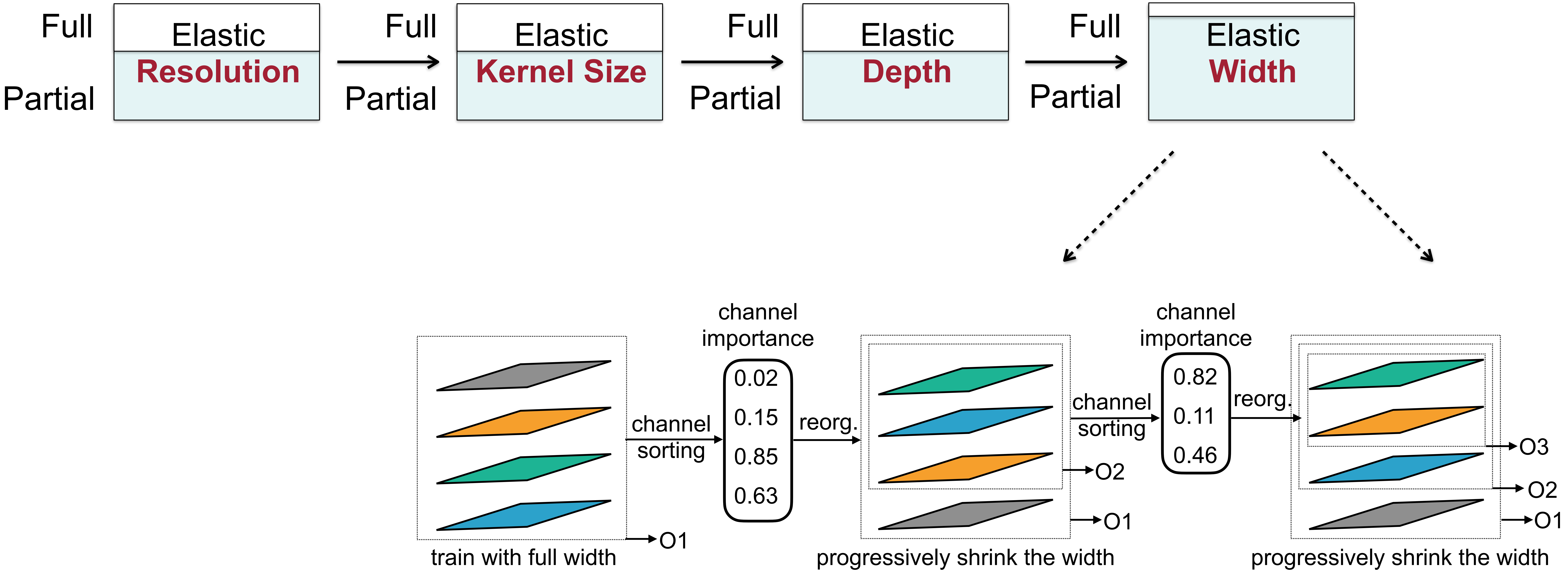
Full / Partial → **Elastic Resolution** → Full / Partial → **Elastic Kernel Size** → Full / Partial → **Elastic Depth** → Full / Partial → **Elastic Width**



Gradually shrink the width

Keep the most important channels when shrinking via channel sorting

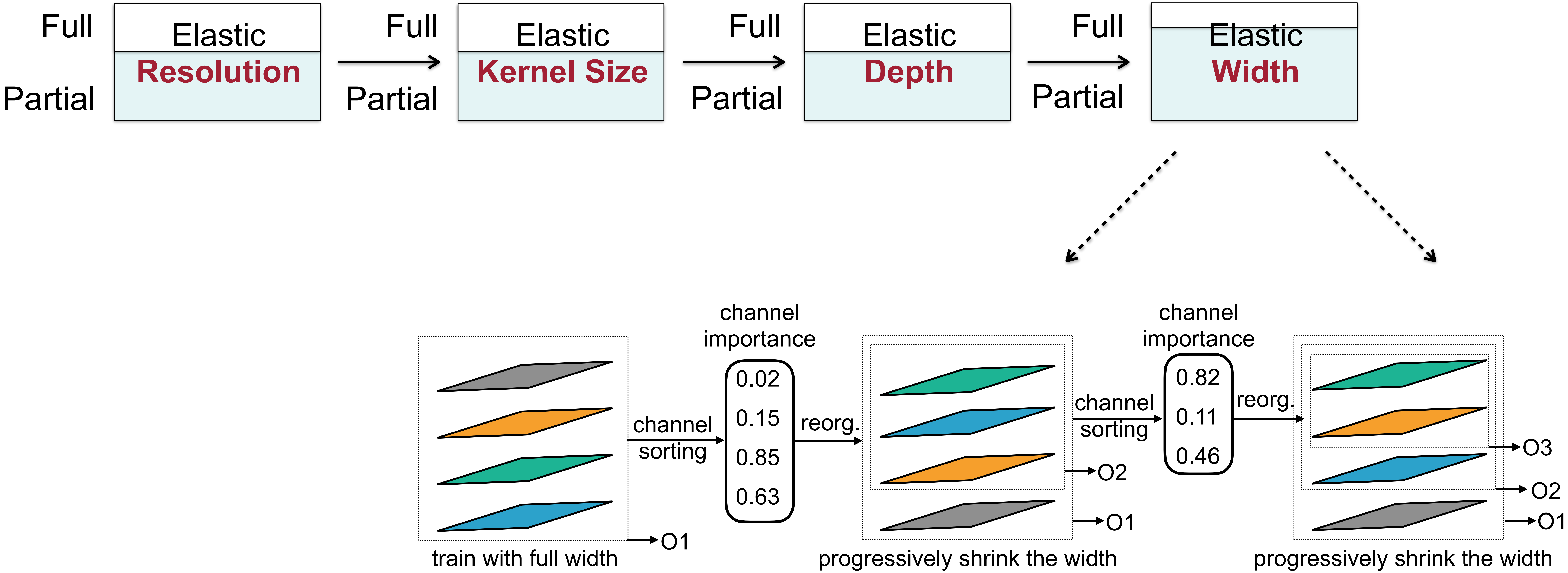
Progressive Shrinking



Gradually shrink the width

Keep the most important channels when shrinking via channel sorting

Progressive Shrinking

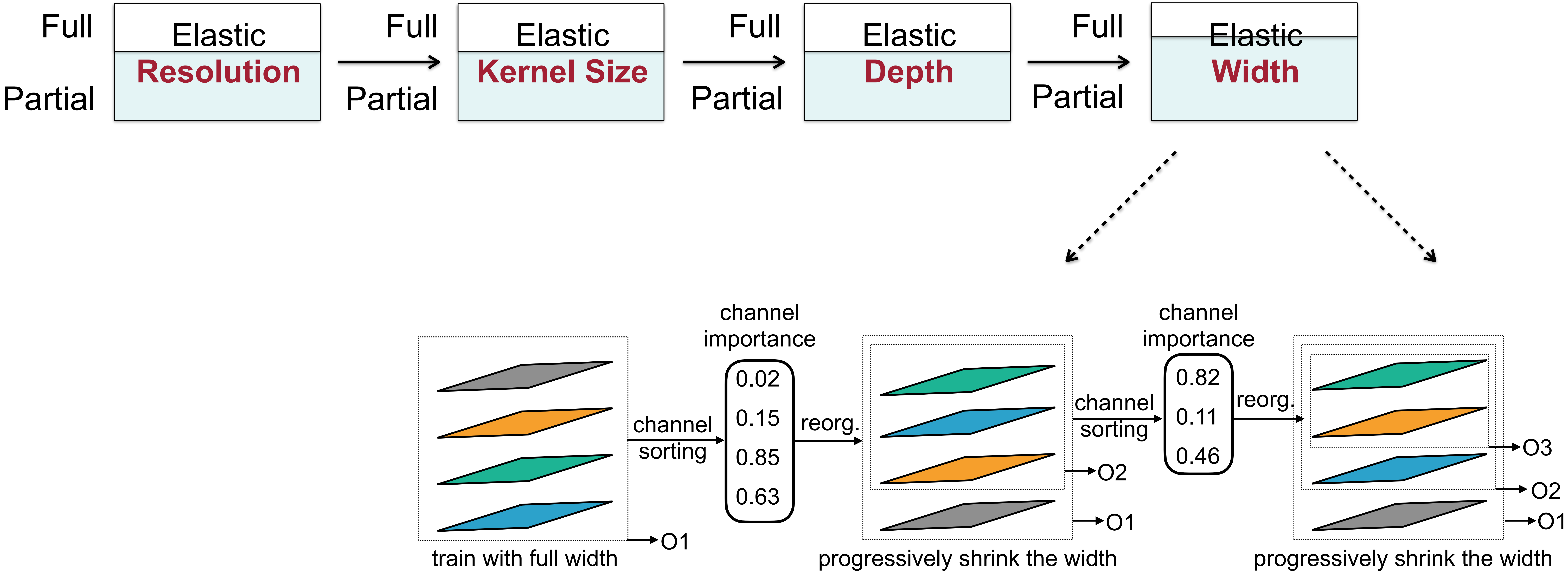


Gradually shrink the width

Keep the most important channels when shrinking via channel sorting

[Once-for-All](), **ICLR'20**

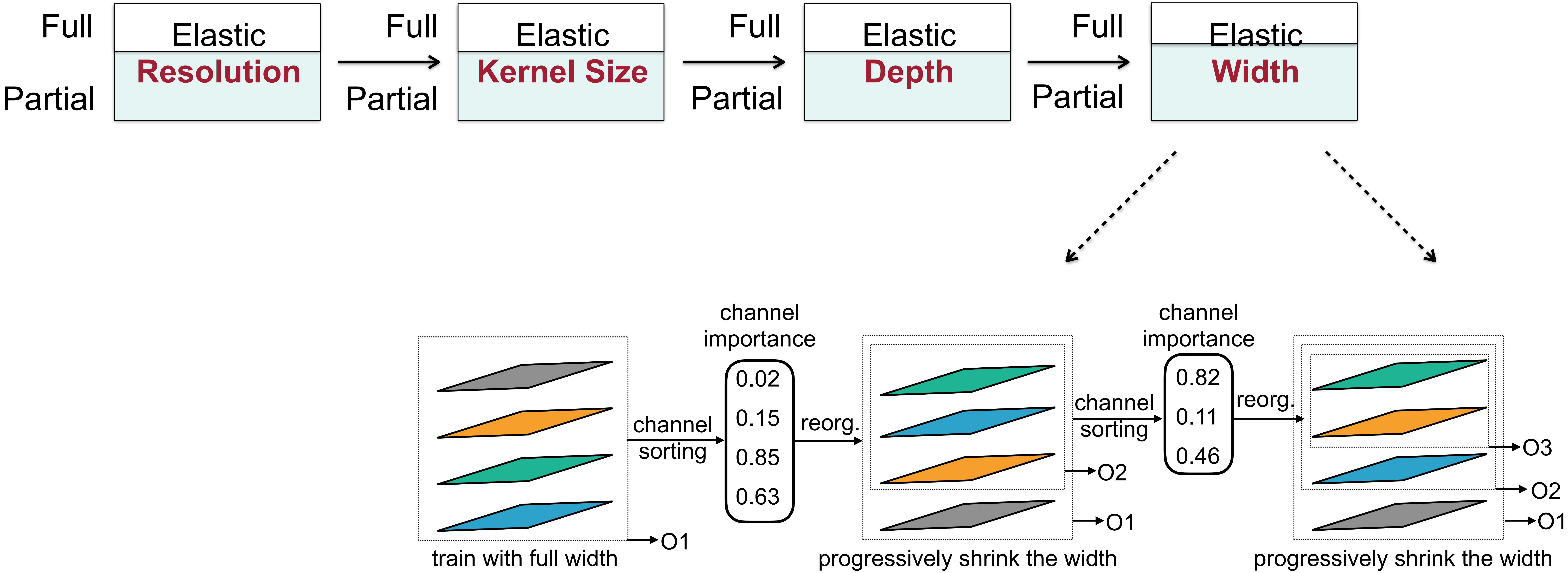
Progressive Shrinking



Gradually shrink the width

Keep the most important channels when shrinking via channel sorting

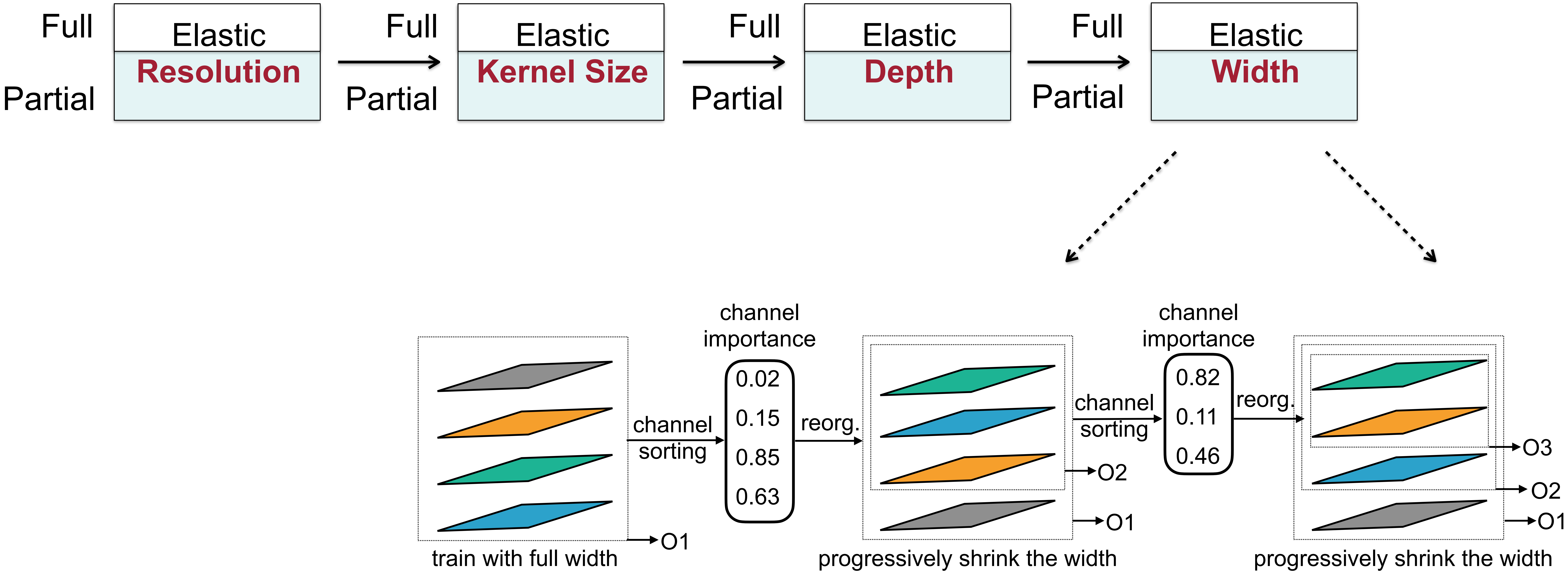
Progressive Shrinking



Gradually shrink the width

Keep the most important channels when shrinking via channel sorting

Progressive Shrinking

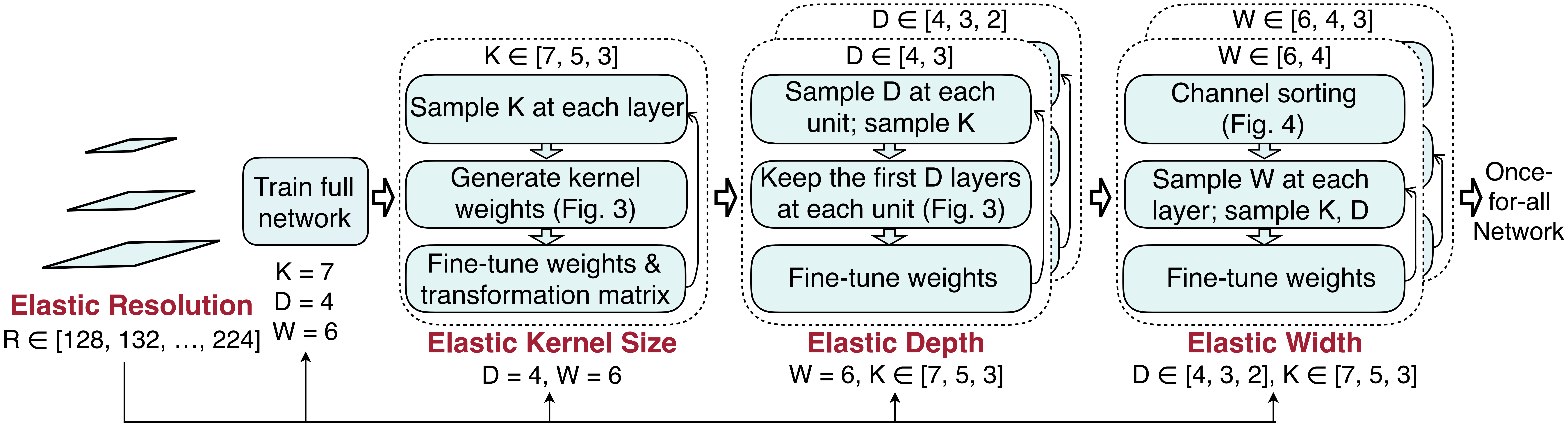


Gradually shrink the width

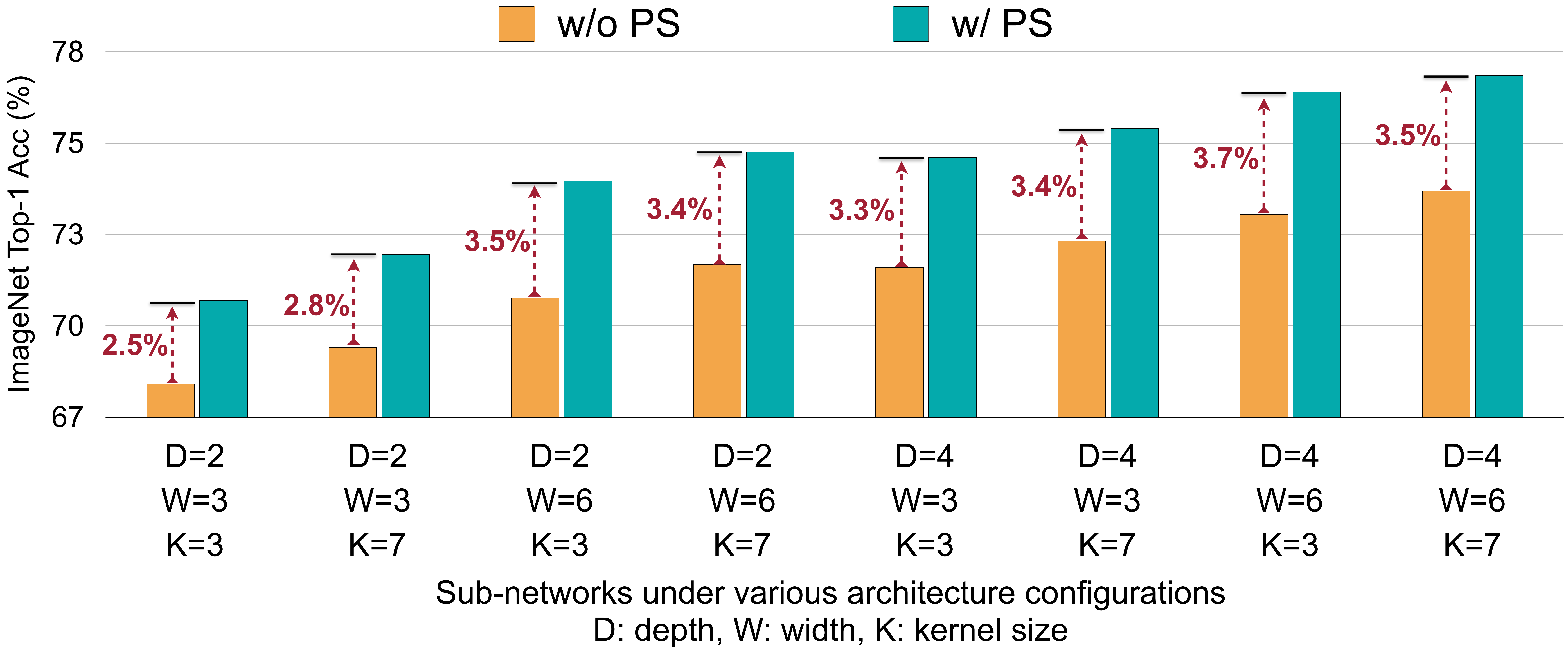
Keep the most important channels when shrinking via channel sorting

Progressive Shrinking

put it together:



Performances of Sub-networks on ImageNet



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

How about search?

training

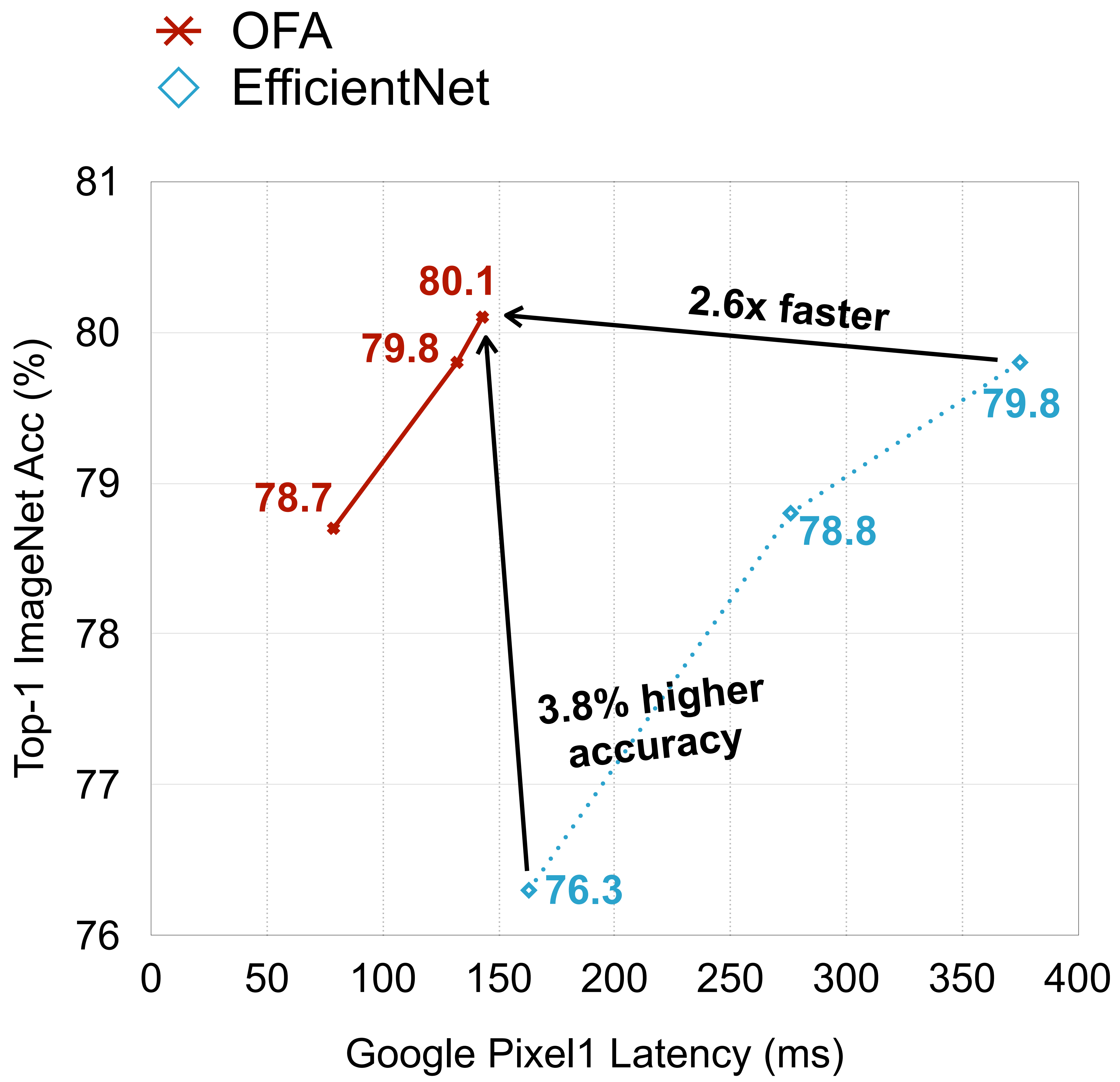
decouple

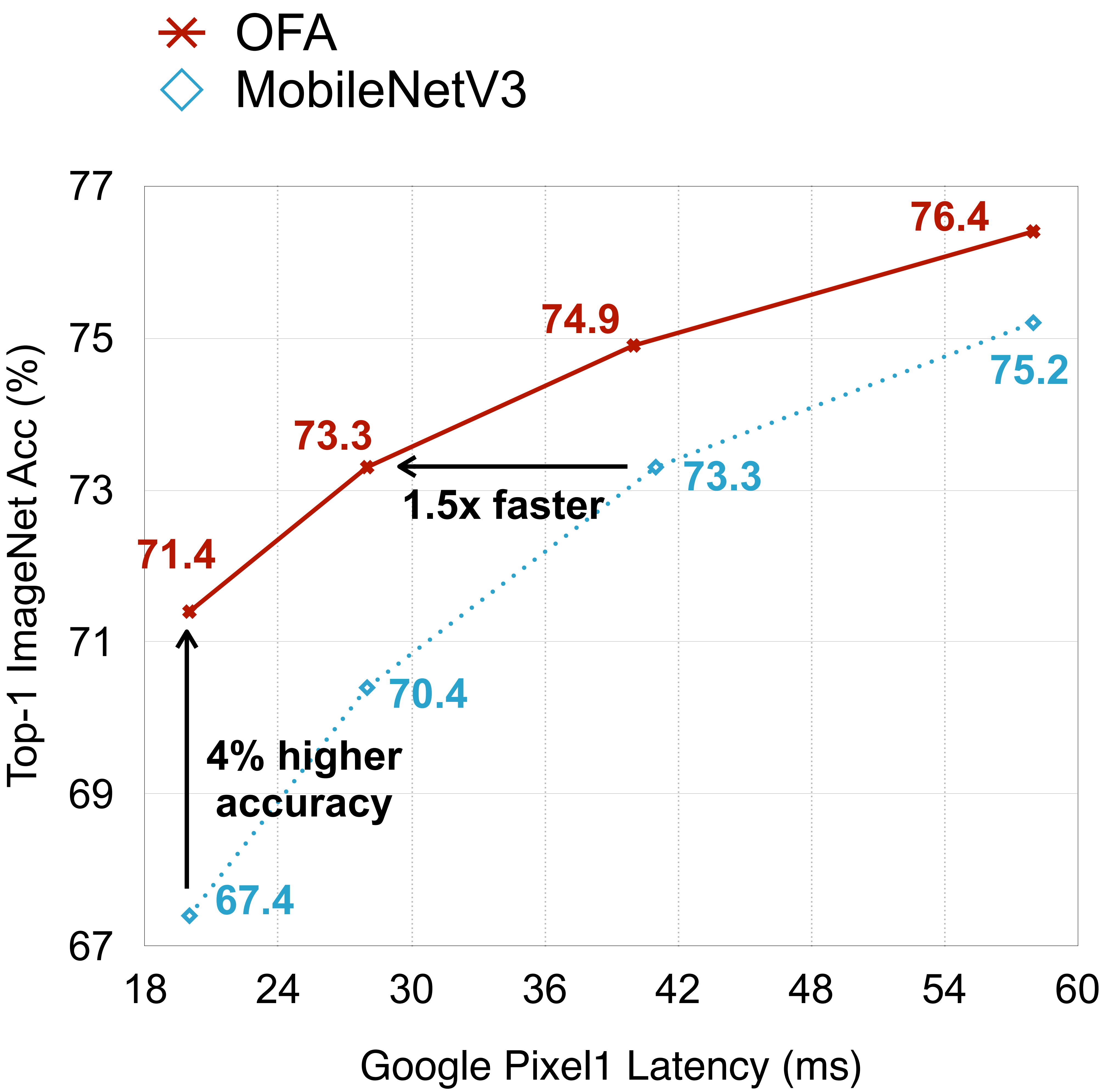
search

```
for OFA training iterations:
    forward-backward();

for devices:
    for search episodes:
        sample from OFA;  //with evolution
        if good_model: break;
    direct deploy without training;
```

2.6x faster than EfficientNet
1.5x faster than MobileNetV3

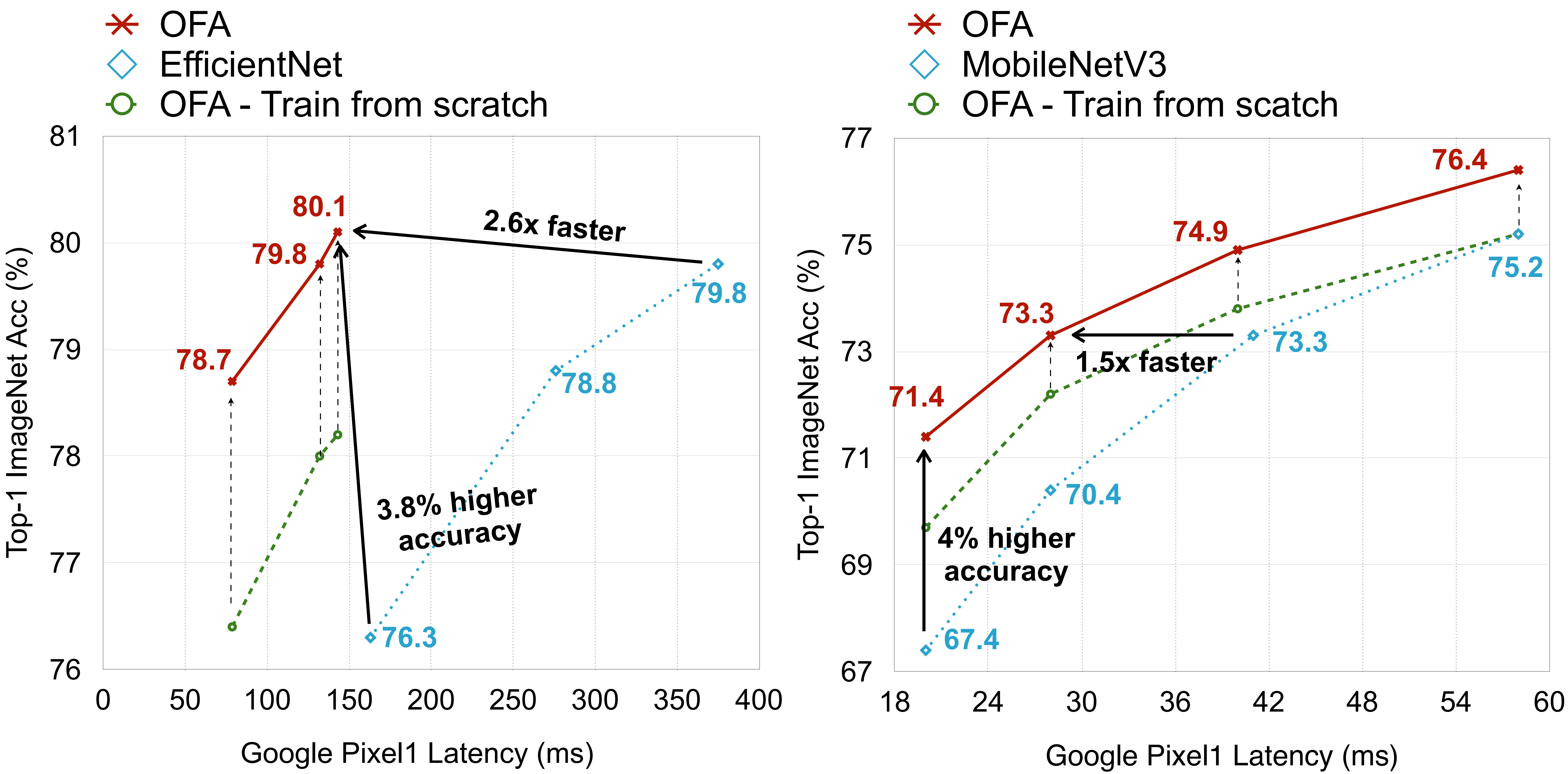




- Training from scratch cannot achieve the same level of accuracy

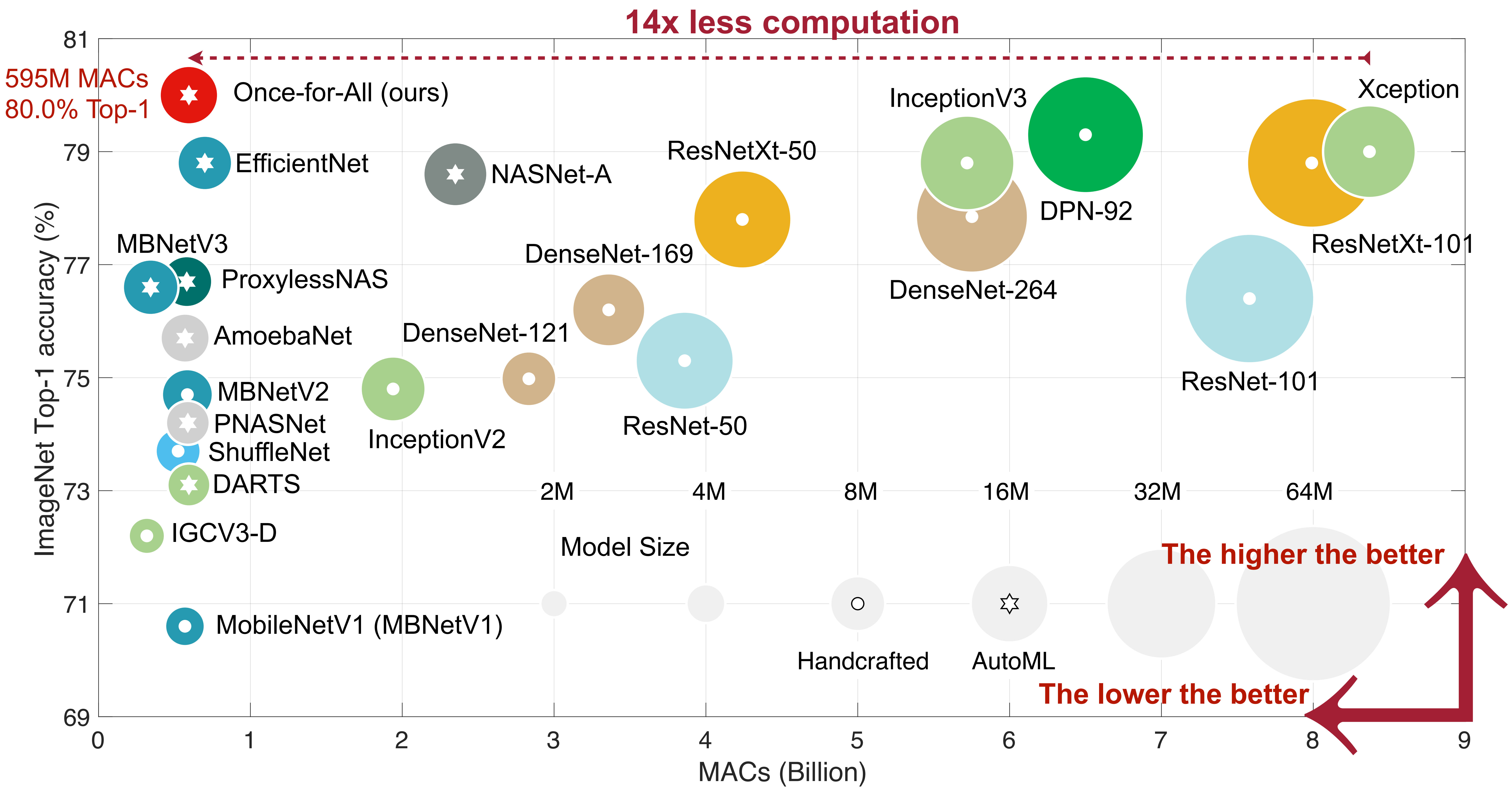
Once-for-All, ICLR'20

More accurate than training from scratch



- Training from scratch cannot achieve the same level of accuracy

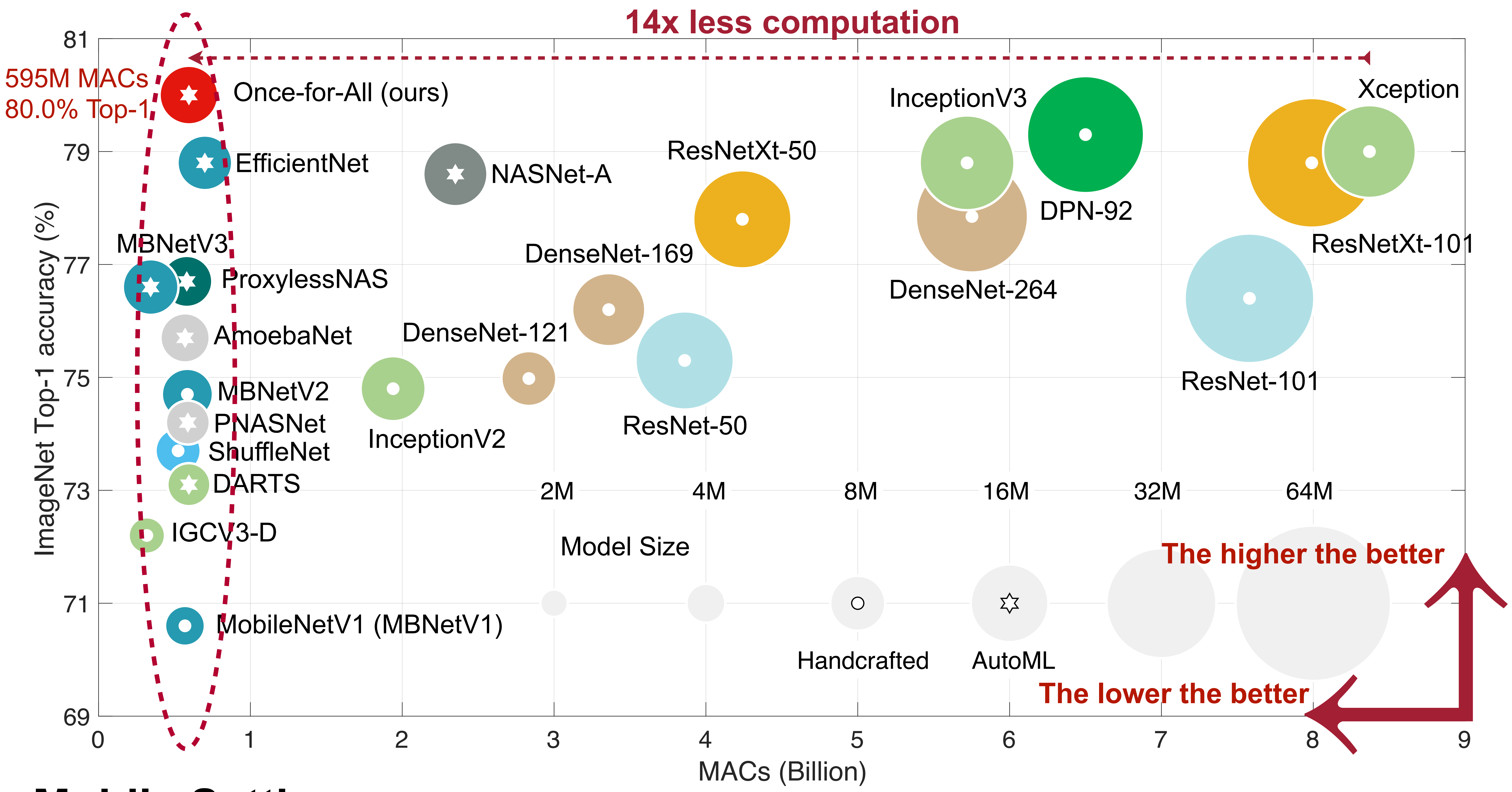
OFA: 80% Top-1 Accuracy on ImageNet



- Once-for-all sets a new state-of-the-art **80% ImageNet top-1 accuracy** under the mobile vision setting (< 600M MACs).

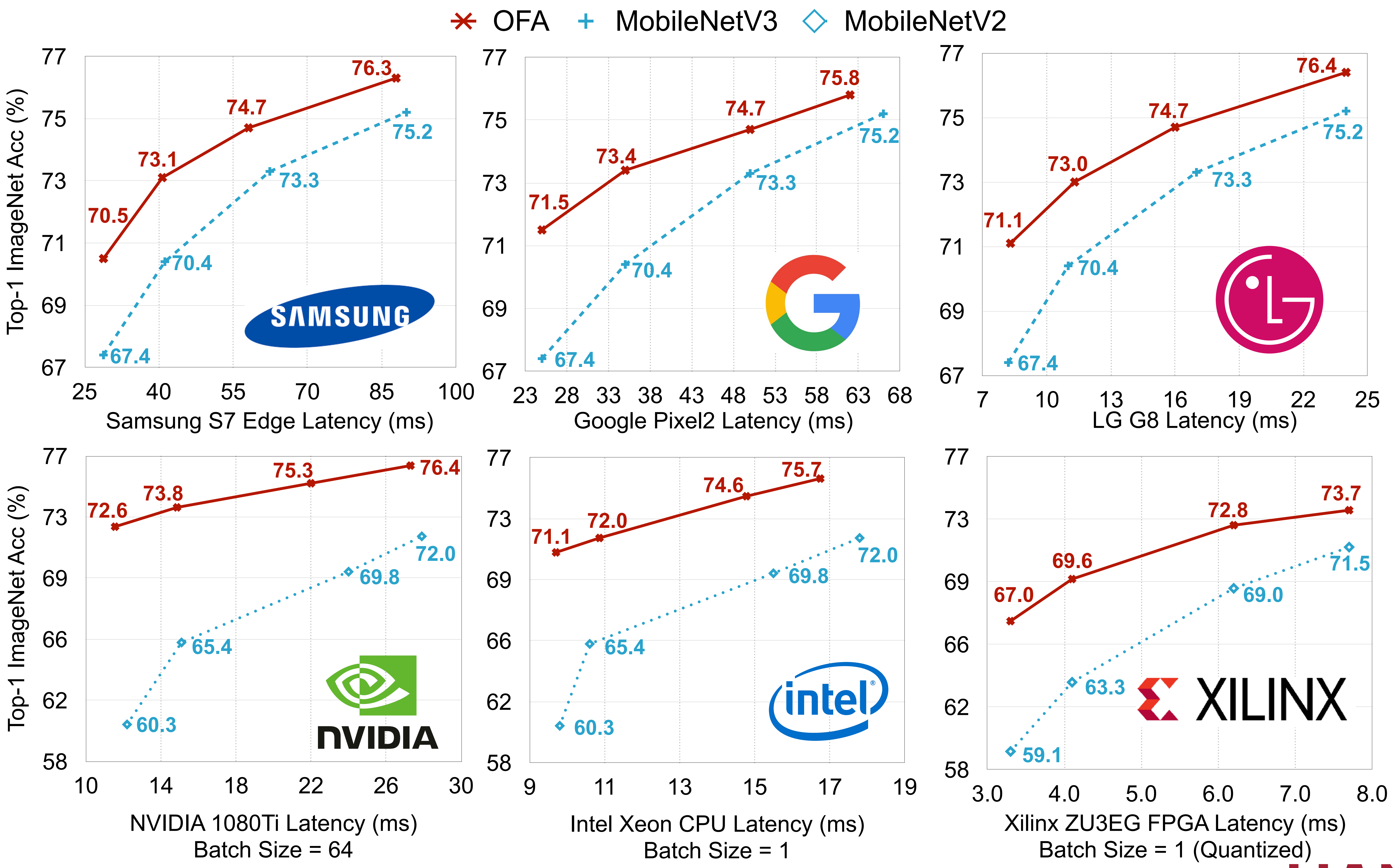
[Once-for-All](), **ICLR'20**

OFA: 80% Top-1 Accuracy on ImageNet



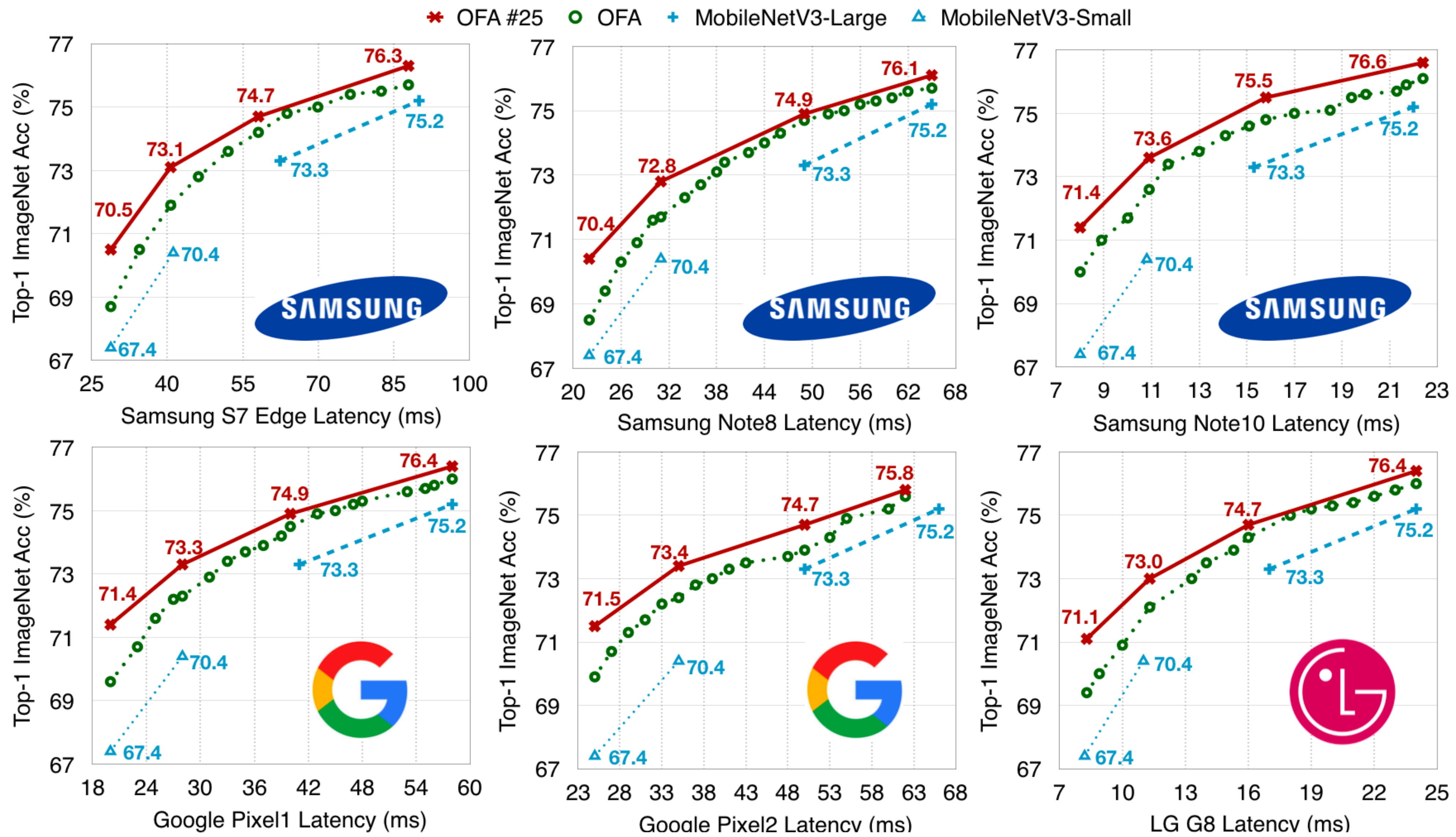
Mobile Setting

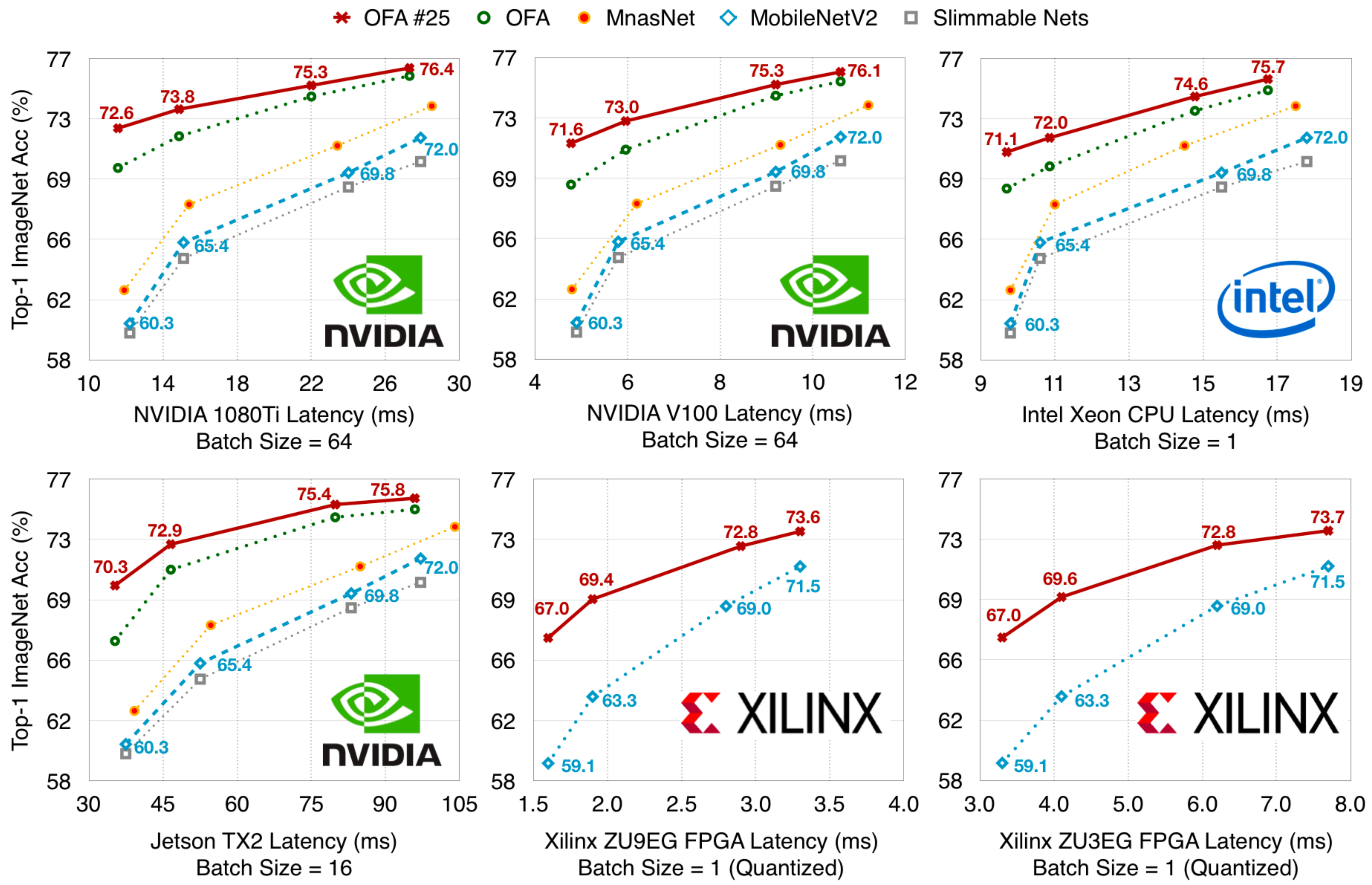
OFA Enables Fast Specialization on Diverse Hardware Platforms



Once-for-All, ICLR'20

Diverse Hardware Platforms, 50+ Pretriained Models are Released





OFA based on FLOPs

- flops@595M_top1@80.0_finetune@75
- flops@482M_top1@79.6_finetune@75
- flops@389M_top1@79.1_finetune@75

OFA for Mobile Phones

LG G8

- LG-G8_lat@24ms_top1@76.4_finetune@25
- LG-G8_lat@16ms_top1@74.7_finetune@25
- LG-G8_lat@11ms_top1@73.0_finetune@25
- LG-G8_lat@8ms_top1@71.1_finetune@25

Samsung Note8

- note8_lat@65ms_top1@76.1_finetune@25
- note8_lat@49ms_top1@74.9_finetune@25
- note8_lat@31ms_top1@72.8_finetune@25
- note8_lat@22ms_top1@70.4_finetune@25

Google Pixel1

- pixel1_lat@143ms_top1@80.1_finetune@75
- pixel1_lat@132ms_top1@79.8_finetune@75
- pixel1_lat@79ms_top1@78.7_finetune@75
- pixel1_lat@58ms_top1@76.9_finetune@75
- pixel1_lat@40ms_top1@74.9_finetune@25
- pixel1_lat@28ms_top1@73.3_finetune@25
- pixel1_lat@20ms_top1@71.4_finetune@25

Samsung Note10

- note10_lat@64ms_top1@80.2_finetune@75
- note10_lat@50ms_top1@79.7_finetune@75
- note10_lat@41ms_top1@79.3_finetune@75
- note10_lat@30ms_top1@78.4_finetune@75
- note10_lat@22ms_top1@76.6_finetune@25
- note10_lat@16ms_top1@75.5_finetune@25
- note10_lat@11ms_top1@73.6_finetune@25
- note10_lat@8ms_top1@71.4_finetune@25

Google Pixel2

- pixel2_lat@62ms_top1@75.8_finetune@25
- pixel2_lat@50ms_top1@74.7_finetune@25
- pixel2_lat@35ms_top1@73.4_finetune@25
- pixel2_lat@25ms_top1@71.5_finetune@25

Samsung S7 Edge

- s7edge_lat@88ms_top1@76.3_finetune@25
- s7edge_lat@58ms_top1@74.7_finetune@25
- s7edge_lat@41ms_top1@73.1_finetune@25
- s7edge_lat@29ms_top1@70.5_finetune@25

OFA for Desktop (CPUs and GPUs)

1080ti GPU (Batch Size 64)

- 1080ti_gpu64@27ms_top1@76.4_finetune@25
- 1080ti_gpu64@22ms_top1@75.3_finetune@25
- 1080ti_gpu64@15ms_top1@73.8_finetune@25
- 1080ti_gpu64@12ms_top1@72.6_finetune@25

V100 GPU (Batch Size 64)

- v100_gpu64@11ms_top1@76.1_finetune@25
- v100_gpu64@9ms_top1@75.3_finetune@25
- v100_gpu64@6ms_top1@73.0_finetune@25
- v100_gpu64@5ms_top1@71.6_finetune@25

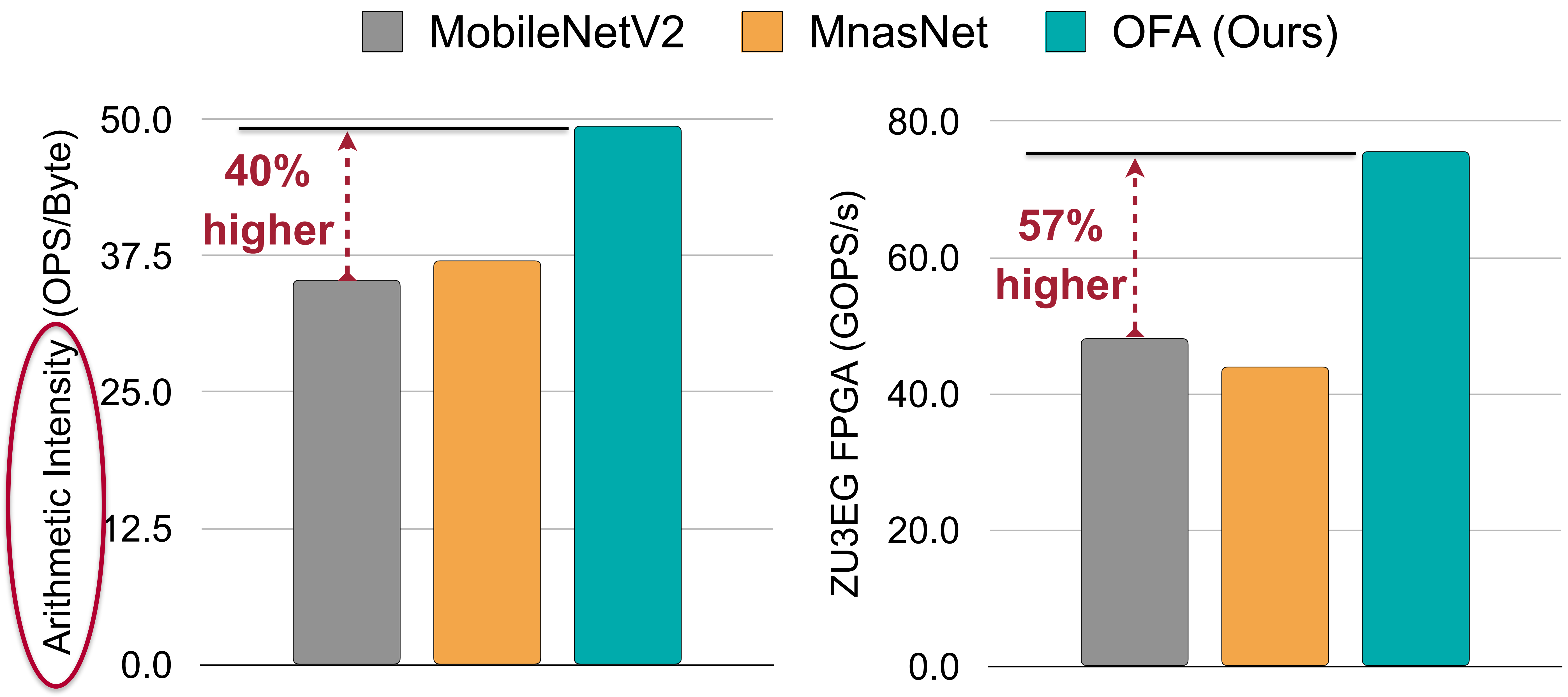
Jetson TX2 GPU (Batch Size 16)

- tx2_gpu16@96ms_top1@75.8_finetune@25
- tx2_gpu16@80ms_top1@75.4_finetune@25
- tx2_gpu16@47ms_top1@72.9_finetune@25
- tx2_gpu16@35ms_top1@70.3_finetune@25

Intel Xeon CPU with MKL-DNN (Batch Size 1)

- cpu_lat@17ms_top1@75.7_finetune@25
- cpu_lat@15ms_top1@74.6_finetune@25
- cpu_lat@11ms_top1@72.0_finetune@25
- cpu_lat@10ms_top1@71.1_finetune@25

OFA for FPGA Accelerators



Measured results on XILINX FPGA

- Non-specialized neural networks do not fully utilize the hardware resource. There is a large room for improvement via neural network specialization.

We need Green AI
Solve the Environmental Problem of NAS

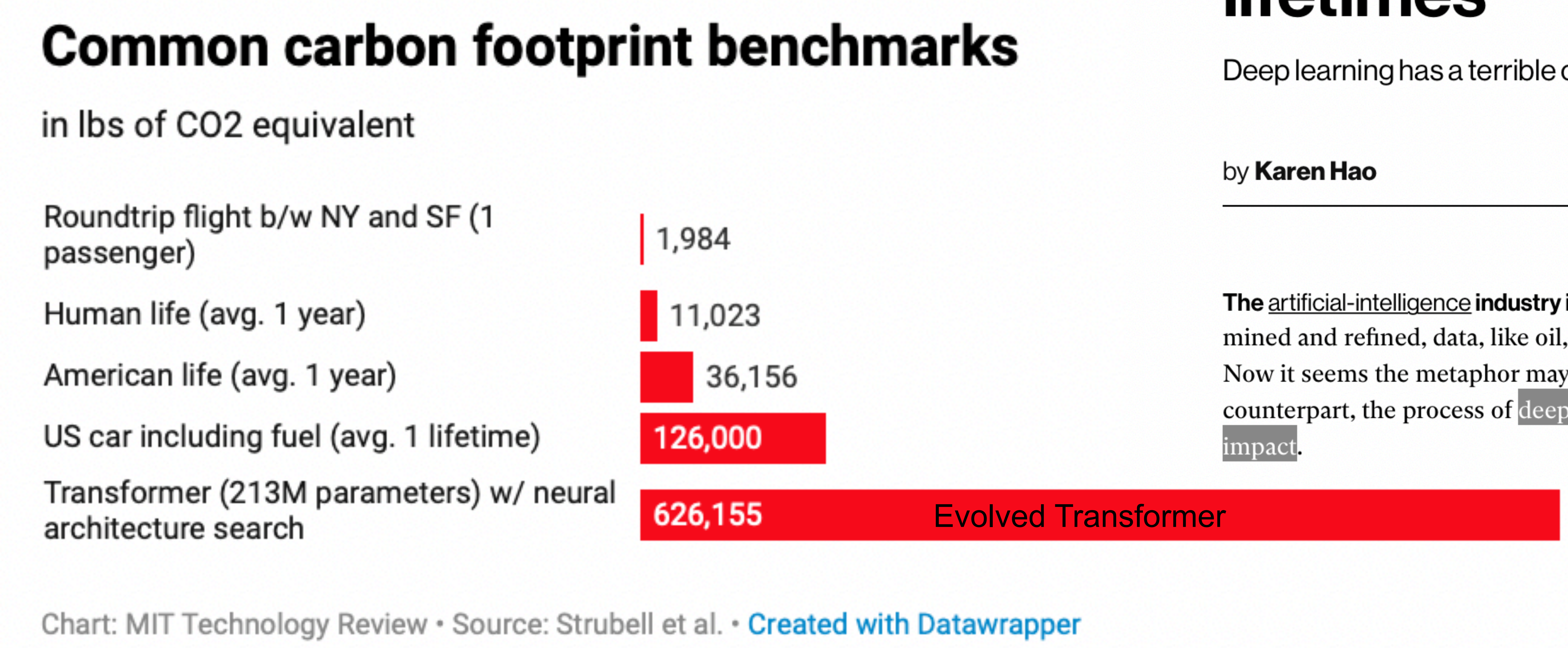
Artificial intelligence / Machine learning

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

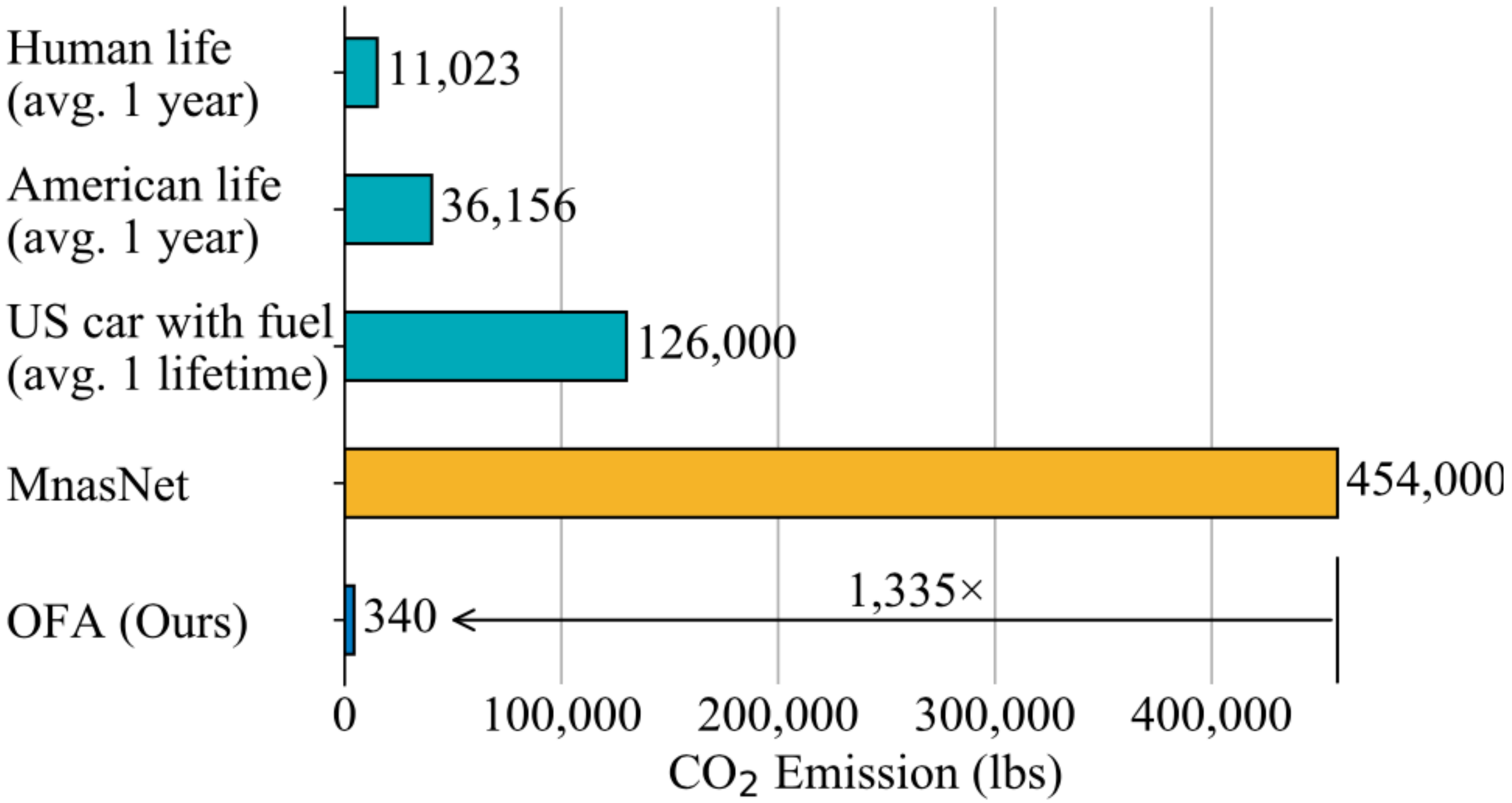
Deep learning has a terrible carbon footprint.

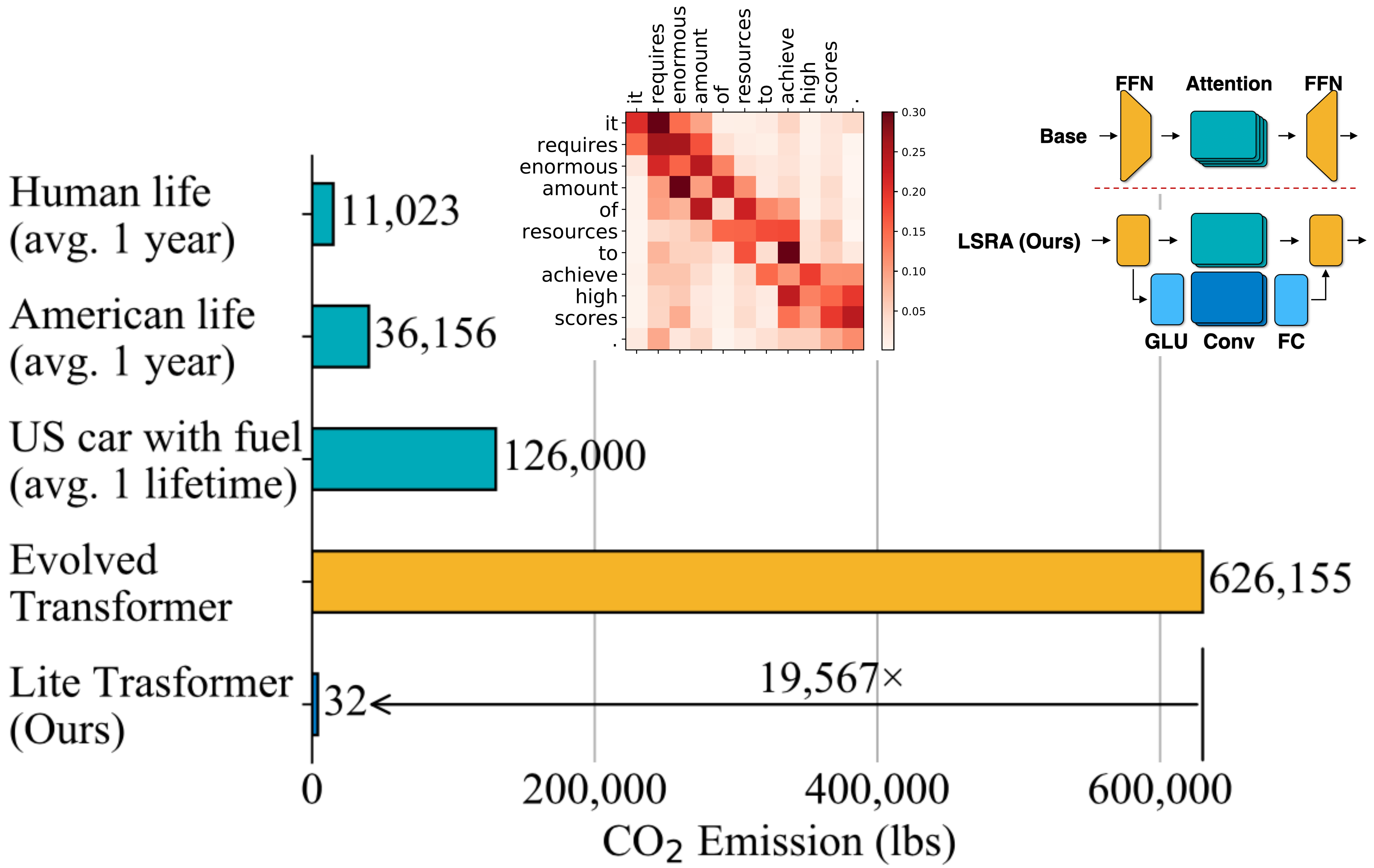
by **Karen Hao** June 6, 2019

The artificial-intelligence **industry is often compared to the oil industry: once** mined and refined, data, like oil, can be a highly lucrative commodity. Now it seems the metaphor may extend even further. Like its fossil-fuel counterpart, the process of deep learning has an outsize environmental impact.



How to save CO_2 emission





1. Once for all: **Amortize** the search cost across **many** sub-networks and deployment scenarios

2. Lite-transformer: **Human-in-the-loop** design. Apply human insights of HW&ML, rather than "**just search it**"

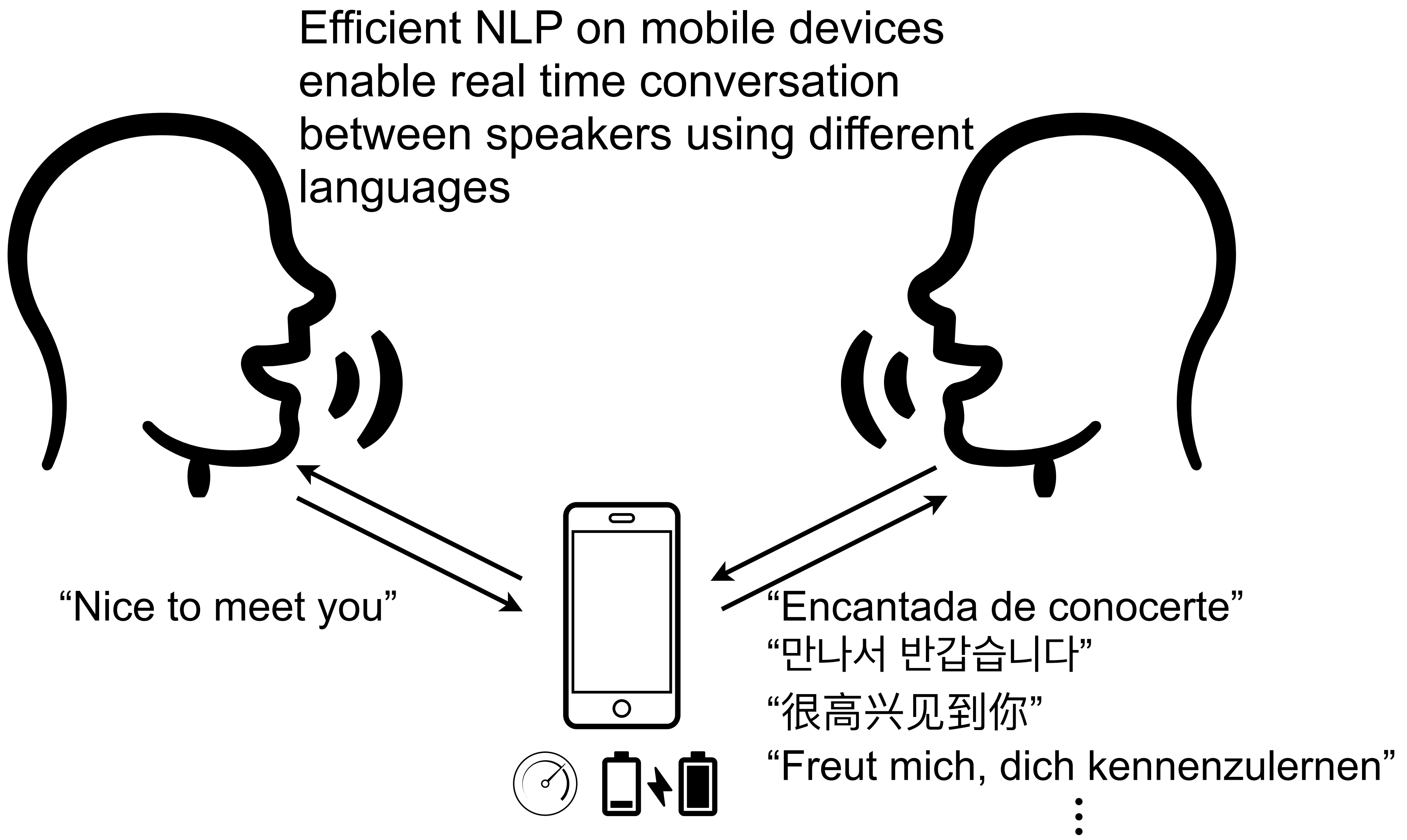
Once-for-All, ICLR'20

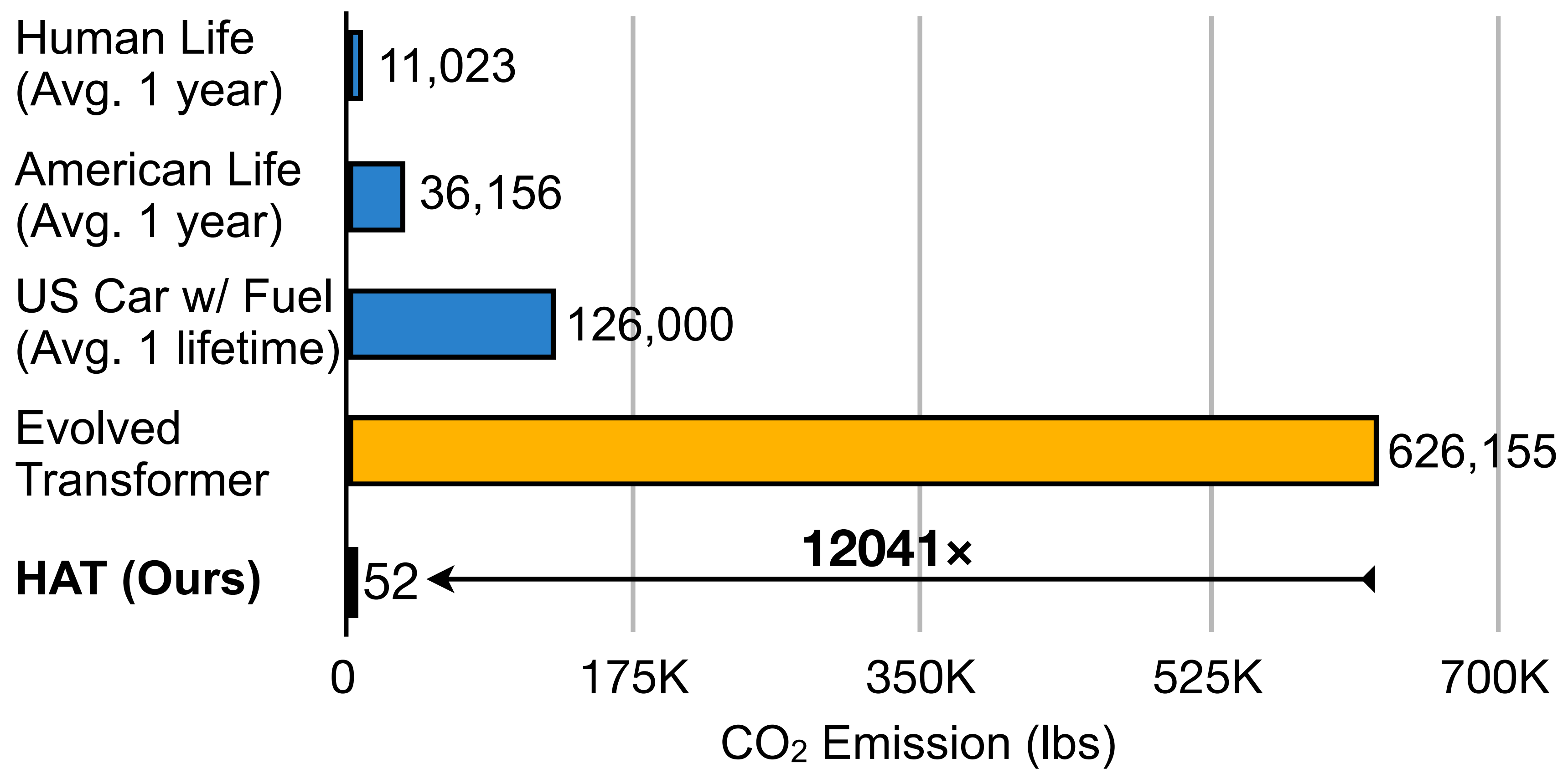
Lite Transformer, ICLR'20

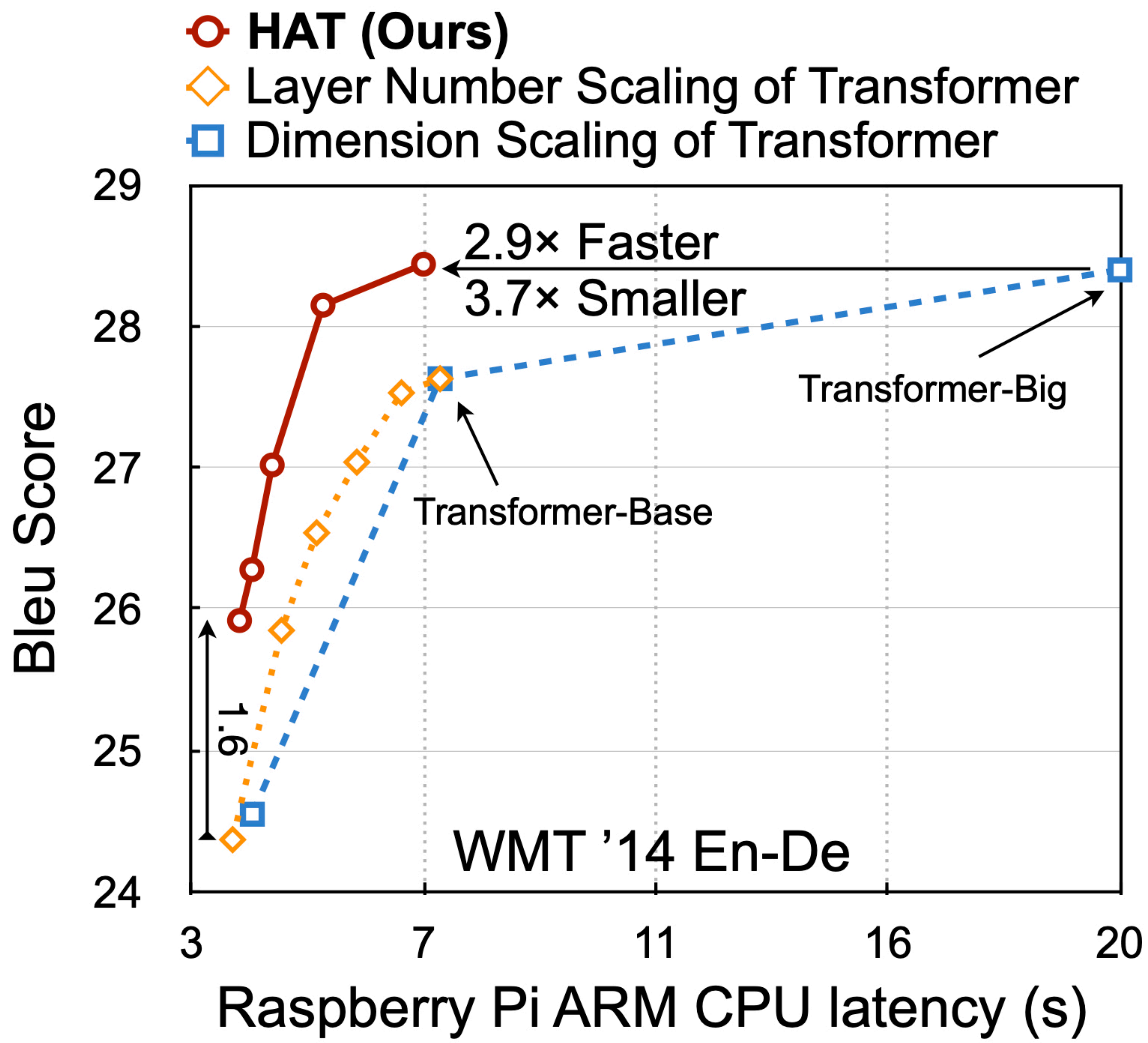
OFA has broad applications

- Efficient Transformer
- Efficient Video Recognition
- Efficient 3D Vision
- Efficient GAN Compression

OFA's Application: Hardware-Aware Transformer

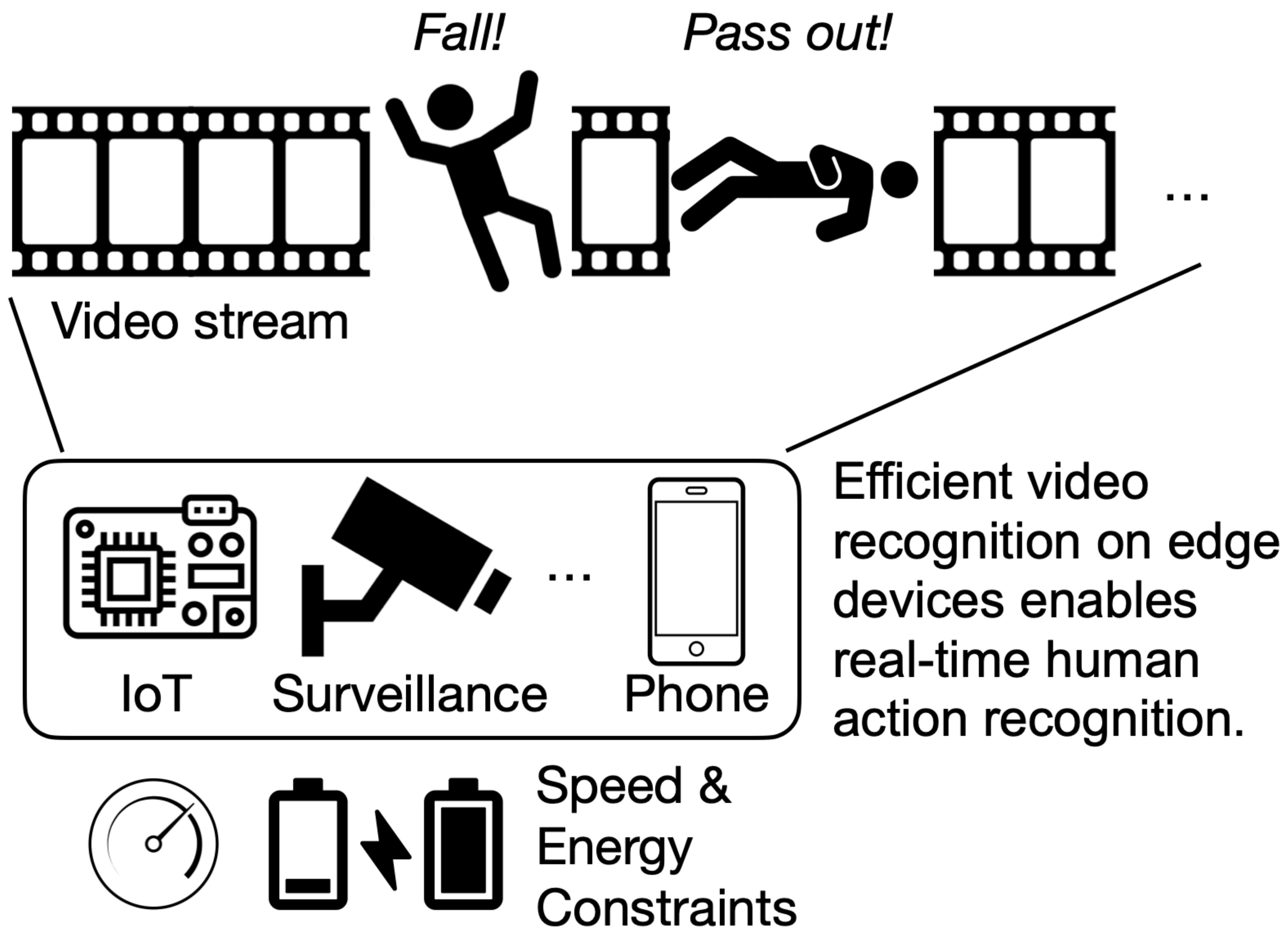


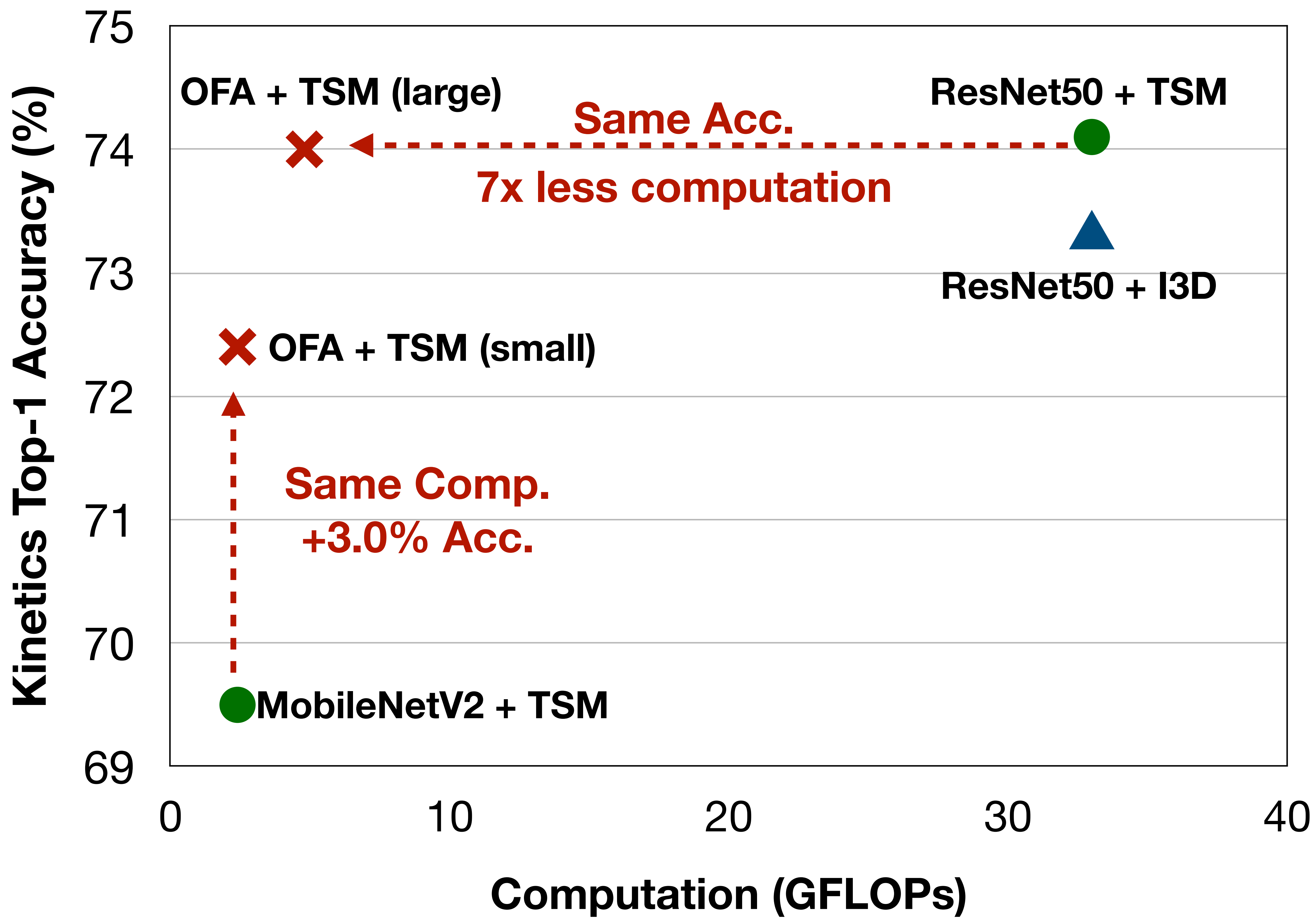




3.7x smaller model size, same performance on WMT'14 En-De;
3x, 1.6x, 1.5x faster on Raspberry Pi, CPU, GPU than Transformer Baseline
12,000x less CO_2 than evolved transformer

OFA's Application: Efficient Video Recognition



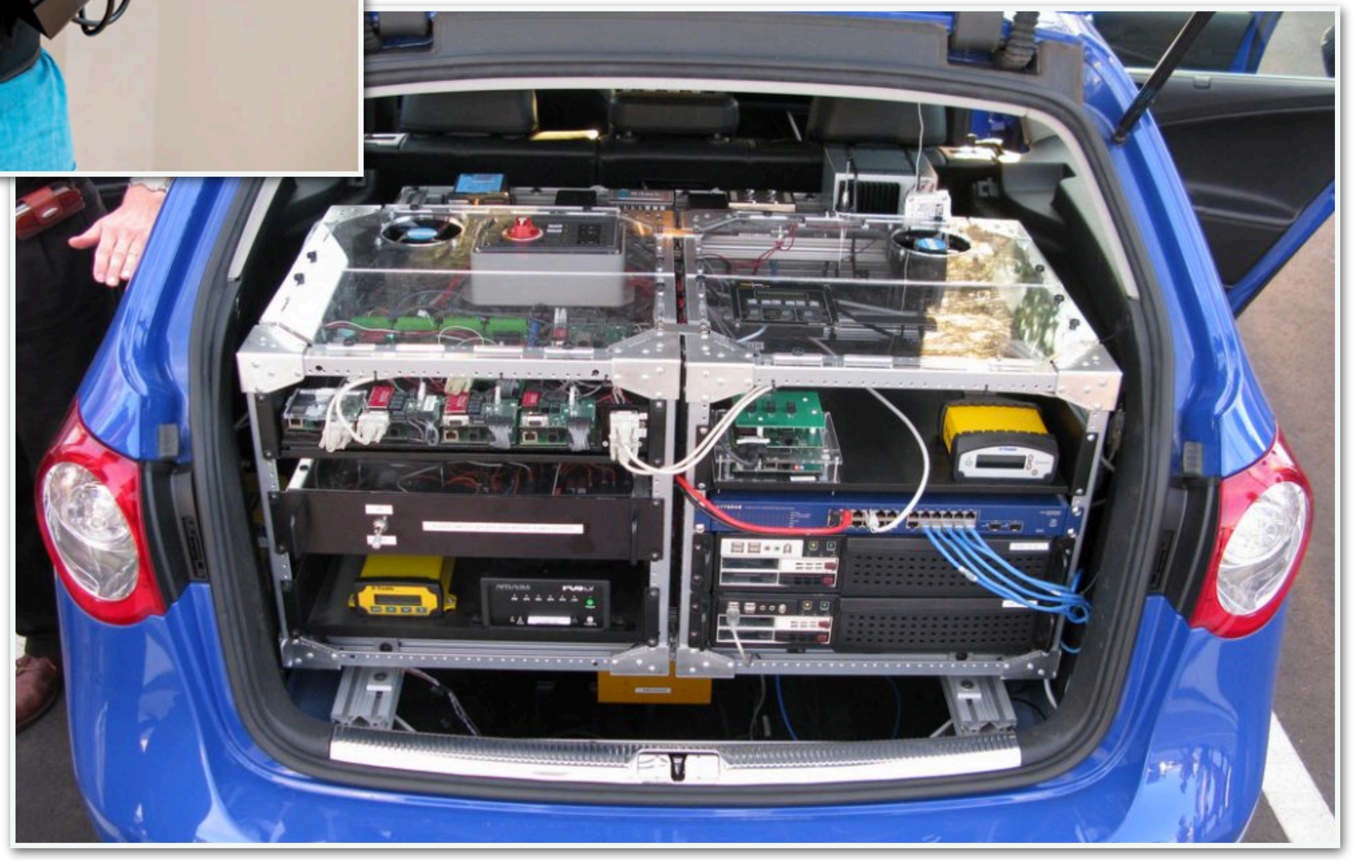


7x less computation, same performance as TSM+ResNet50
same computation, **3%** higher accuracy than TSM+MobileNet-v2

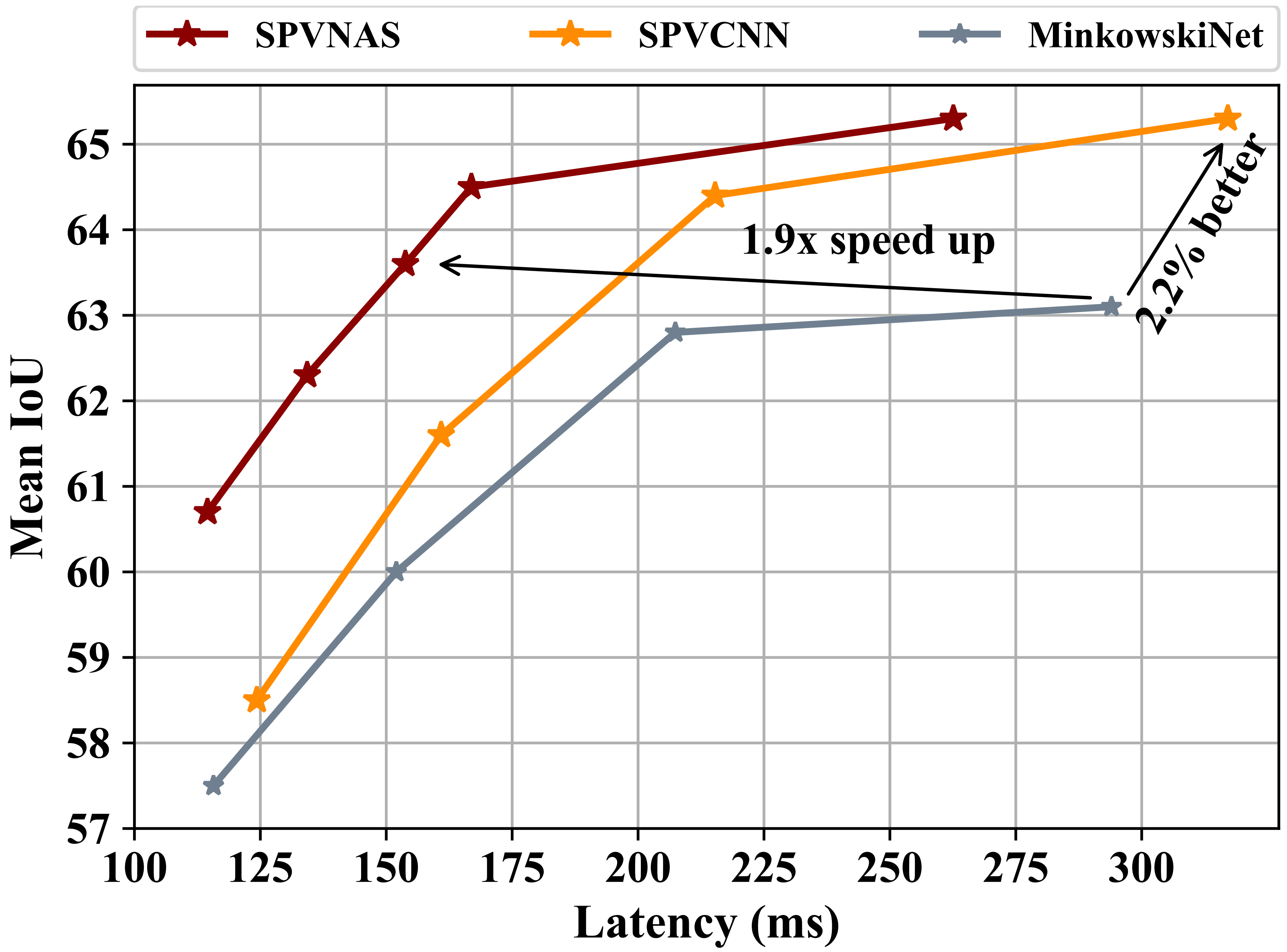
TSM, ICCV'19

OFA's Application: Efficient 3D Recognition

AR/VR: a whole backpack of computer



self-driving: a whole trunk of GPU



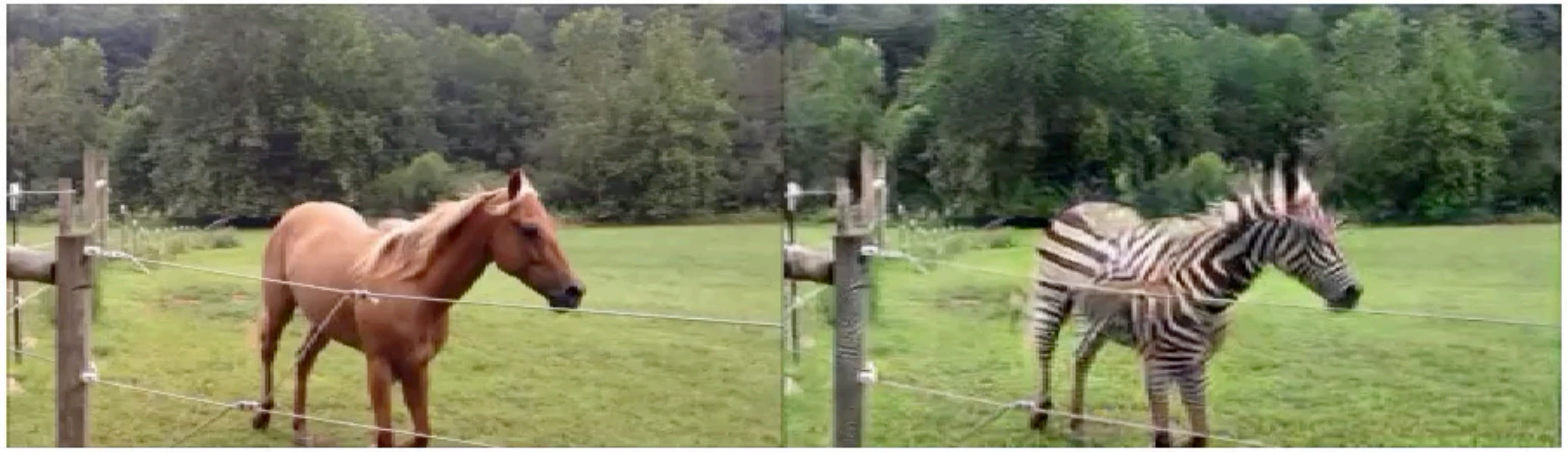
Accuracy *v.s.* **Latency** Tradeoff

4x FLOPs reduction and **2x** speedup over MinkowskiNet
3.6% better accuracy under the same computation budget.

followup of PVCNN, NeurIPS'19 (spotlight)

OFA's Application: GAN Compression

Accelerating Horse2zebra by GAN Compression



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

Measured on NVIDIA **Jetson Xavier GPU**
Lower FID indicates better Performance.

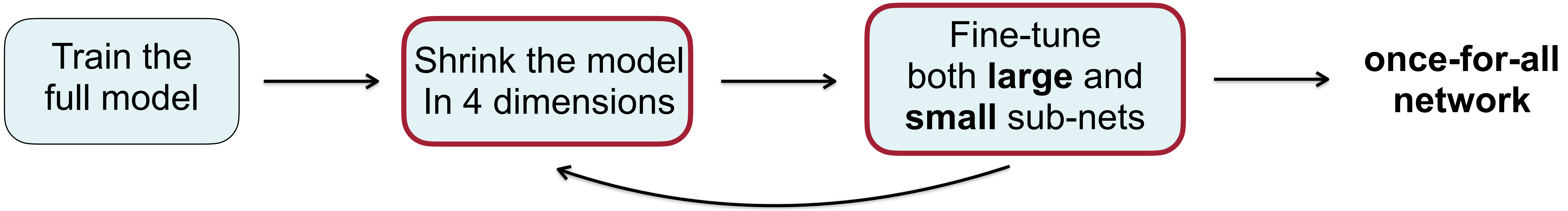
Model		CycleGAN	Pix2pix	GauGAN
Metric	FID (↓)	61.5→65.0	24.2→26.6	–
	mAP (↑)	–	–	58.9 → 58.4
MAC Reduction		21.2×	11.8×	8.8×
Memory Reduction		2.0×	1.7×	1.8×
Xavier Speedup	CPU	1.65s (18.5×)	3.07s (9.9×)	21.2s (7.9×)
	GPU	0.026s (3.1×)	0.035s (2.4×)	0.10s (3.2×)
Nano Speedup	CPU	6.30s (14.0×)	8.57s (10.3×)	65.3s (8.6×)
	GPU	0.16s (4.0×)	0.26s (2.5×)	0.81s (3.3×)
1080Ti Speedup		0.005s (2.5×)	0.007s (1.8×)	0.034s (1.7×)
Xeon Silver 4114 CPU Speedup		0.11s (3.4×)	0.15s (2.6×)	0.74s (2.8×)

8-21x FLOPs reduction on CycleGAN, Pix2pix, GauGAN
1.7x-18.5x speedup on CPU/GPU & Mobile CPU/GPU

Summary: Once-for-All Network

- We introduce once-for-all network for **efficient inference on diverse hardware platforms**.
- We present an effective **progressive shrinking** approach for training once-for-all networks.

Progressive Shrinking



- Once-for-all network **surpasses MobileNetV3 and EfficientNet** by a large margin under all scenarios, setting a new state-of-the-art **80% ImageNet Top1-accuracy** under the mobile setting (**< 600M MACs**).
  - **First place** in the 3rd Low-Power Computer Vision Challenge, DSP track @ICCV'19
  - **First place** in the 4th Low-Power Computer Vision Challenge @NeurIPS'19, both classification & detection.
- Released **50+ different pre-trained OFA models** on diverse hardware platforms (CPU/GPU/FPGA/DSP).

```
net, image_size = ofa_specialized(net_id, pretrained=True)
```

- Released the **training code & pre-trained OFA network** that provides diverse sub-networks without training.

```
ofa_network = ofa_net(net_id, pretrained=True)
```

Project Page: https://ofa.mit.edu

References

Model Compression & NAS

- Once-For-All: Train One Network and Specialize It for Efficient Deployment, ICLR’20
- ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR’19
- APQ: Joint Search for Network Architecture, Pruning and Quantization Policy, CVPR’20
- HAQ: Hardware-Aware Automated Quantization with Mixed Precision, CVPR’19
- Defensive Quantization: When Efficiency Meets Robustness, ICLR’19
- AMC: AutoML for Model Compression and Acceleration on Mobile Devices, ECCV’18

Efficient Vision:

- GAN Compression: Learning Efficient Architectures for Conditional GANs, CVPR’20
- TSM: Temporal Shift Module for Efficient Video Understanding, ICCV’19
- PVCNN: Point Voxel CNN for Efficient 3D Deep Learning, NeurIPS’19

Efficient NLP:

- Lite Transformer with Long Short Term Attention, ICLR’20
- HAT: Hardware-aware Transformer, ACL’20

Hardware & EDA:

- SpArch: Efficient Architecture for Sparse Matrix Multiplication, HPCA’20
- Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning, DAC’20

Make AI Efficient:
Tiny Computational Resources
Tiny Human Resources

MIT HAN LAB
Hardware, AI and Neural-nets

Media Coverage: MIT News · MIT Technology Review · WIRED · engadget · IEEE SPECTRUM

Website: songhan.mit.edu github.com/mit-han-lab youtube.com/c/MITHANLab