Foundations of Edge AI

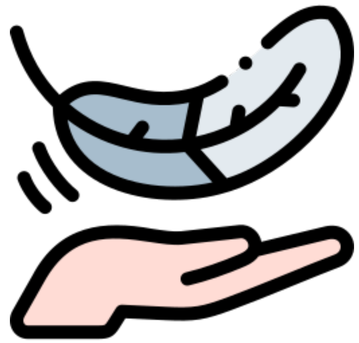
Lecture12

Neural Architecture Search (Cont'd)

Lanyu (Lori) Xu

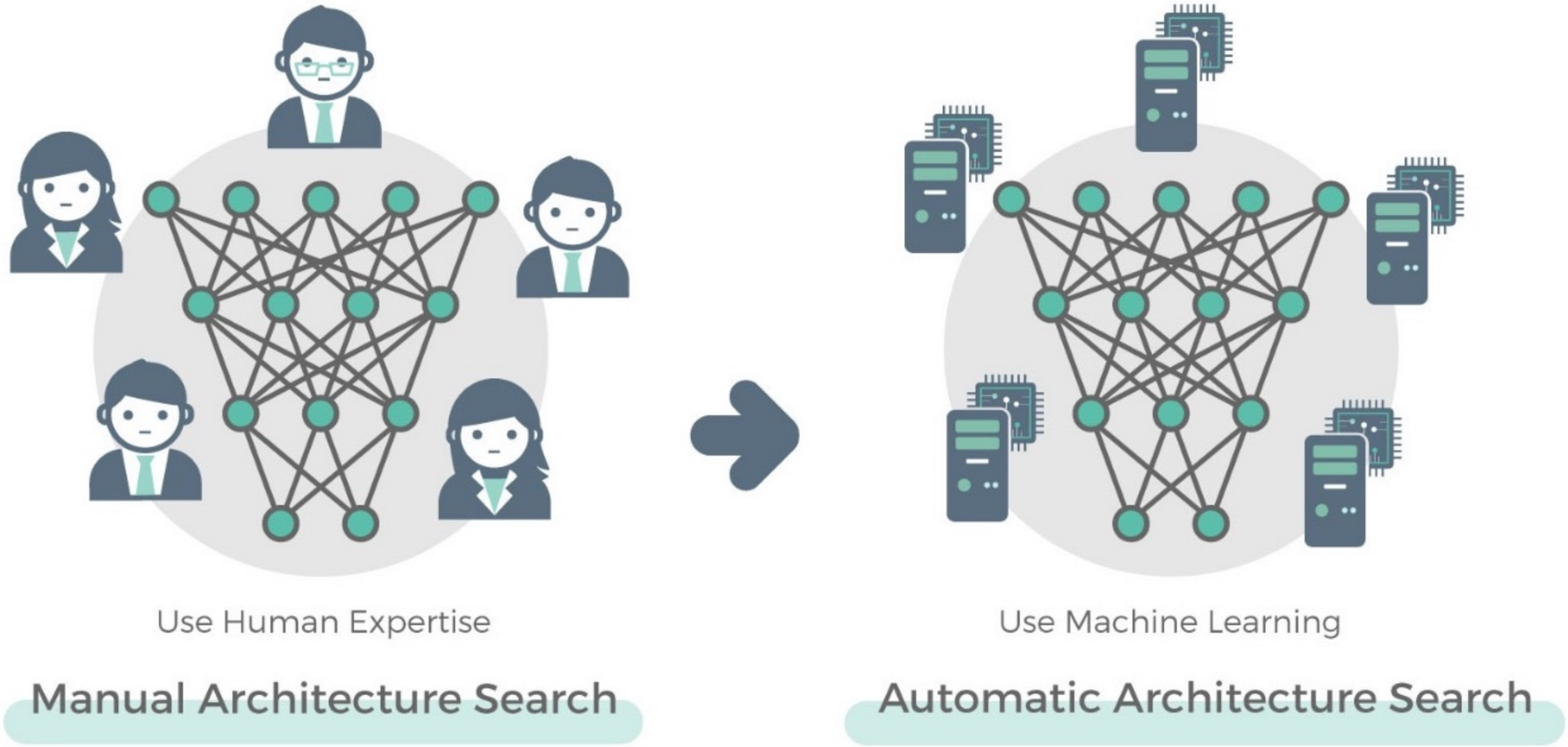
Email: lxu@oakland.edu
Homepage: https://lori930.github.io/
Office: EC 524

Trade-off Between Efficiency and Accuracy



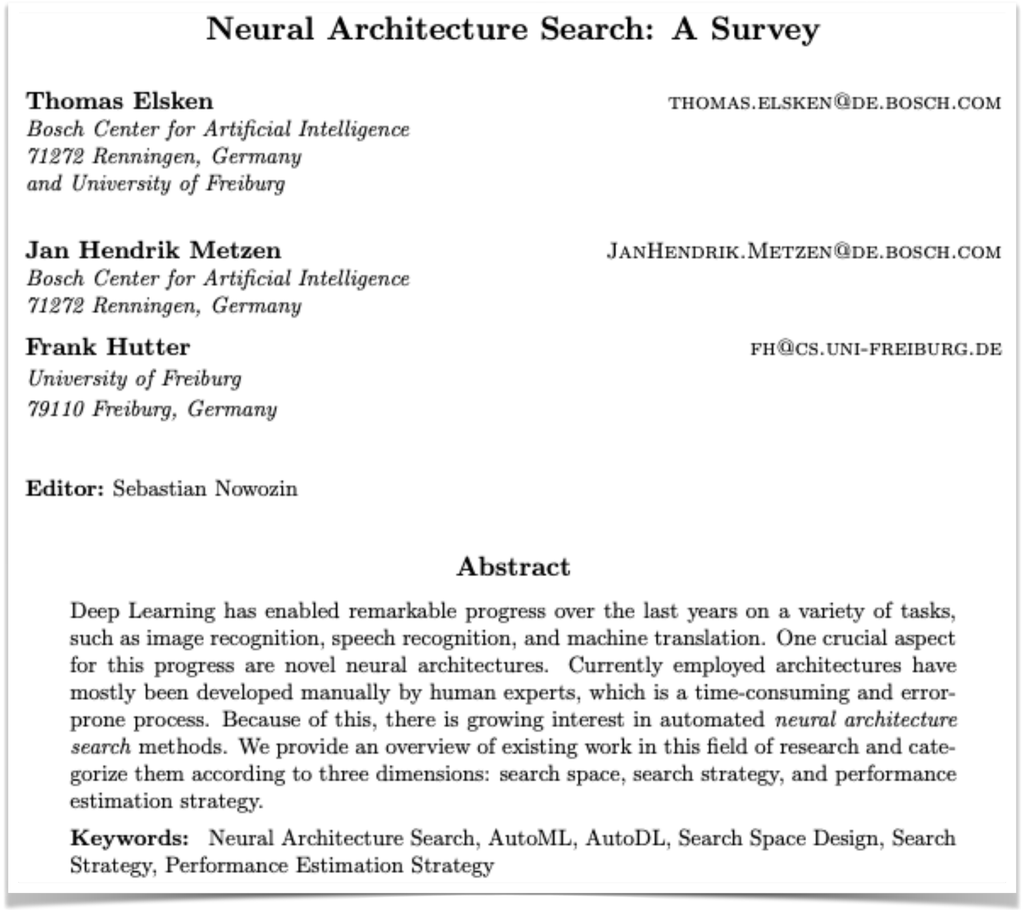
From Manual Design to Automatic Design

Huge design space, manual design is unscalable



Framework of Neural Architecture Search

The goal of NAS is to find the best neural network architecture in the search space, maximizing the objective of interest (e.g., accuracy, efficiency, etc.)



Neural Architecture Search: A Survey

Thomas Elsken THOMAS.ELSKEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany
and University of Freiburg

Jan Hendrik Metzen JANHENDRIK.METZEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany

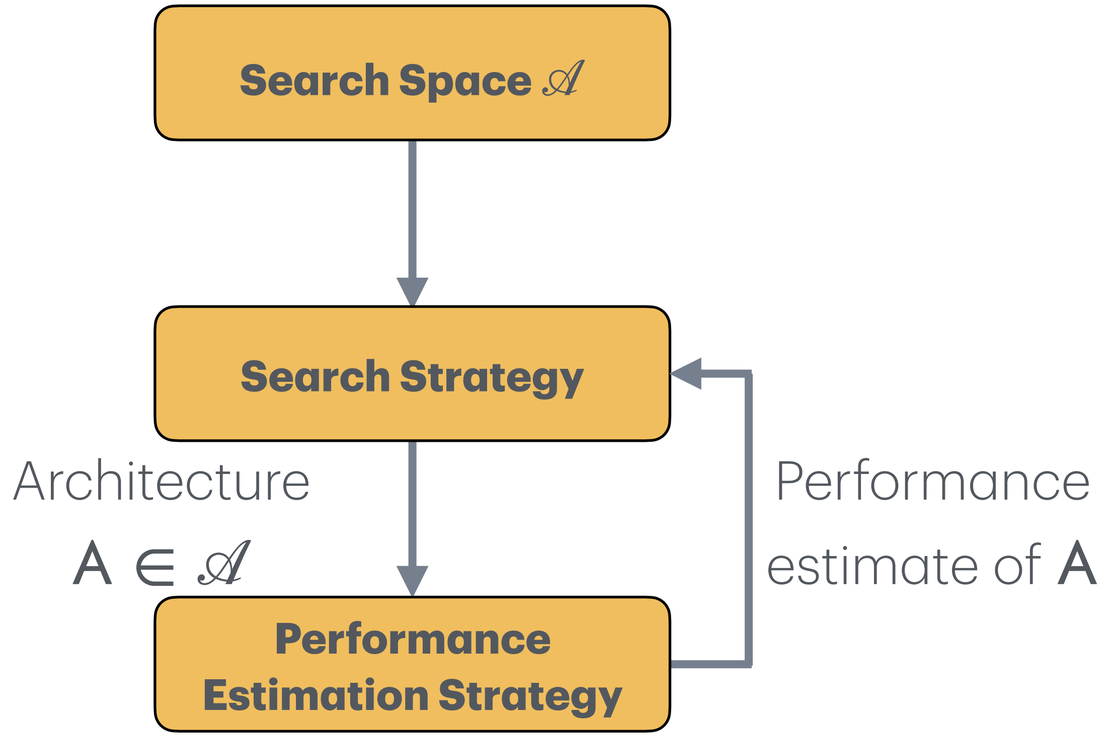
Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg
79110 Freiburg, Germany

Editor: Sebastian Nowozin

Abstract

Deep Learning has enabled remarkable progress over the last years on a variety of tasks, such as image recognition, speech recognition, and machine translation. One crucial aspect for this progress are novel neural architectures. Currently employed architectures have mostly been developed manually by human experts, which is a time-consuming and error-prone process. Because of this, there is growing interest in automated *neural architecture search* methods. We provide an overview of existing work in this field of research and categorize them according to three dimensions: search space, search strategy, and performance estimation strategy.

Keywords: Neural Architecture Search, AutoML, AutoDL, Search Space Design, Search Strategy, Performance Estimation Strategy

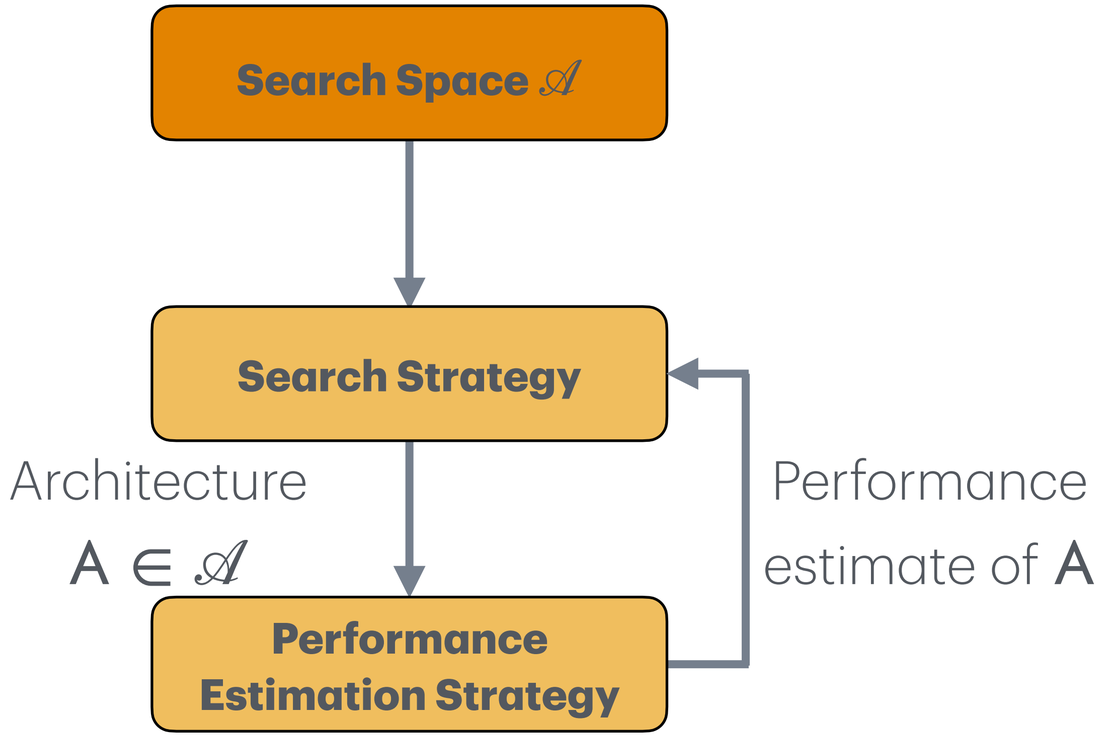


Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.

Search Space in NAS

Search space is a set of candidate neural network architectures

- **Cell-level** search space
- **Network-level** search space
  - Depth dimension
  - Resolution dimension
  - Width dimension
  - Kernel size dimension
  - Topology connection
- Deisgn the search space

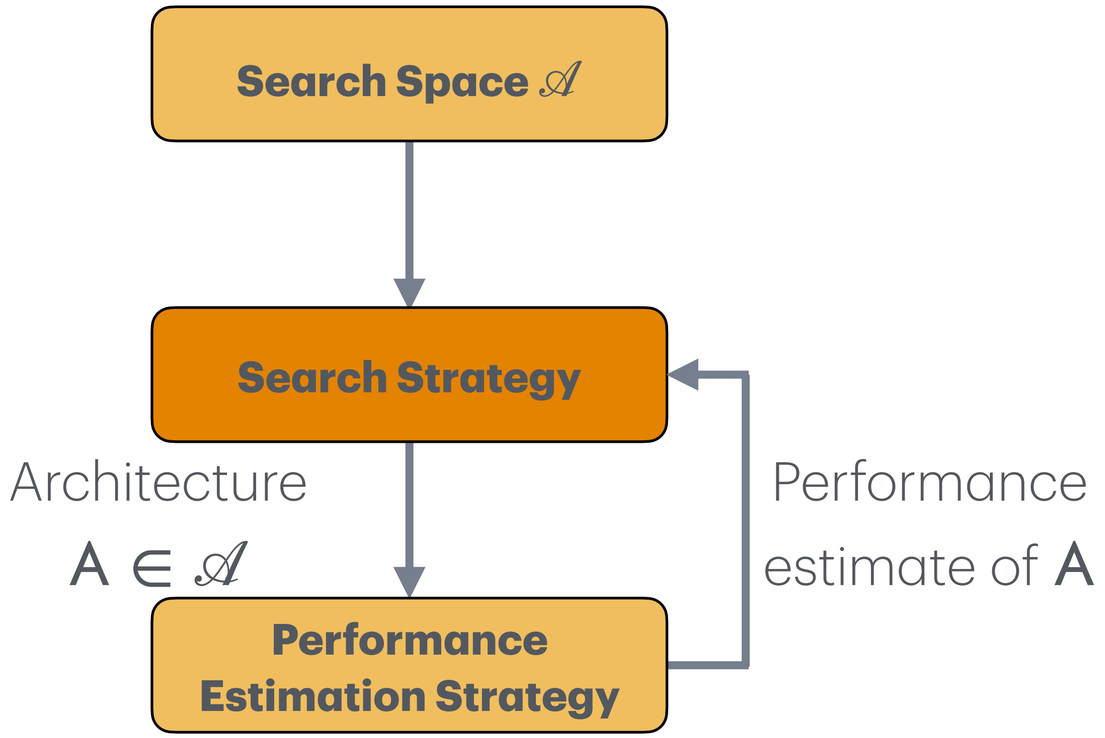


Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.

Search Strategy in NAS

Search strategy defines how to explore the search space

- Grid search
- Random search
- Reinforcement learning
- Gradient descent
- **Evolutionary search**



Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.

Framework of Neural Architecture Search

Performance estimation strategy defines how to estimate/predict the performance of a given neural network architecture in the design space

Neural Architecture Search: A Survey

Thomas Elsken THOMAS.ELSKEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany
and University of Freiburg

Jan Hendrik Metzen JANHENDRIK.METZEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany

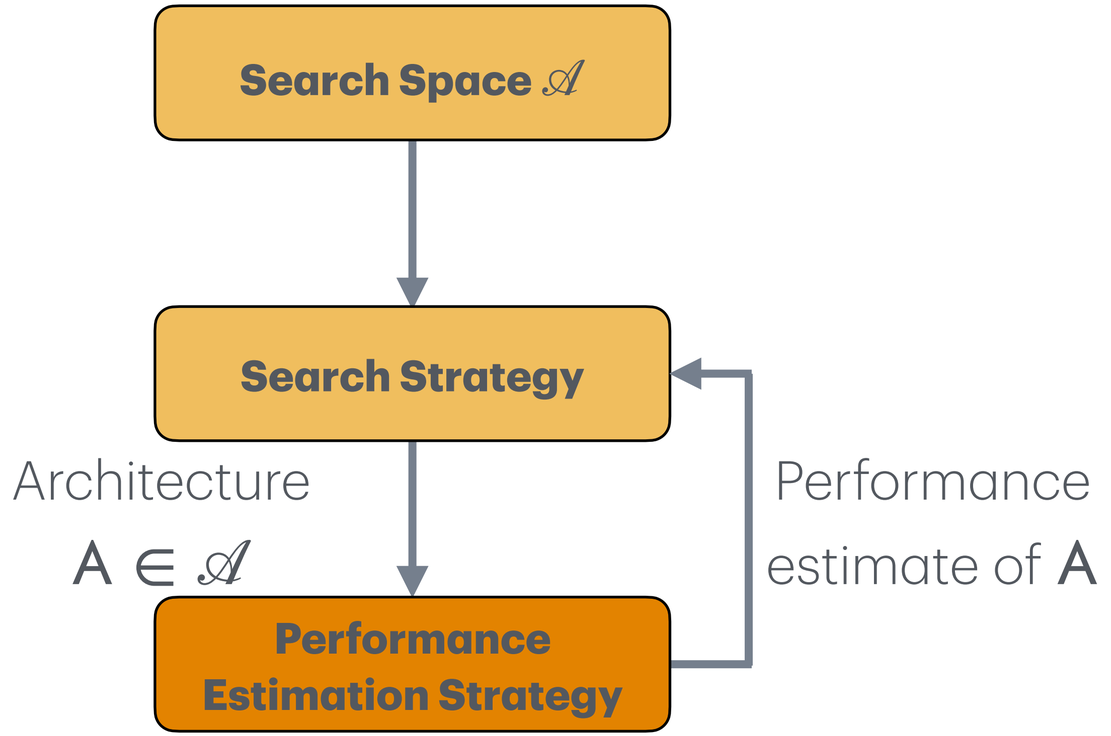
Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg
79110 Freiburg, Germany

Editor: Sebastian Nowozin

Abstract

Deep Learning has enabled remarkable progress over the last years on a variety of tasks, such as image recognition, speech recognition, and machine translation. One crucial aspect for this progress are novel neural architectures. Currently employed architectures have mostly been developed manually by human experts, which is a time-consuming and error-prone process. Because of this, there is growing interest in automated *neural architecture search* methods. We provide an overview of existing work in this field of research and categorize them according to three dimensions: search space, search strategy, and performance estimation strategy.

Keywords: Neural Architecture Search, AutoML, AutoDL, Search Space Design, Search Strategy, Performance Estimation Strategy



Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.

Lecture Plan

Today we will:

- Learn efficient and hardware-aware NAS
  - Accuracy estimation strategy
  - Zero-shot NAS
  - Hardware-aware NAS
  - Neural-hardware architecture co-search

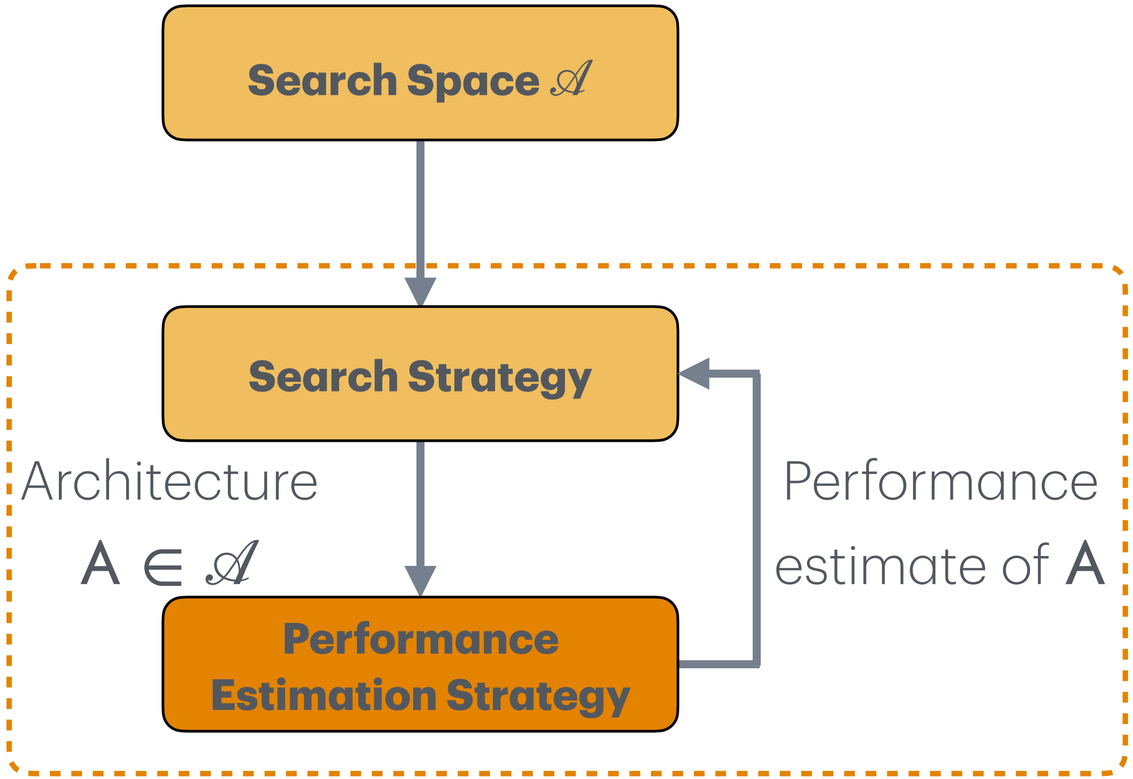
Accuracy Estimation Strategy

Accuracy Estimation Strategy

How to estimate/predict the performance of a given neural network architecture in the design space

- **Train from scratch**
- **Inherit weight**
- **Hypernetwork**

```
For search episodes:
  For training iterations:
    forward-backward();
  If good_model: break;
  For post-search training iterations:
    forward-backward();
```



Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.

Train from scratch

Train the RNN with RL to maximize the expected accuracy

- Train the given model from scratch on the training set
- Evaluate the trained model on the validation set to get accuracy R
- Train **12,800 model architectures** on **CIFAR10** takes **22,400 GPU hours**

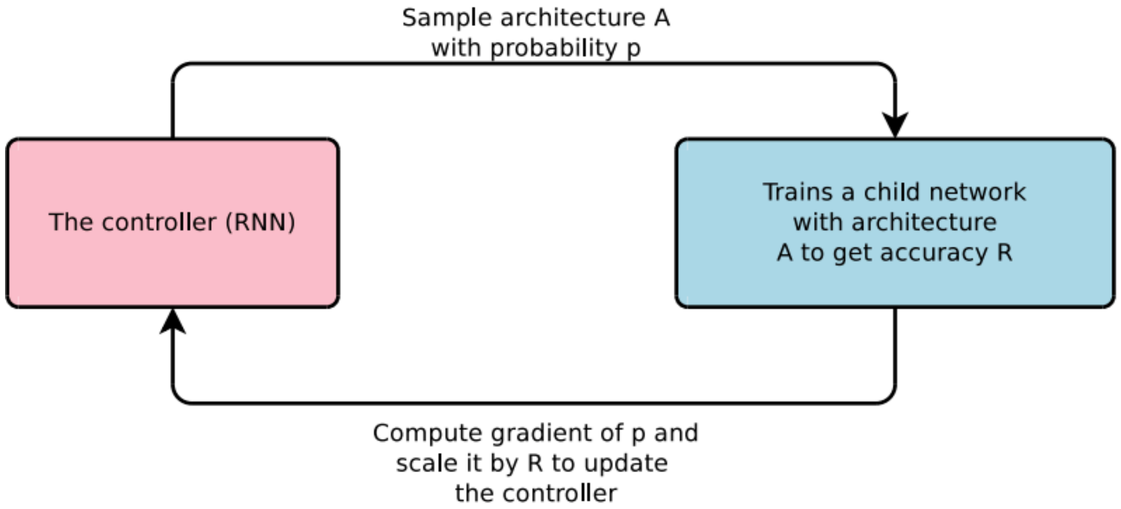
😖 **Prohibitive training cost**

Large-scale datasets?

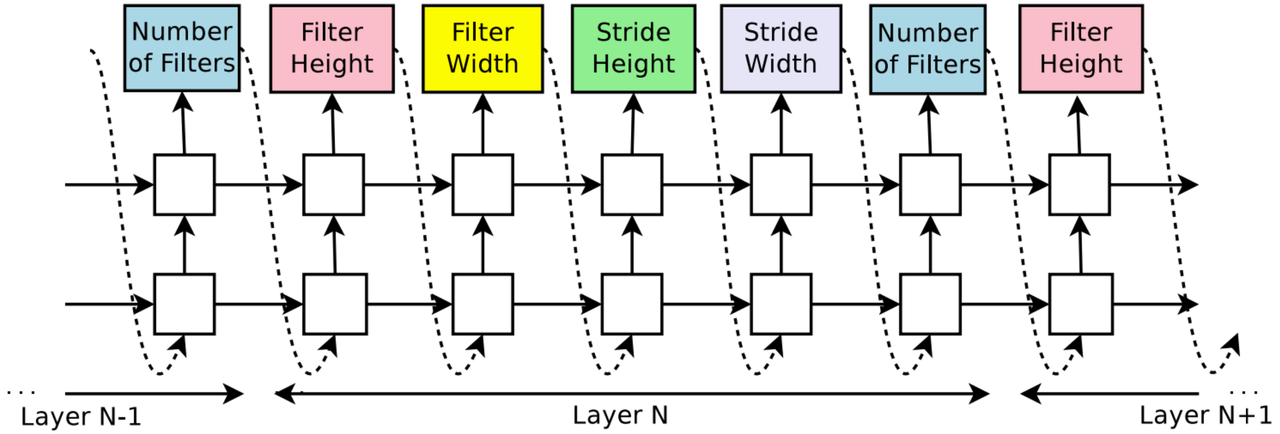
ImageNet

COCO

…



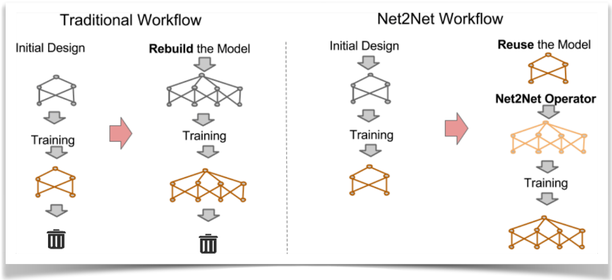
Overview of RL-based NAS



The RNN controller samples a simple convolutional network

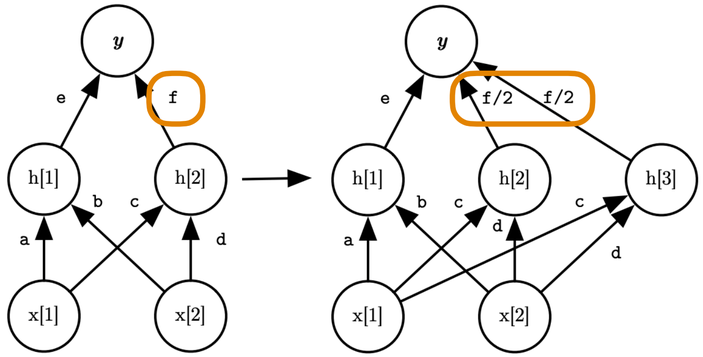
Zoph, B. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.

Inherit Weight

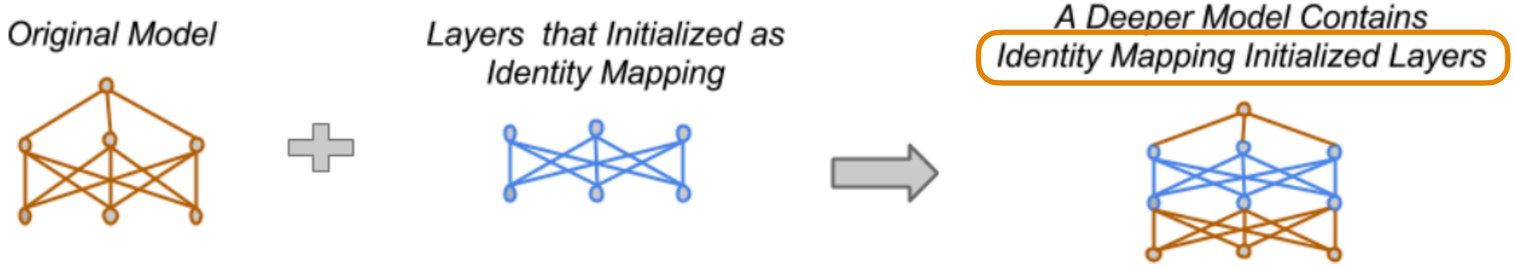


Net2Net: accelerating learning via knowledge transfer

- Instead of training from scratch, **inherit weights from a parent** model to reduce the training cost
- **Function-preserving transformations**
  - **Net2WiderNet**: replace a model with an equivalent model that has more units in each hidden layer
  - **Net2DeeperNet**: replace a model that satisfies some properties with an equivalent, deeper model



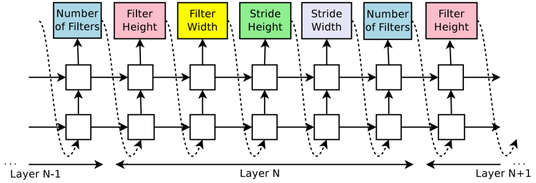
Net2WiderNet



Net2DeeperNet

Chen, T., Goodfellow, I., & Shlens, J. (2015). Net2net: Accelerating learning via knowledge transfer. arXiv preprint arXiv:1511.05641.

Inherit Weight

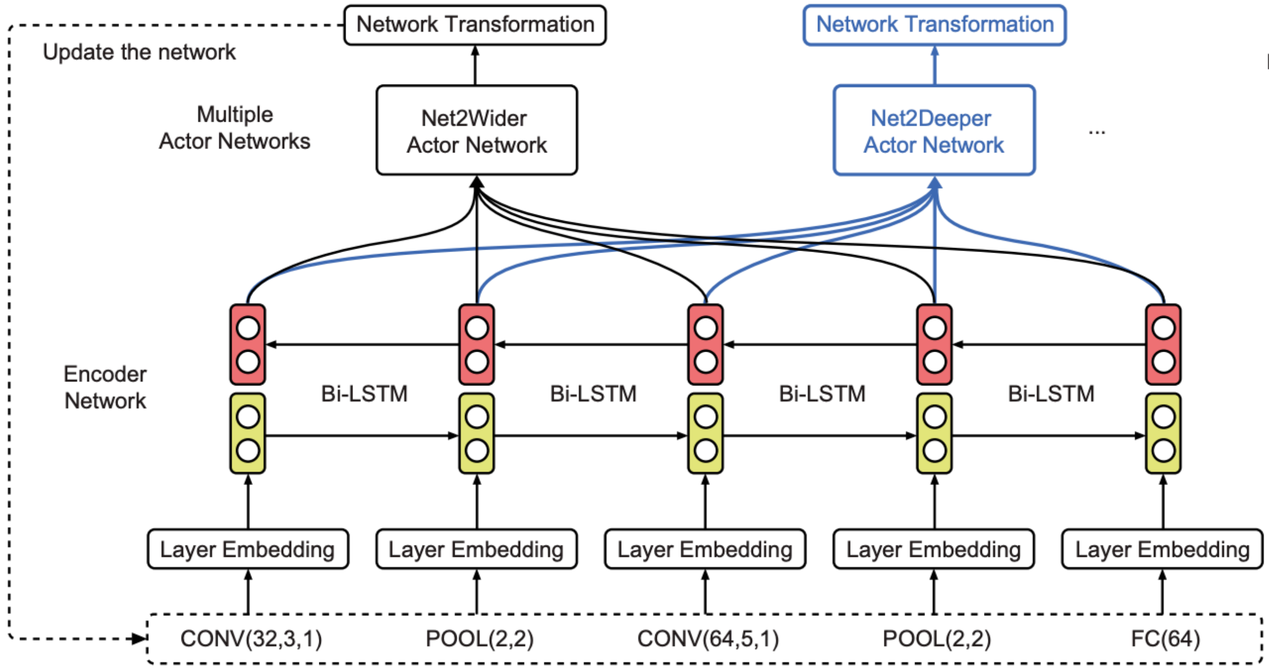


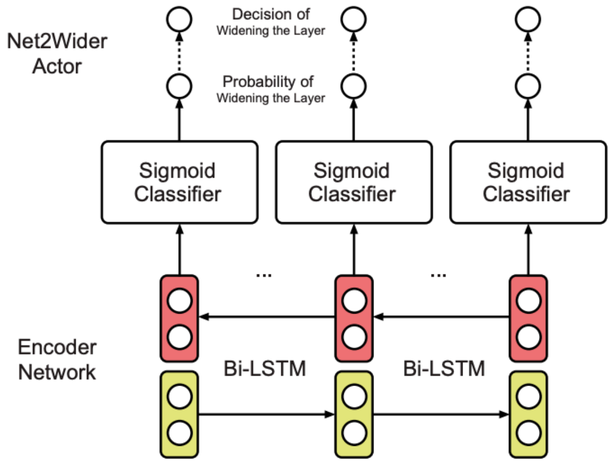
The RNN controller samples a simple convolutional network

EAS: Efficient architecture search by network transformation

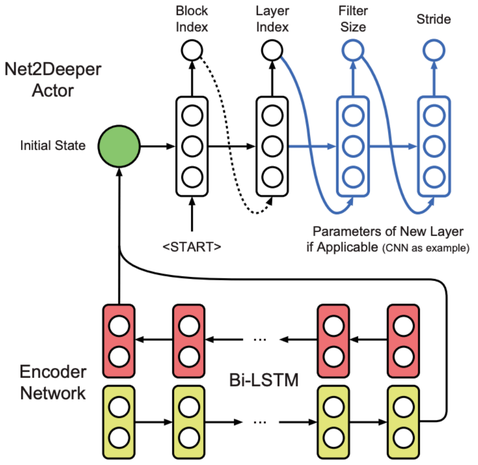
- A **meta-controller** (RL agent): grow the network depth or layer width

- **Generate network transformation actions** to update the model architecture, instead of directly generating the model architecture





Net2Wider actor

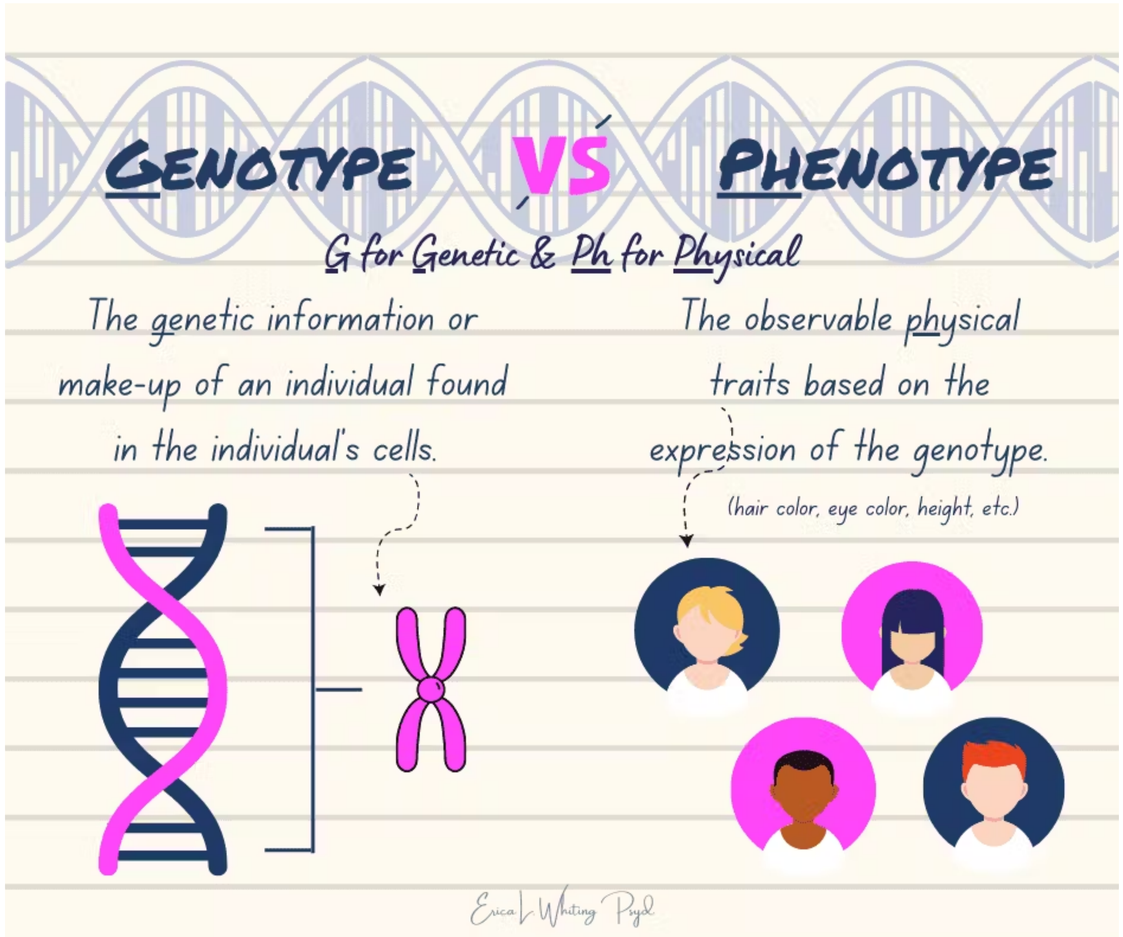


Net2Deeper actor

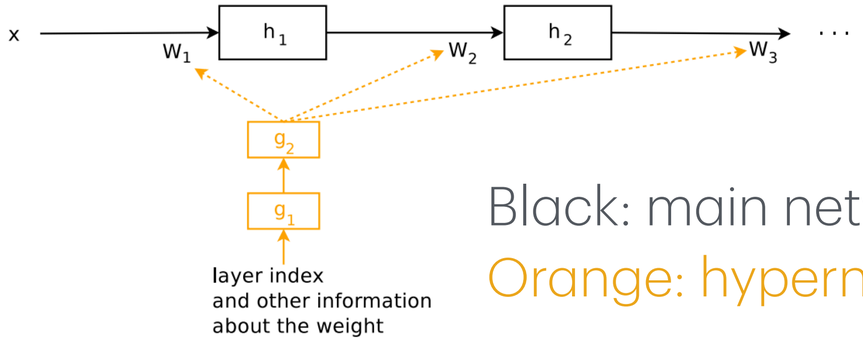
Cai, H., Chen, T., Zhang, W., Yu, Y., & Wang, J. (2018, April). Efficient architecture search by network transformation. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

Hypernetwork

Use one network (hypernetwork) to generate the weight for another network (main network)



- Hypernetwork (**genotype**)
  - Encodes and generates the weights for the main network
- The main network (**phenotype**)
  - Performance is determined by the weights generated by the hypernetwork



Black: main network

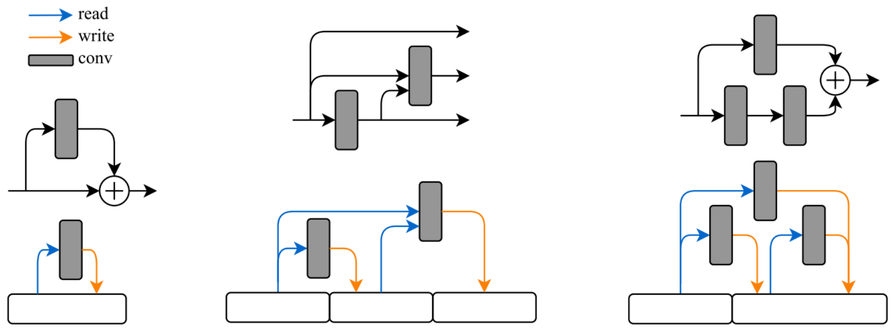
Orange: hypernetwork

Source Ha, D., Dai, A., & Le, Q. V. (2016). Hypernetworks. arXiv preprint arXiv:1609.09106.

Hypernetwork in NAS

SMASH: One-shot Model Architecture Search through Hypernetworks

- Hypernetwork: **generate the weights** of a main model conditioned on that model's architecture
- Compare the relative validation performance of networks with Hypernetwork-generated weights
- Search over a wide range of architectures at the cost of a single training run



Memory-bank reprsentations of ResNet, DenseNet, and FractalNet blocks

Algorithm 1 SMASH

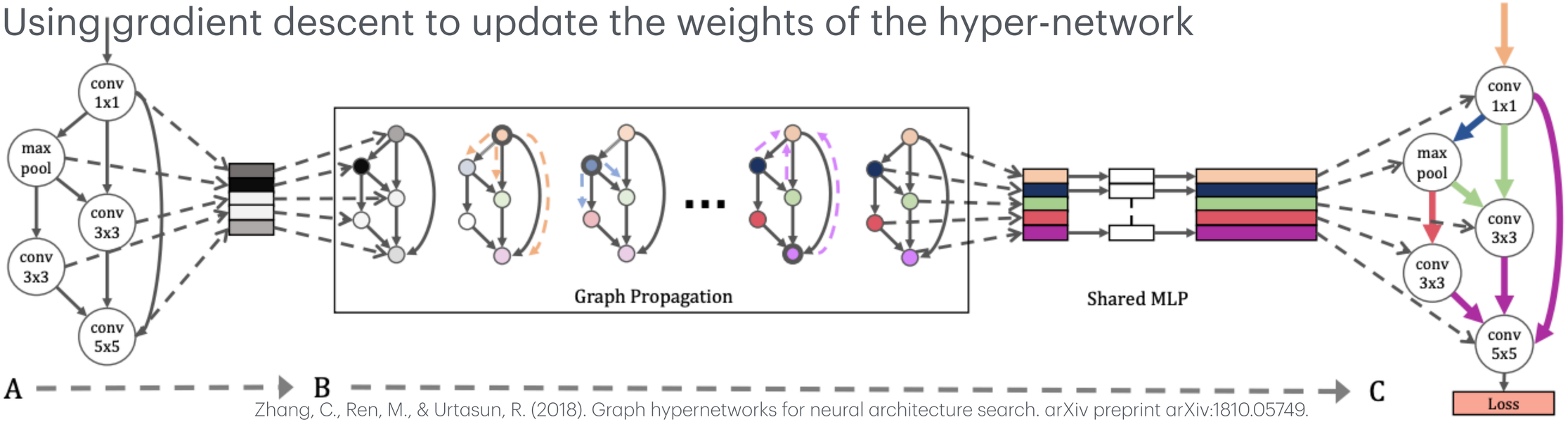
input Space of all candidate architectures, $\mathbb{R}_c$
  Initialize HyperNet weights H
  loop
    Sample input minibatch x_i, random architecture c and architecture weights $W = H(c)$
    Get training error $E_t = f_c(W, x_i) = f_c(H(c), x_i)$, backprop and update H
  end loop
  loop
    Sample random c and evaluate error on validation set $E_v = f_c(H(c), x_v)$
  end loop
  Fix architecture and train normally with freely-varying weights W

Brock, A., Lim, T., Ritchie, J. M., & Weston, N. (2017). Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344.

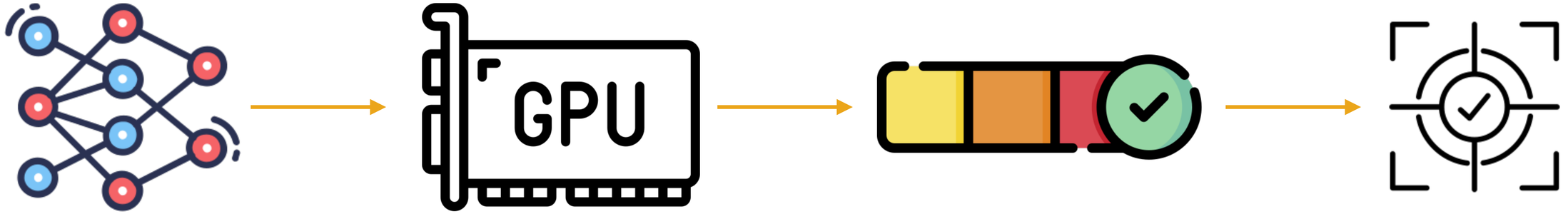
Hypernetwork in NAS

GHN: Graph Hypernetworks for Neural architecture search

- Intuition: a NN is a computing graph, graph is a good representation of the NN architecture.
- At each training step, a random model architecture is sampled from the search space
- The hypernetwork generates weights based on the model architecture
- Using gradient descent to update the weights of the hyper-network



Zhang, C., Ren, M., & Urtasun, R. (2018). Graph hypernetworks for neural architecture search. arXiv preprint arXiv:1810.05749.

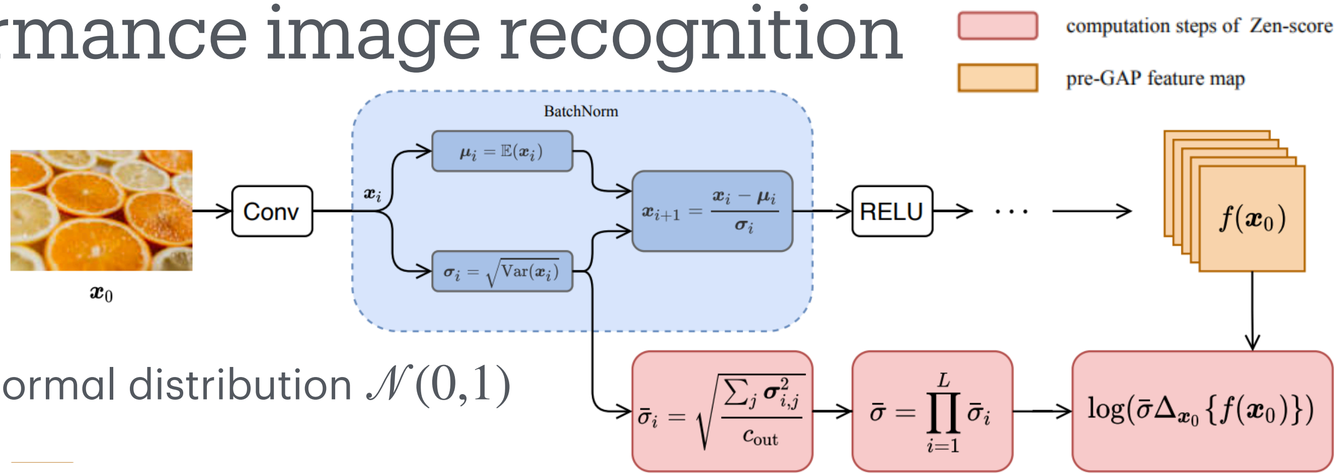


Zero-shot NAS

Estimate the accuracy without training the model.

Zen-NAS

A zero-shot NAS for high-performance image recognition



1. Initialize the random input $x \sim \mathcal{N}(0,1)$
2. Add a small **perturbation** to x and obtain $x' = x + \epsilon$
3. Initialize all the weights of the neural network f to a normal distribution $\mathcal{N}(0,1)$
4. calculate the log difference: $z_1 = \log(f(x') - f(x))$
5. **+ batch normalization variance:**

Hypothesis: A good model should be sensitive to input perturbations

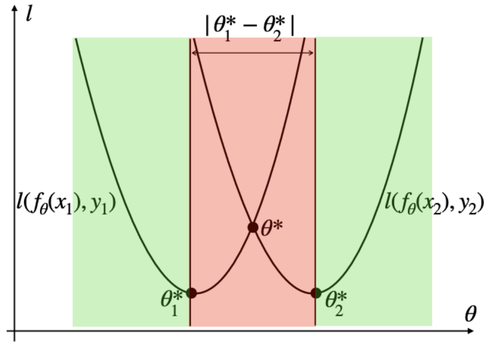
- Consider BN for layer i: $x_i = (x_i - \mu_i)/\sigma$. The square mean $\overline{\sigma_i} = \sqrt{\frac{\sum_j \sigma_{i,j}^2}{c_o}}$
- Fianlly, $z_2 = \sum_i \log \overline{\sigma_i}$, zen score $z = z_1 + z_2$

Lin, M., Wang, P., Sun, Z., Chen, H., Sun, X., Qian, Q., ... & Jin, R. (2021). Zen-nas: A zero-shot nas for high-performance image recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 347-356).

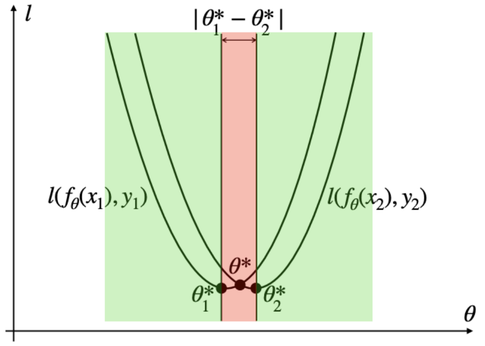
GradSign

Model performance inference with theoretical insights

- **Intuition**: a **good** model has denser sample-wise local minima, i.e., the distance between local minima points θ_1^* and θ_2^* should be **smaller**. In this case, there is a higher probability in (b) that **the gradients of different samples have the same sign (green region)**



(a) Optimization landscape with **sparser** sample-wise local optima corresponding to **worse** $J(\theta^*)$.



(b) Optimization landscape with **denser** sample-wise local optima corresponding to **better** $J(\theta^*)$.

Algorithm 1: GradSign

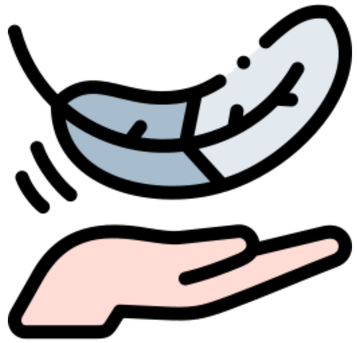
Result: GradSign score τ_f for a function class f_θ
Given $\mathcal{S} = \{(x_i, y_i)\}_{i \in [n]}$, randomly select initialization point θ_0;
Initialize $g[n, m]$;
for $i = 1, 2, \cdots, n$ **do**
  for $k = 1, 2, \cdots, m$ **do**
    $g[i, k] = \text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k)$
  end
end
$\tau_f = \sum_k |\sum_i g[i, k]|$;
return τ_f

i, n corresponds to samples

k, m corresponds to layers

$|\sum_{i=1}^{n} g[i, k]| \leq n$, "=" is achieved only when the gradiens at all samples have the same sign

Zhang, Z., & Jia, Z. (2021). Gradsign: Model performance inference with theoretical insights. arXiv preprint arXiv:2110.08616.



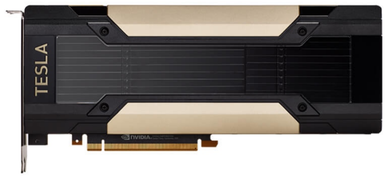
Hardware-Aware NAS

From General Model to Specialized Models

- Different platforms have different properties (degree of parallelism, cache size, memory bandwidth, etc.)

- Customize the models for each platform can achieve the best accuracy—efficiency trade-off.

Specialized GPU model



Specialized CPU model

Specialized mobile model

Specialized Pi model

One model for all hardware

🤩 **Accuracy-efficiency trade-off**

😖 **Expensive design cost**

Efficiently Search on Target Task & Hardware

Conventional NAS relies on proxy task

- Conventional NAS is **very expensive**
  - NASNet: **48,000 GPU hours** on CIFAR ≈ 5 years on a single GPU
  - DARTS: **100GB GPU memory** if directly searched on ImageNet
- Therefore, **proxy tasks** are utilized:
  - CIFAR-10
  - Small architecture space (e.g., low depth)
  - Fewer epochs training←**But the converge speed could be diverse.**
  - FLOPS and parameter counts🤔 **Does MACs/FLOPS translate to real hardware efficiency?**

😣 Limitations of Proxy task

- **Suboptimal for the target task**
- **Blocks need to share the same structure**
- **Not optimized for the target hardware**

Can we **directly** learn NAS on the large-scale target task & hardware, while allowing all blocks to have different structures?

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Efficiently Search on Target Task & Hardware

ProxylessNAS

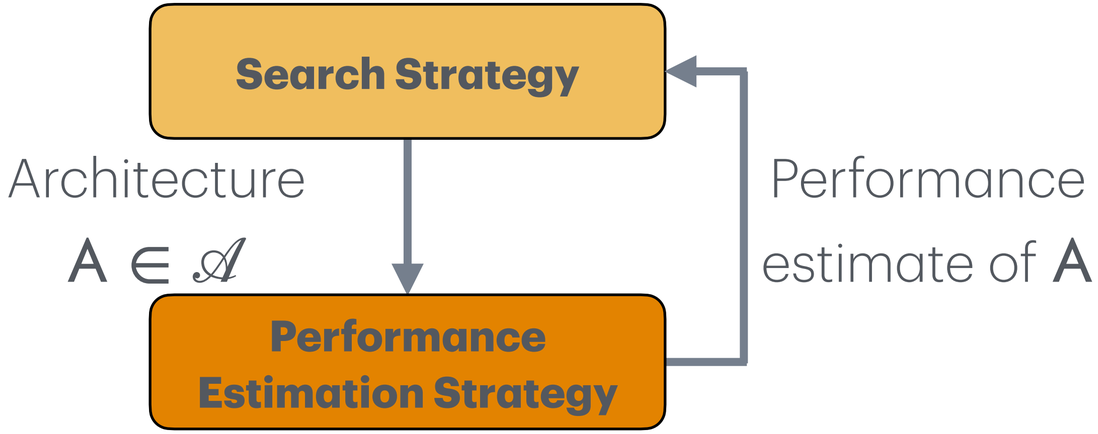
- Conventional NAS is **very expensive**
  - NASNet: **48,000 GPU hours** on CIFAR ≈ 5 years on a single GPU
  - DARTS: **100GB GPU memory** if directly searched on ImageNet
- Therefore, **proxy tasks** are utilized:
  - CIFAR-10
  - Small architecture space (e.g., low depth)
  - Ferer epochs training
  - FLOPS and parameter counts

🤩 ProxylessNAS

- ImageNet
- Large architecture space
- Full training
- Profiled latency

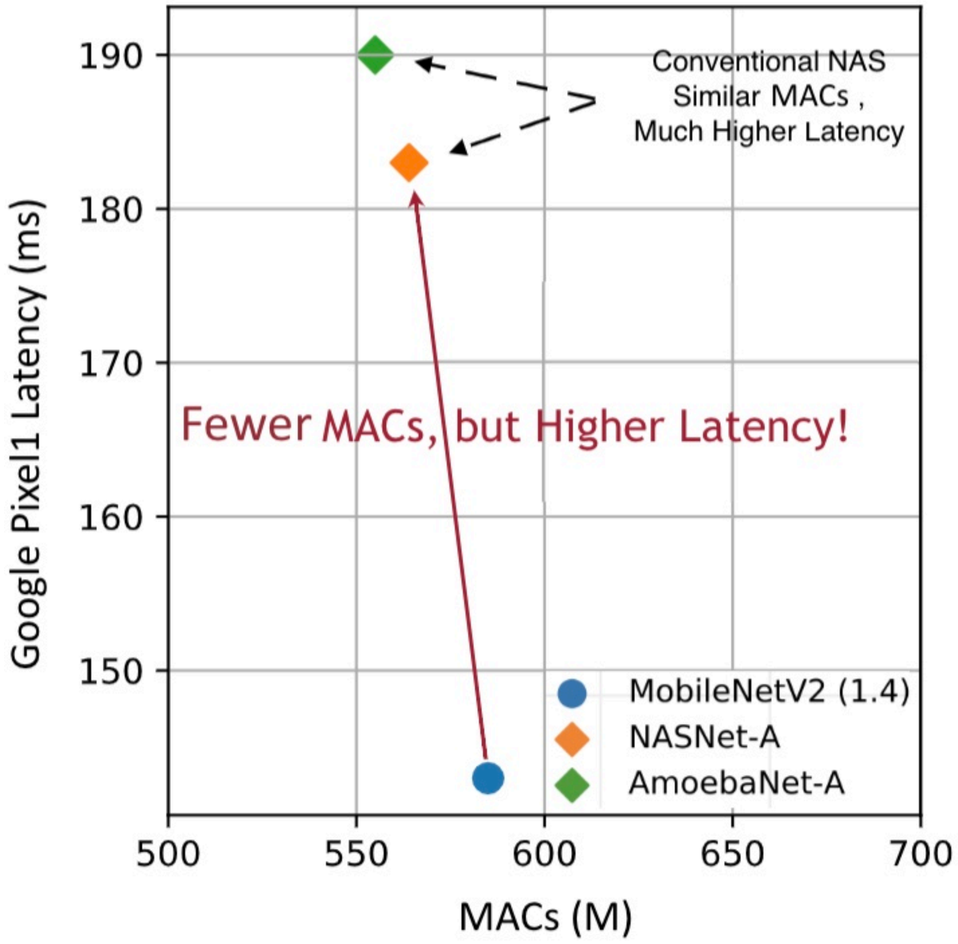
Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

How to evaluate the performance for target task & hardware?



Search for Efficient Model

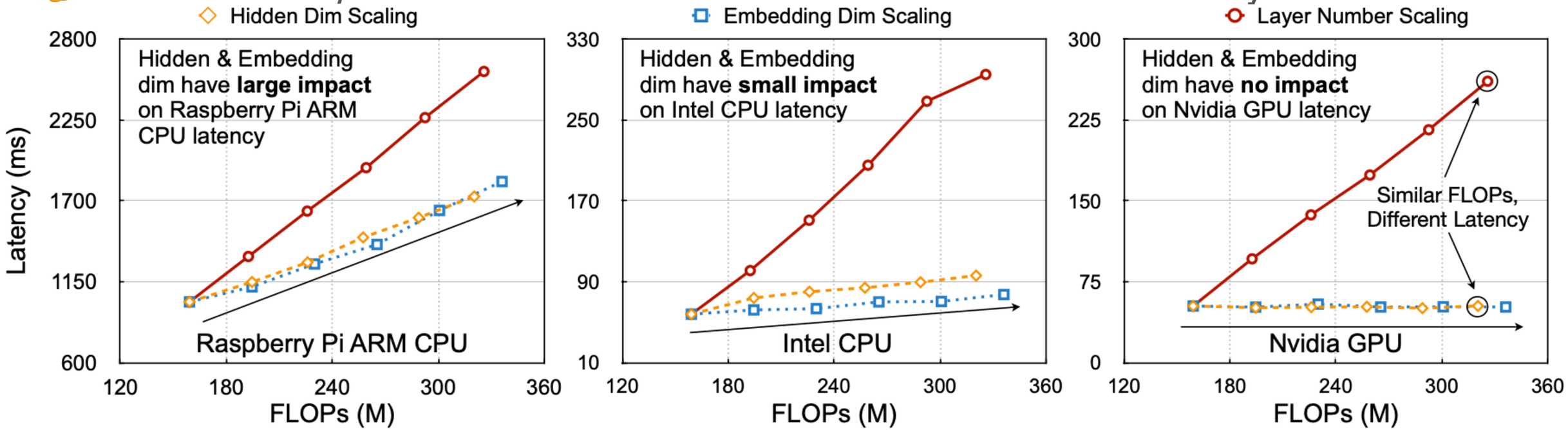
🤔 Does MACs/FLOPS translate to real hardware efficiency?



Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Search for Efficient Model

🤔 Does MACs/FLOPS translate to real hardware efficiency?



- MACs/FLOPS does **not** translate to the real measured latency
- Latency influencing factors of different hardware are contrasting
- We need to consider **hardware latency feedback** to design specialized models for different hardware

Wang, H., Wu, Z., Liu, Z., Cai, H., Zhu, L., Gan, C., & Han, S. (2020). Hat: Hardware-aware transformers for efficient natural language processing. arXiv preprint arXiv:2005.14187.

Incorporate Hardware Feedback into NAS

🤔 How?

- Measure the latency on-device?

🤩 **Accurate**

Measure the latency

NAS

Architecture

Updates
(measured latency)

😣 **Slow**

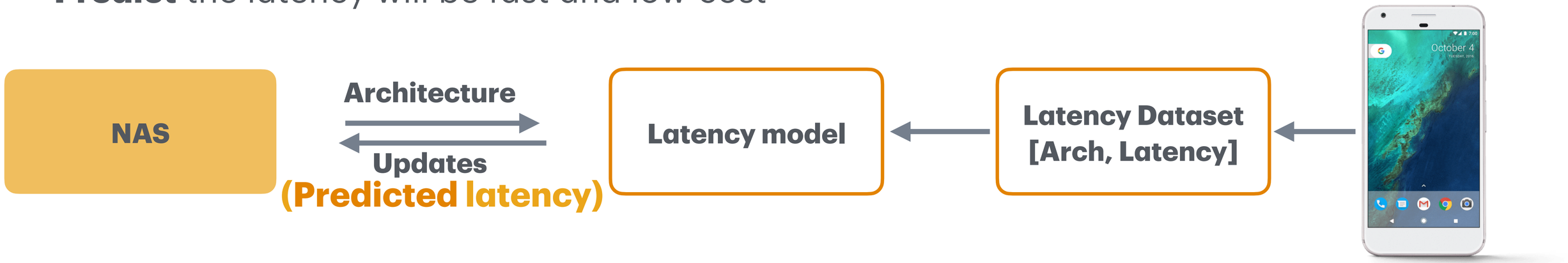
😣 **Expensive**

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 2820-2828).

Incorporate Hardware Feedback into NAS

🤔 How?

- Measure the latency on-device is slow and expensive
- **Predict** the latency will be fast and low-cost

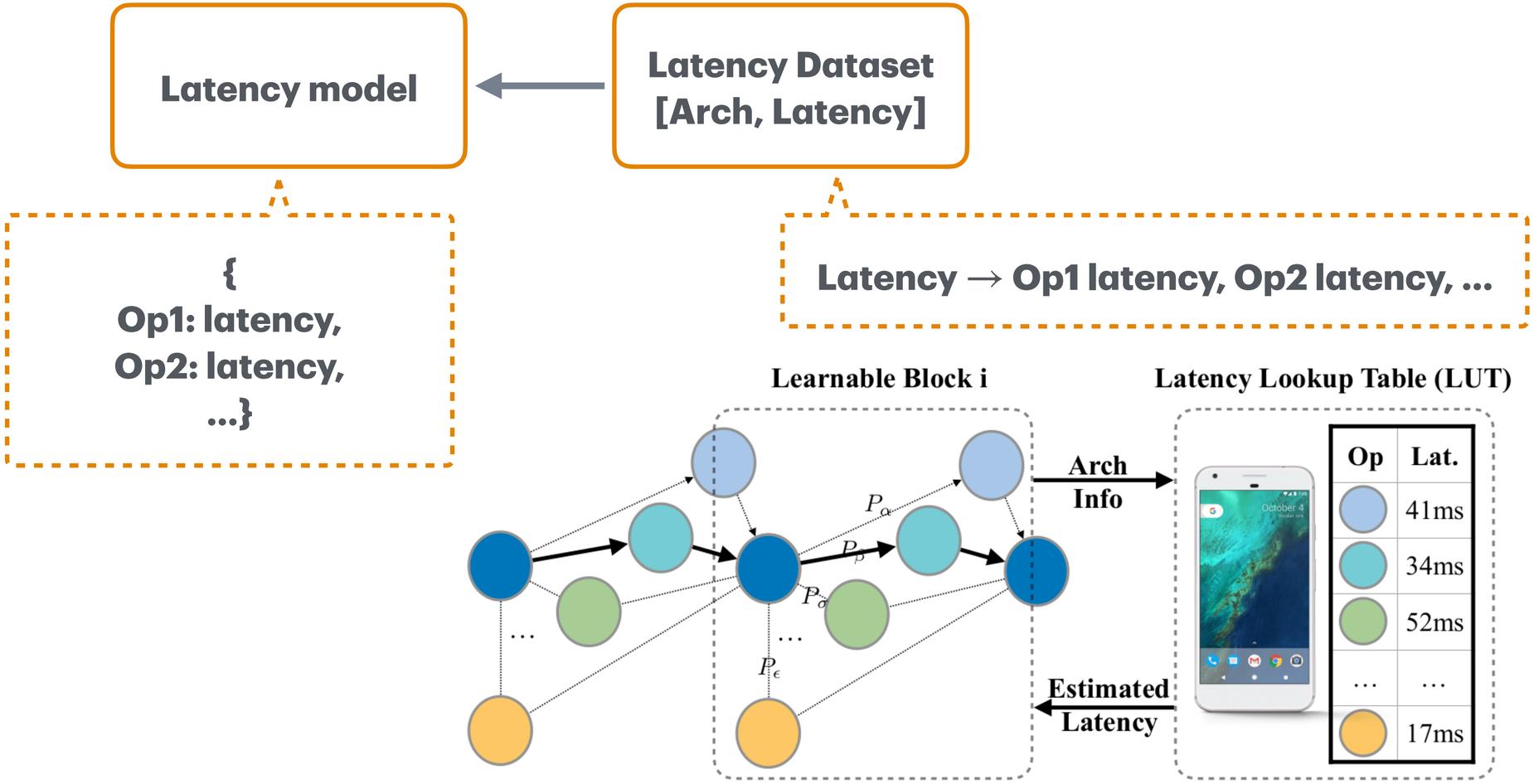


Profile the layer-wise latency
Profile the network-wise latency

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Latency Prediction

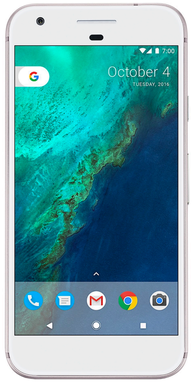
Profile the layer-wise latency with a latency lookup table (LUT)

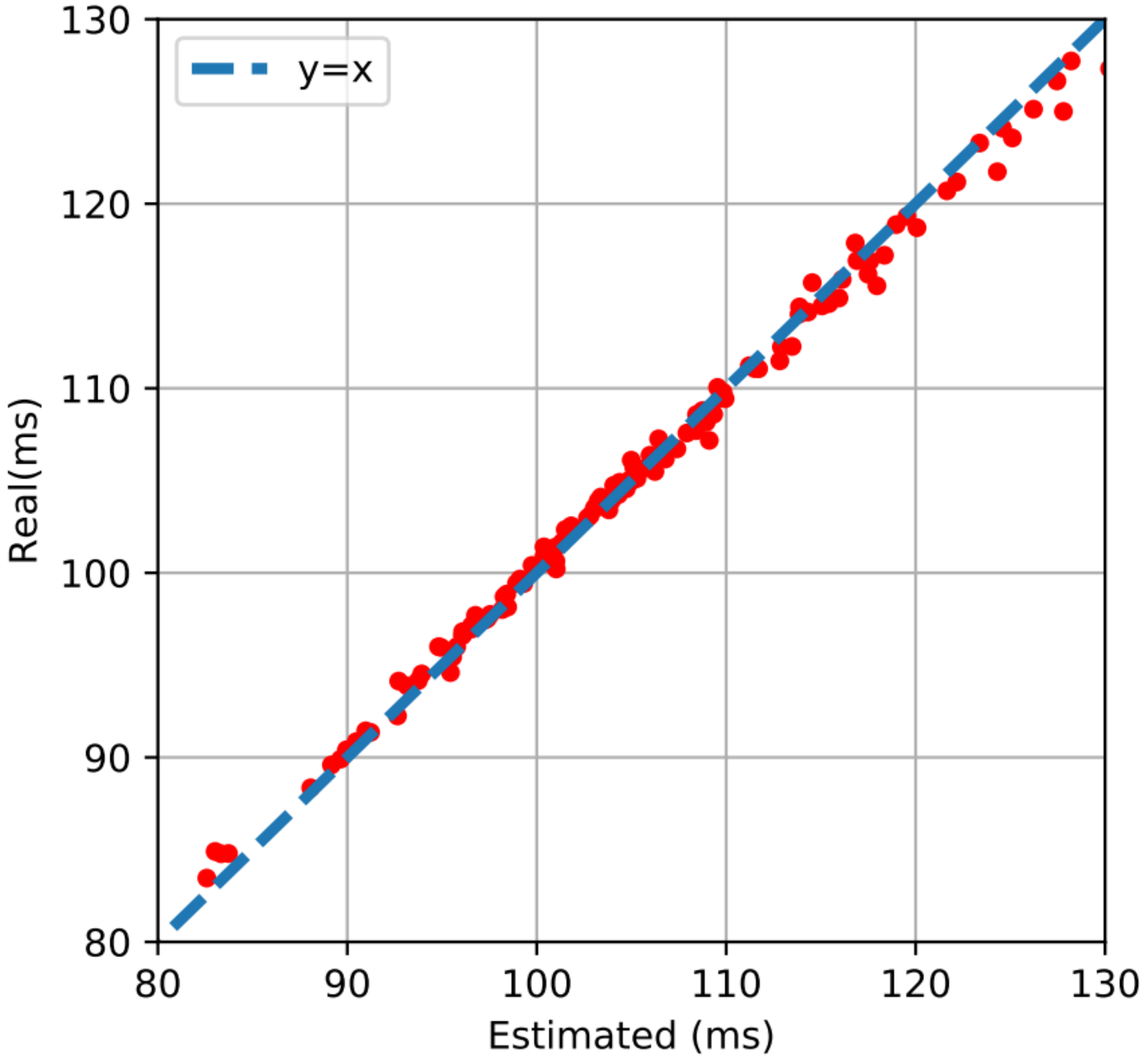


Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Latency Prediction

- Latency prediction model can be used to replace the expensive mobile farm infrastured with little error (RMSE=0.75ms) introduced

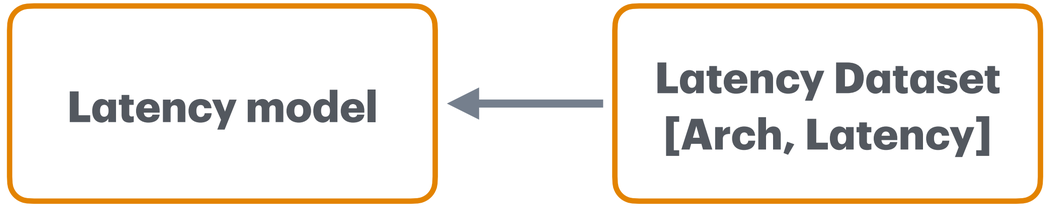


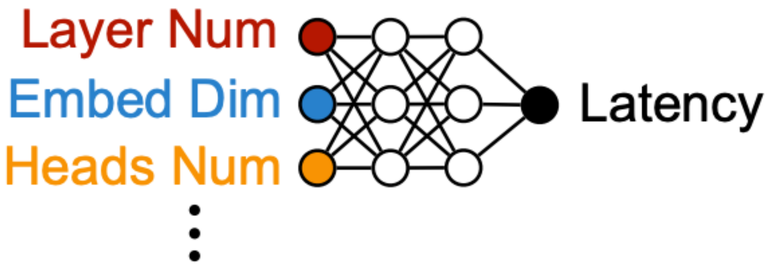


Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Latency Prediction

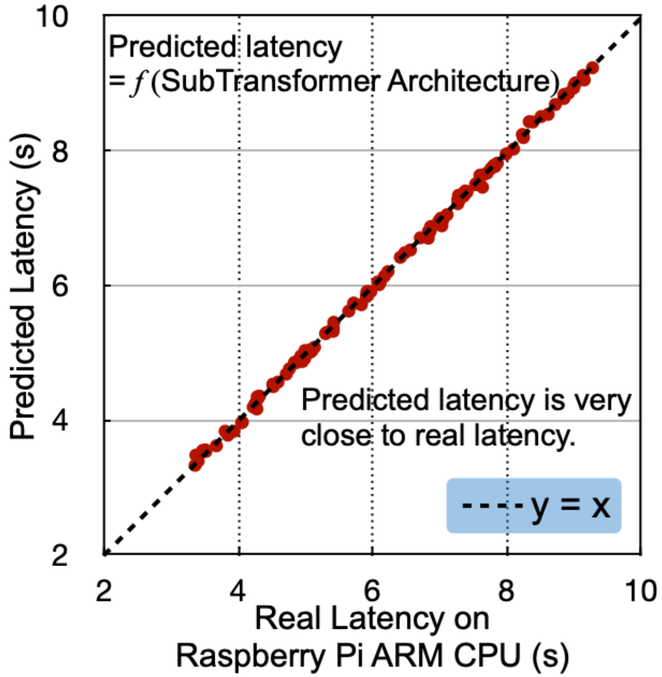
Profile the network-wise latency with a latency prediction model





Latency prediction model

- Train a latency prediction model for each hardware to provide fast and accurate (RMSE=0.1s) latency feedback



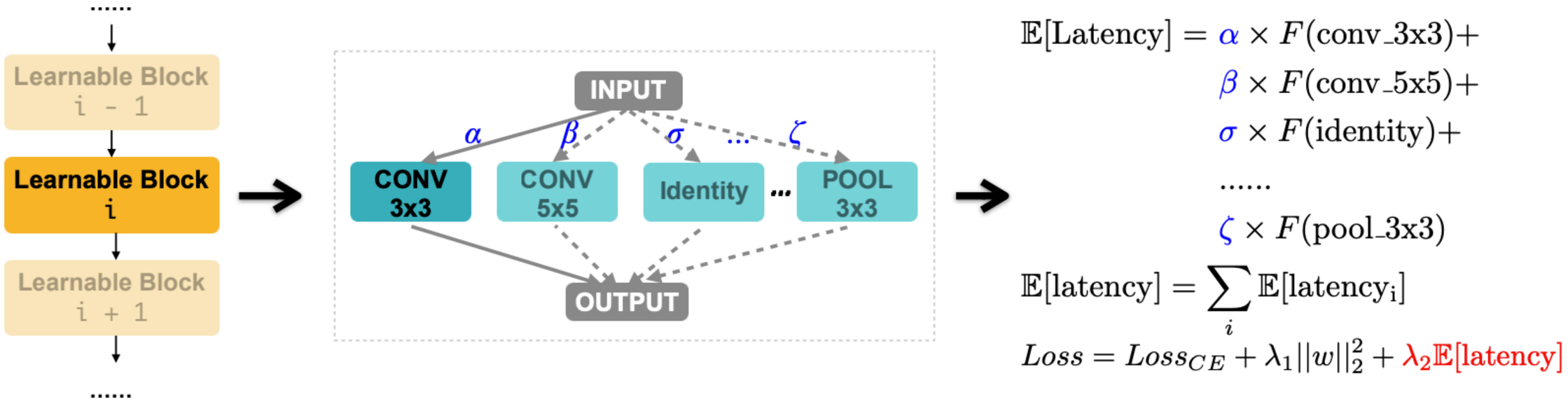
Wang, H., Wu, Z., Liu, Z., Cai, H., Zhu, L., Gan, C., & Han, S. (2020). Hat: Hardware-aware transformers for efficient natural language processing. arXiv preprint arXiv:2005.14187.

Search Strategy: Gradient Descent

ProxylessNAS

- Is it also possible to take latency into account for gradient-based search

- Here, F is a latency prediction model (typically a regressor or a lookup table), with such formulation, we can calculate an additional gradient for the architecture parameters from the latency penalty term



$$\mathbb{E}[\text{Latency}] = \alpha \times F(\text{conv\_3x3}) + \beta \times F(\text{conv\_5x5}) + \sigma \times F(\text{identity}) + \ldots\ldots + \zeta \times F(\text{pool\_3x3})$$

$$\mathbb{E}[\text{latency}] = \sum_i \mathbb{E}[\text{latency}_i]$$

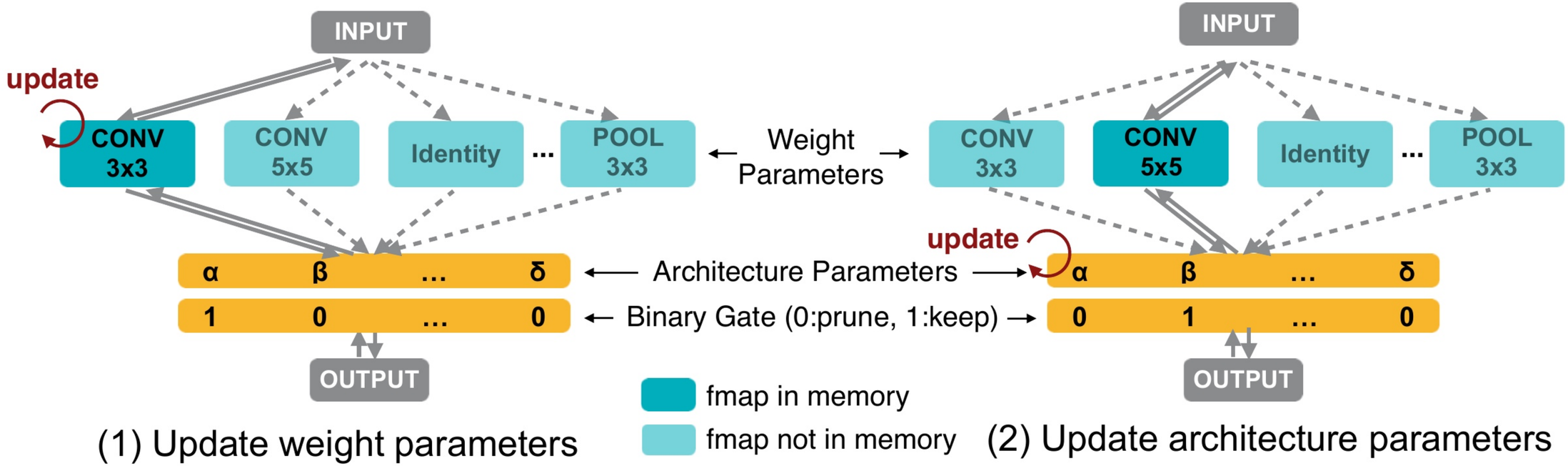
$$Loss = Loss_{CE} + \lambda_1 ||w||_2^2 + \lambda_2 \mathbb{E}[\text{latency}]$$

Making latency differentiable by introducing latency regularization loss

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Efficiently Search on Target Task & Hardware

ProxylessNAS: Path-level pruning and binarization

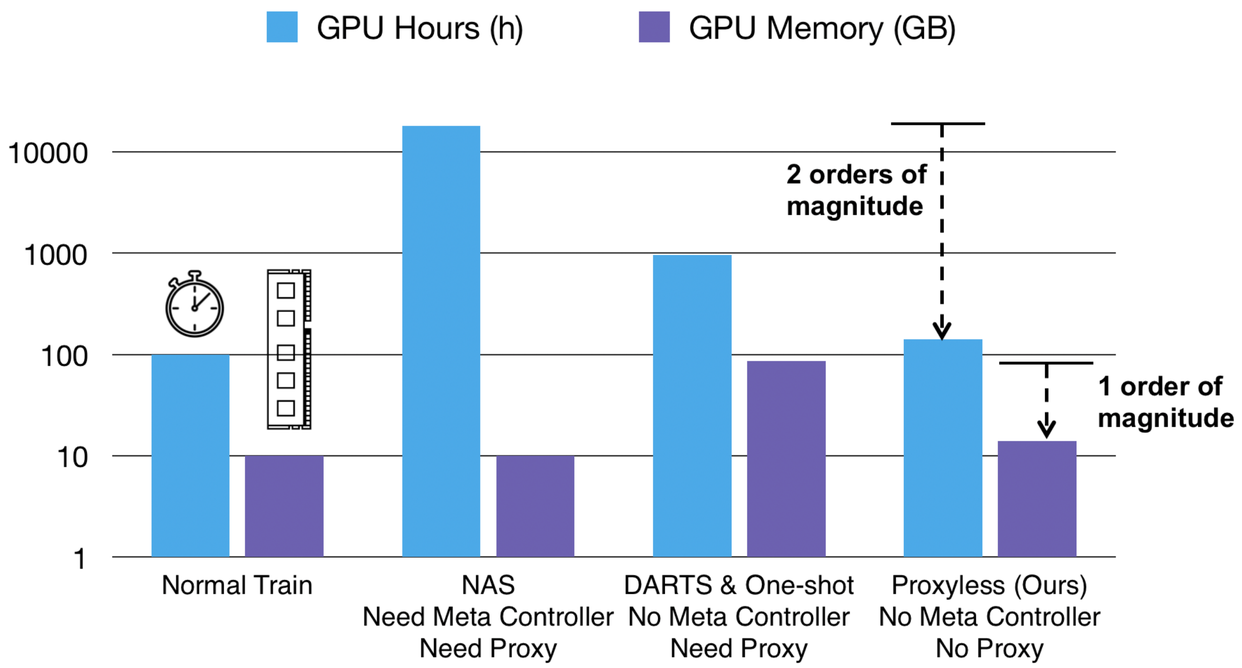


- Build the over-parameterized network **with all candidate paths**
- Simplify NAS to be **a single training process** of an over-parameterized network
- **Pruning** redundant paths based on architecture parameters
- **Binarize** the architecture parameters and allow only **one path of activation to be active**

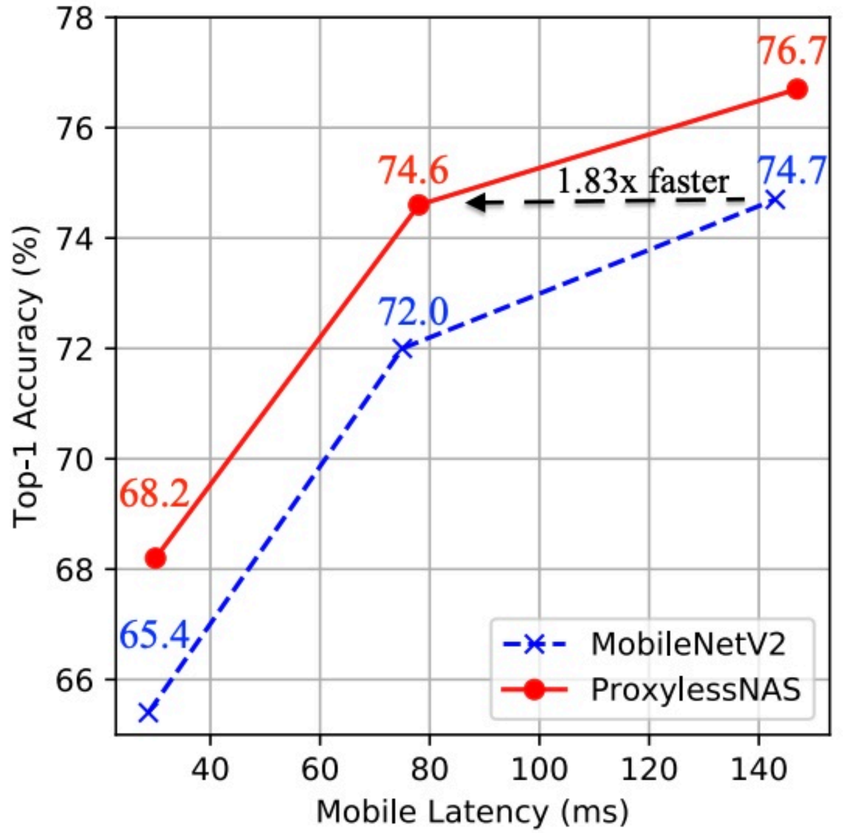
Reduces memory footprint from O(N) to O(1)

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Specialized Models for Different Hardware



The cost of ProxylessNAS is at the same level as regular training



ProxylessNAS outperforms MobileNetV2

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Specialized Models for Different Hardware

- GPU: V100
- CPU: 2.40GHz Intel(R) Xeon(R) CPU E5-2640 v4
- Google Pixel 1 phone

Model	Top-1	GPU	CPU	Mobile
Specialized for GPU	75.1	**5.1ms**	204.9ms	124ms
Specialized for CPU	75.3	7.4ms	**138.7ms**	116ms
Specialized for Mobile	74.6	7.2ms	164.1ms	**78ms**

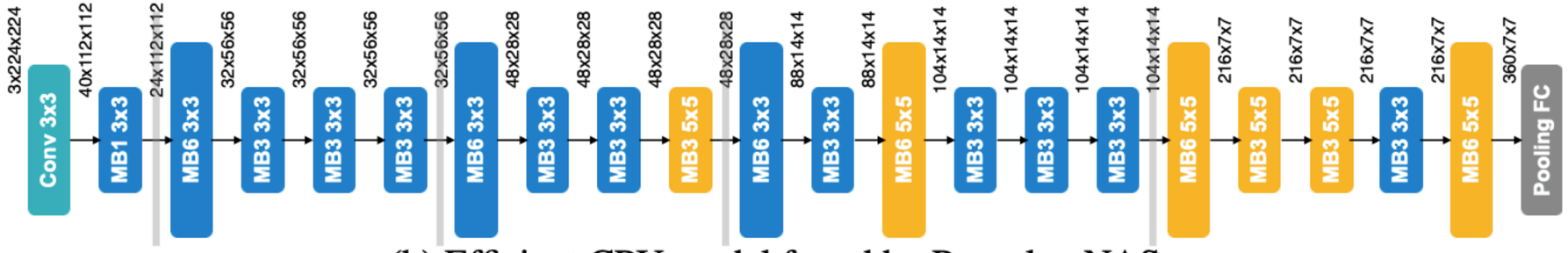
Hardware prefers speacialized models

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Specialized Models for Different Hardware

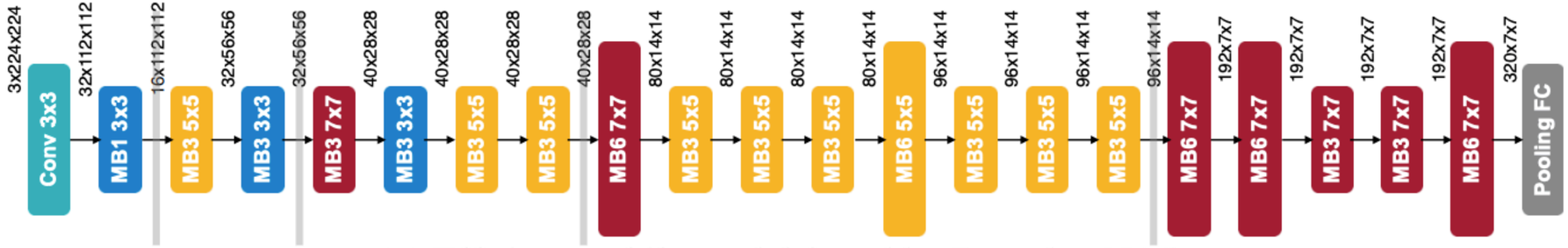
(a) Efficient GPU model found by ProxylessNAS.

Shallower, wider



(b) Efficient CPU model found by ProxylessNAS.

Deeper



(c) Efficient mobile model found by ProxylessNAS.

Wider

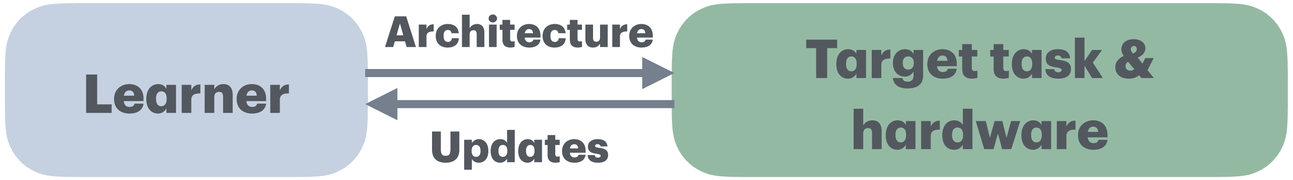
Visualization of ProxylessNAS

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Efficiently Search on Target Task & Hardware

ProxylessNAS - Takeaways

- **Directly** learn neural network architectures on large datasets (e.g., ImageNet) and on target hardware without relying on proxy tasks
- A **path-level pruning** and path-level **binarization** to reduce memory consumption
- The architecture search uses a gradient-based approach (latency regularization loss) to handle hardware objectives (e.g., latency), and the best architecture is discovered by **optimizing both accuracy and hardware efficiency on the specific hardware**
- Once the best architecture is found, it is **trained from scratch** on the dataset and deployed to the target hardware



🤔 **Can we make it better?**

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

How to improve the design productivity for diverse platforms, especially edge devices?

```
For search episodes:
  For training iterations:
    forward-backward();
  If good_model: break;
  For post-search training iterations:
    forward-backward();
```

👉 Manually design

👉 Use NAS to find a specialized NN and train it from scratch for each device

👉 ?

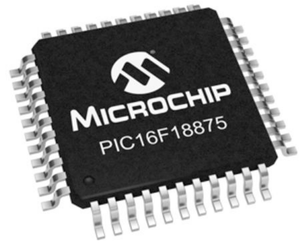
Diverse Platforms and Efficiency Constraints

Cloud AI (10^{12} FLOPS)

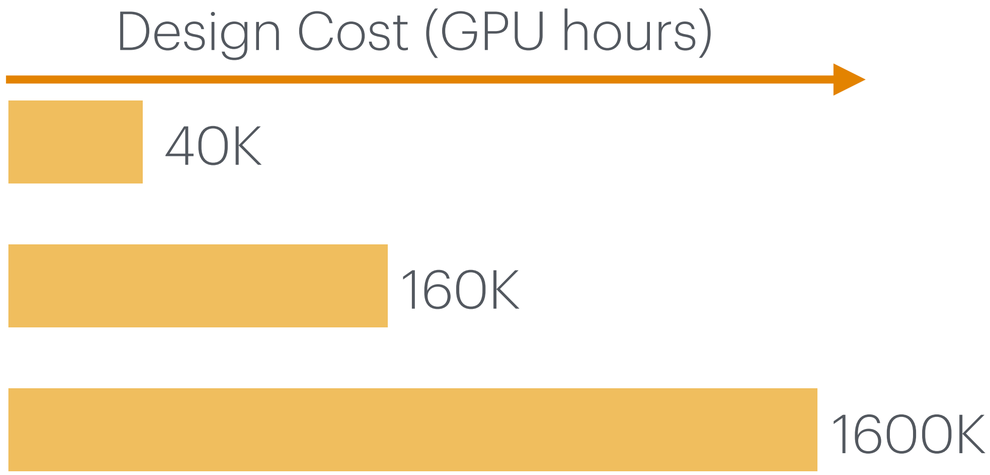
Mobile AI (10^{9} FLOPS)

2023 2019 2017 2014

Tiny AI (10^{6} FLOPS)




```
For many devices:
  For search episodes:
    For training iterations:
      forward-backward();        💰💰💰
    If good_model: break;
    For post-search training iterations:
      forward-backward();        💰💰💰
```



Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 2820-2828).

Hardware-Aware NAS with Once-for-All Network

Rather than training each network from scratch, can we train multiple models at the same time?

- 👉 Manually design
- 👉 Use NAS to find a specialized NN and train it from scratch for each device
- 👉 Once-for-All network

```
For OFA training iterations:
  forward-backword();
For many devices:
    For search episodes:
      For training iterations:
        forward-backward();
      Sample from OFA;        🤑🤑🤑
      If good_model: break;
```

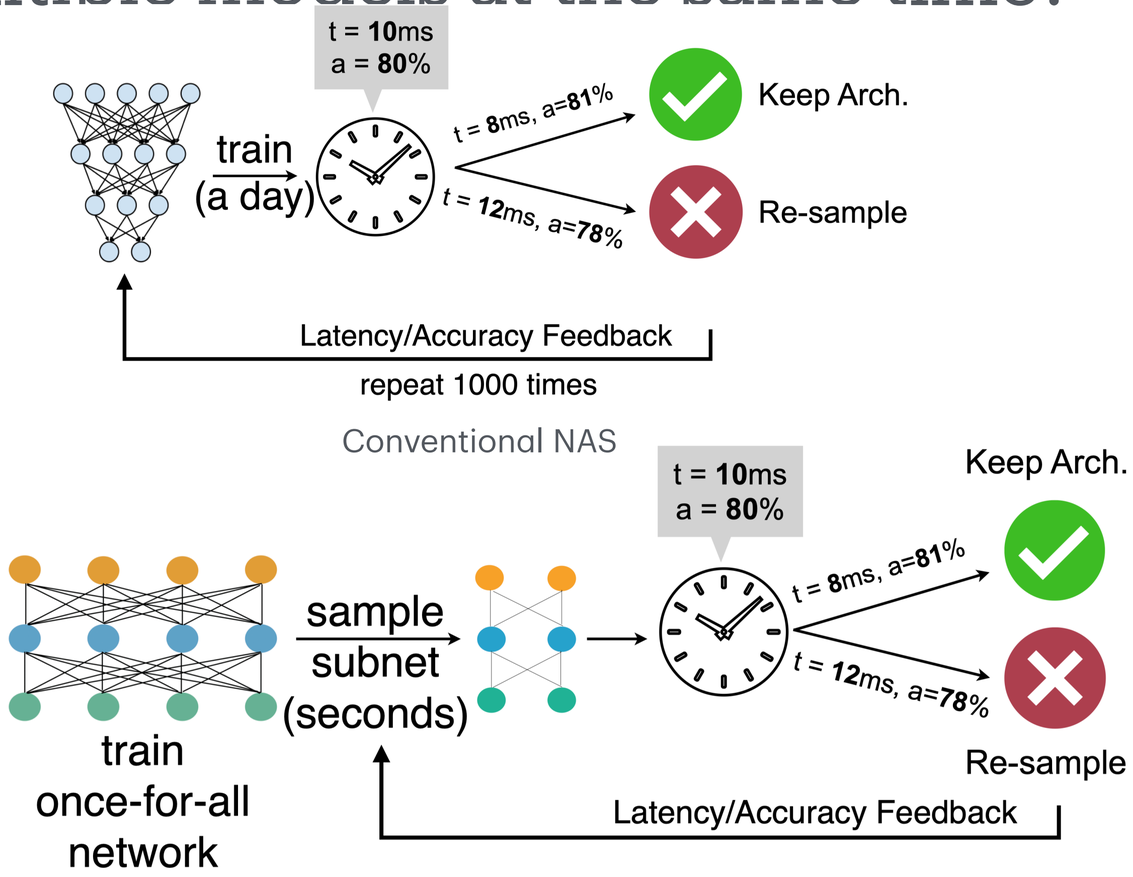
- **Decouple** training and search
  - Training:
    - Train a once-for-all network, sparsely activated
  - Search:
    - Select sub-network, get accuracy and latency
    - Repeat and select the best

🤑 **Amortize the cost**

Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Hardware-Aware NAS with Once-for-All Network

Rather than training each network from scratch, can we train multiple models at the same time?

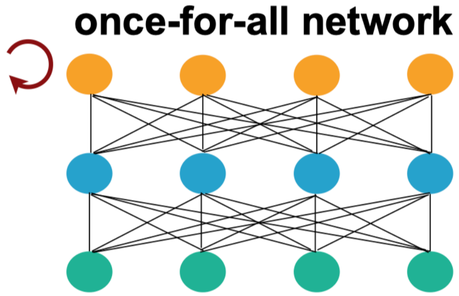


- **Decouple** training and search
  - Training:
    - Train a once-for-all network, sparsely activated
  - Search:
    - Select sub-network, get accuracy and latency
    - Repeat and select the best

Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Once-for-All Network

Decouple module training and architecture design



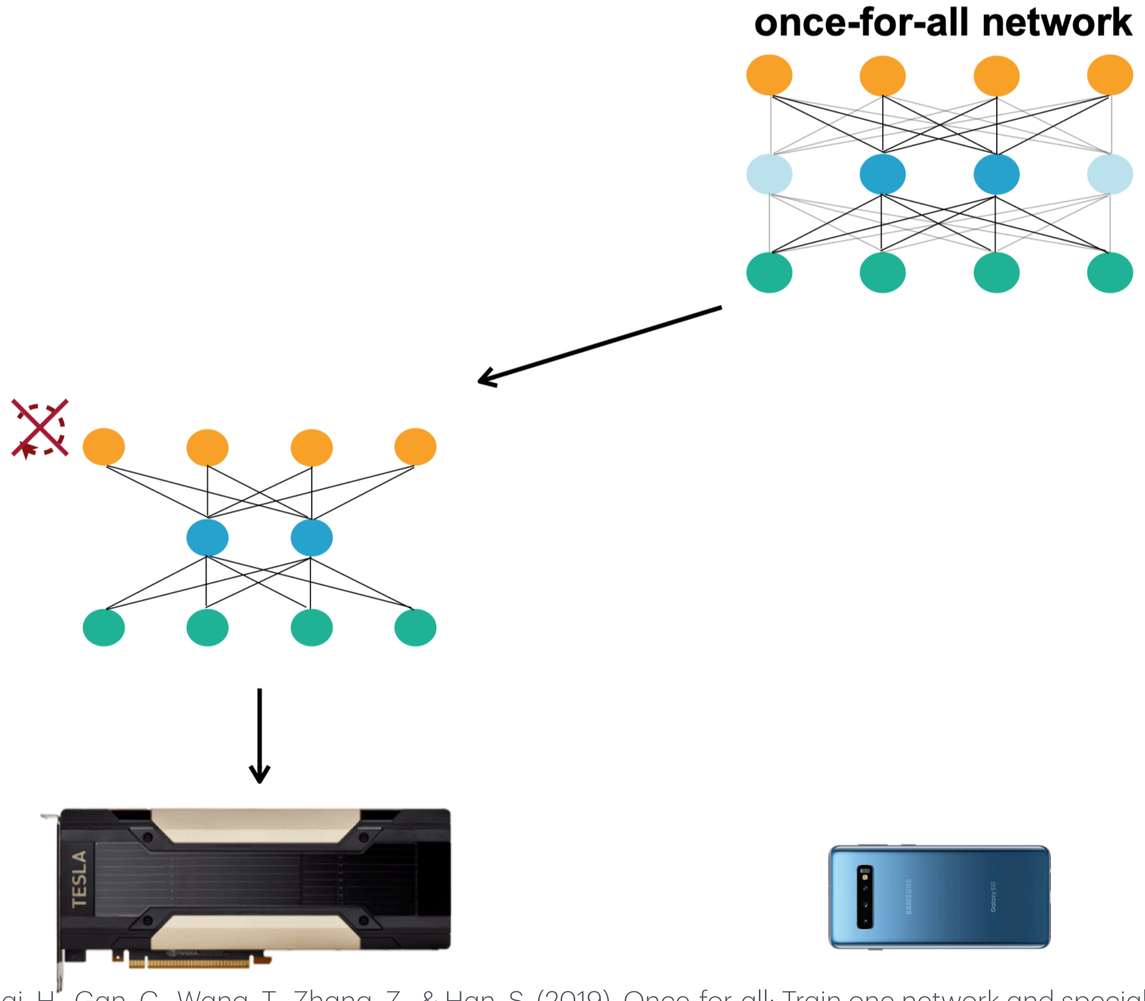
Train once

OFA network contains many child networks that are sparsely activated

Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Once-for-All Network

Decouple module training and architecture design



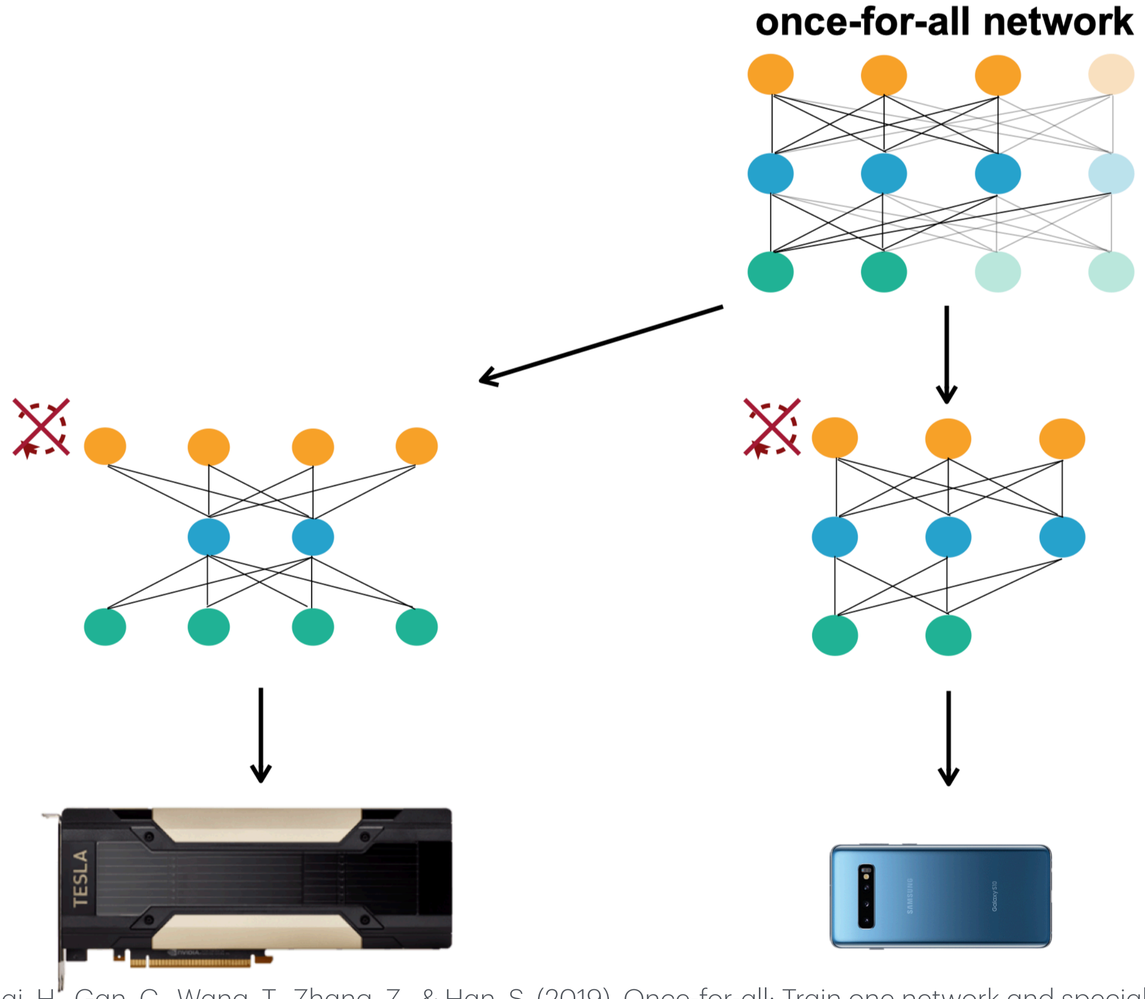
Train once

Get many

Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Once-for-All Network

Decouple module training and architecture design



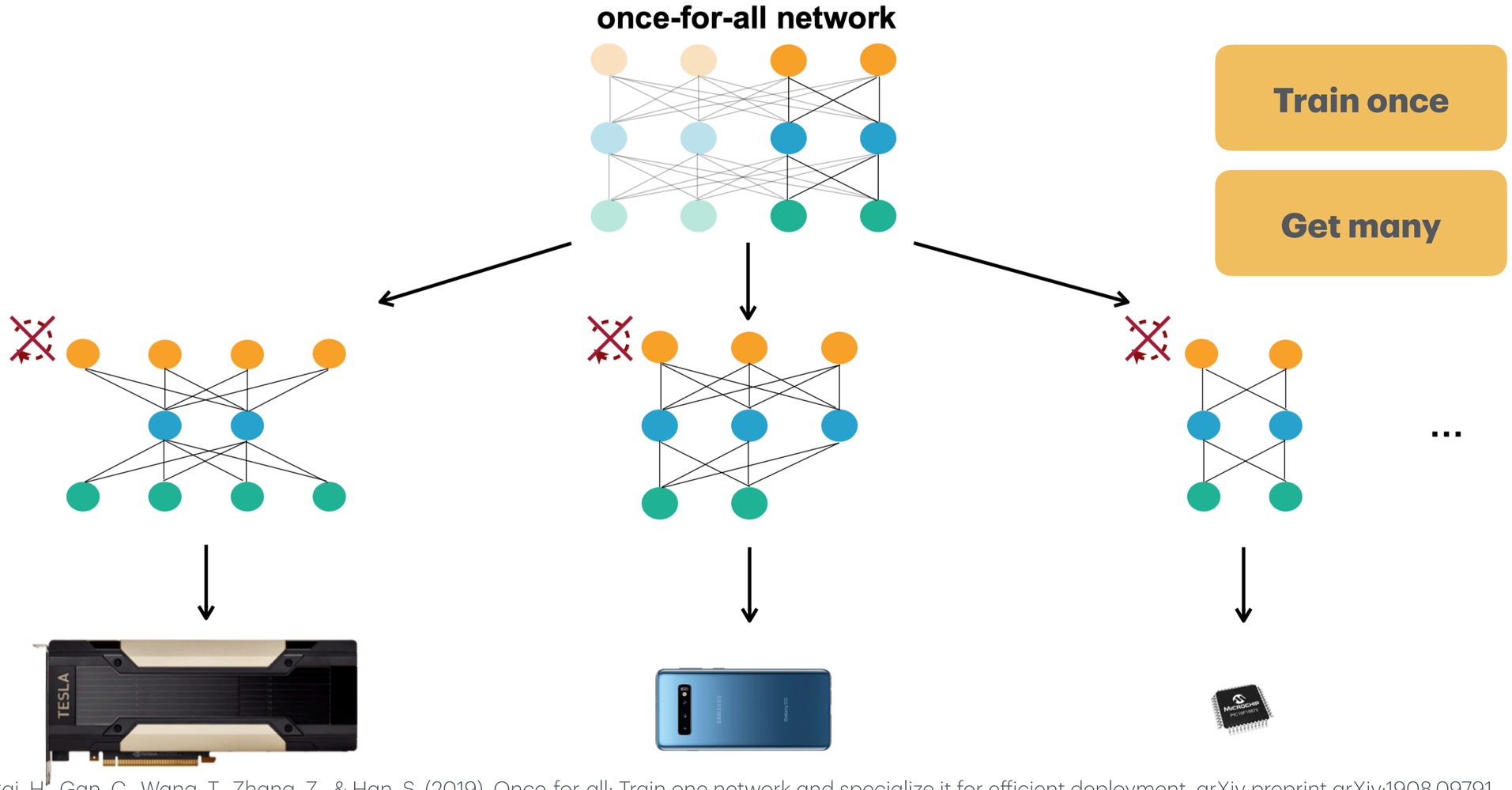
Train once

Get many

Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Once-for-All Network

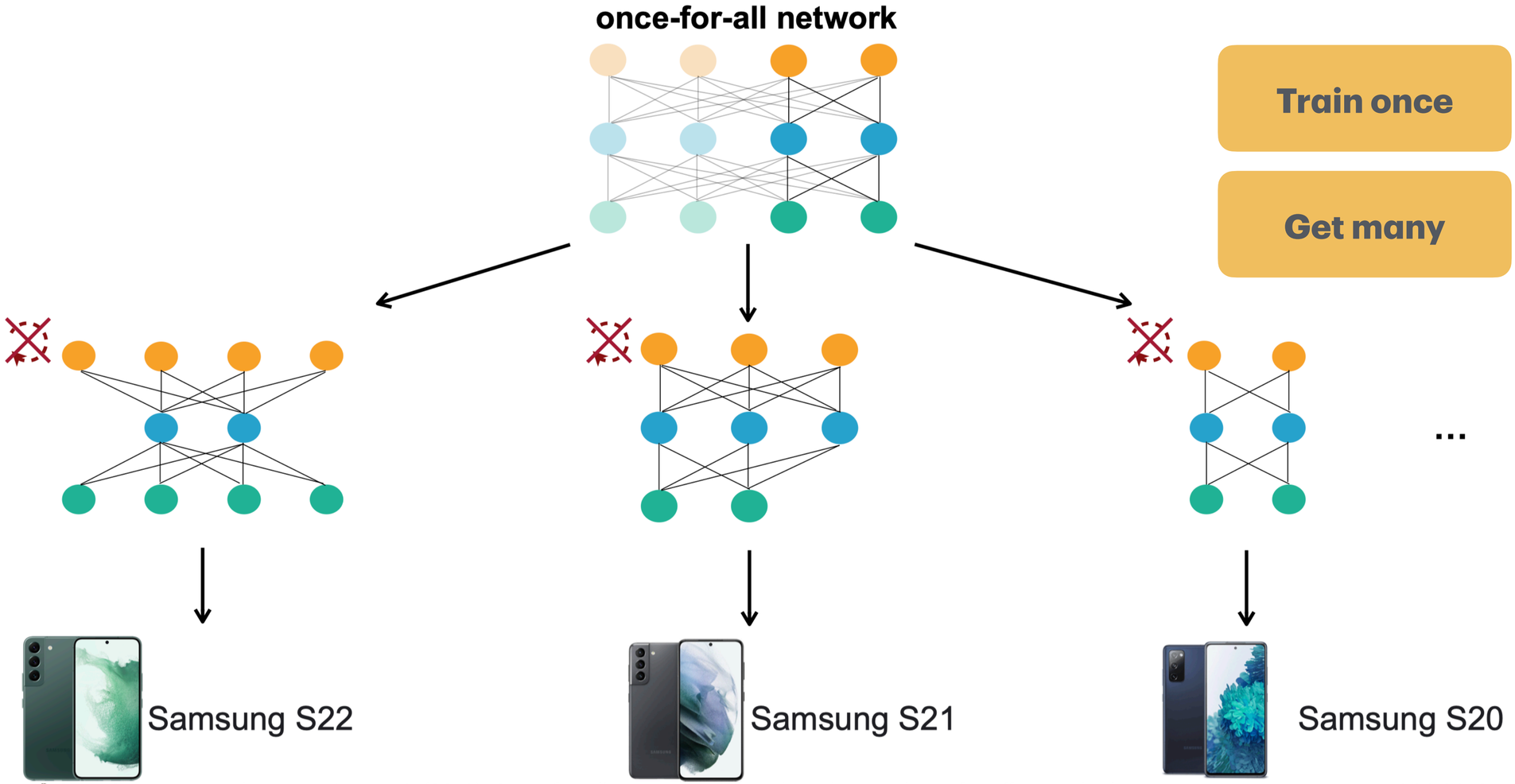
Decouple module training and architecture design



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Once-for-All Network

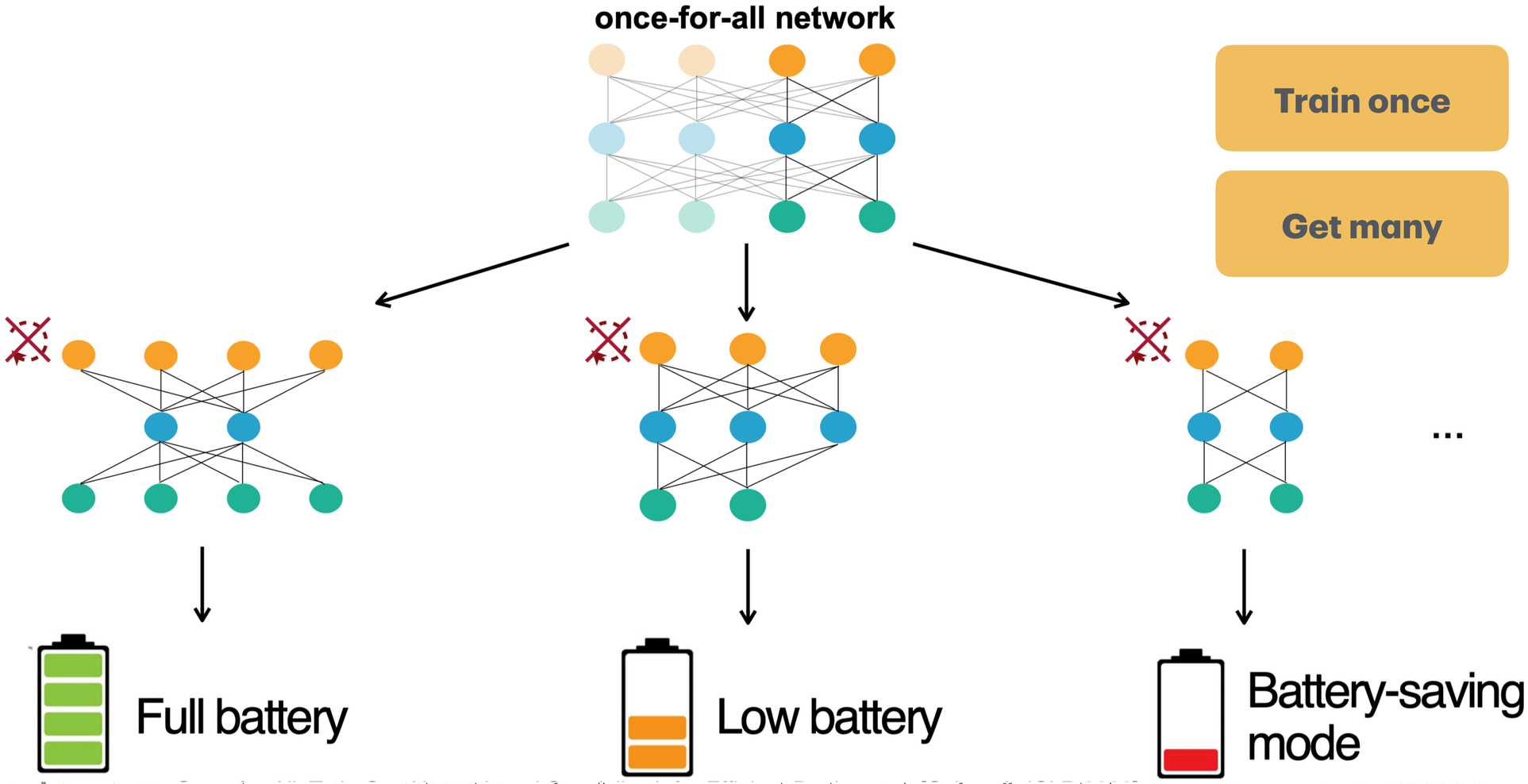
Decouple module training and architecture design



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

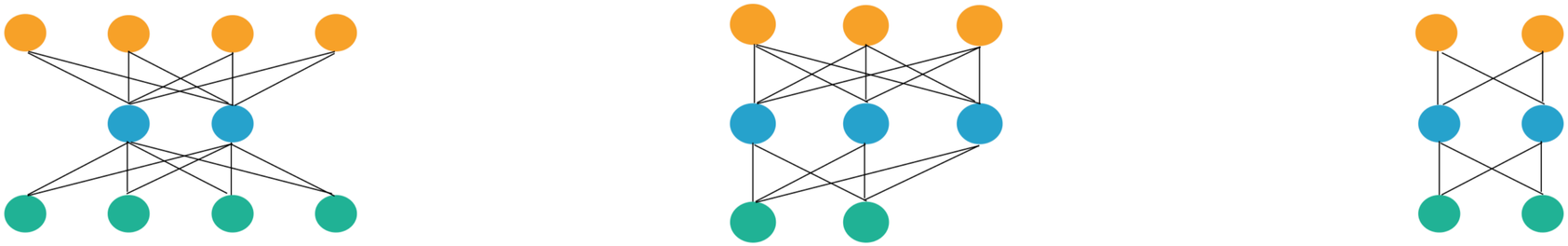
Once-for-All Network

Decouple module training and architecture design



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

How to prevent different subnetworks from interfering with each other?



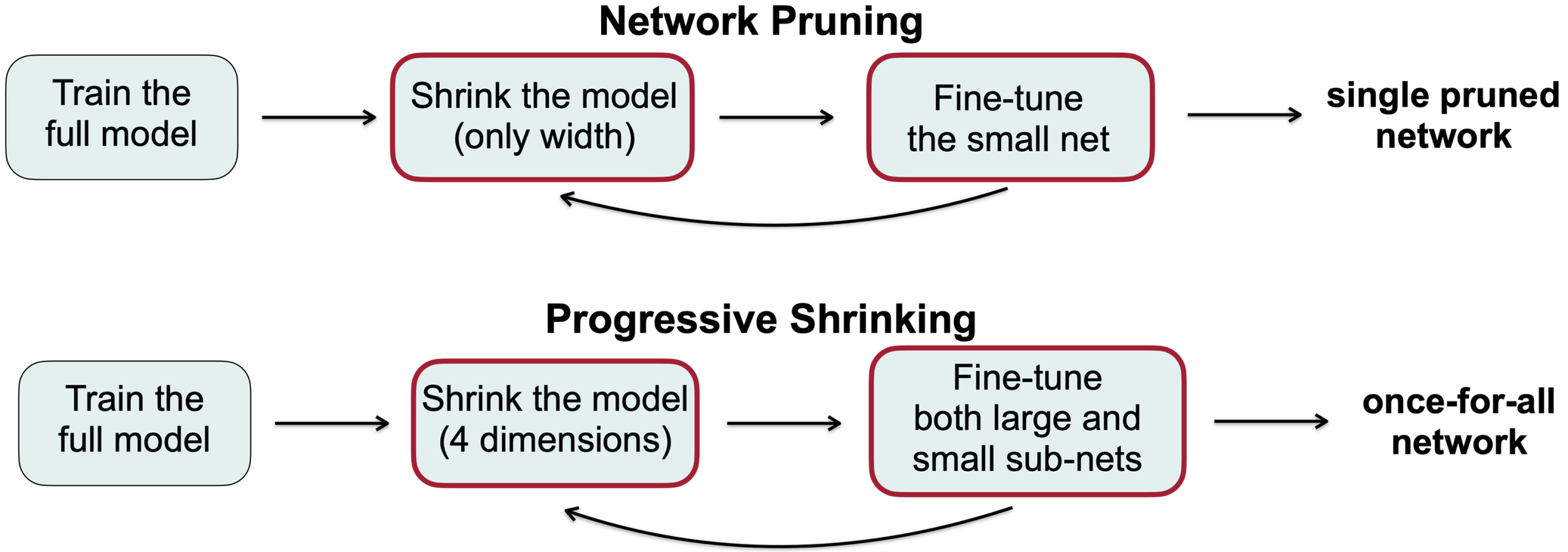
Solution: Progressive Shrinking

Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

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

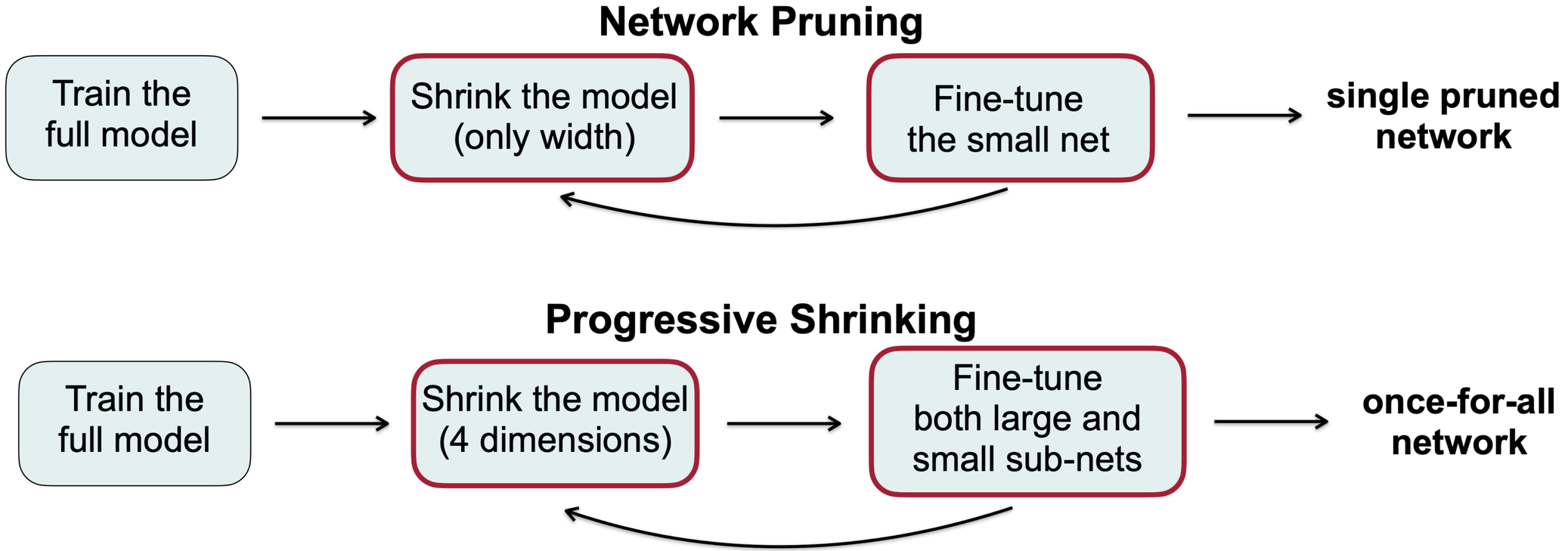
- Directly optimizing the once-for-all network from scratch is much more challenging than training a normal neural network given so many sub-networks to support



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

A generalized network pruning with much higher flexibility across 4 dimensions

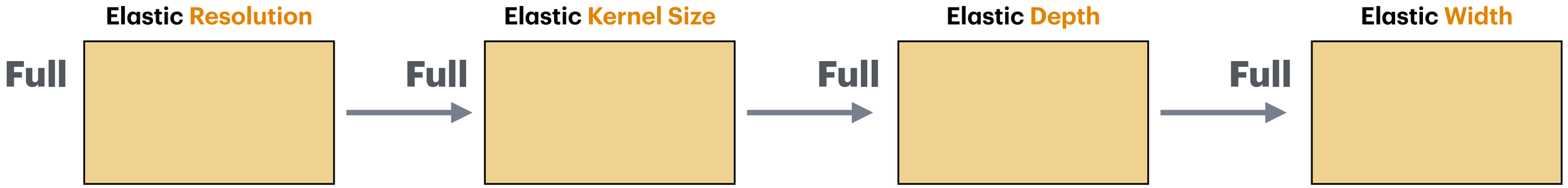


- Small sub-networks are **nested** in large sub-networks
- Cast the training process of the once-for-all network as a **progressive shrinking and joint fine-tuning** process

Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

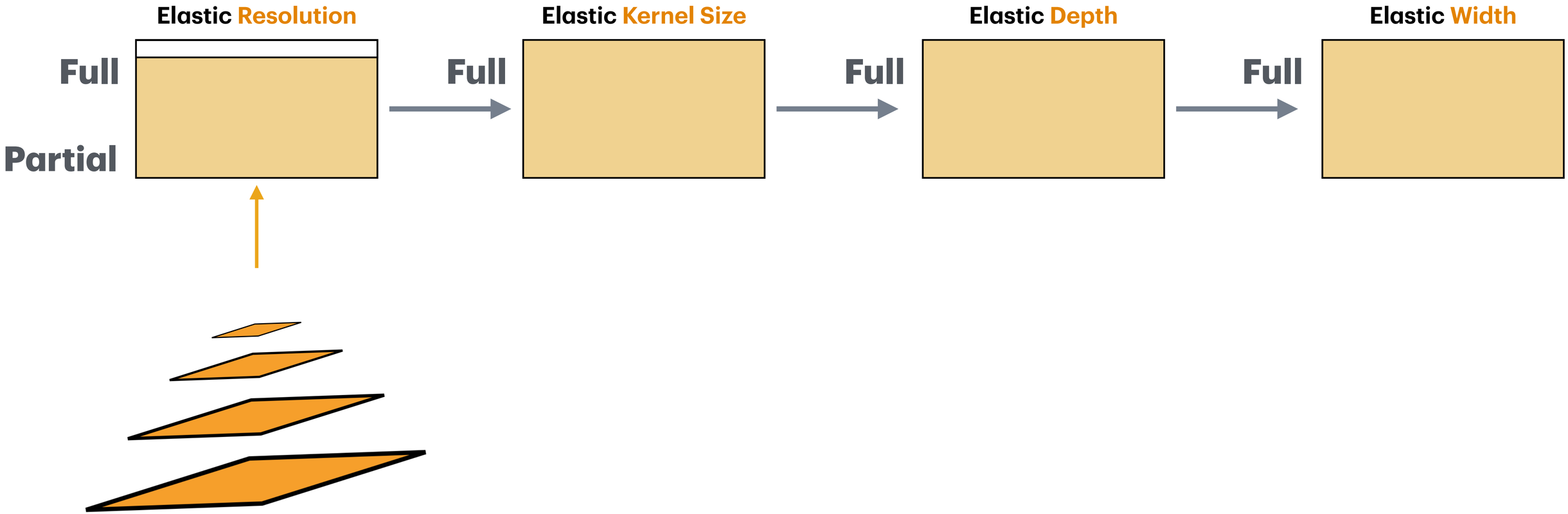
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

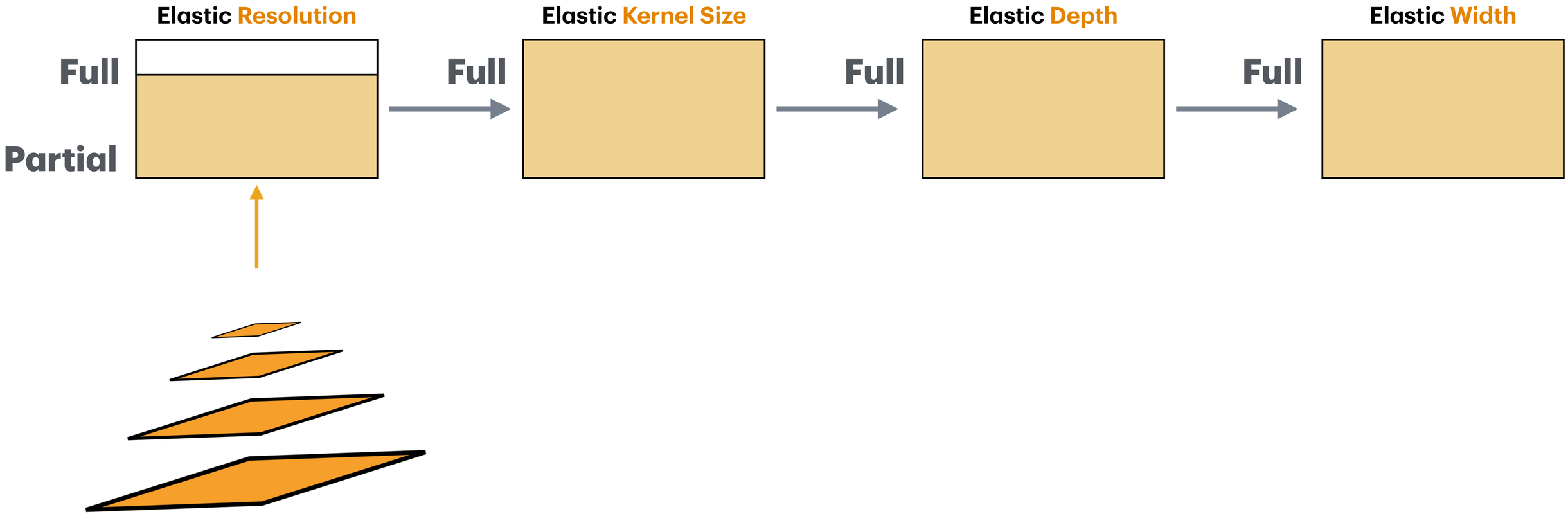
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

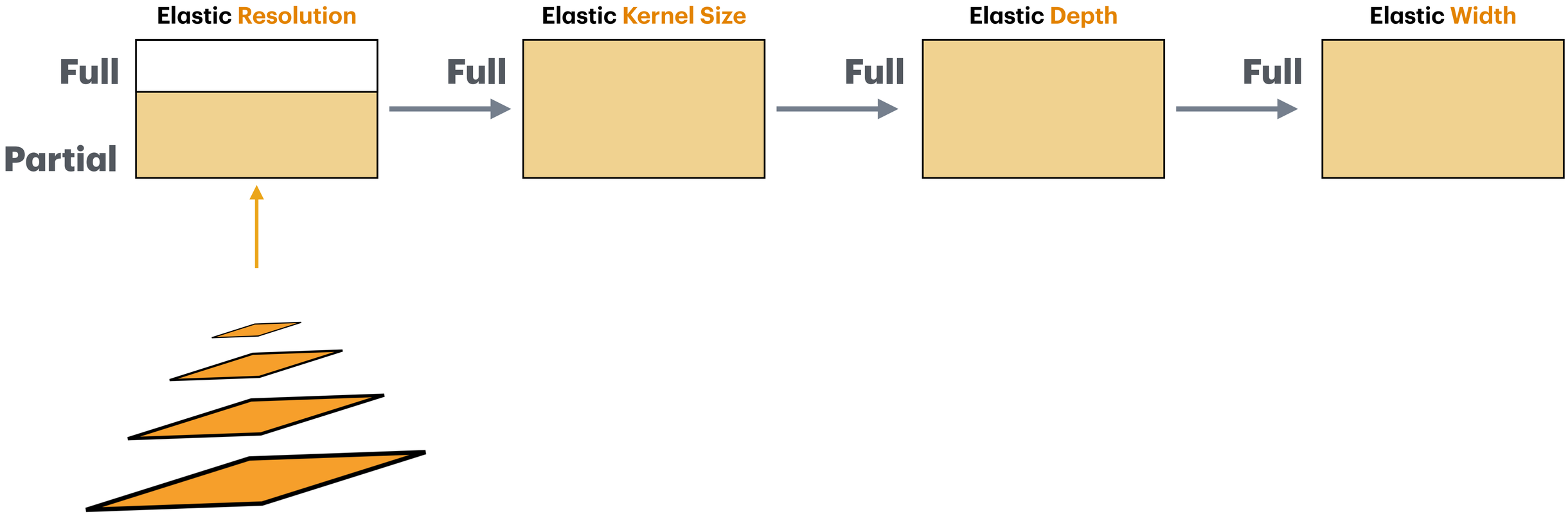
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

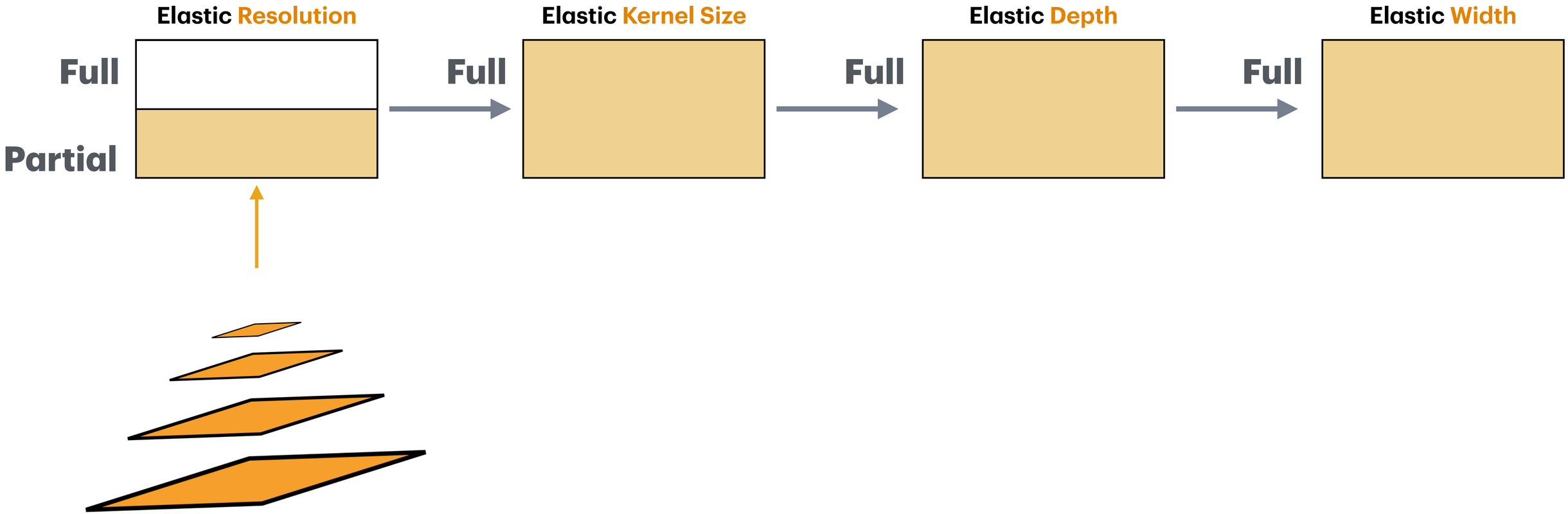
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

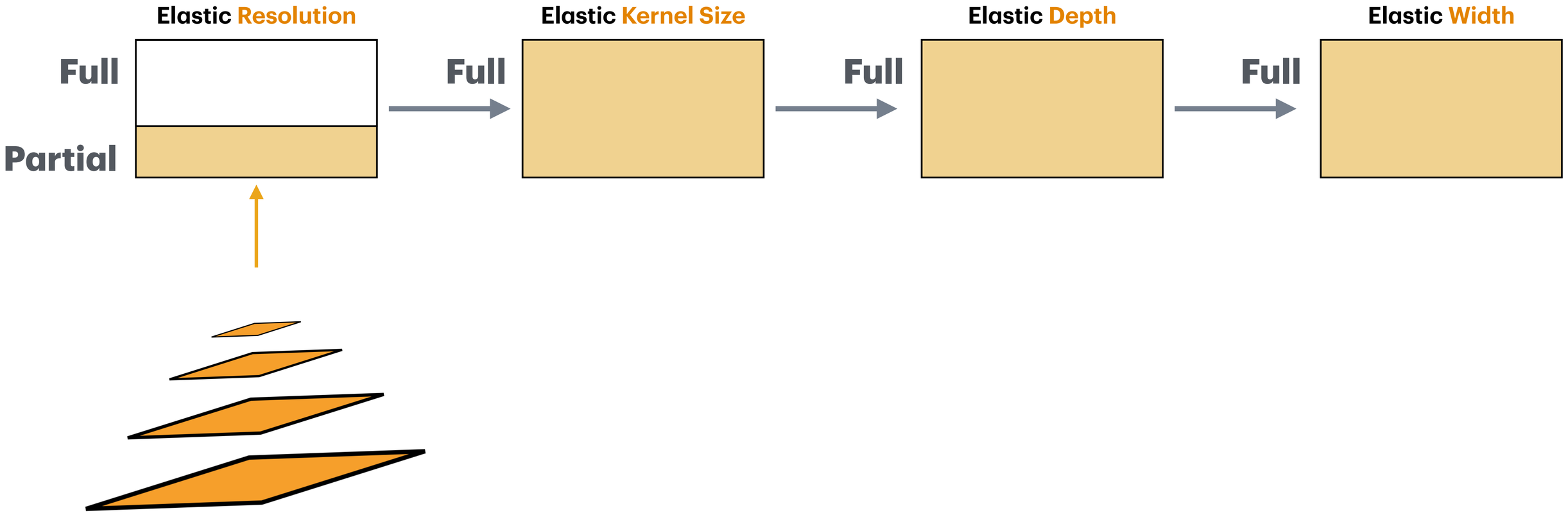
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

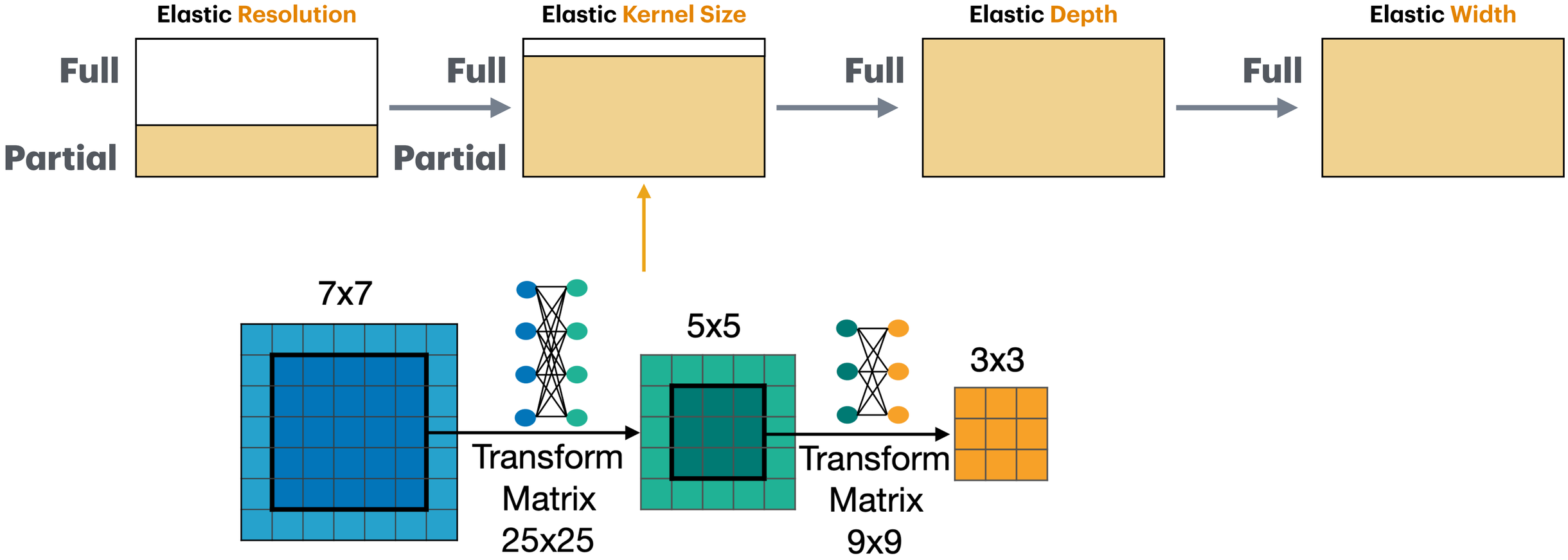
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

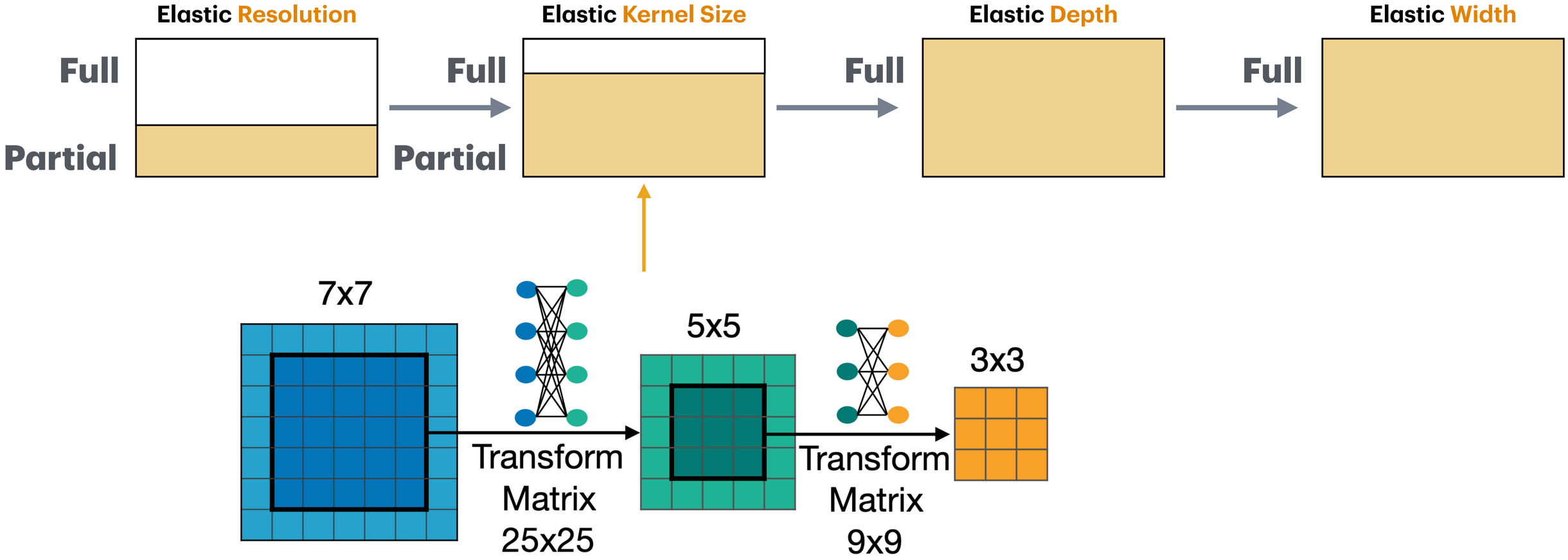
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

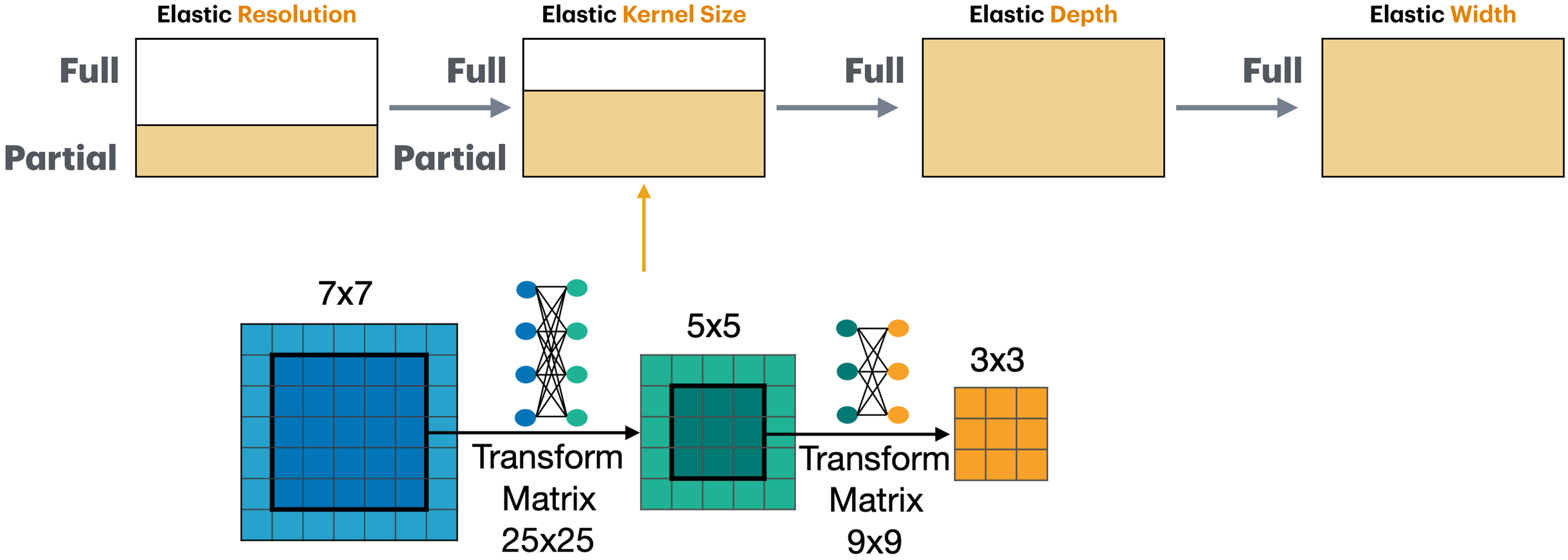
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

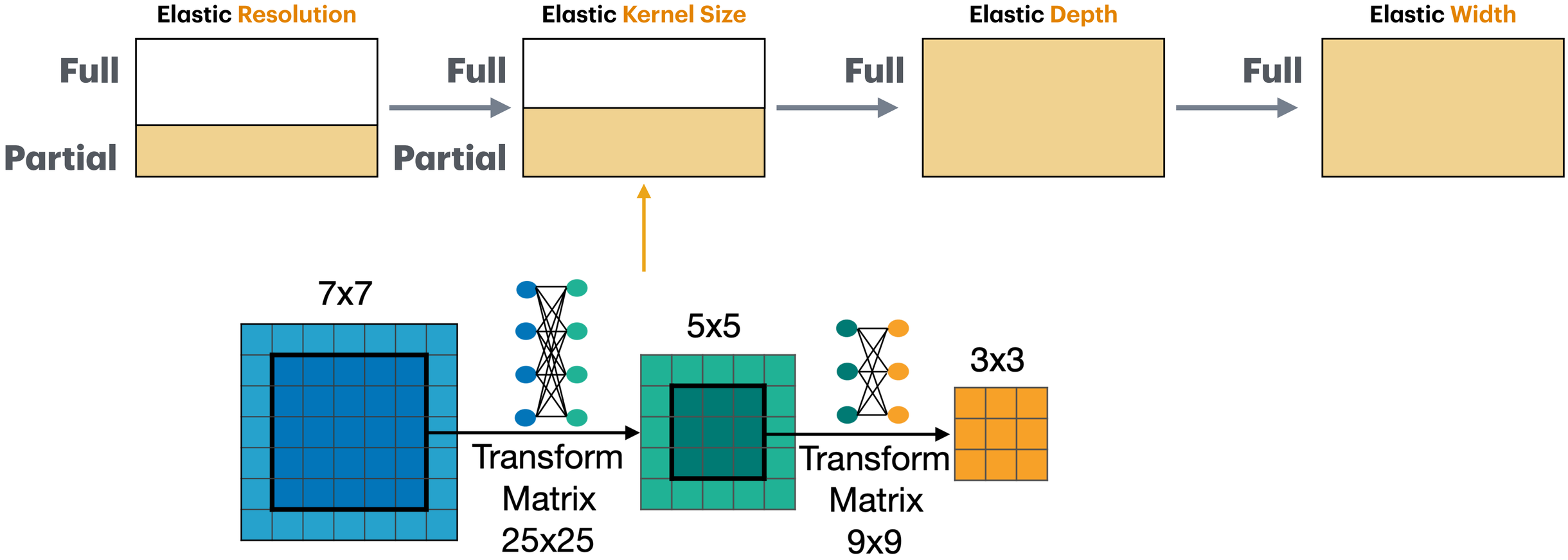
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

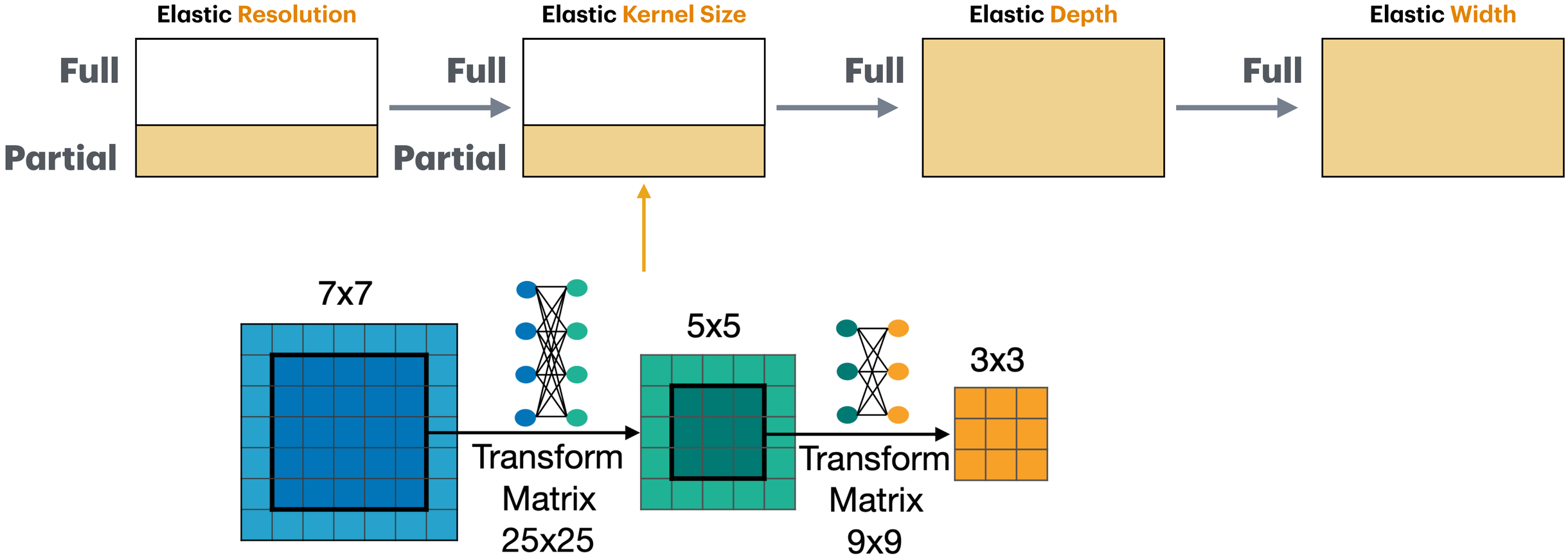
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

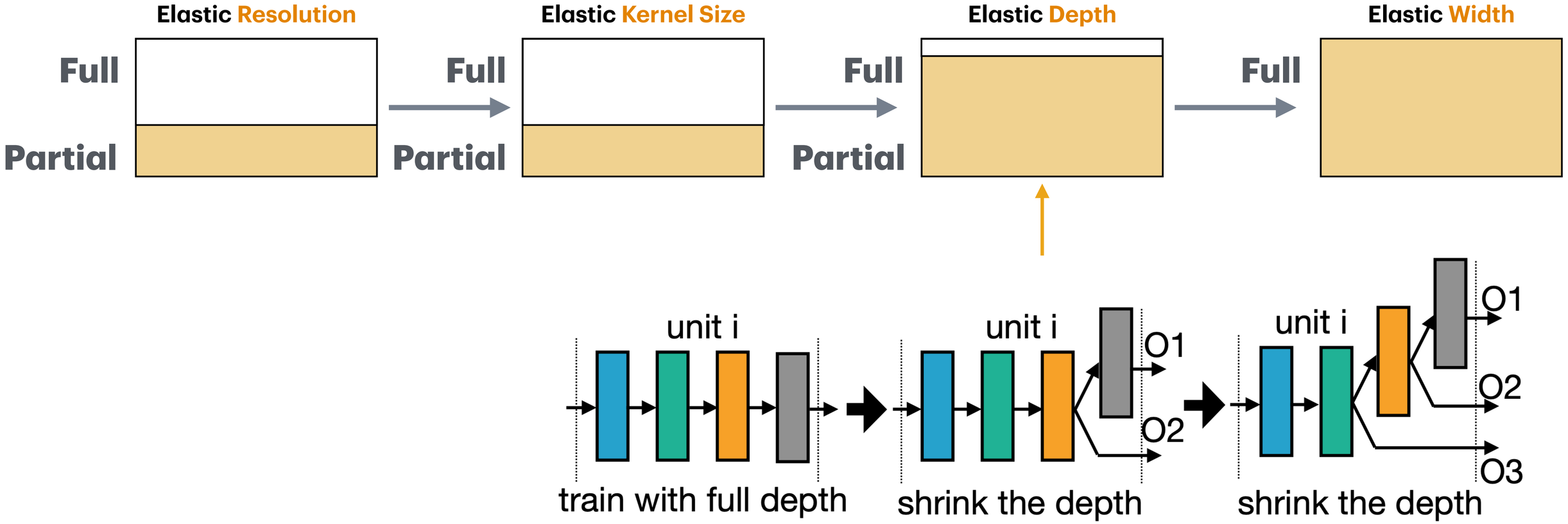
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

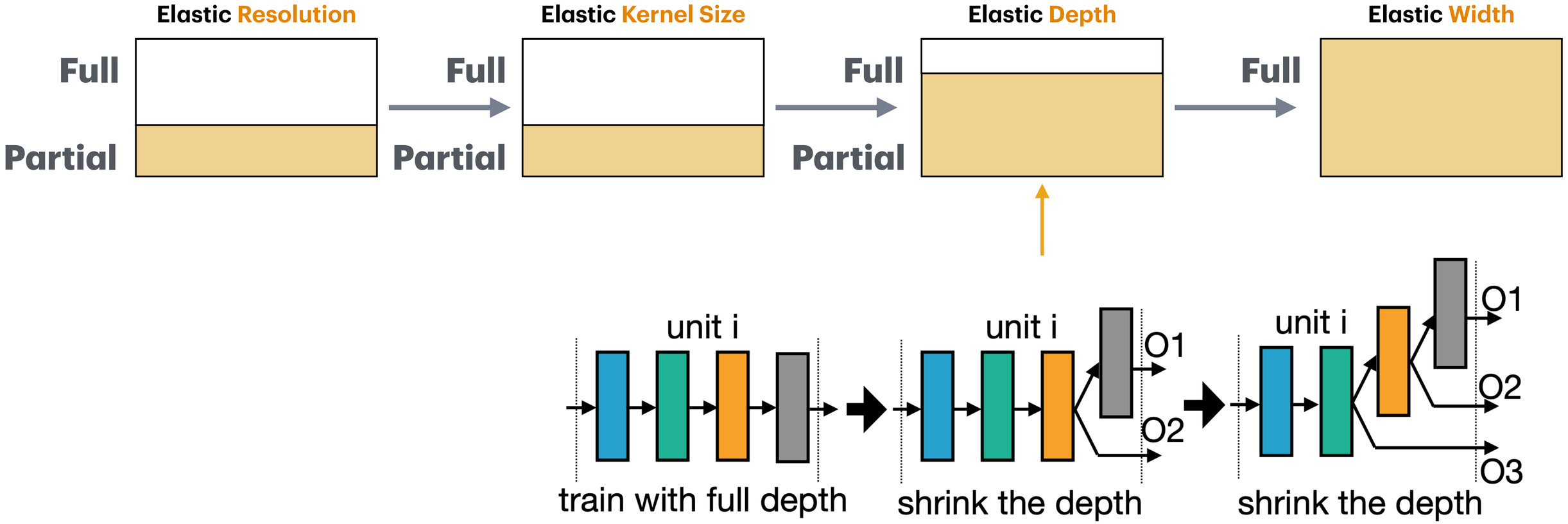
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

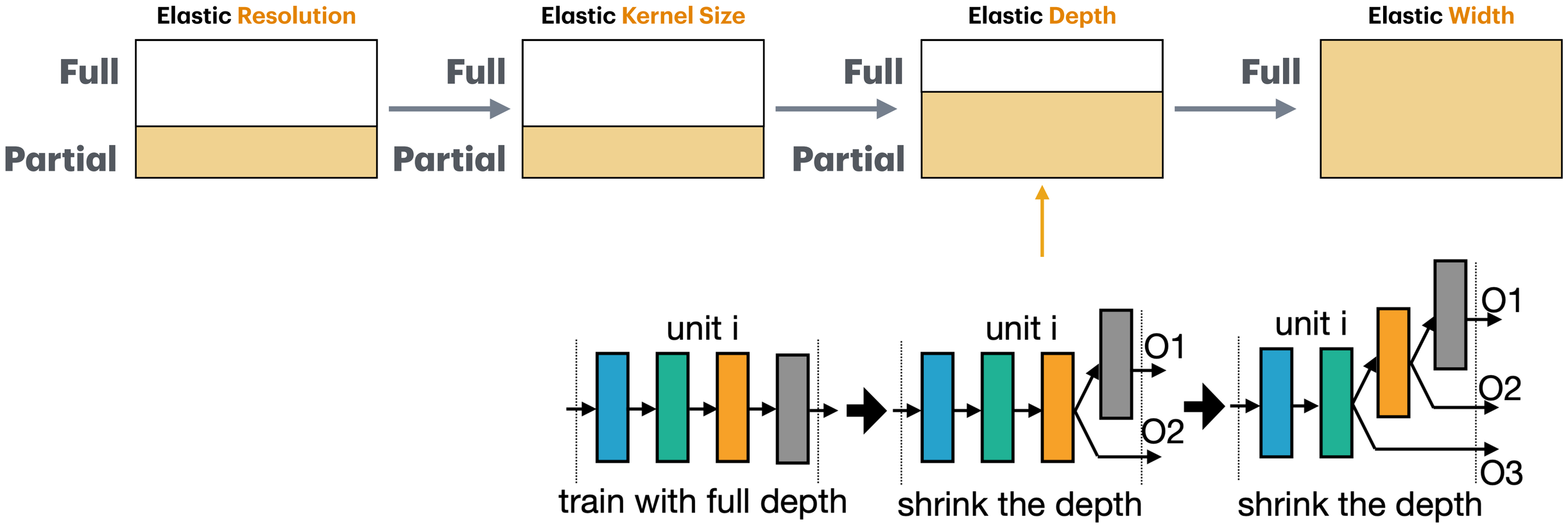
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

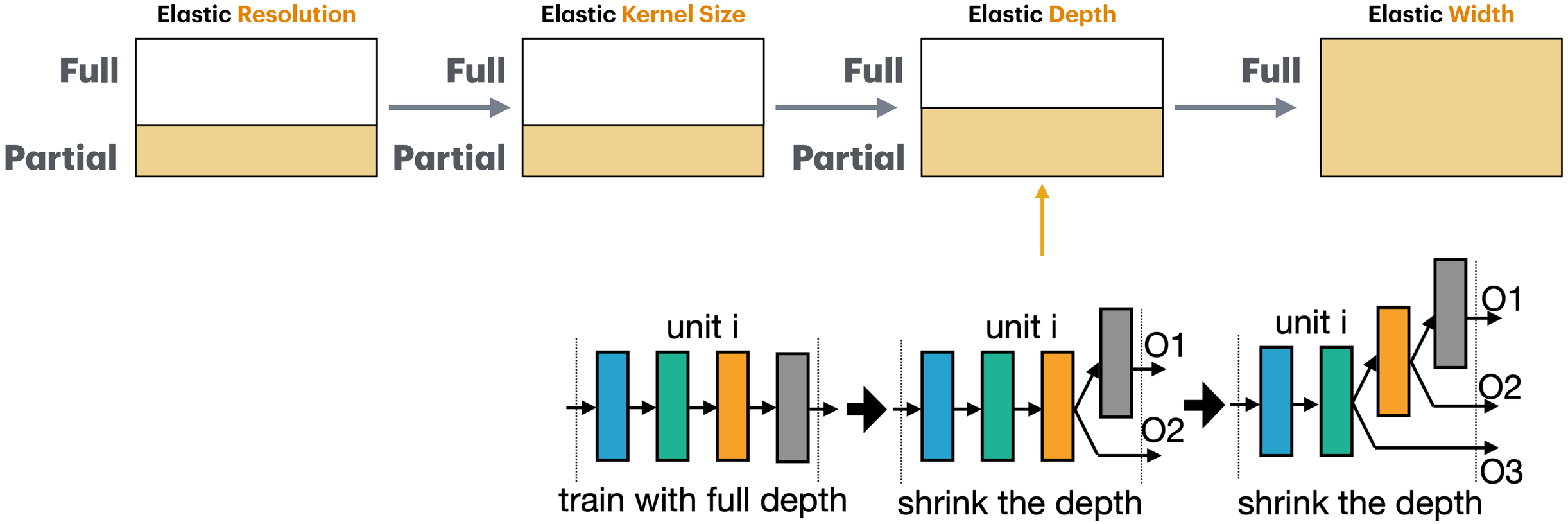
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

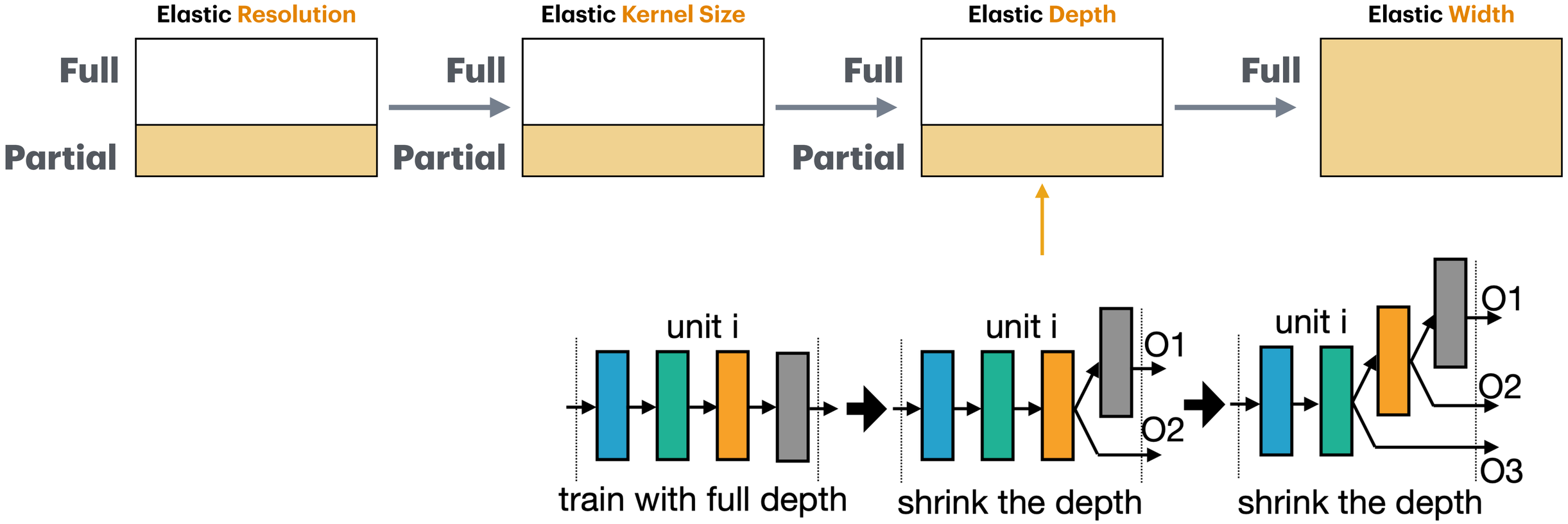
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

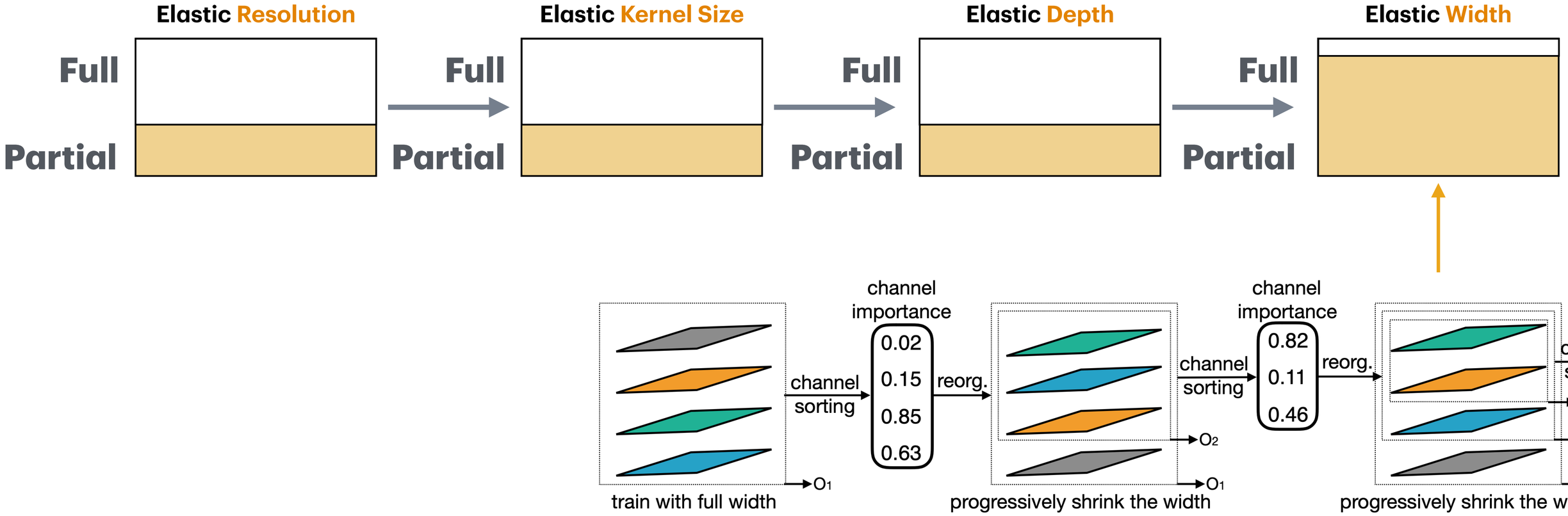
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

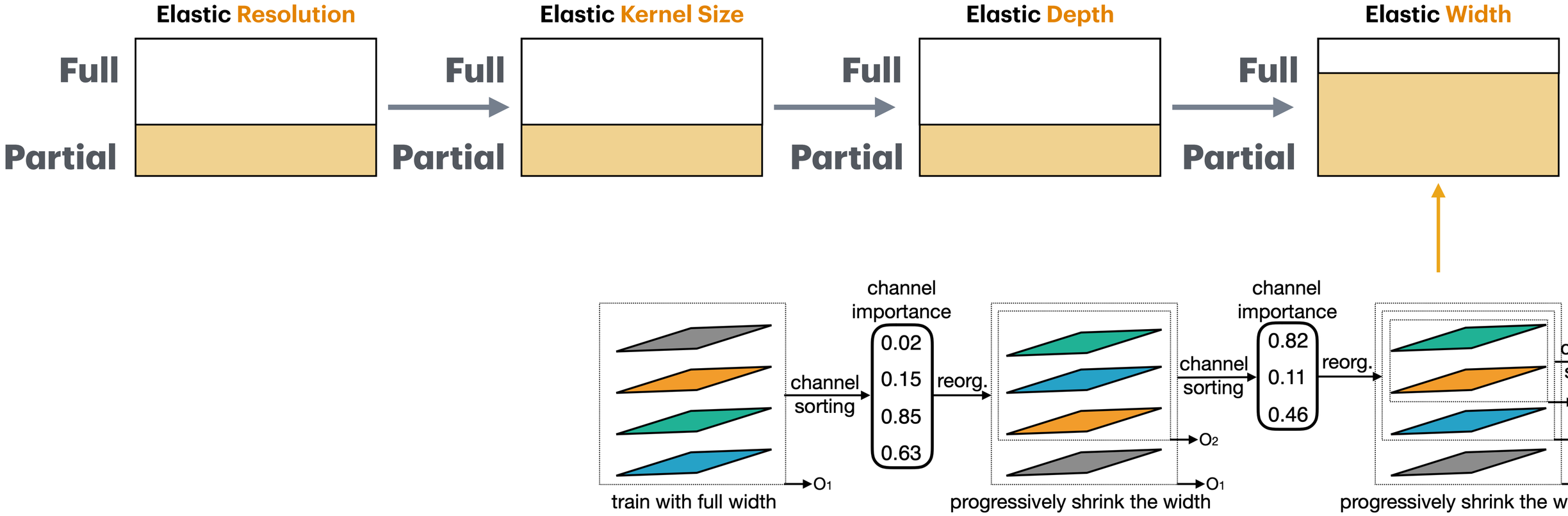
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

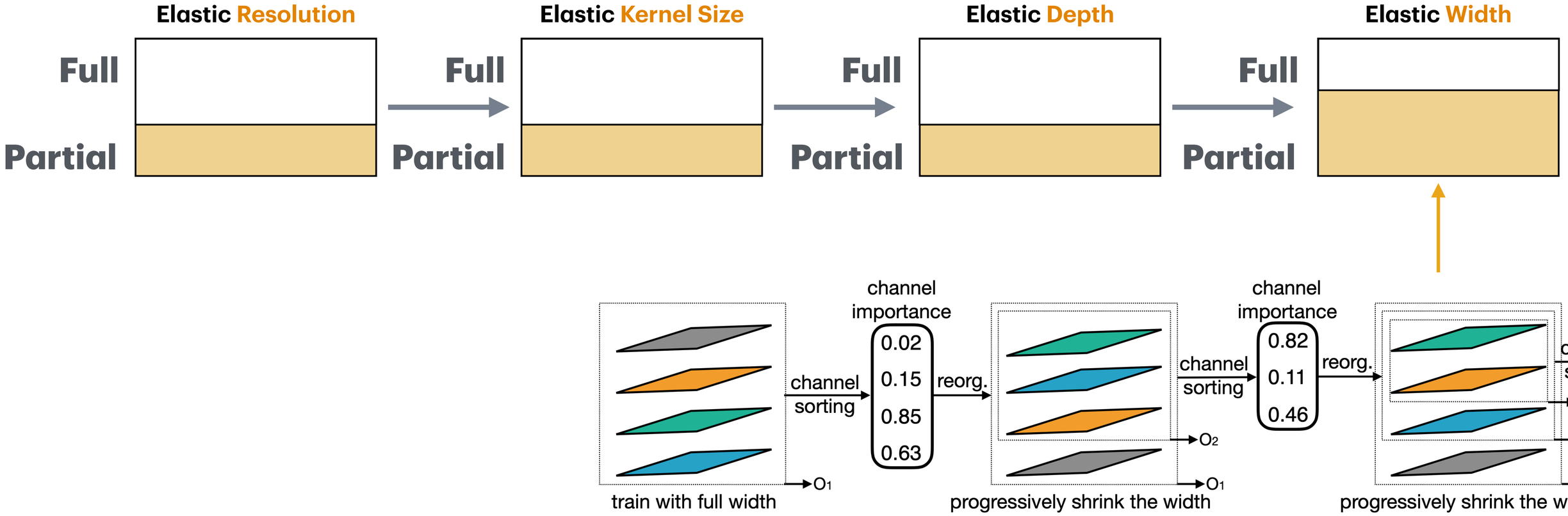
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

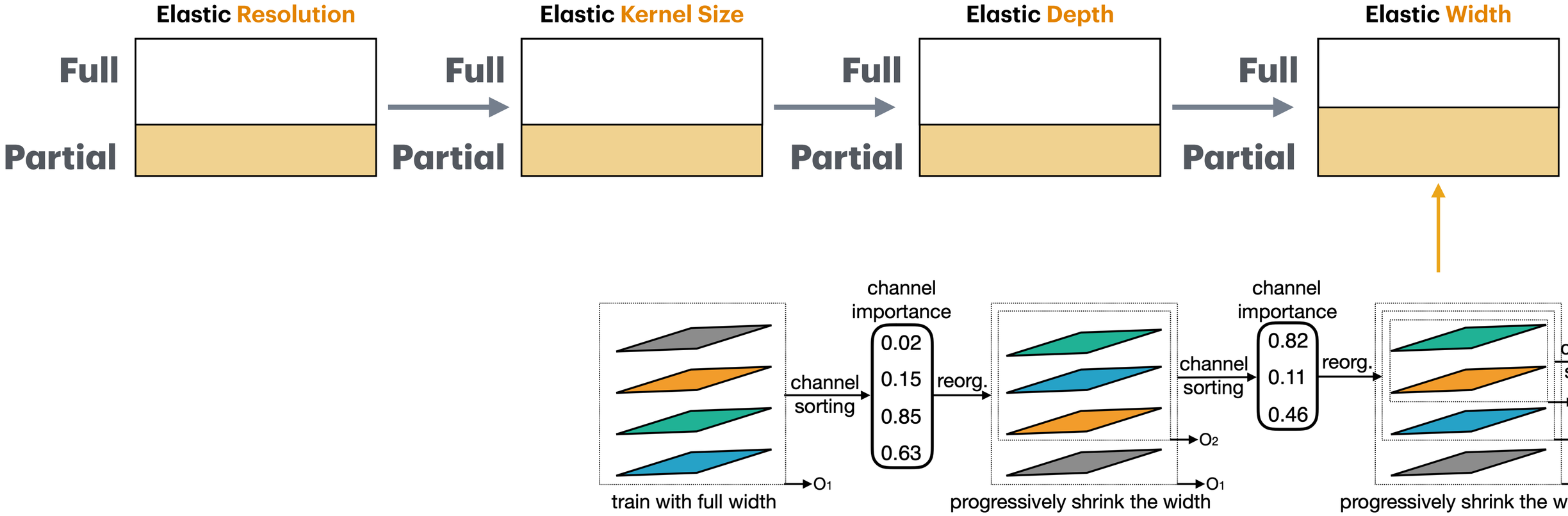
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking

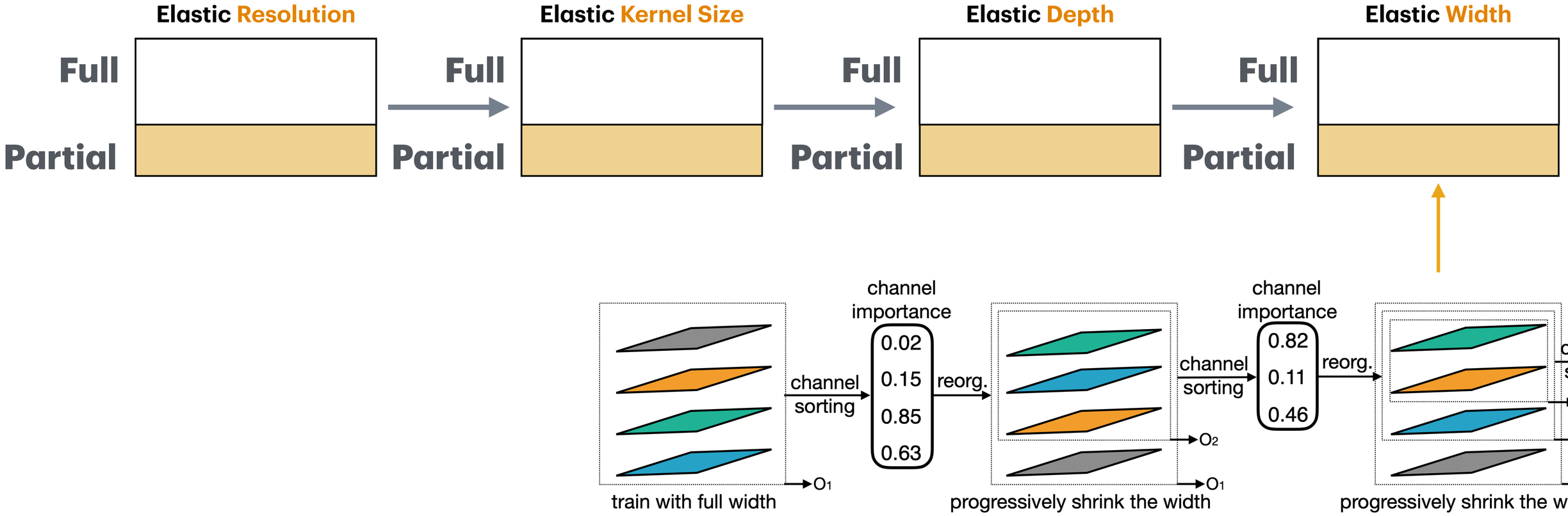
Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

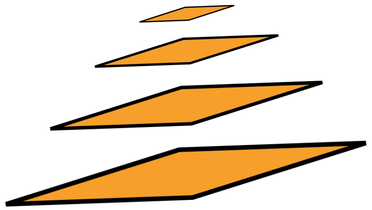
Progressive Shrinking

Progressively prune the kernel size, depth, resolution



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Progressive Shrinking: Put it Together



Train full network

$K = 7$
$D = 4$
$W = 6$

Elastic resolution

$R \in [128,132,...,224]$

$K \in [7,5,3]$

Sample K at each layer

Generate kernel weights

Fine-tune weights & transformation matrix

Elastic kernel size

$D = 4, W = 6$

$D \in [4,3,2]$

Sample D at each unit; sample K

Keep the first D layers at each unit

Fine-tune weights

Elastic Depth

$W = 6, K \in [7,5,3]$

$W \in [6,4,3]$

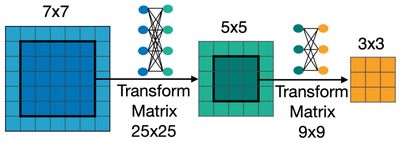
Channel sorting

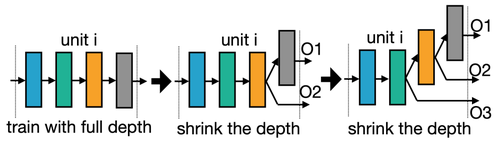
Sample W at each layer; sample K, D

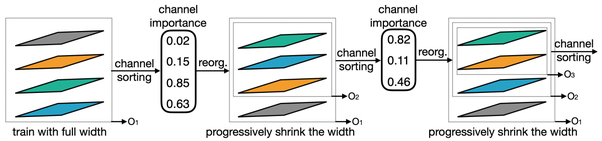
Fine-tune weights

Elastic Width

$D \in [4,3,2], K \in [7,5,3]$

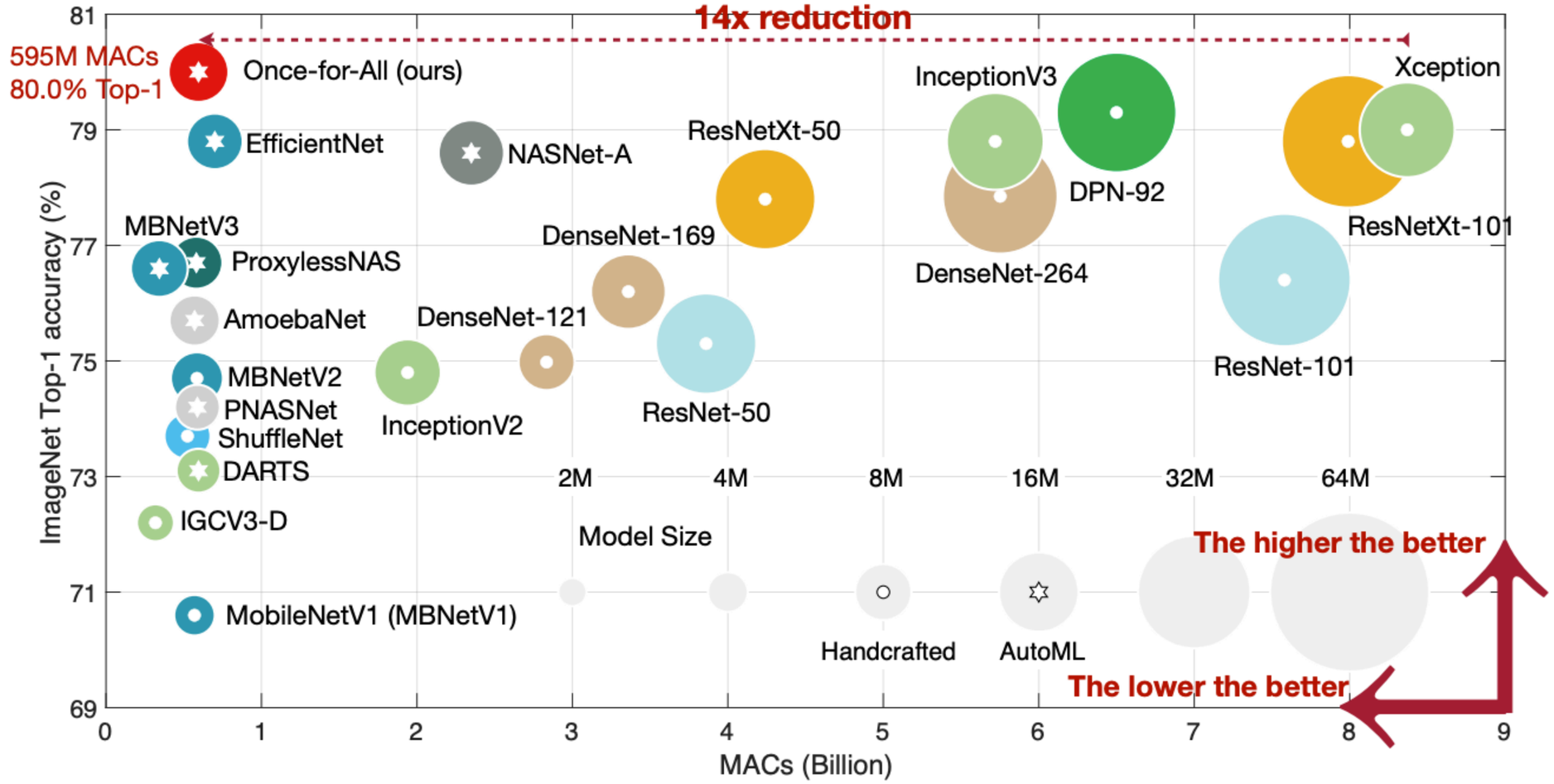






OFA vs. Others

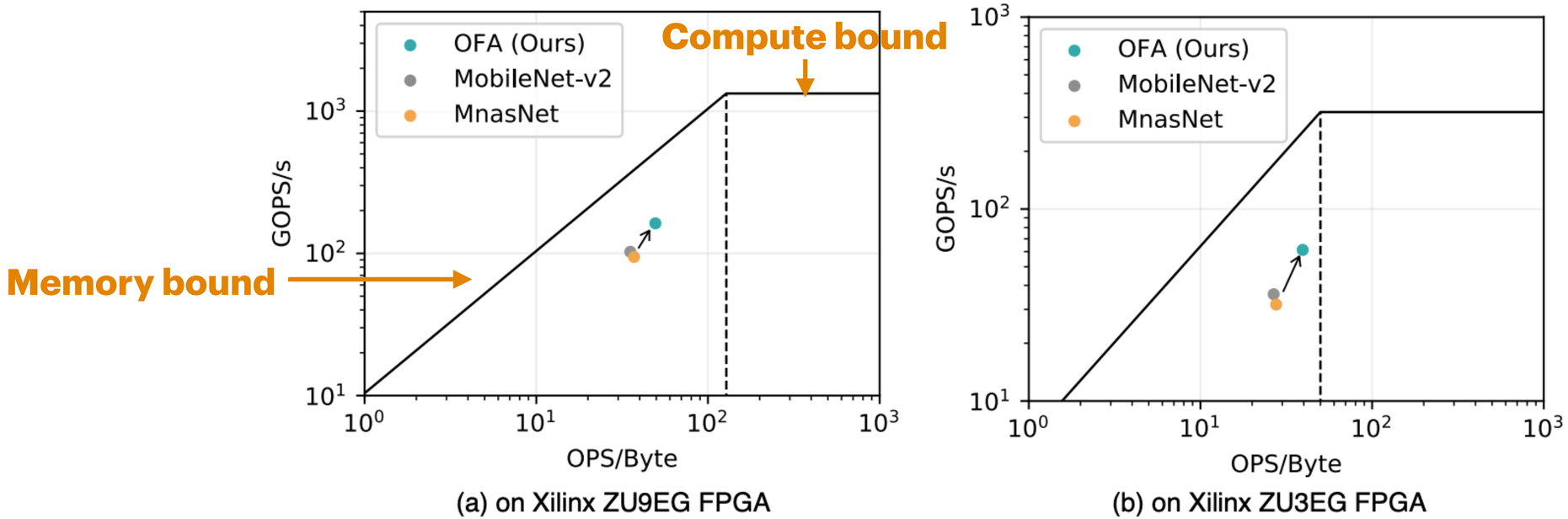
80% Top-1 Accuracy on ImageNet



Once-for-All Network: Roofline Analysis

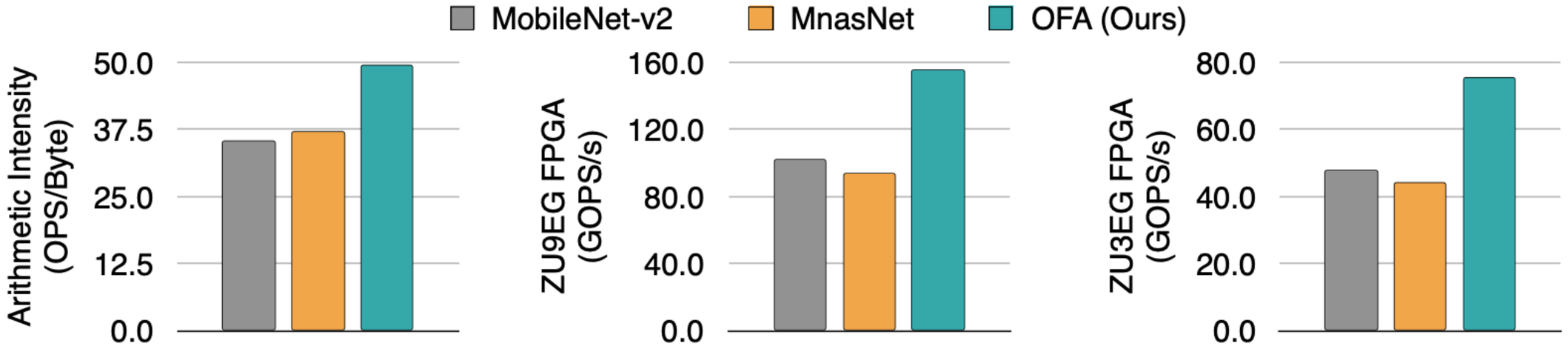
Computation is cheap; memory is expensive

- OFA-designed model has **higher arithmetic intensity** (Ops/Byte)
  - Less memory bounded
  - Higher utilization and performance without changing the RTL (Register Transfer Level)



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

OFA for FPGA Accelerators



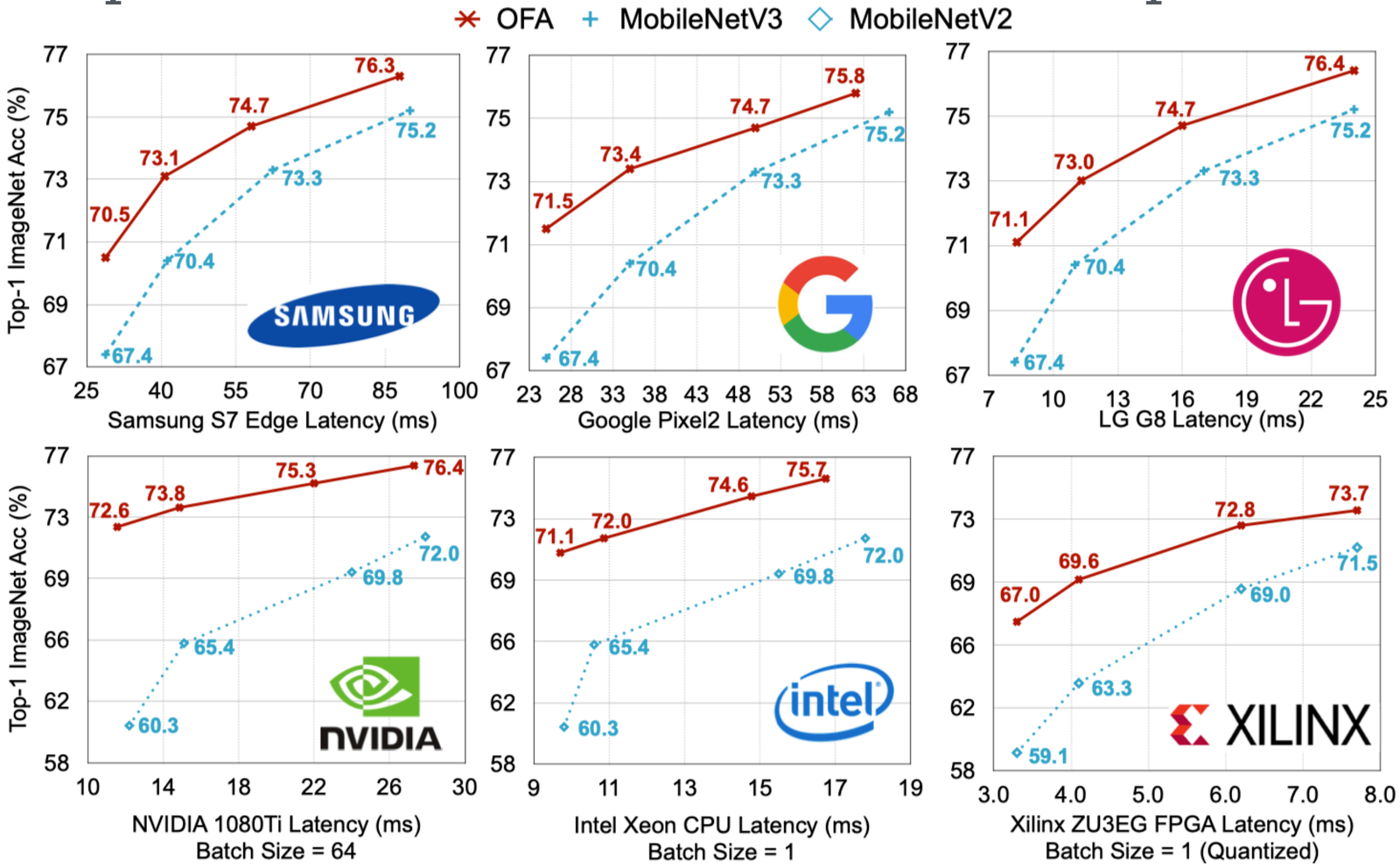
Measured results on XILINX FPGA

- Non-specialized neural networks do **not fully utilize** the hardware resource. There is a large room for improvement via neural network specialization
- OFA models improve the **arithmetic intensity** (OPS/Byte) and **utilization** (GOPS/s)

Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

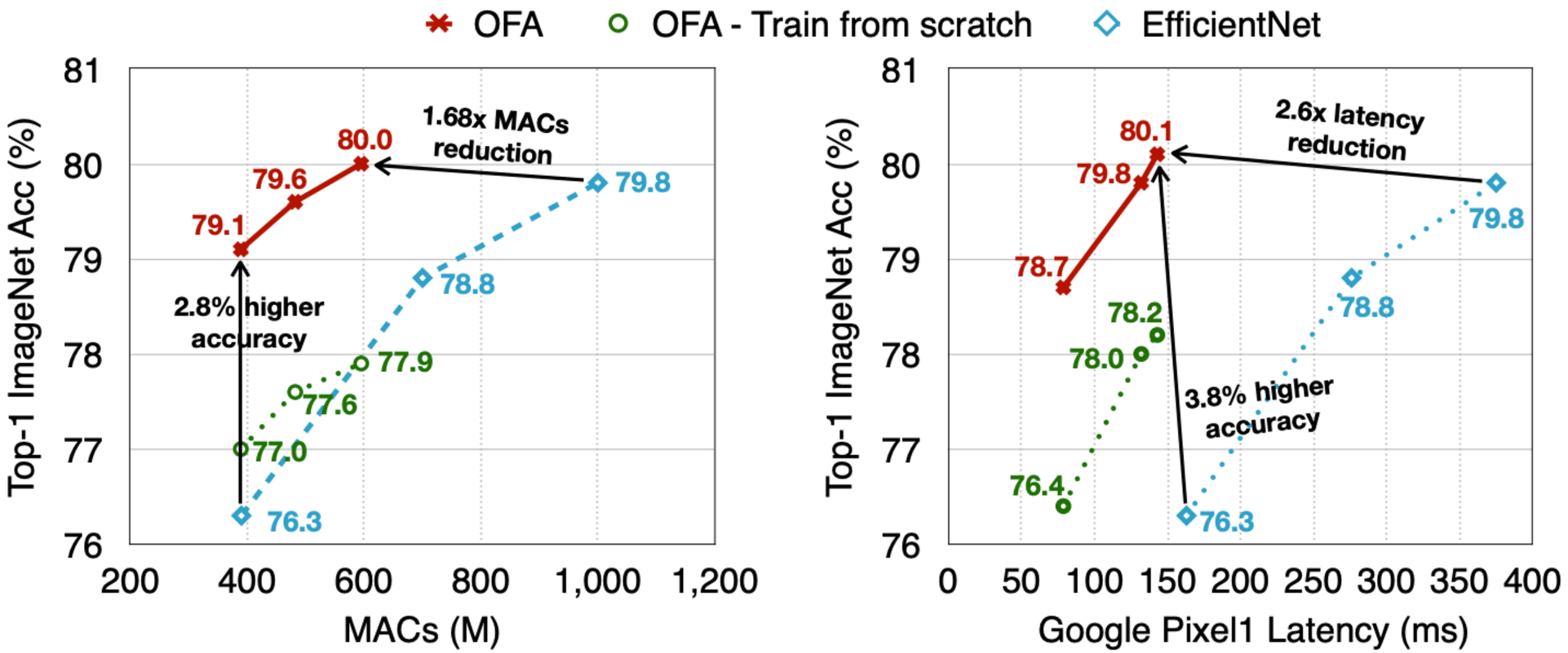
OFA on Diverse Hardware Platforms

Enables fast specialization on diverse hardware platforms



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

OFA Achieves Higher Accuracy than Training from Scratch



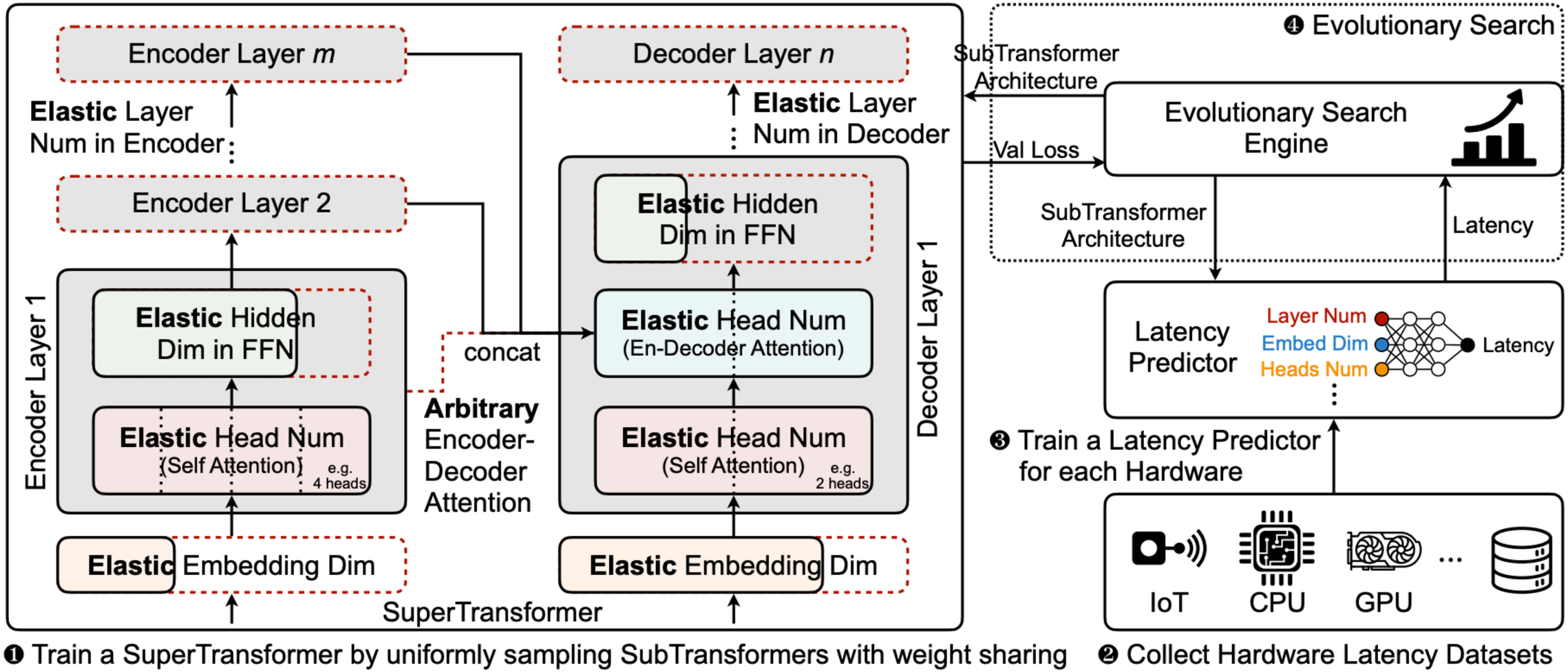
- Training from scratch cannot achieve the same level of accuracy.
- Over-parameterization may help with training smaller networks. [NetAug, ICLR'22]

Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

OFA Has Broad Applications

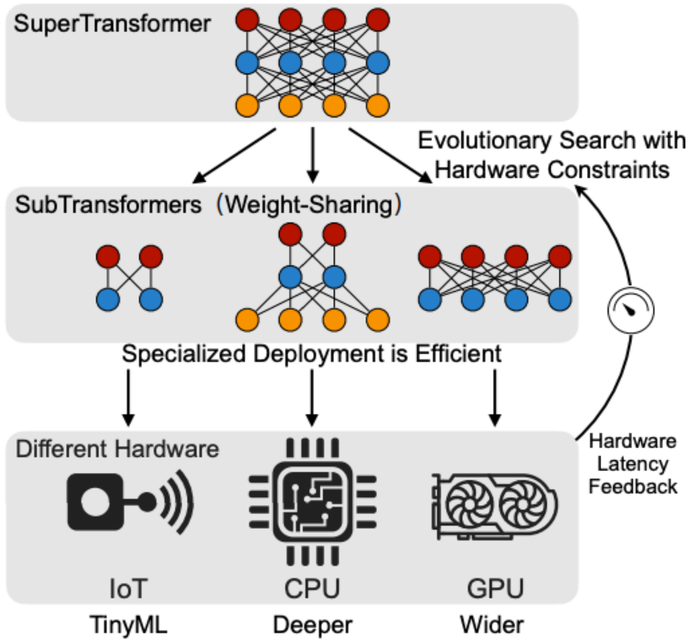
- Natural Language Processing [HAT, ACL'20]
- Video Recognition [TSM, ICCV'19]
- Point Cloud [PVCNN, NeurIPS'19; PVNAS, TPAMI'22; SPVNAS, ECCV'20]
- GAN [GAN Compression, CVPR'20; Anycost GAN, CVPR'21]
- Pose Estimation [Lite Pose, CVPR'22]
- Large Language Model [Flextron, ICML'24]
- Quantum AI [QuantumNAS, HPCA'22]

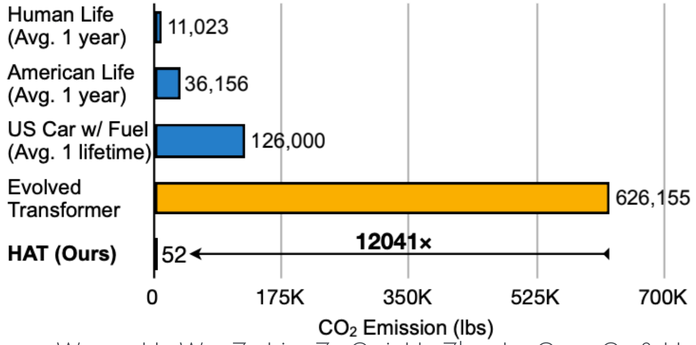
HAT: Hardware-Aware Transformers for Efficient Natural Language Processing

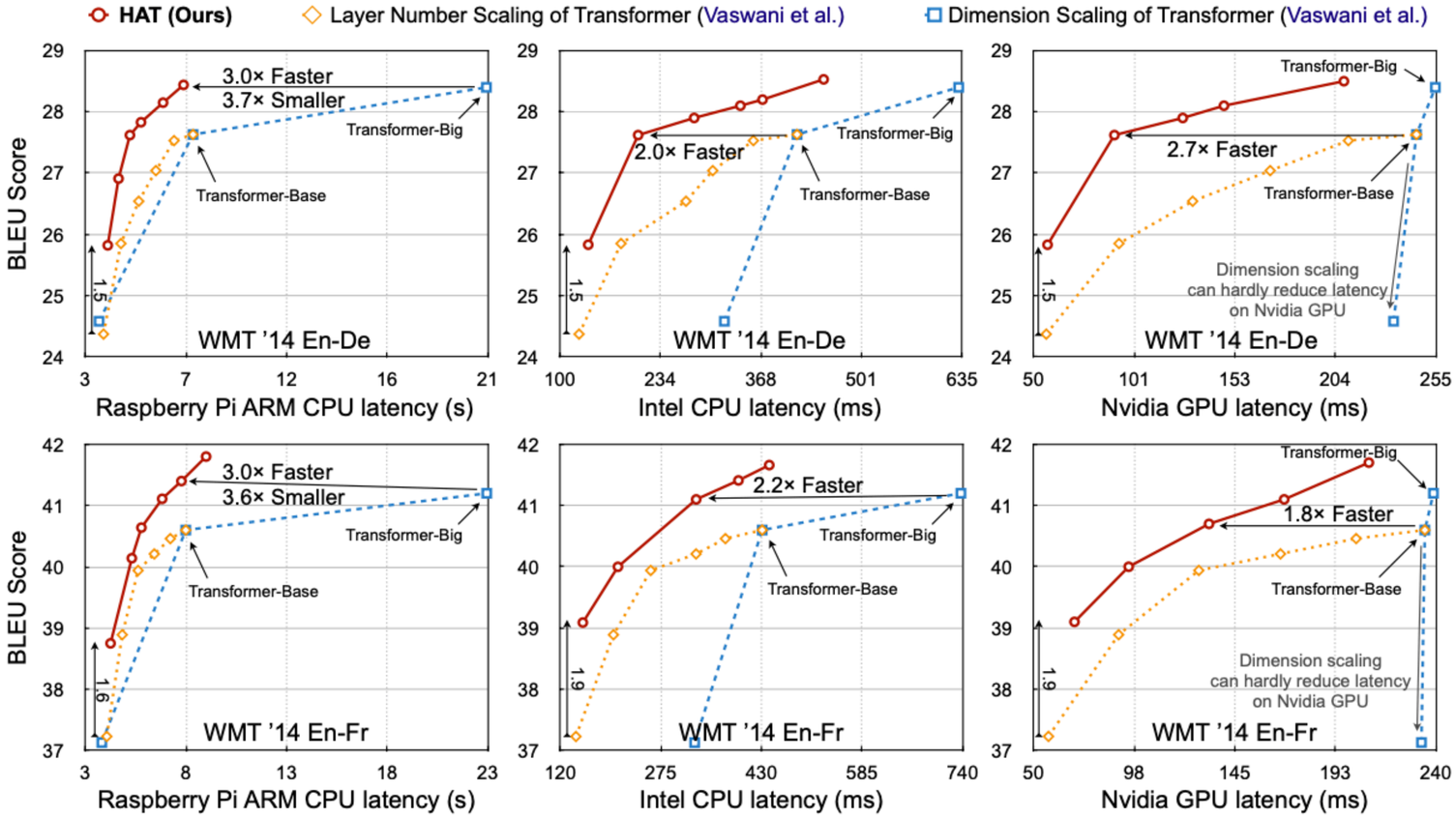


Wang, H., Wu, Z., Liu, Z., Cai, H., Zhu, L., Gan, C., & Han, S. (2020). Hat: Hardware-aware transformers for efficient natural language processing. arXiv preprint arXiv:2005.14187.

HAT: Hardware-Aware Transformers for Efficient Natural Language Processing



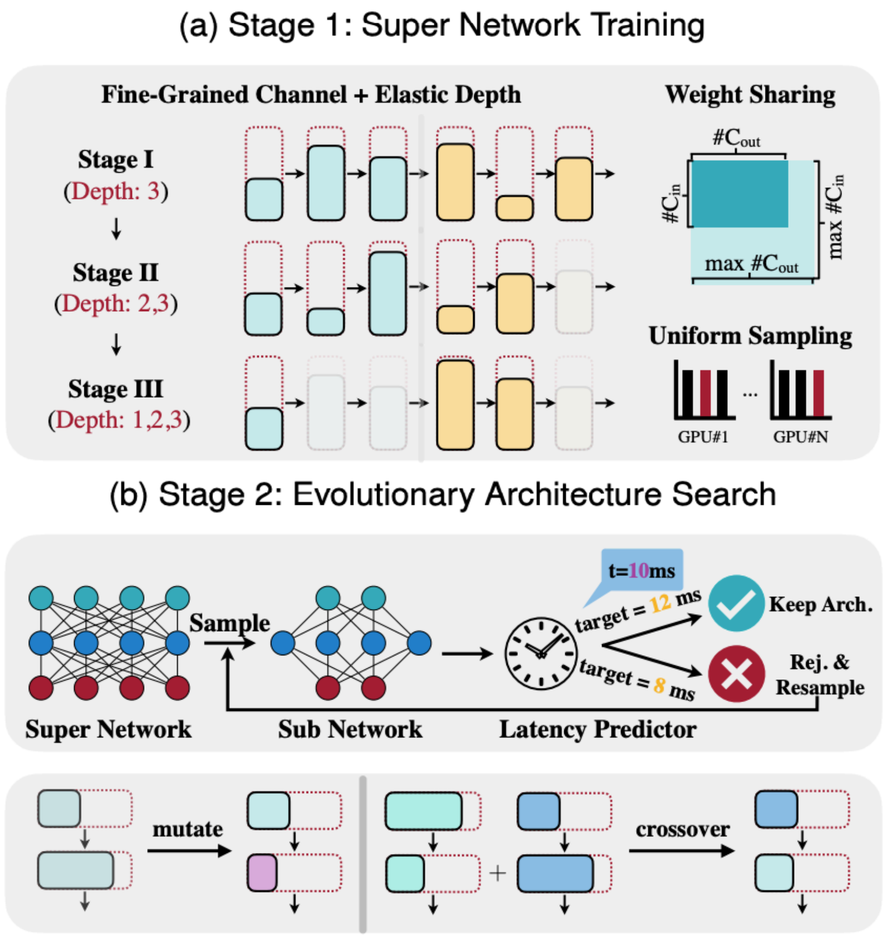


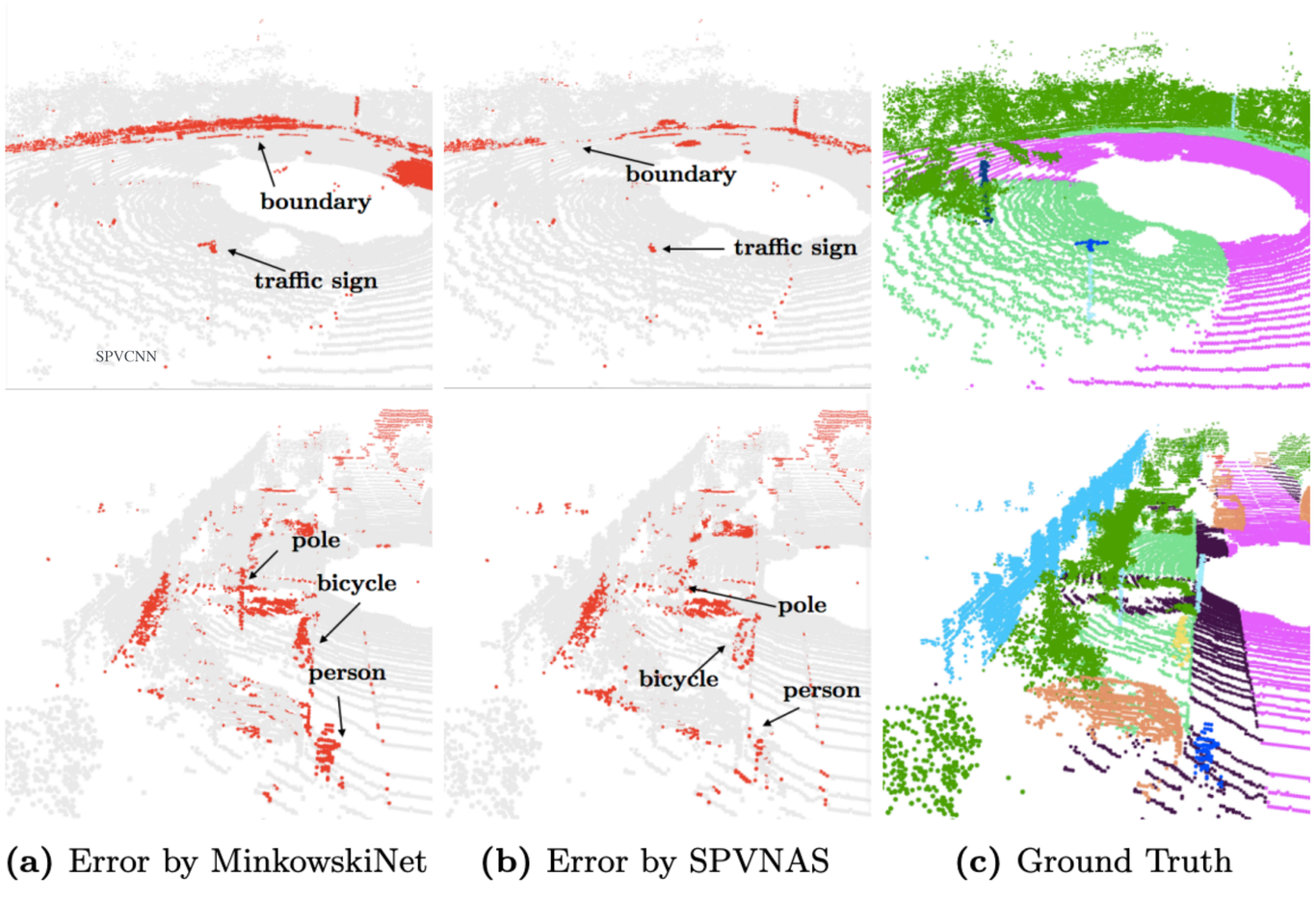


Wang, H., Wu, Z., Liu, Z., Cai, H., Zhu, L., Gan, C., & Han, S. (2020). Hat: Hardware-aware transformers for efficient natural language processing. arXiv preprint arXiv:2005.14187.

OFA for Point Cloud Understanding

SPVNAS



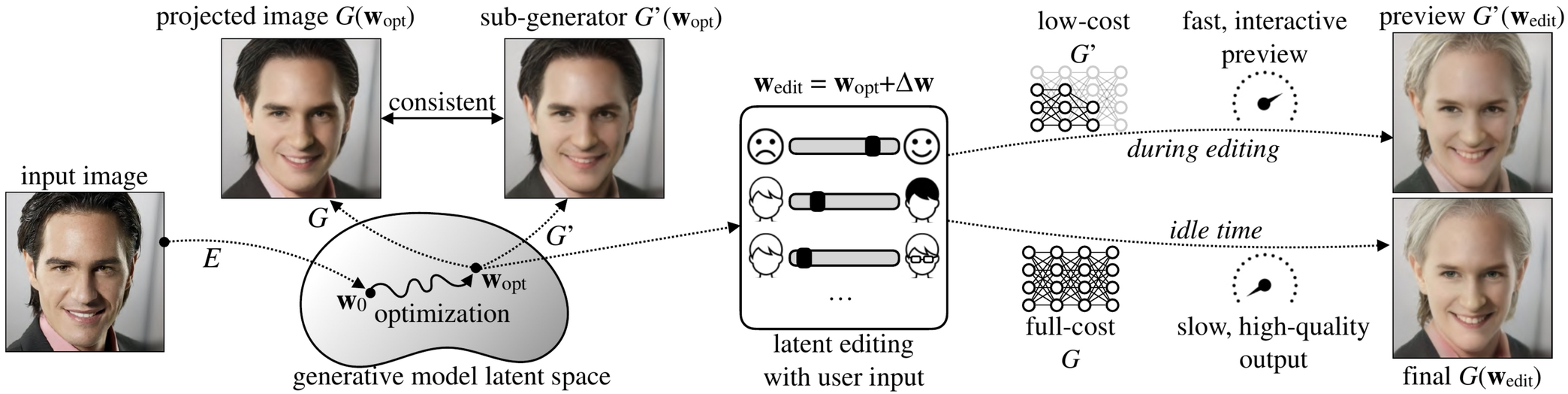


Liu, Z., Tang, H., Zhao, S., Shao, K., & Han, S. (2021). Pvnas: 3d neural architecture search with point-voxel convolution. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11), 8552-8568.
Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., & Han, S. (2020, August). Searching efficient 3d architectures with sparse point-voxel convolution. In European conference on computer vision (pp. 685-702). Cham: Springer International Publishing.

Efficient Image Generation

AnycostGAN achieves interactive image synthesis and editing on a laptop

- Generative Adversarial Network (GAN) is computationally heavy and slow
- Difficult for interactive photo editing on mobile device (iPad)
- Anycost GAN with once-for-all network: Train once, get (1) Small sub-net: low cost, fast prototyping; (2) Large sub-net: high-quality, finalization

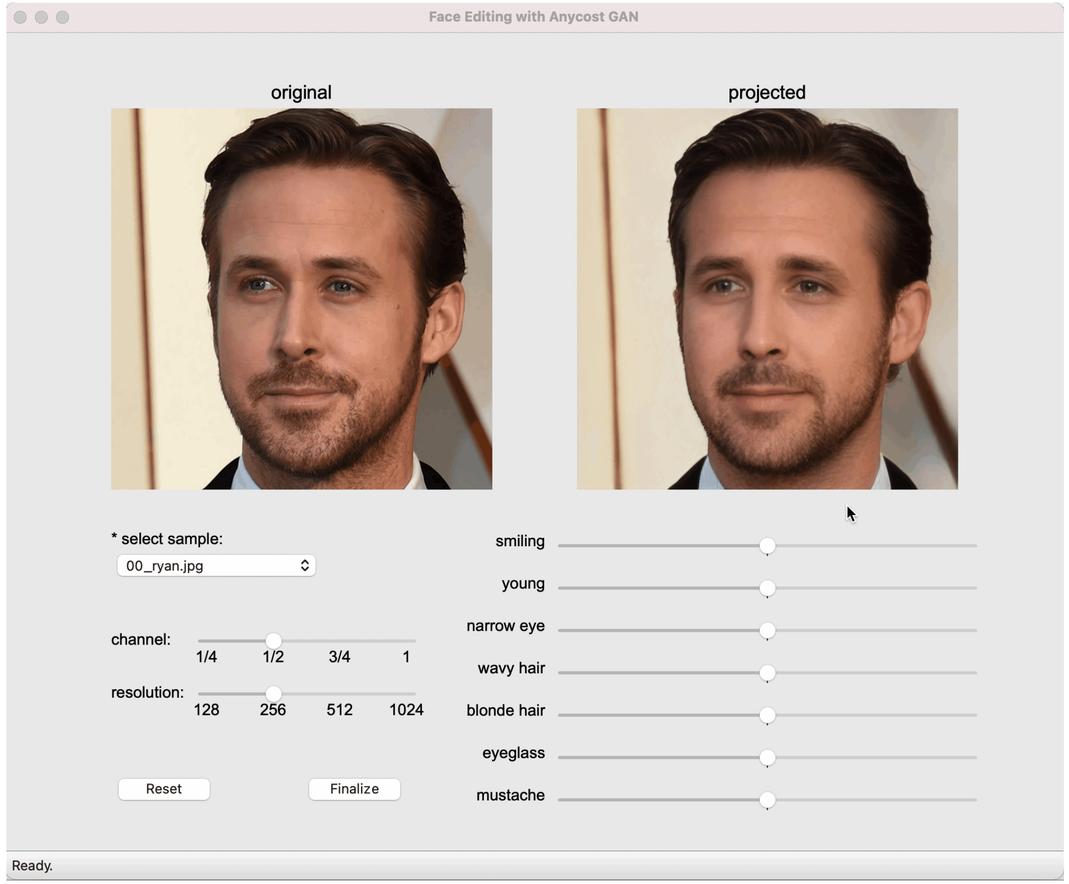


Lin, J., Zhang, R., Ganz, F., Han, S., & Zhu, J. Y. (2021). Anycost gans for interactive image synthesis and editing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 14986-14996).

OFA for GAN

Anycost GAN

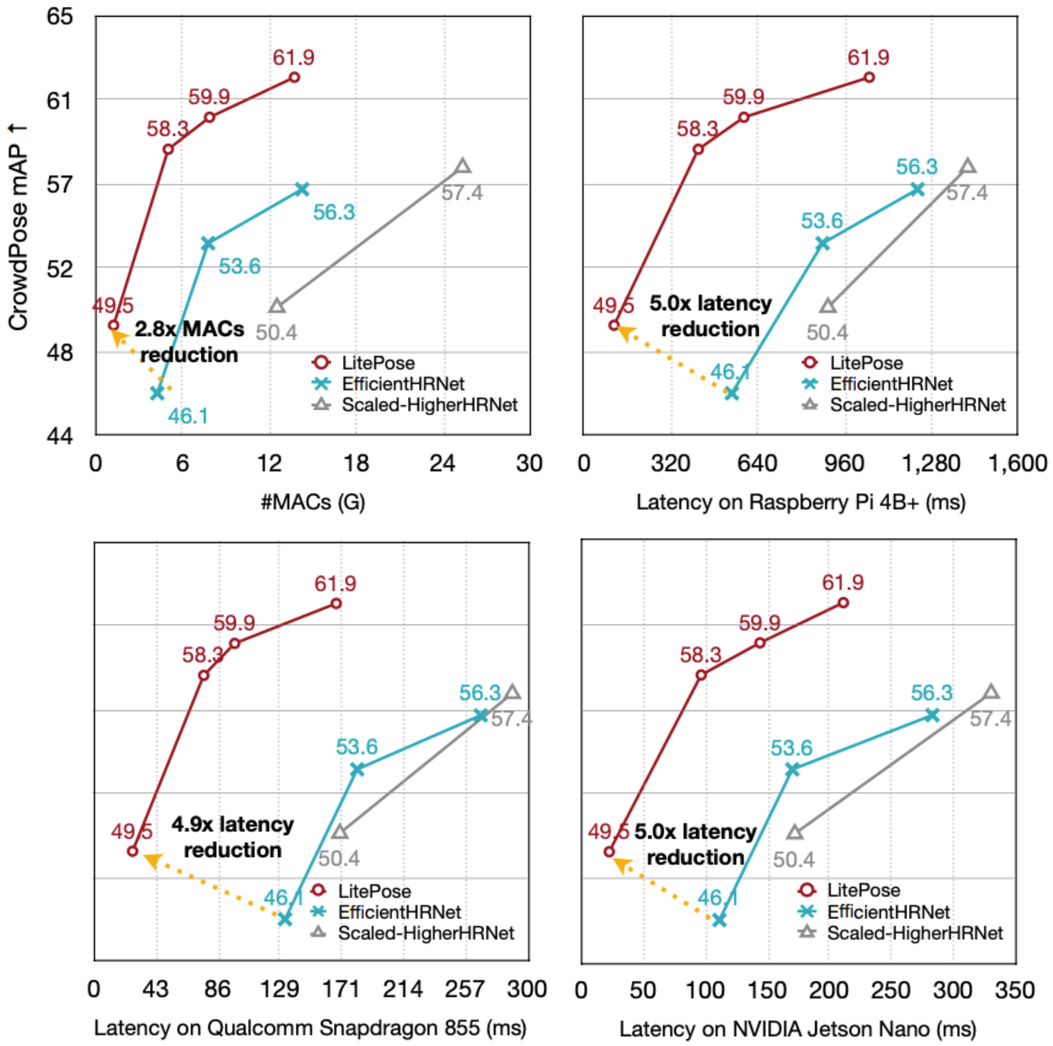
Anycost GAN generates consistent outputs under various computational budgets.

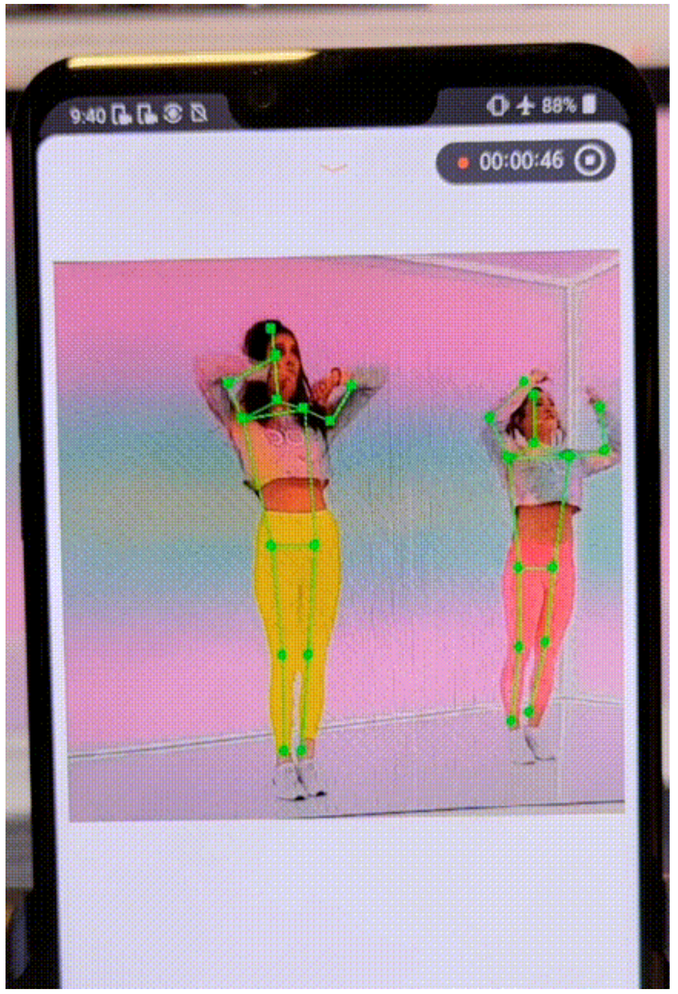


Anycost GAN in interactive image editing.

Lin, J., Zhang, R., Ganz, F., Han, S., & Zhu, J. Y. (2021). Anycost gans for interactive image synthesis and editing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 14986-14996).

OFA for Pose Estimation





On-device pose estimation by Hardware-aware NAS

Wang, Y., Li, M., Cai, H., Chen, W. M., & Han, S. (2022). Lite pose: Efficient architecture design for 2d human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13126-13136).

OFA for Gaze Estimation

Try it on your own

Gaze Estimation Pipeline

Input Image
3x160x128

Face Detection Model	
arch	YOLOX + proxylessNAS backbone
dataset	WIDERFACE
input	3x160x128 image
output	face bounding boxes
MACs	26M
Params	314K
8Gen1 latency	GPU: 3.8ms DSP: 0.28ms

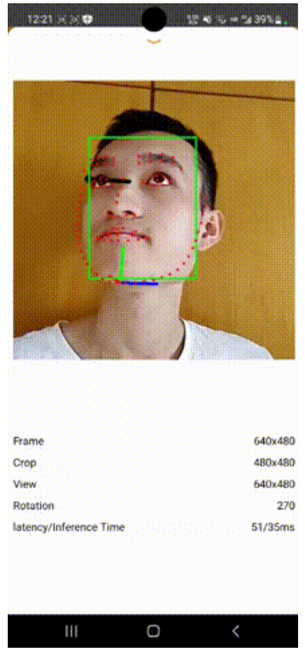
Crop face

Facial Landmark Detection Model	
arch	PFLD + proxylessNAS backbone
dataset	WFLW
input	3x112x112 cropped face image
output	98 landmark coordinates
MACs	16M
Params	454K
8Gen1 latency	GPU: 2.2ms DSP: 0.22ms

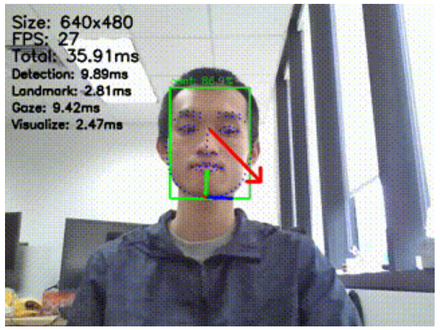
Estimate head pose and align face using landmarks

Gaze Estimation Model	
arch	AGE-Net + proxylessNAS backbone
dataset	ETH-XGaze
input	3x120x120 aligned face image 3x60x60 aligned left/right eye images
output	gaze direction (pitch & yaw angle)
MACs	38M
Params	1707K
8Gen1 latency	GPU: 10ms DSP: 0.48ms

Pipeline Outputs
- Face bounding boxes
- Facial landmarks
- 3D head pose
- 3D gaze direction



On Android

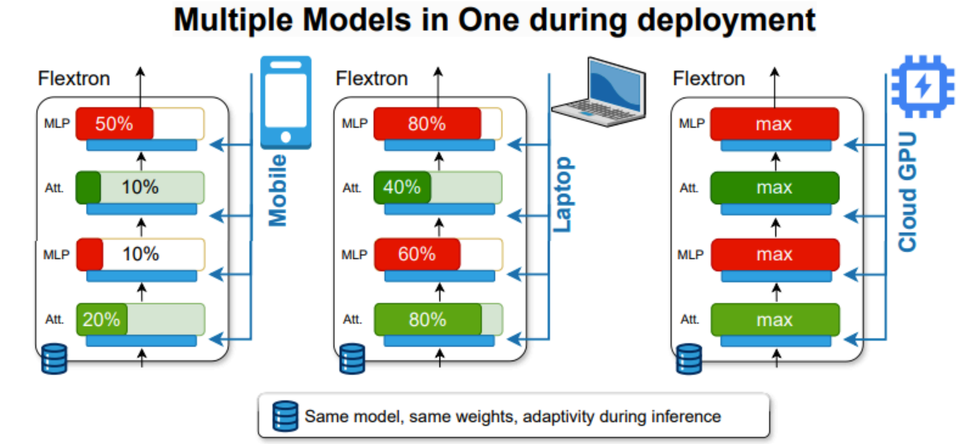


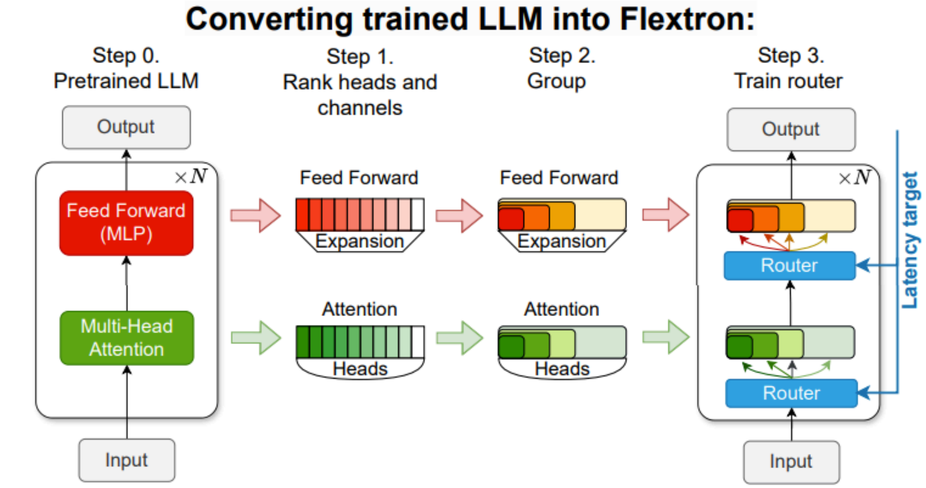
On Raspberry Pi 4

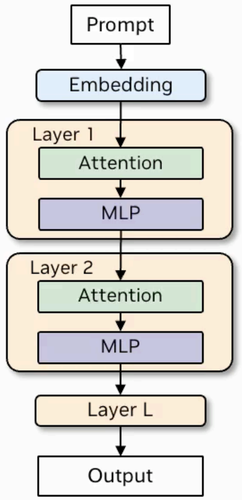
Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

OFA for Large Language Models

FlexTron: Many-in-One Flexible Large Language model







7B parameters

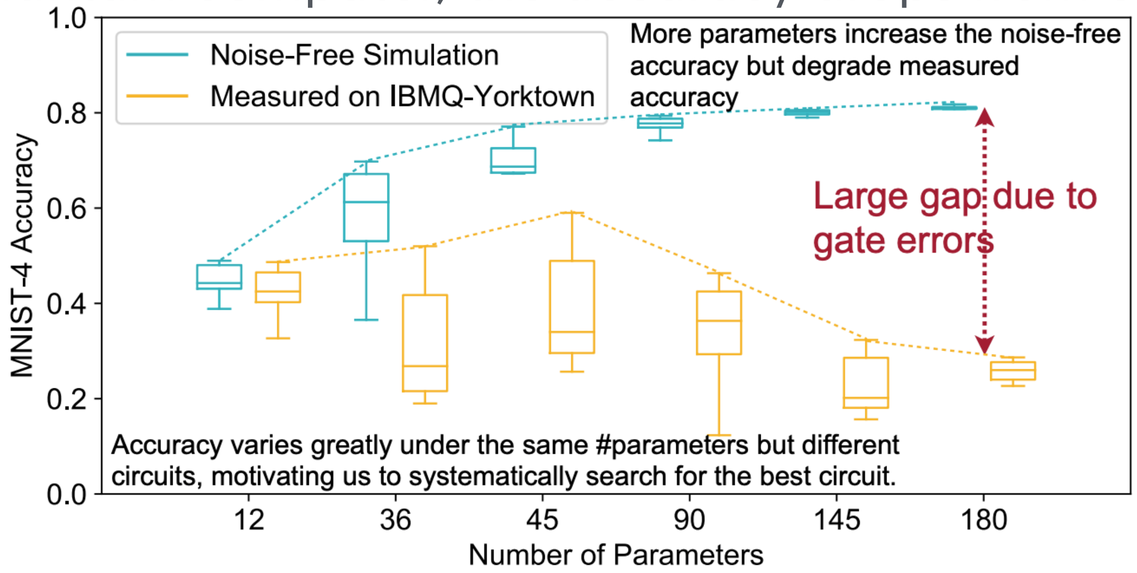
FLEXTRON

Train One Model, Get Many

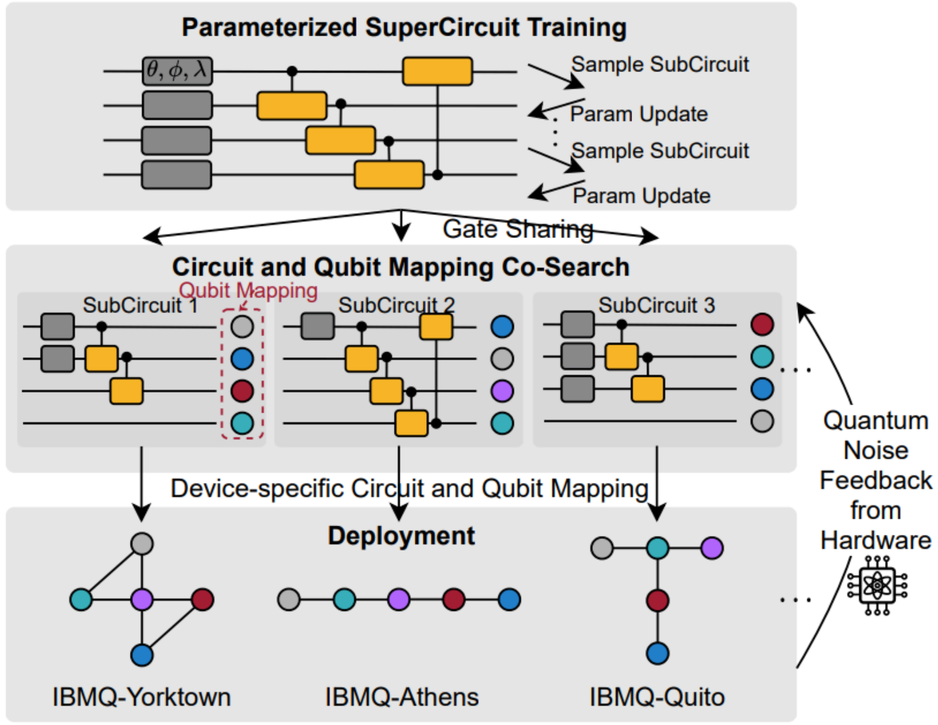
Cai, R., Muralidharan, S., Heinrich, G., Yin, H., Wang, Z., Kautz, J., & Molchanov, P. (2024). Flextron: Many-in-One Flexible Large Language Model. arXiv preprint arXiv:2406.10260.

OFA for Quantum AI

- Quantum noise is the bottleneck for quantum neural nets, noise-free simulator vs. real Quantum computer, the accuracy drops from 87% to 47%



- **QuantumNAS**: **N**oise-**A**ware **S**earch for robust circuit architecture by training a "super circuit", then search a "sub-circuit" that is robust to noise; prune small-magnitude quantum gates



Noise-adaptive circuit and qubit mapping co-search

  - Implemented in torchquantum

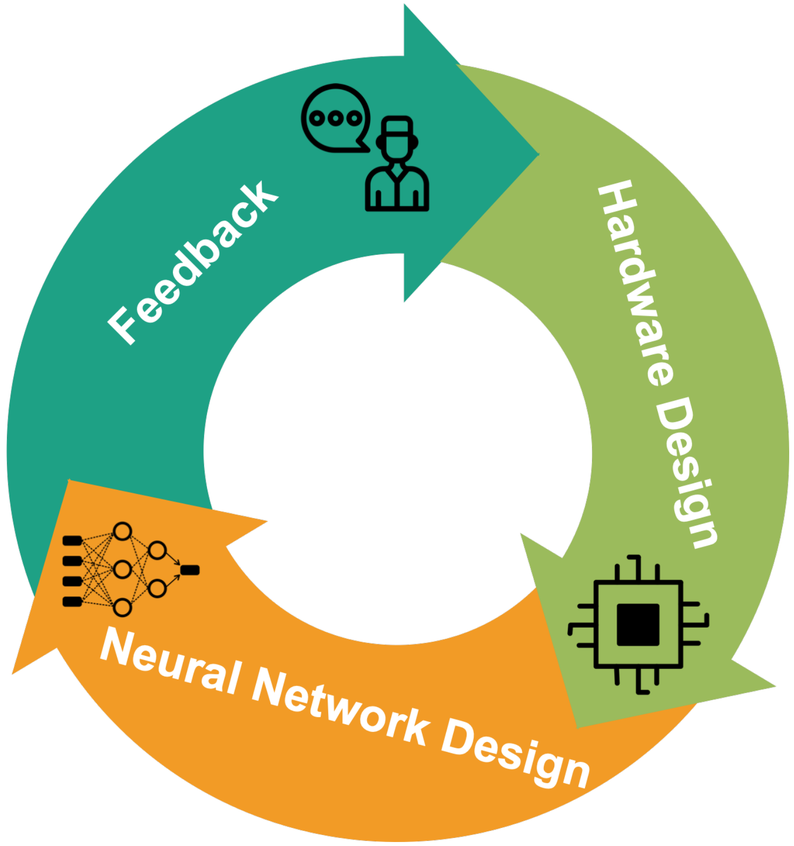
Wang, H., Ding, Y., Gu, J., Lin, Y., Pan, D. Z., Chong, F. T., & Han, S. (2022, April). Quantumnas: Noise-adaptive search for robust quantum circuits. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 692-708). IEEE.

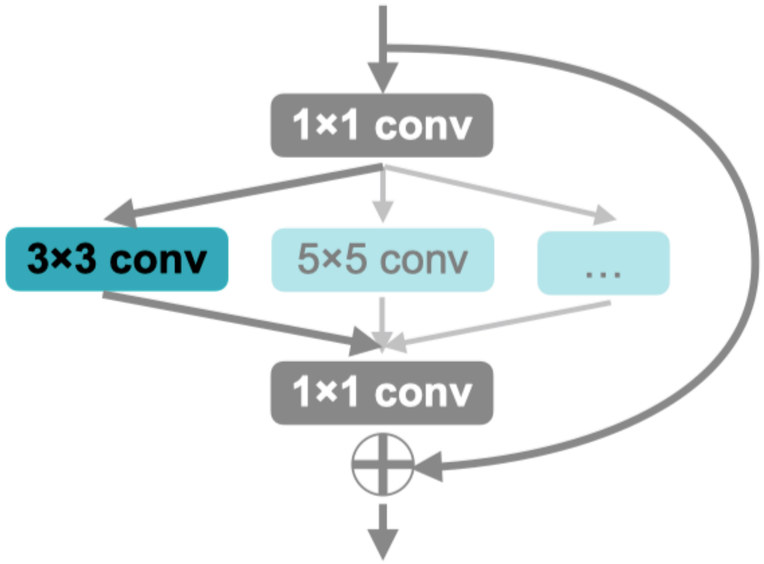
OFA Tutorial

Neural-Hardware Architecture Co-Search

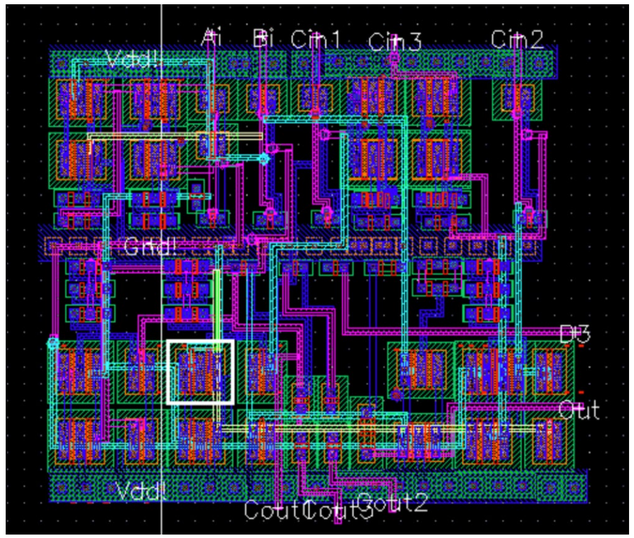
NAAS: Neural Accelerator Architecture Search

Both **neural architecture** and **accelerator architecture** design are important to enable specialization and acceleration





Neural Architecture



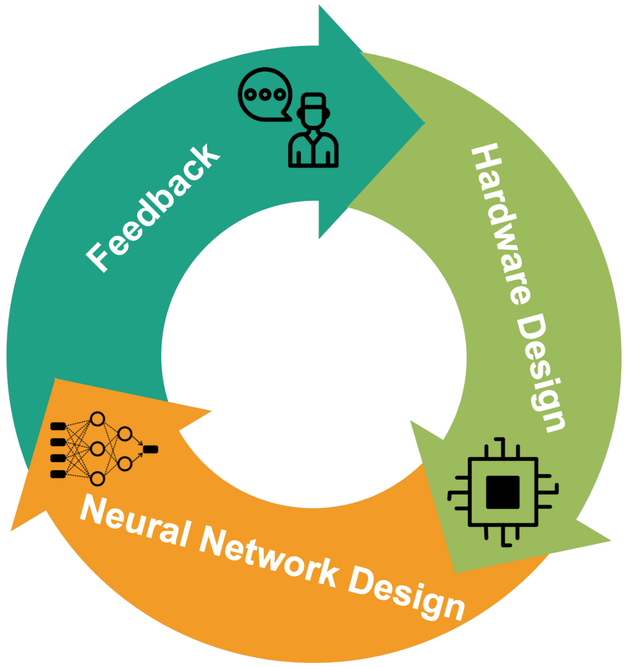
Accelerator Architecture

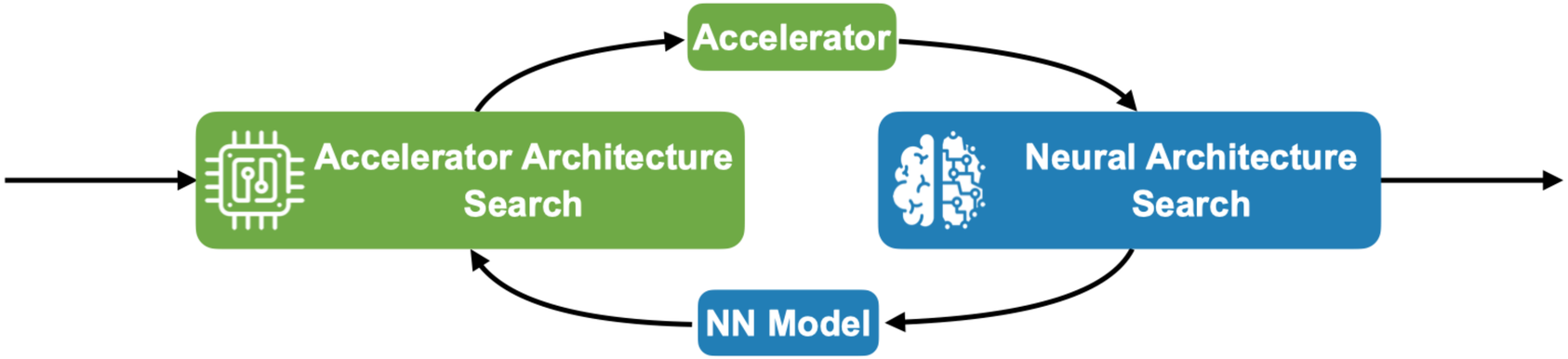
Lin, Y., Yang, M., & Han, S. (2021, December). NAAS: Neural accelerator architecture search. In 2021 58th ACM/IEEE Design Automation Conference (DAC) (pp. 1051-1056). IEEE.

NAAS: Neural Accelerator Architecture Search

Joint search

- Searching accelerator and neural architecture in one optimization loop offers highly matched solutions

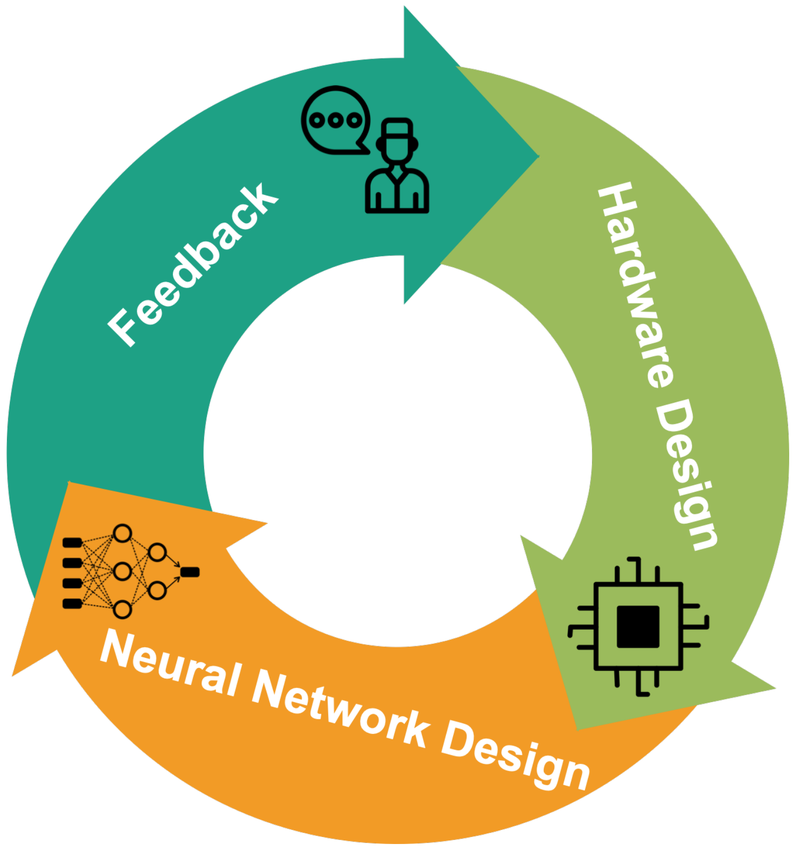




Lin, Y., Yang, M., & Han, S. (2021, December). NAAS: Neural accelerator architecture search. In 2021 58th ACM/IEEE Design Automation Conference (DAC) (pp. 1051-1056). IEEE.

NAAS: Neural Accelerator Architecture Search

Design Space



	Key Dimensions
Accelerator	Local Buffer Size, Global Buffer Size, #PEs
	Compute Array Size, PE Connectivity
Compiler	Loop Orders, Loop Tiling Size, Dataflow
Neural Network	#Layers, #Channels, Kernel Size, Bypass
	(Input/Weight) Quantization Precision

To be discussed in the TinyEngine lecture.

Lin, Y., Yang, M., & Han, S. (2021, December). NAAS: Neural accelerator architecture search. In 2021 58th ACM/IEEE Design Automation Conference (DAC) (pp. 1051-1056). IEEE.

Lab 4 is out

References

- Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.
- Zoph, B. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.
- Chen, T., Goodfellow, I., & Shlens, J. (2015). Net2net: Accelerating learning via knowledge transfer. arXiv preprint arXiv:1511.05641.
- Cai, H., Chen, T., Zhang, W., Yu, Y., & Wang, J. (2018, April). Efficient architecture search by network transformation. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).
- Ha, D., Dai, A., & Le, Q. V. (2016). Hypernetworks. arXiv preprint arXiv:1609.09106.
- Brock, A., Lim, T., Ritchie, J. M., & Weston, N. (2017). Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344.
- Zhang, C., Ren, M., & Urtasun, R. (2018). Graph hypernetworks for neural architecture search. arXiv preprint arXiv:1810.05749.
- Lin, M., Wang, P., Sun, Z., Chen, H., Sun, X., Qian, Q., ... & Jin, R. (2021). Zen-nas: A zero-shot nas for high-performance image recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 347-356).
- Zhang, Z., & Jia, Z. (2021). Gradsign: Model performance inference with theoretical insights. arXiv preprint arXiv:2110.08616.
- Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.
- Wang, H., Wu, Z., Liu, Z., Cai, H., Zhu, L., Gan, C., & Han, S. (2020). Hat: Hardware-aware transformers for efficient natural language processing. arXiv preprint arXiv:2005.14187.

References

- Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 2820-2828).
- Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.
- Liu, Z., Tang, H., Zhao, S., Shao, K., & Han, S. (2021). Pvnas: 3d neural architecture search with point-voxel convolution. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11), 8552-8568.
- Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., & Han, S. (2020, August). Searching efficient 3d architectures with sparse point-voxel convolution. In European conference on computer vision (pp. 685-702). Cham: Springer International Publishing.
- Lin, J., Zhang, R., Ganz, F., Han, S., & Zhu, J. Y. (2021). Anycost gans for interactive image synthesis and editing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 14986-14996).
- Wang, Y., Li, M., Cai, H., Chen, W. M., & Han, S. (2022). Lite pose: Efficient architecture design for 2d human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13126-13136).
- Wang, H., Ding, Y., Gu, J., Lin, Y., Pan, D. Z., Chong, F. T., & Han, S. (2022, April). Quantumnas: Noise-adaptive search for robust quantum circuits. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 692-708). IEEE.
- Cai, R., Muralidharan, S., Heinrich, G., Yin, H., Wang, Z., Kautz, J., & Molchanov, P. (2024). Flextron: Many-in-One Flexible Large Language Model. arXiv preprint arXiv:2406.10260.
- Lin, Y., Yang, M., & Han, S. (2021, December). NAAS: Neural accelerator architecture search. In 2021 58th ACM/IEEE Design Automation Conference (DAC) (pp. 1051-1056). IEEE.
- Neural Architecture Search II [MIT 6.5940]