HACScale: Hardware-Aware Compound Scaling for Resource-Efficient DNNs

Hao Kong[1,2], Di Liu[1], Xiangzhong Luo[2], Weichen Liu[2] and Ravi Subramaniam[3]

[1] HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore
[2] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[3] Innovations and Experiences – Business Personal Systems, HP Inc., Palo Alto, California, USA

Content

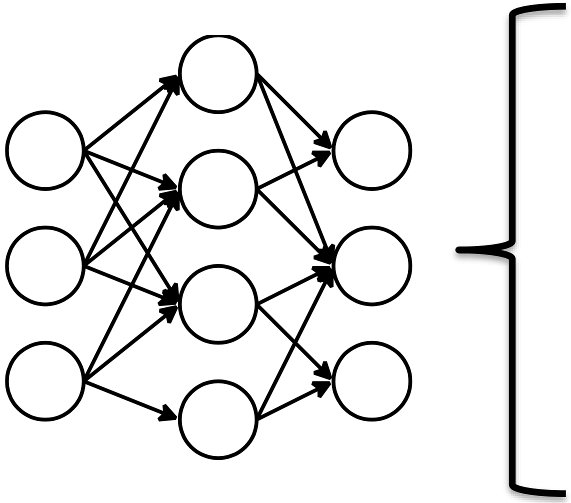
1. Background
2. Hardware-Aware Compound Scaling
3. Importance-Aware Width Scaling
4. Experiments
5. Conclusion

Content

1. Background
2. Hardware-Aware Compound Scaling
3. Importance-Aware Width Scaling
4. Experiments
5. Conclusion

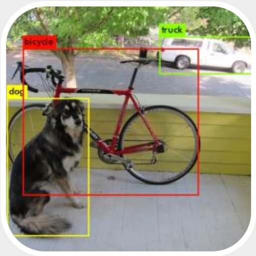
Background

- Deep neural networks (DNNs) have unlocked unprecedented breakthroughs among a wide range of real-world applications.

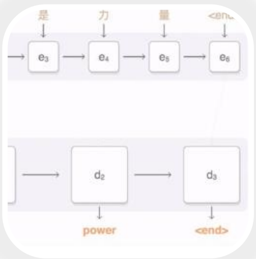


Semantic Segmentation

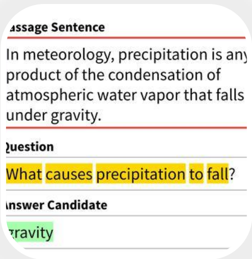
Image Classification



Object Detection



Machine Translation



Question Answering

Speech Synthesis

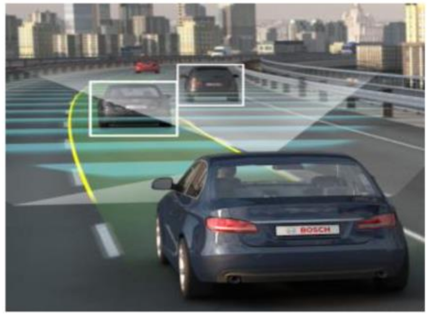
Background: Need for Accurate Models

- User experience
- Property safety
- Safety-critical applications

Virtual reality

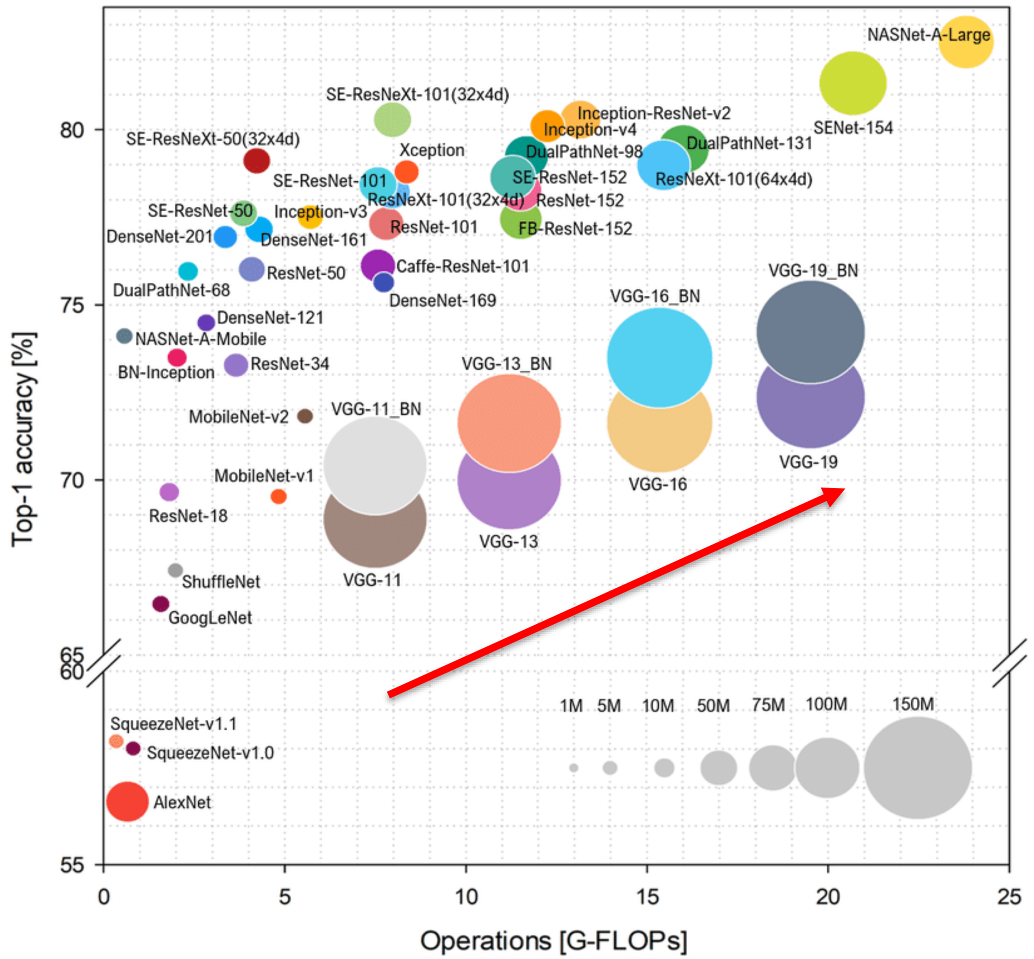
Smart city

Biometric identification



Autonomous driving

Background: Need for Accurate Models



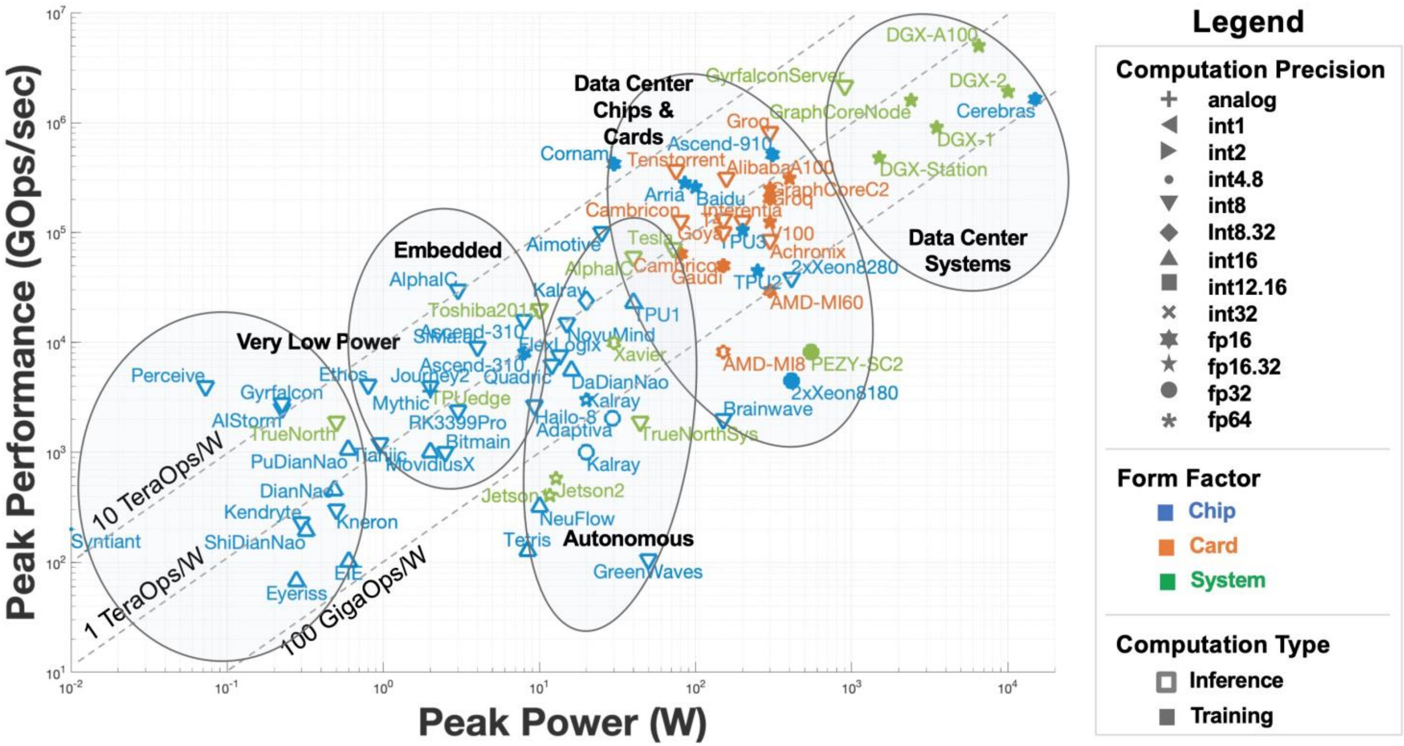
How to design an accurate model?

- Architecture innovation
  - Highly demanding for expertise!
  - High search cost!
- Enlarging the size
  - Easy to implement!
  - Flexibility!

[1] Bianco, Simone et al. “Benchmark Analysis of Representative Deep Neural Network Architectures.” *IEEE Access* 6 (2018): 64270-64277.

Background: Advances in AI Hardware

- New chip architecture
  - ASIC, FPGA, GPU, TPU
  - Efficient for DNNs
- Larger Memory
  - GV100 – 32GB
  - A100 – 80GB
- Higher memory bandwidth
  - A100 – 1.94 TB/s



[1] Reuther, Albert, et al. "Survey of machine learning accelerators." 2020 IEEE High Performance Extreme Computing Conference (HPEC). IEEE, 2020.

Background: Advances in AI Hardware

ResNet-18 model
- Parameters: 11.68 M
- Memory access: 46.72 MB/image
- Computation: 1.74 GOPs/image

13.66ms

AGX Xavier
32 TOPS
30 W
137 GB/s

- By fully utilizing the hardware, we can achieve higher accuracy without sacrificing the latency.

TABLE I: Accuracy and latency of Wide-ResNet model with different width scaling on NVIDIA Jetson Xavier.

Model	Scaling factor	Accuracy(%)	Latency(ms)
Wide-ResNet	1	88.09	5.5
	2	92.71	5.58
	4	94.19	5.58

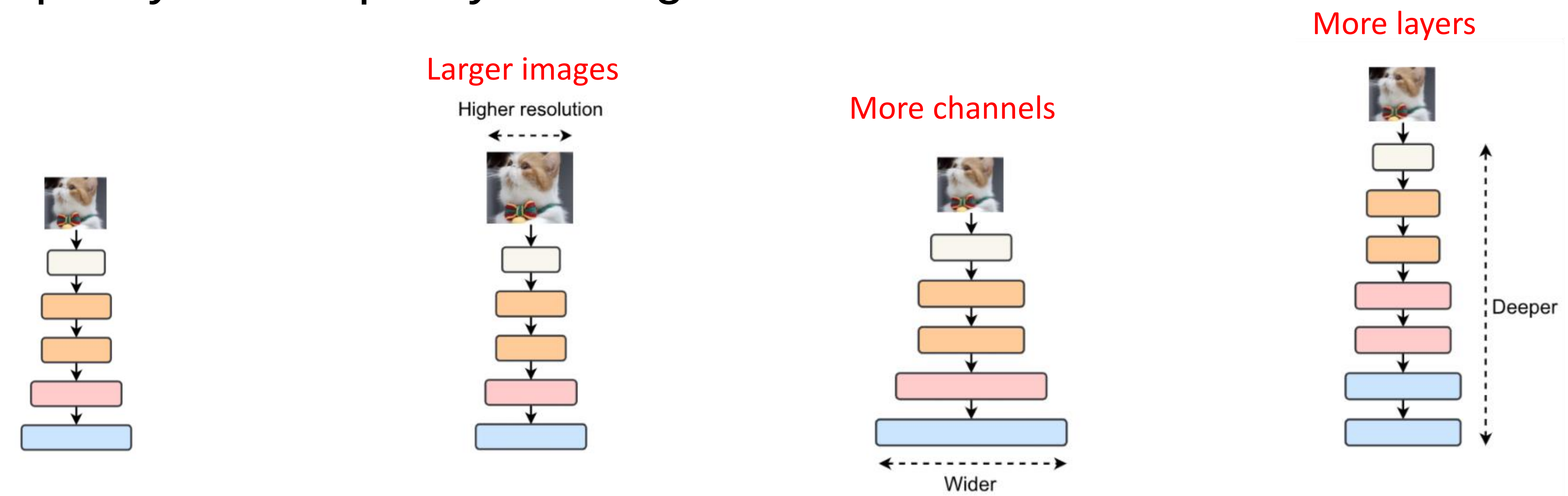
[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Content

1. Background
2. Hardware-Aware Compound Scaling
3. Importance-Aware Width Scaling
4. Experiments
5. Conclusion

Scaling Dimensions

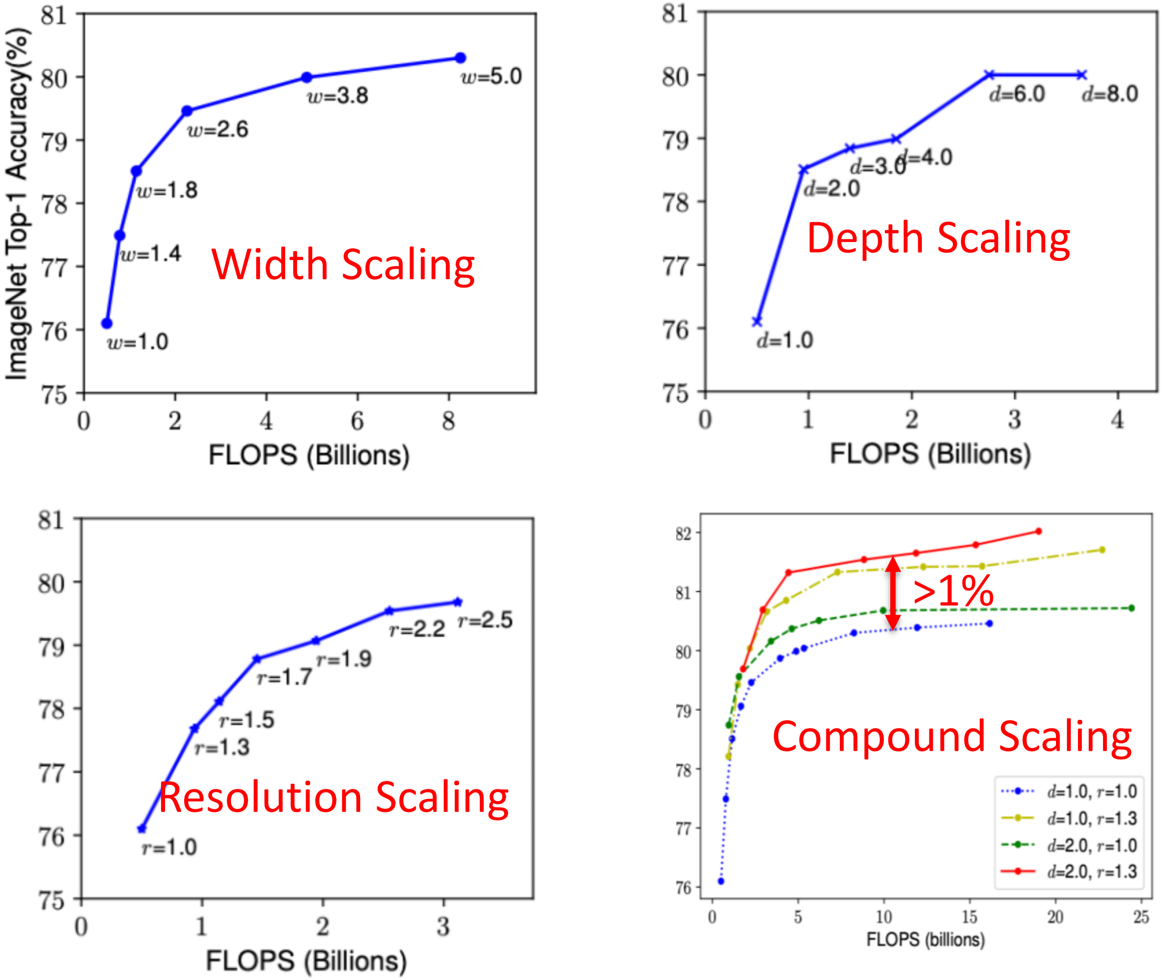
- Model scaling improves the model accuracy by increasing the model capacity or the quality of images.



(a) Baseline (b) Resolution scaling (c) Width scaling (d) Depth scaling

Scaling Dimensions

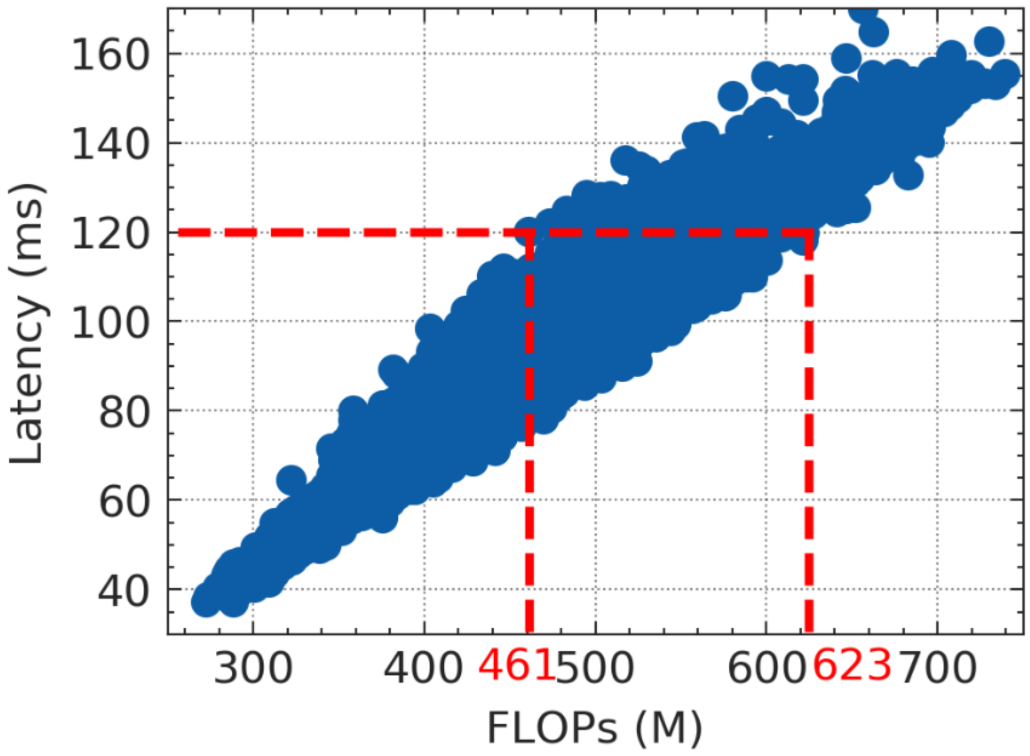
- Single-dimension scaling
  - Simple
  - Limited improvement
- Compound scaling
  - Higher accuracy
  - More complicated



[1] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International Conference on Machine Learning. PMLR, 2019.

Objectives and Constraints

- Objectives of compound scaling
  - Higher accuracy
- Constraints of compound scaling
  - ~~FLOPs~~
  - Latency
  - Memory

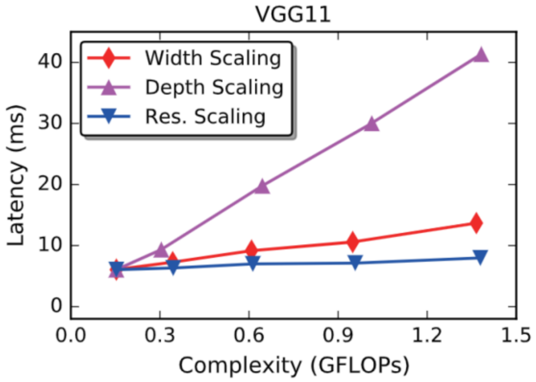


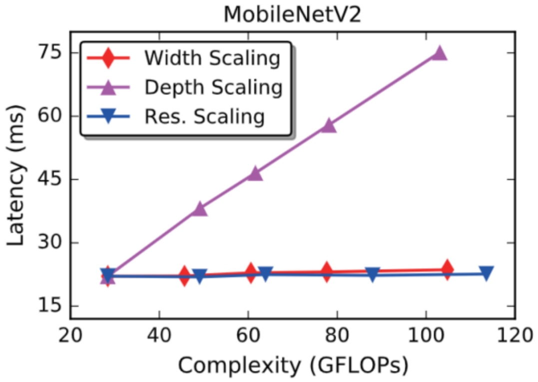
Problem Formulation

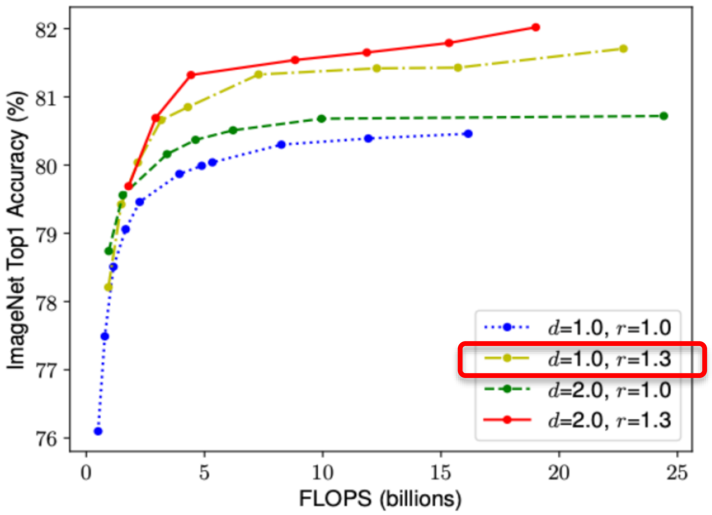
- The main goal of our scaling strategy is to maximize the accuracy under a given latency constraint $\mathcal{L}$ and memory constraint $\mathcal{M}$.

$$
\begin{aligned}
\max_{d,r,\Theta} \quad & Accuracy(\mathcal{N}(d,r,\Theta)) \\
s.t. \quad & \mathcal{N} = \bigodot_{i=1\ldots n\cdot d} \mathcal{F}_i(X_{<r\cdot H_i, r\cdot W_i, w_i\cdot C_i>}) \\
& Latency(\mathcal{N}(d,r,\Theta)) \leq \mathcal{L} \\
& Memory(\mathcal{N}(d,r,\Theta)) \leq \mathcal{M}
\end{aligned}
$$

Efficiency V.S. Accuracy







Compound Scaling

- Scaling all dimensions may not be the most efficient.
  - Higher latency
  - Larger design space
- W+R scaling
  - Good accuracy
  - Smaller design space
  - More efficient to hardware

[1] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International Conference on Machine Learning. PMLR, 2019.

Hardware-Aware Compound Scaling

- The resource allocation among different dimensions is formulated as a hyperparameter optimization problem.

$$d = 1, r = \sqrt{S}^{\alpha}, w = \sqrt{S}^{1-\alpha}$$

- (d, r, w): The scaling factors
- S:The computation budget
- α: The resource allocation hyperparameter

Obtained by random search

Content

1. Background
2. Hardware-Aware Compound Scaling
3. Importance-Aware Width Scaling
4. Experiments
5. Conclusion

Importance-Aware Width Scaling

- Different layers in DNNs have different impact on the accuracy and inference latency.

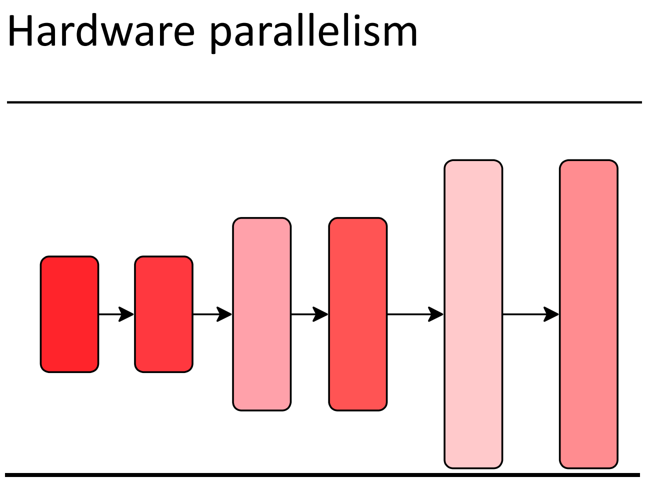


Fig. 1: Baseline model

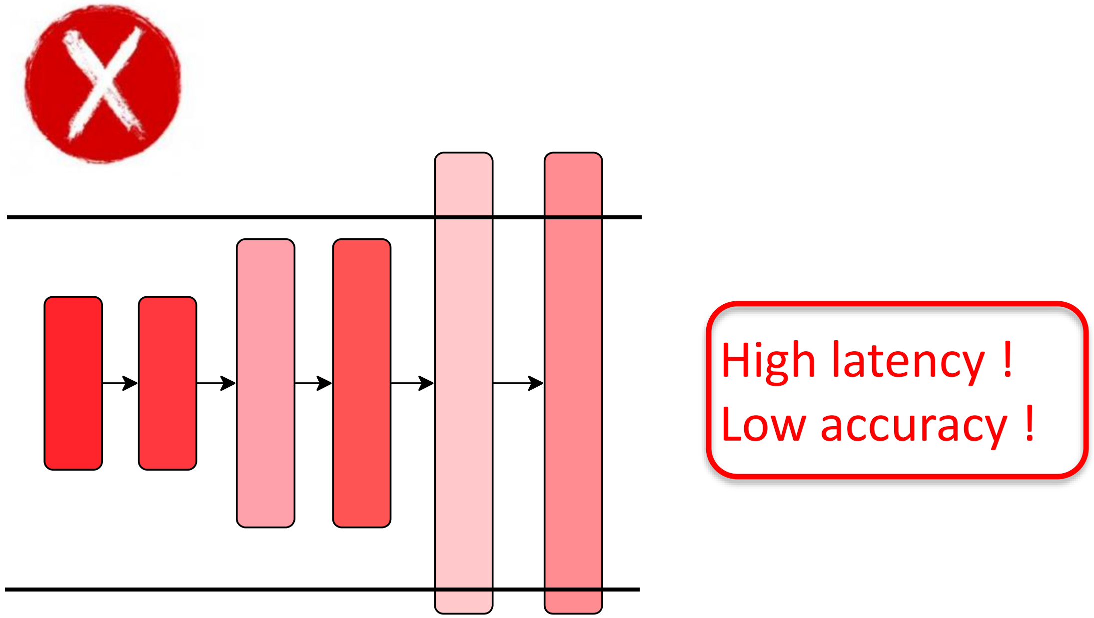


Fig. 2: Uniform scaling

[1] Molchanov, Pavlo, et al. "Importance estimation for neural network pruning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

Importance Estimation

- We model the importance with the gradient from backpropagation.
  - PROS:
    - More accurate than L1-Norm or L2-Norm
    - Can estimate the global importance
  - CONS:
    - Time-consuming training process

$$\mathcal{I}_l = (\mathbb{E}(K) - \mathbb{E}(K|k_l = 0))^2$$
$$\approx \sum_{m \in k_l} (\frac{\partial \mathcal{L}}{\partial m} \cdot m)^2 = \sum_{m \in k_l} (g_m \cdot m)^2$$

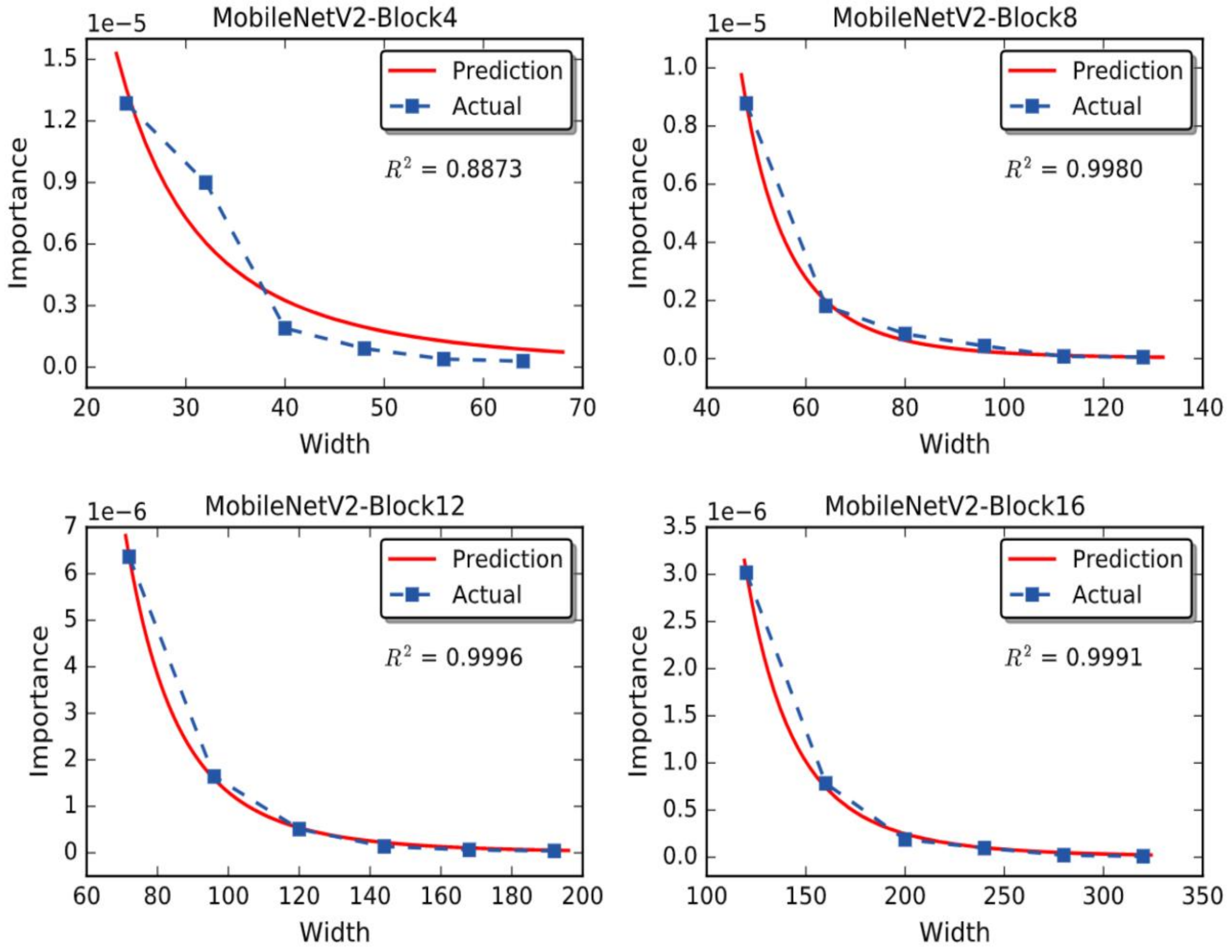
[1] Molchanov, Pavlo, et al. "Importance estimation for neural network pruning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

Importance Predictor

The formulation of the predictor:

$$I_l = a_l C_l^{b_l}$$

1. Sampling networks with different width configuration
2. Training the networks to get the gradients
3. Fitting the predictor with the gradients



Importance-Aware Width Scaling

- The advantages of importance-aware width scaling:
  - Higher accuracy
  - Less parameters
  - Less time overhead

Importance predictor

Algorithm 1: Importance-aware width scaling

Require: The importance prediction model of each layer $\{\mathcal{I}_l = a_l \cdot C_l^{b_l}\}_{l=1}^{n}$, the resolution scaling coefficient r, the baseline width configuration $\{C_l\}_{l=1}^{n}$, the latency constraint $\mathcal{L}$, and the memory constraint $\mathcal{M}$.

Ensure : The width scaling coefficient of each layer $\Theta = \{w_l\}_{l=1}^{n}$

Initialize the network $\mathcal{N}$ with r and $\{C_l\}_{l=1}^{n}$, the scaling stride $s = 10$, and the width scaling coefficients $\{w_l = 1\}_{l=1}^{n}$;

for *Each layer l in the network $\mathcal{N}$* **do**

  Compute importance $\mathcal{I}_l = a_l \cdot C_l^{b_l}$;

end

while $Latency(\mathcal{N}) \leq \mathcal{L}$ *and* $Memory(\mathcal{N}) \leq \mathcal{M}$ **do**

  Find the most important layer $i = \underset{l}{\operatorname{argmax}}\ \mathcal{I}_l$;

  Update the scaling coefficient $w_i = \frac{C_i + s}{C_i} \cdot w_i$;

  Update the layer width $C_i = C_i + s$;

  Update the importance $\mathcal{I}_i = a_i \cdot C_i^{b_i}$;

end

return $\Theta = \{w_l\}_{l=1}^{n}$

Add 10 channels to the layer

Importance-Aware Width Scaling

- Different layers in DNNs have different impact on the accuracy and inference latency.

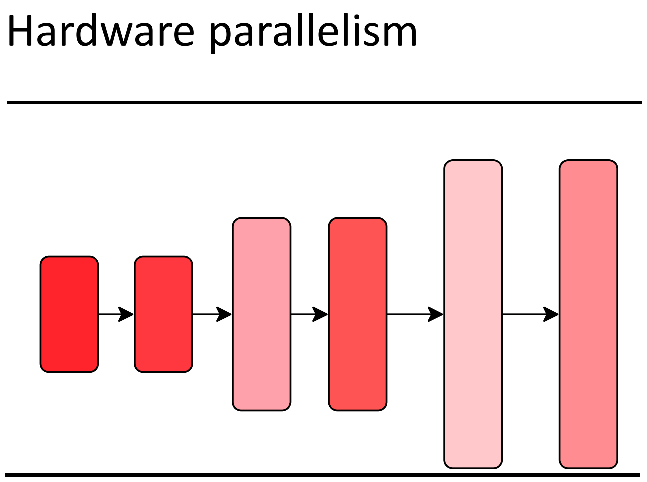


Fig. 1: Baseline model

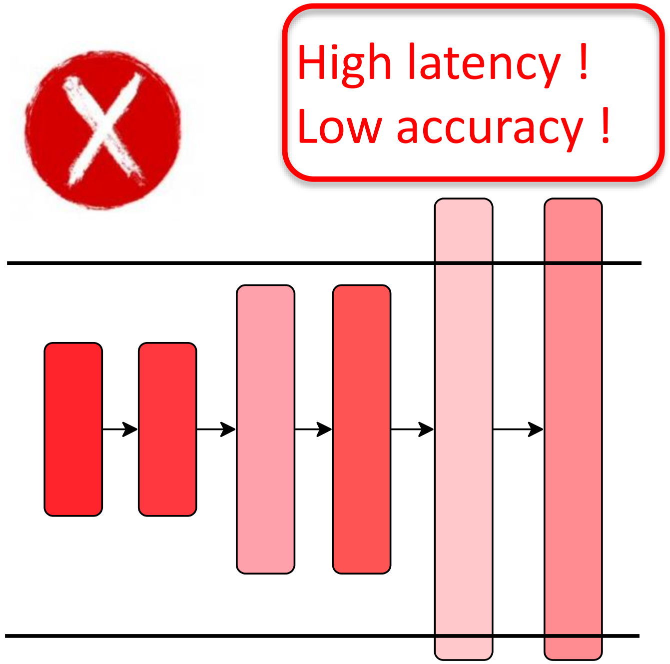


Fig. 2: Uniform scaling

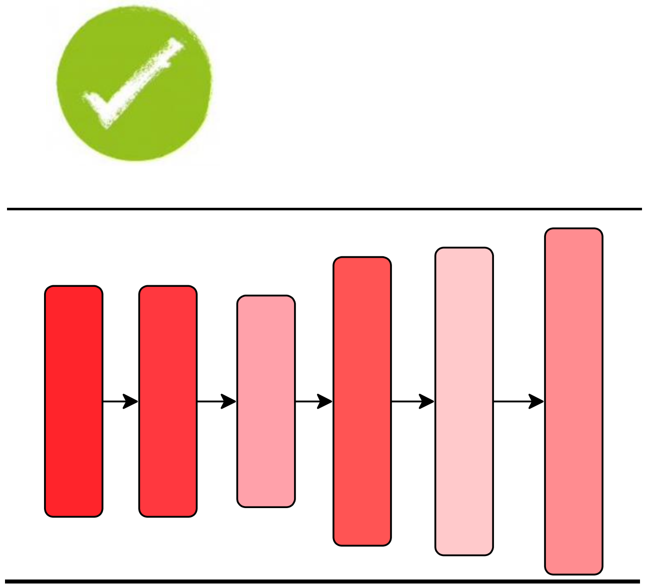


Fig. 3: Importance-aware scaling

[1] Molchanov, Pavlo, et al. "Importance estimation for neural network pruning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

Content

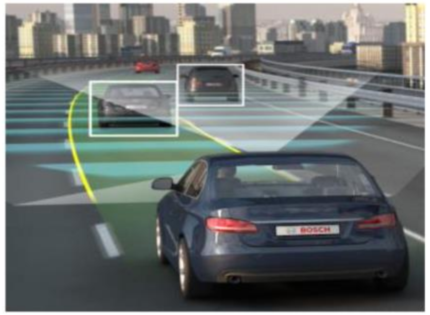
1. Background
2. Hardware-Aware Compound Scaling
3. Importance-Aware Width Scaling
4. Experiments
5. Summary

Experiments: Settings

- We test different scaling approaches under two application scenarios:

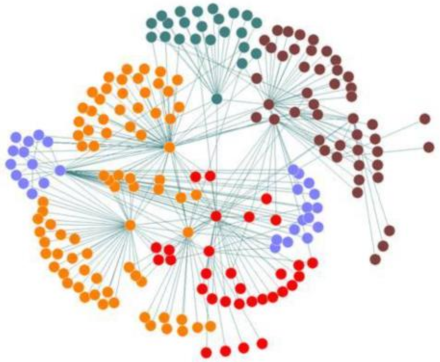
Tight latency constraint

Loose latency constraint

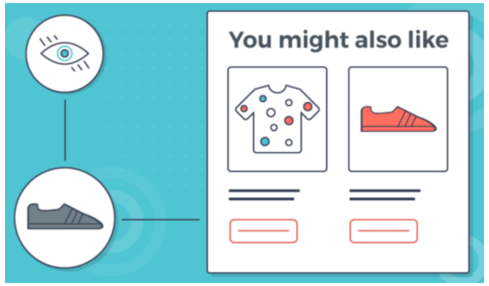


Autonomous driving

Drones



Network analysis



Recommendation

Experiments: ImageNet-1k

- HACScale achieves higher accuracy under the same latency constraint.

Constraint	Method	Params (M)	Latency (ms)†	Power (W)	↑ Utilization (GFLOPS)	↑ Power Efficiency (GFLOPs/J)	↑ Top1 (%)
ImageNet ResNet18							
	Baseline [1]	11.68	13.66	14.87	127.03 (1x)	8.54 (1x)	70.56 (+0.0)
Tight	Zagoruyko et al. [6]	16.33	13.87	22.34	170.58 (1.34x)	7.64 (0.89x)	72.24 (+1.68)
	Szegedy et al. [8]	11.68	13.48	25.6	241.34 (1.9x)	9.43 (1.1x)	71.12 (+0.56)
	NeuralScale [9]	12.74	16.73	20.68	178.62 (1.41x)	8.64 (1.01x)	72.37 (+1.81)
	HACScale (our)	12.96	13.74	24.73	**244.52 (1.92x)**	**9.89 (1.16x)**	**72.97 (+2.41)**
Loose	Zagoruyko et al. [6]	19.54	29.4	29.87	99.8 (0.79x)	3.34 (0.39x)	73.25 (+2.69)
	ResNet34 [1]	21.79	27.38	18.54	131.02 (1.03x)	7.07 (0.83x)	73.5 (+2.94)
	Tan et al. [10]	18.74	31.12	23.9	91.16 (0.72x)	3.81 (0.45x)	73.57 (+3.01)
	Dollar et al. [11]	16.21	19.35	21.27	144.09 (1.13x)	6.77 (0.79x)	72.62 (+2.06)
	NeuralScale [9]	17.96	18.52	29.3	215.1 (1.69x)	7.34 (0.85x)	72.89 (+2.33)
	HACScale (our)	17.14	19.21	33.21	**250.54 (1.97x)**	**7.54 (0.88x)**	**74.47 (+3.91)**

Experiments: CIFAR-10

- The superiority of HACScale also applies to other models and datasets.

Constraint	Method	Params (M)	Latency (ms)†	Power (W)	↑ Utilization (GFLOPS)	↑ Power Efficiency (GFLOPs/J)	↑ Top1 (%)
CIFAR-10 VGG11							
	Baseline [12]	9.2	11.92	7.62	12.58 (1x)	1.65 (1x)	92.06 (+0.0)
Tight	Zagoruyko et al. [6]	20.76	12.88	12.23	26.63 (2.12x)	2.18 (1.32x)	92.72 (+0.66)
	VGG16 [12]	14.73	18.45	9.14	17.02 (1.35x)	1.86 (1.13x)	93.83 (+1.77)
	Tan et al. [10]	16.28	15.37	11.9	32.11 (2.55x)	2.69 (1.63x)	92.88 (+0.82)
	Dollar et al. [11]	17.07	13.25	11.74	24.82 (1.97x)	2.11 (1.28x)	92.69 (+0.63)
	NeuralScale [9]	14.44	12.56	11.94	48.15 (3.83x)	4.03 (2.44x)	93.26 (+1.20)
	HACScale (our)	12.24	12.7	12.92	**113.57 (9.03x)**	**8.79 (5.33x)**	**94.29 (+2.23)**
Loose	Zagoruyko et al. [6]	36.89	28.63	12.98	21.25 (1.68x)	1.63 (0.99x)	92.98 (+0.92)
	VGG19 [12]	20.4	23.43	10.61	17.03 (1.35x)	1.64 (0.99x)	93.71 (+1.65)
	Tan et al. [10]	21.2	24.98	12.38	35.75 (2.84x)	2.89 (1.75x)	94.27 (+2.21)
	Dollar et al. [11]	29.74	24.58	13.64	23.4 (1.86x)	1.7 (1.03x)	92.78 (+0.72)
	NeuralScale [9]	36.9	29.19	13.98	48.58 (3.86x)	3.48 (2.11x)	93.31 (+1.25)
	HACScale (our)	21.1	25.5	14.15	**99.89 (7.94x)**	**7.06 (4.28x)**	**94.38 (+2.32)**

Experiments: Analysis of Parameter Efficiency

- With importance-aware width scaling, we achieve higher accuracy with less parameters.

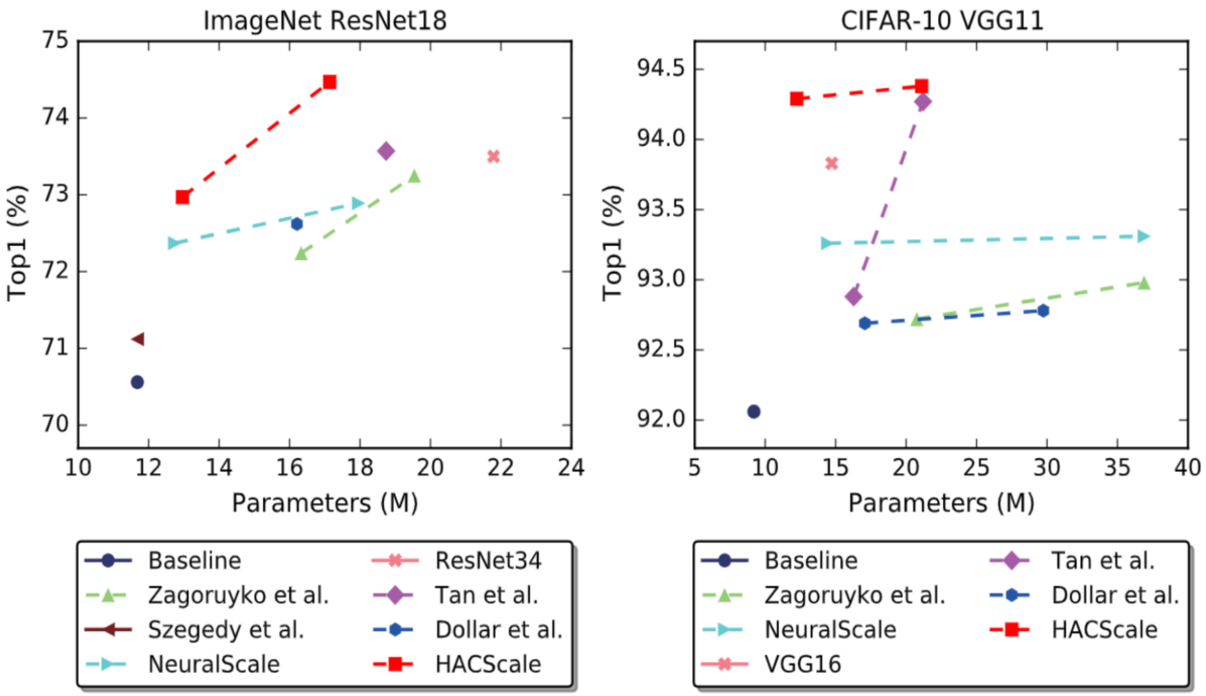


Fig. 4. The parameter efficiency of different scaling methods. The baseline models for ImageNet and CIFAR-10 are ResNet18 and VGG11, respectively.

Conclusion

- We propose a new scaling method, HACScale
  - Study the impact of different dimensions on accuracy and latency
  - Propose to combine width scaling and resolution scaling
  - Propose importance-aware width scaling
- Experimental results
  - Higher accuracy
  - Better parameter efficiency
  - Better energy efficiency

Thanks!