AutoSlim: Towards One-Shot Architecture Search for Channel Numbers

Jiahui Yu **Thomas Huang**
University of Illinois at Urbana-Champaign

Abstract

We study how to set the number of channels in a neural network to achieve better accuracy under constrained resources (FLOPs, latency, memory footprint or model size). A simple and one-shot solution, named AutoSlim, is presented. Instead of training many network samples and searching with reinforcement learning, we train a single slimmable network to approximate the network accuracy of different channel configurations. We then iteratively evaluate the trained slimmable model and greedily slim the layer with minimal accuracy drop. By this single pass, we can obtain the optimized channel configurations under different resource constraints. We present experiments with MobileNet v1, MobileNet v2, ResNet-50 and RL-searched MNasNet on ImageNet classification. Notably, by setting optimized channel numbers, our AutoSlim-MobileNet-v2 at 305M FLOPs achieves **74.2% top-1** accuracy, 2.4% better than default MobileNet-v2 (301M FLOPs), and even 0.2% better than RL-searched MNasNet (317M FLOPs). Our AutoSlim-ResNet-50 at 570M FLOPs, without depthwise convolutions, achieves **1.3% better** accuracy than MobileNet-v1 (569M FLOPs). Code and models are available at: `https://github.com/JiahuiYu/slimmable_networks`.

1 Introduction

The number of channels of a neural network plays a critical role in its affordability on resource constrained platforms, such as mobile phones, wearables and Internet of Things (IoT) devices. The most common constraints, i.e., latency, FLOPs and runtime memory footprint, are all bound to the number of channels. For example, in a single convolution or fully-connected layer, the FLOPs (number of Multiply-Adds) increases linearly by the output channels. The memory footprint can also be reduced by reducing the number of channels in bottleneck convolutions for most vision applications.

Despite its importance, the number of channels has been chosen mostly based on heuristics. In this work, we present *AutoSlim*, a simple and one-shot architecture search method for the number of channels. Our main idea lies in training a slimmable network [1] to approximate the network accuracy of different channel configurations. Yu et al. [1, 2] introduced slimmable networks that can run at arbitrary width with equally or even better performance than same architecture trained individually. Although the original motivation is to provide instant and adaptive accuracy-efficiency trade-offs, we find slimmable networks are especially suitable as benchmark performance estimators for several reasons: (1) Training slimmable models is much faster than the brute-force approach. (2) A trained slimmable model can execute at arbitrary width, which can be used to approximate relative performance among different channel configurations. (3) The same trained slimmable model can be applied on search of optimal channels for different resource constraints.

In *AutoSlim*, we first train a slimmable model for a few epochs (e.g., 10% to 20% of full training epochs) to quickly get a benchmark performance estimator. We then iteratively evaluate the trained slimmable model and greedily slim the layer with minimal accuracy drop on validation set (for

ImageNet, we randomly hold out $50K$ samples of training set as validation set). After this single pass, we can obtain the optimized channel configurations under different resource constraints (e.g., network FLOPs limited to 150M, 300M and 600M). Finally we train these optimized architectures individually or jointly (as a single slimmable network) for full training epochs. We experiment with various networks including MobileNet v1, MobileNet v2, ResNet-50 and RL-searched MNasNet on the challenging setting of 1000-class ImageNet classification. AutoSlim achieves better results (with much lower search cost) compared with three baselines: (1) the default channel configuration of these networks, (2) channel pruning methods on same network architectures [3, 4] and (3) reinforcement learning based architecture search methods [5, 6].

2 Related Work

Channel Pruning. Channel pruning methods [7, 4] aim at reducing effective channels of a large neural network to speedup its inference. Both training-based, inference-time and initialization-time pruning methods have been proposed [7, 4] in the literature.

Neural Architecture Search (NAS). Recently there has been a growing interest in automating the neural network architecture design. Significant improvements have been achieved by these automatically searched architectures in many vision and language tasks. However, most neural architecture search methods did not include channel configuration into search space, and instead applied human-designed heuristics. More recently, the RL-based searching algorithms are also applied to prune channels [5] or search for filter numbers [6] directly.

Slimmable Networks. Slimmable networks were firstly introduced in [1]. Yu et al. further introduced universally slimmable networks, extending slimmable networks to execute at arbitrary width, and generalizing to networks both with and without batch normalization layers. Meanwhile, two improved training techniques, *the sandwich rule* and *inplace distillation*, were proposed [2] to enhance training process and boost testing accuracy. Moreover, with the proposed methods, one can train nonuniform universally slimmable networks, where the width ratio is not uniformly applied to all layers. In other words, each layer in a nonuniform universally slimmable network can adjust its number of channels independently during inference. While the original motivation [1, 2] of slimmable networks is to provide instant and adaptive accuracy-efficiency trade-offs at runtime for different devices, we present an approach that uses slimmable networks for searching channel configurations of deep neural networks.

3 AutoSlim: Network Slimming by Slimmable Networks

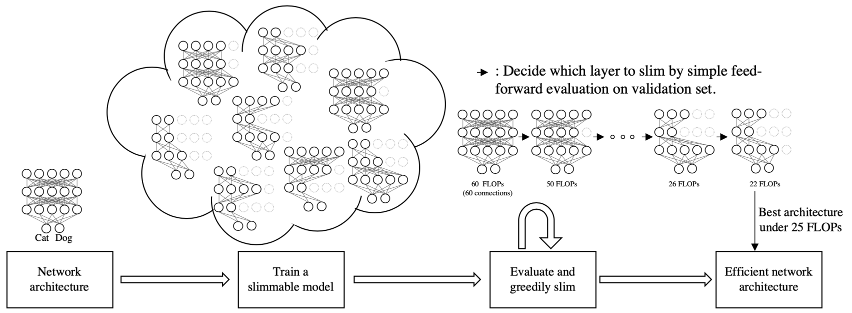
Cat Dog
: Decide which layer to slim by simple feed-forward evaluation on validation set.
60 FLOPs (60 connections)
50 FLOPs
26 FLOPs
22 FLOPs
Best architecture under 25 FLOPs
Network architecture
Train a slimmable model
Evaluate and greedily slim
Efficient network architecture

Figure 1: The flow diagram of our proposed approach *AutoSlim*.

The goal of channel configuration search is to optimize the number of channels in each layer, such that the network architecture with optimized channel configuration can achieve better accuracy under constrained resources. The constraints can be FLOPs, latency, memory footprint or model size. Our approach is conceptually simple, and it has two essential steps:

(1) Given a network architecture, we first train a slimmable model for a few epochs (e.g., 10% to 20% of full training epochs). During the training, many different sub-networks with diverse channel configurations have been sampled and trained. Thus, after training one can directly sample its

Table 1: ImageNet classification results with various network architectures. Blue indicates the network pruning methods [8, 4, 5, 9], Cyan indicates the network architecture search methods [6] and Red indicates our results using *AutoSlim*.

Group	Model	Parameters	Memory	CPU Latency	FLOPs	Top-1 Err. (gain)
200M FLOPs	ShuffleNet v1 1.0× [10]	1.8M	4.9M	46ms	138M	32.6
	ShuffleNet v2 1.0× [11]	-	-	-	146M	30.6
	MobileNet v1 0.5× [12]	1.3M	3.8M	33ms	150M	36.7
	MobileNet v2 0.75× [13]	2.6M	8.5M	71ms	209M	30.2
	AMC-MobileNet v2 [5]	2.3M	7.3M	68ms	211M	29.2 (1.0)
	MNasNet 0.75× [6]	3.1M	7.9M	65ms	216M	28.5
	AutoSlim-MobileNet v1	1.9M	4.2M	33ms	150M	32.1 (4.6)
	AutoSlim-MobileNet v2	4.1M	9.1M	70ms	207M	27.0 (3.2)
	AutoSlim-MNasNet	4.0M	7.5M	62ms	217M	26.8 (1.7)
300M FLOPs	ShuffleNet v1 1.5× [10]	3.4M	8.0M	60ms	292M	28.5
	ShuffleNet v2 1.5× [11]	-	-	-	299M	27.4
	MobileNet v1 0.75× [12]	2.6M	6.4M	48ms	325M	31.6
	MobileNet v2 1.0× [13]	3.5M	10.2M	81ms	300M	28.2
	NetAdapt-MobileNet v1 [9]	-	-	-	285M	29.9 (1.7)
	AMC-MobileNet v1 [5]	1.8M	5.6M	46ms	285M	29.5 (2.1)
	MNasNet 1.0× [6]	4.3M	9.8M	76ms	317M	26.0
	AutoSlim-MobileNet v1	4.0M	6.8M	43ms	325M	28.5 (3.1)
	AutoSlim-MobileNet v2	5.7M	10.9M	77ms	305M	25.8 (2.4)
	AutoSlim-MNasNet	6.0M	10.3M	71ms	315M	25.4 (0.6)
500M FLOPs	ShuffleNet v1 2.0× [10]	5.4M	11.6M	92ms	524M	26.3
	ShuffleNet v2 2.0× [11]	-	-	-	591M	25.1
	MobileNet v1 1.0× [12]	4.2M	9.3M	64ms	569M	29.1
	MobileNet v2 1.3× [13]	5.3M	14.3M	106ms	509M	25.6
	MNasNet 1.3× [6]	6.8M	14.2M	95ms	535M	24.5
	AutoSlim-MobileNet v1	4.6M	9.5M	66ms	572M	27.0 (2.1)
	AutoSlim-MobileNet v2	6.5M	14.8M	103ms	505M	24.6 (1.0)
	AutoSlim-MNasNet	8.3M	14.2M	95ms	532M	24.5
Heavy Models	ResNet-50 [14]	25.5M	36.6M	197ms	4.1G	23.9
	ResNet-50 0.75× [14, 1]	14.7M	23.1M	133ms	2.3G	25.1
	ResNet-50 0.5× [14, 1]	6.8M	12.5M	81ms	1.1G	27.9
	ResNet-50 0.25× [14, 1]	1.9M	4.8M	44ms	278M	35.0
	He-ResNet-50 [4, 8]	-	-	-	≈2.0G	27.2
	AutoSlim-ResNet-50	23.1M	32.3M	165ms	3.0G	24.0
		20.6M	27.6M	133ms	2.0G	24.4
		13.3M	18.2M	91ms	1.0G	26.0
		7.4M	11.5M	69ms	570M	27.8

sub-network architectures for instant inference, using the correspondent computational graph and same trained weights.

(2) Next, we iteratively evaluate the trained slimmable model on the validation set. In each iteration, we decide which layer to slim by comparing their feed-forward evaluation accuracy on validation set. We greedily slim the layer with minimal accuracy drop, until reaching the efficiency constraints. No training is required in this step.

The flow diagram of our approach is shown in Figure 1. Our approach is also flexible for different resource constraints, since the FLOPs, latency, memory footprint and model size are all deterministic given a channel configuration and a runtime environment. By a single pass of greedy slimming in step (2), we can obtain the (FLOPs, latency, memory footprint, model size, accuracy) tuples of different channel configurations. It is noteworthy that the latency and accuracy are relative values, since the latency may be different across different hardware and the accuracy can be improved by training the network for full epochs. In the setting of optimizing channel numbers, we benefit from these relative values as performance estimators.

The Search Space. The executable sub-networks in a slimmable model compose the search space of channel configurations given a network architecture. To train a slimmable model, we simply apply two width multipliers [2] as the upper bound and lower bound of channel numbers. For example, for all mobile networks, we train a slimmable model that can execute between $0.15\times$ and $1.5\times$. In each training iteration, we randomly and independently sample the number of channels in each layer.

Greedy Slimming After training a slimmable model, we evaluate it on the validation set (on ImageNet we randomly hold out $50K$ images in training set as validation set). We start with the largest model (e.g., $1.5\times$) and compare the network accuracy among the architectures where each layer is slimmed by one channel group. We then greedily slim the layer with minimal accuracy drop. During the iterative slimming, we obtain optimized channel configurations under different resource constraints. We stop until reaching the strictest constraint (e.g., 50M FLOPs or 30ms CPU latency).

4 Experiments

Table 1 summarizes our results on ImageNet classification with various network architectures including MobileNet v1 [12], MobileNet v2 [13], MNasNet [6], and one large model ResNet-50 [14]. As shown in Table 1, our models have better top-1 accuracy compared with the default channel configuration of MobileNet v1, MobileNet v2 and ResNet-50 across different computational budgets. We even have improvements over RL-searched MNasNet [6], where the filter numbers are already included in its search space.

References

[1] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[2] Jiahui Yu and Thomas Huang. Universally slimmable networks and improved training techniques. *arXiv preprint arXiv:1903.05134*, 2019.

[3] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. *arXiv preprint arXiv:1707.06342*, 2017.

[4] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 1398–1406. IEEE, 2017.

[5] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[6] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*, 2018.

[7] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2755–2763. IEEE, 2017.

[8] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[9] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300, 2018.

[10] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.

[11] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[13] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv preprint arXiv:1801.04381*, 2018.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.