RESIZABLE NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we present a deep convolutional neural network (CNN) which performs arbitrary resize operation on intermediate feature map resolution at stage-level. Motivated by weight sharing mechanism in neural architecture search, where a super-network is trained and sub-networks inherit the weights from the super-network, we present a novel CNN approach. We construct a spatial super-network which consists of multiple sub-networks, where each sub-network is a single scale network that obtain a unique spatial configuration, and the convolutional layers are shared across all sub-networks. Such network, named as *Resizable Neural Networks*, are equivalent to training *infinite* single scale networks, but has no extra computational cost. Moreover, we present a training algorithm such that all sub-networks achieve better performance than individually trained counterparts. On large-scale ImageNet classification, we demonstrate its effectiveness on various modern network architectures such as MobileNet, ShuffleNet, and ResNet.

To go even further, we present three variants of resizable networks: 1) Resizable as Architecture Search (*Resizable-NAS*). On ImageNet, Resizable-NAS ResNet-50 attain **0.4%** higher on accuracy and **44%** smaller than the baseline model. 2) Resizable as Data Augmentation (*Resizable-Aug*). While we use resizable networks as a data augmentation technique, it obtains superior performance on ImageNet classification, outperform AutoAugment by **1.2%** with ResNet-50. 3) Adaptive Resizable Network (*Resizable-Adapt*). We introduce the adaptive resizable networks as dynamic networks, which further improve the performance with less computational cost via data-dependent inference.

1 INTRODUCTION

Scale is the fundamental component of the physical world, and it has been an active research topic in the field of computer vision. The latest design of scales for the Convolutional Neural Networks (CNN) fall into two categories: single scale networks and multi-scale networks. In single scale networks, such as ResNet (He et al. (2016)), only a fixed scale representation is learned from the feature maps. Alternatively, in the multi-scale networks, e.g., image pyramid (Sermanet et al., 2013; Dalal & Triggs, 2005)/feature pyramid (Ghiasi et al., 2019; Chen et al., 2017a)/filter pyramid structure (Szegedy et al., 2015), features from different resolutions are fused in a network; thus the models learn various type of scales from the objects. As a result, these architectures obtain state-of-the-art performance on scale critical tasks, such as object detection (Lin et al. (2017)) and semantic segmentation (Chen et al. (2017a)). It can be observed that both single scale networks and multi-scale networks essentially perform representation learning on a predefined feature map scales. This natural and intuitive choice, despite their success on various computer vision tasks, limits the CNNs to a constrained predefined scaling mechanism for dealing with scale variation problem ((He et al., 2019)).

Our approach is motivated by the recent emergence of weight sharing approach on neural architecture search (NAS) (Pham et al. (2018); Bender et al. (2018); Guo et al. (2019)), where an over-parameterized super-network is trained and sub-networks inherit the weights from the super-network. Generally, there are a large number of architectural configurations within a super-network.

In this work, we propose a convolutional neural network with infinite single scale networks, named as *Resizable Neural Networks*. It can be viewed as a spatial super-network that includes vast sub-networks which varies in spatial configurations. Specifically, a resizable network contains multiple

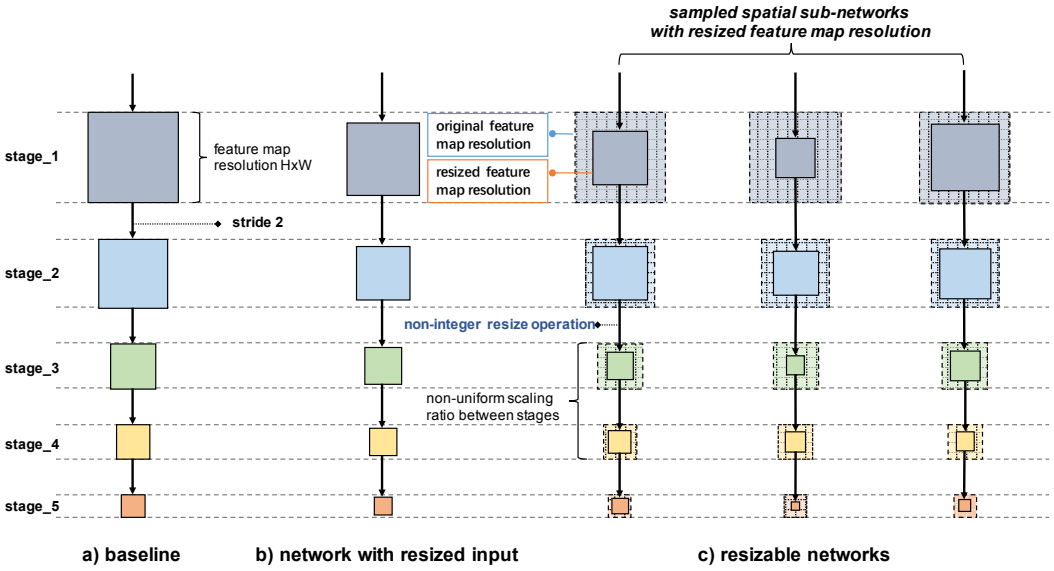

Figure 1: **Resolution scaling**. (a) is a baseline network example. (b) are conventional scaling on input resolution. (c) is our proposed resizable networks. It resizes the feature map resolution by a non-interger scaling ratio at the stage-level. Different spatial sub-networks are jointly trained in a single network with shared convolution. The imaginary line represents the original feature map resolution, and the full line represents the actual feature map resolution after resize operation.

sub-networks: each sub-network obtains a unique feature map scale. The resize operation for feature map resolution is performed on the stage-level by a non-integer scaling ratio. These sub-networks, called spatial sub-networks, are trained jointly in the resizable networks, and the weights of all convolution layers are shared. Figure 1 gives an illustration of the resizable networks compared with the baseline network and input resize only network. Note that the network with resized input resolution is a special form of spatial sub-networks.

Using the approach of resizable networks, the network capacity can be significantly increased as an exponential number of scales for input has been trained by the networks implicitly. Additionally, we present a training algorithm to train the proposed resizable networks, which resolve the unfair sampling issue as well as the discrepancy of batch normalization statistics between different spatial sub-networks. With our proposed training algorithm, we can omit the process of retraining, which is a standard procedure in weight sharing NAS, and obtain higher accuracy for all spatial sub-networks than individually trained counterparts.

In this paper, we first validate the effectiveness of resizable neural networks compared with stand-alone networks. We demonstrate superior performances of Resizable Neural Network on efficient networks such as MobileNet V1, ShuffleNet V2 as well as deep networks, i.e., ResNet-50. The experiments cover the range of modern neural architecture setting such as non-residual block, residual bottleneck, depth-wise convolution, and deep networks.

Variants of Resizable Neural Network. We further demonstrate three resizable network variants: resizable as architecture search (*Resizable-NAS*), resizable as data augmentation (*Resizable-Aug*) and resizable for data-dependent inference (*Resizable-Adapt*).

Resizable-NAS. We demonstrate that our resizable networks can adaptively switch between sub-networks to fulfil the efficiency constraints on the target platform, thus achieve the run-time model compression objective. We show that there consistently exists smaller sub-network with similar (even higher) accuracy than the baseline. For example, Resizable-NAS ResNet-50 is **44%** smaller on model size and **0.4%** higher on accuracy compare with the baseline. This result is significantly

better than state-of-the-art compression methods, i.e., the accuracy of NISP (Yu et al. (2018b)) is dropped by 0.9% with the same computational complexity.

Resizable-Aug. We introduce Resizable-Aug, a data augmentation technique using resizable network training scheme. We are able to improve ResNet-50 by **2.4%** compare with baseline. This result is also higher than state-of-the-art data augmentation methods, for instance, it is **1.2%** higher than AutoAugment (Cubuk et al. (2018)) on ImageNet classification task with no search cost. Additionally, combine our augmentation technique on a new baseline model, we obtain a model achieve **77.8%** with **365M** FLOPs. This is significantly better than previous state-of-the-art model EfficientNet-B0 (Tan & Le (2019)) which trained with AutoAugment (Cubuk et al. (2018)), **1.5%** higher on accuracy with **25M** less FLOPs.

Resizable-Adapt. We introduce an auxiliary module called *ResizeLearner*. By integrating into resizable networks, it allows data-dependent inference. This is achieved by adopting ResizeLearner to assign the optimal spatial sub-network to each sample. Compared with the naive resizable networks, this approach not only lowers the computational cost during inference, but improve representation learning via switch the spatial policy from implicit to explicit. In particular, Resizable-Adapt ResNet-50 improve the performance of Resizable-NAS ResNet-50 up to **0.5%**, depends on the target spatial sub-networks.

2 Related Work

Multi-scale Training. Our work is most related to multi-scale learning, which is important in image recognition (Li et al. (2019); Wang et al. (2019); Chen et al. (2018)), object detection(Lin et al. (2017); Singh et al. (2018); Singh & Davis (2018)) and instance segmentation (Zhao et al. (2017); Chen et al. (2017b)). There are several approaches to fusing information of images at different visual resolutions. The naive approach is image pyramid (Sermanet et al. (2013); Dalal & Triggs (2005); Felzenszwalb et al. (2009); Cai et al. (2016)), where an input image is passed through a model multiple times at different resolutions. Our work introduce the resizable networks which comprise infinite single scale network, that acted as a implicit multi-scale networks.

Dynamic Neural Networks. Our work is related to dynamic neural networks. Dynamic neural networks aim to train a single network to support different architectural configuration, especially adjust networks according to input samples. Slimmable networks(Yu et al. (2018a)) train a single model to support multiple width configuration by switching width at runtime. These dynamic networks are limited by two factors: 1) Performance. Most dynamic networks methods sacrifice accuracy in exchange of adaption in inference. 2) Flexibility. The flexibility previous methods are limited, and the degree of flexibility are limited to width or depth, and the number of sub-networks choice is . For example, slimmablenet (Yu et al. (2018a)) have only four to eight switches to adapt in inference. Our proposed model is highly flexible on spatial-level, while obtain an excellent performance gain over all individual network.

Model Compression. Compressing Over-parametrized model network with minimal performance drop is appealing on resource-constrained devices. Modern compression methods including channel pruning (Liu et al. (2018); He et al. (2017)), knowledge distillation(Hinton et al. (2015); Furlanello et al. (2018); Romero et al. (2014); Zhang et al. (2018); Chen et al. (2015a)), hashing (Chen et al. (2015b)), network binarization(Courbariaux et al. (2016); Rastegari et al. (2016)). Our work explore network compression at the runtime on spatial-level, which is complementary to existing compression methods.

Neural Architecture Search. Recently the design of efficient neural networks has largely shifted from leveraging human knowledge to automatic methods, which is known as neural architecture search(NAS). Designing search space is of the most important in NAS on various vision tasks such as image recognition (Radosavovic et al. (2019); Cai et al. (2018); Howard et al. (2019); Guo et al. (2019); Tan et al. (2019)), instance segmentation (Liu et al. (2019)) and object detection(Ghiasi et al. (2019); Chen et al. (2019)). Resizable networks can be consider as a designed search space, where each spatial sub-networks is a optional architecture.

Data Augmentation. Data augmentation is effective for improving the accuracy of modern neural network in various computer vision tasks (Szegedy et al. (2015)). Many new augmentation techniques such as Cutout (DeVries & Taylor (2017)) and Mixup (Zhang et al. (2017)) on input image

have shown promising results. Recently, a new trend is raised that automatically learn the data transformation (Ho et al. (2019)). Cubuk et al. (2018) leverage RNN controller to automatically search the best data augmentation combination for training in a large augmentation sub-policy space. We explore the data augmentation technique on feature map spatial-level by letting the network learns different feature map scales.

3 RESIZABLE NEURAL NETWORK

In this section, we present the resizable networks. Section 3.1 gives a description of resizable networks. Section 3.2 describe the training algorithm. Section 3.2 to 3.4 describe three variants of resizable networks respectively.

3.1 DEFINITION OF RESIZABLE NETWORKS

Let $s_j^i \in [0, 1]$ be a *scale factor*, each s_j^i is a jth scale ratio at ith stage. We use a set of scale factors to describe a network. We denote the $S = \{s_j^i\}_{i=1,j\in N}^n$ to be the *spatial configuration*, where N denotes the number of total stages in a network. Let $\boldsymbol{x}_{in}$ denotes input tensor as $\mathrm{C} \times H_{in} \times W_{in}$, where C denotes the number of channels, H_{in} denotes the height and W_{in} denotes the width of the input tensor. We can formalize the feature map resize operation as:

$$\boldsymbol{x}_{out} = G(\boldsymbol{x}_{in}; s_j^i) \tag{1}$$

where $G(\cdot;\cdot)$ denotes the resize operation. $\boldsymbol{x}_{out}$ is the output tensor as $\mathrm{C} \times H_{out} \times W_{out}$ after resize operation. The typically choice for the scale is $s_j^i = 0.5$ (He et al. (2016)) and $G(\cdot;\cdot)$ use the max/average pooling or convolutional layer with stride equal to 2. Thus we can write the conventional single scale network as $S = \{0.5_j\}_{j\in N}$. Unlike the conventional single scale network, in the *spatial sub-networks*, the s_j^i is a arbitrary number and $G(\cdot;\cdot)$ represent the bilinear interpolation. This leads to a flexible scaling policy that enable the feature map resolution change to an arbitrary size. Thus we can formalized the resizable networks as the ensemble of an infinite number of single scale networks:

$$RS = \{S_k\}_{k=1}^N \tag{2}$$

where RS denotes the resizable networks, N is the total number of single scale networks. S_k is some single scale network with flexible scaling policy as we described in equation 1. From the equation 2, resizable networks can be viewed as a set of infinite single scale network, where each single scale network S_k obtain a unique spatial configuration while all networks share the same convolutional layers. We present a training algorithm for resizable networks in Section 3.2.

3.2 TRAINING DETAILS

Fair Sampling. The naive approach to train a resizable networks is to uniformly select a spatial configuration at each optimization step and update the weights accordingly. However, such sampling method brings out training bias. Each convolutional layer at different spatial configuration is trained with different mini-batch data and learning rate, which causes the issue of unfair training between different sub-networks. To be concrete, some convolutional layers can go through more training iterations than the others. To reduce the training bias, we introduce a fair sampling method. The ideology behind our proposed sampling is simple: to fairly train every scale factors. Specifically, we propose to train multiple configurations in each iteration and accumulate the gradients of shared convolutional layers. For each iteration, we uniformly sample scale ratios in a non-repetitive manner: if a convolutional layer with the particular scale factor has already been chosen, then this particular layers will not be picked again in the same iteration. Thus we ensure all scale ratios for convolution is trained fairly.

Scale-Aware Batch Normalization. One issue with training a network layer with different feature map size is the discrepancy of both feature mean and variance between different sampled scales. Thus shared batch normalization (BN) (Ioffe & Szegedy (2015)) layer brings instability into the training process, degrade the model performance. To address this issue, we use scale-aware batch normalization in the network. Specifically, the shared convolutional layer is followed by a group of BN layers, where the number of BN layers in this group is equal to the number of scale factors.

Thus each scale ratio is corresponding to an independent BN layer which is scale-aware. Note that the extra parameters and expense of memory introduced by scale-aware batch normalization is negligible.

Training Algorithm. Algorithm 1 illustrates the training framework to efficiently and effectively train a resizable network combine with fair sampling and BN calibration techniques. During training, we adopt fair sampling, which gradients are accumulated and fairly sample spatial configurations in each iteration and then update the weights.

Algorithm 1 Training resizable neural networks RS.

Require: Initialize resizable network RS. Predefined spatial list L
for i = 1, ..., n_{iters} **do**
  Get data x and target y of current mini-batch.
  Clear gradients for all parameters.
  for j = 0, ..., len(L) **do**
    Fair sample spatial sub-network, RS'.
    Select the scale-aware batch normalization for each stage.
    Execute spatial sub-networks. $\hat{y} = RS'(x)$.
    Compute gradients for RS' based on loss.
  end
  Update weights by accumulated gradients.
end

3.3 Resizable Neural Network As Model Search

Using the training algorithm in Section 3.2, the resizable networks now contain multiple spatial sub-networks, where all sub-networks attain good performance. Thus, with a trained super-network, the primary objective is to select the best-performing spatial sub-network given different efficiency constraints (FLOPs, latency, energy, etc.). Generally, various search methods can be combined with the resizable networks for searching the best-fit sub-networks. For simplicity, we present a naive approach. We enumerate all spatial sub-networks and build a lookup table to include information such as accuracy, FLOPs and latency. Therefore, given a target hardware and efficiency constraints, we can query the lookup table to get the best-fit sub-network in the resizable networks. There is no computational cost of querying the lookup table.

Additionally, when the best-fit spatial sub-network is chosen from the lookup table, we can fine-tune this sub-network to increase the accuracy on target dataset further. It allows the sub-network to recover its representation learning ability when it is under-trained in the resizable networks. It also mitigates the co-adaptation issue which we will discuss in Section 3.3. Empirically, we find that fine-tune 15 epochs is sufficient, and the fine-tuning process increases the accuracy considerably.

3.4 Resizable Neural Network as Data Augmentation

In Section 3.2, we present a training algorithm to effectively train resizable networks, which make sure that all the sub-networks obtain good accuracy. Such training algorithm compromises the accuracy between the target network and the others. In Section 3.3, we further show that significant performance enhancement can be achieved by fine-tuning the target sub-network. This performance gain is not only coming from more training on the target network but to alleviate the co-adaptation issue, which we will discuss below.

Co-adaptation. Co-adaptation is a common phenomenon when network operations in a single model are shared (Bender et al., 2018). Because of the co-adaptation, when a spatial sub-network is trained, the ability of other sub-networks will be impaired if they shared the same convolutional layers with a particular scale factor. When we adopt resizable networks as a data augmentation technique, we aim to achieve the best performance on the target network. If we train a resizable network naively, the shared convolutional layers can co-adapt. Thus we introduce a simple technique to alleviate the issue of co-adaptation, named as scheduled switching.

Table 1: This table shows the result of Top-1 accuracy of spatial configurations in resizable networks compare with individually trained counterparts.

Individual Networks			**Resizable Networks**		
Name	FLOPs	Top-1 Acc.	Name	FLOPs	Top-1 Acc.(gain)
Individual-MobileNetV1	569M	70.6%	Resizable-MobileNetV1	569M	72.0%(+1.4%)
	462M	70.0%		462M	71.3%(+1.3%)
	391M	69.5%		391M	70.7%(+1.2%)
	291M	68.1%		291M	69.5%(+1.4%)
Individual-ShuffleNetV2	299M	72.6%	Resizable-ShuffleNetV2	299M	73.3%(+0.7%)
	251M	71.5%		251M	72.2%(+0.7%)
	199M	70.3%		199M	70.9%(+0.6%)
	147M	69.2%		147M	69.8%(+0.6%)
Individual-ResNet-50	4.1G	76.4%	Resizable-ResNet-50	4.1G	77.1%(+0.7%)
	2.8G	75.4%		2.8G	76.6%(+1.2%)
	2.0G	74.2%		2.0G	75.2%(+1.0%)
	1.1G	71.6%		1.1G	72.3%(+0.7%)

Scheduled Switching. Instead of training the resizable networks follow the Section 3.2, we decompose the entire training procedure into two-stage. At the first stage, we follow the same training algorithm as we described in the Section 3.2. At the second stage, we stop the current training algorithm at some scheduled point, and switch to the normal optimization pipeline, where only the target network is sampled and optimized. This simple technique leverage the advantages of implicit multi-scale learning from the resizable networks, as well as resolve the layer co-adaption problem.

3.5 Adaptive Resizable Neural Network

There are two drawbacks in the original resizable networks. First, it is inefficient to process all images to one sub-network, as the algorithm spends equal energy at every sample. Secondly, the scaling policy in the original resizable network is implicit, which may be sub-optimal. To tackle these issues, we introduce a ResizeLearner, such that resizable networks can make data-dependent inference by assigning samples to different spatial sub-networks. This explicit scaling policy, while less computational power is used, enhance the ability of the networks.

The pipeline of adaptive resizable network is simple yet effective: After a resizable network is trained, we initialize the ResizeLearner. It is composed of three convolutional layers along with a pooling layer and followed by the softmax, attached at the last stage of the original network. Note that we freeze the parameters of the original network, such that only the ResizeLearner update the weights. For the training, we randomly sample 50k images from the training data, validate on all spatial sub-networks for each sample, and take one with the highest accuracy as the target. During inference, we assign each sample with the optimal spatial sub-networks generated by ResizeLearner.

4 Experiments

In this section, we validate the resizable networks on ImageNet classification from several applications. For consistency, we use RS- to indicate resizable network, ft to indicate resizable-nas networks by fine-tuning, RS-NAS to denote the fine-tuned target spatial sub-network, RS-Aug to represent resizable augmentation. RS-Adapt to denote adaptive resizable networks. Implementation details can be found in the Appendix A.1. Section 4.1 give experiments for resizable networks compared with individual networks. Section 4.2-4.4 gives the result of resizable-nas, resizable-aug and resizable-adapt network respectively.

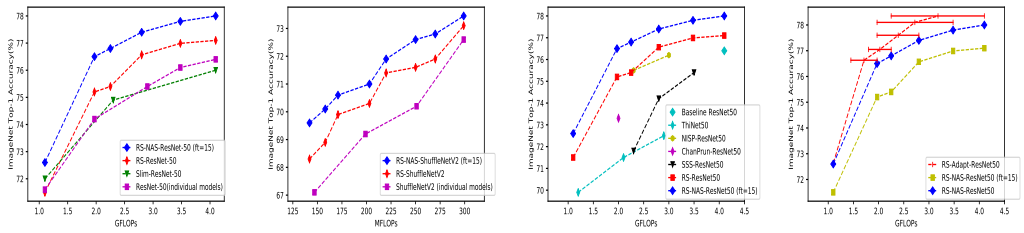

Figure 2: (a)-(b): Resizable-NAS ResNet-50 and ShuffleNetV2 compare with the individually trained networks and naive resizable approach. (c): Resizable-NAS ResNet-50 compare with state-of-the-art model compression methods. (d) Resizable-Adapt ResNet-50 compare with the Resizable-NAS ResNet-50. The horizontal line mean the FLOPs range of sub-network used in resizable-adapt network for data-dependent inference.

Table 2: This table summarizes the validation Top-1 accuracy on ImageNet dataset for ResNet-50 architecture. We compare Resizable-Aug with multiple state-of-the-art augmentation methods and regularization methods. Bold indicates the best results. * denotes results reported in the original papers.

Model	**Top-1 Acc.(gain)**	**Top-5 Acc.(gain)**
Baseline ResNet-50 (He et al. (2016))	76.4%	93.1%
ResNet-50 + Cutout (DeVries & Taylor (2017))	76.5%	93.2%
ResNet-50 + MixUp* (Zhang et al. (2017))	76.7%	93.4%
ResNet-50 + AutoAugment* (Cubuk et al. (2018))	77.6%	93.8%
ResNet-50 + Resizable-Aug	**78.8%**(+2.4%)	**94.4%**(+1.3%)

4.1 Comparison between Resizable Networks with Stand-Alone Network.

We show in Table 1 the top-1 classification accuracy for both individually trained networks and resizable networks give the same spatial configurations. Due to the space limit, we only take several spatial configurations for comparison. We present top-1 test accuarcy and FLOPs, which is the primary concerns in inference time, of all networks for reference. Compared to independently trained networks, the spatial sub-networks in resizable networks achieved significantly better top-1 accuracy, across various network architectures and computational complexity. This result confirm our hypothesis that the implicit multi-scale training improve the model capacity.

4.2 The result of Resizable-NAS Networks.

Comparison with Naive Resizable Networks. The Figure 2(a) and 2(b) demonstrate the results of resizable networks and resizable-nas network with ResNet-50 and ShuffleNetV2. The only difference between these two networks is that the resizable-nas network can 15 epochs fine-tune on target sub-network. It shows the notable performance improvement on individual sub-network when it is selected and fine-tuned compare with the baseline method.

Comparison with State-of-the-art Model Compression Methods. We compare with the following state-of-the-art model compression methods: NISP (Yu et al. (2018b)), ThiNet (Luo et al. (2017)), ChannelPruning (He et al. (2017)), Sparse Structure Selection ResNet50 (Huang & Wang (2018)) and SlimmableNets (Yu et al. (2018a)), AMC (He et al., 2018) and Netadapt (Yang et al., 2018). We present the ResNet-50 based results at Figure 2(c), and the detailed results of Resizable-NAS MobileNetV1 as well as Resizable-NAS ResNet-50 can be found in the Table 4. Resizable-NAS ResNet-50 demonstrate strong performance, especially when the compression ratio is large than 50%. It outperforms listed state-of-the-arts compression methods by a large margin over diverse computational complexity constraints. We emphasize that our Resizable-NAS network only trained

Table 3: Comparison with state-of-the-art architectures on ImageNet (200M to 400M FLOPs). * denotes reported results employ AutoAugment.

Model	FLOPs (M)	Params (M)	Top-1 Acc.	Top-5 Acc.
ShuffleNetV2 1.5x (Ma et al., 2018)	300	-	72.6%	-
MobileNetV2 1.0x (Sandler et al., 2018)	300	3.4	72.0%	91.0%
MobileNetV3 Large (Howard et al., 2019)	291	5.4	75.2%	92.2%
MnasNet-A2 (Tan et al. (2019))	340	4.8	75.6%	92.7%
EfficientNet-B0* (Tan & Le (2019))	390	5.3	76.3%	93.2%
ResizableNet w/o Aug	365	6.7	77.4%	93.6%
ResizableNet	365	6.7	77.8%	93.6%

once, and fine-tune 15 epochs on target sub-networks, where the other compression methods find the optimal structure and retrain accordingly. Comparing to the other methods, our proposed model is more efficient while performing better. Notice even the version without the fine-tuning process can achieve similar (even better) performance than other model compression methods.

4.3 Comparison with State-of-the-art Augmentation methods

On ResNet-50. We compares the Resizable-Aug ResNet-50 with various state-of-the-art data augmentation methods (i.e., Cutout (DeVries & Taylor (2017)), Mixup (Zhang et al. (2017)) and AutoAugment (Cubuk et al. (2018))). The results are summarized in Table 2. It shows that Resizable-Aug improve the ResNet-50 baseline by 2.4%, outperform AutoAugment by 1.2% using the same the baseline.

On Efficient Models. We present the resizable augmentation results based on network that are search by AutoML. The details of the training strategy and network architecture are presented in the Appendix A.3. Table 3 shows the results. By adopting the resizable augmentation strategy, we improve the already high accuracy network from 77.4% to 77.8%, achieve a new state-of-the-art result. The new models is 1.5% better than EfficientNet-B0 (Tan & Le (2019)), which is trained AutoAugment (Cubuk et al., 2018) with 25M less FLOPs.

4.4 Resizable-Adapt Networks Results

Figure 3(d) shows the results of Resizable-Adapt ResNet-50. The dynamic inference is performed on multiple computation complexity threshold. During inference, we ensure that only sub-networks that are smaller than the target network is used for inference. The performance gain is range from 0% to 0.5%. It can be observed that the adaptive resizable ResNet-50 consistently obtain better performance than the baseline. This is a surprising result since most dynamic network struggle at the trade-off between efficiency and accuracy. Less computational complexity in dynamic network always lowers the accuracy compared with the baseline. Moreover, when the target network is large, sometimes the samples are still better predicted by the small sub-network. This gives the evidence that the explicit way to deal with the input scale is indeed beneficial.

5 Conclusion

We introduced a type of convolutional neural network, named as Resizable Neural Networks. The resizable networks consist of infinite single scale networks, where each single scale networks obtain a unique spatial configuration. Based on this, we present a training algorithm to train this network, such that all spatial sub-networks in the resizable networks attain good accuracy. Moreover, we apply three variants of resizable networks. Specifically, we present Resizable-NAS, Resizable-Aug, and Resizable-Adapt. The resizable-NAS network permits run-time model compression by searching for the best-fit sub-network under efficiency constraints. Moreover, we present adaptive resizable networks, which perform data-dependent dynamic inference. Overall, we believe that the resizable network's approach provides an interesting and practical new perspective on designing a convolution neural network on the spatial dimension.

REFERENCES

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pp. 549–558, 2018.

Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. Path-level network transformation for efficient architecture search. *arXiv preprint arXiv:1806.02639*, 2018.

Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *european conference on computer vision*, pp. 354–370. Springer, 2016.

Chun-Fu Chen, Quanfu Fan, Neil Mallinar, Tom Sercu, and Rogerio Feris. Big-little net: An efficient multi-scale feature representation for visual and speech recognition. *arXiv preprint arXiv:1807.03848*, 2018.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4): 834–848, 2017a.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017b.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015a.

Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pp. 2285–2294, 2015b.

Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. Detnas: Neural architecture search on object detection. *arXiv preprint arXiv:1903.10979*, 2019.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. 2005.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7036–7045, 2019.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019.

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*, pp. 770–778, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

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)*, pp. 784–800, 2018.

Zewen He, He Huang, Yudong Wu, Guan Huang, and Wensheng Zhang. Consistent scale normalization for object recognition. *arXiv preprint arXiv:1908.07323*, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, and Xi Chen. Population based augmentation: Efficient learning of augmentation policy schedules. *arXiv preprint arXiv:1905.05393*, 2019.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*, 2019.

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.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 304–320, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Yi Li, Zhanghui Kuang, Yimin Chen, and Wayne Zhang. Data-driven neuron allocation for scale aggregation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11526–11534, 2019.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 82–92, 2019.

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

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

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)*, pp. 116–131, 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. *arXiv preprint arXiv:1905.13214*, 2019.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3578–3587, 2018.

Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, pp. 9310–9320, 2018.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

Huiyu Wang, Aniruddha Kembhavi, Ali Farhadi, Alan L Yuille, and Mohammad Rastegari. Elastic: Improving cnns with dynamic scaling policies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2258–2267, 2019.

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)*, pp. 285–300, 2018.

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

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, 2018b.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4320–4328, 2018.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.

Table 4: This table compares the Top-1 accuracy of Resizable Network with several state-of-the-art model compression methods given FLOPS. Bold indicates highest top-1 accuracy under certain FLOPS constraint. Blue indicates dynamic networks.

Model	Top-1 Acc.	Gain	FLOP	Compres. Ratio
ResNet50 (He et al. (2016))	76.4%	+0%	4.1G	0%
RS-NAS-ResNet50 (ft=15)	**77.8%**	**+1.4%**	**3.5G**	**15%**
RS-ResNet50	77.1%	+0.7%	3.5G	15%
RS-NAS-ResNet50 (ft=15)	**77.4%**	**+1.0%**	**2.8G**	**32%**
RS-ResNet50	76.6%	+0.2%	2.8G	32%
SSS-ResNet50 (Huang & Wang (2018))	74.2%	-2.2%	2.8G	32%
ThiNet-70 (Luo et al. (2017))	72.5%	-3.9%	2.9G	29%
NISP-ResNet50-A (Yu et al. (2018b))	76.2%	-0.2%	3.0G	27%
RS-NAS-ResNet50 (ft=15)	**76.8%**	**+0.4%**	**2.3G**	**44%**
RS-ResNet50	75.4%	-1.0%	2.3G	44%
Slim ResNet50 (Yu et al. (2018a))	75.0%	-1.4%	2.3G	44%
SSS-ResNet50 (Huang & Wang (2018))	75.6%	-0.8%	2.3G	44%
NISP-ResNet50-B (Yu et al. (2018b))	75.5%	-0.9%	2.3G	44%
RS-NAS-ResNet50 (ft=15)	**76.5%**	**+0.1%**	**2.0G**	**51%**
RS-ResNet50	75.2%	-1.2%	2.0G	51%
ChanPrun-ResNet50 (He et al. (2017))	73.3%	-3.1%	2.0G	51%
ThiNet-50 (Luo et al. (2017))	72.5%	-3.9%	2.1G	49%
RS-NAS-ResNet50 (ft=15)	**72.8%**	**-3.6%**	**1.1G**	**73%**
RS-ResNet50	72.3%	-4.1%	1.1G	73%
Slim-ResNet50 (Yu et al. (2018a))	72.0%	-4.4%	1.1G	73%
ThiNet-30 (Luo et al. (2017))	69.9%	-6.5%	1.2G	71%
MobileNetV1 (Howard et al. (2017))	70.9%	+0%	569M	0%
RS-NAS-MobileNetV1 (ft=15)	**71.9%**	**+1.0%**	**391M**	**31%**
RS-NAS-MobileNetV1 (ft=15)	**70.9%**	**+0%**	**291M**	**49%**
RS-MobileNetV1	69.5%	-1.4%	291M	49%
Slim-MobileNetV1 (Yu et al. (2018a))	69.5%	-1.4%	317M	44%
NetAdapt-MobileNetV1 (Yang et al. (2018))	70.1%	-0.9%	285M	50%
AMC-MobileNetV1 (He et al. (2018))	70.5%	-0.4%	285M	50%

A Appendix

A.1 Implementation Details.

Resize operation is place at the first non-stride block at each stage in network. We apply bilinear interpolation to resize the feature map to target size. Resize factor are uniformly taken from a predefined list of possible resize choice, except when it contradicted to fair sampling rule. Our implementation is based on the Pytorch (Paszke et al. (2017)). The input resolution for all train and test images for ImageNet classification are 224 x 224.

Other than ResNet-50, which we report baseline higher than the original paper, we have trained Mobilenet v1, Mobilenet v2 and Shufflenet v2 baseline to match the performance as reported in original paper. And based on the re-implemented settings, we perform both baseline and resizable networks experiments.

For ResNet-50, we train for 100 epochs using weight decay of 1e-4. We use batch size of 512 and do learning rate decay by 10 at epochs 30, 60, 80, 90. The other resizable network train for 200 epochs and double the epochs of learning rate decay.

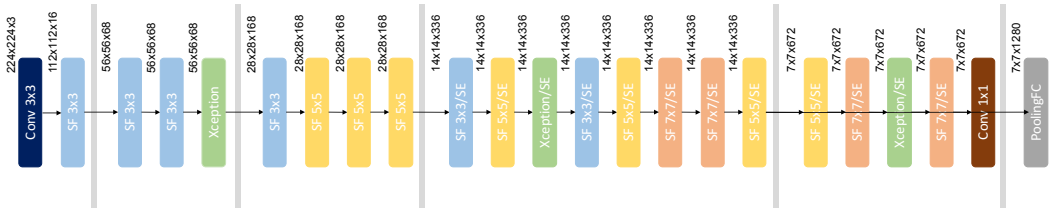

Figure 3: Architecture of ResizableNet. We highlight the input and output tensor shape. Conv denotes convolution layer. SF denotes ShuffleNetV2 module. SE denotes Squeeze-and-Excitation module.

For Mobilenet v1, Mobilenet v2 and Shufflenet v2, we use batch size of 1024 and weight decay of 4e-5. We train both models for 240 epochs using linear learning rate decay where base learning rate is set to 0.5. All models are train across synchronous SGD across 8 GPUs. The resizable networks train for 480 epochs

The resizable augmentation network always use the same training setting as the baseline method. Extending the training epochs is unnecessary.

A.2 Detail Results of Resizable-NAS Networks

The detail results of Resizable-NAS ResNet-50 and Resizable-NAS MobileNetV1 can be found at Table 4.

A.3 Details of ResizableNet

Training Details. ResizableNet is trained using resizable-aug technique as we discuss in the Section 3.4. It is trained for 480 epochs with batch size 1024 on 8 GPUs. The network parameters are optimized using an SGD optimizer with an initial learning rate of 0.5 (decayed linearly after each iteration), a momentum of 0.9 and a weight decay of 3×10^{-5}. Additional enhancements including label smoothing (Szegedy et al. (2016)) and Dropout with 0.5 on last FC layer. We follow the common practice to use inception augmentation.

Detailed Network Architecture. The network architecture along with feature map resolution and channels number are shown in Figure 4.

B Ablation Study

B.1 Is neural network naturally resizable?

It is intuitive to ask the question: is neural network naturally resizable? The answer is no. We present the result of directly inference on a neural network with different spatial configurations and compare the result with resizable neural network. The naive network is trained with scale-aware batch normalization. Figure 5 shows that naively trained neural network can not adjust feature map size, which result in a extremely low test accuracy.

B.2 Scale-Aware BN Analysis.

Effect on Resizable Augmentation. Spatial BN calibration is especially important for resizable augmentation. The following table compare two network adopt resizable augmentation with and without BN calibration. We can observed that without BN calibration, the resizable augmentation is only slightly better than the baseline model. The improvement is negligible. On the other hand, when BN calibrated for spatial configurations, is boost the performance compare with the baseline. The accuracy is increased by 1.1% on ShuffleNetV2 and 2.4% on ResNet-50.

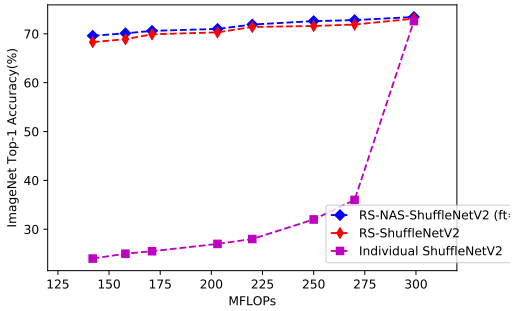

Figure 4: **Is network naturally resizable?** We compare the result of sub-networks in naive ShuffleNetV2, RS-ShuffleNetV2 and RS-NAS-ShuffleNetV2.

Table 5: This table shows the result of Top-1 accuracy of ResNet-50 and ShuffleNetV2 use resizable augmentation technique with (w/) and without (w/o spatial BN calibration.)

Model	Baseline	**w/o BN Calibration**	**w/ BN Calibration**
Resizable-Aug ShufflenetV2	72.6%	72.8%	73.7%
Resizable-Aug ResNet-50	76.4%	76.9%	78.8%