Smart Scissor: Coupling Spatial Redundancy Reduction and CNN Compression for Embedded Hardware

Hao Kong[1,2], Di Liu[2], Shuo Huai[1,2], Xiangzhong Luo[1], Weichen Liu[1],
Ravi Subramaniam[3], Christian Makaya[3], and Qian Lin[3]
[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore
[3]HP Inc., Palo Alto, California, USA

ABSTRACT

Scaling down the resolution of input images can greatly reduce the computational overhead of convolutional neural networks (CNNs), which is promising for edge AI. However, as an image usually contains much spatial redundancy, e.g., background pixels, directly shrinking the whole image will lose important features of the foreground object and lead to severe accuracy degradation. In this paper, we propose a dynamic image cropping framework to reduce the spatial redundancy by accurately cropping the foreground object from images. To achieve the instance-aware fine cropping, we introduce a lightweight foreground predictor to efficiently localize and crop the foreground of an image. The finely cropped images can be correctly recognized even at a small resolution. Meanwhile, computational redundancy also exists in CNN architectures. To pursue higher execution efficiency on resource-constrained embedded devices, we also propose a compound shrinking strategy to coordinately compress the three dimensions (depth, width, resolution) of CNNs. Eventually, we seamlessly combine the proposed dynamic image cropping and compound shrinking into a unified compression framework, Smart Scissor, which is expected to significantly reduce the computational overhead of CNNs while still maintaining high accuracy. Experiments on ImageNet-1K demonstrate that our method reduces the computational cost of ResNet50 by 41.5% while improving the top-1 accuracy by 0.3%. Moreover, compared to HRank, the state-of-the-art CNN compression framework, our method achieves 4.1% higher top-1 accuracy at the same computational cost. The codes and data are available at https://github.com/ntuliuteam/smart-scissor

1 INTRODUCTION

Modern convolutional neural networks (CNNs) continue to break previous accuracy records with the advances in large-scale datasets [3, 4, 21, 28] and network architecture innovation [22, 24, 34, 39]. Naturally, the accuracy improvement comes at the cost of higher computational overhead [2, 7, 34]. Recently, there is a trend deploying CNNs in edge environments [31] to mitigate the latency and privacy concerns. However, the prohibitive computational cost impedes the deployment of advanced models onto resource-constrained edge devices. To enable edge applications like autopilot to benefit from the development of neural networks, efforts have been made to compress CNNs for a better trade-off between accuracy and model overhead, such that they can be easily deployed onto various edge hardware devices.

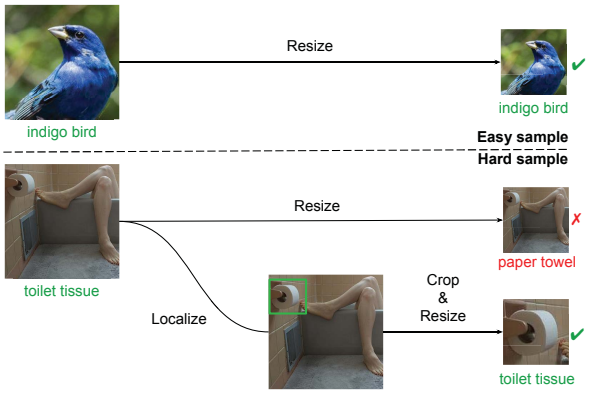

Figure 1: The prediction results of our pretrained ResNet-50 model. For easy samples, the network can still generate correct predictions at a smaller resolution (e.g. 112 × 112 for ImageNet). For hard samples, simply resizing the images to a smaller resolution can lead to misclassification, while the dynamic cropping strategy can correctly classify hard samples at a smaller resolution.

The computational cost, i.e., the Multiply-Accumulate Operations (MACs) of a CNN mainly result from gigantic network architectures and high-resolution input images (e.g., 224×224 for ImageNet). To reduce the cost, on the one hand, network architecture pruning [8, 19, 37, 41] devotes to removing the redundancy in network architectures to construct more compact networks. For instance, MobileNets [29] and Slimmable networks [41] yield 'thinner' networks by using less channels in each layer, which reduces the computation and memory footprint. On the other hand, as the MACs of a CNN reduce quadratically with respect to the image resolution, many works resize the input images to a smaller resolution (e.g., 112×112) to reduce the computational cost [29, 33, 47]. However, different images correspond to diverse classification difficulties. As demonstrated in Figure 1, easy samples with clear foreground can be correctly recognized even at a smaller resolution. While for hard samples, as the foreground object only occupies

a small portion of the whole image, directly shrinking the image will lose the details of the object, leading to a wrong prediction. In contrast, if we only crop the foreground patch for inference, hard samples can also be correctly classified at a small resolution. However, existing image preprocessing methods, e.g., ResizedCenterCrop (RCC), crop all images in a static manner and cannot achieve such instance-aware fine cropping, which motivates our work.

In this paper, we propose a dynamic image cropping (DIC) framework to facilitate the inference with low-resolution images, thereby reducing the costs of CNNs and boosting the deployment of CNNs onto resource-constrained edge devices. DIC first efficiently localizes the most discriminative foreground of the input image with a lightweight foreground predictor, then the detected foreground region will be preserved and the redundant background will be discarded. By this means, DIC is capable of generating fine-cropped images with less spatial redundancy, thereby improving the inference accuracy even under low-resolution settings. To train the foreground predictor on large-scale classification datasets, we first employ a visualization technique, Grad-CAM [30], to generate the salience map of each training image to indicate the most discriminative foreground. Based on the salience map, we then automatically generate a bounding box annotation for each training image. Subsequently, we use the image-box pairs to train the predictor in a supervised manner. Once the foreground predictor is trained, it can be directly applied to any CNN backbones without modification.

Meanwhile, to deal with the redundancy in network architectures for higher execution efficiency, we propose a compound shrinking (CS) strategy to jointly compress the three dimensions (depth, width, resolution) of a CNN. We first investigate the impact of shrinking each dimension on accuracy. Based on that, we then calculate the optimal shrinking coefficient for each dimension to coordinate the shrinking of different dimensions to achieve the highest compression rate while still maintaining the accuracy. Eventually, DIC and CS are seamlessly combined to form a deep compression framework, Smart Scissor, where an image is firstly cropped by DIC, then the cropped patch will be resized to the resolution calculated by CS and sent to the model compressed by CS for efficient inference. With our approach, both the spatial redundancy in images and the architecture redundancy in networks are reduced and thus the execution of CNNs on embedded hardware is accelerated.

The main contributions of this paper are summarized as follows:

(1) We propose a dynamic image cropping framework to reduce the spatial redundancy in images. We introduce a lightweight foreground predictor to efficiently localize the foreground object and conduct instance-aware dynamic cropping. With the finely cropped images, CNNs can yield accurate predictions using a smaller resolution, which greatly reduces the computational cost of CNNs.
(2) We also propose a compound shrinking strategy to coordinately shrink the three dimensions (depth, width, resolution) of a CNN. We first quantify the impact of each dimension on accuracy, and then compute the optimal shrinking coefficient for each dimension accordingly. By this means, we can greatly reduce the redundancy in network architectures while still maintaining high accuracy.
(3) We seamlessly combine the dynamic image cropping and compound shrinking into a deep compression framework, which is able to optimally adapt the model cost to meet different resource constraints of embedded hardware.

Experiments on ImageNet-1K demonstrate that our method reduces the MACs of ResNet50 by 41.5% while improving the top-1 accuracy by 0.3%. In addition, our method achieves 4.2% higher top-1 accuracy with similar MACs and 54.3% less parameters compared to the most widely used image cropping method, ResizedCenterCrop (RCC). In low MACs regime, our approach also outperforms HRank [19] by a remarkable improvement (4.1%) in top-1 accuracy.

2 RELATED WORK

Smart Scissor is mainly related to object discovery and neural network compression, so we discuss the related works in this section.

Object discovery: Object discovery algorithms have made impressive progress. Among them supervised object detection (SOD) [1, 10, 23, 27] achieves relatively high accuracy on well annotated datasets, such as COCO [21] and Pascal VOC [6]. However, the difficulties in building larger detection datasets hinder the further development of SOD. Weakly supervised object discovery (WSOD) [30, 38, 44, 46] is able to coarsely localize objects of interest with only image-level labels, which makes WSOD applicable to more large-scale datasets without bounding box annotations, such as ImageNet [4]. Class activation map (CAM) [46] is able to visualize the importance of different regions of the input image by utilizing the output of the pooling layer at the end of CNNs. Grad-CAM [30] further generalizes CAM by using the gradients to calculate the importance of different pixels, which makes it applicable to more CNN architectures. ACoL [45] designs a CNN architecture with two branches to adversarially learn the full region of objects. CutMix [42] randomly blends different regions of images to improve the localization accuracy. However, the huge computational cost and latency make both the SOD (e.g., SSD) and WSOD inapplicable in our context, i.e., resource-constrained embedded hardware.

CNN compression: To compress CNNs for better on-device performance, different approaches have been proposed to remove the redundancy of CNNs. Deep Compression [9] removes neurons from kernels, which achieves a high compression rate. However, the obtained sparse networks need special hardware to accelerate. [19, 25, 29, 41] build compact CNNs by removing channels from each layer, which can speed up the execution of CNNs on off-the-shelf hardware. Among them MobileNetV2 [29] and Slimmable networks [41] uniformly reduce the channels in all layers. Instead, HRank [19] and Taylor pruning [25] prune the channels in a layer-wise manner. Besides compressing the network architecture, some works focus on reducing the spatial redundancy in images. MobileNetV2 [29] and MnasNet [33] use smaller resolutions to improve the performance on mobile devices. DR-ResNet [47] proposes a dynamic resolution network to reduce the computational cost. GFNet [35] employs reinforcement learning to select multiple small patches instead of using the whole image to accelerate CNNs. However, these methods only consider reducing the redundancy in a single dimension and thus only achieve a limited compression rate. In contrast, jointly compressing all dimensions promises a better trade-off between the compression rate and accuracy.

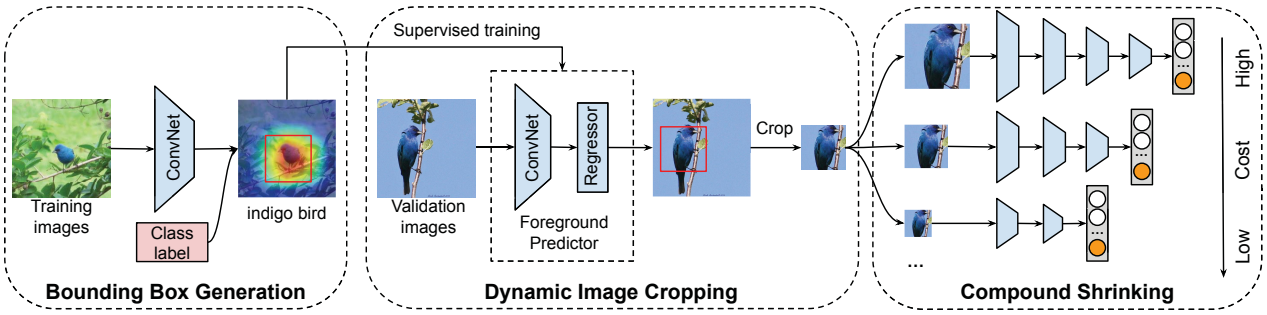

Figure 2: The overview of the proposed framework, Smart Scissor (SS), which mainly consists of three components: bounding box generation (BBG), dynamic image cropping (DIC), and compound shrinking (CS).

3 THE SMART SCISSOR FRAMEWORK

In this section, we first outline the design of Smart Scissor and then describe each submodule in detail.

As demonstrated in Figure 2, given a classification dataset $\mathcal{D}$ and a classification network $\mathcal{N}$. We first exploit Grad-CAM to generate the salience map of all training images in $\mathcal{D}$, and then we generate a bounding box for each image according to the salience map and form a bounding box label set $\mathcal{B}$. Thereafter, we utilize the image-box pairs $\{\mathcal{D}_i, \mathcal{B}_i\}$ to train a lightweight predictor. Meanwhile, for a given MACs budget $\mathcal{M}$, we use CS to quickly calculate the desired resolution and shrink the network accordingly. During inference, an image will be first fed into the predictor to quickly output the boundary of the foreground object, then the detected object is cropped. Finally, the cropped patch is resized to the calculated resolution and sent to the compressed network $\mathcal{N}_s$ for fast inference.

3.1 Bounding Box Generation

As aforementioned, dynamically cropping the foreground for inference is promising in reducing computation and improving accuracy. However, for classification datasets like ImageNet, there is no out-of-the-box position annotation for the foreground object. Moreover, the position of the foreground object varies in different images, which makes it difficult to efficiently localize the foreground object.

To address this limitation, we first use Grad-CAM [30] to automatically generate the position annotations. Specifically, let the class label of the given image be c. We first perform forward inference with a well-trained CNN (e.g. ResNet50) to obtain the prediction score p^c for class c, and then conduct backpropagation to compute the gradient of the score p^c with respect to each activation of the last convolutional layer. Thereafter, the gradients are aggregated within each channel via global average pooling. The obtained scalar for each channel can be seen as the weight of the channel, which can be calculated as follows:

$$a_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{pooling} \overbrace{\frac{\partial p^c}{A_{ij}^k}}^{gradients} \tag{1}$$

where a_k^c is the weight of channel k for class c, and A_{ij}^k is a single activation indexed by i and j in the 2-D feature map of channel k. With the weights of all channels determined, the salience map for class c can be obtained by computing the weighted sum of all feature maps over the channel dimension, which is formulated as:

$$L_{Grad\_CAM}^c = ReLU \underbrace{\left(\sum_k a_k^c A^k\right)}_{linear\ combination} \tag{2}$$

where A^k is the 2-D feature map of channel k, and ReLU is used to eliminate the impact of negative activations. Finally, the obtained salience map is upsampled to the same size as the input image via bi-linear interpolation algorithm.

With the salience map generated, we then introduce a simple yet effective strategy to determine the bounding box of the foreground object. Initially, we set the box as the boundary of the image. Subsequently, we shrink the four sides of the box simultaneously, and once a side reaches our preset salience threshold t, the side is frozen. The bounding box is determined after all sides are frozen. Note that it is crucial to appropriately select the value of t for the final result. As demonstrated in Figure 3, a too small threshold will result in residual background redundancy, while a too large threshold will lose some object features. Therefore, we conduct empirical experiments to determine the optimal threshold value. As shown in Table 1, we achieve the highest accuracy when the threshold t is set to 0.5. Therefore, we set $t = 0.5$ in the following experiments. Note that more fine-grained searching for t may further improve the accuracy, but it also increases the search cost. Finally, the generated box annotations are saved in the form of $[X_{min}, Y_{min}, X_{max}, Y_{max}]$, which denotes the position of the foreground in the image.

Table 1: The impact of using different salience thresholds on prediction accuracy. The model is trained and evaluated on ImageNet-1K. $t = 0$ means using the original images without Grad-CAM cropping.

Model	#Params	#MACs	t	Acc@1
ResNet50	25.6 M	4.1 B	0 (Baseline)	76.02 %
			0.25	76.45 %
			0.5	**76.88** %
			0.75	76.32 %

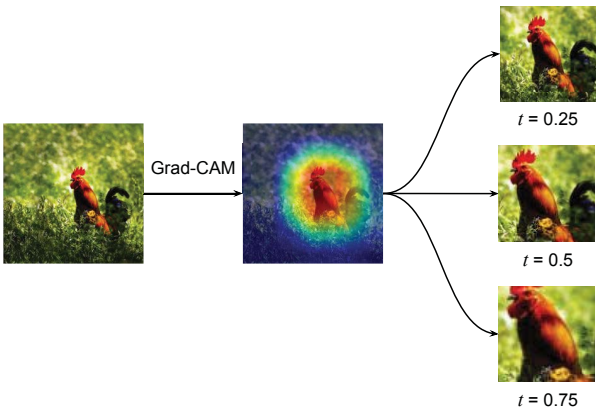

Figure 3: By applying different values of the salience threshold t, we can obtain different cropped images. The larger the threshold value, the more radical the cropping.

3.2 Dynamic Image Cropping

Figure 4 shows that we are capable of accurately localizing the foreground of images with Grad-CAM. However, Grad-CAM cannot be directly applied to edge applications because of the time-consuming backpropagation process. Moreover, Grad-CAM requires the class label as weak supervision, which is unavailable for validation images. To address these issues, we design a foreground predictor to efficiently localize the foreground of input images.

3.2.1 Predictor Architecture. Although dozens of object detectors have been proposed to detect objects from images, such as SSD [23] and Faster R-CNN [27], these detector architectures are completely inapplicable to our task. Particularly, to accurately detect multiple objects from a single image, existing detectors are usually equipped with a heavy feature extractor and a complex region proposal network (RPN), which are extremely time-consuming and unnecessary for image classification where each image only contains one object. Therefore, we discard the design of traditional detectors. Instead, we design the foreground predictor as a lightweight plain CNN, whose detailed architecture is summarized in Table 2. It consists of several residual bottleneck blocks [11] and a fully connected layer as the single-box regressor. As shown in Figure 5, a residual bottleneck contains two convolutional layers with 1×1 kernels and one convolutional layer with 3×3 kernels in the middle. The computational cost mainly results from the 3×3 convolutional layer. Therefore, to reduce the cost and accelerate the predictor, we only stack two residual bottleneck blocks in each stage and each block is only equipped with a small number of channels. Consequently, the proposed predictor only contains 0.27M parameters and 0.09B MACs, which is negligible compared to popular object detectors (e.g., Faster R-CNN with 134.7M (499×) parameters and 15.1B (167.8×) MACs [16]). Moreover, the small overhead of the foreground predictor can be complemented by the CS module.

3.2.2 Training of Foreground Predictor. We train the predictor in a supervised manner. First, we generate a bounding box label set $\mathcal{B}$ for all training images as described in Subsection 3.1, then the labels are utilized to train the predictor. We use the mean square error

Figure 4: The bounding boxes generated with the salience threshold $t = 0.5$, which accurately localize the key object in each image.

(MSE) as the loss function. Let $\mathcal{P}_i = [X^p_{min}, Y^p_{min}, X^p_{max}, Y^p_{max}]$ be the output of the predictor, and $\mathcal{B}_i = [X^g_{min}, Y^g_{min}, X^g_{max}, Y^g_{max}]$ be the generated box label, the loss function can be formulated as:

$$\begin{aligned}\mathcal{L}_{box} &= MSELoss(\mathcal{P}_i, \mathcal{B}_i) \\ &= \frac{1}{4}((X^g_{min} - X^p_{min})^2 + (Y^g_{min} - Y^p_{min})^2 \\ &\quad + (X^g_{max} - X^p_{max})^2 + (Y^g_{max} - Y^p_{max})^2)\end{aligned} \tag{3}$$

To balance the training overhead and prediction accuracy, we train the predictor with Adam [15] optimizer for 40 epochs. The initial learning rate is set to 1e-3, and the learning rate is scheduled using exponential decay [17]. The training of the box predictor is decoupled with backbone networks. Once the predictor is trained, it can be directly applied to different classification backbones without any training overhead. During inference, the trained predictor will quickly localize the foreground object of the input image and generate a finely cropped image, which significantly reduces the redundancy in the input image.

3.3 Compound Shrinking

The proposed DIC significantly reduces the redundancy in images, improving the computational efficiency. We observe that redundancy also exists in network architectures (e.g., redundant parameters), and only removing the redundancy in images loses the opportunity to further compress the model for embedded hardware. Besides, [34] demonstrates that jointly adjusting different dimensions promises higher accuracy. To this end, we propose a compound shrinking (CS) strategy to jointly compress the three dimensions (depth, width, resolution) of CNNs to reduce the redundancy in images as well as networks while maintaining the accuracy.

Intuitively, shrinking different dimensions has different impacts on accuracy and model overhead. The core of our compound shrinking strategy is to calculate a shrinking coefficient for each dimension according to their trade-off between accuracy and model overhead. A larger coefficient denotes more radical shrinking. More specifically, the dimension with a steep accuracy drop during shrinking will be assigned a small shrinking coefficient to prevent severe accuracy degradation. To calculate the shrinking coefficients, we first quantify the trade-off of each dimension between accuracy

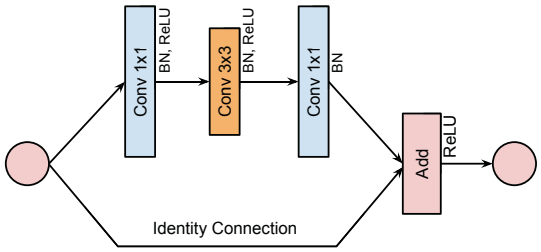

Figure 5: The architecture of the residual bottleneck block, which is widely used in many state-of-the-art networks.

Table 2: The architecture of the proposed box predictor. #C denotes the number of channels and #L denotes the number of layers.

Stage	Block	Resolution	#C	#L
1	Conv 3×3	224 × 224	16	1
2	Residual Bottleneck	112 × 112	16	2
3	Residual Bottleneck	56 × 56	32	2
4	Residual Bottleneck	28 × 28	32	2
5	Residual Bottleneck	14 × 14	64	2
6	Pooling & Linear	7 × 7	4	1
#Params: 0.27M #MACs: 0.09B				

and model overhead. Here we use MACs as the metric to measure the cost of models, because all three dimensions are related to the MACs of a model while only the depth and width can affect the model parameters. Given a MACs budget $\mathcal{M}$, we first obtain the accuracy drops resulting from separately shrinking different dimensions, which can be represented as:

$$\Delta A_s(\mathcal{M}) = A_0 - A_s(\mathcal{M}) \tag{4}$$

where $s \in \{d, w, r\}$ represents the shrunk dimension, $A_s(\mathcal{M})$ denotes the accuracy of the shrunk model, and A_0 denotes the accuracy of the original model. To comply with the rule that the steeper the drop in accuracy, the smaller the coefficient of the corresponding dimension, we design the following equation to determine the shrinking coefficient for each dimension:

$$C_s(\mathcal{M}) = \frac{\sqrt[3]{\Delta A_d(\mathcal{M}) \cdot \Delta A_w(\mathcal{M}) \cdot \Delta A_r(\mathcal{M})}}{\Delta A_s(\mathcal{M})} \tag{5}$$

where $C_s(\mathcal{M})$ denotes the shrinking coefficient of the dimension s ($s \in \{d, w, r\}$). Through Equation 4 and Equation 5, we are able to efficiently calculate the coefficients once we obtain the accuracy degradation of the three dimensions in the given MACs regime.

However, the training cost of the compressed models to calculate the accuracy drop is still non-negligible. To mitigate the training overhead, we propose a dimension-wise accuracy estimator to quickly estimate the accuracy of the compressed models and calculate the accuracy degradation resulting from shrinking different dimensions in the given MACs regime. First, we sample a couple of models with different MACs by separately shrinking the

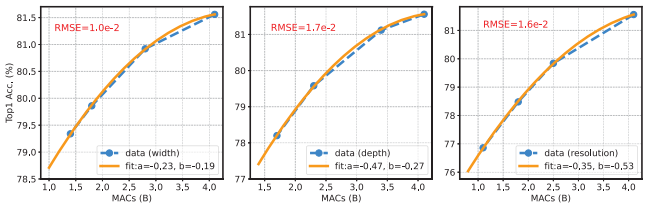

Figure 6: The actual accuracy (blue dotted line) and the estimated accuracy (yellow line) over MACs by separately shrinking the three dimensions. The low root mean square error (RMSE) indicates that the accuracy estimator can well fit existing data.

three dimensions. As demonstrated in Figure 6, the accuracy distribution of the three dimensions along MACs can be well fitted by a quadratic polynomial. Therefore, we design a simple yet effective polynomial estimator to predict the accuracy with respect to the target MACs $\mathcal{M}$. The estimator can be formulated as follows:

$$A_s(\mathcal{M}) = a_s(\mathcal{M} - \mathcal{M}_0)^2 + b_s(\mathcal{M} - \mathcal{M}_0) + A_0 \tag{6}$$

where $\mathcal{M}_0$ is the MACs of the original model. a_s and b_s are the hyperparameters to fit for dimension s ($s \in \{d, w, r\}$). Subsequently, we train the dimension-wise estimator using least square regression with the aforementioned sampled data. Figure 6 shows that the proposed estimator can well fit existing data. Due to the simple and intuitive design of the estimator, we only need to sample and train very few models to train the estimator, and this cost is a one-time cost. With the accuracy estimator established, we are capable of directly calculating the accuracy drop and subsequently the shrinking coefficient of each dimension across a wide range of MACs regimes. Consequently, the cost of assigning the coefficients is significantly reduced compared to directly training models to obtain the coefficients.

4 EXPERIMENTAL RESULTS

In this section, we conduct experiments to comprehensively evaluate Smart Scissor. We compare Smart Scissor with many state-of-the-art (SOTA) methods in terms of accuracy, MACs, on-device latency, and throughput. In addition, we also do ablation experiments to validate the efficacy of each component in our framework.

4.1 Hardware Devices

The on-device latency and throughput are two important metrics to evaluate the efficiency of CNNs. To demonstrate the on-device performance of our method, we select two representative embedded GPU platforms, NVIDIA AGX Xavier and NVIDIA Jeton Nano, and Intel i7-9750H@2.6GHz CPU to deploy the proposed framework and measure the latency and throughput. The hardware specifications are listed in Table 3.

4.2 Datasets

We evaluate our method on two large-scale datasets, ImageNet-1K [4] and ImageNet-100. ImageNet-1K (a.k.a. ILSVRC 2012) is the most widely exploited dataset for image classification tasks, which consists of 1,000 classes, and each class contains about 1,000

Table 3: The specifications of three different hardware platforms used in our experiments.

Device	Power	Memory/Cache	Performance
AGX Xavier	15 W	32 GB	11 TOPS
Jetson Nano	5 W	4 GB	0.47 TFLOPS
i7-9750H	45 W	12 MB	0.4 TOPS

high-resolution (224×224) images for training and 50 images for validation. ImageNet-100 is a subset of ImageNet-1K, which is composed by 100 randomly selected classes from ImageNet-1K. The detailed categories of ImageNet-100 is provided in the code repository. All images are preprocessed following a simple configuration as [26, 29].

4.3 Networks

We implement the Smart Scissor framework with two popular CNN models, ResNet50 [11] and RegNet-X [26] as the baseline backbone networks. For each model, we apply Smart Scissor to reduce the model overhead by removing both the spatial redundancy of images and the architecture redundancy of networks. Moreover, we also report the results of other SOTA model compression approaches on the two models for comparison.

4.4 Optimization Settings

We use a stochastic gradient descent (SGD) optimizer with a momentum of 0.9 to train the classification networks. On both ImageNet-100 and Imagenet-1K, models are first trained with a batch size of 1,024 for 100 epochs without dynamic image cropping, where the first 5 epochs are for warmup. At this stage, the initial learning rate is set to 2.0 and decayed by exponential learning rate policy with a decay factor of 0.02. Then, we use the proposed dynamic image cropping to fine-tune the pretrained models for 20 epochs with a constant learning rate of 5e-4. To prevent over-fitting, we also use label smoothing with $\epsilon = 0.1$.

4.5 Results on ImageNet-100

We conduct experiments on ImageNet-100 with both ResNet50 and RegNet-X. In practise, to better demonstrate the improvement of each component, we implement two versions of Smart Scissor, Smart Scissor with only the DIC module (SS-DIC) and Smart Scissor with both the DIC and the CS module (SS-DIC-CS). As a comparison, we use the most popular image cropping method, Resized-CenterCrop (RCC) to crop and resize images. Finally, all models are deployed to the hardware platforms to evaluate the latency and throughput.

ResNet50: As shown in Table 4, SS-DIC-CS outperforms the competitors on all metrics. Specifically, compared to the baseline ResNet50 (RCC-Baseline), SS-DIC improves the accuracy by 0.9% with a negligible increase in model parameters (1.2%) and MACs (2.4%), while SS-DIC-CS further pushes up the accuracy improvement to 1.1% with a parameter reduction of 27% and a MACs reduction of 26.8%. In the low complexity regime, SS-DIC-CS achieves 2.4% higher accuracy than RCC with only 33% model parameters

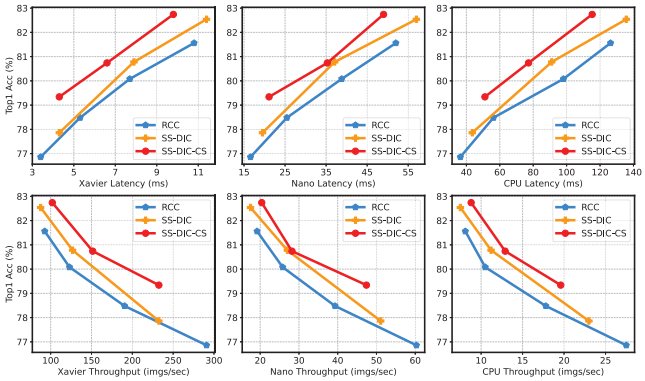

Figure 7: The real performance of ResNet50 compressed by different methods on three distinct hardware devices. The accuracy is measured on ImageNet-100.

Table 4: Results of ResNet50 on ImageNet-100.

Method	#Params	#MACs	↓ MACs	Acc@1
RCC-Baseline	23.7 M	4.1 B	0.0%	81.6%
SS-DIC	24.0 M	4.2 B	-2.4%	82.5%
SS-DIC-CS	**17.3 M**	**3.0 B**	**26.8%**	**82.7%**
RCC	23.7 M	3.0 B	26.8%	80.1%
SS-DIC	24.0 M	2.6 B	36.6%	80.8%
SS-DIC-CS	**14.5 M**	**2.4 B**	**41.5%**	**81.5%**
RCC	23.7 M	1.1 B	73.2%	76.9%
SS-DIC	24.0 M	1.2 B	70.7%	77.9%
SS-DIC-CS	**7.8 M**	**1.0 B**	**75.6%**	**79.3%**

(7.8 M v.s. 23.7 M) and less MACs. Meanwhile, Figure 7 indicates that both SS-DIC and SS-DIC-CS outperform RCC by a large margin across a wide spectrum in terms of latency and throughput. Particularly, SS-DIC-CS achieves 79.4% top-1 accuracy with a latency of 4.3 ms on Xavier, which is 0.9% higher in accuracy and 18.9% lower in latency compared to RCC (78.5% top1 accuracy, 5.3 ms). At the same time, the throughput of SS-DIC-CS on Xavier is 232.5 *imgs/sec*, which is 22% higher than RCC (190.5 *imgs/sec*). On Nano and Intel i7-9750H CPU, SS-DIC-CS also improves the throughput by 20.6% and 10.1%, and reduces the latency by 17% and 9.1%, respectively.

RegNet-X: The experimental results of RegNet-X are summarized in Table 5, where we also observe a significant improvement of our method. SS-DIC-CS outperforms the baseline RegNet-X (RCC-Baseline) with an improvement of 1.3% in accuracy and a reduction of 18.8% in MACs. Meanwhile, SS-DIC-CS reduces the model parameters by 25% (6.3 M v.s. 8.4 M). In the low MACs regime, SS-DIC-CS observes a remarkable 2.8% improvement in accuracy with only 30% parameters (2.5 M v.s. 8.4 M) compared to RCC. As for the real performance on hardware, SS-DIC-CS obtains an accuracy of 84% with 3.6 ms latency on Xavier, which is 0.3% higher in accuracy and 30.8% lower in latency than RCC (83.7% top-1 accuracy, 5.2 ms). Similarly, the latency reductions on Nano and Intel CPU are 31% and 34.6%, respectively. Besides, SS-DIC-CS observes a

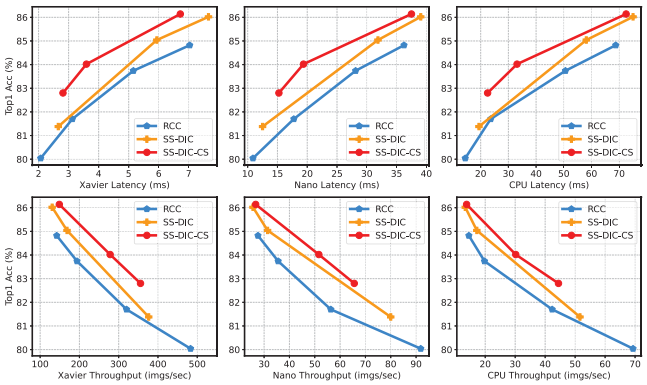

Figure 8: The real performance of RegNet-X compressed by different methods on three distinct hardware devices. The accuracy is measured on ImageNet-100.

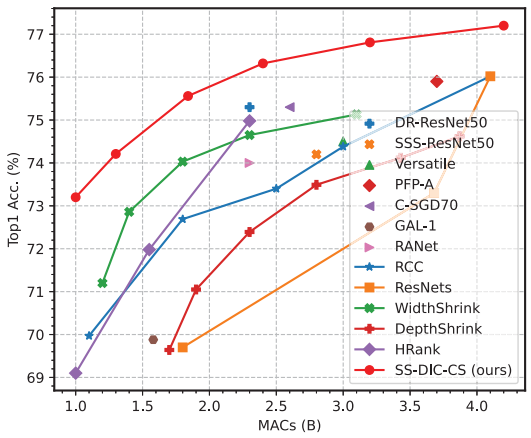

Figure 9: Comparison of our Smart Scissor with other state-of-the-art model compression methods. The baseline model is ResNet50 and the dataset is ImageNet-1K.

Table 5: Results of RegNet-X on ImageNet-100.

Method	#Params	#MACs	↓ MACs	Acc@1
RCC-Baseline	8.4 M	1.6 B	0.0%	84.8%
SS-DIC	8.7 M	1.3 B	18.8%	85.0%
SS-DIC-CS	**6.3 M**	**1.3 B**	**18.8%**	**86.1%**
RCC	8.4 M	0.7 B	56.3%	82.3%
SS-DIC	8.7 M	0.8 B	50.0%	83.8%
SS-DIC-CS	**3.6 M**	**0.6 B**	**62.5%**	**84.0%**
RCC	8.4 M	0.4 B	75.0%	80.0%
SS-DIC	8.7 M	0.5 B	68.8%	81.4%
SS-DIC-CS	**2.5 M**	**0.4 B**	**75.0%**	**82.8%**

44.9% throughput improvement (51.6 *imgs/sec* v.s. 35.6 *imgs/sec*) on Nano and a 52.5% throughput improvement (30.2 *imgs/sec* v.s. 19.8 *imgs/sec*) on Intel CPU compared to RCC.

4.6 Results on ImageNet-1K

We evaluate our approach on ImageNet-1K, and compare the evaluation results with many SOTA CNN compression frameworks. In addition, we also compare our compressed models with many popular backbone architectures in different computation regimes.

Comparison with SOTA compression methods: Starting with the baseline ResNet50 model, we implement different model shrinking methods, including RCC, width shrinking (WidthShrink) [29, 43], depth shrinking (DepthShrink) [11], SS-DIC, and SS-DIC-CS, to compress the three dimensions of the model to different MACs regimes and compare their performance. In addition, we also report the performance of multiple SOTA model compression techniques from the related papers, including DR-ResNet50 [47], SSS-ResNet50 [13], Versatile-ResNet50 [36], PFP-A-ResNet50 [18], C-SGD70-ResNet50 [5], GAL-1 [20], HRank [19], and RANet [40]. The comparison results are summarized in Figure 9, which shows that our method achieves the highest accuracy across a wide range of MACs. Particularly, compared to the baseline ResNet50, our method achieves 1.2% accuracy improvement (76.0% to 77.2%) with a negligible increase in MACs (4.1 B to 4.2 B). As we continue to reduce the MACs budget, SS-DIC-CS reduces the MACs by 41.5% (4.1 B to 2.4 B) while still achieving 0.3% higher accuracy (76.0% to 76.3%). In the low MACs regime, SS-DIC-CS remarkably improves the accuracy by 4.2% (70.0% to 74.2%) compared to RCC with similar MACs. In comparison with other SOTA compression methods, our method also achieves the best trade-off between MACs and accuracy. For example, SS-DIC-CS achieves 3.3% higher accuracy (73.2% v.s. 69.9%) than GAL-1 [20] with 37.5% less MACs (1.0B v.s. 1.6B).

Comparison with popular backbones: In this experiment, we apply Smart Scissor to ResNet50 and compare it with other models from the ResNet family, such as ResNet101 and ResNet34, etc. In addition, we also conduct extensive comparison with other popular backbones like DenseNets [12] and the Inception family [14, 32]. As demonstrated in Table 6, in the highest MACs regime, SS-DIC only uses 53.2% of the MACs (4.2 B v.s. 7.9 B) of DenseNet161 to achieve the same level of accuracy, while in the lowest MACs regime, SS-DIC-CS obtains the highest top-1 accuracy (75.6%), which is 5.8% and 2.1% higher than ResNet18 (69.8%) and BN-Inception (73.5%), respectively. The comparison results with other backbones reveal that our method can achieve promising results without redesigning the network architecture, which avoids the extremely time-consuming exploration of the architecture design space.

4.7 Visualization

We visualize the bounding boxes generated from both Grad-CAM and our predictor in Figure 10. We can see that the foreground of the most of images only occupies part of the whole images, thus performing inference on the whole image is unnecessary and inefficient, which coincides with our motivation. Moreover, Figure 10 validates that, even though the foreground predictor only contains very limited computation and parameters for higher execution efficiency, it can also accurately localize the main object. With the accurate and quick prediction of the foreground, we are able to remove the background redundancy in images, thereby accelerating the execution of CNNs on resource-constrained edge devices.

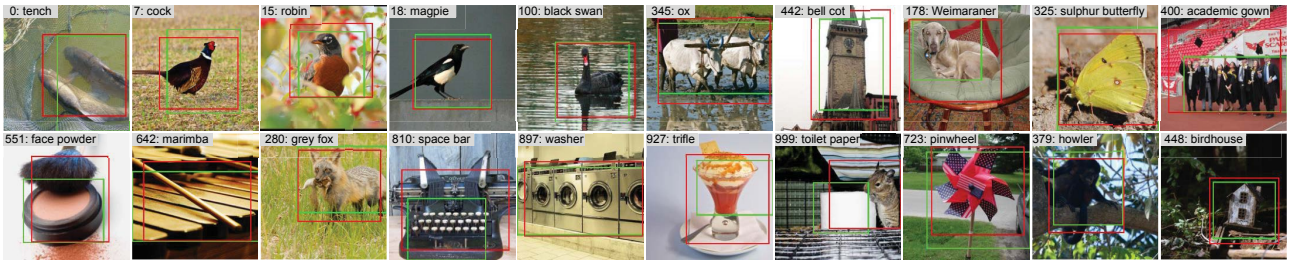

Figure 10: Visualization of the predicted bounding boxes (red) and the ground truth bounding boxes generated from Grad-CAM (green). Our predictor achieves a high localization accuracy of 62% mAP on ImageNet-1K validation set. The images above are randomly selected from ImageNet-1K.

Table 6: Comparison with other popular backbone networks.

Model	#Params	#MACs	↓ MACs	Acc@1
ResNet50[11]	25.6 M	4.1 B	0.0%	76.0%
ResNet101[11]	44.6 M	7.9 B	-92.7%	**77.4%**
DenseNet161[12]	28.7 M	7.9 B	-92.7%	77.1%
InceptionV3[32]	27.2 M	5.8 B	-41.5%	77.3%
SS-DIC	**25.9 M**	**4.2 B**	**-2.4%**	77.2%
ResNet34[11]	21.8 M	3.7 B	9.8%	73.3%
DenseNet169[12]	**14.2 M**	3.4 B	17.1%	75.6%
SS-DIC	25.9 M	3.1 B	24.4%	76.3%
SS-DIC-CS	15.4 M	**2.4 B**	**41.5%**	**76.3** %
ResNet18[11]	11.7 M	1.8 B	56.1%	69.8%
DenseNet121[12]	**8.0 M**	2.9 B	29.3%	74.6%
BN-Inception[14]	11.2 M	2.1 B	48.8%	73.5%
SS-DIC	25.9 M	1.9 B	53.7%	74.9%
SS-DIC-CS	13.5 M	**1.8 B**	**56.1%**	**75.6%**

Table 7: Results of ablation experiments on ImageNet-1K.

Method	#Params	#MACs	Latency	Acc@1
ResNet50	25.6 M	4.1 B	10.6 ms	76.0%
RCC	25.6 M	1.1 B	3.0 ms	70.0%
RandomCrop	25.6 M	1.1 B	2.9 ms	69.1%
SS-DIC	25.9 M	1.2 B	3.8 ms	**73.1%**
ResShrink	25.6 M	1.1 B	3.0 ms	70.0%
WidthShrink	8.9 M	1.2 B	5.2 ms	71.2%
DepthShrink	11.1 M	1.7 B	5.0 ms	69.6%
SS-CS	11.4 M	1.1 B	4.6 ms	**71.5%**
SS-DIC-CS	11.7 M	1.3 B	5.5 ms	**74.2%**

4.8 Ablation Study

To validate the efficacy of both the DIC module and the CS module, we perform comprehensive ablation experiments on ImageNet-1K. First, we only keep the DIC module to separately evaluate its efficacy. The results in Table 7 show that SS-DIC outperforms RCC and RandomCrop remarkably. Specifically, SS-DIC improves the top-1 accuracy by 3.1% compared to RCC with a similar model cost. Subsequently, we evaluate the CS component by comparing it with single-dimension shrinking methods. The results also demonstrate the efficacy of the CS component. Finally, we combine the two modules into the completed framework SS-DIC-CS. Because of the novel design of DIC and CS, we significantly improve the accuracy with a slightly higher latency on Xavier, optimizing the trade-off between accuracy and latency. The ablation experiments reveal that both DIC and CS contribute to the final performance.

5 CONCLUSION

In this paper, we propose a deep compression framework, Smart Scissor, to comprehensively reduce the redundancy in both input images and network architectures, thereby facilitating the efficient inference on edge devices. First, we rethink the image cropping strategy for CNNs, and we find that existing static cropping methods introduce much redundancy in images. To reduce the redundancy in images, we propose a dynamic image cropping framework. The proposed framework first employs a foreground predictor with negligible overhead to quickly localize the important foreground region, and only the detected foreground will be utilized for inference, which remarkably reduces the computational cost. Meanwhile, to remove the redundancy in network architectures, we also propose a compound shrinking strategy, which coordinately shrinks the three dimensions of CNNs according to their importance to the final accuracy. By this means, the compound shrinking strategy is expected to comprehensively compress CNNs. Finally, the dynamic image cropping and compound shrinking are integrated to work coordinately for the optimal trade-off between accuracy and costs. Extensive and comprehensive experiments demonstrate the efficacy and efficiency of our approach.

6 ACKNOWLEDGEMENT

This study is partially supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner, HP Inc., through the HP-NTU Digital Manufacturing Corporate Lab (I1801E0028). This work is also partially supported by Nanyang Technological University, Singapore, under its NAP (M4082282).

REFERENCES

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934* (2020).

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models Are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, Vol. 33. 1877–1901.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3213–3223.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 248–255.

[5] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. 2019. Centripetal SGD for Pruning Very Deep Convolutional Networks with Complicated Structure. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4943–4953.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88, 2 (June 2010), 303–338.

[7] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research* 23, 120 (2022), 1–39.

[8] Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. 2021. Network Pruning via Performance Maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9270–9280.

[9] Song Han, Huizi Mao, and William J Dally. 2015. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv preprint arXiv:1510.00149* (2015).

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2961–2969.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 770–778.

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely Connected Convolutional Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4700–4708.

[13] Zehao Huang and Naiyan Wang. 2018. Data-Driven Sparse Structure Selection for Deep Neural Networks. In *Proceedings of the European Conference on Computer Vision*. 304–320.

[14] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*. 448–456.

[15] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[16] Yuxi Li, Jiuwei Li, Weiyao Lin, and Jianguo Li. 2018. Tiny-DSOD: Lightweight Object Detection for Resource-Restricted Usages. In *British Machine Vision Conference 2018*.

[17] Zhiyuan Li and Sanjeev Arora. 2020. An Exponential Learning Rate Schedule for Deep Learning. In *International Conference on Learning Representations*.

[18] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. 2019. Provable Filter Pruning for Efficient Neural Networks. In *International Conference on Learning Representations*.

[19] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. 2020. HRank: Filter Pruning Using High-Rank Feature Map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1529–1538.

[20] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. 2019. Towards Optimal Structured CNN Pruning via Generative Adversarial Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2790–2799.

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*. 740–755.

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. SSD: Single Shot Multibox Detector. In *European Conference on Computer Vision*. 21–37.

[24] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A ConvNet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[25] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance Estimation for Neural Network Pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11264–11272.

[26] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Designing Network Design Spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10428–10436.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*, Vol. 28.

[28] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. 2021. ImageNet-21K Pretraining for the Masses. In *Advances in Neural Information Processing Systems*, Vol. 34.

[29] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetv2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4510–4520.

[30] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 618–626.

[31] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal* 3 (2016), 637–646.

[32] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking The Inception Architecture for Computer Vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2818–2826.

[33] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2820–2828.

[34] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*. 6105–6114.

[35] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. 2020. Glance and Focus: A Dynamic Approach to Reducing Spatial Redundancy in Image Classification. In *Advances in Neural Information Processing Systems*, Vol. 33. 2432–2444.

[36] Yunhe Wang, Chang Xu, Chunjing Xu, Chao Xu, and Dacheng Tao. 2018. Learning Versatile Filters for Efficient Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 31.

[37] Zi Wang, Chengcheng Li, and Xiangyang Wang. 2021. Convolutional Neural Network Pruning with Structural Redundancy Reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14913–14922.

[38] Xiu-Shen Wei, Chen-Lin Zhang, Jianxin Wu, Chunhua Shen, and Zhi-Hua Zhou. 2019. Unsupervised Object Discovery and Co-Localization by Deep Descriptor Transformation. *Pattern Recognition* 88 (2019), 113–126.

[39] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. FBnet: Hardware-Aware Efficient Convnet Design via Differentiable Neural Architecture Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10734–10742.

[40] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. 2020. Resolution Adaptive Networks for Efficient Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2369–2378.

[41] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2019. Slimmable Neural Networks. In *International Conference on Learning Representations*.

[42] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6023–6032.

[43] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. In *British Machine Vision Conference 2016*.

[44] Chen-Lin Zhang, Yun-Hao Cao, and Jianxin Wu. 2020. Rethinking the Route Towards Weakly Supervised Object Localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13460–13469.

[45] Xiaolin Zhang, Yunchao Wei, Jiashi Feng, Yi Yang, and Thomas S Huang. 2018. Adversarial Complementary Learning for Weakly Supervised Object Localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1325–1334.

[46] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning Deep Features for Discriminative Localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2921–2929.

[47] Mingjian Zhu, Kai Han, Enhua Wu, Qiulin Zhang, Ying Nie, Zhenzhong Lan, and Yunhe Wang. 2021. Dynamic Resolution Network. In *Advances in Neural Information Processing Systems*, Vol. 34.