Spatially Adaptive Feature Refinement for Efficient Inference

Yizeng Han, Gao Huang, *Member, IEEE*, Shiji Song, *Senior Member, IEEE*, Le Yang, Yitian Zhang, and Haojun Jiang

***Abstract*— Spatial redundancy commonly exists in the learned representations of convolutional neural networks (CNNs), leading to unnecessary computation on high-resolution features. In this paper, we propose a novel *Spatially Adaptive feature Refinement* (SAR) approach to reduce such superfluous computation. It performs efficient inference by adaptively fusing information from two branches: one conducts standard convolution on input features at a *lower spatial resolution*, and the other one *selectively refines* a set of regions at the original resolution. The two branches complement each other in feature learning, and both of them evoke much less computation than standard convolution. SAR is a flexible method that can be conveniently plugged into existing CNNs to establish models with reduced spatial redundancy. Experiments on CIFAR and ImageNet classification, COCO object detection and PASCAL VOC semantic segmentation tasks validate that the proposed SAR can consistently improve the network performance and efficiency. Notably, our results show that SAR only refines less than 40% of the regions in the feature representations of a ResNet for 97% of the samples in the validation set of ImageNet to achieve comparable accuracy with the original model, revealing the high computational redundancy in the spatial dimension of CNNs.**

***Index Terms*— Dynamic networks, spatially adaptive inference, convolutional neural networks.**

I. Introduction

ALTHOUGH CNNs have achieved significant improvements in accuracy on many computer vision tasks [1]–[6], their high computational demand still hinders the employment and application of deep models on resource-constrained platforms. Recent work has focused on improving the inference efficiency of CNNs for better applicability. Popular approaches include designing lightweight network architectures [7]–[9], pruning redundant components [10]–[12] and performing adaptive inference [11], [13]–[15]. Apart from exploiting the redundancy in model *structure*, reducing the *spatial* redundancy in features has also become a recent research trend [16]–[20].

Existing literature typically explores to reduce the spatial redundancy in CNNs from two perspectives. The first prevalent solution is sampling salient pixels in feature maps to perform sparse computation [18], [21]–[23]. The other line of work focuses on building static [16], [17], [24] or dynamic [19] models with multi-scale feature representations. Although being shown effective, the aforementioned methods still have their limitations. The spatially dynamic convolution realized in [22], [23], [25], [26] neglect the unselected regions during inference, resulting in inevitable loss of global information and degraded representation power. Moreover, pixel-level sparse convolution might raise challenges for practical efficiency on hardware [18]. Despite that the efficiency of CNNs can be increased via building multi-scale structures [16], [17], [19], [24], uniform computation is allocated on different spatial locations, leading to intrinsic redundancy. In all the previous approaches, the decision of whether a pixel should be calculated is made equally for all the channels at the corresponding spatial position, limiting the flexibility of the inference procedure.

In this paper, we propose a novel *Spatially Adaptive feature Refinement* (SAR) method, which effectively addresses the aforementioned issues with a two-branch structure (see Fig. 1 for an overview): a *base branch* which generates coarse-grained features with a lower resolution, and a *refinement branch* which dynamically selects a series of patches for feature refinement with the original resolution. The output is obtained by fusing the representations from two branches. The proposed approach is efficient and effective because: 1) processing features with low resolution enables the base branch to efficiently learn the high-level representations; 2) the refinement branch selectively performs convolution only on a subset of informative high-resolution regions, and the patch-level dynamic computation could be implemented with efficient operators for practical speedup on hardware. Furthermore, considering the same spatial location can be of different importance for different channels, the refinement branch is split into multiple sub-branches (groups) along the channel dimension, each of which selects and refines an independent set of regions, resulting in improved flexibility and effectiveness of feature refinement.

To identify the salient regions for refinement, the proposed SAR exploits a region selection module to produce a

Manuscript received April 1, 2021; revised August 30, 2021 and October 21, 2021; accepted October 24, 2021. Date of publication November 9, 2021; date of current version November 15, 2021. This work was supported in part by the National Science and Technology Major Project of the Ministry of Science and Technology of China under Grant 2018AAA0100701, in part by the National Natural Science Foundation of China under Grant 61906106 and Grant 62022048, and in part by the Beijing Academy of Artificial Intelligence. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Peihua Li. *(Corresponding author: Shiji Song.)*

Yizeng Han, Shiji Song, and Haojun Jiang are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: hanyz18@mails.tsinghua.edu.cn; shijis@tsinghua.edu.cn; jhj20@mails.tsinghua.edu.cn).

Gao Huang is with the Department of Automation, Tsinghua University, Beijing 100084, China, and also with Beijing Academy of Artificial Intelligence, Beijing, 100084, China (e-mail:gaohuang@tsinghua.edu.cn).

Le Yang is with the School of Information and Communications Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: yangle15@xjtu.edu.cn).

Yitian Zhang is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: markcheung9248@gmail.com).

Digital Object Identifier 10.1109/TIP.2021.3125263

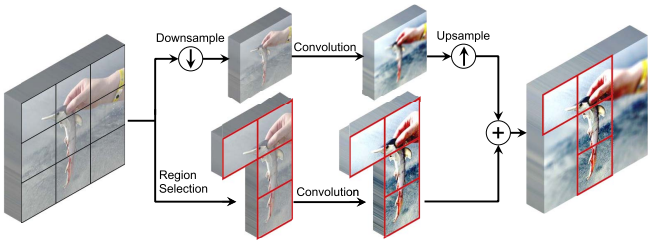


Fig. 1. An illustration of the proposed *Spatially Adaptive feature Refinement* (SAR) method. The base (top) branch processes features at a lower resolution. The refinement (bottom) branch only refines the adaptively selected (framed in red) regions at the original resolution to complement the insufficient local details in low-resolution representations.

binary-valued mask. Each element of the mask determines whether to refine the corresponding spatial location on the high-resolution feature in the refinement branch. The module is trained to adaptively select regions conditioned on the low-resolution representations generated by the base branch. Due to the inability to propagate gradients through the modules making discrete decisions, Gumbel Softmax [27], [28] is adopted to perform differentiable approximations for discrete stochastic nodes and enable training SAR in an end-to-end fashion.

SAR is generic and can be plugged into popular CNN architectures, establishing dynamic models with reduced spatial redundancy. Extensive evaluations on challenging computer vision benchmarks demonstrate that SAR consistently increases the inference efficiency of popular CNNs, including ResNet [2], ResNeXt [29], WideResNet (WRN) [30] and MobileNet-v2 [8]. For example, our method reduces the computation of ResNeXt29-8×32 and WRN29×10 by 38% and 62% respectively with even higher accuracy on CIFAR-100. Moreover, the computation of ResNet50 can be reduced by 41% while achieving 1.0% higher Top-1 accuracy on ImageNet. Experiments on downstream tasks, such as COCO object detection and PASCAL VOC semantic segmentation further validate the notable effectiveness of SAR. In particular, the network performance and efficiency can be increased simultaneously.

II. Related Work

Improving the efficiency of CNNs is attracting great research interest in recent years. Lightweight models have been built by adopting depth-wise convolutions [7], [8] or sparsely connected group convolutions [9], [31]–[33]. Meanwhile, neural architecture search (NAS) [34]–[37] and network pruning methods [10], [12], [38] are other lines of work for improving CNN efficiency. Apart from static models, dynamic networks are widely studied due to their favorable properties such as efficiency and adaptiveness [15]. Prevalent approaches to building dynamic models include conditionally skipping the execution of network layers [39]–[41] or convolutional channels [11], [42]. Besides, networks with auxiliary predictors are designed to allow early exiting "easy" examples [13], [19], [43]. However, few researchers consider the redundancy on the spatial dimension of features. Recent work has shown that the efficiency of CNNs can be significantly improved by allocating adaptive amount of computation to different pixels [18], [21] or exploiting dynamic feature resolutions [19], and our proposed method comprehensively reduces the spatial redundancy from the perspectives of both feature resolution and spatial-wise adaptive inference.

Spatially adaptive inference is realized to perform convolution only on sparsely sampled areas, where the sparse areas can be obtained from prior knowledge [25] or predicted by the networks [20], [22], [23], [26], [44]. However, none of these methods has sought to effectively learn coarse-grained feature representations for the unselected regions, and usually trade the network performance for efficiency. Instead of using naive interpolation as in [18], our method adaptively fills the unselected areas with features obtained by convolution on *low-resolution* representations, and thus is able to provide more sufficient global information with cheap computation. Moreover, in contrast to the aforementioned approaches which generally make a uniform decision for all the channels at a spatial location, the proposed SAR is able to conduct refinement on independent regions for different *channel groups*, leading to increased flexibility of spatially adaptive inference. In terms of practical efficiency, our *patch-level* adaptive computation could be implemented with efficient operators, making it easier to gain realistic speedup than the existing *pixel-level* approaches.

Multi-scale representation plays an important role in many vision tasks such as image classification [13], [24], object detection [4], [45] and pose estimation [46]. Researchers also utilize multi-scale representations for building efficient networks. Representatively, big-little network [16] and octave convolution [17] are *static* multi-branch structures with reduced spatial redundancy. Resolution adaptive network [19] conditionally increases the feature resolution and allows dynamic early exiting. However, these models treat different spatial locations with uniform computation, leading to intrinsic computational redundancy on unimportant regions. Instead, our method *dynamically* refines a subset of areas on the high-resolution features, resulting in improved adaptiveness and efficiency.

III. Method

Existing solutions to reducing the spatial redundancy in CNNs all have their limitations: 1) previous spatially adaptive inference approaches [18], [22], [23] typically result in degraded performance due to the lost information on unselected areas, and pixel-level sparse convolution might raise challenges for realistic efficiency; 2) conventional multi-scale structures [16], [17] inevitably induce redundant computation on less informative regions; 3) none of these methods has considered the divergence of region saliency among different channels.

To this end, our *Spatially Adaptive feature Refinement* (SAR) is proposed to reduce the spatial redundancy of CNNs in a more effective way: 1) coarse-grained representations are generated by processing low-resolution features to guarantee the learning of global information; 2) important regions with high resolution are strategically sampled and refined to produce task-relevant local information; 3) the content-aware sampling strategy allows the model to refine different locations

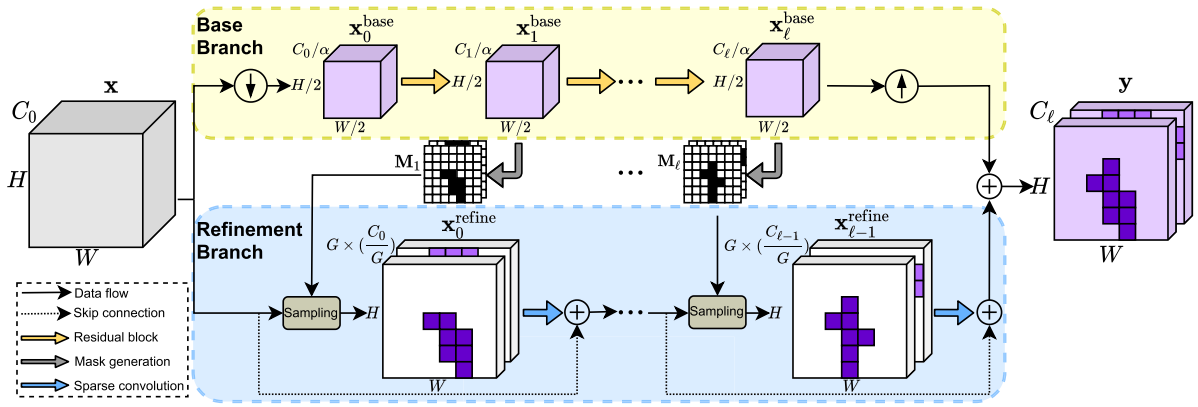


Fig. 2. The architecture of a module implementing the SAR method. The module is composed of a base branch (top) and a refinement branch (bottom). Note that in the refinement branch, only the colored areas are selected to perform the refinement.

conditioned on different channel groups; 4) the patch-level adaptive computation could be implemented with efficient operators. Finally, the two-part features are adaptively fused to complement each other.

In the following, we first introduce SAR in terms of its architecture design and inference paradigm. The implementation details and training strategies will then be presented.

A. Architecture Design

To perform *Spatially Adaptive feature Refinement*, we design a SAR-module with a two-branch structure as shown in Fig. 2: the base branch (top) processes features at a small scale, and the refinement branch (bottom) is designed to conduct convolutions on a set of regions at the original resolution. The regions are selected based on a content-aware stochastic region selection module. Such generic modules can be conveniently plugged in popular CNN architectures to build our SAR-networks.

1) Base Branch: The base branch (Fig. 2 (top)) processes coarse-grained features at a low resolution. Without loss of generality, we denote the input and output of the base branch as $\mathbf{x} \in \mathbb{R}^{C_0 \times H \times W}$ and $\mathbf{y}^{\text{base}} \in \mathbb{R}^{C_\ell \times H \times W}$, which share the same size of $H \times W$ at the spatial dimension, and ℓ is the number of convolutional blocks in a CNN stage.[1] The input and output channel numbers are denoted as C_0, C_ℓ respectively. The output of the base branch can be written as

$$\mathbf{y}^{\text{base}} = \uparrow \left(\mathcal{F}^{\text{base}} \left(\downarrow (\mathbf{x}) \right) \right), \tag{1}$$

where $\downarrow$ denotes a convolutional block with stride $s > 1$ that downsamples the input to the scale of $\frac{H}{s} \times \frac{W}{s}$, and $\mathcal{F}^{\text{base}}(\cdot)$ denotes a series of convolutional blocks processing the low-resolution features in the base branch. The upsample $\uparrow$ is implemented by the nearest interpolation, which consumes negligible computations compared to the convolution operations.

2) Region Selection Module: The region selection module is responsible for identifying the salient regions in the input based on the coarse-grained representations generated by the base branch (Sec. III-A.1). It yields a spatial mask deciding which locations should be processed by the refinement branch with the original high resolution (Sec. III-A.3).

As illustrated in Fig. 2, for the n-th block ($n = 1, 2, \cdots, \ell$) in the refinement branch, the input feature is divided into $K \times K$ grids (e.g. 7×7) in the spatial dimension without overlapping, and each grid is corresponding to one small patch with the size of $\frac{H}{K} \times \frac{W}{K}$ in the input feature. Moreover, to select independent locations for G sub-branches (groups) in the refinement branch, the output of the n-th region selection module is G binary-valued masks $\mathbf{M}_n(g) \in \{0, 1\}^{K \times K}$, $g = 1, 2, \cdots, G$.

To produce $\mathbf{M}_n(g)$, we first generate a real-valued tensor $\bar{\mathbf{M}}_n \in \mathbb{R}^{(2G) \times K \times K}$ by performing convolutions on the output of the n-th convolutional block in the base branch $\mathbf{x}_n^{\text{base}}$:

$$\bar{\mathbf{M}}_n = \mathbf{W}_{1\times1} * \text{AvgPool}(\text{ReLU}(\text{BN}(\mathbf{W}_{3\times3} * \mathbf{x}_n^{\text{base}}))), \tag{2}$$

where $\mathbf{W}_{k\times k} * (\cdot)$ denotes convolution with a $k \times k$ kernel size, and the adaptive pooling layer downsamples the input to the size of $K \times K$ regardless the input size. Note that our region selection modules are lightweight, as their channel numbers are much smaller than those in the backbone network.

The obtained $\bar{\mathbf{M}}_n$ is then reshaped into $\tilde{\mathbf{M}}_n \in \mathbb{R}^{2 \times G \times K \times K}$, which is directly binarized at the inference stage for making discrete decisions on whether to refine the patches at different locations in the input feature. During training, the binomial distribution parameterized by $\tilde{\mathbf{M}}_n(:, g, i, j)$ determines the probability that the corresponding patch is selected. Gumbel Softmax [27], [28] is adopted (details in Sec. III-C.2) to enable the end-to-end training of these decision functions.

3) Refinement Branch: The refinement branch (see Fig. 2 (bottom)) is built to perform convolutions on the high-resolution regions sampled by the region selection module (Sec. III-A.2). As the same spatial location can be of different importance for multiple channel groups, we divide the branch into several sub-branches (groups) along the channel dimension, and refines independent spatial locations in each sub-branch.

Specifically, for the n-th convolutional block in the refinement branch, the input $\mathbf{x}_{n-1}^{\text{refine}}$ is split into G groups, denoted as $\mathbf{x}_{n-1}^{\text{refine}}(g) \in \mathbb{R}^{\frac{C_{n-1}}{G} \times H \times W}$, $g = 1, 2, \cdots, G$. In the g-th sub-branch, the patches in $\mathbf{x}_{n-1}^{\text{refine}}(g)$ at the locations corresponding to the 1 elements in $\mathbf{M}_n(g)$ are selected to be refined at the

[1] Here we refer to a stage as a stack of multiple residual blocks processing feature maps at the same resolution.

original resolution. The procedure can be represented by

$$\mathbf{x}_n^{\text{refine}}(g) = \mathcal{F}_{n;g}^{\text{refine}}(\mathbf{x}_{n-1}(g) \otimes (\uparrow \mathbf{M}_n(g))), \tag{3}$$

where $\mathcal{F}_{n;g}^{\text{refine}}(\cdot)$ is the n-th convolutional block in the g-th sub-branch, and $\otimes$ denotes element-wise multiplication. The final output of G sub-branches are concatenated to obtain the resulting feature of the refinement branch:

$$\mathbf{y}^{\text{refine}} = \text{Concat}(\mathbf{x}_\ell^{\text{refine}}(1), \mathbf{x}_\ell^{\text{refine}}(2), \cdots, \mathbf{x}_\ell^{\text{refine}}(G)). \tag{4}$$

4) Adaptive Feature Fusion: The output of the two branches are directly fused through channel-wise weighted summation:

$$\mathbf{y} = \boldsymbol{a} \odot \mathbf{y}^{\text{base}} + (1 - \boldsymbol{a}) \odot \mathbf{y}^{\text{refine}}, \tag{5}$$

where $\odot$ is channel-wise multiplication, and the weighting factor $\boldsymbol{a} \in \mathbb{R}^{C_\ell}$ is generated from lightweight modules made of average pooling and linear layers.

5) Building SAR-Networks: We can conveniently establish dynamic models with the proposed SAR-module. Take ResNet as an example, a SAR-ResNet is built by replacing the original stages with the two-branch structure in Fig. 2. All the residual blocks can be directly used without modification, except that the channel numbers could be adjusted (see Sec. III-B). Note that we do not substitute the last stage of a network where the feature maps are at the smallest scale. More details of the implementation of SAR are presented in Appendix A.

B. Complexity Analysis

We further introduce a few variables to control the computational cost of the proposed SAR. Specifically, the *width* (i.e. number of channels) of the two branches are adjusted by two hyperparameters, α and β, respectively. First, the channel number in the base branch is compressed by a factor of α to generate coarse-grained features efficiently. Second, we can widen the refinement branch by β to increase the representation power when processing the selected high-resolution regions. By introducing α together with the reduction factor s for spatial resolution mentioned in Sec. III-A.1, the floating point-operations (FLOPs) of the base branch is reduced by

$$r^{\text{base}} \approx \frac{\sum_n \frac{C_{n-1}}{\alpha} \cdot \frac{C_n}{\alpha} \cdot \frac{H}{s} \cdot \frac{W}{s}}{\sum_n C_{n-1} \cdot C_n \cdot H \cdot W} = \frac{1}{s^2 \times \alpha^2}. \tag{6}$$

Particularly, when $s = \alpha = 2$, the base branch only evokes around 1/16 computation compared to a regular CNN stage.

Moreover, we define the activation rate for the n-th convolutional block in the refinement branch

$$r_n^{\text{refine}} = \frac{\sum_{g,i,j} \mathbb{I}(\mathbf{M}_n(g,i,j) = 1)}{G \times K \times K} = \frac{\sum_{g,i,j} \mathbf{M}_n(g,i,j)}{G \times K \times K}, \tag{7}$$

which represents the fraction of the refined areas. Given the widen factor β and the group (sub-branch) number G, the actual FLOPs of the block is mainly controlled by r_n^{refine}. Therefore, the proposed model is able to achieve a dynamic trade-off between accuracy and computational cost by optimizing r_n^{refine} to approximate different targets (see Sec. III-C.3).

Despite the two-branch architecture design, our SAR is both parameter-efficient and computation-efficient because: 1) the base branch has *reduced* spatial resolution and channel numbers; 2) the refinement branch is divided into multiple sub-branches along the channel dimension with the total width *unchanged*; 3) only a subset of high-resolution feature patches are *selectively* processed; 4) the adaptive feature fusion is extremely lightweight thanks to the global pooling operation and the compressed channel numbers. For example, employing our method in ResNet50 with $G = 4, s = 2, \alpha = 2, \beta = 1$ can reduce the computational cost and the amount of parameters by 48.8% and 8.3% respectively without sacrificing accuracy.

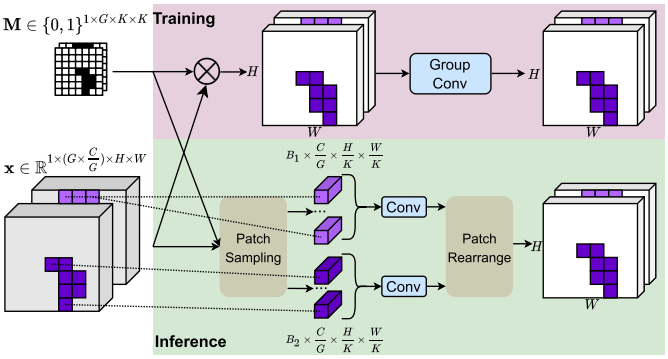


Fig. 3. Two different computational paradigms in the refinement branch for training and inference phases. During inference, B_1 and B_2 patches are sampled in the two sub-branches, respectively. In each sub-branch, the sampled patches are concatenated along the batch dimension.

C. Inference and Training

1) Computational Paradigms of the Refinement Branch: Different computational paradigms are realized for feature refinement in training and inference stages. During training, the unselected areas are masked by zero values (Eq. 3), and group convolution [29] is used to facilitate the parallel computation of G sub-branches (see Fig. 3 top). However, to practically avoid the computation on the unselected regions at the inference stage, *patch-level* sparse convolution is achieved through a *sampling-convolution-rearranging* procedure (see Fig. 3 bottom, the whole procedure is implemented with efficient operators in PyTorch [47]). Specifically, we sample the selected patches with the spatial size of $\frac{H}{K} \times \frac{W}{K}$ from every group of the input feature, and concatenate them along the *batch* dimension. Then regular convolution can be performed on the *"mini-batch"* consisting of these selected patches. The refined patches are finally rearranged to obtain the high-resolution output.

2) End-to-End Training of Region Selection Decisions: In SAR, whether each location of a high-resolution feature should be refined is decided based on a generated mask. Such *discrete* decisions made by the argmax function are non-differentiable, and therefore are hard to be optimized with gradient descent. The straight-through version of the Gumbel Softmax [27], [28] estimator is used to allow making discrete decisions in forward propagation and estimating gradients in backward propagation simultaneously. Specifically, after producing the real-valued mask $\tilde{\mathbf{M}}_n \in \mathbb{R}^{2 \times G \times K \times K}$ (Sec. III-A.2), we obtain the binary-valued mask $\mathbf{M}_n$ in the forward propagation by

$$\mathbf{M}_n(g,i,j) = \arg\max_{k \in \{0,1\}} (\log(\tilde{\mathbf{M}}_n(k,g,i,j)) + \mathbf{Z}(k,g,i,j)), \tag{8}$$

where $\mathbf{Z}$ denotes the *i.i.d* noise sampled from a Gumbel distribution independent to $\tilde{\mathbf{M}}$. Through this sampling strategy, the probability of $\mathbf{M}_n(g, i, j) = 1$ depends on $\tilde{\mathbf{M}}_n(k, g, i, j)$.

In the backward propagation, the gradient with respective to $\tilde{\mathbf{M}}_{k,g,i,j}$ is calculated based on the Softmax relaxation of Eq. 8 with a temperature term τ:

$$\hat{\mathbf{M}}_n(g, i, j) = \frac{\exp\left\{\left(\log\left(\tilde{\mathbf{M}}_n(1, g, i, j)\right) + \mathbf{Z}(1, g, i, j)\right)/\tau\right\}}{\sum_{k=0}^{1}\exp\left\{\left(\log\left(\tilde{\mathbf{M}}_n(k, g, i, j)\right) + \mathbf{Z}(k, g, i, j)\right)/\tau\right\}}. \tag{9}$$

The Softmax goes towards the argmax while $\tau \to 0$. We follow the settings in [18] and let τ decrease exponentially from τ_0 to 0.01 during the training process, which facilitates the optimization for the selection module, and the binary masks are generated with more certainty when the training ends.

3) Optimization Objective: To achieve a trade-off between network performance and computational cost, a loss item is introduced to control the amount of the patches selected for refinement. To be specific, let $r_{b,n}$ denote the activation rate[2] (as derived in Eq. 7) of the n-th block for the b-th input in a mini-batch of size B, $b = 1, 2, \cdots, B, n = 1, 2, \cdots, N$, where N is the total number of blocks. We aim to optimize the overall activation rate for the n-th block in a mini-batch to approximate a target rate t, i.e.

$$\begin{aligned}\mathcal{L}_{\text{refine}} &= \frac{1}{N}\sum_{n=1}^{N}(\bar{r}_n - t)^2 = \frac{1}{N}\sum_{n=1}^{N}(\frac{1}{B}\sum_{b=1}^{B} r_{b,n} - t)^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}(\frac{1}{B}\sum_{b=1}^{B}\frac{\sum_{g,i,j}\mathbf{M}_{b,n}(g, i, j)}{G \times K \times K} - t)^2. \end{aligned} \tag{10}$$

Note that an alternative to minimizing the loss item in Eq. 10 is optimizing the activation rate for every input sample instead of all the inputs in a mini-batch to approximate t. In this way, the additional loss item can also be designed as

$$\mathcal{L}'_{\text{refine}} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{B}\sum_{b=1}^{B}(r_{b,n} - t)^2. \tag{11}$$

However, we find the loss item in Eq. 10 leads to slightly better performance than that in Eq. 11 (experimental comparisons are provided in Appendix D) for allowing to 1) refine the salient regions in every input sample and 2) select different amounts of areas according to the complexity of different samples.

During training, each region selection module is initialized to select the whole feature maps, i.e. $\mathbf{M}_n(g, i, j) = 1$, $\forall n, g, i, j$. We minimize the loss function $\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda\mathcal{L}_{\text{refine}}$, where $\mathcal{L}_{\text{cls}}$ is the classification loss, and λ balances the two items.

IV. Experiments

In this section, SAR is applied to three classic vision tasks for evaluation, i.e. image classification, object detection and semantic segmentation. We first give an introduction about the experimental settings for image classification (Sec. IV-A). Then the effectiveness of SAR is validated by performing ablation studies and hyperparameter studies (Sec. IV-B). Comparisons with competing approaches are conducted in terms of the trade-off between classification accuracy and the average FLOPs over the validation set (Sec. IV-C on CIFAR and Sec. IV-D on ImageNet). Finally, the proposed method is evaluated on two downstream tasks, i.e. COCO object detection (Sec. IV-E) and PASCAL VOC semantic segmentation (Sec. IV-F).

A. Experimental Settings for Image Classification

We apply the proposed SAR on three challenging image classification benchmarks, i.e. CIFAR-10, CIFAR-100 [2] and ImageNet (ILSVRC2012 [48]).

The CIFAR-10 and CIFAR-100 datasets consist of 32×32 RGB images, with 10 and 100 classes of natural scene objects, respectively. Both CIFAR datasets contain 50,000 training images and 10,000 test images. Following [3], we apply a set of transformations to augment the training set of two CIFAR datasets, including 1) randomly flipping each image with probability 0.5 at each iteration, and 2) padding 4 pixels to each side of the images, and then performing a random cropping to produce 32×32 sized images at each iteration.

The ImageNet dataset contains 1.2 million training images and 50,000 validation images with 1000 classes. Following the common practice in [3], we apply the standard data augmentation and the images are cropped to a size of 224×224.

We construct our SAR-networks with $G = 2, s = 2, \alpha = 2$, $\beta = 1$ if not mentioned otherwise. All the models are trained by stochastic gradient descent (SGD) in an end-to-end fashion [49]. More training details are presented in Appendix B.

B. Effectiveness of SAR

In this subsection, ablation studies are first conducted on the CIFAR dataset to evaluate the effectiveness of SAR, and then we explore how the hyperparameters affect the network performance. The number of training epochs is fixed as 300.

1) Ablation Studies: As SAR requires fusing features from two branches, we first validate the effectiveness of our architecture design by comparing it with two variants with only the base branch and only the refinement branch, respectively.

Moreover, to validate the content-aware region selection paradigm, two data-independent selection strategies (with the same two-branch structure) are tested, including center cropping and uniform sampling. For center cropping, the refinement branch always refines approximately 1/2 of the areas at the center of high-resolution features. For uniform sampling, half of the mask elements are set as 1 with an equal interval of 2.

All the experiments are conducted on CIFAR-100 with the same ResNet32-based architecture, and we choose two target activation rates (0.3 and 0.5) for our method. Note that for fair comparisons among models with similar complexity, we do not compress the feature channels of the network with only

[2]Here we omit the superscript "refine" for simplicity.

TABLE I

ABLATION STUDIES OF THE STRUCTURE DESIGN AND THE REGION SELECTION STRATEGY ON CIFAR100

Model	Sampling	FLOPs	Top-1 Acc.
ResNet32	Base branch	82M	71.03±0.09
	Refinement branch	125M	72.41±0.32
	Center cropping	105M	73.24±0.09
	Uniform sampling	98M	72.04±0.08
	SAR (Ours, $t=0.3$)	91M	73.33±0.27
	SAR (Ours, $t=0.5$)	102M	**73.97**±0.10

TABLE II

CIFAR-100 ACCURACY OF THE PROPOSED SAR WITH DIFFERENT GROUP NUMBER G AND TARGET ACTIVATION RATE t

Model	G	t	FLOPs	Top-1 Acc.
SAR-ResNet32	1	0.3	138M	72.59±0.21
		0.5	173M	73.43±0.31
		0.7	213M	74.01±0.18
	2	0.3	91M	73.33±0.27
		0.5	102M	73.97±0.10
		0.7	121M	73.96±0.17
	4	0.3	65M	72.66±0.15
		0.5	74M	73.47±0.16
		0.7	84M	73.81±0.27

the base branch by setting $\alpha = 1$. From the results in Table I, we can clearly draw the conclusions as follows:

- The networks with our two-branch architecture design consistently achieve higher accuracy when evoking comparable FLOPs. Even with the uniform sampling strategy, the refinement branch still has a positive effect on classification. This indicates the necessity and effectiveness of our feature refinement mechanism and its superiority over the methods that only reduce the feature resolution (base branch) or merely conduct selective convolutions on different spatial locations (refinement branch).
- Our content-aware sampling strategy significantly outperforms the other two data-independent sampling methods, which demonstrates that the module can adaptively select salient patches for different inputs and effectively improve the network performance with similar computation.

2) Hyperparameter Studies: We further test the effect of the group number G and the target activation rate t in our SAR. The experiments are conducted based on ResNet32 with the default settings for other hyperparameters.

The results are shown in Table II, from which we can see that the group number G has a dominant effect on the overall performance of the proposed method, and larger activation rates lead to improved classification accuracy together with larger FLOPs for refining more feature areas. Furthermore, we witness that the models with $G > 1$ are superior to those with $G = 1$ in terms of the trade-off between accuracy and computation. Specifically, the accuracy of our SAR with $G = 2, t = 0.5$ is higher than that of the model with $G = 1, t = 0.5$ by 0.50% with 41% less computation. This meets our intuition that the same spatial location can be of different importance for multiple channel groups, and a larger group number allows our SAR to refine different regions for different groups, improving the flexibility and the efficiency of networks. Note that an inappropriately large G would result in a drop of network performance due to insufficient model complexity. We observe that $G = 2$ and $G = 4$ can effectively model the importance variations among channels and maintain the representation power of our SAR-networks, reaching a sweet spot between complexity and performance. More empirical studies on other hyperparameters are provided in Appendix C

TABLE III

RESULTS ON THE CIFAR-100 DATASET. ALL THE MODELS ARE TRAINED FOR 300 EPOCHS AS IN [50] FOR FAIR COMPARISON, AND THE RESULTS OF THE COMPETING METHODS ARE OBTAINED FROM [50]

Model	Method	FLOPs	Top-1 Acc.
ResNet56	Baseline	126M	72.00±0.40
	LCCL [22]	76M	68.37
	FPGM [51]	59M	69.66
	TAS [50]	61M	72.25
ResNet110	Baseline	254M	73.69±0.45
	LCCL [22]	173M	70.78
	FPGM [51]	121M	72.55
	TAS [50]	120M	73.16
ResNet32×2	Baseline	271M	75.33±0.07
	SAR ($G4\beta1, t=0.5$)	74M	73.47±0.16
	SAR ($G4\beta1, t=0.7$)	84M	73.81±0.27
	SAR ($G4\beta2, t=0.3$)	127M	74.94±0.27
	SAR ($G4\beta2, t=0.6$)	182M	**75.70**±0.23

TABLE IV

CIFAR-100 RESULTS WITH SAR IMPLEMENTED ON WIDERESNET AND RESNEXT BACKBONE NETWORKS

Model	Method	FLOPs	Top-1 Acc.
WRN28×10	Baseline	5.24G	80.36±0.15
	SAR ($G2\beta1, t=0.3$)	1.59G	79.63±0.17
	SAR ($G2\beta1, t=0.5$)	1.97G	**80.92**±0.12
ResNeXt29-8×32	Baseline	2.06G	78.05±0.64
	SAR ($G2\alpha1, t=0.5$)	1.28G	79.19±0.25
	SAR ($G2\alpha1, t=0.7$)	1.44G	79.58±0.15
	SAR ($G1\alpha1, t=0.5$)	1.46G	78.88±0.19
	SAR ($G1\alpha1, t=0.8$)	1.81G	**80.13**±0.39

C. Results on CIFAR

1) CIFAR-10: We compare our SAR with many modern dynamic deep models on CIFAR-10, including convolution with adaptive inference graph (Conv-AIG) [39], batch-shaping (BAS) [42] and dynamic convolution (DynConv) [23]. Following [42], the training procedure is extended to 500 epochs for better convergence of the stochastic sampling modules and fair comparison. All models are implemented on the basis of ResNet20 or ResNet32. With the setting of $G2s2\alpha2\beta1$, the FLOPs of our models lie in the same range as the competitors.

The results are shown in Figure 4, from which we witness that the proposed SAR is superior to other dynamic models from two aspects: 1) SAR-ResNet20 obtains 0.8% higher top-1 accuracy than ResNet20-BAS [42] with similar FLOPs (∼37M); 2) when achieving comparable accuracy with DynConv32 [23] (∼93.9%), SAR-ResNet32 consumes 20% fewer FLOPs.

2) CIFAR-100: On CIFAR100, our method is compared with low-cost collaborative layer (LCCL) [22], which performs

TABLE V

IMAGENET RESULTS. THE ACCURACY VARIATIONS OF ALL BASELINES ARE CALCULATED BASED ON THEIR ORIGINAL PAPERS FOR FAIR COMPARISON

Model	Method	Type	Dynamic	FLOPs	Top-1 Acc.	Top-1 Acc. Variation	FLOPs Drop
ResNet34	Baseline	–	✗	3.67G	74.2	–	–
	SFP [51]	Pruning	✗	2.13G	71.8	−2.1	42%
	FPGM [12]	Pruning	✗	2.13G	72.1	−1.8	42%
	LCCL [22]	Spatial	✓	2.70G	73	−0.4	25%
	BAS [42]	Channel	✓	2.27G	73.2	−0.1	38%
	CGNet [52]	Spatial & Channel	✓	–	71.3	−1.1	50%
	SFSI [18]	Spatial	✓	2.53G	73.7	−0.5	31%
	SFSI [18]	Spatial	✓	2.16G	73.0	−1.2	41%
	SAR ($G4s4\alpha2, t=0.6$)	Spatial & Channel	✓	1.99G	**74.5**	**+0.3**	46%
	SAR ($G2s2\alpha2, t=0.5$)	Spatial & Channel	✓	**1.87G**	74.3	+0.1	**49%**
ResNet50	Baseline	–	✗	4.10G	76.8	–	–
	ThiNet [53]	Pruning	✗	2.59G	72.0	−0.8	37%
	SSS-ResNet [54]	Pruning	✗	2.82G	74.2	−1.9	31%
	TAS [50]	Pruning	✗	2.31G	76.2	−1.26	44%
	Conv-AIG [39]	Skip layers	✓	–	75.8	−0.3	24%
	ACT [14]	Skip layers	✓	3.19G	73.1	−1.4	22%
	SACT [21]	Spatial	✓	3.31G	73.3	−1.3	19%
	BAS [42]	Channel	✓	2.07G	75.7	−0.4	50%
	SAR-dil ($G2s2\alpha2, t=0.5$)	Spatial & Channel	✓	3.06G	77.6	+0.8	25%
	SAR-dil ($G2s2\alpha4, t=0.5$)	Spatial & Channel	✓	2.80G	77.4	+0.6	32%
	SAR ($G2s4\alpha1, t=0.5$)	Spatial & Channel	✓	2.55G	**78.0**	**+1.2**	38%
	SAR ($G2s2\alpha2, t=0.5$)	Spatial & Channel	✓	2.40G	77.8	+1.0	41%
	SAR ($G4s2\alpha2, t=0.7$)	Spatial & Channel	✓	2.22G	77.6	+0.8	46%
	SAR ($G4s2\alpha2, t=0.5$)	Spatial & Channel	✓	2.12G	77.3	+0.5	48%
	SAR ($G8s2\alpha2, t=0.5$)	Spatial & Channel	✓	**1.99G**	77.1	+0.3	**51%**

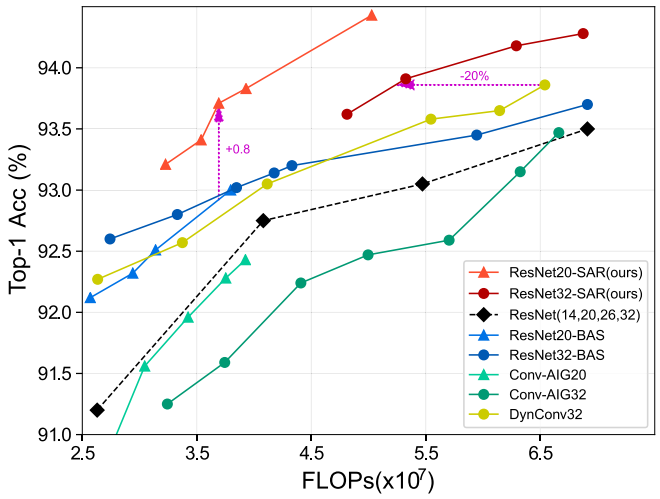


Fig. 4. Comparison with other dynamic methods on CIFAR-10.

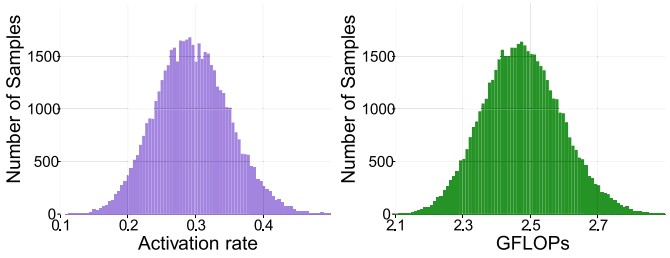


Fig. 5. Histogram for the activation rates and FLOPs of SAR-ResNet50 with $G = 1, t = 0.3$ on the validation set of ImageNet.

pixel-level spatially adaptive inference without our two-branch architecture design. Moreover, some pruning based methods, e.g. filter pruning via geometric median (FPGM) [12] and transformable architecture search (TAS) [50], are included as comparisons. Note that we implement SAR on ResNet32 with a wider refinement branch ($2\times$) and large group number ($G = 4$) to compare our networks to baseline models with similar FLOPs. From the results in Table III, we can see that our method outperforms the competitors consistently at various computational complexity levels.

We also validate our SAR by implementing it on two other backbone CNNs, i.e. WideResNet (WRN) [30] and ResNeXt [29]. The results in Table IV demonstrate the notable generality and effectiveness of our SAR in different architectures. For instance, the computation of WRN28 $\times$ 10 can be reduced by 62% while achieving higher accuracy, and the Top-1 accuracy of ResNeXt29-8 $\times$ 32 is increased by more than 2% even with less computation. It is worth noting that the performance of SAR-ResNeXt with $G = 1$ (the last two rows in Table IV) can clearly prove the gain of accuracy and efficiency brought by performing our *Spatially Adaptive feature Refinement*, and a larger group number is used in other experiments to further improve the trade-off between accuracy and efficiency.

D. Results on ImageNet

Following the training settings in [18], [22], we train the models for 200 epochs on the ImageNet dataset. Apart from the competing models mentioned in previous subsections, other spatially adaptive inference methods including stochastic feature sampling and interpolation (SFSI) [18], channel gating networks (CGNet) [52], networks with adaptive computa-

TABLE VI
SPEED TEST RESULTS OF THE PROPOSED SAR AND COMPARISONS WITH COMPETING METHODS ON CPU AND GPU

Model	Type	Dynamic	Params	FLOPs	CPU Latency (ms)		GPU latency (ms)	Top-1 Acc.
					Single-thread	Multi-thread		
ResNet50	Baseline	✗	25.6M	4.1G	389±17	371±14	**11.6**±0.4	76.8
BL-ResNet50 [16]($\alpha=2,\beta=2$)	Multi-scale	✗	27.0M	2.9G	325±17	334±17	**12.8**±0.1	77.3
BL-ResNet50 [16] ($\alpha=1,\beta=2$)	Multi-scale	✗	30.1M	4.3G	411±19	418±20	13.8±0.8	**77.9**
Oct-ResNet50 [17] ($\alpha=0.5$)	Multi-scale	✗	25.6M	2.4G	325±28	324±27	27.3±1.0	77.4
DynConv-ResNet101 [23] (sparsity ≈ 0.3)	Spatial	✓	44.9M	3.0G	928±39	900±43	56.6±6.9	75.6
SAR-ResNet50 ($G2s2\alpha2, t=0.5$)	Spatial & Channel	✓	**25.0M**	**2.4G**	**281**±13	**243**±13	21.9±0.5	**77.8**
SAR-ResNet50 ($G4s2\alpha2, t=0.5$)	Spatial & Channel	✓	**23.4M**	**2.2G**	**279**±15	**242**±13	32.9±0.8	77.3
SFSI-ResNet34 [18] (sparsity ≈ 0.5)	Spatial	✓	21.3M	2.2G	704±35	703±34	100.0±3.0	73.0
SAR-ResNet34 ($G2s2\alpha2, t=0.5$)	Spatial & Channel	✓	21.8M	**1.9G**	**219**±16	**183**±10	**17.8**±0.6	**74.3**

TABLE VII
RESULTS OF INCORPORATING SAR WITH EFFICIENT CNNS

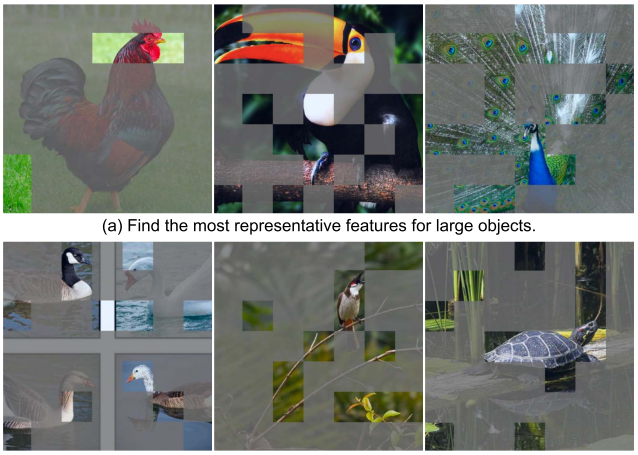
Model	Method	FLOPs	Top-1 Acc.
ResNeXt50 (32×4d)	Baseline	4.23G	77.8
	SAR ($G2s2\alpha1$)	3.23G	**78.4**
	SAR ($G2s4\alpha1$)	**2.96G**	78.1
MobileNet-v2	Baseline (×0.75)	**0.21G**	69.8
	SAR ($G2s2\alpha1$)	**0.21G**	**70.3**

tion time (ACT [14] and SACT [21]) are also tested as baselines. Additionally, we include some pruning approaches, e.g. ThiNet [53] and data-driven sparse structure selection (SSS-ResNet [54]) as comparisons.

1) Main Results: The results in Table V demonstrate that the proposed SAR improves the computational efficiency of the backbone networks and outperforms other methods by a large margin. For example, SAR-ResNet34 ($G4s4\alpha2\beta1, t=0.6$) reduces 46% of FLOPs with even 0.3% accuracy improvement, compared with the pruning based SFP [51] and FPGM [51] that reduce 42% FLOPs with 2.1% and 1.8% accuracy drop, respectively. More remarkably, we observe that our method is significantly superior to the competitors on ResNet50: even a higher classification accuracy can be achieved when the computational cost is reduced by 51%. In addition, SAR-ResNet50 with $G2s2\alpha2\beta1, t=0.5$ improves the accuracy by 1.0% while reducing 41% FLOPs. By performing refinement only on the strategically selected regions and fusing low-resolution and high-resolution features, SAR can efficiently learn both global and task-relevant local information, improving the overall efficiency without sacrificing the model performance.

Moreover, based on SAR-ResNet50 ($G1s2\alpha2\beta1, t=0.3$) which achieves 76.8% accuracy on ImageNet, we plot the histogram for the activation rate and FLOPs on the validation set. From the results in Fig. 5, we find that: 1) ∼30% of the patches are selected for refinement averagely; 2) the distribution of both the real activation rates and the FLOPs follow a Normal-like distribution, revealing that our SAR enables the network to adaptively allocate the computation for each input image; 3) for more than 97% of the samples, less than 40% of feature areas in SAR-ResNet50 are refined by the model, leading to the same classification accuracy as the original model with 39.7% less computation.

It is worth noting that to effectively increase the reception field in the base branch, an alternative approach to the adopted large-stride convolution is dilated convolution [55].



(a) Find the most representative features for large objects.

(b) Locate smaller objects, and avoid expensive computation on background areas.

Fig. 6. Visualization results on ImageNer. Only the brilliant patches are selected to be processed at the original resolution by the refinement branch.

We empirically studied the variants of SAR-ResNet50 using dilated convolution (see SAR-dil in Table V), and find that they achieve comparable accuracy to the original implementation while consuming more computation.

2) Inference Latency: Apart from the theoretical efficiency (FLOPs), we also measure the realistic latency of our proposed SAR. Big-little ResNet (BL-ResNet50) [16], Octave-ResNet50 [17], DynConv-ResNet101 [23] and SFSI-ResNet34 [18] are selected as baselines. The FLOPs and Top-1 accuracy of the tested models are generally within the same range for clear comparison. The experiments are conducted under the same hardware environments: Intel-core CPU (i7-6700HQ @ 2.60GHz) and TITAN XP GPU. Both single-threaded and multi-threaded modes with batch size 1 are tested, and the input images are with the size of 224×224.

The results in Table VI show that thanks to the proposed *patch-level* adaptive computation paradigm and our efficient implementation, SAR can achieve better trade-off between accuracy and latency than *pixel-level* dynamic models [18], [23] on either CPU or GPU. Notably, when the models are deployed on CPU, our SAR demonstrates significant speedup compared to both dynamic and static networks. Interestingly, most competing methods fail to benefit from the multi-threaded mode in the scenario where the input images are fed sequentially. In contrast, by concatenating the selected patches along the batch dimension (Sec. III-C.1), regular convolution could be performed on the formed *"mini-batches"*, and thus the efficiency of SAR could be further

TABLE VIII
OBJECT DETECTION RESULTS ON THE MS COCO DATASET

Detection Framework	Backbone	Backbone FLOPs	mAP	AP_{50}	AP_{75}	mAP_S	mAP_M	mAP_L
Faster R-CNN	ResNet50 (Original)	88.03G	36.4	58.4	39.3	21.8	40.2	46.1
	SAR-ResNet50 ($G2s2\alpha2, t = 0.5$)	**62.63G**	36.7	58.9	39.2	21.6	40.1	48.3
	SAR-ResNet50 ($G2s4\alpha1, t = 0.5$)	65.79G	**37.3**	**59.7**	**39.9**	21.2	**40.7**	**49.2**
RetinaNet	ResNet50 (Original)	88.03G	35.5	55.2	37.8	20.1	39.1	47.3
	SAR-ResNet50 ($G2s2\alpha2, t = 0.5$)	**62.63G**	36.7	57.4	38.9	**20.5**	40.6	**49.7**
	SAR-ResNet50 ($G2s4\alpha1, t = 0.5$)	65.79G	**37.1**	**57.7**	**39.5**	20.2	**41.2**	49.6

TABLE IX
SEMANTIC SEGMENTATION RESULTS ON PASCAL VOC

Framework	Backbone	Backbone FLOPs	mIoU(%)
DeepLab-v3	ResNet50 (Original)	34.1G	77.3
	SAR-ResNet50	**25.8G**	**78.2**
PSPNet	ResNet50 (Original)	34.1G	77.0
	SAR-ResNet50	**25.8G**	**78.2**

increased in the multi-threaded mode. For example, compared to Oct-ResNet50 achieving 77.4% accuracy with ∼324ms latency, SAR-ResNet50 yields 77.8% accuracy with ∼243ms.

As the acceleration of our dynamic operations and the excessive execution of multiple sub-branches has not been fully supported by modern deep learning libraries, the theoretical efficiency of SAR is not fully translated into GPU latency. For instance, when we increase G from 2 to 4, the GPU latency increases even with fewer FLOPs. However, it can be observed that SAR is superior to the competitive pixel-level dynamic networks [18], [23], and achieves comparable GPU latency to some static models. We believed that the future work on the co-design of computing hardware and libraries could facilitate harvesting the theoretical efficiency gains of our method.

3) SAR on Efficient CNNs: We also evaluate SAR on ResNeXt [29] and MobileNet-v2 [8]. The results in Table VII demonstrate that SAR can be incorporated with group (or depth-wise) convolutions to further improve their efficiency.

4) Visualization: We visualize the refined regions selected by the 3-rd block of our SAR-ResNet50 with $G1s2\alpha2\beta1, t = 0.3$ (whose accuracy is the same as the original ResNet50) on the validation set of ImageNet, and the results are shown in Fig. 6 (see more illustrations in Appendix E). The effectiveness of SAR can be validated from two aspects:

- For the images containing large objects, SAR is able to **find the most representative features** for more precise recognition (see the head of the chicken, the beak of the bird and the patterns of the peacock in Fig. 6 (a));
- For the images containing small objects which have higher spatial redundancy (more areas of background), our SAR can help **locate the objects** and avoid redundant computation on the less informative areas, which improves the overall efficiency of the network. For example, even four swans are contained in one image (Fig. 6 (b) left), our region selection module is able to precisely locate all of their heads with merely the classification label as supervision. Similarly, SAR successfully locate the bird and the turtle though they only occupy a small part in the images (Fig. 6 (b) middle and right).

E. Object Detection

1) Experimental Settings: We further validate our SAR on COCO object detection. Our models are trained on the 118k images of the COCO training set and evaluated on the 5k images of the COCO 2017 validation set. The mean average precision (mAP) is used to measure the network performance. Two detection frameworks are utilized in the experiments, including Faster R-CNN [56] with Feature Pyramid Network [4], and RetinaNet [57]. Due to the generality of SAR, we can conveniently replace the backbone networks with ours that are pre-trained on ImageNet, and the whole models are finetuned on COCO with SGD for 12 epochs with the hyper-parameters suggested in [4], [57]. The input images are resized to a short side of 800 and a long side not exceeding 1333.

2) Results: We evaluate SAR-ResNet50 with two hyperparameter settings as our backbone. The results are shown in Table VIII, where AP_{50} and AP_{75} denote the average precision (AP) over 50% and 75% IoU thresholds. The metric mAP takes the average of AP values over thresholds within the range of 50%–95%, and mAP_S, mAP_M and mAP_L denote mAP for the objects at different scales (small, middle and large).

As shown in Table VIII, our method significantly improves the network performance mAP in both detection frameworks with fewer FLOPs. Specifically, for ResNet50, SAR improves the mAP by 0.9% and 1.6% for Faster R-CNN and RetinaNet respectively while saving around 25% computation. It is worth noting that if we downsample the features with an inappropriately large scale $s = 4$, the capability of the model to recognize small objects might be decreased. However, both mAP_M and mAP_L could be increased by a large margin even with less computation. All these results demonstrated the effectiveness of the proposed method on object detection.

F. Semantic Segmentation

1) Experimental Settings: The proposed SAR is applied to the semantic segmentation task on PASCAL VOC [58]. The VOC dataset includes 20 object categories and one background class. The original dataset contains 1464, 1449, and 1456 images for training, validation, and testing, respectively. The augmented data provided by [59] is used in our experiments, leading to 10582 training samples. All the models are pre-trained on ImageNet and then finetuned on the VOC dataset.

2) Results: We substitute the backbone network of two segmentation frameworks (DeepLab-v3 [5] and PSPNet [45]) with our SAR-ResNet50 ($G2s4\alpha1, t = 0.5$). The results are shown in Table IX, from which we can observe that SAR

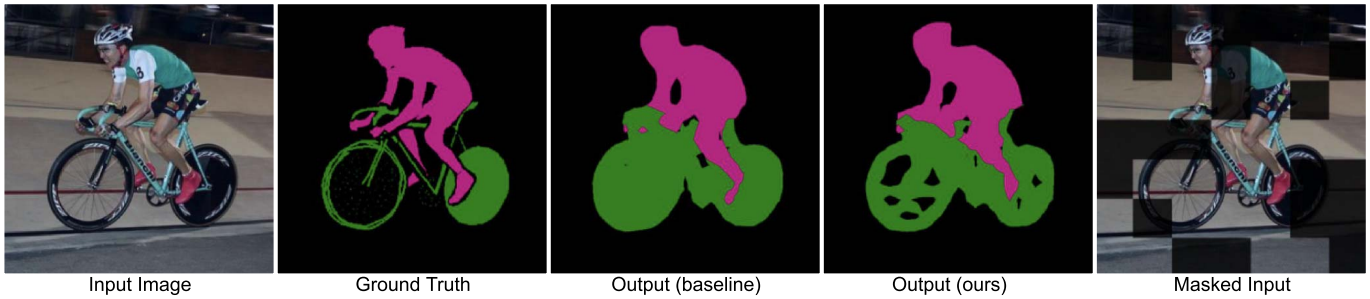


Fig. 7. Visualization on VOC semantic segmentation. The 5-th column is obtained by multiplying the input image with our generated mask, and the bright areas are selected by our region selection module to refine.

can consistently improve the performance of both baselines in terms of mIoU while reducing the overall computational cost.

The effectiveness of SAR is further analyzed by visualizing the segmentation results on VOC. From the illustration in Fig. 7, it can be clearly verified that the region selection module of our SAR successfully identifies salient foreground in the input image, leading to more precise segmentation, especially for the front wheel of the bicycle.

V. Conclusion

In this paper, we propose a *Spatially Adaptive feature Refinement* (SAR) method to reduce the commonly existing spatial redundancy in deep CNNs from the perspective of both multi-scale representations and *patch-level* spatially adaptive inference. SAR avoids a substantial amount of unnecessary computation on high-resolution features by fusing the coarse-grained features obtained from low-resolution representations and the selectively refined regions with high resolution. The adaptiveness of the method is further improved by refining independent spatial locations for different channel groups. The approach is sufficiently generic to be implemented in a variety of deep CNNs with minimal modifications. Experimental results on image classification, object detection and semantic segmentation reveal the considerable redundancy in the spatial dimension of features. By effectively exploiting such redundancy, our proposed SAR significantly reduces both theoretical computation and realistic inference latency of deep CNNs without sacrificing accuracy, achieving a better trade-off between network performance and computational cost.

Appendix A
Network Structure

We take SAR-ResNet34 and SAR-ResNet50 with $G1s2\alpha2\beta1$ as examples, and show the details of their network structures in Table X. Note that a BasicBlock is a residual block composed of two 3×3 convolution layers, and a Bottleneck is a residual block composed of $1 \times 1, 3 \times 3, 1 \times 1$ convolutions, which keeps the same as in a regular ResNet. A $[\text{Bottleneck}, C] \times n$ denotes that n Bottlenecks are cascaded. In each one, the first 1×1 and the 3×3 have $C/4$ output channels, and the last 1×1 has C output channels. The subscripts B and R represent the base branch and the refinement branch respectively. If $G > 1$, all the convolution layers in the refinement branch are split along the channel dimension into G sub-branches with the total width unchanged. To align the spatial size and the channel size of features from two branches, the nearest upsampling and a 1×1 convolution are applied on the output of the base branch.

To summarize, we substitute the first 3 stages in a ResNet by our SAR-modules. In each stage, the base branch processes the features with low resolution and compressed channels, and the refinement branch performs patch-level spatially adaptive computation.

TABLE XI
CIFAR-100 Accuracy of the Proposed SAR With Different α, β, s and Target Activation Rate t

Model	α	β	s	t	FLOPs	Top-1 Acc.
SAR-ResNet32	1	1	4	0.3	101M	72.54±0.50
				0.5	113M	73.90±0.13
				0.7	132M	74.11±0.15
	1	1	2	0.3	125M	73.17±0.11
				0.5	141M	74.70±0.22
				0.7	160M	74.43±0.14
	2	1	2	0.3	91M	73.33±0.27
				0.5	102M	73.97±0.10
				0.7	121M	73.96±0.17
	2	2	2	0.3	223M	75.19±0.19
				0.5	268M	76.05±0.22
				0.7	344M	76.03±0.45

TABLE XII
CIFAR-100 Results With Different Training Strategies

Model	Strategy	t	FLOPs	Top-1 Acc.
SAR-ResNet32	1	0.3	125M	72.85±0.19
		0.5	141M	74.38±0.11
		0.7	160M	74.38±0.20
	2	0.3	125M	72.97±0.16
		0.5	141M	74.22±0.10
		0.7	160M	74.57±0.24
	3	0.3	125M	73.17±0.11
		0.5	141M	74.70±0.22
		0.7	160M	74.43±0.14

TABLE XIII
Performance Comparison of Two Additional Loss Items

Model	Loss	FLOPs	Top-1 Acc.
SAR-ResNet32 ($t=0.3$)	Batch-target	125M	**73.17**±0.11
	Sample-target	118M	72.43±0.24
SAR-ResNet32 ($t=0.5$)	Batch-target	141M	**74.70**±0.22
	Sample-target	139M	74.44±0.17
SAR-ResNet32 ($t=0.7$)	Batch-target	160M	**74.43**±0.14
	Sample-target	161M	74.22±0.29

TABLE X
NETWORK CONFIGURATIONS OF SAR-RESNET34 AND SAR-RESNET50. OUTPUT SIZE IS ILLUSTRATED IN THE PARENTHESIS

Layers	SAR-ResNet34	SAR-ResNet50
Convolution Pooling	7×7, 64, s2 (112×112) MaxPooling (56×56)	
SAR-Module	$[\text{BasicBlock}_B, 128]$ ×2 $[\text{BasicBlock}_R, 256]$ ×2 BasicBlock, 256, s2 (28×28)	$[\text{Bottleneck}_B, 128]$ ×2 $[\text{Bottleneck}_R, 256]$ ×2 Bottleneck, 256, s2 (28×28)
SAR-Module	$[\text{BasicBlock}_B, 256]$ ×3 $[\text{BasicBlock}_R, 512]$ ×3 BasicBlock, 512, s2 (14×14)	$[\text{Bottleneck}_B, 256]$ ×3 $[\text{Bottleneck}_R, 512]$ ×3 Bottleneck, 512, s2 (14×14)
SAR-Module	$[\text{BasicBlock}_B, 512]$ ×5 $[\text{BasicBlock}_R, 1024]$ ×5 BasicBlock, 1024 (14×14)	$[\text{Bottleneck}_B, 512]$ ×5 $[\text{Bottleneck}_R, 1024]$ ×5 Bottleneck, 1024 (14×14)
ResBlock	[BasicBlock, 1024] ×3, s2 (7×7)	[Bottleneck, 1024] ×3, s2 (7×7)
Pooling	7× 7 Average Pooling	
FC, Softmax	1000	

APPENDIX B
EXPERIMENTAL SETTINGS

A. CIFAR-10 and CIFAR-100

All the models are trained using SGD with an initial learning rate of 0.1×batch size/64 which decays with a cosine shape, and we apply 0.0001 and 0.9 as the weight decay and the momentum, respectively. The networks are trained on 1 Nvidia 2080Ti GPU with a batch size of 128. We train the models for independent numbers of epochs in different sections. Specifically, in Sec. IV-B, to validate the effectiveness of our method, the models are trained for 300 epochs. In Sec. IV-C.1, we follow the settings in [42] and train the models on CIFAR-10 (Fig. 4) for 500 epochs. In Table III, models are trained for 300 epochs for fair comparisons with [50], while the models in Table IV are trained for 500 epochs.

B. ImageNet

We use SGD as the optimizer which starts with a 5-epoch warmup procedure, where the learning rate linearly increases to 0.1×batch size/256. Then the learning rate decays with a cosine shape. The weight decay and momentum parameters are set to 0.0001 and 0.9 respectively. Following [18], [50], the models are trained for 200 epochs with a batch size of 256 on 4 Nvidia 2080Ti GPUs. Label smoothing [60] is introduced to prevent overfitting.

C. Optimization of Region Selection

For all our experiments, the temperature τ in the Gumbel Softmax module decreases from 1.00 to 0.01 exponentially, allowing the mask generation to be optimized at the beginning and proceed with more certainty at the end of training. To avoid discarding certain regions too early, the target activation rate t is first set to 1 in the beginning, and decays by three steps until the middle of the training procedure. The factor λ in the loss function is set to 1 and 0.5 for CIFAR and ImageNet respectively.

APPENDIX C
HYPERPARAMETERS STUDY

In Sec. IV-B.2, the effects of the group number G and the target rate t are analyzed, and here we discuss how α and β are used to control the channel numbers in the base branch and the refinement branch. Moreover, the effect of s, which denotes the downsample stride for features in the base branch will be analyzed as well. The experiments are conducted on CIFAR-100 with different target activation rates, and all the models are trained for 300 epochs.

Fig. 8. Failure cases of the region selection module.

Here we have G fixed as 2, adjusting α, β and s for SAR-ResNet32 on CIFAR-100. From the results in Table XI, we can conclude that the network performance and the computational cost can be adjusted in a large range with different settings of α, β and s. It is worth noting that if we set $G4\alpha 2\beta 2s2, t = 0.3$ (see Table III in the paper), the model would achieve a 74.94% accuracy while using 127M FLOPs, and it obviously outperforms the model with $G2\alpha 1\beta 1s2, t = 0.3$. This phenomenon meets our intuition that a large channel number together with a large group number for the refinement branch could further increase the adaptiveness and flexibility of our proposed method.

APPENDIX D
ADDITIONAL LOSS ITEM

In the paper, we optimize the average activation rate in each mini-batch to approximate a given target activation rate t (batch-target), rather than optimizing every input sample to select a fixed amount of areas (sample-target). The overall training loss can be represented by $\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{refine}}$, where for the second item, the batch-target and sample-target loss can be written as Eq. 10 and Eq. 11 respectively. In this section, we empirically compare the two forms of loss.

The results on SAR-ResNet32 ($G2s2\alpha 1\beta 1$) are shown in Table XIII, which indicates that the batch-target loss item (Eq. 10) consistently leads to better performance. This can be due to that it allows the network to refine different amount

Fig. 9. More success cases of the region selection module.

of patches for different input samples, and the adaptive computation is both spatial-wise and sample-wise. In contrast, the sample-target loss item (Eq. 11) lacks the flexibility to treat different inputs dynamically.

APPENDIX E
VISUALIZATION RESULTS

In this section, more visualization results are presented. We first analyze some failure cases where some non-ideal patches are selected, and then more success cases are shown.

A. Failure Cases

We illustrate some failure cases, which can be summarized into three types of failure cases:

1) Fail to localize. The method may occasionally fail to cover the target objects (see Fig. 8 left, the region selection module seems to focus on the wood rather than the bird). This case is quite uncommon.

2) Select the background. As Fig. 8 (middle) illustrates, besides the foreground objects, unnecessary patches containing the background are sometimes selected, leading to a suboptimal solution to reducing the spatial redundancy. Note that such cases are also infrequent.

3) Select the unlabeled objects. As shown in Fig. 8 (right), when the input image contains multiple categories of objects (e.g. an image of a fish held by a human), the region selection module may select patches covering the objects of unlabeled categories (e.g. the human). From another perspective, this situation can further validate that our region selection module is content-aware and able to recognize objects without being limited by classification labels. This property enables our method to be effective on other tasks such as object detection.

B. Success Cases

The success cases refer to that appropriate number of semantic patches are selected, covering **the most representative features**, or precisely **localizing the target objects** with few background areas selected, or both. More results are shown in Fig. 9.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.

[3] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger, "Convolutional networks with dense connectivity," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, May 23, 2019, doi: 10.1109/TPAMI.2019.2918284.

[4] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.

[5] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*.

[6] Z. Huang, C. Wang, X. Wang, W. Liu, and J. Wang, "Semantic image segmentation by scale-adaptive networks," *IEEE Trans. Image Process.*, vol. 29, pp. 2066–2077, 2019.

[7] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[9] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient cnn architecture design," in *Proc. ECCV*, Sep. 2018, pp. 116–131.

[10] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. ICLR*, 2017, pp. 1–13.

[11] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Proc. NIPS*, 2017, pp. 2178–2188.

[12] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4340–4349.

[13] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *Proc. ICLR*, 2018, pp. 1–14.

[14] A. Graves, "Adaptive computation time for recurrent neural networks," 2016, *arXiv:1603.08983*.

[15] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," 2021, *arXiv:2102.04906*.

[16] C.-F. Chen, Q. Fan, N. Mallinar, T. Sercu, and R. Feris, "Big-little net: An efficient multi-scale feature representation for visual and speech recognition," in *Proc. ICLR*, 2019, pp. 1–20.

[17] Y. Chen *et al.*, "Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution," in *Proc. ICCV*, Oct. 2019, pp. 3435–3444.

[18] Z. Xie, Z. Zhang, X. Zhu, G. Huang, and S. Lin, "Spatially adaptive inference with stochastic feature sampling and interpolation," in *Proc. ECCV*, 2020, pp. 531–548.

[19] L. Yang, Y. Han, X. Chen, S. Song, J. Dai, and G. Huang, "Resolution adaptive networks for efficient inference," in *Proc. CVPR*, Jun. 2020, pp. 2369–2378.

[20] Y. Wang, K. Lv, R. Huang, S. Song, L. Yang, and G. Huang, "Glance and focus: A dynamic approach to reducing spatial redundancy in image classification," in *Proc. NIPS*, 2020, pp. 1–16.

[21] M. Figurnov *et al.*, "Spatially adaptive computation time for residual networks," in *Proc. CVPR*, Jul. 2017, pp. 1039–1048.

[22] X. Dong, J. Huang, Y. Yang, and S. Yan, "More is less: A more complicated network with less inference complexity," in *Proc. CVPR*, Jul. 2017, pp. 5840–5848.

[23] T. Verelst and T. Tuytelaars, "Dynamic convolutions: Exploiting spatial sparsity for faster inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2320–2329.

[24] T.-W. Ke, M. Maire, and S. X. Yu, "Multigrid neural architectures," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6665–6673.

[25] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun, "SBNet: Sparse blocks network for fast inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8711–8720.

[26] S. Kong and C. Fowlkes, "Pixel-wise attentional gating for scene parsing," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1024–1033.

[27] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. ICLR*, 2017, pp. 1–13.

[28] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proc. ICLR*, 2017, pp. 1–20.

[29] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1492–1500.

[30] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016, pp. 1–15.

[31] G. Huang, S. Liu, L. V. D. Maaten, and K. Q. Weinberger, "CondenseNet: An efficient DenseNet using learned group convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2752–2761.

[32] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. CVPR*, Jun. 2018, pp. 6848–6856.

[33] L. Yang *et al.*, "CondenseNet v2: Sparse feature reactivation for deep networks," in *Proc. CVPR*, Jun. 2021, pp. 3569–3578.

[34] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. CVPR*, Jun. 2018, pp. 8697–8710.

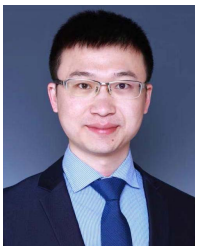
[35] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2820–2828.

[36] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. ECCV*, Sep. 2018, pp. 19–34.

[37] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jan. 26, 2021, doi: 10.1109/TPAMI.2021.3054824.

[38] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. ICCV*, Oct. 2017, pp. 2736–2744.
[39] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *Proc. ECCV*, Sep. 2018, pp. 3–18.
[40] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proc. ECCV*, Sep. 2018, pp. 409–424.
[41] S. Leroux, P. Molchanov, P. Simoens, B. Dhoedt, T. Breuel, and J. Kautz, "IamNN: Iterative and adaptive mobile neural network for efficient image classification," in *Proc. ICML*, 2018, pp. 1–4.
[42] B. E. Bejnordi, T. Blankevoort, and M. Welling, "Batch-shaped channel gated networks," in *Proc. ICLR*, 2020, pp. 1–14.
[43] H. Li, H. Zhang, X. Qi, Y. Ruigang, and G. Huang, "Improved techniques for training adaptive deep networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1891–1900.
[44] A. Kirillov, Y. Wu, K. He, and R. Girshick, "PointRend: Image segmentation as rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9799–9808.
[45] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, Jul. 2017, pp. 2881–2890.
[46] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proc. CVPR*, Jun. 2019, pp. 5693–5703.
[47] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.
[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Jun. 2009, pp. 248–255.
[49] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. ICLR*, 2017, pp. 1–16.
[50] X. Dong and Y. Yang, "Network pruning via transformable architecture search," in *Proc. NIPS*, 2019, pp. 1–12.
[51] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 1–8.
[52] W. Hua, Y. Zhou, C. M. De Sa, Z. Zhang, and G. E. Suh, "Channel gating neural networks," in *Proc. NIPS*, 2019, pp. 1–11.
[53] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. ICCV*, Oct. 2017, pp. 5058–5066.
[54] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. ECCV*, Sep. 2018, pp. 304–320.
[55] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*.
[56] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[57] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
[58] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, 2014, pp. 740–755.
[59] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proc. ICCV*, Nov. 2011, pp. 991–998.
[60] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 4278–4284.

Yizeng Han received the B.S. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2018, where he is currently pursuing the Ph.D. degree in control science and engineering with the Department of Automation, Institute of System Integration. His current research interests include computer vision and deep learning, especially in dynamic neural networks.



Gao Huang (Member, IEEE) received the B.S. degree from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, in 2009, and the Ph.D. degree from the Department of Automation, Tsinghua University, Beijing, in 2015. He was a Postdoctoral Researcher with the Department of Computer Science, Cornell University, Ithaca, USA, from 2015 to 2018. He is currently an Assistant Professor with the Department of Automation, Tsinghua University. His research interests include machine learning and computer vision.

Shiji Song (Senior Member, IEEE) received the Ph.D. degree in mathematics from the Department of Mathematics, Harbin Institute of Technology, Harbin, China, in 1996. He is currently a Professor with the Department of Automation, Tsinghua University, Beijing, China. He has authored over 180 research papers. His current research interests include pattern recognition, system modeling, optimization, and control.

Le Yang received the B.S. degree from the Department of Automation, Northwestern Polytechnical University, in 2015, and the Ph.D. degree from the Department of Automation, Tsinghua University, Beijing, in 2021. He is currently an Assistant Professor with the School of Information and Communications Engineering, Xi'an Jiaotong University. His main research interests include machine learning and computer vision, especially in efficient deep learning and dynamic neural networks.

Yitian Zhang received the B.S. degree from the Department of Material Science and Engineering, Huazhong, University of Science and Technology, Wuhan, China, in 2020. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Northeastern University, Boston, USA. His current research interests include computer vision and deep learning.

Haojun Jiang received the B.S. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2020, where he is currently pursuing the Ph.D. degree with the Department of Automation. His research interests mainly lie on computer vision, especially in efficient neural networks and dynamic neural networks.