Dynamic Dual Gating Neural Networks

Fanrong Li[1,2], Gang Li[1], Xiangyu He[1], Jian Cheng[1,2,3]*
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Future Technology, University of Chinese Academy of Sciences,
[3]CAS Center for Excellence in Brain Science and Intelligence Technology
lifanrong2017@ia.ac.cn, gangli0426@gmail.com, {xiangyu.he, jcheng}@nlpr.ia.ac.cn

Abstract

In dynamic neural networks that adapt computations to different inputs, gating-based methods have demonstrated notable generality and applicability in trading-off the model complexity and accuracy. However, existing works only explore the redundancy from a single point of the network, limiting the performance. In this paper, we propose dual gating, a new dynamic computing method, to reduce the model complexity at run-time. For each convolutional block, dual gating identifies the informative features along two separate dimensions, spatial and channel. Specifically, the spatial gating module estimates which areas are essential, and the channel gating module predicts the salient channels that contribute more to the results. Then the computation of both unimportant regions and irrelevant channels can be skipped dynamically during inference. Extensive experiments on a variety of datasets demonstrate that our method can achieve higher accuracy under similar computing budgets compared with other dynamic execution methods. In particular, dynamic dual gating can provide 59.7% saving in computing of ResNet50 with 76.41% top-1 accuracy on ImageNet, which has advanced the state-of-the-art. Codes are available at https://github.com/lfr-0531/DGNet.

1. Introduction

In recent years, deep convolutional neural networks (CNNs) have made great success in various computer vision tasks, including image recognition [10, 33], object detection [28, 30], and segmentation [25, 40]. However, CNNs achieve impressive accuracy at the cost of huge computational complexity and intensive memory footprint, which pose challenges to the deployment of state-of-the-art models on resource-constrained devices.

Various methods have been proposed to improve the

*Corresponding author.

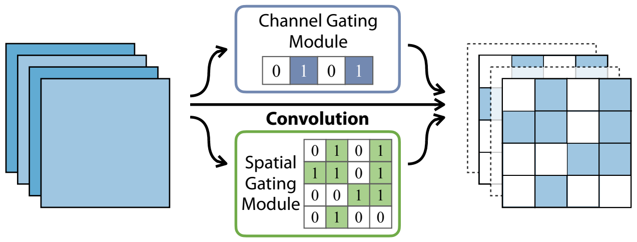


Figure 1: Illustration of dual gating. In each convolutional block, spatial and channel gating modules use the intermediate feature maps to predict the informative features along two separate dimensions. Then the unimportant computations can be skipped at run-time.

computational efficiency of CNN inference. Pruning is a common approach to reduce the model size and computations. Most of the existing methods discard the computations of redundant weights or filters by following specific criteria [13, 26, 18, 43, 41, 11, 12]. However, such methods execute the same calculation for different inputs. This seems to be suboptimal because "hard" inputs usually require more computations. Therefore, dynamic computing has also attracted a number of researchers. Unlike the static pruning methods, these techniques allocate different calculations to different inputs to save computations on those "easy" samples and keep a high accuracy of the overall model. In dynamic computing, methods based on the gating functions are general and flexible approaches and have achieved a good trade-off between computation and accuracy [37, 7, 36, 1, 39].

However, there are limitations in the existing methods. The convolutions extract informative features across spatial and channel dimensions, but previous works only leverage the redundancy from a single point of the networks, affecting the performance. Besides, simply combining those two dimensions can not get a good result, and it is still a challenge to integrate the spatial and channel redundancies for better efficiency. In this paper, we propose dual gating, a new dynamic computing method to reduce the model computations at run-time. CNN with dual gating (DGNet) exploits the redundant features from those two principal axes, spatial and channel. We call such redundancies spatial sparsity and channel sparsity, respectively. In the spatial domain, we try to detect foreground regions and avoid the computations on the background, which can keep the key information in the model. Meanwhile, in the channel domain, we estimate the most relevant channels to the current input and skip the computations of those unimportant ones.

To achieve this, DGNet uses the channel and spatial gating modules to identify the channel and spatial sparsity, respectively. The channel gating module produces a fine-grained binary mask that turns the output feature maps on or off. And we use the spatial gating module to get a tiled binary mask over the spatial dimension to decide regions to be evaluated. Compared with the fine-grained spatial gating, the tile-based spatial mask can keep more informative features under the same computing saving. Both gating modules are lightweight, and the overhead of parameters and computation is negligible. In addition, we propose a simple and efficient method to integrate the gating modules with existing networks and jointly train end to end. As a result, this can not only reduce computations but also allow the salient information to flow freely throughout the model.

Our experiments demonstrate that by plugging our dual gating modules, we can obtain appreciable computation reductions with even higher accuracy than the baseline models on CIFAR-10, ImageNet, and COCO datasets. Compared with the state-of-the-art dynamic computing and static pruning methods, we can consistently improve the performance under similar computing budgets. We then conduct ablation studies to quantitatively evaluate improvements of our proposed methods, including the tiled spatial gating and the efficient integration. Finally, we visualize the trained DGNet and observe that the computation mainly focuses on the target objects and those essential features, making full use of the sparsity in both spatial and channel dimensions.

Our contributions can be summarized as follows:

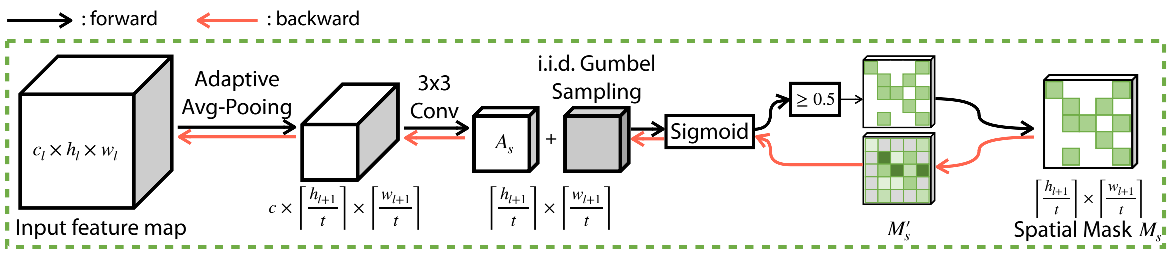
- We propose dual gating, a new dynamic computing method, which leverages spatial and channel sparsities to improve the computation efficiency at run-time.
- We design the spatial and channel gating modules, which can be integrated with commonly used CNN architectures and trade off the accuracy and computation more flexibly. Importantly, the proposed method allows the salient information to flow freely throughout the model.
- We verify the performance on the CIFAR-10, ImageNet, and COCO datasets. The proposed DGNet achieves state-of-the-art results compared with other dynamic computing and static pruning methods.

2. Related Work

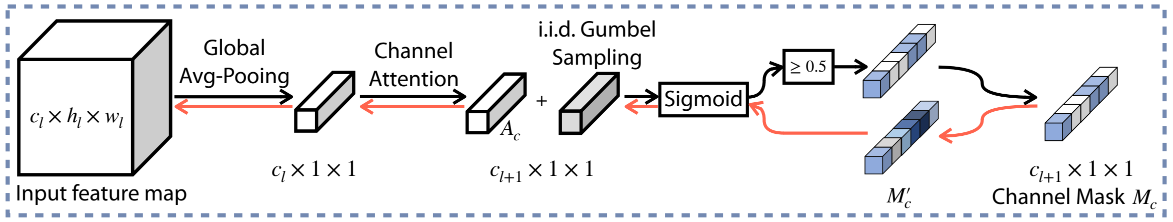
Static Pruning. Pruning methods have been widely studied to tackle the problem of enormous computation in CNNs. Early works [9, 8] focus on pruning the individual weights. This will lead to irregular sparsity that is not friendly to the hardware. In recent years, channel pruning method has become a more promising way, which removes the computations of those unimportant channels based on specific criteria. [18] and [11] use filter norms to approximate the importance of the corresponding channels, while [41] and [13] use the reconstruction errors to guide the channel pruning. FPGM [12] calculates the geometric median of the filters and prunes the filters near it within the same layer. HRank [20] measures the importance based on the rank of the filters. There are some other methods prune via learning-based methods, such as [24, 6]. All the above methods are static and permanently remove the computation of the weights or the whole filters, which may lose the representation capability of the models.

Dynamic Computing. Dynamic computing is a promising alternative to reduce model complexity by skipping part of an existing model based on the input images. Such methods can make decisions based on different criteria to select the part of the network to be executed. BranchyNet [34] and MSDNet [16] use the confidence-based criteria to explore the early exit method, which divides the model into multiple stages and processes the simple inputs with fewer stages in the network than the complex ones. Besides, there are some methods to build an additional policy network to learn the decision of dynamic computation, such as BlockDrop [38], GaterNet[3], and SBNet[29]. Among them, SBNet uses a foreground mask network to predict a tiled spatial mask and guide the sparse computation of all the layers in the model. In contrast, our approach predicts the spatial mask in each block, which is more flexible and can take full advantage of the spatial sparsity.

Compared with the methods above, methods based on gating functions have demonstrated notable generality and applicability, which can be applied to different aspects of the networks. SkipNet [37] and ConvNet-AIG [35] dynamically skip the processing of the whole blocks based on the observation that individual blocks can be removed without interfering with other blocks in the residual networks [10]. But such coarse-grained methods lead to considerable accuracy loss. Some methods exploit spatial sparsity to reduce the computations. [5], [36] and [39] learn a pixel-wise execution mask for each block and only calculate on these predicted locations. CGNet [15] and PGNet [42] are two fine-grained methods that leverage spatial sparsity to allocate different calculations to different output activations. There are also some methods that focus on channel sparsity for efficient inference. FBS [7] ranks the channels and selects the top-k ones as the essential channels. Batch-shaping


: forward
: backward
Adaptive Avg-Pooing
3x3 Conv
i.i.d. Gumbel Sampling
Sigmoid
≥ 0.5
$c_l \times h_l \times w_l$
Input feature map
$c \times \lceil \frac{h_{l+1}}{t} \rceil \times \lceil \frac{w_{l+1}}{t} \rceil$
A_s
$\lceil \frac{h_{l+1}}{t} \rceil \times \lceil \frac{w_{l+1}}{t} \rceil$
M'_s
Spatial Mask M_s

(a) Spatial Gating Module



(b) Channel Gating Module

Figure 2: The details of the Spatial Gating Module and Channel Gating Module for training are illustrated in (a) and (b). Note that, for the spatial gating module, when h and w are not divisible by t, the tiles on the right and bottom edges will be slightly smaller, but the other tiles will still be with size $t \times t$.

[1] uses the Gumbel-Softmax trick to learn the gating decisions, while DGC [32] adaptively selects the connections within group convolution for individual samples on the fly. However, the above methods only explored a single point of the network, while the convolutions extract informative features across spatial and channel dimensions. Our method leverage both spatial and channel sparsities to improve the computation efficiency and empirically verify that exploiting both can consistently improve the performance.

3. Methodology

In this section, we introduce our dual gating method. First, we propose the spatial and channel gating modules in detail. Next, we illustrate how to integrate the dual gating with the existing models. Finally, we present the training losses, which constrain the models to target computations.

Traditional CNNs are composed of multiple sequential convolutional layers, and the output of one layer can be used as the input of the next layer. We assume that a neural network has L layers, and the input of the l-th layer is denoted as X_l. Formally, the l-th convolutional layer can be defined as

$$X_{l+1} = \phi_l(X_l) \tag{1}$$

where $X_l \in R^{c_l \times h_l \times w_l}$, $X_{l+1} \in R^{c_{l+1} \times h_{l+1} \times w_{l+1}}$ are the input and output feature maps, and ϕ_l denotes the operation function of the l-th layer.

3.1. Spatial Gating Module

Spatial sparsity commonly exists in feature maps. As discussed in [19], the information in the background of images contributes less to the final results, which means that not all regions in the spatial dimension are with the same importance. Therefore, we do not need to execute convolutional operations across all regions in the image. Our spatial gating module aims to estimate the informative regions over the spatial dimension.

Figure 2a illustrates the structure of the spatial gating module for training, which splits the output feature maps into tiles on the spatial dimension with tile size $t \times t$ and estimates which tiles needed to be evaluated. First, we aggregate the local information of input feature maps using an adaptive average pooling operation ($AdaAvgPooling$) and then use a standard 3×3 convolution to produce a 2D spatial attention map $A_s \in R^{\lceil \frac{h_{l+1}}{t} \rceil \times \lceil \frac{w_{l+1}}{t} \rceil}$:

$$A_s = f_{3\times3}(AdaAvgPooling(X_l)) \tag{2}$$

where $f_{3\times3}$ denotes the 3×3 convolution. The value $A_s(i,j)$ represents the importance of the tile located at (i,j) position on the output feature map.

Here we choose the tile-based mask for two reasons. First, this can reduce the computations of producing the attention map, making the computing overhead of the gating module negligible. Then considering that the spatial information has a strong local correlation, pixels in the same tile often contain similar information, and using the tile-based mask helps keep more essential features passing throughout different layers, which will be discussed later in detail.

In the inference phase, we can mask the execution of those unimportant tiles directly based on the spatial attention A_s. The binary spatial gating mask M_s can be written as follows:

$$M_s(i,j) = \begin{cases} 1 & A_s(i,j) \geq 0 \\ 0 & Otherwise \end{cases} \tag{3}$$

However, during training, there is a challenge that the spatial gating module can not directly backpropagate as the binary spatial mask is non-differentiable. Here we utilize the Gumbel-Softmax reparameterization technique [27] to relax the discrete binary masks to continuous variables.

Specifically, given the spatial attention A_s, the probability of the execution of the spatial tiles can be defined as $P_s^1 = \sigma(A_s)$, where σ is the sigmoid function. In contrast, the probability that the tiles are not executed is $P_s^0 = 1 - \sigma(A_s)$. Then, the spatial gating M_s can be modeled as binary random variables with probabilities $P(M_s(i,j) = 1) = P_s^1(i,j)$, and the sampling process of M_s can be reparameterized as

$$M_s = \arg\max_k(\log(P_s^k) + g_k), \forall k = 0, 1 \tag{4}$$

where $\{g_k\}_{k=\{0,1\}}$ are i.i.d. random variables that follow the Gumbel distribution.

Because $\arg\max$ is not continuous, Gumbel-Softmax trick replaces the $\arg\max$ with a softmax. Here for the binary special case, the differentiable sample M'_s from the Gumbel-Softmax relaxation can be expressed as follows:

$$M'_s = \frac{\exp(\frac{\log(P_s^1)+g_1}{\tau})}{\Sigma_{k\in\{0,1\}}\exp(\frac{\log(P_s^k)+g_k}{\tau})} = \sigma\left(\frac{A_s+g_0-g_1}{\tau}\right) \tag{5}$$

In addition, the difference between the two Gumbels is a Logistic distribution, and the sampling process can be written as $g_0 - g_1 \overset{d}{=} \log U - \log(1-U)$, where $U \sim Uniform(0,1)$. Then we have

$$M'_s = \sigma\left(\frac{A_s + \log U - \log(1-U)}{\tau}\right) \tag{6}$$

where τ is the temperature of the softmax that controls the difference between the softmax and argmax functions. We choose $\tau = 2/3$ in our experiments as suggested by [27]. In the training phase, to narrow the mismatch between the operation of training and inference, we directly add a step function to M'_s to get a binary mask during the forward pass and use the continuous M'_s to produce the gradient during the backward pass, as shown in Figure 2a.

3.2. Channel Gating Module

As the saliency of a specific channel is not static for different inputs, dynamically selecting important channels for execution is a promising method to reduce the computation while preserving the representation capability of the model as much as possible. Therefore, we build a channel gating module to identify the unimportant channels which can be skipped during inference based on the input images.

The structure of the channel gating module is illustrated in Figure 2b. We first aggregate the spatial information of

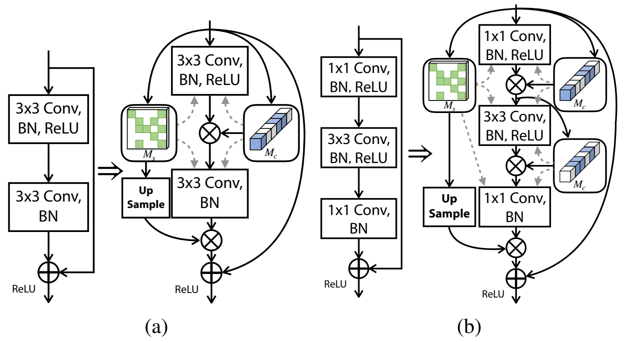


Figure 3: Integration of the dual gating into the residual blocks. (a) Apply dual gating to the basic block. (b) Apply dual gating to the bottleneck block. The dashed arrows denote the inference paths.

the feature maps by using a global average pooling operation ($GlbAvgPooling$) to generate a context descriptor. Then feed the descriptor to a lightweight network to get channel attention $A_c \in R^{c_{l+1}}$. Similar to the SE block [14], the channel attention network composes two sequential fully connected layers and has $\frac{c}{r}$ neurons in the hidden layer to reduce the computations, where r is the reduction ratio. In short, the channel attention can be represented as follows:

$$A_c = W_1 * \delta\left[norm(W_0 * GlbAvgPooling(X_l))\right] \tag{7}$$

where $W_0 \in R^{\frac{c_l}{r} \times c_l}$, $W_1 \in R^{c_{l+1} \times \frac{c_l}{r}}$, δ is the ReLU function, and $norm$ denotes the batch normalization. In our experiments, we set $r = 4$.

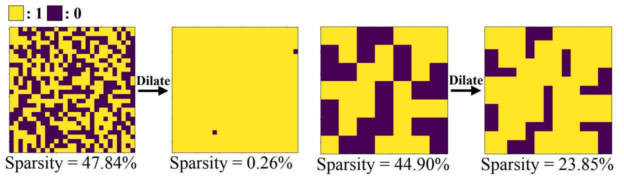
As the same with the spatial gating module, we directly use the channel attention to generate the binary channel mask M_c during inference. In the training phase, the channel gating module also uses the Gumbel-Softmax relaxation to get the continuous mask M'_c for back-propagation. Therefore, the channel gating module can also be integrated with CNNs and learn the salient channels end-to-end.

3.3. Integration with existing models

Our dual gating can be easily integrated with the existing models. Here we use the residual blocks as an example to illustrate the use of our method, as shown in Figure 3. The design principle of the integration is to avoid applying the two gating modules in the same layer during training, so that more salient information can flow freely in the model.

For the spatial gating, each block composes only one gating module, and all the convolutions in the block share the same spatial mask. During training, because of the tile-based gating method, we use an up-sample module to generate a new spatial mask M_s^{up}, then multiply the M_s^{up} to the normalized result of the last convolution operation in the block. In this way, we can mask the less informative regions and only pass the important features to the next block.

In the inference phase, since 3×3 convolutions exist,



(a) Tile size $t = 1$ (b) Tile size $t = 4$

Figure 4: Visualization of the dilated binary spatial mask with $t = 1$ and $t = 4$.

for any position that needs to be executed, the corresponding 3×3 positions in the previous layer also need to be preserved to avoid the mismatch in the input of the 3×3 convolution. One way to solve this problem is to use a dilated mask for the previous layer, which has been exploited in [36]. However, this will lose the sparsity of the previous layer, especially for the fine-grained masks with $t = 1$. As shown in Figure 4, after dilated, a 28×28 fine-grained mask almost loses the sparsity completely. Another approach ignores the mismatch problem and directly shares the same mask among all convolutional layers within the block, which ensures that layers in the same block have the same spatial sparsity. Such method is not suitable for the fine-grained methods due to the large mismatch between the dilated mask and the original one. But it can be an alternative for the tile-based methods, as there is no huge gap between those two masks. In our method, we directly use the same tiled mask for convolutional layers in each block. In this way, compared with fine-grained dilation methods, our method can get masks with lower spatial sparsity to achieve similar computation reduction, which can keep more informative features passing throughout the model.

For the channel gating module, we only insert it between two convolutions in the blocks to preserve more information passing to the next block. During training, we directly multiply the binary mask M_c to the activated result of the convolutional layers to mask those unimportant channels. In the inference phase, the binary mask can be viewed as the output sparsity of the previous layer of the channel gating module, and it can also represent the input sparsity of the later layer.

3.4. Training Loss

To encourage the gating mask to be sparse, we introduce a sparse regularization term into the training loss. This guides the average FLOPs of the dynamic model to a specific target ratio. Let T_d denote the target rate. Then the sparse regularization is defined as:

$$L_{spar} = \left(\frac{\sum_{l=1}^{L} F_l}{\sum_{l=1}^{L} F_{l,ori}} - T_d \right)^2 \tag{8}$$

where F_l denotes the average FLOPs of the l^{th} block including the computation of the gating modules, and $F_{l,ori}$ represents the FLOPs of the l^{th} original residual block.

Initialization is another important issue for training dynamic models. Without proper initialization, the spatial sparsity often converges to a suboptimal state, where exist layers with all the gates on or off. [36] adds a bound loss to guide the early optimization. Here we also introduce a bound regularization to constrain both spatial and channel sparsity to a target budget $\sqrt{T_d}$ at the beginning. And during training, we will relax the constraint to be within the range of $[p\sqrt{T_d}, 1-p(1-\sqrt{T_d})]$. The lower and upper bound regularization terms are:

$$\begin{aligned} L_{b,low} &= \sum_{l=1}^{L} \sum_{k\in\{s,c\}} \max(0, p\sqrt{T_d} - |M_k^l|_d)^2 \\ L_{b,up} &= \sum_{l=1}^{L} \sum_{k\in\{s,c\}} \max(0, p(1-\sqrt{T_d}) - 1 + |M_k^l|_d)^2 \end{aligned} \tag{9}$$

where $|\cdot|_d$ calculates the density of the binary masks, and we use an exponential annealing $p = \exp(-\alpha \cdot epoch)$ to gradually loose the bound. We choose $\alpha = 0.02$ for CIFAR-10 and $\alpha = 0.05$ for ImageNet.

Therefore, the training loss is:

$$L = L_{task} + \lambda L_{spar} + \gamma(L_{b,low} + L_{b,up}) \tag{10}$$

where λ and γ are the weights for the regularization terms and L_{task} is the task specific training loss.

4. Experiments

In this section, we validate the effectiveness of our proposed DGNets on the standard benchmarks: CIFAR-10[17], ImageNet[4], for image classification, and COCO 2017[23] for object detection. We first compare our results with other dynamic methods as well as static pruning methods. Then carry out ablation studies to evaluate different aspects of our proposed method. Last, we will end this section with the visualization of the dual gating method.

4.1. Experiment settings

For CIFAR experiments, we use a pertained static model to initialize, and then train DGNets with a batch-size of 128, using an SGD optimizer with a momentum of 0.9 and weight decay of $5e^{-4}$. No weight decay was applied on the gating modules. We start with a learning rate of 0.1, divide it by 10 at epoch 150 and 225 with a total of 300 epochs. For the gating modules, since spatial size in deep layer are always small, which is not suitable for large tile size, we use different tile size in different layers. For ResNet models on CIFAR-10, the tile sizes of layers in the three stages are set to $\{4, 2, 2\}$, and we set $\lambda = 5$ and $\gamma = 1$.

For ResNet models on ImageNet experiments, DGNets are trained with batch-size 256 and learning rate 0.05 for 100 epochs. The learning rate is reduced by the cosine

Table 1: Comparison of accelerated ResNet on CIFAR-10.

Model	Method	Acc. (%)	Acc. ↓(%)	FLOPs
ResNet-20	LCCL [5]	91.43	0.10	3.20E7 (20.3%↓)
	DynConv [36] *	91.62	0.63	2.34E7 (41.2%↓)
	Batch-shaping [1]	91.75	1.00	2.20E7 (46.3%↓)
	DGNet (50%)	92.27±0.15	-0.03	2.28E7 (42.7% ↓)
	DGNet (60%)	91.90±0.23	0.34	1.89E7 (52.4% ↓)
ResNet-32	LCCL [5]	90.74	1.59	4.70E7 (31.2% ↓)
	DynConv [36]	92.57	1.00	3.37E7 (51.9% ↓)
	Batch-shaping [1]	92.80	0.70	3.30E7 (52.2% ↓)
	DGNet (50%)	93.21±0.14	0.01	3.82E7 (43.4% ↓)
	DGNet (60%)	92.96±0.06	0.26	3.08E7 (54.4% ↓)
ResNet-110	LCCL [5]	93.44	0.19	1.63E8 (34.2% ↓)
	SkipNet [37]	93.30	0.30	1.22E8 (50.5% ↓)
	DynConv [36] *	92.82	1.43	1.30E8 (47.1% ↓)
	DGNet (60%)	94.33±0.06	-0.08	1.08E8 (56.1% ↓)
	DGNet (70%)	93.87±0.09	0.38	8.11E7 (67.2% ↓)

* results are reproduced by using their released code.

scheduler. And the tile sizes of layers in the four stages are set to $\{8, 4, 2, 1\}$. Other settings are with the same to CIFAR-10 with a weight decay of $1e^{-4}$. For MobileNet-V2 [31], we use SGD with the momentum of 0.9 and the weight decay of $4e^{-5}$. We start with a learning rate of 0.05 and reduce it by the cosine scheduler with a total of 200 epochs. The tile size in the first bottleneck block is set to 16. And in the sequential blocks, the tile sizes decrease in proportion to the size of feature maps and are set to a minimum of 2. For all ImageNet experiments, we initialize DGNets with pre-trained models and use the same data augmentation strategies with PyTorch official examples.

For COCO experiments, we evaluate dual gating using single-stage object detection of RetinaNet [22] and the two-stage Faster R-CNN with Feature Pyramid Network (FPN) [21]. The average mAP over different IoU thresholds from 0.5 to 0.95 is used for evaluation. ImageNet pre-trained ResNet-50 with dual gating is chosen as the default backbone model. Our training code and the parameters are based on mmdetection [2]. For the gating modules, the tile sizes of layers in the four stages of the backbone model are set to $\{16, 8, 4, 2\}$, and we set $\lambda = 5$ and $\gamma = 0$.

4.2. Image Classification

Results on CIFAR-10. We test our dual gating on ResNet-20, 32, and 110 with different target sparsity: 50%, 60%, and 70%, and we run each experiment three times and report the "mean $\pm$ std". As shown in Table 1, our DGNets can outperform previous dynamic computing methods. Specifically, compared with SkipNet [37], which skips the processing of the whole residual blocks, on ResNet-110, DGNet can achieve a less accuracy drop (-0.05% v.s. 0.30%) with more computational cost reduction (56.1% v.s. 50.5%). Compared with the methods LCCL [5] and DynConv [36], which exploit spatial sparsity, our approach can obtain better accuracy with fewer FLOPs. Compared with the dynamic channel pruning method Batch-shaping [1], our method can also achieve a smaller accuracy drop with fewer FLOPs.

Results on ImageNet. We apply our method to ResNet-18, 34, 50, and MobileNet-V2 with different target sparsities. Table 2 shows the results on the validation set. We first compare dual gating with other state-of-the-art dynamic methods, and we can observe that DGNets achieve better performance. On ResNet-18, DGNet removes 49.4% FLOPs and can achieve even better performance than the baseline model, which has never been achieved by the previous dynamic methods, including dynamic spatial methods LCCL [5] and DynConv [36], dynamic channel method Batch-shaping [1], and fine-grained dynamic spatial method CGNet [15]. On ResNet-34, DGNet obtains 73.01% top-1 accuracy with 59.3% FLOPs reduction, which also outperforms other dynamic methods. Moreover, on ResNet-50, DGNet can also obtain higher accuracy than the baseline model with 59.7% FLOPs reduction, significantly better than the method ConvNet-AIG [35], DynConv [36], and Batch-shaping [1]. We then compare dual gating with the state-of-the-art static pruning methods. Since dynamic computing can keep the representation capability of the model as much as possible, dual gating shows better results than the static pruning methods across different networks. Similar improvements can also be found when applying the dual gating to MobileNet-V2. This demonstrates the effectiveness of our proposed dual gating, leveraging redundancies from both spatial and channel dimensions. It can not only achieve high accuracy of the model, but also reduce the calculation of the model as much as possible.

To demonstrate the inference speedup, we evaluated the real speedup of our method on two different hardware environments: a CPU (I7-6700, 24G RAM, and Ubuntu 16.04 OS) and an embedded FPGA accelerator (Ultra96 SoC)[1]. Table 3 shows the speedup of DGNet. There is a gap between the theoretical speedup and CPU speedup, mainly because that the non-zero indexing leads to inefficient computation. But FPGA accelerators can implement this process efficiently.

4.3. MS COCO Object Detection

To show the generalization ability of our proposed dual gating method, we conduct experiments on object detection. MS COCO dataset is used for evaluation, and the average FLOPs in the backbone network over the whole validation set is used to evaluate the model complexity. We set the target sparsity of the backbone model to be 60%. During the evaluation, input images are resized to 800 pixels in the shorter edge. The results are shown in Table 4. We can observe that, on both RetinaNet and Faster-RCNN, DGNet can save 59.29% of FLOPs of the backbone network with a

[1]More details are described in the appendix.

Table 2: Comparison of accelerated ResNet and MobileNet-V2 on ImageNet.

Model	Method	Dynamic	Top 1 accuracy (%)			Top 5 accuracy (%)			FLOPs
			Baseline	Accelerated	Acc ↓	Baseline	Accelerated	Acc ↓	
ResNet-18	FPGM [12]	✗	70.28	68.41	1.87	89.63	88.48	1.15	1.05E9 (41.8% ↓)
	LCCL [5]	✓	69.98	66.33	3.65	89.24	86.94	2.30	1.23E9 (34.6% ↓)
	CGNet [15]	✓	69.20	68.80	0.40	-	-	-	0.98E9 (48.2% ↓)
	DynConv [36] *	✓	69.76	66.97	2.79	89.08	87.22	1.86	1.08E9 (41.5% ↓)
	Batch-shaping [1]	✓	69.70	68.75	0.95	-	-	-	1.05E9 (42.0% ↓)
	DGNet (50%)	✓	69.76	**70.12**	**-0.36**	89.08	**89.22**	**-0.14**	9.54E8 (**49.4%** ↓)
	DGNet (60%)	✓	69.76	69.38	0.38	89.08	88.94	0.14	7.88E8 (58.2% ↓)
ResNet-34	FPGM [12]	✗	73.92	72.63	1.29	91.62	91.08	0.54	2.17E9 (41.1% ↓)
	LCCL [5]	✓	73.42	72.99	0.43	91.36	91.19	0.17	2.78E9 (24.8% ↓)
	CGNet [15]	✓	72.40	71.30	1.10	-	-	-	1.83E9 (50.5% ↓)
	DynConv [36] *	✓	73.31	71.75	1.56	91.42	90.47	0.95	2.01E9 (44.1% ↓)
	Batch-shaping [1]	✓	73.30	72.55	0.75	-	-	-	1.75E9 (52.2% ↓)
	DGNet (60%)	✓	73.31	**73.01**	**0.30**	91.42	**90.99**	**0.43**	1.50E9 (**59.3%** ↓)
	DGNet (70%)	✓	73.31	71.95	1.36	91.42	90.46	0.96	1.21E9 (67.2% ↓)
ResNet-50	FPGM [12]	✗	76.15	74.83	1.32	92.87	92.32	0.55	1.91E9 (53.5% ↓)
	HRank [20]	✗	76.15	74.98	1.17	92.87	92.33	0.54	2.30E9 (41.3% ↓)
	ConvNet-AIG [35]	✓	76.13	75.25	0.88	92.88	92.39	0.49	2.56E9 (32.6% ↓)
	DynConv [36] *	✓	76.13	74.40	1.73	92.86	91.83	1.03	2.25E9 (42.4% ↓)
	Batch-shaping [1]	✓	76.10	74.40	1.70	-	-	-	1.75E9 (57.2% ↓)
	DGNet (60%)	✓	76.13	**76.41**	**-0.28**	92.86	**93.05**	**-0.19**	1.65E9 (**59.7%** ↓)
	DGNet (70%)	✓	76.13	75.12	1.01	92.86	92.34	0.52	1.31E9 (67.9% ↓)
MobileNet-V2	MetaPruning [24]	✗	71.88	71.20	0.68	-	-	-	2.17E8 (22.5% ↓)
	DGC [32]	✓	72.00	70.70	1.30	90.60	89.80	0.80	2.45E8 (18.3% ↓)
	DGNet (50%)	✓	71.88	**71.62**	**0.26**	90.27	**90.05**	**0.22**	1.60E8 (**44.0%** ↓)

* results are reproduced by using their released code.

Table 3: Comparison of real speedups on CPU and FPGA.

Model	Method	Top 1 Acc.	FLOPs	CPU Speedup	FPGA Speedup
ResNet-18	Baseline	69.76%	1.89E9	-	-
	DGNet (60%)	69.38%	7.88E8	1.24	1.87
ResNet-34	Baseline	73.31%	3.69E9	-	-
	DGNet (60%)	73.01%	1.50E9	1.21	1.93
	DGNet (70%)	71.95%	1.21E9	1.41	2.26

neglectable accuracy drop, which demonstrates the generalization ability of the proposed dynamic computing method.

4.4. Ablation Study

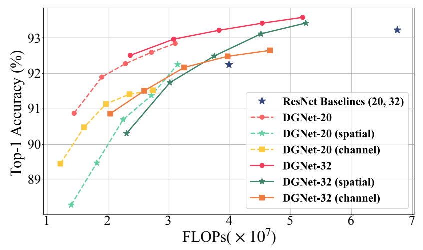
We conduct a series of ablation studies aiming to evaluate different aspects of our proposed dual gating method.

Importance of dual gating. To demonstrate the effectiveness of the dual gating method, we experimentally compare DGNet with the other two variants, DGNet (spatial) and DGNet (channel), which only use spatial or channel gating modules, respectively. On CIFAR-10 dataset, we use ResNet-20 and ResNet-32 with different target densities $T_d \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$. Figure 5a shows the experimental results. We can observe that DGNet (spatial) can achieve better performance than DGNet (channel) under high computation budgets. This is because DGNet (spatial) only leverages the spatial sparsity, but keeps all the kernels during inference, preserving the model representation capacity. However, with low computation budgets, DGNet (channel) can outperform DGNet (spatial). We suspect this is because the spatial redundancy is limited, and the methods that only leverage spatial sparsity are not suitable for an extremely low computation budget. Nevertheless, DGNets outperform both two methods, as the dual gating method can combine advantages of the other two variants and trade-off the accuracy and computation with more freedom.

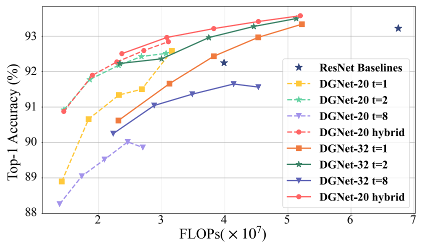
Table 4: Object detection results (bounding box AP) on COCO 2017.

Model		$AP_{0.5:0.95}$	$AP_{0.5}$	$AP_{0.75}$	FLOPs
RetinaNet	Baseline	36.5	55.4	39.1	7.05E10
	DGNet (60%)	36.1	55.3	38.1	2.87E10
Faster-RCNN	Baseline	37.4	58.1	40.4	7.05E10
	DGNet (60%)	37.2	58.3	40.1	2.86E10

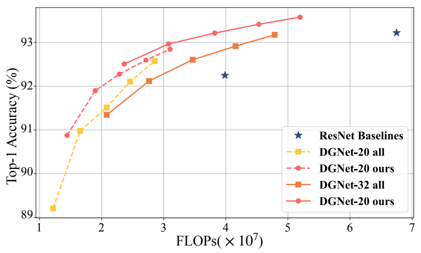
Different spatial tile sizes. Besides the investigation above, we conduct an ablation study of the DGNet accuracy with different tile sizes. Experiments still use ResNet-20 and ResNet-32 with target densities $T_d \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$. Figure 5b shows the accuracy of DGNets with different spatial tile sizes, i.e., 1×1, 2×2, 8×8, and the hybrid setting we used. When we choose 1×1 tile size, the spatial gating becomes a fine-grained method. Compared with the others with tile size 2×2 and the hybrid setting, fine-grained gating leads to an apparent decline in



(a) Different gating methods



(b) Different spatial tile sizes



(c) Different integration strategies

Figure 5: Ablation results on CIFAR-10. (a) shows the effect of our dual gating method on ResNet-20 and ResNet-32. (b) gives the comparison with different spatial tile sizes, where 'hybrid' means the strategies in our proposed method. Other settings use the same tile size in all layers. (c) shows the impact of different integration strategies, where 'all' means the method that inserts spatial gating modules to all convolutional layers in residual blocks.

accuracy, as the fine-grained spatial method will lose more informative features as discussed in Section 3.3. However, the large tile size is not always a good choice, and it can also lead to poor performance. DGNets with 8×8 tile size obtains a larger accuracy drop than other settings. This is because the 8×8 tile size is too large for the features in deep layers, and each tile will compose both important and unimportant features, affecting the estimation of the spatial gating module. Therefore, the large tile size is only suitable for large images, and our proposed hybrid tile setting is a better choice.

Effectiveness of the integration method. To demonstrate the effectiveness of our proposed integration principle, we compare it with a simple integration method, which is a natural way to insert spatial gating modules to each convolutional layer. Figure 5c gives the experimental results of ResNet-20 and ResNet-32 on CIFAR-10. We can observe that our method consistently improves the performance over different sparsities. This is because methods that apply both spatial and channel gating modules to the same convolutional layer will lose much more information than our approach, hindering the flow of salient information across the model.

4.5. Gating Visualization

To demonstrate that dual gating can keep informative features flow throughout the models, we visualize the spatial cost map and the feature maps of the first layer in the first block of ResNet-34 with dual gating, as shown in Figure 6. The spatial cost map indicates the number of blocks executed at each tile, and we can obtain it by upsampling the spatial gating masks to the same size as images and accumulate them all. As we can see, the spatial gating module can focus the computation on the essential tiles, and the channel gating modules can also select the salient features. This is also the reason that our dual gating can achieve better performance.

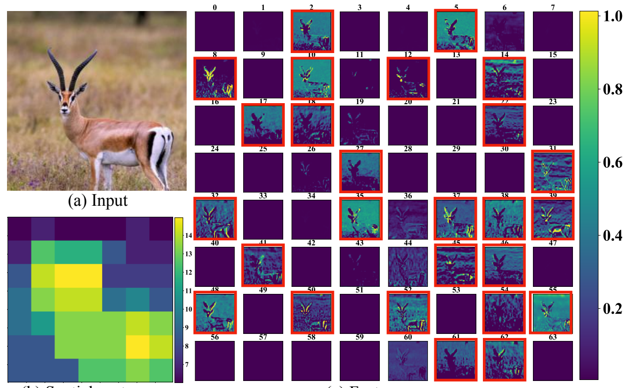


Figure 6: Input image (a) and visualization of the spatial cost map (b) and feature maps (c). Dual gating selects channels with red borders.

5. Conclusion

In this paper, we propose dual gating, a new dynamic computing method, which leverages both spatial and channel sparsities to tradeoff accuracy and computational complexity. Specifically, our method uses spatial and channel gating modules, which can be easily integrated with commonly used CNN architectures and trained end-to-end, to skip redundant computations at run-time. The proposed dual gating method is validated on a variety of computer vision tasks with various network architectures, showing state-of-the-art performance compared with other dynamic computing and static pruning methods.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China (No.61972396), National Key Research and Development Program of China (No. 2020AAA0103402), the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDA27040300).

References

[1] Babak Ehteshami Bejnordi, Tijmen Blankevoort, and Max Welling. Batch-shaping for learning conditional channel gated networks. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 3, 6, 7

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 6

[3] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. You look twice: Gaternet for dynamic filter selection in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9164–9172, 2019. 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 5

[5] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1895–1903, 2017. 2, 6, 7

[6] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 760–771, 2019. 2

[7] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng zhong Xu. Dynamic channel pruning: Feature boosting and suppression. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[8] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 1387–1395, 2016. 2

[9] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1135–1143, 2015. 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1, 2

[11] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, page 2234–2240, 2018. 1, 2

[12] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4335–4344, 2019. 1, 2, 7

[13] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1398–1406, 2017. 1, 2

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018. 4

[15] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G. Edward Suh. Channel gating neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1886–1896, 2019. 2, 6, 7

[16] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[18] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 2

[19] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6459–6468, 2017. 3

[20] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1526–1535, 2020. 2, 7

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 6

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017. 6

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755, 2014. 5

[24] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3295–3304, 2019. 2, 7

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. 1

[26] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5068–5076, 2017. 1

[27] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017. 4

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 1

[29] Mengye Ren, Andrei Pokrovsky, Bin Yang, and Raquel Urtasun. Sbnet: Sparse blocks network for fast inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8711–8720, 2018. 2

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015. 1

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. 6

[32] Zhuo Su, Linpu Fang, Wenxiong Kang, Dewen Hu, Matti Pietikäinen, and Li Liu. Dynamic group convolution for accelerating convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 138–155, 2020. 3, 7

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 1

[34] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016. 2

[35] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 730–741, 2018. 2, 6, 7

[36] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2317–2326, 2020. 1, 2, 5, 6, 7

[37] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 420–436, 2018. 1, 2, 6

[38] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8817–8826, 2018. 2

[39] Zhenda Xie, Zheng Zhang, Xizhou Zhu, Gao Huang, and Steve Lin. Spatially adaptive inference with stochastic feature sampling and interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 531–548, 2020. 1, 2

[40] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016. 1

[41] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9194–9203, 2018. 1, 2

[42] Yichi Zhang, Ritchie Zhao, Weizhe Hua, Nayun Xu, G. Edward Suh, and Zhiru Zhang. Precision gating: Improving neural network efficiency with dynamic dual-precision activations. In *International Conference on Learning Representations, (ICLR)*, 2020. 2

[43] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 875–886, 2018. 1