Context-aware Dynamic Block

Yingcheng Su, Shunfeng Zhou, Yichao Wu, Xuebo Liu, Tian Su, Ding Liang, Junjie Yan
SenseTime Research
{suyingcheng, zhoushunfeng, wuyichao, liuxuebo, liangd, yanjunjie}@sensetime.com
tiansu@seu.edu.cn

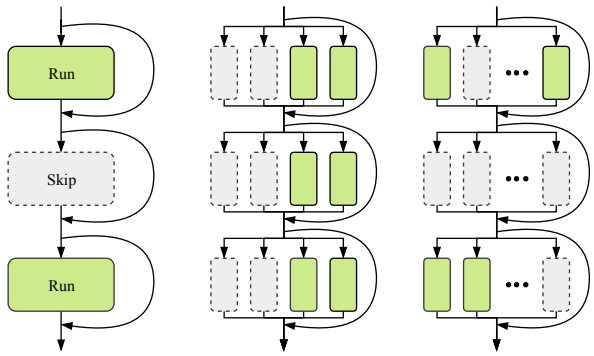
Abstract

Although deeper and larger neural networks have achieved better performance nowadays, the complex network structure and increasing computational cost cannot meet the demands of many resource-constrained applications. An effective way to address this problem is to make use of dynamic inference mechanism. Existing methods usually choose to execute or skip an entire specific layer through a switch structure, which can only alter the depth of the network. In this paper, we propose a dynamic inference method called Context-aware Dynamic Block (CDB), which provides more path selection choices in terms of network width and depth during inference. The execution of CDB is determined by a context-aware group controller, which can take into account both historical and object category information. The proposed method can be easily incorporated into most modern network architectures. Experimental results on ImageNet and CIFAR-100 demonstrate the superiority of our method on both efficiency and overall classification quality. To be specific, we integrate CDB block into ResNet-101 and find that our method significantly outperforms their counterparts and saves 45.1% FLOPs.

1. Introduction

Deep neural networks (DNNs) have dominated the field of computer vision due to the superior performance in all kinds of tasks. Since DNNs generally adopt static designs [12, 13], there exist severe conflicts between the redundant and insufficient number of parameters needed for simple samples and difficult samples respectively. Meanwhile, it is a tendency that the network architecture is becoming deeper and more complex [20, 19, 7] to yield higher accuracy. However, the great computing expense of deeper networks contradicts the demands of many resource-constrained applications, such as mobile platforms, which prefer lightweight network [18, 14] to acquire a short response time.

An elegant solution is to make use of dynamic inference mechanism [22, 21, 1], reconfiguring the inference path according to the input sample adaptively to meet a better accuracy-efficiency trade-off. Prevalent dynamic inference techniques can be roughly divided into two categories: layer-wise [22, 21] and channel-wise [1]. Layer-wise methods, as shown in Fig. 1a, generally employ a gate to determine the execution status of a specified block unit consisting of several layers. However, this method can only alter the depth of the network. Moreover, the design of the gate module always neglects the context information, and it is too rough to make decisions based on the output of the connected previous layer. Channel-wise method, as shown in Fig. 1b, manually adjusts the number of active channels of the same model based on the resource budgets. Strictly speaking, it is not a dynamic process while choosing the active channels. In addition, setting the same fixed number of active channels for each block in the network



(a) Block-wise (b) Channel-wise (c) CDB

Figure 1: Comparison with existing dynamic inference methods. Block-wise methods [21, 22] roughly choose to execute or skip an entire block and channel-wise method [1] can select network's width according to resource constraints, but it can not adjust network's width for different samples, while our method split each block to multiple channel groups so that the input sample can choose a more efficient and faster path during inference.

negatively affects the flexibility of the dynamic inference mechanism. In this paper, we attempt to combine the merits of these two methods, and propose a novel dynamic inference method called Context-aware Dynamic Block (CDB), which can provide more path selection choices in terms of network width and depth during inference.

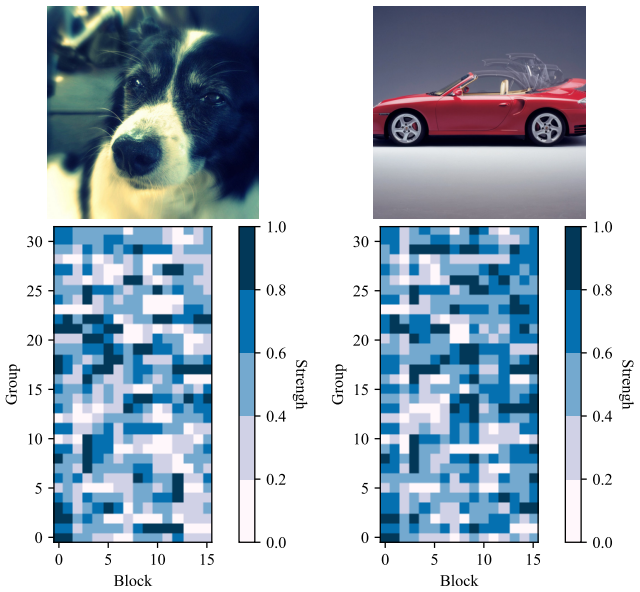
Fig. 2 visualizes the activation of ResNet-50 [7] trained on ImageNet [4] while feeding with different images. we

We note that the activated channels for an input image only take a part of the whole neural network and images with different contents have different respond areas. Fig. 2 visualizes the activation of ResNet-50 [7] trained on ImageNet [4] while feeding with different images. We note that the importance of each group channel for different samples is not equal. The activated channels vary with different samples indicating that the inference path should be chosen dynamically according to the current input.

Motivated by this, we propose a novel dynamic inference method called Context-aware Dynamic Block (CDB) which conquers these drawbacks inherently. Unlike existing dynamic inference methods, the proposed CDB predefines several channel groups at each level of the network to have more path selection choices for different instances, as shown in Fig. 1c. In order to obtain more reasonable network structure, we carefully design context-aware group controller to take both historical information and object category into consideration. In addition, we introduce resource-constrained loss which integrates FLOPs constraint into the optimization process to make the computational complexity controllable. Noting that the proposed CDB can be incorporated into most modern network architecture.

The contributions are summarized as follows.

- We propose a novel dynamic inference method called *Context-aware Dynamic Block*. CDB sets multiple channel groups at each block of the network so that each input sample can have more path selection choices during the inference.
- We carefully design a module called *Context-aware Group Controller*, which takes into account both historical and object category information. Therefore we can obtain better network structure optimization results.
- We introduce resource-constrained loss which integrates FLOPs constraint into optimization target to control the computational complexity of the target network.
- Experimental results demonstrate the superiority of our method on both efficiency and overall classification quality. To be specific, we integrate context-aware dynamic block into ResNet-101 and find that our method outperforms their counterparts and saves 45.1% FLOPs.



(a) Dog (b) Car

Figure 2: Visualization of the activation of ResNet-50 trained on ImageNet. We group the feature maps of each block into 32 groups and visualize their mean values after normalization. The activated groups vary with different samples.

2. Related Work

An efficient and fast network structure is very important for numerous computer vision tasks. In this section, we present a brief introduction to related works including network architecture design, model pruning, distilling and adaptive computation graph.

It also generates significant computational costs. However, lightweight architecture plays important roles in various real scenarios, such as MobileNet [10, 18] and ShuffleNet [25, 14]. Recently, model structure learning for deep learning models has attracted increasing attention. Several methods [2, 27, 26] have been explored to learn network architectures automatically, and thus achieve better or comparable performance in comparison to handcrafted design structures.

Model Pruning and Distilling. In order to trade off between performance and latency, model pruning and distilling were proposed in recent years. Deep Compression [6] can reduce the number of connections by 9x to 13x without loss of performance on VGG16, while methods using sparse connections require specific software and hardware

to speed up. [8, 15, 23] try to prune channels and network depth to reduce computational complexity without accuracy drop. Model distilling methods [9, 16] use a large model to transfer the learned knowledge to small model. Experimental results show these small models distilled from large models perform better than trained from scratch.

Adaptive Computation Graph. Unlike static pruning, adaptive computation graph aims to change runtime inference topology of model based on samples, which can be divided into two classes: block-wise and channel-wise based methods. SkipNet [21] and ConvNet-AIG [22] apply a block-wise dynamic mechanism to networks like ResNet and reduce abundant computational cost with consistent accuracy. [1] proposed a channel-wise method which can select different channels' number during inference. However, the pre-defined width multipliers negatively affect the flexibility of the dynamic inference mechanism.

3. Methodology

In this section, we introduce the proposed context-aware dynamic block in detail. It includes three parts: the structure of dynamic block, context-aware group controller and the optimization approach.

3.1. Dynamic Block

As shown in Fig. 2, the activated channels of a trained DNN vary in terms of different samples, which motives us to rethink the role of different channels while representing different objects. Can we remove the unimportant channels (redundant computation) according to different samples in a dynamic way? The intuitive way to do this is to dynamically select the inference path on the fly for different samples according to their feature characteristics.

To achieve this goal, we interpret optimizing the network structure as executing or skipping of each channel during the inference stage. However, too many subdivision of channels is not necessary and numerous channel controllers would increase the computational complexity. In practice, we gather several channels of a block into channel groups, as shown in Fig. 1c. In this way, each channel group responds separately in each block leading the inference paths more diverse and flexible than other dynamic inference methods. We also use context-aware group controller to decide which channel groups will be executed. We will introduce the details in Sec. 3.2.

A key issue is how to split one block into N channel groups. The guiding principle is the total FLOPs and parameter quantities of these channel groups need to be consistent with the original block for fair comparison. Taking MobileNet V2 [18] as an example, each block contains one pointwise convolution with expansion ratio E, one depthwise convolution and one squeezed pointwise convolution. we simply divide the origin block into N channel groups, the expansion ratio of each group is set to E/N. Thus the sum of every channel groups' FLOPs and parameters are approximate to the original block as depthwise convolution takes up most of FLOPs and parameters. While for ResNet, it is not that straightforward. Suppose the shape of the input tensor is $H \times W \times C_{in}$, convolution with kernel size k and stride s produces C_{out} channels. We can obtain the FLOPS and parameters

$$\begin{aligned} \text{FLOPS}_{origin} &= H \times W \times C_{in} \times C_{out} \times k^2/s^2 \\ \text{Params}_{origin} &= C_{in} \times C_{out} \times k^2 \end{aligned} \tag{1}$$

If we simply set the number of channels of each group to $1/N$ of the origin blocks, then the new FLOPS and parameters are

$$\begin{aligned} \text{FLOPS}_{new} &= H \times W \times C_{in} \times C_{out} \times k^2/(s^2 \times N) \\ \text{Params}_{new} &= C_{in} \times C_{out} \times k^2/N \end{aligned} \tag{2}$$

This result is unfair to subsequent comparisons. Therefore, we carefully design the channel groups split strategy for ResNet, the detailed structure of CDB-50 based on ResNet-50 is shown in Tab. 1. The split strategy of our CDB-50 is similar to ResNeXt [24].

3.2. Context-aware Group Controller

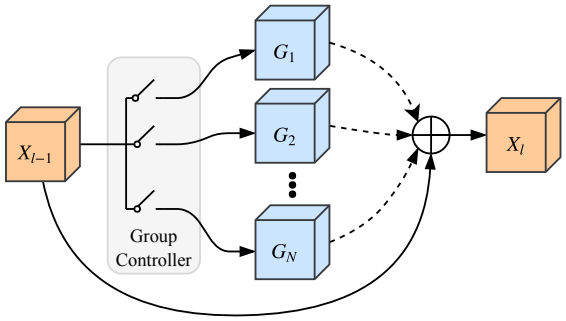
We propose the context-aware group controller to predict the status of each channel group (on/off). It is the inference paths optimizer of context-aware dynamic block. Suppose we split l-th block into N groups, as shown in Fig. 3a, the output of l-th block is the combination of an identity connection and N groups which are controlled by a group controller. Formally,

$$\boldsymbol{X_l} = \boldsymbol{X_{l-1}} + \sum_{n}^{N} s_{l,n} \mathcal{F}_{l,n}(\boldsymbol{X_{l-1}}) \tag{3}$$

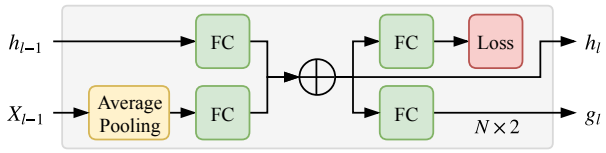
where $\boldsymbol{X_l}$ is output of l-th block, $s_{l,n} \in \{0, 1\}$ refers to the off/on status and $\mathcal{F}_{l,n}(\boldsymbol{X_{l-1}})$ refers to the output of n-th channel group of l-th block.

3.2.1 Contextual information embedding

In the inference phase, the network will automatically choose the paths based on the input features and historical information. There are two problems to be solved here. The first one is the embedding of the input features. Considering that the control modules make decisions based on the global spatial information, we achieve this process by applying global average pooling to compress the high dimension features to one dimension along channels. We further use a fully connected layer followed by an activation layer to map the pooling features to a low-dimensional space. Specifically, $\boldsymbol{X_{l-1}} \in \mathbb{R}^{H \times W \times C}$ represents the input



(a) The overview of our gate mechanism



(b) Context-aware group controller

Figure 3: The framework of the context-aware group controller.

features of l-th block, we calculate the c-th channel statistic by:

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_{i,j,c}^{l-1} \tag{4}$$

The final embedding feature $\boldsymbol{V_{l-1}} \in \mathbb{R}^d$ is:

$$\boldsymbol{V_{l-1}} = \mathcal{F}(\boldsymbol{z}, \boldsymbol{W_1}) = \sigma(\boldsymbol{W_1 z}) \tag{5}$$

where $\boldsymbol{z} = [z_1, z_2, \cdots, z_c]$, $\boldsymbol{W_1} \in \mathbb{R}^{d \times c}$, σ is the ReLU [5] function. The second problem is the aggregation of historical information. We first employ a fully connected layer followed by ReLU function to map the historical hidden features into the same subspace with $\boldsymbol{V_{l-1}}$. Then we perform an addition operation on the hidden feature and $\boldsymbol{V_{l-1}}$ to get the result of the current state. Formally,

$$\begin{aligned} \boldsymbol{h'_{l-1}} &= \mathcal{F}(\boldsymbol{h_{l-1}}, \boldsymbol{W_2}) = \sigma(\boldsymbol{W_2} * \boldsymbol{h_{l-1}}) \\ \boldsymbol{h_l} &= \boldsymbol{V_{l-1}} + \boldsymbol{h'_{l-1}} \end{aligned} \tag{6}$$

where $\boldsymbol{W_2} \in \mathbb{R}^{d \times d}$, σ represents the ReLU function. Bias terms are omitted for simplicity. The status predictions of each group at l-th block is made through $\boldsymbol{h_l}$ by using a softmax trick which we will introduce in section 3.3.

3.2.2 Supervised learning of group controller

Deep CNNs compute feature hierarchies at every layer, producing feature maps with different depths and resolutions. This can also be considered as a feature extraction process from coarse to fine. The proposed dynamic group network has several groups at each block, and we hope that different classes would have varied paths. However, if the path selection mechanism is trained only by optimizing the classification loss at the last layer, it will be difficult for the group controller to learn the category information. As a result, it may select the paths leading to sub-optimal network structure. To solve this problem, we introduce category loss to every group controller to enable all of them to become category-aware. Considering that predicting each class as a different category by the controller is computationally expensive and improper, we cluster samples into fewer categories than original classes. For the ImageNet dataset [4], we cluster all the 1000 classes samples into 58 big categories with the help of hierarchical structure of ImageNet provided in [4]. The number of 58 is to balance the number of samples in different categories. For the CIFAR-100 dataset [11], it groups the 100 classes into 20 superclasses. We use the 20 superclasses as the big categories directly. Then cross entropy loss is employed to supervise each group controller as shown in Fig. 3b. Formally, the category loss of l-th controller can be written as follow

$$\mathcal{L}_l = \sum_{j}^{K} k_j \log(p_j) \tag{7}$$

where p_j represents the probability of j-th class. $k_j \in \{0, 1\}$ refers to the ground truth, K indicates the number of categories. It is worth noting that the loss weights of each block's controller are not always equal since the features of different layers have different semantic information. Deep layers have stronger semantic information than shallow layers. For CDB-50 shown in Tab 1, there are four stages composed of 3, 4, 6, 3 stacked blocks respectively, resulting in 16 controllers. The loss weight of the first stage is set to 0.0001, and it will increase by a factor of 10 in the next stages. CDB-101 follows the same principle. The loss of supervised controller can be represented as follows

$$\mathcal{L}_{ctg} = \sum_{l}^{L} \alpha_l \mathcal{L}_l \tag{8}$$

where α_l denotes the loss weight of l-th controller and L denotes the number of blocks. The category information will be removed after training, so it will not generate any extra computational burden during testing.

The group controller is desirable for its lightweight characteristic during the optimization of network structure. The dimension of the hidden layer d is set to 32 in all experiments. This setting generates only little computational cost and can be omitted compared to the whole computation of the network. If we take ResNet-50 as the backbone, the to-

tal 16 group controllers only generate about 0.02% FLOPs of the original ResNet-50.

3.3. Optimization

Softmax Trick with Gaussian Noise. A simple way to decide whether to execute or omit a channel group is to take the choice as a binary classification problem and choose the path with the maximum score of the corresponding prediction scores. Yet we find that the predicted execution status of a group channel would switch on and off frequently in this situation, which is not desirable. In order to increase the stability of the training process of our network, we add Gaussian noise $\Delta \sim \mathcal{N}(0,1)$ before the softmax layer of the controller module. Formally, We set N as the number of channels groups. Let $\boldsymbol{g_l} = \boldsymbol{W_3 h_l} + \boldsymbol{b_3}$, $\boldsymbol{W_3} \in \mathbb{R}^{2N \times d}$, $\boldsymbol{b_3}$ is the bias term. $\boldsymbol{g_l}$ is then reshaped to $N \times 2$ for the final predictions. The activation can be written as follows:

$$\boldsymbol{s_l} = \arg\max(\text{softmax}(\boldsymbol{g_l} + \Delta)) \tag{9}$$

where $\boldsymbol{s_l} = [s_{l,1}, s_{l,2}, \cdots, s_{l,N}]$ refers to the status of each channel groups of l-th block.

Resource-constrained Loss. The resource constraint comes from two aspects. The first one is the execution rate of each block's group controllers in a mini-batch. We want the network to choose the important channel groups of each block automatically. A execution rate constraint is employed to encourage each block to reach the target rate e. Let z_l denotes the execution rate of l-th block within a mini-batch, we define the execution rate loss of l-th block as:

$$\hat{\mathcal{L}}_l = (e - z_l)^2 \tag{10}$$

The total execution rate loss can be written as follow:

$$\mathcal{L}_{exec} = \sum_{l}^{L} \hat{\mathcal{L}}_l \tag{11}$$

The other constraint is the total FLOPs. Even if we have limited the execution rate, it can not meet the desired FLOPs as we expect yet. Thus, we explicitly add the target FLOPs rate into loss to obtain specific FLOPs during training. The FLOPs loss can be formalized as

$$\mathcal{L}_{flops} = (\frac{f}{f_{total}} - r)^2 \tag{12}$$

where f_{total} and f represent the full FLOPs and the actual execution FLOPs of the network respectively, and r denotes the target FLOPs rate. We set $e = r$ in all experiments since they have strong positive correlation and similar values. f is computed from each mini-batch during training by

$$f = \sum_{l}^{L} \sum_{i}^{N} s_{l,i} f_{l,i} \tag{13}$$

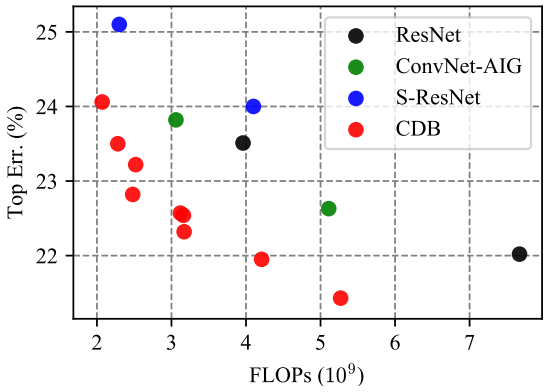


Figure 4: Top-1 error vs. FLOPs on ImageNet. The proposed CDB models outperform other methods by a large margin in both computational cost and accuracy.

where $f_{l,i}$ indicates the FLOPs of i-th group at l-th block of the network. $s_{l,i} \in \{0, 1\}$ stands for the status of the corresponding group. Thus, the resource-constrained loss is defined as

$$\mathcal{L}_{res} = \mathcal{L}_{exec} + \mathcal{L}_{flops} \tag{14}$$

The total training loss is

$$\mathcal{L}_{total} = \alpha_1 \mathcal{L}_{ctg} + \alpha_2 \mathcal{L}_{res} + \alpha_3 \mathcal{L}_{cls} \tag{15}$$

where $\mathcal{L}_{cls}$ is the classification loss. In our experiments $\alpha_1 = \alpha_2 = \alpha_3 = 1$. The joint loss would be optimized by mini-batch stochastic gradient descent.

4. Experiments

In this section, we evaluate the performance of the proposed CDB on benchmark datasets including ImageNet and CIFAR-100. Experimental results demonstrate the superiority of our method on both efficiency and accuracy. In the following part, we compare our method with existing dynamic inference methods and show the effectiveness of our context-aware group controller. Additionally, we explore the inside process of our CDB by visualizing the inference paths of different categories.

4.1. Training Setup

ImageNet. The ImageNet dataset consists of 1.2 million training images and 50K validation images of 1000 classes. We train networks on the training set and report the top-1 and top-5 errors on the validation set. We apply standard practice and perform data augmentation with random horizontal flipping and random-size cropping to 224×224 pixels. We follow the standard Nesterov SGD optimizer with momentum 0.9 and a mini-batch of 256. The cosine learning rate scheduler is employed for better convergence

Output size	ResNet-50	CDB-50, $N = 2$	CDB-50, $N = 3$
112×112	conv, 7×7, stride 2		
56×56	max pool, 3×3, stride 2		
	$\begin{bmatrix} \text{conv}, 1 \times 1, 64 \\ \text{conv}, 3 \times 3, 64 \\ \text{conv}, 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1 \times 1, 38 \\ \text{conv}, 3 \times 3, 38 \\ \text{conv}, 1 \times 1, 256 \end{bmatrix} \times N \times 3$	$\begin{bmatrix} \text{conv}, 1 \times 1, 30 \\ \text{conv}, 3 \times 3, 30 \\ \text{conv}, 1 \times 1, 256 \end{bmatrix} \times N \times 3$
28×28	$\begin{bmatrix} \text{conv}, 1 \times 1, 128 \\ \text{conv}, 3 \times 3, 128 \\ \text{conv}, 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv}, 1 \times 1, 76 \\ \text{conv}, 3 \times 3, 76 \\ \text{conv}, 1 \times 1, 512 \end{bmatrix} \times N \times 4$	$\begin{bmatrix} \text{conv}, 1 \times 1, 60 \\ \text{conv}, 3 \times 3, 60 \\ \text{conv}, 1 \times 1, 512 \end{bmatrix} \times N \times 4$
14×14	$\begin{bmatrix} \text{conv}, 1 \times 1, 256 \\ \text{conv}, 3 \times 3, 256 \\ \text{conv}, 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv}, 1 \times 1, 152 \\ \text{conv}, 3 \times 3, 152 \\ \text{conv}, 1 \times 1, 1024 \end{bmatrix} \times N \times 6$	$\begin{bmatrix} \text{conv}, 1 \times 1, 120 \\ \text{conv}, 3 \times 3, 120 \\ \text{conv}, 1 \times 1, 1024 \end{bmatrix} \times N \times 6$
7×7	$\begin{bmatrix} \text{conv}, 1 \times 1, 512 \\ \text{conv}, 3 \times 3, 512 \\ \text{conv}, 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1 \times 1, 304 \\ \text{conv}, 3 \times 3, 304 \\ \text{conv}, 1 \times 1, 2048 \end{bmatrix} \times N \times 3$	$\begin{bmatrix} \text{conv}, 1 \times 1, 240 \\ \text{conv}, 3 \times 3, 240 \\ \text{conv}, 1 \times 1, 2048 \end{bmatrix} \times N \times 3$
1×1	global average pool, 1000-d fc, softmax		

Table 1: Architecture of CDB-50 for ImageNet.

Model	Top-1 Err. (%)	Params (10^6)	FLOPs (10^9)	FLOPs Ratio (%)
ResNet-50 [7]	24.7	25.56	3.8	-
ResNet-50 (PyTorch Official) [17]	23.85	25.56	3.96	100.0
ResNet-50†(ours)	23.51	25.56	3.96	100.0
ConvNet-AIG-50 [t=0.7] [21]	23.82	26.56	3.06	77.3
S-ResNet-50-0.75 [1]	25.1	19.2	2.3	58.1
CDB-50, N=2 [r=0.4]	24.06	24.67	2.07	52.3
CDB-50, N=2 [r=0.5]	23.50	24.67	2.28	57.6
CDB-50, N=2 [r=0.6]	23.22	24.67	2.52	63.6
CDB-50, N=2 [r=0.7]	22.57	24.67	3.12	78.8
CDB-50, N=3 [r=0.7]	22.54	25.81	3.16	79.8
CDB-50, N=4 [r=0.7]	**22.32**	25.81	3.17	80.1
ResNet-101 [7]	23.6	44.54	7.6	-
ResNet-101 (PyTorch Official) [17]	23.63	44.55	7.67	100.0
ResNet-101†(ours)	22.02	44.55	7.67	100.0
ConvNet-AIG-101[t=0.5] [21]	22.63	46.23	5.11	66.6
CDB-101, N=2 [r=0.3]	22.82	43.12	2.48	32.3
CDB-101, N=2 [r=0.5]	21.95	43.12	4.21	54.9
CDB-101, N=2 [r=0.7]	**21.43**	43.12	5.57	72.6

† Our implementations of ResNet-50, ResNet-101, CDB-50, CDB-101 use stride=2 in conv3 $\times$ 3 layers just as the PyTorch community does [17] which is slightly different from the original paper.

Table 2: Comparison on heavyweight networks on ImageNet. We compare our CDBs with the heavyweight networks ResNet-50, ResNet-101, and other dynamic networks ConNet-AIGs and slimmable network. Results show that our models outperform the other models in both accuracy and computational complexity.

and the initial learning rate is set to 0.1. For different scale models, We use different weight decays, 0.0001 for ResNet and 0.00004 for MobileNet. All models are trained for 120 epochs from scratch.

Model	Top-1 Err. (%)	Params (10^6)	FLOPs (10^9)	FLOPs Ratio (%)
MobileNet V2 [18]	28.0	3.47	-	-
MobileNet V2 (ours)	28.09	3.50†	0.30	100.0
S-MobileNet V2-0.75 [1]	31.1	2.7	0.23	76.7
CDB-MobileNetV2, N=2 [r=0.7]	28.30	3.63	0.22	73.3
CDB-MobileNetV2, N=2 [r=0.8]	28.15	3.63	0.24	80.0
CDB-MobileNetV2, N=2 [r=0.9]	**27.74**	3.63	0.27	90.0
MobileNetV2 (1.4) [18]	25.3	6.06	-	-
MobileNetV2 (1.4) (ours)	25.30	6.09†	0.57	100.0
CDB-MobileNetV2 (1.4), N=2 [r=0.7]	26.03	6.29	0.42	73.7
CDB-MobileNetV2 (1.4), N=2 [r=0.8]	25.53	6.29	0.47	82.5
CDB-MobileNetV2 (1.4), N=2 [r=0.9]	**25.26**	6.29	0.52	91.2

† Our implementation of MobileNet V2 is based on PyTorch and its parameter quantities are counted by PyTorch Summary [3].

Table 3: Comparison on lightweight networks on ImageNet. Our CDBs based on MobileNetV2 can achieve remarkable results comparing to other lightweight models.

CIFAR-100. The CIFAR-100 datasets consist of 60,000 color images of 10, 000 classes. They are split into the training set and testing set by the ratio of 5:1. Considering the small size of images (32×32) in CIFAR, we follow the same setting as [7] to construct our CDBs for a fair comparison. We augment the input image by padding 4 pixels on each side with the value of 0, followed by random cropping with a size of 32×32 and random horizontal flipping. We train the network using SGD with the momentum of 0.9 and weight decay of 0.0001. The mini-batch size is set to 256, and the initial learning rate is set to 0.1. We train the networks for 200 epochs and divide the learning rate by 10 twice, at the 100th epoch and 150th epoch respectively.

4.2. Comparison with Other Methods

We compare our method with ResNet-50, ResNet-101, MobileNetV2 and other dynamic inference methods. We denote N as the number of channel groups of each block, r as the FLOPs rate target. As shown in Tab. 2, 4 and Fig. 4, our method outperforms in both accuracy and computational complexity.

Tab. 2 shows ImageNet results on both ResNet-50 and ResNet-101. Our CDB achieves remarkable results. For CDB-50, when $N = 2$, $r = 0.5$, our method achieves similar performance with ResNet-50 and saves more than 42.4% FLOPs. When we set $N = 4$, $r = 0.7$, our CDB-50 reduces 1.19% Top-1 error while still saves 19.9% FLOPs. For CDB-101, we outperform ResNet-101 and save 45.1% FLOPs in the same time when we set $N = 2$, $r = 0.5$. Meanwhile, our method outperforms other dynamic inference methods, i.e. ConvNet-AIG [21] and slimmable network [1] obviously in both accuracy and computational complexity. For example, with respect to ResNet-50, our CDB-50 with N=2 and r=0.4 achieves comparable performance with ConvNet-AIG yet greatly reduces the FLOPs by one third.

Model	FLOPs (10^9)	Top-1 Err.(%)
ResNet-50 [7]	0.33	27.55
CDB-50, N=2 [t=0.5]	0.18	28.24
CDB-50, N=2 [t=0.7]	0.22	27.34
CDB-50, N=4 [t=0.7]	0.26	**26.15**

Table 4: Test error on CIFAR-100. The CDBs reduce 1.4% Top-1 error while saving about 21.2% FLOPs.

As shown in Tab. 3, 4, when we apply CDB to lightweight network MobileNetV2 or perform experiments on CIFAR-100 dataset with ResNet-50, the same conclusion is obtained. In Tab. 3, our CDB with $N = 2$ and $r = 0.9$ can save 10% FLOPs and achieves better top1 error with MobileNetV2. Tab. 4 shows that CDB-50 with $N = 4$ and $t = 0.7$ can even outperform ResNet-50 by 1.4% on CIFAR-100 with only 79% FLOPs.

In summary, our method performs superbly in accuracy and computational complexity for both heavy and lightweight networks, which demonstrates its great applicability to different networks and robustness on different datasets.

4.3. Efficiency of Context-aware Group Controller

In order to show the effectiveness of the proposed context-aware group controller, we conduct four groups of

Method	CA	CS	Top-1 Err.(%)
ResNet-50 [7]			23.51
CDB-50			23.25
CDB-50	√		23.09
CDB-50		√	23.20
CDB-50	√	√	**22.57**

Table 5: The effectiveness of contextual information and group controller with supervised learning on ImageNet. The FLOPs target is set to 0.7, and the number of group N is set to 2. "CA" represents employing contextual information and "CS" represents employing group controller with supervised learning in this table.

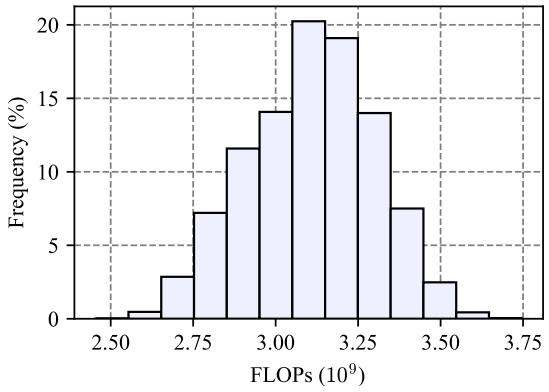


Figure 5: Distribution of FLOPs on the ImageNet validation set using CDB-50 with $N = 2$, $r = 0.7$.

experiments on ImageNet dataset with different configurations. Tab. 5 shows the comparison of different models. Noted that even if we remove both the context-aware mechanism and group controller's category supervised learning, our CDB-50 can get better results than ResNet-50 with less computation burden. If we apply contextual information and group controller with supervised learning separately, additional promotions are obtained. After aggregating these two improvements, we can boost the performance by 0.68%, demonstrating the efficiency of the proposed context-aware group controller. It is worth noting that the context-aware mechanism only introduces a fully connected layer with 32 hidden neurons, the additional computation cost can be omitted. The supervised learning of the group controllers may generate minor additional computational cost during training, yet it will be removed at the testing stage.

4.4. Analysis

Ablation study of N and r. We adopt different values of N and r to explore their impacts on the performance. As shown in Tab. 2, we set $N = 2, 3, 4$, while keep $r = 0.7$ on CDB-50. The model with $N = 4$ obtains the lowest test error rate, indicating that bigger N can lead to more path selection choices and consequently better performance. We further keep $N = 2$ and change r to 0.4, 0.5 and 0.6 respectively. Larger r leads to more computational cost that verifies the effectiveness of our resource-constrained mechanism. The model with larger FLOPs rate gains higher performance since more computation units are involved. The CDB can achieve a better accuracy-efficiency trade-off in terms of the computational budgets.

Visualization of dynamic inference paths. The inference paths vary across images leading to different computation cost. Fig. 5 shows the distribution of FLOPs on the ImageNet validation set using our CDB-50 model with $N = 2$, $r = 0.7$. The proportion of images with FLOPs in the middle is the highest, and images do occupy different computing resource which is guided by computational constraint. We further visualize the execution rates of each group at each block within the categories of animals, artifacts, natural objects, geological formations as shown in Fig. 6. We can see that some group channels, especially group channels at the first two blocks, are executed all the time and the execution rates of other group channels vary from categories. One reason could be that different categories share the same shallow layers' features which are important for classification. As the layer goes deeper, the semantic information of the features becomes stronger, which depends on categories.

5. Conclusion

In this paper we present a novel dynamic inference method called Context-aware Dynamic Block (CDB). This method sets multiple channel groups at each block so that the input sample can choose a more efficient and reasonable path during inference. We also carefully design a gate structure called context-aware group controller to obtain better network structure optimization results. Furthermore, we introduce resource-constrained loss which integrates FLOPs constraint into the optimization target to control the computational complexity of the target network. Experimental results demonstrate the superiority of our method on both efficiency and accuracy.

References

[1] Anonymous. Slimmable neural networks. In *Submitted to International Conference on Learning Representations*, 2019. under review. 1, 3, 6, 7

[2] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 2

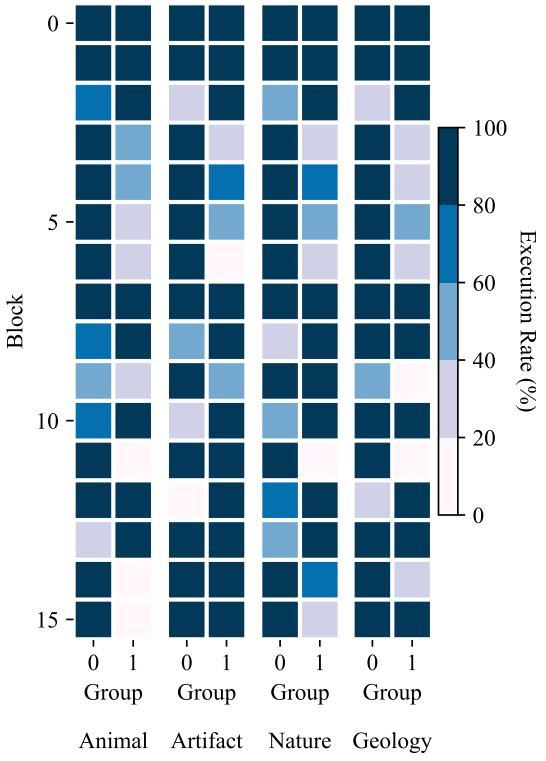


Figure 6: Execution rates of channel groups for different categories on CDB-50 with $N = 2$, $r = 0.7$.

[3] S. Chandel. sksq96/pytorch-summary, Sep 2018. 7

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2, 4

[5] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011. 4

[6] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2, 6, 7, 8

[8] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, volume 2, 2017. 3

[9] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[11] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 4

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[13] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009. 1

[14] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *arXiv preprint arXiv:1807.11164*, 2018. 1, 2

[15] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 3

[16] A. Polino, R. Pascanu, and D. Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018. 3

[17] PyTorch. torchvision.models. 6

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 1, 2, 3, 7

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[21] A. Veit and S. Belongie. Convolutional networks with adaptive inference graphs. In *European Conference on Computer Vision*, pages 3–18. Springer, 2018. 1, 3, 6, 7

[22] X. Wang, F. Yu, Z.-Y. Dou, and J. E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. *arXiv preprint arXiv:1711.09485*, 2017. 1, 3

[23] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016. 3

[24] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017. 3

[25] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017. 2

[26] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2423–2432, 2018. 2

[27] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2