Turkish Journal of Electrical Engineering and Computer Sciences

Volume 31 | Number 5 Article 6

9-29-2023

Dynamic deep neural network inference via adaptive channel skipping

MEIXIA ZOU

XIUWEN LI

JINZHENG FANG

HONG WEN

WEIWEI FANG

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2023) 31: 828 – 843

doi:10.55730/1300-0632.4020

Dynamic deep neural network inference via adaptive channel skipping

Meixia ZOU, Xiuwen LI, Jinzheng FANG, Hong WEN, Weiwei FANG*
School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China

Received: 16.03.2023 • **Accepted/Published Online:** 11.08.2023 • **Final Version:** 29.09.2023

Abstract: Deep neural networks have recently made remarkable achievements in computer vision applications. However, the high computational requirements needed to achieve accurate inference results can be a significant barrier to deploying DNNs on resource-constrained computing devices, such as those found in the Internet-of-things. In this work, we propose a fresh approach called adaptive channel skipping (ACS) that prioritizes the identification of the most suitable channels for skipping and implements an efficient skipping mechanism during inference. We begin with the development of a new gating network model, ACS-GN, which employs fine-grained channel-wise skipping to enable input-dependent inference and achieve a desirable balance between accuracy and resource consumption. To further enhance the efficiency of channel skipping, we propose a dynamic grouping convolutional computing approach, ACS-DG, which helps to reduce the computational cost of ACS-GN. The results of our experiment indicate that ACS-GN and ACS-DG exhibit superior performance compared to existing gating network designs and convolutional computing mechanisms, respectively. When they are combined, the ACS framework results in a significant reduction of computational expenses and a remarkable improvement in the accuracy of inferences.

Key words: Deep neural networks, channel skipping, gating network, dynamic convolution

1. Introduction

Deep neural networks (DNNs) have gained significant traction in recent years and have demonstrated outstanding performance in diverse domains such as computer vision, natural language processing [1], and autonomous robotics control systems. However, the extraordinary performance of DNNs comes at a cost of high computational complexity and memory requirement [2, 3]. For instance, ResNet38 utilizes 2.25 million parameters and performs 667M multiply-accumulate (MAC) operations to process a single input image [4]. The high computational and memory demands can be supported by powerful hardware such as high-performance CPUs, GPUs, or TPUs. However, deploying DNNs on edge devices like Internet-of-things (IoT) [5, 6] has become a very challenging task, since these devices generally have constrained computation and storage resources. Using the Raspberry Pi 3B single-board computer as a typical example, the time required for one inference can vary from a few hundred milliseconds to tens of seconds, depending on the complexity of the underlying DNN model [7]. The complexity of DNN models hinders their deployment in many applications, such as the navigation, control and sensing of robots. To bridge the gap between the constrained resources of computing devices and the computational complexity of DNN models, it is imperative to develop new techniques to achieve a desired trade-off between resource consumption and model accuracy in DNN inference.

*Correspondence: fangww@bjtu.edu.cn

Researchers have explored various model compression approaches to accelerate DNN inference, such as model pruning [8], knowledge distillation [9], and designing compact architectures [10]. In general, these techniques attempt to enhance inference efficiency in a "static" way. This means that during inference, we cannot dynamically adjust the models since both the computational graph and network parameters are predetermined before deploying the model [11]. However, it has been pointed out in [12, 13] that altering the model architectures or removing less significant neurons may lead to a decline in inference capability, thereby affecting the accuracy of classifying challenging input instances. Moreover, the static compression-based approaches may not be able to reduce computational redundancy optimally since all input instances are processed with identical computation, disregarding their classes or levels of difficulty. Dynamic inference, as an alternative approach to execute inference tasks, allows for adapting model structures or network parameters during runtime to the input complexity or resource constraints [14]. The fundamental truth is that images with varying levels of complexity need different amounts of computation, where easily distinguishable ones tend to require less computation, whereas cluttered or ambiguous ones tend to require more computation [15]. Given this fact, a simple solution is to perform inference by utilizing dynamic network depths, either by allowing simple inputs to be processed using shallow exits to produce predictions [16], or by bypassing the unnecessary computations at intermediate layers in a strategic manner [11]. Generally, skipping layers is a more feasible and efficient approach than early exiting because it does not require adding additional branching structures and intermediate classifiers.

However, there are three main challenges facing the current layer-wise skipping technique that inspire us to design alternative solutions. Firstly, the existing strategies, such as SkipNet [13] and Blockdrop [17], chose to either skip or execute an entire layer or block during the inference process. However, these coarse-grained skipping strategies may negatively impact the model's performance, as different layers and channels make unique contributions to feature extraction. To address this issue, we propose a finer-grained skipping scheme called adaptive channel skipping (ACS), which operates at the channel granularity. Secondly, it is worth noting that many existing dynamic inference solutions utilize gating networks that are based on convolutional neural networks (CNNs) [13, 15]. These solutions primarily aim to skip within chain-topology models, such as deep residual network (ResNet). However, recent research has shown that such designs can result in significant computational costs when applied to more complex backbones, such as densely connected networks (DenseNet) [18], as demonstrated in [11]. Thus, our latest innovation, the ACS-GN gating network, utilizes a combination of 1×1 and 3×3 convolutional operations to maximize efficiency and minimize cost for both ResNet and DenseNet models. Lastly, it is important to note that incorporating the ACS-GN into the original backbone network may result in an increase in computational costs and potentially impact the inference performance. To mitigate these impacts, we develop the ACS-DG approach which replaces the standard convolution operation in ACS-GN with dynamic grouping convolution [19]. Overall, the contributions of this research can be summarized as follows:

- For input-dependent dynamic inference, ACS not only surpasses current skip-layer-based solutions but also proposes a new gate network, ACS-GN, which enables channel skipping to be more precise and adaptive. ACS-GN is a flexible approach that can be effortlessly applied to both chained backbone networks (e.g., ResNet) and densely connected backbone networks (e.g., DenseNet).

- In order to enhance computational efficiency and accelerate the inference process, ACS introduces a novel method called ACS-DG, which replaces the standard 3×3 convolution used in ACS-GN with a grouping convolution that dynamically adjusts the number of groups.
- We assess the performance of ACS on the CIFAR10 and SVHN datasets by utilizing ResNet and DenseNet as the foundational backbone models. Our experiments have shown that ACS is capable of generating effective skipping decisions that can significantly reduce the computational costs involved in model inference. Furthermore, the accuracy of inference achieved by ACS is higher than that of the original model and other dynamic skipping schemes, such as those described in references [11] and [13].

The structure of the rest of this paper is as follows. Section 2 provides an overview of related work. In Section 3, we present the design of ACS, including the technical details of ACS-GN and ACS-DG. Section 4 outlines our evaluation of ACS using ResNet and DenseNet models on CIFAR10 and SVHN datasets, and presents our analysis of the experimental results. Finally, in Section 5, we conclude the paper and discuss future work in this area.

2. Related work

Improving the speed of DNN model inference is a critical concern for implementing AI applications in real-world scenarios. This section provides an overview of previous research on developing efficient deep neural network models, which can be broadly classified into two categories: static and dynamic. As space is limited, readers who are interested in delving deeper into related topics can refer to [9, 14, 20] for further information.

2.1. Static compression

Recently, there has been a lot of attention on reducing the size of models and speeding up their inference. Several techniques have been suggested, such as model pruning [8], knowledge distillation [9], and designing compact architectures [10]. In the following paragraphs, we will provide a brief overview of these methods.

The objective of model pruning is to decrease the computational cost and memory usage by eliminating unimportant neurons, filters, or channels from targeted DNN models. Unstructured pruning enables the selective pruning of weights below a specified threshold [8], which often results in high compression rates with minimal impact on inference accuracy. However, it leads to irregular sparsity within the weight matrix, which can be challenging to process efficiently on typical DNN hardware for inference acceleration. On the other hand, structured pruning removes groups of consecutive parameters, such as entire channels/filters/subblocks of DNN weights, to generate structured sparse patterns [21–23]. It is more hardware-friendly than unstructured pruning and can achieve high acceleration performance by leveraging hardware parallelism. However, structured pruning tends to suffer from noticeable accuracy degradation as the pruning rate increases [24], making it difficult to determine the optimal pruning rate.

Knowledge distillation utilizes the transfer of knowledge to condense DNN models. A target compact model (student) can acquire knowledge from a more sophisticated yet unwieldy teacher model by imitating the prediction distribution founded on the same inputs [25]. The concept was initially put forth by Hinton et al., who proposed aligning the outputs of the student and teacher models by minimizing the KL-divergence of the category distribution [26]. Several studies concentrate on determining what knowledge to condense, with an

emphasis on response-based knowledge (i.e. derived from output probability distribution [26]), feature-based knowledge (i.e. originating from intermediate feature representation [27]), and relation-based knowledge (i.e. arising from relationships between layers or inputs [28]). Another line of research places greater emphasis on how to distill knowledge. In the traditional approach, knowledge distillation involves a two-stage process where the teacher model is assumed to be pretrained offline and ready for one-way condensation [26]. Without this assumption, online distillation enables a group of student models to learn from each other and receive updates simultaneously during the training process [29].

In addition to model pruning and knowledge distillation, there are other ways to reduce the complexity and cost of models by using new building blocks with efficient operations. As an example, MobileNets replace the conventional convolution with a unique combination of 3×3 depthwise convolution and 1×1 pointwise convolution [30]. ShuffleNet achieves a balance between accuracy and cost by employing pointwise group convolution and channel shuffle [31]. In AdderNet [32], the computational cost is reduced by replacing a large number of multiplications in the DNN with lighter-weight additions. Recently, there has been a proposal for neural architecture search (NAS) as a method to create the best possible DNN architectures using optimization algorithms [3].

Although effective, these static compression-based approaches generally have two distinct drawbacks [15]. On the one hand, removing layers, filters, or neurons from a DNN model may compromise its representational power, as sparsely activated neurons may be indispensable for handling difficult inputs. On the other hand, treating all input instances indiscriminately is inefficient, i.e. deep network embeddings are only needed for challenging images that require accurate classification but may be wasted on easy images. These issues have prompted the emergence of new dynamic inference solutions.

2.2. Dynamic inference

Although static compression-based methods make DNN deployment more feasible by reducing model complexity, they cannot maximize efficiency and accuracy by exploiting input characteristics. In contrast, dynamic solutions can adapt model structures or parameters based on input instances [14], resulting in advantages in terms of efficiency, accuracy, latency, among other factors.

Considering the presence of spatial redundancy, some work attempts to perform adaptive inference by processing only a portion of the input data in the spatial dimension. For example, octave convolution (OctConv) [33] saves storage and computation by reducing the resolution of low-frequency feature maps. Spatially adaptive computation time (SACT) [34] allocates different numbers of execution layers to different regions of the input image. Glance and focus network (GFNet) [35] gradually performs classification on a series of image blocks given a high-resolution image until sufficient confidence is obtained.

Our work belongs to a category of dynamic solutions that differentiate between easy and difficult input instances. BranchyNet [16] enhances the model by incorporating additional branch classifiers, which allow for early exit of certain inputs based on confidence-based criteria. BlockDrop [17] builds an additional policy network based on reinforcement learning to determine which blocks to be removed in the pretrained ResNet model while maintaining inference accuracy. SkipNet [13] proposes modifying residual networks by adding gate networks so that the decision of skipping blocks is based on a hybrid learning algorithm that combines supervised learning and reinforcement learning. However, this layer-skipping strategy is coarse-grained and

may result in significant accuracy reduction. Some recent studies [11, 15] have started to use finer-grained dynamic skipping of layer channels. In [15], two auxiliary networks with shallow architectures are suggested to facilitate dynamic skipping of layers and channels. This method of skipping is specially developed for DNN models which utilize a block-based residual design. To address this issue, input-adaptive dynamic inference (IADI) [11] proposes a gating design that is applicable to both chain-like backbone networks (e.g., ResNet) and dense connection backbone networks (e.g., DenseNet), demonstrating significant universality and applicability. Compared to SkipNet and BlockDrop, IADI has been shown to be more computationally efficient.

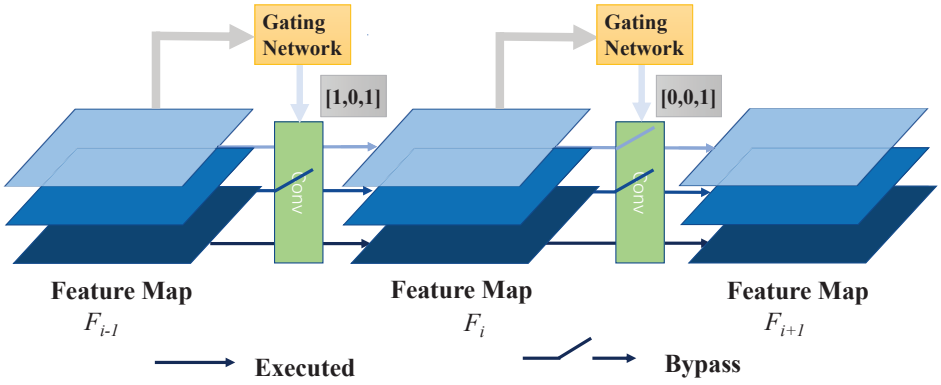

Figure 1. An illustration of channel skipping with gating networks in chain-like backbone.

3. ACS: adaptive channel skipping

In this section, we provide a detailed introduction to our adaptive channel skipping scheme (ACS). Firstly, we present our new CNN-based gating network called ACS-GN. Then, we explain the design of the group convolution ACS-DG, and demonstrate how it can be integrated with ACS-GN to enhance its computational efficiency. Lastly, we discuss the training strategy that implements constraints to ensure the model produces the desired computational results.

3.1. Basic gating network design of ACS-GN

ACS-GN allows for selective activation or execution of individual channels within each convolutional layer, depending on the input provided. As shown in Figure 1, the gating networks are placed between layers, and map the output from the previous layer or group of layers to a sequence of binary decisions, which determine which channels to skip in the subsequent layer. In the following, we will describe how ACS-GN can be incorporated into ResNet and DenseNet, which are commonly studied DNN models for dynamic inference [11, 13, 17].

The decision of whether to skip channels for a 3×3 convolution layer i with k channels can be formulated as:

$$F_i^{out} = C_i(G_i(F_i^{in})) * F_i^{in}, \tag{1}$$

where F_i^{in} and F_i^{out} denote the input and output feature maps, C_i denotes the convolutional operation for the i-th layer, and the gating network G_i outputs a size-k vector in which each value denotes the

binary skipping decision for a channel, respectively. During inference, a convolution operation will only be executed for channels with a nonzero gating value, while channels with a gating value of zero will be skipped and not calculated. Note that F_i^{in} is not necessarily equal to F_{i-1}^{out}, due to the existence of the shortcut connections in ResNet and the dense connections in DenseNet [4, 18]. Take DenseNet as an example, $F_i^{in} = ReLU(BN([F_0^{out}, F_1^{out}, \ldots, F_{i-1}^{out}]))$, where [] refers to the concatenation of the feature maps of all preceding layers.

An effective gating network can dynamically identify the subset of computations that need to be executed to significantly reduce computational costs while minimizing the impact on inference accuracy. For layer skipping, recurrent neural networks (RNNs) can be a suitable choice to serve as the only gating network for sequentially determining the optimal routing for all layers [13]. However, to make decisions on which channels to skip, it is crucial to consider the relevance between channels within the same layer, rather than focusing on cross-layer correlations alone. The recurrent design is not suitable here, since different layers generally have different number of channels (i.e. k) [11]. Due to these two reasons, we turn to gate design based on CNN, associating a gate with each 3×3 convolutional layer.

Referring to Figure 2, our ACS-GN design is inspired by FFGate-I proposed in SkipNet [13] and IADI [11] (Note that FFGate-II is specifically designed for deep networks with more than 100 layers), while considering performance and cost. Specifically, ACS-GN consists of a 1×1 bottleneck layer, and two 3×3 convolutional layers with stride of 2 and 1 respectively, followed by a global average pooling layer and a fully-connected layer. On the one hand, the max pooling operation in FFGate-I can only decrease the height and width of the input, whereas a 1×1 convolution can provide filter-level pooling to further reduce the channel dimension. On the other hand, we follow FFGate-I and use two 3×3 convolutional layers to extract informative features, instead of one layer as in IADI but adjust their order for computation reduction. Output of ACS-GN is a vector with k dimensions, where each value is rescaled using a Sigmoid function to a range of (0,1), and then quantized to 0 or 1 by comparing it to a threshold α that determines the skipping ratio. Despite the added cost of incorporating CNN gates, implementing fine-grained channel skipping with ACS-GN ultimately reduces overall resource consumption.

3.2. Improving gating network efficiency with ACS-DG

Integrating multiple gating networks unavoidably increases the computational cost of the original backbone network. To mitigate the computational cost incurred by the gating networks, we enhance the efficiency of ACS-GN by incorporating dynamic group convolution. Group convolution, which groups adjacent channels together for computation, has been demonstrated to be efficient, compact, and simple to implement in previous studies [36]. However, conventional methods for group convolution rely heavily on human expertise and trial-and-error to determine the appropriate number of groups, using the same number of groups across different depths of convolutional layers often leads to suboptimal results [19]. Our ACS-DG is inspired by dynamic grouping convolution (DGConv), which was proposed in [19]. A brief overview of DGConv is provided below.

Generally, the output of a convolution operation with kernel size k and stride 1 on an input $F^{in} \in \mathbb{R}^{H \times W \times C^{in}}$ can be defined as

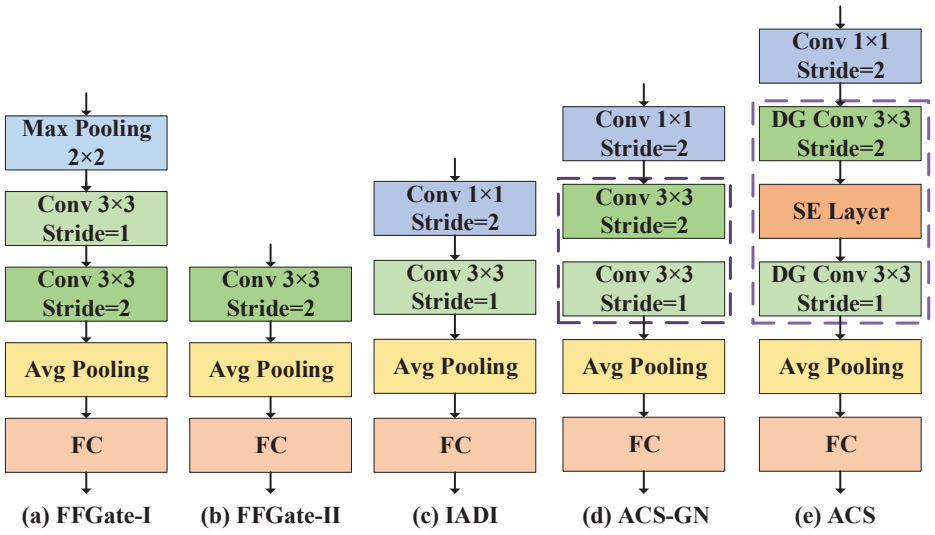

Figure 2. Comparison of the existing gating network structure.

$$o_{hw} = \sum_{x=0}^{k-1} \sum_{y=0}^{k-1} f_{(h+x)(w+y)}(M \odot \omega_{xy}), \tag{2}$$

where $h \in \{1, \ldots, H\}$, $w \in \{1, \ldots, W\}$, $o_{ij} \in \mathbb{R}^{C^{out}}$ denotes the output unit, $f_{(h+x)(w+y)} \in \mathbb{R}^{C^{in}}$ denotes the input units of F, $\omega_{xy} \in \mathbb{R}^{C^{in} \times C^{out}}$ denotes the convolution weights, and $\odot$ denotes element-wise product. Here, $M \in \{0,1\}^{C^{in} \times C^{out}}$ is a binary relationship matrix that can represent a variety of convolution operations in the literature. For example, Equation (2) represents a regular convolution when $M = \mathbf{1}$, a depthwise convolution [30] when M is an identity matrix, and a conventional group convolution [36] when M is a binary block-diagonal matrix. DGConv aims to construct an optimal structured matrix M while reducing the total number of parameters.

According to DGConv [19], because $C^{in} = C^{out}$ for the convolutions in ACS-GN, M can be decomposed into a number of 2×2 submatrices as

$$M = M_1 \otimes M_2 \otimes \ldots M_Z, \tag{3}$$

where $\otimes$ denotes a Kronecker product, and $Z = \log_2 C^{in}$. The submatrix M_z is calculated as

$$M_z = g_z \mathbf{1} + (1 - g_z)\mathbf{I} \tag{4}$$

where $\mathbf{1}$ and $\mathbf{I}$ are 2×2 matrix of ones and identity matrix respectively, and g_z is the z-th component in the binary gate vector $g = \{g_1, \ldots, g_Z\} \in \{0,1\}^Z$. In the training, it is handled by training a continuous gate vector $\hat{g}$ as extra parameters, and then converting $\hat{g}$ to g using the $sign$ function (i.e. $sign(x) = 0$ when $x < 0$, and $sign(x) = 1$ otherwise.). In this way, we can obtain a relationship matrix M that has a group

structure. Figure 3 shows some illustrative examples of structures of M and the groups differentiated by colors in M when $C_{in} = C_{out} = 8$.

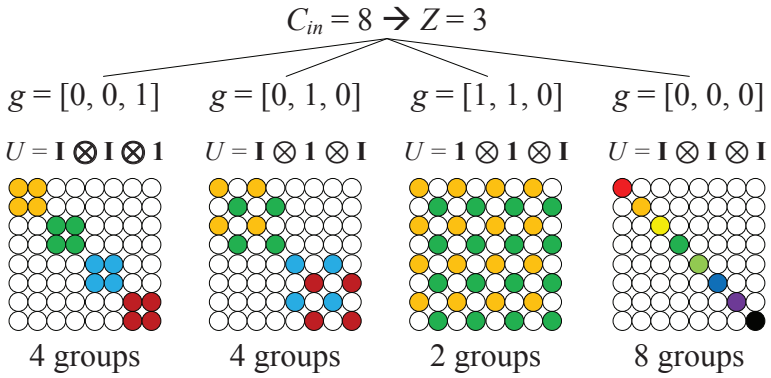

Figure 3. Illustration of structures and groups with relationship matrix M.

With this new approach, the regular 3×3 convolution used in ACS-GN can be replaced by DGConv. To address the potential negative impact of group convolution on inference accuracy [36], we enhance our ACS-DG design by incorporating the widely-used attention mechanism. Specifically, we insert a Squeeze-and-Excitation Network (SENet) block [37] between the two DGConv layers. The core of SENet is a Squeeze-and-Excitation block, which consists of a squeeze module and an excitation module. The squeeze module in SENet gathers global spatial information through computation of channel-wise statistics via global average pooling, while the excitation module leverages the aggregated information to capture channel-wise dependencies, thereby enhancing both the representational power and prediction capabilities of the network. Finally, we achieve our integrated and formal ACS gating design, which is depicted in Figure 2e. In the experiment, we will compare the five different types of gating designs as shown in Figure 2, and demonstrate the effectiveness and superiority of ACS.

3.3. Training strategy for skipping policy learning

In ACS, the backbone and the gating networks are jointly optimized through supervised learning. Traditionally, only cross-entropy loss is used, and there are usually few or no skipping operations as more feature information tends to result in higher classification accuracy. To achieve a balance between accuracy and computational cost, we suggest implementing a regularization method that takes into consideration the available resources. This approach will assist in selecting which channels to skip and limit the amount of computation used.

The loss function for the model in ACS can be expressed as follows:

$$L = L_{ce}(B, G) + \alpha L_{reg}(B, G), \tag{5}$$

where L_{ce} and L_{reg} used to represent the classification loss and resource-aware regularization, respectively. The parameters of the backbone and gating networks are denoted as B and G, while α represents the weighted coefficient that determines the skipping threshold. The dynamic loss L_{reg} is determined by the number of channels involved in the convolutional computation and is evaluated based on the actual inference floating-point operations per second (FLOPs). Therefore, a penalty will be imposed if the computational cost is too

high or if we intend to reduce more computation with a large α. This enforces the DNN inference to be aware of the instance and considers both accuracy and cost factors. However, one challenge with binary skipping decisions is that they are nondifferentiable, meaning they cannot be directly back-propagated. Therefore, we use the softmax approximation technique[11, 13], treating the softmax outputs as the probability of ACS-GN propagating gradients during back-propagation.

4. Performance evaluation

In this section, we perform comprehensive experiments on image classification tasks to showcase the efficiency and effectiveness of our adaptive channel skipping scheme [1]. Section 4.1 outlines the experimental setup, which includes the datasets used, DNN models, baselines, training specifics, and hyperparameter settings. In Section 4.2, we begin by comparing ACS-GN against state-of-the-art gating network designs. We then investigate ACS-GN's performance when integrated with ACS-DG. Finally, we evaluate ACS against IADI, the current state-of-the-art work on channel skipping.

4.1. Experimental setup

4.1.1. Datasets

We perform experiments on two widely used image classification benchmarks, i.e. CIFAR10 and SVHN, both of which consist of 32×32 colored images [13]. The CIFAR10 dataset is comprised of 50,000 training samples and 10,000 testing samples, which are categorized into 10 different classes. The SVHN dataset contains 604,000 training samples and 26,000 testing samples that have been labeled for 10 distinct classes. Prior to training, the training samples in both datasets were preprocessed by normalizing them using the channel means and standard deviation. These two datasets have been extensively adopted in previous studies in the literature [13, 14, 18].

4.1.2. Models

We assess our designs using three commonly used DNN architectures: ResNet38, DenseNet41, and DenseNet53, which have all been extensively utilized in previous studies related to dynamic computation skipping [11, 38]. As per the design guidelines specified in [4], ResNet38 consists of a single convolutional layer, succeeded by 18 residual blocks (which includes two convolutional layers in each block) and one fully connected layer. As per the design guidelines outlined in [18], DenseNet41 and DenseNet53 are comprised of a first convolutional layer, succeeded by three dense blocks (each dense block consists of two convolutional layers), followed by a transition layer, and a fully connected layer. These two models differ in terms of the number of two-layer structures present in them. Specifically, DenseNet41 comprises 6 two-layer structures, whereas DenseNet53 has 8 two-layer structures. The growth rate is set as 32 for the DenseNet models [11].

4.1.3. Comparisons

In all experiments, the baseline models are the original DNN models. Specifically, ACS-GN is compared with existing gating designs such as FFGate-I, FFGate-II [13], and IADI [11]. Then, the performance of ACS-DG

[1]The source code of ACS is available at: https://github.com/fangvv/ACS.

is compared with regular convolution to evaluate its improvement over ACS-GN. Finally, the ACS and IADI channel hopping methods are compared as two representative approaches to showcase their overall performance.

4.1.4. Training details

To enable the gating network of ACS to adapt to the feature statistics, we develop a two-step training procedure that differs from that of SkipNet or IADI. In the first step, we insert the gating network into the backbone model and train them together with zero skipping ratio. After obtaining the pretrained model, we then set the value of α to a desired target and fine-tune the entire model. In general, the training time of the first stage is similar to that of the original model, while the total training time increases by 40%–60%.

4.1.5. Hyperparameter settings

In the experiments, we utilize a stochastic gradient descent (SGD) optimizer with a momentum of 0.9 and a weight decay of 1e-4. We train the models using a batch size of 128 and an initial learning rate of 0.1 with a cosine decay schedule. We train for a total of 120 epochs for CIFAR10 and 50 epochs for SVHN. The default skipping ratio α is set to 0.5.

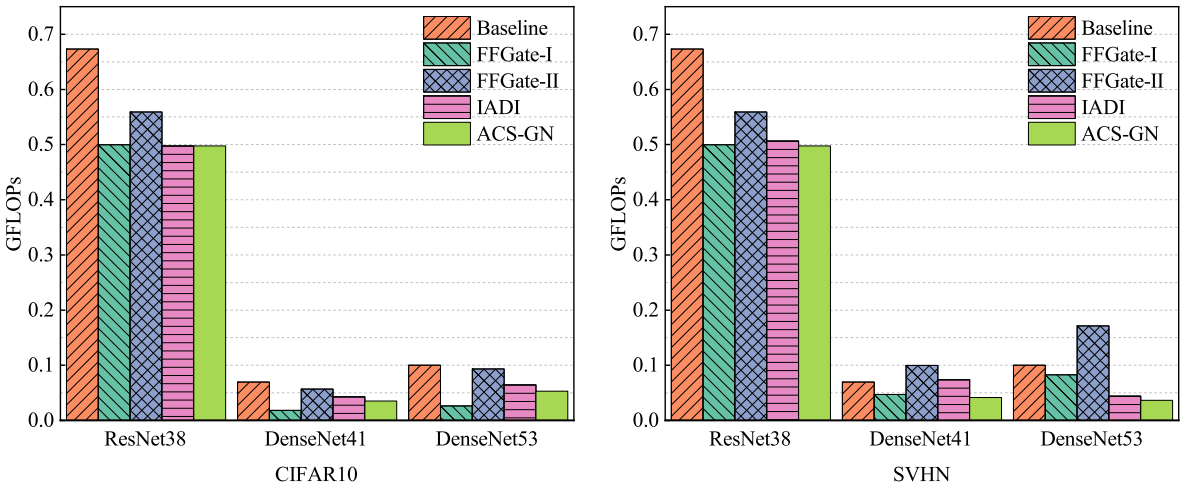

Figure 4. (a) Comparison of computational cost for different gating designs [11, 13] and DNN models [4, 18] on CIFAR10. (b) Comparison of computational cost for different gating designs [11, 13] and DNN models [4, 18] on SVHN.

4.2. Experiment results

4.2.1. Evaluation for ACS-GN

For the initial set of experiments, we apply the ACS-GN method to two datasets and compare its performance to other gating designs. The comparison results of accuracy and Giga FLOPs (GFLOPs) are presented in Figure 4 and Table 1 respectively, and these comparisons have yielded several observations. Firstly, the layer-skipping gates, FFGate-I and FFGate-II, have a drawback in that they only permit a coarse-grained choice to execute the whole layer, which may lead to a significant loss in accuracy. Out of the four designs, FFGate-II has the lowest accuracy and highest cost compared to the others, including the baseline design. This is because its simplified

design not only fails to effectively make skipping decisions but also adds additional cost to the backbone. Second, with the exception of FFGate-II, the other three gates have a similar computational cost when used with ResNet38. However, for both deeper and densely-connected DenseNet models, the costs of these three are quite different. Third, ACS-GN always performs better than IADI. Compared with IADI, ACS-GN can achieve a higher accuracy improvement (CIFAR10: 0.25%∼1.47% vs. 0.39%∼2.353%, SVHN: 0.034%∼0.131% vs. 0.104%∼0.135%) with more computational cost reduction (CIFAR10: 26.0%∼35.5% vs. 26%∼49.1%, SVHN: –6.0%∼55% vs. 26.1%∼63.4%). These results demonstrate the effectiveness and advantages of our proposed gate design, which not only improves the accuracy of the model but also minimizes the computational cost of the model as much as possible.

4.2.2. Evaluation for ACS-DG

For our second round of experiments, we begin by examining how the grouping strategy impacts inference performance. To do so, we use the example of DenseNet41 on CIFAR10. The experiment outcomes are detailed in Table 2. The results clearly indicate that static grouping, which uniformly reduces model parameters, suffers from suboptimal performance. In contrast, ACS-DG is uniquely positioned to learn the optimal group strategy for each convolutional operation in a layer-by-layer approach. The model is able to achieve higher accuracy than the baseline with a 25.35% reduction in FLOPs, which is significantly better than static grouping methods.

To further illustrate the effectiveness of ACS-DG, we employ DenseNet41 to demonstrate how it can enhance the performance of ACS-GN. Table 3 outlines the outcomes of this experiment. Our results indicate that ACS-DG is able to reduce computational costs by up to 25.35% and 50.91% on CIFAR10 and SVHN, respectively, while achieving a slightly higher accuracy. These findings validate the design and usefulness of ACS-DG.

Table 1. Comparison of inference accuracy for different models [4, 18] and datasets.

Dataset	Approach	ResNet38	DenseNet41	DenseNet53
CIFAR10	Baseline	91.667%	86.699%	86.609%
	FFGate-I	↓ 90.274%	↓ 74.109%	↓ 75.130%
	FFGate-II	↓ 87.640%	↓ 81.180%	↓ 80.699%
	IADI	↑ 91.917%	↑ 87.861%	↑ 88.081%
	ACS-GN	↑ **92.057%**	↑ **88.301%**	↑ **88.962%**
SVHN	Baseline	98.072%	97.491%	97.587%
	FFGate-I	↓ 93.146%	↓ 91.967%	↓ 93.023%
	FFGate-II	↓ 90.213%	↓ 92.358%	↓ 93.177 %
	IADI	↑ 98.126%	↑ 97.525%	↑ 97.718%
	ACS-GN	↑ **98.176%**	↑ **97.618%**	↑ **97.722%**

4.2.3. Evaluation for ACS

In the third set of experiments, we conduct a comparison between ACS and IADI, which is the latest state-of-the-art technique on channel skipping, usable for both ResNet and DenseNet architectures. In an effort to maintain fairness within the comparison, the training setting of IADI is kept in line with the original research paper [11]. Additionally, we create an IADI variant that integrates our ACS-DG approach. The results of this experiment are then charted in Table 4. Compared to the baseline, all three skipping schemes can reduce FLOPs

Table 2. Comparison of computational cost and inference accuracy for different number of convolution groups in DenseNet41.

Dataset	Number of groups	GFLOPs	Acc(%)
CIFAR10	Static (baseline)	0.03542	88.301
	Static, 2 groups	0.03050	88.121
	Static, 4 groups	0.02725	87.720
	Static, 8 groups	0.02885	88.371
	Dynamic (ACS-DG)	**0.02644**	**88.642**

Table 3. Comparison of computational cost and inference accuracy for different convolutional approaches [18, 19] in DenseNet41.

Dataset	Approach	GFLOPs	Acc(%)
CIFAR10	Baseline	0.06965	86.699
	ACS-GN	0.03542	↑ 88.301
	ACS (GN + DG)	**0.02644**	↑ **88.642**
SVHN	Baseline	0.06965	97.491
	ACS-GN	0.04154	↑ 97.618
	ACS (GN + DG)	**0.02039**	↑ **97.668**

and improve accuracy, but ACS performs the best in terms of accuracy. For ResNet38, the three schemes have nearly identical computational costs of roughly 0.5 GFLOPs, which is consistent with the findings reported in Figure 4. These results imply that the DNN model does not leave much room for more optimization. Nonetheless, the differences in FLOPs among the three techniques are greater for DenseNet41. Our proposed ACS-DG helps IADI reduce FLOPs by 6.7% on CIFAR10 and 42.1% on SVNH. The FLOPs reduction of ACS alone account for only 61.9% and 27.6% of IADI on CIFAR10 and SVNH, respectively. Overall, our ACS method demonstrates higher or comparable performance in terms of both cost and accuracy compared to the IADI approach.

Table 4. Comparison of computational cost and inference accuracy between IADI and ACS.

Dataset	Approach	GFLOPs	Acc(%)
CIFAR10	ResNet38-Baseline	0.67313	91.667
	ResNet38-IADI	0.49763	↑ 91.917
	ResNet38-IADI + ACS-DG	**0.51552**	↑ **92.228**
	ResNet38-ACS	**0.50160**	↑ **93.139**
	DenseNet41-Baseline	0.06965	86.699
	DenseNet41-IADI	0.04271	↑ 87.861
	DenseNet41-IADI + ACS-DG	**0.03984**	↑ **88.281**
	DenseNet41-ACS	**0.02644**	↑ **88.642**
SVHN	ResNet38-Baseline	0.67313	98.072
	ResNet38-IADI	0.50631	↑ 98.126
	ResNet38-IADI + ACS-DG	**0.50631**	↑ **98.167**
	ResNet38-ACS	**0.51020**	↑ **98.194**
	DenseNet41-Baseline	0.06965	97.491
	DenseNet41-IADI	0.07376	↑ 97.525
	DenseNet41-IADI + ACS-DG	**0.04271**	↑ **97.548**
	DenseNet41-ACS	**0.02039**	↑ **97.668**

4.2.4. Impact of parameter α

In order to balance computational cost and inference accuracy, a hyperparameter called α is introduced, which determines the skipping ratio. In our experiments on CIFAR10, we only use the default value of α as 0.5. Table 5 provides a breakdown of the cost and accuracy results for different α values on the CIFAR10 dataset, ranging from 0.2 to 0.6. When α is large, the models skip more channels, resulting in a significant reduction in computational cost. For the two DNN models, the variations in their accuracies show distinct trends as α increases. Therefore, adjusting the value of α based on specific application demands, such as accuracy sensitivity or cost sensitivity, can help trade-off performance and cost.

Table 5. Comparison of the effect of α on CIFAR10.

α	Model	GFLOPs	Acc(%)
-	ResNet38-Baseline	0.67313	91.667
	DenseNet41-Baseline	0.06965	86.699
0.2	ResNet38	0.69736	92.819
	DenseNet41	0.04372	88.331
0.4	ResNet38	0.66659	93.369
	DenseNet41	0.04154	88.812
0.5	ResNet38	0.50160	93.139
	DenseNet41	0.02644	88.642
0.6	ResNet38	0.43393	92.889
	DenseNet41	0.01128	86.188

5. Conclusion and future work

Efficient inference on resource-constrained devices is a complex issue, as there is typically a tension between the resource-demanding nature of DNN models and the resource limitations of edge devices. Dynamic neural networks have emerged as a promising solution to this problem [14]. In this paper, we propose a new scheme, called ACS, which enables instance-based, fine-grained dynamic inference. Our approach incorporates an efficient gating network, which can be easily applied to popular DNN architectures to skip unnecessary calculations during inference. We evaluate ACS on benchmark datasets like CIFAR10 and SVHN, using typical network backbones ResNet and DenseNet. We demonstrate that ACS performs even better than the existing state-of-the-art dynamic inference approaches, such as FFGate-I, FFGate-II, and IADI. With ACS, we achieve significant reductions in computational FLOPs while maintaining a higher level of accuracy.

However, it should be noted that the current experimental tests conducted for our proposed ACS have only been limited to small-scale datasets and software implementation. Moving forward, we intend to further evaluate the performance of ACS by testing it on other commonly used datasets such as KITTI [39], as well as implementing it in more complex network architectures such as VGG or GoogLeNet [3]. Implementing and evaluating our ACS on a real FPGA-based platform is another important issue [11], as skip operations are currently not well supported by commonly used hardware such as CPUs or GPUs [40]. Interested readers are referred to relevant papers [41, 42] and books [43, 44] on this topic, and can choose the Xilinx Zynq boards for implementation [11, 42].

Acknowledgments

This work has received funding from the National Science Foundation of China (NSFC) under Grant 62172031, the Beijing Municipal Natural Science Foundation under Grant L191019, and the Open Research Fund Program of Key Laboratory of Industrial Internet and Big Data, China National Light Industry, Beijing Technology and Business University, under Grant IIBD-2021-KF03.

References

[1] Hu X, Chu L, Pei J, Liu W, Bian J. Model complexity of deep learning: A survey. Knowledge and Information Systems 2021; 63 (10): 2585-2619.

[2] Wang W, Zhu L. Structured feature sparsity training for convolutional neural network compression. Journal of Visual Communication and Image Representation 2020; 71: 102867.

[3] Cai H, Lin J, Lin Y, Liu Z, Tang H et al. Enable deep learning on mobile devices: Methods, systems, and applications. ACM Transactions on Design Automation of Electronic Systems (TODAES) 2022; 27 (3): 1-50.

[4] He K, Zhang X, Ren S et al. Deep residual learning for image recognition.In: IEEE 2016 Conference on Computer Vision and Pattern Recognition; Las Vegas, NV, USA; 2016. pp. 770-778.

[5] Fang W, Xue F, Ding Y, Sun J. EdgeKE: An on-demand deep learning IoT system for cognitive big data on industrial edge devices. IEEE Transactions on Industrial Informatics 2020; 17 (9): 6144-6152.

[6] Cao L. Decentralized ai: Edge intelligence and smart blockchain, metaverse, web3, and desci. IEEE Intelligent Systems 2022; 37 (3): 6-19.

[7] Hadidi R, Cao J, Xie Y, Asgari B, Krishna T et al. Characterizing the deployment of deep neural networks on commercial edge devices.In:IEEE 2019 International Symposium on Workload Characterization; Orlando, FL, USA; 2019.pp. 35-48.

[8] Han S, Pool J, Tran J, Dally J. Learning both weights and connections for efficient neural network. Advances in Neural Information Processing Systems, 2015, 28.

[9] Berthelier A, Chateau T, Duffner S, Garcia, Blanc C. Deep model compression and architecture optimization for embedded systems: A survey. Journal of Signal Processing Systems 2021; 93 (8): 863-878.

[10] Huai S, Zhang L, Liu D, Liu W, Subramaniam R. ZeroBN: Learning Compact Neural Networks For Latency-Critical Edge Systems. In: ACM/IEEE 2021 Design Automation Conference; Phoenix, AZ, USA; 2021. pp. 151-156.

[11] Wang Y, Shen J, Hu T, Xu P, Nguyen T et al. Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference. IEEE Journal of Selected Topics in Signal Processing 2020; 14 (4): 623-633.

[12] Huang G, Chen D, Li T, Wu F, van der Maaten L. Multi-scale dense networks for resource efficient image classification. arXiv preprint arXiv:1703.09844, 2017.

[13] Wang X, Yu F, Dou Z, Darrell T et al. Skipnet: Learning dynamic routing in convolutional networks. In: European 2018 Conference on Computer Vision; Munich, Germany; 2018. pp. 409-424.

[14] Han Y, Huang G, Song S, Gonzalez J. Dynamic neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

[15] Xia W, Yin H, Dai X, Jha N. Fully dynamic inference with deep neural networks. IEEE Transactions on Emerging Topics in Computing, 2021.

[16] Liang Z, Zhou Y. Dispense Mode for Inference to Accelerate Branchynet. In: IEEE 2022 International Conference on Image Processing; Bordeaux, France; 2022. pp. 1246-1250.

[17] Wu Z, Nagarajan T, Kumar A, Rennie S, Davis L et al. Blockdrop: Dynamic inference paths in residual networks.In: IEEE 2018 Conference on Computer Vision and Pattern Recognition; 18-23 June, 2018; Salt Lake City, UT, USA; 2018. pp. 8817-8826.

[18] Huang G, Liu Z, Van Der Maaten L, Weinberger K. Q. Densely connected convolutional networks. In: IEEE 2017 conference on computer vision and pattern recognition; Hawaii; 2017. pp. 4700-4708.

[19] Zhang Z, Li J, Shao W, Peng Z, Zhang R et al. Differentiable learning-to-group channels via groupable convolutional neural networks. In: IEEE/CVF 2019 International Conference on Computer Vision; Seoul, Korea (South); 2019. pp. 3542-3551.

[20] Liang T, Glossner J, Wang L, Shi. S, Zhang. X. Pruning and quantization for deep neural network acceleration: A survey. Neurocomputing 2021; 461: 370-403.

[21] He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks. In: IEEE 2017 International Conference on Computer Vision; Venice, Italy; 2017. pp. 1389-1397

[22] Li H, Kadav A, Durdanovic I, Samet H, Graf P. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710, 2016.

[23] Ao R, Tao Z, Yuhao W, Sheng L, Peiyan D et al. Darb: A density-adaptive regular-block pruning for deep neural networks. In: AAAI 2020 Conference on Artificial Intelligence; Hilton New York Midtown, New York, USA ; 2020. pp. 5495-5502.

[24] Ma X, Guo FM, Niu W, Lin X, Tang J et al. Pconv: The missing but desirable sparsity in dnn weight pruning for real-time execution on mobile devices. In: AAAI 2020 Conference on Artificial Intelligence; Hilton New York Midtown, New York. USA; 2020. pp. 5117-5124.

[25] Gou J, Yu B, Maybank SJ, Tao D. Knowledge distillation: A survey. International Journal of Computer Vision, 2021; 129 (6): 1789-1819.

[26] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015; 2 (7).

[27] Xu K, Rui L, Li Y, Gu L. Feature normalized knowledge distillation for image classification. In: European 2020 Conference on Computer Vision. Springer, Cham; 2020. pp. 664-680.

[28] Park W, Kim D, Lu Y, Cho M. Relational knowledge distillation. In: IEEE/CVF 2019 Conference on Computer Vision and Pattern Recognition; Long Beach, CA, USA; 2019. pp. 3967-3976.

[29] Chen D, Mei JP, Wang C, Feng Y, Chen C. Online knowledge distillation with diverse peers. In: AAAI 2020 Conference on Artificial Intelligence; Hilton New York Midtown, New York. USA; 2020. pp. 3430-3437.

[30] Howard AG, Zhu M, Chen B, Kalenichenko D, Weyand T et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv: 1704.04861, 2017.

[31] Zhang X, Zhou X, Lin M, Sun J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: IEEE/CVF 2018 Conference on Computer Vision and Pattern Recognition; Salt Lake City, UT, USA; 2018. pp. 6848-6856.

[32] Chen H, Wang Y, Xu C, Shi B, Xu C et al. AdderNet: Do we really need multiplications in deep learning? In: IEEE/CVF 2020 Conference on Computer Vision and Pattern Recognition; Seattle, WA, USA; 2020.pp.1468-1477.

[33] Chen Y, Fan H, Xu B, Yan Z, Kalantidis Y et al. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In: IEEE/CVF 2019 International Conference on Computer Vision; Seoul, Korea (South); 2019. pp. 3435-3444.

[34] Figurnov M, Collins MD, Zhu Y, Zhang L, Huang J et al. Spatially adaptive computation time for residual networks. In: IEEE 2017 Conference on Computer Vision and Pattern Recognition; Honolulu, HI, USA; 2017. pp.1039-1048.

[35] Wang Y, Lv K, Huang R, Song S, Yang L et al. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. Advances in Neural Information Processing Systems 2020; 33: 2432-2444.

[36] Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. In: IEEE 2017 Conference on Computer Vision and Pattern Recognition; Honolulu, HI, USA; 2017. pp. 1492-1500.

[37] Hu J, Shen L, Sun G. Squeeze-and-excitation networks. In: IEEE 2018 Conference on Computer Vision and Pattern Recognition; salt lake city, UT, USA; 2018. pp. 7132-7141.

[38] Hao C, Jin N, Qiu C, Ba K, Wang X et al. Balanced convolutional neural networks for pneumoconiosis detection. International Journal of Environmental Research and Public Health 2021; 18 (17): 9091.

[39] Geiger A, Lenz P, Stiller C, Urtasun R. Vision meets robotics: The kitti dataset. The International Journal of Robotics Research 2013; 32 (11): 1231-1237.

[40] Gao C, Saha S, Lu Y, Saha Y, McDonald-Maier K. D et al. Deep Learning on FPGAs with Multiple Service Levels for Edge Computing. In: IEEE 2022 International Conference on Automation and Computing; Bristol, United Kingdom; 2022. pp. 1-6.

[41] Tsai H, Yuan H, Ming S. Implementation of FPGA-based Accelerator for Deep Neural Networks. In: IEEE 2019 International Symposium on Design and Diagnostics of Electronic Circuits & Systems; Cluj-Napoca, Romania; 2019. pp. 1-4.

[42] Venieris S, Christos Bouganis. fpgaConvNet: Mapping regular and irregular convolutional neural networks on FPGAs. IEEE Transactions on Neural Networks and Learning Systems 2018; 30 (2): 326-342.

[43] Amos R, Jagath C. FPGA Implementations of Neural Networks. Springer New York, NY, USA: 2006.

[44] Songlin S, Jiaqi Z, Zixuan Z, Shaokang W. Experience of PYNQ. Springer Singapore, Singapore: 2023.