SCAN: A Scalable Neural Networks Framework Towards Compact and Efficient Models

Linfeng Zhang[1], **Zhanhong Tan**[1], **Jiebo Song**[2], **Jingwei Chen**[3]
Chenglong Bao[*4], **Kaisheng Ma**[*1]
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
[4]Yau Mathematical Sciences Center, Tsinghua University
[2]Institute for Interdisciplinary Information Core Technology, [3]HiSilicon
{zhang-lf19, tanzh19}@mails.tsinghua.edu.cn
{kaisheng, clbao}@mail.tsinghua.edu.cn
songjb@iiisct.com, jean.chenjingwei@hisilicon.com

Abstract

Remarkable achievements have been attained by deep neural networks in various applications. However, the increasing depth and width of such models also lead to explosive growth in both storage and computation, which has restricted the deployment of deep neural networks on resource-limited edge devices. To address this problem, we propose the so-called SCAN framework for networks training and inference, which is orthogonal and complementary to existing acceleration and compression methods. The proposed SCAN firstly divides neural networks into multiple sections according to their depth and constructs shallow classifiers upon the intermediate features of different sections. Moreover, attention modules and knowledge distillation are utilized to enhance the accuracy of shallow classifiers. Based on this architecture, we further propose a threshold controlled scalable inference mechanism to approach human-like sample-specific inference. Experimental results show that SCAN can be easily equipped on various neural networks without any adjustment on hyper-parameters or neural networks architectures, yielding significant performance gain on CIFAR100 and ImageNet. Codes are be released on `https://github.com/ArchipLab-LinfengZhang/pytorch-scalable-neural-networks`.

1 Introduction

Recently deep learning has evolved to become one of the dominant techniques in areas like natural language processing [6, 2] and computer vision [23, 22]. To achieve higher accuracy, over-parameterized models [26, 32] have been proposed at the expense of explosive growth in storage and computation, which is not available for certain application scenes such as self-driving cars and mobile phones. Various techniques have been utilized to address this problem, including pruning [8, 9], quantization [4, 24], lightweight neural networks [12] design and knowledge distillation [11, 25, 1].

Another rising star in this domain named scalable neural networks has attracted increasing attention due to its effectiveness and flexibility [37, 20, 17]. The scalability of neural networks refers to its ability to adjust the trade-offs between response time and accuracy on the fly. As a result, scalable neural networks can always accomplish inference in budgeted and limited time, which is important for real-world applications. Researchers have explored scalability through the lens of depth (layers) and width (channels). Built upon DenseNet [14], MSDNet [13] directly trains multiple classifiers

[*] Corresponding Authors.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

from features at different levels according to their depths. Yu *et al.* proposes switchable batch normalization which enables neural networks to work with arbitrary channels [34]. However, most existing scalable neural networks still suffer from two drawbacks. Firstly, in MSDNet, multiple classifiers which share the same backbone neural network interfere with each other, leading to accuracy loss compared with training them individually. Secondly, in slimmable neural networks, computation of narrow classifiers can't be reused by wide classifiers, which means the inference of wide classifiers has to predict from scratch, increasing the inference time.

In this paper, we propose SCAN, a scalable neural network framework to overcome aforementioned difficulties. By dividing neural networks according to its own depth straightforwardly, the computation of each classifier can be shared. Through knowledge distillation and attention mechanism, multiple classifiers in the same backbone networks can benefit from each other instead of creating negative interaction. Substantial experimental results show that SCAN is a generic and effective neural networks framework which can be easily equipped with various neural networks without any adjustment in architectures or hyper-parameters.

The proposed SCAN framework is inspired by human vision systems. When a human being is asked to identify some images, most of easy images can be recognized instantly. Only fuzzy or easy-to-mix challenging images require further consideration. In SCAN framework, images which are easy to be classified are predicted by shallow (shallow classifier) classifiers, consuming extremely little computation. Deep classifiers only involve in the prediction of challenging samples. Compared to traditional neural networks in which all the samples are treated with equal efforts, SCAN can obtain a high ratio acceleration through human-like sample-specific dynamic inference.

The contributions of this paper are summarized as follows:

- To the best of our knowledge, this is the first work combining model compression and acceleration with attention mechanism, which provides a novel choice for lightweight models design. Compared to existing lightweight design, the proposed mechanism is more hardware friendly with features of input-specific adaptive networks and reuse of the backbone. To verify its effectiveness and generalization, SCAN is evaluated on various neural networks and datasets without adjustment in hyper-parameters and networks architectures.
- Through this method, a two-fold improvement can be achieved on either accuracy or acceleration. Firstly, significant accuracy gain can be observed, especially on the shallow classifiers. Secondly, a high ratio of acceleration can be obtained via scalable inference mechanism.

2 Related work

Adaptive computation graph: Adaptive computation graph is proposed to attain flexible and dynamic neural networks acceleration [13, 15, 29, 31]. Compared with constant and static computation graph, it can meet various demands from diverse application scenes and inputs. SkipNet [29] targets to skip redundant layers in over-parameterized models like ResNet. A reinforcement learning based on auxiliary gating module is proposed to decide whether to skip or not. BlockDrop [31] refines this method by producing a sample-specific dropping strategy, further enhancing acceleration.

However, serious complication hides behind. Complex construction of skipping or dropping path severely prevents hardware from better coordination, which makes aforementioned algorithm counterproductive. To address this problem, the proposed SCAN simply divides the neural networks into 3 or 4 sections, which can be constructed as static graphs individually. The scalability and flexibility only exist among them instead of inside them, which is not only harmonious with hardware but also gains a higher acceleration ratio.

Attention mechanism: Attention mechanism of neural networks has been extensively utilized in various fields of deep learning, yielding state-of-the-art results. It facilitates neural networks to focus on valuable information of inputs, aiming to avoid interference from redundant messages. Firstly, proposed in machine translation, attention mechanism aims to align the words in the source language and target language [2]. Then, it has been applied in other application of natural language processing [27, 30, 6], evolving to an indispensable module in neural networks architectures. Motivated by its success in NLP, attention mechanism has also been employed in computer vision

tasks. Conspicuous performance gain has also been observed in images recognition [28], images caption [33], and fine-grained classification [7].

In this paper, a simplified squeezing-expansion attention module has been proposed to facilitate the training of shallow classifiers, improving the accuracy of shallow classifiers in various networks.

Model compression: The phenomenon that over-parameterized models can't be deployed on edge devices for their excessive requirements of storage and computation has stimulated research of models compression and acceleration. The typical methods include pruning, quantization, compact models design and knowledge distillation. Pruning [8, 9] is to cut off the redundant connections or channels in pre-trained neural networks. Quantization [24, 4] targets at replacing the 32 bits float numbers with fewer bits. Knowledge distillation [11, 3, 39, 36, 1] aims to transfer the knowledge of over-parameterized models to a small model in order to approach higher accuracy with fewer parameters and computation. In addition, some researchers try to design compact models [12, 16] which has fewer parameters yet still a high accuracy.

Aforementioned work has thoroughly exploited the redundancy in weights and neural networks architectures yet ignored the drawback that samples with diverse difficulty are treated equally. The proposed SCAN framework is orthogonal and complementary to aforementioned work, targeting at exploring more acceleration possibility through unbalanced sample-specific disposal.

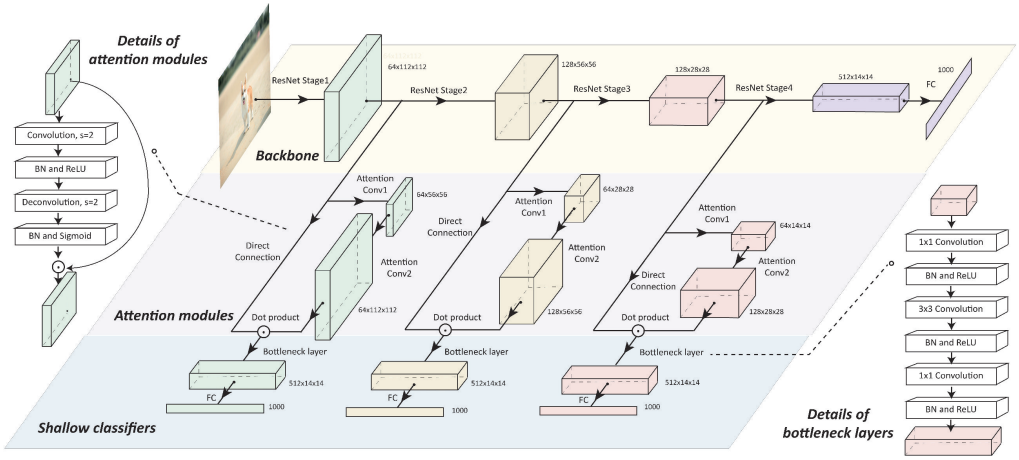

Figure 1: The architecture of ResNet18 equipped with SCAN. (i) The whole neural networks can be divided into three sections: backbone, attention modules and shallow classifiers. (ii) The backbone section is just identical to the origin model. (iii) Additional attention modules are attached after the intermediate features of backbone.(iv) Features refined by attention modules will be feed into the shallow classifiers, which consist of a bottleneck layer and a fully connected layer.

3 SCAN framework

In this section, we introduce the proposed SCAN architecture as 3 parts, as shown in Figure 1. Firstly, to obtain human-like scalable inference, classifiers with varying response time are indispensable. According to self distillation [38], a bottleneck layer and a fully connected layer are organized as shallow classifiers. In addition, knowledge distillation is utilized to facilitate the training of shallow classifiers, which will be further introduced in Section 3.1.

Secondly, although shallow classifiers permit instant prediction, it also leads to dramatic decline on accuracy. To address this problem, a simplified attention module is proposed to compensate for the accuracy of shallow classifiers, which will be further introduced in Section 3.2.

We further propose a threshold-based strategy to manage all the shallow classifiers to corporate together. Moreover, a genetic algorithm is designed to search for proper thresholds, which will be brought forth in Section 3.3.

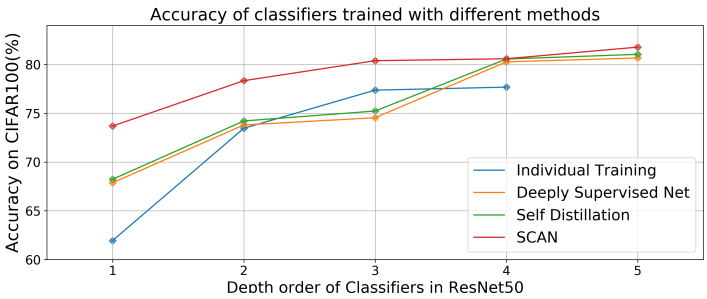

Figure 2: The accuracy of classifiers trained with different methods.

3.1 Self distillation

Self distillation provides an effective method to construct and train shallow classifiers which share the same backbone neural network. It firstly divides neural networks into several sections depending on their depth. Then a bottleneck layer and fully connected layer are attached after the intermediate features as shallow classifiers, which are regarded as the student models in knowledge distillation. In the training period, the knowledge of the deepest classifier is distilled into each shallow classifier, whose function loss can be written as

$$loss = \sum_{i=1}^{C} loss_i = \sum_{i=1}^{C} \Big((1-\alpha) \cdot CrossEntropy(q^i, y) + \alpha \cdot KL(q^i, q^C) + \lambda \cdot ||F_i - F_C||_2^2 \Big) \quad (1)$$

where C denotes the number of classifiers. q^i and q^C represent the outputs of softmax in the i_{th} classifier and the deepest classifier respectively. y represents corresponding labels. F_i and F_C signify the feature maps in the i_{th} classifier and the deepest classifier respectively. $CrossEntropy$, KL denote the well-known cross entropy loss and Kullback–Leibler divergence respectively. Experiments results show that significant performance gain can be observed on not only shallow classifier but also the deepest classifiers. Motivated by its impressive achievement, self distillation is utilized to construct and train the shallow classifiers in SCAN.

3.2 Attention modules

Figure 2 provides accuracy comparison of four methods training shallow classifiers in ResNet50 on CIFAR100. The X axis is the depth of classifiers, where x=5 indicates the ensemble of all the classifiers [19], and Y axis denotes accuracy. It's observed that evident accuracy decay can be observed as the depth of classifiers decreases. For example, 13% and 8% drop on accuracy exists on the shallowest and second shallowest classifier in self distillation. Moreover, as depicted in Figure 2, the 3_{th} classifier of self distillation and DSN [21] is lower than individual training, which may be caused by the negative interaction among classifiers in one backbone neural network [13]. Features desired by different classifiers are mixed up in the sharing backbone neural network. It's unattainable for each classifier to detach its own features automatically. To address this problem

Algorithm 1 Scalable Inference

Input: Samples X, Thresholds $\Sigma = \{\sigma_i\}^N$, Classifiers $C = \{c_i\}^N$, Multi-classifiers model M
Output: Predicted labels Y

Y := None
for i from 1 to N **do**
    logit = M.getSoftmaxOutputs(X, i)
    if max(logits) > σ_i **then**
        Y := argmax(logits)
        Break
if Y = None **then**
    Y := M.getEnsemblePrediction()
return Prediction

and further enhance the performance of shallow classifiers, attention modules are utilized to obtain classifier-specific features from the sharing backbone neural network. Inspired by RAN [28], we propose a simplified attention modules including one convolution layer for downsampling and one deconvolution layer for upsampling. A sigmoid activation is attached after attention modules to obtain attention maps between 0 and 1. Then, the attention maps are involved in a dot product operation with origin features, yielding classifier-specific features. Its forward computation can be formulated as

$$Attention\ Maps(W_{conv}, W_{deconv}, F) = \sigma(\phi(\psi(F, W_{conv})), W_{deconv}) \tag{2}$$

where ψ and ϕ denote convolution function and deconvolution function respectively. F represents the input features and σ signifies a sigmoid function. Notes that batch normalization and ReLU activation function after convolution and deconvolution layers are omitted here.

Experiments results demonstrate that attention modules in SCAN lead to dramatic accuracy boost in shallow classifiers. For instance, 5.46%, 4.13% ,and 5.16% accuracy gain can be observed on the shallow classifiers in ResNet50 on CIFAR100, compared with self distillation [38].

3.3 Scalable inference mechanism

It is generally acknowledged that the prediction of neural networks with a higher confidence (softmax value) is more likely to be right. In this paper, we exploit this observation to determine whether a classifier gives a right or wrong prediction. As described in Algorithm 1, we set different thresholds for shallow classifiers. If the maximal output of softmax in shallow classifier is larger than the corresponding threshold, its results will be adopted as the final prediction. Otherwise, the neural networks will employ a deeper classifier to predict, until the deepest one or the ensemble prediction. Because most of the computation for shallow classifiers is included by that for deep classifiers, there is no much extra computation introduced.

However, threshold controlled scalable inference causes another problem, that is the choice of thresholds for difference classifiers. Thresholds matters: (i) A lower threshold for shallow classifiers results in that most samples will be predicted by shallow classifiers, indicating more rapid response yet lower accuracy. (ii) Similarly, a higher threshold leads to a phenomenon that most samples will be determined by deeper classifiers, indicating precise prediction yet longer response time. (iii) By adjusting the value of thresholds, flexible accuracy-response time trade-offs can be approached on the fly. Instead of designing thresholds manually, we propose a genetic algorithm based method to search the most optimal thresholds as is depicted in Algorithm 1.

Genes Coding: Genes, a binary sequence is ought to be decoded into its corresponding threshold. To guarantee the accuracy of the model, we empirically restrict the lower bound for thresholds as 0.70. Its decode function can be formulated as

$$\sigma_i = 1 - \frac{0.3}{N} \cdot \sum_{n=1}^{N} S(n) = 1 \tag{3}$$

Here $S(n)$ indicates the n_{th} bit in the sequence of genes. σ denotes the threshold of i_{th} gene. N is the length of binary sequence utilized to express one threshold. The more bits "1" there are in this sequence, the lower the threshold is.

Fitness: Another crucial issue is to choose the metrics for computing the fitness for thresholds. Targeting at accelerating the models and improving the performance at the same time, these two elements are taken into consideration for the fitness metrics, which can be formulated as

$$fitness = acceleration\ ratio + \beta \cdot (accuracy - baseline) \tag{4}$$

where β is a hyper-parameter to balance the impact of these two elements. Adjustment of β leads to trade-offs between accuracy and acceleration.

4 Experiments

We evaluate SCAN on two benchmark datasets: CIFAR100 [18] and ImageNet (ILSVRC2012) [5] and three kinds of neural networks with difference depth and width: VGG [26], ResNet [10] and Wide ResNet [35]. During training periods, common techniques like data argumentation (random

Algorithm 2 Threshold Searching

Input: Genes G, Multi-classifiers model M, Dataset D , Generations g
Output: Optimal Thresholds $\Sigma = \{\sigma_i\}^N$

RandomlyInitialize(G)
for i from 1 to g **do**
    fitness = getFitness(G, D, M) // Calculate fitness of each gene according to Equation 4.
    G := weightedSelect(G, fitness) // Drop the genes with low fitness.
    G := crossover(G) // Each two genes cross over, generating new genes.
    G := mutate(G) // Each bit of genes may mutate with a low possibility.
Σ := decode(G) // Decode the genes into thresholds according to Equation 3.
return Σ

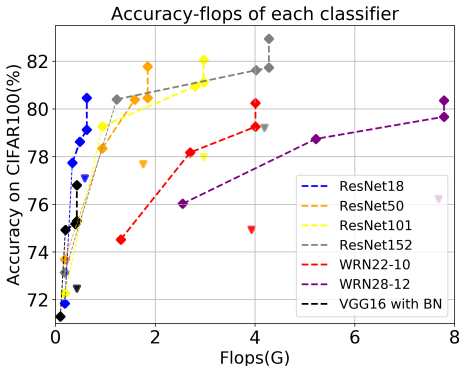

Figure 3: Accuracy and computation of each classifier on CIFAR100.

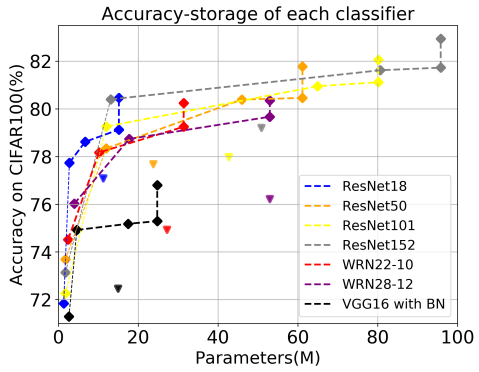

Figure 4: Accuracy and storage of each classifier on CIFAR100.

cropping and horizon flipping), learning rate decay, l_2 regularization are equipped. To fit the size of tiny images in CIFAR, we slightly adjust the kernel size and strides of convolution and pooling layers. The recommended value for hyper-parameters λ and α in Equation 1, and N in Equation 3 are 0.5, 5e-7 and 30, respectively. Note that the reported ImageNet (ILSVRC2012) accuracy is evaluated on validation set. All the experiments are conducted by PyTorch1.0 on GPU devices.

4.1 Results on CIFAR100

Experiments results on CIFAR100 are depicted in Figure 3 and 4. The squares connected in the same line from left to right denote the classifier¼ to classifier¼ and their ensemble. All of them share the same backbone neural network. The triangle with the same color out of lines denotes the corresponding baseline.

It is observed that (i) In all the situations, the classifier¾ equipped with SCAN outperforms its baseline. (ii) 2.17X acceleration and 3.20X compression have been achieved on average with no accuracy drop. (iii) Compared with the corresponding baseline, 4.05% accuracy increment can be obtained with 4.4% relative computation increment on average. (iv) The ensemble of all classifiers in one neural network leads to 1.11% performance gain with almost no incremental computation and storage. (v) Shallow classifiers benefit more from SCAN than deeper classifiers. (vi) Deeper or wider neural networks benefit more from SCAN than shallower or thinner neural networks.

4.2 Results on ImageNet

As shown in Table 4.1, the same tendency of accuracy increment on all the classifiers can also be observed: (i) On average, 1.26% accuracy gain on ImageNet can be achieved, varying from 1.41% on ResNet50 as maximum to 1.08% on ResNet101 as minimum. (ii) The depth of classifiers impacts their accuracy more significantly, which indicates there is less redundancy in ImageNet compared with CIFAR100.

Table 1: Experiments results of accuracy (%) on CIFAR100.

Models	Baseline	Classifier1/4	Classifier2/4	Classifier3/4	Classifier4/4	Ensemble
VGG16(BN)	72.46	71.29	74.92	75.18	75.29	76.80
VGG19(BN)	72.25	71.52	74.02	74.15	74.43	75.43
ResNet18	77.09	71.84	77.74	78.62	79.13	80.46
ResNet50	77.68	73.69	78.34	80.39	80.45	81.78
ResNet101	77.98	72.26	79.26	80.95	81.12	82.06
ResNet152	79.21	73.14	80.40	81.73	81.62	82.94
WRN20-8	74.61	74.52	78.17	79.25	/	80.04
WRN44-8	76.22	76.02	78.74	79.67	/	80.35

Table 2: Experiments results of accuracy (%) on ImageNet.

Models	Baseline	Classifier1/4	Classifier2/4	Classifier3/4	Classifier4/4
ResNet18	68.02	48.25	58.00	65.32	69.32
ResNet50	74.47	53.86	66.54	73.57	75.88
ResNet101	75.24	52.32	65.33	74.51	76.32

4.3 Results of scalable inference

Experiments results for scalable inference are depicted in Figure 5. The X axis is the acceleration ratio compared to its baseline. The Y axis is the top 1 accuracy evaluated on CIFAR100 and ImageNet. The squares connected in the same line denotes the results of different thresholds for the same neural network. The triangles on x=1 denote the baselines of different models.

It is observed that:(i) Compared with deploying one of the shallow classifier individually, scalable inference leads to a higher acceleration ratio. (ii) Compared with baselines on CIFAR100, 4.41X acceleration can be achieved with no drop on accuracy on average, varying from 2.41X on ResNet18 as minimum to 6.23X on ResNet152 as maximum. (iii) Compared with baselines on ImageNet, 1.99X acceleration can be achieved with no accuracy drop on average, varying from 1.54X on ResNet50 as minimum to 2.43X on ResNet101 as maximum. (iv) More accuracy gain can be observed on deeper neural networks which is in accordance with the general observation that over-parameterized models have more potential to be compressed and accelerated.

5 Discussion

What have attention modules learned? In SCAN, attention modules are introduced to obtain classifier-specific features, leading to significant performance gain on shallow classifiers. We further visualize the spatial attention maps as depicted in Figure 7. The heat maps indicate learned attention maps, where the value of each pixel is computed as the mean value of pixels in the same position of all channels.

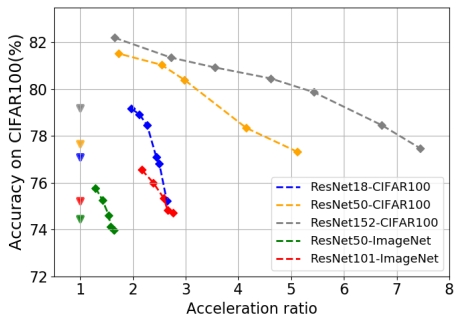

Figure 5: Acceleration ratio and accuracy of different thresholds.

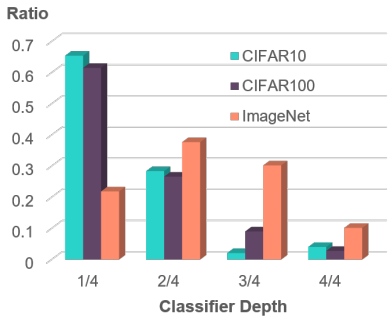

Figure 6: Statistics of samples predicted in each classifier on three kinds of datasets.

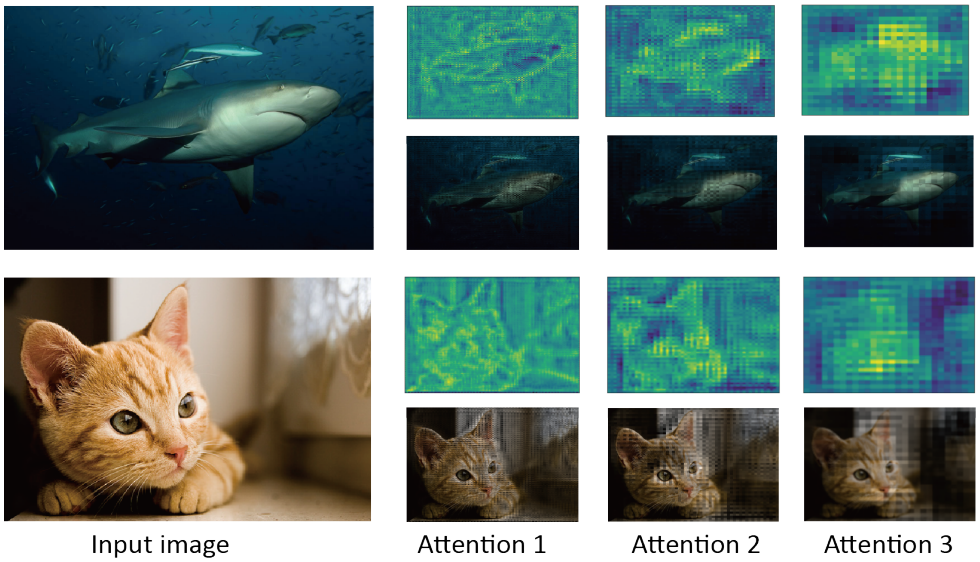

Figure 7: Visualization of attention maps in shallow classifiers.

As is depicted in Figure 7, all the classifiers pay their attention on the same spatial position - the bodies of a shark and a cat, while ignoring the backgrounds, which indicates that all of the attention modules have learned to find the most informative pixels. The attention maps in classifiers ¼ seems to concentrate on the details of shark's and cat's features such as their outlines. In contrast, the attention maps in deeper classifiers ¾ focus more on the texture features, which indicates deep classifiers that have a larger receptive filed are more likely to predict based on global and low frequency information while shallow classifiers incline to be dominated by local and high frequency information.

How many samples are predicted by shallow classifiers? The determinants of the acceleration ratio in SCAN is the number of samples predicted by shallow classifiers, which varies from thresholds, datasets and neural networks. Figure 6 shows the statistics of samples predicted by each classifier of ResNet18 on CIFAR10, CIFAR100 and ImageNet with the same thresholds. It's observed that:(i) More than half samples in CIFAR10 and CIFAR100 can be classified in the shallowest classifier, which consumes the least computation compared with others. (ii) In ImageNet, more samples have to be predicted in the last two classifiers, which indicates the classification of ImageNet data is beyond the capacity of shallow classifiers. Based on these observation, two possible usage are proposed as follows:

Firstly, the number of samples predicted in different classifier can be utilized as a guidance of models compression. For example, the classifier¼ in Figure 6 provides extremely little valid prediction in CIFAR10 and CIFAR100, indicating there is much more redundancy and compression potential.

Secondly, it also can be utilized as a metric if datasets difficulty. It's not rigorous to measure the difficulty of datasets by prediction accuracy because different datasets consist of different numbers of categories. SCAN provides a possible solution - the ratio of samples predicted by shallow classifiers.

Future works: Although SCAN has achieved significant acceleration and boost on accuracy, we still believe it has more potential. Firstly, more creditable judgement of whether the prediction of shallow classifiers should be adopted remains to be explored. This issue can be formulated as a binary classification problem which may be addressed by machine learning algorithms.

Secondly, continued optimization on the structure of shallow classifiers is necessary. The success of attention modules in SCAN proves that tiny adjustment on shallow classifiers can lead to dramatic accuracy boost, indicating that more compact and efficient shallow classifiers can be achieved by a well designed structure.

6 Conclusion

We have proposed a novel neural networks training and inference framework named SCAN, whose benefits can be seen in three folds: Firstly, self distillation and attention modules are utilized to train compact and efficient shallow classifiers to achieve static acceleration and compression. Secondly, SCAN exploits the diversity in prediction difficulty to accomplish human-like sample-specific conditional execution, yielding scalability and a high acceleration ratio. Thirdly, compared with self distillation and its corresponding baseline, more significant accuracy gain can be achieved on the ensemble of all classifiers with a negligible growth on computation and storage.

7 Acknowledgement

This paper is supported by Institute for Interdisciplinary Information Core Technology, Beijing Academy of Artificial Intelligence and Zhongguancun Haihua Institute for Frontier Information Technology.

References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NeurIPS*, pages 2654–2662, 2014.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[3] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *SIGKDD*, pages 535–541, 2006.

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, pages 3123–3131, 2015.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2018.

[7] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, volume 2, page 3, 2017.

[8] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[9] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, pages 1135–1143, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS*, 2014.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *CVPR*, 2017.

[13] 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 *ICLR*, 2018.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[15] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661, 2016.

[16] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. In *ICLR*, 2016.

[17] Eunwoo Kim, Chanho Ahn, and Songhwai Oh. Nestednet: Learning nested sparse structures in deep neural networks. In *CVPR*, pages 8669–8678, 2018.

[18] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[19] Ashish Kumar, Saurabh Goyal, and Manik Varma. Resource-efficient machine learning in 2 kb ram for the internet of things. In *ICML*, pages 1935–1944. JMLR. org, 2017.

[20] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

[21] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015.

[22] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37, 2016.

[24] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542, 2016.

[25] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[28] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *CVPR*, pages 3156–3164, 2017.

[29] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, pages 409–424, 2018.

[30] Xuejian Wang, Lantao Yu, Kan Ren, Guanyu Tao, Weinan Zhang, Yong Yu, and Jun Wang. Dynamic attention deep model for article recommendation by learning human editors' demonstration. In *SIGKDD*, pages 2051–2059, 2017.

[31] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, pages 8817–8826, 2018.

[32] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995, 2017.

[33] BaJ XuK, CourvilleA KirosR, et al. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 20148–2057, 2015.

[34] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *ICLR*, 2019.

[35] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[36] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

[37] Amir R Zamir, Te-Lin Wu, Lin Sun, William B Shen, Bertram E Shi, Jitendra Malik, and Silvio Savarese. Feedback networks. In *CVPR*, pages 1308–1317, 2017.

[38] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *arXiv preprint:1905.08094*, 2019.

[39] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, pages 4320–4328, 2018.