Attention Based Pruning for Shift Networks

Ghouthi Boukli Hacene, Carlos Lassance, Vincent Gripon, Matthieu Courbariaux, Yoshua Bengio

Attention Based Pruning for Shift Networks

Ghouthi Boukli Hacene[1,2], Carlos Lassance[1,2], Vincent Gripon[1,2]
[1] Université de Montréal, MILA, [2]IMT Atlantique, Lab-STICC

***Abstract*—In computer vision and many other domains, Convolutional Layers (CLs) are the key to the accuracy and performance of deep learning methods. However, it is often required to assemble a very large number of CLs with tons of parameters to reach state-of-the-art accuracy, thus resulting in complex and demanding systems that are poorly fitted to resource-limited devices. Recently, methods have been proposed to replace the convolution by the combination of a shift operation and a 1x1 convolution. The result operation, called Shift Layers (SLs) is an efficient alternative to CLs that allow to reach similar accuracy on various tasks with faster computations and less parameters. In this contribution, we introduce Shift Attention Layers (SALs), a particular version of SLs using an attention mechanism that learns which shifts are the best at the same time the network function is trained. We demonstrate SALs are able to outperform vanilla SLs in various tasks in vision while significantly reducing computations and parameters for the inference.**

I. Introduction

Convolutional Neural Networks (CNNs) are the state-of-the-art in many challenges in computer vision, such as image classification, object detection and face recognition [20]. To achieve top-rank accuracy, CNNs rely on a the use of a very large number of trainable parameters, and considerable computational complexity. This is why there has been a lot of interest in the past few years towards the compression of CNNs, so that they can be deployed in embedded systems. The main purpose of compressing DNNs is to reduce memory footprint and/or computational complexity while having a limited impact on accuracy.

Prominent works to compress neural networks include binarizing (or quantifying) weights and activations [2], [19], [16], [15], [13], [**?**], [25], pruning network connections during or before training [6], [5], [14], [1], using grouped convolutions [**?**], [8], [17], introducing new components [29], [12], using convolutional decomposition [21], [18], or searching for a lightweight neural network architectures [10].

Recently, the authors of [24], [4] have proposed to replace the convolution operator with the combination of shifts and 1x1 convolutions. In their work, the shifts are hand-crafted and all parameters to be trained are in the convolutions. These operations are particularly well fitted to embedded devices [4], [27]. In [11], the authors have proposed to learn shifts by making them differentiable. Their solution implies to use interpolation of pixel and neuron values throughout the CNN architecture.

In this paper, we introduce the Shift Attention Layer (SAL), which can be seen as an alternative to [11]. This layer starts with a vanilla convolution and learns to transform it into a shift layer throughout the training of the network function. It uses an attention mechanism [23] that selects the best shift for each feature map of the architecture. We demonstrate it is able to significantly outperform original shift layers [24] and the method introduced in [11], even though it requires more parameters during training. It is thus of particular interest for implementing the inference process on resource-limited systems.

The outline of the paper is as follows. in Section II we present the related works. In Section III we explain the proposed method. In Section IV we perform experiments using challenging computer vision datasets. Finally, In Section V we discuss future work and conclude.

II. Related Work

In this section we introduce works aiming at reducing complexity and memory footprints of CNNs.

In a first line of works, authors have proposed to binarize weights and activations [2], [3], [16], and then replace all multiplication operations by low cost multiplexers. Other works have proposed to use K-means to quantize weights [5], [22], [25] and thus reduce CNN model size.

Another line of work is to rely on network pruning [6]. For example Li et al [14] use the sum of absolute weights of each channel to select and prune redundant ones. In the same vein, He et al [7] proposed a novel Soft Filter Pruning (SFP) approach to prune dynamically filters in a soft manner. Using the reconstruction error of the last layer before classification, Yu et al [28] introduce the Neuron Importance Score Propagation (NISP) method to estimate the importance of each neuron in each layer in the backward propagation of the scores obtained from the reconstructed error. Huang et al [9] propose a reinforcement learning based method, in which an agent is trained to maximise a reward function in order to improve the accuracy when pruning selected channels. Yamamoto et al [26] use a channel-pruning technique based on attention statistics by adding attention blocks to each layer and update them during training process to evaluate the importance of each channel.

For mobile applications with limited resources, specific neural network architectures have been introduced. Iandola et al [10] propose a fire module to define SqueezeNet, a very small neural network with an acceptable accuracy. Howard et al [8] use depthwise separable convolutions to define MobileNet and build lightweight CNNs.

Other approaches focus on decomposition of convolution. Simonyan et al [18] reduce the number of parameters of VGG by replacing a 5×5 convolution by two 3×3 convolutions. In [20], the authors decompose 7×7 convolutions into 1×7 and 7×1 convolutions. More closer to our proposed work, Wu et al [24] introduce a shift operation as an alternative to

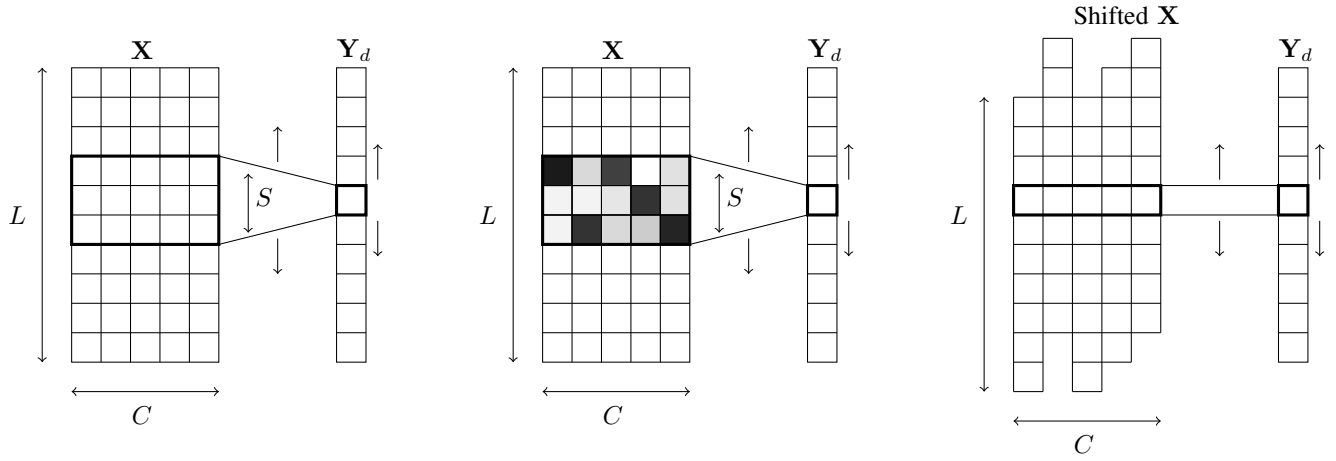

Fig. 1. Overview of the proposed method: we depict here the computation for a single output feature map d. The figure on the left represents a standard convolutional operation: the weight filter $\mathbf{W}_{d,\cdot,\cdot}$ containing SL weights is moved along the spatial dimension of the input to produce each output in $\mathbf{Y}_d$. In the middle figure, we depict the attention tensor $\mathbf{A}$ on top of the weight filter: the darker the cell, the most important the corresponding weight has been identified to be. At the end of the training process, $\mathbf{A}$ should contain only binary values with a single 1 per slice $\mathbf{A}_{d,c,\cdot}$. In the right figure, we depict the corresponding obtained shift layer: for each slice along the input feature maps, the cell with the highest attention is kept and the others are disregarded. As a consequence, the initial convolution with a kernel size S has been replaced by a convolution with a kernel size 1 on a shifted version of the input $\mathbf{X}$.

spatial convolutions. The authors deconstruct 3×3 spatial convolution into 1×1 convolution and a shift operation to reduce the complexity and memory usage of CNNs. In [11], the authors propose an active shift layer (ASL) to improve the accuracy of the shift operation method, and replace the handcrafted shifts by learnable parameters which are optimised during back-propagation.

In this contribution, we introduce Shift Attention Layer (SAL), a novel pruning-shift operation method. SAL uses pruning concept in such a way to not only reduce the memory footprint and the number of operations in CNNs, but also to replace the convolution by only a shift operation followed by a 1×1 convolution, and thus reduces CNNs memory and complexity, and make easier the implementation of large CNNs on embedded systems.

III. Methodology

In this section, we first review the classical spatial convolution operation and see how it can be related to the shift operation defined in [24]. We then introduce our proposed method.

A. Convolution/Shift operation

Let us consider the 1d case (other cases can be easily derived). Let us denote by $\mathbf{X} \in \mathbb{R}^{C \times L}$ an input tensor of a convolutional layer, where C is the number of channels and L is the dimension of each feature map. We denote $\mathbf{W} \in \mathbb{R}^{D \times C \times S}$ its weight tensor, where D is the number of output channels, S the kernel size, and $\mathbf{Y} \in \mathbb{R}^{D \times L}$ its output tensor. Disregarding downsampling and padding (i.e. border effects), the convolution operation is defined through Equation (1) and depicted in Figure 1: left.

$$y_{d,\ell} = \sum_{c=1}^{C} \sum_{\ell'=1}^{S} x_{c,\ell+\ell'-\lceil S/2 \rceil} w_{d,c,\ell'}, \forall d, \ell \ . \tag{1}$$

A shift layer is obtained when connections in $\mathbf{W}$ are pruned so that remains exactly one connection for each slice $\mathbf{W}_{d,c,\cdot}, \forall d, c$. We denote $\ell_{d,c}$ the index such that $w_{d,c,\ell_{d,c}}$ is not pruned. Then the shift operation is defined in Equation (3) and depicted in Figure 1: right.

$$y_{d,\ell} = \sum_{c=1}^{C} x_{c,\ell+\ell_{d,c}-\lceil S/2 \rceil} w_{d,c,\ell_{d,c}} \tag{2}$$

$$= \sum_{c=1}^{C} \tilde{x}_{c,\ell} \tilde{w}_{d,c} \ , \tag{3}$$

where $\tilde{x}_{c,\ell} = x_{c,\ell+\ell_{d,c}-\lceil S/2 \rceil}$ and $\tilde{w}_{d,c} = w_{d,c,\ell_{d,c}}$. We observe that the shift operation boils down to shifting the input tensor $\mathbf{X}$ then convoluting with a kernel of size 1 (in the equation, the kernel $\tilde{\mathbf{W}}$ is indexed only by the input and output feature maps). In the original work [24], the authors proposed to arbitrarily predetermine which shifts are used before training the architecture. To the contrary, in this work, we propose a method that learns which shifts to perform during the training process.

B. Shift Attention Layer (SAL)

The idea we propose is to enrich a classical convolution layer with a selective tensor $\mathbf{A}$ which aims at identifying which connections should be kept in each slice of the weight tensor. As such, we introduce $\mathbf{A} \in \mathbb{R}^{D \times C \times L}$ a tensor containing as many elements as weights in the weight tensor. Each value of $\mathbf{A}$ is normalised between 0 and 1 and represents how important the corresponding weight in $\mathbf{W}$ is (c.f. Figure 1: middle). At

the end of the training process, $\mathbf{A}$ becomes binary, with only one nonzero element per slice $\mathbf{A}_{d,c,\cdot}$, corresponding to the weights in $\mathbf{W}$ that should be kept.

More precisely, each slice $\mathbf{A}_{d,c,\cdot}$ is normalised using a softmax function with temperature T. The temperature is increased smoothly along the training process. Note that in order to force the mask $\mathbf{A}$ to be selective, we first normalise each slice $\mathbf{A}_{d,c,\cdot}$ so that it has 1 standard deviation (sd). Algorithm 1 summarises the training process of one layer. At the end of training, the selected weight in each slice $\mathbf{W}_{d,c,\cdot}$ corresponds to the maximum value in $\mathbf{A}_{d,c,\cdot}$.

Algorithm 1 SAL algorithm of one layer

Inputs: Input tensor $\mathbf{X}$,
Initial softmax temperature T, Constant $\alpha > 1$.

for each training iteration **do**
  $T = \alpha T$
  for $d := 1$ to D **do**
    for $c := 1$ to C **do**
      $\mathbf{A}_{d,c,\cdot} = \frac{\mathbf{A}_{d,c,\cdot}}{sd(\mathbf{A}_{d,c,\cdot})}$
      $\mathbf{A}_{d,c,\cdot} = Softmax(\mathbf{A}_{d,c,\cdot}, T)$
    end for
  end for
  $\mathbf{W}_A = \mathbf{W} \cdot \mathbf{A}$ ($\cdot$ is the pointwise multiplication)
  Compute standard convolution as described in Equation 1 using input tensor $\mathbf{X}$ and weight tensor $\mathbf{W}_A$ instead of $\mathbf{W}$.
  Update $\mathbf{W}$ and $\mathbf{A}$ via back-propagation.
end for

Note that contrary to the vanilla shift layers where the same number of shifts is performed in every direction, at the end of the training process the resulting shift layer can have an uneven distribution of shifts, as shown in the next section. This comes with a price, which correspond to the memory needed to retain which shift is performed for each input feature map. In order to be fair, we thus reduce the number of feature maps in the networks we use in the experiments so that the total memory is comparable.

IV. Experiments

In this section we present the benchmark protocol used to evaluate the proposed method, and then compare the obtained performance with CNNs baseline, pruning methods and shift-module based methods.

A. Benchmark Protocol

We compute our evaluation on three vision datasets: CIFAR10, CIFAR100 and ImageNet ILSVRC 2012. To evaluate the proposed method, we test the architecture Resnet-20/56 on CIFAR10 and the architecture Resnet-20/50 on CIFAR100 using the following parameters: the training process consists of 300 epochs for Resnet-20 and 400 epochs for Resnet-50/56, the learning rate starts at 0.1 and is divided by 10 after each 100 epochs. For all these evaluations, the training batch size is 128, the initial/final softmax temperature are $0.15/50$, and the temperature is increased at each step (each time a mini batch is processed).

For ImageNet ILSVRC 2012, we tested Resnet-w32 defined in [11] and Resnet-50 using the following parameters: the total number of epochs is 90, the training batch size is 256, the learning rate starts at 0.1 and is divided by 10 after each 60 epochs, initial/final softmax temperature are $0.15/60$.

B. Results

The first experiment is performed on CIFAR10 and CIFAR100, and compares the performance of the proposed method with shift-based module methods (c.f. Table I) and pruning methods(c.f. Table II). Table I shows that our method achieves a better accuracy with fewer parameters than the baseline and other shift-module based method. Table II shows that the proposed method is comparable in term of accuracy and number of parameters/floating point operations (FLOPs) with other pruning methods.

As a second evaluation, we plot the average of $\mathbf{A}$ at the end of the learning process along the channel dimension. As such, we observe the proportion of each position in slices $\mathbf{A}_{d,c,\cdot,\cdot}$ that are kept at the end of the training process. In Figure IV-B, a heat-map represents the amount of kept weights through $\mathbf{W}_{d,c,\cdot,\cdot}, \forall d, c$ slices, for the 4 first layers (first row), and the 4 last layers (second row), of Resnet-20 trained on CIFAR10, and where attention tensors $\mathbf{A}$ values are initially drawn uniformly at random. We observe that at the end of the training process, initial layers seem to yield a uniform distribution of kept weights. To the contrary, for the last layers, there is a clear asymmetry that favours corners. This interestingly suggests that shift-layers would benefit from a non regular number of shifts in each direction.

To see how much the initialisation of $\mathbf{A}$ is related to the amount of kept weights, we then perform another experiment where $\mathbf{A}$ is initialized uniformly at random but then the centre value $\mathbf{A}_{d,c,\lfloor S/2 \rfloor,\lfloor S/2 \rfloor}$ is changed to the maximum over the corresponding slice $\max(\mathbf{A}_{d,c,\cdot,\cdot})$. We observe in Figure IV-B that almost all kept weights in the first layer are slices centres. Subsequent layers yield an almost uniform distribution, and we observe the same kept weights distribution for the last layers as the previous experiment. This suggests that the initialisation of $\mathbf{A}$ is not the cause for the first layers kept weights distribution. We also plot a heat-map of kept weights in Resnet-56 trained on CIFAR10, and where attention tensor $\mathbf{A}$ values are initially drawn uniformly at random. Figure IV-B shows that for the first layers, the number of kept weights is more important on the centre row than other positions. However, we see on the last layers that there is more kept weights in the corners than other positions.

From all these experiments, we consistently observe that in deeper layers, the method tends to keep more weights in corner positions than others, and this independently from initialisation process or neural network architecture.

In a third experiment, we observe the effect of initial and final temperature choices on accuracy. Figure 5 represents the

TABLE I

COMPARISON OF ACCURACY AND NUMBER OF PARAMETERS BETWEEN BASELINE ARCHITECTURE (RESNET20), SHIFTNET, ASNET, AND SANET (THE PROPOSED METHOD) ON CIFAR10 AND CIFAR100.

	CIFAR10		CIFAR100	
	Accuracy	Params (M)	Accuracy	Params (M)
Baseline	94.66%	1.22	73.7%	1.24
ShiftNet [11]	93.17%	1.2	72.56%	1.23
ASNet [24]	94.53%	0.99	76.73%	1.02
SANet (ours)	**95.52%**	**0.98**	**77.39%**	**1.01**

TABLE II

COMPARISON OF ACCURACY, NUMBER OF PARAMETERS AND NUMBER OF FLOATING POINT OPERATIONS (FLOPS) BETWEEN BASELINE ARCHITECTURE (RESNET-56 FOR CIFAR10, AND RESNET-50 FOR CIFAR100), SANET (THE PROPOSED METHOD) , AND SOME OTHER PRUNING METHODS ON CIFAR10 AND CIFAR100. NOTE THAT THE NUMBER BETWEEN () REFERS TO THE RESULT OBTAINED BY THE BASELINE USED FOR EACH METHOD.

	CIFAR10			CIFAR100		
	Accuracy	Params (M)	FLOPs (M)	Accuracy	Params (M)	FLOPs (M)
Pruned-B [11]	93.06%(93.04)	0.73(0.85)	91(126)	73.6%(74.46)	7.83(17.1)	616(1409)
NISP [24]	93.01%(93.04)	0.49(0.85)	71(126)	–	–	–
PCAS [24]	93.58%(93.04)	0.39(0.85)	56(126)	73.84%(74.46)	4.02(17.1)	475(1409)
SANet (ours)	**93.6**%(93.04)	**0.36**(0.85)	**42**(126)	**77.6**%(78)	**3.9** (16.9)	**251**(1308)

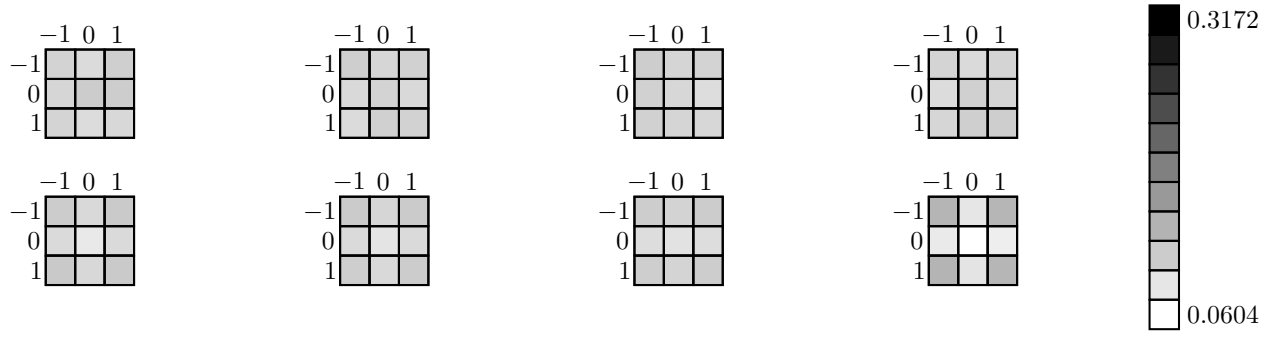

Fig. 2. Heat maps representing the average values in $\mathbf{A}$ for various layers in the Resnet-20 architecture trained on CIFAR10. In this experiment, values in $\mathbf{A}$ are initialized uniformly at random.

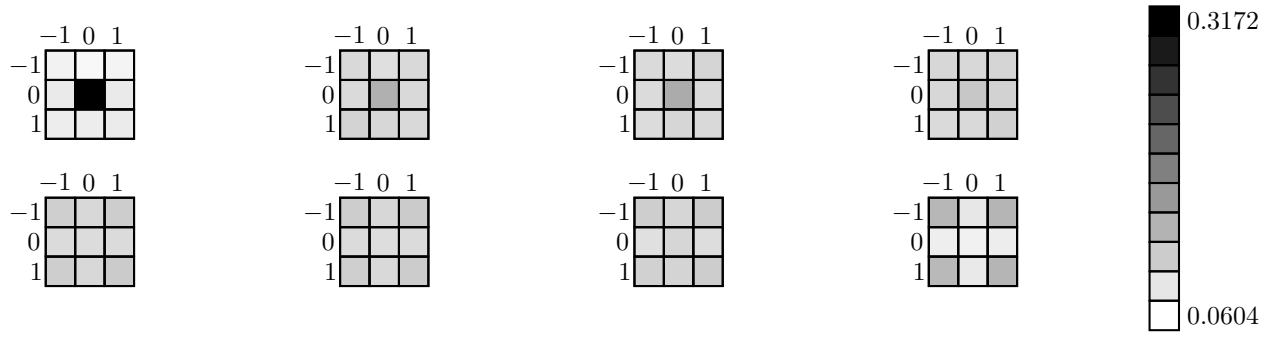

Fig. 3. Heat maps representing the average values in $\mathbf{A}$ for various layers in the Resnet-20 architecture trained on CIFAR10. In this experiment, values in $\mathbf{A}$ are initialized uniformly at random but the centre value that takes the maximum over the corresponding slice.

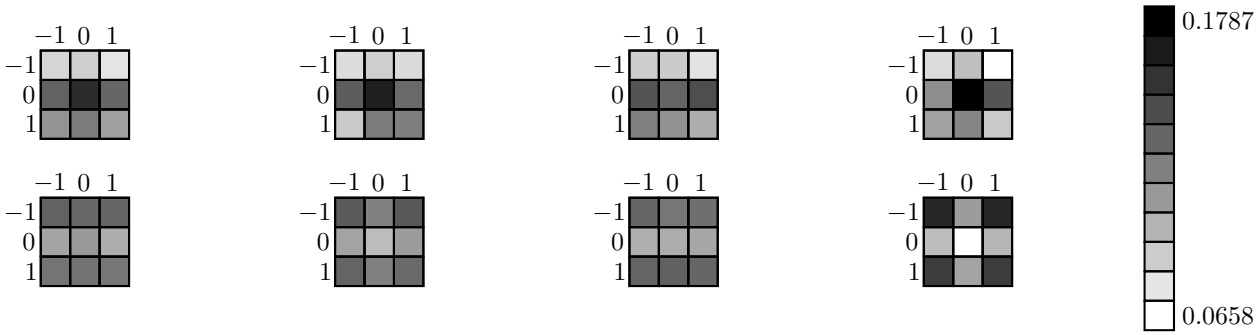

Fig. 4. Heat maps representing the average values in $\mathbf{A}$ for various layers in the Resnet-56 architecture trained on CIFAR10. In this experiment, values in $\mathbf{A}$ are initialized uniformly at random.

evolution of accuracy of Resnet-20 trained on CIFAR10 as function of final temperature while initial temperature is fixed at 0.15. It shows that the accuracy is highly increased when the final temperature value is increased from 1 to 5, then beyond this point, the effect of final temperature is negligible. Note that when the final temperature is small, obtained values in $\mathbf{A}$ at the end of the training process can be far from binary. In all cases, we round the values in $\mathbf{A}$ to the nearest integer before computing the accuracy. This experiment states that final temperature value needs to be big enough so the softmax can push the highest value to 1 and the other values to 0. Figure 6 shows the behaviour of the accuracy of Resnet-20 trained on CIFAR10 when initial temperature is changed and final temperature is fixed at 50. We see an interesting region between 0.1 and 0.15 in which the accuracy is better.

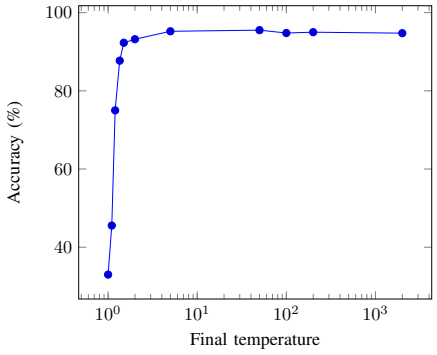

Fig. 5. Evolution of accuracy of Resnet-20 trained on CIFAR10 as function of final temperature.

V. Conclusion

In this contribution, we proposed a novel attention-based pruning method that transforms a convolutional layer into a shift layer. The result network provides interesting improvements in terms of memory and computation time, while keeping high accuracy. Compared to existing methods, we showed that the proposed method results in improved accuracy for similar memory budgets as existing shift-layer based methods, using challenging vision datasets such as CIFAR10/100 and

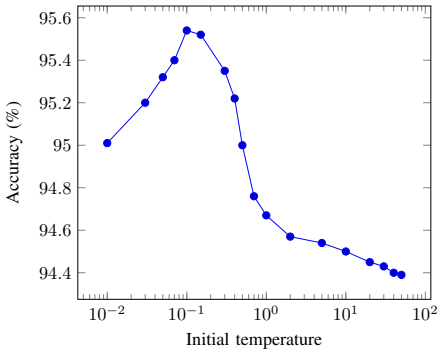

Fig. 6. Evolution of accuracy of Resnet-20 trained on CIFAR10 as function of initial temperature.

Imagenet. It provides an interesting alternative to channel-pruning methods.

Future work will focus on how to reduce complexity of the training process and combine the proposed method with other out-of-the-shelf techniques to compress deep learning architectures such as quantization.

References

[1] Arash Ardakani, Carlo Condo, and Warren J Gross. Sparsely-connected neural networks: towards efficient vlsi implementation of deep neural networks. *arXiv preprint arXiv:1611.01427*, 2016.

[2] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[3] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[4] Ghouthi Boukli Hacene, Vincent Gripon, Matthieu Arzel, Nicolas Farrugia, and Yoshua Bengio. Quantized guided pruning for efficient hardware implementations of convolutional neural networks. *arXiv preprint arXiv:1812.11337*, 2018.

[5] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[6] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[7] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

[8] 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. *arXiv preprint arXiv:1704.04861*, 2017.

[9] Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann. Learning to prune filters in convolutional neural networks. *arXiv preprint arXiv:1801.07365*, 2018.

[10] 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. *arXiv preprint arXiv:1602.07360*, 2016.

[11] Yunho Jeon and Junmo Kim. Constructing fast network through deconstruction of convolution. *arXiv preprint arXiv:1806.07370*, 2018.

[12] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4284–4293. IEEE, 2017.

[13] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[14] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[15] Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. *arXiv preprint arXiv:1510.03009*, 2015.

[16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

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

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Guillaume Soulié, Vincent Gripon, and Maëlys Robert. Compression of deep neural networks on the fly. In *International Conference on Artificial Neural Networks*, pages 153–160. Springer, 2016.

[20] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015.

[22] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[24] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. *arXiv preprint arXiv:1711.08141*, 2017.

[25] Junru Wu, Yue Wang, Zhenyu Wu, Zhangyang Wang, Ashok Veeraraghavan, and Yingyan Lin. Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. *arXiv preprint arXiv:1806.09228*, 2018.

[26] Kohei Yamamoto and Kurato Maeno. Pcas: Pruning channels with attention statistics. *arXiv preprint arXiv:1806.05382*, 2018.

[27] Yifan Yang, Qijing Huang, Bichen Wu, Tianjun Zhang, Liang Ma, Giulio Gambardella, Michaela Blott, Luciano Lavagno, Kees Vissers, John Wawrzynek, et al. Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded fpgas. *arXiv preprint arXiv:1811.08634*, 2018.

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

[29] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018.