Regularized Training Framework for Combining Pruning and Quantization to Compress Neural Networks

1st Qimin Ding
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
moquan@mail.nwpu.edu.cn

2nd Ruonan Zhang
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
rzhang@nwpu.edu.cn

3rd Yi Jiang
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
peipeiv88nwpu@hotmail.com

4th Daosen Zhai
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
zhaidaosen@163.com

5th Bin Li
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
libin@nwpu.edu.cn

Abstract—Many convolutional neural networks(CNNs) have been proposed to solve computer vision tasks such as image classification and image segmentation. However the CNNs usually contain a large number of parameters to determine which consumes very high computation and power resources. Thus, it is difficult to deploy the CNNs on resource-limited devices. Network pruning and network quantization are two main methods to compress the CNNs, researchers often apply these methods individually without considering the relationship between them. In this paper, we explore the coupling relationship between network pruning and quantization, as well as the limits of the current network compression training method. Then we propose a new regularized training method that can combine pruning and quantization within a simple training framework. Experiments show that by using the proposed training framework, the finetune process is not needed anymore and hence we can reduce much time for training a network. The simulation results also show that the performance of the network can over-perform the traditional methods. The proposed framework is suitable for the CNNs deployed in portable devices with limited computational resources and power supply.

Index Terms—convolutional neural networks, network compression, training framework, coupling relationship, fuzzy rules.

I. Introduction

With the rapid development in recent years, deep convolutional neural networks (CNNs) keep promoting the algorithm performance on various computer visual data sets. But the improvement of network performance is often based on the increasing scale of network models. The CNNs consume much computational resources power and time at runtime, so that these powerful models can only be deployed on GPU clusters. In recent years, in order to deploy the CNNs on small portable devices such as FPGAs, more and more researchers are beginning to study how to compress CNNs. The typical two methods are network pruning and network quantification.

Nowadays, the network pruning [1]–[3] and quantification [4], [5] are two independent research fields and have their own model training methods. The training methods of network pruning are usually divided into two types. The first one is to gradually prune the redundant network parameters or network structure in the network training process, The second method is to note down the parameters and structures needed to be pruned and they are cut off after the training process. The process of network quantification is often divided into two major steps [6]. The first step is to train a float-point model. The second step is to finetune the trained float-point model and quantize the network during the finetuning process.

In practice, we need to conduct network pruning and network quantization on a CNN in multiple steps: training a full-precision floating-point model first, and then using the network pruning method to prune the full-precision float-point model. After the network has been pruned, quantization process will quantize the network towards target quantization bits. Using this training method to train a CNN can indeed make use of network pruning and quantification together. But such a training framework has the following drawbacks: the training process does not combine the two model compression methods but simply performs the two methods repeatedly. Such a training framework does not consider the coupling relationship between the model compression methods.

In this paper, we first investigate the coupling rela-

This work was supported in part by the National Natural Science Foundation of China under Grant 61901381, in part by the China Postdoctoral Science Foundation under Grant BX20180262 and Grant 2018M641019, in part by the Natural Science Basic Research Plan in Shaanxi Province under Grant 2019JQ-631.

tionship between the network pruning and quantization. Motivated by this relationship, we propose a new regularized training framework to combine network pruning and quantization together. In this framework, we use some rules to regularize the training process and make the training framework to choose the best combination of the two network compression methods. In order to combine the two methods, we plug-in pruning and quantization so that the training framework can choose the specific compression method during each epoch of the training peroid.

Experiments show that without our training framework, network compressed by pruning and quantization suffers a lot from performance drop even though the performance after pruning and quantization is satisfactory individually. By using our proposed training framework, network can choose different compression methods in each training epoch dynamically and the performance is much higher than that of the traditional method.

The rest of the paper is organized as follows: Sec.II describes network pruning as well as quantization individually. The proposed training framework as well as its three key components are presented in Sec.III. The simulation and measurement results and algorithm comparisons are presented in Sec.IV. Finally, Sec.V concludes the paper and points out future research issues.

II. RELATED WORK

A. Network Pruning

Network pruning attempts to prune the parameters or structures in a network that have less impact on the network performance to reduce the amount of network computation. In [1], the greedy method is used to calculate the Taylor weight of the convolution kernel parameters in the network finetune process to select the convolution kernel to be pruned; Reference [3] introduces the Dropout idea into the convolution kernel, and the Drop convolution kernel is masked. The authors in [9] introduce pruning terms into the loss function of the network, and regard pruning as a part of the network iterative optimization function. Reference [10] considers the regularization term on the convolution kernel sparse coefficient to sparse the convolution kernel to achieve pruning. The concept of geometric median is used in [11] to measure the convolution kernel similarity to determine whether the convolution kernel needs to be pruned. Structured pruning has been applied in [12] and has achieved good pruning results.

Channel-level pruning methods prune the convolutional channel which make the CNNs slimmer, by pruning one convolutional layer, the layers related to current layer also need to be pruned. Reference [10] proposes a channel-level pruning method which uses the next layer's feature map to guide the pruning in current layer. Other methods such as [11] proposes an iterative two-step algorithm to prune channels by minimizing the feature map errors. Due to the efficiency of channel-level pruning, this paper selects this kind of pruning method as a plug-in component in the framework.

B. Network Quantization

Network quantization attempts to replace the commonly used 32-bit network parameters with low-bit parameters, thereby reducing the amount of bits used to store the parameters in the model and facilitating the acceleration of FPGA. Reference [12] first proposes using 16-bit parameters for network training, and verified the feasibility of compression network parameter bits. In [13], parameters in the network are logarithmized, and the numerical interval of the compression parameters is quantified. Reference [14] uses histogram equalization to approximate the network parameters of the original Gaussian distribution to increase the amount of information per bit of data. Reference [4] uses the K-means algorithm to cluster convolution kernel parameters into several groups and then uses a limited number of values to represent each group. The method of approximating the original floating-point matrix using two matrices with only integer values has been proposed in [15].

III. METHODOLOGY

Regularized training framework has three key components: plug-in pruning and quantization, training framework, and rules to choose compression methods.

A. Plug-in Pruning and Quantization

In most of the proposed network compression frameworks, one compression method is applied in the entire training process without any change. For example, if the network is compressed by quantization, then no other compression method such as pruning is applied during the whole training period. The reason is that the ordinary pruning and quantization method requires continuity to ensure the result of network compression. This requirement restricts the usage of different compression methods together.

In order to handle this problem, we introduce plug-in pruning and quantization. Different from the ordinary method in which pruning and quantization are designed to adapt to specific convolutional layers or ReLU [24] layers in the network, plug-in pruning and quantization only operate with inputs and outputs of layers in the network without considering where they are generated. This can decouple the pruning and quantization process from specific network layers.

For convolutional layers, plug-in quantization only operates with a 4-D kernel and the outputs generated from current layer by using Fix Quantization or other methods. With a quantized 4-D kernel, if layer's input parameters are quantized, then the output of the layer is also quantized. Nevertheless, when input parameters are float-point numbers (such as images in the datasets), we need to quantize the output as well so that the input of the next layer will be quantized. Because ReLU layers cannot modify a quantized parameter into a float-point parameter, if the output from convolutional layers are quantized, the parameters in all the network will be quantized values.

Plug-in pruning is generally more difficult because it should have the ability to analyze the chain relationship between difference convolutional layers. Fig.1 shows a simple chain relationship between different convolutional layers.

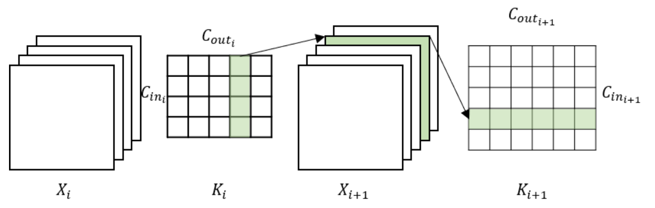

Fig. 1. A simple chain relationship in CNNs. The number of input channels in kernel K_{i+1} should equal the number of output channels in kernel K, hence when the kernel K_i is pruned, kernel K_{i+1} should also be pruned.

In Fig.1, let X_i be the input feature map of the convolutional layer i , K_i is the convolution kernel of this layer, and each small cell in K represents the convolution kernel pattern of size (H_{out} ,W_{out}). When some input channels in kernel K_i are pruned by a plug-in pruning method, some of the output channels in the output feature map X_{i+1} will be deprecated. But we should notice that the output of the convolution layer i is also the input of the convolution layer $i+1$ (ReLU and BatchNorm [16] layer are ignored because these two kinds of layers do not change the network structure.), In order to match the input and output channels between different layers, the number of input channels $C_{in_{i+1}}$ of the convolution kernel K_{i+1} also needs to be pruned correspondingly.

B. Training Framework

The training framework proposed in this paper can be represented in Fig.2. The "automatically search for compression methods" module in Fig.2 is based on the plug-in network pruning and quantization.

Suppose that the L is the number of layers in a CNN, E is the total number of epochs. K_i is the convolution kernel in the *ith* layer, R_i is the ReLU function applied on the output of the *ith* layer. Then we describe the function of the *ith* layer as (1), hence the entire network can be represented as (2). We use P and $Q_{m,n}$ to represent plug-in pruning and quantization where m is the quantization bit for convolution layers and n is the quantization bit for ReLU layers, then we use a set $\mathcal{C} = \{P, Q_{m,n} | m, n \subset \mathbb{N}, m, n \leq 32\}$ to donate all the optional compression methods during training. The purpose of the framework is to select a hash-map $\mathcal{Z} = \{e \rightarrow c | e \leq E, c \subset \mathcal{C}\}$ to compress K_i and R_i in the entire network during each epoch. In order to store the information of a specific epoch during training, we group all the trainable parameters into a set $\mathcal{S}$ together with the status of the network optimizer such as the learning rate and the value of current weight decay.

$$f^i = R_i(K_i \bigotimes f^{i-1}) \tag{1}$$

$$\mathcal{F} = f^L \bullet f^{L-1} \bullet f^{L-2} \bullet\bullet\bullet f^1 \tag{2}$$

Algorithm 1 Proposed Training Framework

```
Input: The set of compression methods for current training epoch, $\mathcal{C}$; The rule to guide the framework, $\mathcal{R}$;
Output: Compression method used in each epoch $\mathcal{Z}$;
1: while $e \leq E$ do
2:   Initialize a hash-map $\mathcal{M} = \{c \rightarrow v\}$, where $c$ is a compression method in $\mathcal{C}$ and $v$ is the score of this compression method.
3:   Collect $\mathcal{S}$ from the CNN;
4:   for compression method $c$ in $\mathcal{C}$ do
5:     Initialize the CNN and the optimizer with $\mathcal{S}$;
6:     Train and compress the CNN with $c$ by using plug-in pruning and quantization;
7:     Calculate the score $v$ of the current compression method $c$ by $\mathcal{R}$;
8:     Insert a map $c \rightarrow v$ into $\mathcal{M}$
9:   end for
10:  Select the compression method $c$ with the highest score in $\mathcal{M}$
11:  Insert a map $e \rightarrow c$ into $\mathcal{Z}$
12: end while
13: return $\mathcal{Z}$
```

By using this training framework, plug-in pruning and quantization can be selected by the framework during the training process with the guidance of the rule. When the training process is completed, networks are compressed not only by pruning but also quantization. Experiments will show that training framework can search for different compression methods and combine them together. In this way the coupling relationship between pruning and quantization are considered by using this approach.

C. Rules for Selecting a Suitable Compression Method

When training networks by using the proposed training framework, it is very important to guide the framework to choose the right compression method and apply it in current training epoch.

In this section, we give three rules that can guide the framework to choose the right compression method in the searching space.

- The accuracy of the network. After the network has been trained by the current epoch, training framework will use the test set in the dataset to validate the performance of the network. The metric used to

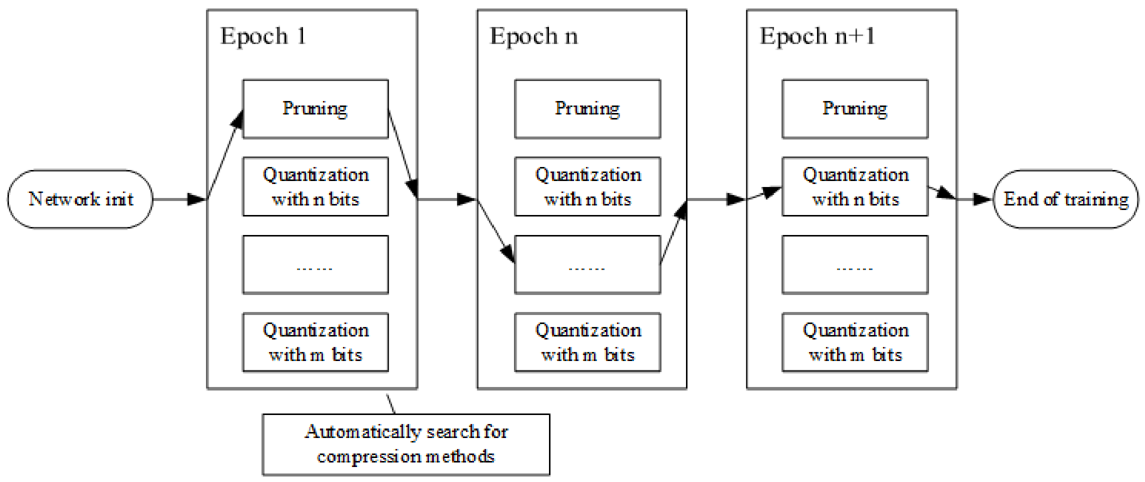

Fig. 2. Our training framework. During each training epoch, this framework will search the most suitable compression method to train the network. Many rules can be applied to control the this searching process, such as latency, accuracy and so on.

measure the performance of the network on the test set can be the top-1 accuracy.

- Latency of the network when it is implemented on FPGA. One of the purposes of network compression is to accelerate the network forwarding period. Therefore, we can count the time consumed by the model's forward propagation, and it is reasonable to choose the compression method according to the latency.
- Number of float-point operation used to forward a network. The fewer floating-point operations a network needs in forward propagation, the shorter time the network will consume, and the more efficient the network is.

IV. EXPERIMENTS

We conducted extensive experiments to validate the proposed training framework. First, we compare the plug-in pruning and plug-in quantization with traditional pruning and quantization. Second, we use top-1 accuracy as the rule to guide the framework to select the compression methods in the searching space.

A. Plug-in and No-plug-in Pruning and Quantization

Before we can use no-plug-in pruning and quantization as components in the proposed framework, we should firstly prove it is reasonable to replace ordinary pruning and quantization with them.

We choose ResNet18 [17] as the target network when we test two pruning methods. We use the dataset CIFAR-100 [18] that is similar with CIFAR-10 and it has 100 classes, every class contains 600 including 500 training images and 100 testing images. All the 100 classes in the CIFAR-100 are grouped into 20 super classes. Each image comes with a "fine" label (the class it belongs to) and a "coarse" label (the superclass it belongs to). The goal of the quantized network is to classify all the images in the test set into the correct classes.

Details of the pruning process are as follows. The optimizer used to train the network is Stochastic Gradient Descent(SGD) [19], in which the momentum is 0.9 and the weight decay [20] is 0.0001. By using 8 Nvidia Titan XP GPUs, we set the total batch size to be 256. No tricks or whistles are used except a multi-step learning rate. The initial learning rate is 0.1 and when the training epoch becomes 40, 60, and 90, the learning rate will change to one tenth of itself. The total number of epochs for pruning is 110. In each training epoch, the pruning method will prune specific numbers of channels from all the convolutional layers permanently according to the pruning rate in the range of 0 to 0.3. We use the pruning method presented in [21] in the experiments. We list the top-1 accuracy of the pruned network in Table.I.

TABLE I
TOP-1 ACCURACY OF PLUG-IN AND NO-PLUG-IN PRUNING METHODS

MODE	Pruning Rate			
	0	0.1	0.2	0.3
plug-in	91.37	91.25	91.04	90.65
no-plug-in	92.20	92.24	91.20	90.83

Table.I shows that the plug-in pruning method has slightly difference compared with the ordinary method because the accuracy drop for these two methods are almost the same.

Then we choose ResNet50 as the target network when we test the plug-in and no-plug-in quantization method, the two methods are both designed based on [22]. The dataset used in this experiment is ImageNet [23] which is an image database organized according to the WordNet hierarchy (currently only the nouns). In ImageNet, each node of the hierarchy is depicted by hundreds and thousands of images. The whole dataset contains 1.2 million training images and 50000 validation images of

1000 classes. The purpose of the network is to classify all the test images into its correct class. The training process of the network is the same as pruning, but no pruning rate will affect the training process. We use different combinations of quantization bits to test the plug-in and no-plug-in method.The results are listed in Table.II. We can see that the plug-in and no-plug-in quantization methods are similar with each other.

TABLE II
TOP-1 ACCURACY OF PLUG-IN AND NO-PLUG-IN QUANTIZATION METHODS

MODE	Quantization Bits[a]			
	[32,32]	[16,16]	[8,8]	[4,8]
plug-in	75.48	75.21	74.31	72.75
no-plug-in	75.86	75.47	74.28	73.05

[a]The first number is the convolutional layer's quantization bit and the second number is the quantization bit in ReLU layers.

B. Training Framework with Rule of Accuracy

Since we have validated the effectiveness of the plug-in pruning and quantization, we conduct experiments to validate our proposed training framework in this subsection. We still use ResNet50 as the target network and apply both pruning and quantization to this network. The dataset used in this subsection is ImageNet because CIFAR-100 is too simple to distinguish the ordinary training framework and our training framework.

For the ordinary training framework, we need to take two steps to prune and quantize the network. In the pruning period, we use SGD as the optimizer and the number of total training epochs is 100. In each epoch, the pruner will prune 40 channels from the entire network permanently until 30% of the network has been pruned. The initial learning rate is 0.2 and, when the training epoch comes to 40, 60, and 80, the learning rate will decrease to one tenth of its current value. After the network has been pruned, we take the second step to quantize the network and the optimizer used in this step is also SGD with the momentum of 0.9 and the weight decay of 0.0001. We choose 3 combinations of quantization bits to train the quantized network. The number of epochs used in the second step is 40.

By using plug-in pruning and quantization, we can set our training framework very simple, we only take one step and train the network for 100 epochs. The optimizer is also SGD with the momentum of 0.9 and the weight decay of 0.0001. The rule used in this subsection is top-1 accuracy that the network achieved on the ImageNet dataset. If one of the epochs is chosen to conduct pruning the network, the framework will prune 50 channels from the entire network until 30% of the network has been pruned. If one of the epochs is chosen to conduct quantization, the framework will use the plug-in quantization module to quantize the network to target quantization bits.

TABLE III
TOP-1 ACCURACY OF DIFFERENT TRAINING METHODS

MODE	Quantization Bits[b]			
	[32,32]	[16,16]	[8,8]	[4,8]
ordinary	74.35	73.12	72.84	71.02
proposed	74.29	73.85	73.29	72.86

[b]Pruning rate is 30% for the both training frameworks

Due to the hard compression caused by both quantization and pruning, both the training frameworks suffer from performance drop. But the proposed training framework still achieve better results than the ordinary training framework, as listed in Table.III.

We draw the choices that both frameworks made under [8,8] bits quantization and compare the different behaviors of two training frameworks in Fig.3.

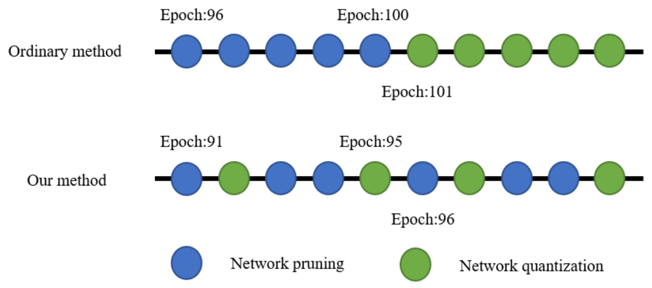

Fig. 3. Choices made by different training frameworks.

The ordinary framework cannot choose a specific compression method in each epoch and ignores the coupling relationship between different compression methods. Even though the pruning and quantization methods are the same in the both training frameworks, the proposed training framework can choose suitable compression method in each training epoch and consequently the performance of the trained network over performs that of the ordinary framework.

V. Conclusion

Combining network pruning and quantization is an important part in compressing neural networks, which is challenging due to the separation of the network pruning and quantization. In this paper, we propose a training framework by using plug-in pruning and quantization as well as some selection rules. We first describe plug-in pruning and quantization and prove the efficiency of them. Then we propose some rules to guide the training framework to select suitable compression method in each training epoch. We perform extensive simulations to evaluate and compare the performance of the proposed and existing training framework by using ResNet18 and ResNet50 in the CIFAR-100 and the ImageNet datasets. The results of the experiments show the proposed framework can keep more top-1 accuracy of the CNNs when compressing the networks very hard. More methods to combine network pruning and quantization as well as some new fusion rules to guide the framework during training are interesting research topics for future works.

References

[1] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. International Conference on Learning Representations (ICLR), 2017.

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

[3] M. A. Carreira-Perpi ˜n´an and Y. Idelbayev. "Learning-compression" algorithms for neural net pruning. In Proc. of the 2018 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'18), pages 8532–8541, Salt Lake City, UT, June 18–22 2018.

[4] Y. Choi, M. El-Khamy, and J. Lee. Towards the limit of network quantization. In Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017.

[5] D. Miyashita, E. H. Lee, and B. Murmann. Convolutional neural networks using logarithmic data representation. arXiv preprint arXiv:1603.01025, 2016

[6] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems, pages 1135–1143, 2015

[7] S. Hou and Z. Wang. Weighted channel dropout for regularization of deep convolutional neural network. In AAAI, 2019.

[8] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In ICCV, 2017.

[9] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[10] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In The IEEE International Conference on Computer Vision (ICCV), Oct 2017.

[20] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," Advances Neural Inform. Processing Syst., vol. 4, pp. 951–957, 1992

[11] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in The IEEE International Conference on Computer Vision (ICCV), Oct 2017.

[12] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep Learning with Limited Numerical Precision," CoRR, vol. abs/1502.02551, 2015.

[13] D. Miyashita, E. H. Lee, and B. Murmann. Convolutional neural networks using logarithmic data representation. arXiv preprint arXiv:1603.01025, 2016.

[14] S.-C. Zhou, Y.-Z. Wang, H. Wen, Q.-Y. He, and Y.-H. Zou. Balanced quantization: An effective and efficient approach to quantized neural networks. Journal of Computer Science and Technology, 32(4):667–682, 2017

[15] P. Wang and J. Cheng. Fixed-point factorized networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[18] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 and cifar-100 datasets. URl: https://www. cs. toronto. edu/˜ kriz/cifar.html, 2009

[19] L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In Y. Lechevallier and G. Saporta, editors, Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010), pages 177–187, Paris, France, August 2010. Springer.

[21] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. In IJCAI, 2018.

[22] P. Gysel. Ristretto: Hardware-oriented approximation of convolutional neural networks. Master's thesis, University of California, 2016.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. ImageNet: A large-scale hierarchical image database. In CVPR, 2009.

[24] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In AISTATS, 2011.