Sparseness Ratio Allocation and Neuron Re-pruning for Neural Networks Compression

Li Guo[1], Dajiang Zhou[1], Jinjia Zhou[2,3], and Shinji Kimura[1]
[1] Graduate School of Information, Production, and Systems, Waseda University, Kitakyushu, Japan.
[2] Graduate School of Science, and Engineering, Hosei University, Tokyo, Japan.
[3] JST, PRESTO, Tokyo, Japan
Email: guoli@toki.waseda.jp

***Abstract*—Convolutional neural networks (CNNs) are rapidly gaining popularity in artificial intelligence applications and employed in mobile devices. However, this is challenging because of the high computational complexity of CNNs and the limited hardware resource in mobile devices. To address this issue, compressing the CNN model is an efficient solution. This work presents a new framework of model compression, with the sparseness ratio allocation (SRA) and the neuron re-pruning (NRP). To achieve a higher overall spareness ratio, SRA is exploited to determine pruned weight percentage for each layer. NRP is performed after the usual weight pruning to further reduce the relative redundant neurons in the meanwhile of guaranteeing the accuracy. From experimental results, with a slight accuracy drop of 0.1%, the proposed framework achieves 149.3× compression on lenet-5. The storage size can be reduced by about 50% relative to previous works. 8-45.2% computational energy and 11.5-48.2% memory traffic energy are saved.**

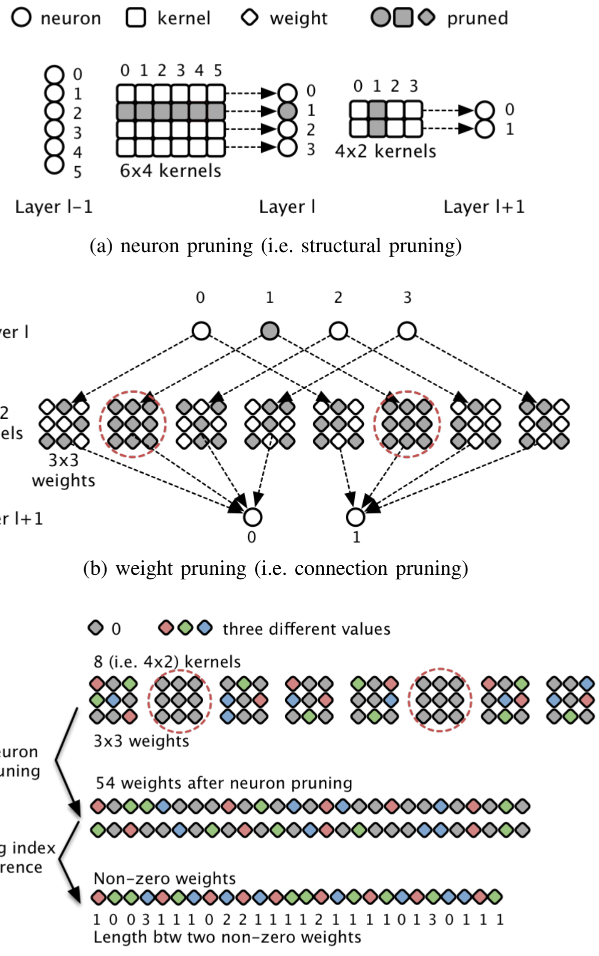
***Index Terms*—Model compression, connection/neuron pruning, sparseness ratio allocation, neuron re-pruning.**

I. Introduction

Convolutional neural networks (CNNs) achieve the state-of-the-art performance in various modern artificial intelligence (AI) applications [1], and have been gaining popularity in mobile devices like smart-phones. However, CNNs usually come with high computational complexity, while the resource in terms of energy and storage for mobile devices are limited. Hence, this becomes the crucial challenge.

Model compression is an efficient solution to reduce the storage size of CNN model and the related energy consumption of memory traffic from DRAM for buffering model [2]. It is usually composed of pruning, quantization and encoding. The pruning can be further divided to two categories of weight [3] [4] [5] and neuron (i.e. structural) pruning [6] [7]. When structural pruning is exploited, the computational energy can also be reduced.

Compared to the other portions of quantization and encoding, pruning contributes the most for reducing the model size, but its overall efficiency highly depends on the sparseness ratio in each layer. Considering only one layer, a larger sparseness ratio clearly achieves a higher compression ratio, and it can be chosen by the acceptable accuracy drop after retraining. However, for a deep CNN model, it is impossible for retraining and testing all cases and choosing an efficient one. Hence, except for improving the sparseness ratio, how to allocate the pruned weight percentage to each layer should also be studied.



(a) neuron pruning (i.e. structural pruning)
(b) weight pruning (i.e. connection pruning)
(c) quantization (i.e. weight sharing)

Fig. 1. Existing techniques for neural networks model compression.

This paper presents a new model compression framework with sparseness ratio allocation and neuron re-pruning. The contributions are described as follows:

1) Sparseness ratio allocation to avoid repeated pruned percentage settings: determines the pruned percentage for each layer without retraining process.

2) Neuron re-pruning to improve compression ratio: removes the relative redundant neurons for a pre-weight-pruned model. The model size can be further reduced by up to 28.3%

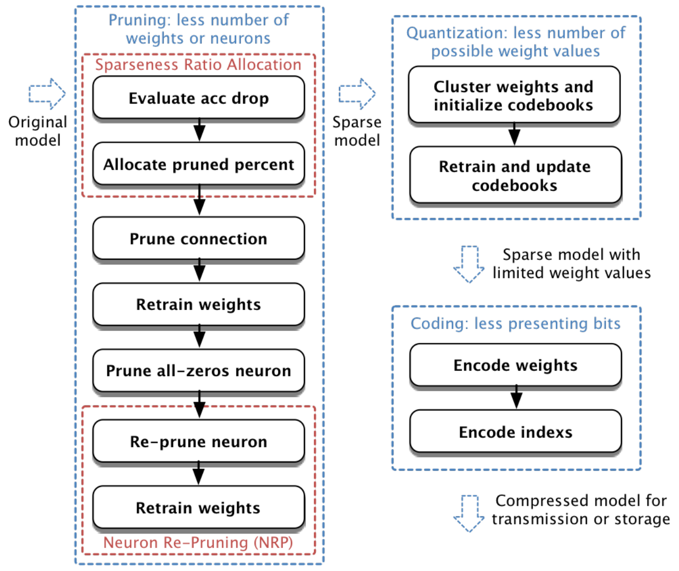

Original model
Pruning: less number of weights or neurons
Sparseness Ratio Allocation
Evaluate acc drop
Allocate pruned percent
Prune connection
Retrain weights
Prune all-zeros neuron
Re-prune neuron
Retrain weights
Neuron Re-Pruning (NRP)
Sparse model
Quantization: less number of possible weight values
Cluster weights and initialize codebooks
Retrain and update codebooks
Sparse model with limited weight values
Coding: less presenting bits
Encode weights
Encode indexes
Compressed model for transmission or storage

Fig. 2. New framework for neural networks model compression.

with nearly no accuracy drop.

3) Higher hardware resource efficiency: with the proposed work, the storage size for lenet-5-caffe is reduced by 50%, relative to [4], and 8-45.2% and 11.5-48.2% energy is saved respectively from computation and memory traffic.

II. Model Compression Overview

The three basic techniques for compressing CNN model will be briefly introduced and shown in Fig. 1.

1) Pruning: Pruning on neuron or weight is usually first performed to remove negligible weights, and followed by a retraining process to guarantee accuracy, as shown in Fig. 1a and 1b respectively. For the former, it mainly aims to reduce the computational complexity, while the latter is focusing on reducing the model size. For the neuron pruning (see Fig. 1a), if one neuron in layer l is pruned, the corresponding weights in the two neighboring layers are pruned. The structure of CNN is modified after pruning, so the computation related to the pruned neuron can be obviously omitted. The weight pruning makes the model sparse as given in Fig. 1b. Due to the irregular sparseness, it is difficult for reducing the amount of computation. However, under the same accuracy drop, more weights can be pruned compared to the neuron pruning, which is beneficial for reducing the CNN model size.

According to the study of previous works, an efficient and most widely used way for choosing the pruned unit (i.e. weight or neuron) is to prune the one with smaller absolute value or sum value first. We also follow it in this work.

2) Quantization: Weights in each layer are presented with a constant number ofF shared codes, so code indexes are encoded instead of the original weight values.

3) Encoding: According to the probability distribution of the code indexes from quantization, weights are encoded by Huffman coding.

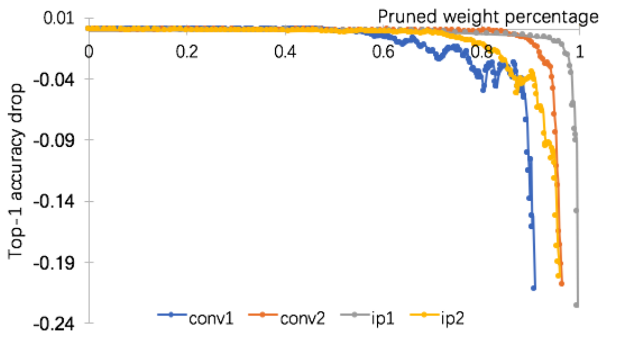


Fig. 3. Evaluated curves between the pruned weight percentage (p_p, without %) and the accuracy drop for Lenet-5-caffe. Weight pruning is exploited.

III. Proposed Sparseness Ratio Allocation and Neuron Re-Pruning Scheme

A. New Model Compression Framework

Figure 2 shows the proposed framework for model compression. Except for the basic processes presented in the above section II, two additional portions of sparseness ratio allocation (SRA) and neuron re-pruning (NRP) are exploited to achieve a better tradeoff between compression ratio and accuracy. In this framework, we focus on optimizing the pruning stage, so follow the quantization and encoding processes in deep compression [4], which are introduced in Section II.A.

For the pruning, SRA is added to explore the potential of compression by allocating different sparseness ratio to each layer, while NRP is to further reduce the redundant overhead bits for presenting pruned weights by re-pruning the neurons with high sparseness ratio. Hence, in the proposed framework, there are three steps for pruning. At first, for each layer l, a pruned percentage (defined as $p_{p,l}$) is determined by SRA. Weight pruning is then performed base on the p_p setting. p_p of the weights with smaller absolute values are pruned and retrained. At last, NRP is performed to further reduce the redundant neurons.

B. Sparseness Ratio Allocation based on Weight Pruning

Based on the weight pruning, the proposed SRA consists of three steps. At first, for each layer, the curve between the pruned weight percentage (i.e. p_p) and the corresponding test accuracy drop is evaluated. Then the layer-wise acceptable accuracy drop (defined as acc_d) is determined according to a given drop degree. Finally, the sparseness ratio (i.e. p_p) for each layer is obtained from the above curve and acc_d.

To estimate the direct influence of pruning one layer to accuracy, the original pre-trained model is pruned and tested. To reduce the complexity, there is no retraining process after pruning, and only one layer is pruned for each time. The curves between p_p and acc_d for lenet-5-caffe are shown in Fig. 3.

Then, for layer l, the accuracy drop ($acc_{d,l}$) is determined as presented in Eq. 1. In order to improve compression ratio in the meanwhile of guarantee accuracy, three factors are considered for sparseness ratio allocation: for each layer, a higher spareness ratio leads to a more serious accuracy drop;

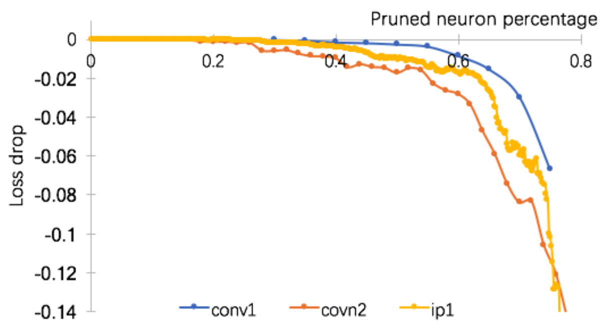


Fig. 4. The evaluated curves between the pruned neuron percentage (without %) and the loss drop for Lenet-5-caffe. Neuron pruning is performed on a pre-weight-pruned CNN model.

for a layer with more weights, a more aggressive pruning on it may achieve a higher overall compression ratio; for a layer closer to the output, the influence of pruning will be propagated to the accuracy more directly.

$$acc_{d,l} = max[p_{w,l} \times (1 + l), 1.0] \times d_d \tag{1}$$

where $p_{w,l}$ is the percentage of the weights in layer l compared to in the whole model. d_d is short for drop degree, which is a defined parameter to control the tradeoff between higher compression ratio and lower accuracy drop. For a certain CNN model, with a larger d_d, a smaller model size can be obtained.

Finally, the pruned weight percentage ($p_{p,l}$) for each layer l is determined according to the curve (e.g. in Fig. 3) and $acc_{d,l}$.

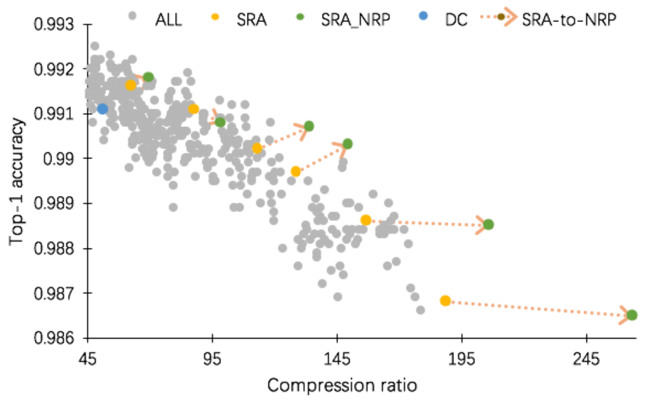
C. Neuron Re-pruning

Considering one layer l, although $p_{p,l}$ is exploited to guide weight pruning, the sparseness ratios for each kernel or neuron are obviously not equal. For these neurons with a high sparseness ratio, most bits are used to present the pruned weight, while not the unpruned weights. With few unpruned weights, these neuron can be pruned with a slight influence to accuracy.

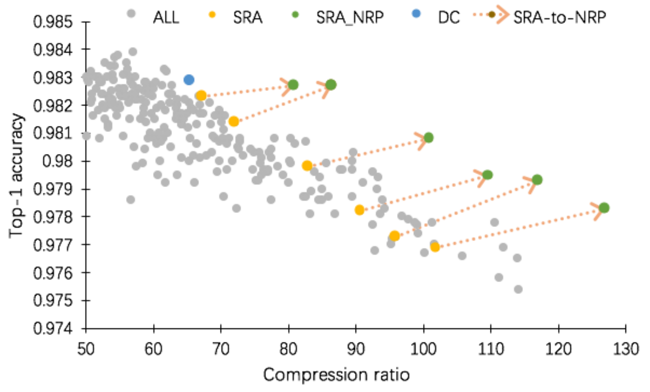
In previous work of deep compression [4], after weight pruning, these neurons, with all pruned previous (as output neuron) or next (as input neuron) weights, are pruned. Although this neuron pruning leads to no accuracy drop, but the percentage of these kind of neurons is very low or even zero.

Therefore, a neuron re-pruning process is added after the usual weight pruning in this proposed framework Fig. 2. Pruning is only performed on output neurons of each layer, except for the last output layer. The sum of related previous layer weights is calculated for each neuron. The neuron is pruned in the order of the sum value from low to high.

Following the basic process of weight pruning, the curve between the percentage of pruned neuron and the loss drop is evaluated based on a pre-pruned model. Without retraining process, an example for lenet-5-caffe is presented in Fig. 4. Compared to the weight pruning, the loss drop is exploited instead of the accuracy, since the loss is more sensitive than the accuracy. In this NRP process, guaranteeing the accuracy is more significant than the compression ratio, so the more sensitive measurement of loss is used.



(a) Lenet-5-Caffe (original size 1.72 MB, accuracy 0.9914)



(b) Lenet-300-100 (original size 1.07 MB, accuracy 0.9828)

Fig. 5. Experimental results of compression ratio and top-1 accuracy on MNIST. These gray nodes are for various p_p setting within a reasonable range. For lenet-5-caffe, the tested p_p ranges for four layers are 20-80%, 60-90%, 90-99%, and 50-80%, while they are 88-97%, 75-90%, and 50-90% for lenet-300-100. DC follows the p_p setting in [3] [4]. Yellow nodes are for applying SRA only, while these green nodes are for SRA and NRP.

For each layer, the acceptable loss drop before retraining is set to a constant value, which is 0.01$\times$ the loss of pre-pruned model. So the pruned neuron number for each layer can be determined based on the above evaluated curve. According to the experiments in Section IV, up to 34% more neurons in one layer can be pruned without accuracy drop.

IV. Experimental Results

To evaluate the compression performance and the corresponding accuracy, the basis processes for model compression are integrated to Caffe [8]. A tradeoff between the model size and the accuracy drop is presented. Furthermore, the influence of the proposed work to CNN accelerators is also discussed and evaluated from two aspects of storage and energy consumption.

A. Experimental Condition

Except for the proposed two portions of SRA and NRP, the other processes of weight pruning, quantization and encoding follow the algorithm in deep compression (DC) [4]. Two models on MNIST are evaluated, the lenet-5 from caffe and the lenet-300-100. For the retraining process of pruning and

TABLE I
REMAINING WEIGHT PERCENTAGE FOR EACH LAYER WITH THE PROPOSED SPARSENESS RATIO ALLOCATION (SRA)

lenet-5	weight	remaining weight percentage (%)					
	#	0.01^*	0.02^*	0.03^*	0.05^*	0.07^*	0.09^*
conv1	0.5k	72.0	72.0	72.0	65.8	65.8	52.4
conv2	25k	15.9	15.0	12.7	10.9	9.6	8.6
ip1	400k	5.0	2.9	2.0	1.7	1.2	0.8
ip2	5k	60.4	53.5	49.4	29.3	27.4	27.4
lenet	weight	remaining weight percentage (%)					
300-100	#	0.015^*	0.02^*	0.03^*	0.04^*	0.05^*	0.06^*
ip1	235k	7.2	6.2	4.9	4.3	3.9	3.6
ip2	30k	9.9	9.9	7.7	6.7	6.0	5.7
ip3	1k	59.8	57.1	56.8	52.8	45.1	45.1

Note: * *the drop degree (d_d) in Eq. 1.*

TABLE II
REMAINING OUTPUT NEURON NUMBERS FOR EACH LAYER WITH THE PROPOSED NEURON RE-PRUNING (NRP)

lenet-5	output	remaining neuron # (neuron # reduced by NRP)					
	neu. #	0.01^*	0.02^*	0.03^*	0.05^*	0.07^*	0.09^*
conv1	20	17 (3)	17 (3)	17 (3)	16 (4)	16 (4)	16 (4)
conv2	50	48 (2)	46 (4)	43 (7)	44 (6)	43 (7)	42 (8)
ip1	500	427 (73)	416 (74)	377 (92)	385 (69)	293 (133)	269 (117)
lenet	output	remaining neuron # (neuron # reduced by NRP)					
300-100	neu. #	0.015^*	0.02^*	0.03^*	0.04^*	0.05^*	0.06^*
ip1	300	203 (76)	202 (74)	188 (83)	184 (84)	185 (83)	175 (90)
ip2	100	69 (12)	69 (12)	69 (12)	70 (9)	68 (11)	68 (11)

Note: * *the drop degree (d_d) in Eq. 1. (#) indicates the number of neuron further reduced by applying NRP.*

quantization, the base_lr for lenet-5-caffe is set to 0.005, while it is 0.01 for lenet-300-100. The number of quantization steps is 9. One for the zero value, and the other 8 for non-zero codes. These codes are encoded based on Huffman coding. 4 bits are used to present the length between two unpruned weights. If the length is longer than 32, zeros will be padded.

B. Lenet-5-caffe and Lenet-300-100 on MNIST

Fig. 5 shows the experimental results for lenet-5-caffe and lenet-300-100. Except for the DC [4] and the proposed SRA/NRP, various p_p settings within a reasonable range are also evaluated. These gray nodes in Fig. 5 indicates the achievable accuracy under a compression ratio.

From the comparison between these gray nodes and the yellow nodes for applying SRA, the SRA is efficient for providing the p_p for each layer without retraining many times. The difference among these nodes of SRA are the drop degree (d_d) setting. The detailed p_ps for each layer are given in TABLE I. NRP further improves the compression ratio. TABLE II presents the pruned neuron numbers. Nodes with NRP in Fig. 5 are superior to all gray nodes, so the proposed framework with SRA and NRP is more efficient than previous.

TABLE III
REDUCED PERCENTAGE OF COMPUTATIONAL ENERGY CONSUMPTION (%) FOR LENET-5-CAFFE

	layer	0.01^*	0.02^*	0.03^*	0.05^*	0.07^*	0.09^*
SRA	conv1	0.0	0.0	0.0	0.0	0.0	0.0
	conv2	8.0	10.1	15.5	21.8	27.0	30.8
SRA+	conv1	15.0	15.0	15.0	20.0	20.0	20.0
NRP	conv2	21.7	25.6	32.4	38.1	42.2	45.2

Note: * *the drop degree (d_d) in Eq. 1. The percentage is calculated relative to the original CNN model.*

TABLE IV
REDUCED PERCENTAGE OF ENERGY CONSUMPTION FOR MEMORY TRAFFIC (%) RELATIVE TO DC FOR LENET-5-CAFFE

	conv1	conv2	ip1	ip2	total
weight	-9.1	-25.0	63.7	-181.6	48.2
input features	–	15.0	8.0	15.1	11.5
output features	15.0	8.0	15.1	–	13.6

Note: *For a similar accuracy, the drop degree (d_d) of SRA is set to 0.02. When drop_degree is set to a larger value, more energy can be saved.*

C. Evaluation of the influence to CNN accelerator

1) Storage: Under a similar accuracy drop, the proposed work further reduces the required storage size for lenet-5-caffe by 48.2% (see Fig. 5a), relative to the pruning percentage (p_p) setting in DC [4].

2) Energy efficiency: Energy consumption consists two portions of computation and memory traffic from external DRAM. CNN accelerators [9] [10] are usually kernel-based processing. Hence, if all weights within a kernel are pruned, the related computations can be omitted, and the energy can be saved. The reduced percentages of computational energy for only convolutional layers are listed in TABLE III, due to their much higher complexity than fully connected layers. For memory traffic, it is composed of buffering weights, input and output features. The amounts of them are respectively depends on the model size, input and output neuron numbers. Thus, the reduced energy is estimated by the compression ratio and pruned neuron numbers, and shown in TABEL IV. From the above evaluation, up to 45.2% computational energy and 11.5-48.2% memory traffic energy can be save with SRA and NRP.

V. CONCLUSION

This paper provides a sparseness ratio allocation to determine the pruned weight percentage for each layer, and a neuron re-pruning to remove the relative redundant neurons. Compared to deep compression, under a similar accuracy, this work can further reduce the model size by 48.2% on lenet-5-caffe and 24.4% on lenet-300-100. For the future work, we will evaluate the proposed SRA and NRP scheme on deeper CNN models and larger dataset of ImageNet.

ACKNOWLEDGMENT

This work is supported by JST, PRESTO Grant Number JPMJPR1757 Japan, by Waseda Univ. Graduate Program for Embodiment Informatics (FY2013-FY2019), and by Research Fellowships of Japan Society for the Promotion of Science for Young Scientists.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, ”Deep learning,” Nature, vol. 521, no. 7553, pp. 436-444, May 2015.

[2] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, ”EIE: efficient inference engine on compressed deep neural network,” in ACM/IEEE Int. Symposium on Computer Architecture (ISCA), pp. 243-254, June 2016.

[3] S. Han, J. Pool, J. Tran, W. J. Dally, ”Learning both Weights and Connections for Efficient Neural Networks,” in Advances in Neural Information Processing Systems (NIPS), 2015.

[4] S. Han, H. Mao, W. J. Dally, ”Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding,” International Conference on Learning Representations (ICLR), 2016.

[5] T.-J. Yang, Y.-H. Chen, V. Sze, ”Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[6] R. Abbasi-Asl, B. Yu, ”Structural Compression of Convolutional Neural Networks Based on Greedy Filter Pruning,” arXiv preprint arXiv:1705.07356, 2017.

[7] 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.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, ”Caffe: Convolutional architecture for fast feature embedding,” arxiv preprint arxiv: 1408.5093, 2014.

[9] Y.-H. Chen, T. Krishna, J. Emer, V. Sze, ”Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks,” IEEE Journal of Solid State Circuits (JSSC), vol. 52, no. 1, pp. 127-138, Jan. 2017.

[10] S. Wang, D. Zhou, X. Han, and T. Yoshimura, ”Chain-NN: An Energy-Efficient 1D Chain Architecture for Accelerating Deep Convolutional Neural Networks,” in Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1032-1037, Mar. 2017.