Simplified Neural Network Based on Auxiliary layer and Adaptive Pruning Rate

Reza Esfandiarpoor
Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran 84156-83111
r.esfandiarpoor@ec.iut.ac.ir

Mohsen Hajabdollahi
Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran 84156-83111
m.hajabdollahi@ec.iut.ac.ir

Nader Karimi
Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran 84156-83111
nader.karimi@cc.iut.ac.ir

Abstract—**Using of neural networks has increased in many applications by its increasing power to model, analysis and solve the problems in many different areas. The performance of neural networks is growing and so is the complexity of their structure. Although in recent years pruning and other similar works have been introduced as an appropriate solution to decrease their complexity, they are still very complex. In this paper, a method is proposed to reduce the complexity of neural networks by inserting an auxiliary layer and employing adaptive pruning rate. By insertion of an auxiliary layer after pruning, the network can be fitted on the given problem with less computational nodes. Simulation results on MNIST dataset represent a reduction of %22 in the number of network weights in comparison with the base pruning method. Also, an energy analysis performed on this dataset shows that it can be implemented with low energy consumption.**

Keywords-component; Artificial neural network, structural complexity, auxiliary layer, pruning.

I. INTRODUCTION

In recent years, deep neural networks (DNNs) are used extensively in many applications. Applying DNN in some applications leads to using powerful hardware devices such as many processing cores and powerful GPU devices. Although the powerful hardware devices can overcome the computational power required by DNNs but in real time and power-restricted applications this computational power cannot be provided. Recently many efforts have investigated the problem of simplifying neural networks by using different methodologies. In [1] a new method for training neural networks was presented to make network weights and activations in the form of binary. Gradients were computed during training and back propagated to the network parameters and made the binary weights and activations. In [2] the effects of limited precision in numerical representation of deep neural networks were investigated. Training deep neural networks with fixed-point representation was obtained based on stochastic rounding. Finally, by using 16-bit fixed-point representation, improved energy consumption in case of MNIST and CIFAR10 was obtained. Direct quantization has a crucial effect on the network accuracy [3]. Based on this observation in [3] a new quantization by ternary weights using retraining was used to make better results with fixed-point representation. Fixed-point variables were used during feed forward phase, but network sensitivities were back propagated using high precision values. In [4] an efficient approximation of convolutional neural networks based on two methodologies including binary weights and XOR net was presented. Filters were approximated using binary values in binary networks. In XOR networks, both filters and CNN inputs were in the form of binary. Using binary weights, 2.9 percent loss of accuracy was obtained in case of ALEX net classifier.

Pruning was introduced as another approach to simplify DNNs structure and reduce the storage and computation required by DNNs. The complexity of the network structure in DNNs makes it difficult to implement on mobile devices with limited power resources. In recent years pruning was performed with a greedy approach that makes it inefficient. In [5] a new compression method named dynamic network surgery was proposed which appropriately pruned the network connections. In this paper, those connections which are removed can be returned to the network structure after pruning. Simulation results show the compression ratio of 10.8 and 17.7 in case of LeNet and AlexNet, respectively.

In [6] a new method for reducing the computational complexity of neural networks was presented. Learning important connections leads to efficient simplification of the network parameters. Connections are pruned after accurate training and then the network is retrained for compensation of the error due to loss of connections. A 9× and 13× reduction in network parameters, in case of AlexNet and VGG-16, was obtained respectively.

Magnitude-based pruning of the weights may not adequately reduce the computational cost in the convolutional layers [7]. In [7] a simplification method for convolutional neural networks was proposed in which filters with a small effect on accuracy were pruned. This proposed method doesn't have irregular sparsity problem in the conventional pruning of CNNs. Simulation results showed 34- % and 38 % reduction in the network complexity in the case of VGG-16 and CIFAR10, respectively. CNNs were rarely applied on battery equipped mobile devices due to their great energy consumption [8]. In [8] a pruning based on energy

978-1-5386-4916-9/18/$31.00 ©2018 IEEE

consumption consideration was proposed which progressively pruned each layer. First, the layer's weights were pruned and locally fine-tuned. Then, the global fine tuning was performed, which in the case of AlexNet and GoogleNet produced reductions of 3.7× and 1.6× were achieved, respectively.

Increasing number of real-time mobile applications such as semantic segmentation leads to the implementation of DNNs on these devices [9]. In [9] a new deep neural network architecture was proposed for low latency applications. This new architecture based on feature down-sampling, factorizing filters and dilated convolution yields fast and accurate segmentation in case of CamVid, Cityscapes, and SUN dataset.

In [10] a deep neural network specialized for FPGA implementation was developed. On-chip memory usage has been possible with quantization in the form of 3-bit weights. Simulation results on MNIST dataset showed 5 Watts of power consumption on the quarter speed of a GPU implementation.

In most recent pruning studies the removed nodes cannot be involved again and have not any chance to properly come back to the network structure. Inserting auxiliary layer after pruning creates another chance to previously removed neurons to be involved in network training process. Therefore in this paper, a new method for pruning neural networks based on auxiliary layer insertion is proposed. An auxiliary layer including neurons is inserted into the pruned network structure. Static pruning rate leads to hard or soft pruning so an adaptive pruning rate is applied for efficient pruning.

The remainder of this paper is organized as follows. In Section II, pruning methods are briefly reviewed. In Section III, the proposed method, which is pruning with auxiliary layer insertion, is explained. Experimental results are presented in Section IV, and finally, this study is concluded in Section V.

II. NEURAL NETWORK PRUNING

In recent years, pruning has been introduced as an efficient way to reduce the neural networks' complexity and simplification of its structure. Some of these methods are summarized as follows.

Pruning can be performed by using local network parameters statistics. In [11] a new statistical pruning heuristic is proposed based on variance analysis to decide which units to prune [12].

Network pruning has been developed to remove redundancy, reduce the network complexity and avoid overfitting [6]. In Fig. 1, basic pruning method is illustrated which is based on learning both weights and connections. This method has three steps. In the First step, the network is trained conventionally. After that, the network connections are pruned. All connections with weights below a threshold are removed from the network. Finally, for error compensation, the network is retrained.

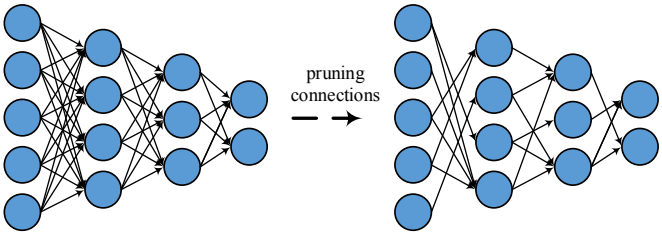

Figure 1. Basic pruning method

III. PROPOSED METHOD

Basic pruning schema proposed in [6] is considered. As it was mentioned in the previous section, after training the network weighs, the connections which are lower than a threshold are removed. For improving the basic pruning method, two effective improvements are proposed as follows.

A. Adaptive pruning rate

As it was mentioned in the previous section, pruning is an iterative process in which weights are pruned with a static pruning rate. In this way, weights, which are lower than a threshold, are removed, and there is not any chance to return to the network structure. Network weights are changed during pruning, so analysis of the changes in the network weights during the prune-train process can be useful for better pruning. For analyzing the effect of the static pruning rate on the network weights, an MLP structure on MNIST dataset is applied. MLP with 80-60-40 configuration is trained with basic accuracy 96.46 and then pruning is performed. Two static pruning rates are considered as follows.

1. Pruning with small pruning rate

In the first experiment, the network is pruned with static pruning rate of 0.2. Also, the maximum accuracy drop which is allowed is 0.5. Pruning results are illustrated in Table I. As illustrated in Table I, network is pruned after 13 steps. Also, the experiment is performed with the maximum allowed accuracy drop equal to 1 which took 30 steps to prune the network.

TABLE I. PRUNING WITH STATIC PRUNING RATE 0.2

Prune-train step	Number of remaining weights	Accuracy	Mean of weights
1	9570	0.9628	0.4624
2	8484	0.9628	0.5222
3	8239	0.9621	0.5379
4	8143	0.9624	0.5464
5	8034	0.9621	0.5668
6	7842	0.9620	0.5896
7	7701	0.9616	0.6088
8	7569	0.9613	0.6277
9	7446	0.9609	0.6445
10	7354	0.9608	0.6733
11	7163	0.9605	0.6926
12	7108	0.9603	0.7330
13	6872	0.9601	0.8005

As it is shown in Table I, pruning with small rate leads to a lot of iterations. A high number of iterations is because of the small changes that the network takes, so many iterations are required. Small pruning rate in the case of complex networks takes a lot of iteration steps which can be impossible for commodity hardware.

Algorithm1: Adaptive pruning

```
Basic training of the network
while accuracy > base accuracy
    Adaptive pruning (PR=0.1, step = 0.05)
    while True:
        PR += step
        Prune with PR rate
        if (Test accuracy drop > threshold)
            break
    PR -= step
    Prune with PR rate
    Retraining
    Go to while
END
```

It is depicted in Table I that the mean values of the weights are increased in each step. Small pruning rate decreases the possibility of removing the weights which are useful in early stages but become non-useful in the next iterations. For better understanding, a condition in which a sample network weight is useful in the early stages of the prune-train process is considered. It is possible that after some iterations a new structure is created in which the sample weight is not useful. In the pruning process with small pruning rate, network weights become larger and larger, and the network structure cannot be modified significantly. As the network is not able to modify any weights, it gets away from the optimum pruning.

2. Pruning with large pruning rates

The same experiment as before is performed with a large pruning rate of 0.4. In this experiment in each pruning step, lots of weights are removed. With maximum allowed accuracy drop of 0.5, the network is not capable of reaching an accuracy of 0.96, and the pruning fails. High pruning rates in some applications may result in a fast pruning process, but it may lead to high variations of the weights in the training process. Removing the weights with large values in the first steps of the train-prune process can be led to a harsh variation in the network structure in initial steps. Error due to the harsh variation in the early steps of the prune-train process, may not be compensated with retraining.

3. Adaptive Pruning Rate

As it was mentioned previously, after pruning the network is retrained, but the static pruning rate can lead to two possible problems. First, in the small pruning rate, the problem of lots of pruning iterations and inefficient pruning arises. Second, for a large pruning rate, the problem of great variation in the network weights and failing the network pruning process arises. Moreover, the static pruning rate becomes more inefficient when the notion of the large and small pruning rates is different in different applications. For adaptive pruning, the previous experiment is performed with adaptive pruning rate which begins with 0.1 in early steps of the pruning and is adapted in each iteration. Adaptive pruning procedure is represented in the form of pseudo code in the Algorithm 1. Variable PR is the pruning rate, which is adapted in each step. Variable “base accuracy” is the acceptable accuracy, and the “threshold” is the allowed decrease of accuracy from the “base accuracy”.

The results of the experiment are illustrated in Table II. As it is shown in Table II, the iteration steps required for pruning is 7, and a better-pruned network is obtained in comparison with the pruning with small pruning rate. In each step of pruning, pruning rate is adapted which allows the network to become stable by iterations. Also, pruning rate is gradually increased by increasing the mean value of the network weights, so the problem of newly non-useful weights which have high value is moderated.

TABLE II, PRUNING WITH ADAPTIVE PRUNING RATE

Prune-train step	Pruning rate	Number of remaining weights	Accuracy	Mean of weights
1	0.19	9789	0.9623	0.4823
2	0.24	7790	0.9620	0.5914
3	0.29	7093	0.9602	0.6527
4	0.26	6759	0.9606	0.6840
5	0.3	6482	0.96212	0.7463
6	0.29	6042	0.9610	0.8057
7	0.33	5667	0.9600	0.9156

B. Pruning based on auxiliary layer insertion

As it can be seen, after conventional pruning, there is no chance for pruned weights to come back to the network structure again. In the process of network pruning, it is possible that a non-useful weight in the current situation could be useful in the next steps. By inserting an auxiliary layer after pruning, another chance can be reproduced for useful weights which were removed previously. The pruning schema based on the auxiliary layer is illustrated in Fig. 2. As illustrated in Fig. 2, in the first step the network is pruned in such a way that the loss of accuracy is negligible. In the next step, the second pruning process is performed by inserting an auxiliary layer.

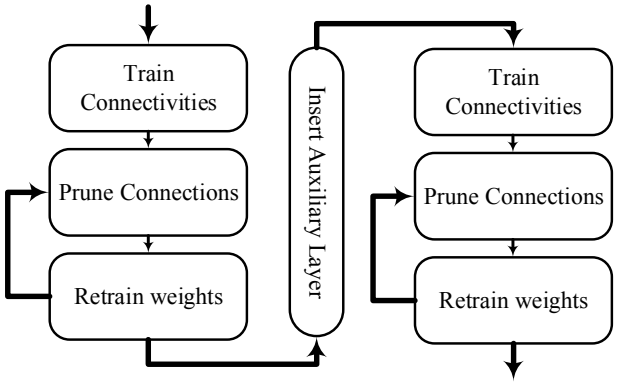

Figure 2. Pruning process based on auxiliary layer insertion.

As illustrated in Fig. 3, inserting an auxiliary layer after pruning could change the network structure. By changing the network structure, new opportunities are given to the connections which are removed previously. In fact, the inserted layer has the role of transferring the information which was lost in the previous structure. Since the loss of transferred information can be anywhere in the network structure, the inserted layer is added in a way that can be a link between one-half of the neurons to the other half of them. Due to the pruning step which is performed after the layer insertion, a fully connected layer is applied. Fully connected layers have a better operation in transferring the information between the network elements.

Experimental Results

The performance and the accuracy of the simplified version of the neural network have been tested using the MNIST dataset containing 60000 training images with the size of 28 × 28. For more simplification, all images of the dataset are downsampled to the size of 10 × 10 and then the neural network is applied. The performance of the proposed method is evaluated using 10-fold cross-validation method. Before any pruning method, the network is trained to reach the acceptable accuracy. The base accuracy of the network is considered as 0.9646. All the pruning processes are performed in such a way that does not affect the network accuracy.

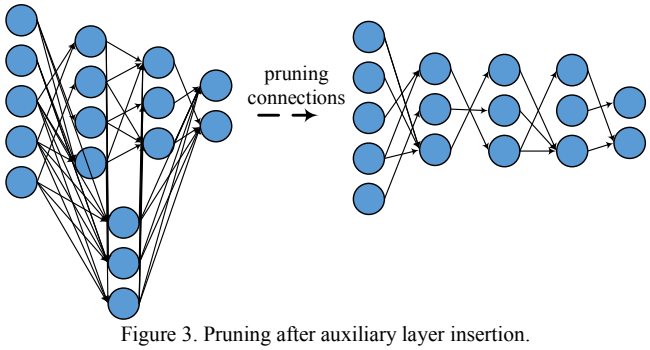

Figure 3. Pruning after auxiliary layer insertion.

C. Adaptive pruning rate

In the first experiment, the network structure is pruned with different pruning rates. Static Pruning rates 0.2, 0.3 and 0.4 are considered, and the remaining weights are reported in Table III. In the pruning process, the network is pruned with a limited accuracy drop. This experiment is performed with two cases of accuracy drop which are 0.46% and 1.0%. After retraining the pruned network, the accuracy drop should not be less than the specified value. Experimental results show that adaptive pruning rate is used for both cases of accuracy drop. In adaptive pruning rate approach, the network is pruned with small pruning rate in the beginning steps of the process. However, the pruning rate increases gradually in the next steps which leads to an efficient pruning. It is useful to note that the network pruning in case of pruning rate of 0.3 is better than the others. This static proper pruning rate must obtained through many experiments which can be cumbersome for networks with complex structures. Therefore, adaptive pruning rate gives proper results without a large number of experiments.

D. Pruning based on auxiliary layer insertion

From the previous experiment, the network structure was obtained by adaptive pruning rate and insertion of the auxiliary layer. Three cases are presented for the number of neurons in the auxiliary layer. Simulation results in Table IV, indicate that all three cases of auxiliary neurons reach lower number of remaining weights than the base adaptive pruning without auxiliary layer insertion. Also, it is observed that in case of inserting 40 neurons, the network has better results after pruning than the others.

TABLE III. REMAINING WEIGHTS AFTER PRUNING WITH DIFFERENT PRUNING RATES.

Accuracy drop / pruning rate	0.46	1.0	Average
0.2	6872	4587	5730
0.3	6302	4290	5296
0.4	Failed	4638	#
Adaptive	5667	4769	5218

TABLE IV. REMAINING WEIGHTS AFTER PRUNING BASED ON DIFFERENT AUXILIARY LAYERS.

Number of neurons	Remaining weights	Accuracy
20	5124	0.9604
40	4389	0.9601
60	4747	0.9607

E. Energy consumption based on computational operations

In this section energy consumption of the network's computational operations is presented. In Table V, rough energy costs for various operations are given for 45nm CMOS technology [13]. Pruned network structure resulted from the proposed method contains 4389 weights which are in the form of 32-bit floating point operations. While other simplification studies such as [2], [5] and [6] have energy cost equal to 358 nJ, 17 nJ and 82 nJ respectively, our pruned network has energy cost equal to 16 nJ.

TABLE V. ROUGH ENERGY COSTS FOR VARIOUS OPERATIONS IN 45NM

Operation	Multiply	Add
8-bit Integer	0.2pJ	0.03pJ
32-bit Integer	3.1pJ	0.1pJ
16-bit Floating Point	1.1pJ	0.4pJ
32-bit Floating Point	3.7pJ	0.9pJ

A small number of weights used in the resulted network structure shows that the proposed pruning method can be applied for simplification of the neural network structure. It is important to note that, the proposed pruning method reduces the network structure with the accuracy of 96.46 percent with a loss of accuracy lower than 0.5 percent. Our proposed pruning schema can be used for other network configurations which have already been pruned.

IV. CONCLUSION

A pruning method based on auxiliary layer insertion was presented in this study. Although removed connections due to the pruning were not important, it might become important after retraining. Inserting the auxiliary layer after pruning process has provided a new chance for the removed connections to come back to the network structure. Also, adaptive pruning rate during network pruning has led to a better structure pruning. Simulation results in case of MNIST dataset resulted in a network structure with only 4389 weights. Also proposed pruning method yielded 0.22 % reduction in the number of the network weights than the conventional pruning method. Proposed pruning method can be used for all networks with complex structures, especially the pruning rate which is updated in each step, reduces the training time significantly.

REFERENCES

[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio. "Binarized neural network", *Advances in neural information processing systems*, pp. 4107-4115, 2016.

[2] S. Gupta, A. Agrawal, K. Gopalakrishnan and P. Narayanan. "Deep learning with limited numerical precision", *International Conference on Machine Learning*, pp. 1737-1746, 2015.

[3] K. Hwang and W. Sung. "Fixed-point feedforward deep neural network design using weights+ 1, 0, and− 1", *IEEE Workshop on Signal Processing Systems,* pp. 1-6, *2014*.

[4] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. "Xnor-net: Imagenet classification using binary convolutional neural networks", Springer *European Conference on Computer Vision*, pp. 525-542, 2016.

[5] Y. Guo, A. Yao and Y. Chen. "Dynamic network surgery for efficient dnns", In *Advances In Neural Information Processing Systems*, pp. 1379-1387, 2016.

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

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

[8] T.J. Yang, Y.H. Chen and V. Sze. "Designing energy-efficient convolutional neural networks using energy-aware pruning", *arXiv preprint arXiv:1611.05128*, 2016.

[9] A. Paszke, A. Chaurasia, S. Kim and E. Culurciello. "Enet: A deep neural network architecture for real-time semantic segmentation", *arXiv preprint arXiv:1606.02147*, 2016.

[10] J. Park and W. Sung. "FPGA based implementation of deep neural networks using on-chip memory only", *IEEE International Conference Acoustics, Speech and Signal Processing (ICASSP),* pp. 1011-1015, 2016.

[11] A.P Engelbrecht. "A new pruning heuristic based on variance analysis of sensitivity information", *IEEE Transactions on Neural Networks*, *12*(6), 1386-1399, 2001.

[12] A.P. Engelbrecht, L. Fletcher and I. Cloete. "Variance analysis of sensitivity information for pruning multilayer feedforward neural networks", *IEEE International Joint Conference on Neural Networks,. IJCNN'99,* Vol. 3, pp. 1829-1833, *1999.*

[13] M. Horowitz. "computing's energy problem and what we can do about it", *IEEE International Conference* on *Solid-State Circuits Digest of Technical Papers (ISSCC),* pp. 10-14, 2014