R&D Project

A Comparative Study of Sparsity Methods in Deep Neural Network for Faster Inference

Desiana Dien Nurchalifah

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr Paul G. Plöger
Deebul Nair

September 2019

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date

Desiana Dien Nurchalifah

Abstract

Your abstract

Acknowledgements

I would like to express my gratitude towards the following people for helping me with this research and development project:

1. Both of my supervisors, Deebul Nair and Prof. Dr Paul G. Plöger, whose knowledge and insight into the subject had helped me during these 8 months.

2. All the people involved in the building of Platform for Scientific Computing at Bonn-Rhein-Sieg University for GPU and HPC consumption of this research.

3. My friend, Aldo Aditya, that helped me with the valuable advice for the writing of this research.

4. And to my parents who had given me support and encouragement.

To all of them, much obliged.

Contents

List of Figures

List of Tables

1

Introduction

Neural networks (NNs) are algorithms that are designed after a human brain, modeled to process a function of interest such that it is able to acquire knowledge from the environment and store the knowledge in synaptic weights. The first model of a neuron in NN was proposed in 1943, where Warren McCulloch and Walter Pitts create a model that consists of a single neuron and activation function using threshold logic unit (TLU). Weights as input are processed using TLU, where if it exceeds certain threshold, output will be high valued.

Network from the neurons defined as layered structure of neurons where it may range from single-layer to multiple-layer of neurons. The architecture of NN can either be deep or shallow. Deep Neural Networks (DNN) are NN with layer consists of more than 3. As DNNs are able to process tasks end-to-end, there are four kind of neural networks that is currently developed. Simple neural network, convolutional neural network, recurrent neural network, and hybrid neural network.

Data in each of the applications are represented and processed using DNNs. The performance of these tasks are aligned with the size of the network, where it is likely to increase as the network is made larger and deeper. As an example, residual network (ResNet) performance comparison on 20 layers network with 0.27 millions parameters and 110 layers network with 1.7 millions parameters provide 8.75% errors and 6.43% errors respectively.

However, a large network requires a large support of resources. Therefore, it is only applicable to recent hardware development, such as NVIDIA GPU. Large network is not able to be processed on devices such as mobile phones, Internet of Things (IoT), and wearables as three main problems resurfaced as stated in Network Slimming (Liu et al. 2017):

1. The size of the neural network : neural network contains millions of parameters that needs to be processed upon inferencing. These parameters consumes memory, for example, convolutional neural network training using ImageNet dataset consumes 300MB which considered a large consumption for mobile devices

2. Run-time memory : upon importing the neural network, while processing tasks, there are also responses created from the neural network that consumes more part of the memory and hence it burden resource-constrained devices to provide even more memory to allocate upon processing tasks which is possible for high-end devices such as GPU

3. Inference time : image classification tasks which mainly uses convolutional neural networks consists of several layers of convolution processes that requires minutes to process on each layer. This application is not possible on real-time application on mobile devices as it requires long time to process

This leads to development in compression of DNNs, hence a compressed DNN is able to be processed on mobile devices. There are various ways to compress a DNN, including application of sparsity to the network.

A sparse network is a network that contains fewer links from one part of the network, such as neuron or layers, to another as the connections among the network is eliminated or pruned based on certain conditions. The conditions to prune is based on the importance of the respective part of the network whether it possess the ability to contribute to the network performance. With these elimination, larger network can be built to obtain much better performance of the network. Sparsity by pruning, where network is trained, pruned and fine-tuned repeatedly until network become sparse, is mostly chosen method as pruning is proven to remove parameters on the network without harming accuracy of the process.(Han et al., 2015)

Sparsity on the network can be achieved in two ways, that categorizes as follows:

1. Structured method: a pruning technique that eliminates the whole layer of a network, leaving a network that is undamaged. There are three possible methods on recent development in structured pruning, that are channel pruning, filter pruning and layer pruning. This method conserves convolutional structure of the network, therefore it requires no specialized hardware or software to accelerate the process. Previous development in this approaches are: pruning channel based on weights value, activation and deactivation of channel connections randomly, pruning neurons, pruning based on average percentage of zeros in the output and group sparsity

2. Unstructured method: sparsity is referred to zero values in a subset of model parameters thus making the model able to be stored using sparse matrix format. These representation are stated as there are unnecessary parameters that can be eliminated from the network. Sparsity is a consequence of pruning technique which eliminates based on certain parameters and leaving a sparse network to be trained. Weight pruning is one of the examples, where it eliminates unimportant connections based on weights value. Smaller weights are pruned and leaves a network with zero weights. Residue network with sparse properties is able to save memory as model is sparse. Although this method provides higher compression rate than other methods, utilizing this method contains drawbacks such as requirements of specialized hardware or software to accelerate the processing, otherwise improvement in speed does not happen. As an example, on the work of The Lottery Ticket (Frankle and Carbin, 2019), speed to process the weight pruning was not reported.

Recent advances in sparsity include The Lottery Ticket and SNIP(Lee et al., 2018) stated that network that is sparse can be trained and thus deployed on a hardware. Although other work such by Liu et al. (2019) and Gale et al. (2019) stated that sparse network cannot be trained form scratch and therefore sparsity is introduced by structural pruning which is hardware-friendly. This raises the question of whether sparsity can actually be trained and deployed in a hardware. From the viewpoints of sparsity deployed in a hardware, several work supports that it is applicable as long as it is processed by removing the whole layer of the network. For example, in the work by NVIDIA (Molchanov et al., 2017), not only pruning whole feature-map is able to be applied on a hardware, it also speeds up the process by 3.4 times.

NetAdapt (Yang et al., 2018) also prune the network on filter-level that gains 1.7 times inference speed up. Similar works by removing filter are introduced in ThiNet(Luo et al., 2018) and a work by Chin et al. (2018) that not only structured sparsity can be implemented in a hardware, it also provides better performance in time consumption. These current advances in pruned DNNs have only been compared to the original network in accuracy measures whereas speed is not taken into evaluations. Therefore in this research project, it aims to compare and evaluate pruning method, that can be deployed on hardware without any required specialization either on the architecture or library dependencies. This work focuses on evaluating and experimenting which sparsity that is introduced in a hardware to achieve better time processing. It compares whether sparsity, either structured or non-structured in a hardware is able to gain better performance in time. It combines pruning methods, quantizing, and knowledge distillation. State-of-the-art pruning methods evaluated include both open-source and closed-source where the latter will be replicated based on the algorithm provided. Observations in this paper include as follows:

1. Pruned DNNs are able to be trained on hardware without specializations
2. Evaluation of speed in both combination of pruning, quantization and knowledge distillation and individual methods
3. Identifications of benefits and drawbacks on three categories of acceleration after deployment

Evaluation criteria of the methodologies are based on speeds, accuracy, floating point operations (FLOPs), and the number of parameters used.

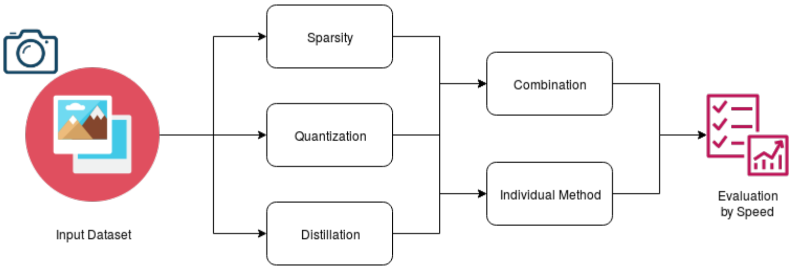

Figure 1.1: Figure 1.1 Research Illustration

1.1 Motivation

1.2 Challenges and Difficulties

1.2.1 Individual Methods

1.2.2 Combined Methods

1.3 Problem Statement

2

State of the Art

There are several ways to obtain faster inference for DNN. One of the method is model compression. Compressing models of DNN can be divided into three major classes: [15] removal of DNN structure that is redundant, approximation of DNN function, and architecture search which creates and designs a compact DNN. This research is concentrated on the first two methods.

Model compression by removal of redundant structures, or pruning, are extensively studied. Pruning the network is an example that is widely used as it is able to simplify DNN models while also retain the performance of the original models. [13]

Knowledge distillation, on the other hand, is an example of a method that belongs in approximating DNN function. A large computationally expensive model is able to be compressed into a single computational efficient neural network. [10].

In this section,each method will be introduced and then explained how it is used in the experiments on section 4.

2.1 Pruning Methods

Pruning in neural network field is an act to reduce the extent of the network by removal of superfluous or unwanted parts. Pruning methods are divided into two main categories: unstructured pruning (fine-grained pruning) and structured pruning (coarse-grained pruning) [12]

2.1.1 Unstructured Pruning

Unstructured or fine-grained pruning is a method to eliminate weight parameters of the neural network that are deemed unnecessary. As the result network will be sparse, specialized hardware or libraries will be needed in order to fasten the inference of the network.[9]

Several works developed in this methods are individual weight pruning based on Hessian matrix, [3] deep compression by training, pruning, and fine tuning [2], and variational dropout [11]

2.1.2 Structured Pruning

Structured pruning or coarse-grained pruning is a method to eliminate channels or layers of DNN. Channels or layers are determined through the importance either globally (automatic pruning) or locally. Structured pruning is delved more as it requires no specialized hardware or software. [14].

In line to structured pruning methods, In [1], Patricle filter is used to obtain smaller DNN.

2.2 Quantization Method

2.3 Knowledge Distillation Methods

Deep Neural Network (DNN) operates with long inference time as the network is deep and contains a lot of parameters. Knowledge distillation [5] aims to reduce this by creating a student network where it consists of less number of parameters and less depth that learns from a teacher, in this case, is a DNN.

There are two main points that it differs from transfer learning, the first point is that the model architecture in transfer learning is similar with shared layers, while in knowledge distillation (KD) the architecture between the students and the teacher are not the same. The second point is that in transfer learning, it copies all the weights of a DNN, while in KD it only imitates the Teacher.

Therefore, in this learning method, the student is learning from a Dark Knowledge

2.4 Limitations of previous work

3

Methodology

The goal of this research is to compare methods of model compression to gain faster inference with minimum tradeoff to accuracy. Sections below elaborates setup and experiment design to the work.

3.1 Setup

The network is tested on GenuineIntel Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz. Processor running on 64 bits with 15GB RAM to represent embedded devices.

Inference measurement is done using modified benchmark program from MLMark that measures inference by using ”Latency” and ”Throughput” metrics.

Latency used is defined by time of processing of a single input through one iteration in miliseconds. Measurement is based on industrial metric where it is taking 95% of the time the machine is able to perform well. The Python program takes warming up predictions and then calculate average inference over 10 iterations.

Meanwhile in throughput, unit is in frames per second as the model's task is classifications of images. Throughput is calculated by multiplication of iterations and batch sizes divided by amount of time required to process.

3.2 Experimental Design

Design of the experiments conducted are divided into four sections:

1. Comparison of pruning methods
2. Utilization of quantization
3. Comparison of knowledge distillation methods
4. Integration of three methods

These four sections are compared using elaborated metrics, that are latency, throughput, and accuracy.

Networks are trained using PyTorch 1.0.1 and Python 3.6.9 that runs on CUDA 9.2 and cuDNN 7.4. Hardware for training has specifications as follows:

1. Nvidia Tesla V100 SXM2 GPU with 5120 Cuda cores
2. 16 GB memory
3. system interface PCIe 3.0 x16
4. Nvidia Volta architecture

Experiment is limited to CIFAR-10 [6] dataset. It is a well-known dataset to compare model compression techniques.

4

Evaluation

Below are the results of model compression implementations.

4.1 Pruning Methods

Methods compared include L1 norm pruning [7], L1 pruning with batch normalization and induced sparsity [8], and weight pruning [2]. These methods are compared based on the FLOP reductions that it acclaims. The network used as a baseline is ResNet-56 with pre-activation. Pre-activation on ResNet is used as it is able to alleviate the problem of vanishing gradients on deeper networks. [4]

In the figure, the dot in the middle is the measurement, right and left dot is the standard deviation of the measurement.

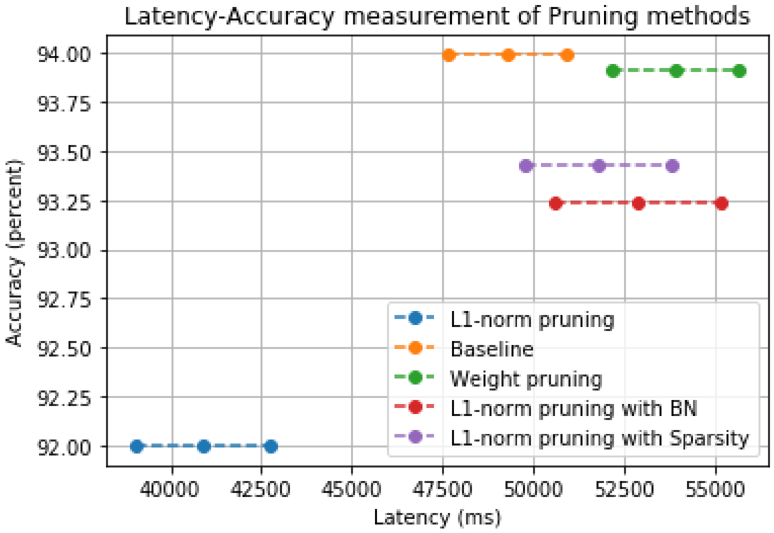

Figure 4.1: Comparison of Latency for Pruning Methods

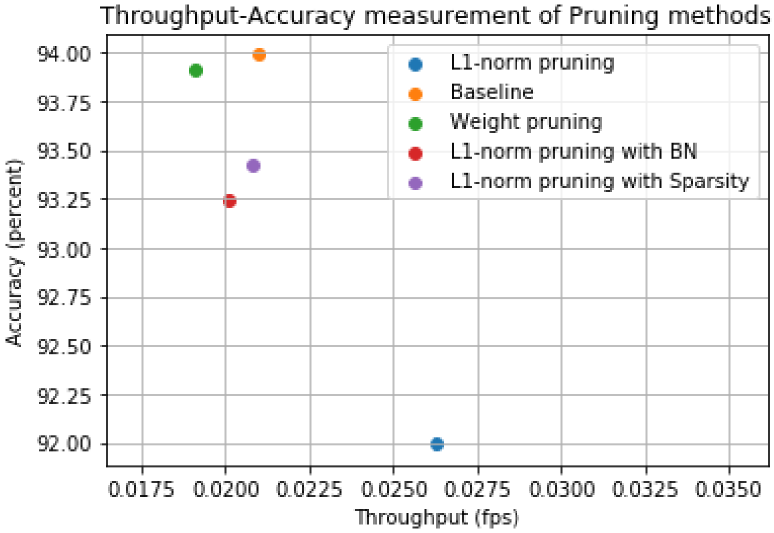

Figure 4.2: Comparison of Throughput for Pruning Methods

5

Results

5.1 Comparison of Pruning Methods

From the results, comparison on the latency of the network provides weight pruning, as an example of unstructured method, performs worse than structured methods. While L1 pruning without adding batch normalization or sparsity performs the best among all methods. Although latency is obtained the best on L1 norm, it includes trade-off with the lower performance on accuracy.

Meanwhile in throughput measurement, the higher the number of frame-per-second, provides faster inference of the network. In this figure, it is also concluded that L1 norm pruning exceeds among modified methods and unstructured pruning.

5.2 Use case 2

5.3 Use case 3

6

Conclusions

Based on the results obtained, sparsity of the network affect negatively to latency. There is also a trade-off for accuracy and latency, where the lower the latency is tolerated, the lower accuracy of neural network will be. Sparsity of the network also affects latency by prolonging the latency itself.

6.1 Contributions

6.2 Lessons learned

6.3 Future work

A

Design Details

Your first appendix

B

Parameters

Your second chapter appendix

References

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *J. Emerg. Technol. Comput. Syst.*, 13 (3):32:1–32:18, February 2017. ISSN 1550-4832. doi: 10.1145/3005348. URL `http://doi.acm.org/10.1145/3005348`.

[2] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.

[3] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 164–171. Morgan-Kaufmann, 1993. URL `http://papers.nips.cc/paper/647-second-order-derivatives-for-network-pruning-optimal-brain-surgeon.pdf`.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

[5] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

[6] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[7] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ArXiv*, abs/1608.08710, 2016.

[8] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[9] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *ArXiv*, abs/1810.05270, 2018.

[10] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *ArXiv*, abs/1902.03393, 2019.

[11] Dmitry Molchanov, Arsenii Ashukha, and Dmitry P. Vetrov. Variational dropout sparsifies deep neural networks. In *ICML*, 2017.

[12] V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, Dec 2017. doi: 10.1109/JPROC.2017.2761740.

[13] Yehui Tang, Shan You, Chang Xu, Boxin Shi, and Chao Xu. Bringing giant neural networks down to earth with unlabeled data. *ArXiv*, abs/1907.06065, 2019.

[14] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2074–2082. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6504-learning-structured-sparsity-in-deep-neural-networks.pdf`.

[15] Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. Learning intrinsic sparse structures within long short-term memory. *ArXiv*, abs/1709.05027, 2017.