A REPORT ON DEEP COMPRESSION: *COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING*

MD. Muzakker Hossain, 18-37801-2; Zahin Awosaf, 18-37064-1; MD. Loton, 18-37583-1.

1. INTRODUCTION:

Neural networks are computer systems of interconnected nodes that act just like human brain neurons. Using algorithms, they can recognize hidden patterns and correlations in raw data, cluster and classify it, and – over time – continuously learn and improve [2]. The applications of neural networks are widely used nowadays. For example, suppose you are visiting Russia and saw a road sign but couldn't figure out the meaning but with help of google translation you can easily find out the meaning and it is an example of many applications of neural networks. But the implication of neural networks requires heavy datasets and storage space. For example, a dataset VGG-16 requires storage capacity of 552 MB which makes it difficult to deploy neural networks on different embedded systems and mobile networks. To resolve this issue "Deep Compression" is introduced. Deep Compression is a combination of three series data processing elements. Firstly, a network pruning is performed upon the networks and it can reduce the number of connections by 9x to 13x. Then, network quantization and weight sharing further compresses the pruned network by reducing the number of bits that represent each connection from 32 to 5 which reduces the weight by 27x to 31x. Finally, with the help of Huffman Coding the weights and index are encoded which improves the compression rate by 35x to 49x. This enormous compression rate doesn't only resolves the storage issue but also reduces energy consumption without any loss of accuracy. For example, the dataset VGG-16 which requires 552 MB of storage capacity can be reduced to 11.3 MB meaning the networks can be implemented into SRAM cache rather than DRAM memory which will be a perfect fitting into embedded systems and mobile networks.

2. TECHNIQUES PERFORMED:

The three staged pipeline to attain deep compression are – Network Pruning, Trained Quantization and Huffman Coding.

2.1 NETWORK PRUNING:

Network pruning is a common approach to reducing a heavy network by minimizing redundancy throughout the heavy network to achieve a light-weight network. In this method, in order to achieve comparable performance with reduced parameters, a complex over-parameterized network is first trained, then pruned is done based on certain criteria and finally fine-tuned.

2.2 TRAINED QUANTIZATION:

Network quantization is carried out to further simplify the pruned network by limiting the amount of effective weights that need to be processed by sharing the same weight with multiple connections, and then fine-tuning those shared weights, resulting in a substantial reduction in memory requirements and computational costs.

To calculate the compression rate, given k clusters, only log2(k) bits is needed to encode the index. In general, for a network with n connections and each connection is represented with b bits [1], constraining the connections to have only k shared weights will result in a compression rate of:

$$r = \frac{nb}{nlog_2(k) + kb}$$

For example, for the weights of a single layer neural network with four input units and four output units, there are 4x4 = 16 weights originally but there are only 4 shared weights: similar weights are grouped together to share the same value [1].

Originally 16 weights must be stored where each has 32 bits, now it is possible to store only 4 effective weights, each has 32 bits, along with 16 2-bit indices giving a compression rate of 16*32 / (4*32 + 2*16) = 3.2 = 3.2.

2.3 HUFFMAN CODING:

Huffman code is a special type of optimal prefix code widely used to compress lossless data. It uses variable-length codewords to encode source symbols. The table is derived from the occurrence probability for each symbol, where fewer bits represent more common symbols. In the datasets the probability distribution of quantized weights and the sparse matrix index of the fully connected layer are biased around the two peaks. Experiments show that Huffman coding these non-uniformly distributed values saves 20% – 30% of network storage [1].

3. EXPERIMENTS:

To validate the hypothesis the researchers experimented on four different datasets of different network architecture. Two of them were selected from MNIST and others were selected from ImageNet datasets. The first experiment was conducted on MNIST dataset with LeNet-300-100 and LeNet-5 network where LeNet-300-100 is a fully connected network with two hidden layers with 300 and 100 neurons each and LeNet-5 is a convolutional network that has two convolutional layers with two fully connected layers. The experiment shows that the compression pipeline can save 35x to 49x parameter storage with no loss of accuracy. The weights of LeNet-300-100 was 266K before the deep compression, after pruning (P) the weights reduced to 8%, then further reduced to 3.1% after Pruning + Quantization (P+Q) and finally Huffman coding (P+Q+H) gave a marginal gain by reducing it to 2.49%. For LeNet-5 the initial weights were 431K, after compressing the three staged reduction were (P = 8%), (P+Q) = 3.05% and finally (P+Q+H) = 2.55%. Most of the saving comes from pruning and quantization (compressed 32x) and Huffman coding contributed a marginal gain (compressed 40x) [1].

The research was further experimented with ImageNet ILSVRC-2012 datasets which has 1.2M training examples and 50k validation examples. AlexNet Caffe model was used as a reference model of having 61M parameters. After applying Deep Compression (P+Q+H) it reduced to 2.88% of its original size with a promising result of top-1 accuracy of 57.2% and top-5 accuracy of 80.3%. AlexNet's performance inspired researchers to apply the compression technique to larger and newer networks. VGG-16 networks has far more convolutional with only three fully-connected layers. To realize a substantial reduction in the number of effective weights, researchers adopted similar methods and actively compressed both convolutional and fully-connected layers. Until compression, the VGG-16 had a weight of 138M and was drastically reduced to 2.05 percent, which is 49 times the actual size. The initial parameter size of 552 MB was then reduced to 11.3 MB, which means that it is compact enough to fit into the SRAM on-chip, removing the need to store the model in energy-consuming DRAM memory.

4. DISCUSSION:

4.1 PRUNNING AND QUANTIZATION WORKING TOGETHER:

The experimental data for accuracy vs. compression rate under different compression methods shows that pruning and quantization works better when combined than being applied individually. The compression rates were about 13% and over 20% for pruning and quantization respectively, but when they were combined the compression rate drops remarkably to 7%. Question may arise about the distortion of bits while doing quantization as pruning reduces the number of parameters from the weights. But the experiment result shows that the accuracy rate remains same for quantization with or without pruning the networks. Moreover with the help of pruning the numbers of parameters are reduced enormously which shrinks the massive weights of the networks. On pruned networks, quantization works well because unpruned AlexNet has 60 million weights to quantify, while pruned AlexNet has just 6.7 million weights to quantify. Provided the same number of centroids, there is less error in the latter.

4.2 SPEED UP & ENERGY EFFICIENCY:

The main aim of this compression technique is to fit the immensely larger neural networks into different embedded system and mobile networks. The primary focus is on latency-based mobile applications that require real-time inference along with the connection between embedded systems. To benchmark the performance and energy efficiency of the compressing technique, it was compared in three different hardware systems such as, the NVIDIA GeForce GTX Titan X and the Intel Core i7 5930K as desktop processors and NVIDIA Tegra K1 as mobile processor[1]. They obtain 3x to 4x speed over the dense network as the pruned networks have smaller memory and mitigate the data transfer overhead, particularly for large matrices that can not fit into the caches. To find out the energy efficiency the researchers multiplied power consumption with computation time to get energy consumption, then normalized it to CPU and got 3x to 7x less energy consumption for pruned networks.

4.3 RATION OF WEIGHTS, INDEX & CODEBOOK

Comparison between different compression techniques such as, Baseline Caffemodel (BVLC), Fastfood-32-AD, Fastfood-16-AD, Collins & Kohli, SVD, Pruning, Pruning + Quantization and Deep Compression (Pruning + Quantization + Huffman Coding) were performed on AlexNet where the Deep Compression gave the best result of 35x reduced size and the parameters were down to 6.9MB where only Pruning + Quantization were 27x of compression rate with parameters size of 8.9MB. The Deep Compression performed better with a negligible 1% accuracy loss.

5. FUTURE SCOPE & CONCLUSION:

As some libraries do not support indirect matrix entry lookup (for which quantization with weight sharing is not applicable) the fully advantage of Deep Compression may not be availed. To solve this issue, either a software solution of writing customized GPU kernels that support this indirect matrix entry or a hardware solution of building custom ASIC architecture is required. In this way an energy domination by on-chip SRAM access instead of off-chip DRAM access can be expected. Deep Compression is a promising compression technique that makes it easier for mobile applications to use complex neural networks where application size and download bandwidth are limited.

REFERENCES:

[1] Han, S., Mao, H. and Dally, W.J., 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*.

[2] “Neural Networks: What they are and why they matter”, [online], Available: https://www.sas.com/en_us/insights/analytics/neural-networks

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