Compressing Deep Neural Networks

Tuguldur Baigalmaa
Earlham College
801 National Road West
Richmond, IN, USA
tbaiga13@earlhame.edu

ABSTRACT

Deep neural networks have become an important area of research due to the recent success on a number of recognition tasks, such as speech recognition, computer vision and natural language processing. However, the increasing depth, that comes with increasing accuracy of such models, makes it both computationally and memory intensive to deploy to mobile platforms and embedded systems. In this survey paper, we will examine a range of compression methodologies that can be used to decrease model size and increase runtime efficiency.

Keywords

Network Pruning; Quantization; Huffman Coding; Hessian Matrix

1. INTRODUCTION

There is an increasing demand to be able to run deep neural networks on mobile platforms and embedded systems. Deploying the model directly to the target platform that has limited resources has several advantages, such as less network bandwidth, real-time inference and storage space reduction. Although most of the compression methodologies mentioned in the survey paper have been applied to image recognition tasks using Convolutional Neural Networks (CNNs), such as AlexNet and LeNet, recent research has also expanded to cover Neural Machine Translation and Speech Recognition tasks using Long Short-Term Memory (LSTM) Recurrent Neural Networks since the general concepts are equally applicable.

2. GENERIC COMPRESSION METHODS

In this section, we'll explore network pruning, quantization, weight sharing and Huffman Coding as generic methodologies to compress a neural network.

2.1 Network Pruning

After a training phase, neural networks are typically overparameterized with a significant redundancy. *Network pruning* or *weight pruning* has been widely researched as a means to reduce the number of redundant parameters. There are different approaches to pruning the networks. The initial naive implementation relied on a magnitude-based approach, which prunes parameters that have a parameter close to zero. Among the most popular are the Optimal Brain Damage (OBD)(Le Cun et al., 1989) and Optimal Brain Surgeon (OBS) (Hassibi and Stork, 1993) techniques, which involve

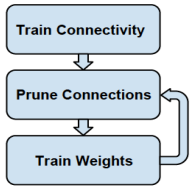

Figure 1: Network Pruning pipeline.

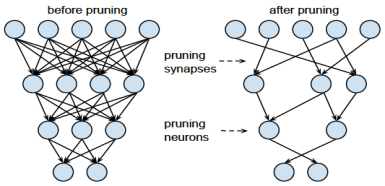

Figure 2: Synapses and neurons before and after pruning.

computing the Hessian matrix of the loss function with respect to the parameters, in order to assess the *saliency*of each parameter. Parameters with low saliency are pruned from the network and the remaining sparse network is retrained. However, both OBD and OBS were proven to be computationally complex especially for larger networks (Augasta and Kathirvalavakumar, 2013). Recent advancement in magnitude-based network pruning has focused on pruning parameters that have a magnitude below a certain threshold to preserve the original model accuracy (Han et al., 2015a). However, as noted by Han et al., there is a trade-off between accuracy and the number of the parameters. If too many parameters are pruned away, the network becomes less accurate. The trade-off curve is drawn on Figure 3.

The network pruning pipeline outlined on Figure 1 can be iterated to further compress the network. However, there is no clear method to determine as to how many iterations should be run and what the threshold should be in order to preserve the optimal number of parameters. The average percentage of zero activations (APoZ) was proposed as a measure to evaluate the importance of each neuron in a network to address the issue mentioned above. APoZ is defined as follows:

$$APoZ_c^{(i)} = APoZ(O_c^{(i)}) = \frac{\sum_k^N \sum_j^M f(O_{c,j}^{(i)}(k) = 0)}{N \times M} \quad (1)$$

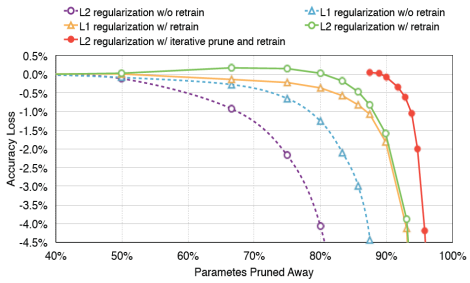

Figure 3: Trade-off curve between parameter reduction and loss in top-5 accuracy of AlexNet.

where $f(\cdot) = 1$ if true $f(\cdot) = 0$ if false, M denotes dimension of the output feature map of $O_c^{(i)}$, and N is the total number of validation examples.

Then, the neurons with an APoZ value larger than one standard deviation from the mean APoZ are pruned away iteratively. However, the researchers were able to reach a compression rate of only 2.59x as compared to 13x (Han et al. 2016) on the standard VGG-16 model while preserving the original accuracy. However, the statistical approach of pruning redundant neurons still remains as a viable alternative to other approaches, which involve intensive computational power and manual tuning. Hu et al. also note that their approach is tailored more towards using GPU for computation instead of CPU in Han et al. The most recent paper from Han et al. has also introduced a computationally efficient network pruning approach with load-balance-aware pruning (Han et al., 2016b). Their newly proposed approach involves dividing the matrix into different sub-matrices, which can be compressed in parallel.

2.2 Quantization

The stored sparse structure after network pruning is compressed further through quantization. Both network quantization and weight sharing further compress the network by reducing the number of bits required to represent each weight. In quantization, weights are represented in 8-bits. Recently, a more complex quantization routine has been proposed to use the Hessian-weighted distortion measure as a loss function for quantization errors and has been proved effective in increasing the compression rate compared to the results of the Deep Compression pipeline (Choi et al., 2016). The researchers have managed to preserve the original accuracy, while achieving impressive compression rates. They have achieved compression rates of 51.25, 40.65. The full statistics can be seen on Table 1:

However, there is a performance trade-off as it can be computationally expensive to calculate the second order derivatives (Han et al., 2015).

2.3 Weight Sharing

The network is further compressed by having multiple connections share the same weight. k-means clustering technique is used to identify shared weights for each layer of a trained network. The weights that fall into the same cluster share the same weight. Three approaches for centroid initialization are proposed: Forgy (random), density-based and linear initialization (Han et al. 2015). Linear initialization has proven to work the best because there are only a few

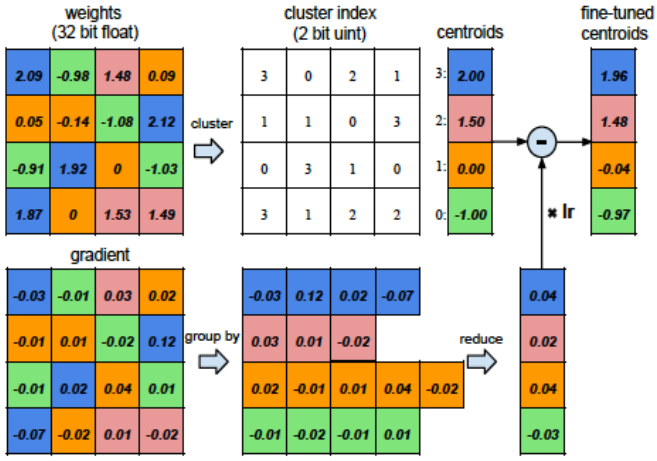

Figure 4: Weigth sharing by scalar quantization (top) and centroids fine-tuning (bottom).

large weights that significantly affect the output variable.

2.4 Huffman Coding

Huffman code is an optimal prefix code commonly used for lossless data compression(Van Leeuwen, 1976). It encodes source symbols with variable length codewords. More common symbols are represented with fewer bits. Han et al. note that Huffman coding saves an additional 20-30% network storage. Choi et al. also confirm the usage of Huffman coding, noting that it is one of the most optimal coding schemes when distribution of a source is provided. Huffman coding also works especially well in the pipeline, following uniform quantization.

3. COMPRESSING LSTM

LSTM is a Recurrent Neural Network (RNN) architecture that has numerous applications in speech recognition and natural language processing. Although the majority of the compression research had been focused on traditional CNNs that are used for image recognition, there have been recent attempts to compress RNNs using a similar set of methodologies (Han et al., 2016b). They create a load-balance-aware pruning pipeline, as well as, quantizing the model weights into 12-bit integers using linear quantization strategy. Although the paper in question also addresses hardware accelerators that could be used to further improve the efficiency of the RNN, it is outside the scope of this paper. Experimental results indicate that they were able to compress LSTM by 12 times without sacrificing accuracy of the model. See et al. also take on the challenge of compressing a LSTM. They compress a Neural Machine Translation Model (NMT) using LSTM. Although their general approach to network pruning is also magnitude-based, they have tried experimenting with alternative approaches (class-blind, class-uniform, class-distribution) and found that class-bind pruning has been proven to outperform other approaches. Class-blind pruning takes in all parameters, sorts them by magnitude and prunes the x% of the smallest magnitude, regardless of the weight class. As usual, they also retrain the model after pruning to adjust the weights to accommodate for the pruned parameters. They were able to compress the

			Accuracy %	Compression ratio
LeNet5	Original model		99.25	-
	Pruned model		99.27	10.13
	Pruning + Quantization all layers + Huffman coding	k-means	99.27	44.58
		Hessian-weighted k-means	99.27	47.16
		Uniform quantization	99.28	51.25
		Iterative ECSQ	99.27	49.01
	Deep compression (Han et al., 2015a)		99.26	39.00
ResNet	Original model		92.58	-
	Pruned model		92.58	4.52
	Pruning + Quantization all layers + Huffman coding	k-means	92.64	18.25
		Hessian-weighted k-means	92.67	20.51
		Uniform quantization	92.68	22.17
		Iterative ECSQ	92.73	21.01
AlexNet	Original model		57.16	-
	Pruned model		56.00	7.91
	Pruning + Quantization all layers + Huffman coding	k-means	56.12	30.53
		Hessian-weighted k-means	56.04	33.71
		Uniform quantization	56.20	40.65
	Deep compression (Han et al., 2015a)		57.22	35.00

Table 1: Summary of network quantization results with Huffman Coding for pruned models.

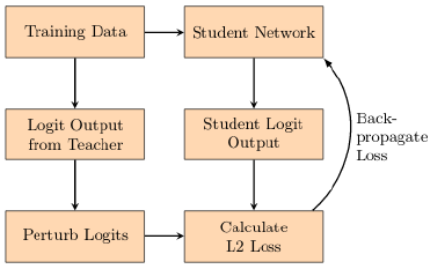

Figure 5: Training Shallow Students using the proposed Logit Perturbation Method.

model to achieve a 65.2% reduction in storage space.

4. KNOWLEDGE DISTILLATION

Knowledge distillation is a technique described by Hinton et al. (2015) and involves training a small 'student' network on the outputs of a large 'teacher' network. The knowledge distillation approach is fundamentally different, but the main idea is still to reduce the model size by reducing redundancy. Shen et al. have succeeded in creating a model with 400x fewer parameters while outperforming AlexNet on a specific dataset *Caltech Pedestrian Dataset*. The main idea is to create a smaller network that mimics the behavior of the larger network. As a result of the sparsely generated network, the student model performs 8x faster than the teacher with 21x less memory usage. Figure 5 illustrates the process for training the student.

Sau et al. have also demonstrated the usage of knowledge distillation technique for reducing the model parameters. They were also able to achieve compression rate of up to 33x on the MNIST, SVHN and CIFAR datasets. Another potentially significant idea they introduce is to train the student network on multiple teachers, which could potentially improve network accuracy even further.

5. CONCLUSION

The recent research on compressing deep neural networks is finally proving it possible to deploy sophisticated models into mobile platforms and embedded systems. Research has been done both on traditional feedforward CNNs, but also, RNN architectures, which use backpropagation. In both cases, the general ideas of network pruning, quantization and weight sharing have proven to significantly reduce the number of parameters, while preserving original, sometimes exceeding, original model accuracy.

6. REFERENCES

[1] Song Han, Jeff Pool, John Tran and William J. Dally. *Learning both Weights and Connections for Efficient Neural Networks*. arXiv:1506.02626 [cs], 2015.

[2] Song Han, Huizi Mao and William J. Dally. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. arXiv:1510.00149 [cs], 2015.

[3] Hengyuan Hu, Rui Peng, Yu-Wing Tai and Chi-Keung Tang. *Network Trimming: A Data-Driven Neuron*

Pruning Approach towards Efficient Deep Architectures arXiv:1607.03250 [cs], 2016.

[4] Yoojin Choi, Mostafa El-Khamy and Jungwon Lee. *Towards the Limit of Network Quantization* arXiv:1612.01543 [cs], 2016.

[5] Abigail See, Minh-Thang Luong and Christopher D. Manning. *Compression of Neural Machine Translation Models via Pruning* arXiv:1606.09274 [cs], 2016.

[6] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, Huazhong Yang and William J. Dally. *ESE: Efficient Speech Recognition Engine with Compressed LSTM on FPGA* arXiv:1612.00694 [cs], 2016.

[7] Bharat Bhusan Sau and Vineeth N. Balasubramanian. *Deep Model Compression: Distilling Knowledge from Noisy Teachers* arXiv:1610.09650 [cs], 2016.

[8] Jonathan Shen, Noranart Vesdapunt, Vishnu N. Boddeti and Kris M. Kitani. *In TeacherWe Trust: Learning Compressed Models for Pedestrian Detection* arXiv:1612.00478 [cs], 2016.