BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2023
http://excel.fit.vutbr.cz

Evolutionary Optimization of Convolutional Neural Networks

Vojtěch Čoupek*

Abstract

The topic of this paper is the weight compression of convolutional neural networks. It uses a Weight-Sharing technique to obtain a high level of compression ratio. The technique is applied layer-wise to amplify the compression effect; however, it requires parameter optimization, which is done by Genetic algorithm or Particle Swarm Optimization. Furthermore, additional compression using Quantization and post sharing fine-tuning are explored. The implementation presented in this paper is then tested on Le-Net-5, which leads to 20× weight size reduction with 0.62% accuracy loss on MNIST and Mobilenet_v2 showing 4.86 times compression with 2.26% accuracy loss on Imagenet subdataset called Imagenette [1].

*xcoupe01@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Modern convolutional neural networks are robust solutions to handle tasks otherwise hard to code, such as the classification of pictures, segmentation of MRI data, etc. In recent years, the main focus is on improving accuracy by modifying the structure and enlarging the number of trainable parameters. However, these complicated neural networks consume a lot of energy to evaluate solutions that are not viable for mobile devices like cellphones.

Sze et al. [2] proves that the vast majority of energy consumed by inferring a neural network comes from loading the parameters from storage. So the energy consumption should be reduced by compressing the network's data. But compressing the network too much can lead to accuracy losses, so it needs to be in balance.

The net can be compressed in a number of ways:

1. Refining the data (like weights)
2. Refining the net structure
3. Refining the operator

This paper explores the data refinement approach where the main representative is Quantization, which takes the original weights and transforms them into other representations in order to seek compression (i.e. from float32 to float16). This approach can be beneficial because it allows hardware optimisation for further energy savings. This method can also be used during training to let the network adapt to it. However, the end compression is limited by the employed number of representations.

Another solution, which is discussed in this paper, is Weight-Sharing, that takes a group of weights, runs clustering algorithm to rank them into the given number of groups (typically K-means), and then replaces them with keys to these groups. A translation table with a key and group representing the value is created. When the net is loaded, the keys in the net structure are replaced with the values from the table by the corresponding key. This technique is applicable for different weight groups (model-wise or layer-wise groups). This paper explores layer-wise Weight-Sharing, because it demonstrates better performance and is used in many papers (for example in [3]). The number of representatives/clusters chosen for each layer is determined by an optimization algorithm in a given range. No retraining is done through the compression to ease the computational demands of this approach.

A dynamic target based fitness function is presented for the optimisation. The premise is that the fitness function is defined as a distance to some ideal unreachable target (in the accuracy and compression axis), but it is hard to tell what that target should be without closely studying the network first. The

dynamic target allows to reduce the need of testing the network and permits the target to be dynamically shifted to push the optimisation for better results. In some scenarios, no net knowledge is needed to get good results using this approach.

Furthermore, this paper explores the benefits of additional Quantization on the representatives in the translation table to achieve additional compression. In the project, float32 is the base representation, which can be converted through Quantization to float16 or float8 representations. Also a fine-tuning method is explored to retrieve some accuracy by modifying the clustering space.

The approach was tested on two types of Le-Net-5 (two types - Tanh and ReLu activation functions) and Mobilenet_v2 with the following results.

2. Poster Commentary

The poster describes the proposed solution for the approach. The Figure 1 shows the whole process of the net compression in a graph-like form (on Le-Net-5 example). The computation goes from top to bottom. There are shown the two inputs that are needed for the compression. These are the target neural network and the layer ranges for a number of clusters.

Then, the search space optimization on the cluster ranges is done. This is proposed by Dupuis et al. [4]. It takes the network and, for each layer and each cluster option in this layer, it tries to compress only the current layer's weights and measures the compression rate, while the premise is that if the accuracy drops too low, the cluster value is not useful and will not be included in the search space. All layers and their clusters number precision are plotted in the line graph. This is shown in the Figure 1 in the "Range Optimization" section.

Next, the optimization and weight sharing is depicted. Firstly, the new random population is created. Each member of the population represents the number of clusters for each layer of the net (e.g., there will be 5 values for Le-Net because Le-Net has 5 layers). After that, the members are scored by combining compression rate (CR) and accuracy (ACC) by equation 1. To compute these metrics, it is necessary to perform the weight sharing and evaluate the compressed network. The process of weight sharing is shown in the figure 1 in the "Weight Sharing" section with the description of the weight compression performance. As mentioned in Sect. 1, additional Quantization is carried out, which is also shown in the Figure 1 in section "Additional Quantization" with a picture of different types that are used in the project. Lastly, the CR and ACC metrics are used to update the fitness target (ACC_{target} and CR_{target}) if necessary and then every candidate solution is scored by equation 1. The scores for each candidate solution are processed by the optimization algorithms shown in the Figure 1 in the "Optimization algorithm" section and the cycle is repeated.

After the optimization completion, the fine-tuning is performed. The proposed fine-tuning's idea is based on modulating the space for the clustering algorithm. The modulated space is shown in figure 1 in the "Fine-tuning" section. The space is modulated by the Tanh function because it enlarges the distances in the function's zero point while shrinking the relative space on the edges of the space. It is based on the idea that most of the weights are around zero point and the weight distribution is usually normal. By this modulation, it is possible to shift some clusters towards the zero point and have more precision for weights located there. The resulting weight clustering is shown in figure 1 in the "Fine-tuning" section (can also be used to further explain clustering of weights and Weight-Sharing principle). After that, the network is compressed.

On the right side in the first paragraph, goals and brief step-by-step description of the approach is presented. It discusses the core ideas behind the project and also describes the dynamic fitness target approach. This is easier to understand by observing how the fitness and compression rate are computed in equations 1 and 2. A graph in figure 2 can be used to further explain this functionality.

The results are then presented separately for each network with a corresponding dataset. Firstly, the Le-Net-5 results are discussed in table 1. It shows the best obtained results for each net type and Quantization setting. Below that in Figure 2, a graph depicting the search on Le-net-5 ReLu is shown, which highlights the difference in the performance for various Quantization approaches and with and without the fine-tuning method. In this graph, it is clearly observable that the additional Quantization can be beneficial and can lead to better results, but this way has also its limits (float16 performed better than float32 but float8 did not perform well). Also, as mentioned above, the fitness function and dynamic target can be explained using this graph. In table 2, results on Mobilenet_v2 network are presented. Note that the float8 is crossed out because no viable solution was found.

References

[1] Jeremy Howard. Imagenette. https://github.com/fastai/imagenette/.

[2] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[3] Junru Wu, Yue Wang, Zhenyu Wu, Zhangyang Wang, Ashok Veeraraghavan, and Yingyan Lin. Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5359–5368. PMLR, 2018.

[4] Etienne Dupuis, David Novo, Ian O'Connor, and Alberto Bosio. A heuristic exploration of retraining-free weight-sharing for cnn compression. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 134–139, 2022.