LegoNet: Memory Footprint Reduction Through Block Weight Clustering

22-12-2023

Introduction

In an era where neural networks are becoming increasingly central to technological advancements, the efficiency of these models is a paramount concern. This is particularly critical when deploying complex models on devices with constrained memory resources, like smartphones or IoT devices. "LegoNet: Memory Footprint Reduction Through Block Weight Clustering"[1] introduces an approach to address this challenge using weight clustering. By clustering blocks of weights within neural networks, LegoNet promises to reduce the memory footprint of these models without compromising on accuracy or necessitating retraining.

This blog post aims to reproduce and critically analyze LegoNet's methodology and claims, offering insights into its potential impact on the field of neural network efficiency. Our aim is to accurately reproduce the original paper's results using the PyTorch framework, with an emphasis on verifying LegoNet's efficiency in memory-constrained environments. Furthermore, we are open sourcing our code as a contribution to the community, fostering further exploration and innovation in neural network optimization.

LegoNet Overview

LegoNet introduces a novel technique for compressing neural networks by clustering blocks of weights into representative blocks. This approach, unlike traditional methods, does not compromise the model's accuracy or require retraining. By leveraging this weight clustering strategy, LegoNet significantly reduces the memory footprint of large neural network models. This is particularly beneficial for deploying complex models in memory-constrained environments like mobile devices or embedded systems. The paper presents a thorough analysis of LegoNet's performance, demonstrating its effectiveness in comparison to other state-of-the-art methods.

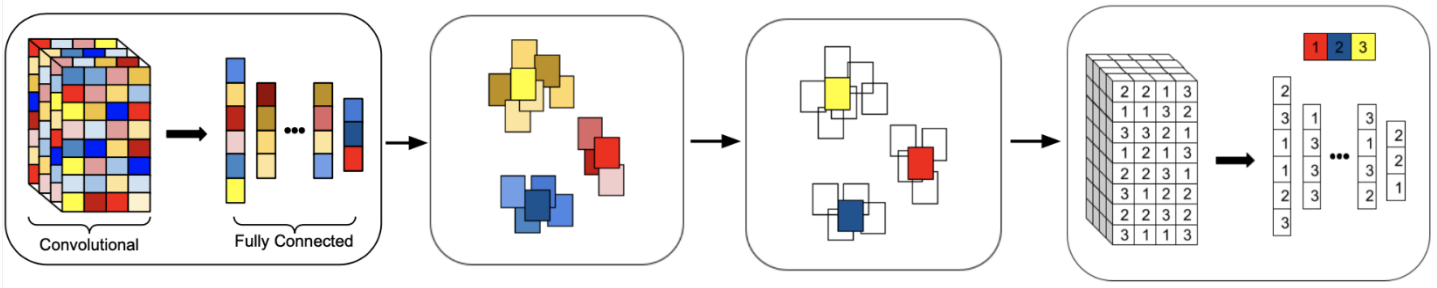


Figure 1: Overview of LegoNet. First a trained model's weight matrices are chunked into blocks, or Legos. Then the Legos are clustered into groups. Then the blocks are replaced by the index of the cluster they belong to. At inference time, the centroid representative of this cluster is used as the weight values.

The LegoNet algorithm initializes with input parameters: the model to compress (M), the number of clusters (K), and the dimension of Lego pieces (b). Subsequently, the model's weights are divided into b × b blocks, agnostic to layer types or positions. These blocks undergo clustering, and the centers become Legos for model reconstruction. Each block is compared to the centers, determining the minimal Euclidean distance. An illustration of LegoNet is presented in Figure 1.

Compression Ratio

For an uncompressed model M, let P be the number of parameters in the model and wordlength (32bits in PyTorch) be the number of bits required to represent the model elements. Therefore, the model size is:

$$|M| = P \times worldlength$$

If weights are grouped into clusters by employing a clustering algorithm such as K-means, the shared values can be stored in a codebook, and the weight matrix values are replaced with their cluster index. Therefore, the model size would be:

$$|M'| = P \times |\log K| + K \times worldlength$$

, where |Log K| is the number of bits required to store K values and K x worldlength is the size of the codebook. This paper proposed LegoNet, a block weight clustering method in which each weight is clustered into groups of b x b, and the index is stored using |Log K| bits. They defined the compression ratio as:

$$CR = \frac{|M|}{|M'|} = \frac{P \times worldlength}{P \times \frac{\lceil \log_2 \quad K \rceil}{b \times b}}$$

Experiments

Models and Datasets. We assessed the proposed method using pre-trained VGG16 [5] and ResNet18 [6] models obtained from Torchhub and Torchvision [2], conducting evaluations on CIFAR-10 [3] and ImageNet [4] datasets.

Hyperparameters. We conducted an evaluation of Legonet under two settings. Initially, we examined the performance with K=50 and b=4, evaluating accuracy by replacing the blocks with cluster centers. Additionally, we assessed the performance with b=2 and K=128.

Replicating Main Experiments

Our experiments aimed to replicate Table 1 from the paper, examining the performance of these models on the ImageNet and CIFAR-10 datasets. Our methodology closely mirrored that of the original study, where we reported the compression ratio (CR) and Accuracy Loss for each model and dataset.

Model	Dataset	Baseline Acc.		Accuracy	CR	Accuracy	CR	
		Reported	Ours	Reported	Reported	Ours	Replaced Blocks	Ours
ResNet-18	CIFAR-10	83.70	92.44	81.61	128	90.54	90	85.33
				83.44	64	91.32	97	18.28
ResNet-18	ImageNet	69.76	69.76	67.66	128	61.83	53	85.33
				69.21	64	61.04	90	18.28
VGG16	CIFAR-10	90.02	93.22	87.62	128	70.96	58	85.33
				90.07	64	91.08	96	18.28
VGG16	ImageNet	71.59	71.59	68.96	128	66.52	90	85.33
				71.57	64	69.97	97	18.28

Table 1: Main experiments performed on CIFAR-10 and ImageNet using ResNet-18 and VGG16. Experiments replicate Table 1 of the original paper. The reported values include Top-1 Accuracy on the test set and Compression Ratio (CR) for each model and dataset. Additionally, the Replaced Blocks indicates the percentage of blocks replaced with clusters centers in our experiments, with accuracy reported under two different CR settings: 85.33 (K=50, b=4), and 18.28 (K=128, b=2).

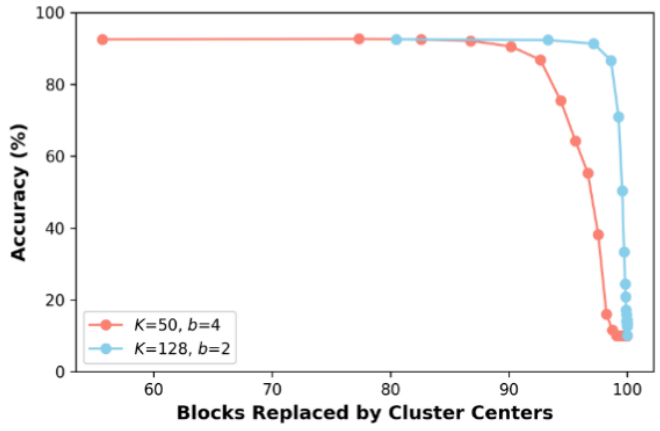
From the results of Table 1, we notice a gap between the reported and our baseline accuracy of both VGG16 and ResNet-18 in CIFAR-10. Moreover, we examined the compressed models at compression ratios (CR) of 64x (LegoNet-A) and 128x (LegoNet-C), as initially indicated in the paper. However, when we compressed the model at CR=85 and 18 times, we observed a significant loss in accuracy, even with a relatively small compression ratio of 18 times. In particular, for ImageNet dataset, authors had claimed a

0% accuracy loss with a 64x reduction in memory footprint (Table 2 of original paper), something we could not clearly achieve in our experiments.

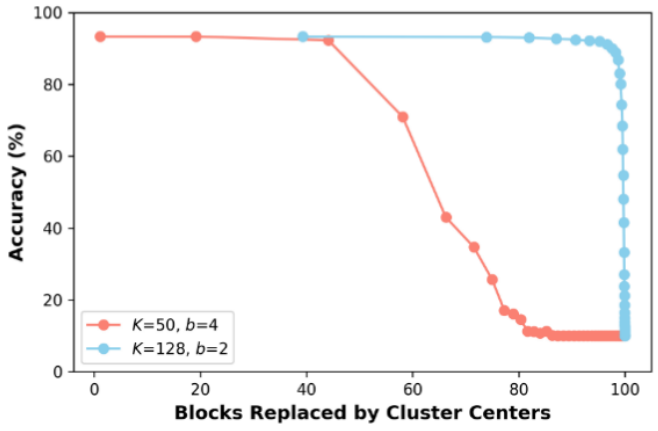
Effect of Blocks Replacement on Model Performance

To further investigate the effect of replacing model weights with their closest cluster center in each block we performed experiments using VGG16 and ResNet-18 on CIFAR-10 and ImageNet, where we varied the number of blocks being replaced with their closest cluster center.

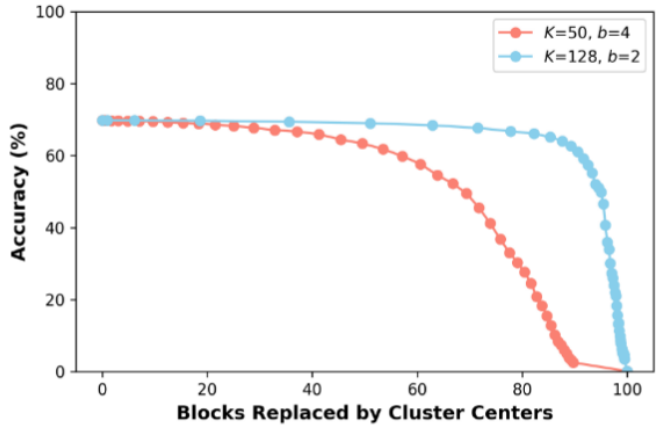
In Figure 2, we notice that while the model exhibits a robustness in replacement of blocks with cluster centers for low levels of replacement (between 20-80% based on the underlying classification task difficulty and dataset), replacing all blocks with cluster centers has a catastrophic effect on model performance. This further verifies our finding from Table 1, where an accuracy loss was observed for high CR values.



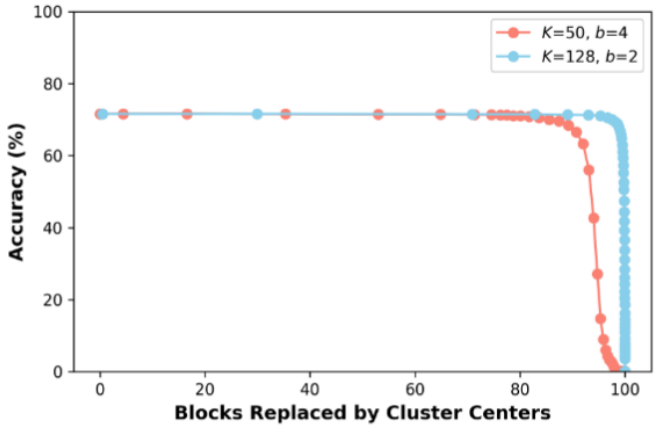
(a) ResNet18 on CIFAR-10



(b) ResNet18 on ImageNet



(c) VGG16 on CIFAR-10



(d) VGG16 on ImageNet

Figure 2: Effect of block replacement in model performance. In all experiments, the Top-1 Accuracy on test set is reported.

Our experiments indicate that, although this method shows promise for achieving significant compression (in terms of memory/storage requirements) by clustering network blocks with cluster centers, it fails to replicate the claims made in the original papers. Despite this, we believe that a deeper exploration of this clustering approach might be beneficial, particularly in distributed machine learning contexts like Federated Learning. In these scenarios, where model parameter communication is a major training bottleneck, transmitting only a limited number of blocks could substantially reduce communication costs.

Contributions

This blog post is the result of a collaborative effort by Vasileios Tsouvalas and Saeed Khalilian Gourtani, created during the Winter School on Efficient Deep Learning, part of ASCI Course A16. Both authors have contributed equally to the re-implementation of LegoNet and the writing of this blog post.

Citations

[1] J. Bingham, N. Green and S. Zonouz, "LegoNet: Memory Footprint Reduction Through Block Weight Clustering," *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, Falerna, Italy, 2022, pp. 1-6, doi: 10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927846.

[2] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf\

[3] Krizhevsky, Alex. (2012). Learning Multiple Layers of Features from Tiny Images. University of Toronto.

[4] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255).

[5] Karen Simonyan, & Andrew Zisserman. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. (2015). Deep Residual Learning for Image Recognition.