Analysis of Collaborative Compression: Extensive Experiments and Better Implementation

Yuchen Liu

Abstract

Channel pruning and tensor decomposition are both typical CNN compression techniques, but they usually penalize network performance tremendously when pursuing high compression rates. In this paper, we extend the experiment on Collaborative Compression mechanism, which incorporates both strategies and achieves high accuracy after fine-tuning. In order to compress custom blocks in ResNet-56, we add a layer to the original CC pipeline. Moreover, we implement a new workflow for loading the compressed CNN architecture, so that model compression and testing are separated. We compress the model at the stride of 3 and 9, generating 14 fine-tuned models. Comparing their confusion matrices evaluated on CIFAR-10, we discuss the effect of CC comprehensively and the limitation of fine-tuning. Our study contributes to the comprehension of multi-stage CNN compression and facilitates the implementation of Collaborative Compression (CC) technique.

I. Introduction

Convolutional Neural Networks(CNN) have achieved state-of-the-art performance on various computer vision tasks, such as image classification[1], [2], video segmentation[5] and object detection[4], [3]. Despite the improvement in generalization ability, modern CNN architectures are typically over-parameterized[6]. As the size of CNN architectures grows, it becomes more challenging to deploy these models on resource-constrained edge devices that are limited in memory, energy, and computation power. To meet such demand, various methods have been invented and discussed. The mainstream techniques to compress and accelerate CNN include sparsification[7], channel pruning[8], quantization[9], [11], knowledge distillation[12], [13], and tensor decomposition[14], [19]. Some techniques specify training requirements and some compress CNN architectures on the pre-trained models, and they all have separate stages for training and compression.

This paper aims to extensively explore a newly proposed CNN compression scheme, [23]Collaborative Compression (CC), which joints channel pruning and tensor decomposition to compress CNN models by simultaneously learning the model sparsity and low-rankness. This method demonstrates superior performance gains over previous ones on various datasets and backbone architectures. Significantly, CC claims to control the compression rate automatically by a simultaneous trade-off between the sparsity and low-rankness in weights, attaining high model accuracy after fine-tuning.

Thus, it's intriguing to explore CC's ability to investigate the compression sensitivity of each convolutional layer in CNN and determine target channels to prune based on previous layers' compression. Specifically, this paper intends to discuss CC's influence on each sections of CNN as well as the global performance. Compared with model compression, the subsequent fine-tuning process takes the majority of time , whose number of epochs is a hyper-parameter to be automated.

II. Related Work

In general, network pruning can be categorized into either unstructured or structured pruning. Notably, the latter one is capable of reducing both the size and reference time of CNN models, as a result of less FLOP operations. For example, filter pruning[26] removes redundant filters by different importance measurements. Similarly, channel pruning[27] deletes input channels without dimensional mismatch in multi-branch networks, such ResNets[24] and DenseNets. Furthermore, tensor decomposition[10] approximates weights of low-rank filters based on the intrinsic low-rankness of convolutional parameters, which gained more attention in recent years as they can achieve higher compression rates in comparison to other methods.

The low-rank approximation is based on factoring weight tensors into lightweight representations which leads to much fewer parameters and consequently fewer computations[22]. Many canonical methods such as Singular Value Decomposition(SVD), Tucker[16], and CP decomposition[19] have played important roles in this field. Due to their inherent nature, these canonical methods are unable to provide a good compression rate while maintaining the model's accuracy. To solve their problems, plenty of more advanced tensor decomposition methods are introduced. Studies[17],[18] propose the training-aware compression technique which compresses the network during training by utilizing regularization. However, one significant drawback of training-aware compression is that they are difficult to balance the compression rate and accuracy given the target compression rate.

Inspired by the flaw, [23] Collaborative Compression is proposed to automatically compress CNN to a custom global compression rate, and achieve slightly penalized performance after fine-tuning.

III. Methods

A. Collaborative Compression Scheme

As shown in Fig.1, CC first determines the compression rate of each layer by global compression rate optimization(GCRO), then compress each layer independently and synchronously by multi-step heuristic compression(MSHC). Considering that compression of a layer will affect the importance of the remaining compression units, MSHC prunes unimportant channels step-by-step.

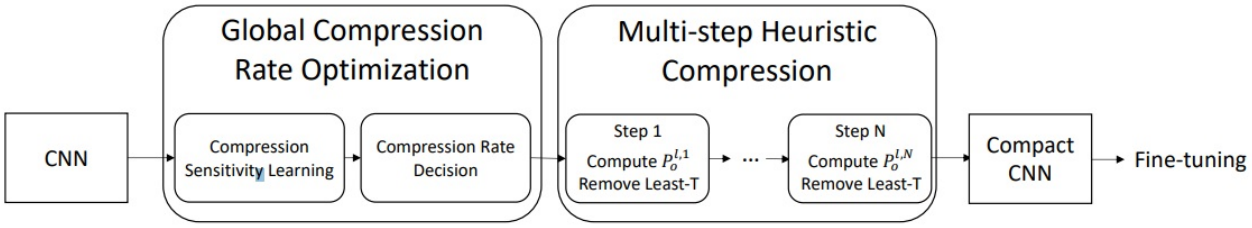

Global Compression Rate Optimization
Multi-step Heuristic Compression
CNN
Compression Sensitivity Learning
Compression Rate Decision
Step 1 Compute $P_o^{l,1}$ Remove Least-T
...
Step N Compute $P_o^{l,N}$ Remove Least-T
Compact CNN
Fine-tuning

Fig. 1: Compression pipeline of Collaborative Compression.

1) Global Compression Rate Optimization: Inspired by [15], CC measures the information loss at the l-th layer by adopting the first-order Taylor-based approximation of the network loss on the compressed weights $\mathcal{M}^l$

$$I^l = \left[L\left(\overline{\mathcal{W}}^l\right) - L\left(\mathcal{W}^l\right)\right]^2 \approx \sum_{i \in Q^l}\left(\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i^l} * \left(\overline{\mathcal{W}}_i^l - \mathcal{W}_i^l\right)\right)^2 = \mathrm{S}\left[\left(\mathcal{G}^l * \left(\overline{\mathcal{W}}^l - \mathcal{W}^l\right)\right)^2\right]$$

Then CC utilizes a greedy algorithm to evaluate the information loss under different compression rates. Given the whole network compression rate C, CC decides the compression rate of each layer by optimizing

$$\min_{\mathcal{R}}\left(\sum_{i=1}^{L} F^i \cdot R^i - C \cdot F\right)^2$$

2) Multi-Step Heuristic Compression: Given that the importance of units will change due to the removal of other units because of the mutual influence between different compression operations, CC proposes a new importance metric for trimming convolutional layers according to the nature of the Markov Decision Process.

$$P_o^{l,(t)} = I_o^{l,(t)} + \gamma \frac{1}{\left|\mathcal{U}^{l,(t)}\right| - 1} \sum_{i \in \mathcal{U}^{l,(t)} \setminus o} I_{i|o}^{l,(t)}$$

The time complexity of the above computation is $\mathcal{O}((c^l + r^l)^3)$. To speed up evaluation to $\mathcal{O}(c^l + r^l)$, they approximate the process by introducing a balance hyper-parameter T, which is the number of removed units after each evaluation."

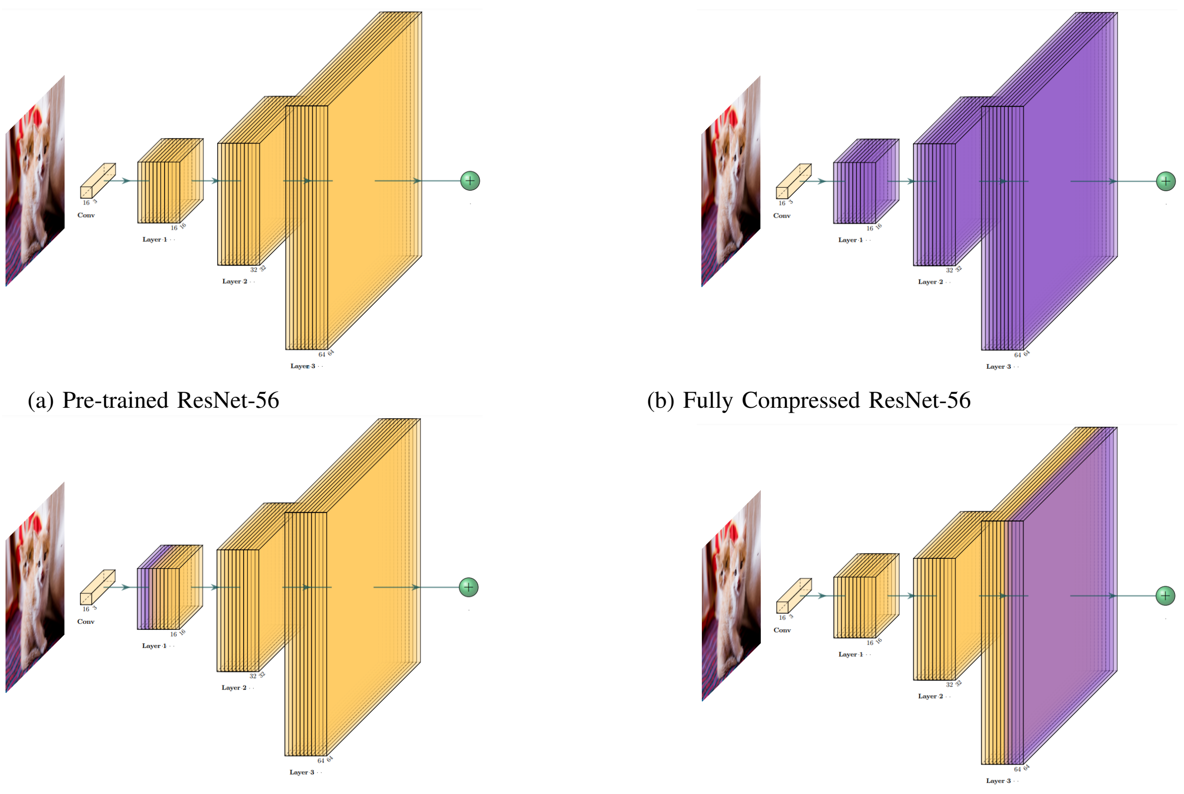
B. Extensive Study with ResNet-56

In order to separate testing from compression pipeline, it's necessary to store some relevant parameters of each compression unit, so that the compressed architecture can be reconstructed. For example, the shape of each weight in each convolutional layers is determined after compression, and we need this extra information to generate the custom ResNet-56 architecture, where the saved model can be loaded. We only need to record 2 sets of parameters, the complete ResNet-56 as in Fig.2a and the fully compressed one as in Fig.2b, and ablation models can be generated by masking.

The first ablation aims to understand the relationship between compression rate and accuracy of compressed models. To that end, assign a range of global compression rates to CC pipeline, compare test accuracy between different compressed models and original base model.

The second ablation intends to inspect the actual performance of collaborative compression. One metric is the difference between the assigned compression rate and actual number of parameters after compression. Besides, we'd like to know how CC affects each residual unit. Since there are 27 units in ResNet-56, compress every 3 units to generate 9 ablation models, fine-tune 100 epochs and test the performance discrepancies. Fig.2c and Fig.2d demonstrate 2 outcome models. Intuitively, taking away redundancy in every 3 units shouldn't hurt the overall performance, and fine-tuning may restore the accuracy a little bit.

In addition to overall test accuracy, various metrics can be derived from the prediction confusion matrix, such as per-class accuracy, variance of that, the most difficult class to predict, the most common mislabel per class.



(a) Pre-trained ResNet-56

(b) Fully Compressed ResNet-56

(c) Only compress block 1, 2, 3

(d) Only compress block 25, 26, 27

Fig. 2: Examples of ResNet-56 ablations.

IV. Experiments and Discussion

The experiments in mainly written in Pytorch on Google Colab Platform, including compression, fine-tuning, testing and visualization. To reserve computation power, I choose ResNet-56 as the base model and evaluate on CIFAR-10[25] dataset. Leaving the first convolutional unit uncompressed, there are 3 layers in ResNet-56, each layer is consisted of 9 residual blocks, and each block has 2 compressible units. Thus, we can compress certain units out of 54 to compress and fine-tune the custom model. The input images in CIFAR-10, whose shape is 32 x 32, has 10 classes. The training set contains 50, 000 images, while test set 10, 000.

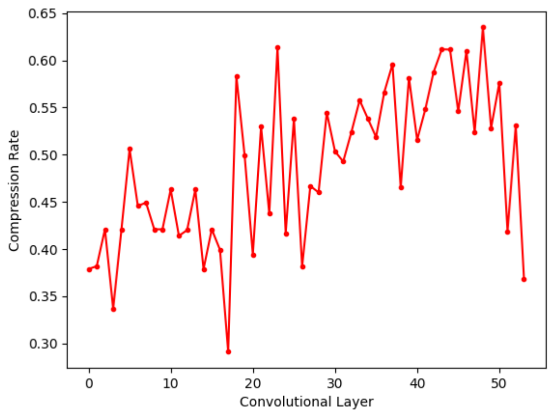
A. *Experiment Settings*

To calibrate with the original paper[23], all networks are fine-tuned via SGD with momentum 0.9 and mini-batch size of 256. Every compressed model is fine-tuned for 100 epochs, checkpoint models every 10 epochs and save the model that scores the highest. The inference test loads the best model, and generate its confusion matrix evaluated on CIFAR-10 test set. The learning rate is set to 0.1 and is multiplied by 0.1 at 50% and 75% of the total epoch number.

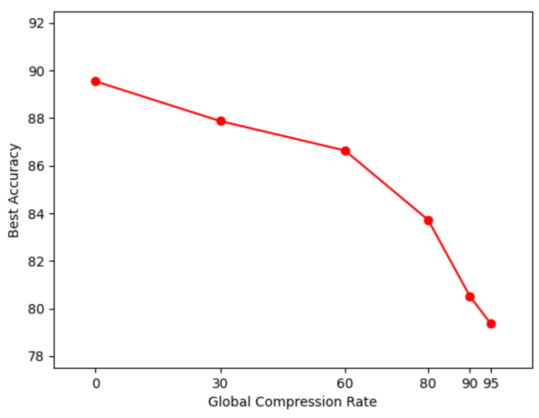
B. *Compression Rates Study*

First of all, we re-implement the entire pipeline with global compression rate at 50%. Load the pre-trained model as in Fig.2a with 853018 parameters, collaborative compression of 27 blocks yields the model in Fig.2b with 455599 parameters. The top-1 accuracy is 88.79%, close to the result in [23]. The actual number of parameters seems to have small discrepancy with the assigned one. What's more, we also inspect the compression rates of 54 convolutional layers, as illustrated in Fig.3a. The serrated curve implies that some layers are less redundant in the view of CC, which dynamically adjusts compression rate for each layer. Specifically, the importance of each CNN layer varies, even if we take the compression that is already performed into consideration.

To investigate how the collaborative compression penalizes the inference ability of CNNs and how fine-tuning recovers it, ResNet-56 is compressed by a range of compression rates, from 0 to 95%. After fine-tuning 300 epochs, the compressed models appear to converge. As shown in Fig.3b, the top-1 accuracy on CIFAR-10 drops along with increasing global compression rate, and it becomes harder for fine-tuning to recover accuracy. For example, if ResNet-56 is compressed at 95%, even after 100 epochs of fine-tuning, its accuracy can hardly improve to 80%.



(a) Compression rate of 54 convlutional layers



(b) Per-class accuracy variance

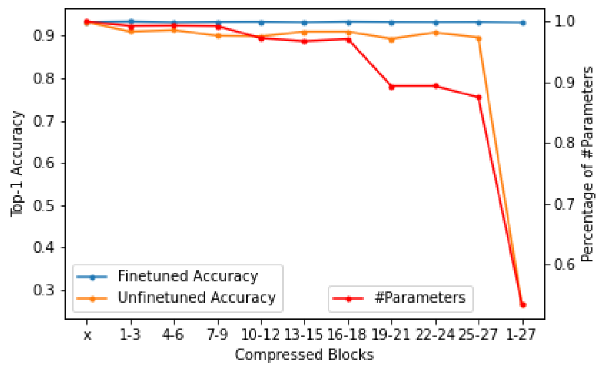
Fig. 3: Compression rates study of CC on ResNet-56

C. Regional Compression Study

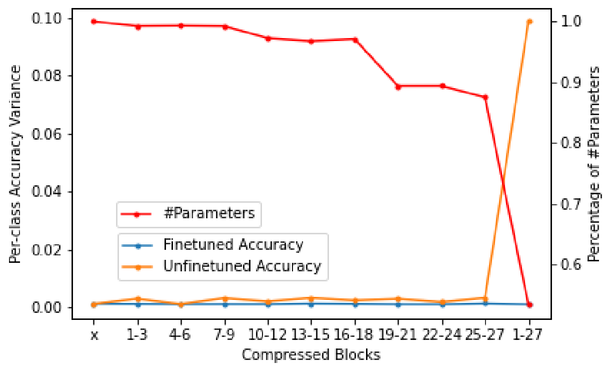
In order to compress custom block, we add a middle-layer to Fig.1 between GCRO and MSHC. Among block 1 to 27, we can choose a custom continuous section to compress, while the other blocks are masked with the original pre-trained weights. Afterwards, we fine-tune the ablation model as before. Here chooses every 3 blocks as the stride of experiment. When testing the custom models, architectures are also generated by masking Fig.2a and Fig.2b.

TABLE I: Test Results for Ablation Models Before and After Fine-tuning

Compressed Blocks	**x**	**1-3**	**4-6**	**7-9**	**10-12**	**13-15**	**16-18**	**19-21**	**22-24**	**25-27**	**1-27**
Finetuned Accuracy	0.9327	0.934	0.932	0.933	0.9332	0.932	0.9335	0.9328	0.9325	0.9328	0.9313
Unfinetuned Accuracy	0.9327	0.9102	0.9137	0.9004	0.899	0.91	0.91	0.8931	0.908	0.8962	0.2643
Finetuned Var.	0.0012	0.0029	0.0012	0.0010	0.0011	0.0010	0.0013	0.0012	0.0010	0.0013	0.0010
Unfinetuned Var.	0.0012	0.0029	0.0012	0.0032	0.0021	0.0032	0.0024	0.0029	0.0019	0.0033	0.0986



(a) Top-1 Accuracy



(b) Per-class accuracy variance

Fig. 4: Test accuracy on Ablations(stride=3)

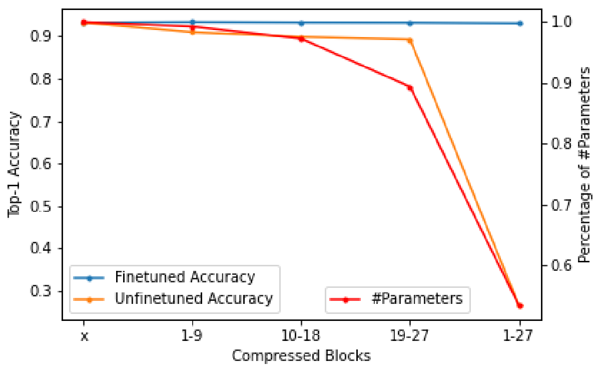
As shown in Fig.2, the first data point x represents the uncompressed, pre-trained ResNet-56 in Fig.2a, while the last one $1-27$ represents the fully compressed model in Fig.2b.

First, the red curve presents a step-like pattern, indicating that the first layer has the least compressed parameters, while the third layer has the most compressed parameters. Notably, the numbers of parameters in 3 layers also follow the pattern. Besides, The parameters compressed by each block in each layer are roughly similar.

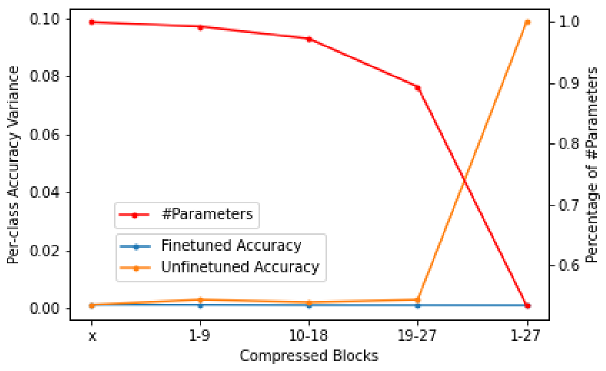
According to Fig.4a and Fig.4b, the fully compressed model's top-1 accuracy and per-class accuracy variance are tremendously improved by fine-tuning. The partially compressed models also benefit from fine-tuning slightly, and one compressed model even achieve higher accuracy than the base model.

Even though the third layer, namely block 19-27, has more pruned parameter, the unfine-tuned model is not necessarily weaker than others. This shows that CC select redundant parameters to prune, at a universal and stable metrics, the unit's importance.

We note that the fully compressed and fine-tuned model's inference ability is slightly penalized. Given that 50% of model size is reduced, the loss of accuracy is quite acceptable.



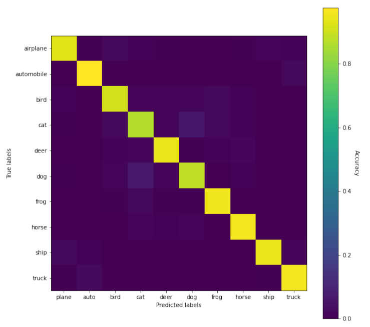
(a) Top-1 Accuracy



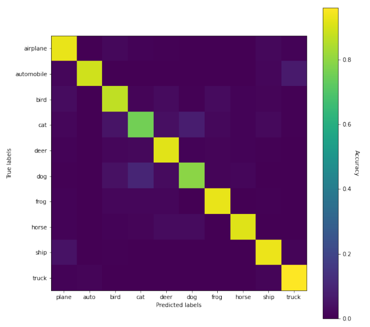
(b) Per-class accuracy variance

Fig. 5: Test accuracy on Ablations(stride=9)

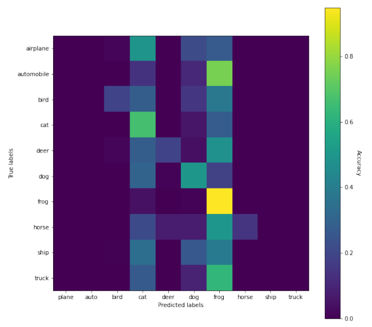
Hoping to observe it in another granularity, we also compress ResNet-56 layer by layer, namely every 9 blocks. Unlike previous case, the top-1 accuracy tends to decline when deeper layers are compressed, according to Fig.5a. Similarly, the fine-tuning recovers accuracy successfully.



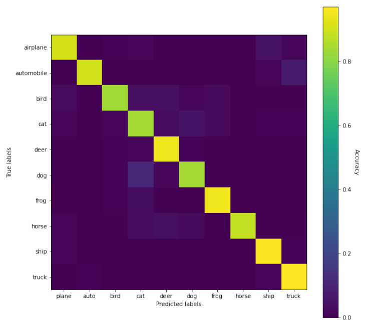
(a) Pre-trained ResNet-56



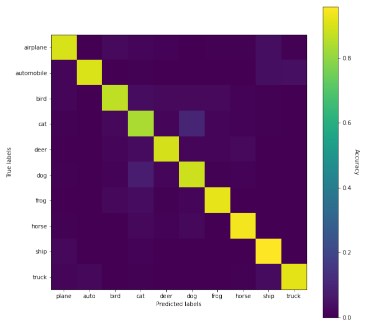
(b) Fully compressed, finetuned



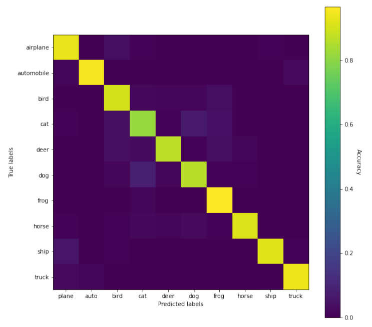
(c) Fully compressed, unfinetuned



(d) Block 4-6 compressed, finetuned



(e) Block 22-24 compressed, finetuned



(f) Block 22-24 compressed, unfinetuned

Fig. 6: Confusion matrices generated from results of prediting 10000 CIFAR-10 images.

Furthermore, some confusion matrices are illustrated in 6. Comparing Fig.6b with Fig.6c, it's obvious to see the tremendous improvement made by fine-tuning on the fully compressed network. When only a portion of network is pruned, the inference ability is not hurt, but the behavior of prediction is slightly altered. For instance, when blocks 22-24 are compressed, it's more accurate to predict horse, dog, compared to that when blocks 4-6 are compressed. Observing the confusion matrices closely, Fig.6b roughly follows the pattern shown in Fig.6a, while other partially compressed models break the prediction behavior. This demonstrates that collaborative compression comprehensively preserves information in the original networks, both the ability to predict correctly and the inclination to predict wrongly.

TABLE II: The most difficult class to predict for ablations

Blocks	**x**	**1-3**	**4-6**	**7-9**	**10-12**	**13-15**	**16-18**	**19-21**	**22-24**	**25-27**	**1-27**
Finetuned	cat	cat	cat	cat	cat	cat	cat	cat	cat	cat	cat
Unfinetuned	cat	dog	dog	cat	dog	cat	cat	cat	cat	plane	auto

Last but not least, we look into the most difficult class to predict for all ablation models. For all fine-tuned models, cat is the most challenging class to predict in CIFAR-10 test set. In contrast, some unfine-tuned models, even with high accuracy, present quite different behaviors from the base model.

V. Conclusion

In this paper, we perform extensive study on a novel unified compression framework, Collaborative Compression, for CNN. We add a layer between GCRO and MSHC to choose blocks to compress, and generate both fine-tuned and unfine-tuned ablations to observe their performance. Moreover, we provide a better implementation to import and infer the compressed model generated by CC pipeline. Our work reveals the CC's effect on partially compressed CNNs and the limited strength of fine-tuning in some cases. CC proves to be a effective method of reducing model size while preserving inference accuracy.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 6 (June 2017), 84–90.
[2] Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.
[3] Redmon, Joseph and Ali Farhadi. “YOLOv3: An Incremental Improvement.” ArXiv abs/1804.02767 (2018): n. pag.
[4] Ren, Shaoqing et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2015): 1137-1149.
[5] Chen, Liang-Chieh et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation.” European Conference on Computer Vision (2018).
[6] Misha Denil, Babak Shakibi, Laurent Dinh, Marc’Aurelio Ranzato, and Nando de Freitas. 2013. Predicting parameters in deep learning. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS’13). Curran Associates Inc., Red Hook, NY, USA, 2148–2156.
[7] Ding, Xiaohan et al. “Global Sparse Momentum SGD for Pruning Very Deep Neural Networks.” Neural Information Processing Systems (2019).
[8] He, Yang et al. “Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration.” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018): 4335-4344.
[9] Ba, Jimmy and Rich Caruana. “Do Deep Nets Really Need to be Deep?” NIPS (2013).
[10] Zhang, X. et al. “Accelerating Very Deep Convolutional Networks for Classification and Detection.” IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (2015): 1943-1955.
[11] Banner, Ron et al. “Post training 4-bit quantization of convolutional networks for rapid-deployment.” Neural Information Processing Systems (2018).
[12] Hinton, Geoffrey E. et al. “Distilling the Knowledge in a Neural Network.” ArXiv abs/1503.02531 (2015): n. pag.
[13] Jiao, Jianbo et al. “Geometry-Aware Distillation for Indoor Semantic Segmentation.” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019): 2864-2873.
[14] Lin, Shaohui et al. “Holistic CNN Compression via Low-Rank Decomposition with Knowledge Transfer.” IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (2019): 2889-2905.
[15] Molchanov, Pavlo et al. “Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning.” ArXiv abs/1611.06440 (2016): n. pag.
[16] Tucker, Ledyard R. ”Implications of factor analysis of three-way matrices for measurement of change.” Problems in measuring change 15.122-137 (1963): 3.
[17] Yu, Xiyu, et al. ”On compressing deep models by low rank and sparse decomposition.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[18] Tung, Frederick, and Greg Mori. ”Clip-q: Deep network compression learning by in-parallel pruning-quantization.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[19] Harshman, Richard A. ”Foundations of the PARAFAC procedure: Models and conditions for an” explanatory” multimodal factor analysis.” (1970): 1-84.
[20] Oseledets, Ivan V. ”Tensor-train decomposition.” SIAM Journal on Scientific Computing 33.5 (2011): 2295-2317.
[21] Garipov, Timur, et al. ”Ultimate tensorization: compressing convolutional and fc layers alike.” arXiv preprint arXiv:1611.03214 (2016).
[22] Hameed, Marawan Gamal Abdel, et al. ”Convolutional neural network compression through generalized Kronecker product decomposition.” Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 36. No. 1. 2022.
[23] Li, Yuchao et al. “Towards Compact CNNs via Collaborative Compression.” 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021): 6434-6443.
[24] He, Kaiming, et al. ”Deep Residual Learning for Image Recognition.” arXiv, 2015, https://doi.org/10.48550/arXiv.1512.03385. Accessed 9 Dec. 2022.
[25] Krizhevsky, Alex. “Learning Multiple Layers of Features from Tiny Images.” 2009.
[26] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. ”Pruning filters for efficient convnets.” International Conference on Learning Representations (ICLR), 2016.
[27] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. ”Rethinking the value of network pruning.” International Conference on Learning Representations (ICLR), 2018.