QUANTIZATION AWARE FACTORIZATION FOR DEEP NEURAL NETWORK COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Tensor approximation of convolutional and fully-connected weights is an effective way to reduce parameters and FLOP in neural networks. Due to memory and power consumption limitations of mobile or embedded devices, the quantization step is usually necessary when pre-trained models are deployed. A conventional post-training quantization approach applied to networks with decomposed weights yields a drop in accuracy. Therefore, our goal is to develop an algorithm that finds tensor approximation directly with quantized factors and thus benefit from both compression techniques while keeping the prediction quality of the model. Namely, we propose to use Alternating Direction Method of Multipliers (ADMM) for approximating a float tensor by a tensor of Canonical Polyadic format (CP), whose factors are close to their quantized versions. This leads to lower approximation error after quantization and smaller quality drop in model predictions while maintaining the same compression rate.

1 INTRODUCTION

Recent years more and more applications based on deep learning algorithms appear on the market, and many of them are developed to work on mobile and edge devices. However, neural networks which are in the core of such algorithms can not usually be used as it is, because they either do not satisfy memory and energy constraints of devices or perform the inference phase not fast enough to satisfy the user needs. Thus, many approaches to reduce and accelerate neural networks have been proposed in the literature.

Conventionally, people distinguish approaches based on pruning, tensor/factorization methods, quantization, knowledge distillation, and architecture search. Recent years some researchers try to combine different techniques to gain benefits from several methods simultaneously. The availability of additional information, such as training data or loss/gradients values, yields additional opportunities to improve performance of compressed models. From this point of view, we distinguish among techniques that do not use additional data, use data but do not perform training, and those that run several training epochs to tune parameters of the compressed model.

In our paper we propose a method that jointly performs factorization and quantization of neural networks. Namely, we replace a linear (convolutional) layer represented with parameters in FLOAT32 format with a decomposed layer, which is constructed as a sequence of linear (convolutional) layers with weights in lower precision format (e.g., INT8/INT6/INT4). By applying our method we benefit from reduction in number of parameters and operations due to factorization and further model size reduction and acceleration due to lower bit representation of weights.

Our main contributions are the following:

- We introduce an ADMM-based algorithm, which approximates a tensor with a CP-factorization, where factors are represented in lower precision format (e.g., INT8/INT6/INT4).
- We propose a neural networks compression method that simultaneously performs factorization and quantization of weight tensors based on introduced tensor approximation technique. As far as we are concerned, this is the first work that introduces weights approximation by CP-decomposition with lower precision factors.

- We show that in terms of compression/accuracy trade-off our approach performs better that conventional tensor approximation followed by quantization of factors.

2 Related Work

There are many papers related to neural networks speed-up and compression. In this section we review the most relevant to our proposed technique.

2.1 Factorization

Various tensor decompositions(Kolda & Bader, 2009; Cichocki et al., 2016) are used in Deep Neural Networks to factorize weights(Panagakis et al., 2021; Kossaifi et al., 2019; Gusak et al., 2019) or activations(Cui et al., 2020; Gusak et al., 2021). This approach allows to store less parameters during training and inference and to fasten the computation by reducing the number of elementary operations required. For linear layers, prior works have utilized Tensor-Train (TT) format (Novikov et al., 2015). For convolutional layers, whose weights are 4d tensors, authors in Lebedev et al. (2015) have used CP(Canonical Polyadic or CANDECOMP/PARAFAC). Authors in (Phan et al., 2020) have also decomposed convolutional layers through CP-decomposition but proposed to flatten spacial dimension and factorize 3d tensors instead of the original 4d. Other tensor approximations techniques like Tucker(Kim et al., 2016) or block term decompositions have been exploited by researchers as well. In our paper we focus on CP decomposition for convolutional layers with kernel spacial dimensions greater than 1 and matrix factorization for other linear layers. We prefer to study these type decompositions to others as they showed promising accuracy/speed-up trade-off in existing research papers.

2.2 Quantization

Due to memory/power consumption limitations of real devices, the quantization step is necessary, when pre-trained models are deployed. A comprehensive overview on Neural Networks quantization was given in Nagel et al. (2021) and Gholami et al. (2022). Quantization might greatly impedes the quality of a neural network prediction. There are different techniques to restore the quality of neural networks after quantization, some of them might require additional data or training. Best accuracy restoration can be obtained by usage of quantization-aware training: fine-tuning the quantized model while restricting model weights to be on the quantization grid. However, it requires dataset and resources to train the model.

2.3 ADMM

Alternating Direction Method of Multipliers or ADMM(Boyd et al., 2010) is an optimization algorithm for convex problems. The advantages of ADMM algorithm are that it can be efficiently parallelized(Shang et al., 2014; Liavas & Sidiropoulos, 2015) and the convergence is guaranteed in convex case(Boyd et al., 2010). However, for non-convex problems the algorithm is not guaranteed to converge or might not converge to a global minimum. Authors in Diamond et al. (2016) proposed using ADMM in conjunction with local improvement methods to find sufficient heuristic solutions.

In Huang et al. (2016) authors have applied ADMM to a task of finding a CP-decomposition that satisfied particular constraints like non-negativity, sparsity, etc.. They called this approach AO-ADMM, since one of the objectives was an Alternating Optimization(in particular, Alternating Least Squares) that is used to find CP factors.

Authors in Yin et al. (2021) have embedded ADMM into training neural network to impose low TT-rank structure on its weights. After training, the weights are decomposed into a TT-format and fine-tuned.

Leng et al. (2018) aimed to apply ADMM to training neural network with weights that lie on a grid of scaled powers of two. The first objective function in their case is model prediction loss, the second - projection of weight tensors on the grid of scaled powers of two. The authors solve a discrete non-convex constraint problem by alternating between optimizing a scaling factor and projecting onto a fixed grid. The method converges, however, the projection even on the unscaled grid is a non-convex problem and is not guaranteed to converge to global minimum.

3 METHODOLOGY

3.1 CONVOLUTIONAL LAYER FACTORIZATION

As was proposed in Phan et al. (2020), we represent convolution layer weights as a 3-way tensor by flattening spacial dimension and decompose it into 3 consecutive convolutional layers. Let $\mathbf{K} \in \mathbb{R}^{T \times S \times D \times D}$ be a kernel tensor, corresponding to convolutional layer with S input channels, T output channels, and $D \times D$ spacial convolution. Let $\overline{\mathbf{K}} \in \mathbb{R}^{T \times S \times D^2}$ denote the K tensor after reshape operation. Using rank-R CP decomposition, one element of $\overline{\mathbf{K}}$ can be represented as follows:

$$\overline{\mathbf{K}}(t, s, dd) \cong \sum_{r=1}^{R} \overline{K}^{dd}(dd, r) K^s(s, r) K^t(t, r) \tag{1}$$

Then, in order to reshape convolutional layer back to original shape, we reshape factor-matrix $\overline{K}^{dd}$ to a tensor $\mathbf{K}^{dd}$ of shape $D \times D \times R$.

$$\mathbf{K}(t, s, j, i) \cong \sum_{r=1}^{R} \mathbf{K}^{dd}(j, i, r) K^s(s, r) K^t(t, r) \tag{2}$$

Therefore, having an input X to the convolutional layer, the output tensor Y is calculated as follows:

$$\mathbf{Y}(t, h', w') = \sum_{h=h'-\delta}^{h'+\delta} \sum_{w=w'-\delta}^{w'+\delta} \sum_{s=1}^{S} \mathbf{K}(t, s, h - h' + \delta, w - w' + \delta) \mathbf{X}(s, h, w) \tag{3}$$

Substituting kernel expression into the last formula, performing rearrangements and grouping summands, we obtain the following four consecutive expressions for the approximate evaluation of the convolution

$$\begin{aligned}
\mathbf{Z}^1(r, h, w) &= \sum_{s=1}^{S} K^s(s, r) \mathbf{X}(s, h, w), \\
\mathbf{Z}^2(r, h', w') &= \sum_{h=h'-\delta}^{h'+\delta} \sum_{w=w'-\delta}^{w'+\delta} \mathbf{K}^{dd}(h - h' + \delta, w - w' + \delta, r) \mathbf{Z}^1(r, h, w), \\
\mathbf{Y}(t, h', w') &= \sum_{r=1}^{R} K^t(t, r) \mathbf{Z}^2(r, h', w'),
\end{aligned}$$

where $\delta = D/2$.

Therefore, we substitute original convolution with a sequence of smaller convolutions: point-wise convolution that reduces the number of input channels from S to R, convolution with the same spacial dimension as the original but with R number of input and output channels, and another point-wise convolution that changes the number of channels from R to T. The last convolution in sequence adds the original bias.

Downsampling skip-connection layers are used in ResNet architecture to bring the output of the previous layer and the residual connection activation to the same spacial dimension and same number of channels. They are implemented by a Convolutional layer with kernel size 1. Such layer is equivalent to a Linear layer with weight matrix of shape $S \times T$ acting on the input with flattened spacial dimension and, thus, can be decomposed into two matrix factors and replaced with a sequence of two convolutional layers both with kernel size 1. Since downsampling layers have a significant impact on model accuracy, in Experiments section (4) we demonstrate results with both factorized and unfactorized downsampling layers.

3.2 Quantization

Quantizing of neural network means transforming its weights and/or activations into low-bit fixed-point representations (e.g. INT8). It saves memory occupied by model parameters, reduces the amount of data transfer, size and energy consumption of the MAC operation (Nagel et al., 2021).

In our work we use signed symmetric min-max per-tensor quantization (Nagel et al., 2021) in which tensors are mapped to their int versions in the following way:

$$x_{\text{int}} = \text{clip}\left(\frac{\lfloor x \rceil}{scale}; -2^{b-1}, 2^{b-1} - 1\right), \tag{4}$$

where $\lfloor x \rceil$ denotes rounding x to the nearest integer, scale is a stepsize of the quantization grid and b is a number of bits in quantization. Real-valued approximation is obtained by multiplying by scale: $x \approx scale * x_{\text{int}}$. For per-tensor MinMax quantization (Nagel et al., 2021) scale is determined from minimum and maximum values of tensor that is being quantized:

$$scale = \frac{\max\{X\} - \min\{X\}}{2^b - 1} \tag{5}$$

3.3 Quantization-aware factorization

Finding a decomposition that produces less error after quantization can be formulated as a constrained tensor factorization problem. A method called AO-ADMM(a hybrid of the Alternating Optimization and the Alternating Direction Method of Multipliers) has been shown(Huang et al., 2016) to successfully solve problems that involve non-negativity, sparsity or simplex constraints. In our work we extend the field of its application by introducing a constraint function that ensures low quantization error of the derived factors and construct a corresponding algorithm.

Searching for a factorization of $n \times m$ matrix X that satisfies quantization constraint (i.e. obtained factors are equal to their quantized versions) means minimizing the following function:

$$\underset{A,B}{\text{minimize}}\ \frac{1}{2}\|X - AB^T\|_F^2 + I_Q(A) + I_Q(B) \tag{6}$$

where A is a matrix of shape $n \times r$, B - of shape $r \times m$ ($r \leq n, m$) and I_Q is an indicator function such that $I_Q(Y) = 0$ when $Y \in Q$ and $I_Q(Y) = +\infty$ if $Y \notin Q$. In case of symmetric min-max per-tensor quantization (Nagel et al., 2021) Q is a space of tensors whose elements belong to a discrete set $\{(-2^{b-1}) * scale, ..., (2^{b-1} - 1) * scale\}$.

We introduce an auxilary variable $\tilde{B}$ and formulate a constrained Alternating Least Squares (ALS) problem. ALS procedure fixes one variable and minimizes the objective function over the other. For fixed factor A the subproblem takes the following form:

$$\begin{aligned} &\underset{B,\tilde{B}}{\text{minimize}}\ \frac{1}{2}\|X - A\tilde{B}\|_F^2 + I_Q(B) \\ &\text{subject to } B = \tilde{B}^T \end{aligned} \tag{7}$$

ADMM method introduces a dual variable U for equality constraint $B = \tilde{B}^T$ and alternates between optimizing each part of the objective function:

$$\begin{aligned} \tilde{B} &\longleftarrow \underset{\tilde{B}}{\text{argmin}}\ \left(\frac{1}{2}\|X - A\tilde{B}\|_F^2 + \frac{\rho}{2}\|B - \tilde{B}^T + U\|_F^2\right) \\ B &\longleftarrow \underset{B}{\text{argmin}}\ \left(I_Q(B) + \frac{\rho}{2}\|B - \tilde{B}^T + U\|_F^2\right) \\ U &= U + B - \tilde{B}^T \end{aligned} \tag{8}$$

Solving the outlined problem for the auxiliary variable $\tilde{B}$ raises the following update:

$$\begin{aligned} &(X^T - \tilde{B}^T A^T)A + \rho(B - \tilde{B}^T + U) = 0 \\ &X^T A + \rho(B + U) = \tilde{B}^T(A^T A + \rho I) \\ &\tilde{B}^T = (X^T A + \rho(B + U))(A^T A + \rho I)^{-1} \\ &\tilde{B} = (A^T A + \rho I)^{-1}(X^T A + \rho(B + U))^T \end{aligned} \tag{9}$$

For efficiency, 9 is solved through Cholesky decomposition of $A^TA + \rho I$.

Solving 8 for factor B gives the minimum distance between $\tilde{B}^T - U$ and set Q:

$$B \leftarrow \operatorname*{argmin}_{B \in Q} \|B - \tilde{B}^T + U\|_F^2 \tag{10}$$

$$B = proj_Q(\tilde{B}^T - U) \tag{11}$$

Algorithm 1 Solve 6 using ADMM

```
    Input: B, U ,K, G
    Output: B, U
 1: $\rho \leftarrow trace(G)/F$
 2: $L \leftarrow Cholesky(G + \rho I)$
 3: repeat
 4:     $\tilde{B} \leftarrow L^{-T}L^{-1}(K + \rho(B + U))^T$
 5:     $B_0 \leftarrow B$
 6:     $B \leftarrow proj_Q(\tilde{B}^T - U)$
 7:     $U \leftarrow U + B - \tilde{B}^T$
 8:     $r \leftarrow \|B - \tilde{B}^T\|_F^2/\|B\|_F^2$
 9:     $s \leftarrow \|B - B_0\|_F^2/\|U\|_F^2$
10: until $r < \epsilon$ and $s < \epsilon$
```

Overall, the ADMM procedure for factor B is defined in Algorithm 1. For $X \approx A\tilde{B}^T$ matrix factorization G is a Gram matrix of the factor that is fixed on current iteration of Alternating Least Squares and K is an original matrix multiplied by a fixed factor (A^TA and X^TA in this case). We adopt expressions for r, s and ρ from Huang et al. (2016). Similarly, the task is solved for the remaining factor through minimizing $||X - \tilde{A}^TB^T||_F^2$ over set Q. The procedure continues alternating between factors until convergence.

For 3-way tensors, the matrix multiplication of two factors in 6 is replaced with the Canonical Polyadic decomposition (CP):

$$\operatorname*{minimize}_{A,B,C} \frac{1}{2}\|\mathbf{X} - \sum_{r=1}^{R} A_{:,r} \otimes B_{:,r} \otimes C_{:,r}\|_F^2 + I_Q(A) + I_Q(B) + I_Q(C) \tag{12}$$

where $\mathbf{X}$ is a tensor of shape $n \times m \times k$, A is a matrix of shape $n \times r$, B - matrix of shape $m \times r$, C - matrix of shape $k \times r$, $\otimes$ denotes the outer product. The constrained ALS subproblem for factor B takes the following form:

$$\begin{aligned} &\operatorname*{minimize}_{B,\tilde{B}} \frac{1}{2}\|X_{(2)} - \tilde{B}^T(C \odot A)^T\|_F^2 + I_Q(B) \\ &\text{subject to } B = \tilde{B}^T \end{aligned} \tag{13}$$

where $X_{(1)}$ is a matrix unfolding of tensor X by mode 1, $\odot$ denotes Khatri-Rao product. The ADMM procedure is the same as in Algorithm 1, with $G = A^TA * C^TC$ and $K = X_{(2)}(C \odot A)$ (Smith et al., 2017). Optimization with respect to factors A and C is done in the same manner (Smith et al., 2017). Alternating between factors proceeds until convergence.

The set Q is non-convex, thus, it is not guaranteed that solution converges to global or even local minimum. Moreover, as shown in Figure 1, convergence depends on initialization1. Authors in Diamond et al. (2016) proposed to use Multistart, Polishing and Neighbor search to improve solution. However, in our case, search among neighbors(even sampling search) of current iteration factor would require number of computations comparable to the number of parameters in a layer. Polishing on Q reduces to simple recomputation of $\tilde{B}_{k+1}$ with B_{k+1} instead of B_k since the only convex restriction(Diamond et al., 2016) that can be chosen in this case is the trivial one. Multistart, however, proved to be a plausible technique to obtain a better solution (Figure 1).

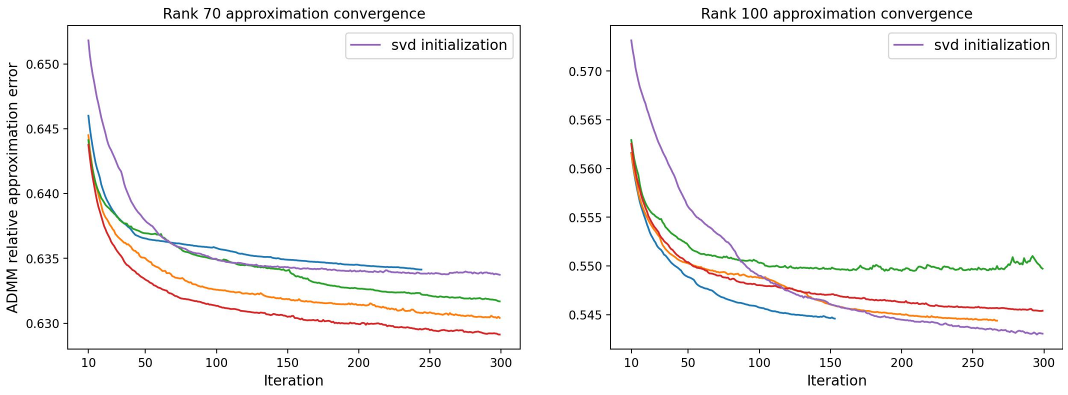

Figure 1: Convergence of ADMM algorithm for layer3.0.conv1 of ResNet18 model. Different colours indicate different random initialization seeds. Initialization using svd decomposition is labeled explicitly.

4 EXPERIMENTS

We use pretrained ResNet18 from PyTorch model zoo and ImageNet dataset to evaluate our method. For each convolutional layer of the network we compute two different procedures: (i) CP-factorization with parafac followed up with quantization; (ii) ADMM algorithm that outputs already quantized factors. Obtained factors are then loaded into a sequence of two (for downsample layers) or three convolutional layers that replace the original layer (3.1). The fully-connected layer in the tail of the model is decomposed into a sequence of two linear layers with a matrix factorization.

Factorization ranks To choose a rank for each layer factorization, we set up a parameter reduction rate (i.e. how many times the number of the obtained layer parameters is lower than the original) and define the rank as $N/(n+m)/rate$ for a fully-connected and downsample layers and $N/(n+m+k)/rate$ for other convolutional layers. N is the number of parameters in the original layer, n, m and k - dimensions of reshaped convolution weights (3.1), $rate$ denotes the parameter reduction rate. In our evaluations we experimented with which layers to leave unfactorized for better model performance. Thus, the factorization reduction rate is not always precisely the parameter reduction rate of the whole model (Table 2).

Calibration Since ResNet18 model has a lot of BatchNorm layers, training only BatchNorm parameters(the procedure called ”calibration”) can substantially boost the model prediction quality. We perform a few iterations of training with all parameters frozen except for BatchNorm Layers. In all our experiments we calibrate on 100 batches of size 512 from ImageNet training set with a single fixed random seed for a data-sampler.

Parafac Parafac algorithm from Tensorly[1] package is used to factorize convolutions and fully-connected layer. The obtained factors are then quantized and placed as layers of a reduced model. The results are used as a baseline for comparison to ADMM (Table 2).

First, we experimented with individual layers of ResNet18 model. Figure 2 shows neural network accuracy after factorization, after per-tensor quantization of factors (admm outputs already quantized factors), and calibration of the transformed model. With parafac factorization higher rank leads to better model predictions. Moreover, the accuracy is higher than that of a model factorized with admm, since parafac only had one optimization objective - Least Squares without quantization constraints. However, when obtained factors are quantized, the accuracy begins to decrease as the rank increases. Whereas with our method the accuracy is consistent with rank. We theorize that quantizing parafac factors performs poorly for higher ranks because the number of elements in tensor is bigger and, thus, the scale of quantization (Equation 5) is wider. After calibration admm

[1]https://github.com/tensorly

Table 1: Accuracy of ResNet18 model with quantized (INT8, INT6, INT4) weights and FP32 activations. Average ImageNet validation accuracy (%) over 3 runs of factorization algorithm with different random initialization seeds.

Bit-width	**W8**			**W6**			**W4**		
Param Reduction Rate	**2x**	**4x**	**8x**	**2x**	**4x**	**8x**	**2x**	**4x**	**8x**
ADMM	52.67	4.01	0.11	42.90	2.18	0.12	2.58	0.16	0.10
ADMM Calibrated	65.48	39.07	6.01	63.94	36.32	5.51	52.55	15.27	1.36

Table 2: Accuracy of ResNet18 model with quantized (INT8, INT6, INT4) weights and FP32 activations. Average ImageNet validation accuracy (%) over 3 runs of factorization algorithm with different random initialization seeds. Factorization is not performed on downsample layers and first convolutional layer, their weights are only quantized.

Bit-width	**W8**			**W6**			**W4**		
Param Reduction Rate	**1.97x**	**3.83x**	**7.23x**	**1.97x**	**3.83x**	**7.23x**	**1.97x**	**3.83x**	**7.23x**
ADMM	64.01	25.63	0.40	59.73	18.75	0.34	12.31	0.88	0.18
ADMM Calibrated	66.55	52.31	19.87	65.62	49.41	17.13	56.05	31.45	5.72
Parafac Quantized	0.10	0.09	0.10	0.10	0.10	0.10	0.10	0.10	0.10
Parafac Quantized Calibrated	0.14	0.17	0.50	0.10	0.11	0.10	0.10	0.10	0.11

performes almost on the level of a baseline, whereas quantized parafac loses up to 15% of model accuracy for some reduction rates.

Our second experiment was quantizing one of the four ResNet18 modules, that contains four convolutional layers and one downasample layer for skip-connection. Figure 3 shows the same trend as Figure 2 with admm performing better then the quantized parafac approach. In case of modules, the error induced by parafac factors quantization sums up and brings model accuracy to the level of random guessing amoung 1000 of ImageNet classes. Calibration helps, but the resulting accuracy of quantized parafac is below 50%.

Finally, we transform the whole ResNet18 model. Table 1 shows results for factorization-quantization with ADMM before and after calibration. However, we discovered that first convolution and downsample layers have a significant impact on model accuracy, while being relatively light-weighted layers. Thus, we can achieve almost the same parameter compression rate while obtaining higher model accuracy both before and after calibration if we factorize all layers except for first convolution and downsample layers. Nevertheless, weights of these layers are quantized so that all weights of the model have the same bit-width. Table 2 shows results both for admm and parafac in this experiment setting. It is notable, that quantized parafac approach doesn't regain accuracy even after calibration step.

To explore how reduction rate and low-precision format correspond to memory, we plot a model size-accuracy dependency (Figure 4). It is clear from the graph, for example, that factorizing with 2x reduction rate and INT4 quantization constraint is more beneficial in terms of memory than INT8 format with 2x reduction rate factorization.

5 CONCLUSION

In the current paper we proposed a technique for compressing neural networks that performs joint factorization and quantization. We introduced ADMM-based approximation algorithm to replace weights with float32 elements with their factorized representations, whose factors contain elements of lower precision (int8/int6/int4).

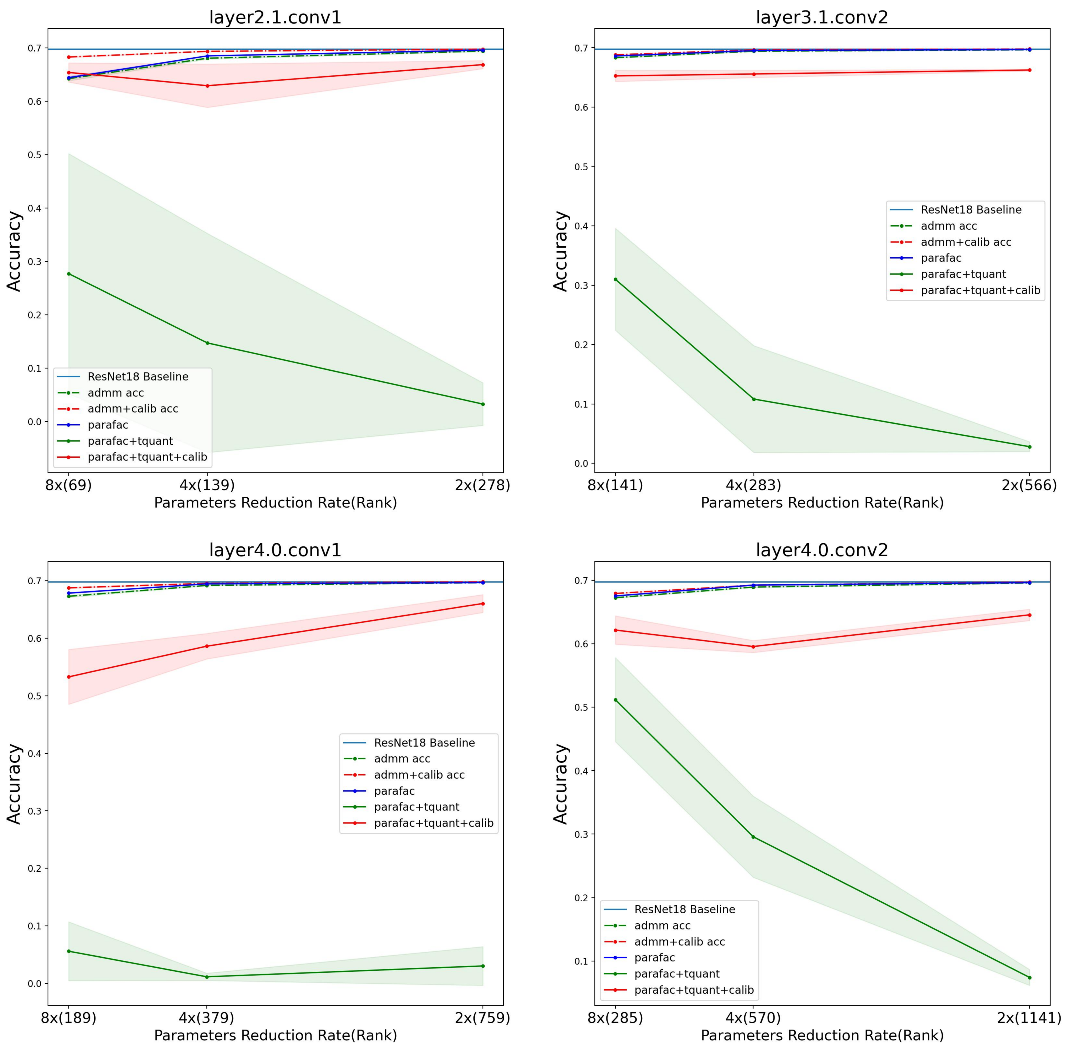

Figure 2: Average ImageNet validation accuracy (%) over 3 runs of factorization algorithm with different initialization random seeds. The factorization and 8bit quantization is performed on a single layer of a network. (Admm algorithm outputs already quantized factors; Parafac factors are quantized.) Then for each method the model is calibrated on 100 batches of size 512.

Our method allows to benefit from both factorization and quantization techniques, resulting in decrease of model size and acceleration in model inference. Our experiments have shown significant advantage of our proposed method in comparison over a naive approach of consequent factorization and quantization of weights.

Nevertheless, there is a lot of opportunities to improve our method by incorporating more sophisticated tensor and quantization techniques.

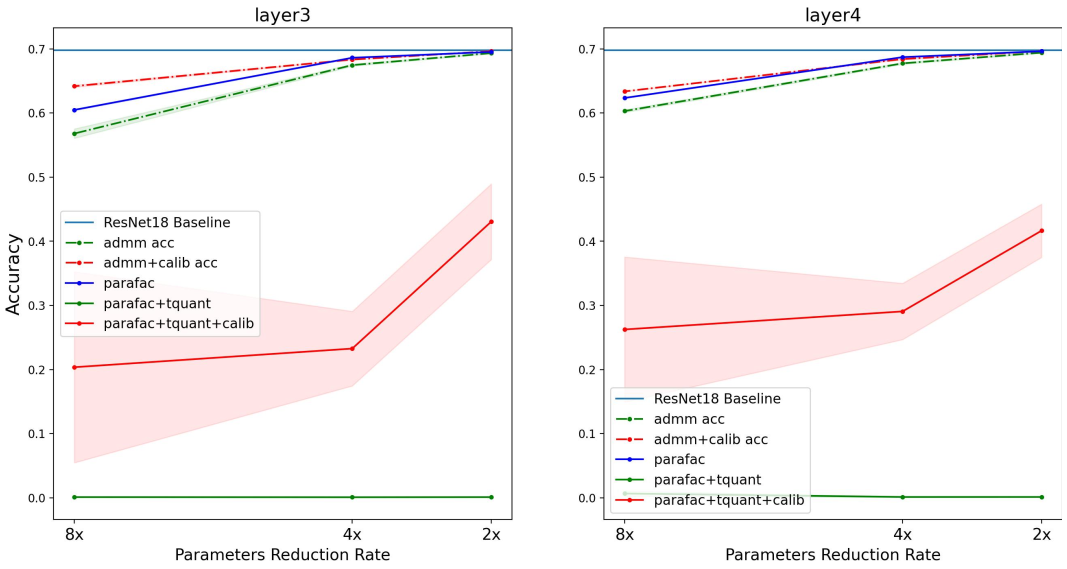

Figure 3: Average ImageNet validation accuracy (%) over 3 runs of factorization algorithm with different initialization random seeds. The factorization and 8bit quantization is performed on a module of a network. (Admm algorithm outputs already quantized factors; Parafac factors are quantized.) Then for each method the model is calibrated on 100 batches of size 512.

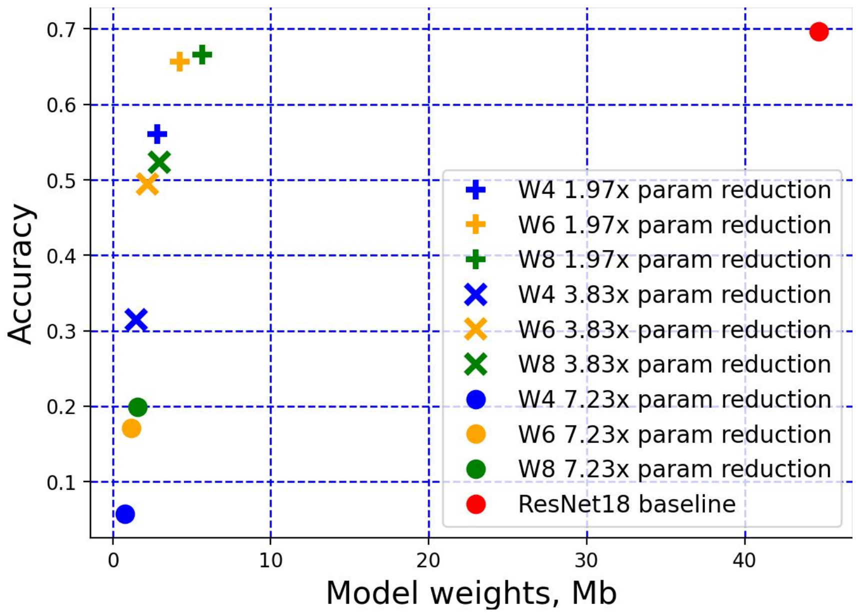

Figure 4: Average ImageNet validation accuracy (%) over 3 runs of factorization algorithm with different initialization random seeds. X axis represents models weights size in megabytes. Red dot indicates original ResNet model weights size and baseline accuracy.

References

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers, 2010. ISSN 19358237.

Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh Huy Phan, Qibin Zhao, and Danilo P. Mandic. Tensor networks for dimensionality reduction and large-scale optimization part 1 low-rank tensor decompositions. *Foundations and Trends in Machine Learning*, 9, 2016. ISSN 19358245. doi: 10.1561/2200000059.

Chunfeng Cui, Kaiqi Zhang, Talgat Daulbaev, Julia Gusak, Ivan Oseledets, and Zheng Zhang. Active Subspace of Neural Networks: Structural Analysis and Universal Attacks. *SIAM Journal on Mathematics of Data Science*, 2(4):1096–1122, 2020. paper.

Steven Diamond, Reza Takapoui, and Stephen Boyd. A general system for heuristic solution of convex problems over nonconvex sets. 9 2016.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference, 2022.

Julia Gusak, Maksym Kholiavchenko, Evgeny Ponomarev, Larisa Markeeva, Philip Blagoveschensky, Andrzej Cichocki, and Ivan Oseledets. Automated MUlti-Stage Compression of Neural Networks. In ***Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)***, pp. 0–0, 2019. paper.

Julia Gusak, Talgat Daulbaev, Evgeny Ponomarev, Andrzej Cichocki, and Ivan Oseledets. Reduced-Order Modeling of Deep Neural Networks. *Computational Mathematics and Mathematical Physics Journal*, 2021. paper.

Kejun Huang, Nicholas D. Sidiropoulos, and Athanasios P. Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. volume 64, 2016. doi: 10.1109/TSP.2016.2576427.

Y. D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. 2016.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications, 2009. ISSN 00361445.

Jean Kossaifi, Adrian Bulat, Georgios Tzimiropoulos, and Maja Pantic. T-net: Parametrizing fully convolutional nets with a single high-order tensor. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7822–7831, 2019.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. 2015.

Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. 2018.

Athanasios P. Liavas and Nicholas D. Sidiropoulos. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 63, 2015. ISSN 1053587X. doi: 10.1109/TSP.2015.2454476.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *ArXiv*, abs/2106.08295, 2021.

Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov. Tensorizing neural networks. volume 2015-January, 2015.

Yannis Panagakis, Jean Kossaifi, Grigorios G Chrysos, James Oldfield, Mihalis A Nicolaou, Anima Anandkumar, and Stefanos Zafeiriou. Tensor methods in computer vision and deep learning. *Proceedings of the IEEE*, 109(5):863–890, 2021.

Anh Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavský, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. volume 12374 LNCS, 2020. doi: 10.1007/978-3-030-58526-6_31.

Fanhua Shang, Yuanyuan Liu, and James Cheng. Generalized higher-order tensor decomposition via parallel admm. volume 2, 2014. doi: 10.1609/aaai.v28i1.8913.

Shaden Smith, Alec Beri, and George Karypis. Constrained tensor factorization with accelerated ao-admm. 2017. doi: 10.1109/ICPP.2017.20.

Miao Yin, Yang Sui, Siyu Liao, and Bo Yuan. Towards efficient tensor decomposition-based dnn model compression with optimization framework. 2021. doi: 10.1109/CVPR46437.2021.01053.