# A LOW RANK CONVOLUTION OPERATION GENERATING NUMEROUS FEATURE MAPS

**Reza Mohajernoei**
Department of Computer Science
Tarbiat Modares University
m.reza@modares.ac.ir

**Mansoor Rezghi***
Department of Computer Science
Tarbiat Modares University
rezghi@modares.ac.ir

**ABSTRACT**

Convolutional Neural Network is an essential tool in machine learning used in many applications and computers on various scales. For instance, one can see their role in different applications such as Machine Vision, Natural Language Processing, and financial data. Recently, Convolutional Neural Networks have shown satisfying and even fantastic performance in different applications. This ability of Neural Networks sometimes surpasses human categorization or recognition. Despite their advantages, Neural Networks involve some challenges as well. Numerous parameters consumption requires powerful calculation tools to solve the problems. Some solutions have been provided to overcome this issue. Different methods exist to improve Convolutional Neural networks. One of these approaches is based on tensor decompositional methods. Because The parameters (Kernel-Filter) between two layers of Convolutional Neural Network is always a four-dimensional tensor. In tensor methods, the parameter numbers are usually sought to be reduced using tensor decompositions. Therefore different approximations of convolution operations could be found by these methods. Hence, the concept of tensor, as a critical solution, got into Neural Networks literature. It should be noted that although recently have been tensor appearing in a lot of articles and research, Neural Networks had been based on a specialized condition of tensor application because tensors were a generalization of weights vectors and in-layer connections matrices. Consequently, tensors have shown a significant usage to replace the lower dimensions, representations of the layer-wise connections approach towards utilizing this concept have been extending. In this research, we tend to announce a mechanism inspired by some studies in this field and combining them with algebraic concepts to improve the Convolutional Neural Networks, which consume fewer resources with more negligible effect on the accuracy, and in the end, the experimental results will be reported.

***Keywords*** Deep Neural Networks · Tensor product · Contractive dot · Convolutional operation · Convolutional Neural Networks

## 0.1 Introduction

Convolutional Neural Networks have shown their advantages over the other methods in the last decade. Initially announced before the millennium [4] and evolved during the recent years, Convolutional Neural Network has a wide range of applications such as computer vision [6, 5, 7] , natural language processing [9, 8] , and biomedical [10, 11] or financial analysis [13, 12] . Despite their prominent ubiquity, Convolutional Networks suffer from several difficulties, such as the need for the mass amount of data to be appropriately trained, numerous parameters to build a network which is also called memory complexity; learning these parameters in every deep learning solution is a complex optimization task [26] . Hence, the convolutional neural network is a time-consuming solution, also. A higher number of parameters expands the search space, and consequently, the accuracy grows. In line with more advanced hardware and new generations of processing units [25] , the network designers and researchers announced new architectures in which millions of parameters were utilized to extract better results with current facilities [20, 21, 19, 22] . Although

*https://www.modares.ac.ir/ rezghi

this equipment paved the way for many high-quality and precise results in different fields, advances, in turn, caused inattention to the essence of thriftiness about the resource.

Although they are known as a real solution for most of the machine learning tasks, and they are still under development, the curse of resource consumption has remained an issue. Besides introducing novel architectures requiring less floating-point operations and memory, many research pieces have focused on hashing technique [28] , pruning [29] and fine-tuning [31] CNN. However, few have tried changing the traditional convolution operation mechanism, which affects the overall memory and computation cost by itself [32, 33, 34, 35, 36]. Across a CNN, from input towards output, we encounter tensors with different dimensions and sizes [37] . Hence, multilinear operations suggest solutions to overcome this challenge and have turned into a fundamental remedy for such a problem [32, 38] . In this work, while applying a multilinear method instead of the old-fashioned convolution with less resource usage, we generate even more feature maps for the network's successive layers. Theoretical and experimental results show that the new operation outperforms the traditional ones in parameter consumption and computation complexity.

Techniques such as pruning and fine-tuning the networks have turned out to be helpful but still redundant[30]. One of the trickiest ways to reduce both the time and memory complexity of a network is to change the convolution's essential core functions. Some works suggested changing the mechanism of convolution, whether by changing the filter[32, 33, 34, 35, 36, 26] or even the order and stages of convolution in a layer[24, 20]. Some tried to change the way fully connected perform using linear algebraic decompositions[38].

In this paper, we will not propose an architecture involving multiple convolutions one after another[19, 21, 20, 22], nor are we going to break the convolution into several stages[24]. Instead, we get major inspiration from changing the core functionality of convolution thanks to tensorial and linear algebraic concepts[32]. We will at first discuss similar ideas on designing novel convolutions, those which have shown their advantages in practice. Then the experiments demonstrate why this method is a suitable alternative for the standard convolution. Finally, a conclusion will sum the discussion in a nutshell.

## 0.2 Related Work

Besides a means of better illustrating the competence of Deep Neural Networks [1] Tensor methods have been used in different machine learning applications, such as learning latent variable models [2] and developing recommender systems [3], Several research applied tensor learning [14] and tensor decomposition [15, 16] to Deep Neural Networks. Furthermore, they have been used in multi-task learning [17], and sharing residual units [18] as well.

Recently, Convolutional neural networks have drawn the spotlight on themselves though they use a high amount of unnecessary memory and calculations [38]. In some architectural solutions, such as Resnets [19], changing the order of convolutional layers along a network has been a method to increase the accuracy of the convolutional neural networks in different applications. The expense of memory and time has led researchers to design well-known cost-efficient networks [19, 20, 21, 22]. The efficiency results from different aspects. Some efforts have shown their advantages successfully, such as TRL networks which concentrated on the final layers, made use of tensor decompositions to lessen the cost of convolutional neural networks[38]. Another outstanding idea helping to improve Convolutional Neural Networks is the bottleneck layer deployed by ResNets architecture which has been undeniably successful in image classification, and object detection[19]. Another critical example is MobileNets comprising several layers in each of which separates the convolution operation into two stages[20] such that At the first stage, depthwise convolution is applied on the input, then its output is fed to the second stage performing one by one convolution which is called pointwise [24]. This idea has turned out to be reasonable in both performances and resource consumption[20]. Considering the research on improving Neural Networks in terms of the core convolution function[32], it is evident that most researchers have spent significant effort and time manipulating the existing structures and known architectures, modifying the order of layer placements, normalizations, pooling layers, and even activation functions. With the rise of AlexNets, Convolutional Neural network has been revolutionized; the excellence of Graphical Processing Units has neutralized even the dramatic increase in the number of parameters [25] . On the other hand, only recently has potential linear algebra power shown its role in reordering the convolution operation [26, 32, 38]. An exquisite idea to reduce the cost of a Convolutional Neural Network is a low-rank approximation of the convolution operation. [32] A good example of low rank approximation can be shown by the formula that follows:

$$\tilde{W}_n^c \simeq \sum_{k=1}^{K} V_k^c (H_n^k)^T$$

In this formula, subscripts and superscripts stand for the channel index and the corresponding kernel and $K$ is the hyper parameter which works as a rank controller of the approximation matrix. Here, $\tilde{W}_n^c$ just represents a two-dimensional kernel of the $n$th filter that applies on $c$th channel. Also, $V_i^c \in R^{R\times w_1\times 1\times C}$ and $H_n^i \in R^{N\times 1\times w_2\times R}$. With this filter and applied on the input $X$, convolution operation applies as follows:

$$\tilde{W}_n * X = \sum_{c=1}^{C}\sum_{i=1}^{R} H_n^i (V_i^c)^T * X^c = \sum_{i=1}^{R} H_n^i * \left(\sum_{c=1}^{C} V_i^c * X^c\right)$$

in which $X \in R^{X\times Y\times C}$ and $\tilde{W}_n \in R^{w_1\times w_2\times C}$ .

This method aims at convolving with two types of parameter sharing. First is sharing all right singular vectors $H_n^k$ along all the input channels $C$ and in the $n$th filter, while the second ensures parameter sharing via left singular vectors $V_k^c$ along all of the $N$ filters [32] .

## 0.3 Proposed Method

Before getting over the main description of the main idea of the proposed method, we need to introduce some notations using which we will describe the method.

subsection prerequisites

**Rank-1 tensor [23]**

An N-way tensor $X \in R^{I_1\times I_2\times\cdots\times I_N}$ is rank one if it can be written as the outer product of $N$ vectors, i.e.,

$$X = a^{(1)} \bullet a^{(2)} \bullet \cdots \bullet a^{(N)}.$$

The symbol " $\bullet$ " represents the vector outer product. This means that each element of the tensor is the product of the corresponding vector elements:

$$x_{i_1 i_2 \ldots i_N} = a_{i_1}^{(1)} a_{i_2}^{(2)} \ldots a_{i_N}^{(N)} \quad for\ \ all \quad 1 \le i_n \le I_n.$$

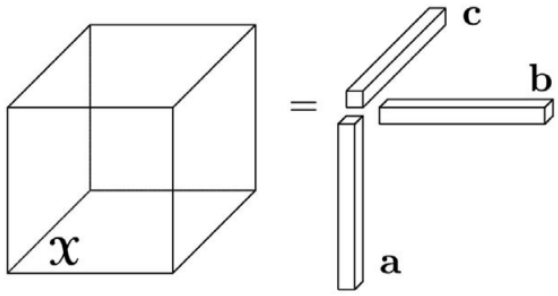


Figure 1: $Rank-one\ third-order\ tensor, X = a \bullet b \bullet c.\ The (i,j,k)\ element\ of\ X\ is\ given\ by\ x_{ijk} = a_i b_j c_k.$

Figure 0.3 illustrates $X = a \bullet b \bullet c$, a third-order rank-one tensor.

**Fibers [23]**

Fibers are the higher-order analogue of matrix rows and columns. A fiber is defined by fixing every index but one. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. Third-order tensors have the column, row, and tube fibers, denoted by $x_{:jk}$, $x_{i:k}$, and $x_{ij:}$, respectively; see **??** When extracted from the tensor, fibers are always assumed to be oriented as column vectors.

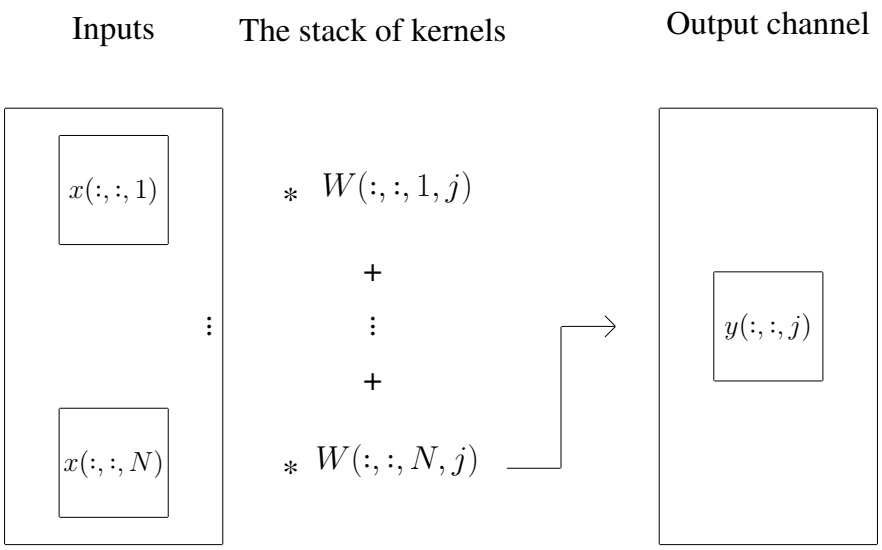


Figure 2: The $j$th channel is generated by summation of product of two tensors in the manner shown above in a standard convolution operation

**Tensor-matrix multiplication[27]**

Let us assume $X \in R^{J\times K\times L}$ and $W \in R^{M\times J}$. The Tensor-matrix multiplication is defined as follows:

$$y_{mkl} = \sum_{j=1}^{J} W_{mj} xjkl.$$

Where $Y \in R^{M\times K\times L}$This means that all column vectors (mode-1 fibers) in the order-3 tensor are multiplied by the matrix W . Similarly, mode-2 multiplication by a matrix X means that all row vectors (mode-2 fibers) are multiplied by the matrix X. Mode-3 multiplication is analogous. In the case when tensor-matrix multiplication is performed in all modes in the same formula, we omit the subscripts and write:

$$(A, B, C) \bullet X$$

Where the mode of each multiplication is understood from the order in which the matrices are given.

### 0.3.1 Thrifty output feature map generator as a Convolutional Layer

We can develop tensor methods to reduce the problem size being inspired by previous papers[26, 20]. We can present a new generator formula like:

$$Y(:,:,j) = \sum_{l=1}^{N} W(:,:,l,j) * X(:,:,l)$$

where the bias addition expression has been omitted for simplicity and $*$ represents the convolution operation and has been defined, before. To simplify the formula, we eliminate the index $j$ from it:

$$Y = \sum_{l=1}^{N} W(:,:,l) * X(:,:,l)$$

In other words, as in figure 0.3.1 the $(i, j)$th element of the output matrice $Y$ obtains by the formula:

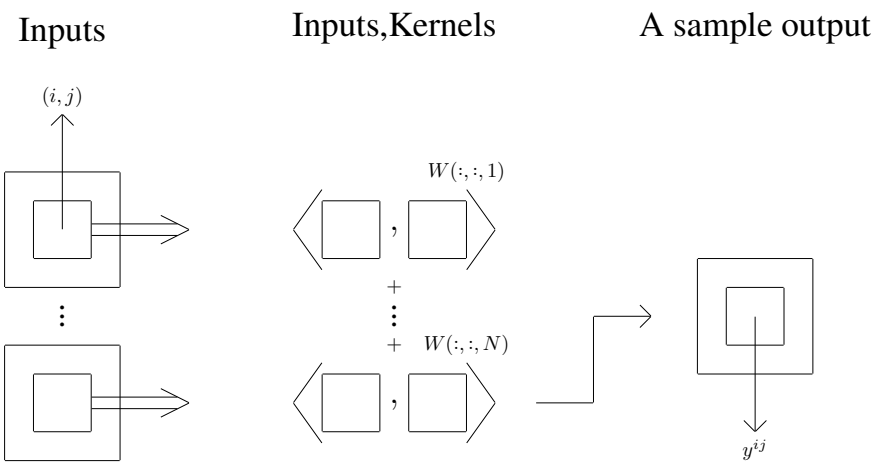


Figure 3: The final matrice is output by reshaping the resulting vector

$$y^{i,j} = \sum_{l=1}^{N} \left\langle X_l^{i,j}, W(:,:,l) \right\rangle \tag{1}$$

In which, the spacial size of $X_l^{i,j}$th patch is equal to the $l$ th channel of the filter. Supposing $x^{i,j} \in R^{W_1 \times W_2 \times N}$ and $x^{i,j}(:,:,l) = X_l^{i,j}$ then :

$$y^{i,j} = \sum_{l=1}^{N} \left\langle X_l^{i,j}, W(:,:,l) \right\rangle = \left\langle x^{i,j}, W \right\rangle \tag{2}$$

In which: $\left\langle , \right\rangle$ Represents inner product of two tensors. In [26], assuming $W = a \bullet b \bullet c$ Means outer product of 3 vectors and therefore:

$$y^{i,j} = \left\langle x^{i,j}, a \bullet b \bullet c \right\rangle \tag{3}$$

Based on the inner and matrix product property between a tensor and a matrix(vector,) [27]:

$$y^{i,j} = (a,b,c)_{1:3} \cdot x^{i,j} \tag{4}$$

In fact, [26] Assumes that tensor shaped filter

$W \in R^{W_1 \times W_2 \times N}$ is of rank 1. Generally if we have supposed $W = \sum_{r=1}^{R} a_r \bullet b_r \bullet c_r$ then we would have 4: as the form, bellow:

$$y^{i,j} = \sum_{r=1}^{R} (a_r \bullet b_r \bullet c_r) \cdot x^{i,j} \tag{5}$$

Its advantage is in network parameters reduction. meaning in $W \in R^{W_1 \times W_2 \times N}$ , we have $W_1 W_2 N$ parameters, while in 4 , we have $W_1 + W_2 + N$ parameters. In5 this is equal to $R(W_1 + W_2 + N)$ applying much thriftiness in parameters consumption. In this method, if the number of output feature maps is $M$, we will have $M$ times equation 1 In fact, it is being supposed that :

$$W(:,:,:,i) = \sum_{r=1}^{R} a_r^i \bullet b_r^i \bullet c_r^i \tag{6}$$

And eventually, the consumed parameters are equal to $MR(W_1 + W_2 + N)$

## 0.4 Experiments

Our focus was to reduce the redundancy in Convolutional Neural networks by modifying the primary operation in this paper that allows us to use the same architecture for either weak and robust systems, whether a smartphone or a corporational server computer. Although our proposed method might suffer a tinge of accuracy loss, it can still benefit more straightforward tasks in simpler devices. We used a computer with Ubuntu 18.04 operating system for this paper, and Tensorflow 1.14 and Cuda 9.0 installed on it, equipped with 32 GB of RAM.

Since the Multiple linear methods first inspired us to develop a method that saves more memory and calculus operations, besides the standard convolution, we compared the proposed method's performance to it. As far as our equipment, we took the deepest possible network, a 56 layer Resnets architecture. To achieve high accuracy results in many applications, Resnets is maybe the most valuable and omnipresent architecture that uses the standard convolution operation as its core feature extractor. With this background and having the official Resnets edition 1.11 official source code, we could replace the core convolution operation utilized officially with the proposed one, resulting in really parsimonious memory and calculation consumption with even higher iteration, yet lower floating-point calculations. Therefore, we have not even changed the network architecture, and the standard architecture remains intact with its convolution, pooling, and even its bottle-neck layers. We did, instead, change the code module in which the primary convolution was called and called our proposed convolution function to see if it could result in better or possibly nearly equal accuracy.

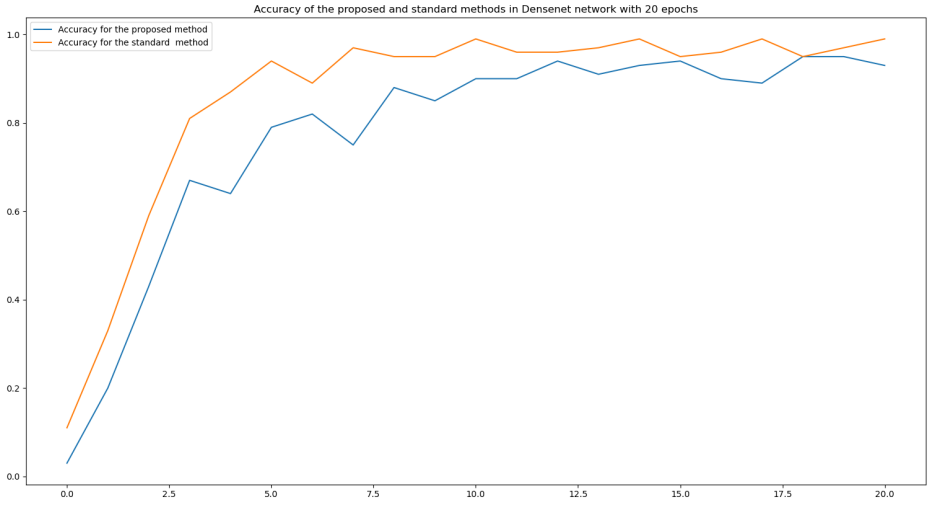


Figure 4: Having the Densenet network trained in 20 epochs, with MNIST dataset with comparable accuracy.

In the process of the whole experiment, we did not use the rank hyperparameter R more than one, and the activation function for all the experiments was ReLU. For the experiments of Densenet network [39] with the MNIST dataset containing 60000 of 28 by 28-pixel images where the proportion of train to test was 80 to 20, we only used convolutional kernel with three and stride of 1 had only four filters in each convolutional layer. The experiments IST via Densenet network are shown in diagrams 0.4 and 0.4. For all the experiments alongside bias vectors. Also, we utilized SGD to optimized the networks. On the other hand, All we did to the standard Resnets network was calling the proposed convolution and rerunning the network to compare with the leading network and table 1 were resulted while calling the program with it is default parameters except for the maximum train steps and channels order set to channels-last attribute. We experimented with all the networks trained from scratch, and the trainable variables were initialized with random values.

Having these settings,datasets and topologies we performed 10 experiments on both proposed or standard resnets lead to results reported in table 1 mentioning the accuracy and loss for each experiment as well as their corresponding average accuracy and loss and finally, the number of floating point calculations has been reported in the last row.

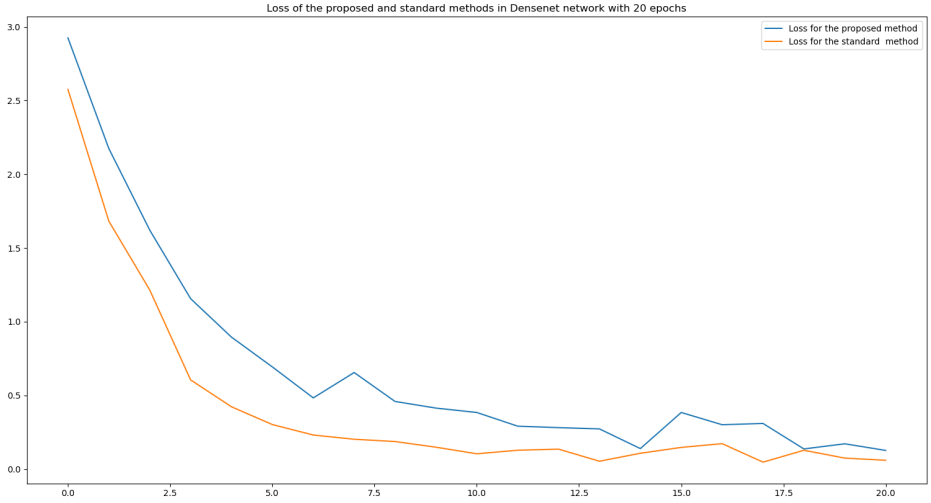


Figure 5: Having the Densenet network trained in 20 epochs, with MNIST dataset we can reach promising loss.

Table 1 mentions that the the Resnets-56 with proposed method can reach Resnets-56 with the standard convolution with 20000 iterations somwhere between 40000 and 45000 iterations. Table 2 shows the number of parameters used in Resnets-26 and Resnets-56 for the standard convolution, the multilinear and the proposed method, respectively. Which means the more the networks expands in terms of depth, the more proposed method shows its advantage over the other two methods. In Resnets-26, the proposed convolution consumes about one third the parameters the standard convolution uses. While in shallower networks it can be slightly less thrifty in parameter usage. The reason is that while we have deeper ResNets networks, we deploy much more parameters and obviously, we expect to use dramatically less parameters in the network. In table 3 we have reported the count of floating-point calculations during each iteration for ResNets-26 and ResNets-56 with the standard, multinlinear and the proposed convolution, respectively. It is clear that the multilinear and the proposed method have reduced the number of calculations of each iteration, drastically.

Table 1: The results of experiments with the standard Resnets 26 in 20000 iterations and Resnets 56 with proposed convolution with 40000 and 45000 iterations with Cifar-10 data, respectively. Also in the last row, sum of the calculations during the whole training phase has been reported.

| Row | 20000 it Standard ResNets-26 Accuracy | 20000 it Standard ResNets-26 Loss | 40000 it Proposed ResNets-56 Accuracy | 40000 it Proposed ResNets-56 Loss | 45000 it Proposed ResNets-56 Accuracy | 45000 it Proposed ResNets-56 Loss |
|---|---|---|---|---|---|---|
| 1 | 76.99 | 0.99 | 71.87 | 0.93 | 78.12 | 0.73 |
| 2 | 74.75 | 0.98 | 75.00 | 0.87 | 78.90 | 0.70 |
| 3 | 70.84 | 1.21 | 69.53 | 0.90 | 76.56 | 0.93 |
| 4 | 75.67 | 1.01 | 73.43 | 0.80 | 78.90 | 0.70 |
| 5 | 75.60 | 1.01 | 78.90 | 0.84 | 78.90 | 0.74 |
| 6 | 82.35 | 0.81 | 76.56 | 0.85 | 78.90 | 0.80 |
| 7 | 86.50 | 0.83 | 80.46 | 0.79 | 78.12 | 0.71 |
| 8 | 82.06 | 0.82 | 73.43 | 0.88 | 76.56 | 0.80 |
| 9 | 71.18 | 1.19 | 75.25 | 0.84 | 78.12 | 0.76 |
| 10 | 74.98 | 1.07 | 78.12 | 0.77 | 75.00 | 0.85 |
| Avg | 76.49 | 0.99 | 75.00 | 0.84 | 77.80 | 0.77 |
| Calculations | $1.31864168 \times 10^{12}$ | | $1.32904 \times 10^{9}$ | | $1.49517 \times 10^{9}$ | |

Table 2: The number of parameters used in ResNets56 and ResNets26 for the standard convolution and MLconv and the proposed method .

| Network Version | Standard Convolution | MLconv | Proposed Method |
|---|---|---|---|
| ResNets-26 | 367280 | 34800 | 29046 |
| ResNets-56 | 851120 | 46960 | 33226 |

Table 3: The number of calculations in ResNets-56 and ResNets-26 for the standard convolution and MLconv and the proposed method .

| Network Version | Standard Convolution | MLconv | Proposed Method |
|---|---|---|---|
| ResNets-26 | 65932084 | 26905268 | 26716996 |
| ResNets-56 | 137284404 | 44950708 | 44658116 |

## 0.5 Conclusions

This paper proposed a new convolution operation and output feature map generation mechanism, which dramatically reduces parameter consumption and calculations. We can flexibly adjust the rank hyperparameter for each convolution to enhance the training ability; meanwhile, we are free to choose the count of the kernels in each filter batch corresponding to each dimension. The memory consumption is exponentially lower than the standard convolution. It can be used in devices and systems with far less computational and memory ability or even a deeper structure. Also, The accuracy and loss results have been shown in the number of parameters and floating-point calculations and the experiments section.

# Bibliography


[1] N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis. CoRR, abs/1509.05009, 2015. 1

[2] A. Anandkumar, R. Ge, D. J. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. Journal of Machine Learning Research, 15(1):27732832, 2014.

[3] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In Proceedings of the fourth ACM conference on Recommender systems, pages 7986. ACM, 2010.

[4] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521, no. 7553 (2015): 436-444.

[5] [1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580587, 2014.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You only look once: Unified, real-time object detection, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779788, 2016.

[7] M. A. Waris, A. Iosifidis, and M. Gabbouj, Cnn-based edge filtering for object proposals, Neurocomputing, 2017.

[8] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly,A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 8297, 2012.

[9] A. Graves, A.-r. Mohamed, and G. Hinton, Speech recognition with deep recurrent neural networks, in Acoustics, speech and signal processing (icassp), 2013 ieee international conference on, pp. 66456649, IEEE, 2013.

[10] M. Zabihi, A. B. Rad, S. Kiranyaz, M. Gabbouj, and A. K. Katsaggelos, Heart sound anomaly and quality detection using ensemble of neural networks without segmentation, in Computing in Cardiology Conference (CinC), 2016, pp. 613616, IEEE, 2016.

[11] X. An, D. Kuang, X. Guo, Y. Zhao, and L. He, A deep learning method for classification of eeg data based on motor imagery, in International Conference on Intelligent Computing, pp. 203210, Springer, 2014.

[12] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, Using deep learning to detect price change indications in financial markets, in European Signal Processing Conference (EUSIPCO), Kos, Greece, 2017.

[13] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, Forecasting stock prices from the limit order book using convolutional neural networks, in Business Informatics (CBI), 2017 IEEE 19th Conference on, vol. 1, pp. 712, IEEE, 2017.

[14] H. Sedghi and A. Anandkumar. Training input-output recurrent neural networks through spectral methods. CoRR, abs/1603.00954, 2016.

[15] M. Janzamin, H. Sedghi, and A. Anandkumar. Generalization bounds for neural networks through tensor factorization. CoRR, abs/1506.08473, 2015.

[16] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. CoRR, abs/1412.6553, 2014.

[17] Y. Yang and T. M. Hospedales. Deep multi-task representation learning: A tensor factorisation approach. CoRR, abs/1605.06391, 2016.

[18] Y. Chen, X. Jin, B. Kang, J. Feng, and S. Yan. Sharing residual units through collective tensor factorization in deep neural networks. 2017

[19] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

[20] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[21] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012): 1097-1105.

[22] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[23] Kolda, Tamara G., and Brett W. Bader. "Tensor decompositions and applications." SIAM review 51, no. 3 (2009): 455-500.

[24] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251-1258. 2017.

[25] Lym, Sangkug, Donghyuk Lee, Mike O'Connor, Niladrish Chatterjee, and Mattan Erez. "DeLTA: GPU Performance Model for Deep Learning Applications with In-Depth Memory System Traffic Analysis." In 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 293-303. IEEE, 2019.

[26] Tran, Dat Thanh, Alexandros Iosifidis, and Moncef Gabbouj. "Improving efficiency in convolutional neural networks with multilinear filters." Neural Networks 105 (2018): 328-339.

[27] Rezghi, Mansoor, and Lars Elden. "Diagonalization of tensors with circulant structure." Linear Algebra and its Applications 435, no. 3 (2011): 422-447.

[28] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, Y. Chen, Compressing neural networks with the hashing trick, in: Proceedings of the International Conference on Machine Learning (ICML), 2015, pp. 22852294.

[29] L. Y. Pratt, Comparing biases for minimal network construction with back-propagation, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 1988, pp. 177185

[30] Xie, Guotian. "Redundancy-Aware Pruning of Convolutional Neural Networks." Neural Computation 32, no. 12 (2020): 2532-2556.

[31] J. Lin, O. Morere, V. Chandrasekhar, A. Veillard, H. Goh, Deephash: Getting regularization, depth and fine-tuning right, in: Proceedings of the International Conference on Multimedia Retrieval (ICMR), 2017, pp. 133141.

[32] C. Tai, T. Xiao, Y. Zhang, X. Wang, and E. Weinan, Convolutional neural networks with low-rank regularization,

[33] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, A. Y. Ng, Tiled convolutional neural networks, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2010, pp. 12791287.

[34] M. D. Zeiler, G. W. Taylor, R. Fergus, Adaptive deconvolutional networks for mid and high level feature learning, in: Proceedings of the International Conference on Computer Vision (ICCV), 2011, pp. 20182025.

[35] M. D. Zeiler, D. Krishnan, G. W. Taylor, R. Fergus, Deconvolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 25282535.

[36] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, in: Proceedings of the International Conference on Learning Representations (ICLR), 2016.

[37] Novikov, Alexander, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P. Vetrov. "Tensorizing neural networks." Advances in neural information processing systems 28 (2015): 442-450.

[38] Kossaifi, Jean, Zachary C. Lipton, Arinbjörn Kolbeinsson, Aran Khanna, Tommaso Furlanello, and Anima Anandkumar. "Tensor regression networks." Journal of Machine Learning Research 21 (2020): 1-21.

[39] Huang, Gao, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. "Densely connected convolutional networks. arXiv 2016." arXiv preprint arXiv:1608.06993 1608 (2018).