DOI: 10.15439/2021F127
ISSN 2300-5963 ACSIS, Vol. 26

An impact of tensor-based data compression on deep neural network accuracy

Jakub Grabek*† and Bogusław Cyganek*†
*Department of Electronics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
Email: *jakub.grabek@qed.pl, cyganek@agh.edu.pl*
†QED Software Sp. z o.o.,
Mazowiecka 11/49, 00-052 Warszawa, Poland

***Abstract*—The emergence of the deep neural architectures greatly influenced the contemporary big data revolution. However, requirements on large datasets even increased a necessity for efficient data storage. The storage problem is present at all stages, from the dataset creation up to the training and prediction stages. However, compression algorithms can significantly deteriorate the quality of data and in effect the classification models. In this article, an in-depth analysis of the influence of the tensor-based lossy data compression on the performance of the various deep neural architectures is presented. We show that the Tucker and the Tensor Train decomposition methods, with properly selected parameters, allow for very high compression ratios, while conveying enough information in the decompressed data to achieve only a negligible or very small drop in the accuracy. The measurements were performed on the popular deep neural architectures: AlexNet, ResNet, VGG, and MNASNet. We show that further augmentation of the tensor decompositions with the ZFP floating-point compression algorithm allows for finding optimal parameters and even higher compressions ratios at the same recognition accuracy. Our experiments show data compressions of 94%-97% that result in less than 1% accuracy drop.**

I. Introduction

The tendency of generating huge volumes of data dynamically increases. In this light data compression methods are of high importance. This is also true in the ML/AI domain, where modern deep neural architectures require larger training data sets. On the other hand, it has been shown that tensor decomposition methods allow to achieve huge compression ratios on multidimensional data [1]–[4]. Not surprisingly then that the tensor decompositions have been applied to compress weights of the deep neural networks [5], [6]. There are also works focusing on problem of learning from compact representations of images [7], [8]. However, to the best of our knowledge, there are no studies on the influence of the training data compression with tensor decompositions on the accuracy of the deep neural networks. This paper fills this gap providing an in-depth analysis of the Tucker and Tensor Train (TT) decomposition based data compression methods on the performance of the common deep network architectures such as AlexNet, ResNet, VGG, and MNASNet. The networks are trained in different scenarios, and due to the batch processing not all data need to be decompressed at the same time. Furthermore, we propose an additional step of data compression based on the ZFP floating-point lossy compression method [9]. Our experimental results show that the properly setup tensor decompositions followed by the ZFP module allow for as high as 94%-97% data compression ratios with less than 1% drop in accuracy of the deep neural networks.

The rest of the paper is organized as follows. Section II describes the related works. In Section III the two tensor decomposition methods used in our experiments are discussed. Section IV presents the ZFP floating-point compression method. In Section V neural network models utilized during experiments are described. Our proposed tensor-based training method is explained in Section VI. Experimental results with a discussion on results are described in Section VII. Finally, Section VIII concludes the paper.

II. Related Works

Compression methods gained much attention over the recent few decades. Since the seminal works of Lempel & Ziv in the lossless compression [10], much more diversed methods have been proposed in the lossy compression domain. Soon it was realized that various matrix, such as the well-known SVD one, and tensor decompositions can lead to significant data reductions [11]. However, the latter depends on many parameters, such as the tensor rank [2], [12]. In the case of multidimensional data, such as images, video, etc., the tensor-based approach offers much more possibilities, as will be discussed. In the era of deep neural networks, tensor-based methods proved to be superior in compressing their weights. However, it is also possible to compress their training and testing data - in this paper, we explore this branch.

Balle et al. proposed an image compression method consisting of nonlinear transformations for analysis and signal synthesis [13]. Three stages of convolutional linear filters with nonlinear activation functions were used to create both transforms. It achieved better rate-distortion performance than the standard JPEG and JPEG 2000 compression methods.

The tensor approach was explored by Zhang et al. [14]. The hyperspectral images were stacked into 3D tensors in which spatial-spectral structure is preserved. The data - approximated and stored in projection matrices - achieved a high compression ratio with a low value of introduced artifacts.

Aidini et al. used the CANDECOMP/PARAFAC tensor decomposition to the compression method [15]. The multispectral image time series was expressed as a linear combination of the learned tensors and the quantization of the coefficients using the learned encoding dictionary.

Watkins and Sayeh proposed deep neural networks based method for gray image compression. The network is based on the autoencoder structured network, capable of both compressing and decompressing images [16] with a high compression ratio.

A multispectral image compression method based on the convolutional neural network was proposed by Li and Liu [5]. The processing path consists of the encoder and decoder parts. Both parts use CNN in combination with discrete cosine transform and nonnegative tensor decomposition (NDT) in pseudo-autoencoder structure. The method shows improved computational efficiency with comparable PSNR values compared to direct NDT in the wavelet domain.

Friedland et al. [17] analyze an impact of artifacts from perceptual compression on deep learning, concluding that classification accuracy is tightly connected to compression rate and data quantization. The loss of classification efficiency was mainly due to artifacts introduced during the compression process.

Similarly, in the PhD thesis on the impact of standard image compression techniques on CNN performance, Dejan concluded that network trained on JPEG encoded images partially relies on artifacts introduced by the compression [18].

In the paper, the authors do not use weight compression [6], [19], [20]. This topic is an additional option for further saving space in neural networks and can be implemented in future works. In recent years, tensors gained much attention from the ML/AI community, also in the context of data decompositions for compression [2]–[4], [12]. However, there is no work to analyze the influence of tensor-based compression of the training and testing data on the performance of the deep neural networks. We fill this gap, starting in the next section with a brief introduction to tensors.

III. Tensor Compression

Tensors are mathematical objects which can be regarded as multidimensional arrays of data, in which each separate dimension corresponds to a different degree of freedom of a measurement. Such an approach provides tools that extend the classical matrix analysis and which can take into account correlations hidden in data, to yield better results in various applications, such as compression or filtering [2].

Tensors extend the notion of vectors and matrices into higher-dimensional objects [2], [3], [21]–[23]. As discussed below, they allow for better representation and processing of the multidimensional signals and, in effect, also for higher compression ratios.

The multidimensional compression can be based on the following tensor product

$$\hat{\mathcal{T}} = \mathcal{T} \times_1 \mathbf{F}_1 \times_2 \mathbf{F}_2 ... \times_P \mathbf{F}_P \tag{1}$$

where decompressed version of input tensor $\mathcal{T}$ is denoted as $\hat{\mathcal{T}}$, whereas $\mathbf{F}_i$ is the ith factor matrix. The key idea is that the set of factor matrices and their product with $\mathcal{T}$, from the right side of (1), require much less storage space than the original tensor $\mathcal{T}$, while its recovered version $\hat{\mathcal{T}}$ is close enough in terms of the chosen norm, allowing e.g. for proper CNN training, as will be discussed. Also in the above equation, the k-th modal product $\mathcal{T} \times_k \mathbf{M}$ of a tensor $\mathcal{T} \in \Re^{N_1 \times N_2 \times ... \times N_P}$ and a matrix $\mathbf{M} \in \Re^{Q \times N_k}$ is used. The result is also a tensor $\mathcal{S} \in \Re^{N_1 \times N_2 \times ... N_{k-1} \times Q \times N_{k+1} \times ... \times N_P}$, whose elements are expressed as follows:

$$\mathcal{S}_{n_1 n_2 ... n_{k-1} q n_{k+1} ... n_P} = (\mathcal{T} \times_k \mathbf{M})_{n_1 n_2 ... n_{k-1} q n_{k+1} ... n_P} = \sum_{n_k=1}^{N_k} t_{n_1 n_2 ... n_{k-1} q n_{k+1} ... n_P} m_{q n_k} \tag{2}$$

As shown below, the compression matrices $\mathbf{F}_i$, called factors, can be obtained using the Tucker decomposition of tensors [12]. Thanks to the proper selection of the ranks of the tensor decomposition factors, decomposition usually well separates useful signal from its high-frequency components while taking multidimensional characteristics of the signal into account. The decomposition procedure of the tensor $\mathcal{T}$ is done by calculation of an approximating tensor $\hat{\mathcal{T}}$ that is close to the input tensor in terms of the Frobenius norm. Hence, a minimization function is defined as follows

$$\Theta(\hat{\mathcal{T}}) = ||\hat{\mathcal{T}} - \mathcal{T}||_F^2 \tag{3}$$

A. Tucker-based methods

The concept of the Tucker decomposition of a 3D tensor is presented in Figure 1. Assuming that the approximating tensor $\hat{\mathcal{T}}$ contains the same amount of useful information as the original tensor $\mathcal{T}$, it can be expressed as follows

$$\hat{\mathcal{T}} = \mathcal{Z} \times_1 \mathbf{S}_1 \times_2 \mathbf{S}_2 \times_3 ... \times_P \mathbf{S}_P \tag{4}$$

where $\mathcal{Z} \in \Re^{R_1 \times R_2 \times ... \times R_P}$ is a core tensor and $\mathbf{S}_i \in \Re^{N_i \times R_i}$ are the so-called mode matrices. Using algebraic operations, from Equation (4), the formula for the core tensor is obtained:

$$\mathcal{Z} = \hat{\mathcal{T}} \times_1 \mathbf{S}_1^T \times_2 \mathbf{S}_2^T \times_2 ... \times_P \mathbf{S}_P^T \tag{5}$$

Combining Equation (5) with Equations (3) and (4) yields

$$\Theta(\hat{\mathcal{T}}) = ||\hat{\mathcal{T}} - \mathcal{T} \prod_{k=1}^{P} \times_k (\mathbf{S}_k \mathbf{S}_k^T)||_F^2 \tag{6}$$

The Tucker decomposition in Equation (6) reads that a tensor $\mathcal{T}$ is approximated by its projection onto space spanned by the matrices $\mathbf{S}_k$. To compute the series of $\mathbf{S}_k$ matrices, the alternating method can be used [2], [21], [24]–[26]. Also, let's observe that $\mathbf{S}_i \mathbf{S}_i^T$ in (6) is equivalent to the factor matrix $\mathbf{F}_i$ from (1).

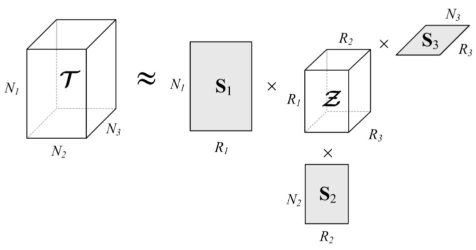


Fig. 1. Visualization of the Tucker decomposition of a 3D tensor.

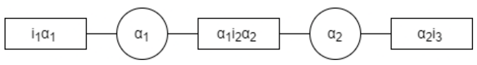


Fig. 2. Visualization of the Tensor Train network.

The approximation in Equation (4) includes only the components conveying the majority of energy available in the signal. However, to ensure the best quality, the estimation of the proper ranks R_1, R_2, and R_3 of the mode matrices S_i is necessary. Although fixed values can be used as a first approximation, the proper ranks need to be based on the signal content in real dynamic systems with unpredictable noise values. Multiple methods were presented to help solve this problem, i.e., Muti and Bourennane [22] using the minimum description length parameter (MDL) computed for each dimension separately, allowing optimal rank selection.

The Tucker format is stable, but its computational complexity grows exponentially with input tensor dimensions. It makes the method suitable only for "small" dimensions [27], [28].

B. Tensor-Train

To mitigate the computational complexity problem, a D-th order tensor $\mathcal{T} \in \Re^{N_1 \times N_2 \times \ldots \times N_D}$ can be represented as $\mathcal{T}(j_1, j_2, ..., j_d) = \mathbf{G}_1[j_1]\mathbf{G}_2[j_2]...\mathbf{G}_d[j_d]$, where $\mathbf{G}_k[j_k]$ is factor matrix with size of $r_{k-1} \times r_k$ $r_0 = r_d = 1$ and $j_k \in \{1, 2, ..., n_k\}$ $(k \in 1, 2, ..., d)$ [45].. The set of r_k is collectively called Tensor Train ranks. The $\mathbf{G}_k[j_k]$ belonging to the same mode can be stacked into 3-rd order core tensor $\mathcal{G}_k \in \Re^{N_1 \times r_{k-1} \times r_k}$ allowing the $\mathcal{T}$ be represented as follows:

$$\mathcal{T} = \mathcal{G}_1 \times^1 \mathcal{G}_2 \times^1 ... \times^1 \mathcal{G}_d \tag{7}$$

where $\times^1$ is called mode-(N,1) contracted product, presented here [29].

The decomposition can be presented graphically by the linear tensor network [30], [31], illustrated in Figure 2. There are two types of nodes: rectangular and circular. Rectangular contains spatial indices (i_k), some auxiliary indices (α_k), and a tensor with these indices associated with such nodes. On the other hand, a circular node is a link and contains only the auxiliary indices. If an auxiliary index is present in two cores, the cores are connected. To evaluate an input tensor, all tensor in rectangles need to be multiplied, and then summation is performed over all auxiliary indices.

Compared to Tucker Decomposition, the Tensor Train format has lower spatial complexity, making it more computationally efficient for tensors with larger dimensions [32].

IV. Fixed-Rate Compressed Floating-Point Arrays

The need for floating-point array compression is expressed by multiple lossy and lossless compression algorithms developed throughout the years. The most widely spread are image compression methods allowing encoding 2D and 3D arrays. For instance, PNG and JPEG-LS use linear prediction; JPEG - the block transform coding; JPEG2000 relies on the higher-order wavelets.

The Fixed-Rate Compressed Floating-Point Arrays (ZFP) compression scheme is based on ideas developed to compress 2D image data efficiently [33]. The input 3D array is divided into small, fixed-size blocks of dimensions 4 x 4 x 4, stored using a user-specified number of bits, which can be accessed independently. The method compresses the block performing the following steps:

1) Align the values in a block to a common exponent;
2) Convert the floating-point values in a block to a common exponent;
3) Convert the floating-point values to a fixed-point representation;
4) Apply an orthogonal block transform to decorrelate the values;
5) Order the transform coefficients by the expected magnitude;
6) Encode the resulting coefficients one "bit plane" at a time

The conversion to fixed-point is done by expressing each block value with respect to the largest floating-point exponent in a block, which is stored uncompressed resulting in normalized values in the range (-1, +1).

The prepared values are transformed to a basis allowing the spatially correlated values to be mostly decorrelated, as this results in many near-zero coefficients that can be compressed efficiently.

A separable orthogonal transform in d dimensions is employed to take advantage of regularly gridded data, resulting in a basis that is the tensor product of 1D basis functions. The proposed transform, due to coefficient selection, replaces divisions and multiplications into bitshifts. This choice achieves near-optimal results in terms of decorrelation efficiency and coding gain and is very efficient from a computational perspective. Further details of the ZFP method can be accessed from [33].

V. Neural Network Models

The advent of modern ML/AI methods, especially deep neural networks (DNN), resulted in a real IT revolution [34]. In a very short time, people realized real power in these systems that can be directly trained from data with marvelous results. Hence, the term "data" gained even more importance. With these came the eruption of modern deep neural network architectures, such as AlexNet [35], VGG [36], ResNet [37], and dozens of their derivatives [38]–[41]. These were possible thanks to novel scientific achievements such as a solution to the vanishing gradient problem in very deep networks, optimization algorithms, and thanks to the availability of the efficient general-purpose graphics processing units (GGPU).

For example, in the computer vision domain, neural networks dominate in majority of tasks, such as object detection & recognition, segmentation, filtering, to name a few. However, performance of the neural networks depends heavily on availability of the high quality labeled data. However, this can be jeopardized by many factors, such as compression-decompression processes, we focus upon in this work.

To examine the impact of the proposed compression method, state-of-the-art models capable of high accuracy with reasonable low training time were used. Their architectures are shortly described below.

AlexNet

The neural network contains eight learned layers [35] - five convolutional and three fully connected. It introduces such features as ReLU nonlinearity, the capability of training on multiple GPUs, local response normalization, and overlapping pooling. The new approach combined with a high emphasis on overfitting reduction results in a highly accurate model, even today. The network is recognized as a milestone, and solutions proposed in the article are now considered as standard by AI/ML community.

ResNet

The ResNet networks implements skip connections with ReLU nonlinearity and batch normalization [41]. It allows to mitigate vanishing gradient problem and speeds up the training process, which positively impacts the feasibility of training deeper neural networks.

VGG

The model uses small receptive fields, decreasing the number of trainable parameters and increasing the ReLU unit count [36]. Such an approach makes the decision function more discriminative, which in turn increases overall network performance.

MNASNet

It is a neural architecture for mobile devices [42]. It bases on Factorized Hierarchical Search Space approach, which balances the diversity of layers and the size of total search space. The resulting network generally runs faster and uses less computational power.

VI. Tensor-based Data Processing

The main goal of our approach is to decrease data size with the lossy compression process without a significant impact on the quality of object prediction by the benchmark neural-network architectures. The proposed method requires an input in a tensor form. Based on images selected to process, the width (W) and height (H) parameters, describing tensor dimensions, are selected based on the biggest image in a set. The last parameter depth (D) depends only on the number of input images. The input tensor is assembled utilizing the method shown in Algorithm 1 and produces $W \times H \times D$ data chunk used for further processing.

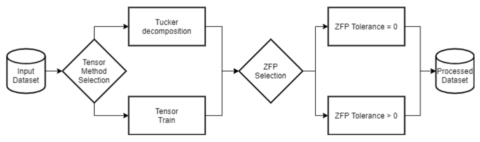


Fig. 3. Block diagram of proposed method's compression step.

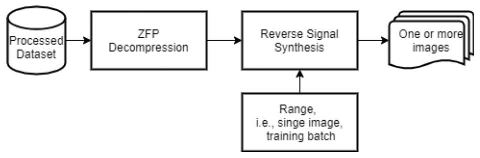


Fig. 4. Block diagram of proposed method's decompression step.

The next step is a tensor decomposition, as presented in Figure 3. Depending on the selected method, the result is a set of mode matrices (Tensor Train) or core tensor and mode matrices (Tucker decomposition). These approximated signals contain the most relevant information, and higher frequency components are removed. Smaller rank values selected for the method translate to higher compression and more significant high-frequency attenuation. In the next step, obtained results are compressed with the ZFP algorithm using a tolerance argument ZFP_t that controls the compression quality. Such compressed data objects can be stored on a disk for further use or be utilized as an intermediate step in real-time processing.

Before the next step, processed data needs to be decompressed, as presented in the Figure 4. First, using the ZFP decompression method, then in the reverse signal synthesis process, the aforementioned matrices are merged into the result tensor. However, the reverse signal synthesis can be calculated for the entire tensor, a single image, or a set of consecutive images. Such flexibility allows decrease computational requirements and allows the method to be used as part of modern neural-network training architecture.

VII. Experimental Results

The presented method was implemented in Python, using the NumPy, SciPy, and scikit-image packages. TensorLy library was used for tensor decomposition [43], and ZFP library for the additional compression of multidimensional floating-point arrays [33]. As the benchmark, the following neural network architectures were selected: AlexNet, ResNet-18, ResNet-34, VGG-11, VGG-13, and MNASNet0.5.

Presented Experiments were performed on a server computer, equipped with 256 GB of RAM, 64-core processor AMD Ryzen Threadripper 3990X with the 2.9 GHz base clock, and 64-bit Ubuntu 20.04.2 LTS OS.

The quantitative results were measured in terms of compression ratio (Cr) and object detection accuracy of neural

Algorithm 1 Tensor assembler.

```
1: $T = \emptyset$
2: while $I \neq \emptyset$ do
3:     load original image $I_i$ and prepare container for resized image $I_p$
4:     calculate $x$ and $y$ offsets needed to place $I_i$ in the center of $I_p$
5:     resize selected image from $I$ to dimensions specified in $W$ and $H$, keeping aspect ratio of original data and save it to $I_p$
6:     append $I_p$ to $T$
7:     remove $I_i$ from $I$
8: end while
9: return $T$
```

network (Acc). The compression ratio is defined as follows:

$$Cr = \frac{\text{Uncompressed data size}}{\text{Compressed data size}} \tag{8}$$

Furthermore, an object detection accuracy of a neural network is described as the proportion of correct predictions over the total examined cases:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

where TP, TN, FP, FN represents true positive, true negative, false positive, and false negative values achieved by the network model, on the tested dataset, respectively.

For the quantitative evaluation of the proposed method, Imagenette [44] was selected. The dataset is a subset of 10 easily classified classes from the Imagenet dataset and was selected to decrease the time needed for development and tests. The original structure contains train and validation sets with 9469 and 3925 images, respectively. For better quality estimation, the presented accuracy of tested neural networks is measured using a test set containing 1500 images. The before-mentioned set was separated from the original subsets using 900 images from the train set and 600 images from the validation set. The final image count for used image sets is as follows: training 8569 images, validation 3325 images, and test 1500.

The input data was processed using the tensor-based compression methods described in Section VI. The collection of test datasets was created depending on selected parameters and methods, as shown in Table I.

To cover a broad spectrum of possible utilization of the proposed method, it was decided to check an impact of tensor based data compression/decompression on networks depending on the learning process. Both random weight initialization and the transfer learning technique were used. For the latter, models were pre-trained on the Imagenet dataset.

The training process was conducted for each of the prepared datasets and each of the selected benchmark neural networks, and results were measured. All prepared data plots from Figure 5 to Figure 19 are presented below.

Denoting dimensions of the input tensor as $W \times H \times D$, values of the corresponding ranks for tensor methods were calculated as $[0.5W, 0.5H, D]$ for Tucker decomposition. Similarly, in the case of the tensor train method, dimensions of the factor matrices were set to $[1, 0.5min(W, H), 0.5min(W, H), 1]$. Obtained compression ratios for different tensor methods and values of ZFP tolerance are presented in Table II. Results of the neural network accuracy are presented in Table III for the random weight initialization and in Table IV for the transfer learning, respectively. The results are discussed below.

For the transfer learning technique, it can be observed that TT achieved better accuracy for the selected ranks. Depending on the tested neural network, it achieves 0.6 - 2.5% worse results compared to the original dataset. At the same time, the TT method provides users with a lower compression ratio, between 14.92 and 16.86 for lossless and lossy ZFP compression, respectively. On the other hand, the Best rank method achieves higher compression 20.41 - 21.39, with lower network accuracy however. In this case the difference between original data and tested datasets is higher, from 2.1% to 6.3%. Additionally, for nearly all cases, methods combining tensor-based algorithm with lossy ZFP yield better results than ZFP lossless ones. Depending on the tested neural network, the lossy ZFP version achieves 0.6 - 1.1% and 2.1 - 4.1% difference between original for the TT and Best rank methods, respectively.

In the random weight training case, again TT achieves better accuracy, with an accuracy loss between 3.3% - 7.2% depending on the tested neural network. The Best rank method yields 6.6% - 13.5% drop in accuracy. In the considered context, for nearly all tested cases, lossy ZFP compression does not change the accuracy achieved by tested neural networks.

Hence, the proposed method combines high compression capabilities with good retention of important signals for the detection process. For the most common neural network training technique used today - transfer learning - the method achieves results very close to the values obtained on the original dataset.

Higher compression rate impacts all tested networks' quality by removing high-frequency components from the images, which means lowering object detection accuracy by 0.6 - 1.1% and 2.1 - 4.1% for the Tensor train and Best rank methods, respectively. However, tensor methods inherently allow an easy change in the compression rate by selecting different ranks during the decomposition process, making it possible to find an acceptable trade-off for a given application.

VIII. Conclusion

In this paper, a novel framework and experiments on data compression/decompression in order to measure the impact on the deep neural network training and prediction are presented. The compression/decompression is based on tensor decomposition methods combined with the floating-point array compression. We show that the presented methods can achieve very high compression ratios while still preserving enough

TABLE I
DATASETS USED IN NETWORK QUALITY ASSESSMENT.

	Name	**Tensor compression type**	**ZFP tolerance**	**Validation set compressed?**
A	original	-	-	false
B	best_rank	Best rank	lossless	true
C	best_rank_comp	Best rank	1e-3	true
D	best_rank_orig_val	Best rank	lossless	false
E	best_rank_comp_orig_val	Best rank	1e-3	false
F	tensor_train	Tensor train	lossless	true
G	tensor_train_comp	Tensor train	1e-3	true
H	tensor_train_orig_val	Tensor train	lossless	false
I	tensor_train_comp_orig_val	Tensor train	1e-3	false

TABLE II
COMPRESSION RESULTS.

Tensor compression type	**ZFP tolerance**	**Compression ratio (Space saving)**	**Complete processing time [h:mm:ss]**
Best rank	lossless	20.41 (0.951)	2:46:02
Best rank	1e-3	21.39 (0.953)	2:47:09
Tensor train	lossless	14.92 (0.932)	1:27:38
Tensor train	1e-3	16.86 (0.941)	1:27:43

TABLE III
NETWORK ACCURACY VERSUS DATASET TYPE. TRAINING RESULTS FOR RANDOM WEIGHT INITIALIZATION. DATASETS: ORIGINAL (A), BEST_RANK (B), BEST_RANK_COMP (C), BEST_RANK_ORIG_VAL (D), BEST_RANK_COMP_ORIG_VAL (E), TENSOR_TRAIN (F), TENSOR_TRAIN_COMP (G), TENSOR_TRAIN_ORIG_VAL (H), TENSOR_TRAIN_COMP_ORIG_VAL (I)

	AlexNet	**ResNet-18**	**ResNet-34**	**VGG-11**	**VGG-13**	**MNASNet0.5**
A	0.7687	0.8025	0.7962	0.8018	0.7806	0.7389
B	0.6497	0.6683	0.6581	0.6838	0.6713	0.6033
C	0.6555	0.7292	0.7304	0.7139	0.6945	0.6454
D	0.6482	0.681	0.7083	0.6851	0.6566	0.6191
E	0.6662	0.693	0.6856	0.6823	0.6741	0.6321
F	0.7093	**0.7554**	0.7605	0.7422	0.7228	0.6652
G	0.7271	0.7521	0.7475	0.7427	**0.7338**	**0.681**
H	**0.7389**	0.7493	**0.7623**	0.7417	0.7335	0.6795
I	0.7284	0.7439	0.7475	**0.7483**	0.7264	0.6561

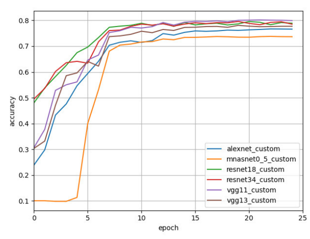
TABLE IV
NETWORK ACCURACY VERSUS DATASET TYPE. TRAINING RESULTS FOR TRANSFER LEARNING TECHNIQUE. DATASETS: ORIGINAL (A), BEST_RANK (B), BEST_RANK_COMP (C), BEST_RANK_ORIG_VAL (D), BEST_RANK_COMP_ORIG_VAL (E), TENSOR_TRAIN (F), TENSOR_TRAIN_COMP (G), TENSOR_TRAIN_ORIG_VAL (H), TENSOR_TRAIN_COMP_ORIG_VAL (I)

	AlexNet	**ResNet-18**	**ResNet-34**	**VGG-11**	**VGG-13**	**MNASNet0.5**
A	0.9381	0.9689	0.9664	0.9791	0.9783	0.9592
B	0.8943	0.9378	0.9434	0.9577	0.9595	0.9177
C	0.8961	0.9411	0.9434	0.9562	0.9575	0.8963
D	0.893	0.9394	0.9338	0.9572	0.9597	0.9185
E	0.8854	0.9378	0.9437	0.9572	0.9618	0.9141
F	0.9124	0.9549	0.9498	0.9654	0.9697	0.9335
G	**0.9302**	**0.961**	**0.959**	0.971	**0.972**	**0.9508**
H	0.9243	0.9575	0.9582	**0.9715**	0.9715	0.9501
I	0.9261	0.9585	0.9575	0.9674	0.9689	0.9503

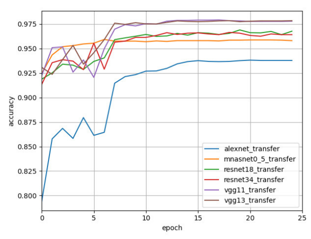
significant information in data to achieve high accuracy of object detection in the neural models.

The utilized algorithms can smoothly change the achieved compression rate, which impacts network accuracy during the training process, allowing users to find parameters that are optimal for a given application.

Furthermore, storing data in the proposed form allows for a selective decompression of a single or a group of images without the need to decompress the entire tensor. The accuracy drop in respect to the original data is visible, but for the problems where storage or data transfer speed are important, it can increase performance both during training and normal operation. Alternatively, we can say that thanks to data compression a larger amount of data can be transferred and

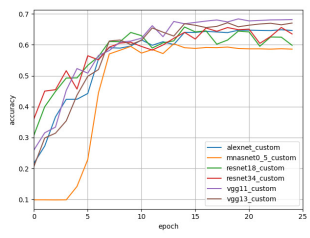


(a) Random initialization.

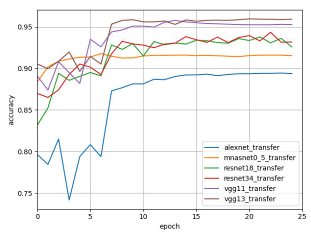


(b) Transfer learning.

Fig. 5. Comparison between all used architectures trained on the original dataset.

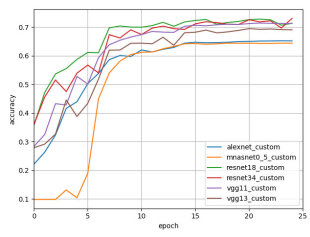


(a) Random initialization.

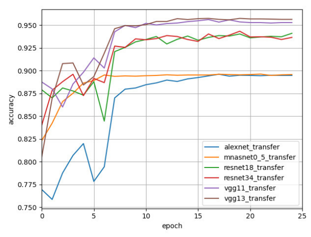


(b) Transfer learning.

Fig. 6. Comparison between all used architectures trained on the dataset compressed with Tucker decomposition and lossless ZFP.

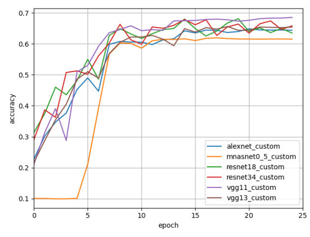


(a) Random initialization.

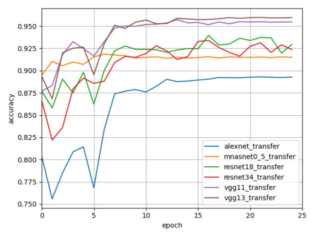


(b) Transfer learning.

Fig. 7. Comparison between all used architectures trained on the dataset compressed with Tucker decomposition and lossy ZFP.

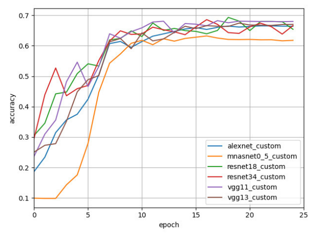


(a) Random initialization.

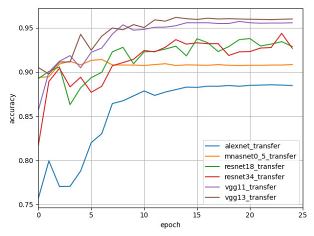


(b) Transfer learning.

Fig. 8. Comparison between all used architectures trained on the dataset compressed with Tucker decomposition and lossless ZFP with original validation set.

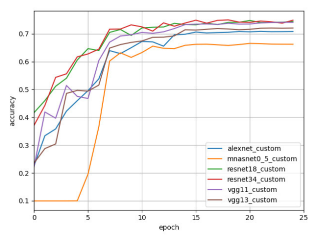


(a) Random initialization.

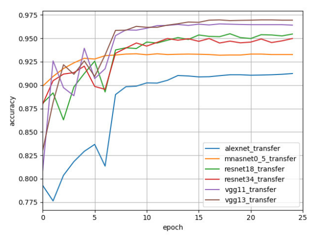


(b) Transfer learning.

Fig. 9. Comparison between all used architectures trained on the dataset compressed with Tucker decomposition and lossy ZFP with original validation set.

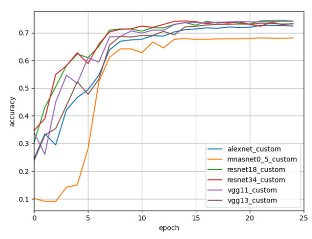


(a) Random initialization.

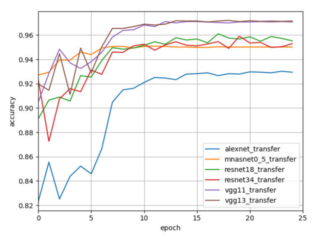


(b) Transfer learning.

Fig. 10. Comparison between all used architectures trained on the dataset compressed with Tensor Train algorithm and lossless ZFP.

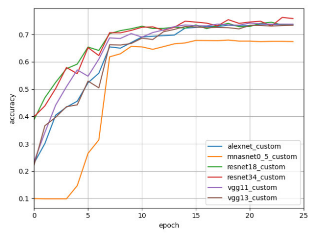


(a) Random initialization.

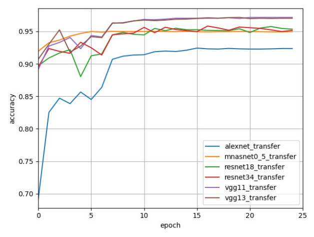


(b) Transfer learning.

Fig. 11. Comparison between all used architectures trained on the dataset compressed with Tensor Train algorithm and lossy ZFP.

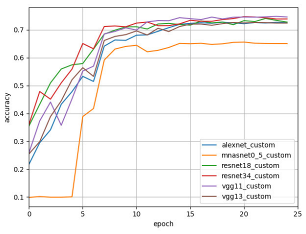


(a) Random initialization.

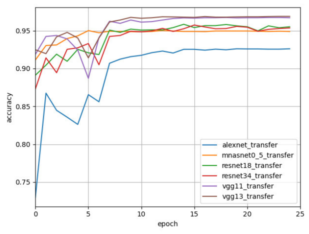


(b) Transfer learning.

Fig. 12. Comparison between all used architectures trained on the dataset compressed with Tensor Train algorithm and lossless ZFP with original validation set.

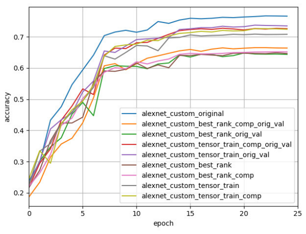


(a) Random initialization.

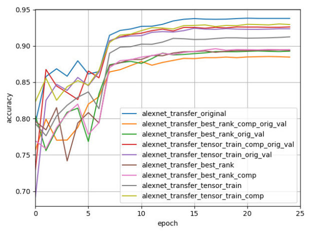


(b) Transfer learning.

Fig. 13. Comparison between all used architectures trained on the dataset compressed with Tensor Train algorithm and lossy ZFP with original validation set.

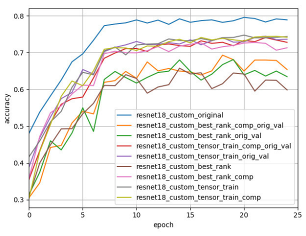


(a) Random initialization.

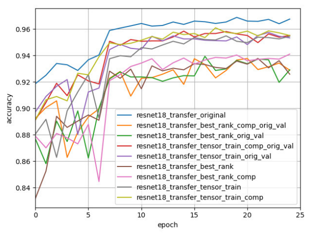


(b) Transfer learning.

Fig. 14. AlexNet training results for each tested dataset.

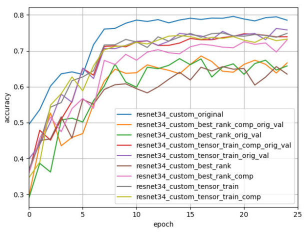


(a) Random initialization.

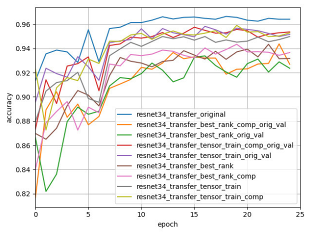


(b) Transfer learning.

Fig. 15. ResNet-18 training results for each tested dataset.

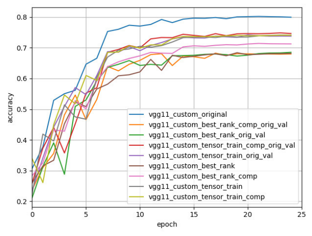


(a) Random initialization.

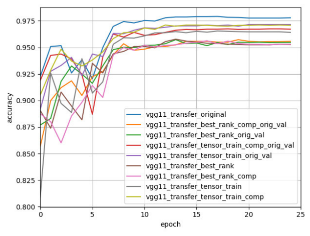


(b) Transfer learning.

Fig. 16. ResNet-34 training results for each tested dataset.

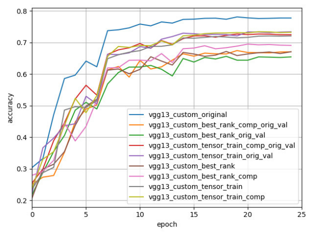


(a) Random initialization.

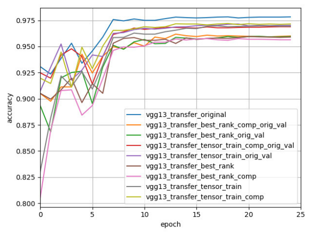


(b) Transfer learning.

Fig. 17. VGG-11 training results for each tested dataset.



(a) Random initialization.



(b) Transfer learning.

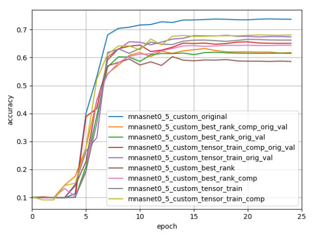
Fig. 18. VGG-13 training results for each tested dataset.

used for training. Also, although the method was developed for images, it can be useful for other data types, such as physical or industry measurements, etc.

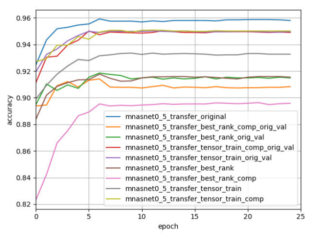
Summarizing, the best results were achieved using the transfer learning technique, where a dataset is processed with Tensor Train decomposition paired with the lossy version of the ZFP algorithm. In this setting 0.6% - 0.7% drop in accuracy of the deep neural networks in respect to the original dataset was achieved for all tested methods. The best accuracy, both in respect to the original and processed dataset, was obtained with the VGG-11 and VGG-13 models.

Acknowledgment

This research was co-funded by Smart Growth Operational Programme 2014-2020, financed by European Regional Development Fund, in frame of project POIR.01.01.01-00-0570/19, operated by National Centre for Research and Development in Poland.



(a) Random initialization.



(b) Transfer learning.

Fig. 19. MNASNet0.5 training results for each tested dataset.

REFERENCES

[1] COCOCCIONI, M., et al. Novel arithmetics in deep neural networks signal processing for autonomous driving: Challenges and opportunities. In *IEEE Signal Processing Magazine*, **2020**, 38.1: 97-110. doi: 10.1109/MSP.2020.2988436

[2] Cyganek, B. *Object Detection and Recognition in Digital Images: Theory and Practice*; John Wiley & Sons: New York, NY, USA, 2013. doi: 10.1002/9781118618387

[3] Kolda, T.; Bader, B. Tensor Decompositions and Applications. *SIAM Rev. 51.3* **2009**, *51*, 455–500. doi: 10.1137/07070111X

[4] Cyganek, B., Thumbnail Tensor—A Method for Multidimensional Data Streams Clustering with an Efficient Tensor Subspace Model in the Scale-Space, Sensors, 19(19), 4088, 2019, doi: 10.3390/s19194088

[5] LI, J., LIU, Z., Multispectral transforms using convolution neural networks for remote sensing multispectral image compression. In *Remote Sensing 11.7: 759*, **2019**. doi: 10.3390/rs11070759

[6] CHOI, Y., EL-KHAMY, M., LEE, J., Universal deep neural network compression. In *IEEE Journal of Selected Topics in Signal Processing, 14.4*, **2020**, pp. 715-726. doi: 10.1109/JSTSP.2020.2975903

[7] Przyborowski M., et al. Toward Machine Learning on Granulated Data – a Case of Compact Autoencoder-based Representations of Satellite Images. In *2018 IEEE International Conference on Big Data (Big Data)*, **2018**, pp. 2657-2662, doi: 10.1109/BigData.2018.8622562.

[8] WANG, N; YEUNG, D. Y., Learning a deep compact image representation for visual tracking. In *Advances in neural information processing systems*, **2013**

[9] Lindstrom, P., Fixed-Rate Compressed Floating-Point Arrays. In *IEEE Transactions on Visualization and Computer Graphics 20(12)* **2014**, pp. 2674-2683, doi:10.1109/TVCG.2014.2346458

[10] ZIV, J., LEMPEL, A., Compression of individual sequences via variable-rate coding. In *IEEE transactions on Information Theory*, **1978**, 24.5: 530-536. doi: 10.1109/TIT.1978.1055934

[11] Cyganek, B., A Framework for Data Representation, Processing, and Dimensionality Reduction with the Best-Rank Tensor Decomposition. Proceedings of the ITI 2012 34th International Conference Information Technology Interfaces, June 25-28, 2012, Cavtat, Croatia, pp. 325-330, doi:10.2498/iti.2012.0466, **2012**.

[12] De Lathauwer, L.; De Moor, B.; Vandewalle, J. On the best rank-1 and rank-(R1, R2,..., Rn) approximation of higher-order tensors. *Siam J. Matrix Anal. Appl.* **2000**, *21*, 1324–1342. doi: 10.1137/S0895479898346995

[13] BALLÉ, J., LAPARRA, V., SIMONCELLI, E. P., End-to-end optimized image compression. In *arXiv preprint arXiv:1611.01704*, **2016**.

[14] ZHANG, L., et al. Compression of hyperspectral remote sensing images by tensor approach. In *Neurocomputing, 147*, **2015**, pp. 358-363. doi: 10.1016/j.neucom.2014.06.052

[15] AIDINI, A., TSAGKATAKIS, G., TSAKALIDES, P., Compression of high-dimensional multispectral image time series using tensor decomposition learning. In: *2019 27th European Signal Processing Conference (EUSIPCO). IEEE*, **2019**. p. 1-5. doi: 10.23919/EUSIPCO.2019.8902838

[16] WATKINS, Y. Z., SAYEH, M. R., Image data compression and noisy channel error correction using deep neural network. In *Procedia Computer Science, 95*, **2016**, pp. 145-152. doi: 10.1016/j.procs.2016.09.305

[17] FRIEDLAND, G., et al. On the Impact of Perceptual Compression on Deep Learning. In *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, **2020**, p. 219-224. doi: 10.1109/MIPR49039.2020.00052

[18] DEJEAN-SERVIÈRES, M., et al. Study of the impact of standard image compression techniques on performance of image classification with a convolutional neural network. **2017**. PhD Thesis. INSA Rennes; Univ Rennes; IETR; Institut Pascal.

[19] ULLRICH, K., MEEDS, E., WELLING, M., Soft weight-sharing for neural network compression. In *arXiv preprint arXiv:1702.04008*, **2017**.

[20] JIN, S. et al. DeepSZ: A novel framework to compress deep neural networks by using error-bounded lossy compression. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, **2019** p. 159-170. doi: 10.1145/3307681.3326608

[21] DENG, Lei, et al. Model compression and hardware acceleration for neural networks: A comprehensive survey. In *Proceedings of the IEEE*, **2020**, 108.4: 485-532. doi: 10.1109/JPROC.2020.2976475

[22] Muti, D.; Bourennane, S. Multidimensional filtering based on a tensor approach. *Signal Process.* **2005**, *85*, 2338–2353. doi: 10.1016/j.sigpro.2004.11.029

[23] Cyganek, B.; Smołka, B. Real-time framework for tensor-based image enhancement for object classification. *Proc. SPIE* **2016**, *9897*, 98970Q. doi: 10.1117/12.2227797

[24] Cyganek, B.; Krawczyk, B.; Wozniak, M. Multidimensional Data Classification with Chordal Distance Based Kernel and Support Vector Machines. *Eng. Appl. Artif. Intell.* **2015**, *46*, 10–22. doi: 10.1016/j.engappai.2015.08.001

[25] Cyganek, B.; Wozniak, M. Tensor-Based Shot Boundary Detection in Video Streams. *New Gener. Comput.* **2017**, *35*, 311–340. doi: 10.1007/s00354-017-0024-0

[26] Marot, J.; Fossati, C.; Bourennane, S. Fast subspace-based tensor data filtering. In Proceedings of the 2009 16th IEEE International Conference on Image Processing (ICIP), Cairo, Egypt, 7–10 November 2009; pp. 3869–3872. doi: 10.1109/ICIP.2009.5414048

[27] Khoromskij, B. N., Khoromskaia, V., Multigrid accelerated tensor approximation of function related multidimensional arrays. In *SIAM J. Sci. Comput., 31*, **2009**, pp. 3002–3026. doi: 10.1137/080730408

[28] Oseledets, I. V., Savostianov, D. V., Tyrtyshnikov, E. E., Tucker dimensionality reduction of three-dimensional arrays in linear time. In *SIAM J. Matrix Anal. Appl., 30*, **2008**, pp. 939–956. doi: 10.1137/060655894

[29] Lee, N., Cichocki, A., Fundamental tensor operations for large-scale data analysis using tensor network formats. In *Multidimensional Syst. Signal Process., vol. 29, no. 3*, **2017**, pp. 921–960 doi: 10.1007/s11045-017-0481-0

[30] Hubener, R., Nebendahl, V., Dur, W., Concatenated tensor network states. In *New J. Phys., 12*, **2010**, 025004. doi: 10.1088/1367-2630/12/2/025004

[31] Van Loan, C. F., Tensor network computations in quantum chemistry Technical report, available online at www.cs.cornell.edu/cv/OtherPdf/ZeuthenCVL.pdf, **2008**.

[32] Oseledets, I., Tensor-Train Decomposition. In *SIAM J. Scientific Computing. 33.*, **2011**, pp. 2295-2317. doi: 10.1137/090752286.

[33] Lindstrom, P., Fixed-Rate Compressed Floating-Point Arrays. In *IEEE Transactions on Visualization and Computer Graphics vol. 20*; **2014**, doi :10.1109/TVCG.2014.2346458.

[34] Lemley, J., Deep Learning for Consumer Devices and Services: Pushing the limits for machine learning, artificial intelligence, and computer vision. In *IEEE Consumer Electronics Magazine vol. 6, Iss. 2*; **2017** doi: 10.1109/MCE.2016.2640698

[35] Krizhevsky, A., Sutskever, I., Hinton, G. E., ImageNet classification with deep convolutional neural networks. In *Communications of the ACM. 60 (6)* pp. 84–90. doi: 10.1145/3065386

[36] Simonyan, K., Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *arXiv preprint arXiv:1409.1556*. **2014**

[37] He, Kaiming, et al. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* **2016**, pp. 770-778 doi: 10.1109/CVPR.2016.90

[38] Krizhevsky, A., et al., ImageNet classification with deep convolutional neural networks. In *Proc. 25th Int. Conf. Neural Inf. Process. Syst. (NIPS), vol. 1.*, Red Hook, NY, USA: Curran Associates, **2012**, pp. 1097–1105. doi: 10.1145/3065386

[39] Simonyan K. and Zisserman A., Very deep convolutional networks for large-scale image recognition. In *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, Y. Bengio and Y. LeCun, Eds., **2015**, pp. 1–14.

[40] Xie S., et al., Aggregated residual transformations for deep neural networks.In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, **2017**, pp. 5987–5995. doi: 10.1109/CVPR.2017.634

[41] Szegedy, S. et al., Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, S. P. Singh and S. Markovitch, Eds., **2017**, pp. 4278–4284.

[42] Tan, M., et al. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* **2019**, pp. 2820-2828 doi: 10.1109/CVPR.2019.00293

[43] Kossaifi, J.; Panagakis, Y.; Kumar, A.; Pantic, M. TensorLy: Tensor Learning in Python. *arXiv preprint* **2018**, arXiv:1610.09555.

[44] Howard, J., imagenette dataset, https://github.com/fastai/imagenette/

[45] Oseledets, I. V., Tensor-train decomposition. In *SIAM J. Sci. Comput., vol. 33, no. 5*, **2011**, pp. 2295–2317 doi: 10.1137/090752286