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

Jakub Grabek*† and Boguslaw 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 compression 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 diversified 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
The multidimensional compression can be based on the following tensor product

$$\hat{T} = T \times_1 F_1 \times_2 F_2 \times_p F_p \quad (1)$$

where decompressed version of input tensor $T$ is denoted as $\hat{T}$, whereas $F_i$ is the $i$th factor matrix. The key idea is that the set of factor matrices and their product with $T$, from the right side of (1), require much less storage space than the original tensor $T$, while its recovered version $\hat{T}$ 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 $T \times_k M$ of a tensor $T \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_p}$ and a matrix $M \in \mathbb{R}^{Q \times N_k}$ is used. The result is also a tensor $S \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_{k-1} \times Q \times N_{k+1} \times \ldots \times N_p}$, whose elements are expressed as follows:

$$S_{n_1 n_2 \ldots n_k-1 q n_{k+1} \ldots n_p} = (T \times_k M)_{n_1 n_2 \ldots n_{k-1} q n_{k+1} \ldots n_p} = \sum_{n_k=1}^{N_k} t_{n_1 n_2 \ldots n_{k-1} q n_{k+1} \ldots n_p} m_{n_k} \quad (2)$$

As shown below, the compression matrices $F_i$, 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 $T$ is done by calculation of an approximating tensor $\hat{T}$ that is close to the input tensor in terms of the Frobenius norm. Hence, a minimization function is defined as follows

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

A. Tucker-based methods

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

$$\hat{T} = Z \times_1 S_1 \times_2 S_2 \times_3 \ldots \times_p S_p \quad (4)$$

where $Z \in \mathbb{R}^{R_1 \times R_2 \times \ldots \times R_p}$ is a core tensor and $S_i \in \mathbb{R}^{N_i \times R_i}$ are the so-called mode matrices. Using algebraic operations, from Equation (4), the formula for the core tensor is obtained:

$$Z = \hat{T} \times_1 S_1^T \times_2 S_2^T \times_3 \ldots \times_p S_p^T \quad (5)$$

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

$$\Theta(\hat{T}) = ||\hat{T} - T \prod_{k=1}^P \times_k (S_k S_k^T)||_F^2 \quad (6)$$

The Tucker decomposition in Equation (6) reads that a tensor $T$ is approximated by its projection onto space spanned by the matrices $S_k$. To compute the series of $S_k$ matrices, the alternating method can be used [2], [21], [24]–[26]. Also, let’s observe that $S_k S_k^T$ in (6) is equivalent to the factor matrix $F_i$ from (1).
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 (GPPU).

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 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: \( \text{load original image } I_i \text{ and prepare container for resized} \)
4: \( \text{image } I_p \)
5: \( \text{calculate } x \text{ and } y \text{ offsets needed to place } I_i \text{ in the center} \)
6: \( \text{of } I_p \)
7: \( \text{resized selected image from } I \text{ to dimensions specified} \)
8: \( \text{in } W \text{ and } H \text{, keeping aspect ratio of original data and} \)
9: \( \text{save it to } I_p \)
10: \( \text{append } I_p \text{ to } T \)
11: \( \text{remove } I_i \text{ from } I \)
12: end while
13: return \( T \)

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

\[
Cr = \frac{\text{Uncompressed data size}}{\text{Compressed data size}}
\]  

(8)

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

\[
Acc = \frac{TP + TN}{TP + TN + FP + FN}
\]  

(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 \([1, 0.5\min(W, H), 0.5\min(W, H), 1]\). 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Tensor compression type</th>
<th>ZFP tolerance</th>
<th>Validation set compressed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>original</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>B</td>
<td>best_rank</td>
<td>Best rank</td>
<td>lossless</td>
</tr>
<tr>
<td>C</td>
<td>best_rank_comp</td>
<td>Best rank</td>
<td>1e-3</td>
</tr>
<tr>
<td>D</td>
<td>best_rank_orig_val</td>
<td>Best rank</td>
<td>lossless</td>
</tr>
<tr>
<td>E</td>
<td>best_rank_comp_orig_val</td>
<td>Best rank</td>
<td>1e-3</td>
</tr>
<tr>
<td>F</td>
<td>tensor_train</td>
<td>Tensor train</td>
<td>lossless</td>
</tr>
<tr>
<td>G</td>
<td>tensor_train_comp</td>
<td>Tensor train</td>
<td>1e-3</td>
</tr>
<tr>
<td>H</td>
<td>tensor_train_orig_val</td>
<td>Tensor train</td>
<td>lossless</td>
</tr>
<tr>
<td>I</td>
<td>tensor_train_orig_val</td>
<td>Tensor train</td>
<td>1e-3</td>
</tr>
</tbody>
</table>

TABLE II
COMPRESSION RESULTS

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

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)

<table>
<thead>
<tr>
<th></th>
<th>AlexNet</th>
<th>ResNet-18</th>
<th>ResNet-34</th>
<th>VGG-11</th>
<th>VGG-13</th>
<th>MNASNet0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.7687</td>
<td>0.8025</td>
<td>0.7962</td>
<td>0.8018</td>
<td>0.7806</td>
<td>0.7389</td>
</tr>
<tr>
<td>B</td>
<td>0.6497</td>
<td>0.6683</td>
<td>0.6581</td>
<td>0.6838</td>
<td>0.6715</td>
<td>0.6033</td>
</tr>
<tr>
<td>C</td>
<td>0.6555</td>
<td>0.7304</td>
<td>0.7139</td>
<td>0.6945</td>
<td>0.6854</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0.6482</td>
<td>0.7081</td>
<td>0.6851</td>
<td>0.6566</td>
<td>0.6191</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>0.6662</td>
<td>0.693</td>
<td>0.6856</td>
<td>0.6823</td>
<td>0.6741</td>
<td>0.6321</td>
</tr>
<tr>
<td>F</td>
<td>0.7093</td>
<td>0.7554</td>
<td>0.7605</td>
<td>0.7422</td>
<td>0.7228</td>
<td>0.6652</td>
</tr>
<tr>
<td>G</td>
<td>0.7271</td>
<td>0.7521</td>
<td>0.7475</td>
<td>0.7427</td>
<td>0.7338</td>
<td>0.681</td>
</tr>
<tr>
<td>H</td>
<td>0.7309</td>
<td>0.7493</td>
<td>0.7623</td>
<td>0.7417</td>
<td>0.7335</td>
<td>0.6795</td>
</tr>
<tr>
<td>I</td>
<td>0.7284</td>
<td>0.7439</td>
<td>0.7475</td>
<td>0.7483</td>
<td>0.7264</td>
<td>0.6561</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th></th>
<th>AlexNet</th>
<th>ResNet-18</th>
<th>ResNet-34</th>
<th>VGG-11</th>
<th>VGG-13</th>
<th>MNASNet0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.7681</td>
<td>0.8025</td>
<td>0.7962</td>
<td>0.8018</td>
<td>0.7806</td>
<td>0.7389</td>
</tr>
<tr>
<td>B</td>
<td>0.6947</td>
<td>0.6683</td>
<td>0.6581</td>
<td>0.6838</td>
<td>0.6715</td>
<td>0.6033</td>
</tr>
<tr>
<td>C</td>
<td>0.6555</td>
<td>0.7304</td>
<td>0.7139</td>
<td>0.6945</td>
<td>0.6854</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0.6482</td>
<td>0.7081</td>
<td>0.6851</td>
<td>0.6566</td>
<td>0.6191</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>0.6662</td>
<td>0.693</td>
<td>0.6856</td>
<td>0.6823</td>
<td>0.6741</td>
<td>0.6321</td>
</tr>
<tr>
<td>F</td>
<td>0.7093</td>
<td>0.7554</td>
<td>0.7605</td>
<td>0.7422</td>
<td>0.7228</td>
<td>0.6652</td>
</tr>
<tr>
<td>G</td>
<td>0.7271</td>
<td>0.7521</td>
<td>0.7475</td>
<td>0.7427</td>
<td>0.7338</td>
<td>0.681</td>
</tr>
<tr>
<td>H</td>
<td>0.7309</td>
<td>0.7493</td>
<td>0.7623</td>
<td>0.7417</td>
<td>0.7335</td>
<td>0.6795</td>
</tr>
<tr>
<td>I</td>
<td>0.7284</td>
<td>0.7439</td>
<td>0.7475</td>
<td>0.7483</td>
<td>0.7264</td>
<td>0.6561</td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

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

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


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