Guaranteed Compression Rate for Activations in CNNs using a Frequency Pruning Approach

Sebastian Vogel*, Christoph Schorn*, Andre Guntoro*, Gerd Ascheid†
*Robert Bosch GmbH, Renningen, Germany
†RWTH Aachen University, Aachen, Germany
{sebastian.vogel, christoph.schorn, andre.guntoro}@de.bosch.com, gerd.ascheid@ice.rwth-aachen.de

***Abstract*—Convolutional Neural Networks have become state of the art for many computer vision tasks. However, the size of Neural Networks prevents their application in resource constrained systems. In this work, we present a lossy compression technique for intermediate results of Convolutional Neural Networks. The proposed method offers guaranteed compression rates and additionally adapts to performance requirements. Our experiments with networks for classification and semantic segmentation show, that our method outperforms state-of-the-art compression techniques used in CNN accelerators.**

***Index Terms*—Deep Neural Networks, Convolutional Neural Networks, Compression, Embedded Systems**

I. Introduction

In recent years, Convolutional Neural Networks (CNNs) have shown remarkable success in computer vision tasks, such as image classification, object detection, and semantic segmentation. On one side, CNN are developed with ever increasing size and performance [1]. On the other side, research efforts are made to reduce their massive resource requirements with novel architectural approaches [2] or quantization methods [3]. However, most of the proposed methods reduce the computational workload as well as the size of network parameters but do not dedicate their technique to intermediate results of neural networks.

In this paper, we propose a technique to compress intermediate results of a neural network using a lossy approach. While processing a CNN our method tiles intermediate data into small chunks of data which are transformed into the frequency domain by using a Discrete Cosine Transformation (DCT). Some of the values in the frequency domain are pruned before storing the transformed data chunk to memory.

With this paper we make the following contributions:

- We propose a novel lossy compression technique for intermediate data of CNNs.
- With guaranteed compression rates, our approach can countervail high memory bus utilization in critical real-time systems.
- We propose an optimization technique to adapt the pruning rate of each layer.
- We evaluate the proposed technique on a variety of CNNs and data sets and compare it to state-of-the-art techniques deployed in CNN accelerators.

Convolutional Neural Networks typically incorporate regular structures of layered convolution operations, pooling or upsampling functions, and possibly fully-connected layers. The underlying mathematical expression for convolutional as well as fully-connected layers is a weighted sum of input values. Subsequently a bias is added followed by a nonlinearity function. The resulting output y of a layer can hence be written as

$$y = \Phi\left(b + \sum xw\right). \tag{1}$$

In most modern CNNs the rectified linear unit (ReLU) is used as a nonlinearity function. ReLU clips negative values to 0 and acts as the identity function for positive values. To reduce the network size towards the output, pooling functions can be implemented. A popular approach is the maximum pooling which chooses the highest value out of a number of neighboring values.

II. Related Work

Recent advances in research propose various approaches to enhance performance of CNNs even further. Some are based on an increasing number of network layers [4] others additionally rely on a growing number of parameters [1]. However, with increasing network size the compute and memory requirements rise as well. Therefore, countermeasures have beeen proposed based on novel network architectures, e.g. CNNs with depthwise separable convolutions [2]. Other proposals reduce the size of parameters, e.g. by quantizing the network to a reduced bit width [3], [5].

Some compression approaches require specialized hardware modules in order to fully benefit from the proposed method. Weight pruning leads to sparse matrices which thus can be compressed in memory. Therefore, a dedicated compression engine enables loading compressed parameters from memory. Hardware accelerators with dedicated sparse matrix-matrix-multiplication engines can even profit by the matrix sparsity to reduce computation time without decompressing parameters [6]. However, additional information has to be stored in order to code the exact position of zeros and non-zeros respectively by using a sparsity map encoding (SPM) [7]. Furthermore, when using such an approach for intermediate data of CNNs, the actual compression rate depends on the input data and hence underlies a stochastic process. Another approach is to store neighboring zeros in a Run Length Encoding (RLE) format. Such technique is employed for storing

978-3-9819263-2-3/DATE19/©2019 EDAA

intermediate values in the Eyeriss accelerator [8]. Again this cannot provide a guaranteed compression rate.

Using our method, no additional information such as run-length encodings or sparsity maps have to be stored. Also, as we prune a predefined amount of data in the frequency domain, our technique results in a guaranteed compression rate. We evaluate the impact on classification performance of our lossy frequency pruning approach. To the best of our knowledge, we are the first to propose and evaluate a lossy compression technique for intermediate data in CNNs.

III. Analysis and Compression Methods

A. Analysis

While recent advances in neural network architectures reduce the amount of network parameters compared to earlier network designs (compare parameter size of VGG16 [9] versus MobileNet in Table I), the amount of intermediate data does not reduce in the same order. In fact, when increasing the input data size from rather small images (e.g. 224×224 as for VGG16 and MobileNet for ImageNet [10] classification) to high-resolution images typically used in applications such as object detection or semantic segmentation, the amount of intermediate data surpasses that of network parameters by far (compare ratio of weights to activations in Table I). The FCN8s [11] network used for semantic segmentation works at an input resolution of 2048×1024 at three color channels. The Dilated Model [12] uses an input resolution of 512×256. Both networks are evaluated on the Cityscapes data set [13].

To understand the size of intermediate data of neural networks in more detail Fig. 1 shows the distribution of data per layer in four different CNNs. As can be seen in Fig. 1, especially the first layers contain most of the intermediate data. Therefore, compression techniques are most effective when employed in the first layers.

The CNNs we evaluate in this paper have been quantized to 16 bit. Therefore, weights as well as activations are represented by fixed-point numbers with 16 bit.

B. Run-Length Encoded Compression

Most CNNs exhibit the ReLU activation function. Hence, a nonnegligible amount of activation values is zero. A special coding for zeros can help compress the size of data to be stored in Random Access Memory (RAM). To make use of such a compression, the authors in [8] propose to compress zero values in a 5 bit run and store nonzero values in 16 bit. Three run and value (level) pairs are stored into a 64 bit word,

TABLE I
For models with high input resolution (FCN8s, Dilated Model) the main data load emerges from intermediate data. The numbers are based on 16-bit-quantized CNNs.

model	size (weights / activations)	ratio
VGG16 [9]	268 MB / 36.1 MB	7.43
MobileNet [2]	8.44 MB / 6.22 MB	1.36
FCN8s [11]	268 MB / 1.60 GB	0.16
Dilated Model [12]	10.6 MB / 139 MB	0.07

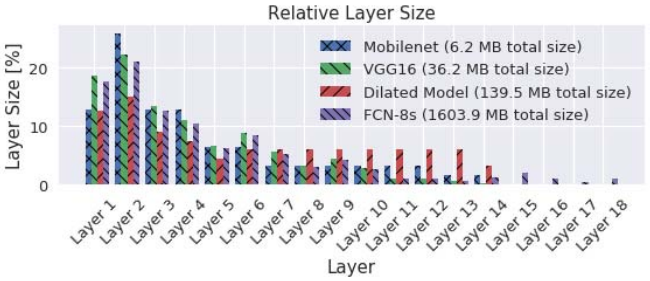

Fig. 1. The relative size of intermediate data of each layer is shown for a variety of neural networks. It can be seen that the first layers which typically extract low-level features incorporate most of the neural network size. Therefore, compression is most effective if applied to these layers.

where the final bit indicates, whether this run is the last. In case of neighboring nonzero values, a run of length 0 has to be stored. The authors use this technique in their hardware accelerator to reduce the amount of data stored in external memory. We use this technique to evaluate the effectiveness of RLE against our approach.

C. Sparsity Map Based Compression

Another way to compress intermediate data is to store only nonzero values and their respective position. This principle has been proposed and evaluated by the authors of [7]. Their hardware accelerator compresses and decompresses intermediate data before transferring it to memory. First, all nonzero values of a given data chunk are extracted. Then their respective position is coded in a so called sparsity map (SPM). Each bit in the SPM corresponds to a nonzero value.

D. Frequency Pruning Approach

We were inspired by the JPEG-compression principle of pruning high-frequency components of small tiles in images to reduce the data footprint. While humans are still able to recognize the content of images after JPEG-compression, we expect CNNs to be resilient to the lossy compression also.

Our frequency pruning approach works as depicted in Fig. 2. First, each feature map of a CNN is tiled into quadratic small data chunks of size 8×8, or 7×7 where applicable. Then, each tile is transformed into the frequency domain using a DCT. To reduce data size, we prune some of the frequency components before transferring data to memory. When data is read back from memory to be processed in a later stage, tiles are filled up with zeros and retransformed from the frequency domain using Inverse Discrete Cosine Transformation (IDCT).

In a first setting, we prune each convolutional layer of a neural network in the same way. I.e. each feature map is tiled, each tile is transformed into the frequency domain and the pruning mask (see Fig. 2) of all layers is identically carrying zeros in the high-frequency region. In Fig. 2, we plot the high-frequency region of a 2D-DCT-transformed matrix in the lower right corner. Using this approach, a potential hardware compression module does not require layerwise configuration it is rather configured once per inference of a neural network.

Such a compression module reduces the data size required for each tile. We call this pruning "direct compression".

Nevertheless, the first layers of CNNs contain more data and therefore contribute to most of the workload in terms of data transfers. In fact, our method can also be used with different pruning masks for each layer. Therefore, we propose to adapt the pruning mask of each layer separately. To do so, we implement an elementwise multiplication of the frequency component tiles with a so called compression matrix A (see (2)). The elements a_{ij} of A are initialized with value 1. To compress a pre-trained network, we fix all network parameters and do not change their values during the optimization of the attenuation matrix A.

$$A \in \{a^{n \times n} | a \in [0, 1]\} \tag{2}$$

We use the backpropagation algorithm to learn the values of A. To focus only on the important frequency components, we extend the target loss function with an L2-regularizing term:

$$0.0001 \cdot \|A\|_2 \tag{3}$$

We found the weighting of 10^-4 to be suitable. Using this optimization method, the elements of A are trained to the needs of the network regarding its task while weighting the frequency components of each tile. After optimization, we prune values of a_{ij} which are smaller than a given threshold a_th. All remaining values are set to 1. Hence, a compression module prunes some predefined frequency components and stores the nonpruned values into memory. No multiplication has to take place additionally. This adaptive compression principle is depicted in Fig. 3.

The resulting compression matrices $A^{(l)}$ of a layer (l) are adapted to the application of the neural network and therefore only take the nonredundant frequency components into account. The compression module of a potential hardware accelerator is configured layerwise based on the required compression rate. Especially in Systems on Chip (SoC) a CNN accelerator IP might encounter high bus utilization. Therefore, a dynamic compression module for intermediate data with guaranteed reduction rate is beneficial.

IV. Evaluation

We evaluate the compression techniques proposed in [7] and [8] on MobileNet. As can be seen in Fig. 4, the compression based on sparsity maps outperforms that of Run Length Encoding in terms of data reduction. However, both methods

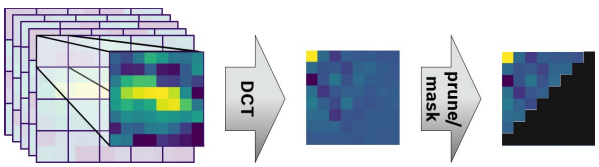

Fig. 2. Our compression method is based on the tiling of intermediate results into small data chunks of quadratic tiles. These tiles are transformed into the frequency domain using a DCT. High-frequency components are pruned before transferred to memory.

Fig. 3. Trained elements of the compression matrices below a threshold a_{th} are pruned whereas the remaining values are set to 1.

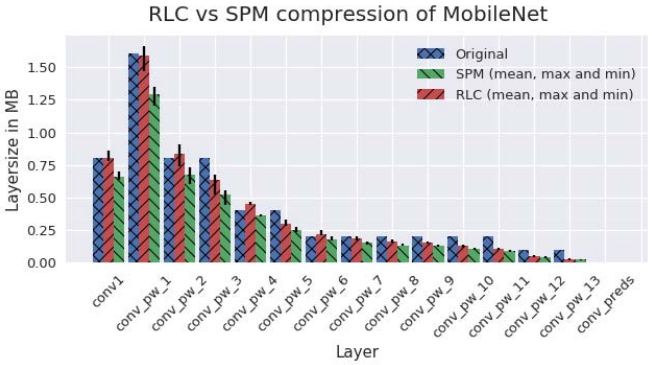

Fig. 4. We evaluated the compression principles based on Run Length Encoding (RLE) as proposed in [8] and based on Sparsity Maps (SPM) as proposed in [7] on MobileNet. The original layer size as well as the resulting layer size after compression can be seen.

incorporate a stochastic nature and therefore no compression rate can be guaranteed. Especially in safety-critical real-time systems where the worst case bus utilization has to be taken into account, these methods may increase the worst case amount of data when compared to the original data size. This effect can be seen for layers 1, 2, 3, 5, and 7.

Also, especially the first layers are not as sparse as deeper layers in the network (compare relative data reduction of SPM in Fig. 7). However, the first layers make up most of the size of the network. Thus compression techniques based on sparsity are not as effective.

To evaluate the direct compression approach we report the resulting performance of MobileNet and VGG16 when each layer is compressed identically. All convolutional layers of VGG16 as well as of MobileNet are frequency pruned with

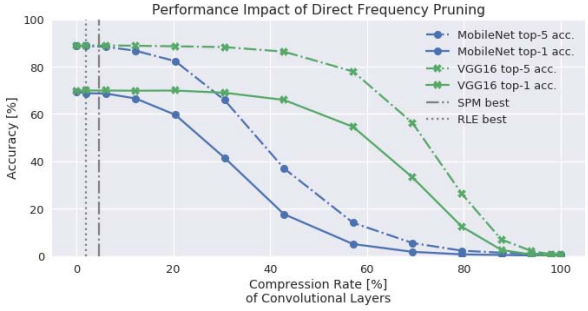

Fig. 5. Performance versus compression rate of MobileNet and VGG16 using our frequency pruning approach. VGG16 is resilient to our lossy pruning approach. MobileNet compression surpasses RLE or SPM with little reduction in performance.

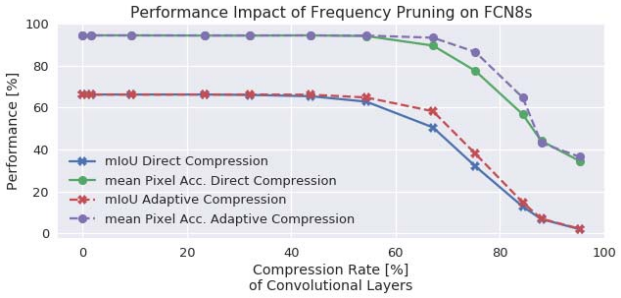

Fig. 6. Performance versus compression rate of FCN8s for direct compression and adaptive compression. As can bee seen, the adaptive compression leads to higher compression rates at less performance drop.

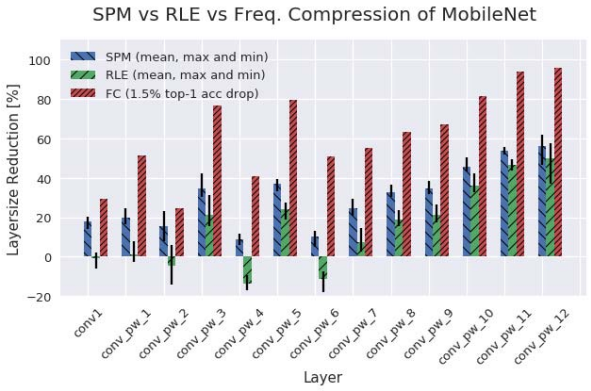

Fig. 7. Three compression methods are compared in terms of portion of data compressed in each layer. Higher is better.

tile size 8×8, or 7×7 where applicable. In Fig. 5 the top-1 accuracy as well as the top-5 accuracy are depicted for different compression rates achieved by pruning high-frequency components. In our experiments, we start pruning the lower-right values of frequency transformed tiles. With increasing compression rate, the resulting compressed tile contains zeros in the lower-right triangular part. Fig. 5 shows that VGG16 can be compressed almost $3\times$ more compared to MobileNet at the same performance. The best compression rates of RLE and SPM are shown as vertical lines. Frequency pruning outperforms these approaches with little to no degradation in classification performance.

The results of our adaptive compression approach evaluated on MobileNet and VGG16 are depicted in Fig. 7. Three compression methods are compared in terms of amount of data compressed in each layer. As can be seen, in some layers RLE actually increases the amount of data. Especially in the first layers, RLE and sparsity maps do not achieve high compression rates. With an allowed drop of top-1 accuracy of 1.5 %, frequency compression (FC) outperforms the other compression principles in each layer by far. Please note that the compression rates reported for FC are guaranteed.

Finally, we report the results of our proposed method on the FCN8s network. Fig. 6 shows the results of the direct compression approach compared to the adaptive procedure. The FCN8s network with an input resolution of 2048×1024 at RGB can be compressed up to 67.19% with a drop in mean Intersection-over-Union (mIoU) of 8.1 % for the adaptive frequency pruning. This corresponds to a data reduction by 524 MB (compare Table I) of intermediate data at 16 bit quantization. In case of direct compression the performance drops by 16.3% for the same data reduction rate. Compared to the direct procedure, the adaptive frequency pruning approach achieves a slightly higher overall compression rate.

V. Conclusion and Outlook

In this paper we presented a compression technique for intermediate data of CNNs. In contrast to previously proposed compression methods such as Run Length Encoding or sparsity maps, our technique allows for guaranteed compression rates. Our method prunes frequency components of intermediate data tiles and hence incorporates a lossy behavior. However, our analysis shows that CNNs a resilient to this kind of pruning to a great extent. To the best of our knowledge we are the first to evaluate lossy compression techniques for intermediate data of CNNs.

References

[1] Y. Huang *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *CoRR*, 2018. [Online]. Available: https://arxiv.org/abs/1811.06965

[2] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[3] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[5] S. Vogel, M. Liang, A. Guntoro, W. Stechele, and G. Ascheid, "Efficient hardware acceleration of cnns using logarithmic data representation with arbitrary log-base," in *International Conference On Computer Aided Design (ICCAD)*, San Diego, 2018.

[6] S. Han *et al.*, "Eie: Efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 243–254.

[7] A. Aimar *et al.*, "Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *CoRR*, vol. abs/1706.01406, 2017. [Online]. Available: http://arxiv.org/abs/1706.01406

[8] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, jan 2017.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[10] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, apr 2015.

[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3431–3440.

[12] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015. [Online]. Available: http://arxiv.org/abs/1511.07122

[13] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.