At the core of any inference procedure in deep neural networks are dot product operations, which are the component that require the highest computational resources. One common approach to reduce the complexity of these operations is to prune and/or quantize the weight matrices of the neural network and to employ data structures, e.g., sparse matrix data structures, that can efficiently exploit the statistics present in the matrix. However, most of these common data structures make strong statistical assumptions about the distribution of the elements in the matrix, which may not always be satisfied, in particular, when we consider the entire set of highly compressible matrices.

We present new efficient data structures for representing matrices that exhibit low entropy statistics relative to the maximum likelihood estimates of the probability mass distribution of their elements. We show that these formats are especially suitable for representing neural networks that have been compressed using state-of-the-art lossy compression techniques. We show that the proposed data structures can not only be regarded as a generalization of sparse formats, but are also more energy and time efficient under practically relevant assumptions. For instance, we experimentally show that we are able to attain up to x16 compression ratios, x1.7 speed ups and x20 energy savings when we lossless convert state-of-the-art networks such as AlexNet, VGG-16, ResNet152 and DenseNet into the new data structures.

1 Introduction

Deep neural networks [9][14] nowadays became the state-of-the-art in many fields of machine learning. However, most deep neural network models require the computation of many dot product operations between large matrices when performing inference. This requires a great amount of computational resources which difficult or even prohibit their deployment into resource constrained devices.

This fact motivated an entire research field of model compression [1] that aimed to reduce the complexity of inference of deep neural networks. For instance, some of the most popular techniques include sparsification of the network [6,8,7,13] or quantization of the weight elements [17][11]. These approaches aim to reduce both storage and execution complexity of the dot product operations involved in the inference procedure, by either leveraging on the sparsity of the weight matrices, e.g., by representing them in one of the sparse data structures and performing the dot products accordingly, or by converting the element values into low bit-length numerical representations. However, highest compression ratios and execution efficiencies can be achieved when one employs compression schemes that either implicitly [6,10,12] or explicitly [2,16,4] aim to minimize the entropy of the weight elements relative to the maximum likelihood estimate of their probability mass distribution.
However, weight matrices that exhibit low entropy statistics are not necessarily sparse, neither the cardinality of the elements has to be low. For instance, figure 1 plots the distribution of the weight elements of the last classification layer of VGG-16 [15] (1000 × 4096 dimensional matrix), after having applied uniform quantization on the weight elements. We stress that the prediction accuracy and generalization of the network was not affected by this operation. As we can see, on the one hand, the distribution of the compressed layer does not satisfy the sparsity assumption, i.e., there is not one particular element (such as 0) that appears specially frequent in the matrix. On the other hand, although the cardinality of the weight elements is in this case smaller than the original size (7 bits as opposed to 32), one can not efficiently exploit this property with regards to the dot product operation on most conventional hardware, since they do not support multiplications on mixed-precision numerical representations. Moreover, the entropy of the matrix is about $H = 5.5$, which is smaller than 7 bits, thus indicating that even higher efficiency gains may be obtained. Although [6] proposed an optimal entropy coder for representing these type of matrices, specialized hardware is needed in order to efficiently run inference under the resulting representation [5]. Hence, under our knowledge, there is currently no simple representation that can be applied on this type of matrices, that is able to simultaneously achieve high compression ratios and low computational complexity relative to the dot product operation.

2 The compressed entropy row (CER) and compressed shared weight row (CSER) representations

Notice, that we expect the ratio $\bar{k}/n$ between the average number of unique elements $\bar{k}$ that appear in a particular row (or column) and the respective number of columns (or rows) $n$ to be low, for matrices that exhibit low entropy statistics. The intuition behind this is that we can interpret the number $2^H$ as the effective number of unique elements that a random variable with entropy value $H$ outputs. Indeed, only 15 distinct values dominate the entries of the weight matrix displayed in figure 1, which is only 1.5% of the number of columns of the matrix. Hence, in this work, we propose two data structures that exploit this property, which we named compressed shared weight row (CSER) and compressed entropy row (CER). Both apply the same principles, but differ only slightly on their statistical assumptions.

The CER and CSER representations are able to save significant amounts of storage requirements by leveraging on the fact that we only need to store the unique elements that appear on the matrix rows once. In contrast, although sparse data structures already implicitly apply this technique for the most frequent element (namely, the 0 value), they store the other values multiple times (the non zero entries), inducing high redundancies in their final representations. Moreover, we can design efficient dot product algorithm by implicitly encoding the distributive law of multiplications in the CER and CSER representations. By incorporating the distributive law we can not only save a great number of multiplications, but also a great number of expensive read/write operations associated to them.
Table 1: Storage, number of operations, time and energy gains of performing a matrix-vector multiplication of different state-of-the-art neural networks, after their weight matrices have been compressed down to 7 bits and, subsequently, converted into the different data structures. The performance gains are relative to the dense representation. The accuracy is measured with regard to the validation set (in parenthesis we show the accuracy of the uncompressed model).

<table>
<thead>
<tr>
<th>Size [MB], #ops [G], time [s], energy [J]</th>
<th>Accuracy [%]</th>
<th>dense</th>
<th>CSR</th>
<th>CER</th>
<th>CSER</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>68.51 (68.71)</td>
<td>553.43, 15.08</td>
<td>x0.71, x0.88</td>
<td>x2.11, x1.40</td>
<td>x2.11, x1.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.37, 2.70</td>
<td>x0.85, x0.76</td>
<td>x1.27, x2.37</td>
<td>x1.29, x2.38</td>
</tr>
<tr>
<td>ResNet152</td>
<td>78.17 (78.25)</td>
<td>240.77, 10.08</td>
<td>x0.76, x0.93</td>
<td>x2.08, x1.42</td>
<td>x2.10, x1.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.00, 1.92</td>
<td>x0.93, x1.25</td>
<td>x1.30, x3.73</td>
<td>x1.31, x3.74</td>
</tr>
<tr>
<td>DenseNet</td>
<td>77.09 (77.12)</td>
<td>114.72, 7.14</td>
<td>x1.04, x1.11</td>
<td>x2.74, x1.66</td>
<td>x2.79, x1.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.53, 0.51</td>
<td>x1.10, x1.95</td>
<td>x1.43, x6.40</td>
<td>x1.45, x6.57</td>
</tr>
</tbody>
</table>

The particular encoding/decoding procedure as well as the associated dot product pseudocodes are described more extensively in our main manuscript [18].

Concretely, we can show that the storage requirement as well as the dot product complexity (as measured relative to each element in the matrix) of the CSER and CER representations asymptotically fulfill

\[
\text{complexity} = O((1 - p_0) \log_2 n) + O\left(\frac{k}{n}\right) + O\left(\frac{1}{n}\right)
\]

where \(p_0\) is the maximum likelihood estimate of the probability mass distribution of the most frequent value, \(n\) the row size and \(k\) the average number of unique elements that appear in a row. Hence, equation (1) states that the efficiency of the CER and CSER representations depend partially on the sparsity level of the matrix (the term \((1 - p_0)\)) and the per row cardinality of it’s elements (the term \(k\)). Consequently, the CER and CSER representations will result in being particularly efficient when both terms are simultaneously minimized. Notice, that both terms decrease as the entropy of the matrix statistics decreases. Also, as we mentioned in the introduction, most frameworks that achieve state-of-the-art compression ratios aim to directly minimize these two components simultaneously [6, 10, 12, 2, 16, 4]. In addition, (1) also states that the CER and CSER data structures can efficiently exploit a wider set of matrices, in particular those matrices whose element statistics fail to be efficiently executed by the dense and sparse representations. Finally, we would like to stress once more that the CER and CSER data structures do not require specialized hardware in order to run efficiently, as long as their complexity (1) is sufficiently small.

3 Experiments

We applied our representations on compressed weight matrices of deep neural networks and benchmarked their efficiency with regards to 4 metrics: 1) the storage requirement, 2) the total number of operations needed to perform a matrix-vector multiplication, 3) the respective time complexity and 4) the respective energy consumption. We run all our experiments on a conventional CPU hardware.

We were able to attain up to x16 compression ratios, x1.7 speed ups and x20 energy savings (compared to the original dense representation) after converting the weight matrices of the by [6] compressed AlexNet model into the CER and CSER representations. These are significant gains compared to, e.g., the common CSR sparse format which attained x6, x1.7 and x5 gains respectively. We also benchmarked our data structures on networks that were compressed using lossy compression techniques that do not require to retrain the network in the process. This case is of particular interest, since in many real world scenarios one may have a full-sized model but no access to the training data. Also, in most cases, the statistics of the compressed weight matrices do not fulfill the common prior assumptions of most conventional data structures in such scenario (e.g., figure [1]). In contrast, the CER and CSER are able to exploit the statistical properties present, and consequently attain higher gains in efficiency throughout all four above mentioned benchmarks (see table [1]).

References


