Reduction of DNNs in edge computing

Asier Garmendia-Orbegozo (asier.garmendiao@ehu.eus)
University of the Basque Country
Jose David Nuñez-Gonzalez
University of the Basque Country
Miguel Angel Anton
Tecnalia Basque Research and Technology Alliance (BRTA)

Article

Keywords:

Posted Date: July 27th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1862445/v1

Reduction of DNNs in edge computing

Asier Garmendia-Orbegozo[1,*], Miguel Angel Anton[2], and J. David Nuñez-Gonzalez[1]

[1]University of the Basque Country UPV/EHU, Department of Applied Mathematics, Eibar, 20600, Spain
[2]TECNALIA, Basque Research and Technology Alliance (BRTA), San Sebastian, 20009, Spain
[*]asier.garmendiao@ehu.eus

ABSTRACT

Application of Deep Neural Networks (DNN) in Edge Computing has emerged as a consequence of the need of real time and distributed response of different devices in a large number of scenarios. For this end, shredding these original structures is urgent due to the high number of parameters needed to represent them. As a consequence, the most representative components of different layers are kept in order to maintain the network's accuracy as close as possible from the entire network's ones. To do so, two different approaches have been developed in this work. First, Sparse Low Rank Method (SLR) has been applied to two different Fully Connected (FC) layers to watch their effect on the final response, and similarly the method has been applied to one of these layers. In the contrary, in this last case, the relevances of the previous FC layer's components were considered taking into account the connections to each of the components from the other FC layer, considering the relationship of relevances across layers. Experiments had been carried out in well-known architectures to conclude whether the relevances throughout layers have less effect on the final response of the network than the independent relevances intra-layer.

Introduction

The use of Deep Neural Networks (DNN) in different scenarios related to Machine Learning (ML) applications has developed in such a way that nowadays neural network designs have billions of parameters with great capability of prediction, being one of the most used type of architecture in prediction tasks. Specifically, some of those applications include image, sound and textual data recognition. In contrast with other of ML algorithms the DNNs has achieved a remarkable accuracy. However, the use of these networks in memory and processing resource constrained devices is limited due to the amount of data needed to develop these architectures and the high computational costs at training them. Consequently, different reduction techniques are essential to fit these former networks in resource constrained devices, such as edge devices.

Among others, the most used and effective way to shrink these networks is the use of techniques such as pruning and quantization. The former one consists in removing parameters (neurons or weights) that have negligible contribution while maintaining the accuracy of the classifier. On the other hand, quantization involves replacing datatypes to reduced width datatypes, by transforming data to fit in new datatypes' shapes. By this way, reduced networks are able to compete with the original ones in terms of accuracy, even improving these in some cases in which overfitting issues were hindering their predictability. Moreover, by reducing the width of data edge devices could face the storage issue mentioned above and collect larger datasets in constrained memory sizes.

Specially Convolutional Neural Networks (CNN) became a widely used network structure in image recognition tasks. Such a success is built upon a large number of model parameters and convolutional operations. As a result, the huge storage and computation cost make these models difficult to be deployed on resource-constrained devices, such as phones and robots, needing to adopt different reduction techniques.

In this work we introduce a new method to develop weight pruning in well-known architectures. We adopt the posture that the last Fully Connected (FC) Layer, Final Response Layer (FRL), is the most relevant to the final decision. Moreover, the relevance of weights of this final layer are propagated to the previous layers, making non-independent each neuron of the previous layers in terms of relevance. Consequently, the connections of each neuron have a direct relationship with their predictability in the final decision of the network, needing to consider them. After factorizing the weight matrices of FC Layers, we sparsified them only considering the most relevant parts and propagate this relevance to the previous FC layers by considering the connections between different FC layers. Similarly, we performed a parallel process in which the sparsification of matrices had been carried out independently between layers, only considering the relevance intra-layer. Finally, we state the validity of the supposition of backpropagating the relevance within layers.

State of the Art

There have been several attempts to reduce DNNs dimensionality by applying techniques mentioned above. Pruning techniques consist in removing part of connections(weights) or neurons from the original network so as to reduce the dimension of the original structure by maintaining its ability to predict. The core of these techniques reside on the redundancy that some elements add to the entire architecture. Memory size and bandwidth reduction are addressed with these techniques. Redundancy is lowered and overfitting is faced in some scenarios. Different classifications of works based on this ability are made depending on element pruned, structured/unstructured (simmetry) and static/dynamic.

Static pruning is the process of removing elements of a network structure offline before training and inference processes. During these last processes no changes are made to the network previously modified. However, after removing different elements of the architecture it is interesting a fine-tuning or retraining of the pruned network. This is due to the changes that suffer the network by removing big part of its elements. Thus, some computational effort is needed in order to reach comparable accuracy to the original network.

The pruning has been carried out by following different criteria. In[1] and[2] they used the second derivative of the Hessian matrix to reduce the dimension of the original architecture. Optimal Brain Damage (OBD) and Optimal Brain Surgeon (OBS) respectively work under three assumptions. Quadratic: the cost function is near quadratic. Extremal: the pruning is done after the network converged. Diagonal: sums up the error of individual weights by pruning the result of the error caused by their co-consequence. Additionally, OBS avoids the diagonal assumption and improves neuron removal precision by up to 90% reduction in weights for XOR networks. Using Taylor expansions of first order[3][4] were also an alternative to the previous ones to tackle networks' dimension issues, as a criterion to approximate the change of loss in the objective function as an effect of pruning.

Some works were based on the magnitude of the elements themselves. It is undoubtedly true that near-zero values of weights make far less contribution to the results than others that surpass certain threshold value. By this way, removing connections that may appear unneeded the original network is shrunk. It is an interesting approach to develop this process layer-by-layer to not affect brutely to the performance of the resulting network. LASSO[5] was introduced as a penalty term. It shrinks the least absolute valued feature's corresponding weights increasing weight sparsity. This operation has been shown to offer a better performance than traditional procedures such as OLS by selecting the most significantly contributed variables instead of using all the variables, achieving approximately 60% more sparsity than OLS. The problem with LASSO is that is an element-wise pruning technique leading to unstructured network and sparse weight matrices. By performing this technique group-wise, as it does Group LASSO[6] removing entire groups of neurons and maintaining the original network's structure, this last issue was solved. Groups are made based on geometry, computational complexity or group sparsity among others.

Singular Value Decomposition (SVD) is an efective and promissing technique to shred convolutional or FC layers by reducing the number of parameters needed to represent them. Not only it has been useful for image classification tasks, but also in object detection[7] scenarios and others related with DNN based acoustic modeling[8][9]. Low-rank decomposition for convolution layers as well as fully connected layer was applied in several works. Kholiavchenko et al.[10] proposed an iterative approach to low-rank decomposition by applying dynamic rank selection to image classification and object detection models. One of its negative aspects was that iteratively applying low-rank decomposition needs longer time and higher computational resources for rank selection in deeper models. The alternative proposed by[11] assumes the properties of both low-rank and sparseness of weight matrices while aiming to reconstruct the original matrix. In[12] mixing the concepts of sparsity and existence of un-equal contributions of neurons towards achieving the target, Sparse Low Rank (SLR) method is proposed, a method that scatters SVD matrices to compress them by conserving lower rank for unimportant neurons. As a result, it is feasible to reduce 3.6×storage space of SVD without much variance on the model accuracy. Speedup in the computation was other advantage that has the structured sparsity obtained by the presented approach.

The majority of the works had paid attention to the individual pruning of layers not considering the connection between different layers. In[13] stated that the last FC layer is the one that has the most relevant effect on the final response of the entire network. Considering this last, they proposed to prune the previous layer of the network considering the connections of neurons with the neurons of this last FC layer called Final Response Layer (FRL). By this way, the relevances of the neurons considered independently for the FRL were backpropagated to the previous layer's neurons. The pruning of the rest of the layers is carried out similarly, scoring the relevance of the neurons considering the connections with the posterior layers' neurons.

Other alternatives have been proposed to carry out static pruning. In[14] was presented an innovative method for Convolutional Neural Networks (CNN) pruning called Layer-wise relevance propagation. Each unit's relevance to the final decision is measured, and the units that are below a predefined threshold are removed from the original structure. As a last step, each component's relevance is recalculated, by calculating the total relevance per layer to keep it constant through iterations. Thus, each unit's relevance is recalculated to maintain this value. In[15] a method of pruning redundant features along with their related feature maps according to their relative cosine distances in the feature space is proposed, achieving smaller networks with a significant download in post-training inference computational costs and achieving a decent performance. Redundancy can be

minimized while inference cost (FLOPS) is reduced by 40% for VGG-16, 28%/39% for ResNet-56/110 models trained on CIFAR-10, and 28% for ResNet-34 trained on ImageNet database with near negligible loss of accuracy. To fix the decrease in accuracy after pruning, models were retrained for some iterations maintaining all hyper-parameters untouched.

Theory

In this section we describe the methodology proposed in order to improve the results obtained in the literature for different neural networks and datasets.

The approach we present in this study follows this methodology. First, traditional low-rank decomposition SVD is applied to the weight matrix of the final FC layer, called FRL. Next, input and output weights in the layer are selected for sparsification using different neuron selection strategies. Then, sparsification is applied to the selected input and output neuron components in the decomposed matrices. With the most relevant neurons of the final FC layer obtained we back propagate their relevance to the prior FC layer, following the idea proposed by[13], and we obtained the relevance of the neurons composing the prior FC layer. Finally, we repeated the process of sparsification for the decomposed matrices of the prior FC layer. In parallel, we performed the same process of sparsification but only considering the relevance of each individual layer for the last two FC layers. The results and comparative of both methodologies are summarized in Section .

Single Value Decomposition(SVD)

One way for decomposing matrices representing the weights of neural networks is the use of low-rank factorization. A convolutional neural network is composed by a large number of convolutional layers and fully connected layers. By applying this technique to convolutional kernels weights optimization of the inference speed of the convolution operation could be obtained due to the reduction in the time needed for multiplication with factorized matrices compared to that of multiplication with 3D weights of kernels..

In a FC layer having m input and n output neurons, activation $a \in \mathbb{R}$ n of the layer with n nodes is represented as

$$a = \mathbf{g}(\mathbf{W}^T\mathbf{X} + \mathbf{b}) \tag{1}$$

where $\mathbf{X}$ represents the input to the layer and $\mathbf{g()}$ represents any of the possible activation functions. FC layers connections form a weight matrix $\mathbf{W} \in \mathbb{R}^{mn}$ and a bias vector $\mathbf{b} \in \mathbb{R}^n$ where each parameter in the weight matrix $\mathbf{W}$ is $w^{ij} \in \mathbb{R}$ $(1 \leq i \leq m, 1 \leq j \leq n)$ and bias matrix $\mathbf{b}$ is $b^j \in \mathbb{R}(1 \leq j \leq n)$. The proposed approach is applied to the weight matrix $\mathbf{W}$ after training the entire model. SVD approach decomposes the weight matrix $\mathbf{W}$ as $\mathbf{W} = \mathbf{USV}^T$ where $\mathbf{U} \in \mathbb{R}^{m \times m}$, $\mathbf{V}^T \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\mathbf{S} \in \mathbb{R}^{m \times n}$ is a diagonal matrix.

Sparse Low Rank Decomposition

The matrix $\mathbf{S}$ is a diagonal matrix containing n non-negative singular values in a decreasing order. The k singular values that are most significant are kept by Truncated SVD where the decomposed matrices $\mathbf{U}, \mathbf{S}$, and $\mathbf{V}T$ become $\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}^T \in \mathbb{R}^{m \times k}, \mathbb{R}^{k \times k}, \mathbb{R}^{k \times n}$. By this way, the original weights $\mathbf{W}$ are replaced into reconstructed approximated weight $\hat{\mathbf{W}}$ as $\hat{\mathbf{W}} = \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^T$.

In SVD we have diagonal matrix sigma $\mathbf{S}$ with the most significant singular values from the upper left to lower right in a decreasing order. In the truncation process the first k rows of $\mathbf{U}$ and columns of $\hat{\mathbf{V}}^T$ are kept.

Simulating the approach driven by[12] we compressed truncated matrices $\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}$ based on the importance of the m input and n output neurons, i.e. we represented few columns of $\hat{\mathbf{U}}$ and rows of $\hat{\mathbf{V}}^T$ with a rank lower than k , called as reduced rank rk. By this way, only rk most significant rows and columns are kept in $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}^T$ respectively due to the order of importance of $\mathbf{W}$ that starts from left to right through columns of $\hat{\mathbf{U}}$ and top to bottom through rows of $\hat{\mathbf{V}}^T$. We coonsidered only the most significant rows (rm) and columns (rn) from each column and row from $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}^T$ respectively, following the cost criteria, briefly explained in the next subsection.

When the matrices $\hat{\mathbf{U}}, \hat{\mathbf{S}}$ and $\hat{\mathbf{V}}^T$ are sparsified with sr and rr , the total number of non-zero parameters of the $\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}^T$ become $k(m - rm + n - rn + 1) + rk(rm + rn)$, which is lesser than the number of non-zero parameters of truncated SVD $k(m + n + 1)$.

Pruning fully connected layers is much more effective in terms of accuracy, time and energy efficiency than pruning convolutional layers as shown in[16], which contributes to bigger losses in prediction capability with the same rate of reduction of parameters.. Those are usually placed in the first positions in DNN and those are more sensitive than the ones that are placed in the last positions in many cases. In this study, we followed the approach directed by[12] sparsifying SVD matrices achieving a low compression rate without big losses in accuracy. We used as a metric of sparsification the compression rate defined in[12], as the ratio between the parameters needed to define the sparsified decomposed matrices and the original weights' matrix parameters. In our case, we analyzed their 3 variants of applying SLR, that were based in cost, weights and activations, and we

proposed two new variants that sums the importance of cost and weights and cost and activations due to the fact that each of them performed as best variant in different compression rate regimes.

Overall, the most relevant attribute was the cost so we decided to establish as the criteria for selection of the rows and columns for sparsification.

Selection of rows and columns based on cost

A neuron's importance is defined by whether there is change or not in the network performance after removing it. Let c be the default cost of the neural network with original trained weight W estimated for the p training samples, computed using any loss function. Let $\hat{c}$ be the value of cost of the network with sparsified weights $\hat{\mathbf{W}}$. By truncating with reduced rank rr a specific row of $\hat{\mathbf{U}}$ or column of $\hat{\mathbf{V}}^T$ we have the absolute change in cost is or os. Those are calculated as follows:

$$is_i = |c - \hat{c}_i| \tag{2}$$

$$os_j = |c - \hat{c}_j| \tag{3}$$

As the sparsification process purpose is to ensure that the functionality of the network does not change after compression, and not to reduce the overall network cost or improve accuracy but only the absolute change in the cost value is considered.

Propagation of relevance between layers

As it is known, the majority of neural networks can be formulated as a nested function. Thus, we can define a network with n hidden layers as a $F^{(n)} = f^{(n)} \circ f^{(n-1)} \circ ... \circ f^{(1)}$. Each layer can be represented as follows:

$$f^{(n)}(x) = \sigma^{(n)}(\mathbf{w}^{(n)}x + \mathbf{b}^{(n)}) \tag{4}$$

Where $\sigma^{(n)}$ is the activation function of each layer, $\mathbf{w}^{(n)}$ is the corresponding layers connections' weight function and $\mathbf{b}^{(n)}$ is the bias of each layer. At this stage it is legible to say that each of these layers are interconnected and each of them has direct relevance on the final decision of the entire network. Consequently, weigths from the FRL, that is the last Fully Connected Layer, backpropagate their relevance to the prior layers as proposed in[13]. As a result, the relevance of each neuron in the final decision is the composition of weights that are interconnected until the FRL corresponding element's relevance. The sumation of the corresponding relevances is given by the equation 5.

$$s_k = |\mathbf{w}^{(k+1)}|^\top |\mathbf{w}^{(k+2)}|^\top ... |\mathbf{w}^{(n)}|^\top s_n \tag{5}$$

The absolute value of the weights that are connected to each of the neurons of the FRL are multiplied by the relevance of these in the FRL.

$$s_{k,j} = \sum_i |w_{i,j}^{(k+1)}| s_{k+1,i} \tag{6}$$

The equation 6 shows the relevance of the j-th neuron in the k-th layer, that propagates the relevances of the neurons from the posterior $k+1$-th layer that are connected with it.

By introducing this idea to the SVD matrices, keeping only the most relevant rows of U matrices we can consider only the most relevant neurons of that layer. The procedure in the FC layers that are not the FRL, is similar to the original SLR method except for the sparsification of the U matrices where the relevance propagated through the posterior layers is considered to determine the most relevant neurons. This relevance is propagated following the Equation 6.

Material and methods

In this section, we present the datasets and models used for experimentation.

We used two well-known models for image recognition, VGG-16[17] and Lenet5[18], where VGG architecture is much known for its memory intensive FC layers. It is worth to note that VGG is the commonly used architecture with FC layers where other popular image recognition models, such as ResNet, Inception, MobileNet, ResNet, DenseNet, and object detection models, do not have FC layers except the final softmax layer. Table 1 and Table 2 show the specifications of each network structure.

Those two different approaches were tested on different well-known datasets, Cifar10 (VGG16), Cifar100 (VGG16) and MNIST (Lenet5). Each of them contain 32x32 images (color images in Cifar10/Cifar100 and grey scale images in MNIST). In case of Cifar10 and MNIST there are 10 different classes, and 100 in Cifar100. All of them have been trained using default 10,000 test images and 50,000 and 60,000 train images for Cifar and MNIST datasets respectively. Different compression rates

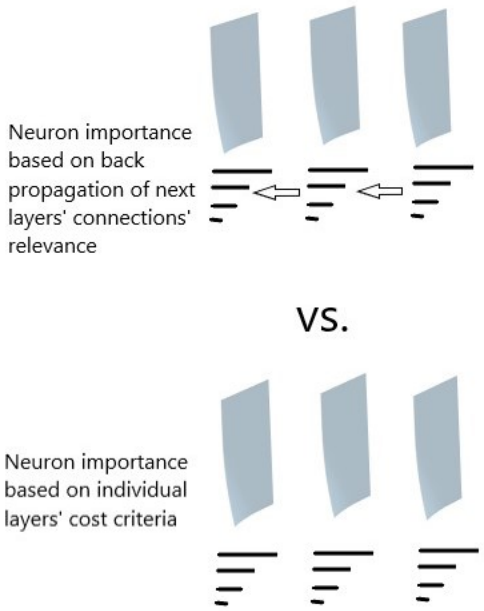

Figure 1. Comparative of the proposed approaches.

were applied for sparsifying SVD matrices, so that, for each dataset we obtained different performance metrics for each method. Overall, we were able to state which method was the best in each case. The datasets used for experiments comprise good mix of different image types, sizes and number of classes. CIFAR-10 and CIFAR-100 have general purpose image classes where MNIST dataset contains handwritten digit images.

The environment in which all development of our work had been processed is a x64 Ubuntu 20.04.4 LTS Operating System equipped with a Intel Core i7-11850H working at 2,5 GHz X 16 and 32 GB DDR-4 RAM, and a NVIDIA GPU.

Proposed approach

As cited above, the intentionality of this research was to determine the connection of relevances between different layers. To do so, we opted for applying the approach presented by[12] in two different FC layers. First, we applied it independently. To show that there exists a direct relationship between neurons from different layers, we considered the relevance of the FRL and backpropagate it until the second FC layer we pruned in the parallel process. By this way, we could see the effect of backpropagating the relevance throughout layers and see the correlation between them.

We applied SLR approach proposed by[12] to obtain the information about the most relevant parts forming the FRL. By this way, we were able to know the relevances to the final decision of each of the neurons comprising this last FC layer. To calculate the relevance propagated to the previous layers we used the insight introduced in Section , by multiplying each of the absolute value of weights that were connected with each neuron of the next layer with the relevance of these neurons from the next layer, for each neuron comprising the layer in question. Finally, after obtaining the relevances for each neuron from the layer, we sparsified the weight matrix of this layer as we did for the FRL but sparsifying the U matrix in the following way. We considered only the rows that obtained the highest value after the sumation of multiplications of absolute weights of connections with each of the relevances of neurons connected from the next layer, instead of considering the original relevances of neurons as we did for the FRL.

At the same time, we carried out sparsification of the same number of layers only considering the independent relevances of each layer, following the criteria proposed by[12]. In this work they present 3 different criteria to determine which elements of each layer were more relevant to the final decision of the network. Overall, the criteria based in the cost of weights was the most adequate to reduce the dimensionality of the problem and mantain the performance of the architecture as high as possible. Both approaches graphical representation is given by Figure 1.

In case of VGG16 the FRL corresponds to FC7 and the backpropagation of the relevances has been carried out until FC6. FRL and previous FC layer of Lenet5 are FC4 and FC3 respectively.

Table 1. VGG16 model trained for 32x32 images.

Layer name	Layer type	Feature map	Output size of images	Kernel size	Stride	Activation
Input	Image	1	32x32x3	-	-	-
Conv-1	2 x Conv	64	32x32x64	3x3	1	relu
Pool1	Maxpool	64	16x16x64	3x3	2	relu
Conv-2	2 x Conv	128	16x16x128	3x3	1	relu
Pool2	Maxpool	128	8x8x128	3x3	2	relu
Conv-3	2 x Conv	256	8x8x256	3x3	1	relu
Pool3	Maxpool	256	4x4x256	3x3	2	relu
Conv-4	2 x Conv	512	4x4x512	3x3	1	relu
Pool4	Maxpool	512	2x2x512	3x3	2	relu
Conv-5	2 x Conv	512	2x2x512	3x3	1	relu
Pool5	Maxpool	512	1x1x512	3x3	2	relu
Flatten	Flatten	-	512	-	-	relu
FC6	Dense	-	4096	-	-	relu
FC7	Dense	-	4096	-	-	relu
FC8	Dense	-	# of classes	-	-	softmax

Table 2. Lenet5 model trained for 32x32 images.

Layer name	Layer type	Feature map	Output size of images	Kernel size	Stride	Activation
Input	Image	1	32x32x3	-	-	-
Conv-1	1 x Conv	6	28x28x6	5x5	1	tanh
Pool1	Avgppool	6	14x14x6	2x2	2	tanh
Conv-2	1 x Conv	16	10x10x16	5x5	1	tanh
Pool2	Avgppool	16	5x5x16	2x2	2	tanh
Flatten	Flatten	-	400	-	-	tanh
FC3	Dense	-	120	-	-	relu
FC4	Dense	-	84	-	-	relu
FC5	Dense	-	# of classes	-	-	softmax

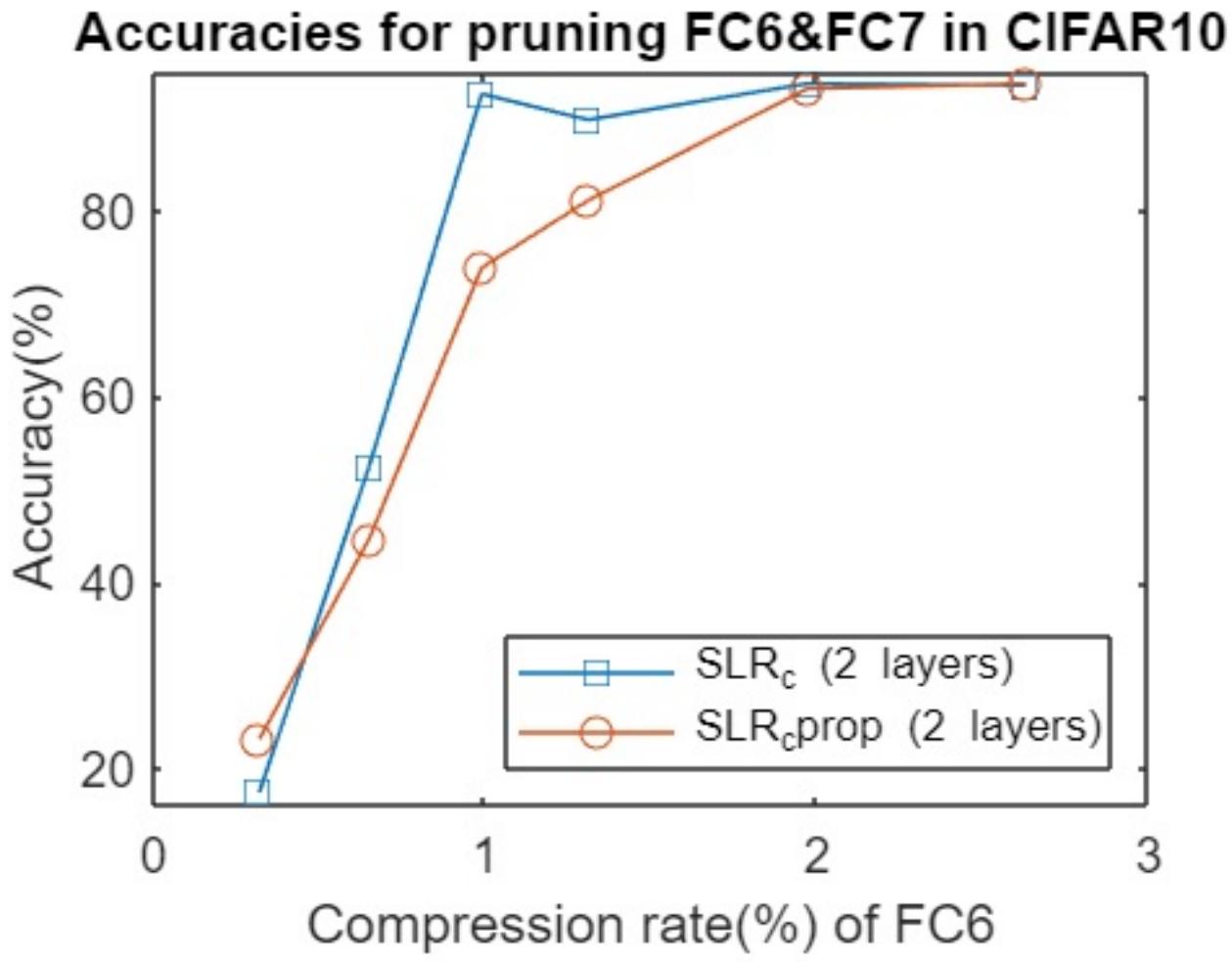

Figure 2. Accuracies for pruning VGG16 network on Cifar10 dataset for different compression rates.

Experimental

In this section details about the entire experimentation process are described. The results obtained are summarized as well.

Performance metrics

Evaluation metrics used for determining which of the methods used is best for keeping the performance of the former network as high as possible are the accuracy and the compression rate defined in[12]. This last determines the relationship of parameters between sparsified decomposed matrices and the original network's weight matrices. We used FRL's previous FC layer's compression rate to check the accuracy of the resultant network on different compression rate regimes.

Results

Each of the variant proposed in this work, considering or not the relevance between layers, have been tested on well-known open source datasets for image recognition Cifar10, Cifar100 and MNIST. All of them have been trained using default 10,000 test images and 50,000 and 60,000 train images for Cifar and MNIST datasets respectively. In each case, we opted for stablishing the same reduction rate (0.5) and sparsity rate (0.5) defined in[12], and we tested each variant with different rank k, which determines the number of columns and rows kept in the sparsified $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}^T$ matrices.

Figure 2 shows the accuracies obtained after testing both pruning techniques for VGG16 architecture on Cifar10 dataset. As it is clear, there is no significant difference between both methods when applying an extremely low compression rate, which means that very few parameters of the original matrices are kept. Similarly, we could observe the same pattern when a higher number of parameters are kept in the original decomposed matrices, but there are significant differences between both

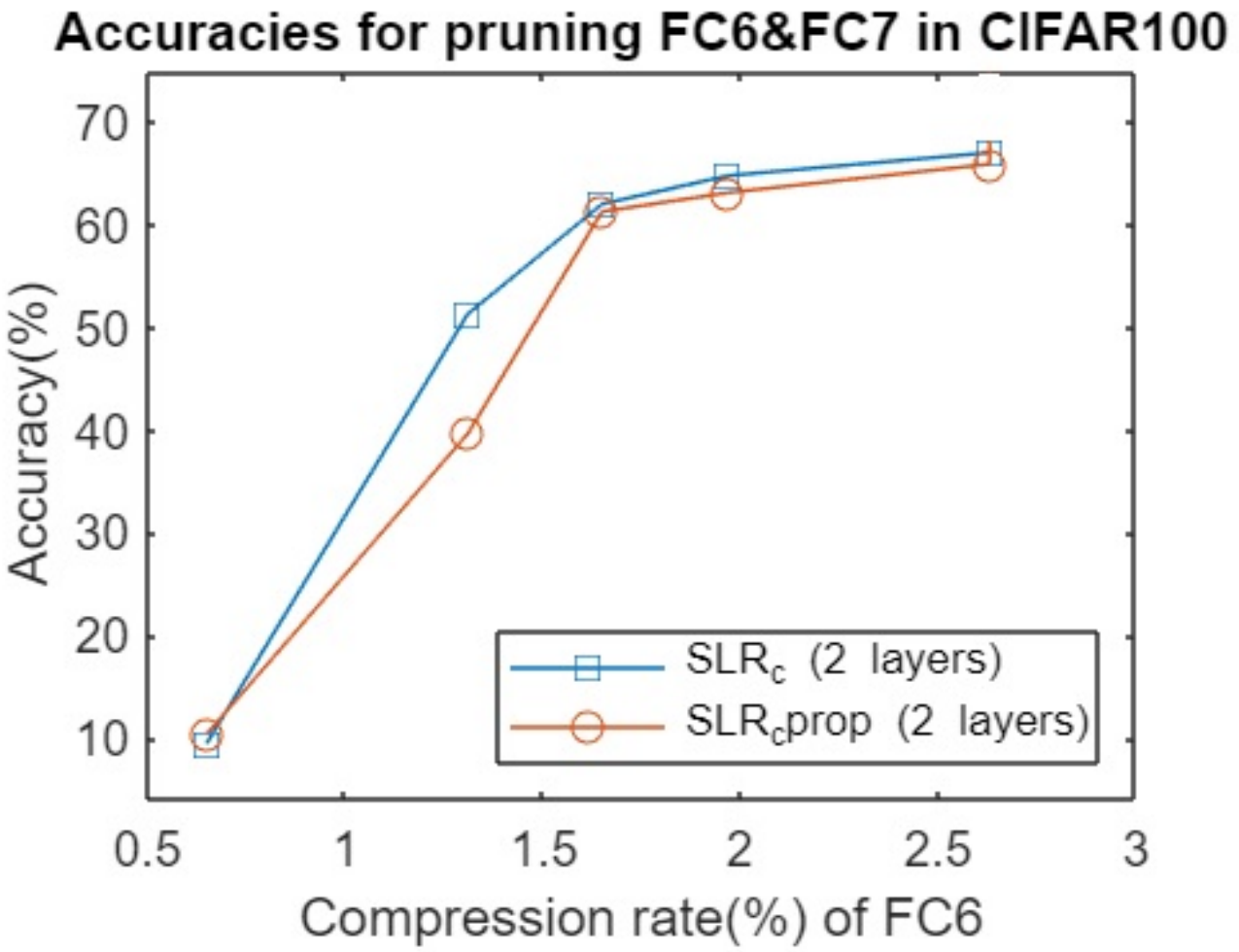

Figure 3. Accuracies for pruning VGG16 network on Cifar100 dataset for different compression rates.

compression rate regimes. In this case, applying independently SLR method to different FC layers offers higher accuracy with the same compression rate, i.e. keeping the same number of connections between neurons.

Similarly, there could figure out the same pattern in case of Cifar100 dataset for the same network architechture. The application of SLR method independently without any consideration of propagation of relevances across layers. In this case for lower compression rate regimes the difference is high as well. Figure 3 shows the results summarized for Cifar100 dataset.

Finally, Lenet5 architecture was pruned following both methodologies on MNIST dataset. Figure 4 shows the accuracies obtained for different compression rates. As it could be observed, for the majority of the pruning rates applied applying SLR independently in different layers offers better performance than considering the backpropagation rule of the relevances from the FRL. However, in certain compression rate regime the last one outperforms the former one, but the difference is insignificant compared to the overall performance result.

Discussion

As demonstrated in the previous section, the introduction of the concept of the backpropagation of the relevances from the FRL to the rest of the layers of the original network does not always outperform the supposition of the relevances independently within layers. By this way, the breakthrough presented by[13] is not preserved in this experimentation, being more important for the final result the relevances of each layer at the moment of pruning the connections between different layers. For relatively high compression rate regimes, where the number of pruned connections is not so high, the performance metrics are almost identical for both architectures applied for the 3 different datasets. On the contrary, for lower compression rates, where the accuracies are decent and enough for the majority of the edge computing contexts (60%-80%), the difference is clear in favour

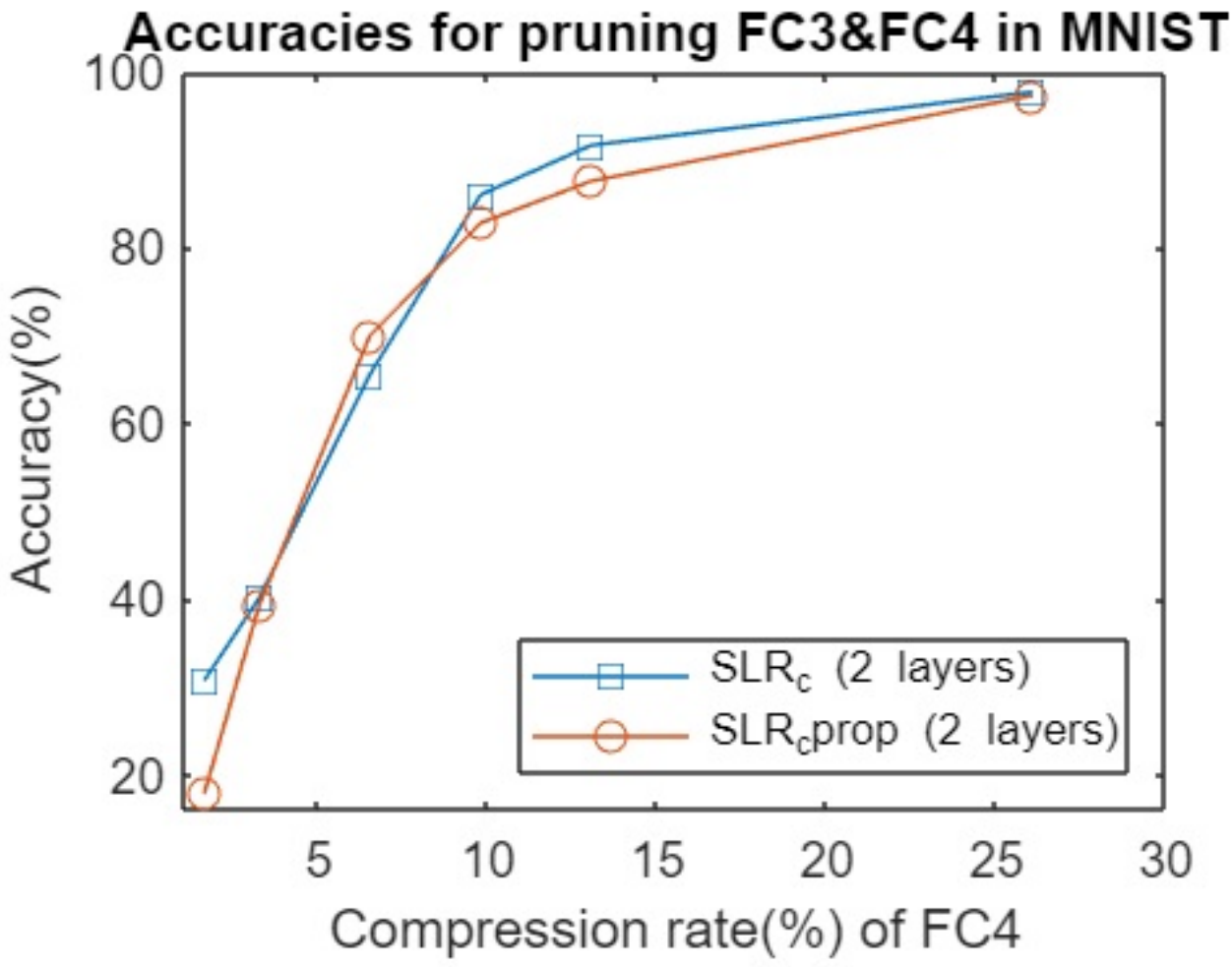

Figure 4. Accuracies for pruning Lenet5 network on MNIST dataset for different compression rates.

of independent appliaction of SLR method. For very low compression rate regimes the performance metrics do not follow a distinguishable pattern showing the randomness of both methods when an excessive pruning is carried out in any of the mentioned architectures.

This shows that the components of each layer could have certain influence on the rest of the network components, but the main contribution to the final result of each component is more connected with other aspects rather than the connections' weights' absolute values across layers. In this case, the cost defined as the difference on the accuracy between the case when a certain component is eliminated from the original network and the original structure's accuracy, showed that could be more crucial when selecting which connections to remove when pruning the original network.

To summarize, the proposition presented by[13] echoed in Equation 6 is not fully conserved in this experimental process, challenging its validity in for every architecture of convolutional neural network focused on image recognition. In fact, the ranking of connection's relevance proposed by[12] offers optimal result in terms of accuracy and network compression, needing a very low percentage of parameters for representing sparsified matrices compared to the original network's matrices. However, the computational cost of calculating each matrices' components costs might be too high and ineffective in many scenarios, needing an alternative way for solving this issue. Attending these weights' absolute values offers near identical result in terms of accuracy offering a $\sim$ 150x faster solution. In applications where time response would be crucial may be more adequate the use of this last alternative.

References

1. LeCun, Y., Denker, J. & Solla, S. Optimal brain damage. In Touretzky, D. (ed.) *Advances in Neural Information Processing Systems*, vol. 2 (Morgan-Kaufmann, 1989).

2. Hassibi, B., Stork, D. & Wolff, G. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, 293–299 vol.1, DOI: 10.1109/ICNN.1993.298572 (1993).

3. Molchanov, P., Tyree, S., Karras, T., Aila, T. & Kautz, J. Pruning convolutional neural networks for resource efficient inference, DOI: 10.48550/ARXIV.1611.06440 (2016).

4. Yu, C., Wang, J., Chen, Y. & Wu, Z. Transfer channel pruning for compressing deep domain adaptation models. In U., L. H. & Lauw, H. W. (eds.) *Trends and Applications in Knowledge Discovery and Data Mining*, 257–273 (Springer International Publishing, Cham, 2019).

5. Muthukrishnan, R. & Rohini, R. Lasso: A feature selection technique in predictive modeling for machine learning. In *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, 18–20, DOI: 10.1109/ICACA.2016.7887916 (2016).

6. Yuan, M. & Lin, Y. Model selection and estimation in regression with grouped variables. *J. Royal Stat. Soc. Ser. B (Statistical Methodol.* **68**, 49–67, DOI: https://doi.org/10.1111/j.1467-9868.2005.00532.x (2006). https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2005.00532.x.

7. Girshick, R. Fast r-cnn, DOI: 10.48550/ARXIV.1504.08083 (2015).

8. Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E. & Ramabhadran, B. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 6655–6659, DOI: 10.1109/ICASSP.2013.6638949 (2013).

9. Xue, J., Li, J. & Gong, Y. Restructuring of deep neural network acoustic models with singular value decomposition. In *INTERSPEECH* (2013).

10. Kholiavchenko, M. Iterative low-rank approximation for cnn compression, DOI: 10.48550/ARXIV.1803.08995 (2018).

11. Yu, X., Liu, T., Wang, X. & Tao, D. On compressing deep models by low rank and sparse decomposition. 67–76, DOI: 10.1109/CVPR.2017.15 (2017).

12. Swaminathan, S., Garg, D., Kannan, R. & Andres, F. Sparse low rank factorization for deep neural network compression. *Neurocomputing* DOI: 10.1016/j.neucom.2020.02.035 (2020).

13. Yu, R. *et al.* Nisp: Pruning networks using neuron importance score propagation, DOI: 10.48550/ARXIV.1711.05908 (2017).

14. Yeom, S.-K. *et al.* Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognit.* **115**, 107899, DOI: 10.1016/j.patcog.2021.107899 (2021).

15. Ayinde, B. O., Inanc, T. & Zurada, J. M. Redundant feature pruning for accelerated inference in deep neural networks. *Neural Networks* **118**, 148–158, DOI: https://doi.org/10.1016/j.neunet.2019.04.021 (2019).

16. Han, S., Pool, J., Tran, J. & Dally, W. J. Learning both weights and connections for efficient neural networks, DOI: 10.48550/ARXIV.1506.02626 (2015).

17. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556* (2014).

18. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278 – 2324, DOI: 10.1109/5.726791 (1998).

Acknowledgements (not compulsory)

Acknowledgements should be brief, and should not include thanks to anonymous referees and editors, or effusive comments. Grant or contribution numbers may be acknowledged.

Author contributions statement

All the authors have contributed equally to this work.

Additional information

There were no competing interests in this work.

Data availability statement

The raw data used in this work is available at https://keras.io/api/datasets/ were a brief explanation of each dataset is given, as well as explanatory information of using them.