Aggressively prunable MAM²-based Deep Neural Oracle for ECG acquisition by Compressed Sensing

This is the author's version of the article that has been presented at IEEE BioCAS2022
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS54905.2022.9948676

Aggressively prunable MAM²-based Deep Neural Oracle for ECG acquisition by Compressed Sensing

Philippe Bich[‡], Luciano Prono[‡], Mauro Mangia[*], Fabio Pareschi[‡†], Riccardo Rovatti[*†] and Gianluca Setti[‡†]
[‡]DET, Politecnico di Torino, Italy - Email: {philippe.bich, luciano.prono, fabio.pareschi, gianluca.setti}@polito.it
[*]DEI, [†]ARCES, University of Bologna, Italy - Email: {mauro.mangia, riccardo.rovatti}@unibo.it

***Abstract*—The growing interest in Internet of Things (IoT) and mobile biomedical applications is pushing the investigation on approaches that can be used to reduce the energy consumption while acquiring data. Compressed Sensing (CS) is a technique that allows to reduce the energy required for the acquisition and compression of a sparse signal, transferring the complexity to the reconstruction stage. Many works leverage the use of Deep Neural Networks (DNNs) for signal reconstruction and, assuming that also this operation has to be performed on a IoT device, it is necessary for the DNN architecture to fit in small and low-energy devices. Pruning techniques, that can reduce the size of DNNs by removing unnecessary parameters and thus decreasing storage requirements, can be of great help in this effort. In this work, a novel Multiply and Max&Min (MAM²) map-reduce paradigm trained with the vanishing contributes technique and then pruned with the activation rate method is proposed. The result is a naturally and aggressively pruned DNN layer structure. This structure is used to reduce the complexity of a DNN-based CS reconstructor and its performance is verified. As an example, MAM²-based layers still retain the baseline accuracy of the CS decoder with 94% of the parameters pruned against 25% when using classic MAC-based layers only.**

I. Introduction

Nowadays, the necessity of low-cost and low-power hardware and software implementations for a wide range of applications has become essential. In particular, the acquisition and elaboration of bio-signals by means of edge devices has become even more important. A typical example can be found in Internet of Things (IoT) for medical applications, whose aim is to continuously collect data and elaborate them to monitor the patient health state [1], as schematized in Fig. 1. As the reliability and life-time of such a system, generally running on batteries, is fundamental, lowering the energy required for any devices is paramount.

The Compressed Sensing (CS) technique [2]–[4] is very promising when searching for this kind of energy reduction. Assuming that an input signal is *sparse* (condition often found in actual real scenarios), CS simultaneously samples and compresses an input signal by means of a simple linear projection on a set of sensing waveforms.

However, the reconstruction of the compressed signal is a quite complex task. Many different reconstruction methods have been proposed such as Spectral Projected Gradient for ℓ_1 Minimization (SPGL1) [5], the Generalized Approximate Message Passing (GAMP) [6], the Orthogonal Matching Pursuit (OMP) [7] or the Compressive Sampling Matching Pursuit (CoSaMP) [8]. Also, there exist many methods that leverage the learning capabilities of the Deep Neural Networks (DNNs) to reconstruct the compressed signal [9]–[20]. In these cases, while trying to reduce the signal reconstruction complexity, it is fundamental to reduce as much as possible the size of DNNs structures. In fact, DNNs typically use a massive number of trainable parameters and are often redundant; hence it is possible to prune the unnecessary parameters by removing interconnections between neurons that do not appreciably influence the accuracy of the DNN task.

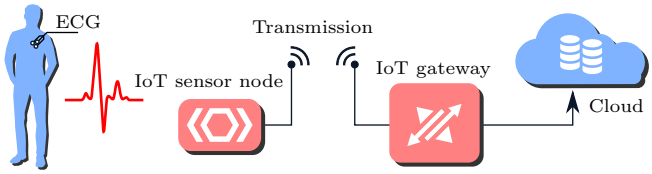

Fig. 1. Example of medical application of IoT: a sensor node continuously acquire ECG data from a patient and transmit it so that it can be analyzed. The analysis can happen on an IoT device on the edge or directly in the cloud on servers.

Pruning is a fundamental operation to achieve low-power, low-latency, and lightweight machine learning inference. This problem has been addressed in multiple ways and many solutions have been proposed in the literature. The simplest approach is to score parameters of a DNN according to their absolute value and prune the ones with the lowest scores. Many works can be found that are based on this strategy. Some algorithms prune individual parameters [21] while others try to remove entire neurons, filters or channels [22]. In [23], weights are removed layer by layer while in [24] weight scores are related to the global state of the network. It is possible to prune all the parameters in a single step [25] but there also exist methods that prune the network iteratively.

In this work, we focus on the reduction of the computational complexity of the *Trained Compressed Sensing with Support Oracle* (TCSSO), a DNN-based CS decoder described in [9], and we test it on a synthetic Electrocardiogram (ECG) dataset. To do so, we propose a DNN layer based on a novel Multiply and Max&Min (MAM^2) map-reduce paradigm that is naturally prone to aggressive pruning. In fact, this structure is capable of *automatically selecting in one shot*, during its training, the weights that are *really indispensable* for the task to be performed.

The paper is structured as follows. In Section II the CS problem and the TCSSO are briefly described. Then the description of the novel MAM^2-based DNN layer can be found in Section III. Section IV shows how this novel layer is capable of reducing the number of parameters of the TCSSO without degrading its performance. Finally, the conclusion is drawn.

II. CS Decoder with Deep Neural Oracles

Given a sensor readings stream, it is possible to split it up into subsequent windows of n samples each. Each window is represented by a vector $\boldsymbol{x} = (x_1, ..., x_n)$. Let us assume that

This is the author's version of the article that has been presented at IEEE BioCAS2022
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS54905.2022.9948676

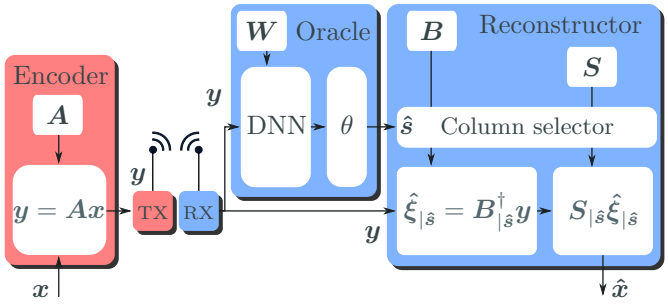

Fig. 2. Structure of the TCSSO framework for CS. In this work we focus on the reduction of the number of weights $\boldsymbol{W}$ used by the DNN in the oracle.

any vector $\boldsymbol{x}$ is κ-sparse, meaning it can be represented as $\boldsymbol{x} = \boldsymbol{S}\boldsymbol{\xi}$, where $\boldsymbol{S}$ is an orthonormal sparsity matrix and up to κ values in $\boldsymbol{\xi}$ are non-zero.

Signal compression with CS is performed by projecting each window $\boldsymbol{x}$ on the rows of a predetermined sensing matrix $\boldsymbol{A}$ of size $m \times n$, resulting in a measurement vector $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ with $\boldsymbol{y}$ of size $m < n$.

It is possible to leverage the sparsity property of signal $\boldsymbol{x}$ in order to reconstruct the signal from measurement $\boldsymbol{y}$. In this work, we focus on the TCSSO framework where the decoder splits the reconstruction problem in two steps: first a DNN-based oracle *divines* the support of $\boldsymbol{\xi}$, i.e., the positions of the non-zero values, then, by using this information, the signal is reconstructed by means of the Moore-Penrose pseudoinverse operation. The TCSSO structure is represented in Fig. 2.

More in detail, the first step, which is the DNN-based oracle, can be described as

$$\boldsymbol{o} = F(\boldsymbol{y}),\ \hat{\boldsymbol{s}} = \theta(\boldsymbol{o}) \tag{1}$$

where $F(\cdot)$ represent the trained DNN with input $\boldsymbol{y}$ and output $\boldsymbol{o} \in [0,1]^n$, where each of the outputs $(\boldsymbol{o}_1, ..., \boldsymbol{o}_n)$ is the probability that the corresponding value in $\boldsymbol{\xi}$ is non-zero. Function $\theta(\cdot)$ thresholds $\boldsymbol{o}$ in order to obtain the predicted support $\hat{\boldsymbol{s}} \in \{0,1\}^n$, which indicates where $\boldsymbol{\xi}$ contains non-zero values.

The second step is instead

$$\hat{\boldsymbol{\xi}}_{|\hat{\boldsymbol{s}}} = \boldsymbol{B}^{\dagger}_{|\hat{\boldsymbol{s}}}\boldsymbol{y},\ \hat{\boldsymbol{x}} = \boldsymbol{S}_{|\hat{\boldsymbol{s}}}\hat{\boldsymbol{\xi}}_{|\hat{\boldsymbol{s}}} \tag{2}$$

where $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{S}$, the operator $\cdot_{|\hat{\boldsymbol{s}}}$ produces a matrix containing only the columns of the input matrix indicated by $\hat{\boldsymbol{s}}$, and $\cdot^{\dagger}$ is the pseudoinverse operation, $\hat{\boldsymbol{\xi}}_{|\hat{\boldsymbol{s}}} \in \mathbb{R}^m$ is the reconstructed sparsity vector containing only the non-zero contributes and $\hat{\boldsymbol{x}}$ is the reconstructed signal.

In terms of memory footprint, the DNN-based oracle is the most demanding in the workflow, thus a reduction of its parameters is desirable. This can be achieved with a structure that easily allows pruning, i.e., the removal of DNN parameters.

III. Non-conventional layer description

A. Layer architecture

Given a DNN layer ℓ, in a standard feed-forward fully connected neural network the output $\boldsymbol{a}^{(\ell)}$ of size $N^{(\ell)}$ of a MAC-based dense layer is computed as follows

$$\boldsymbol{z}^{(\ell)} = \boldsymbol{W}^{(\ell)}\boldsymbol{a}^{(\ell-1)} + \boldsymbol{b}^{(\ell)} \tag{3}$$

$$\boldsymbol{a}^{(\ell)} = f^{(\ell)}(\boldsymbol{z}^{(\ell)}) \tag{4}$$

where $\boldsymbol{a}^{(\ell-1)}$ of size $N^{(\ell-1)}$ is the input column vector, $\boldsymbol{W}^{(\ell)}$ of size $N^{(\ell)} \times N^{(\ell-1)}$ is the matrix of the weights, $\boldsymbol{b}$ of size $N^{(\ell)}$ is the bias row vector and $f^{(\ell)}(\cdot)$ is the activation function.

Defined $\boldsymbol{W}^{(\ell)}_{ij}$ as the value at row i and column j of matrix $\boldsymbol{W}^{(\ell)}$, it may be convenient to rethink the MAC paradigm supposing that the sum operation producing the i-th output $\boldsymbol{v}_i = \sum_{j=1}^{N^{(\ell)}} \boldsymbol{W}^{(\ell)}_{i,j}\boldsymbol{a}^{(\ell-1)}_j + \boldsymbol{b}^{(\ell)}_i$ is dominated by few arguments, i.e., the summation can be roughly approximated by the sum of few dominant entries. Furthermore, an extreme application of this idea is the hypothesis that the whole sum may be represented by only two arguments as suggested in [26]. This can be implemented by substituting the accumulate operation with the maximum and minimum operations as follows. First we define

$$\boldsymbol{v}^{(\ell)}_i = \boldsymbol{W}^{(\ell)}_{i,\cdot} \odot \boldsymbol{a}^{(\ell-1)} \tag{5}$$

where $\odot$ is the Hadamard product, which performs element-wise products between scalars contained in two vectors, and $\boldsymbol{W}_{i,\cdot}$ is the i-th row of the weight matrix. Then, the i-th entry of the vector $\boldsymbol{z}^{(\ell)}$ is rewritten as

$$\boldsymbol{z}^{(\ell)}_i = \max_{1\le j\le N^{(\ell)}} \boldsymbol{v}^{(\ell)}_i + \min_{1\le j\le N^{(\ell)}} \boldsymbol{v}^{(\ell)}_i + \boldsymbol{b}^{(\ell)}_i \tag{6}$$

with the final output still computed using (4). We will refer to this map-reduce paradigm as Multiply and Max&Min (MAM^2, *MAM-squared*) and we build our new type of layer over this.

B. Training a MAM^2-based layer: the vanishing contributes technique

To ease the train process of MAM^2-based layers, we substitute the $\max_j(\cdot)$ and $\min_j(\cdot)$ operations with a function whose purpose is acting as a bridge between the MAC and the MAM^2 paradigms. For this reason, we define a function of an input $\boldsymbol{v}$ and a parameter $\beta \in [0,1]$ as

$$s_j(\boldsymbol{v};\beta) = \begin{cases} \boldsymbol{v}_j & \text{if } j = \underset{1\le j\le N^{(\ell)}}{\text{argmax}}\ \boldsymbol{v}_j \\ \boldsymbol{v}_j & \text{if } j = \underset{1\le j\le N^{(\ell)}}{\text{argmin}}\ \boldsymbol{v}_j \\ \beta\boldsymbol{v}_j & \text{otherwise} \end{cases} \tag{7}$$

With this, during training the output of a MAM^2-based layer ℓ can be described as

$$\boldsymbol{z}^{(\ell)}_i = \sum_{j=1}^{n} s_j(\boldsymbol{v}^{(\ell)}_i, \beta) + \boldsymbol{b}^{(\ell)}_i \tag{8}$$

where $\boldsymbol{v}^{(\ell)}_i$ is defined in (5).

These properties allow us to gradually transition from a MAC-based network to a MAM^2-based one during training. Indicating with p the number of training epoch and with $\beta[p]$ the actual value of β used at epoch p, it is convenient to start with $\beta[1] = 1$ and then change gradually to $\beta[p] = 0$ for large values of p. We call this technique *vanishing contributes*. This method allows a better DNN accuracy when compared to [26], where the max/min based paradigm is trained as-it-is, with a great improvement especially for networks with more than one hidden layer. Note that only the training is performed with a non-constant value of β. After training, pruning and tests are

This is the author's version of the article that has been presented at IEEE BioCAS2022
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS54905.2022.9948676

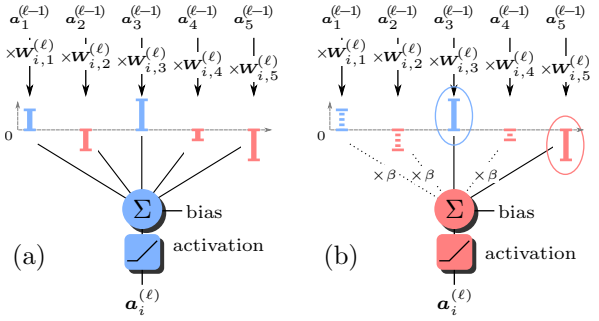

Fig. 3. Comparison of a MAC-based neuron (a) and a MAM^2-based neuron (b). The MAM^2-based neuron takes the maximum and minimum values as-they-are, while it multiplies the others by a value $\beta[e]$. This way, when $\beta = 1$ the two structures are equivalent, when $\beta = 0$ we perform the operations in (6) and intermediate values of β serve as a bridge between the two paradigms.

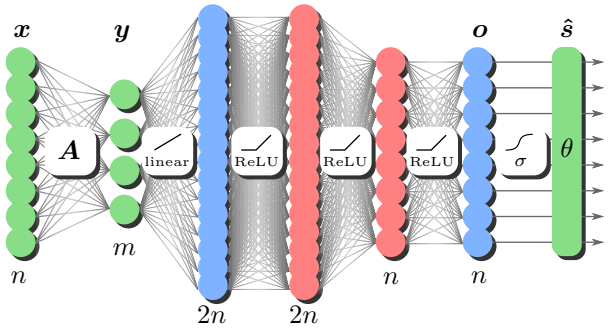

Fig. 4. Structure of the DNN-based oracle. The first layer is used to optimize the sensing matrix $\boldsymbol{A}$ and has no bias nor activation function. In the TCSSO, the compressed signal $\boldsymbol{y}$ is fed directly to the second layer. The third and the fourth hidden layers are the ones that need most of the parameters and are the ones that are being pruned.

performed with $\beta = 0$, e.g., with hard max/min functions as in (6). This map-reduce paradigm for a single neuron is schematized in Fig. 3, where it is compared with the classic MAC-based neuron.

C. Pruning a MAM^2-based layer: the activation rate method

When training is complete, MAM^2-based layers can be pruned with the *activation rate* method. During data inference, $\max_i(\cdot)$ and $\min_i(\cdot)$ select only two values $\boldsymbol{W}_{i,j}^{(\ell)}\boldsymbol{a}_j^{(\ell-1)}$ for each $i = 1, 2, .., N^{(\ell)}$, while the others are discarded. The weights $\boldsymbol{W}_{i,j}^{(\ell)}$ associated with the values that have not been discarded are then considered *activated*. So, we keep track of the number of times each weight has been activated. For each weight, the ratio between this number and the total number of inferences that are performed (i.e., the number of instances that are fed to the DNN in the pruning phase) is defined as the *activation rate*. After this, a threshold on the activation rate is set and the weights that are less activated are removed. The higher the activation rate threshold, the larger the number of pruned weights. Furthermore, in all the performed tests, most final weights have a null activation rate, showing that in our case this structure is naturally prone to aggressive pruning.

IV. Performance and size

We assess here the performance of the TCSSO decoder that employs the MAM^2-based layer described in Section III. For this purpose, we use a synthetic ECG dataset to train and test the decoder. The performance is compared to a size-equivalent TCSSO decoder built only with classic fully-connected MAC-based layers. Pruning is performed on both MAM^2-based and MAC-based structures in order to minimize their complexity. Finally, the accuracy vs the number of parameters kept for the two structures is evaluated and their performance is compared.

A. ECG dataset

The synthetic ECG dataset is generated as described in [27] and used as in [9], [10]. The dataset employed for the performance assessment is composed of 800 000 windows of size $n = 256$, of which 720 000 are used to train the TCSSO encoder/decoder, 40 000 compose the validation set while the remaining 40 000 are used to test the performance. Each window has been pre-sparsified, meaning that κ-sparsity has been enforced for each window by removing the $n - \kappa$ smallest entries of ξ. This has been done to ensure that the conditions for CS are met, while a further insight on the signal sparsity problem and how to treat realistic signals can be found in [10]. For each signal, the true support vector $\boldsymbol{s}$ is retrieved based on the non-zero elements of $\boldsymbol{\xi}$ and used as training label.

White noise has been added to the signals, so that the Intrinsic Signal to Noise Ratio (ISNR) is 60 dB.

B. Performance figures

In order to assess the performance of the CS decoder, different metrics have been defined in literature [28], [29]. In particular, the Reconstructed Signal to Noise Ratio (RSNR) is measured in dB and defined as

$$\text{RSNR} = 20\log_{10}\frac{\|\boldsymbol{x}\|_2}{\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2} = \left(\frac{\|\boldsymbol{x}\|_2}{\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2}\right)_{\text{dB}} \tag{9}$$

The performance over a batch of signals can be expressed as the Average RSNR (ARSNR) value

$$\text{ARSNR} = \mathbf{E}\left[\left(\frac{\|\boldsymbol{x}\|_2}{\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2}\right)_{\text{dB}}\right] \tag{10}$$

where $\mathbf{E}[\cdot]$ is the expected value.

C. Oracle DNN configuration

The structure of the DNN used in the oracle is described in Fig. 4. The first layer is used as an encoder to optimize the sensing matrix $\boldsymbol{A}$, whose scalar values correspond to the weights of the layer. It has n inputs and m outputs, uses a linear activation function and has no bias. It accepts as input the uncompressed signal $\boldsymbol{x}$ and generates the measurements $\boldsymbol{y}$. After training, this layer is detached and used for encoding/compression.

The following four fully-connected layers accept the measurements $\boldsymbol{y}$ as input and are used for support retrieval and have $2n$-$2n$-n-n neurons, respectively. All the layers but the last use a ReLU activation function, while the last uses a sigmoid function. The outputs, $\boldsymbol{o}$, are limited in $[0, 1]$ and then thresholded to obtain the support vector $\boldsymbol{s}$, which contains only zeros and ones.

This is the author's version of the article that has been presented at IEEE BioCAS2022
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS54905.2022.9948676

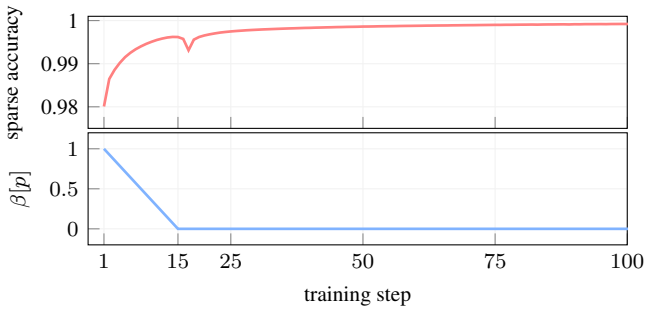

Fig. 5. Sparse accuracy trend on the training set during the training of the DNN-based oracle, along with $\beta[p]$ trend, where p is the epoch number. After an initial accuracy increase, there is an accuracy drop when $\beta[p] = 0$ but the DNN quickly recovers its classification capability.

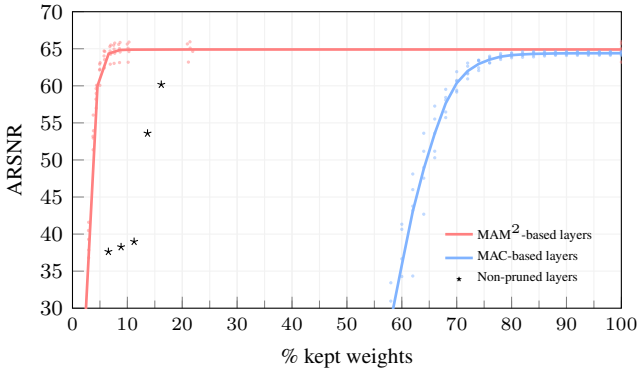

Fig. 6. ARSNR vs the percentage of kept weights in the third and fourth hidden layers of the TCSSO ($m = 64$) after pruning. The mean performance over 5 tests is highlighted. Points marked with a star indicate non-pruned models trained directly with a smaller number of neurons.

For this work, we use $n = 256$, $m = 64$ and a threshold value of 0.1.

In order to assess the performance of the layer described in Section III, we use it to reduce the parameters needed by the DNN. We substitute the second and third and layers in the decoder with two MAM^2-based layers keeping the same number of neurons, as they contain about 80% of the total parameters (see Fig. 4).

We compare the DNN that contains the MAM^2-layers with one completely built over classic MAC-based layers.

D. Oracle DNN training

During training, pairs of signals $\boldsymbol{x}$ and supports $\boldsymbol{s}$ are fed to the oracle DNN. Training is performed with Adam optimizer [30] with learning rate = 0.001 and batch size = 256 for 350 epochs. The cost function to be minimized is the component-wise clipped cross-entropy between the true support $\boldsymbol{s}$ and the DNN output $\boldsymbol{o}$

$$C(\boldsymbol{s}, \boldsymbol{o}) = -\sum_{i|\boldsymbol{s}_i=1} \mathrm{L}_\epsilon(\boldsymbol{o}_i) - \sum_{i|\boldsymbol{s}_i=0} \mathrm{L}_\epsilon(1-\boldsymbol{o}_i) \qquad (11)$$

where $\mathrm{L}_\epsilon(\cdot)$ is a clipped log function defined as $\min\{\log_2(1-\epsilon), \max\{\log_2(\epsilon), \log_2(\cdot)\}\}$ and $\epsilon = 10^{-5}$.

While training the MAM^2-based layers, we start with $\beta[1] = 1$ at epoch 1 and decrease it linearly to $\beta[15] = 0$ until epoch 15. After this, $\beta[p]$ is kept to 0 for $p > 15$ (i.e., we use (6)).

Accuracy trend during training can be seen in Fig. 5. As expected, when $\beta[p] > 0$ in the first 15 epochs all the contributes to the summation have not completely vanished yet. When $\beta[p] = 0$ all the contributes but the maximum and the minimum vanish, producing an initial decrease in accuracy that is quickly retrieved during the successive epochs.

E. Performance and size

The DNN-based oracle is trained and then the third and the fourth hidden layers are pruned, as they contain most of the parameters. The TCSSO decoder that employs the MAM^2-based layers is pruned using the activation rate method while the size-equivalent TCSSO that use only MAC-based layers is pruned using a magnitude-based approach, i.e., the parameters are scored according to their absolute value and the ones with the lowest scores are pruned. This is a typical approach for pruning MAC-based layers without retraining. The scores used for pruning are evaluated globally on the two layers.

Results can be found in Fig. 6, where the ARSNR of the reconstructed signal is shown against the percentage of remaining parameters in the weight matrices of the third and of the fourth hidden layers. For each of the two TCSSO configurations, 5 different training and pruning processes have been run.

The MAM^2-based implementation presents the same baseline accuracy of the MAC-based structure. When both the solutions are pruned, MAM^2 layers perform better, allowing a further and significant reduction of the size of the DNN. As an example, in Fig. 6 it is shown that the MAM^2-based layers can still retain the baseline accuracy with over 94% of their weights pruned while the MAC-based layers do the same with not more than 25% of pruned parameters. In order to further verify this result, we trained five more TCSSO configurations reducing the number of parameters without pruning, i.e., we select a lower number of neurons for the third and fourth hidden layers in the model[1]. As shown in Fig. 6, none of this models achieve the same ARSNR of the model trained and pruned with the MAM^2-based layers.

V. Conclusion

We have presented a novel DNN layer based on a MAM^2 map-reduce paradigm, naturally prone to aggressive pruning, capable of reducing the number of parameters in a DNN-based CS decoder. This new type of layer is trained through the *vanishing contributes* technique, that allows to seamlessly transform a classic MAC-based structure in a MAM^2-based one. When employed in the decoder, MAM^2-based layers pruned with the activation rate method allow a further reduction of the total number of parameters compared to classic MAC-based layers. This is achieved with no retraining thus shortening the design-to-deployment time of the decoder.

[1]We keep the ratio 2:1 used in the original TCSSO which contains 512 and 256 neurons in the third and fourth hidden layers, respectively. The number of neurons for these two layers are 112-56, 96-48, 80-40, 64-32 and 48-24, defining each of the 5 extra configurations.

REFERENCES

[1] A. Pantelopoulos and N. G. Bourbakis, "A Survey on Wearable Sensor-Based Systems for Health Monitoring and Prognosis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 40, no. 1, pp. 1–12, Jan. 2010. doi:10.1109/TSMCC.2009.2032660

[2] R. G. Baraniuk, E. Candes, R. Nowak, and M. Vetterli, "Compressive Sampling [From the Guest Editors]," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 12–13, Mar. 2008. doi:10.1109/MSP.2008.915557

[3] D. L. Donoho, "Compressed Sensing," *IEEE Trans. on Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006. doi:10.1109/TIT.2006.871582

[4] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. on Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006. doi:10.1109/TIT.2005.862083

[5] E. van den Berg and M. P. Friedlander, "Probing the pareto frontier for basis pursuit solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, Jan. 2009. doi:10.1137/080714488

[6] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *2011 IEEE international symposium on information theory proceedings*, Jul. 2011, pp. 2168–2172. doi:10.1109/ISIT.2011.6033942

[7] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007. doi:10.1109/TIT.2007.909108

[8] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, May 2009. doi:10.1016/j.acha.2008.07.002

[9] M. Mangia, L. Prono, A. Marchioni, F. Pareschi, R. Rovatti, and G. Setti, "Deep Neural Oracles for Short-window Optimized Compressed Sensing of Biosignals," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 3, pp. 545–557, Jun. 2020. doi:10.1109/TBCAS.2020.2982824

[10] L. Prono, M. Mangia, A. Marchioni, F. Pareschi, R. Rovatti, and G. Setti, "Deep Neural Oracle With Support Identification in the Compressed Domain," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 4, pp. 458–468, Dec. 2020. doi:10.1109/JETCAS.2020.3039731

[11] A. Mirrashid and A. A. Beheshti, "Compressed remote sensing by using deep learning," in *9th international symposium on telecommunications (IST)*, Dec. 2018, pp. 549–552. doi:10.1109/ISTEL.2018.8661112

[12] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "ReconNet: non-iterative reconstruction of images from compressively sensed measurements," in *IEEE conference on computer vision and pattern recognition (CVPR)*, Jun. 2016, pp. 449–458. doi:10.1109/CVPR.2016.55

[13] A. Mousavi and R. G. Baraniuk, "Learning to invert: signal recovery via deep convolutional networks," in *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, Mar. 2017, pp. 2272–2276. doi:10.1109/ICASSP.2017.7952561

[14] M. Mangia, A. Marchioni, L. Prono, F. Pareschi, R. Rovatti, and G. Setti, "Low-power ECG acquisition by compressed sensing with deep neural oracles," in *2020 2nd IEEE international conference on artificial intelligence circuits and systems (AICAS)*, Aug. 2020, pp. 158–162. doi:10.1109/AICAS48895.2020.9073945

[15] W. Shi, F. Jiang, S. Zhang, and D. Zhao, "Deep networks for compressed image sensing," in *IEEE international conference on multimedia and expo (ICME)*, Jul. 2017, pp. 877–882. doi:10.1109/ICME.2017.8019428

[16] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A deep learning approach to structured signal recovery," in *3rd annual allerton conference on communication, control, and computing (allerton)*, Sep. 2015, pp. 1336–1343. doi:10.1109/ALLERTON.2015.7447163

[17] L. Prono, M. Mangia, A. Marchioni, F. Pareschi, R. Rovatti, and G. Setti, "Low-power fixed-point compressed sensing decoder with support oracle," in *2020 IEEE international symposium on circuits and systems (ISCAS)*, Oct. 2020, pp. 1–5. doi:10.1109/ISCAS45731.2020.9180502

[18] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, "Deep fully-connected networks for video compressive sensing," *Digital Signal Processing*, vol. 72, pp. 9 – 18, 2018. doi:10.1016/j.dsp.2017.09.010

[19] J. Zhang and B. Ghanem, "ISTA-Net: interpretable optimization-inspired deep network for image compressive sensing," in *IEEE/CVF conference on computer vision and pattern recognition*, Jun. 2018, pp. 1828–1837. doi:10.1109/CVPR.2018.00196

[20] B. Sun, H. Feng, K. Chen, and X. Zhu, "A deep learning framework of quantized compressed sensing for wireless neural recording," *IEEE Access*, vol. 4, pp. 5169–5178, 2016. doi:10.1109/ACCESS.2016.2604397

[21] C. Louizos, M. Welling, and D. P. Kingma, "Learning Sparse Neural Networks through L0 Regularization," in *6th International Conference on Learning Representations (ICLR 2018)*, Feb. 2018.

[22] Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 1398–1406. doi:10.1109/ICCV.2017.155 ISSN: 2380-7504.

[23] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both Weights and Connections for Efficient Neural Network," in *28th International Conference on Neural Information Processing Systems (NIPS'15)*, vol. 28, Oct. 2015.

[24] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," in *7th International Conference on Learning Representations (ICLR 2019)*, Sep. 2018.

[25] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the Value of Network Pruning," in *7th International Conference on Learning Representations (ICLR 2019)*, Sep. 2018.

[26] L. Prono, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "A Non-conventional Sum-and-Max based Neural Network layer for Low Power Classification," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2022.

[27] P. E. McSharry, G. D. Clifford, L. Tarassenko, and L. A. Smith, "A dynamical model for generating synthetic electrocardiogram signals," *IEEE Trans. on Biom. Eng.*, vol. 50, no. 3, pp. 289–294, Mar. 2003. doi:10.1109/TBME.2003.808805

[28] D. Gangopadhyay, E. G. Allstot, A. M. R. Dixon, K. Natarajan, S. Gupta, and D. J. Allstot, "Compressed sensing analog front-end for bio-sensor applications," *IEEE J. Solid-State Circuits*, vol. 49, no. 2, pp. 426–438, Feb. 2014. doi:10.1109/JSSC.2013.2284673

[29] M. Shoaran, M. H. Kamal, C. Pollo, P. Vandergheynst, and A. Schmid, "Compact low-power cortical recording architecture for compressive multichannel data acquisition," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 6, pp. 857–870, Dec. 2014. doi:10.1109/TBCAS.2014.2304582

[30] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017.