A Multiply-And-Max/Min Neuron Paradigm for Aggressively Prunable Deep Neural Networks

A Multiply-And-Max/Min Neuron Paradigm for Aggressively Prunable Deep Neural Networks

Luciano Prono, *Member, IEEE*, Philippe Bich, *Graduate Student Member, IEEE*, Chiara Boretti, *Graduate Student Member, IEEE*, Mauro Mangia, *Member, IEEE*, Fabio Pareschi, *Senior Member, IEEE*, Riccardo Rovatti, *Fellow, IEEE*, and Gianluca Setti, *Fellow, IEEE*

***Abstract*— The growing interest in the Internet of Things (IoT) and mobile artificial intelligence applications is pushing the investigation on deep neural networks (DNNs) that can operate at the edge using low-resources/energy devices. To obtain such a goal, several pruning techniques have been proposed in the literature. They aim to reduce the number of interconnections—and consequently the size, and the corresponding computing and storage requirements—of DNNs that traditionally rely on classic multiply-and-accumulate (MAC) neurons. In this work, we propose a novel neuron structure based on a multiply-and-max/min (MAM) map-reduce paradigm, and we show that by exploiting this new paradigm it is possible to build naturally and aggressively prunable DNN layers, with a negligible loss in performance. This novel structure allows a greater interconnection sparsity when compared to classic MAC-based DNN layers. Moreover, most of the already existing state-of-the-art pruning techniques can be used with MAM layers with little to no changes. To test the pruning performance of MAM, we employ different models—AlexNet, VGG-16 and the more recent ViT-B/16—and different computer vision datasets—CIFAR-10, CIFAR-100, and ImageNet-1K. Multiple pruning approaches are applied, ranging from single-shot methods to training-dependent and iterative techniques. As a notable example, we test MAM on the ViT-B/16 model fine-tuned on the ImageNet-1K task and apply one-shot gradient-based pruning. We remove interconnections until the model experiences a 6% decrease in accuracy. While the selected MAC-based layers need at least 38.2% remaining interconnections, MAM-based layers achieve the same accuracy with only 0.1%.**

***Index Terms*— Computer vision, deep neural network (DNN), map-reduce, pruning, vision transformer (ViT).**

Received 20 February 2023; revised 7 December 2023 and 12 July 2024; accepted 1 January 2025. Date of publication 17 January 2025; date of current version 6 August 2025. This work was supported by the Future Artificial Intelligence Research (FAIR) through European Union Next-Generation EU (Piano Nazionale di Ripresa e Resilienza (PNRR)–Missione 4 Componente 2, Investimento 1.3–D.D. 1555 11/10/2022) under Grant PE00000013. *(Corresponding author: Luciano Prono.)*

Luciano Prono, Philippe Bich, and Chiara Boretti are with the Department of Electronic and Telecommunication, Politecnico di Torino, 10129 Turin, Italy (e-mail: luciano.prono@polito.it; philippe.bich@polito.it; chiara.boretti@polito.it).

Mauro Mangia and Riccardo Rovatti are with the Department of Electrical, Electronic, and Information Engineering, University of Bologna, 40136 Bologna, Italy, and also with the Advanced Research Center on Electronic Systems (ARCES), University of Bologna, 40125 Bologna, Italy (e-mail: mauro.mangia@unibo.it; riccardo.rovatti@unibo.it).

Fabio Pareschi is with the Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Turin, Italy, and also with the Advanced Research Center on Electronic Systems (ARCES), University of Bologna, 40125 Bologna, Italy (e-mail: fabio.pareschi@polito.it).

Gianluca Setti is with the King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia (e-mail: gianluca.setti@kaust.edu.sa).

Digital Object Identifier 10.1109/TNNLS.2025.3527644

I. Introduction

DEEP neural networks (DNNs) are structures capable of solving complex tasks with the use of a massive number of trainable parameters. They are in such a large number that they result in being greatly redundant. Hence, it is possible to remove the unnecessary parameters by pruning parts of the structure that do not appreciably influence the accuracy of the DNN [1].

Pruning is a necessary operation to achieve low-power and lightweight inference, which is fundamental for the implementation of neural networks on mobile devices with limited battery capacity and constrained computational capabilities [2]. The use of DNNs at the edge on mobile Internet of Things (IoT) devices is of great interest due to the high potential it unlocks [3], [4], [5] and for its wide range of applications from augmented reality [6], natural language processing [7], computer vision [8], and compressed sensing for biomedical signals [9].

Classical pruning approaches in the literature [1] simply leverage methods to select the interconnections or entire neurons to be pruned in a DNN without modifying its inherent structure, which is based on the typical multiply-and-accumulate (MAC) paradigm. In standard MAC-based neurons, the output is computed by first modulating inputs independently of each other (*map* operation), then by aggregating the outcomes into a single quantity (*reduce* operation), and finally by adjusting the value through an activation function. For these classical neurons, *map* is multiplication, while *reduce* is accumulation. It is worth noting that once a MAC-based neuron is pruned, the map operation produces fewer outputs since fewer inputs are taken into consideration and this can significantly modify the output of the neuron. Hence, our idea is to find a *different reduce operation* that is less sensible to this reduction of elements.

To do so, let us consider that, in principle, sum (accumulate) may be seen as a special case of a general aggregation of N

quantities v_i, whose resulting output o is computed as

$$o = \left(\sum_{i=1}^{N} v_i^p \right)^{1/p}.$$

Here, the parameter p controls how much the largest v_i prevails on the smallest ones (or vice versa) in deciding the result: when $p = 1$ one obtains the sum and the contribution is equal, when $p \to \infty$ is the maximum, while when $p \to -\infty$ is the minimum.

This observation drives our intuition to modify the sum-based reduce with a *max/min-based reduce*, leading to a new model of a neuron, relying on our novel multiply-and-max/min (MAM) paradigm, which is capable of explicitly identifying which inputs are used to compute the output and which are not. This, to the best of our knowledge, is an unprecedented route to non-structured pruning. In fact, such an apparently simple modification has the advantage of making the behavior of the individual neurons (and, as we will see, of the resulting network) less sensitive to pruning. At the same time, this neuron remains easily implementable in hardware, without introducing a significant computational overhead compared to the use of classical neurons based on the sum operation. This is because the multiplication operation is unchanged, while the comparison operations, which are required to select the maximum and minimum values, are performed through successive subtractions, i.e., computationally similar to the accumulation operation. We better elaborate on this in Appendix A.

The equivalence in functionality between the original and the alternative neurons cannot be assured a priori as it is not on a neuron-by-neuron basis. Nonetheless, we will show that the "same" global behavior of the DNN can be retrieved after the substitution of MAC-based neurons with MAM-based ones for the whole network. Contrary to MAC, our paradigm is found to be naturally prone to pruning as each neuron typically learns to select always the same few interconnections when performing the maximum and minimum reduction operations. At the same time, almost any pruning algorithm already available in the literature that can be applied to MAC-based layers can be applied to MAM-based ones with little to no changes, to boost its performance.

Focusing on fully connected (FC) layers built upon MAM neurons, the following points summarize the novelty of this work.

1) We formalize the definition of the MAM operation and the *vanishing contributions* training approach.
2) We present MAM pruning properties by means of a driving example: a DNN trained on MNIST.
3) We show how MAM-based networks outperform standard DNN adopting MAC neurons in terms of pruning performance for some well-known computer vision DNN models (AlexNet and VGG-16) and on a more recent transformer-based model (ViT-B/16).
4) We analyze the performance of MAM with AC/DC [10] and the *lottery ticket* [11] pruning approaches, to further show how it is possible to use MAM in conjunction with training-dependent pruning methods.
5) With the *lottery ticket* iterative pruning approach, we show that employing a MAM-based DNN drastically reduces the number of training processes needed to prune the initial architecture.

The rest of the article is structured as follows. In Section II, we provide an extended description of the novel MAM map-reduce paradigm together with an explanation of the vanishing contributions technique. Then, in Section III, we introduce an overview of DNN pruning and describe some pruning scoring strategies. In Section IV, we present a first insight into the properties of MAM by training and pruning a simple FC-based model. In Section V, we show extensive comparisons between MAM layers and classic MAC layers used in multiple DNN structures and with multiple image classification datasets, pruned with non-structured one-shot pruning methods. In Section VI, we show how MAM can also be coupled with more complex pruning techniques such as AC/DC [10] and the lottery ticket iterative pruning [11]. Finally, the conclusion is drawn.

II. MAM-Based Layer Architecture

In this section, we describe our novel MAM layer architecture, which is naturally prone to aggressive pruning. After a formal definition of the novel DNN layer, we explain the *vanishing contributions* technique that is used for training. In this work, we focus on FC layers to ease the analysis. In fact, we intend MAM to be used exclusively for FC layers; deploying it on convolutional layers would require further analysis.

A. Layer Description

In a standard DNN, we define the output of a feed-forward FC MAC-based layer as a column vector $\boldsymbol{y} \in \mathbb{R}^M$ computed as

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \tag{1}$$

$$\boldsymbol{y} = f(\boldsymbol{z}). \tag{2}$$

Here, $\boldsymbol{x} \in \mathbb{R}^N$ is the input column vector, $\boldsymbol{W} \in \mathbb{R}^{M \times N}$ is the weights matrix, $\boldsymbol{b} \in \mathbb{R}^M$ is the bias column vector, and $f(\cdot)$ is the activation function, which is applied element-wise to $\boldsymbol{z}$ to obtain the output vector $\boldsymbol{y}$.

As we seek a new family of neurons changing the aggregation phase, we need to explicitly separate the multiply and the reduce operations of (1). To do so, first, we isolate the map phase (i.e., multiply) operation by defining the *weighted inputs* matrix $\boldsymbol{V} \in \mathbb{R}^{M \times N}$ as

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{w}_1 \odot \boldsymbol{x}^\top \\ \boldsymbol{w}_2 \odot \boldsymbol{x}^\top \\ \vdots \\ \boldsymbol{w}_M \odot \boldsymbol{x}^\top \end{bmatrix} \tag{3}$$

where $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M$ is the ith row of $\boldsymbol{W}$ and $\odot$ is the element-wise product. More explicitly

$$v_{ij} = w_{ij} x_j \quad \text{with } i \in \{1, .., M\},\ j \in \{1, .., N\} \tag{4}$$

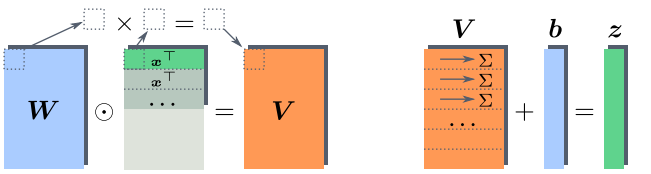

Fig. 1. Split of the multiply-and-accumulate operations for the MAC map-reduce paradigm. The classic MAC operation as a matrix–vector dot product $\boldsymbol{W}\boldsymbol{x}$ is decomposed in the multiply operation (a), where we multiply element-wise each row of $\boldsymbol{W}$ with $\boldsymbol{x}^\top$, and in the accumulate operation (b) where the elements of $\boldsymbol{V}$ are summed along the rows of the matrix.

where v_{ij} and w_{ij} are the scalars at row i and column j of $\boldsymbol{V}$ and $\boldsymbol{W}$, respectively, and x_j is the jth element of the input vector $\boldsymbol{x}$. With this notation, the reduce phase can be simply indicated as

$$z_i = \sum_{j=1}^{N} v_{ij} + b_i \quad \text{with } i \in \{1, .., M\} \tag{5}$$

where z_i is the ith element of vector $\boldsymbol{z}$. Fig. 1 shows a summary of the multiply-and-accumulate splitting.

It is worth noting that from the point of view of the architecture, each weight in w_{ij} (and, consequentially, each value v_{ij}) is associated with an interconnection whose spatial properties are determined by i, indicating the *postsynaptic* connection, and j, indicating the *presynaptic* connection. With reference to (5), the pruning process has the meaning of removing the interconnections associated with the less significant contributions from the accumulate operation, which are typically the values of $\boldsymbol{V}$ with the smallest magnitudes. When pruning the largest possible amount of interconnections, only a couple per row of $\boldsymbol{V}$ is kept.[1]

Following this lead, we force each neuron of the layer to take into account only the (potentially) two most significant contributions, that is to say, the maximum and the minimum values of each row of $\boldsymbol{V}$, as suggested in [12], [13]. It is important to stress that this operation does not remove any interconnection, i.e., it is not a pruning method. Each time an input is inferred, matrix $\boldsymbol{V}$ is evaluated (so all the weights contribute). Only *then*, maximum and minimum are selected for each neuron (and they are not necessarily the same for two different inputs). The result is that, for each different input vector, the maximum and minimum values that are selected are different, generating different sub-nets for each inference.

Furthermore, the reason to use maximum and minimum—and not, for example, the couple of values with the largest absolute values—is twofold. First, by doing so, the hardware implementation of the corresponding neuron will have a small to negligible overhead compared to the one of traditional MAC-based neurons (more on this matter in Appendix A). Second, the two contributions are capable of balancing themselves,[2] resulting in a good stability of the neuron output. With this choice, (5) modifies as

$$z_i = \mathop{\mathcal{M}}_{j=1}^{N} v_{ij} + b_i \quad \text{with } i \in \{1, .., M\} \tag{6}$$

where $\mathcal{M}$ is a suitable operator defined as

$$\mathop{\mathcal{M}}_{j=1}^{N} v_{ij} \triangleq \max_{j\in\{1,..,N\}} v_{ij} + \min_{j\in\{1,..,N\}} v_{ij}. \tag{7}$$

We will refer to the map-reduce operations in (3) and (6) as the Multiply-And-Max/min paradigm (MAM). These equations, together with (2), describe the novel MAM-based layer, while Fig. 2 shows a visual representation of (6).

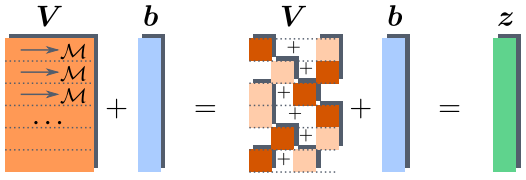

Fig. 2. Max/min reduce operation. For each row of $\boldsymbol{V}$, the minimum and maximum values are selected, then added together.

B. Bridge From MAC to MAM: The Vanishing Contributions Method

In general, training DNN layers directly on the basis of (6) results in poor performance. In fact, as (6) forward-propagates exactly two values for each row of $\boldsymbol{V}$, only the weights associated with them can be updated. This means that only a few weights can be updated for each training iteration, greatly slowing the optimization process. Consequently, to ease the training process of MAM-based layers, we define a layer structure that is the bridge between the MAC and the MAM layers. This can be obtained by introducing a parameter $\beta \in [0, 1]$, so that (5) and (6) can be affinely combined as

$$z_i = \beta \sum_{j=1}^{N} v_{ij} + (1-\beta) \mathop{\mathcal{M}}_{j=1}^{N} v_{ij} + b_i \quad \text{with } i \in \{1, .., M\}. \tag{8}$$

During training, by using (8), it is possible to gradually transition from a MAC-based to a MAM-based layer. This enables, in the first phase of the training, the backward propagation of the output error through all the interconnections. In particular, this is achieved in the following way. Starting with $\beta = 1$ during the first training iterations, the value of β is linearly reduced[3] for each training iteration until it reaches $\beta = 0$. When $\beta = 0$, (8) is reduced to (6) and β is kept fixed for the rest of the training iterations, keeping only the contributions of MAM. We call this the *vanishing*

[1] Keeping a single contribution would make the reduce operation collapse to an identity function.

[2] Empirical evidence shows that the maximum is typically associated with a large positive value, while the minimum to a large but negative value.

[3] Many nonlinear strategies have also been tested to transition from $\beta = 1$ to $\beta = 0$ and each of them has shown a different impact on the trend of the validation accuracy during training. However, every strategy seems to yield the same final accuracy. All the results presented in this work are obtained using the linear decay.

contributions method. The usage of this technique allows us to obtain a much better performance of the resulting DNN when compared to the previous embryonic work in [12], where the paradigm based on max and min is trained as-it-is. It is important to highlight that (8) is used only during the training phase, while pruning of the layers and test inferences are performed strictly with (6) (i.e., $\beta = 0$). Moreover, this technique allows us to use pretrained MAC-based models and convert them into MAM neurons without the need to retrain the network anew, which is often impractical for very large models.

III. Classification of Pruning Techniques

To profitably frame MAM and its application in the field of DNN pruning, we introduce an overview of the classification of the pruning methods proposed in the literature.

From the general point of view, DNN pruning techniques can be categorized into two main classes of methods, namely *structured* and *non-structured*. Structured pruning [14], [15], [16], [17], [18] refers to the removal of entire neurons—or filters/channels, in the case of convolutional layers. Removing a neuron means taking out all the trainable parameters related to it, namely the input weights and the bias. While these techniques are typically not the most effective in literature, they de facto nullify the necessity of designing ad hoc implementations for the pruned model, resulting in totally hardware-friendly methods. In contrast, non-structured pruning methods [19], [20] allow the removal of single interconnections (and, consequentially, of the related weights), granting more degrees of freedom to the pruning process and resulting in better compression results. Nonetheless, this comes with an in-hardware computational/memory overhead due to the encoding of the resulting sparse weights matrices [21], [22], [23].

Different criteria can be employed to select the neurons/interconnections to be removed. As a first criterion, a *score* is associated with each interconnection depending on the influence it has on the model and the interconnections with the lowest scores are pruned. Score comparison can be performed *locally* [24], i.e., among the interconnections of the same DNN layer, or *globally* [11], i.e., among the interconnections of the whole DNN model. As a second criterion, a *regularization* term [25], [26], [27] can be imposed on the loss function during training to promote the sparsity of the structure.

The DNN pruning process can be scheduled in different ways, mainly it can be performed *one-shot* or *iteratively*. One-shot pruning [28], [29] is performed in one step after the model has been trained and no further fine-tuning is performed afterward. This results in very fast and computationally efficient pruning processes, even though it requires waiving deeper optimizations. On the other hand, iterative pruning [11], [30], [31] is performed iteratively through multiple "train-and-prune" loops. Pruning at each step only a small part of the DNN and then retraining brings very good compression results, with the drawback of a very high total computational cost for the generation of the compressed model.

Finally, other examples of pruning techniques non-exhaustively include knowledge extraction from feature maps [32], neuroevolution methods that take inspiration from biological mechanisms [33], and the global capture of correlations among layers with second-order structured pruning [34]. An alternative to pruning to design small neural models starting from large ones is instead represented by knowledge distillation [35].

While virtually any of the aforementioned classes of pruning methods could be applied to MAM-based DNNs, in this work, we focus on non-structured, score-based pruning techniques. This is mainly because, as we will see: 1) the MAM paradigm naturally promotes a non-structured sparsity of the DNN layers and 2) trained MAM-based structures are intrinsically sparse, avoiding the need of introducing regularization terms during the training process (the vanishing contributions technique does not require any additional terms in the cost function). Nonetheless, we test MAM-based structures both with local and global scores and with one-shot and iterative pruning techniques.

A. Scoring Strategies for Pruning

As stated before, MAM-based structures are designed to be pruned—until proven otherwise—with any pruning method present in the literature. In particular, we test the MAM-based structures in combination with non-structured, score-based pruning techniques. To keep the analysis simple, we adhere to the criteria indicated in [1]. Additionally, we introduce a criterion that leverages the properties of the operator $\mathcal{M}$, whose purpose is the better understanding of MAM properties.

1) Random Scores: With the random pruning (RP) approach, the interconnections are randomly scored. This means that each interconnection is pruned independently with a probability equal to the fraction of the network that is being pruned. This method is used as a "debugging tool" to verify the effectiveness of the other pruning methods, as stated in [1].

2) Magnitude-Based Scores: Starting from the assumption that the smaller the magnitude of weight is, the less it potentially influences the output of the DNN, one of the simplest ways to define the scores for interconnection is by looking at the magnitude of its associated weight (that is, its absolute value). The corresponding score function is

$$g(\iota) = |w_\iota|$$

where ι is an interconnection and w_ι is the weight associated with it. The score comparisons can be global or local, resulting in two distinct approaches, namely global magnitude pruning (GMP) and layer-wise magnitude pruning (LMP), respectively. GMP does not impose constraints on the pruning process, whereas LMP results in equally pruned layers.

3) Probability of Selection Pruning: We introduce this method solely to show the properties of MAM neurons. In fact,

the pruning performance of this scoring method is worse compared to the strategies previously described. When performing a single inference through a MAM layer, operator $\mathcal{M}$ selects only two values for each row of $\boldsymbol{V}$, as Fig. 2 shows and as in (6). In this way, when inferring a single input, we know that most of the interconnections—each associated with a value in $\boldsymbol{V}$ and, by extension, to a weight in $\boldsymbol{W}$—are not used. Using this property, it is possible to define a new score to be assigned to each interconnection. By using a dedicated *pruning dataset*.[4] and for each inferred input, we keep track of the interconnections that are chosen. Over the whole dataset, it is possible to evaluate for each interconnection the approximate probability of being selected by operator $\mathcal{M}$, thus defining a set of scores that defines how often an interconnection is used by the DNN.

4) Gradient-Based Scores: While GMP and LMP are quite effective approaches with very straightforward implementations, they do not take into account the statistics of the data that is fed to the DNN as a potential information to guide the pruning process. To alleviate this problem, the loss function $C(\boldsymbol{x}, \text{DNN})$ is defined and evaluated on the model DNN with input $\boldsymbol{x}$. Each interconnection ι is scored as

$$g(\iota) = \mathbf{E}\left[\left|\frac{\partial C(\boldsymbol{x}, \text{DNN})}{\partial w_\iota} w_\iota\right|\right].$$

Here, the operator $\mathbf{E}[\cdot]$ evaluates the expected value on all the inputs $\boldsymbol{x}$ of a dedicated *pruning dataset*. This approach is called global gradient pruning (GGP) and layer-wise gradient pruning (LGP) for its global and local variants, respectively. The gradient is evaluated on a dedicated dataset (which is not the test set on which we evaluate the final results). This method is especially powerful with MAM neurons because it implicitly incorporates the probability of selection pruning (PSP) method. In fact, with MAM, when an interconnection is not selected, the associated gradient is 0 (and so is the associated score). This means that the interconnections that are selected less often are inclined to have lower scores. In practice, GGP and LGP incorporate three different pieces of information that compose the final scoring, namely the information on the magnitude of the weights, their influence on the cost function (gradient) and the probability of selection in MAM neurons. This makes GGP and LGP the most effective scoring approaches.

IV. First Insight Into MAM Properties

As a first example of how MAM-based layers can be used to obtain aggressively pruned networks, we employ a simple DNN composed of three FC layers trained on the MNIST dataset [36]. The two hidden layers contain 256 neurons each with a ReLU activation function. Fig. 3 shows the model which is defined as FC-DNN, where FC-DNN-MAC is the version with MAC-based hidden layers and FC-DNN-MAM the one with MAM-based hidden layers.

For any model in this work, the output layer is always MAC-based, which is needed to keep a good accuracy in the model.

[4]For any pruning method requiring a pruning dataset, we use the validation set and we evaluate the final result with the test set (so we guarantee the generalization capabilities of the approaches).

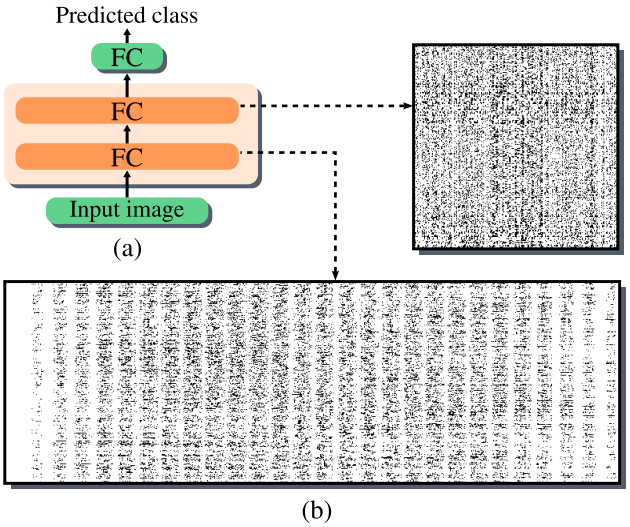

Fig. 3. Structure of the three FC layers DNN for MNIST classification (FC-DNN) (a). MAM is deployed in the highlighted FC layers. The whole network contains 269 322 trainable parameters. We also show the maps of the probability of selection (b) for matrices $\boldsymbol{V}$ of the two hidden layers. Black dots correspond to values of $\boldsymbol{V}$ used at least once (i.e., with non-zero probability), corresponding to 15.7% of the total interconnections for the first layer and 19.0% for the second.

In fact, we found in practice that we cannot always guarantee the best accuracy of the model when the output layer is MAM-based, with an accuracy drop ranging from 1% to more than 5% for different models. We can assume that this is due to the fact that *at least* the output layer needs to actually combine all the contributions in the network (which MAM does not, since it combines only the maximum and minimum contributions per neuron).

While the exact reason behind this behavior is yet to be analyzed, this constraint does not significantly influence the overall pruning performance, as the number of interconnections in the output layer is typically low.

We first study the behavior of the MAM layers by using the PSP scoring approach. Then, to prune the two structures for comparison, we employ GMP, LMP, GGP, and LGP scoring approaches as discussed in Section III-A.

A. Training Setup and Behavior of MAM Layers

Through this work, training processes and tests are performed within a PyTorch framework, with the MAM operation implemented in C++ as a custom PyTorch layer.[5]

During training, pairs of images and labels from the MNIST dataset, which is composed of a total of 70 000 b/w images of size 28×28, are fed to FC-DNN. Each pixel in the image is a value normalized between 0 and 1. To improve the generalization capabilities of the trained network, data augmentation is performed. This in particular consists of image rotation, shifting, scaling, horizontal and vertical shrinking, and shearing [36]. The trainable parameters are updated using the Adam optimizer [37]. Overall, training with the FC-DNN + MNIST is performed for 80 epochs with an initial learning rate set to 0.001 and a batch size of 128.

[5]GitHub repository at https://github.com/SSIGPRO/mamtorch

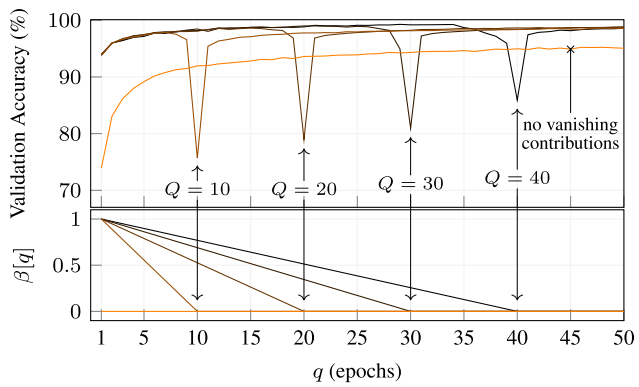

Fig. 4. Top-1 accuracy trend on the validation set during the training of the FC-DNN-MAM with MNIST and $\beta[q]$ trend. Different values of Q are tested. After an initial accuracy increase, there is an accuracy drop when $\beta[Q] = 0$, but the DNN quickly recovers its classification capability. With Q big enough, the sensibility of the final accuracy with respect to Q is not significant.

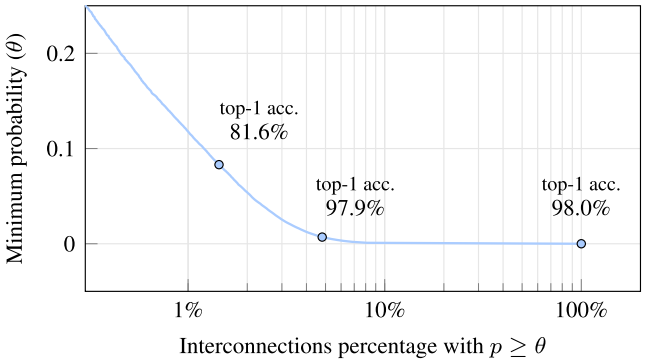

Fig. 5. FC-DNN-MAM: For different threshold values of θ, the plot shows how many interconnections (in percentage) have a probability of selection $p \geq \theta$. Also, top-1 accuracy is shown when keeping only the aforementioned interconnections. For $\theta = 0$, any interconnection is kept. For $\theta > 0$, most interconnections are removed, meaning that *i)* most interconnections have $p = 0$ and are never used and *ii)* most interconnections can be safely removed without affecting the performance.

Training MAM layers requires starting with standard MAC-based layers and then transitioning to MAM following (8), defined by the vanishing contributions method of Section II-B. More in detail, we define for each training epoch q a value $\beta[q]$ to be substituted to β in (8). We start with $\beta[1] = 1$ at epoch 1 and we decrease it linearly to $\beta[Q] = 0$ at the Qth epoch. After this, $\beta[q]$ is set to 0 for $q > Q$.

Fig. 4 shows the accuracy trend during training with different values of Q.

When $\beta[q] > 0$ in the first Q epochs, the MAC contribution from (8) is still present and the validation accuracy increases epoch by epoch as expected. When $\beta[q] = 0$, the MAC contribution disappears completely causing a sudden performance drop. The following epochs readjust the parameters and the MAM-only structure recovers its accuracy. Nonetheless, as convergence with MAM is slower compared to standard MAC, full recovery of the performance requires some extra epochs.

As a rule of thumb, Q should be chosen small but greater than a few epochs. When Q is too small, the training process fails to train the DNN to its maximum possible accuracy, while on the other hand, high values of Q make the convergence slow. For FC-DNN-MAM + MNIST, we set $Q = 30$.

When training is completed, on the test set and without pruning, FC-DNN-MAC achieves 98.8% top-1 accuracy and FC-DNN-MAM achieves 98.0% top-1 accuracy. Even though FC-DNN-MAC performs slightly better than FC-DNN-MAM, we reiterate that the aim of MAM-based structures is to be subject to aggressive pruning to be implemented on low-cost edge devices with limited computational power. Because of this, to make a complete comparison, we need to look at the performance of the two models when they are pruned.

B. Natural Sparsity of MAM Layers

In order to highlight the sparsification properties of MAM, we apply the PSP scoring approach from Section III-A to the trained DNN. The validation set, employed as pruning set, is used as input, and the probability of selection for each element of matrix $\boldsymbol{V}$ is evaluated for each layer (we still evaluate the final accuracy on a test dataset, different from the validation/pruning dataset). In Fig. 3, we highlight the positions of the values in $\boldsymbol{V}$ with non-zero probability for the two MAM-based layers, showing that most values have a null probability of being selected. This suggests how MAM results in an intrinsically sparse structure—and because of this, it is prunable. In another analysis, it is possible to consider/keep only the interconnections with a probability of selection larger than a given threshold (i.e., those with a large enough probability of being used in the computation of the output). Fig. 5 shows the trend of kept interconnections with a varying probability threshold, showing that only a few interconnections have a high probability of being used. For example, we can see from the plot that only 10% of the interconnections have a probability of selection higher than 0 and only 1% of the interconnections have a probability of selection of at least 0.1. Along with that, we also highlight top-1 accuracy at different working points, showing that the performance drops only when we start removing interconnections that have a non-negligible probability of being used.

These sparsification properties can be measured in practice when pruning the DNN with GMP, LMP, GGP, or LGP from Section III-A. Table I shows the remaining weights in the models after pruning, maintaining top-1 accuracy above the best result (98.8% achieved with FC-DNN-MAC) reduced by 3%. For the sake of completeness, we also inserted PSP for the MAM layers: as stated before, the results are worse compared to the other approaches. Finally, the number of FLOPs is indicated, counting two FLOPs per weight for MAC (multiply and addition) and three FLOPs per weight for MAM (multiply, comparison for the search of maximum and comparison for the search of minimum). Further details on the number of FLOPs can be found in Appendix A.

To better highlight the behavior of MAM, Fig. 6 shows the variation of top-1 accuracy with a decreasing number of

TABLE I
PERCENTAGE OF REMAINING WEIGHTS AND FLOPS IN THE PRUNED FC LAYERS WHEN THE ACCURACY IS 95.8% (3% ACCURACY LOSS)

	FC-DNN + MNIST			
	MAC		MAM	
	Weights kept	kFLOPs	Weights kept	kFLOPs
GMP	35.84%	191.35	4.64%	38.08
LMP	37.47%	200.03	4.52%	37.13
GGP	19.22%	102.86	2.98%	24.83
LGP	20.47%	109.51	**2.61%**	**21.87**
PSP	–	–	6.40%	53.63

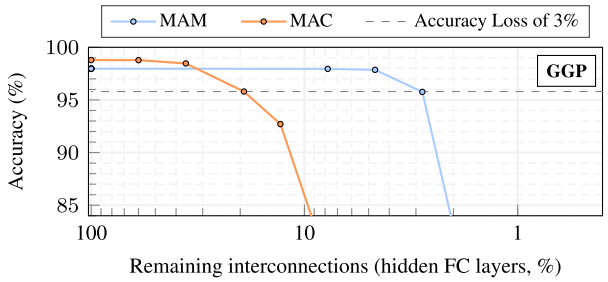

Fig. 6. FC-DNN + MNIST + one-shot pruning: Top-1 accuracy versus percentage of kept interconnections in the two hidden FC layers when pruned with GGP. The dashed line indicates when the accuracy is 95.8% (i.e., 3% accuracy loss). Interconnections are increasingly removed going from left to right.

kept interconnections. If we focus on the knees of the curves, when the number of remaining interconnections is low and the performance degradation is still small, we see that MAM layers retain the classification capabilities of the network much better compared to standard MAC.

V. PRUNING PERFORMANCE OF MAM LAYERS

In this section, we expand the discussion on MAM-based DNN layers and we show a comparison between MAC-based and MAM-based FC layers on models commonly known in the literature for computer vision tasks, i.e., with structures capable to solve image classification tasks.

A. Computer Vision Benchmark Datasets

We perform benchmarks on three different computer vision datasets, namely CIFAR-10, CIFAR-100 [38], and ImageNet-1K [39].

1) *CIFAR-10:* This task involves in the classification of animals and vehicles among ten different classes. The dataset consists of a total of 60 000 color images of size 32×32. It is split into 45 000 training images, 5000 validation images, and 10 000 test images. Data augmentation is performed and consists of random image rotation, shifting, and changes in saturation, contrast, and brightness. Each pixel in the images is normalized between 0 and 1, and all samples are resized to be of size 224×224 using bilinear interpolation.
2) *CIFAR-100:* This task involves in the classification of items belonging to 100 different classes. It consists of a total of 60 000 color images of size 32×32. It is split into the same number of images for training, validation, and test, as in CIFAR-10. The same data augmentation employed for CIFAR-10 is used. Each pixel in the image is normalized between 0 and 1, and all samples are resized using a bilinear interpolation to dimensions of 224×224.
3) *ImageNet-1K:* This dataset contains more than 1.2 million RGB images of different sizes representing objects belonging to 1000 different classes. The training set is composed of 1 281 167 images, while the validation and the test set are 50 000 images each.[6] Data augmentation on the training set is performed by means of random horizontal flip, random saturation, and random changes in contrast and brightness. All the images are resized to 224×224 pixels using a bilinear interpolation.

B. Deep Neural Models and Training

Here, we present the computer vision models used for the tests. They are represented in Fig. 7. As stated in Section II, to simplify our analysis, we do not discuss the pruning of convolutional layers and we focus on the portions of DNN that contain most of the interconnections, namely, the FC layers. The models are as follows.

1) *AlexNet:* This is a rather outdated convolutional neural network used for computer vision tasks [40]. It is composed of five convolutional layers and two FC hidden layers, followed by an output FC layer. It uses 47.2 million interconnections. The two FC hidden layers contribute to 90% of the interconnections in the model and are the ones that we substitute with MAM layers. We refer to the original MAC-based model as AlexNet-FCMAC and to the model containing MAM FC layers as AlexNet-FCMAM. We perform a full training of AlexNet + CIFAR-10 and AlexNet + CIFAR-100 for 70 epochs, with an initial learning rate of 0.001 and a batch size of 256. Dropout after the two hidden FC layers is set to 0.5 for AlexNet-FCMAC as indicated in [40], and to 0.2 for AlexNet-FCMAM, with $Q = 12$ for the vanishing contributions method.
2) *VGG-16:* This is a large convolutional neural network with much greater approximation capabilities compared to AlexNet [41]. It is composed of ten convolutional layers and two FC hidden layers, followed by an output FC layer. It uses 138.4 million interconnections. As for AlexNet, the two FC layers contribute to most of the interconnections in the model (86%) and are the ones that we substitute with MAM layers. We refer to the original MAC-based model as VGG-16-FCMAC and to the model containing MAM FC layers as VGG-16-FCMAM. VGG-16 + CIFAR-100 and VGG-16 + ImageNet-1K are trained

[6]As the labels of the official ImageNet test set are not disclosed, we used the official validation set as the test set and 50 000 images from the training as the validation set to ease the evaluation process.

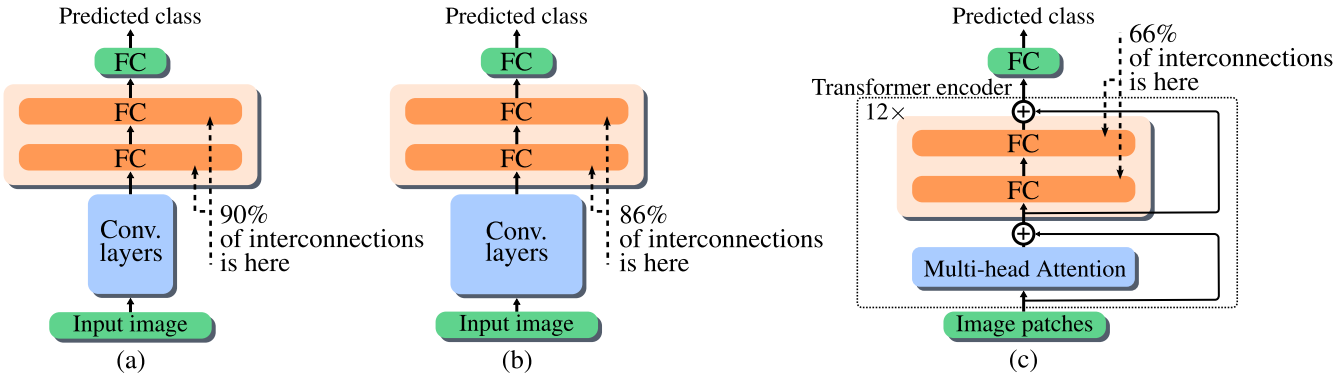

Fig. 7. DNN models tested in this work, namely (a) AlexNet, (b) VGG-16, and (c) ViT-B/16. The highlighted FC layers, which contain most of the interconnections in the networks, are the ones in which we employ MAM and that we ultimately prune. In (c), we prune the FC layers of 10 out of 12 sequential blocks.

TABLE II
TOP-1 ACCURACY OF THE NON-PRUNED NETWORKS WHEN TRAINED WITH MAC OR MAM NEURONS ON DIFFERENT DATASETS

	AlexNet		**VGG-16**		**ViT-B/16**	
	MAC	MAM	MAC	MAM	MAC	MAM
CIFAR-10	90.34%	89.88%	–	–	–	–
CIFAR-100	65.52%	64.63%	89.56%	89.12%	92.57%	87.23%
ImageNet-1k	–	–	63.36%	63.04%	80.06%	75.62%

starting from the pre-trained version of the VGG-16 model.[7] First, the trainable parameters of the hidden FC layers are randomly reinitialized. Training is then performed using stochastic gradient descent (SGD) for 45 epochs. Only for VGG-16 + ImageNet-1K, the parameters of the convolutional layers are kept fixed during training for the first 35 epochs. We set the learning rate to 0.001, the momentum coefficient to 0.9, and the batch size to 256. Dropout after the two hidden FC layers is set to 0.5 for VGG-16-FCMAC and to 0.15 for VGG-16-FCMAM. For MAM-based layers, we employ the vanishing contributions method with $Q = 5$.

3) *ViT-B/16:* This is a modern, transformer-based DNN, which leverages the self-attention mechanism on computer vision tasks [42], [43]. It is composed of 12 sequential transformer encoders. Each transformer encoder contains a multihead attention block followed by two hidden FC layers (two skip connections are also present, as shown in Fig. 7). An FC output layer follows the 12 encoders. As for AlexNet and VGG-16, we substitute MAM in each of the two hidden FC layers highlighted in Fig. 7 with the exception of the first two.[8] Overall, ViT-B/16 uses approximately 86 million interconnections, of which 56% is in the hidden FC layers that we substitute with MAM. We refer to the original MAC-based model as ViT-B/16-FCMAC and to the model containing MAM FC layers as ViT-B/16-FCMAM. The MAC and the MAM-based models are both trained starting from the available MAC-based network[9] that has been pre-trained on ImageNet-21K [39]. We fine-tune the model for 50 epochs and we use the Adam optimizer with an initial learning rate of 1×10^{-4} and each time there is no progress on validation for 3 epochs, the learning rate is multiplied by 0.3. Furthermore, we use a batch size of 128 and mix-up data-augmentation technique [44] with a strength (α parameter) of 1.0. In the case of the MAM-based model, the initial 12 epochs are used to complete the vanishing contributions transition.

[7]We used the VGG-16 + ImageNet-1K PyTorch pre-trained model.

[8]The insertion of MAM in the first two blocks results in a significant reduction of the overall accuracy.

C. Pruning Performance

The models are trained with both MAM FC layers and MAC FC layers, and Table II shows the results of the *non-pruned* models. As mentioned before, models containing MAM layers perform worse compared to standard models, but this is within expectation since MAM neurons' purpose is not performance but granting good pruning capabilities to the DNN.

After training, for each model under test, we prune the selected hidden FC layers to compare the performance of MAM layers versus standard MAC. We first test the models with the RP scoring approach as suggested in [1], and we confirm that randomly pruning the interconnections gives bad results, i.e., the performance drops after a few interconnection removals. Then, we proceed to evaluate the performance with

[9]We used the ViT-B/16 Hugging Face pre-trained model.

TABLE III

PERCENTAGE OF REMAINING WEIGHTS AND FLOPS IN THE PRUNED FC LAYERS OF THE ALEXNET MODEL (47.2 MILLION INTERCONNECTIONS) AT 3% TOP-1 ACCURACY LOSS (87.34% FOR CIFAR-10 AND 62.52% FOR CIFAR-100)

	AlexNet + CIFAR-10				**AlexNet + CIFAR-100**			
	MAC		MAM		MAC		MAM	
	Weights kept	MFLOPs	Weights kept	MFLOPs	Weights kept	MFLOPs	Weights kept	MFLOPs
GMP	1.01%	1.11	0.06%	0.11	25.01%	27.28	0.26%	0.44
LMP	0.94%	1.03	0.07%	0.13	26.05%	28.42	0.30%	0.51
GGP	0.37%	0.41	0.04%	0.08	17.95%	19.58	**0.21%**	**0.36**
LGP	0.48%	0.53	**0.03%**	**0.06**	12.68%	13.84	0.24%	0.41

TABLE IV

PERCENTAGE OF REMAINING WEIGHTS AND FLOPS IN THE PRUNED FC LAYERS OF THE VGG-16 MODEL (138.4 MILLION INTERCONNECTIONS) AT 3% TOP-1 ACCURACY LOSS (86.56% FOR CIFAR-100 AND 60.36% FOR ImageNet-1K)

	VGG-16 + CIFAR-100				**VGG-16 + ImageNet-1K**			
	MAC		MAM		MAC		MAM	
	Weights kept	MFLOPs	Weights kept	MFLOPs	Weights kept	MFLOPs	Weights kept	MFLOPs
GMP	8.71%	20.83	0.012%	0.06	10.82%	25.88	**0.04%**	**0.16**
LMP	3.80%	9.09	0.01%	0.05	10.36%	24.78	0.07%	0.27
GGP	0.30%	0.73	**0.007%**	**0.04**	35.93%	85.91	**0.04%**	**0.16**
LGP	4%	9.57	0.013%	0.06	42.12%	100.71	0.09%	0.34

TABLE V

PERCENTAGE OF REMAINING WEIGHTS AND FLOPS IN THE PRUNED FC LAYERS OF THE ViT-B/16 MODEL (86.0 MILLION INTERCONNECTIONS) AT 6% TOP-1 ACCURACY LOSS (86.57% FOR CIFAR-100 AND 74.06% FOR ImageNet-1K)

	ViT-B/16 + CIFAR-100				**ViT-B/16 + ImageNet-1K**			
	MAC		MAM		MAC		MAM	
	Weights kept	MFLOPs	Weights kept	MFLOPs	Weights kept	MFLOPs	Weights kept	MFLOPs
GMP	29.5%	27.8	**0.001%**	**0.001**	41.1%	38.7	21.0%	29.7
LMP	28.0%	26.4	**0.001%**	**0.001**	40.5%	38.2	10.0%	14.2
GGP	29.0%	27.4	**0.001%**	**0.001**	48.6%	45.9	**0.1%**	**0.14**
LGP	30.0%	28.3	**0.001%**	**0.001**	50.0%	47.2	**0.1%**	0.14

GMP, LMP, GGP, and LGP scoring approaches. We look at the top-1 accuracy of Table II for the non-pruned MAC models and we remove interconnections until we provoke, with respect to best achieved results, a drop in accuracy of 3% for AlexNet and VGG-16, and of 6% for ViT-B/16. Tables III–V show the results for AlexNet, VGG-16, and ViT-B/16, respectively. Additionally, as a relevant example, Fig. 8 shows the top-1 accuracy trend with a decreasing number of kept interconnections for ViT-B/16 + ImageNet-1K. Plots and figures take into consideration only the layers in which we substitute MAM.

Compared to what happens to MAC-based layers, MAM-based ones intrinsically rely on fewer interconnections, in accordance to what is discussed in Section IV-B. In fact, when the DNN is pruned, the interconnections in MAM layers can be removed at a much larger rate compared to standard MAC layers.

A notable case is ViT-B/16 + ImageNet-1K. Here, by using GMP and LMP, MAM still introduces a good prunability to the FC layers, but the results are not as good as compared to the other models. This could be mainly attributed to the combined complexity of the model and the task. Nonetheless, by using GGP and LGP, it is possible to prune MAM to a much larger extent. This is because gradient-based scoring approaches implicitly rely on the concept of probability of selection defined for MAM. In fact, when an interconnection is not selected during inference, the variation it introduces on the cost function is null (and so is the score), automatically excluding it from the set of kept interconnections.

Tables III–V also show the computational complexity in terms of FLOPs, where we count two FLOPs per weight for MAC and three FLOPs per weight for MAM. Even though MAM introduces a computational overhead compared to MAC, the computational complexity of the whole layers is much lower due to the large number of interconnections removed. Further insights on the computational complexity of MAM can be found in Appendix A.

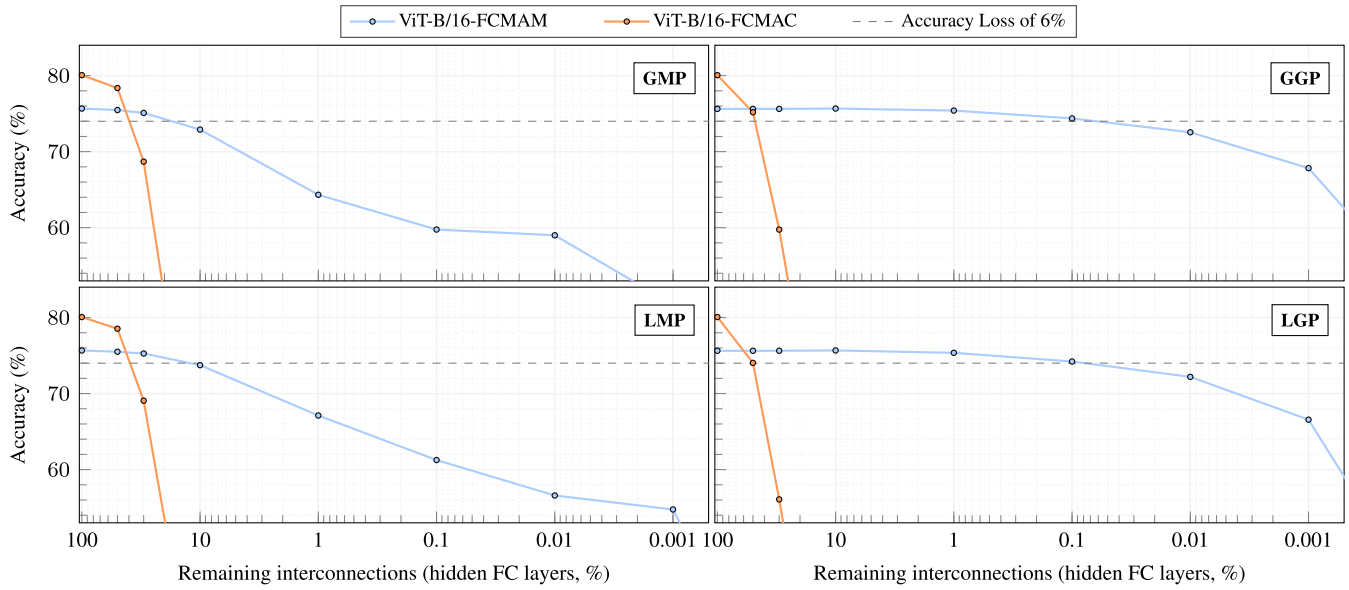

Fig. 8. ViT-B/16 + ImageNet-1K + one-shot pruning: Top-1 accuracy versus percentage of kept interconnections in the hidden FC layers (of 10 out of 12 transformer blocks) when pruned with GMP, LMP, GGP, and LGP. The dashed line indicates when the accuracy is 78.15% (i.e., 3% accuracy loss). Interconnections are increasingly removed going from left to right.

Of course, we need to consider that we are pruning *only* the FC layers. Because of this, in the case of AlexNet and VGG-16, the global reduction in terms of FLOPs is indeed negligible, since most of the computational complexity is due to the convolutional layers. The same is not for ViT-B/16, being all the structure is based on FC layers, so in this case, we globally manage to reduce the computational complexity by more than two times.

VI. MAM on Training-Dependent Pruning Methods

We showed the performance of MAM layers when pruning is applied post-training, i.e., the training phase of the DNN is independent of the pruning performance we want to achieve. However, it is possible to apply more complex, training-dependent pruning approaches to MAM neurons. Here, we present two examples: we test MAM in conjunction with the AC/DC method [10] and the *lottery ticket* iterative pruning approach [11]. We test them on AlexNet + CIFAR-100 and avoid the pre-trained models, namely VGG-16 and ViT-B/16. In fact, training anew these large DNNs would require large computational resources making the use of these pruning methods not practical.

On this matter, we report some examples from [42]. The authors state that, using a standard cloud TPUv3 with eight cores, it takes approximately 30 days to train a ViT-L/16 model on the public ImageNet-21K dataset while training a ViT-H/14 model on the JFT-300M dataset [45] takes around 312 days. Nonetheless, training these models on such large datasets is fundamental to obtain high performance when fine-tuning them on smaller datasets—the same applies also for the ViT-B/16 model used in this work. A similar problem holds for the VGG-16 model trained on ImageNet-1K [41]. Despite notable hardware enhancements since its introduction, a full retraining of this model on ImageNet-1K on a modern GPU still requires several days. For structures that need high computational resources for training, we consider more appropriate the use of post-training, one-shot pruning techniques.

A. Performance With AC/DC Pruning

The alternating compressed/decompressed training (AC/DC) [10] is a technique that aims to reduce the computational training cost through a specific training procedure. This technique produces a sparse version of the model. First, the DNN is trained as-it-is for a few epochs (warm-up phase). Then, the percentage of interconnections to be removed is selected, and the DNN is trained by alternating repeatedly between two different phases, namely compression and decompression. During compression, the selected percentage of interconnections is temporarily removed (with GMP) and the pruned DNN is trained. During decompression, the removed interconnections are restored and training continues with all the interconnections updated. Training ends with a compression phase that is used to fine-tune the model and that produces the final sparse model. At this point, the compressed (sparse) model can be further pruned by using one-shot post-training GMP. To produce MAM-based models with this approach, we morph the MAC neurons into MAM neurons during the last compression phase (with the vanishing contributions technique).

We test AC/DC with different target sparsities (i.e., of 70%, 80%, 85%, 90%, and 95%) on AlexNet + CIFAR-100. Higher percentages of target sparsity result in poor top-1 accuracy results. We train the model for 70 epochs starting with 8 epochs of warm-up followed by alternating compressed and decompressed training phases (11 phases in total, which last 5 epochs each) and by 7 final epochs of fine-tuning. As reported in Table VI, even with this approach, the MAM-based model can be compressed more than the MAC-based one.

TABLE VI

AC/DC: PERCENTAGE OF REMAINING WEIGHTS AND FLOPS IN THE PRUNED FC LAYERS OF THE AlexNet MODEL (47.2 MILLION INTERCONNECTIONS) AT 3% TOP-1 ACCURACY LOSS (62.52%)

	AlexNet + CIFAR-100			
	MAC		MAM	
	Weights kept	MFLOPs	Weights kept	MFLOPs
GMP	25.01%	27.28	0.26%	0.44
ACDC@70	12.86%	14.03	0.26%	0.44
ACDC@80	8.73%	9.53	0.24%	0.41
ACDC@85	7.21%	7.87	0.21%	0.36
ACDC@90	3.25%	3.55	0.18%	0.31
ACDC@95	2.17%	2.37	**0.13%**	**0.23**

B. Performance With Lottery Ticket Iterative Pruning

As a second training-dependent pruning approach, we consider the well-known Lottery Ticket hypothesis in [11] tailored on the non-structured pruning of FC layers. This hypothesis suggests that a randomly initialized DNN contains a subnetwork (i.e., a pruned network) that, if trained again with the same starting conditions (i.e., training parameters initialization), can achieve the same accuracy of the original network after being trained for, at most, the same number of iterations.

In order to select the right sub-network, the authors propose the following approach. Given a DNN with a total number of interconnections P_0, training is performed for a certain number of epochs starting from a given set of randomly initialized parameters. Then, ΔP_0 interconnections are removed from the current number of interconnections P_0 using GMP, and the weights associated with the remaining interconnections are reset to the same initialized values as before. Again, the DNN is trained and ΔP_1 interconnections are removed from the remaining $P_1 = P_0 - \Delta P_0$ interconnections. This is performed iteratively, removing at the generic kth iteration ΔP_k from the remaining P_k interconnections, until the desired size of the network is reached. Typically, ΔP_k is proportional to the remaining number of interconnections P_k, i.e., $\Delta P_k = \alpha P_k$, with $\alpha \in (0, 1)$ defining the fraction of interconnections removed for each iteration.

We apply this iterative pruning approach on AlexNet + CIFAR-100 (both with MAM and MAC FC hidden layers) removing interconnections until only 0.02% of the original number remains. The larger α, the lower the number of repeated training processes (iterations) necessary to reach that number of interconnections. Conversely, using a lower number of iterations results in a poorer performance.

Fig. 9 shows the final accuracy we obtain for the pruned DNN given a certain number of iterations. While MAC requires at least 10 iterations (i.e., ten full training of the DNN) to reach a top-1 accuracy over 60%, the use of MAM results in a much lower number of iterations for the same result (six full training of the DNN). This is crucial for enabling the use of iterative pruning methods with large DNNs that need long training times and expensive hardware for training.

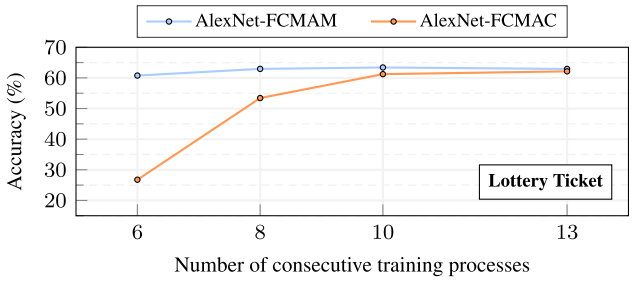

Fig. 9. AlexNet + CIFAR-100 + Lottery Ticket pruning: Top-1 accuracy with 0.02% remaining weights in the hidden FC layers versus the number of consecutive training processes. AlexNet-FCMAM can reach good accuracy with less iterations compared to AlexNet-FCMAC.

VII. CONCLUSION

We have presented and elaborated on a novel MAM map-reduce paradigm, which allows for building neurons that are naturally prone to aggressive pruning and can be deployed at the edge. The resulting FC layers are trained through the vanishing contributions technique, which seamlessly transforms a classic MAC-based structure into a MAM-based one, resulting in a structure with a comparable performance in terms of accuracy. MAM-based layers can be pruned with multiple state-of-the-art non-structured techniques, ranging from one-shot methods to iterative ones. When employed with classical computer vision DNN models, such as AlexNet, VGG-16, and ViT-B/16 on multiple classification tasks, such as MNIST, CIFAR-10, CIFAR-100, and ImageNet-1K, MAM-based structures allow a further reduction in the total number of interconnections compared to a classic MAC-based DNN, together with a reduction of the training cost necessary to obtain such pruned networks in the case of iterative methods. Moreover, despite the reduction in terms of computational cost (FLOPs) being negligible in CNN-based structures (as the largest contribution is given by convolutional layers), it is not the same for transformer-based structures such as ViT-B/16, for which the global reduction of computational cost is consistent.

APPENDIX A
ON THE COMPUTATIONAL COMPLEXITY OF MAM

While the pruning capabilities of MAM neurons constitute the main focus of this work, it naturally prompts consideration of their computational performance when compared with standard MAC structures.

On a high-level view of the matter, it is possible to consider the computational complexity of MAC as two FLOPs per weight (one multiplication and one addition). Regarding MAM operation, we count one multiplication and two additions (one comparison for the maximum and one for the minimum) per weight, with a total of three FLOPs per weight.

Additionally, since MAC operation is very common—it is employed in any application that requires the use of linear algebra—it is highly optimized on general-purpose hardware (including CPUs, GPUs, and even microcontroller units) both from the point of view of computation and memory access. As an example, MAC operations often benefit from

the fused multiply-add (FMA) operation [46] that allows to perform multiplication and addition together within one round. Of course, it is not the same for the MAM operation, which needs a multiplication and two comparison operations associated with two branching instructions (they select whether to update the maximum/minimum value in the output register or not). In particular, branching tends to disrupt the pipeline structure present in most modern processors, slowing down the processing with MAM operations. Nevertheless, the branch predictor mechanism—which is present on most processors—along with the fact that the output register needs to be updated only a few times on average, greatly alleviates this problem.

Nonetheless, if we allow ourselves the use of ad hoc architectures (e.g., we design custom hardware accelerators for deploying MAM), MAM operation can easily match MAC in terms of computational complexity. This is because: 1) the flow-disrupting branch operation could be implemented with multiplexer structures, with a negligible computational overhead and 2) since *typically* the inputs in the layer are positive (being image data, or the output of a ReLU activation), we know beforehand which inputs are associated with the maximum and the minimum, reducing the FLOPs from 3 to 2.

Moreover, MAM neurons have been developed to work with pruned layers. Because of this, their computational performance should be evaluated in a context in which data is sparse, which calls again for the use of specialized hardware to get the best from this approach.

Finally, MAM could unlock some computational optimization during the training phase of the network. First, we hypothesize that looking at the probability of selection of the interconnections at an early training stage could unlock sparse training. This is because interconnections that are not selected in an early stage of training are typically the same that are not selected when training is completed. Second, during backpropagation, the computation of the gradient travels only through the interconnections that are selected by MAM (only 2 per neuron), strongly reducing the number of weight updates. This could introduce large energy savings when performing fine-tuning on edge devices.

ACKNOWLEDGMENT

This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

REFERENCES

[1] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. V. Guttag, "What is the state of neural network pruning," in *Proc. Mach. Learn. Syst.*, vol. 2, Mar. 2020, pp. 129–146.

[2] Y. Jiang et al., "Model pruning enables efficient federated learning on edge devices," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10374–10386, Jul. 2023, doi: 10.1109/TNNLS.2022.3166101.

[3] M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for AI-enabled IoT devices: A review," *Sensors*, vol. 20, no. 9, p. 2533, 2020, doi: 10.3390/s20092533.

[4] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019, doi: 10.1109/JPROC.2019.2921977.

[5] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020, doi: 10.1109/TWC.2019.2946140.

[6] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. 12th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2014, pp. 68–81, doi: 10.1145/2594368.2594383.

[7] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "FastGRNN: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2018, pp. 9031–9042.

[8] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 426–438, doi: 10.1145/2789168.2790123.

[9] L. Prono, M. Mangia, A. Marchioni, F. Pareschi, R. Rovatti, and G. Setti, "Deep neural Oracle with support identification in the compressed domain," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no. 4, pp. 458–468, Dec. 2020, doi: 10.1109/JETCAS.2020.3039731.

[10] A. Pešte, E. Iofinova, A. Vladu, and D. Alistarh, "AC/DC: Alternating compressed/decompressed training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, Jan. 2021, pp. 8557–8570.

[11] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, 2018.

[12] 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 *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2022, pp. 712–716, doi: 10.1109/ISCAS48785.2022.9937576.

[13] P. Bich, L. Prono, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "Aggressively prunable MAM^2-based deep neural Oracle for ECG acquisition by compressed sensing," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2022, pp. 163–167, doi: 10.1109/BioCAS54905.2022.9948676.

[14] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1398–1406, doi: 10.1109/ICCV.2017.155.

[15] Z. Chen, T.-B. Xu, C. Du, C.-L. Liu, and H. He, "Dynamical channel pruning by conditional accuracy change for deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 799–813, Feb. 2021, doi: 10.1109/TNNLS.2020.2979517.

[16] Y. He, P. Liu, L. Zhu, and Y. Yang, "Filter pruning by switching to neighboring CNNs with good attributes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 8044–8056, Oct. 2023, doi: 10.1109/TNNLS.2022.3149332.

[17] K. Persand, A. Anderson, and D. Gregg, "Taxonomy of saliency metrics for channel pruning," *IEEE Access*, vol. 9, pp. 120110–120126, 2021, doi: 10.1109/ACCESS.2021.3108545.

[18] W. Niu et al., "PatDNN: Achieving real-time DNN execution on mobile devices with pattern-based weight pruning," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.* New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 907–922, doi: 10.1145/3373376.3378534.

[19] X. Chen, J. Zhu, J. Jiang, and C.-Y. Tsui, "Tight compression: Compressing CNN model tightly through unstructured pruning and simulated annealing based permutation," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6, doi: 10.1109/DAC18072.2020.9218701.

[20] A. R. Aswani, R. Chithra, and A. P. James, "Unstructured weight pruning in variability-aware memristive crossbar neural networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2022, pp. 3458–3462, doi: 10.1109/ISCAS48785.2022.9937284.

[21] C. Yang, Y. Wang, and J. D. Owens, "Fast sparse matrix and sparse vector multiplication algorithm on the GPU," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshop*, May 2015, pp. 841–847, doi: 10.1109/IPDPSW.2015.77.

[22] S.-F. Hsiao and H.-J. Chang, "Sparsity-aware deep learning accelerator design supporting CNN and LSTM operations," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–4, doi: 10.1109/ISCAS45731.2020.9180404.

[23] J.-F. Zhang, C.-E. Lee, C. Liu, Y. S. Shao, S. W. Keckler, and Z. Zhang, "SNAP: An efficient sparse neural acceleration processor for unstructured sparse deep neural network inference," *IEEE J. Solid-State Circuits*, vol. 56, no. 2, pp. 636–647, Feb. 2021, doi: 10.1109/JSSC.2020.3043870.

[24] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 1. Cambridge, MA, USA: MIT Press, Dec. 2015, pp. 1135–1143.

[25] A. Formanek and D. Hadházi, "Compressing convolutional neural networks by L0 regularization," in *Proc. Int. Conf. Control, Artif. Intell., Robot. Optim. (ICCAIRO)*, Dec. 2019, pp. 155–162, doi: 10.1109/ICCAIRO47923.2019.00032.
[26] A. Ren et al., "ADMM-NN: An algorithm-hardware co-design framework of DNNs using alternating direction methods of multipliers," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2019, pp. 925–938, doi: 10.1145/3297858.3304076.
[27] F. Martinini, A. Enttsel, A. Marchioni, M. Mangia, R. Rovatti, and G. Setti, "Structured pruning in deep neural networks with trainable probability masks," in *Proc. IEEE 66th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2023, pp. 1020–1024, doi: 10.1109/MWSCAS57524.2023.10405945.
[28] M. S. Zhang and B. C. Stadie, "One-shot pruning of recurrent neural networks by Jacobian spectrum evaluation," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.
[29] T. Chen et al., "Only train once: A one-shot neural network training and pruning framework," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34. Red Hook, NY, USA: Curran Associates, Jan. 2021, pp. 19637–19651.
[30] J. T. C. Min and M. Motani, "DropNet: Reducing neural network complexity via iterative pruning," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, vol. 119, Jul. 2020, pp. 9356–9366.
[31] M. Zullich, E. Medvet, F. A. Pellegrino, and A. Ansuini, "Speeding-up pruning for artificial neural networks: Introducing accelerated iterative magnitude pruning," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 3868–3875, doi: 10.1109/ICPR48806.2021.9412705.
[32] H. Zhang, L. Liu, H. Zhou, W. Hou, H. Sun, and N. Zheng, "AKECP: Adaptive knowledge extraction from feature maps for fast and efficient channel pruning," in *Proc. 29th ACM Int. Conf. Multimedia*. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 648–657, doi: 10.1145/3474085.3475228.
[33] H. Zhang, K. Hao, L. Gao, B. Wei, and X. Tang, "Optimizing deep neural networks through neuroevolution with stochastic gradient descent," *IEEE Trans. Cognit. Develop. Syst.*, vol. 15, no. 1, pp. 111–121, Mar. 2023, doi: 10.1109/TCDS.2022.3146327.
[34] M. Nonnenmacher, T. Pfeil, I. Steinwart, and D. Reeb, "SOSP: Efficiently capturing global correlations by second-order structured pruning," in *Proc. Int. Conf. Learn. Represent.*, Oct. 2021.
[35] C. Tan and J. Liu, "Improving knowledge distillation with a customized teacher," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 2, pp. 2290–2299, Feb. 2024, doi: 10.1109/TNNLS.2022.3189680.
[36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE*, Jan. 1998, vol. 86, no. 11, pp. 2278–2324, doi: 10.1109/5.726791.
[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[38] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.
[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.
[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25. Red Hook, NY, USA: Curran Associates, May 2012, pp. 1–9.
[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent.*, May 2015.
[42] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Oct. 2020.
[43] Y. Liu et al., "A survey of visual transformers," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 6, pp. 7478–7498, Jun. 2024, doi: 10.1109/TNNLS.2022.3227717.
[44] H. Zhang, M. Cissé, Y. Dauphin, and D. López-Paz, "Mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Represent.*, Apr. 2017.
[45] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 843–852, doi: 10.1109/ICCV.2017.97.
[46] R. K. Montoye, E. Hokenek, and S. L. Runyon, "Design of the IBM RISC system/6000 floating-point execution unit," *IBM J. Res. Develop.*, vol. 34, no. 1, pp. 59–70, Jan. 1990, doi: 10.1147/rd.341.0059.

Luciano Prono (Member, IEEE) received the M.Sc. degree in electronics engineering and the Ph.D. degree "cum laude" in electrical, electronics, and communication engineering from Politecnico di Turin, Turin, Italy, in 2019 and 2023, respectively.

He was a Visiting Ph.D. Student with the Institute of Neuroinformatics (University of Zurich and ETH), Zürich, Switzerland, in 2022. He is currently working as an Assistant Professor with the Department of Electronics and Telecommunications, Politecnico di Turin. His research interests include low-power systems and applications of artificial intelligence, mainly in the fields of the IoT, tinyML, edge computing, compressed sensing, and neuromorphic computing.

Philippe Bich (Graduate Student Member, IEEE) received the B.Sc. degree in computer engineering and the M.Sc. degree (Hons.) in mechatronic engineering from Politecnico di Turin, Turin, Italy, in 2018 and 2021, respectively, where he is currently pursuing the Ph.D. degree with the Department of Electronics and Telecommunications (DET).

In 2021, he was a Visiting Student at the Boston University Robotics Laboratory, Brookline, MA, USA, under the supervision of Prof. John Baillieul. His research interests include robotics and artificial intelligence, specifically in the areas of quantization and pruning of large deep neural networks to optimize both their training and inference processes.

Chiara Boretti (Graduate Student Member, IEEE) received the M.Sc. degree (Hons.) in mechatronic engineering from Politecnico di Turin, Turin, Italy, in 2021, where she is currently pursuing the Ph.D. degree in electrical, electronics, and communications engineering.

She is also affiliated with the Interdepartmental Center for Service Robotics (PIC4SeR), Politecnico di Turin. Her research interests include robotic visual perception, utilizing both conventional frame-based cameras and cutting-edge neuromorphic sensors, such as event-based cameras. Her interests extend to neuromorphic computing and to the development of algorithms for resource-constrained systems.

Mauro Mangia (Member, IEEE) received the B.Sc. and M.Sc. degrees in electronic engineering and the Ph.D. degree in information technology from the University of Bologna, Bologna, Italy, in 2005, 2009, and 2013, respectively.

He was a Visiting Ph.D. Student with the Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, from 2009 to 2012. He is currently an Associate Professor with the Statistical Signal Processing Group, Department of Electrical, Electronic and Information Engineering, University of Bologna. He is also a Member of both the Advance Research Center for Electronic Systems (ARCES), University of Bologna, and Alma Mater Research Institute for Human-Centered Artificial Intelligence, University of Bologna. His research interests include nonlinear systems, explainable artificial intelligence, machine learning and artificial intelligence, anomaly detection, the Internet of Things, big data analytics, and optimization.

Dr. Mangia was a recipient of the 2013 IEEE CAS Society Guillemin-Cauer Award and the 2019 IEEE BioCAS Transactions Best Paper Award. He was also a recipient of the Best Student Paper Award at the International Symposium on Circuits and Systems (ISCAS) 2011. He was the Web and Social Media Chair of ISCAS2018.

Fabio Pareschi (Senior Member, IEEE) received the Dr. Eng. degree (Hons.) in electronic engineering from the University of Ferrara, Ferrara, Italy, in 2001, and the Ph.D. degree in information technology from the University of Bologna, Bologna, Italy, in 2007, under the European Doctorate Project (EDITH).

He is currently an Associate Professor with the Department of Electronic and Telecommunication, Politecnico di Turin, Turin, Italy. He is also a Faculty Member with ARCES, University of Bologna. His research interests include analog and mixed-mode electronic circuit design, statistical signal processing, compressed sensing, dc–dc converters, random number generation and testing, and electromagnetic compatibility.

Dr. Pareschi was a recipient of the 2019 IEEE BioCAS Transactions Best Paper Award, the Best Paper Award at ECCTD 2005, and the Best Student Paper Award at EMC Zurich 2005 and IEEE EMCCompo 2019. From 2010 to 2013, he was an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS and IEEE OPEN JOURNAL OF CIRCUITS AND SYSTEMS from 2020 to 2022.

Riccardo Rovatti (Fellow, IEEE) received the M.S. degree in electronic engineering and the Ph.D. degree in electronics, computer science, and telecommunications from the University of Bologna, Bologna, Italy, in 1992, and 1996, respectively.

He is currently a Full Professor of electronics with the University of Bologna. He has authored more than 300 technical contributions to international conferences and journals and two volumes. His research interests include mathematical and applicative aspects of statistical signal processing, on machine learning for signal processing, and on the application of statistics to nonlinear dynamical systems.

Dr. Rovatti was a Distinguished Lecturer of the IEEE CAS Society from 2017 to 2018. He was a recipient of the 2004 IEEE CAS Society Darlington Award, the 2013 IEEE CAS Society Guillemin-Cauer Award, and the 2019 IEEE BioCAS Transactions Best Paper Award. He received the Best Paper Award at ECCTD 2005 and the Best Student Paper Award at the EMC Zurich 2005 and ISCAS 2011. He is an IEEE Fellow for his contribution to nonlinear and statistical signal processing applied to electronic systems.

Gianluca Setti (Fellow, IEEE) received the Dr.Eng. degree (Hons.) and the Ph.D. degree in electronic engineering from the University of Bologna, Bologna, Italy, in 1992 and 1997, respectively.

From 1997 to 2017, he was with the Department of Engineering, University of Ferrara, Ferrara, Italy, as an Assistant Professor from 1998 to 2000, an Associate Professor from 2001 to 2008, and a Professor from 2009 to 2017 of circuit theory and analog electronics. From 2017 to 2022, he was a Professor of electronics, signal, and data processing at the Department of Electronics and Telecommunications (DET), Politecnico di Torino, Turin, Italy. Since November 2022, he has been the Dean of the Computer, Electrical, Mathematical Sciences and Engineering (CEMSE) Division and a Professor of Electrical and Computer Engineering (ECE) Program at the King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia. He held several positions, such as Visiting Professor/Scientist at EPFL (2002 and 2005), UCSD (2004), IBM (2004 and 2007) and at the University of Washington, Washington, DC, USA (2008 and 2010). He is the coeditor of the book *Chaotic Electronics in Telecommunications* (CRC Press, Boca Raton, 2000), *Circuits and Systems for Future Generation of Wireless Communications* (Springer, 2009), and *Design and Analysis of Biomolecular Circuits* (Springer, 2011); the co-author of the book *Adapted Compressed Sensing for Effective Hardware Implementations* (2018); and one of the guest editors of the May 2002 Special Issue of the IEEE Proceedings on "Applications of Nonlinear Dynamics to Electronic and Information Engineering." His research interests include nonlinear circuits, recurrent neural networks, electromagnetic compatibility, compressive sensing and statistical signal processing, biomedical circuits and systems, power electronics, design and implementation of the IoT nodes, circuits and systems for machine learning, and applications of AI techniques for anomaly detection and predictive maintenance.

Dr. Setti was a Distinguished Lecturer of the IEEE CAS Society (2004–2005 and 2013–2014), a member of the CASS Board of Governors (2005–2008) of the IEEE CAS Society, and served as the 2010 CAS Society President. He received the 2013 IEEE CAS Society Meritorious Service Award and the co-recipient of the 2004 IEEE CAS Society Darlington Award, the 2013 IEEE CAS Society Guillemin-Cauer Award, the 2019 IEEE Transactions on Circuits and Systems Best Paper Award, the Best Paper Award at ECCTD2005, and the Best Student Paper Awards at EMCZurich2005 and ISCAS2011. He also received the 1998 Caianiello Prize for the Best Italian Ph.D. Thesis on neural networks. In 2012, he was the Chair of the IEEE Strategic Planning Committee of the Publication Services and Products Board (PSPB-SPC) and in 2013–2014, he was the first non-North-American Vice President of the IEEE for Publication Services and Products. He served in the program committee of many conferences and was, in particular, the Special Sessions Co-Chair of ISCAS2005 (Kobe) and ISCAS2006 (Kos), the Technical Program Co-Chair of NDES2000 (Catania), ISCAS2007 (New Orleans), ISCAS2008 (Seattle), ICECS2012 (Seville), BioCAS2013 (Rotterdam), MWSCAS2023 (Phoenix), and the General Co-Chair of NOLTA2006 (Bologna) and ISCAS2018 (Florence). He served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS (1999–2002 and 2002–2004) and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS (2004–2007), the Deputy-Editor-in-Chief for *IEEE Circuits and Systems Magazine* (2004–2007), and the Editor-in-Chief for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS (2006–2007) and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS (2008–2009). He served in the Editorial Board for IEEE ACCESS (2013–2015) and *Proceedings of the IEEE* (2015–2018). From 2019 to 2024, he served for two terms as the first non-U.S. Editor-in-Chief of *Proceedings of the IEEE*, the flagship journal of the institute.