Short Papers

On the Universal Approximation Properties of Deep Neural Networks Using MAM Neurons

Philippe Bich, *Student Member, IEEE*, Andriy Enttsel, *Student Member, IEEE*, Luciano Prono, *Member, IEEE*, Alex Marchioni, *Member, IEEE*, Fabio Pareschi, *Senior Member, IEEE*, Mauro Mangia, *Member, IEEE*, Gianluca Setti, *Fellow, IEEE*, and Riccardo Rovatti, *Fellow, IEEE*

***Abstract*—As neural networks are trained to perform tasks of increasing complexity, their size increases, which presents several challenges in their deployment on devices with limited resources. To cope with this, a recently proposed approach hinges on substituting the classical Multiply-and-ACcumulate (MAC) neurons in the hidden layers with other neurons called Multiply-And-Max/min (MAM) whose selective behavior helps identify important interconnections, thus allowing aggressive pruning of the others. Hybrid MAM&MAC structures promise a 10x or even 100x reduction in their memory footprint compared to what can be obtained by pruning MAC-only structures. However, a cornerstone of maintaining this promise is the assumption that MAC&MAM architectures have the same expressive power as MAC-only ones. To concretize such a cornerstone, we take here a step in the theoretical characterization of the capabilities of mixed MAM&MAC networks. We prove, with two theorems, that two hidden MAM layers followed by a MAC neuron with possibly a normalization stage is a universal approximator.**

***Index Terms*—Deep neural network, multiply-and-max/min, universal approximation theorem.**

I. Introduction

Deep Neural Networks (DNNs) have achieved widespread adoption due to their remarkable success in diverse domains [1], [2]. The evolution of these networks from the simple Multi Layer Perceptron (MLP), the subsequent introduction of Convolutional Neural Networks (CNNs) [3] and the recent development of Transformers [4] that are now exploited for solving almost any task, showcase a progressive shift towards architectures of higher complexity and larger scale. What all these systems have in common is the structure of the neuron, which is based on the Multiply-and-ACcumulate (MAC) preactivation stage, which allows DNNs to work as universal approximations [5], [6], [7].

Due to the increasing interest in mobile artificial intelligence [8] but also in light of the concerns surrounding the resource consumption of large models [9], there has been a growing emphasis on designing low-resource networks. As an example, a popular technique used to decrease the memory footprint of DNN is represented by pruning [10], which consists in removing weights from the neural model without sacrificing its accuracy. The challenge of reducing network complexity has also prompted researchers to investigate alternative neuron structures capable of delivering performance comparable to networks using standard MAC-based neurons, while consuming less memory and power. In [11], we proposed and studied a novel neuron model that is designed to structurally facilitate aggressive pruning. These neurons are based on the Multiply-And-Max/min (MAM) paradigm, and they can be substituted to classical MAC neurons in the hidden layers of a DNN while substantially preserving the network performance. As in a standard MAC-based neuron, in a MAM neuron the input elements are modulated independently by multiplying them with their respective weights. However, in a MAM neuron, instead of summing all the resulting products into a single quantity, only the maximum and the minimum products are considered.

It is shown empirically in [11] that starting from an architecture originally designed using MAC neurons, one may substitute them with MAM neurons in several hidden layers and use a proper training strategy to recover a performance close to the original MAC-only network. Then, existing pruning strategies may be applied to the new network to reduce the number of weights. As an example, Table I reports some results from [11]. It shows that when Global/Layer-wise Magnitude, or Global/Layer-wise Gradient pruning techniques are applied to the MAM version of three well-known classification architectures, the number of surviving weights is reduced up to 2 orders of magnitude more than what can be obtained for MAC versions. The increased sparsification observed in MAM-based networks not only reduces memory requirements but can also lead to faster inference, as demonstrated in [12]. Even in the absence of specialized hardware support for MAM operations, these effects have been empirically measured, confirming that higher sparsity can translate into efficiency gains at the full network level.

Beyond such empirical evidence, notice how the sensibility of the MAM approach is based on two key assumptions. The first is that hybrid MAM&MAC networks have the same expressive power of MAC-only networks. The second is that MAM&MAC networks allow an extremely aggressive pruning, but this can be intuitively attributed to the $\min$ and $\max$ operators within each MAM neuron, which highlight the only two important contributions in each inference of every single neuron.

Received 19 July 2024; revised 20 March 2025; accepted 6 May 2025. Date of publication 15 May 2025; date of current version 6 August 2025. Recommended for acceptance by Z. Wei. *(Corresponding author: Luciano Prono.)*

Philippe Bich and Luciano Prono are with the Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Torino, Italy (e-mail: philippe.bich@polito.it; luciano.prono@polito.it).

Andriy Enttsel, Alex Marchioni, 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: andriy.enttsel@unibo.it; alex.marchioni@unibo.it; mauro.mangia@unibo.it; riccardo.rovatti@unibo.it).

Fabio Pareschi is with the Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Torino, 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/TPAMI.2025.3570545

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

TABLE I
PERCENTAGE OF REMAINING WEIGHTS IN THE PRUNED FULLY CONNECTED LAYERS OF ALEXNET, VGG-16 AT 3% TOP-1 ACCURACY LOSS AND VIT-B/16 AT 6% TOP-1 ACCURACY LOSS USING DIFFERENT PRUNING SCORES

		AlexNet + CIFAR-100		VGG-16 + ImageNet-1K		ViT-B/16 + ImageNet-1K	
		MAC	MAM	MAC	MAM	MAC	MAM
Top-1 Accuracy		65.52%	64.63%	63.36%	63.04%	80.06%	75.62%
% NON-PRUNED WEIGHTS	Global Magnitude	25.01%	0.26%	10.82%	**0.04%**	41.1%	21.0%
	Layer-wise Magnitude	26.05%	0.30%	10.36%	0.07%	40.5%	10.0%
	Global Gradient	17.95%	**0.21%**	35.93%	**0.04%**	58.6%	**0.1%**
	Layer-wise Gradient	12.68%	0.24%	42.12%	0.09%	50.0%	0.1%

Yet, the first assumption is far from being trivial, as most of the classical theorems on the expressive power of neural networks deeply exploit the MAC operations of the neurons and the fact that nonlinearity is concentrated in the activation functions and not in the previous linear combination of inputs. Here, we provide theoretical ground to this first assumption by proving two universal approximation theorems for hybrid MAM&MAC architectures as well as asymptotic trends for the complexity of the corresponding networks.

The remainder of this work is structured as follows. In Section II, we list related works that demonstrate the approximation capabilities of several neural network models. In Section III, we present a high-level description of the mathematical model that we use to derive the two theorems on the approximation capabilities of DNNs using MAM neurons. The two theorems are formally stated in Section IV. An example of a function approximation and quantitative results regarding the convergence of the approximation error are reported in Section V. The proof of these theorems can be found in Section VI. Finally, conclusion is drawn.

II. RELATED WORKS

The development of models with universal approximation capabilities has been a significant breakthrough in many fields of science and engineering. In 1989, [5] proved that a network with a single hidden layer could approximate any continuous function, given enough hidden neurons. Some years later, [13] and [14] showed that also fuzzy systems could approximate any continuous function to arbitrary accuracy. These works were later extended to multiple inputs and outputs, demonstrating the universal approximation properties of fuzzy systems more broadly [15]. In the following years, a large number of researchers have studied the universal approximation properties of neural networks with MAC neurons in the case of bounded depth and arbitrary width [6], [7], bounded width and arbitrary depth [16], [17] and bounded width and depth [18]. In the recent work [19], authors obtained the optimal minimum width bound of a neural network with arbitrary depth to retain universal approximation capabilities.

Research in this field is still very active and aims at proving the universal approximation capabilities of networks with different architectural or computational paradigm choices, such as deep convolutional neural networks [20], dropout neural networks [21], networks representing probability distributions [22], graph neural networks with random weights [23] and spiking neural networks [24]. In addition, in [25] the authors exploit a neural network's universal approximation property requirement to design novel architectures competitive with the state-of-the-art.

Some works also propose the usage of sorting activation functions, i.e., GroupSort [26], [27], [28]. Here, GroupSort is demonstrated to improve the properties of Lipschitz neural networks, improving their robustness against adversarial examples. At first glance, the sorting functions can be associated with the use of maximum and minimum functions. However, the schemes presented in these works are structurally different from what is presented in this manuscript. Multiply-And-Max/min neurons are not merely another form of a non-polynomial activation; they define an entirely different type of network where the scalar product is not used in intermediate layers. This is an extremely important difference since, to the best of our knowledge, all the universal approximation works – including both the classical results for ReLU networks [6] and the latest works [17], [19], [29], [30] – crucially rely on representations of the kind

$$u = \Psi\left(\langle \boldsymbol{w}, \boldsymbol{v} \rangle + b\right) \tag{1}$$

where $\boldsymbol{v}$ and $\boldsymbol{w}$ are the equally-dimensioned inputs vector and weights vector, b is a bias value, $\Psi(\cdot)$ is a nonlinearity function (e.g., the commonly used ReLU function) and u is the output of the neuron. In particular, this representation is fundamentally defined by the scalar product $\langle \cdot, \cdot \rangle$. It is clear that existing results on the universal approximation properties of neural networks heavily depend on the scalar product in the intermediate layers, a property that MAM networks do not possess.

III. GENERAL MODEL AND PROOF STRATEGY

To demonstrate the approximation capabilities of a hybrid MAM&MAC structure, we build and program two network architectures and prove that their approximation error can approach zero by increasing their complexity. In this section, we present the general idea behind this construction, while the detailed description and the proofs of the theorems can be found in Section VI. In our design, we use two MAM layers followed by a MAC layer, as hidden layers typically contain more weights and thus benefit the most from MAM. Moreover, this architecture aligns with the empirical setup used in [11].

In each MAM neuron, all inputs are multiplied by a corresponding weight, resulting in a series of *weighted inputs*. Then, only the maximum and minimum of the weighted inputs are taken and summed together (as opposed to standard MAC neurons, which sum *all* the weighted inputs). A bias value is then added, and an activation function is applied. In this work, we use the well-known ReLU activation. If $v_1, v_2, \dots$ are the inputs and $w_1, w_2, \dots$ are the weights, the MAM neuron can be described as follows

$$u = \left[\max_j w_j v_j + \min_j w_j v_j + b\right]^+ \tag{2}$$

where b is the bias and $[\cdot]^+ = \max\{0, \cdot\}$ represents the commonly used ReLU nonlinearity. Notably, (2) differs fundamentally from (1), as the aggregation (the reduce operation) is different. In fact, it relies on the maximum and minimum operations rather than the summation within the scalar product. We stress that this key distinction makes the existing universal approximation studies not directly applicable to this case.

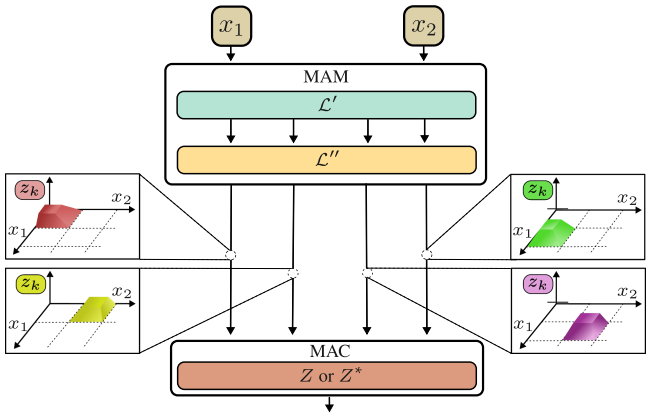

Fig. 1. Structure of the overall neural network when $N = 2$. The composition of the two MAM layers $\mathcal{L}''(\mathcal{L}'(\boldsymbol{x}))$ is used to generate the building blocks z_k that the final MAC layer Z^* or Z properly combines to create the output of the network.

We adopt MAM and MAC neurons to build a two-part DNN structure. In the first part, MAM neurons are used to build specific weakly unimodal piecewise linear functions $z(\boldsymbol{x})$, with $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ the N inputs of the DNN for which we will assume $x_i \in [0, 1]$. In the second part, all the $z(\boldsymbol{x})$ computed by the first part are combined to approximate the target function. This is shown in Fig. 1 for the special case $N = 2$, though the general architecture remains the same for any N. The first layers $\mathcal{L}'$ and $\mathcal{L}''$ produce "truncated pyramids" (which in the case of $N > 2$ become "truncated hyperpyramids") whose positions, truncations, and slopes depend on the parameters of these two layers.

If we index the outputs of the first part of the network as $z_k(\boldsymbol{x})$, the third layer first applies weights c_k and finally combines them in the output of the network. The choice of the final combination, the shapes, and the positions of $z_k(\boldsymbol{x})$ are intertwined. In fact, inputs can affect the output of the network only if $z_k(\boldsymbol{x}) \neq 0$ for some k. If we collect in $\mathbb{Y} \subset \mathbb{X} = [0, 1]^N$ the set of points for which this fails, we require that its Lebesgue measure is null, i.e., $\mu(\mathbb{Y}) = 0$

Clearly, this can be obtained by making the supports of $z_k(\boldsymbol{x})$ overlap and thus $\mathbb{Y} = \emptyset$. In this case, not to complicate the choice of network parameters, it is advisable to avoid the fact that the value of the output depends on how many $z_k(\boldsymbol{x})$ are non-null at each point. This can be obtained by adopting a final normalization stage that is quite common in the literature and yields the network output

$$Z^*(\boldsymbol{x}) = \frac{\sum_k c_k z_k(\boldsymbol{x})}{\sum_k z_k(\boldsymbol{x})}. \tag{3}$$

Although each $z_k(\boldsymbol{x})$ is piecewise linear and could be used, in principle, to reproduce both punctual and differential behaviors, normalization makes the gradient of the input-output relationship of the network difficult to control.

To recover the ability of reproducing both punctual and differential behaviors, we additionally consider a final layer with a simple linear combination

$$Z(\boldsymbol{x}) = \sum_k c_k z_k(\boldsymbol{x}). \tag{4}$$

In this case, the resulting input-output relationship is piecewise linear and the overlap between the $z_k(\boldsymbol{x})$ can be administered to guarantee that the network parameters can easily control both the punctual and differential behaviors. This leaves $\mathbb{Y} \neq \emptyset$ and another subset $\mathbb{Y}'$ with $\mathbb{Y} \subset \mathbb{Y}' \subset \mathbb{X}$ in which universal approximation cannot be guaranteed but whose measure can be made to vanish by increasing network complexity.

IV. Main Results

Within the above framework, we prove two theorems that describe the universal approximation properties of DNNs using MAM neurons in the hidden layers. Theorem 1 guarantees the strongest uniform approximation of targets that are only required to be continuous. However, it requires a last layer of the form of (3) that needs normalization. Theorem 2 guarantees a looser approximation that allows deviation in a vanishing-measure subdomain. However, this weaker approximation, combined with a stronger smoothness assumption on the target function, allows us to leverage the simpler linear last layer in (4) and to show that the first-order differential behavior can also be reproduced. In both cases, we give the asymptotic trend of the number of parameters in a network whose approximation error does not exceed a prescribed threshold.

Let us indicate with $\mathcal{Z}^*$ the family of functions in (3) while with $\mathcal{Z}$ the analogous family of functions in (4). Smoothness conditions on our target functions $f : \mathbb{X} \mapsto \mathbb{R}$ are formalized by assuming that they belong to $\mathcal{C}^d(\mathbb{X})$, i.e., that their dth order derivatives are continuous.

Theorem 1: For any function $f \in \mathcal{C}^0(\mathbb{X})$ and any prescribed level of tolerance $\epsilon > 0$, there is a $Z^* \in \mathcal{Z}^*$ such that

$$\sup_{\boldsymbol{x} \in \mathbb{X}} |f(\boldsymbol{x}) - Z^*(\boldsymbol{x})| \leq \epsilon.$$

If f is also Lipschitz and $\#\mathrm{p}$ is the number of parameters in the network, then the above inequality holds for

$$\#\mathrm{p} \sim \epsilon^{-N}$$

as $\epsilon \to 0$.

Theorem 2: For any function $f \in \mathcal{C}^2(\mathbb{X})$ and any prescribed levels of tolerance $\epsilon, \epsilon', \xi > 0$, there are a domain $\mathbb{D} \subset \mathbb{X}$ and a $Z \in \mathcal{Z}$ such that

$$\sup_{\boldsymbol{x} \in \mathbb{D}} |f(\boldsymbol{x}) - Z(\boldsymbol{x})| \leq \epsilon$$

$$\sup_{\boldsymbol{x} \in \mathbb{D}} \left| \frac{\partial f}{\partial x_j}(\boldsymbol{x}) - \frac{\partial Z}{\partial x_j}(\boldsymbol{x}) \right| \leq \epsilon'$$

$$\mu\left(\mathbb{X} \setminus \mathbb{D}\right) \leq \xi.$$

If $\#\mathrm{p}$ is the number of parameters in the network then the above inequalities are satisfied for

$$\#\mathrm{p} \sim \max\left\{\epsilon^{-\frac{N}{2}}, \epsilon'^{-N}, \xi^{-N}\right\}$$

as $\epsilon, \epsilon', \xi \to 0$.

The proofs of both theorems are reported in Section VI and are constructive. In particular, subnetworks in the cascade $\boldsymbol{z}(\boldsymbol{x}) = \mathcal{L}''(\mathcal{L}'(\boldsymbol{x}))$ are identified and programmed to make each $z_k(\boldsymbol{x})$ a weakly unimodal piecewise linear function of the inputs, whose maximum is 1 and is reached in a hyper-rectangular subset of the domain, while the function vanishes for points far from the center of that hyper-rectangle. The shapes and positions of these functions can then be designed along with the weights c_k so that their combination by means of (3) or (4) is capable of arbitrarily approximating the target function.

A. Limitations of the Current Approach

Theorems 1 and 2 are based on networks in which constraints are placed on neither the width of the layer nor the total number of neurons. Hence, despite proving universal approximation capabilities, they do

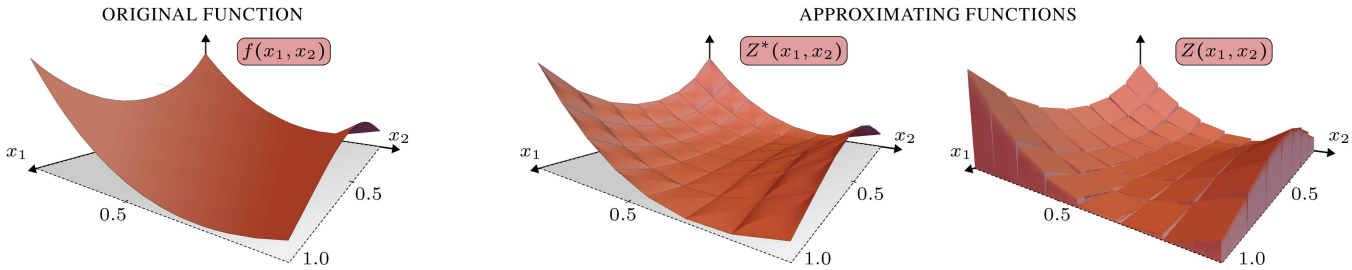

Fig. 2. Three dimensional plot of the target function $f(x_1, x_2)$ and of its two approximations $Z^*(x_1, x_2) \in \mathcal{Z}^*$ implied by Theorem 1 and $Z(x_1, x_2) \in \mathcal{Z}$ by Theorem 2 with $N = 2$ and $n = 7$.

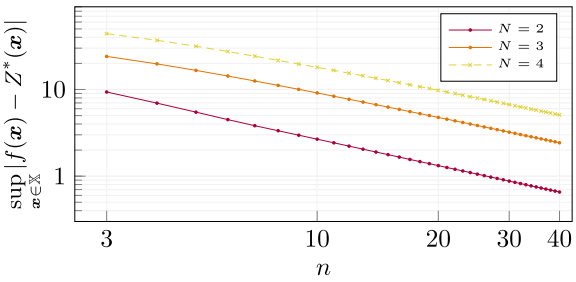

Fig. 3. Convergence of the approximation error for the Rosenbrock function for Theorem 1 with a varying number of input dimensions N and of divisions of the domain n.

not imply *efficient* approximation. However, such a theoretical limitation is never experienced strongly in practice, since MAM networks can guarantee acceptable performance in real use cases.

V. Examples

The purpose of this section is to visually represent the approximation capabilities of a neural network employing MAM neurons, as delineated in Theorems 1 and 2. As a case study, we utilize the $\mathcal{C}^\infty(\mathbb{X})$ Rosenbrock function [31], centered in $(0.5, \ldots, 0.5) \in \mathbb{R}^N$, which is defined as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N-1} \left[100 \left(x_{i+1} - x_i^2 + 2x_i - 0.75\right)^2 + (1.5 - x_i)^2\right] \quad (5)$$

where $f : \mathbb{X} \to \mathbb{R}$ and $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{X}$, i.e., $\boldsymbol{x} \in \mathbb{X}^N$.

The complexity of implied networks will be summarized by an integer parameter n whose precise role in the construction of the network is defined in Section VI. What is important to know here to appreciate these examples is that the number of neurons in a network is approximately proportional to n^N Setting $N = 2$ and $n = 7$, Fig. 2 shows visual examples of the Rosenbrock function and the approximation results for both theorems. Note how, in this $N = 2$-dimensional domain, $n = 7$ identifies a 7×7 grid whose cells correspond to locally tuned behaviors of the input-output relationship of the networks.

Furthermore, in Figs. 3 and in 4 we provide quantitative results illustrating the decrease in the approximation error for both theorems. Point-wise and differential errors are reported, whose approximately linear trends in double logarithmic plots against n confirm the inverse polynomial relationship between network complexity and errors. Due to the definition of the Rosenbrock function, for a given i, $\frac{\partial f}{\partial x_i}$ is independent of N if $N > 2$. Hence, the two tracks of differential errors for $N = 3$ and $N = 4$ coincide.

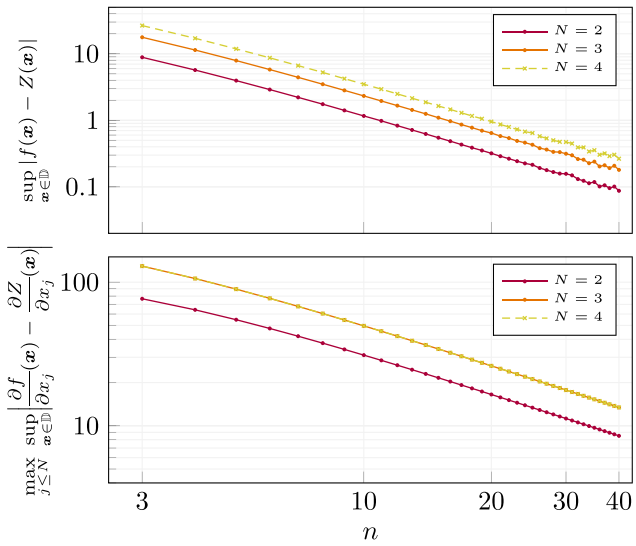

Fig. 4. Convergence of the approximation error for the Rosenbrock function and the maximum approximation error of its partial derivatives for Theorem 2 varying the number of input dimensions N and of divisions of the domain n. Plots with $N \geq 3$ are superimposed due to the nature of the target function.

VI. Network Construction and Proofs of Theorems

A. Network Construction

This subsection aims to show that the first two layers of our network can be programmed so that the outputs of the second hidden layer are specific weakly unimodal piecewise-linear functions $z_k(\boldsymbol{x})$ of the inputs. The arrangement of $\mathcal{L}'$ is reported in Fig. 5. It contains neurons in which only one weight is non-null so that only one input affects the output. This simplifies (2) into

$$u = [wv + b]^+ \quad (6)$$

where v is the only input connected with the neuron with $w \neq 0$. The profile of (6) is piecewise linear, in which the breakpoint is set by the bias and the slope of the non-horizontal piece is set by the weight. In $\mathcal{L}'$, these neurons appear in pairs. Referring to Fig. 5(a), each pair is fed by the same input and contains a *left* neuron with parameters w^{L} and b^{L} and a *right* neuron with parameters w^{R} and b^{R}. By setting $w^{\mathrm{L}} < 0$, $w^{\mathrm{R}} > 0$ and $0 \leq b^{\mathrm{L}} \leq b^{\mathrm{R}} \leq 1$ we obtain the left y^{L} and right y^{R} parts of a trapezoidal profile depending on the common input.

Looking at Fig. 5(b), each input is connected to multiple pairs of neurons in $\mathcal{L}'$. The second hidden layer $\mathcal{L}''$ is then fully connected to $\mathcal{L}'$. In each of its neurons, all the weights are set to 0 but the $2N$ weights connecting the outputs of N pairs in $\mathcal{L}'$, each pair depending

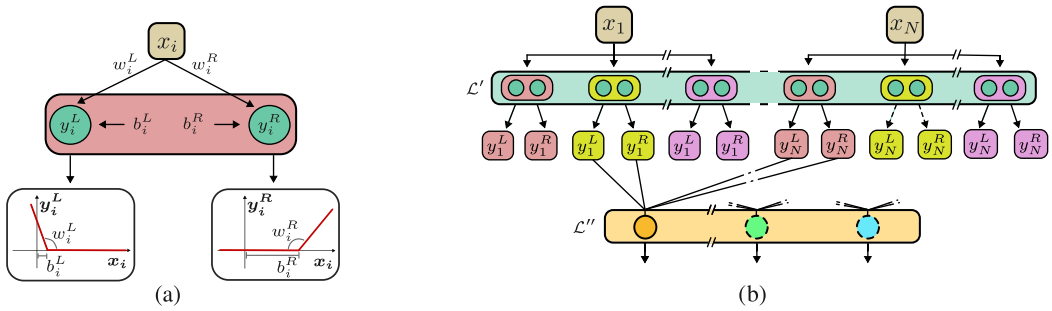

Fig. 5. In (a) the representation of a couple of MAM neurons with non-null weights for input x_i while in (b) the structure of the two MAM layers $\mathcal{L}'$ and $\mathcal{L}''$.

on a different input. These $2N$ connections have weight -1 and the bias equal to 1. The following lemma establishes the link between the inputs of the network and the outputs of $\mathcal{L}''$.

Lemma 1: Let z be any of the outputs of $\mathcal{L}''$. For $N > 1$ and any choice of the quantities $\omega_1, \ldots, \omega_N \in [0, 1]$, $l_1, \ldots, l_N \geq 0$, $\delta_1^{\mathrm{L}}, \ldots, \delta_N^{\mathrm{L}} \geq 0$, and $\delta_1^{\mathrm{R}}, \ldots, \delta_N^{\mathrm{R}} \geq 0$, the two MAM hidden layers can be programmed to yield

$$z(\boldsymbol{x}) = [1 - \Delta(\boldsymbol{x})]^+ \tag{7}$$

where

$$\Delta(\boldsymbol{x}) = \max_{i \in \{1,..,N\}} \left\{ 0, \frac{|x_i - \omega_i| - l_i}{\begin{cases} \delta_i^{\mathrm{L}} & \text{if } x_i < \omega_i \\ \delta_i^{\mathrm{R}} & \text{if } x_i \geq \omega_i \end{cases}} \right\}. \tag{8}$$

Proof of Lemma 1: We focus on a single second hidden layer neuron and assume that the $2N$ neurons of the previous layer that are connected to it by non-null weights have outputs $y_1^{\mathrm{L}}, y_1^{\mathrm{R}}, y_2^{\mathrm{L}}, y_2^{\mathrm{R}}, \ldots$, where the index indicates the input on which that output depends, i.e., y_i^{L} and y_i^{R} depend only on x_i.

For y_i^{L} the non-null input weight $w^{\mathrm{L}} = {}^{-1}/_{\delta_i^{\mathrm{L}}}$ and the bias $b^{\mathrm{L}} = {}^{(\omega_i - l_i)}/_{\delta_i^{\mathrm{L}}}$, while for y_i^{R} the non-null input weight $w^{\mathrm{R}} = {}^{1}/_{\delta_i^{\mathrm{R}}}$ and bias $b^{\mathrm{R}} = {}^{(-\omega_i - l_i)}/_{\delta_i^{\mathrm{R}}}$. By recalling (6) one gets

$$y_i^{\mathrm{L}} = \left[\frac{-x_i + \omega_i - l_i}{\delta_i^{\mathrm{L}}} \right]^+ \quad \text{and} \quad y_i^{\mathrm{R}} = \left[\frac{x_i - \omega_i - l_i}{\delta_i^{\mathrm{R}}} \right]^+. \tag{9}$$

The output of the second hidden layer neuron we are focusing on is therefore

$$\begin{aligned} z &= \left[\max_{i \in \{1,\ldots,N\}} \{0, -y_i^{\mathrm{L}}, -y_i^{\mathrm{R}}\} + \min_{i \in \{1,..,N\}} \{0, -y_i^{\mathrm{L}}, -y_i^{\mathrm{R}}\} + 1 \right]^+ \\ &= \left[1 - \max_{i \in \{1,..,N\}} \{y_i^{\mathrm{L}}, y_i^{\mathrm{R}}\} \right]^+. \end{aligned} \tag{10}$$

Note now that, if $x_i \geq \omega_i$ then $y_i^{\mathrm{R}} \geq 0$ and $y_i^{\mathrm{L}} = 0$ while, if $x_i < \omega_i$ then $y_i^{\mathrm{R}} = 0$ and $y_i^{\mathrm{L}} \geq 0$. Hence, without loss of generality, we may assume that $x_i \geq \omega_i$ for $i = 1, \ldots, N$, being all other cases a variation of this one by suitable symmetry and scaling. With this, $y_i^{\mathrm{L}} = 0$ for $i = 1, \ldots, N$ and (10) becomes

$$\begin{aligned} z &= \left[1 - \max_{i=1,\ldots,N} \left[\frac{x_i - \omega_i - l_i}{\delta_i^{\mathrm{R}}} \right]^+ \right]^+ \\ &= \left[1 - \max_{i=1,\ldots,N} \left\{ 0, \frac{x_i - \omega_i - l_i}{\delta_i^{\mathrm{R}}} \right\} \right]^+ \end{aligned} \tag{11}$$

that is equivalent to the thesis. □

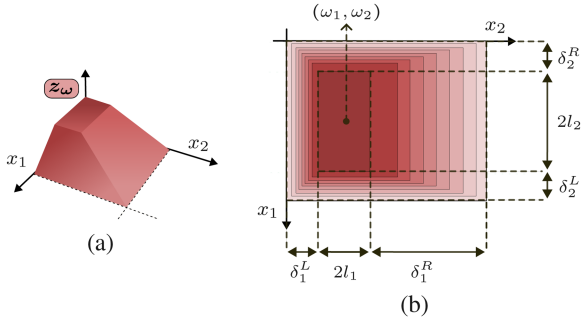

Fig. 6. In (a) the three dimensional representation of a generic $z_{\boldsymbol{\omega}}(\boldsymbol{x})$ for $N = 2$ and its contour plot in (b) showing the role of the various parameters.

To interpret Lemma 1 note that $\Delta(\boldsymbol{x})$ is a scaled measure of how far the input vector $\boldsymbol{x}$ is from the hyper-rectangle centered at $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_N)$ with sides $2l_1, \ldots, 2l_N$. Hence, $z(\boldsymbol{x})$ is maximum and equal to 1 if $\boldsymbol{x}$ belongs to such a hyper-rectangle and has a piecewise-linear decreasing profile when $\boldsymbol{x}$ gets further from $\boldsymbol{\omega}$. Fig. 6 reports an example of a $z(\boldsymbol{x})$ when $N = 2$.

In the following, we will assume that each neuron in the second hidden layer matches a whole subnetwork as implied by Lemma 1. With this, we may re-index the outputs of the second hidden layer as $z_{\boldsymbol{\omega}}(\boldsymbol{x})$ associating each of them with the center of the hyper-rectangle in which $z_{\boldsymbol{\omega}}(\boldsymbol{x}) = 1$. The same is done with the corresponding weights $c_{\boldsymbol{\omega}}$ in the output layers.

B. Universal Approximation Properties With Normalized Linear Output Neuron

Given a positive integer n, define $\Omega = \{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}^N$ and include in the two hidden layers all the subnetworks implied by Lemma 1 to implement the function $z_{\boldsymbol{\omega}}(\boldsymbol{x})$ for each $\boldsymbol{\omega} \in \Omega$. In each of these subnetworks set $\delta_i^{\mathrm{L}} = \delta_i^{\mathrm{R}} = \delta = {}^{1}/_{n}$ for $i = 1, \ldots, N$ and $l_i = 0$ for $i = 1, \ldots, N$. With this, $z_{\boldsymbol{\omega}}(\boldsymbol{x})$ is an $(N+1)$-dimensional pyramid whose base is an N-dimensional hypercube with sides of length 2δ and center in $\boldsymbol{\omega}$.

Given this configuration and the definition of the building blocks, it is easily calculated that the first hidden layer consists of $2nN$ neurons, each with 2 non-null parameter, while the second hidden layer consists of $|\Omega| = (n+1)^N$ each with $2N+1$ non-null parameters, and the last layer has a single neuron combining the $|\Omega|$ outputs of the second hidden layer using $|\Omega|$ further parameters. Hence the total number of neurons and parameters of the network with normalized linear output

are

$$\#\mathrm{n} = 2nN + (n+1)^N + 1 \sim n^N \tag{12}$$

$$\#\mathrm{p} = 4nN + (2N+1)(n+1)^N + (n+1)^N \sim 2(N+1)n^N \tag{13}$$

where the asymptotic trend is for $n \to \infty$.

Proof of Theorem 1: Note first that for any given $\boldsymbol{x} \in \mathbb{X}$, only a limited number of functions $z_{\boldsymbol{\omega}}(\boldsymbol{x})$ are not null. In particular, if $k_i = \lfloor nx_i \rfloor$ for $i = 1, \dots, N$ is the largest integer not exceeding nx_i, then $z_{\boldsymbol{\omega}}(\boldsymbol{x}) > 0$ only if $\boldsymbol{\omega}$ belongs to the set $\Omega_{\boldsymbol{x}} = \{k_1\delta, (k_1+1)\delta\} \times \cdots \times \{k_N\delta, (k_N+1)\delta\}$ that contains the 2^N corners of the N-dimensional hypercube $C_{\boldsymbol{x}} = [k_1\delta, (k_1+1)\delta] \times \cdots \times [k_N\delta, (k_N+1)\delta]$. Hence, we may evaluate $Z(\boldsymbol{x})$ focusing on functions $z_{\boldsymbol{\omega}}(\boldsymbol{x})$ with $\boldsymbol{\omega} \in \Omega_{\boldsymbol{x}}$.

Define the functions

$$\zeta_{\boldsymbol{\omega}}(x) = \frac{z_{\boldsymbol{\omega}}(\boldsymbol{x})}{\sum_{\boldsymbol{\omega}' \in \Omega} z_{\boldsymbol{\omega}'}(\boldsymbol{x})} \tag{14}$$

that are such that $\sum_{\boldsymbol{\omega} \in \Omega} \zeta_{\boldsymbol{\omega}}(\boldsymbol{x}) = \sum_{\boldsymbol{\omega} \in \Omega_{\boldsymbol{x}}} \zeta_{\boldsymbol{\omega}}(\boldsymbol{x}) = 1$ for any $\boldsymbol{x} \in \mathbb{X}$, and set $c_{\boldsymbol{\omega}} = f(\boldsymbol{\omega})$ for each $\boldsymbol{\omega} \in \Omega$.

The error $|f(\boldsymbol{x}) - Z^*(\boldsymbol{x})|$ in Theorem 1 can be written as

$$\begin{aligned} &\left| f(\boldsymbol{x}) - \sum_{\boldsymbol{\omega} \in \Omega_{\boldsymbol{x}}} f(\boldsymbol{\omega}) \zeta_{\boldsymbol{\omega}}(\boldsymbol{x}) \right| \\ &= \left| \sum_{\boldsymbol{\omega} \in \Omega_{\boldsymbol{x}}} [f(\boldsymbol{x}) - f(\boldsymbol{\omega})] \zeta_{\boldsymbol{\omega}}(\boldsymbol{x}) \right| \\ &\leq \max_{\boldsymbol{x} \in \mathbb{X}} \max_{\substack{\boldsymbol{\xi} \in C_{\boldsymbol{x}} \\ \boldsymbol{\omega} \in \Omega_x}} |f(\boldsymbol{\xi}) - f(\boldsymbol{\omega})| . \end{aligned} \tag{15}$$

Since $f : \mathbb{X} \mapsto \mathbb{R}$ is continuous on the compact domain $\mathbb{X}$, it is also uniformly continuous and, for any given level of tolerance $\epsilon > 0$, there is a Δx such that for any $\boldsymbol{x}', \boldsymbol{x}'' \in \mathbb{X}$ with distance $\|\boldsymbol{x}' - \boldsymbol{x}''\|_2 \leq \Delta x$ we have $|f(\boldsymbol{x}') - f(\boldsymbol{x}'')| \leq \epsilon$. For a given $\boldsymbol{x}$, the distance between any $\boldsymbol{\xi} \in C_{\boldsymbol{x}}$ and any $\boldsymbol{\omega} \in \Omega_{\boldsymbol{x}}$ is $\|\boldsymbol{\xi} - \boldsymbol{\omega}\|_2 \leq \delta\sqrt{N}$. Since $\delta = 1/n$ we can select n so that

$$|f(\boldsymbol{x}) - Z^*(\boldsymbol{x})| \leq \max_{\boldsymbol{x} \in \mathbb{X}} \max_{\substack{\boldsymbol{\xi} \in C_{\boldsymbol{x}} \\ \boldsymbol{\omega} \in \Omega_{\boldsymbol{x}}}} |f(\boldsymbol{\xi}) - f(\boldsymbol{\omega})| \leq \epsilon$$

Actually, if f is Lipschitz with constant L, then we know that the above inequality is satisfied for $\epsilon \geq L\delta\sqrt{N} = \frac{L}{n}\sqrt{N}$. For large n (and thus small ϵ), this can be paired with (13) to say that the number of parameters needed to meet the given tolerance should grow as ϵ^{-N} as the tolerance decreases. □

C. Universal Approximation Properties With Linear Output Neuron

In this case, the approximation capabilities of our network over the whole domain depend on the local behaviour of subnetworks converging not in a single second-hidden-layer neuron but in $2N$ second-hidden-layer neurons.

Formally speaking, given a positive integer n, define $\Omega = \left\{\frac{1}{2n}, \frac{3}{2n}, \frac{5}{2n}, \dots, \frac{2n-1}{2n}\right\}^N$, and set

$$\ell = \frac{1}{2}\left(\sqrt{\frac{n+2}{n}} - 1\right) \tag{16}$$

as well as $\delta = \ell^2$ so that $2(\ell + \delta) = {}^1\!/_n$.

For any $\boldsymbol{\omega} \in \Omega$ we include subnetwork neurons of the second hidden layer with outputs labeled $z_{\boldsymbol{\omega}^{1-}}, z_{\boldsymbol{\omega}^{1+}}, \dots, z_{\boldsymbol{\omega}^{N-}}, z_{\boldsymbol{\omega}^{N+}}$ as well as all the previous neurons needed to compute such outputs. The expression of each $z_{\boldsymbol{\omega}^{j\pm}}$ is given by Lemma 1 and thus is defined by the center point $\boldsymbol{\omega}^{j\pm} = (\omega_1^{j\pm}, \dots, \omega_N^{j\pm})$, by the slopes $\delta_1^{\mathrm{L},j\pm}, \dots, \delta_N^{\mathrm{L},j\pm}$ and $\delta_1^{\mathrm{R},j\pm}, \dots, \delta_N^{\mathrm{R},j\pm}$, as well as by the side lengths $l_1^{j\pm}, \dots, l_N^{j\pm}$.

In a subnetwork, everything depends on two quantities δ and ℓ that are used to set

$$\omega_i^{j\pm} = \begin{cases} \omega_i & \text{if } i \neq j \\ \omega_i \pm \ell & \text{if } i = j \end{cases} \qquad l_i^{j\pm} = \begin{cases} \ell & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

$$\delta_i^{\mathrm{R},j+} = \delta \qquad \delta_i^{\mathrm{R},j-} = \begin{cases} \delta & \text{if } i \neq j \\ 2\ell & \text{if } i = j \end{cases}$$

$$\delta_i^{\mathrm{L},j+} = \begin{cases} \delta & \text{if } i \neq j \\ 2\ell & \text{if } i = j \end{cases} \qquad \delta_i^{\mathrm{L},j-} = \delta$$

for $i, j = 1, \dots, N$.

To give some intuitive grounding to the above definitions, Fig. 7 reports example profiles for 4 output functions $z_{\boldsymbol{\omega}^{1-}}, z_{\boldsymbol{\omega}^{1+}}, z_{\boldsymbol{\omega}^{2-}}, z_{\boldsymbol{\omega}^{2+}}$ with $N = 2$. According to the construction of the network, the first hidden layer contains $2nN$ neurons, each defined by 2 non-null parameters. The second hidden layer consists of $2Nn^N$ neurons each having $2N + 1$ non-null parameters. In the last layer, there is a single weight for each of the $2Nn^N$ outputs of the second hidden layer. Overall, the resulting network has

$$\#\mathrm{n} = 2nN + 2Nn^N + 1 \sim 2Nn^N \tag{17}$$

$$\#\mathrm{p} = 4nN + (2N+1)2Nn^N + 2Nn^N \sim 4N(N+1)n^N \tag{18}$$

in which we have reported the asymptotic trends when $n \to \infty$.

Given an $\boldsymbol{\omega}$, we may define the subset

$$X_{\boldsymbol{\omega}}^{\blacksquare} = \left\{ \boldsymbol{x} \in \mathbb{X} \mid \max_{i=1,\dots,N} \{|x_i - \omega_i|\} \leq \ell \right\}. \tag{19}$$

The approximation capabilities depend on the behavior of the output of the subnetworks in each $X_{\boldsymbol{\omega}}^{\blacksquare}$ and thus in $\mathbb{D} = \bigcup_{\boldsymbol{\omega} \in \Omega} X_{\boldsymbol{\omega}}^{\blacksquare}$. The following lemma holds.

Lemma 2:

$$\mu\left(\bigcup_{\boldsymbol{\omega} \in \Omega} X_{\boldsymbol{\omega}}^{\blacksquare}\right) \geq 1 - \frac{N}{2n}$$

Proof of Lemma 2: From (19) we get that the $X_{\boldsymbol{\omega}}^{\blacksquare}$ are disjoint and their individual measure is $(2\ell)^N$.

Since the cardinality of Ω is n^N we have

$$\mu\left(\bigcup_{\boldsymbol{\omega} \in \Omega} X_{\boldsymbol{\omega}}^{\blacksquare}\right) = (2\ell)^N n^N = \left(\sqrt{n^2 + 2n} - n\right)^N \geq 1 - \frac{N}{2n}$$

where we have exploited (16). □

Lemma 3: Given any choice of $N + 1$ coefficients a and b_j for $j = 1, \dots, N$, one may choose $2N$ weights $c^{j\pm}$ with $j = 1, \dots, N$ such that

$$Z_{\boldsymbol{\omega}}(\boldsymbol{x}) \equiv \sum_{j=1}^{N} c^{j\pm} z_{\boldsymbol{\omega}^{j\pm}}(\boldsymbol{x}) = a + \sum_{j=1}^{N} b_j x_j \tag{20}$$

for any $\boldsymbol{x} \in X_{\boldsymbol{\omega}}^{\blacksquare}$, where $Z_{\boldsymbol{\omega}}(\boldsymbol{x})$ remains implicitly defined.

Proof of Lemma 3: Due to the definition of $\boldsymbol{\omega}^{j\pm}$ we have

$$\begin{aligned} X_{\boldsymbol{\omega}}^{\blacksquare} &= [\omega_1 - \ell, \omega_1 + \ell] \times \cdots \times [\omega_N - \ell, \omega_N + \ell] \\ &= \left[\omega_1^{1-}, \omega_1^{1+}\right] \times \cdots \times \left[\omega_N^{N-}, \omega_N^{N+}\right] \end{aligned}$$

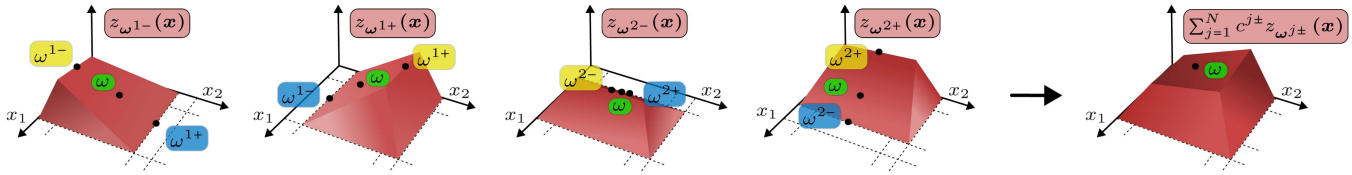

Fig. 7. Three dimensional plots of the functions $z_{\omega^{1-}}, z_{\omega^{1+}}, z_{\omega^{2-}}, z_{\omega^{2+}}$ with $N = 2$ that are combined to build $\sum_{j=1}^{N} c^{j\pm} z_{\omega^{j\pm}}(\boldsymbol{x})$.

Hence, if $\boldsymbol{x} \in X_{\boldsymbol{\omega}}^{\blacksquare}$ we know that $\omega_j^{j-} \leq x_j \leq \omega_j^{j+}$ for $j = 1, \ldots, N$. Moreover, since by definition for any $i, j = 1, \ldots, N$ and $i \neq j$ we have $\omega_i^{j+} - \omega_i^{j-} = 2\ell$ and $\omega_i^{j-} + \omega_i^{j+} = 2\omega_i$, then $|x_i - \omega_i^{j\pm}| \leq \ell$ when $i \neq j$. Therefore, one can apply Lemma 1 and compute $\Delta(\boldsymbol{x})$, for which all the terms in (8) but $|x_j - \omega_j^{j\pm}|$ are non-positive, thus yielding $z_{\omega^{j\pm}}(\boldsymbol{x}) = 1 - {}^{|x_j - \omega_j^{j\pm}|}/_{(2\ell)}$.

Without loss of generality, translate $X_{\boldsymbol{\omega}}$ so that $\boldsymbol{\omega} = (\ell, \ldots, \ell)$. This implies $\omega_j^{j-} = 0$ and $\omega_j^{j+} = 2\ell$ for $j = 1, \ldots, N$, thus yielding $z_{\omega^{j-}}(\boldsymbol{x}) = 1 - \frac{x_j}{2\ell}$ and $z_{\omega^{j+}}(\boldsymbol{x}) = \frac{x_j}{2\ell}$. With this,

$$\begin{aligned}\sum_{j=1}^{N} c^{j\pm} z_{\omega^{j\pm}}(\boldsymbol{x}) &= \sum_{j=1}^{N} \left[c^{j-} \left(1 - \frac{x_j}{2\ell}\right) + c^{j+} \frac{x_j}{2\ell} \right] \\ &= \sum_{j=1}^{N} c^{j-} + \sum_{j=1}^{N} \left(c^{j+} - c^{j-}\right) \frac{x_j}{2\ell}\end{aligned}$$

that can yield any affine function $f(x) = a + \sum_{j=1}^{N} b_j x_j$ by setting, for $j = 1, \ldots, N$,

$$c^{j-} = \frac{a}{N} \quad \text{and} \quad c^{j+} = c^{j-} + 2\ell b_j \tag{21}$$

□

The above characterization of the output of Z-subnetworks allows to prove their local approximation capabilities.

Lemma 4: Given any function $f \in \mathcal{C}^2(\mathbb{X})$, there is a constant $M > 0$ such that

$$\begin{aligned}|f(\boldsymbol{x}) - Z_{\boldsymbol{\omega}}(\boldsymbol{x})| &\leq MN^2\ell^2 \\ \left| \frac{\partial f}{\partial x_j}(\boldsymbol{x}) - \frac{\partial Z_{\boldsymbol{\omega}}}{\partial x_j}(\boldsymbol{x}) \right| &\leq MN\ell\end{aligned}$$

for any $\boldsymbol{x} \in X_{\boldsymbol{\omega}}^{\blacksquare}$ and any $\boldsymbol{\omega} \in \Omega$.

Proof of Lemma 4: Since $f \in \mathcal{C}^2(\mathbb{X})$ and $\mathbb{X}$ is compact, $H \geq 0$ exists such that

$$\left| \frac{\partial^2 f}{\partial x_i x_j}(\boldsymbol{x}) \right| \leq H \tag{22}$$

for any $\boldsymbol{x} \in \mathbb{X}$ and $i, j = 1, \ldots, N$. Since $\boldsymbol{x} \in X_{\boldsymbol{\omega}}^{\blacksquare}$, and thus $|x_i - \omega_i| \leq \ell$, the above bound can be used jointly with the Taylor expansions of f and its derivatives around $\boldsymbol{\omega}$

$$\begin{aligned}f(\boldsymbol{x}) = f(\boldsymbol{\omega}) &+ \sum_{i=1}^{N} \frac{\partial f}{\partial x_i}(\boldsymbol{\omega})(x_i - \omega_i) \\ &+ \sum_{i=1}^{N} \sum_{j=1}^{N} R_{i,j}(\boldsymbol{x}) \left(x_i - \omega_i\right)\left(x_j - \omega_j\right)\end{aligned} \tag{23}$$

$$\frac{\partial f}{\partial x_i}(\boldsymbol{x}) = \frac{\partial f}{\partial x_i}(\boldsymbol{\omega}) + \sum_{j=1}^{N} S_{i,j}(\boldsymbol{x}) \left(x_j - \omega_j\right) \quad i = 1, \ldots, N \tag{24}$$

where $R_{i,j}$ and $S_{i,j}$ are Taylor theorem remainder terms. Their error terms satisfy

$$\begin{aligned}&\left| \sum_{i=1}^{N} \sum_{j=1}^{N} R_{i,j}(\boldsymbol{x}) \left(x_i - \omega_i\right)\left(x_j - \omega_j\right) \right| \\ &\quad =\leq N^2\ell^2 \frac{1}{2} \max_{k,l=1,\ldots,N} \max_{\boldsymbol{\xi} \in \mathbb{X}} \left| \frac{\partial^2 f}{\partial x_k x_l}(\boldsymbol{\xi}) \right| \\ &\quad =\leq \frac{1}{2} HN^2\ell^2\end{aligned} \tag{25}$$

and

$$\begin{aligned}&\left| \sum_{j=1}^{N} S_{i,j}(\boldsymbol{x}) \left(x_j - \omega_j\right) \right| \\ &\quad =\leq N^2\ell^2 \frac{1}{2} \max_{j=1,\ldots,N} \max_{\boldsymbol{\xi} \in \mathbb{X}} \left| \frac{\partial^2 f}{\partial x_i x_j}(\boldsymbol{\xi}) \right| \\ &\quad =\leq \frac{1}{2} HN\ell \qquad i = 1, \ldots, N.\end{aligned} \tag{26}$$

At this point, one may exploit Lemma 3 to set the weights $c^{j\pm}$ and yield

$$\begin{aligned}Z_{\boldsymbol{\omega}}(\boldsymbol{x}) &= f(\boldsymbol{\omega}) + \sum_{i=1}^{N} \frac{\partial f}{\partial x_i}(\boldsymbol{\omega}) \left(x_i - \omega_i\right) \\ &= \left[f(\boldsymbol{\omega}) - \sum_{i=1}^{N} \frac{\partial f}{\partial x_i}(\boldsymbol{\omega}) \omega_i \right] + \sum_{i=1}^{N} \frac{\partial f}{\partial x_i}(\boldsymbol{\omega}) x_i\end{aligned} \tag{27}$$

which is also such that $\frac{\partial Z_{\boldsymbol{\omega}}}{\partial x_i}(\boldsymbol{x}) = \frac{\partial f}{\partial x_i}(\boldsymbol{\omega})$. Hence, we may program $Z_{\boldsymbol{\omega}}$ to reproduce the behaviour of f and its derivatives in $X_{\boldsymbol{\omega}}^{\blacksquare}$, and the approximation errors can be derived exploiting (23) with (25) and (24) with (26) to obtain

$$|Z_{\boldsymbol{\omega}}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \frac{1}{2} HN^2\ell^2, \tag{28}$$

$$\left| \frac{\partial Z_{\boldsymbol{\omega}}}{\partial x_i}(\boldsymbol{x}) - \frac{\partial f}{\partial x_i}(\boldsymbol{x}) \right| \leq \frac{1}{2} HN\ell \tag{29}$$

for any $\boldsymbol{x} \in \mathbb{D} = \bigcup_{\boldsymbol{\omega} \in [\Omega} X_{\boldsymbol{\omega}}^{\blacksquare}$. This yields the thesis with $M = {}^{H}/_{2}$.□

Proof of Theorem 2: Note first that from (16) we get that if $\bar{\ell} > 0$, the generic inequality $\ell \leq \bar{\ell}$ is satisfied by $n \geq \frac{1}{2} \frac{1}{\bar{\ell} + \bar{\ell}^2}$ and thus by the slightly looser $n \geq \frac{1}{2\bar{\ell}}$.

Lemma 4 implies that to meet the error tolerances we should set $\ell \leq \sqrt{\frac{\epsilon}{MN^2}}$ and $\ell \leq \frac{\epsilon'}{MN}$. In addition to that, Lemma 2 implies that we should also set $n \geq \frac{N}{2\xi}$.

Hence, to satisfy all the requirements is enough to set

$$n \geq \frac{N}{2} \max \left\{ \sqrt{\frac{M}{\epsilon}}, \frac{M}{\epsilon'}, \frac{1}{\xi} \right\}$$

With this, we recall (18) to produce the second part of the thesis. □

VII. Conclusion

We proved two theorems with different assumptions and theses that confirm that hybrid MAM&MAC networks are universal approximators. We also presented an example to show how the two theorems apply in practice and provided quantitative results on the vanishing of approximation errors. These theorems give ground to aggressive pruning strategies whose first step is the substitution of MAC neurons with MAM neurons in the hidden layers of a DNN while still maintaining overall functionality.

Acknowledgment

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from in part by the 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, PE00000013). 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] Y. Li, T. Yao, Y. Pan, and T. Mei, "Contextual transformer networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1489–1500, Feb. 2023.

[2] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 604–624, Feb. 2021.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[4] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., 2017, pp. 6000–6010.

[5] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[7] A. Pinkus, "Approximation theory of the MLP model in neural networks," *Acta Numerica*, vol. 8, pp. 143–195, Jan. 1999.

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

[9] G. Bai et al., "Beyond efficiency: A systematic survey of resource-efficient large language models," Jan. 2024, *arXiv:2401.00625*.

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

[11] L. Prono et al., "A multiply-and-max/min neuron paradigm for aggressively prunable deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jan. 17, 2025, doi: 10.1109/TNNLS.2025.3527644.

[12] P. Bich, L. Prono, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "Multiply-and-Max/min neurons at the edge: Pruned autoencoder implementation," in *Proc. IEEE 66th Int. Midwest Symp. Circuits Syst.*, 2023, pp. 629–633.

[13] L.-X. Wang, "Fuzzy systems are universal approximators," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1992, pp. 1163–1170.

[14] B. Kosko, "Fuzzy systems as universal approximators," *IEEE Trans. Comput.*, vol. 43, no. 11, pp. 1329–1333, Nov. 1994.

[15] R. Rovatti, "Fuzzy piecewise multilinear and piecewise linear systems as universal approximators in Sobolev norms," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 2, pp. 235–249, May 1998.

[16] G. Gripenberg, "Approximation by neural networks with a bounded number of nodes at each level," *J. Approximation Theory*, vol. 122, no. 2, pp. 260–266, Jun. 2003.

[17] B. Hanin and M. Sellke, "Approximating continuous functions by ReLU nets of minimal width," Mar. 2018, doi: 10.48550/arXiv.1710.11278.

[18] N. J. Guliyev and V. E. Ismailov, "On the approximation by single hidden layer feedforward neural networks with fixed weights," *Neural Netw.*, vol. 98, pp. 296–304, Feb. 2018.

[19] Y. Cai, "Achieve the minimum width of neural networks for universal approximation," in *Proc. 11th Int. Conf. Learn. Representations*, 2023, pp. 1–15.

[20] D.-X. Zhou, "Universality of deep convolutional neural networks," *Appl. Comput. Harmon. Anal.*, vol. 48, no. 2, pp. 787–794, 2020.

[21] O. A. Manita, M. A. Peletier, J. W. Portegies, J. Sanders, and A. Senen-Cerda, "Universal approximation in dropout neural networks," *J. Mach. Learn. Res.*, vol. 23, no. 1, Jan. 2022, Art. no. 19.

[22] Y. Lu and J. Lu, "A universal approximation theorem of deep neural networks for expressing probability distributions," in *Proc. Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., 2020, pp. 3094–3105.

[23] C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, and X. Wu, "Are graph convolutional networks with random weights feasible?," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 2751–2768, Mar. 2023.

[24] S.-Q. Zhang and Z.-H. Zhou, "Theoretically provable spiking neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., 2022, pp. 19345–19356.

[25] Z. Wu, M. Xiao, C. Fang, and Z. Lin, "Designing universally-approximating deep neural networks: A first-order optimization approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 9, pp. 6231–6246, Sep. 2024.

[26] C. Anil, J. Lucas, and R. B. Grosse, "Sorting out Lipschitz function approximation," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 291–301.

[27] U. Tanielian, M. Sangnier, and G. Biau, "Approximating Lipschitz continuous functions with GroupSort neural networks," in *Proc. 24th Int. Conf. Artif. Intell. Statist.*, 2021, pp. 442–450.

[28] B. Zhang, D. Jiang, and L. Wang, "Rethinking Lipschitz neural networks and certified robustness: A Boolean function perspective," in *Proc. 36th Conf. Neural Inf. Process. Syst.*, 2022.

[29] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Proc. Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., 2017, pp. 6231–6239.

[30] A. Neufeld and P. Schmocker, "Universal approximation results for neural networks with non-polynomial activation function over non-compact domains," Oct. 2024, *arXiv:2410.14759*.

[31] H. H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *Comput. J.*, vol. 3, no. 3, pp. 175–184, Jan. 1960.