Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(37^{th}cycle)

Towards Smaller, Lighter, and More Transparent AI

By

Philippe Bich

******

Supervisor(s):
Prof. Gianluca Setti, Ph.D, Supervisor

Politecnico di Torino
2025

Declaration

I hereby declare that the content and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Philippe Bich
2025

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Acknowledgements

For many reasons, this journey has not been easy. I am certain it has helped me to grow not only professionally, but also as a person. My deepest thanks go to everyone who supported me over the years and to my colleagues at the SmartData Lab. Together, we shared three unforgettable years and experiences that extended far beyond the academic growth. My only true regret is not becoming a Trashball Maestro, but knowing the title has passed to Luca Vassio still gives me peace, it is in excellent hands.

A special thanks goes to Matteo, with whom I shared many passionate debates about the future of LLMs (and much more), but who also made going to the office a true pleasure. My thanks also go to Nik, Tailai, Kai, Giordano, Francesco, Gabriele and Andrea, the friendship we built has remained strong. I also want to express my gratitude to Luciano, whose guidance over the years was invaluable.

Finally, I am deeply grateful to my parents. Despite the significant challenges we have faced, we are still standing, changed, but stronger. I truly believe the future will repay every sacrifice we have made.

Abstract

In recent years, the evolution of Artificial Intelligence (AI) has been marked by a rapid increase in the complexity of neural models. From early architectures such as LeNet and VGG to more advanced designs like the Vision Transformer (ViT) and the latest Large Language Models (LLMs), the number of parameters of these neural networks has grown from thousands to billions.

Within this rapidly evolving landscape, this thesis explores three key directions that I believe are becoming increasingly central to make the future AI: *i) smaller* through pruning techniques aimed at reducing the size and complexity of neural networks; *ii) lighter* by designing algorithms optimized for low-power AI, enabling efficient deployment in constrained environments; *iii) more transparent* by addressing the opacity of large-scale models via concept-based interpretability methods.

The thesis is structured in three parts. The first introduces a study on a novel pruning technique that leverages a new type of neuron, based on Multiply and Max/Min operations, to achieve aggressive model reduction. The second part presents a work on lightweight algorithms designed for event-based cameras (low-power neuromorphic sensors), tackling key challenges in computer vision. The final part explores the use of variational inference to enhance the responsiveness and performance of state-of-the-art models for concept-based explainability in neural networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The age of billion-parameter models

11^{th} December 2015. I was a freshman at Politecnico di Torino attending my first classes. Meanwhile, on the other side of the world, a new reality was created. One that I would not come to know until much later in my academic journey. OpenAI was founded in San Francisco by Sam Altman and Elon Musk, at a time when the most advanced artificial intelligence models were based on ResNet, a convolutional network with 100 million parameters, which seemed an incredibly powerful model. Fast forward nine years, and this company, which has revolutionized the world, recently unveiled ChatGPT-5, a model that rumors suggest it could count more than 2 trillion parameters. Models that when I began my studies seemed enormous, are now considered toy models. As I finish writing this thesis in July 2025, Meta AI has recently released its flagship open-source Llama 4 models, which have quickly become a benchmark in the academic community, initially thanks to their openness and manageable sizes (though already exceeding 1 billion parameters). Yet, like the rest of the field, they have since grown dramatically: the smallest model in the Llama 4 family available at the time of writing, Llama 4-Scout, now counts 109 billion parameters, while Llama Mavericks reaches 400 billion.

During these years at Politecnico di Torino, I had the privilege of observing an incredibly rapid transition in the field of artificial intelligence, one that has accelerated in recent years to the point where it is almost impossible to

keep up. We have now entered the era of massive models, I call it *the age of billion-parameter models*. This thesis, which marks the culmination of my time at this university that has given me so much and helped me grow both as an engineer and as a person, aims to explore three key topics that have become even more crucial during my PhD journey: the compression of neural models, the exploitation of low-power sensors together with the creation of lightweight networks that harness their properties, as well as the explainability of these models. Now, more than ever, these models resemble black boxes, and their inner workings are increasingly difficult to understand and study.

1.2 Thesis outline

To sum up, this thesis will be organized as follows. In the first part, titled ***Smaller***, I will present a novel type of neuron based on the Multiply-And-Max/Min (MAM) paradigm. Compared to the classical neuron model based on Multiply-and-ACcumulate (MAC), this new neuron allows for aggressive pruning of neural networks while significantly limiting the loss in accuracy. This work was presented in [115] and published in *IEEE Transactions on Neural Networks and Learning Systems* (IEEE TNNLS, 2025). In addition to demonstrating the practical applications of this novel neuron, I also delve into the theoretical aspects, proving that neural networks using layers with MAM neurons are still universal approximators. This work has been published in *IEEE Transactions on Pattern Analysis and Machine Intelligence* (IEEE TPAMI, 2025).

In the second part of this thesis, titled ***Lighter***, I will present a couple of works related to a type of lightweight sensor: the event-based camera. Event cameras are bio-inspired sensors that asynchronously capture changes in a scene with high temporal resolution and low power consumption. This section includes the creation of a dataset for event-camera-based learning and the development of a lightweight algorithm to adapt the sparse and asynchronous nature of DVS (Dynamic Vision Sensor) data for traditional, non-recurrent deep learning models thus avoiding the need for recurrent networks, which are often inefficient in hardware. This work is based on two papers [16, 26] which

were published in the *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE/CVF CVPR 2023 and 2024, respectively).

In the final part of the thesis, titled ***More transparent***, I will discuss another important topic in the context of increasingly large neural networks: explainability, with a particular focus on concept-based explainability. I will introduce *V-CEM*, which leverages variational inference to improve intervention responsiveness in Concept Embedding Models. This work is based on [30] and was accepted at *The 3rd World Conference on Explainable AI* (2025).

At the beginning of each chapter, I will briefly introduce the foundational material necessary to understand the work presented. This will provide the reader with the essential background knowledge required to follow the discussions and results in each section. In the final section of each chapter, I will present some concluding remarks, reflecting on the main findings, potential implications, and areas for future research. Finally, in Section 7, I report some closing thoughts that will aim to synthesize the content and highlight the significance of the contributions discussed in this thesis.

1.3 List of publications

The following list encompasses all the publications published as a Ph.D. candidate:

- **Visual Navigation Using Sparse Optical Flow and Time-To-Transit**, C. Boretti, P. Bich, Y. Zhang, J. Baillieul. In: International Conference on Robotics and Automation (ICRA), 2022.
- **Aggressively Prunable MAM²-based Deep Neural Oracle for ECG Acquisition by Compressed Sensing**, P. Bich, L. Prono, M. Mangia, F. Pareschi, R. Rovatti, G. Setti. In: IEEE Biomedical Circuits and Systems Conference (BioCAS), 2022
- **PEDRo: an Event-based Dataset for Person Detection in Robotics**, C. Boretti, P. Bich, F. Pareschi, L. Prono, R. Rovatti, G. Setti. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023.

- **Event-based Eye Tracking. AIS 2024 Challenge Survey**, Z. Wang et al. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024.
- **Optimizing Vision Transformers: Leveraging Max and Min Operations for Efficient Pruning**, P. Bich, C. Boretti, L. Prono, F. Pareschi, R. Rovatti, G. Setti. In IEEE 6th International Conference on AI Circuits and Systems (AICAS), 2024.
- **Memory in Motion: Exploring Leaky Integration of Time Surfaces for Event-based Eye-tracking**, C. Boretti, P. Bich, L. Prono, F. Pareschi, R. Rovatti, G. Setti. In: 2024 IEEE Biomedical Circuits and Systems Conference (BioCAS), 2024.
- **A Multiply-And-Max/min Neuron Paradigm for Aggressively Prunable Deep Neural Networks**, L. Prono, P. Bich, C. Boretti, M. Mangia, F. Pareschi, R. Rovatti, G. Setti. In: IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2025.
- **On the Universal Approximation Properties of Deep Neural Networks using MAM Neurons**, P. Bich, A. Enttsel, L. Prono, A. Marchioni, F. Pareschi, M. Mangia, G. Setti, R. Rovatti. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2025.
- **V-CEM: Bridging Performance and Intervenability in Concept-based Models**, F. De Santis, G. Ciravegna, P. Bich, D. Giordano, T. Cerquitelli. In: The 3rd World Conference on Explainable Artificial Intelligence (xAI), 2025.
- **Towards Better Generalization and Interpretability in Unsupervised Concept-Based Models**, F. De Santis, P. Bich, G. Ciravegna, P. Barbiero, T. Cerquitelli, D. Giordano. In: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2025.
- **Linearly-Interpretable Concept Embedding Models for Text Analysis**, F. De Santis, P. Bich, G. Ciravegna, P. Barbiero, T. Cerquitelli, D. Giordano. In: Machine Learning, 2025.

The following list includes all the publications authored as a Ph.D. candidate that are currently under review:

- **MESA: A Dynamical Attention-based Pre-processing Pipeline for Event-based Computer Vision Tasks**, P. Bich, L. Prono, C. Boretti, F. Pareschi, R. Rovatti, G. Setti. Submitted to: IEEE Transactions on Circuits and Systems Part II: Express Briefs.
- **FOREST-GC: A conFOrmable Rendering Engine for Synthetic Tree Generation and Counting**, L. Prono, N. Dhieb, P. Bich, C. Boretti, F. Pareschi, M. Brini, H. Ghazzai, R. Rovatti, G. Setti. Submitted to: IEEE Access.

Part I

Smaller

Chapter 2

The MAM neuron

The growing interest in Internet of Things 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 in recent years. 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 first chapter, I will present a novel neuron structure based on the Multiply-And-Max/min (MAM) map-reduce paradigm. After presenting some background material, I will show that by exploiting this new paradigm, it is possible to build naturally and aggressively prunable DNN layers with 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, I employ different models, from the old-school models AlexNet and VGG-16 to the more recent ViT-B/16, and different computer vision datasets such as CIFAR-10, CIFAR-100 and ImageNet-1K. Multiple pruning approaches are applied, ranging from single-shot methods to training-dependent and iterative techniques.

2.1 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 to be 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 [13].

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 [68]. 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 [100, 22, 83] and for its wide range of applications from Augmented Reality [50], Natural Language Processing [79], Computer Vision [156] to Compressed Sensing for biomedical signals [116].

Classical pruning approaches in the literature [13] 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-ACumulate (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 multiply, while *reduce* is accumulate. 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, the idea is to find a *different reduce operation* that is less sensible to this reduction of elements.

To do so, let's 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 the intuition to modify the sum-based reduce with a *max/min-based reduce*, leading to a new model of neuron, relying on the 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 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. I better elaborate on this in Appendix A.1.

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, I will show that the "same" global behaviour of the DNN can be retrieved after the substitution of MAC-based neurons with MAM-based ones for the whole network. Contrary to MAC, this 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 Chapter:

- I formalize the definition of the MAM operation and the *vanishing contributions* training approach;
- I present MAM pruning properties by means of a driving example: a DNN trained on MNIST;

- I 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);
- I analyze the performance of MAM with AC/DC [111] and the *lottery ticket* [41] pruning approaches, to further show how it is possible to use MAM in conjunction with training-dependent pruning methods.
- with the *lottery ticket* iterative pruning approach, I show that employing a MAM-based DNN drastically reduces the number of training processes needed to prune the initial architecture.

The rest of the Chapter is structured as follows. In Section 2.2 I provide an extended description of the novel MAM map-reduce paradigm together with an explanation of the vanishing contributions technique. Then, in Section 2.3, I introduce an overview of DNN pruning and describe some pruning scoring strategies. In Section 2.4, I present a first insight into the properties of MAM by training and pruning a simple FC-based model. In Section 2.5 I 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 2.6, I show how MAM can also be coupled with more complex pruning techniques such as AC/DC [111] and the lottery ticket iterative pruning [41]. Finally, the conclusion is drawn.

2.2 The MAM-based layer architecture

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

2.2.1 Layer description

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

$$z = Wx + b \tag{2.1}$$

$$y = f(z). \tag{2.2}$$

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

As I seek a new family of neurons changing the aggregation phase, I need to explicitly separate the multiply and the reduce operations of (2.1). To do so, first I isolate the map phase (i.e. multiply) operation by defining the *weighted inputs* matrix $V \in \mathbb{R}^{M \times N}$ as

$$V = \begin{bmatrix} w_1 \odot x^\top \\ w_2 \odot x^\top \\ \vdots \\ w_M \odot x^\top \end{bmatrix} \tag{2.3}$$

where $w_1, w_2, \ldots, w_M$ is the i-th row of W and $\odot$ is the element-wise product. More explicitly

$$v_{ij} = w_{ij} x_j \quad \text{with } i \in \{1, .., M\}, j \in \{1, .., N\} \tag{2.4}$$

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

$$z_i = \sum_{j=1}^{N} v_{ij} + b_i \quad \text{with } i \in \{1, .., M\} \tag{2.5}$$

where z_i is the i-th element of vector z. Fig. 2.1 shows a summary of the multiply and accumulate splitting.

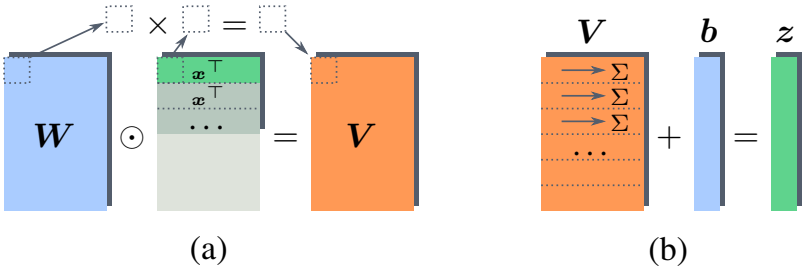

Fig. 2.1 Split up of the multiply and accumulate operations for the MAC map-reduce paradigm. The classic MAC operation as a matrix-vector dot product Wx is decomposed in the multiply operation (a), where I multiply element-wise each row of W with $x^\top$, and in the accumulate operation (b) where the elements of V are summed along the rows of the matrix. Figure from [115].

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 *post-synaptic* connection – and j – indicating the *pre-synaptic* connection. With reference to (2.5), 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 V with the smallest magnitudes. When pruning the largest possible amount of interconnections, only a couple per row of V are kept[1].

Following this lead, I 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 V, as suggested in[117, 11]. 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 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.

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

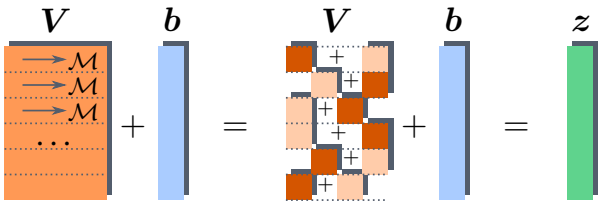

Fig. 2.2 Max/min reduce operation. For each row of V, the minimum and maximum values are selected, then added together. Figure from [115].

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.1). Second, the two contributions are capable of balancing themselves[2], resulting in a good stability of the neuron output. With this choice (2.5) modifies as

$$z_i = \underset{j=1}{\overset{N}{\mathcal{M}}} v_{ij} + b_i \quad \text{with } i \in \{1,..,M\} \tag{2.6}$$

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

$$\underset{j=1}{\overset{N}{\mathcal{M}}} v_{ij} \triangleq \max_{j\in\{1,..,N\}} v_{ij} + \min_{j\in\{1,..,N\}} v_{ij}. \tag{2.7}$$

I will refer to the map-reduce operations in (2.3) and (2.6) as the Multiply-And-Max/min paradigm (MAM). These equations, together with (2.2), describe the novel MAM-based layer, while Fig. 2.2 shows a visual representation of (2.6).

2.2.2 A bridge from MAC to MAM: the vanishing contributions method

In general, training DNN layers directly on the basis of (2.6) results in poor performance. In fact, as (2.6) forward-propagates exactly two values for each row of 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

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

slowing the optimization process. Consequently, to ease the training process of MAM-based layers, I 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 (2.5) and (2.6) can be affinely combined as

$$z_i = \beta \sum_{j=1}^{N} v_{ij} + (1-\beta) \underset{j=1}{\overset{N}{\mathcal{M}}} v_{ij} + b_i \quad \text{with } i \in \{1,..,M\} \tag{2.8}$$

During training, by using (2.8) 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 $\beta = 0$, equation (2.8) is reduced to (2.6) and β is kept fixed for the rest of the training iterations, keeping only the contributions of MAM. I call this the *vanishing contributions method*. The usage of this technique allows to obtain a much better performance of the resulting DNN when compared to the previous embryonic work in [117], where the paradigm based on max and min is trained as-it-is. It is important to highlight that (2.8) is used only during the training phase, while pruning of the layers and test inferences are performed strictly with (2.6) (i.e., $\beta = 0$). Moreover, this technique allows us to use pre-trained 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.

2.3 Pruning techniques classification

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

[3]Many non-linear 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.

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 [56, 25, 55, 110, 106] 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 [24, 5] 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 [145, 61, 153].

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* [51], i.e., among the interconnections of the same DNN layer, or *globally* [41], i.e., among the interconnections of the whole DNN model. As a second criterion, a *regularization* term [40, 122] 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 [154, 23] is performed in one step after the model has been trained and no further fine-tuning is performed afterwards. This results in very fast and computationally efficient pruning processes, even though it requires waiving deeper optimizations. On the other hand, iterative pruning [41, 134, 163] 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 [152], neuroevolution methods that take inspiration from biological mechanisms [151] and the global capture of

correlations among layers with second-order structured pruning [107]. An alternative to pruning to design small neural models starting from large ones is instead represented by Knowledge Distillation [133].

While virtually any of the aforementioned classes of pruning methods could be applied to MAM-based DNNs, in this work I focus on non-structured, score-based pruning techniques. This is mainly because, as we will see, *i)* the MAM paradigm naturally promotes a non-structured sparsity of the DNN layers and *ii)* 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, I test MAM-based structures both with local and global scores and with one-shot and iterative pruning techniques.

2.3.1 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 this work, I evaluate these architectures using unstructured, score-based pruning techniques. Among the many available approaches, I follow the evaluation criteria recommended in [13] for selecting pruning scores; these criteria are summarized below. In addition, I introduce a new criterion that exploits specific properties of the operator $\mathcal{M}$, with the aim of providing deeper insight into the behavior and characteristics of MAM-based models.

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 [13].

Magnitude-based scores

Starting from the assumption that the smaller the magnitude of a weight is, the less it potentially influences the output of the DNN, one of the simplest ways to define the scores for an 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.

Probability of Selection Pruning (PSP)

I 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 V, as Fig. 2.2 shows and as in (2.6). In this way, when inferring a single input, I know that most of the interconnections – each associated with a value in V and, by extension, to a weight in 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, I 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]For any pruning method requiring a pruning dataset, I use the validation set and I evaluate the final result with the test set (so I guarantee the generalization capabilities of the approaches)

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 $C(x, \text{DNN})$ is defined and evaluated on the model DNN with input x. Each interconnection ι is scored as

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

Here, the operator $\mathbf{E}[\cdot]$ evaluates the expected value on all the inputs 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 I evaluate the final results). This method is especially powerful in the context of MAM neurons and it implicitly reproduces the behavior of PSP. Specifically, in MAM, any unselected interconnection has a gradient of zero, which in turn yields a pruning score of zero, mirroring the effect obtained with PSP. 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.

2.4 A first insight into MAM properties

As a first example of how MAM-based layers can be used to obtain aggressively pruned networks, I employ a simple DNN composed of 3 FC layers trained on the MNIST dataset [81]. The two hidden layers contain 256 neurons each with a ReLU activation function. Fig. 2.3 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.

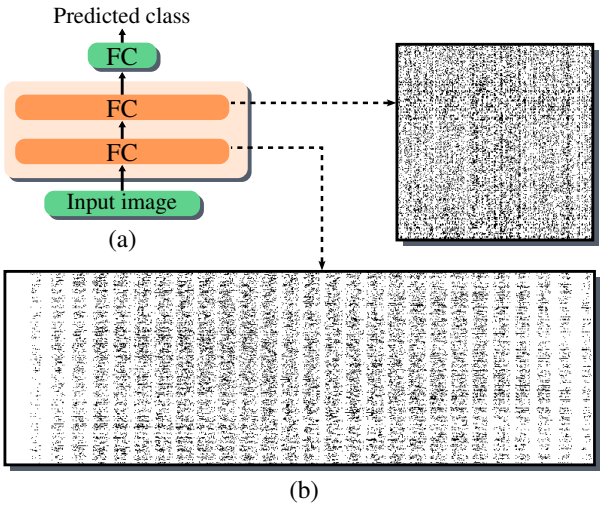

Fig. 2.3 Structure of the 3-layer fully connected network used for MNIST classification (FC-DNN) (a). MAM is applied to the highlighted FC layers. The entire network contains 269 322 trainable parameters. Subfigure (b) shows the probability-of-selection maps for the matrices V of the two hidden layers. In these black–white visualizations, the horizontal axis indexes the input neurons to the layer and the vertical axis indexes the output neurons. Each black dot corresponds to an interconnection with non-zero selection probability, amounting to 15.7% of all possible connections in the first layer and 19.0% in the second. Figure adapted from [115].

For any model in this work, the output layer is always MAC-based, which is needed to keep a good accuracy in the model. In fact, I found in practice that I can not 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. I 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.

I first study the behavior of the MAM layers by using the PSP scoring approach. Then, to prune the two structures for comparison, I employ GMP, LMP, GGP and LGP scoring approaches as discussed in Section 2.3.1.

2.4.1 Training setup and behaviour 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 [81]. The trainable parameters are updated using the Adam optimizer [73]. 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.

Training MAM layers requires starting with standard MAC-based layers and then transitioning to MAM following (2.8), defined by the vanishing contributions method of Section 2.2.2. More in detail, I define for each training epoch q a value $\beta[q]$ to be substituted to β in (2.8). I start with $\beta[1] = 1$ at epoch 1 and I decrease it linearly to $\beta[Q] = 0$ at the Q-th epoch. After this, $\beta[q]$ is kept to 0 for $q > Q$. Fig. 2.4 shows the accuracy trend during training with different values of Q.

When $\beta[q] > 0$ in the first Q epochs, the MAC contribution from (2.8) 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, I set $Q = 30$.

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

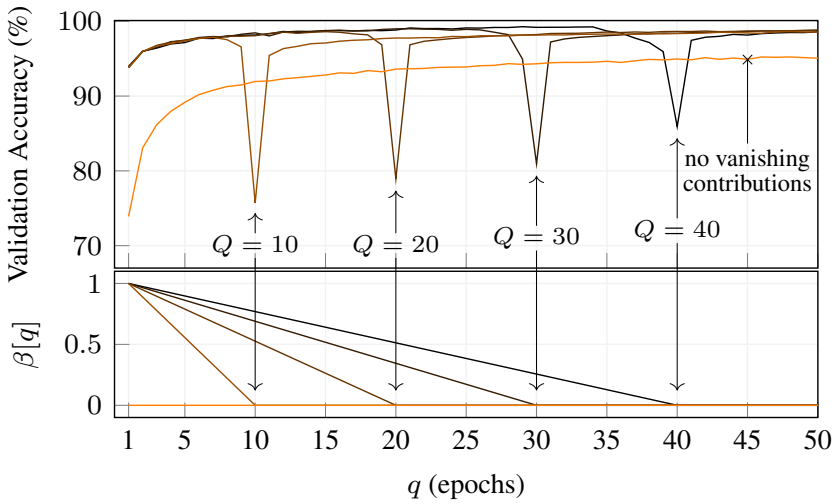

Fig. 2.4 Top: evolution of the Top-1 validation accuracy during training of the FC-DNN-MAM on MNIST for different values of Q. Bottom: corresponding trends of $\beta[q]$. For each Q, the accuracy exhibits a drop when $\beta[Q] = 0$, after which the network rapidly recovers. When Q is sufficiently large, the final accuracy becomes insensitive to the choice of Q. The orange curves depict the reference case in which the vanishing contributions approach is not used (i.e., $\beta[q]$ always zero), included to highlight the effect of vanishing contributions in the other curves. Figure adapted from [115].

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, I 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, I need to look at the performance of the two models when they are pruned.

2.4.2 The natural sparsity of MAM layers

In order to highlight the sparsification properties of MAM, I apply the PSP scoring approach from Section 2.3.1 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 V is evaluated for each layer (I still evaluate the final accuracy on a test dataset, different from the validation/pruning dataset). In Fig. 2.3, I highlight the positions of the values in V with non-zero probability

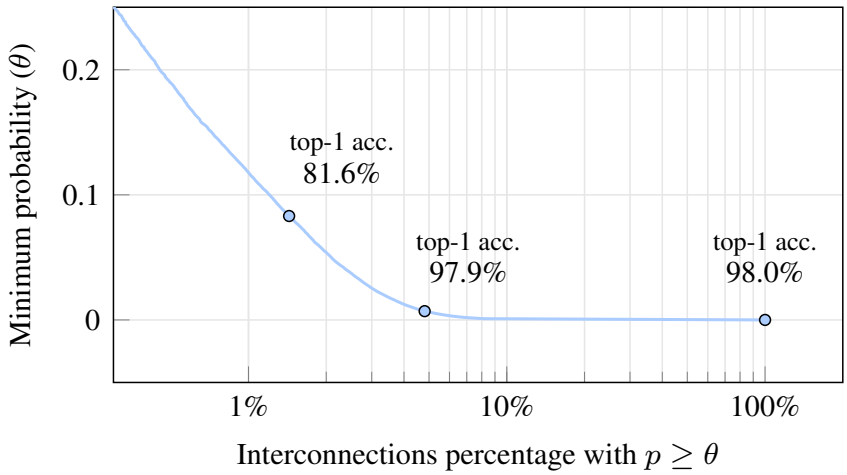

Fig. 2.5 FC-DNN-MAM: for different threshold values of θ, the plot shows how many interconnections (in percentage) have a probability of selection $p \geq \theta$. Also, top-1 accuracy is shown when keeping only the aforementioned interconnections. For $\theta = 0$, any interconnection is kept. For $\theta > 0$, 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. Figure from [115].

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 – and because of this, prunable – structure. 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. 2.5 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, I 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, I also highlight top-1 accuracy at different working points, showing that the performance drops only when I 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 2.3.1. Table 2.1 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%.

Table 2.1 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

For the sake of completeness, I 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 2 FLOPs per weight for MAC (multiply and addition) and 3 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.1.

To better highlight the behaviour of MAM, Fig. 2.6 shows the variation of top-1 accuracy with a decreasing number of 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.

2.5 Pruning performance of MAM layers

In this section, I expand the discussion on MAM-based DNN layers and I 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.

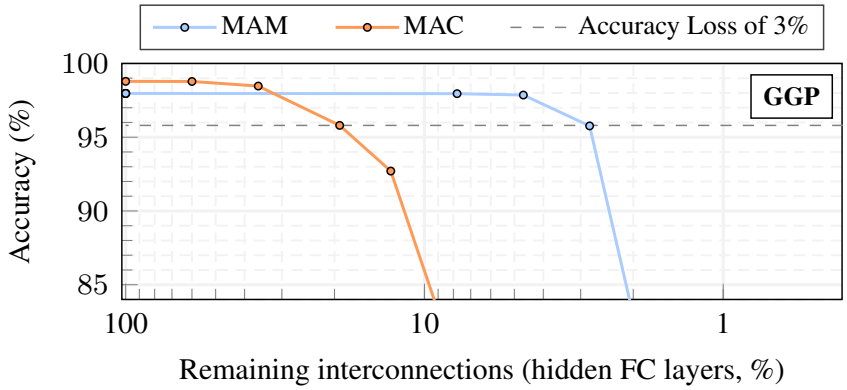

Fig. 2.6 FC-DNN + MNIST + one-shot pruning: top-1 accuracy vs percentage of kept interconnections in the two hidden FC layers when pruned with GGP. The dashed line indicates when the accuracy is 95.8% (i.e., 3% Accuracy loss). Interconnections are increasingly removed going from left to right.

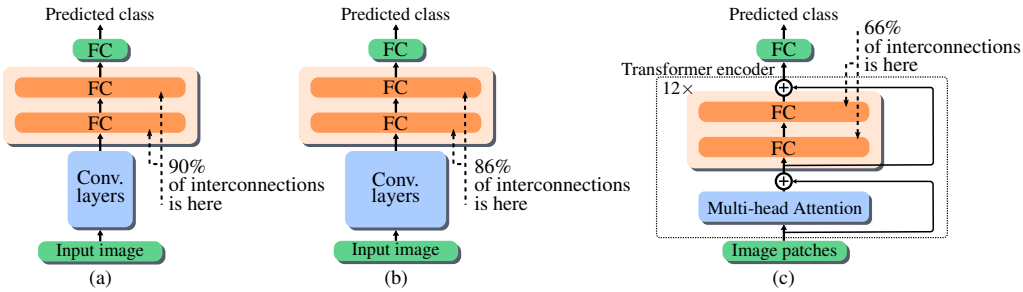

Fig. 2.7 The DNN models tested in this work, namely AlexNet (a), VGG-16 (b) and ViT-B/16 (c). The highlighted FC layers, that contain most of the interconnections in the networks, are the ones in which I employ MAM and that I ultimately prune. In (c), I prune the FC layers of 10 out of 12 sequential blocks. Figure from [115].

2.5.1 Computer vision benchmark datasets

I perform benchmarks on three different computer vision datasets, namely CIFAR-10, CIFAR-100 [77] and ImageNet-1K [33]:

- **CIFAR-10**: This task consists in the classification of animals and vehicles among 10 different classes. The dataset consists of a total of 60 000 colour 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 in random image rotation, shifting and change of 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.

- **CIFAR-100**: This task consists in the classification of items belonging to 100 different classes. It consists of a total of 60 000 colour 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 in and all samples are resized using a bilinear interpolation to be of dimension 224×224.

- **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 change in contrast and brightness. All the images are resized to 224×224 pixels using a bilinear interpolation.

2.5.2 Deep neural models and training

Here I present the computer vision models used for the tests. They are represented in Fig. 2.7. As stated in Section 2.2, to simplify the analysis I do not discuss the pruning of convolutional layers and I focus on the portions of

[6]As the labels of the official ImageNet test set are not disclosed, I 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.

DNN that contain most of the interconnections, namely, the FC layers. The models are:

- **AlexNet**: This is a rather outdated convolutional neural network used for computer vision tasks [78]. It is composed of 5 convolutional layers and 2 FC hidden layers, followed by an output FC layer. It uses 47.2 million interconnections. The 2 FC hidden layers contribute to 90% of the interconnections in the model and are the ones that I substitute with MAM layers. I refer to the original MAC-based model as AlexNet-FCMAC and to the model containing MAM FC layers as AlexNet-FCMAM. I perform a full train of AlexNet + CIFAR-10 and AlexNet + CIFAR-100 for 70 epochs with a starting learning rate of 0.001 and batch size of 256. Dropout after the two hidden FC layers is set to 0.5 for AlexNet-FCMAC as indicated in [78] and to 0.2 for AlexNet-FCMAM, with $Q = 12$ for the vanishing contributions method.

- **VGG-16**: This is a large convolutional neural network with much greater approximation capabilities compared to AlexNet [129]. It is composed of 10 convolutional layers and 2 FC hidden layers, followed by an output FC layer. It uses 138.4 million interconnections. As for AlexNet, the 2 FC layers contribute to most of the interconnections in the model (86%) and are the ones that I substitute with MAM layers. I 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 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. I set the learning rate to 0.001, the momentum coefficient to 0.9 and 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, I employ the vanishing contributions method with $Q = 5$.

- **ViT-B/16**: This is a modern, transformer-based DNN which leverages the self-attention mechanism on computer vision tasks [35, 89]. It is composed

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

Table 2.2 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%

of 12 sequential transformer encoders. Each transformer encoder contains a multi-head attention block followed by 2 hidden FC layers (2 skip connections are also present, as shown in Fig. 2.7). An FC output layer follows the 12 encoders. As for AlexNet and VGG-16, I substitute MAM in each of the 2 hidden FC layers highlighted in Fig. 2.7 with the exception of the first two.[8] I 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 [33]. I fine-tune the model for 50 epochs and I use the Adam optimizer with a starting 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, I use a batch size of 128 and mix-up data-augmentation technique [150] with strength (α parameter) 1.0. In the case of the MAM-based model, the initial 12 epochs are used to complete the vanishing contributions transition.

2.5.3 Pruning performance

The models are trained with both MAM FC layers and MAC FC layers and Table 2.2 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.

[8]The insertion of MAM in the first two blocks results in a significant reduction of the overall accuracy. Overall, ViT-B/16 uses approximately 86 million interconnections, of which 56% is in the hidden FC layers that I substitute with MAM.

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

Table 2.3 Percentage of remaining weights and flops in the pruned fully connected 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).

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

Table 2.4 Percentage of remaining weights and flops in the pruned fully connected 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).

		MAC		MAM	
		Weights kept	MFLOPs	Weights kept	MFLOPs
CIFAR-100	GMP	8.71%	20.83	0.012%	0.06
	LMP	3.80%	9.09	0.01%	0.05
	GGP	0.30%	0.73	**0.007%**	**0.04**
	LGP	4.00%	9.57	0.013%	0.06
ImageNet	GMP	10.82%	25.88	**0.04%**	**0.16**
	LMP	10.36%	24.78	0.07%	0.27
	GGP	35.93%	85.91	**0.04%**	**0.16**
	LGP	42.12%	100.71	0.09%	0.34

Table 2.5 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).

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

After training, for each model under test, I prune the selected hidden FC layers to compare the performance of MAM layers vs standard MAC. I first test the models with RP scoring approach as suggested in [13] and I confirm that randomly pruning the interconnections gives bad results, i.e., the performance drops after a few interconnection removals. Then I proceed to evaluate the performance with GMP, LMP, GGP and LGP scoring approaches. I look at the top-1 accuracy of Table 2.2 for the non-pruned MAC models and I remove interconnections until I 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. Table 2.3, Table 2.4 and Table 2.5 show the results for AlexNet, VGG-16 and ViT-B/16, respectively. Additionally, as a relevant example, Fig. 2.8 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 I 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 2.4.2. 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

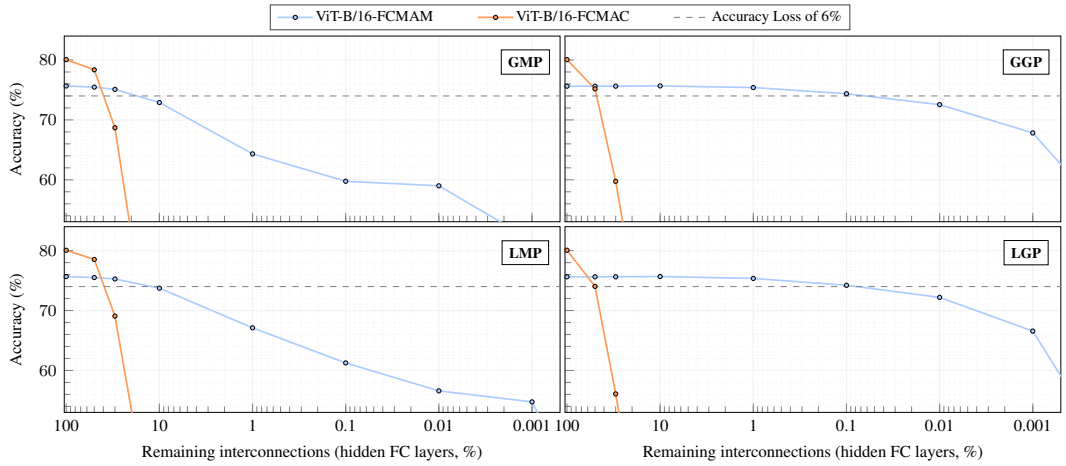

Fig. 2.8 ViT-B/16 + ImageNet-1K + one-shot pruning: top-1 accuracy vs percentage of kept interconnections in the hidden FC layers (of 10 out of 12 transformer blocks) when pruned with GMP, LMP, GGP and LGP. The dashed line indicates when the accuracy is 78.15% (i.e., 3% Accuracy loss). Interconnections are increasingly removed going from left to right. Figure from [115].

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.

Table 2.3, Table 2.4 and Table 2.5 also show the computational complexity in terms of FLOPs, where I count 2 FLOPs per weight for MAC and 3 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.1.

Of course, I 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 based on FC layers, so in this case we globally manage to reduce the computational complexity by more than 2 times.

2.6 MAM on training-dependent pruning methods

I 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 I want to achieve. However, it is possible to apply to MAM neurons other more complex training-dependent pruning approaches. Here I present two examples: I test MAM in conjunction with the AC/DC method [111] and the *lottery ticket* iterative pruning approach [41]. I 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, I report some examples from [35]. Authors state that, using a standard cloud TPUv3 with 8 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 [132] 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 [129]. 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 I consider more appropriate the use of post-training, one-shot pruning techniques.

2.6.1 Performance with AC/DC pruning

The Alternating Compressed / DeCompressed Training (AC/DC) [111] 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 alternating repeatedly 2 different phases, namely compression and decompression. During compression, the selected percentage of interconnections

Table 2.6 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**

is temporarily removed (with GMP) and the pruned DNN is trained. During decompression, the removed interconnections are restored and training continues updating all the interconnections. 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, I morph the MAC neurons into MAM neurons during the last compression phase (with the vanishing contributions technique).

I 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. I 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 2.6, even with this approach the MAM-based model can be compressed more than the MAC-based one.

2.6.2 Performance with lottery ticket iterative pruning

As second training-dependent pruning approach I consider the well-known Lottery Tickets hypothesis in [41] tailored on the non-structured pruning of FC layers. This hypothesis suggests that a randomly initialized DNN contains

a sub-network (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 k-th 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.

I 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. 2.9 shows the final accuracy I obtain for the pruned DNN given a certain number of iterations. While MAC requires at least 10 iterations (i.e., 10 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 (6 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.

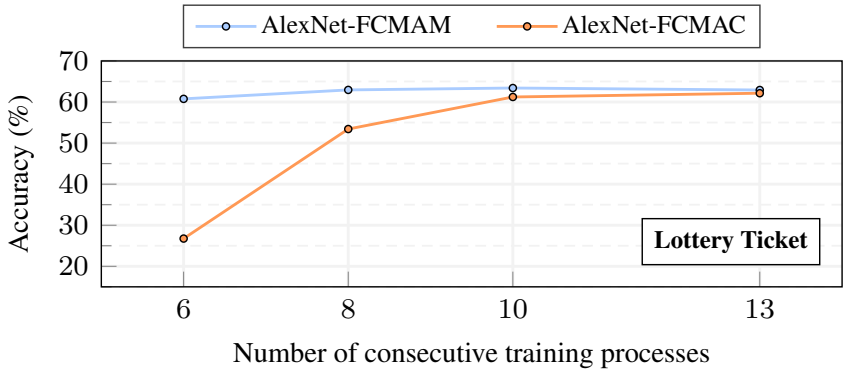

Fig. 2.9 AlexNet + CIFAR-100 + Lottery Ticket pruning: top-1 accuracy with 0.02% remaining weights in the hidden FC layers vs number of consecutive training processes. AlexNet-FCMAM can reach good accuracy with less iterations compared to AlexNet-FCMAC. Figure from [115].

2.7 Conclusion

In this first chapter I have presented and elaborated on a novel MAM map-reduce paradigm which allows to build neurons naturally prone to aggressive pruning that 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 of 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.

Chapter 3

Back to Basics: The Universal Approximation Theorem

In the previous chapter I showed empirically 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 almost the same performance as the original MAC-only network. However, 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.

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, I 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 chapter is structured as follows. In Section 3.1, I list related works that demonstrate the approximation capabilities of several

neural network models. In Section 3.2, I present a high-level description of the mathematical model that I use to derive the two theorems on the approximation capabilities of DNNs using MAM neurons. The two theorems are formally stated in Section 3.3. An example of a function approximation and quantitative results regarding the convergence of the approximation error are reported in Section 3.4. The proof of these theorems can be found in Section 3.5. Finally, conclusion is drawn.

3.1 Related works

The development of models with universal approximation capabilities has been a significant breakthrough in many fields of science and engineering. In 1989, [28] proved that a network with a single hidden layer could approximate any continuous function, given enough hidden neurons. Some years later, [141] and [75] 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 [126]. 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 [59, 112], bounded width and arbitrary depth [46, 52] and bounded width and depth [47]. In the recent work [18], 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 [158], dropout neural networks [97], networks representing probability distributions [94], graph neural networks with random weights [66] and spiking neural networks [155]. In addition, in [144] 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 [4, 135, 149]. 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 here since, with MAM neurons, maximum and minimum operations provide the aggregating part of the computation that does not involve any linear combination, while GroupSort is an activation function that acts on a set of linear combinations of the layer inputs. Another recent alternative to Multi-Layer Perceptrons is offered by Kolmogorov-Arnold Networks (KANs) [91], where weight parameters are replaced by a univariate function parameterized using a spline, but their underlying structure is fundamentally different from the one of layers based on the MAM paradigm.

3.2 General model and proof strategy

To demonstrate the approximation capabilities of a hybrid MAM&MAC structure, I build and program two network architectures and prove that their approximation error can approach zero by increasing their complexity. In this section, I present the general idea behind this construction, while the detailed description and the proofs of the theorems can be found in Section 3.5.

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, I use the well-known ReLU activation. If $v_1, v_2, \ldots$ are the inputs and $w_1, w_2, \ldots$ are the weights, the MAM neuron can be described as follows

$$u = \left[\max_j w_j v_j + \min_j w_j v_j + b \right]^+ \tag{3.1}$$

where b is the bias and $[\cdot]^+ = \max\{0, \cdot\}$ represents the nowadays common ReLU nonlinearity.

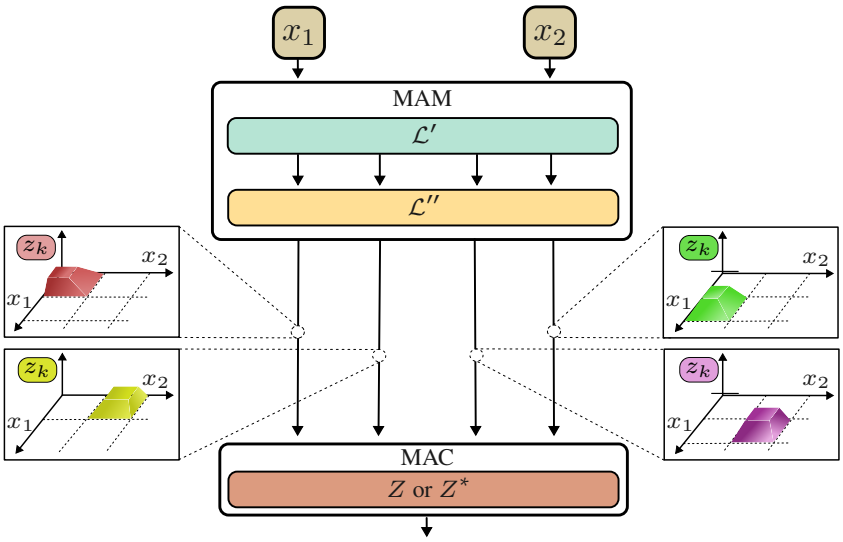

Fig. 3.1 Structure of the overall neural network when $N = 2$. The composition of the two MAM layers $\mathcal{L}''(\mathcal{L}'(\boldsymbol{x}))$ is used to generate the building blocks z_k that the final MAC layer Z^* or Z properly combines to create the output of the network. Figure from [10].

I 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 I 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 Figure 3.1 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 I 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 I collect in $\mathbb{Y} \subset \mathbb{X} = [0,1]^N$ the set of points for which this fails, I 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,

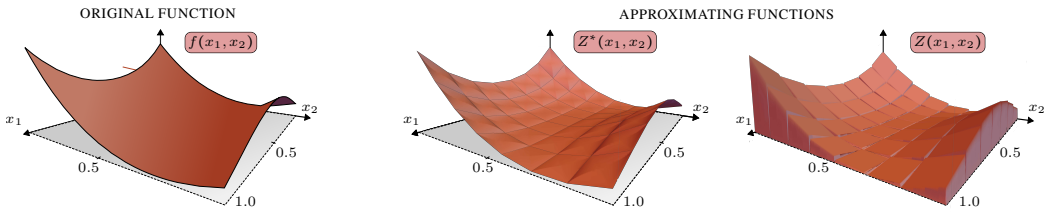

Fig. 3.2 Three dimensional plot of the target function $f(x_1,x_2)$ and of its two approximations $Z^*(x_1,x_2) \in \mathcal{Z}^*$ implied by Theorem 1 and $Z(x_1,x_2) \in \mathcal{Z}$ by Theorem 2 with $N=2$ and $n=7$. Figure from [10].

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

$$Z^*(\boldsymbol{x}) = \frac{\sum_k c_k z_k(\boldsymbol{x})}{\sum_k z_k(\boldsymbol{x})}. \tag{3.2}$$

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, I additionally consider a final layer with a simple linear combination

$$Z(\boldsymbol{x}) = \sum_k c_k z_k(\boldsymbol{x}). \tag{3.3}$$

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.

3.3 Main results

Within the above framework, I 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 (3.2) 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 (3.3) and to show that the first-order differential behavior can also be reproduced. In both cases, I 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 (3.2) while with $\mathcal{Z}$ the analogous family of functions in (3.3). 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 d-th 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*

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

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

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

If #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 3.5 and are constructive. In particular, subnetworks in the cascade $\boldsymbol{z}(\boldsymbol{x}) = \mathcal{L}''\left(\mathcal{L}'(\boldsymbol{x})\right)$ 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 (3.2) or (3.3) is capable of arbitrarily approximating the target function.

3.3.1 Limitations of the current approach

Theorem 1 and Theorem 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 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.

3.4 Examples

The purpose of this section is to visually represent the approximation capabilities of a neural network employing MAM neurons, as delineated in Theorem 1 and Theorem 2. As a case study, I use the $\mathcal{C}^\infty$ Rosenbrock function [125], a classical benchmark in approximation studies due to its smooth but nonlinear structure. Centered at $(0.5, \ldots, 0.5) \in \mathbb{R}^N$, it is defined as follows

$$f(\boldsymbol{x}) = \sum_{i=1}^{N-1} \left[100\left(x_{i+1} - x_i^2 + 2x_i - 0.75\right)^2 + (1.5 - x_i)^2\right] \tag{3.4}$$

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

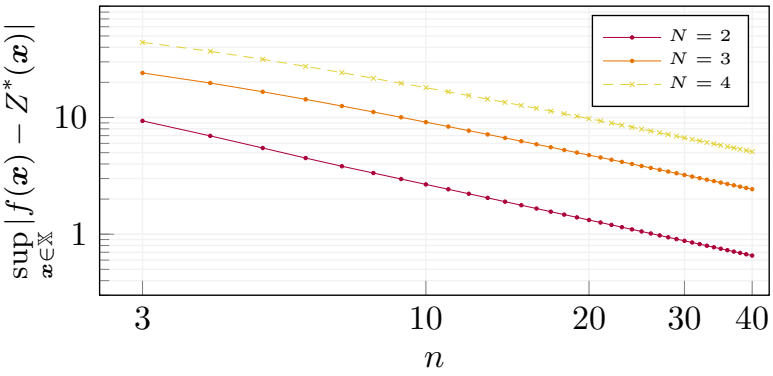

Fig. 3.3 Convergence of the approximation error for the Rosenbrock function for Theorem 1 with a varying number of input dimensions N and of divisions of the domain n. Figure from [10].

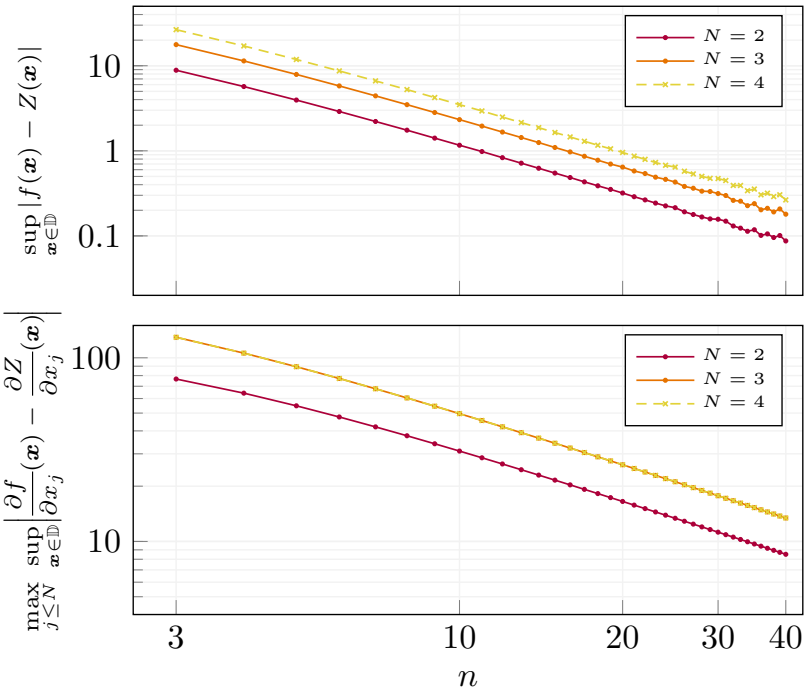

Fig. 3.4 Convergence of the approximation error for the Rosenbrock function and the maximum approximation error of its partial derivatives for Theorem 2 varying the number of input dimensions N and of divisions of the domain n. Plots with $N \geq 3$ are superimposed due to the nature of the target function. Figure from [10].

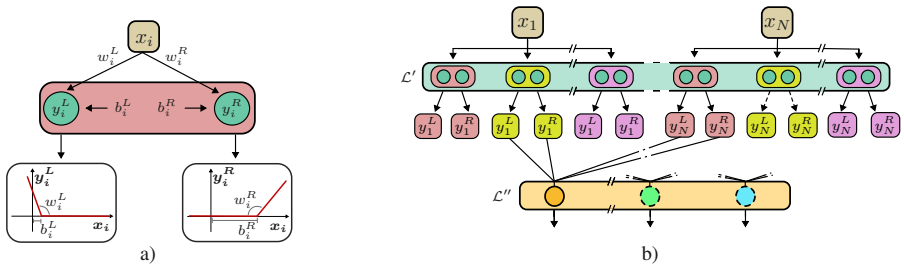

Fig. 3.5 In a) the representation of a couple of MAM neurons with non-null weights for input x_i while in b) the structure of the two MAM layers $\mathcal{L}'$ and $\mathcal{L}''$. Figure from [10].

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 3.5. 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 $n = 7$, Figure 3.2 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 Figure 3.3 and in Figure 3.4 I provide quantitative results illustrating the decrease in the approximation error for both theorems. Pointwise 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.

3.5 Network construction and proofs of Theorems

3.5.1 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 Figure 3.5. It contains neurons in which only one weight is non-null so that only one input affects the output. This simplifies (3.1) into

$$u = [wv + b]^+ \tag{3.5}$$

where v is the only input connected with the neuron with $w \neq 0$. The profile of (3.5) 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 Figure 3.5a, 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 $w^{\text{L}} < 0$, $w^{\text{R}} > 0$ and $0 \leq b^{\text{L}} \leq b^{\text{R}} \leq 1$ I obtain the left y^{L} and right y^{R} parts of a trapezoidal profile depending on the common input.

Looking at Figure 3.5b, 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 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 $\omega_1, \ldots, \omega_N \in [0,1]$, $l_1, \ldots, l_N \geq 0$, $\delta_1^{L}, \ldots, \delta_N^{L} \geq 0$, and $\delta_1^{R}, \ldots, \delta_N^{R} \geq 0$, the two MAM hidden layers can be programmed to yield*

$$z(\boldsymbol{x}) = [1 - \Delta(\boldsymbol{x})]^+ \tag{3.6}$$

where

$$\Delta(\boldsymbol{x}) = \max_{i \in \{1,..,N\}} \left\{ 0, \frac{|x_i - \omega_i| - l_i}{\begin{cases} \delta_i^{L} & \text{if } x_i < \omega_i \\ \delta_i^{R} & \text{if } x_i \geq \omega_i \end{cases}} \right\}. \tag{3.7}$$

Proof of Lemma 1. I 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 $y_1^{\text{L}}, y_1^{\text{R}}, y_2^{\text{L}}, y_2^{\text{R}}, \ldots$, 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 $w^{\text{L}} = -1/\delta_i^{\text{L}}$ and the bias $b^{\text{L}} = (\omega_i - l_i)/\delta_i^{\text{L}}$, while for y_i^{R} the non-null input weight $w^{\text{R}} = 1/\delta_i^{\text{R}}$ and bias $b^{\text{R}} = (-\omega_i - l_i)/\delta_i^{\text{R}}$. By recalling (3.5) one gets

$$y_i^{\text{L}} = \left[\frac{-x_i + \omega_i - l_i}{\delta_i^{\text{L}}}\right]^+ \quad \text{and} \quad y_i^{\text{R}} = \left[\frac{x_i - \omega_i - l_i}{\delta_i^{\text{R}}}\right]^+. \tag{3.8}$$

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

$$\begin{aligned} z &= \left[\max_{i\in\{1,\dots,N\}} \{0, -y_i^{\text{L}}, -y_i^{\text{R}}\} + \min_{i\in\{1,..,N\}} \{0, -y_i^{\text{L}}, -y_i^{\text{R}}\} + 1\right]^+ \\ &= \left[1 - \max_{i\in\{1,..,N\}} \{y_i^{\text{L}}, y_i^{\text{R}}\}\right]^+. \end{aligned} \tag{3.9}$$

Note now that, if $x_i \geq \omega_i$ then $y_i^{\text{R}} \geq 0$ and $y_i^{\text{L}} = 0$ while, if $x_i < \omega_i$ then $y_i^{\text{R}} = 0$ and $y_i^{\text{L}} \geq 0$. Hence, without loss of generality, we may assume that $x_i \geq \omega_i$ for $i = 1, \dots, N$, being all other cases a variation of this one by suitable symmetry and scaling. With this, $y_i^{\text{L}} = 0$ for $i = 1, \dots, N$ and (3.9) becomes

$$\begin{aligned} z &= \left[1 - \max_{i=1,\dots,N} \left[\frac{x_i - \omega_i - l_i}{\delta_i^{\text{R}}}\right]^+\right]^+ \\ &= \left[1 - \max_{i=1,\dots,N} \left\{0, \frac{x_i - \omega_i - l_i}{\delta_i^{\text{R}}}\right\}\right]^+ \end{aligned} \tag{3.10}$$

that is equivalent to the thesis. □

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 $\boldsymbol{\omega} = (\omega_1, \dots, \omega_N)$ with sides $2l_1, \dots, 2l_N$. 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 $\boldsymbol{\omega}$. Figure 3.6 reports an example of a $z(\boldsymbol{x})$ when $N = 2$.

In the following, I will assume that each neuron in the second hidden layer matches a whole subnetwork as implied by Lemma 1. With this, I 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.

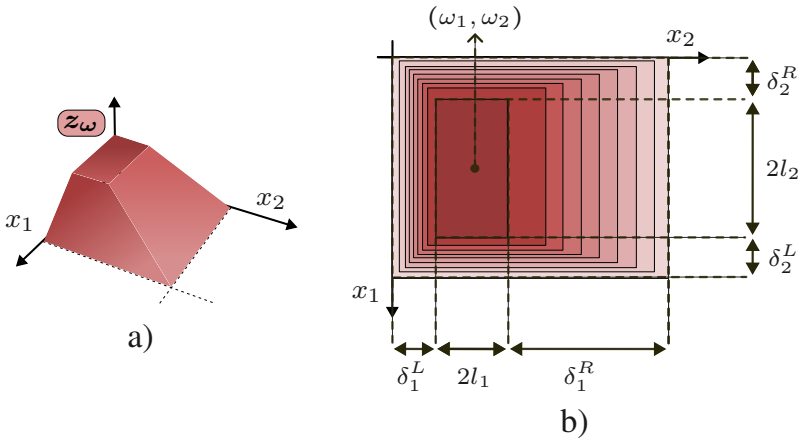

Fig. 3.6 In a) the three dimensional representation of a generic $z_{\boldsymbol{\omega}}(\boldsymbol{x})$ for $N=2$ and its contour plot in b) showing the role of the various parameters. Figure from [10].

3.5.2 Universal approximation properties with normalized linear output neuron

Given a positive integer n, define $\Omega = \left\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\right\}^N$ 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

$$\begin{aligned} \#\mathrm{n} &= 2nN + (n+1)^N + 1 \sim n^N && (3.11) \\ \#\mathrm{p} &= 4nN + (2N+1)(n+1)^N + (n+1)^N \\ &\sim 2(N+1)n^N && (3.12) \end{aligned}$$

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 $i = 1, \ldots, N$ 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

$$\zeta_{\boldsymbol{\omega}}(x) = \frac{z_{\boldsymbol{\omega}}(\boldsymbol{x})}{\sum_{\boldsymbol{\omega}' \in \Omega} z_{\boldsymbol{\omega}'}(\boldsymbol{x})} \tag{3.13}$$

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

$$\begin{aligned} &\left| f(\boldsymbol{x}) - \sum_{\boldsymbol{\omega} \in \Omega_{\boldsymbol{x}}} f(\boldsymbol{\omega}) \zeta_{\boldsymbol{\omega}}(\boldsymbol{x}) \right| \\ &= \left| \sum_{\boldsymbol{\omega} \in \Omega_{\boldsymbol{x}}} [f(\boldsymbol{x}) - f(\boldsymbol{\omega})] \zeta_{\boldsymbol{\omega}}(\boldsymbol{x}) \right| \\ &\leq \max_{\boldsymbol{x} \in \mathbb{X}} \max_{\substack{\boldsymbol{\xi} \in C_{\boldsymbol{x}} \\ \boldsymbol{\omega} \in \Omega_x}} |f(\boldsymbol{\xi}) - f(\boldsymbol{\omega})| . \end{aligned} \tag{3.14}$$

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 $\delta = 1/n$ I 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 I 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 (3.12) to say that the number of parameters needed to meet the given tolerance should grow as ϵ^{-N} as the tolerance decreases. □

3.5.3 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 $\Omega = \left\{\frac{1}{2n}, \frac{3}{2n}, \frac{5}{2n}, \ldots, \frac{2n-1}{2n}\right\}^N$, and set

$$\ell = \frac{1}{2}\left(\sqrt{\frac{n+2}{n}} - 1\right) \tag{3.15}$$

as well as $\delta = \ell^2$ so that $2(\ell+\delta) = 1/n$.

For any $\boldsymbol{\omega} \in \Omega$ I include subnetwork neurons of the second hidden layer with outputs labeled $z_{\boldsymbol{\omega}^{1-}}, z_{\boldsymbol{\omega}^{1+}}, \ldots, z_{\boldsymbol{\omega}^{N-}}, z_{\boldsymbol{\omega}^{N+}}$ 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 $\boldsymbol{\omega}^{j\pm} = (\omega_1^{j\pm}, \ldots, \omega_N^{j\pm})$, by the slopes $\delta_1^{\mathrm{L},j\pm}, \ldots, \delta_N^{\mathrm{L},j\pm}$ and $\delta_1^{\mathrm{R},j\pm}, \ldots, \delta_N^{\mathrm{R},j\pm}$, as well as by the side lengths $l_1^{j\pm}, \ldots, l_N^{j\pm}$.

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 $i, j = 1, \ldots, N$.

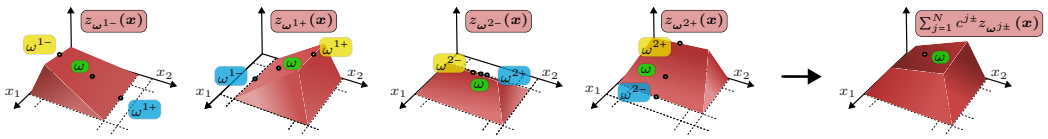

Fig. 3.7 Three dimensional plots of the functions $z_{\boldsymbol{\omega}^{1-}}, z_{\boldsymbol{\omega}^{1+}}, z_{\boldsymbol{\omega}^{2-}}, z_{\boldsymbol{\omega}^{2+}}$ with $N=2$ that are combined to build $\sum_{j=1}^{N} c^{j\pm} z_{\boldsymbol{\omega}^{j\pm}}(\boldsymbol{x})$. Figure from [10].

To give some intuitive grounding to the above definitions, Figure 3.7 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

$$\begin{aligned} \#\text{n} &= 2nN + 2Nn^N + 1 \sim 2Nn^N && (3.16) \\ \#\text{p} &= 4nN + (2N+1)\,2Nn^N + 2Nn^N \\ &\sim 4N(N+1)\,n^N && (3.17) \end{aligned}$$

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

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

$$X_{\boldsymbol{\omega}}^{\blacksquare} = \left\{ \boldsymbol{x} \in \mathbb{X} \;\middle|\; \max_{i=1,\ldots,N} \{|x_i - \omega_i|\} \leq \ell \right\}. \tag{3.18}$$

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 (3.18) 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 I 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 I have exploited (3.15). □

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*

$$Z_{\boldsymbol{\omega}}(\boldsymbol{x}) \equiv \sum_{j=1}^{N} c^{j\pm} z_{\boldsymbol{\omega}^{j\pm}}(\boldsymbol{x}) = a + \sum_{j=1}^{N} b_j x_j \tag{3.19}$$

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

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

Hence, if $\boldsymbol{x}\in X_{\boldsymbol{\omega}}^{\blacksquare}$ I know that $\omega_j^{j-}\leq x_j\leq\omega_j^{j+}$ for $j=1,\dots,N$. Moreover, since by definition for any $i,j=1,\dots,N$ and $i\neq j$ I have $\omega_i^{j+}-\omega_i^{j-}=2\ell$ and $\omega_i^{j-}+\omega_i^{j+}=2\omega_i$, then $\left|x_i-\omega_i^{j\pm}\right|\leq\ell$ when $i\neq j$. Therefore, one can apply Lemma 1 and compute $\Delta(\boldsymbol{x})$, for which all the terms in (3.7) but $\left|x_j-\omega_j^{j\pm}\right|$ are non-positive, thus yielding $z_{\omega^{j\pm}}(\boldsymbol{x}) = 1-\left|x_j-\omega_j^{j\pm}\right|/(2\ell)$.

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

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

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

$$c^{j-} = \frac{a}{N} \quad\text{and}\quad c^{j+} = c^{j-} + 2\ell b_j \tag{3.20}$$

□

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*

$$|f(\boldsymbol{x}) - Z_{\boldsymbol{\omega}}(\boldsymbol{x})| \leq MN^2\ell^2$$

$$\left|\frac{\partial f}{\partial x_j}(\boldsymbol{x}) - \frac{\partial Z_{\boldsymbol{\omega}}}{\partial x_j}(\boldsymbol{x})\right| \leq MN\ell$$

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

$$\left|\frac{\partial^2 f}{\partial x_i x_j}(\boldsymbol{x})\right| \leq H \tag{3.21}$$

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 $\boldsymbol{\omega}$

$$\begin{aligned} f(\boldsymbol{x}) = f(\boldsymbol{\omega}) + \sum_{i=1}^{N} \frac{\partial f}{\partial x_i}(\boldsymbol{\omega})(x_i - \omega_i) + \\ + \sum_{i=1}^{N}\sum_{j=1}^{N} R_{i,j}(\boldsymbol{x})(x_i - \omega_i)(x_j - \omega_j) \end{aligned} \tag{3.22}$$

$$\frac{\partial f}{\partial x_i}(\boldsymbol{x}) = \frac{\partial f}{\partial x_i}(\boldsymbol{\omega}) + \sum_{j=1}^{N} S_{i,j}(\boldsymbol{x})(x_j - \omega_j) \quad i = 1, \ldots, N \tag{3.23}$$

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

$$\begin{aligned} &\left|\sum_{i=1}^{N}\sum_{j=1}^{N} R_{i,j}(\boldsymbol{x})(x_i - \omega_i)(x_j - \omega_j)\right| = \\ &\leq N^2\ell^2 \frac{1}{2} \max_{k,l=1,\ldots,N} \max_{\boldsymbol{\xi} \in \mathbb{X}} \left|\frac{\partial^2 f}{\partial x_k x_l}(\boldsymbol{\xi})\right| = \\ &\leq \frac{1}{2} HN^2\ell^2 \end{aligned} \tag{3.24}$$

and

$$\begin{aligned}
&\left|\sum_{j=1}^{N} S_{i,j}(\boldsymbol{x})(x_j-\omega_j)\right| = \\
&\leq N^2\ell^2\frac{1}{2}\max_{j=1,\ldots,N}\max_{\boldsymbol{\xi}\in\mathbb{X}}\left|\frac{\partial^2 f}{\partial x_i x_j}(\boldsymbol{\xi})\right| = \\
&\leq \frac{1}{2}HN\ell \qquad i=1,\ldots,N.
\end{aligned} \tag{3.25}$$

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

$$\begin{aligned}
Z_{\boldsymbol{\omega}}(\boldsymbol{x}) &= f(\boldsymbol{\omega}) + \sum_{i=1}^{N}\frac{\partial f}{\partial x_i}(\boldsymbol{\omega})(x_i-\omega_i) \\
&= \left[f(\boldsymbol{\omega}) - \sum_{i=1}^{N}\frac{\partial f}{\partial x_i}(\boldsymbol{\omega})\omega_i\right] + \sum_{i=1}^{N}\frac{\partial f}{\partial x_i}(\boldsymbol{\omega})x_i
\end{aligned} \tag{3.26}$$

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 $X^{\blacksquare}_{\boldsymbol{\omega}}$, and the approximation errors can be derived exploiting (3.22) with (3.24) and (3.23) with (3.25) to obtain

$$|Z_{\boldsymbol{\omega}}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \frac{1}{2}HN^2\ell^2, \tag{3.27}$$

$$\left|\frac{\partial Z_{\boldsymbol{\omega}}}{\partial x_i}(\boldsymbol{x}) - \frac{\partial f}{\partial x_i}(\boldsymbol{x})\right| \leq \frac{1}{2}HN\ell \tag{3.28}$$

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

Proof of Theorem 2. Note first that from (3.15) I 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, I recall (3.17) to produce the second part of the thesis. □

3.6 Conclusion

I proved two theorems with different assumptions and theses that confirm that hybrid MAM&MAC networks are universal approximators. I 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.

Part II

Lighter

Chapter 4

Towards Lightweight AI: Leveraging Event-Based Cameras for Efficient Vision

In the second part of this thesis, I shift focus from reducing the number of network parameters to make models *Smaller* in size, to developing AI that is *Lighter* which means here faster and more low-power. I continue working within the field of computer vision, exploring how things change when we move from using traditional RGB cameras to event-based cameras, neuromorphic sensors that are gaining traction in various fields, including robotics.

4.1 Introduction

Event-based cameras [87, 42] are neuromorphic video recording devices that enable low-power and high temporal-resolution acquisition of visual information. They have gained an increasing popularity in a large variety of fields, including surveillance [123, 124], autonomous driving cars [85, 20, 137, 21] and robotics [14, 147, 37, 139]. The sensors used for these devices offer a fundamentally different way of representing visual information compared to traditional frame-based cameras, providing a sparse and asynchronous events stream where each event corresponds to a change in the scene brightness. More specifically, an event is represented by the coordinates of the pixel location where the

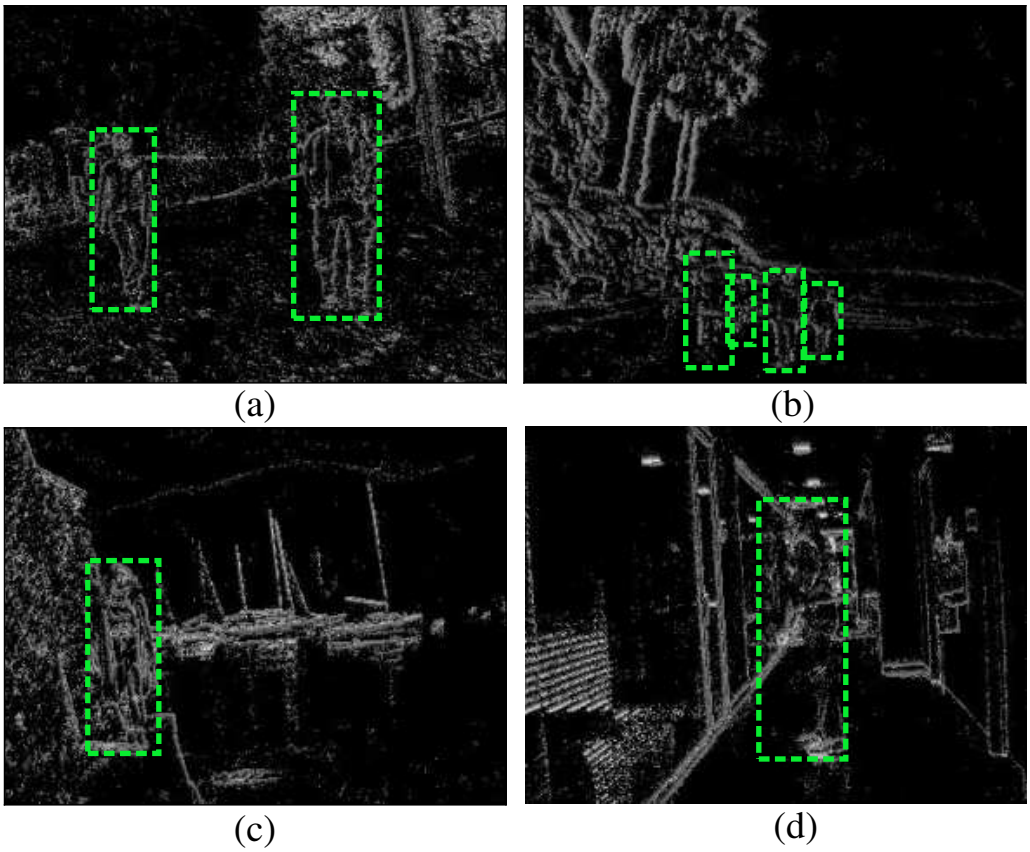

Fig. 4.1 Some of the 43 259 bounding boxes manually annotated contained in PEDRo. The dataset focuses on people and it presents recordings taken in a large variety of environments such as a) woods, b) beaches, c) lakes, d) indoor scenarios. Figure from [16].

illuminance change occurs, as well as by the polarity and the time at which the event has been detected. Event cameras typically provide a very high temporal resolution, with an acquisition time of the order of microseconds and a high dynamic range (up to 120 dB) thanks to the logarithmic characteristic of the pixel response. Moreover, their compact size and low power consumption make them well-suited for deployment in mobile and battery-powered robots. In particular, this kind of sensors has shown great potential for tasks that require fast and accurate motion detection and tracking in a large variety of scenarios with different lighting conditions. As an example, person detection is one of these, to help robots monitoring areas for intruders and navigating through crowded spaces.

The state-of-the-art approach to solve person detection problems is to adopt deep learning architectures, whose proficient training is made possible using large quantities of annotated data. However, the number of event-based datasets is notably smaller compared to what is currently available for frame-based cameras, mainly due to the still limited adoption of event sensors. While synthetic data can be used to tackle this problem, their use can lead to suboptimal performance because of the potential disparities between simulated and real events [43, 131].

In recent years, efforts have been made to create large event-based datasets specifically designed for object detection in the context of autonomous driving [31, 109]. These datasets are recorded using a sensor mounted on the dashboard of a car and data are acquired driving in real-world scenarios. To the best of my knowledge, these are the only datasets containing a large number of labeled people recorded using a moving camera. Because of this, the training of algorithms for person detection in robotics applications becomes a challenge. In fact, robots can move both indoors and outdoors and in a variety of environments which are not limited to roads or highways. Moreover, pedestrians are typically walking on the sidewalks far from the camera, while when dealing with robots navigation, people can be close and their figure may be partially out of frame.

To overcome this, I introduce in this work an event-based dataset for Person Detection in Robotics applications (PEDRo), made of 119 recordings of variable duration taken with a moving DAVIS346 camera. The dataset, containing 43 259 bounding boxes, was manually annotated and focuses on human figures in a large variety of environments, atmospheric and lighting conditions, as in Figure 4.1. To the best of my knowledge, this is also the largest manually annotated event-based dataset for person detection recorded with a moving camera.

4.2 Related Works

Event-based datasets Early examples of event-based datasets have been generated from already existing frame-based datasets by extracting events from frames. In [108], the MNIST [81] and the Caltech-101 [39] datasets have been converted into events stream by moving an event camera in front of a screen

where the frames were displayed. A similar strategy has been adopted in [64], where the authors generate events datasets from image datasets [76, 120, 45], and where the event-based camera was still and the motion was performed by sliding the images on the monitor. These approaches allow to obtain very large datasets without the need for manual labeling, but the 2D characteristic of the screen and its limited refresh rate negatively impact the quality of the events stream. An alternative procedure has been proposed in [43] where Gehrig *et al.* leverage the capabilities of event simulators, like [118, 65], to transform popular frame-based datasets into their event-based counterparts. As such, clearly real data are still needed to fully exploit the properties of the event cameras and to accurately replicate noise and sensor nonidealities.

Over the past few years, the number of event datasets collected by using event-based sensors has in fact increased. For example, in [12, 62] and in [159] two datasets collected in driving scenarios with a single and a stereo configuration of a DAVIS346 camera respectively are presented to estimate quantities such as the steering angle, depth, and odometry.

Event-based datasets for person detection More recently, the number of event-based datasets for person detection has increased. In [101], Shu *et al.* present a small dataset completely focused on pedestrians composed of 12 sequences recorded using a fixed DAVIS346 event camera containing 4670 labeled people. This dataset is mainly used for surveillance and it contains also sequences for action recognition and fall detection. Another example is the one produced by Bolten *et al.* in [15] collected with three fixed CeleX-4 DVS [49] event cameras. It features recordings of an outdoor urban public area, and it has been developed for long time monitoring purposes. In 2020, Prophesee released two large datasets recorded in driving scenarios which are the GEN1 Automotive Detection Dataset [31] and the 1 Megapixel Automotive Detection Dataset [109]. The former presents a total of 255 781 manually-labeled bounding boxes (228 123 cars, 27 658 pedestrians) collected for 39 hours with a Prophesee Gen1 sensor [114] at a resolution of 304×240 pixels, while the latter is the most comprehensive event-based dataset to date and it counts 15 hours of recordings with a resolution of 1280×720 pixels and 25 million automatically-labeled bounding boxes. The people labeled in these two datasets are mainly pedestrians walking on sidewalks.

Event-based cameras in Robotics The interest in event-based cameras is increasing also in robotics and their usage has been investigated for Simulatenous Localization And Mapping (SLAM) [119], control strategies [14, 147], and the estimation of quantities like the pose [104, 164], the optical flow [160, 146, 44] and the trajectory of moving objects [102, 53]. Event-based sensors find applications in diverse tasks, ranging from guiding ground robots [14] or underwater platforms [147] to developing obstacle avoidance or SLAM algorithms for Unmanned Aerial Vehicles (UAVs) [37, 139].

4.3 Dataset

PEDRo is a dataset recorded with a moving camera and completely focused on person detection, specifically designed for service robotics applications[1]. It contains 43 259 bounding boxes associated to 27 000 samples with a duration of 40 ms each, extracted from 119 recordings with an average duration of 18 s. Each sample features at least one and at most six bounding boxes. The dataset presents a wide variety of indoor and outdoor scenarios, ranging from office and house environments to mountains, lakes landscapes and seafronts. The recordings are taken with different meteorological conditions such as sunny, rainy and snowy during day and night. The dataset has been collected in 6 months from September 2022 to February 2023 and the people recorded in PEDRo range from 20 to 70 years of age. Most of the labeled subjects are walking, although there are examples of people standing still, sitting, or running. I obtained informed written consent from all the recorded individuals, and I further protect their privacy by publishing only the events and the labels of the recordings.

4.3.1 Data collection and labeling

The dataset has been entirely recorded using a DAVIS346 event-based camera [1] which has a resolution of 346×260 pixels and outputs simultaneously events and grayscale frames. The camera is in motion, it has been hand-carried to capture

[1]PEDRo is publicly available and it can be obtained via the following link: https://github.com/SSIGPRO/PEDRo-Event-Based-Dataset.git

the events and the position of the sensor varies among recordings. The dataset has been fully manually labeled by the authors using the grayscale frames and all the bounding boxes have been double-checked. In this case, automatic labeling performed by state-of-the-art object detection models like YOLOv8x [69] does not provide reliable results, probably due to the low resolution of the grayscale frames offered by the camera.

In order to highlight this, I evaluate the quality of the predicted bounding boxes using the Intersection over Union (IoU) as a criterion. More specifically, given two bounding boxes, one true (manually labeled) and one predicted, the IoU measures their degree of overlap as the ratio of their intersection area to their union area (e.g., 1 means perfect overlap, 0 no overlap). Since a frame can contain multiple bounding boxes, I select the optimal true-predicted couples and then I evaluate the average IoU. The optimal true-predicted couples are selected as follows:

- the IoU is evaluated for all the combinations of true and predicted bounding boxes;
- the true and predicted bounding boxes associated with the maximum IoU value are selected as an optimal couple and removed from the computation, and this is repeated until there are no couples remaining;
- all the unpaired bounding boxes count as zero IoU value.

With the largest pretrained YOLOv8 model, 22% of the 27000 greyscale frames used for labeling do not reach an average IoU of 0.85 and almost 45% do not reach an average IoU of 0.90. This means that, for this dataset, automatic labeling is not a suitable option to obtain high-quality ground truth. Figure 4.2 shows two frames that result to be wrongly labeled using the YOLOv8x model.

4.3.2 Dataset Format

The 119 recordings that compose the dataset have been split into train, validation and test subsets. To avoid overlap of data, every single recording belongs entirely to one of these three groups. The number of bounding boxes in the

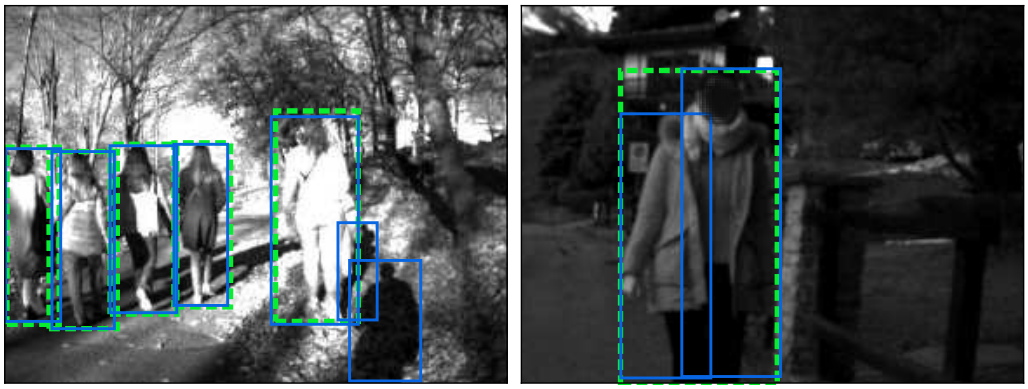

Fig. 4.2 Bounding boxes predicted by the YOLOv8x model are shown with solid light blue lines, while the manually annotated labels are shown with dashed green lines. Figure from [16].

dataset is 43 259, of which 34 243 (79.2%) in train, 4372 (10.1%) in validation and 4179 (9.7%) in test.

Each subset is associated to a text file that lists the recordings and the names of the samples it contains. With sample, I refer to the stream of events collected in a time interval of 40 ms preceding the timestamp of the frame used to obtain the corresponding labels. This time interval is determined by the acquisition rate of the grey-scale images used for the manual labeling process (25 fps).

The events (with positive and negative polarity) are stored in a numpy structure, while their corresponding labels are provided in Pascal VOC format [36]. Each sample can be coupled with its corresponding label by considering the matching names (e.g., file `frame0000001.npy` is associated with `frame0000001.xml`).

4.3.3 Analysis and Statistics

To better highlight some peculiarities of the dataset, I extract some statistics and I compare them to the ones obtained from the GEN1 Automotive Detection Dataset [31]. I select the GEN1 dataset as it is the most extensive automotive dataset with hand-labeled annotations and its spatial resolution is similar to PEDRo.

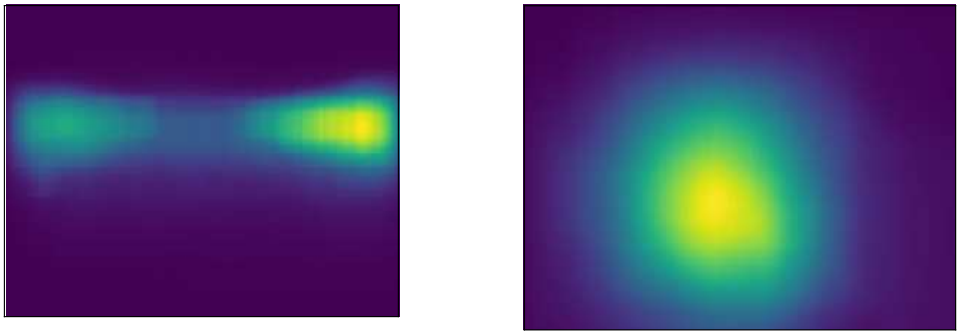

Fig. 4.3 On the left, the heatmap displaying labeled bounding boxes for pedestrians of the GEN1 dataset, while on the right, the heatmap for people in our dataset. Figure from [16].

I start this analysis by looking at the distribution of the bounding boxes in the two datasets using heatmaps, computed by counting the number of bounding boxes covering each pixel over the entire dataset. I normalize the count by dividing it by the total number of bounding boxes in the datasets, resulting in heatmaps containing values in the range $[0, 1]$ as in Figure 4.3. Since the GEN1 dataset is collected using a camera on a vehicle, detected people are typically pedestrians walking on sidewalks, so bounding boxes are confined to the image margins. On the contrary, our dataset focuses more on people which are closer to the center of the image and to the camera. Moreover, Figure 4.4 shows the distribution of sizes of bounding boxes diagonals in the GEN1 and in our dataset. The automotive dataset presents a long tail distribution and most of the bounding boxes have a small diagonal while our dataset features more bounding boxes with a wider range of diagonal sizes. These characteristics highlight how PEDRo features labeled data whose properties are missing in automotive datasets like GEN1.

4.4 Experimental Results

In this section I evaluate the performance of YOLOv8x on PEDRo and I compare it with the performance on GEN1. For each dataset, I start from the pre-trained YOLOv8x architecture and I train the model for 5 epochs with Stochastic Gradient Descent (SGD), learning rate of 0.01 and a batch size of 64. Each input sample consists in events organized in a Surface of Active Events

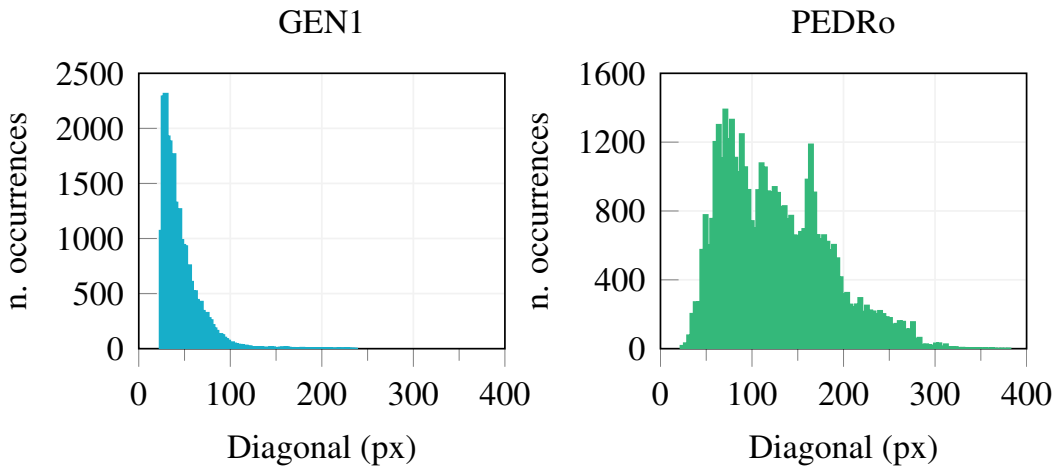

Fig. 4.4 On the left, the distribution of the length (in pixels) of the diagonals of bounding boxes for pedestrians in the GEN1 dataset, while on the right, the diagonals for people in our dataset. Figure from [16].

(SAE) [105, 101]. With SAE, the events from a sample are aggregated together in a single two-channel tensor (a two-channel image). One channel is generated from positive events, while the other from negative ones. For each channel, the value corresponding to each pixel is proportional to the timestamp of the most recent event, i.e. the more recent is the event, the higher is the value. In order to make SAE compliant with YOLOv8x, the values are normalized between 0 and 255 (which is typical for images). Furthermore, since the pre-trained YOLOv8x model accepts 3 channels images, I have removed the third input channel from the model.

To compare the performance, I consider the mean Average Precision value (mAP) [88] which is a widely used metric for understanding the performances of a neural network employed for an object detection task [63, 109]. Figure 4.1 presents the results in terms of mAP_{50} and $mAP_{50:95}$ obtained on different test sets. Here I see that GEN1 and PEDRo actually are uncorrelated datasets, being the mAP values quite low when cross-training YOLOv8x, when a model trained on GEN1 is tested on PEDRo and vice-versa. This means that GEN1, being an automotive dataset focused on pedestrians, is not suited for person detection where human figures are the main subject of the scenes. On the contrary, PEDRo allows the training of structures capable of detecting persons closer to the camera and in different environments, which is relevant for different

Test \ Train	GEN1	PEDRo	GEN1 + PEDRo
GEN1	0.716|0.341	0.487|0.205	0.718|0.342
PEDRo	0.437|0.237	0.895|0.586	0.794|0.504

Table 4.1 Results expressed as $mAP_{50}|mAP_{50:95}$ with YOLOv8x trained on different training sets and evaluated on various test sets.

robotic and surveillance applications. Furthermore, by training YOLOv8x with GEN1 in conjunction with PEDRo, there is no loss in performance on the recognition of pedestrians while the model becomes also capable of recognizing targets contained in PEDRo.

4.5 Conclusions

This paper presents PEDRo which is, to the best of my knowledge, the largest manually annotated event-based dataset recorded with a moving camera and explicitly designed for people detection. This dataset focuses on people, with a wide variety of recording environments and different lighting conditions, making it a relevant addition to other existing event-based datasets, such as the GEN1 Automotive dataset. PEDRo presents an opportunity for enhancing the prediction capabilities of deep learning object detection models with events as input, which can pave the way for novel research avenues and potential applications in diverse fields including but not limited to robotics and surveillance.

Chapter 5

A Dynamical Attention-based Pipeline for Event-based Computer Vision Tasks

As I have shown in the previous chapter, Dynamic Vision Sensors (DVS) offer a unique advantage in capturing changes in luminance asynchronously, providing high temporal resolution and efficiency, making them particularly suitable for applications like robotics and autonomous driving. However, adapting the sparse and asynchronous nature of DVS data for traditional non-recurrent deep learning models, such as convolutional neural networks (CNNs) and transformer-based architectures, poses challenges. In fact, classical methods, such as time surfaces and voxel grids, convert event-based data into a form suitable for frame-based Deep Neural Networks (DNNs). While effective, these methods often sacrifice the fine-grained temporal details intrinsic to DVS data. This can diminish the advantages of DVS in capturing fast-moving or transient phenomena. I aim to contribute addressing this issue and propose a dynamic pre-processing pipeline called *Memory of Events through Spatial Attention* (MESA), that enhances the currently used event-based data representations. This is obtained by storing events in a memory tensor with pixel-wise adaptive forgetting factors generated in real time through a spatial-attention module. Tested on multiple tasks, this method improves performance with minimal computational cost, enhancing the performance of state-of-the-art non-recurrent DNNs across diverse computer vision tasks. In particular, by using MESA, the

accuracy on CIFAR10-DVS with MobileViT-v2s improves by more than 15% and with DETR-ResNet50, the mAP on the PEDRo object detection dataset is three times higher the baseline achieved with time surfaces alone.

5.1 Introduction

In recent years, Deep Neural Networks (DNNs) have emerged as the de facto standard for tackling a wide range of computer vision tasks. In particular, the literature enumerates a wide variety of non-recurrent DNN models, ranging from the widely acclaimed convolutional models [81, 78, 54, 60, 121] to the most recent and highly scalable transformer-based models [35, 19, 99]. These models are employed for tasks ranging from simple image classification [81, 77, 33] to object detection [162], often in dynamic, quickly changing scenarios requiring a high temporal resolution analysis of data in video streams, such as egocentric vision [8] and autonomous driving [142].

In this area, event-based encoding of data is an interesting approach to solve computer vision tasks when processing speed is of the essence. These asynchronous devices respond only to the luminescence changes in the visual scene, generating a stream of temporally and spatially-sparse events with a very high temporal resolution in the order of 1 µs and very low-power and low-memory requirements. This kind of representation adapts well to time-varying models such as the recurrent neural networks [58]. However, these models require a computationally intensive training process [9], which has led the current best-performing models to favor non-recurrent architectures [138].

Because of this, a well-known approach is to use meaningful representations from streams of events that are suitable to standard non-recurrent computer vision models [136, 130]. Popular strategies involve time surfaces [80] and voxel grids [161]. While time surfaces flatten the events within a certain time range to a single image-like matrix, voxel grids are 3-dimensional structures where events are aggregated into small spatial-temporal bins; both the approaches capture changes over time and space in a dense format suitable for processing by standard computer vision models. Unfortunately, these representations encode video data only within a certain time range, meaning that not every piece of input contains enough information for the DNN models to generate

accurate outputs. While the enlargement of the time range would be an intuitive solution to enclose a larger amount of data in a single input tensor, this would also inevitably negate the event-based cameras benefit of having a very high temporal resolution.

I will start from [17] where I developed an approach to accumulate time surfaces over time and store them in *memory channels*. At each time step, the memory channels are scaled by a constant forgetting factor $k \in (0,1)$, which helps to discard outdated, no-longer-relevant information. Although this solution proved effective, the use of constant, manually tuned values of k is clearly suboptimal. Additionally, all values in the time surfaces are uniformly scaled by the same factor, which is also less than ideal.

In this chapter, I explore the possibility of extending the concept of *memory of events* introduced in [17]. To do so, I employ them on diverse datasets used in computer vision for classification and object detection tasks, leveraging various event representations. Furthermore, I propose a flexible, dynamic pre-processing pipeline, called Memory of Events through Spatial Attention (MESA) to generate information-rich inputs from event-based data. Specifically: 1) I accumulate the event representation in a *memory tensor* and apply a dynamic, pixel-by-pixel forgetting factor matrix $\boldsymbol{K}$, generated at each time-step with a spatial-attention module [143]; and 2) I test both time surface and voxel grid representations on multiple classification and object detection tasks and state-of-the-art DNN models.

Using MESA greatly enhances the performance of state-of-the-art estimators in classification and object detection tasks with minimal computational overhead. The rest of this chapter is structured as follows. In Section 5.2 I present the methodology, in Section 5.3 I list the datasets and models used to test the proposed pipeline and I provide information on the training process. In Section 5.4 I present the numerical results of the experiments. Finally, the conclusion is drawn.

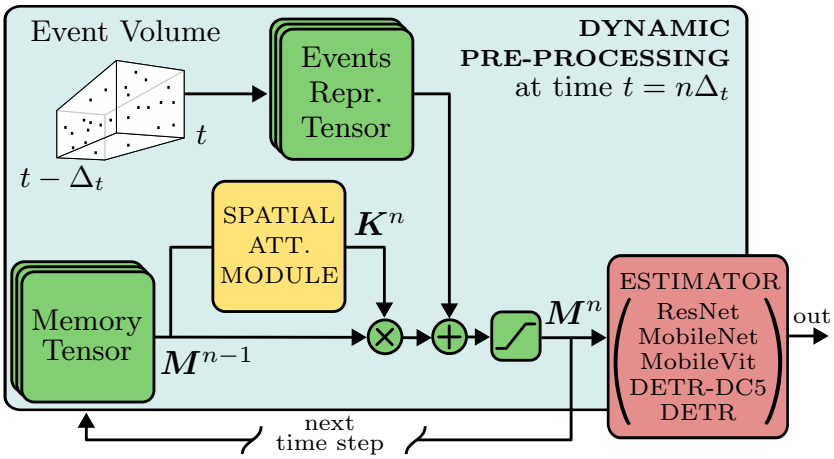

Fig. 5.1 Pre-processing pipeline. A tensor-like representation of an event stream – in the time range $(t, t+\Delta_t)$ – is accumulated (with 0 to 1 clipping) over time and stored in a memory tensor $\boldsymbol{M}^n$. At each time step n, the values corresponding to each pixel of the memory tensor $\boldsymbol{M}^{n-1}$ are dampened with varying strength by a forgetting matrix $\boldsymbol{K}^n$, dynamically generated by the spatial attention module.

5.2 MESA methodology

DVSs generate sparse and asynchronous streams of events that encode the pixel-local changes in the brightness of the scene. Each event is in the form $\boldsymbol{e} = (x, y, t, p)$, where x, y and t are the pixel coordinates and detection time of the luminescence alteration, and $p \in \{+1, -1\}$ is the polarity of the event (indicating whether brightness is increasing or decreasing). In the input pipeline, I employ two event representations, either time surfaces [80], or voxel grids [161], but the same pipeline can be used with many others available in the literature such as TORE volumes [7]. In this work, the employed representations are generated by means of the *Tonic* framework [82].

These event representations are sampled within time intervals of size Δ_t over all the sensor area. A new representation is generated each discrete time step $n = 1, 2, 3, \ldots$ corresponding to times $t = n\Delta_t$ and encoded in a tensor $\boldsymbol{R}^n \in \mathbb{R}^{H \times W \times C}$, where H and W are the number of pixels in the DVS sensor along height and width, respectively, while C is the number of channels, whose definition varies depending on the representation. I define as $\mathcal{E}_p^n(x, y)$ the set of the events with polarity p located at pixel (x, y) in the time range

$t \in ((n-1)\Delta_t, n\Delta_t]$. With $\mathcal{E}^n(x,y)$ I consider both positive and negative events in the corresponding set.

Time surfaces (ts) are image-like matrices where each pixel is defined as

$$\boldsymbol{R}^n_{\text{ts},p}(x,y) = \exp\left[-\frac{1}{\tau}\left(n\Delta_t - \max_{\boldsymbol{e}\in\mathcal{E}^n_p(x,y)} t_{\boldsymbol{e}}\right)\right] \tag{5.1}$$

where $t_{\boldsymbol{e}}$ is the time of event $\boldsymbol{e}$ and τ is a time constant. Using time surfaces, the resulting event representation $\boldsymbol{R}^n_{\text{ts}}$ is a tensor with $C = 2$ channels, namely $\boldsymbol{R}^n_{\text{ts},+1}$ encoding positive events and $\boldsymbol{R}^n_{\text{ts},\text{-}1}$ encoding negative events.

Conversely, voxel grids (vg) are defined as 3-dimensional structures. The time dimension is discretized in B bins of size $\delta_t = \Delta_t/(B-1)$, so the voxel grid is a tensor with $C = B$ channels. Each pixel at position (x,y) of channel z is then defined as

$$\boldsymbol{R}^n_{\text{vg}}(x,y,z) = \sum_{\boldsymbol{e}\in\mathcal{E}^n(x,y)} p_{\boldsymbol{e}} \max\left(0, 1 - |z\delta_t - t^*_{\boldsymbol{e}}|\right) \tag{5.2}$$

where $t^*_{\boldsymbol{e}} = \left[t_{\boldsymbol{e}} - (n-1)\Delta_t\right]/\delta_t$ is the time of event $\boldsymbol{e}$ shifted to the event representation range and normalized to the bin size δ_t and $p_{\boldsymbol{e}}$ is its polarity. The max operator in (5.2) is equivalent to the bilinear sampling kernel defined in [67], which means that the closer an event is to the center of a temporal bin, the more it influences the corresponding value, and a single event can influence two adjacent bins at the same time.

For each time step n, either the time surface or the voxel grid representations are accumulated over time into a *memory tensor* $\boldsymbol{M}^n \in \mathbb{R}^{H\times W\times C}$ as

$$\boldsymbol{M}^n = \left[\boldsymbol{K}^n \odot \boldsymbol{M}^{n-1} + \boldsymbol{R}^n\right]_0^1 \tag{5.3}$$

where $\boldsymbol{K}^n \in (0,1)^{H\times W}$ is the *forgetting matrix* with the same spatial height and width of $\boldsymbol{M}^n$, $\odot$ is an element-wise multiplication operator that scales all the values corresponding to a spatial position in $\boldsymbol{M}^{n-1}$ by the corresponding value in matrix $\boldsymbol{K}^n$ and operator $[\cdot]_0^1$ saturates the argument between 0 and 1. Processing starts at time step $n = 1$, with $\boldsymbol{M}^0$ defined as an all-zero tensor. Fig. 5.1 shows in detail the proposed pre-processing pipeline.

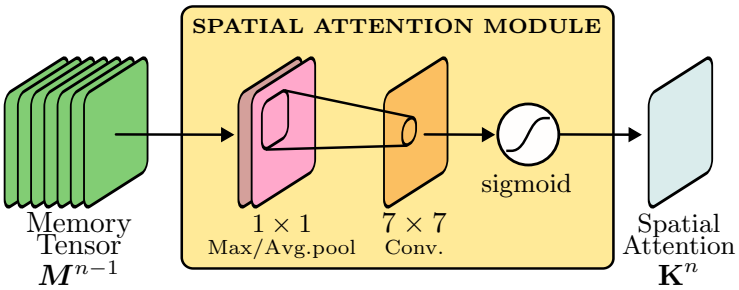

Fig. 5.2 Spatial attention module structure from [143]: a 2 channels tensor is generated by taking the average and the maximum of the memory tensor channels; then a single 7×7 convolution filter is applied and a sigmoid function is applied to get the forgetting matrix $\boldsymbol{K}^n$ with values between 0 and 1.

The forgetting matrix $\boldsymbol{K}^n$ is composed of values in the range $(0, 1)$ and is generated dynamically each time step by a *spatial attention* module [143]. This module selects what parts of the memory tensor are to be "forgotten" quickly (scaling values close to 0) and what parts are to be "remembered" (scaling values close 1). This is done by condensing all the channels of the memory tensor in 2 channels[1] and by applying a 7×7 convolutional filter followed by a sigmoid activation function[2]. This kind of approach allow us to dynamically keep only the important pixel stored in the memory tensor to maximize the performance on the task. Fig. 5.2 shows the spatial attention module structure.

This input pipeline well-adapts to any non-recurrent computer vision model, meaning that any image-driven estimator can be employed off-the-shelf with some fine-tuning. Compared to recurrent DNNs approaches, this moves the time dependency of the model entirely to the input pipeline, greatly simplifying the optimization process, i.e., it is not necessary to unroll the model during training with a backpropagation-through-time approach. Furthermore, I can select Δ_t as small as possible, increasing the throughput of the algorithm without sacrificing the performance. In contrast, simply using time surfaces or voxel grids would be unsuitable for high-throughput applications, since a small Δ_t would lead to inputs with limited information content.

[1] For each pixel I encode the channel-wise maximum and average value.
[2] I apply zero-padding to keep the spatial dimensions unchanged.

5.3 Case studies and training

5.3.1 Datasets

To validate the proposed pre-processing pipeline, I train and test it on multiple computer vision tasks:

- **N-MNIST** [108] (Classification) an event-based version of the MNIST dataset [81], consisting of event-based data representing hand-written digits. It contains 70 000 recordings with each image captured as an event stream.
- **CIFAR-10 DVS** [84] (Classification): a neuromorphic adaptation of the CIFAR-10 dataset, containing 10 000 event streams across 10 classes.
- **N-CALTECH101** [108] (Classification): the event-based version of CALTECH101 dataset [38]. Similar to the original, it contains 101 classes with an average of 50 event streams per class.
- **PEDRo** [16] (Object Detection): the pedestrian detection and tracking dataset recorded with an event-based camera presented in the previous chapter. It includes 43 259 bounding boxes included in 119 recordings, with scenarios involving pedestrians in dynamic environments.

5.3.2 Employed architectures

To solve the aforementioned tasks, I fine-tune and employ multiple state-of-the-art estimators in conjunction with the proposed pre-processing pipeline:

- **ResNet-18** [54] (Classification): a well-known residual network with skip connections that help mitigate vanishing gradients.
- **MobileNet-v3s** [60] (Classification): a lightweight convolutional neural network optimized for mobile devices.
- **MobileViT-v2s** [99] (Classification): the transformer-based version of MobileNet designed for efficient mobile use, integrating transformer blocks for high performance.

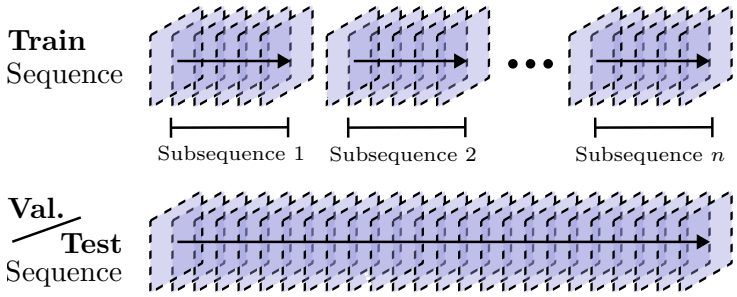

Fig. 5.3 Splitting of the sequence of tensors $\boldsymbol{R}^n$ for training and validation/test. The training subsequences are fed randomly to the algorithm during the training phase.

- **DETR** [19] (Object Detection): the state-of-the-art architecture for object detection [157], eliminating the need for region proposals and anchor boxes by directly predicting bounding boxes and class labels[3].
- **DETR-DC5** [19] (Object Detection): a variant of DETR designed to enhance its performance by incorporating dilated convolutions into the backbone network[4].

5.3.3 Training

The training sets of the employed datasets are composed of multiple event stream recordings. I partition each event stream into small chunks of duration Δ_t and generate a representation tensor $\boldsymbol{R}^n$ for each chunk, resulting in a sequence of tensors. The sequence from the training set is then split into multiple sub-sequences, which are batched and randomly fed to the pre-processing/estimator pair for optimization. On the other hand, validation and test sets are not split in sub-sequences to emulate the behavior of the system. Fig. 5.3 shows the training vs validation/test sequence splitting.

In the experiments, all models were trained for 20 epochs. The training sub-sequences are composed of 20 samples, their overlap is set to 3 samples, $\Delta t = 5\,\text{ms}$, $\tau = 3\,\text{ms}$ for time surfaces and $B = 3$ for voxel grid representations. These parameters were chosen to replicate a high-throughput setting char-

[3] I fine-tuned the model starting from the pre-trained version `facebook/detr-resnet-50` and `facebook/detr-resnet-101` available on HuggingFace.

[4] In this case I fine-tuned the model starting from the pre-trained version available on HuggingFace: `facebook/detr-resnet-50-dc5`.

Table 5.1 Accuracy of the different models across the selected datasets. The best models are highlighted in **bold**. Best models with overlapping standard deviations are considered equally the best. In light gray, the results obtained with [17] and MESA.

Dataset	DNN	Time surfaces			Voxel grids		
		without memory	*static memory*	*with MESA*	*without memory*	*static memory*	*with MESA*
N-MNIST	ResNet-18	82.45 ± 0.95	87.16 ± 0.93	**90.30 ± 0.93**	80.76 ± 0.17	89.60 ± 0.65	**92.97 ± 0.42**
	MobileNet-v3s	73.20 ± 2.01	76.63 ± 2.06	**79.16 ± 0.77**	72.96 ± 0.91	81.38 ± 1.17	**84.52 ± 0.25**
	MobileViT-v2s	82.66 ± 1.04	88.15 ± 0.46	**91.82 ± 0.48**	82.18 ± 0.12	88.93 ± 0.84	**92.67 ± 0.24**
CIFAR-10 DVS	ResNet-18	49.71 ± 0.25	52.70 ± 1.23	**58.91 ± 0.63**	44.90 ± 0.48	49.63 ± 0.81	**53.19 ± 0.08**
	MobileNet-v3s	47.15 ± 0.11	**52.61 ± 0.90**	**53.82 ± 0.48**	42.73 ± 0.18	44.85 ± 0.27	**50.02 ± 0.18**
	MobileViT-v2s	52.86 ± 0.31	58.16 ± 1.28	**65.24 ± 0.80**	47.72 ± 0.83	52.51 ± 0.63	**58.85 ± 0.71**
N-CALTECH101	ResNet-18	64.56 ± 0.52	68.65 ± 0.79	**70.52 ± 0.24**	63.46 ± 0.25	70.26 ± 0.76	**74.36 ± 0.12**
	MobileNet-v3s	58.62 ± 0.19	65.84 ± 0.91	**68.02 ± 0.97**	59.52 ± 0.27	**70.38 ± 1.84**	**72.38 ± 0.23**
	MobileViT-v2s	65.96 ± 0.20	72.93 ± 0.96	**74.83 ± 0.43**	65.39 ± 0.25	75.05 ± 0.66	**79.05 ± 0.23**

Table 5.2 Performance in terms of $\text{mAP}_{.5:.95}$ of the different models on the PEDRo dataset. The best model for each configuration is highlighted in **bold**. I report in light gray the results obtained with the memory-based methods I propose.

Dataset	DNN	Time surfaces			Voxel grids		
		without memory	*static memory*	*with MESA*	*without memory*	*static memory*	*with MESA*
PEDRO	DETR-Resnet50	0.1195 ± 0.0008	0.3285 ± 0.0055	**0.3818 ± 0.0072**	0.2533 ± 0.0125	0.3923 ± 0.0183	**0.4421 ± 0.0243**
	DETR-Resnet101	0.2475 ± 0.0121	0.3965 ± 0.0234	**0.5557 ± 0.0113**	0.3843 ± 0.0211	0.4983 ± 0.0197	**0.5610 ± 0.0098**
	DETR-DC5-Resnet50	0.3423 ± 0.0305	0.5211 ± 0.0148	**0.5824 ± 0.0147**	0.4326 ± 0.0347	0.5070 ± 0.0301	**0.5833 ± 0.0102**

acteristic of event-camera applications. The batch size was set to 512 for classification models and 258 for object detection models. Given the large number of sequences resulting from the small Δ_t, data augmentation was not applied. The optimizer used was AdamW [92], with an initial learning rate set to 1e−4 and a weight decay of 1e−4.

Table 5.3 Relative memory and computational overhead introduced by MESA with respect to the estimator complexity.

DNN	Memory over. (%)	GFLOPS over. (%)
ResNet-18	<+0.001%	+0.004%
MobileNet-v3s	+0.003%	+3.726%
MobileViT-v2s	+0.001%	+3.026%
DETR-based	<+0.001%	<+0.001%
DETR-C5-based	<+0.001%	<+0.001%

5.4 Results

5.4.1 Accuracy

I assess the effectiveness of the proposed dynamic pre-processing pipeline in comparison to the direct use of the representation tensor $\boldsymbol{R}^n$. Fig. 5.4 shows the difference between the two types of inputs. I also perform a further comparison with a static pre-processing pipeline (static memory), as in [17], where $\boldsymbol{K}^n$ in (5.3) is replaced by a fixed value $k = 0.5$.

Table 5.1 and Table 5.2 compare the performance of the different pre-processing strategies for classification (accuracy metric) and object detection (mean average precision, $\text{mAP}_{.5:.95}$ metric) tasks, respectively. The use of a dynamic pre-processing pipeline proves to be highly effective, with MESA consistently emerging as the best option. MESA can achieve accuracy gains exceeding 10% and can improve the $\text{mAP}_{.5:.95}$ by up to 3 times compared to the other solutions.

5.4.2 Computational overhead

The proposed methodology involves the use of a spatial attention module plus the operations in (5.3) to update the memory tensor $\boldsymbol{M}^n$, which is ultimately fed to the estimator. This introduces a computational overhead, that I estimate both in terms of memory (bytes) and amount of Floating Point OperationS (FLOPS). Table 5.3 lists the relative overhead with respect to the computational complexity of the estimator under test. The absolute overhead is constant for any application and equal to 392 bytes (32-bit floating-point parameters) and 0.018 GFPLOS. The computational overhead introduced by the pre-processing pipeline is only a fraction of the total, reaching up to 3.8% for MobileNet-v3s and being negligible for the object detection models.

5.5 Conclusion

In this chapter, I have introduced a dynamic input pre-processing pipeline to enhance the performance of state-of-the-art neural models for event-based

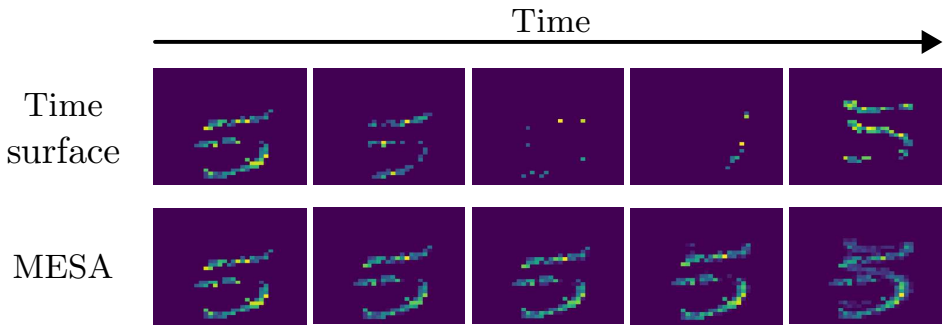

Fig. 5.4 Example of a sequence of time surfaces and a sequence of MESA representing the digit 5 from the N-MNIST dataset.

computer vision tasks. Specifically, I use a spatial attention module to update a memory tensor that accumulates multiple event representations over time. This attention mechanism allows the model to adapt dynamically to incoming data, retaining only the most relevant information and suppressing less important details. As a result, this approach improves neural estimators performance, offering a simple structure compatible with state-of-the-art computer vision models. Extensive tests on various event-based datasets and established neural estimators consistently show performance gains with the proposed pre-processing pipeline.

Part III

More transparent

Chapter 6

V-CEM: Bridging Performance and Intervenability in Concept-based Models

In the third and final part of this thesis, I will focus on another crucial topic that has become increasingly important with the growing size of models and the pervasive presence of AI systems: the explainability of these models. Specifically, I will delve into a subfield of what is known as *eXplainable AI* (xAI), namely *Concept-based eXplainable AI* (C-XAI). C-XAI is a rapidly growing research area that aims to improve the interpretability of AI models by utilizing intermediate, human-understandable concepts.This approach not only enhances model transparency but also enables human intervention, allowing users to interact with these concepts to refine and improve the model's performance. Concept Bottleneck Models (CBMs) explicitly predict concepts before making final decisions, enabling interventions to correct misclassified concepts. While CBMs remain effective in Out-Of-Distribution (OOD) settings with intervention, they struggle to match the performance of black-box models. Concept Embedding Models (CEMs) address this by learning concept embeddings from both concept predictions and input data, enhancing In-Distribution (ID) accuracy but reducing the effectiveness of interventions, especially in OOD scenarios. In this chapter, I propose the Variational Concept Embedding Model (V-CEM), which leverages variational inference to improve intervention responsiveness in CEMs. I evaluated the model on various textual and visual datasets in terms

of ID performance, intervention responsiveness in both ID and OOD settings, and Concept Representation Cohesiveness (CRC), a metric I propose to assess the quality of the concept embedding representations. The results demonstrate that V-CEM retains CEM-level ID performance while achieving intervention effectiveness similar to CBM in OOD settings, effectively reducing the gap between interpretability (intervention) and generalization (performance).

6.1 Introduction

Concept-Bottleneck Models (CBMs) [74] have emerged as a promising approach to interpretable machine learning by making task predictions through intermediate, human-understandable concepts. This architecture enhances model transparency by providing insight into the decision-making process through an interpretable mapping between concepts and outputs. Additionally, CBMs offer a distinctive advantage: the ability for human users to *intervene* on the intermediate concept predictions. This allows users both to rectify misclassified concepts, improving model performance, and to gain a deeper understanding of the relationships between concepts and task labels.

However, CBMs struggle with generalization, exhibiting limited performance. Their performance is constrained by the intermediate bottleneck, which restricts their ability to match the predictive accuracy of black-box models that directly map inputs to outputs. To address this issue, Concept Embedding Models [148, 71] (CEMs) have been introduced. CEMs generate dedicated embedding representations for each concept, thus alleviating the constrained representational capacity of the concept bottleneck. This approach improves model performance achieving black-box accuracy, while preserving a degree of intervenability (i.e., the level of efficacy of intervention) and interpretability. Besides testing model intervenability in In-Distribution (ID) settings, in this chapter I propose testing model intervenability in Out-of Distribution scenarios (OOD). The experiments show that the CBM architecture remains responsive to interventions on concept representations in both ID and OOD settings. In contrast, CEM exhibits very limited intervenability in OOD scenarios. Theoretically, this is due to CBM relying exclusively on predicted concepts for final decisions, whereas CEMs predictions are based on concept embeddings, which

integrate both concept predictions and raw input data. This entanglement negatively impacts CEM's intervenability in OOD settings. To address this challenge, I propose the Variational Concept Embedding Model (V-CEM), which utilizes variational inference to achieve black-box-level accuracy on ID tasks, while maintaining high intervention responsiveness in both ID and OOD scenarios.

In summary, this work makes the following key contributions: i) I demonstrate that while CEMs can achieve higher ID accuracy compared to CBMs, their ability to support interventions in OOD scenarios is significantly limited; ii) I introduce V-CEM, a model that achieves black-box generalization performance under ID conditions, comparable to CEMs; iii) I show that V-CEM retains responsiveness to interventions in both ID and OOD scenarios, similar to CBMs.

The chapter is structured as follows. In Section 6.2, I review related works, and in Section 6.3, I provide the foundational concepts necessary to understand this work. Section 6.4 introduces V-CEM, while Section 6.5 outlines the metrics used to evaluate concept representations. In Section 6.6, I present the results of the experimental campaign. Finally, Section 6.7 offers concluding remarks. Code is publicly available[1].

6.2 Related Works

C-XAI [113] has gained prominence as a solution to the growing demand for machine learning models that provide explanations in human-understandable terms [127]. Unlike traditional feature-based approaches, C-XAI emphasizes associating a model's behavior with human-interpretable concepts, offering a more intuitive and accessible way to understand model decisions.

A foundational approach in this domain is Testing with Concept Activation Vectors (T-CAVs), introduced in [70], which leverages directions in the latent space to measure, post-hoc, a model's sensitivity to predefined human-understandable concepts. In contrast, explainable-by-design models incorporate interpretability directly into their architecture. A prominent exam-

[1] https://github.com/VCEM

ple is CBM [74, 27, 3], which integrates supervised concepts as intermediate representations within the prediction pipeline, enabling more transparent and controllable decision-making. This integration facilitates direct intervention and correction of errors, thus enabling a more interactive approach to model understanding.

Additionally, Concept Embedding Models (CEMs) [148] push the boundaries of the interpretability-performance trade-off by incorporating concept embeddings into the learning process, enabling richer representations while maintaining concept-based explanations. Building on this approach, Prob-CBM [71] employs concept embeddings to capture uncertainty in concept predictions, offering explanations that incorporate both the concept and its associated uncertainty.

However, a critical challenge remains: ensuring robustness in OOD scenarios. OOD generalization is essential in machine learning, as it determines a model's ability to maintain reliable performance when encountering data that deviates from the training distribution.

Methodologies such as Outlier Exposure [57] and ODIN [86] aim to enhance the detection and management of OOD samples during inference. Additionally, studies like [48] underscore the necessity of benchmarking OOD performance through meticulously designed experimental protocols. More recently, other works [140, 6] have demonstrated that the decline in the performance of deep learning models following deployment in real-world applications can be mitigated by incorporating human assistance to support OOD generalization. Motivated by this approach, I seek to investigate the potential of leveraging the interveneability characteristic of concept-based models to enable human interventions under OOD conditions. The interplay between OOD generalization and concept-based explainability remains a relatively unexplored yet promising research direction. Integrating these domains could pave the way for more resilient and interpretable models that provide actionable explanations, even in challenging and unforeseen scenarios.

6.3 Background

Concept Bottleneck Models (CBMs).

Let $x \in X \subset \mathbb{R}^d$ be an input realization, $c \in C \subset [0,1]^k$ represent interpretable concepts, and $y \in Y \subset \{0, \dots, N\}$ denote the task label. CBMs assume a generative process where x determines c, which in turn influences y. A CBM consists of a concept encoder $p(c \mid x)$ and a task classifier $p(y \mid c)$, trained end-to-end to approximate $p(y, c \mid x) = p(y \mid c)p(c \mid x)$. The corresponding Probabilistic Graphical Model (PGM) is shown in Figure 6.1a. Modifying a concept c_j removes its reliance on x. This characteristic is especially crucial in OOD scenarios, as it enables to completely replace the concept representation generated by the concept encoder for a given concept. However, the bottleneck on c, while enhancing interpretability, limits performance in ID settings, resulting in a trade-off between interpretability and accuracy.

Concept Embedding Models (CEMs).

CEM alleviates the usual conflict between interpretability and performance by introducing a rich concept representation, the concept embedding $\mathbf{c} \in \mathbf{C} \subset \mathbb{R}^{k \times m}$, as shown in CEM PGM in Figure 6.1b. Unlike the CBM architecture, CEM defines a new conditional distribution $p(\mathbf{c} \mid c, x)$ that integrates both the input x and the concept c, enabling the generation of concept embeddings that capture concept-specific information enriched by the input instance x. These embeddings are then used to model the distribution $p(y \mid \mathbf{c})$, which predicts task labels. Similarly to CBMs, CEM is trained to approximate $p(y, c \mid x)$. Despite utilizing embeddings, CEM maintains the ability to support concept interventions: modifying a concept influences the conditional distribution $p(\mathbf{c} \mid x, c)$, thereby altering the generated embeddings. The dependence on x, which contributes to high ID performance, still remains after human intervention. As a result, CEM becomes less responsive to interventions in OOD scenarios, as the concept embedding generated in these cases may contain poor-quality information that cannot be overridden by human input.

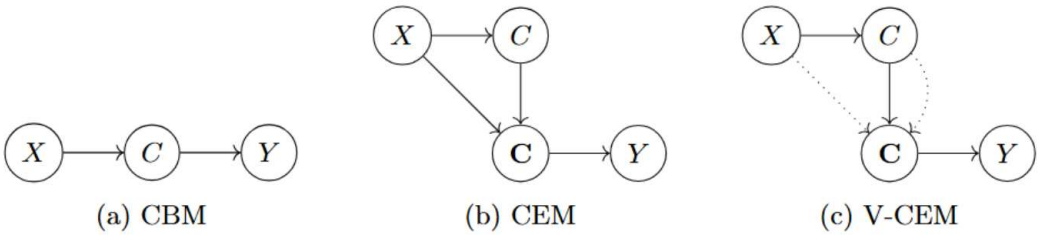

Fig. 6.1 Probabilistic Graphical Models of a) CBMs, b) the CEMs, and c) the proposed V-CEM architecture. Solid lines represent the data generation process, while dotted lines represent inference. Figure from [30].

Intervention.

Interventions in Concept based Models enable humans to correct model errors and gain insights into the relationship between concepts and tasks. This capability is crucial for developing interpretable models by improving transparency, trust, and control over decision-making. For instance, in a classification task where the objective is to categorize birds based on a set of concepts representing their features, if the concept *Red Breast* of an image depicting a *Red Breasted Parrot* is misclassified, the model might assign an incorrect bird label to the image. A human can adjust the concept prediction, which in turn may alter the final task prediction of the model. Different approaches enable various types of interventions: concept intervention [74, 148], where the predicted concept is directly replaced, and concept embedding intervention [71], where the concept's embedding is adjusted. Formally, in concept intervention, the concept $c_j \sim p(c_j \mid x)$ is replaced with $c_j := c'_j$, where c'_j is the concept assigned by the human. In a similar manner, in concept embedding intervention, $\mathbf{c}_j \sim p(\mathbf{c}_j \mid x)$ is substituted with $\mathbf{c}_j := \mathbf{c}'_j$, where $\mathbf{c}'_j$ is the embedding representing concept j that the human uses to correct the misclassified concept.

6.4 Variational CEM

I propose Variational CEM, a methodology to maintain CEM performance in ID settings by leveraging the rich, sample-specific information of the concept embeddings while ensuring their dependence primarily on the underlying concepts. At the same time, V-CEM enables targeted interventions on the concept

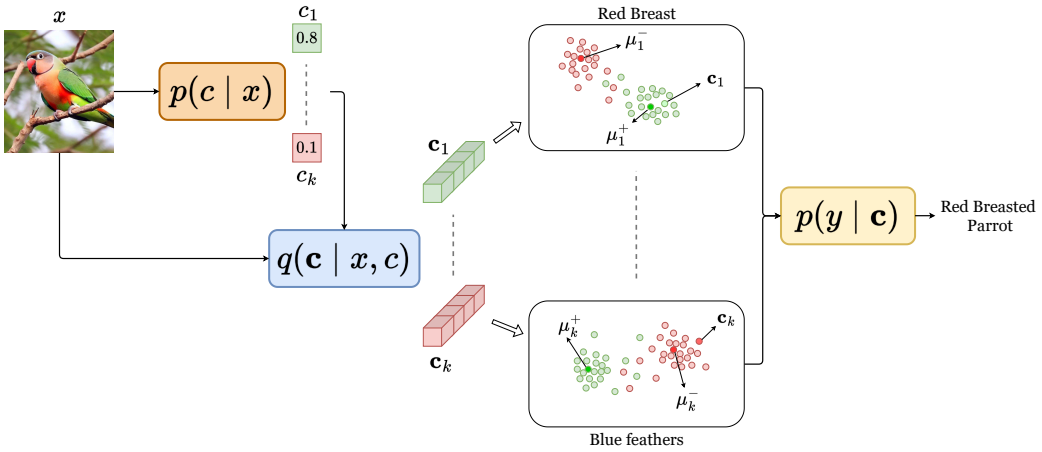

Fig. 6.2 Illustration of the V-CEM architecture. Given an image of a parrot with a red breast, V-CEM concept encoder $p(c|x)$ assigns a high probability to the "Red Breast" concept and a low probability to "Blue Feathers", which is absent. The approximate posterior $q(\mathbf{c}|x, c)$ maps concept predictions to concept embeddings clustered around $\mu^+_{\text{Red Breast}}$ and $\mu^-_{\text{Blue Feathers}}$, respectively. These embeddings are then employed to condition $p(y \mid \mathbf{c})$ and enable a correct label prediction ("Red Breasted Parrot"). Figure from [30].

embeddings that completely override their dependency on the input, ensuring high intervenability also in OOD scenarios. In Section 6.4.1 I describe V-CEM architecture, while in Section 6.4.2 I describe its training.

6.4.1 V-CEM Architecture

As shown in Figure 6.2, V-CEM is composed first of a concept encoder $p(c|x)$, mapping the input data x to an intermediate, interpretable concept layer c. Concept embeddings, $\mathbf{c}$, are generated from $q(\mathbf{c} \mid x, c)$ using both concept predictions and input features. The classification head $p(y|\mathbf{c})$ works on the concept embeddings to produce the final class prediction y.

However, from a probabilistic point of view, I assume a generative process where the concept embeddings $\mathbf{c}$ are only influenced by the interpretable concepts c and not by the input x, which is only used to derive the concept c. Similarly to CEM, the task label y is generated from a distribution conditioned on the concept embeddings. The PGM corresponding to this formulation is

depicted by the solid lines in Figure 6.1c. This generative framework leads to the following factorization:

$$p(x, c, \mathbf{c}, y) = p(x)p(c|x)p(\mathbf{c}|c)p(y|\mathbf{c}) \tag{6.1}$$

With respect to the CEM architecture, I introduce a prior $p(\mathbf{c} \mid c)$, which I will discuss in detail later. Also, notice how the concept embedding probability is only conditioned by the concept predictions $p(\mathbf{c}|c)$. Similarly to CBM and CEM, the objective is to approximate the joint distribution $p(y, c \mid x)$. Since $\mathbf{C}$ is unobservable, I account for its effect on the relationships between X, C, and Y by marginalizing over all possible values of $\mathbf{C}$:

$$p(c, y|x) = \int_{\mathbf{C}} \frac{p(x, c, \mathbf{c}, y)}{p(x)} d\mathbf{c} \tag{6.2}$$

Loss Derivation.

Using a variational inference approach, I define an approximate posterior distribution, $q(\mathbf{c} \mid x, c)$, which, like CEM, generates concept embeddings by conditioning on both the input and the concept (as illustrated by the dotted lines in Figure 6.1c). This allows for amortized inference, as the true posterior $p(\mathbf{c} \mid x, c)$ is intractable. This approach leads to the derivation of the following Evidence Lower Bound (ELBO) for the log-likelihood of the conditional distribution $p(c, y \mid x)$:

$$\log p(c, y|x) \geq \underbrace{-E_q\left[\log \frac{q(\mathbf{c}|x, c)}{p(\mathbf{c}|c)}\right]}_{\text{Pior Matching}} + \underbrace{\log p(c|x)}_{\text{Concept Loss}} + \underbrace{E_q[\log p(y|\mathbf{c})]}_{\text{Task Loss}} \tag{6.3}$$

A comprehensive derivation of the loss function is provided in Appendix B.1. The first term in the ELBO is the Kullback-Leibler (KL) divergence between the approximate posterior $q(\mathbf{c} \mid x, c)$ and the prior $p(\mathbf{c} \mid c)$, ensuring their alignment. I refer to this term as *Prior Matching*. This alignment is crucial, as it encourages the approximate posterior $q(\mathbf{c} \mid x, c)$—which depends on both the input x and concept predictions c—to resemble the prior $p(\mathbf{c} \mid c)$, which is independent of x. Maximizing the second and third terms of the ELBO (Concept and Task loss) optimizes both concept and task accuracy. Since the third term involves

averaging over concept embeddings sampled from the approximate posterior q, I approximate this by using the Monte Carlo method. Specifically, as described in [72, 93], I employ a large batch size and draw a single sample of $\mathbf{c}$ per data point using the reparameterization trick. All the distributions in the ELBO, besides the prior distribution $p(\mathbf{c} \mid c)$, are parameterized by neural networks.

Concept Embedding Encoder.

I assume that each concept c_j is independent of the others. Consequently, I define each concept embedding $\mathbf{c}_j$ as independent of the other concept embeddings and model it as a mixture of two multivariate normal distributions:

$$p(\mathbf{c}_j|c_j) = \delta(c_j)\mathcal{N}(\mathbf{c}_j;\mu_j^+,I) + (1-\delta(c_j))\mathcal{N}(\mathbf{c}_j;\mu_j^-,I),$$

where $\mu_j^+, \mu_j^- \in \mathbb{R}^m$ are learnable embeddings, I is the identity matrix, and $\delta(\cdot)$ represents the Dirac delta function, which evaluates to 1 if $c_j = 1$ and 0 otherwise. Here, μ_j^+ corresponds to the expected embedding when the concept is active ($c_j = 1$), while μ_j^- represents the expected embedding when the concept is inactive ($c_j = 0$). For the sake of simplicity, I define the approximate posterior as a multivariate normal distribution:

$$q(\mathbf{c}_j|x,c_j) = \mathcal{N}(\mathbf{c}_j;\hat{\mu}_j(x,c_j),\text{diag}(\sigma_j(x,c_j)))$$

where $\hat{\mu}_j(x,c_j),\ \sigma_j(x,c_j) \in \mathbb{R}^m$.

Given this definition for the prior and the approximate posterior, the *Prior Matching* term can be expressed in a closed-form solution. A detailed derivation of this formulation is presented in Appendix B.2. During training, the *Prior Matching* term encourages the approximate posterior q to position the multivariate normal distribution near μ_j^+ when $c_j = 1$ and near μ_j^- otherwise. This regularization promotes the formation of dense clusters for each concept state, ensuring that each state is represented by a distinct concept embedding: μ_j^+ for $c_j = 1$ and μ_j^- for $c_j = 0$. By exploiting this property of V-CEM, I can perform concept embedding intervention, thereby decoupling the concept embedding from the raw input data.

6.4.2 V-CEM Training

The model is trained to optimize the ELBO by minimizing its negative counterpart. Assuming each concept c_j follows a Bernoulli distribution, the second term in the ELBO reduces to a sum of binary cross-entropy losses, denoted as L_c. Similarly, if the task variable y follows a categorical distribution, the third term in ELBO corresponds to the expected cross-entropy loss over y, referred to as L_t.

Following standard practices in concept bottleneck models [148], I introduce a weighting parameter $\lambda_t \in [0,1]$ to balance the task loss L_t, allowing for trade-offs between concept learning and task performance. Additionally, a scaling factor $\lambda_p \in [0,\infty)$ is applied to the *Prior Matching* term L_p, influencing the model's regularization. Increasing λ_p progressively aligns V-CEM with a CBM, while setting $\lambda_p = 0$ removes constraints on concept embeddings, making the model function like CEM.

V-CEM is trained by minimizing the following objective function:

$$L = \frac{1}{k} L_c + \lambda_t L_t + \lambda_p L_p \tag{6.4}$$

where L_c is normalized by the number of concepts k. In this work, I set $\lambda_t = 0.1$ and $\lambda_p = 0.05$. An ablation study exploring the effect of varying λ_p on V-CEM's performance is provided in Appendix B.4.

To enhance the responsiveness of V-CEM to ID interventions, the *RandInt* regularization strategy [148, 71] is employed during the training phase, performing random concept embedding interventions with a predefined probability. Additional details about the specific settings of the proposed methodology and the baseline methods are provided in Appendix B.5.2.

6.5 Evaluating Concept Representations

In order to properly evaluate the model's intervenability in OOD settings, particularly when dealing with concept embeddings, concept accuracy might not be sufficient. In this section, I describe two further metrics that I use

for this scope: OOD intervenability and *Concept Representation Cohesiveness (CRC)*.

OOD Intervenability.

Concept interventions are generally used to assess the intervenability of a model [74], i.e., whether a model's predictions change when concept predictions are modified while keeping other factors constant. Model intervenability is normally evaluated ID by replacing concept predictions with concept labels. However, ID concept predictions are often already correct, thus the possibility to obtain a counterfactual prediction is low. Furthermore, for models relying on concept embeddings, this phenomenon is even more evident as part of the task prediction depends on x rather than c. Thus, in this chapter I evaluate model intervenability OOD. More specifically, I propose to analyze responsiveness to interventions under varying conditions by progressively adding random noise $\epsilon \sim N(0, I)$ to the input x. The perturbed input is thus defined as:

$$\tilde{x} = (1-\theta) \cdot x + \theta \cdot \epsilon, \quad \theta \in [0, 1]$$

where θ controls the noise intensity. Interventions are applied randomly on misclassified concepts, with an increasing probability $p_{int} \in [0, 1]$.

Concept Representation Cohesiveness.

Concept embeddings allow concept-based models to avoid the performance trade-off due to the CBM concept-bottleneck layer, as they enrich concept representation with sample-based information. Still, it is fundamental that this information represents the concept and not other input features; otherwise I may incur in the so-called "concept leakage" issue [96, 98], where the concepts encode spurious information related to other concepts. In other words, I would like each point in the concept embedding space $\mathbf{C}$ to represent a different instantiation of an active or inactive concept. As training a decoder for each concept is non-trivial, in this chapter I propose to assess this characteristic through an evaluation of the cohesiveness of the clusters associated with active and inactive concepts. More precisely, I compute *CRC* by splitting all concept embeddings into two clusters according to their concept predictions, and I

compute the corresponding silhouette score as follows:

$$CRC = \frac{1}{|C|} \sum_{i=0}^{|C|} s_i(\mathbf{c}_i, c_i) \tag{6.5}$$

where $|C|$ represents the number of concepts and $s_i(\mathbf{c}_i, c_i)$ represents the silhouette coefficient computed for the i-th concept over concept embedding representation $\mathbf{c}_i$ and considering as clustering labels the concept prediction c_i. For further detail on the computation of s_i I refer the reader to Appendix B.3. A higher silhouette score indicates a denser and tighter concept embedding space. This, in turn, indicates a model more responsive to OOD concept embedding intervention, as it samples from a denser representation.

6.6 Experimental Evaluation

To evaluate V-CEM, I seek to address several key research questions that guide the investigation. Specifically, I aim to answer the following:

(1) Does V-CEM exhibit comparable task performance to Black-box and CEM in ID settings?

(2) Is V-CEM more responsive than concept embedding-based approaches (CEM and Prob-CBM) in OOD scenarios?

(3) How does V-CEM concept representation compare to CBM representation, despite its reliance on concept embeddings?

6.6.1 Experimental Setting

In this section, I outline the experimental setup used to evaluate the performance of V-CEM. Specifically, I present the datasets, the baseline models and the training details.

Datasets.

I conduct experiments on a diverse set of vision and NLP datasets. For vision, I use MNIST Even/Odd and MNIST Addition, which are derived from the MNIST dataset [81] and involve binary classification and digit-sum prediction tasks, respectively. For these two datasets digits are used as concepts. I conduct experiments also on CelebA [90], a large-scale facial attribute dataset, where selected attributes serve as concepts and others as prediction targets. For NLP, I experiment with CEBaB [2], a dataset designed to study causal effects of concepts in sentiment analysis, and IMDB [95], where movie reviews are classified as positive or negative using interpretable aspects. More details on dataset preprocessing and structure are provided in Appendix B.5.

Baselines.

To assess the effectiveness of the proposed methodology, I compare it against several baseline models. For vision tasks, I extract embeddings using a frozen ResNet-34 [54], while for NLP tasks, I use *all-distilroberta-v1*[2] [128]. Both backbones are used without fine-tuning to extract embeddings from the input data. All baselines operate on these precomputed embeddings. The compared models include: (1) a standard Black-box model, implemented using two consecutive linear layers, (2) two variations of CBMs [74]: the first employing a single linear layer to map concepts to the task (CBM+Linear), and the second utilizing two consecutive linear layers (CBM+MLP), (3) Prob-CBM [71], (4) CEM [148]. Training details for all models are reported in Appendix B.5.

6.6.2 Results

The results highlight three key findings: (1) V-CEM outperforms CBMs and Prob-CBM while remaining comparable to CEM and Black-box models in ID settings, (2) it exhibits high responsiveness to interventions in OOD scenarios compared to CEMs and Prob-CBM, and (3) its concept embedding space $\mathbf{C}$ is more cohesive than that of concept embedding-based models.

[2]I use the pretrained model available at https://huggingface.co/sentence-transformers.

Table 6.1 The average task accuracy and corresponding standard deviation in ID settings obtained by the various methodologies across different datasets. V-CEM performance are the highest on average when considering concept-based models, surpassing also Black-box performance on three datasets.

	MNIST E/O	MNIST+	CelebA	CEBaB	IMDB
Black-box	98.56 ±0.01	67.59 ±0.57	**64.66** ±0.07	**80.20** ±0.25	86.98 ±0.48
CBM+Linear	98.82 ±0.04	44.19 ±1.86	49.75 ±0.18	63.66 ±3.48	87.30 ±0.58
CBM+MLP	98.82 ±0.13	68.63 ±0.72	51.01 ±0.51	72.51 ±5.88	86.48 ±1.88
CEM	98.75 ±0.07	69.84 ±0.91	64.49 ±0.08	80.12 ±0.14	86.79 ±0.77
Prob-CBM	97.38 ±0.61	27.31 ±3.92	51.64 ±7.12	77.86 ±0.95	85.90 ±0.38
V-CEM	**98.91** ±0.05	**73.12** ±0.35	64.49 ±0.15	79.62 ±1.29	**87.94** ±0.86

In-Distribution Performance.

In Table 6.1, I present the task accuracy results for the various models evaluated across different datasets in ID settings. The results clearly demonstrate that **V-CEM consistently outperforms traditional CBMs and Prob-CBM** in average ID performance. This trend is consistent across all datasets and this is particularly evident in MNIST Addition, where V-CEM achieves over 40% higher task accuracy compared to Prob-CBM and outperforms CBM+Linear by nearly 30%. The fact that V-CEM is able to surpass the black-box model is in line with the findings reported in [32]. Overall, V-CEM achieves ID performance comparable to CEM and the Black-box model while also attaining the highest average accuracy for MNIST E/O, MNIST+, and IMDB. This is achieved while maintaining similar concept accuracy across all models, as reported in Appendix B.6.

Intervention Responsiveness.

Figure 6.3 illustrates the task accuracy of various models when human intervention is used to correct misclassified concept predictions under varying levels of input noise $\theta \in [0, 1]$, revealing several key insights. As anticipated, CEM shows minimal responsiveness to interventions, underscoring a key limitation: its strong dependence on input, which makes it less effective in the presence of distributional shifts. In contrast, **V-CEM consistently shows greater responsiveness to interventions in OOD settings**, outperforming Prob-

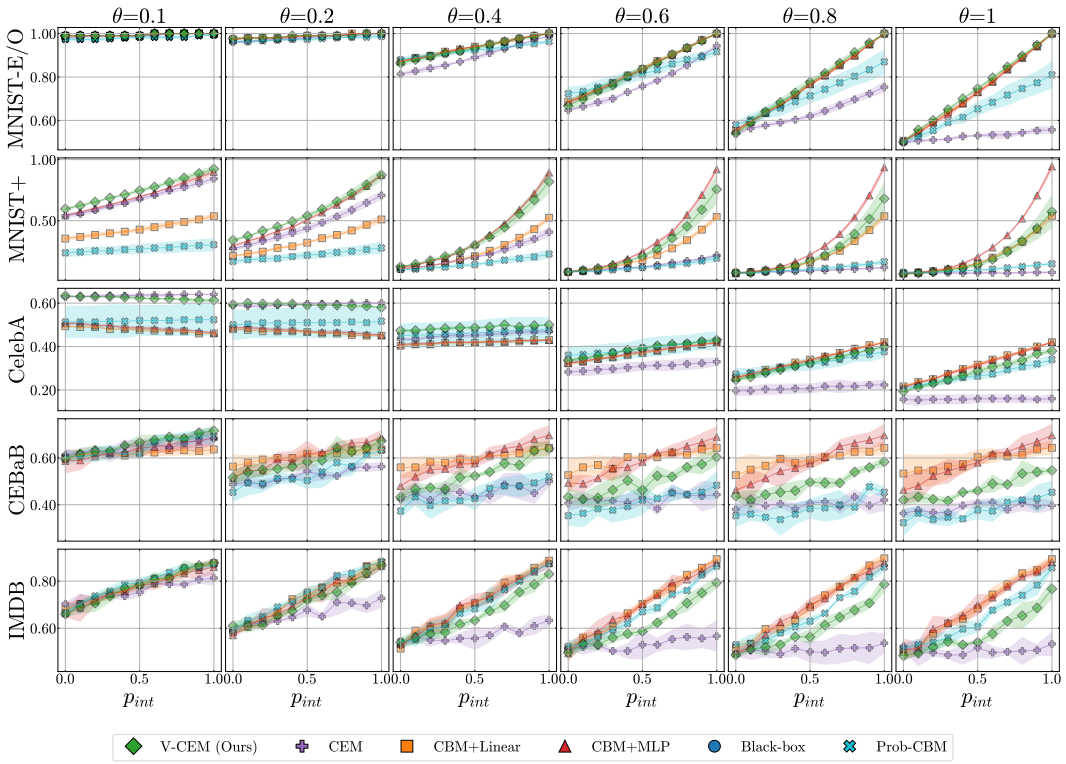

Fig. 6.3 The solid lines represent the mean task accuracy under random interventions at probability p_{int}, while the shaded areas indicate the standard deviation of each method. Results are reported across different models and datasets, under varying levels of input noise $\theta \in [0, 1]$. The Black-box model is not shown since it does not allow human interventions. Figure from [30].

CBM, which only surpasses V-CEM in responsiveness for the IMDB dataset. This suggests that V-CEM retains intervention efficacy by more effectively utilizing concept embeddings. Overall, V-CEM demonstrates responsiveness similar to CBMs while achieving superior performance in the ID scenario.

Concept embedding space evaluation.

As outlined in Section 6.6, to further investigate why V-CEM outperforms CEMs in terms of intervention responsiveness while maintaining performance comparable to CBMs in OOD settings, I propose to analyze the cohesiveness of the concept embedding space.

Ideally, each point in the concept embedding space should correspond to a distinct instance of an active or inactive concept. Specifically, for each concept,

Table 6.2 The average *CRC* values and their respective standard deviations in ID settings evaluated for all methodologies and datasets. The higher the better. V-CEM values are close to CBMs and always higher than both CEM and Prob-CBM.

	MNIST E/O	MNIST+	CelebA	CEBaB	IMDB
CBM+Linear	0.99 $\pm \leq 0.01$	0.92 ± 0.01	0.73 ± 0.01	0.70 ± 0.01	0.73 ± 0.01
CBM+MLP	0.99 $\pm \leq 0.01$	0.91 ± 0.01	0.72 ± 0.01	0.71 ± 0.01	0.74 ± 0.01
CEM	0.65 ± 0.01	0.65 ± 0.02	0.32 ± 0.02	0.33 ± 0.03	0.45 ± 0.04
Prob-CBM	0.73 ± 0.01	0.59 ± 0.02	0.31 ± 0.03	0.41 ± 0.05	0.50 ± 0.02
V-CEM	0.98 ± 0.01	0.85 ± 0.02	0.41 ± 0.03	0.59 ± 0.02	0.67 ± 0.02

I here identify two clusters and compute the *CRC* score across all concepts for each model and dataset. Table 6.2 presents the results, showing that **V-CEM's concept embedding space is more cohesive than that of concept embedding based models** (CEM and PRob-CBM) while remaining comparable to CBMs. Additionally, Figure 6.4 provides a 2D t-SNE visualization comparing the concept embedding spaces of CEM, Prob-CBM and V-CEM for different concepts in the CEBaB dataset, further illustrating this effect.

6.7 Conclusion

In this chapter I introduced V-CEM, a model that achieves ID performance comparable to Black-box and CEM while maintaining strong responsiveness to interventions in both ID and OOD settings. I have shown that its improved performance compared to other concept embedding based models originates from the cohesiveness of its concept embedding space which is ensured by the generative process that is conditioned on the concept prediction only.

Although V-CEM demonstrates strong responsiveness in OOD scenarios, it lacks an inherent mechanism for identifying OOD samples. Implementing such a mechanism would be beneficial, as it could assist human intervention by highlighting concepts associated with samples that deviate from those the model encountered during training. This could improve the model's ability to flag and address potential OOD instances, enhancing its overall reliability and reducing the risk of misclassification. Moreover, V-CEM was evaluated exclusively on OOD scenarios generated by introducing random noise into the

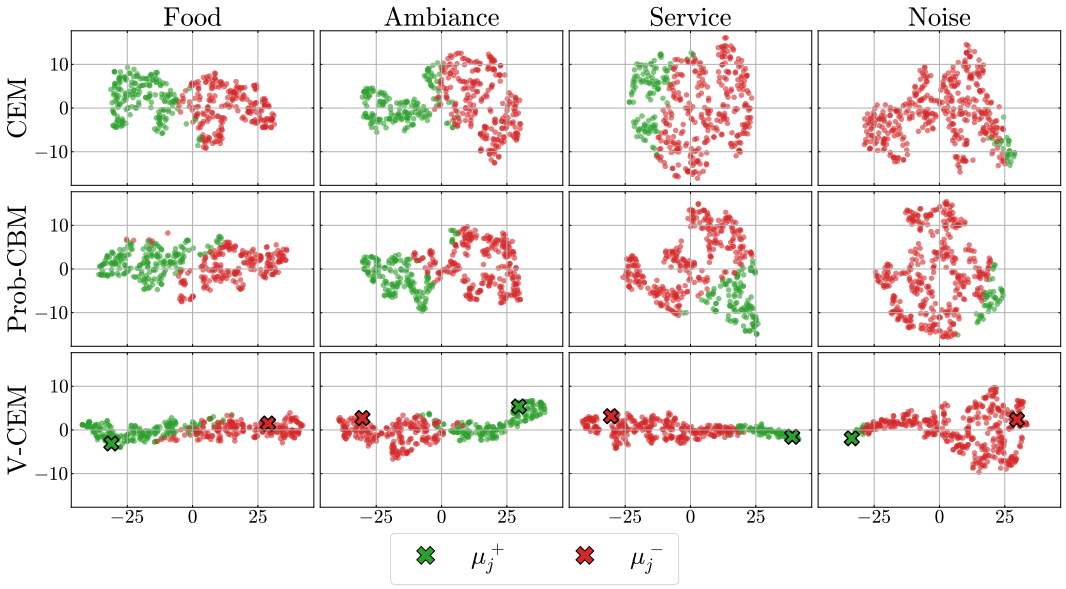

Fig. 6.4 2D t-SNE visualization of the concept embedding space $\mathbf{c}$ for the CEBaB dataset, comparing V-CEM, Prob-CBM and CEM. V-CEM concept representation is much denser than the ones of CEM and Prob-CEM. Figure from [30].

input. Additional experiments are required to assess its performance on other types of distributional shifts.

Future works.

Future research directions include extending V-CEM to handle multimodal inputs, allowing the model to integrate and process information from multiple data sources effectively, thereby creating shared and aligned concept embeddings across modalities. Another promising avenue is the incorporation of generative models as decoders for concepts, leveraging their capabilities to create concept visualizations from V-CEM cohesive concept embedding space. Finally, V-CEM models concepts independently from each other. In scenarios where the presence of a concept significantly affects other concepts, it may be opportune to explicitly model this dependency. Merging V-CEM with the strategy suggested in [34, 29] might offer a method for accomplishing this.

Chapter 7

Conclusions

This dissertation has advanced AI along three key axes –making models smaller, lighter, and more transparent– by proposing novel techniques in each domain. *Smaller AI:* I introduced the Multiply-and-Max/min (MAM) neuron paradigm, a map-reduce framework that replaces standard MAC (multiply-and-accumulate) neurons with neurons naturally prone to aggressive pruning. Using a vanishing-contributions training scheme, I transformed conventional fully-connected layers into MAM-based layers with negligible accuracy loss, enabling state-of-the-art networks (from AlexNet and VGG-16 to ViT-B/16) to retain high performance even when over 99% of weights are pruned. This unprecedented sparsification dramatically reduces model size and computation, opening opportunities to integrate such ultra-compact networks into broader architectures and specialized low-power hardware for edge AI. *Lighter AI:* Leveraging event-based neuromorphic vision, I developed efficient computer vision pipelines that exploit the sparse, high-temporal-resolution data from Dynamic Vision Sensors. In particular, I contributed the PEDRo dataset (the largest manually annotated event-camera dataset for person detection) and proposed the MESA (Memory of Events through Spatial Attention) pre-processing pipeline, which adaptively accumulates and forgets events via a learned attention mechanism. This approach preserves fine-grained temporal information and boosted the performance of standard deep models (e.g., improving classification accuracy by over 15% on N-MNIST and tripling object detection mAP on PEDRo) with only a 3% computational overhead, demonstrating that event-driven sensing can be harnessed for lightweight, low-power vision.

These results pave the way for broader adoption of event-based AI, suggesting further research on advanced event representations, real-time neuromorphic processing, and multimodal sensor fusion. *More Transparent AI:* Finally, to enhance explainability, I introduced the Variational Concept Embedding Model (V-CEM), which leverages variational inference to improve concept-based interpretability and human intervention in neural networks. V-CEM retains the high in-distribution predictive accuracy of concept embedding models while markedly improving their intervenability on concept predictions, even under distribution shifts, thereby bridging the gap between interpretability and performance. Empirical evaluations showed that V-CEM achieves black-box-level accuracy on vision and language tasks and attains intervention effectiveness comparable to classical concept bottleneck models in OOD settings, effectively combining the strengths of both approaches. This contribution provides a more transparent and controllable AI paradigm, and it prompts future explorations such as extending V-CEM to handle multi-modal inputs, incorporating generative concept decoders for richer concept visualization, and explicitly modeling dependencies among concepts to further enhance the transparency and trustworthiness of AI systems.

References

[1] (2019). *DAVIS346 datasheet*. IniVation.

[2] Abraham, E. D., D'Oosterlinck, K., Feder, A., Gat, Y., Geiger, A., Potts, C., Reichart, R., and Wu, Z. (2022). Cebab: Estimating the causal effects of real-world concepts on nlp model behavior. *Advances in Neural Information Processing Systems*, 35:17582–17596.

[3] Alvarez Melis, D. and Jaakkola, T. (2018). Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31.

[4] Anil, C., Lucas, J., and Grosse, R. B. (2019). Sorting out Lipschitz function approximation. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*.

[5] Aswani, A., R, C., and James, A. (2022). Unstructured Weight Pruning in Variability-Aware Memristive Crossbar Neural Networks. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3458–3462.

[6] Bai, H., Zhang, J., and Nowak, R. (2024). Aha: Human-assisted out-of-distribution generalization and detection. *arXiv preprint arXiv:2410.08000*.

[7] Baldwin, R. W., Liu, R., Almatrafi, M., Asari, V., and Hirakawa, K. (2023). Time-ordered recent event (TORE) volumes for event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2519–2532.

[8] Bandini, A. and Zariffa, J. (2023). Analysis of the Hands in Egocentric Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):6846–6866.

[9] Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625.

[10] Bich, P., Enttsel, A., Prono, L., Marchioni, A., Pareschi, F., Mangia, M., Setti, G., and Rovatti, R. (2025). On the universal approximation properties of deep neural networks using mam neurons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(9):8297–8304.

[11] Bich, P., Prono, L., Mangia, M., Pareschi, F., Rovatti, R., and Setti, G. (2022). Aggressively prunable MAM^2-based Deep Neural Oracle for ECG acquisition by Compressed Sensing. In *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 163–167.

[12] Binas, J., Neil, D., Liu, S.-C., and Delbruck, T. (2017). DDD17: End-To-End DAVIS Driving Dataset. *arXiv*.

[13] Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Guttag, J. (2020). What is the State of Neural Network Pruning? *Proceedings of Machine Learning and Systems*, 2:129–146.

[14] Blum, H., Dietmüller, A., Milde, M., Conradt, J., Indiveri, G., and Sandamirskaya, Y. (2017). A neuromorphic controller for a robotic vehicle equipped with a dynamic vision sensor. In *Robotics Science and Systems (RSS)*.

[15] Bolten, T., Pohle-Frohlich, R., and Tonnies, K. D. (2021). DVS-OUTLAB: A Neuromorphic Event-Based Long Time Monitoring Dataset for Real-World Outdoor Scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1348–1357.

[16] Boretti, C., Bich, P., Pareschi, F., Prono, L., Rovatti, R., and Setti, G. (2023). PEDRo: An Event-based Dataset for Person Detection in Robotics. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4065–4070.

[17] Boretti, C., Bich, P., Prono, L., Pareschi, F., Rovatti, R., and Setti, G. (2024). Memory in Motion: Exploring Leaky Integration of Time Surfaces for Event-based Eye-tracking. In *2024 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–5.

[18] Cai, Y. (2023). Achieve the Minimum Width of Neural Networks for Universal Approximation. In *The Eleventh International Conference on Learning Representations*.

[19] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham. Springer International Publishing.

[20] Chen, G., Cao, H., Conradt, J., Tang, H., Rohrbein, F., and Knoll, A. (2020a). Event-Based Neuromorphic Vision for Autonomous Driving: A Paradigm Shift for Bio-Inspired Visual Sensing and Perception. *IEEE Signal Processing Magazine*, 37(4):34–49.

[21] Chen, G., Wang, F., Li, W., Hong, L., Conradt, J., Chen, J., Zhang, Z., Lu, Y., and Knoll, A. (2022). NeuroIV: Neuromorphic Vision Meets Intelligent

Vehicle Towards Safe Driving With a New Database and Baseline Evaluations. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1171–1183.

[22] Chen, J. and Ran, X. (2019). Deep Learning With Edge Computing: A Review. *Proceedings of the IEEE*, 107(8):1655–1674.

[23] Chen, T., Ji, B., Ding, T., Fang, B., Wang, G., Zhu, Z., Liang, L., Shi, Y., Yi, S., and Tu, X. (2021a). Only Train Once: A One-Shot Neural Network Training And Pruning Framework. In *Advances in Neural Information Processing Systems*, volume 34, pages 19637–19651. Curran Associates, Inc.

[24] Chen, X., Zhu, J., Jiang, J., and Tsui, C.-Y. (2020b). Tight Compression: Compressing CNN Model Tightly Through Unstructured Pruning and Simulated Annealing Based Permutation. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6.

[25] Chen, Z., Xu, T.-B., Du, C., Liu, C.-L., and He, H. (2021b). Dynamical Channel Pruning by Conditional Accuracy Change for Deep Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):799–813.

[26] Chris Judy, Nanashi, Z. W. (2024). Event-based eye tracking - AIS2024 CVPR workshop.

[27] Ciravegna, G., Barbiero, P., Giannini, F., Gori, M., Lió, P., Maggini, M., and Melacci, S. (2023). Logic explained networks. *Artificial Intelligence*, 314:103822.

[28] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

[29] De Felice, G., Flores, A. C., De Santis, F., Santini, S., Schneider, J., Barbiero, P., and Termine, A. (2025). Causally reliable concept bottleneck models. *arXiv preprint arXiv:2503.04363*.

[30] De Santis, F., Ciravegna, G., Bich, P., Giordano, D., and Cerquitelli, T. (2025). V-cem: Bridging performance and intervenability in concept-based models. *The 3rd World Conference on eXplainable Artifical Intelligence*.

[31] De Tournemire, P., Nitti, D., Perot, E., Migliore, D., and Sironi, A. (2020). A Large Scale Event-based Detection Dataset for Automotive. *arXiv*.

[32] Debot, D., Barbiero, P., Giannini, F., Ciravegna, G., Diligenti, M., and Marra, G. (2024). Interpretable concept-based memory reasoning. *Advances in Neural Information Processing Systems*, 37:19254–19287.

[33] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[34] Dominici, G., Barbiero, P., Zarlenga, M. E., Termine, A., Gjoreski, M., Marra, G., and Langheinrich, M. (2024). Causal concept graph models: Beyond causal opacity in deep learning. *arXiv preprint arXiv:2405.16507*.

[35] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings of 2021 International Conference on Learning Representations (ICLR)*.

[36] Everingham, M., Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision*, 88(2):303–338.

[37] Falanga, D., Kleber, K., and Scaramuzza, D. (2020). Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40):1–15.

[38] Fei-Fei, L., Fergus, R., and Perona, P. (2004). Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 12 - Volume 12*, CVPRW '04, page 178, USA. IEEE Computer Society.

[39] Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

[40] Formanek, A. and Hadházi, D. (2019). Compressing Convolutional Neural Networks by L0 Regularization. In *2019 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, pages 155–162.

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

[42] Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A. J., Conradt, J., Daniilidis, K., and Scaramuzza, D. (2022). Event-Based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180.

[43] Gehrig, D., Gehrig, M., Hidalgo-Carrió, J., and Scaramuzza, D. (2020). Video to Events: Recycling Video Datasets for Event Cameras. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3583–3592. IEEE Computer Society.

[44] Gehrig, M., Millhäusler, M., Gehrig, D., and Scaramuzza, D. (2021). E-RAFT: Dense Optical Flow from Event Cameras. In *2021 International Conference on 3D Vision (3DV)*, pages 197–206. IEEE Computer Society.

[45] Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 Object Category Dataset. Technical Report 7694, California Institute of Technology.

[46] Gripenberg, G. (2003). Approximation by neural networks with a bounded number of nodes at each level. *Journal of Approximation Theory*, 122(2):260–266.

[47] Guliyev, N. J. and Ismailov, V. E. (2018). On the approximation by single hidden layer feedforward neural networks with fixed weights. *Neural Networks*, 98:296–304.

[48] Gulrajani, I. and Lopez-Paz, D. (2020). In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*.

[49] Guo, M., Huang, J., and Chen, S. (2017). Live demonstration: A 768 × 640 pixels 200Meps dynamic vision sensor. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–1.

[50] Ha, K., Chen, Z., Hu, W., Richter, W., Pillai, P., and Satyanarayanan, M. (2014). Towards wearable cognitive assistance. In *12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '14)*, pages 68–81.

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

[52] Hanin, B. and Sellke, M. (2018). Approximating Continuous Functions by ReLU Nets of Minimal Width.

[53] He, B., Li, H., Wu, S., Wang, D., Zhang, Z., Dong, Q., Xu, C., and Gao, F. (2021). FAST-Dynamic-Vision: Detection and Tracking Dynamic Objects with Event and Depth Sensing. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3071–3078.

[54] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[55] He, Y., Liu, P., Zhu, L., and Yang, Y. (2022). Filter Pruning by Switching to Neighboring CNNs With Good Attributes. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13.

[56] He, Y., Zhang, X., and Sun, J. (2017). Channel Pruning for Accelerating Very Deep Neural Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1398–1406.

[57] Hendrycks, D., Mazeika, M., and Dietterich, T. (2018). Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*.

[58] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

[59] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

[60] Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., and Le, Q. (2019). Searching for MobileNetV3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, Seoul, Korea (South). IEEE.

[61] Hsiao, S.-F. and Chang, H.-J. (2020). Sparsity-Aware Deep Learning Accelerator Design Supporting CNN and LSTM Operations. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4.

[62] Hu, Y., Binas, J., Neil, D., Liu, S.-C., and Delbruck, T. (2020a). DDD20 End-to-End Event Camera Driving Dataset: Fusing Frames and Events with Deep Learning for Improved Steering Prediction. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6.

[63] Hu, Y., Delbruck, T., and Liu, S.-C. (2020b). Learning to Exploit Multiple Vision Modalities by Using Grafted Networks. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 85–101.

[64] Hu, Y., Liu, H., Pfeiffer, M., and Delbruck, T. (2016). DVS Benchmark Datasets for Object Tracking, Action Recognition, and Object Recognition. *Frontiers in Neuroscience*, 10.

[65] Hu, Y., Liu, S. L., and Delbruck, T. (2021). v2e: From Video Frames to Realistic DVS Events. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1312–1321.

[66] Huang, C., Li, M., Cao, F., Fujita, H., Li, Z., and Wu, X. (2023). Are graph convolutional networks with random weights feasible? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2751–2768.

[67] Jaderberg, M., Simonyan, K., Zisserman, A., and kavukcuoglu, k. (2015). Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[68] Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K., and Tassiulas, L. (2023). Model Pruning Enables Efficient Federated Learning on Edge Devices. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10374–10386.

[69] Jocher, G., Chaurasia, A., and Qiu, J. (2023). YOLO by Ultralytics.

[70] Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. (2018). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR.

[71] Kim, E., Jung, D., Park, S., Kim, S., and Yoon, S. (2023). Probabilistic concept bottleneck models. *International Conference on Machine Learning*, pages 16521–16540.

[72] Kingma, D. P. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[73] Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization.

[74] Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. (2020). Concept bottleneck models. In *International conference on machine learning*, pages 5338–5348. PMLR.

[75] Kosko, B. (1994). Fuzzy systems as universal approximators. *IEEE Transactions on Computers*, 43(11):1329–1333.

[76] Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., Fernandez, G., Vojir, T., Hager, G., Nebehay, G., Pflugfelder, R., Gupta, A., Bibi, A., Lukezic, A., Garcia-Martin, A., Saffari, A., Petrosino, A., and Solis Montero, A. (2015). The Visual Object Tracking VOT2015 Challenge Results. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 564–586.

[77] Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

[78] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[79] Kusupati, A., Singh, M., Bhatia, K., Kumar, A., Jain, P., and Varma, M. (2018). FastGRNN: A Fast, Accurate, Stable and Tiny Kilobyte Sized Gated Recurrent Neural Network. In *32nd International Conference on Neural Information Processing Systems (NIPS'18)*, pages 9031–9042.

[80] Lagorce, X., Orchard, G., Galluppi, F., Shi, B. E., and Benosman, R. B. (2017). HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1346–1359.

[81] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[82] Lenz, G., Chaney, K., Shrestha, S. B., Oubari, O., Picaud, S., and Zarrella, G. (2021). Tonic: Event-based datasets and transformations. Zenodo.

[83] Li, E., Zeng, L., Zhou, Z., and Chen, X. (2020). Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing. *IEEE Transactions on Wireless Communications*, 19(1):447–457.

[84] Li, H., Liu, H., Ji, X., Li, G., and Shi, L. (2017). CIFAR10-DVS: An Event-Stream Dataset for Object Classification. *Frontiers in Neuroscience*, 11.

[85] Li, J., Dong, S., Yu, Z., Tian, Y., and Huang, T. (2019). Event-Based Vision Enhanced: A Joint Detection Framework in Autonomous Driving. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1396–1401.

[86] Liang, S. and Li, Y. (2017). Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*.

[87] Lichtsteiner, P., Posch, C., and Delbruck, T. (2006). A 128 X 128 120db 30mw asynchronous vision sensor that responds to relative intensity change. In *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, pages 2060–2069.

[88] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 740–755.

[89] Liu, Y., Zhang, Y., Wang, Y., Hou, F., Yuan, J., Tian, J., Zhang, Y., Shi, Z., Fan, J., and He, Z. (2024a). A Survey of Visual Transformers. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7478–7498.

[90] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[91] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., and Tegmark, M. (2024b). KAN: Kolmogorov-Arnold Networks.

[92] Loshchilov, I. and Hutter, F. (2019). Decoupled Weight Decay Regularization.

[93] Louizos, C., Welling, M., and Kingma, D. P. (2017). Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

[94] Lu, Y. and Lu, J. (2020). A universal approximation theorem of deep neural networks for expressing probability distributions. In *Advances in Neural Information Processing Systems*, volume 33, pages 3094–3105. Curran Associates, Inc.

[95] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In Lin, D., Matsumoto, Y., and Mihalcea, R., editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

[96] Mahinpei, A., Clark, J., Lage, I., Doshi-Velez, F., and Pan, W. (2021). Promises and pitfalls of black-box concept learning models. *arXiv preprint arXiv:2106.13314*.

[97] Manita, O. A., Peletier, M. A., Portegies, J. W., Sanders, J., and Senen-Cerda, A. (2022). Universal approximation in dropout neural networks. *J. Mach. Learn. Res.*, 23(1).

[98] Marconato, E., Passerini, A., and Teso, S. (2022). Glancenets: Interpretable, leak-proof concept-based models. *Advances in Neural Information Processing Systems*, 35:21212–21227.

[99] Mehta, S. and Rastegari, M. (2022). MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer. In *Proceedings of 2022 International Conference on Learning Representations (ICLR)*.

[100] Merenda, M., Porcaro, C., and Iero, D. (2020). Edge Machine Learning for AI-Enabled IoT Devices: A Review. *Sensors*, 20(9):2533.

[101] Miao, S., Chen, G., Ning, X., Zi, Y., Ren, K., Bing, Z., and Knoll, A. (2019). Neuromorphic Vision Datasets for Pedestrian Detection, Action Recognition, and Fall Detection. *Frontiers in Neurorobotics*, 13.

[102] Mitrokhin, A., Fermüller, C., Parameshwara, C., and Aloimonos, Y. (2018). Event-Based Moving Object Detection and Tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9.

[103] Montoye, R. K., Hokenek, E., and Runyon, S. L. (1990). Design of the IBM RISC System/6000 floating-point execution unit. *IBM Journal of Research and Development*, 34(1):59–70.

[104] Mueggler, E., Gallego, G., Rebecq, H., and Scaramuzza, D. (2018). Continuous-Time Visual-Inertial Odometry for Event Cameras. *IEEE Transactions on Robotics*, 34(6):1425–1440.

[105] Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., and Scaramuzza, D. (2017). The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research*, 36(2):142–149.

[106] Niu, W., Ma, X., Lin, S., Wang, S., Qian, X., Lin, X., Wang, Y., and Ren, B. (2020). PatDNN: Achieving Real-Time DNN Execution on Mobile Devices with Pattern-based Weight Pruning. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, pages 907–922, New York, NY, USA. Association for Computing Machinery.

[107] Nonnenmacher, M., Pfeil, T., Steinwart, I., and Reeb, D. (2021). SOSP: Efficiently Capturing Global Correlations by Second-Order Structured Pruning. In *International Conference on Learning Representations*.

[108] Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. (2015). Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience*, 9.

[109] Perot, E., de Tournemire, P., Nitti, D., Masci, J., and Sironi, A. (2020). Learning to Detect Objects with a 1 Megapixel Event Camera. In *Adv. Neural Inform. Process. Syst.*, volume 33, pages 16639–16652.

[110] Persand, K., Anderson, A., and Gregg, D. (2021). Taxonomy of Saliency Metrics for Channel Pruning. *IEEE Access*, 9:120110–120126.

[111] Peste, A., Iofinova, E., Vladu, A., and Alistarh, D. (2021). AC/DC: Alternating Compressed/DeCompressed Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 8557–8570.

[112] Pinkus, A. (1999). Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195.

[113] Poeta, E., Ciravegna, G., Pastor, E., Cerquitelli, T., and Baralis, E. (2023). Concept-based explainable artificial intelligence: A survey. *arXiv preprint arXiv:2312.12936*.

[114] Posch, C., Matolin, D., and Wohlgenannt, R. (2011). A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS. *IEEE Journal of Solid-State Circuits*, 46(1):259–275.

[115] Prono, L., Bich, P., Boretti, C., Mangia, M., Pareschi, F., Rovatti, R., and Setti, G. (2025). A multiply-and-max/min neuron paradigm for aggressively prunable deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems.*

[116] Prono, L., Mangia, M., Marchioni, A., Pareschi, F., Rovatti, R., and Setti, G. (2020). Deep Neural Oracle With Support Identification in the Compressed Domain. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(4):458–468.

[117] Prono, L., Mangia, M., Pareschi, F., Rovatti, R., and Setti, G. (2022). A Non-conventional Sum-and-Max based Neural Network layer for Low Power Classification. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 712–716.

[118] Rebecq, H., Gehrig, D., and Scaramuzza, D. (2018). ESIM: An Open Event Camera Simulator. In *Proceedings of The 2nd Conference on Robot Learning*, pages 969–982. PMLR.

[119] Rebecq, H., Horstschaefer, T., Gallego, G., and Scaramuzza, D. (2017). EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time. *IEEE Robotics and Automation Letters*, 2(2):593–600.

[120] Reddy, K. K. and Shah, M. (2013). Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981.

[121] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

[122] Ren, A., Zhang, T., Ye, S., Li, J., Xu, W., Qian, X., Lin, X., and Wang, Y. (2019). ADMM-NN: An Algorithm-Hardware Co-Design Framework of DNNs Using Alternating Direction Methods of Multipliers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, pages 925–938, New York, NY, USA. Association for Computing Machinery.

[123] Rodríguez-Gomez, J. P., Eguíluz, A. G., Martínez-de Dios, J. R., and Ollero, A. (2020). Asynchronous event-based clustering and tracking for intrusion monitoring in UAS. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8518–8524.

[124] Rodríguez-Gómez, J. P., Eguíluz, A. G., Martínez-De Dios, J. R., and Ollero, A. (2021). Auto-Tuned Event-Based Perception Scheme for Intrusion Monitoring With UAS. *IEEE Access*, 9:44840–44854.

[125] Rosenbrock, H. H. (1960). An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3):175–184.

[126] Rovatti, R. (1998). Fuzzy piecewise multilinear and piecewise linear systems as universal approximators in sobolev norms. *IEEE Transactions on Fuzzy Systems*, 6(2):235–249.

[127] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215.

[128] Sanh, V. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[129] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations (ICLR 2015)*.

[130] Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., and Benosman, R. (2018). HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1731–1740, Salt Lake City, UT, USA. IEEE.

[131] Stoffregen, T., Scheerlinck, C., Scaramuzza, D., Drummond, T., Barnes, N., Kleeman, L., and Mahony, R. (2020). Reducing the Sim-to-Real Gap for Event Cameras. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 534–549.

[132] Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852.

[133] Tan, C. and Liu, J. (2024). Improving Knowledge Distillation With a Customized Teacher. *IEEE Transactions on Neural Networks and Learning Systems*, 35(2):2290–2299.

[134] Tan, C. M. J. and Motani, M. (2020). DropNet: Reducing Neural Network Complexity via Iterative Pruning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9356–9366. PMLR.

[135] Tanielian, U., Sangnier, M., and Biau, G. (2021). Approximating Lipschitz continuous functions with GroupSort neural networks. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS 2021)*.

[136] Tapiador-Morales, R., Maro, J.-M., Jimenez-Fernandez, A., Jimenez-Moreno, G., Benosman, R., and Linares-Barranco, A. (2020). Event-Based Gesture Recognition through a Hierarchy of Time-Surfaces for FPGA. *Sensors*, 20(12):3404.

[137] Tomy, A., Paigwar, A., Mann, K. S., Renzaglia, A., and Laugier, C. (2022). Fusing Event-based and RGB camera for Robust Object Detection in Adverse Conditions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 933–939.

[138] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ukasz Kaiser, Ł., and Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[139] Vidal, A. R., Rebecq, H., Horstschaefer, T., and Scaramuzza, D. (2018). Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001.

[140] Vishwakarma, H., Lin, H., and Vinayak, R. (2023). Human-in-the-loop out-of-distribution detection with false positive rate control. In *NeurIPS Workshop on Adaptive Experimental Design and Active Learning in the Real World*.

[141] Wang, L.-X. (1992). Fuzzy systems are universal approximators. In *[1992 Proceedings] IEEE International Conference on Fuzzy Systems*, pages 1163–1170.

[142] Wang, Y. and Bi, X. (2024). The Application of Computer Vision Target Recognition Technology in Autonomous Driving. In *2024 3rd International Conference on Artificial Intelligence and Autonomous Robot Systems (AIARS)*, pages 519–524.

[143] Woo, S., Park, J., Lee, J.-Y., and Kweon, I. S. (2018). CBAM: Convolutional Block Attention Module. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, pages 3–19, Cham. Springer International Publishing.

[144] Wu, Z., Xiao, M., Fang, C., and Lin, Z. (2024). Designing universally-approximating deep neural networks: A first-order optimization approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–17.

[145] Yang, C., Wang, Y., and Owens, J. D. (2015). Fast Sparse Matrix and Sparse Vector Multiplication Algorithm on the GPU. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 841–847.

[146] Ye, C., Mitrokhin, A., Fermüller, C., Yorke, J. A., and Aloimonos, Y. (2020). Unsupervised Learning of Dense Optical Flow, Depth and Egomotion with Event-Based Sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5831–5838.

[147] Youssef, I., Mutlu, M., Bayat, B., Crespi, A., Hauser, S., Conradt, J., Bernardino, A., and Ijspeert, A. (2020). A Neuro-Inspired Computational Model for a Visually Guided Robotic Lamprey Using Frame and Event Based Cameras. *IEEE Robotics and Automation Letters*, 5(2):2395–2402.

[148] Zarlenga, M. E., Barbiero, P., Ciravegna, G., Marra, G., Giannini, F., Diligenti, M., Precioso, F., Melacci, S., Weller, A., Lio, P., et al. (2022). Concept embedding models. In *NeurIPS 2022-36th Conference on Neural Information Processing Systems*.

[149] Zhang, B., Jiang, D., and Wang, L. (2022). Rethinking Lipschitz Neural Networks and Certified Robustness: A Boolean Function Perspective. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*.

[150] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). Mixup: Beyond Empirical Risk Minimization. In *Proceedings of 2018 International Conference on Learning Representations (ICLR)*.

[151] Zhang, H., Hao, K., Gao, L., Wei, B., and Tang, X. (2023). Optimizing Deep Neural Networks Through Neuroevolution With Stochastic Gradient Descent. *IEEE Transactions on Cognitive and Developmental Systems*, 15(1):111–121.

[152] Zhang, H., Liu, L., Zhou, H., Hou, W., Sun, H., and Zheng, N. (2021a). AKECP: Adaptive Knowledge Extraction from Feature Maps for Fast and Efficient Channel Pruning. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, pages 648–657, New York, NY, USA. Association for Computing Machinery.

[153] Zhang, J.-F., Lee, C.-E., Liu, C., Shao, Y. S., Keckler, S. W., and Zhang, Z. (2021b). SNAP: An Efficient Sparse Neural Acceleration Processor for Unstructured Sparse Deep Neural Network Inference. *IEEE Journal of Solid-State Circuits*, 56(2):636–647.

[154] Zhang, S. and Stadie, B. C. (2020). One-Shot Pruning of Recurrent Neural Networks by Jacobian Spectrum Evaluation. In *International Conference on Learning Representations (ICLR 2020)*.

[155] Zhang, S.-Q. and Zhou, Z.-H. (2022). Theoretically provable spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pages 19345–19356. Curran Associates, Inc.

[156] Zhang, T., Chowdhery, A., Bahl, P. V., Jamieson, K., and Banerjee, S. (2015). The Design and Implementation of a Wireless Video Surveillance System. In *21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*, pages 426–438.

[157] Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., and Chen, J. (2024). DETRs Beat YOLOs on Real-time Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16965–16974.

[158] Zhou, D.-X. (2020). Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*, 48(2):787–794.

[159] Zhu, A. Z., Thakur, D., Özaslan, T., Pfrommer, B., Kumar, V., and Daniilidis, K. (2018a). The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039.

[160] Zhu, A. Z., Yuan, L., Chaney, K., and Daniilidis, K. (2018b). EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras. In *Robotics: Science and Systems XIV*.

[161] Zhu, A. Z., Yuan, L., Chaney, K., and Daniilidis, K. (2019). Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 989–997.

[162] Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2023). Object Detection in 20 Years: A Survey. *Proceedings of the IEEE*, 111(3):257–276.

[163] Zullich, M., Medvet, E., Pellegrino, F. A., and Ansuini, A. (2021). Speeding-up pruning for Artificial Neural Networks: Introducing Accelerated Iterative Magnitude Pruning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3868–3875.

[164] Zuo, Y.-F., Yang, J., Chen, J., Wang, X., Wang, Y., and Kneip, L. (2022). DEVO: Depth-Event Camera Visual Odometry in Challenging Conditions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2179–2185.

Appendix A

MAM: Supplementary Information

A.1 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 2 FLOPs per weight (1 multiplication and 1 addition). Regarding MAM operation, I count 1 multiplication and 2 additions (one comparison for the maximum and one for the minimum) per weight, with a total of 3 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 operation (FMA) [103] 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) I know beforehand which inputs are associated to 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. Firstly, I 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.

Appendix B

V-CEM: Supplementary Information

B.1 Loss derivation

This appendix provides the complete derivation of the loss function that V-CEM is trained to approximate:

$$\log p(c, y|x) = \log \int_{\mathbf{C}} \frac{p(x, c, \mathbf{c}, y)}{p(x)} d\mathbf{c} \tag{B.1}$$

$$= \log \int_{\mathbf{C}} p(c|x)p(\mathbf{c}|c)p(y|\mathbf{c})d\mathbf{c} \tag{B.2}$$

$$= \log \int_{\mathbf{C}} \frac{q(\mathbf{c}|x, c)}{q(\mathbf{c}|x, c)} p(c|x)p(\mathbf{c}|c)p(y|\mathbf{c})d\mathbf{c} \tag{B.3}$$

$$= \log E_q \left[\frac{p(c|x)p(\mathbf{c}|c)p(y|\mathbf{c})}{q(\mathbf{c}|x, c)} \right] \tag{B.4}$$

$$\geq E_q \left[\log \frac{p(c|x)p(\mathbf{c}|c)p(y|\mathbf{c})}{q(\mathbf{c}|x, c)} \right] \tag{B.5}$$

$$= - E_q \left[\log \frac{q(\mathbf{c}|x, c)}{p(\mathbf{c}|c)} \right] + \log p(c|x) + E_q \left[\log p(y|\mathbf{c}) \right] \tag{B.6}$$

I begin by re-expressing the target conditional probability $p(x, y \mid \mathbf{c})$ through marginalization over $\mathbf{C}$ and factorizing the joint distribution $p(x, c, \mathbf{c}, y)$ according to the generative process illustrated in Figure 6.1c. Next, to amortize inference I introduce an approximate posterior distribution $q(\mathbf{c}|x, c)$ (Eq. B.3).

By applying Jensen's inequality, I obtain a lower bound on the log-likelihood, known as the ELBO, as shown in Eq. B.5. Finally, Eq. B.6 expands the ELBO into three terms: the first term is the negative KL divergence between $q(\mathbf{c}|x,c)$ and $p(\mathbf{c}|c)$, which measures the difference between the approximate posterior and the true prior; the second term is the log-likelihood of c, and the third term is the expected log-likelihood of y.

B.2 Prior matching formulation

An important assumption I make, which is a standard assumption for concept based methodologies, is that the different concepts, and therefore the different concepts embeddings, are independent one another. Therefore, $q(\mathbf{c}|x,c) = \prod_{j=1}^{k} q(\mathbf{c}_j|x,c_j)$ and $p(\mathbf{c}|c) = \prod_{j=1}^{k} p(\mathbf{c}_j|c_j)$. This allows to rewrite the *Prior Matching* term as the sum of KL divergences between the approximate posterior and the true prior of each concept:

$$E_q\left[\log \frac{q(\mathbf{c}|x,c)}{p(\mathbf{c}|c)}\right] = \int_{\mathbf{C}} q(\mathbf{c}|x,c) \log \frac{q(\mathbf{c}|x,c)}{p(\mathbf{c}|c)} d\mathbf{c} \tag{B.7}$$

$$= \sum_{j=1}^{k} \int_{\mathbf{C}} \prod_{i=1}^{k} q(\mathbf{c}_i|x,c_i) \log \frac{q(\mathbf{c}_j \mid x,c_j)}{p(\mathbf{c}_j \mid c_j)} d\mathbf{c} \tag{B.8}$$

$$= \sum_{j=1}^{k} \int_{\mathbf{C}} q(\mathbf{c}_j|x,c_j) \log \frac{q(\mathbf{c}_j \mid x,c_j)}{p(\mathbf{c}_j \mid c_j)} d\mathbf{c} \tag{B.9}$$

$$= \sum_{j=1}^{k} E_q\left[\log \frac{q(\mathbf{c}_j|x,c_j)}{p(\mathbf{c}_j|c_j)}\right] \tag{B.10}$$

The prior is modeled as a mixture, governed by the function $\delta(\cdot)$, which selects the appropriate normal distribution based on the value of c_j. As a result, the KL divergence is computed differently depending on whether c_j is active or inactive. When $c_j = 1$, it quantifies the divergence between the approximate posterior and the corresponding normal distribution in the prior for $c_j = 1$. Similarly, when $c_j = 0$, it measures the divergence between the approximate posterior and the prior distribution associated with $c_j = 0$. Defining

$$\mu_j = \begin{cases} \mu_j^+ & \text{if } c_j = 1, \\ \mu_j^- & \text{if } c_j = 0 \end{cases}$$

allows to rewrite the *Prior Matching* term as:

$$E_q\left[\log \frac{q(\mathbf{c}|x,c)}{p(\mathbf{c}|c)}\right] = \frac{1}{2}\sum_{j=1}^{k}\left[||\hat{\mu}_j(x,c) - \mu_j||^2 + \sum_{z=1}^{m}\sigma_{jz}^2(x,c) - m - \sum_{z=1}^{m}\log\,\sigma_{jz}^2(x,c)\right]$$

where $\sigma_{jz}^2(x,c)$ denotes the variance of the concept embedding j for the latent dimension z.

B.3 Concept Representation Cohesiveness

In this work I introduce a novel metric to compute the Concept Representation Cohesiveness, a metric to comprehend how spread the representation are in the concept space which is particularly useful to assess how prone a model is to concept leakage and in turn how likely I can correctly perform concept intervention also OOD. Recalling from Section 6.5, Equation 6.5, I defined CRC as:

$$CRC = \frac{1}{|C|}\sum_{i=0}^{|C|} s_i(\mathbf{c}_i, c_i)$$

More specifically, I now define how to compute s_i (here and in the following I drop the dependency from $\mathbf{c}_i$, c_i):

$$s_i = \frac{1}{2}\left(\frac{b_i^+ - a_i^+}{\max(b_i^+, a_i^+)} + \frac{b_i^- - a_i^-}{\max(b_i^-, a_i^-)}\right),$$

$$a_i^+ = \frac{1}{|\mathcal{C}_i^+|}\sum_{j\in\mathcal{C}_i^+}\frac{1}{|\mathcal{C}_i^+ - 1|}\sum_{k\in\mathcal{C}_i^+, k\neq j}||\mathbf{c}_{ij} - \mathbf{c}_{ik}||_1$$

$$b_i^+ = \frac{1}{|\mathcal{C}^+ - 1|}\sum_{j\in\mathcal{C}_i^+}\frac{1}{|\mathcal{C}^-|}\sum_{k\in\mathcal{C}_i^-}||\mathbf{c}_{ij} - \mathbf{c}_{ik}||_1$$

and where $\mathcal{C}_i^+\mathcal{C}_i^-$ are the set of sample indexes associated to positive and negative concept prediction for concept i and are thus computed as: $\mathcal{C}_i^+ = \mathbb{1}_{c_i>0.5}$ and $\mathcal{C}_i^- = \mathbb{1}_{c_i\leq 0.5}$.

B.4 Ablation on λ_p variation

In this work I introduce a scaling factor $\lambda_p \in [0, \infty)$ to regulate the *Prior Matching* term, allowing fine-grained control over the model's behavior. Increasing λ_p progressively aligns V-CEM with a standard CBM, whereas setting $\lambda_p = 0$ eliminates constraints on concept embedding generation, making the model function similarly to a CEM. In this appendix I show how modifying λ_p modifies the model performance.

In Figure B.1, I report the ID performance of V-CEM on CEBaB and IMDB as an example of performance datasets when modifying λ_p. The observed transition aligns with expectations: for $\lambda_p = 0$, the model achieves good performance similar to CEM, while increasing λ_p leads to performance degradation, making it more similar to that of CBMs.

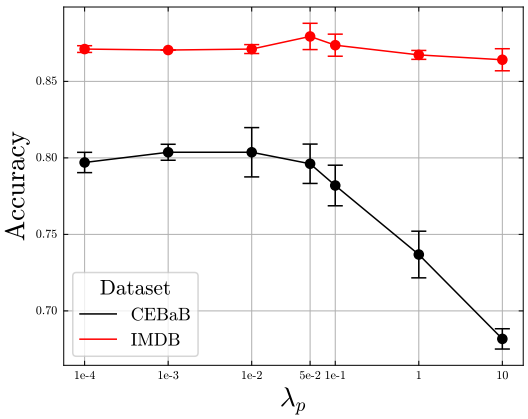

Fig. B.1 Variation in V-CEM's ID accuracy across different values of λ_p on the CEBaB and CelebA datasets. Figure from [30].

Similar results can be observed in Figure B.2, where for low values of λ_p, responsiveness to interventions is weaker -a characteristic typical of CEM- while it improves as λ_p increases, approaching the responsiveness of CBMs. To balance both in-distribution performance and responsiveness to interventions, I set $\lambda_p = 0.05$ in this manuscript.

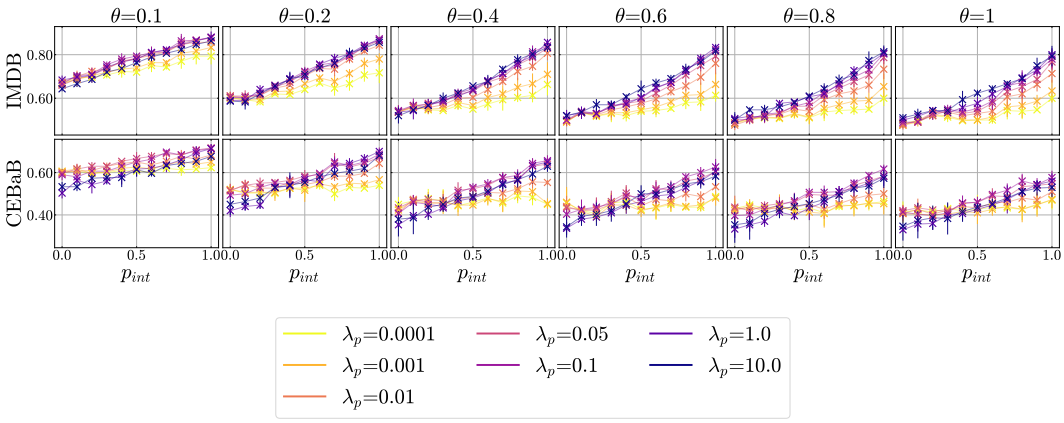

Fig. B.2 Impact of interventions on V-CEM's OOD accuracy across different λ_p values. Figure from [30].

B.5 Dataset details

In this appendix, I provide additional details on the datasets used to evaluate the performance of the tested models, followed by a description of the training procedure.

B.5.1 Datasets

MNIST Even/Odd.

MNIST Even/Odd is a binary classification dataset derived from MNIST, where digits, used as concept labels, are categorized as either even or odd. It consists of 60 000 training images and 10 000 test images, each of size 28×28 and in grayscale. All images were converted to a three-channel format and rescaled to 224×224.

MNIST Addition.

MNIST Addition is constructed by pairing two MNIST digits (used as concepts) and assigning a label equal to the sum of their individual values. The dataset retains the original MNIST structure, containing 60 000 training samples and 10 000 test samples. Each input is a grayscale image formed by concatenating

two MNIST digits side by side. Also in this case, images were converted to a three-channel format and rescaled to 224×224.

CelebA.

CelebA is a large-scale facial attribute dataset containing over 200 000 images of celebrities, each of size 178×218. The dataset is divided into training, validation, and test sets. I use the following attributes as concepts: `No Beard`, `Young`, `Attractive`, `Mouth Slightly Open`, `Smiling`, `Wearing Lipstick`, and `High Cheekbones`, as they are the most balanced attributes in the dataset. The task is a multi-class classification problem, where the goal is to predict the attributes `Wavy Hair`, `Black Hair`, and `Male`. All images are already in RGB format and are rescaled to 224×224.

CEBaB.

CEBaB is a dataset designed to study the causal effects of real-world concepts on NLP models. It includes short restaurant reviews annotated with sentiment ratings at both the overall review level (positive, neutral, and negative reviews) and for four dining experience aspects, which are used as concept labels: `Good Food`, `Good Ambiance`, `Good Service`, and `Good Noise`.

IMDB.

The IMDB dataset consists of 50 000 movie reviews labeled as either positive or negative. To predict the overall review sentiment, I use four interpretable concepts: `Acting`, `Cinematography`, `Emotional arousal`, and `Storyline`.

For datasets that do not provide a validation set, I randomly removed 10% of the training data to create a validation set.

B.5.2 Training details

All models were trained up to 500 epochs, employing an early stopping criterion with a patience of 20 epochs. The Adam optimizer was used along with a

learning rate scheduler that reduced the learning rate by a factor of 0.1 every 100 epochs. The initial learning rate was dataset-specific: $2e{-}3$ for MNIST Even/Odd and MNIST Addition, $1e{-}4$ for CelebA, $5e{-}4$ for CEBaB, and $1e{-}2$ for IMDB. All baseline models were trained with default hyperparameters. Both Prob-CBM and CEM were trained following the *RandInt* technique proposed in [148], setting it to 0.25 for CEM and to 0.5 for Prob-CBM, as suggested in the respective papers. To ensure a fair comparison across different methodologies, I applied the *RandInt* technique during the training of CBM+MLP, CBM+Linear, and V-CEM. Specifically, I set the intervention probability to 0.25 for these approaches. For V-CEM, random interventions were introduced starting from the 20th epoch for the CelebA dataset (given the larger size of the training-set), while for all other datasets, they were applied from the 3rd epoch onward.

For the V-CEM model, the *Prior Matching* term was scaled using a factor of $\lambda = 0.05$. As for Prob-CBM and CEM, I used a concept embedding dimension of 16. Each model was trained using three different random seeds.

B.6 Concept accuracy

In this appendix, I report the concept accuracy values for all models and datasets. The results reported in Table B.1 confirm that, in terms of concept accuracy, the performance of all models is comparable, with V-CEM being on average the best (despite overlapping standard deviations).

Table B.1 Concept accuracy comparison across different datasets in ID settings.

	MNIST E/O	MNIST+	CelebA	CEBaB	IMDB
CBM+Linear	99.44 $\pm$0.01	95.24 $\pm$0.00	83.02 $\pm$0.03	80.33 $\pm$0.74	84.41 $\pm$0.05
CBM+MLP	99.46 $\pm$0.01	95.08 $\pm$0.04	82.91 $\pm$0.05	80.81 $\pm$0.79	84.49 $\pm$0.12
CEM	99.35 $\pm$0.03	94.91 $\pm$0.05	82.77 $\pm$0.04	79.51 $\pm$0.17	83.30 $\pm$0.18
Prob-CBM	99.18 $\pm$0.07	95.08 $\pm$0.09	82.93 $\pm$0.03	80.70 $\pm$0.66	83.48 $\pm$0.41
V-CEM	99.49 $\pm$0.00	95.22 $\pm$0.09	83.04 $\pm$0.01	80.37 $\pm$0.52	84.16 $\pm$0.19

B.7 ID Interventions

In addition to demonstrating strong responsiveness to interventions in OOD settings, V-CEM maintains high accuracy even when interventions occur in ID settings. As shown in Figure B.3, the performance of the various models remains stable across different datasets. This stability is primarily attributed to the high concept accuracy (Table B.1) achieved by these models, which limits the potential for further improvement following interventions. Notably, in MNIST+, accuracy increases linearly with the intervention probability (p_{int}) for all the methodologies. Conversely, for CBM+MLP and CBM+Linear on the CEBaB and CelebA datasets, performance slightly declines post-intervention, likely due to the lower concept accuracy in these datasets (approximately 80%). This observation highlights the greater robustness of concept embedding methodologies to interventions in such scenarios.

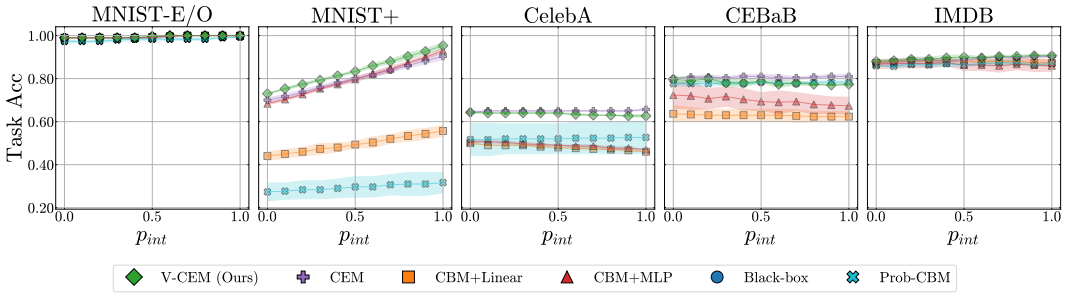

Fig. B.3 Mean and standard deviation of task accuracy with random interventions at probability p_{int} across different models and datasets without noise (ID settings). Figure from [30].