ElegansAI: how a biological neural network would compare with artificial networks?

Francesco Bardozzo (fbardozzo@unisa.it)
University of Salerno https://orcid.org/0000-0003-0199-6623
Andrea Terlizzi
University of Salerno
Pietro Lio
University of Cambridge https://orcid.org/0000-0002-0540-5053
Roberto Tagliaferri
University of Salerno https://orcid.org/0000-0001-8134-9025

Article

Keywords:

Posted Date: June 13th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3005708/v2

Additional Declarations: There is **NO** Competing Interest.

ElegansAI: how a biological neural network would compare with artificial networks?

Francesco Bardozzo*, Andrea F. Terlizzi *, Pietro Lió †, Roberto Tagliaferri*
*NeuroneLab - DISAMIS - University of Salerno - Italy
† Computer Laboratory - University of Cambridge - United Kingdom

***Abstract*—This paper presents ElegansAI, a neural network model that leverages the connectome topology of the *Caenorhabditis elegans* to design and generate advanced learning systems. The objective of this approach is to integrate the intricate circuitry of biological neuronal networks into artificial ones, with the aim of exploring the advantages of incorporating bio-plausible connectome topology in deep learning models. ElegansAI outperforms randomly wired tensor networks, simulated bio-plausible networks, and state-of-the-art models such as transformers and attention-enforced autoencoders. The models achieve a top-1 accuracy of 99.99% on Cifar10 and 99.84% on MNIST Unsup in supervised image classification tasks and unsupervised handwritten digit reconstruction, respectively. The proposed method offers a unique approach to designing and generating connectome-inspired learning systems that harness the functional distribution of biological neuron circuitry. It is shown how bio-plausible structures integrated into artificial neural networks efficiently tackle complex tasks by evaluating evolutionary optimized neuronal motifs.**

***Index Terms*—neuromorphic neural network, bio-plausible AI, multi-dyadic effect, network motif, *C.elegans*, connectomic**

I. Introduction

Over the past decades, scientists have been developing algorithms and machines that take inspiration from neuronal communication mechanisms and nervous system structures. Artificial intelligence (AI) is a broad field with no single definition, encompassing research topics that range from symbolic-reasoning-oriented algorithms to cognitive simulation and neuromorphic machines, ultimately leading to neural networks. These connectionist-oriented models focus on network-based architectures capable of learning from examples and solving various tasks with reasonable generalization capacity. Although these modern problem-solving approaches are widely recognized and applied within the scientific community, there remains ample room for improvement. Our research is focused on the connectome, the structural organization of natural neural circuits, which plays a fundamental role in shaping the behavior of living organisms. Much research has suggested that the connectome neural connections are optimized by evolutionary pressure [1]. Thus, the next logical step is to investigate whether this type of optimized structure can be harnessed to improve the performance of learning algorithms structured as neural networks. For this reason, this paper introduces ElegansAI, a neural network model designed *ex novo*, that leverages the connectome topology of *Caenorhabditis elegans* (*C.elegans*), a small nematode.

a) Related works and critical points: The integration of biological features and structures, such as single neurons functioning through activation functions, brain behaviors, and connectomes, into artificial learning systems has been a long-standing scientific pursuit. A recent editorial in *Nature Machine Intelligence* [2] advocates for an approach to artificial intelligence that aims to better integrate bio-physical information. However, it is worth noting that biological learning systems are currently too complex to be efficiently represented by our knowledge and machines [3]. Effectively, the development of bio-inspired neural models typically may require a balance between operational simplifications and the characteristic aspects of the systems themselves [4], [5]. Despite that, main attempts to develop artificial learning networks by examining and replicating bio-inspired mechanisms can be categorized into three algorithmic approaches. The first approach involves Spike Neural Networks (SNNs) which simulate information communication in the nervous system via spike diffusion [4], [5]. The second approach is focused on Deep Neural Networks (DNNs), which, in a certain sense, enrich synaptic relations by backward-updating learned information [6]. Typically SNNs and DNNs are compared by means of their performance and computational costs [7]. The third approach, which is hybrid, combines SNNs with DNNs by incorporating neural dynamics and time-dependent plasticity features into traditional deep learning paradigms, as shown by recent studies [8], [9], [10]. Concurrently, some studies argue that in DNNs the training through backpropagation poorly approximates brain function [11], [12], while others integrate backpropagation into SNNs models [13], [14], [15], [16]. On one hand, SNNs suggest their suitability for specific applications, but a more universal approach that could be applied to a wider range of problems and applications is still lacking [17], [18]. One of the primary criticism of DNNs is their requirement for a large number of neurons and parameters to enhance learning capacity [11], and their lack of architectural and dimensional bio-plausibility [19]. Despite less mimicking bio-inspired models, DNNs have demonstrated broad effectiveness across various application domains, far outperforming most other machine learning methods in both supervised and unsupervised settings, and continuously evolving towards better architectures. As an example, recent literature for many supervised tasks like image classification has shifted from systems based on convolutional models [20], [21], [22] to attention-based transformers [23], [24], [25], [26], [27], [28]. On the other side, unsupervised reconstruction and/or denoising problems still rely on autoencoder-like architectures [29], [30] or encoder-decoder structures [31], [32], [33]. Concerning the connections

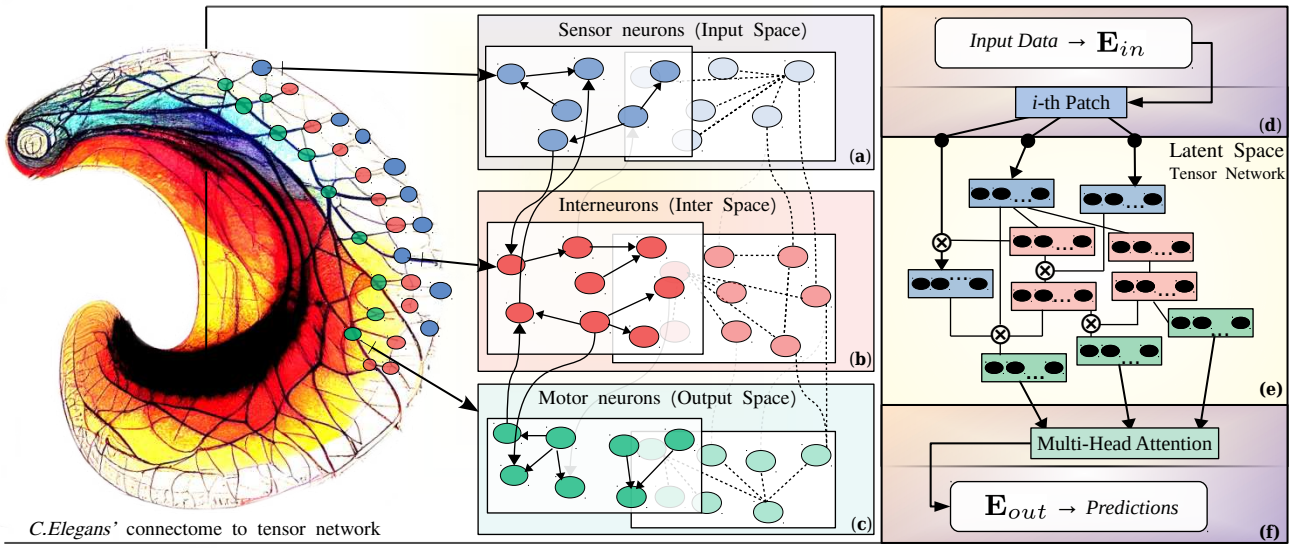

Fig. 1. The connectome of *C.elegans* is represented as a fully connected graph with two overlapping layers, where the solid edges represent chemical and directional synapses and the dashed edges represent electrical and undirected ones. The sensor neurons are represented in blue (Box **(a)**), while interneurons are represented in red (Box **(b)**). Finally, the motor neurons are represented in green (Box **(c)**). The blocks **(d-e-f)** describe the general structure of *ElegansAI*. In Box **(d)**, the first part of the so-called external operational environment ($\mathbf{E_{in}}$) of *ElegansAI* is shown. In detail, $\mathbf{E_{in}}$ is an encoder that may vary from the different deep learning tasks and generates the feature maps in input to the sensor-tensors space. In Box **(e)**, the Tensor Network TN is the resulting layered model produced starting from the reference graph. The TN is the core of the model and it is projected into the middle of the operational environment (in between $\mathbf{E_{in}}$ and $\mathbf{E_{out}}$). The TN takes the configuration of a directed acyclic graph and it is depicted as the latent space of our models. Solid lines into the TN show directed functional associations between tensor unit neurons. The $\otimes$ shows the skip connection by multiplication of the previous tensor units in multiple edge connections. In the output from the TN, the motor unit tensors are collected by tensor stacking and provided to the Multi-Head Attention layer. In turn, the external environment $\mathbf{E_{out}}$ is proximal to the targets (Box **(f)**). The latter is designed with a Multi-Head Attention layer in input to a tensorial module, called *feature condenser* (see also $\mathbf{E_{out}}$ classifier/decoder blue boxes **(e)** and **(c)** of Figure 7 and 8, respectively).

between deep learning and biological neural networks, a recent study [34] showed that a combination of fully convolutional layers with 1-dimensional causal convolutions, consisting of five to eight layers and using up to 1024 artificial neurons, can effectively emulate the learning behavior of an individual biological neuron. It can be argued that the evolution of neural network design has drawn inspiration from biological systems, particularly focusing on the functioning of single neurons / computational units and their activation functions [35], and has even led to more complex connectome-inspired models which focus to connections between groups of several computational units [36], [22], [21], [23]. From a certain perspective, the shift towards neural network models that emphasize the number of connections between simple units and their optimized organization can be seen as a reapproach to biological neural systems. Indeed, the nervous systems and connectomes of animals and insects are well-known to hold promise for developing optimized learning systems [37], [38], [39]. Moreover, it should be noted that in early 2017, Nick Bostrom [40], a philosopher of science, already identified the nematode worm *C.elegans* as a potential model for developing connectome-based artificial intelligence due to its relatively simple yet fully mapped nervous system. In this direction, Sardi et al. [41] show that using online learning mechanisms inspired by brain functioning, such as increased neuronal training frequency, can significantly outperform conventional machine learning methods in the context of online learning. However, it is only in more recent studies, such as Yan et al. [15], that it has been demonstrated that a sparse variant of the backpropagation algorithm can create a bionic structure that resembles the nervous system of *C.elegans*. On the other hand, from a neuroscientific point of view, it is currently unknown whether *C.elegans* employs a mechanism similar to backpropagation, although the neural activity in *C.elegans* may be influenced by learning and adaptation, similar to artificial networks [42], [43]. Recent works have shown that learning systems can mimic natural connectome activity with varying degrees of bio-plausibility [44], [15], [45]. In 2019, taking inspiration from the work of [46], Deep Connectomics Networks (DCNs) [47] were proposed as an extension of DNNs. The work attempted to design small-world neural networks similar to real-world neuronal networks. However, DCNs did not fully reproduce the topology of living connectomes preserved by evolutionary pressure, and are often based on existing architectures like ResNet [21]. Another relevant attempt to create neural models inspired by living organisms' connectomes comes from the work of Hernandez et al. [16], where the authors designed a neural model inspired by the *C.elegans* connectome, using the SNNs and applying the constructed model to toy classification problems. However, this SNN approach is slow in training time due to the high computational cost of the Hodgkin–Huxley model [48]. Inspired by the *C.elegans* nervous system, Chahine et al. [49] propose Liquid Neural Networks (LNNs), a class of neural models with continuous-time dynamics outperforming

various state-of-the-art agents in the drone visual navigation task. Lastly, an interesting connection can be found in Yan et al.'s recent work [15], who propose the backpropagation algorithm with sparsity regularization (BPSR) on bio-inspired networks. This variant of the classical backpropagation algorithm imposes synaptic structure sparsity and it is applied to SNNs with positive results on classical classification problems, such as MNIST [50] and CIFAR-10 [51] datasets.

b) A structurally and efficient bio-plausible artificial intelligence: As shown in the previous paragraphs, some learning models aim to simulate physical-chemical bio-properties of propagation, while others focus on neuronal feedback and memory mechanisms. However, these learning systems lack structural bio-plausibility, despite the potential to improve efficiency and learning capacity, which could hinder the development of artificial ones [52]. For this reason, the purpose of this study is to investigate biologically-plausible artificial deep learning models by examining specific motifs from a reference biological connectome. As shown in Figure 1 - Box **(a-c)**, the connectome of *C.elegans* is chosen as a reference because it has a reasonable size and is characterized by three functional neuron classes: sensor neurons, interneurons, and motor neurons. This simplifies connectome-inspired neural network layering, where sensors serve as an input space, inter-neurons as a latent space, and motor neurons as an output space. The connectome of *C.elegans* is represented as a network of tensors (TN). In Figure 1, Box **(d-f)**, the generated learning model is layered by transforming every neuron into a sequence of fully connected layers and every synapse into a learning graph connection. The Encoder in Figure 1 Box **(d)** is an external system that encodes information from the so called external environment, enabling the layered connectome to learn input information. The Decoder in Figure 1 Box **(d)** transforms the learned information into a form that can be used by the connectome to interact with the external constraints of that environment, where the inputs and outputs depend on the learning problem. To enhance the analysis, ad-hoc models are designed to generate artificial connectomes based on evolutionary features. These generators use a custom Variational Graph Autoencoder [53] architecture that represents most of the structural information (and, by extension, the evolutionary patterns) to be learned. Since the original and the generated artificial connectomes are both based on the motif distribution of sensors, motors, and interneurons, a custom algorithm is designed for comparisons. The learning performance of the original and artificial layered connectomes is analyzed, as well as their motif distributions, highlighting the strengths, scalability, and limitations of transforming connectomes into learning systems which are influenced by structural features built from evolutionary patterns. In conclusion, most bio-inspired models show limited performance, while connectionist models rarely mimics biological networks showing higher accuracy. ElegansAI fills the gap in modeling bio-inspired connectionist-oriented models which follows network evolutionary patterns combined with backpropagation strategy. Section Results II is organized as follows: Firstly, in section II-A, the discussion revolves around clues of evolutionary conservation related to connectome characteristics. Next, in Section II-B, a comparison is made between the neural networks optimized by the nematode connectome and state-of-the-art models. Furthermore, Section II-C presents the findings obtained from the examination of neural networks designed using randomly rewired connectomes. Section II-D delves into the performance evaluation of the original connectome with respect to simulated ones generated by advanced deep-learning graph autoencoders. Finally, in Section II-E, the conclusions drawn from the findings on connectome learning are provided, along with final considerations.

Methods section III is structured as follows: in Section III-A, the design and engineering of supervised and unsupervised ElegansAIs are discussed. Next, in Section III-B, the processes for obtaining, organizing, and generating connectomes at various degrees of similarity with respect to the original one are described. This includes the explanation of randomly generated, simulated through autoencoders, and original connectome data. Finally, in Section III-C, the *MiDEA* algorithm is described. The algorithm investigates multi-dyadic effects on connectome distributions, aiming to uncover insights into evolutionary conservation.

II. Results

A. The evidence of evolutionary conservation on the reference connectome.

An examination of the distribution of various neural operations within the nervous system of the nematode is conducted using an analysis of motifs, by employing algorithms designed for the computation of dyadic and multi-dyadic effects. Specifically, our work investigates the way in which the structure of the network impacts deep learning systems by evaluating if the interactions among neuronal attributes reflect patterns that have been optimized through evolution. In this context, a dyad can be described as a couple of interconnected neurons sharing similar functional traits. Conversely, the multi-dyadic effect provides a broader comprehension of the function and interaction of different neuronal types within structural motifs and shortest paths. Our findings indicate that when using both dyadic and multi-dyadic methods, the distances in the original connectome, compared to those randomly rewired, highlight the fact that the neuronal functions at the node level of the connectome are not randomly organized. Furthermore, it has been confirmed that evolutionary optimization varies for neuronal functionalities associated with both chemical and electrical synapses [52]. More specifically, Supplementary Tables S1 and S2 display a gradual increase in dyadic and anti-dyadic distances for all neuron pairs when comparing the nematode connectome to those that have been randomly rewired (with rewiring percentages ranging from 0.2 to 1.0). This suggests that the functional interplay among motor, sensory, and interneurons are inherent characteristics that gradually diminish in importance as the extent of rewiring increases. This insight is also consistent with earlier research on structural motifs as referenced in [54], [55], and [52]. The influence of dyadic and

anti-dyadic interactions on neuronal distances is more distinct in the context of directed synapses as opposed to undirected ones. This finding is bolstered in the calculation of dyadic-effect information content, as presented in Supplementary Table S3. The functional multi-dyadic/anti-dyadic information content displays a progressive increase, beginning from a value of 0.03 on electrical shortest paths of length 2, up to an average value of 3.91 for chemical shortest paths of length 4. This characteristic is also observed in other biological sequences and is described by the 'short memory' property [56]. Supplementary Tables S1, S2, and S3 also reveal that the occurrence of structured paths in connectomes, signified by the multi-dyadic/anti-dyadic effect, is a highly preserved characteristic that depends on the length of the synaptic path. A minor rewiring involving only 20% of the connections is enough to disrupt this effect in shortest paths composed of less than 4 edges. This consistency is observed across sensor, inter, and motor neurons in both chemical and electrical synapses. Consequently, variation in shortest path lengths (sp2, sp3, sp4) suggests that evolutionary optimization primarily preserves the multi-dyadic effect in the reference connectome, and the relevance of this preservation diminishes as the path length extends. The outcomes are derived from employing both the dyadic-effect algorithm proposed by Park and Barabasi [57], and its extended version, the Multi-Dyadic Effect Algorithm (MiDEA). The latter algorithm is discussed in Section III-C and deepened in Supplementary Section S1. In comparison to the algorithm by Park and Barabasi, the benefit of the MiDEA lies in its ability to differentiate between chemical and electrical synapses, which can be directed or undirected. Furthermore, it provides a separate analysis of the influence of dyadic and anti-dyadic interactions on the shortest paths with respect to their directionality. Supplementary Section S1-E presents visual comparisons of the dyadic/anti-dyadic effect in both Park and Barabasi and MiDEA, demonstrating a propensity in the reference connectome to create connections between neuron clusters with different functional characteristics when these connections are undirected. In contrast, when the connections are directed, there seems to be a partial trend toward establishing connections among neuron groups possessing similar functional attributes. These findings illustrate that the neuronal circuitry motifs in the *C. elegans* connectome have been accurately honed through evolutionary optimization. Thus, these observations offer pivotal directions for constructing deep learning models which mimic these directed and undirected evolutionary patterns.

B. Comparisons with state-of-the-art models

The results of this section show that the proposed models $M1$ and $M2$ (detailed in Section III-A4) outperform SOTA deep learning models on two well-known benchmark datasets. In Table I, our transformer-based ElegansAI $M1$ model shows significant improvements in the classification of images from the *Cifar10* dataset compared to deep-learning and machine-learning SOTA models [1]. Such models include classical vision transformer architectures like *ViT, CvT, CaiT, BiT* or *DeiT* [23], [24], [26], [58], [27], [25], evolutionary-based transformer approaches like μ2Net [28] as well as pure convolutional architectures like EfficientNetV2 [20]. The $M1$ model achieved a Top-1 accuracy of 99.99% on the test set, resulting in complete and accurate classification regarding error accuracy. It is worth noting that despite having fewer training parameters ($107M$) than the second-best transformer, *ViT-H 14* [24], which has *623M* parameters, $M1$ still outperformed it. *EfficientNetV2-L* [20] with $121M$ parameters achieved a Top-1 accuracy of 99.10%, while the *ResNet*-inspired transformer *BiT-L* [58] had a *Top-1 accuracy* of 99.37%.

TABLE I
ELEGANSAI $M1$ VS SOTA MODELS FOR *Cifar10*

Model	**Top-1Acc.**	**Param.**
ElegansAI $M1$ **(ours)**	**99.9**	107M
ViT-H/14 [24]	99.5	632M
μ2Net [28]	99.5	
ViT-L/16 [24]	99.4	307M
CaiT-M-36 U 224 [26]	99.4	
CvT-W24 [25]	99.4	
BiT-L [58]	99.4	
ViT-B [59]	99.3	
Heinsen Routing + BEiT-large 16 224 [60]	99.2	309.5M
ViT-B/16 [61]	99.1	
CeiT-S [62]	99.1	
AutoFormer-S 384 [63]	99.1	23M
TNT-B [64]	99.1	65.6M
DeiT-B [27]	99.1	86M
EfficientNetV2-L [20]	99.1	121M
BPSR SNN ResNet [15]	90.74	260.7M

Moreover, ElegansAI $M2$ has outperformed machine/deep learning-based SOTA models in global benchmarks for unsupervised digit reconstruction. SOTA models include a wide range of machine learning techniques, varying from autoencoder-like architectures like Stacked Capsule Autoencoders or Adversarial Autoencoders [29], [30], to GAN-based methods like CatGAN, InfoGAN or PixelGAN [65], [66], [67], to information theory and topology-based algorithms, like Invariant Information Clustering (IIC) and Sparse Manifold Transform [68], [31]. Table II shows our $M2$ results in comparison with both deep and traditional machine learning problems. All the showed results are collected from the online benchmark repository[2] except for Stacked Capsule AutoEncoder (AE) [29] where instead of reporting $MNIST$ is reported 40×40 $MNIST$ at 98.7 of accuracy. Our model $M2$ reaches a value of 99.78 (*Top-1 Accuracy*) with *MSE* equal to 0.0018 overreaching all the other models in the competition. Moreover, $M2$ overreach the 99.27 of F1-score with respect to *DenMune* [69] that is of 96.6.

C. Comparisons with randomly generated networks

In this section, it is demonstrated that the learning performance of models based on the TN (see Figure 1) of

[1]Cifar-10 Benchmark dataset
[2]MNIST-Unsup - Last queried 6th March 2023

TABLE II
ELEGANSAI $M2$ VS SOTA MODELS FOR MNIST UNSUP

Model	Top-1 Acc.
ElegansAI $M2$ (ours)	**99.8**
IIC [68]	99.3
Sparse Manifold Transform [31]	99.3
SubTab [32]	98.3
Stacked Capsule Autoencoder [29]	98.0
Self-Organizing Map [33]	96.9
Bidirectional InfoGAN [66]	96.6
Adversarial Autoencoder [30]	95.9
CatGAN [65]	95.7
InfoGAN [70]	95.0
PixelGAN AE [67]	94.7
Model	**F1 (%)**
ElegansAI $M2$ (ours)	**99.3**
DenMune [69]	96.6

the reference connectome is significantly superior to those based on randomly rewired connectomes on both Cifar10 and MNIST-Unsup datasets. In Supplementary Section S3 and in Section III-B2, the process of generating *random tensor networks* (r-TNs) which are used here for comparisons is detailed. The dimensions of the r-TNs are comparable with the original TN which is structured by considering *C.elegans* connectome. All model hyperparameters of *ElegansAI* $M1$ and $M2$ models remain unmodified for a fair comparison. Thus, for each experiment, only the r-TN-th connectome changes for each model training by reflecting the different random architectures generated. The ratio between accuracy and epochs in Figures 2 can also be interpreted as learning velocity indicators of the effectiveness of the *ElegansAI* $M1$ and $M2$ models with original connectome, as they achieve higher performance, in comparison to randomly generated ones (as shown by Figures 2, 4. Supplementary Figure 3 shows the model convergence speed and accuracy of $M1$ and $M2$ by tracking at which epoch the minimum loss is reached. With respect to the supervised classification problem of *Cifar10* (see Figure 2), the accuracy on the validation set of the original connectome (label *org* - red dashed line) remains stable across epochs outperforming all the 30 r-TN structured models. The 10 models trained with the Watts-Strogatz (WS) generative algorithm (Label $\mathbf{G_1}$- blue solid line) exhibit slightly better performance, on average, compared to those structured with the Barabasi-Albert (BA) (Label $\mathbf{G_2}$- - green dotted lines) or Erdos-Renyi (ER) (Label $\mathbf{G_3}$ - orange dashed lines) generative algorithms. Similarly, in *MNIST*-Unsup, the WS algorithm follows the higher accuracy values of the original connectome only in the first epochs and then gradually decreases. On the other hand, models structured with the ER algorithm maintain a higher accuracy for both $F1$ and *Accuracy* scores in subsequent epochs without reaching the accuracy of WS. It is noteworthy that Watts and Strogatz [71] demonstrated the small-world property of the *C.elegans* connectome by providing valuable insights into designing networks that are similar to natural ones. This property is also reflected in the performance comparisons discussed above.

D. Comparisons with simulated networks

After the comparison between the learning and prediction capabilities of random and original connectomes, deep learning generators are trained on optimized original connectome motifs to generate new ones. Thus, a second comparison is made evaluating 48 simulated connectomes generated by an ad-hoc designed Variational Graph Autoencoder (VGAE) [53] (see Supplementary Section S3-2). A total of 8 training sets suited for the 16 models consist of hundreds of randomly rewired networks at different levels ranging from a probability of 0.1 to 0.4. Thus, every model learns a graph distribution from a different rewiring level and a different rewiring topology. Once the networks are trained, several graphs are sampled from the posterior distribution of the generative model, conditioned by the original connectome. As it is shown in Supplementary Figure 2, the coherence of the selection criteria is tested a posteriori by evaluating the Jensen-Shannon distance of the simulated connectomes with respect to the motif entropy of the original one. The criteria described in Supplementary Section S3 - (p. Generation VGAE) shows in detail in which ways the 48 connectomes are selected and then converted to a set of TN for $M1$ and $M2$. Similarly to r-TN (see II-C), these TN are named *simulated tensor network* (s-TN). In Figure 3, ElegansAI *M1* and *M2* are compared by means of the Top-1 validation accuracy over the number of training epochs (see also Figure 3).

In detail, Figure 3 shows that the trained models based on the original connectome (indicated by the green dashed line) overreach the average performance of two groups of s-TN predictors. The models trained by using the simulated networks are divided into two groups by thresholding in half the Hamming distance[3] δ_τ from the reference connectome. The thresholding criteria for $M1$ and $M2$ and the s-TN separation are detailed in Supplementary Figure 1. In Figure 3, the results show that the connectomes with $\tau \geqslant 0.5$ have better average performance compared to those with a τ lower than 0.5. The group with $\tau \geqslant 0.5$ is shown with orange bands, while the other with blue bands. As shown in Supplementary Figure 1, the threshold is decided, by observing the distribution of connectomes obtained by leveraging VGAEs that learned how to rewire only the latent spaces of the connectome (those intercepted by interneurons) and those with total rewiring.

Figure 4 shows that s-TNs with respect to the randomly rewired ones, achieve significant performance.

In detail, the s-TNs are divided into two groups. In the first group there are the s-TNs which are generated starting from interneuron-interneuron connectome-rewired training set (s-TNs latent). In the second group there are the s-TNs produced by generators trained on total rewired connectomes (s-TNs total). It is noteworthy that simulated networks s-TNs latent produced by learning the rewiring of only the interneurons achieved better performance than those based on total

[3]It's worth noting that such distance is, in this context, equivalent to the notorious *graph edit distance* [72] since the compared graphs completely share nodes.

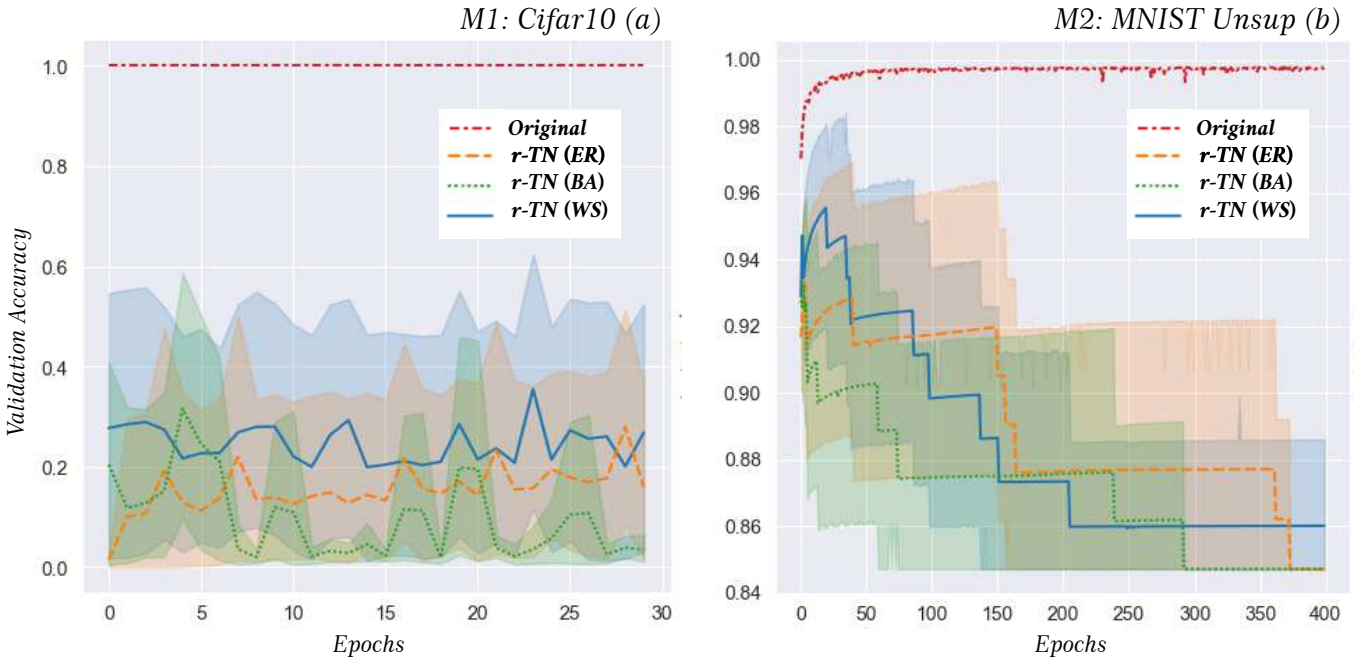

Fig. 2. Figure 2 shows the comparisons on the validation set by using the Top-1 Accuracy over the number of training Epochs for ElegansAI *M1* and *M2*, boxes **(a)** and **(b)**, respectively. The models *M1* and *M2* are structured by the immersion of the original TN (red dashed lines) and of 30 random tensor networks (r-TN) produced by 3 stochastic generators. Specifically, r-TN (BA) is for Barabasi-Albert, r-TN (ER) with Erdos-Renyi and r-TN (WA) for Watts-Strogatz generators. See also Section II-C.

rewiring, highlighting the strength and capability of VGAE to simulate connectomes with features that progressively become more similar to those influenced by evolutionary pressure.

This supports the hypothesis that natural optimization can effectively enhance the learning performance of deep learning models. To explore this hypothesis, the relationship between prediction performance on the validation set and the effects of multi-dyadic and multi-anti-dyadic connections was investigated by dividing the network into chemical and electrical synapses. Figures 5 and 6 demonstrate that multi-dyadic and multi-anti-dyadic connections have an impact on prediction performance on the validation set. The performance of models based on s-TN was divided into two groups based on the median value of the multi-dyadic and multi-anti-dyadic magnitudes, resulting in two distinct clusters: those that had higher magnitudes (generally the nearest to the original ones), and those that show lower magnitudes (usually the farthest to the original ones), represented by orange and blue bands, respectively. As a consequence of the results shown in Section II-A, the motif distributions are better represented by the chemical connections, resulting in a clear separation between the two bands. However, when considering the heterophilic and heterophobic magnitudes in the networks mapped onto electrical connections, the separation in performance slightly deteriorates. Taken together, these findings indicate that the evolutionary features of neuronal circuitry, including the effects of multi-dyadic and multi-anti-dyadic connections, can guide the design of learning algorithms with optimized performances.

E. Conclusion

The comparison between biological and artificial neural networks highlights the remarkable complexity, efficiency, robustness, and flexibility of the former, particularly in the case of highly evolved brains. Although recent advances in artificial networks have been significant, they still fall short of matching the elevated capabilities of biological networks. Ongoing research is essential for a complete comprehension of the mechanisms behind neural network functioning and the advancement of advanced artificial networks that can accurately emulate the complexity and adaptability seen in their biological counterparts. The study demonstrated the growing feasibility of incorporating biological structures of connectomes into artificial neural networks, facilitated by the increasing of innovative methods and models (such as transformers and attention based encoders/decoders) which interact with our tensor network system. This immersion has the potential to enhance outcomes in classification and reconstruction tasks, as evidenced by the performance of our ElegansAI models, specifically *M1* for classification and *M2* for reconstruction. Our work enables researchers to explore new avenues of research that were previously unreachable. As such, the future of neural network research is anticipated to involve greater integration of biological and artificial systems, leading to novel insights and breakthroughs in the field.

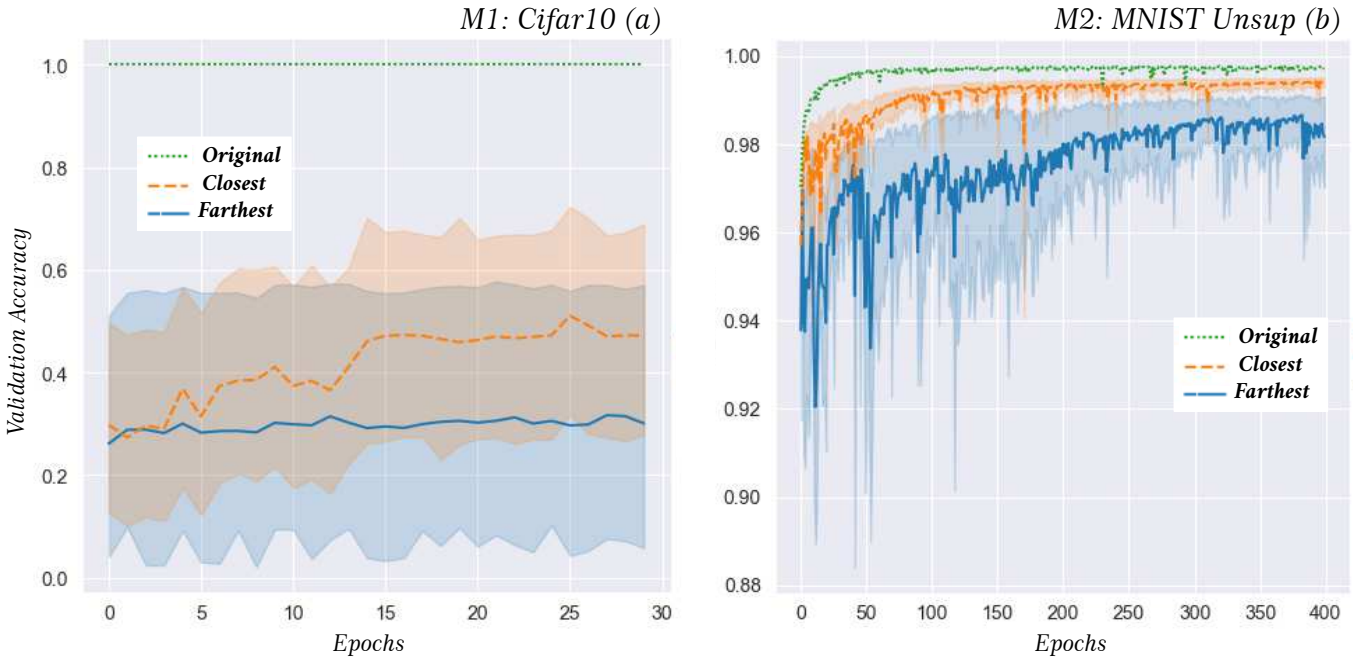

Fig. 3. In Figure 3 comparisons in terms of validation set's Top-1 average accuracy over the number of training epochs for EleganAI *M1* and *M2* are shown in boxes (a) and (b), respectively. The models are structured by the immersion of the original TN (green dashed line) and of two groups of simulated tensor networks (s-TN). Groups 1 (the closest) and 2 (the farthest) stand for s-TNs whose structures are based on generated graphs which are the closest (orange bands) and the farthest (blue bands) to the original connectome according to the Hamming distance, respectively (see Supplementary Section S3-2). The plots show that models based on graphs that are closer to the original connectome tend to have considerably better average performance. See also Section II-D.

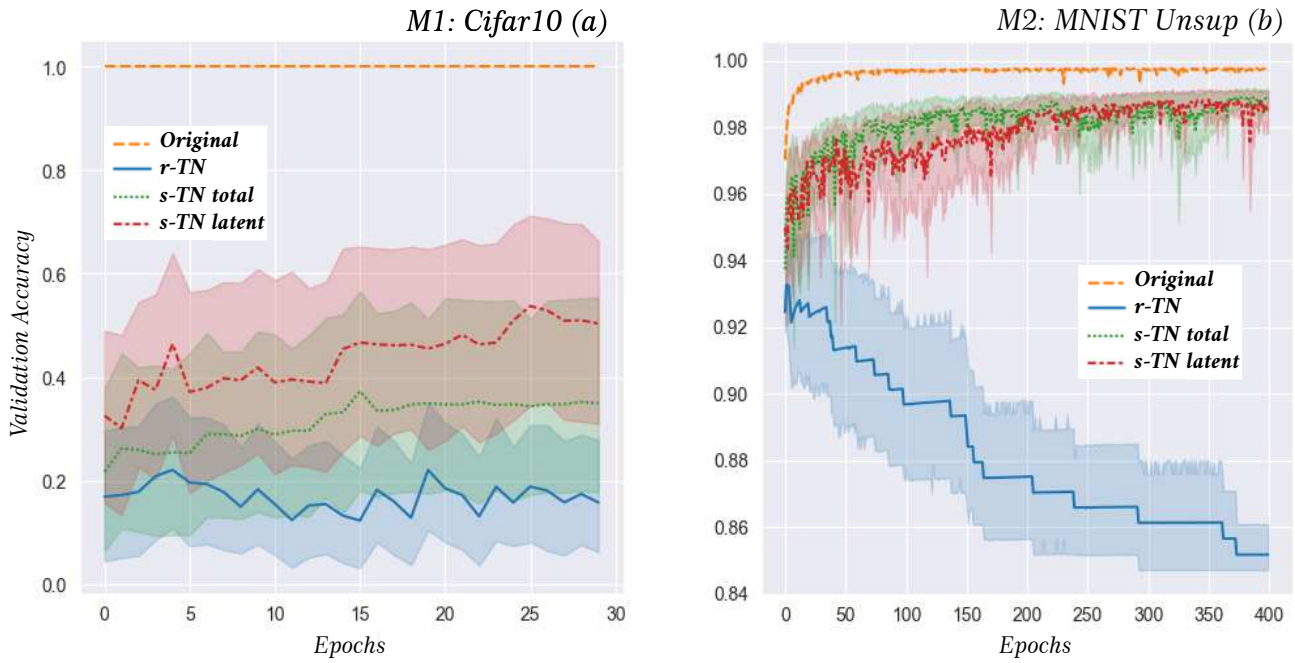

Fig. 4. In Figure 4 the comparisons in terms of validation set Top-1 Accuracy over the number of training epochs for EleganAI *M1* and *M2* are shown in boxes (a) and (b), respectively. The models taken into account are the tensor network based on the original connectome (orange dashed line) the average performance of the random tensor networks (r-TN, blue line), and the average performance of the simulated tensor networks (s-TN, red and green dashed lines). Specifically, s-TN_total and s-TN_latent stand for s-TN whose structure is generated by VGAE models, trained on connectomes with rewired edges in the entire edge set (whole connectome) or only on the latent space edge set, that means rewiring only interneurons connections. The graphs clearly highlight how the more randomness is injected into the network, the more performance degrades. Aside from the high performance reached by the original network, r-TNs show the worse performance, while s-TNs seem to show better average performance the more they are close to the original connectome.

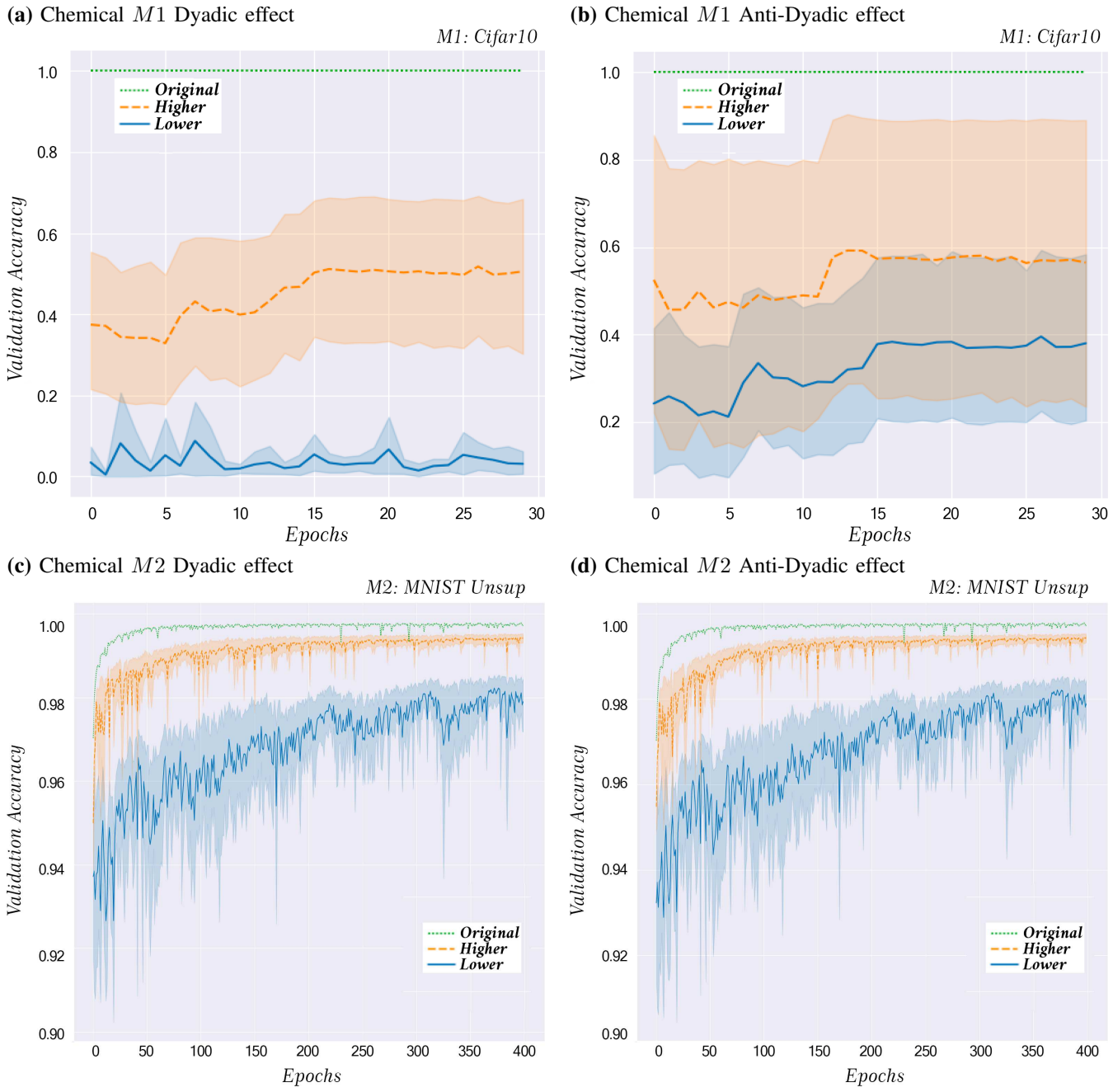

Fig. 5. Figure 5 display the validation set's Top-1 average accuracy over the training epochs for ElegansAI models M1 ((**a-b**)) and M2 ((**c-d**)). The distinct bands (blue and orange) represent the models' performances separated by thresholding of the multi dyadic/anti-dyadic effect measured on directed s-TNs. In detail, the MiDEA algorithm evaluated the motif patterns (effects) only on the chemical (directed) shortest paths of length 2. Successively the s-TNs are grouped based on the normalized intensity of the multi-dyadic/multi-anti-dyadic effects, into two different bins (orange range - higher ($\geqslant 0.5$), blue range - lower (< 0.5)). The s-TNs with the stronger directed dyadic/anti-dyadic effects generally tend to have better performance. See also Section II-D.

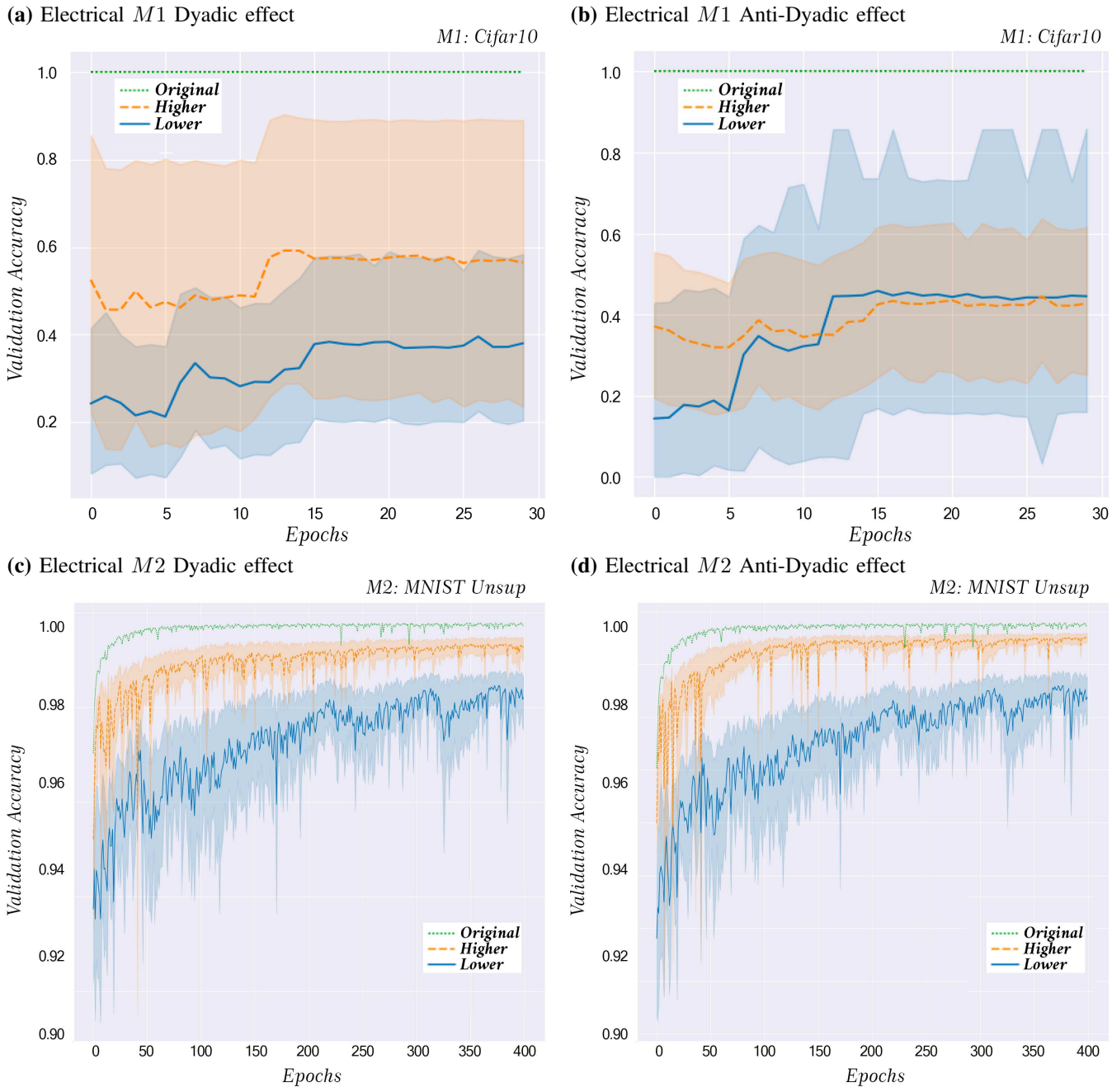

Fig. 6. Figure 6 display the validation set's Top-1 average accuracy over the training epochs for ElegansAI models M1 ((**a-b**)) and M2 ((**c-d**)). The distinct bands (blue and orange) represent the models' performances separated by thresholding of the multi dyadic/anti-dyadic effect measured on undirected s-TNs. In detail, the MiDEA algorithm evaluated the motif patterns (effects) only on the electrical (undirected) shortest paths of length 2. Successively the s-TNs are grouped based on the normalized intensity of the multi-dyadic/multi-anti-dyadic effects, into two different bins (orange range - higher ($\geqslant 0.5$), blue range - lower (< 0.5)). The s-TNs with the stronger undirected dyadic/anti-dyadic effects generally tend to have better performance. See also Section II-D.

III. METHODS

A. ElegansAI

This section provides an overview of the design process for ElegansAI. It begins with the transformation of a connectomic structure, whether it is the original one of *C.elegans*, bio-plausible or randomized, into a tensor network TN. Subsequently, the TN is immersed into well-known deep-learning architectures. Section III-A1 details the construction of the TN starting from a graph/connectome, which specifically mimics the structure of a neural circuitry composed of three classes of neurons: sensor, inter, and motor neurons. Furthermore, Section III-A2 introduces transformer and autoencoder inspired architectures which are implemented to incorporate the TNs in their latent spaces. The architecture parts which encompass the TNs are referred to as the external environment ($\mathbf{E}$). In Section III-A4, these architectures are specifically designed to address classification and reconstruction problems on images. The transformer-like model $M1$ is employed to solve a classification problem based on the *Cifar10* dataset [73], which is a collection of 60000 $32 \times 32 \times 3$ pixel RGB images for 10 classes with 6000 images per class. According to the official repository, 50000 images are used for training and 10000 for testing purposes [4]. Conversely, the autoencoder-inspired model $M2$ works on the *MNIST* dataset [74], [50] in an unsupervised fashion for image reconstruction. *MNIST* is a collection of gray-scaled digits of size 28×28 for a total of 60000 training images and 10000 testing images (according to MNIST official repository [5]. As shown in Figures 7 and 8, the architectures of the different $\mathbf{E}$s vary (contingent upon the specific task being addressed), while the TNs can be considered as interchangeable modules because they are independent of the specific task.

1) The tensor network: The TN resulting from the collection of connectome/graphs S is constructed by allocating a tensor unit θ for each node/neuron, rather than a single tensor unit representation per edge. Therefore, each edge/synapse, chemical or electrical, corresponds to an edge connection at the architectural level between two different tensor units. The transformation algorithm is depicted with the pseudo-code as reported in Supplementary Materials Algorithm 1. The first part of the transformation algorithm is an initialization phase which involves scanning all nodes labeled as sensor neurons on the s-th connectome S_s, then assigning the same feature map θ_{init} from the previous layers of the external environment $\mathbf{E}_{in}$ (see also Supplementary Material Algorithm 1 -*Init Sensors* function). The other associations between tensor units θs are represented in the core of the latent space, and the operations between tensor units are mapped into the so-called computational graph (which allows TensorFlow to track a non-linear mapping of all the mathematical operations between tensors). In the second part of the algorithm (see also Supplementary Materials Algorithm -*Create Tensor Net*), the cascading scan of the s-th adjacency matrix $A_{\hat{S}_s}$ continues, by searching dyadic and anti-dyadic connections. In the first scan, the algorithm searches nodes labeled as motor neurons and interneurons which are linked with sensors. If the i-th sensor node is connected to the j-th motor neuron or interneuron and the latter has not been already allocated, a new tensor unit is allocated, and a functional connection is established from i to j in the latent space architecture. Accordingly, the computational graph is updated. As the whole adjacency matrix is scanned, all directed (chemical) and undirected (electrical) connections are allocated as connections $e(i, j)$ between the involved tensor units. The second scan of the adjacency matrix is used to allocate all connections between interneurons and sensors/motors, and the last scan similarly establishes connections between motors and interneurons/sensors. If a dyadic connection is present on S_s, the algorithm allocates tensor units establishing edge associations between neurons of the same type (i.e sensor to sensor, etc..). To account for multiple incoming edges without information loss or overwriting between dyads and anti-dyads, the transformation algorithm replaces the single tensor unit per neuron with an element-wise multiplication of tensor units of the same tensor shape. As shown in Figure 1-**e**, the skip connection by multiplication is denoted by the symbol $\otimes$. Finally, the output motor unit tensors are collected and stacked. Once the motor tensors are stacked, they are fed into a multi-head attention layer μ_1 that interfaces with the external environment represented by the $\mathbf{E}_{out}$ (Figure 1 - **(f)**) In conclusion, the algorithm follows a logic of edge association consistent with that of the original graph, according to the directionality of the artificial neural architecture and its computational graph. The building of non-linear topology is supported by leveraging the connection modularity of Keras Functional API. Post-processing on the operational latent space tensor network, such as the presence of backward connections and cycles on the computational graphs, are resolved by the Grappler optimizer of Tensorflow [75].

2) The external operational environment: As depicted in Figure 1, the connectome-derived TN is structured as a latent space embedded within the external environment ($\mathbf{E}_{in}, \mathbf{E}_{out}$). Generally, the function of $\mathbf{E}_{in}$ is to encode the input feature maps for the sensor neurons of TN, whereas $\mathbf{E}_{out}$ serves as a decoder in reconstruction tasks or as a classifier in classification tasks by operating with motor neurons. As illustrated in Figures 7 and 8, the external model components can be conceptualized as an artificial exposome that interacts with the artificial connectomes contained within ($\mathbf{LS}$ latent spaces in yellow boxes). Alternatively, each i-th environment ($\mathbf{E}^i_{(in,out)}$) can be considered as the environment of an intelligent agent equipped with motor and sensor tensors functioning as actuators or sensors. Specifically, a transformer-inspired model $M1 : \mathbf{E}^1_{(in)} \rightarrow TN_q \rightarrow \mathbf{E}^1_{(out)}$ with q tensor networks is employed for image classification [23], while an autoencoder-inspired model $M2 : \mathbf{E}^2_{(in)} \rightarrow TN \rightarrow \mathbf{E}^2_{(out)}$ is utilized for unsupervised digit reconstruction. To preserve a clear distinction between the external environments and the tensor

[4]Cifar10 official repository - Last queried on 6th March 2023

[5]MNIST official repository - Last queried on 6th March 2023

networks in all proposed models, no supplementary design modifications, such as incorporating skip connections or others, have been introduced between $\mathbf{E}_{in}$ and $\mathbf{E}_{out}$, aiming to assess the model's expressive capacity and effective complexity [76] of the original, randomly rewired, or generated/simulated TNs.

3) Preprocessing and data augmentation : All images inputted to $M1$ and $M2$ have undergone a preprocessing and data augmentation phase within their respective input environments as usually applied in the Literature [24]. Specifically, in *Cifar10* for $M1$ a central crop of 75%, resulting in 24×24 images is applied. Then, data augmentation is performed by applying 4 transformations: the first transformation is a rotation with a range of 15 degrees, which introduces a degree of variation to the orientation of the images, making the model more robust to rotations. The second transformation is horizontal flipping, which involves mirroring the image along its vertical axis. This transformation is applied with a probability of 0.5, allowing the model to learn from images with reversed orientation. The third and fourth transformations are width and height-shift with a range of 0.1, which involve shifting the images horizontally or vertically by up to 10% of their width or height. This allows the model to learn from images with slight variations in position, which can occur due to changes in camera angle, object placement, or other factors. In $M2$ that focuses on grayscaled MNIST images, instead of performing a central crop, a binary thresholding equal to 0.3 is applied to the images. The binary thresholding simplifies the images and removes any noise or unnecessary details that may not be useful for the digit unsupervised reconstruction. After thresholding, data augmentation is applied by using two types of transformations: width and height shift range to 10% and a zoom range of 10%. These transformations are used to generate slightly different versions of the same digit, which increase the size and diversity of the dataset and prevent overfitting.

4) Model architectures:

a) M1 - Transformer-inspired ElegansAI for Cifar10: The architecture of the external environment $\mathbf{E}_{(in,out)}$ and the latent space (**LS**) for $M1$ is shown in Figure 7. To obtain a set of flattened patches (n_p = 4), the original images on 3 channels are reshaped and patched with equal dimensions. Then, the n_p patches follow two branches. The first branch bypasses the latent space **LS** (blue arrow in Figure 7). Meanwhile, the patches in the second branch enter the **LS**, where a replica of the tensor network (TN_q with $q = [1:4]$) is configured for each q-th flattened patch ($[p_1, p_2, p_3, p_4]$). In the **LS**, as described in the transformation algorithm (see Section III-A1), all the fully connected layers of the q-th TN, named tensor units θs, are allocated with 432 neurons (resulting by flattening the 3 channels $\times$ 144 neurons) and a rectified linear unit *ReLU* is used as the activation function. According to the initialization function of the transformation algorithm (see Section III-A1 - **Init Sensors** function), each input flattened patch is assigned to the group of sensor layers (label "S"), one for each TN replica (see Figure 7 - (c) - blue nodes). Note that each of these TN replicas processing a patch of the input shares weights with all the others, which drastically reduces the trainable parameters, especially compared with other state-of-art transformer networks, like ViT, BEiT or CvT [24], [60], [25], and even some parameter-optimized convolutional architectures like EfficientNetV2 [20]. Once the information flows from sensors to interneurons, the output of the TN in the **LS** is collected from the fully connected layers labeled as "motors" and reshaped according to the size of the initial patches. Thus, for each replica of the TN, a single feature map is extracted by the application of a multi-head attention $\mu_1(H, K)$ with a head-space H equal to the number of allocated θ motor layers and a key space K fixed to 32 (which is approximately one-third of the number of motor neurons). $\mu_1(H, K)$ is applied to both the flattened input sensors and the motor layers. To keep track of relative patch positions along the model, the feature maps in output from the **LS** (violet arrows of Figure 7) are arranged by applying a positional embedding layer (Figure 7 - (d)). Once the features are positionally embedded, they are provided in input to a feature space *condenser* as shown in Figure 7 (e). In both $M1$ and $M2$ setups (see also paragraph III-A4b), the condenser's role is to merge and reduce the feature space in the output obtained from the TNs. Then, these features are selected with respect to a reduced feature space built by applying a second multi-head attention (μ_2) driven by a drop-out of 10%. The $\mu_2(N, C)$ layer has a number of heads N equal to the number of input patches ($N = n_p$) and a key-space C equal to the number of neurons equivalent to the number of possible C-classes (for *Cifar10*, $C = 10$). It is worth noting that multi-head attention layers (μ_1 and μ_2) are commonly used in self-attention mechanisms. However, in this type of transformer, they are applied for encoder-decoder attention mechanisms. In the output from μ_2, for each H, the second-last *Reduce Mean* layer computes the mean of elements across the C dimensions producing a C-dimensional vector. The latter is in input to the last fully connected layer FC with C neurons and a $ReLU$ as an activation function.

b) M2 - Autoencoder-inspired ElegansAI for MNIST: In Figure 8, an autoencoder-like architecture is depicted, which encompasses a single TN. Compared to the preceding transformer-like architecture (see previous paragraph III-A4a), where each individual patch was allocated to a different TN, this architecture immerses a single TN directly into the **LS**. Figure 8 - **(a)** shows how in the external environment $\mathbf{E}_{in}$ an encoder is designed to progressively extract abstract representations of the input features via 2D-convolutional (*2DConv*) and max-pooling (*MaxPool*) layers. Figure 8 - **(d)** shows how the decoder $\mathbf{E}_{out}$ operates for image reconstruction starting from the output of the latent space to the original target via *2DConv* layers supported by bilinear interpolation for upsampling features (*UpSample*). In Figure 8 boxes **(a)** and **(d)**, the number of layers in the encoder is less than that in the decoder. This imbalance could provide certain advantages [77].

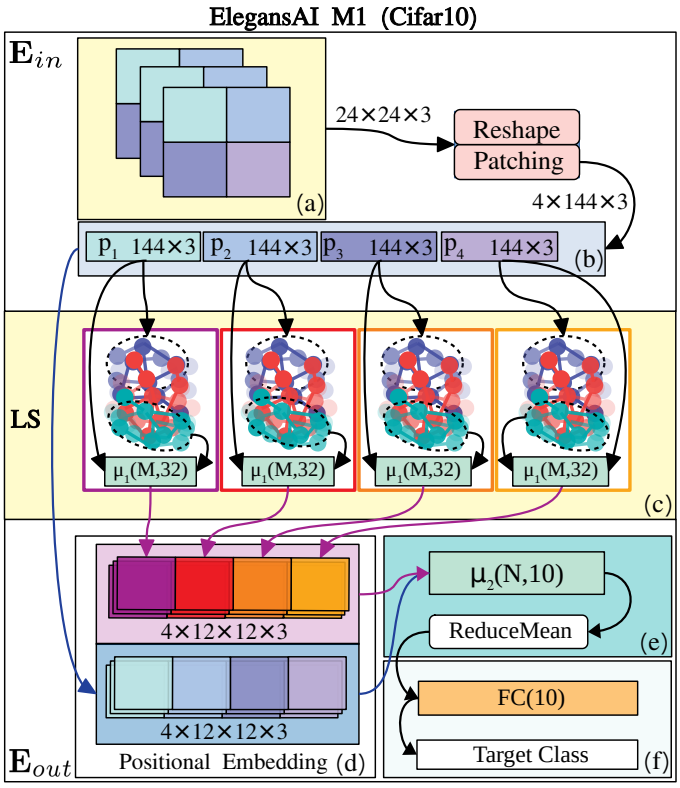

Fig. 7. **Architecture of ElegansAI M1 - *Cifar10*:** Figure 7 illustrates the architecture of the external environment $\mathbf{E}$ and the latent space ($\mathbf{LS}$) for $M1$. In Figure 7 - Box $\mathbf{E_{in}}(\mathbf{a})$, the input layer undergoes patching and reshaping operations to obtain flattened patches. In Box $\mathbf{LS}(c)$, the patches enter the latent space $\mathbf{LS}$ into $q = 4$ independent replicas of TN where sensors are shown in blue, interneurons in red and motors in green. The TN's θ tensor units are fully connected layers with 432 neurons and a rectified linear unit *ReLU* as the activation function. In $\mathbf{E_{out}}(d)$ the output of each TNs is collected from the θs labeled as "motors" and reshaped to match the 12x12x3 size of the initial p_n with $n = [1, 2, 3, 4]$. This is because in $\mathbf{LS}(c)$ a single feature map is extracted, in comparison with input patches p, for each TN replica by using multi-head attention $\mu_1(M, 32)$. Where $M = 86$ is equal to the number of allocated motor layers. In $\mathbf{E_{out}}(d)$ the patches that bypass the $\mathbf{LS}$ (blue arrow) and those from the TNs (violet arrows) are positionally embedded. Then, the two positional embedded layers are provided in input to the feature space *condenser* (Figure 7 - (e)). In this case, the *condenser* composed by a second multi-head attention layer $\mu_2(N, C)$, which has a H equal to the number of N-produced input patches and a $C = 10$ and by a *ReduceMean* the averages the output of μ_2. Then, in $\mathbf{E_{out}}(f)$ it is provided in input to the last fully connected layer FC with 10 neurons.

For instance, given the presence of a dimensionally significant TN in the LS, overloading the model with an extensive-dimensional encoder is unnecessary. As in $M1$, the building procedure of the TN involves the transformation algorithm (see Section III-A1) by allocating fully connected layers of 784 neurons with an Exponential Linear Unit (*ELU*) activation function. As displayed in Figure 8 **(b)**, the feature maps in output from the $\mathbf{E}_{in}$ are flattened and follow two separate branches (blue and violet). The blue branch transports the features to a single fully connected layer of 784 neurons (28×28), after which layer normalization is executed (green *LayerNorm* Box in Figure 8 - **(b)**). With a longer path, the violet branch conveys the features to the single TN; as in $M1$, multi-head attention μ_1 is applied, followed by a fully connected layer of 784 neurons ($\mathbf{FC}(784)$). The latter layer also undergoes layer normalization (Figure 8 - second green *LayerNorm* in the violet path of Box **(b)**). The tensors output from the blue and violet branches are point-wise multiplied to generate a single output tensor. The mechanism of applying layer normalization and multiplication, despite the absence of some tensorial operations, could be regarded as a very simple alternative to the μ_2 multi-head attention in the *condenser* block of $M1$ (Figure 7 - Box **(e)**). The tensor in output from layer *Multiply* is fed into a feature space condenser block (Figure 8 (c)), where a series of fully connected layers, containing 512, 256, and 128 units respectively, further reduce the feature space. The output from $M2$ *condenser* to the decoder of $\mathbf{E_{out}}$ is normalized by using a traditional batch normalization after reshaping the reduced features into a tensorial form of $4\times4\times8$. In Figure 8 Block **(d)**, generally, the architectures are designed with smaller blocks and progressively diminish the number of filters in reconstruction; nevertheless, in this instance, a large number of filters is maintained, while the feature map's dimensions are progressively increased [20], [29], [67]. In all the layers considered within the various parts of $M2$, the *ELU* activation function is employed. The only exception is the final layer that leads to the target, which utilizes a sigmoidal activation function.

5) Training configurations: The models $M1$ and $M2$ of *ElegansAI* are trained with different parameter configurations and optimization functions. In our case, given the complexity of *Cifar10* with respect to *MNIST*, it is important for $M1$ to choose an optimizer that provides balanced importance to rare features. For this reason, the optimizer chosen for $M1$ is *AdaDelta* [78]. *AdaDelta* optimizer adjusts the learning rates based on recent gradient updates instead of storing all past gradients, resulting in a slower convergence on frequent features while also taking into account infrequent ones. The decay rate ρ for *AdaDelta* is set to 0.95. The second hyper-parameter is the precision ϵ which is fixed to $\epsilon = 1.0^{-7}$. The $M1$ *AdaDelta* optimizer is configured with an initial learning rate lr_{M1} equal to 0.01. On the other hand, for the unsupervised reconstruction problem of MNIST-Unsup, *Adam* [79] is chosen for $M2$ because it offers a robust and faster convergence on simpler datasets. The optimizer's learning rate for $M2$ is fixed at 0.001 ($lr_{M2} = 0.001$). The $M1$ model is trained using the original connectome of *C. elegans*, resulting in TN having $107,360,964$ trainable parameters, while for $M2$ with the same TN, the number of trainable parameters decreases to $87,852,914$. However, on simulated connectomes or those that are randomly rewired, the dimension of TN may vary and, accordingly, the number of trainable parameters may also change. Various types of initializers and regularizers can be applied at different levels of the architecture during the optimization procedure to prevent bias and ensure weight regularization. In $M1$ a correct weight updating of the lower layers of the model may be affected by the vanishing gradient problem inflating the whole learning process. Thus, according to [80], the kernel weights of the last fully connected layer

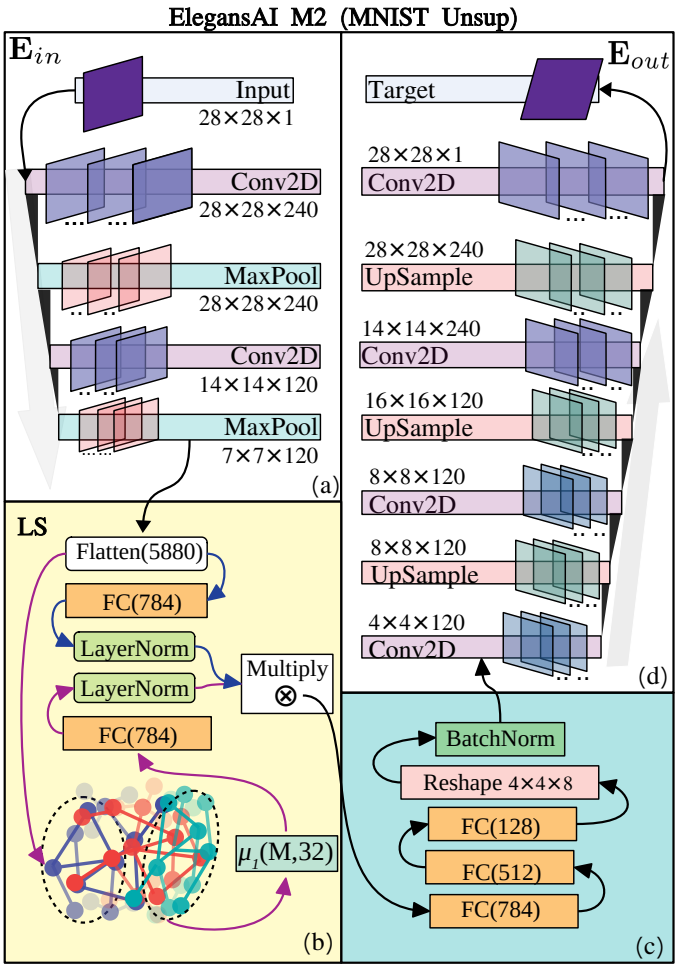

Fig. 8. **Architecture of ElegansAI *M2* - MNIST:** In Figure 8, an autoencoder-inspired architecture is depicted for $M2$. Box **(a)** showcases an encoder situated in the external environment $\mathbf{E}_{in}$, comprised of two successive 2D-convolutional (*2DConv*) layers respectively followed by max-pooling layers (*MaxPool*). Within box **(b)**, the architecture of the latent space **LS** is displayed, featuring two distinct branches. The violet branch directs the extracted features towards the sensor θ of a singular tensor network TN. As depicted in Figure 7, a μ_1 layer is employed to identify the most salient features among motor neurons (illustrated in gray). The blue branch serves as a more direct pathway and in this case, it is utilized to calibrate the TN prediction through the implementation of a point-wise multiplication layer ($\otimes$). Within the **LS**, box **(c)** illustrates a series of three fully connected **FC** layers with gradually decreasing sizes, ranging from 512 to 128. These layers are subsequently reshaped and subjected to batch normalization, prior to being provided as output to the decoder block in $\mathbf{E_{out}}$ (Box **(d)**). The decoder block, as represented in Figure 8 - **(d)**, consists of a sequence of *Conv2D* layers followed by upsampling via bilinear interpolation (*UpSample*). The transformations of tensor shapes after each tensorial operation are shown accordingly.

$FC(C)$ (Figure 7 - Box **f**) are initialized with *Glorot Uniform* distribution. The latter is helpful also to avoid the exploding gradient problem. For $M2$, a more extensive intervention is required to avoid gradient-related issues. Therefore, in $M2$, the kernel weights are initialized utilizing the *Glorot uniform* distribution, while the *bias* weights are initialized with a zero-wise distribution. The final convolutional layer of $M2$ (Figure 8 - Box **(d)** is regularized employing ℓ_1 regularization [81] with a penalty parameter of 0.0001 ($\ell_1 = 1.0^{-4}$). The models' training is improved by using random image selection with a fixed seed to create the batches. The batch size of $M1$ is $b_{M1} = 32$ while for $M2$ it is $b_{M2} = 128$. During the model training, overfitting was prevented by using early stopping. $M1$ and $M2$ performances on the validation sets were monitored according to the evaluation metrics *Top-1 accuracy* (see also Supplementary Section S2-C), and the weights obtained at the end of the best epoch were saved to guarantee maximum accuracy and generalization. Specifically, early stopping occurred at the $30th$ epoch for $M1$ and at the $400th$ epoch for $M2$. As previously discussed, the tasks addressed by $M1$ and $M2$ were intentionally made distinct. $M1$ is a supervised classifier, while $M2$ is an unsupervised grayscale image reconstruction model. The choice of the loss function for each model was accordingly tailored to the specific task. For $M1$, where the ground truth class is represented as an integer, the sparse categorical cross-entropy loss function λ_{M1} was employed. On the other hand, for $M2$, where the goal is to minimize the pixel-by-pixel reconstruction error, the binary cross-entropy loss function λ_{M2} was used.

B. Connectomes Description and Generation

1) The reference connectome: The connectome anterior/pharynx part of the hermaphrodite free-living nematode *C.elegans* consisting of 279 neurons and 3225 chemical and electrical weighted edges is analyzed. The *C.elegans* complex network is accurately reconstructed by [55], [82] and made free-available on the online repository[6]. Several *C.elegans* nervous systems are provided in the Literature, however, the network of *Varshney et al.* [55] is chosen as reference connectome because it is well annotated with respect to the full *C.elegans* network representation of 302 neurons originally provided by *White et al.* in 1986 [83]. The reference connectome, at the node level, presents three-class labels for neurons of type sensors, interneurons, and motors. Furthermore, at the edge level, the authors made available: **(a)** a weight that represents the number of between-neuron connections summing up at 13.000 synapses and **(b)** a binary label that indicates if the synapse is chemical, that is a directed edge, or electrical, an undirected edge. Moreover, the reference connectome is enriched by additional information, such as types of neurotransmitters, neuron soma positions[7], and other details like the neural cell class, at the node level. Furthermore, neurotransmitter type information and neurons' soma position are used to enhance the structural understanding of the generative tensor network models (see also II-D).

[6]https://neurodata.io/project/connectomes/ (GraphML data format) - Last queried on 6th March 2023

[7]Along the length of the nematode from the head to 0.83 mm on x-axis, an adult *C.elegans* has an average length of max of $1.5mm$. However, in the network of *Varshney et al.* [55] only the anterior/pharynx part of the worm is considered.

2) Random and simulated connectomes: In this work, four different classes of networks are transformed into TNs and compared with the TN derived from the original connectome. Three classes of 30 random small world graphs are generated using stochastic algorithms: Erdos-Reenyi $\mathbf{G}_1$ (ER), Barabasi-Albert $\mathbf{G}_2$ (BA), and Watts-Strogatz $\mathbf{G}_3$ (WS). These models generate features both by random wiring edges and random enriching edges and nodes with labels to signify the type of connection and type of neuron. In the WS model, an integer constant is used to set the median density limit, while in the models of BA and ER, the probability of insertion is evaluated at exactly 0.5. Careful pruning is done to avoid creating connected components. These random networks are then converted to neural models that take the name of r-TNs (see also Supplementary Section S3).

To better represent the characteristics of the original connectome, a desirable feature of a generator would be the ability to learn the effects of evolutionary optimization on connectome graphs. This would allow the generator to retain the features learned from ad-hoc rewired reference connectomes and produce new ones. To this end, a total of eight sets of rewired connectomes are collected. Each of these sets is created by randomly rewiring a certain percentage of the edges from the original connectome, creating two types of rewired connectome: the first type (*total*) involves the rewiring of the whole edge set, the second one (*latent*), instead, regards only the rewiring of the interneuron-interneuron edge set. These sets are then used to train a fourth class of graph generators, $\mathbf{G}_4$, which is based on Variational Graph Autoencoders (VGAE). This unsupervised Graph Neural Network (GNN) framework uses latent variables to learn meaningful node embeddings incorporating structural information of the input graph. The VGAE models are trained using two slightly different loss functions: a variational lower bound and a regularized version of it. The latter employs a regularization term that slightly shifts the motif distribution learned by the VGAE towards the original connectome.

After training, the $\mathbf{G}_4$ generators are applied to each VGAE model by using the generation procedure $\mathcal{A}_{\mathbf{G}_4}$ (details in Supplementary Section S3-B(a)). First, a set of $T = 2500$ probabilistic adjacency matrices is sampled from the posterior distribution of the generative model, conditioned by the original connectome graph. The generated adjacency matrices are refined with a strategy that enforces generation diversity while maintaining structural similarities with respect to the reference connectome. Finally, a representative set of networks is chosen by sampling a subset of 6 from the 5000 generated networks for each rewiring percentage and type. The networks are generated by sampling from quantiles, based on distances from the original connectome. These distances are measured using the Jensen-Shannon metric applied to the adjacency matrices (see Supplementary Section S3-B(b)). Although the generators can establish different connections between nodes, the number and the characteristics of the nodes remain unchanged, enabling a comparison between the generated adjacency matrix and the original one. In terms of motifs entropy, the Jensen-Shannon distance increases as the level of rewiring in the generated graphs and the distance from the original connectome increase, as shown in Supplementary Figure 2. The chosen graphs are then converted to neural models producing the so-called s-TNs, type *total* and *latent*. (see also Section III-B2).

C. MiDEA: Multi Dyadic Effect Algorithm

MiDEA is an algorithm tailored to analyze evolutionary optimization in both directed and undirected parts of the connectome. It scrutinizes various neuron types and their circuitry to appraise the dyadic/anti-dyadic effect. As an extended version of the algorithm by Park and Barabasi [57], the process of examining evolutionary conservation involves a structured protocol that includes the assessment of motif distribution attributes and entropy. The analyses of dyadic/anti-dyadic effect motifs on complex networks are also described in our previous works on *E.coli* [84]. MiDEA, in its initial phase (refer to Supplementary Section S1-A), takes into account three distinct neuron categories: sensors, motors, and interneurons. It evaluates the functional relationships among these neurons, accounting for connections both within the same neuron type (dyad) and between different types (anti-dyad). The algorithm then calculates the dyadic effect across these three neuron classes and quantifies its comprehensive magnitude $\widehat{m}$ within the studied connectome. In the second phase (refer to Supplementary Section S1-B), the MiDEA algorithm extends the previously described procedure to every potential shortest path between any two nodes in the connectome, resulting in a collection of multi-dyadic effects that capture the influences along various pathways. During the third phase (see Supplementary Section S1-C), the algorithm gauges the information content linked to these multi-dyadic effects, offering insights into their significance and informative qualities. The fourth phase (refer to Supplementary Section S1-D) involves juxtaposing the original connectome with randomly rewired counterparts. This allows for a discerning understanding of the disparities and commonalities between the original and the rewired connectomes. MiDEA, along with the information content analysis, serves as a tool to grasp the evolutionary optimization of the connectomes before being transformed in TNs (*C. elegans* TN, s-TNs or r-TNs). This examination particularly concentrates on the distribution of neuron types within the context of electrical and chemical synapses, offering a methodological approach for investigating evolutionary characteristics and informing the design of neural networks.

SOURCE CODE AVAILABILITY

The Python code necessary to replicate the experiments is available in an online repository, in compliance with the policies of Nature Machine Intelligence. Subsequently, the code will be made publicly available on a GitHub repository and linked to the Elegans.AI website to facilitate its dissemination.

ACKNOWLEDGMENT

The authors wish to thank the CINECA SuperComputing Application and Innovation department (SCAI) of Bologna

(Italy) for granting MARCONI100. FB extends his sincere gratitude to Prof. Maurizio Galluzzo and Marco Morgan Castoldi for their motivation and belief in this research.

REFERENCES

1 Worrell, J. C., Rumschlag, J., Betzel, R. F., Sporns, O., and Mišić, B., "Optimized connectome architecture for sensory-motor integration," *Network Neuroscience*, vol. 1, no. 4, pp. 415–430, 2017.

2 Intelligence, N. M., "Materializing artificial intelligence," *Nature Machine Intelligence*, vol. 2, no. 11, p. 653, 2020.

3 Sporns, O., "The human connectome: a complex network," *Annals of the new York Academy of Sciences*, vol. 1224, no. 1, pp. 109–125, 2011.

4 Donachy, S., "Spiking neural networks: neuron models, plasticity, and graph applications," Master's thesis, Virginia Commonwealth University, 2015.

5 Almomani, A., Alauthman, M., Alweshah, M., Dorgham, O., and Albalas, F., "A comparative study on spiking neural network encoding schema: implemented with cloud computing," *Cluster Computing*, vol. 22, no. 2, pp. 419–433, 2019.

6 Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

7 Davidson, S. and Furber, S. B., "Comparison of artificial and spiking neural networks on digital hardware," *Frontiers in Neuroscience*, vol. 15, p. 651141, 2021.

8 Woźniak, S., Pantazi, A., Bohnstingl, T., and Eleftheriou, E., "Deep learning incorporating biologically inspired neural dynamics and in-memory computing," *Nature Machine Intelligence*, vol. 2, no. 6, pp. 325–336, 2020.

9 Ambroise, M., Buccelli, S., Grassia, F., Pirog, A., Bornat, Y., Chiappalone, M., and Levi, T., "Biomimetic neural network for modifying biological dynamics during hybrid experiments," *Artificial Life and Robotics*, vol. 22, no. 3, pp. 398–403, 2017.

10 Pantazi, A., Woźniak, S., Tuma, T., and Eleftheriou, E., "All-memristive neuromorphic computing with level-tuned neurons," *Nanotechnology*, vol. 27, no. 35, p. 355205, 2016.

11 Kulkarni, S. R. and Rajendran, B., "Spiking neural networks for handwritten digit recognition—supervised learning and network optimization," *Neural Networks*, vol. 103, pp. 118–127, 2018.

12 Hodassman, S., Vardi, R., Tugendhaft, Y., Goldental, A., and Kanter, I., "Efficient dendritic learning as an alternative to synaptic plasticity hypothesis," *Scientific Reports*, vol. 12, no. 1, pp. 1–12, 2022.

13 Lee, C., Panda, P., Srinivasan, G., and Roy, K., "Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning," *Frontiers in neuroscience*, vol. 12, p. 435, 2018.

14 Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G., "Backpropagation and the brain," *Nature Reviews Neuroscience*, vol. 21, no. 6, pp. 335–346, 2020.

15 Yan, Y., Chu, H., Jin, Y., Huan, Y., Zou, Z., and Zheng, L., "Backpropagation with sparsity regularization for spiking neural network learning," *Frontiers in Neuroscience*, vol. 16, 2022.

16 Hernandez, J., Daza, K., and Florez, H., "Spiking neural network approach based on caenorhabditis elegans worm for classification." *IAENG International Journal of Computer Science*, vol. 49, no. 4, 2022.

17 Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A., and Asari, V. K., "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, p. 292, 2019.

18 Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., and Plank, J. S., "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

19 Dayan, P., Abbott, L. F. *et al.*, "Theoretical neuroscience, vol. 806," 2001.

20 Tan, M. and Le, Q., "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

21 He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

22 Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

23 Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

24 Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

25 Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L., "Cvt: Introducing convolutions to vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 22–31.

26 Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., and Jégou, H., "Going deeper with image transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 32–42.

27 Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H., "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

28 Gesmundo, A. and Dean, J., "An evolutionary approach to dynamic introduction of tasks in large-scale multitask learning systems," *arXiv preprint arXiv:2205.12755*, 2022.

29 Kosiorek, A., Sabour, S., Teh, Y. W., and Hinton, G. E., "Stacked capsule autoencoders," *Advances in neural information processing systems*, vol. 32, 2019.

30 Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B., "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

31 Chen, Y., Yun, Z., Ma, Y., Olshausen, B., and LeCun, Y., "Minimalistic unsupervised learning with the sparse manifold transform," *arXiv preprint arXiv:2209.15261*, 2022.

32 Ucar, T., Hajiramezanali, E., and Edwards, L., "Subtab: Subsetting features of tabular data for self-supervised representation learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 853–18 865, 2021.

33 Khacef, L., Rodriguez, L., and Miramond, B., "Improving self-organizing maps with unsupervised feature extraction," in *Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 23–27, 2020, Proceedings, Part II 27*. Springer, 2020, pp. 474–486.

34 Beniaguev, D., Segev, I., and London, M., "Single cortical neurons as deep artificial neural networks," *Neuron*, vol. 109, no. 17, pp. 2727–2739, 2021.

35 Rosenblatt, F., "Perceptron simulation experiments," *Proceedings of the IRE*, vol. 48, no. 3, pp. 301–309, 1960.

36 Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2012.

37 Dalgaty, T., Miller, J. P., Vianello, E., and Casas, J., "Bio-inspired architectures substantially reduce the memory requirements of neural network models," *Frontiers in neuroscience*, vol. 15, p. 156, 2021.

38 Jürgensen, A.-M., Khalili, A., Chicca, E., Indiveri, G., and Nawrot, M. P., "A neuromorphic model of olfactory processing and sparse coding in the drosophila larva brain," *Neuromorphic Computing and Engineering*, vol. 1, no. 2, p. 024008, 2021.

39 Bouvier, G., Aljadeff, J., Clopath, C., Bimbard, C., Ranft, J., Blot, A., Nadal, J.-P., Brunel, N., Hakim, V., and Barbour, B., "Cerebellar learning using perturbations," *Elife*, vol. 7, p. e31599, 2018.

40 Bostrom, N., *Superintelligence*. Dunod, 2017.

41 Sardi, S., Vardi, R., Meir, Y., Tugendhaft, Y., Hodassman, S., Goldental, A., and Kanter, I., "Brain experiments imply adaptation mechanisms which outperform common ai learning algorithms," *Scientific reports*, vol. 10, no. 1, p. 6923, 2020.

42 Della Vecchia, M., Hakim, V., Pagnani, A., and DISAT, P., "Biologically plausible learning algorithm for recurrent neural networks," *eLife*, 2021.

43 Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M., "Neuroscience-inspired artificial intelligence," *Neuron*, vol. 95, no. 2, pp. 245–258, 2017.

44 Barbulescu, R., Mestre, G., Oliveira, A. L., and Silveira, L. M., "Learning the dynamics of realistic models of c. elegans nervous system with recurrent neural networks," *Scientific Reports*, vol. 13, no. 1, p. 467, 2023.

45 Sakamoto, K., Soh, Z., Suzuki, M., Iino, Y., and Tsuji, T., "Forward and backward locomotion patterns in c. elegans generated by a connectome-based model simulation," *Scientific Reports*, vol. 11, no. 1, pp. 1–13, 2021.

46 Xie, S., Kirillov, A., Girshick, R., and He, K., "Exploring randomly wired neural networks for image recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1284–1293.

47 Roberts, N., Yap, D. A., and Prabhu, V. U., "Deep connectomics networks: Neural network architectures inspired by neuronal networks," *arXiv preprint arXiv:1912.08986*, 2019.

48 Hodgkin, A. L. and Huxley, A. F., "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

49 Chahine, M., Hasani, R., Kao, P., Ray, A., Shubert, R., Lechner, M., Amini, A., and Rus, D., "Robust flight navigation out of distribution with liquid neural networks," *Science Robotics*, vol. 8, no. 77, p. eadc8892, 2023.

50 Deng, L., "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

51 Krizhevsky, A., Nair, V., and Hinton, G., "Cifar-10 (canadian institute for advanced research)," 2010.

52 Liu, J., Yuan, Y., Zhao, P., Liu, G., Huo, H., Li, Z., and Fang, T., "Change of motifs in c. elegans reveals developmental principle of neural network," *Biochemical and Biophysical Research Communications*, vol. 624, pp. 112–119, 2022.

53 Kipf, T. N. and Welling, M., "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

54 Adami, C., Qian, J., Rupp, M., and Hintze, A., "Information content of colored motifs in complex networks," *Artificial Life*, vol. 17, no. 4, pp. 375–390, 2011.

55 Varshney, L. R., Chen, B. L., Paniagua, E., Hall, D. H., and Chklovskii, D. B., "Structural properties of the caenorhabditis elegans neuronal network," *PLoS computational biology*, vol. 7, no. 2, p. e1001066, 2011.

56 Ron, D., Singer, Y., and Tishby, N., "The power of amnesia: Learning probabilistic automata with variable memory length," *Machine learning*, vol. 25, no. 2, pp. 117–149, 1996.

57 Park, J. and Barabási, A.-L., "Distribution of node characteristics in complex networks," *Proceedings of the National Academy of Sciences*, vol. 104, no. 46, pp. 17 916–17 920, 2007.

58 Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N., "Big transfer (bit): General visual representation learning," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, 2020, pp. 491–507.

59 Touvron, H., Cord, M., El-Nouby, A., Verbeek, J., and Jégou, H., "Three things everyone should know about vision transformers," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*. Springer, 2022, pp. 497–515.

60 Heinsen, F. A., "An algorithm for routing vectors in sequences," *arXiv preprint arXiv:2211.11754*, 2022.

61 Tseng, C.-H., Liu, H.-C., Lee, S.-J., and Zeng, X., "Perturbed gradients updating within unit space for deep learning," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 01–08.

62 Yuan, K., Guo, S., Liu, Z., Zhou, A., Yu, F., and Wu, W., "Incorporating convolution designs into visual transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 579–588.

63 Chen, M., Peng, H., Fu, J., and Ling, H., "Autoformer: Searching transformers for visual recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 270–12 280.

64 Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., and Wang, Y., "Transformer in transformer," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 908–15 919, 2021.

65 Springenberg, J. T., "Unsupervised and semi-supervised learning with categorical generative adversarial networks," *arXiv preprint arXiv:1511.06390*, 2015.

66 Hinz, T. and Wermter, S., "Inferencing based on unsupervised learning of disentangled representations," *arXiv preprint arXiv:1803.02627*, 2018.

67 Makhzani, A. and Frey, B. J., "Pixelgan autoencoders," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

68 Ji, X., Henriques, J. F., and Vedaldi, A., "Invariant information clustering for unsupervised image classification and segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9865–9874.

69 Abbas, M., El-Zoghabi, A., and Shoukry, A., "Denmune: Density peak based clustering using mutual nearest neighbors," *Pattern Recognition*, vol. 109, p. 107589, 2021.

70 Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P., "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems*, vol. 29, 2016.

71 Watts, D. J. and Strogatz, S. H., "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

72 Sanfeliu, A. and Fu, K.-S., "A distance measure between attributed relational graphs for pattern recognition," *IEEE transactions on systems, man, and cybernetics*, no. 3, pp. 353–362, 1983.

73 Krizhevsky, A. and Hinton, G., "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009.

74 LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

75 Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

76 Hu, X., Chu, L., Pei, J., Liu, W., and Bian, J., "Model complexity of deep learning: A survey," *Knowledge and Information Systems*, vol. 63, pp. 2585–2619, 2021.

77 Kazerouni, I. A., Dooly, G., and Toal, D., "Ghost-unet: An asymmetric encoder-decoder architecture for semantic segmentation from scratch," *IEEE Access*, vol. 9, pp. 97 457–97 465, 2021.

78 Zeiler, M. D., "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

79 Kingma, D. P. and Ba, J., "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

80 Glorot, X. and Bengio, Y., "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

81 Ma, R., Miao, J., Niu, L., and Zhang, P., "Transformed l1 regularization for learning sparse deep neural networks," *Neural Networks*, vol. 119, pp. 286–298, 2019.

82 Chen, B. L., Hall, D. H., and Chklovskii, D. B., "Wiring optimization can relate neuronal structure and function," *Proceedings of the National Academy of Sciences*, vol. 103, no. 12, pp. 4723–4728, 2006.

83 White, J. G., Southgate, E., Thomson, J. N., Brenner, S. *et al.*, "The structure of the nervous system of the nematode caenorhabditis elegans," *Philos Trans R Soc Lond B Biol Sci*, vol. 314, no. 1165, pp. 1–340, 1986.

84 Bardozzo, F., Lió, P., and Tagliaferri, R., "A study on multi-omic oscillations in escherichia coli metabolic networks," *BMC bioinformatics*, vol. 19, no. 7, pp. 139–154, 2018.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- ElegansAIsupplementary.pdf