Neurocomputing 584 (2024) 127598

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Elegans-AI: How the connectome of a living organism could model artificial neural networks

Francesco Bardozzo [a,*], Andrea Terlizzi [a], Claudio Simoncini [a], Pietro Lió [b], Roberto Tagliaferri [a]

[a] NeuroneLab DISA-MIS, University of Salerno, Via Giovanni Paolo II, 132, 84084, Salerno, Italy
[b] Computer Laboratory, University of Cambridge, 15 JJ Thomson Avenue, CB3 0FD, Cambridge, UK

ARTICLE INFO

Communicated by G. Fenza

Keywords:
Artificial connectomes
C.elegans connectome
Connectomic architectures
Deep connectomic networks
Deep neural network transformers
Echo-state transformers
Multi-dyadic motifs

ABSTRACT

This paper introduces Elegans-AI models, a class of neural networks that leverage the connectome topology of the *Caenorhabditis elegans* to design deep and reservoir architectures. Utilizing deep learning models inspired by the connectome, this paper leverages the evolutionary selection process to consolidate the functional arrangement of biological neurons within their networks. The initial goal involves the conversion of natural connectomes into artificial representations. The second objective centers on embedding the complex circuitry topology of artificial connectomes into both deep learning and deep reservoir networks, highlighting their neural-dynamic short-term and long-term memory and learning capabilities. Lastly, our third objective aims to establish structural explainability by examining the heterophilic/homophilic properties within the connectome and their impact on learning capabilities. In our study, the Elegans-AI models demonstrate superior performance compared to similar models that utilize either randomly rewired artificial connectomes or simulated bio-plausible ones. Notably, these Elegans-AI models achieve a top-1 accuracy of 99.99% on both Cifar10 and Cifar100, and 99.84% on MNIST Unsup. They do this with significantly fewer learning parameters, particularly when reservoir configurations of the connectome are used. Our findings indicate a clear connection between bio-plausible network patterns, the small-world characteristic, and learning outcomes, emphasizing the significant role of evolutionary optimization in shaping the topology of artificial neural networks for improved learning performance.

1. 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 study concentrates on artificial connectomes, which involve the structural organization and transformation of neural circuits from natural systems into artificial counterparts to solve learning tasks. The connectome plays a vital role in shaping the behavior of living organisms, as demonstrated by the complex processes involved in neurogenesis, such as the differentiation and migration of neural cells. Previous studies have indicated that the neural connections in the connectome are refined through evolutionary pressure [1]. Consequently, it becomes pertinent to explore whether this kind of optimized structure can be utilized to enhance the efficiency of learning algorithms that are designed in the form of neural networks. Therefore, this paper presents Elegans-AI which utilizes the connectome topology of *Caenorhabditis elegans* (*C. elegans*), a small nematode for classification and reconstruction tasks.

Neuromorphic and connectomic systems. The effort to incorporate biological elements like neuron functions, brain behaviors, and connectomes into artificial learning systems has been a significant scientific challenge [2–4]. However, it is important to recognize that biological learning systems are too complex to be fully replicated by current technology and knowledge [5]. Creating bio-inspired neural models often involves a trade-off between simplifying operations and maintaining key characteristics of these systems [6,7]. A recent study [8] demonstrated that using convolutional layers with 1-dimensional causal convolutions, with up to 1024 artificial neurons, can effectively mimic

* Corresponding author.
E-mail address: fbardozzo@unisa.it (F. Bardozzo).

https://doi.org/10.1016/j.neucom.2024.127598
Received 12 January 2024; Received in revised form 7 March 2024; Accepted 19 March 2024
Available online 24 March 2024

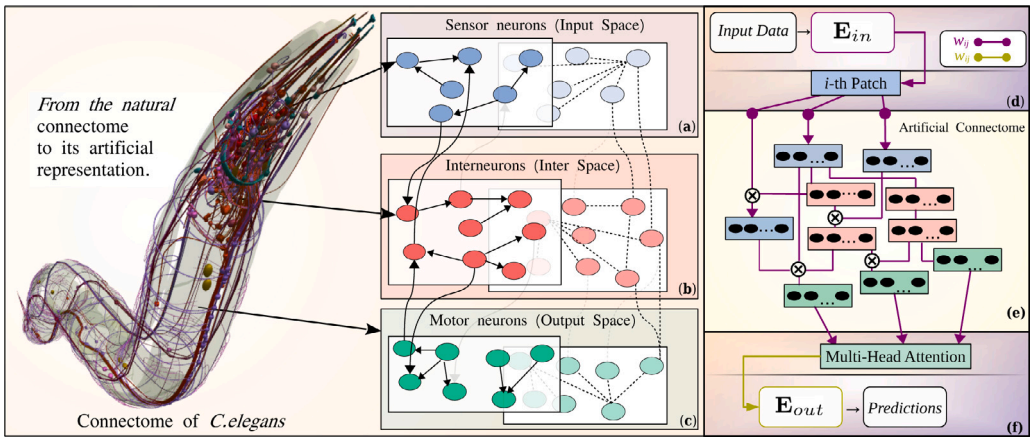


Fig. 1. The connectome of *C.elegans* is represented as a fully connected graph with two overlapping layers, where the solid edges represent chemical and directional synapses and the dashed edges represent electrical and undirected ones. The sensor neurons are represented in blue (Box **(a)**), while 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 architecture of *Elegans-AI* models. In Box **(d)**, the first part of the so-called external operational environment ($\mathbf{E}_{in}$) of *Elegans-AI* is shown. In detail, $\mathbf{E}_{in}$ is an encoder that may vary from the different tasks and generates the feature maps in input to the sensor-tensors space. In Box **(e)**, the artificial connectome is represented as a Tensor Network TN. The TN layered model is a representation starting from the natural connectome. 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 (DAG) and it is depicted as the latent space of our models. Solid lines into the TN show functional associations, links, between tensor units. In echo state networks (ESNs), the violet links (representing weights $\mathbf{w}_{ij}$ between tensor units i-ths and j-ths) are configured as untrainable, while the green links are trainable. Conversely, in deep neural networks (DNNs), both the green and violet links are trainable. The ⊗ 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)**). $\mathbf{E}_{out}$ 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 Figs. 4 and 5, respectively).

the learning behavior of a biological neuron. Neural network designs, inspired by biology, have evolved from focusing on single neurons and their functions [9] to more complex models based on connectomes, which emphasize the connections between multiple computational units [10–13]. Animal and insect nervous systems and connectomes are recognized for their potential in creating optimized learning systems [14–16]. New methods, like deep reservoir networks (DRNs), are emerging for modeling neural networks with graph structures, particularly for understanding the hippocampus's role in classification tasks [17,18], and assessing their biological plausibility. The main efforts to develop artificial learning networks, which replicate bio-inspired mechanisms, fall into two categories: neuromorphic-oriented and connectomic-oriented networks. Neuromorphic-oriented systems are mainly based on the notion of neurons, mimicking how neurons and synapses process and transmit information. On the other hand, connectomic-oriented systems focus on mimicking neural connection maps for developing artificial neural networks and understanding their functions. The key here is the detailed architecture of neural connections, aiming to replicate or use natural patterns to enhance artificial learning methods.

Neuromorphic-oriented structures. Neuromorphic systems encompass various approaches, with Spike Neural Networks (SNNs) being one prominent method. SNNs mimic the nervous system's information communication through spike diffusion [6,7]. Another approach focuses on Deep Neural Networks (DNNs), which enhance synaptic relations by backward-updating learned information [19]. Typically, SNNs and DNNs are evaluated and compared based on their performance and computational costs [20]. A third, more hybrid approach combines the features of SNNs and DNNs. This involves incorporating neural dynamics and time-dependent characteristics into traditional deep learning frameworks, as evidenced in recent studies [21–23]. Concurrently, studies such as Kulkarni and Rajendran [24] and Hodassman et al. [25] argue that the training methods used in DNNs, particularly backpropagation, do not accurately approximate brain function. While other studies integrate backpropagation into SNNs models [26–29]. 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 [30,31]. One of the primary criticisms of DNNs is their requirement for a large number of neurons (trainable parameters) to enhance learning capacity [24], and their lack of architectural and dimensional bio-plausibility [32]. However, 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 [11,12,33] to attention-based transformers [13,34–38]. On the other side, unsupervised reconstruction and/or denoising problems still rely on autoencoder-like architectures [39,40] or encoder–decoder structures [41–43].

Connectomic-oriented structures. The field of connectomic-oriented structures in artificial intelligence has increasingly focused on the *C.elegans* nematode worm, primarily due to its simple yet comprehensively mapped nervous system. Philosopher of science, Nick Bostrom [44], highlighted *C.elegans* as a promising model for connectome-based AI development. In robotics, the abstract representational capabilities of *C.elegans* have been explored, as seen in the works of van Harmelen et al. [45]. In electronics, recent advancements rely in the development of brain organoids being utilized as computational units in reservoir computing systems [46]. The concept of synthetic connectomes, inspired by *C.elegans*, gained traction following reviews like [47,48]. A notable advancement in this field is the work of Sardi et al. [49], which showed that online learning mechanisms, inspired by brain functions and including increased neuronal training frequency, could outperform conventional machine learning methods. Additionally, research by Yan et al. [28] demonstrated the effectiveness of sparse backpropagation algorithms in creating bionic structures that mimic the *C.elegans* nervous system. Although it is not confirmed if *C.elegans* employs a mechanism akin to backpropagation, its neural activities show similarities to recurrent neural networks (RNNs) [50,51], suggesting a viable direction for AI research. For what is concerning Deep Neural Networks (DNNs)

C.elegans connectome as a foundational structure for learning models is shown in [52] and the more recent [53] (works inspired by Li and Talwalkar [54]). Also in the field of Spike Neural Networks (SNNs) [29] developed a *C.elegans*-inspired model. In contrast, to previously trainable approaches, Chahine et al. [55] introduced reservoir feed-foreward networks, always drawing inspiration from the *C.elegans* nervous system. It is important to note that these models have not entirely replicated the evolutionary conserved topology of natural connectomes. For example in [52,53], they incorporate custom skip connections into existing frameworks, such as ResNet [12], rather than fully mimicking the intricate network structures found in nature. On the other hand, recent works like those of Lappalainen et al. [56], show connectome-constrained deep mechanistic networks that incorporate the known fruit fly's connectomic topology, using deep learning techniques to optimize unknown neuronal and synaptic parameters but without showing concrete deep learning applications. Thus a key question remains: do these systems possess structural bio-plausibility that can be used to optimize deep learning models? This concern is crucial for the design of artificial learning models optimized by connectomic-inspired systems [57]. Recent studies, such as those by Lappalainen et al. [56], have been exploring the integration of the Fruit Fly's connectomic structure into deep mechanistic networks. These networks employ deep learning methodologies to fine-tune unidentified neuronal and synaptic parameters, proposing hypotheses for potential connectomic-oriented deep learning applications. As discussed in works like [57] emerges an important research question: Can the structural bio-plausibility of these systems be leveraged to enhance deep learning models? This question is particularly relevant in the context of developing connectome-inspired artificial learning models.

Paper organization and principal contributions. Our research presents a novel approach to integrating the *C.elegans* connectome into Deep Neural Networks (DNNs) and reservoir Echo State Networks (ESNs). We introduce models that incorporate artificial connectomes, aligning with key biological plausibility aspects, including evolutionary patterns and node distribution in connectomic topology [58]. From a neuroscientific perspective, our shift from neuromorphic-based structures to connectome-oriented ones, as undertaken in this study, utilizes these concepts to emulate communication mechanisms found in the hippocampus, with a special emphasis on their role in augmenting learning capabilities [59]. In the Methods Section 2, we provide a detailed account of our methodology. We start by elucidating the process of transforming a biological connectome into an artificial bio-plausible representation. Subsequently, we elaborate on the development of both supervised and unsupervised DNN and ESN Elegans-AIs. We also introduce two distinct methodologies for generating connectomes with varying degrees of similarity to the original. Additionally, we conduct an extensive investigation into the evolutionary conservation of connectomes to gain insights into their optimized architecture for learning tasks.

The Results Section 3 reveals valuable insights into connectomic evolutionary conservation by comparing Elegans-AI with advanced models. It also offers an in-depth analysis of performance across various connectome types, including original, randomly rewired, and simulated versions, using well-established deep-learning benchmarks bridging the gap between bio-inspired models, often limited in performance, and connectome-inspired models, typically lacking structural bio-plausibility.

Our study makes several key contributions to the field of connectomic-oriented artificial intelligence:

- **Model Exploration**: We investigate the learning capabilities of both Deep Neural Network (DNN) and Deep Reservoir Network (DRN) models that incorporate artificial connectomes derived from various sources, i.e. the original *C.Elegans*, randomly rewired versions, and bio-plausible simulated connectomes.

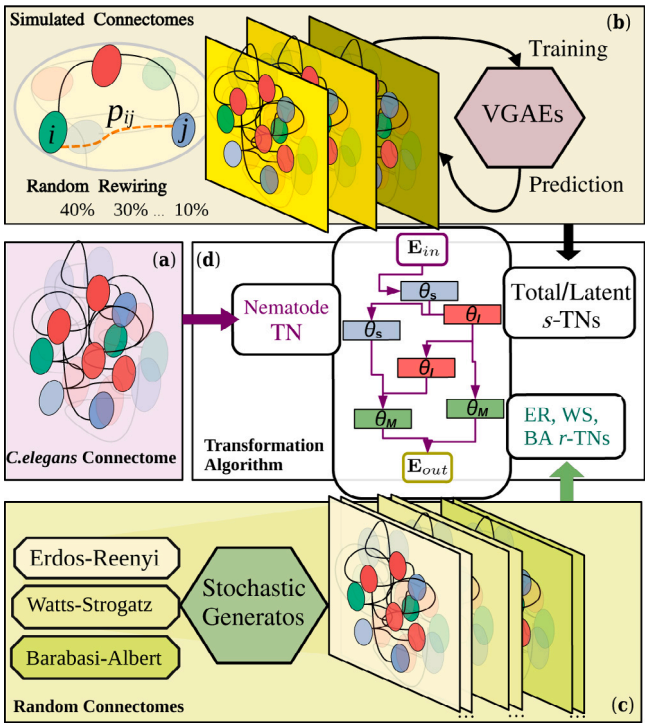


Fig. 2. Fig. 2 provides a schematic representation of the process for generating tensor networks TN using the transformation algorithm 1 from nematode, random, and simulated connectomes (refer to Section 2.5 for more details). The various TNs are integrated into two distinct wrapper models, **M1** and **M2** as described in Section 2.4. In Box **(a)**, the transformation of the nematode connectome directly into a TN is depicted. Meanwhile, Box **(b)** illustrates the process of edge-swap rewiring in the nematode connectome, employing different probabilities p_{ij} for swapping edges between the ith and jth neurons. This edge-swap rewiring can be applied to the entire set of edges (termed *total* edge-swap rewiring) or specifically to connections between interneurons (termed *latent* edge-swap rewiring). For each level of edge-swapping, a VGAE is trained to create simulated connectomes, which are then transformed into corresponding TNs. Box **(c)** focuses on the generation of r-TNs from randomly rewired connectomes, which are produced using stochastic generators. These connectomes maintain the same set of nodes and annotation enrichment as the nematode connectome. Finally, Box **(d)** offers a schematic overview of the models **M1** and **M2**, demonstrating their configurations as either Echo State Networks (ESN) or Deep Neural Networks (DNN). These models incorporate various tensor network sub-architectures, namely nematode TN, r-TN, or s-TN. The customization of these architectures is further refined by integrating sensor, interneuron, and motor tensor units, denoted as θ_S, θ_I, and θ_M, respectively.

- **Evolutionary Structural Explainability**: This approach introduces an interesting method for non-gradient-based evolutionary explainability in artificial learning systems. It involves analyzing and comparing neuron motif distributions and the properties of homophily/heterophily within connectomes, focusing on their learning capabilities. The goal is to gain deeper insights into the effectiveness, scalability, and limitations of artificial connectomes as learning systems, particularly in terms of how they are influenced by evolutionary patterns. This method aims to enhance our understanding of the functioning and development of artificial intelligence models based on bio-plausible neural network structures.
- **Performance Evaluation**: Our study performs learning-oriented experiments and ablation studies on well-established 10-class benchmark datasets, specifically *Cifar10* and *MNIST Unsupervised*. To assess scalability, we expand our analysis to include *Cifar100*. In our evaluations, we consider two distinct memory types: short-term memory (STM) for reservoir models and long-term memory (LTM) for deep learning versions. Both the approaches outperform state-of-the-art (SOTA) models in mostly of the comparisons.

2. Methods

This section starts with an overview of the design process for Elegans-AI models and in which way they are benchmarked. In Section 2.1, it begins with the description of the dataset used for benchmarking our models. Then follows Section 2.2 with the transformation of a connectomic structure, whether it is the *C.elegans* one, bio-plausible (Fig. 2**(b)**) or fully randomized (Fig. 2**(c)**), into tensor networks TNs. In Section 2.3 the TNs are immersed into the latent space of well-known deep-learning models leading to architectures inspired by transformers and autoencoders. These *wrapper* architectures are denoted as the external environments **E** of **M1** (for transformers) and **M2** (for autoencoders). As described in Section 2.4, **M1** and **M2** are specifically designed to address classification and reconstruction problems on images (see Fig. 2**(g)**). As depicted in Figs. 4 and 5, the *wrapper* architectures of the various **E**s differ based on the specific task. However, their latent spaces, the TNs, are consistently designed and structured using the same algorithmic approach, remaining structurally identical across all different tasks. Section 2.5 delves into the extraction, representation, and generation of nematode connectomes. It explains how these connectomes are derived using both stochastic algorithms and Graph Variational Autoencoders (VGAE) as tools for simulating the complex connectomic features and their relations in a connectomic structure. Finally, Section 2.6 introduces a multi-class algorithmic approach to analyze the heterophilic and homophilic properties of these connectomes.

2.1. Benchmark datasets

The transformer-like model $M1$ is employed to solve a classification problem based on the *Cifar10* dataset [60], which consists of 60 000 $32 \times 32 \times 3$ pixel RGB images across 10 classes, with 6000 images per class. According to the official repository, 50 000 images are designated for training and 10 000 for testing purposes. Additionally, $M1$ is also tested on the *Cifar100* dataset, which is similar to *Cifar10* but includes 100 classes containing 600 images each. The *Cifar100* dataset is thus comprised of the same total number of images (60 000), also split into 50 000 for training and 10 000 for testing. Each image in *Cifar100* is a $32 \times 32 \times 3$ pixel RGB image, offering a more challenging classification task due to the higher number of classes.[1] Conversely, the autoencoder-inspired model $M2$ is applied to the *MNIST* dataset [61,62] in an unsupervised manner for image reconstruction. *MNIST* comprises gray-scaled digit images of size 28×28, totaling 60 000 training images and 10 000 testing images, as per the official MNIST repository.[2]

2.2. The C.elegans connectome as a tensor network (TN)

This section details the construction of the TN starting from an artificial graph derived from the mapping of a real connectome, which specifically mimics the structure of a neural circuitry composed of three classes of neurons: sensor labeled as S, interneurons labeled as I, and motor neurons labeled as M (see also Fig. 1). The TN resulting from the collection of connectome/graphs C is constructed by allocating a tensor unit θ for each node/neuron. Therefore, each edge/synapse, chemical or electrical, corresponds to an edge connection at the architectural level between two different tensor units. For the jth connectome **C**, the transformation algorithm ***TA1(C)*** is depicted with the pseudo-code in 1. The first part of the transformation algorithm is an initialization phase which involves scanning all nodes labeled as sensor neurons S on the cth connectome C_c, then assigning the same feature map θ_{init} from the previous layers of the external environment $\mathbf{E}_{in}$ (see Transformation 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 our system to track a non-linear mapping of all the mathematical operations between tensors). In the second part of the algorithm (see Transformation algorithm 1 - *Create Tensor Net*), the cascading scan of the cth adjacency matrix $A_{\hat{C}_c}$ continues, by searching dyadic and anti-dyadic connections. In the first scan, the algorithm searches nodes labeled as motor neurons M and interneurons I which are linked with sensors S. If the ith sensor node is connected to the jth motor neuron

Algorithm 1 Pseudo-code of Tensor Network Algorithm.

Require: The θ tensor unit spec. (shape, type, act. fun.)
Require: The node list V of the c-th connectome $\hat{C}_c$
Require: The adjacency matrix $A_{\hat{C}_c}$ of connectome $\hat{C}_c(v, e)$
Require: The $Alloc(\theta)$ function that returns a boolean value checking if a θ tensor is allocated or not into the computational graph.
Require: The $f_{map}(v)$ function that given a v vertex returns node attributes and its relative position / ID into the nematode connectome.

$\vec{s} \leftarrow None$ ▷ Input Space (Sensors) - Fig. 1 **(a)**
$\vec{i} \leftarrow None$ ▷ Inter Space (Interneurons) - Fig. 1 **(b)**
$\vec{m} \leftarrow None$ ▷ Output Space (Motors) - Fig. 1 **(c)**

function INIT SENSORS($\vec{s}$, θ_{init})
  for $v_i, k_i \leftarrow f_{map}(V[i])$ **do**
    if $v_i[r] = S$ **then**
      $\vec{s}[k_i] \leftarrow \theta_{init}$
  return $\vec{s}$

function CREATE TENSOR NET($\vec{s}, \vec{i}, \vec{m}, A_{\hat{C}_c}, \theta, V$)
  $\vec{\theta}_s = [(\vec{s}, \vec{i}, \vec{m}), (\vec{i}, \vec{s}, \vec{m}), (\vec{m}, \vec{s}, \vec{i})]$
  $\vec{\rho}_l = [(S, I, M), (I, S, M), (M, S, I)]$
  for l **from** $0 \rightarrow len(\vec{\theta}_s)$ **do**
    for $e(i, j) \in A_{\hat{C}_c}$ **do**
      $v_i, k_i \leftarrow f_{map}(V[i])$
      $v_j, k_j \leftarrow f_{map}(V[j])$
      if $v_i[r] = \vec{\rho}_l[l][0] \wedge Alloc(\vec{\theta}_s[l][0][k_i])$ **then**
        if $v_j[r] = \vec{\rho}_l[l][1]$ **then**
          if $!Alloc(\vec{\theta}_s[l][1][k_j])$ **then**
            $\vec{\theta}_s[l][1][k_j] \leftarrow \theta(\vec{\theta}_s[l][0][k_i])$
          if $Alloc(\vec{\theta}_s[l][1][k_j])$ **then**
            $\vec{\theta}_s[l][1][k_j] \leftarrow$
            $\vec{\theta}_s[l][0][k_i] \otimes \vec{\theta}_s[l][1][k_j]$
        if $v_j[r] = \vec{\rho}_l[l][2]$ **then**
          if $!Alloc(\vec{\theta}_s[l][2][k_j])$ **then**
            $\vec{\theta}_s[l][2][k_j] \leftarrow \theta(\vec{\theta}_s[l][0][k_i])$
          if $Alloc(\vec{\theta}_s[l][2][k_j])$ **then**
            $\vec{\theta}_s[l][2][k_j] \leftarrow$
            $\vec{\theta}_s[l][0][k_i] \otimes \vec{\theta}_s[l][2][k_j]$
        if $v_i[r] = v_j[r]$ **then**
          if $!Alloc(\vec{\theta}_s[l][0][k_j])$ **then**
            $\vec{\theta}_s[l][0][k_j] \leftarrow \theta(\vec{\theta}_s[l][0][k_i])$
          if $Alloc(\vec{\theta}_s[l][0][k_j])$ **then**
            $\vec{\theta}_s[l][0][k_j] \leftarrow$
            $\vec{\theta}_s[l][0][k_i] \otimes \vec{\theta}_s[l][0][k_j]$
  return $\vec{\theta}_s$

[1] Cifar10 and Cifar100 repository - Last queried on 6th March 2023.
[2] MNIST official repository - Last queried on 6th March 2023.

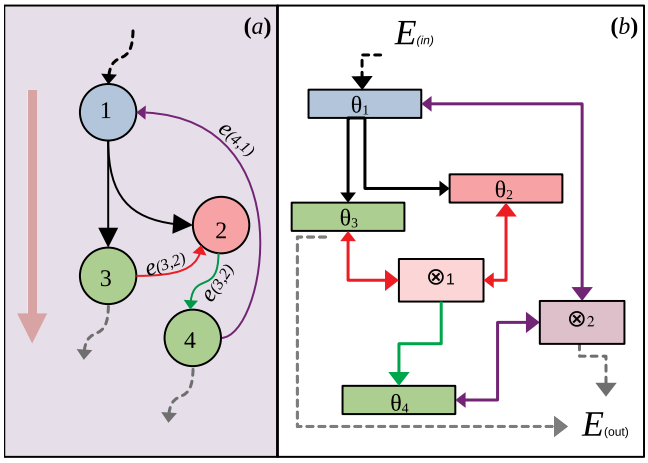


Fig. 3. *Transitioning from the C.elegans connectome in* ***(a)*** *to its Tensor Network (TN) configuration in* ***(b)****:* Fig. 3 depicts the operational framework of TA1 1, which is designed to handle layered feedback loops. This method can be extended to manage multiple nested cycles or feedback synapses which are not suitable in an artificial model. Specifically, while maintaining the flow pattern from sensors to motors, the skip connection ⊗ implemented through **TN1** 1 ensures the retention of connections between neurons, upholds directionality, and prevents the occurrence of endless recursive loops and feedback. In TA1 1, the initial step involves assigning the sensor-representative tensor unit, denoted as θ_1. Following this, the process advances to θ_2, which corresponds to a tensor unit associated with interneuron number 2. Subsequently, the algorithm progresses to the motor-related tensor units θ_3 and θ_4. A primary challenge arises with the need to reassign θ_2, which has already been allocated (and it cannot be reallocated without losing previous connections), to adhere to the synaptic edge $e_{(3,2)}$, indicated by a red arrow. To address this, the algorithm implements a directional skip connection, designated as $\otimes_1$. This connection is crucial for maintaining the integrity of information flowing through both neural pathways. Following the structure of the connectome, the procedure terminates with the formation of an outgoing connection, represented by a green arrow - which facilitates the transition to the next tensor unit to be allocated (θ_4). Then, a similar challenge arises when establishing a connection between θ_4 and the previously allocated θ_1. To address this issue and, in a way, preserve the feedback connection $e_{(4,1)}$ (indicated by a violet arrow), the algorithm employs a similar mechanism as before, but with a necessary adjustment in the flow's directionality - which may affect only chemical synapses. In this instance, the problem is tackled by altering the tensorial information through a skip connection. This modification effectively inverts the directionality from $e_{(4,1)}$ to $e_{(1,4)}$. Such an adjustment is pivotal for maintaining the continuous sensor-to-motor flow, while simultaneously redirecting the output of $\otimes_2$ towards the external environment, denoted as $E_{(out)}$.

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 completing all the possible combinations. When a dyadic connection exists within C_c, the algorithm assigns tensor units, forming edge associations among neurons of identical types (such as sensor to sensor, etc.). As explained in Fig. 3, to manage multiple incoming edges without losing or overwriting previous allocated information between dyads and anti-dyads, the transformation algorithm substitutes the single tensor unit per neuron with an element-wise multiplication of tensor units sharing the same tensor shape. This approach is commonly referred to as a skip connection by multiplication ⊗. As depicted in Figs 1-**e** and in 3-**(b)**, these skip connections by multiplication are symbolized by ⊗. Moreover, as outlined in Fig. 3, the TA1 1 algorithm is capable of resolving cycles and endless feedback loops through skip connections, which are adept at preserving the flow of information from sensors to motors. This strategic redirection ensures that the sensor-to-motor pathway is maintained without interruption and effectively interacts with the external $M1$ and $M2$ wrappers. Contrastingly, in other transformation systems, as cited in [52,53], the principles of connection conservation or the maximal preservation of directionality are not predominant. Commonly, these systems utilize a Directed Acyclic Graph (DAG) framework, wherein neurons possessing solely outgoing edges are classified as input neurons, irrespective of their sensory functions. It is noteworthy that sensors do not invariably possess only outgoing edges. Similarly, neurons functioning as outputs are considered output neurons, regardless of their status as motor neurons, and it is not an absolute rule that motor neurons are restricted to having only outgoing edges. At the end of the transformation process, the output motor unit tensors are collected and stacked. In our learning framework, additional post-processing is applied to the Tensor Network (TN), particularly to address issues such as redundant connections within the computational graphs. This is achieved through the application of the Grappler optimization strategy, as detailed in the TensorFlow whitepaper [63].

2.3. The definition of the external environment $\mathbf{E}_{(in,out)}$

As depicted in Fig. 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 Figs. 4 and 5, the external model components can be conceptualized as an artificial exposome that interacts with the artificial connectomes contained within (**LS** latent spaces in yellow boxes). Alternatively, each ith 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. From the tensor network (TN) to the external environment $\mathbf{E}_{out}$, once the motor tensors are stacked in output from the transformation algorithm 1, they are fed into a multi-head attention layer μ_1 that interfaces with the external environment represented by the $\mathbf{E}_{out}$ (Fig. 1 - **(f)**). The models M1 and M2 employ a μ_1 cross-attention relative to external environmental inputs $\mathbf{E}_{in}$ and their abstraction in the output $\mathbf{E}_{out}$ leveraging higher-level feature diversity [64]. In μ_1 heads the connectome-based asymmetric topological routing from sensory to motor neurons induces structural diversity improving attention focus and model outcomes. Furthermore, the integration of μ_1 layers (as shown in Figs. 4 and 5) are motivated by their correlation with brain regions such as the hippocampus [65,66]; allows for complex representations, underscoring the significance of attention mechanism diversity in advancing model capabilities. These complex representations are observed at the architectural level (see Figs. 1 and 2) as a collection of varied and extensive routes from $\mathbf{E}_{in}$ to $\mathbf{E}_{out}$, systematically arranged in the connectome structured tensor network TN with the transition algorithm ***TA1*** 1 on the specific connectome C, thus $TN = \textbf{\textit{TA1}}(C)$ from sensory to motor neurons. Specifically, a transformer-inspired model $M1 : \mathbf{E}^1_{(in)} \rightarrow TN_q \rightarrow \mathbf{E}^1_{(out)}$ with a number of q tensor networks is employed for image classification [13] (see Fig. 4), while an autoencoder-inspired model $M2 : \mathbf{E}^2_{(in)} \rightarrow TN \rightarrow \mathbf{E}^2_{(out)}$ is utilized for unsupervised digit reconstruction (see Fig. 5). To preserve a clear distinction between the external environments and the tensor 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 of the TNs.

2.4. Model architectures

2.4.1. M1 - Transformer-inspired Elegans-AI for Cifar10 and Cifar100

The architecture of the external environment $\mathbf{E}_{(in,out)}$ and the latent space (**LS**) for $M1$ is shown in Fig. 4. 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 Fig. 4). 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 qth flattened patch ($[p_1, p_2, p_3, p_4]$). In the **LS**, as described in the transformation algorithm (see Section 2.2), all the fully connected layers of the qth TN, named tensor units θs, are allocated with 432 neurons (resulting by flattening the 3 channels × 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 2.2 - **Init Sensors** function), each input flattened patch is assigned to the group of sensor layers (label "S"), one for each TN replica (see Fig. 4 - (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 [34,35,67], and even some parameter-optimized convolutional architectures like EfficientNetV2 [33]. 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 Fig. 4) are arranged by applying a positional embedding layer (Fig. 4 - (d)). Once the features are positionally embedded, they are provided in input to a feature space *condenser* as shown in Fig. 4(e). In both $M1$ and $M2$ setups (see also paragraph 2.4.2), the condenser's role is to merge and reduce the feature space in the output obtained from the TNs. Then, these features are selected for 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 many 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* $C = 10$, and for *Cifar100* $C = 100$). 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.

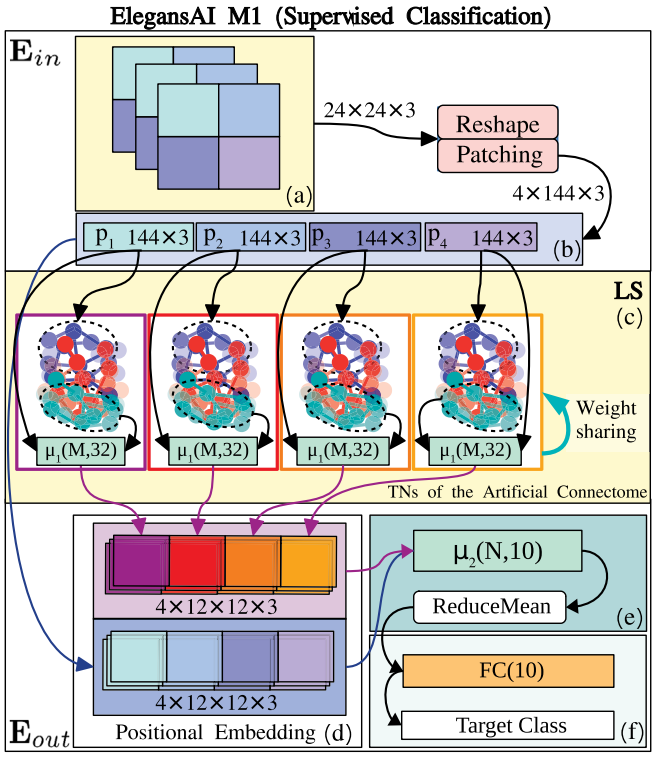


Fig. 4. Architecture of Elegans-AI M1 - Classification Fig. 4 illustrates the architecture of the external environment **E** and the latent space (**LS**) for $M1$. In Fig. 4 - Box $\mathbf{E_{in}}$(**a**), the input layer undergoes patching and reshaping operations to obtain flattened patches. In Box **LS**(c), the patches enter the latent space **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 the *Cifar10* configuration, on $\mathbf{E_{out}}(d)$ the output of each TNs is collected from the θs labeled as motors and reshaped to match the 12 × 12 × 3 size of the initial p_n with $n = [1, 2, 3, 4]$. This is because in **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 **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* (Fig. 4 - (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. In the *Cifar100* configuration the number of neuron-per-layers is switched from 10 to 100 neurons..

2.4.2. M2 - Autoencoder-inspired Elegans-AI for MNIST Unsup

In Fig. 5, an autoencoder-like architecture is depicted, which encompasses a single TN. Compared to the preceding transformer-like architecture (see previous paragraph 2.4.1), where each individual patch was allocated to a different TN, this architecture immerses a single TN directly into the **LS**. Fig. 5 - **(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. Fig. 5 - **(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 bi-linear interpolation for upsampling features (*UpSample*). In Fig. 5 boxes **(a)** and **(d)**, the number of layers in the encoder is less than that in the decoder. This imbalance could provide certain advantages [68]. 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 2.2) by allocating fully connected layers of 784 neurons with an Exponential Linear Unit (*ELU*) activation function. As displayed in Fig. 5**(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 Fig. 5 - **(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 (**FC**(784)) (from the blue path). The latter layer also undergoes 2 layer normalization (Fig. 5 - second green *LayerNorm* in the violet path of Box **(b)**). The tensors output from the black and violet normalized 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 $M1$ (Fig. 4 - Box **(e)**). The tensor in output from layer *Multiply* is fed into a feature space condenser block (Fig. 5(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

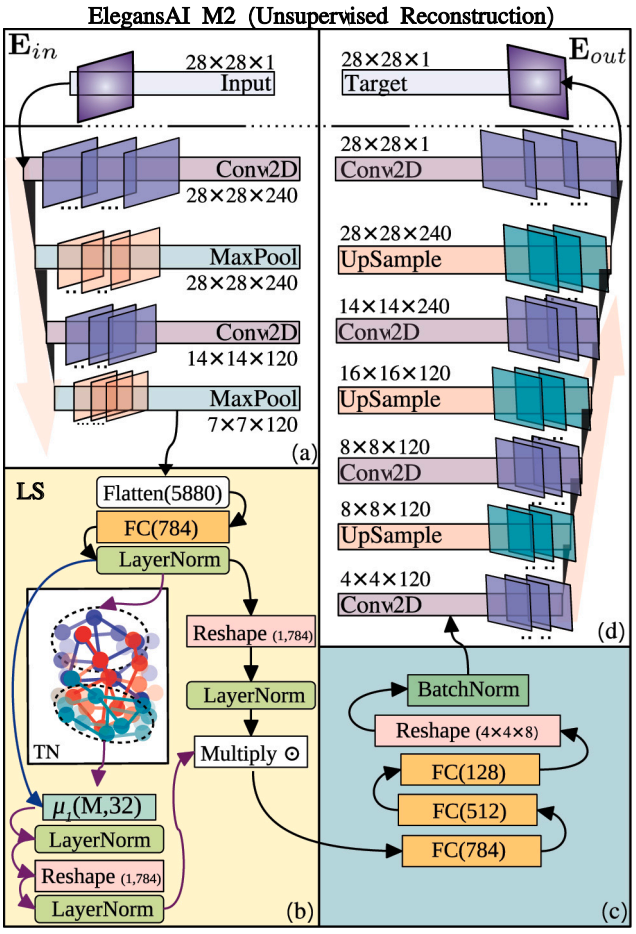


Fig. 5. Architecture of Elegans-AI *M2* - MNIST: In Fig. 5, an autoencoder-inspired architecture is depicted for $M2$. Box **(a)** showcases an encoder situated in the external environment $\mathbf{E}_{in}$, comprised of two successive 2D-convolutional (*2DConv*) layers respectively followed by max-pooling 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 Fig. 4, 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, before being provided as output to the decoder block in $\mathbf{E}_{\mathbf{out}}$ (Box **(d)**). The decoder block, as represented in Fig. 5 - **(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.

features into a tensorial form of $4 \times 4 \times 8$. In Fig. 5 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 [33,39,69]. 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.

2.4.3. The echo state versions of M1 and M2

The architectures described in Sections 2.4.1 and 2.4.2 are transformed into ESN-like models, namely M1 and M2 Echo State Networks (ESNs). Similarly to their DNN counterparts, this modeling approach involves incorporating the non-linear graph topology of artificial connectomes as RNNs in feed-forward and deactivating specific trainable blocks into the $M1$ and $M2$ configurations. However, unlike the original definition of ESNs, in our case, the number of tensor units is equal to the number of neurons, and the connectivity probability is fixed, representing the synaptic connectivity of the nematode connectome (or of those randomly rewired and simulated ones). However, this still adheres to the fundamental definition of ESNs [70] and its extension to deep ESNs [71]. In detail, training is possible only for the blocks that are outputs from the **E***out* layers, whereas the full **E***in* layers and the TN blocks within the artificial connectome are not trainable. This approach reduces the overall count of parameters that can be trained, yet keeps the total parameter count of the model constant in the forward direction. Fig. 1**(d-e-f)** illustrates a visual representation of this concept. It displays a schematic structure of the ESNs $M1$ and $M2$, showcasing the arrangement of neurons and synapses in the connectome. In the figure, in the ESNs configuration, violet weights are non-trainable, whereas green weights are trainable. In the model M1-**E***out*, as depicted in Fig. 4, the trainable layers extend from μ_1 in block **(c)** to the fully connected layer in block **(f)**. This configuration significantly reduces the number of learnable parameters from 107,360,964 in the standard M1 DNN to just 5186 in the *Cifar10* dataset. For the *Cifar100* dataset, the M1 DNN comprises 312,943,440 learning parameters, whereas its ESN variant has a substantially lower count of 34,184,016 parameters. Similarly, in the M2-**E***out* model, illustrated in Fig. 5, the trainable segments span from block **(b)** to block **(d)** in the $M2$ ESN. This results in a total of 1,587,953 learnable parameters, a reduction from the 87,852,914 parameters found in the DNN version of M2.

2.4.4. Trainable and reservoir TN implementations as models for long-term and short-term memory

In the context of $M1$ and $M2$ models, the weights within the TN can be configured as either trainable or non-trainable, contingent upon their designation as connectome-oriented DNNs or ESNs. From a neuroscientific point of view, the encoding of long-term memory models (LTMs) is predominantly achieved through the modulation of synaptic weights, necessitating a structured training model based, i.e., on backpropagation. Conversely, short-term memory models (STMs) are characterized by their reliance on transient network activities. In particular, perturbations in external stimuli can induce prolonged neural activity within a recurrent network framework, resembling an artificial connectome, even after the cessation of the initial stimulus. This mechanism is reflective of the hypothesized role of transient neural activities in supporting STM within cortical networks [72] and it is well-represented by a feed-forward setup. Therefore, investigating how memory impacts learning effectiveness in artificial systems is crucial for the study and design of these architectures in a biologically plausible manner. This necessitates an in-depth examination of the constraints and capabilities of both ESN and DNN architectures, particularly in terms of how variations in network parameters such as size, topology, and input modalities affect the learning trajectory [73]. An analytical comparison of the ESN and DNN configurations delineated in the results (Section 3), aims to elucidate the differential approaches each model employs in *simulating* artificially these memory processes.

2.4.5. 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 [34]. 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 increases the size and diversity of the dataset and prevents overfitting.

2.4.6. Training and hyper-parameter configurations

Models $M1$ and $M2$ of *Elegans-AI* are trained with different parameter configurations and optimization functions. The weights of the convolutional and fully connected layers are initialized with a random distribution [74]. Given the complexity of *Cifar10* and *Cifar100* compared 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* [75]. *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 $\epsilon = 1.0 \times 10^{-7}$. The $M1$ *AdaDelta* optimizer is configured with an initial learning rate lr_{M1} equal to 0.01 for *Cifar10* and 0.0002 for *Cifar100*. On the other hand, for the unsupervised reconstruction problem of MNIST-Unsup, *Adam* [76] 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 number of trainable parameters in DNNs and ESNs structured on the nematode connectome TN is described and compared in paragraph 2.4.3. 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 are applied at different architectures' levels 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 Glorot and Bengio [74], the kernel weights of the last fully connected layer $FC(C)$ (Fig. 4 - Box **f**) are initialized with *Glorot Uniform* distribution. The latter is helpful also in avoiding the exploding gradient problem. For $M2$, a more extensive intervention is required to avoid gradient-related issues. Therefore, in $M2$, all 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 $M2$ (Fig. 5 - Box **(d)** is regularized employing ℓ_1 regularization [77] with a penalty parameter of 0.0001 ($\ell_1 = 1.0 \times 10^{-4}$). In the models, a random image selection with a fixed seed is used to generate the batches. In detail, the batch size of $M1$ is $b_{M1} = 32$ for *Cifar10* and $b_{M1} = 10$ for *Cifar100*, while for $M2$ $b_{M2} = 128$. 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. 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. For fair comparisons, the training and hyperparameters are fixed with the same configurations for DNN and ESN versions.

2.5. Connectomes description and generation

In the initial section, labeled as Section 2.5.1, the manuscript provides a detailed description of the nematode connectome data for *C.elegans*. The subsequent Section 2.5.2 outlines the methodologies employed to generate randomly rewired connectomes. Following this, Section 2.5.3 introduces a specifically tailored Variational Graph Autoencoder (VGAE) [78], designed to generate simulated connectomes that mirror the characteristics of the nematode connectome. Fig. 2 depicts a schematic pipeline of both the procedures to produce *random tensor networks* (r-TN) and *simulated tensor networks* (s-TN) from random and simulated connectomes.

2.5.1. The nematode connectome

The connectome anterior/pharynx part of the hermaphrodite free-living nematode *C.elegans* consisting of 279 neurons and 13.000 chemical and electrical weighted edges are analyzed. The *C.elegans* complex network is accurately reconstructed by Varshney et al. [79],Chen et al. [80] and made free-available on the online repository.[3] Several *C.elegans* nervous systems are provided in the Literature, however, the network of Varshney et al. [79] is chosen as nematode connectome because it is well annotated to the full *C.elegans* network representation of 302 neurons originally provided by White et al. in 1986 [81]. The nematode 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 nematode connectome is enriched by additional information, such as types of neurotransmitters, neuron soma positions,[4] 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 3.2.2). The 3D graphical model of *C.elegans* in Fig. 1 is reproduced by leveraging the tool in [82]. Four different classes of connectomes are transformed into TNs and compared with the TN derived from the nematode connectome as detailed in Sections 2.5.2 and 2.5.3.

2.5.2. Random connectomes and related r-TNs

Three classes of graph generators based on stochastic algorithm are employed: Erdos–Renyi $\mathbf{G}_1$ (ER), Barabasi–Albert $\mathbf{G}_2$ (BA), and Watts–Strogatz $\mathbf{G}_3$ (WS) as schematically shown in Fig. 2 Box **(a-d-e-g)**). 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. 30 graphs are generates using each of $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$, applying careful pruning and edge adding to avoid creating connected components. These random networks are then converted to r-TNs (see also Supplementary Section S3).

2.5.3. Simulated connectomes and related s-TNs

The fourth class of graph generators, $\mathbf{G}_4$, is based on VGAE [78] and it is employed to produce s-TN as depicted in Fig. 2**(b-d)**. In the following, details on the generation procedure and the modeling-experimental

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

[4] 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 1.5 mm. However, in the network of Varshney et al. [79] only the anterior/pharynx part of the worm is considered.

setup used in this portion of the work are provided. The VGAE modeling and experimental setup are described in the first paragraph. While, in the second paragraph, the simulated connectome generation procedure of trained VGAE is described. Adopting VGAE over traditional Graph Neural Networks (GNNs) can be particularly advantageous in contexts where sensitivity to heterophily is necessary [83] - as shown in Section 2.6 the heterophilic/homophilic property is proved in *C.elegans*. VGAE excel in learning latent representations of graphs, capturing complex and diverse structural features beyond simple node similarity or class-based connectivity.

Modeling and experimental setup. Let $G_{origin} = (V_{origin}, E_{origin})$ be the nematode connectome graph, with $|V_{origin}| = N$ nodes. Every node represents a neuron. Let $w : E_{origin} \to \mathbb{R}^+$ be a function that weights edges according to the connectome synapse weights. The rewired connectomes used to train the $\mathbf{G}_4$ models are created by considering 4 different levels of rewiring intensity ($l = 10, 20, 30, 40\%$), either in the whole $E_1 = E_{origin}$ edge set or only in the $E_2 \subset E_{origin}$ edge set (Fig. 2 - **(b)**). The E_2-set includes the interneurons–interneurons synapses as well as interneuron–sensor and interneuron–motor ones. For this reason, a total of 8 different sets of rewired connectomes is collected ($S^1_{10}, S^2_{10}, \dots, S^1_{40}, S^2_{40}$).Where, for every rewiring intensity l, two sets of edges are collected from the nematode connectome that is divided into the two previously defined scopes, E_1 and E_2. Thus, the set S^1_l contains the rewired connectome from scope E_1, while the set S^2_l includes only the edges collected from scope E_2. The VGAE framework consists of two main components: an *inference model* and a *generative model*. The *inference model* q takes the node feature matrix $\mathbf{X}$ and the weighted adjacency matrix $\mathbf{A}$ as input, processing them through two encoders f_μ and f_σ. Each row of the feature matrix $\mathbf{x}_u \in \mathbb{R}^F$ is an F-dimensional vector ($F = 10$), with 9 ($3 + 6$) components corresponding to the one-hot encoding of the role (sensor, motor, interneuron) and cell neurotransmitter type of the u neuron, and 1 component corresponding to the *C.elegans* neuron soma position. Additionally, it is worth noting that each edge $e = (u, v)$, with u, v coordinates in the adjacency matrix $\mathbf{A}$, is weighted according to the synapse weights of the nematode connectome w, when $e \in E_{origin}$. Otherwise, if $e \notin E_{origin}$, it is weighted according to the average weight of the synapses involving the connected nodes u and v in the nematode connectome. The encoders f_μ and f_σ output two matrices $\boldsymbol{\mu}, \log \boldsymbol{\sigma} \in \mathbb{R}^{N \times F'}$, with F' being the size of node embedding. The matrices referred to are the $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma}$ matrices, which respectively represent the mean and log-standard deviation of the latent space multivariate Gaussian distribution. These matrices are calculated using the functions f_μ and f_σ with input matrices $\mathbf{X}$ and $\mathbf{A}$. Thus, the inference model distribution over each node stochastic representation $\mathbf{z}_i$ is defined as an F'-variate normal distribution parametrized by $\boldsymbol{\mu}_i$ and $\text{diag}\left(\boldsymbol{\sigma}_i^2\right)$ as follows:

$$q\left(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}\right) = \prod_{i=1}^{N} \mathcal{N}\left(\mathbf{z}_i \mid \boldsymbol{\mu}_i, \text{diag}\left(\boldsymbol{\sigma}_i^2\right)\right) \tag{1}$$

The entire stochastic representation matrix $\mathbf{Z} \in \mathbb{R}^{N \times F'}$ is obtained using a *reparameterization trick* [84]. Extending the original VGAE architecture [78], the encoders employed f_μ, f_σ are composed of several Reversible Residual [85] Graph Convolutional Network (GCN) [86] blocks (from 80 to 120). Residual connections mitigate the risk of oversmoothing, while the use of a reversible architecture allows for the training of very deep models without encountering memory limitations and lowers the number of parameters. This setup was observed to be the most stable and reliable in the simulated connectomes generation phase 2.5.3. The *generative model* (decoder), is given by an inner product between the stochastic latent representation matrix $\mathbf{Z}$ and itself, on top of which a sigmoid function σ is applied to produce a probabilistic adjacency matrix $\tilde{\mathbf{A}}$ containing $\tilde{A}_{i,j} \in [0, 1]\ \forall i, j = 1, 2, \dots, N$. In addition to the original work [78], a learnable decoder could be leveraged to further process the embeddings in $\mathbf{Z}$ before reconstructing $\tilde{\mathbf{A}}$, which could lead to better reconstruction. Specifically, a simple decoder made up of $K = 5$ point-wise feed-forward layers (similarly to the transformers [13]) is employed, denoted by $\text{FF}_K(\cdot)$. More formally, the decoder distribution is defined as:

$$\begin{aligned} p\left(\mathbf{A} \mid \mathbf{Z}\right) &= \prod_{i=1}^{N} \prod_{j=1}^{N} p\left(A_{i,j} = 1 \mid \mathbf{z}_i, \mathbf{z}_j\right) \\ &= \prod_{i=1}^{N} \prod_{j=1}^{N} \sigma\left(\text{FF}_K\left(\mathbf{z}_i\right)^T \text{FF}_K\left(\mathbf{z}_j\right)\right), \end{aligned} \tag{2}$$

hence the reconstructed probabilistic adjacency matrix is computed as: $\tilde{\mathbf{A}} = \sigma\left(\text{FF}_K\left(\mathbf{Z}\right)^T \text{FF}_K\left(\mathbf{Z}\right)\right)$. Two slightly different loss functions are leveraged to train the generators. The first (traditionally used in the VGAE framework) is the variational lower bound to the parameters of the inference model q and the generative one p:

$$\mathcal{L} = -\mathbb{E}_{(\mathbf{X},\mathbf{A})}\left[\log p\left(\mathbf{A} \mid \mathbf{Z}\right) + \text{KL}\left(q\left(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}\right) \parallel\ p\left(\mathbf{Z}\right)\right)\right] \tag{3}$$

where $-\log p\left(\mathbf{A} \mid \mathbf{Z}\right)$ is the adjacency reconstruction error, and $\text{KL}\left(q\left(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}\right) \parallel\ p\left(\mathbf{Z}\right)\right)$ is the KL-divergence [87] between the inference model distribution and the Gaussian prior $p(\cdot) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The second loss is a regularized version of the first one:

$$\mathcal{L}' = \mathcal{L} - \gamma \log p\left(\mathbf{A}_{origin} \mid \mathbf{Z}\right) \tag{4}$$

Such loss employs a regularization term which slightly shifts and stabilizes the learned graph distribution towards the nematode connectome adjacency matrix $\mathbf{A}_{origin}$ by a small factor of $\gamma \in [0, 1]$. Experimental evidence suggests that a value leading to sufficiently stable generation, without falling into the "Helvetica Scenario" [88], is $\gamma = 0.05$. The models trained to minimize $\mathcal{L}'$ and $\mathcal{L}$ are referred to as *regularized* and *non-regularized*. The corresponding generators are, respectively, denoted as $\mathbf{G}_4(\cdot, \mathcal{L})$ and $\mathbf{G}_4\left(\cdot, \mathcal{L}'\right)$, where $\cdot$ is the rewired connectome set on which the model is trained ($S^1_{10}, S^2_{10}, \dots, S^1_{40}, S^2_{40}$). For example, $\mathbf{G}_4\left(S^2_{30}, \mathcal{L}'\right)$ denotes a regularized generator model trained on the rewired connectomes from S^2_{30} (30% of the G_{origin} edges rewired, only on the latent space). In terms of training configurations, each VGAE model is trained for 2000 epochs using 70% of the corresponding rewired connectome set using the Adadelta [75] optimizer and validated/tested on the remaining 30% (split up between 15% validation set and 15% test set). Early stopping and a dropout rate of 0.1 on each GCN layer are employed to avoid overfitting.

Generation procedure. After the training, the generation procedure $\mathcal{A}_{\mathbf{G}_4}$ (Fig. 2**(c)**) is applied to each VGAE model q:

1. A set $\widetilde{\Sigma} = \left\{\tilde{\mathbf{A}}^{(1)}, \tilde{\mathbf{A}}^{(2)}, \dots, \tilde{\mathbf{A}}^{(T)}\right\}$ of $T = 2500$ probabilistic adjacency matrices is sampled from the posterior distribution of the generative model, conditioned by the nematode connectome graph $G_{original}$:

$$\tilde{\mathbf{A}}^{(i)} = \sigma\left(\text{FF}_K\left(\mathbf{Z}^{(i)}\right)^T \text{FF}_K\left(\mathbf{Z}^{(i)}\right)\right), \text{ where:} \tag{5}$$

$$\mathbf{Z}^{(i)} \sim q\left(\cdot \mid \mathbf{X}_{origin}, \mathbf{A}_{origin}\right), \forall i = 1, \dots, T \tag{6}$$

2. A τ threshold cutoff is applied to each $\tilde{\mathbf{A}}^{(i)}$ to produce a binary adjacency matrix $\mathbf{A}^{(i)}$, and therefore a new set $\Sigma = \left\{\tilde{\mathbf{A}}^{(1)}, \dots, \tilde{\mathbf{A}}^{(T)}\right\}$. Specifically, an ϵ-greedy strategy is employed to select edges to enforce the generation diversity. For each edge (u, v), if $\tilde{A}^{(i)}_{u,v} \geqslant \tau$ then with probability $1 - \epsilon$, $A^{(i)}_{u,v} = 1$, and with probability ϵ, $A^{(i)}_{u,v} = 0$. The exact opposite happens when $\tilde{A}^{(i)}_{u,v} \leqslant \tau$. After each iteration, ϵ decreases exponentially in a simulated annealing fashion. The optimal threshold τ is determined using the AUC-ROC curve based on the VGAE model predictions over the corresponding validation set.
3. Finally, a heuristic based on searching similarities in the nematode connectome is used to post-process the adjacency matrices through electric (undirected) and chemical (directed) synapses (edges). Specifically, for each edge e designated by the ith generated adjacency $\mathbf{A}^{(i)}$:

(a) - if the e belongs to the nematode connectome as a direct/undirect/both synapse, with weight $w(e)$, then e, is added to the simulated connectome as a synapse of the same type(s), with a weight of $\lceil w(e)+\theta \rceil$, where $\theta \sim \mathcal{N}(0,2)$.
(b) - otherwise, if $e=(u,v)$ does not belong to the nematode connectome, then its synapse type it is chosen uniformly across the three options: direct, undirect or both. The weight of the added connection(s) is calculated as the ceiling of the average weight of the synapses involving u and v in the nematode connectome, plus a small $\theta \sim \mathcal{N}(0,2)$.

After the simulated connectomes have been generated, a representative set of networks is chosen. Specifically, among the 5000 networks generated for each rewiring type and level (2500 created with $\mathbf{G}_4(\cdot,\mathcal{L})$ and 2500 created with $\mathbf{G}_4(\cdot,\mathcal{L}')$), a subset $R(\cdot)$ of $|R(\cdot)|=6$ networks is sampled according to distances from the nematode connectome. As an example, $R(S^2_{30})$ contains 6 simulated connectomes sampled among the 5000 generated by $\mathbf{G}_4(S^2_{30},\mathcal{L})$ and $\mathbf{G}_4(S^2_{30},\mathcal{L}')$. 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 nematode one. To quantify the distances, the Jensen–Shannon metric is applied to the adjacency matrices. Such an adjacency matrix-only metric can be employed since the compared graphs completely share the same nodes. Of the selected $|R(\cdot)|=6$, two simulated networks are chosen from the first quartile of distances, two from the fourth quartile, and one each from the second and third quartiles. The chosen graphs are then converted to neural models producing the so-called s-TNs.

2.6. Evolutionary structural explainability: analyzing connectome learning capacity with heterophilic/homophilic effects

The Multi-dyadic Effect Analysis (MiDEA) is a post-training technique used to examine the heterophilic network characteristics of connectome-inspired models. It specifically concentrates on the interactions of multiple dyadic and anti-dyadic motifs within a tensor network (TN). This form of explainability is distinct from gradient-based methods [89]. A motif and learning-based approach is pivotal for deepening our comprehension of natural neural network functions and advancing their artificial counterparts [56,57]. Our algorithm, drawing on the research of Park and Barabasi [90], explores evolutionary optimization in tensor networks (TNs) comprising sensor (S), interneuron (I), and motor (M) nodes, focusing on their heterophilic/homophilic characteristics. This methodology is also applied in our *E. coli* studies [91,92], offers a comprehensive understanding of complex network dynamics. Following TA1's procedures 1, it assesses neuron types and connections to understand dyadic and anti-dyadic effects, comparing these with results from randomly rewired and simulated connectomes after training. This comparison is crucial for comprehending the impact of particular neural configurations on the learning abilities of network models and determining the extent to which the learning capacity of a trained system correlates with its motif structure. Initially, MiDEA (Section 2.6.1) evaluates relationships between neurons and calculates the dyadic effect's magnitude. The second phase (Section 2.6.2) expands this to all shortest paths, analyzing a range of multi-dyadic effects. In the third phase, detailed in Section 2.6.3, it is computed the multi-dyadic information content.

2.6.1. Tensor network multi dyadic/anti-dyadic (heterophilic/homophilic) effect analysis

The dyadic/anti-dyadic effect between tri-class neurons can be explored with a configuration defined as follows: let n_s represent sensor neurons, n_i inter-neurons, and n_m motor neurons, making the total neuron count $N=n_s+n_m+n_i$, where s, m, and i represent sensor, motor, and inter-neuron, respectively. Edges in the connectome, denoted as m, are categorized by neuron synapse typology: $m_{m,m}$, $m_{s,s}$, and $m_{i,i}$. The total edge count in a directed network is $M_d = m_{m,s}+m_{s,m}+m_{m,m}+m_{s,s}+m_{i,m}+m_{m,i}+m_{i,s}+m_{s,i}+m_{i,i}$. An interesting upper bound of M_d is $N(N-1)$. The network density δ_d [93] for a directed graph is the observed M_d divided by $N(N-1)$, as shown in Eq. (7):

$$\delta_d = \frac{M_d}{N(N-1)} \tag{7}$$

In undirected networks, where directionality is absent, the total edge count M_u differs from M_d. Here, heterogeneous connections (e.g., $m_{m,s}=m_{s,m}$) are counted once, leading to an upper bound of $\frac{N(N-1)}{2}$. The network density δ_u is defined in Eq. (8):

$$\delta_u = \frac{2M_u}{N(N-1)} \tag{8}$$

Random variables like $X_{(m,s)}$, $X_{(s,m)}$, and others describe the occurrence of similar or different neuron types in a directed network. The expected interaction value between different neurons is modeled in Eq. (9):

$$E[X_{(s,m)}] = n_s \cdot n_m \cdot \delta_d, \tag{9}$$

with $E[X_{(s,m)}]=E[X_{(m,s)}]$ in undirected networks. Eq. (9) also describes variables like $X_{m,i}$, $X_{i,m}$, etc.

The expected value for the occurrence of the same neuron type is defined in Eqs. (10), (11), and (12):

$$E[X_{(s,s)}] = \frac{n_m(n_m-1)}{2}\delta_d \tag{10}$$

$$E[X_{(m,m)}] = \frac{n_s(n_s-1)}{2}\delta_d \tag{11}$$

$$E[X_{(i,i)}] = \frac{n_i(n_i-1)}{2}\delta_d \tag{12}$$

Substituting δ_d with δ_u in Eqs. (9), (10), (11), and (12) yields a similar formulation for undirected paths. These equations describe the probability of linking neuron pairs according to Gilbert's model for random networks [94,95]. Deviations from expected values in $m_{s,m}$, $m_{m,s}$, etc., suggest non-random neuron arrangements. The magnitude of the dyadic/anti-dyadic effect is measured against its expected value, as shown in Eq. (13):

$$\widehat{m_{s,m}} = \frac{m_{s,m}}{E[X_{(s,m)}]} \tag{13}$$

For dyadic effects, magnitudes $\widehat{m_{s,s}}$ and $\widehat{m_{m,m}}$ are calculated using Eqs. (14) and (15):

$$\widehat{m_{s,s}} = \frac{m_{s,s}}{E[X_{(s,s)}]} \tag{14}$$

$$\widehat{m_{m,m}} = \frac{m_{m,m}}{E[X_{(m,m)}]} \tag{15}$$

Similarly for $\widehat{m_{i,i}}$.

A magnitude $\widehat{m_{x,y}}$ with $x=y$ greater than 1 indicates a dyadic effect, suggesting that such neurons connect more or less densely than expected in a random setup. When $\widehat{m_{x,y}}$ with $x\neq y$ is greater than 1, it shows a heterophilic effect, and less than 1 indicates a homophobic effect.

2.6.2. Shortest neural paths heterophilic/homophilic analysis

Traditionally, the heterophilic/homophilic effect is observed in bi-class neuron types [90]. However, when three distinct types of neurons are considered, this effect extends to all three classes, becoming a dyadic effect on three neuron types. The idea of a multi heterophilic/homophilic effect arises when we average its dyadic effect across neural pathways involving more than two elements. For instance, consider a shortest path **sp** with a depth of n, commencing from a sensor neuron (s_1) and culminating at a motor neuron (m_n): $\mathbf{sp} := s_1 \to i_2 \to i_3 \cdots \to m_n$. In this scenario, one can quantify its anti-dyadic effect ($m_{(s_1 \to m_n)}$) by averaging the interactions of endpoint pairs

(in a 2-by-2 manner) from s_1 to m_n. In this context, our focus shifts to the connectome topology, particularly in identifying and examining the shortest neural pathways. To be precise, we define the shortest paths as those with a maximum of four nodes and three edges connecting any two nodes. This approach allows us to simultaneously investigate evolutionary influences and gain insights into how the dyadic effect extends beyond pairs of nodes.

2.6.3. Tensor network tri-class entropy analysis for heterophilic/homophilic effects

In this phase, the focus is on the impact of rewiring on heterophilic configurations across three classes of neurons and their shortest paths (see 2.6.2), see supplementary materials Tables S3. The information content of the network is expected to be sensitive to varying degrees of rewiring. Accordingly, our entropy model incorporates the dyadic/anti-dyadic effect as a significant event. This is based on the number of edges M_d and M_u, as defined in the earlier Section 2.6.1.

For directed networks, we define S_d as the set of dyadic/anti-dyadic elements: $S_d = [m_{s,m}, m_{m,s}, m_{s,s}, m_{s,i}, m_{i,s}, m_{s,i}, m_{m,i}, m_{i,i}, m_{m,m}]$. Conversely, in undirected networks, S_u comprises elements: $S_u = [m_{s,m}, m_{s,i}, m_{m,i}, m_{i,i}, m_{s,s}, m_{m,m}]$. The cardinality of S_d in the directed case is n^k, where $k = 2$ represents a pair of neurons, and $n = 3$ denotes the three neuron classes. In the undirected scenario, the cardinality of S_u is $\binom{n+k-1}{k}$.

The ith event Y_i, indicating the occurrence of a specific dyadic or anti-dyadic effect within the neuronal network, is calculated as the ratio of each element in S_d (or S_u) to M_d (or M_u). Consequently, the dyadic/anti-dyadic entropy $\widehat{H(Y)}$ is defined in Eq. (16) as:

$$\widehat{H(Y)} = -\sum_{i=1}^{|S|} Y_i log_2\left(Y_i\right) = -\sum_{i=1}^{|S|} \frac{S_i}{M} log_2\left(\frac{S_i}{M}\right), \tag{16}$$

where M is equivalent to M_d (or M_u) and S corresponds to S_d (or S_u). The dyadic/anti-dyadic information content ($IC(Y)$), as defined in Eq. (17), involves a comparative analysis between the entropy of 5000 randomly rewired *C.elegans* networks ($H(Y)_R$) and the nematode network ($\widehat{H(Y)}$):

$$IC(Y) = \widehat{H(Y)}_R - \widehat{H(Y)}. \tag{17}$$

This phase thus quantifies the extent of the network heterophilic effect, offering insights into the structural and informational changes induced in rewired and simulated tensor networks.

3. Results and discussion

In Section 3.1, the study examines models $M1$ and $M2$, which use the nematode TN, comparing them with state-of-the-art models in classification and reconstruction. Section 3.2 conducts a ablation analysis of $M1$ and $M2$'s learning performances with the nematode TN, versus randomized r-TN and simulated versions s-TN. Section 3.3 investigates evolutionary optimization of connectomic architectures, focusing on their heterophilic/homophilic traits in relation to learning performance for non-gradient structural explainability. Similarly, Section 3.4 explores and compares the small-world property of the analyzed connectomic architectures.

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

In Table 1, our transformer-based $M1$ models, encompassing both Deep Neural Network (DNN) and Echo State Network (ESN) versions, exhibit substantial improvements in image classification tasks on the *Cifar10* and *Cifar100* benchmark datasets, surpassing a range of state-of-the-art (SOTA) models in deep learning and machine learning.[5,6]

Table 1
Elegans-AI $M2$ vs. SOTA models for MNIST Unsup.

Model	Top-1 Acc.
Elegans-AI $M2$ DNN (ours)	**99.8**
IIC [100]	99.3[a]
Sparse Manifold Transform [41]	99.3[a]
Elegans-AI $M2$ ESN (ours)	98.5[b]
SubTab [42]	98.3
Stacked Capsule Autoencoder [39]	98.0
Self-Organizing Map [43]	96.9
Bidirectional InfoGAN [99]	96.6
Adversarial Autoencoder [40]	95.9
CatGAN [98]	95.7
InfoGAN [102]	95.0
PixelGAN AE [69]	94.7
Model	**F1 (%)**
Elegans-AI $M2$ DNN (ours)	**99.3**
DenMune [101]	96.6[a]
Elegans-AI $M2$ ESN (ours)	94.9[b]

Highest accuracy is in **bold**.
[a] The second-best.
[b] The third-best.

These models include a variety of architectures, such as classical vision transformers like *ViT, CvT, CaiT, BiT*, and *DeiT* [13,34–37,96], evolutionary-based transformer models like μ2Net [38], and purely convolutional architectures such as *EfficientNetV2* [33].

The DNN and ESN versions of our $M1$ model achieved a remarkable Top-1 accuracy of 99.99% on both the *Cifar10* and *Cifar100* test sets, indicating a near-perfect classification performance. Notably, on the *Cifar10* dataset, our $M1$ model, with fewer training parameters ($107M$), outperformed the second-best transformer, *ViT-H 14* [34], which has a significantly larger parameter count of $623M$. *EfficientNetV2-L* [33], with $121M$ parameters, achieved a Top-1 accuracy of 99.10%, while the *ResNet*-inspired transformer *BiT-L* [96] reached a Top-1 accuracy of 99.37%. On the *Cifar100* dataset, EfficientNet V2-L, with $120M$ parameters, achieved a Top-1 Accuracy of 96.08%, using a robust combination of ad-hoc loss functions that simultaneously minimize loss value and loss sharpness (SAM) [97]. In contrast, our DNN $M1$ model achieved the best accuracy with $313M$ parameters, while the ESN version required only $34M$ parameters. Interestingly, on *Cifar10*, the ESN $M1$ model, with approximately $5K$ learning parameters, demonstrated the same performance as the DNN $M1$.

Moreover, our Elegans-AI DNN $M2$ model has outperformed machine/deep learning-based SOTA models in global benchmarks for unsupervised digit reconstruction. These SOTA models encompass a wide range of machine learning techniques, from autoencoder-like architectures such as Stacked Capsule Autoencoders or Adversarial Autoencoders [39,40], to GAN-based methods like CatGAN, InfoGAN, or PixelGAN [69,98,99], and algorithms based on information theory and topology, like Invariant Information Clustering (IIC) and Sparse Manifold Transform [41,100].

Table 2 presents our $M2$ results in comparison with both deep learning and traditional machine learning models. The results are compiled from the online benchmark repository,[7] except for the Stacked Capsule AutoEncoder (AE) [39], which reports a 40 × 40 MNIST accuracy of 98.7%. Our DNN $M2$ model achieves a Top-1 Accuracy of 99.78% with an MSE of 0.0018, surpassing all other models in Table 2. Additionally, DNN $M2$ exceeds an F1-score of 99.27%, compared to the 96.6% achieved by *DenMune* [101]. The ESN version of $M2$, with an accuracy of 98.5%, does not outperform some models listed in Table 2.

[5] Cifar10 Benchmark dataset.
[6] Cifar100 Benchmark dataset.
[7] MNIST-Unsup - Last queried 6th March 2023.

Table 2
Elegans-AI $M1$ vs. SOTA models for *Cifar10* and *Cifar100*.

Model	Cifar10		Cifar100	
	Top-1 Acc.	Trainable params	Top-1 Acc.	Trainable params
Elegans-AI M1 DNN (ours)	**99.9**	107M	**99.9**	313M
Elegans-AI M1 ESN (ours)	**99.9**	5K	**99.9**	34M
EfficientNet V2-L (SAM) [97]	99.1	121M	96.08[a]	120M
ViT-H/14 [34]	99.5[a]	632M	–	–
μ2Net [38]	99.5[a]	111K	94.95[b]	100K
ViT-L/16 [34]	99.4[b]	307M	–	–
CaiT-M-36 U 224 [36]	99.4	86M	–	–
CvT-W24 [35]	99.4	276.7M	94.09	276.7M
BiT-L [96]	99.4	928M	93.51	928M
ViT-B [103]	99.3	928M	–	–
Heinsen Rout.+BEiT-l. 16 224 [67]	99.2	309.5M	93.8	309.8M
ViT-B/16 [104]	99.1	86M	93.9	86.5M
CeiT-S [105]	99.1	24.2M	–	–
AutoFormer-S 384 [106]	99.1	23M	–	–
TNT-B [107]	99.1	65.6M	–	–
DeiT-B [37]	99.1	86M	–	–
EfficientNetV2-L [33]	99.1	121M	92.3	121M
BPSR SNN ResNet [28]	90.74	260.7M	–	–
Swin-L + ML-Decoder [108]	–	–	95.1	–
ViT-B-16(ImageNet-21K-PT) [109]	–	–	94.2	87M
Astroformer [110]	–	–	93.36	161.75M
CaiT-M-36 U 224 [36]	–	–	93.1	86M
ViT-L(attn fine-tune) [103]	–	–	93.0	306M
TResNet-L-V2 [111]	–	–	92.6	77.1M
EfficientNetV2-M [33]	–	–	92.2	55M
BiT-M(ResNet) [96]	–	–	92.17	235M

Highest accuracy is in **bold**.
[a] The second-best.
[b] The third-best.

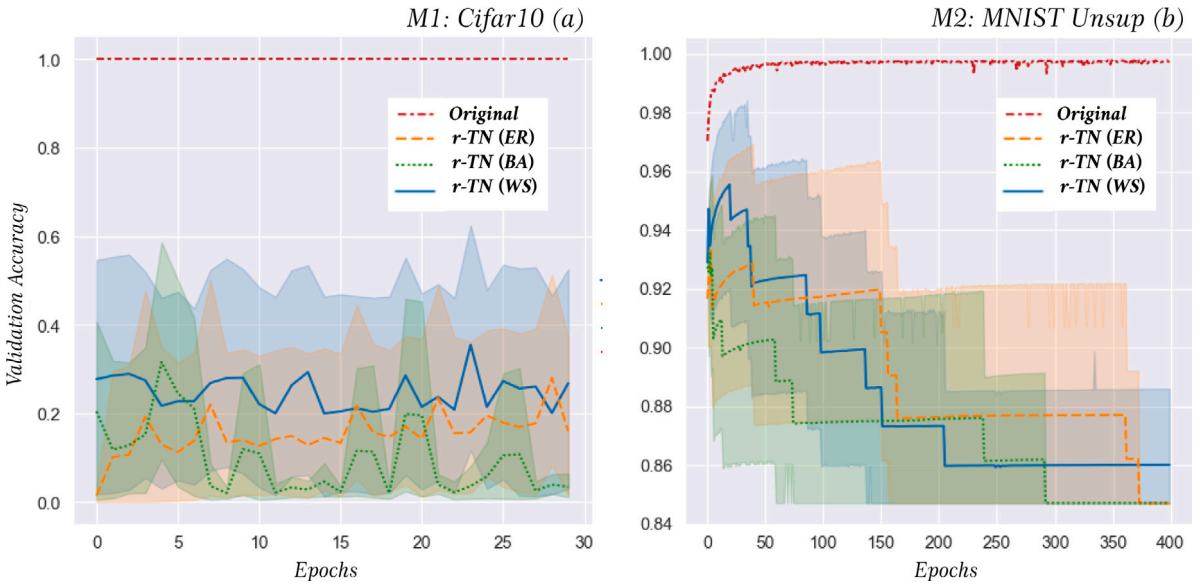


Fig. 6. Fig. 6 shows the comparisons on the validation set by using the Top-1 Accuracy over the number of training Epochs for Elegans-AI *M1* and *M2*, boxes **(a)** and **(b)**, respectively. The models *M1* and *M2* are structured by the immersion of the original TN (red dashed lines) and of 30 random 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 3.2.1.

3.2. Ablation study: Evaluating nematode TN against r-TNs and s-TNs

The ablation study detailed in the document concentrates on evaluating the learning performances of the nematode tensor network (TN), comparing it with both randomly rewired tensor networks (r-TN) and tensor networks simulated by VGAE (s-TN) (see Fig. 2). The primary objective of this study is to dissect and comprehend the influence of varying network structures on the learning efficiency and capabilities of the models. Following this, Section 3.2.1 explores two specific ten-class problems, *Cifar10* and *MNIST Unsup*, to evaluate how models $M1$ and $M2$, when integrated with the nematode TN, compare to the randomly rewired tensor networks (r-TN). Similarly, Section 3.2.2 investigates these two ten-class problems to contrast the performance of $M1$ and $M2$ models using the nematode TN with the VGAE simulated tensor networks (s-TN). Additionally, Section 3.2.3 provides a comparative analysis, examining the performance of $M1$ and $M2$ models with the nematode TN against both r-TN and two variations of s-TN: s-TN-*total* and s-TN-*latent*. As shown in Fig. 2**(d)**, the s-TNs *latent* are

generated by our VGAE models trained on connectomes rewired on the interneuron–interneuron edges only. In the second variant, there are the s-TNs *total* produced by generators trained on totally rewired connectomes (for more details see Section 2.5.3).

3.2.1. Comparing nematode TN with r-TNs

In this section, it is proved that the learning performance of DNN models based on the TN (see Fig. 1) of the nematode connectome is significantly superior to those based on randomly rewired connectomes on both Cifar10 and MNIST-Unsup datasets. In Supplementary Section 2.5.2, the process of generating *random tensor networks* (r-TNs) is detailed. The dimensions of the r-TNs are comparable with the original TN. All model hyperparameters of *Elegans-AI* DNN and ESN $M1$ 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 Fig. 6 can also be interpreted as learning velocity indicators of the effectiveness of the *Elegans-AI* DNN $M1$ and $M2$ models with nematode TN, as they achieve higher performance, in comparison to randomly generated ones (as shown by Figs. 6, 8). Supplementary Figure 3 shows the model convergence speed and accuracy of DNN $M1$ and $M2$ by tracking at which epoch the minimum loss is reached. Concerning the supervised classification problem of *Cifar10* (see Fig. 6), the accuracy on the validation set of the nematode 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 (blue solid line) exhibit slightly better performance, on average, compared to those structured with the Barabasi–Albert (BA) (green dotted lines) or Erdos–Renyi (ER) (orange dashed lines) generative algorithms. Similarly, in *MNIST*-Unsup, the DNN models trained with WS r-TNs follow the higher accuracy values of the nematode connectome only in the first epochs and then gradually decrease. On the other hand, DNN 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 r-TNs. The same experimental set-up is built up by considering the reservoir version of $M1$ and $M2$. Notably, in this scenario, ESN M1 and M2 models based on WS reservoirs demonstrate better performance with a lower number of learning parameters. The WS random networks in ESN $M1$ (Supplementary Figure 5 (a)) match completely the performance of the nematode network at several epochs, in the ESN $M2$ (Supplementary Figure 5 (b), the WS (and also BA) networks slightly enhance the nematode connectome learning capacity, particularly during the initial epochs. What emerges is that the classification of *Cifar10* compared to the unsupervised reconstruction of MNIST creates a gap in predictions within the ESNs, which could be related to the fact that a 10-class classification task can be seen as a short-term memory problem [112,113], whereas reconstruction is more associated with a long-term memory problem [114], which in turn is better addressed with Elegans-AI DNN versions.

3.2.2. Comparing nematode TN with s-TNs

After the comparison between the learning and prediction capabilities of random and nematode connectomes, deep learning generators are trained on optimized nematode connectome motifs to generate new ones. Thus, a second comparison is made evaluating 48 simulated connectomes generated by our ad-hoc VGAE model (see Section 2.5.3). In Fig. 7, Elegans-AI *M1* and *M2* are compared through the Top-1 validation accuracy over the epochs (see also Fig. 7), showing that the trained models based on the nematode 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 Jensen–Shannon distance δ_τ from the nematode connectome. The thresholding criteria for $M1$ and $M2$ and the s-TN separation are detailed in Supplementary Figure 1. Thus, in Fig. 7, 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, aligning with the intuition that networks closer to the naturally optimized one should show better performance. Accordingly, as shown in Supplementary Figure 1, the threshold is decided by observing the distribution of connectomes obtained by leveraging VGAEs.

3.2.3. Comparisons of nematode TN vs. r-TNs and s-TN variants

Fig. 8 illustrates that the simulated tensor networks (s-TNs) outperform the randomly rewired tensor networks (r-TNs) in terms of significant performance gains. This figure presents a comparison focusing on the validation set Top-1 Accuracy across various training epochs for the deep neural networks *M1* and *M2*, depicted in panels (a) and (b), respectively. The models compared include the tensor network based on the nematode connectome (represented by an orange dashed line), the average performance of the random tensor networks (r-TN, shown with a blue line), and the average performance of the simulated tensor networks (s-TN, indicated by red and green dashed lines). Specifically, s-TN *total* and s-TN *latent* refer to s-TN structures generated by VGAE models trained on connectomes with rewired edges either in the entire edge set (the whole connectome) or only in the latent space edge set, which involves rewiring solely the interneurons connections. The data demonstrate that as more randomness is introduced into the network, there is a corresponding decline in performance. While the nematode network achieves high performance, the r-TNs exhibit the poorest performance, while the s-TNs tend to display better average performance, especially when their structure more closely resembles the nematode connectome. This observation is further elucidated in Supplementary Figure 2, which shows that the difference in multi-dyadic motif entropy is correlated with the Jensen–Shannon distance. As the level of rewiring in the s-TN graphs increases, their distance from the nematode connectome also increases, highlighting the impact of structural modifications on performance. Accordingly, it is significant to note that the simulated networks, specifically the s-TN *latent*, which are developed through learning the rewiring of only the interneurons, show better performance compared to those networks that underwent total rewiring. This observation again lends support to the hypothesis that natural optimization processes can enhance the learning performance of DNN $M1$ and $M2$ models on s-TNs - those with long-term memory capabilities. The performance changes drastically when examining the results of reservoir computing $M1$ and $M2$ models, as detailed in Supplementary Figure 6. In these cases, the performance achieved by the reservoir models with s-TN contrasts with the higher effectiveness demonstrated by random tensor networks (r-TNs) on the same problem setups, as also shown in Supplementary Figure 6. In reservoir approaches, this may depend on different small word properties of s-TNs and r-TNs as detailed in Section 3.4.

3.3. Evolutionary optimization of C.elegans connectome in artificial learning systems

In light of the findings from our comparative analysis and ablation study detailed in Section 3.2, a critical question arises: what does it make the evolutionary optimization of the nematode connectome TN and its associated tensor network so significant, particularly when contrasted with other trained models of similar dimensionality but different neuronal organization? Furthermore, why do various TN versions, randomized and simulated with VGAE, exhibit such disparate learning performances? While a definitive answer remains elusive, this section aims to shed light on potential reasons why the *C. elegans* TN exhibits a structurally bio-plausible evolutionary optimized configuration compared to TNs in both deep and reservoir configurations. Section 3.3.1 compares tensor networks using heterophilic/homophilic properties and entropy methods, contrasting the *C. elegans* connectome with its randomized versions to examine evolutionary conservation. Section 3.3.2 delves into the information content

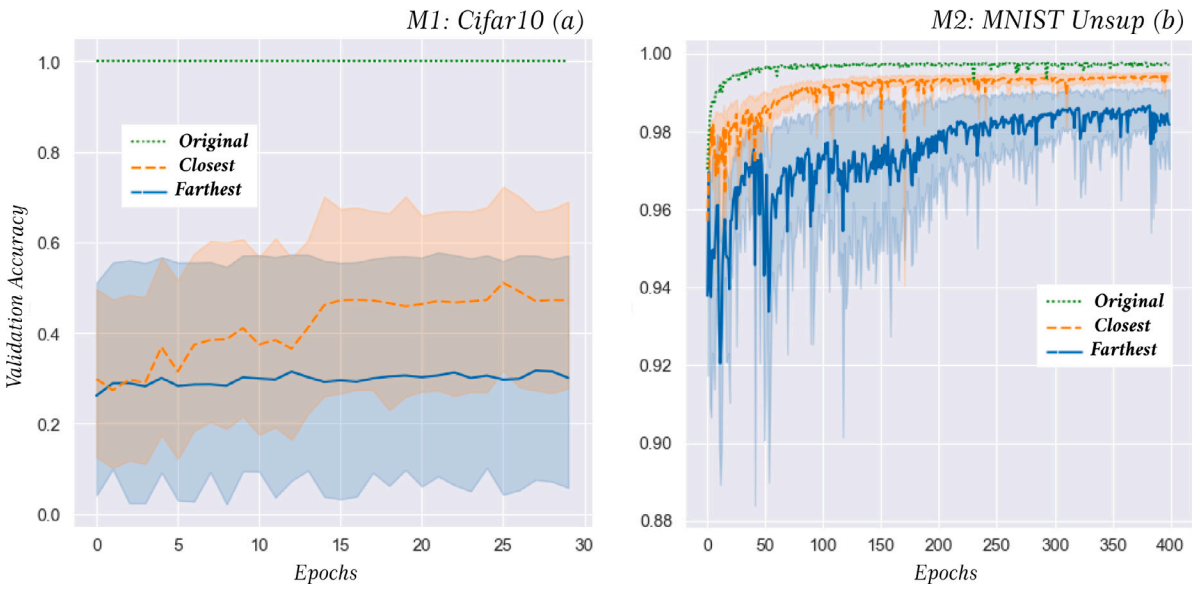


Fig. 7. In Fig. 7 comparisons in terms of validation set's Top-1 average accuracy over the number of training epochs for Elegans-AI *M1* and *M2* are shown in boxes (a) and (b), respectively. The models are structured by the immersion of the original TN (green dashed line) and of two groups of simulated 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 nematode connectome according to the Jensen–Shannon distance, respectively (see Supplementary Section S3-2). The plots show that models based on graphs that are closer to the nematode connectome tend to have considerably better average performance. See also Section 3.2.2.

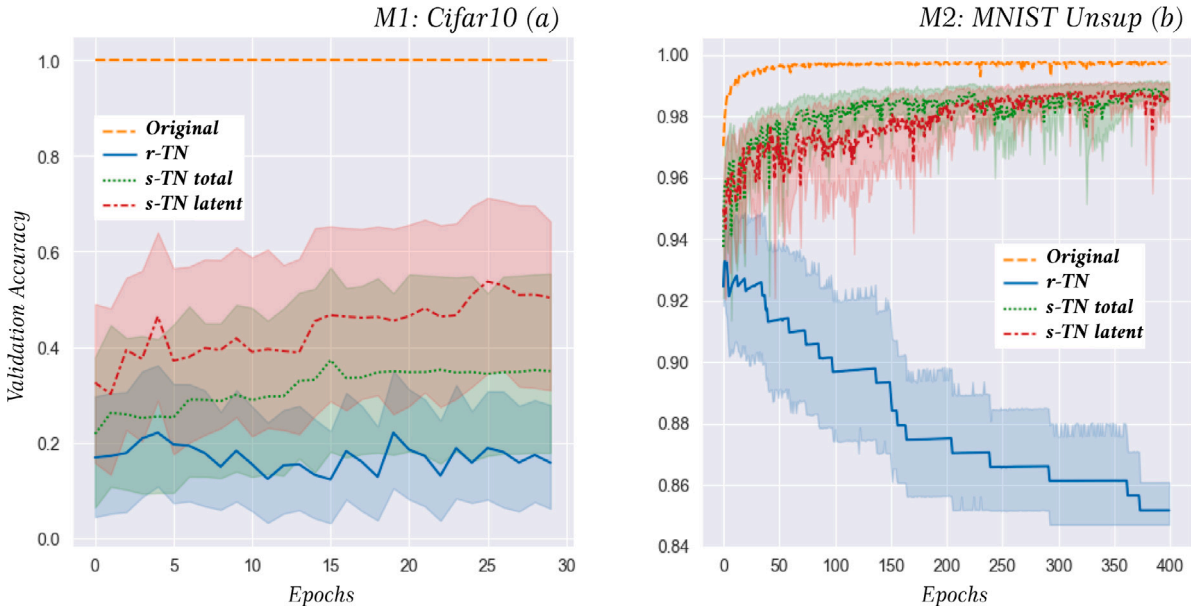


Fig. 8. In Fig. 8 the comparisons in terms of validation set Top-1 Accuracy over the number of training epochs for Elegans-AI *M1* and *M2* are shown in boxes (a) and (b), respectively. The models taken into account are the tensor network based on the nematode connectome (orange dashed line) the average 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 highlight how the more randomness is injected into the network, the more performance degrades. Aside from the high performance reached by the nematode network, r-TNs show the worse performance, while s-TNs seem to show better average performance the more they are close to the nematode connectome.

of the heterophilic/homophilic effect along shortest pathways, aiming to understand structural preservation within networks. Section 3.3.3 explores the relationship between the heterophilic/homophilic features and learning capacity on Cifar10 and MNIST-Unsup benchmarks, evaluating its explanatory power in classification and reconstruction problems, thus contributing to evolutionary structural explainability in TN.

3.3.1. *Heterophilic/Homophilic effect of C.elegans connectome*

Evolutionary optimization varies for functionalities associated with chemical and electrical synapses [57]. The *C.elegans* connectome heterophilic/homophilic distances, compared to randomly rewired ones, demonstrate the non-random organization of neuronal functions at the neuron-type level. There is a noticeable increase in dyadic and anti-dyadic distances for neuron pairs in the nematode connectome versus randomly rewired versions, indicating the inherent importance

of functional interplay among motor, sensory, and interneurons. As indicated in Supplementary Tables S1 and S2, the extent of the heterophilic/homophilic effect intensifies in correlation with an increased percentage of rewiring in random configurations. The data reveals an average diversity of 50% in the nematode connectome compared to the randomly rewired networks for electrical synapses, and a 62% diversity for chemical synapses, highlighting their more pronounced specificity. The MiDEA algorithm demonstrates that electrical synapses are more significantly influenced by the heterophilic effect, whereas the homophilic effect is more pronounced in chemical synapses (see Supplementary Figure S4 - the nematode connectome is indicated with a red dot). This aligns with previous research on *C.elegans* structural motifs [57,79,115]. Moreover, and

3.3.2. Heterophilic/Homophilic effect information content on shortest paths

The heterophilic/homophilic information content (IC), detailed in Supplementary Table S3, shows a progressive increase in heterophilic/homophilic information content along electrical and chemical shortest paths. This 'short memory' property, observed in other biological sequences, is described in [116]. Supplementary Tables S1, S2 also highlight that structured paths in connectomes are highly conserved and depend on synaptic path length. Even minor rewiring disrupts this effect in shortest paths with fewer than four edges. Moreover, variations in shortest path lengths (sp2, sp3, sp4) suggest that evolutionary optimization primarily preserves the heterophilic/heterophobic effect in the nematode connectome, with its relevance diminishing as path length increases. These results are obtained using both Park and Barabasi's dyadic-effect algorithm [90] - only for sp2 - and our extended MiDEA approach (Section 2.6).

3.3.3. Heterophilic/Homophilic effect and prediction performances

In this section, the learning ability as measured by classification/reconstruction accuracy on the validation set and the influence of heterophilic/homophilic connections is explained. In detail, the focus is on analyzing the nematode TN, r-TNs, and s-TNs, specifically looking at their chemical and electrical synaptic connections separately within our three-colored network (see Methods 2.6). Fig. 9 illustrates how chemical and electrical synapses, through their respective heterophilic effects, have distinct impacts on the prediction performances of s-TNs with respect to the nematode connectome by underlying the specific contributions of different wiring configuration of the distinct synaptic types. Moreover, as illustrated in Supplementary Materials Figure 3 (red dots), the nematode's neural network exhibits a pronounced heterophilic effect in its electrical connections compared to its chemical ones, and this property rapidly degrades even with small percentages of random rewiring on r-TNs. The heterophilic/homophilic property is slightly better preserved in the chemical connections (see 3.3.1), and this analysis reveals a clear separation between the two bands in the learning performances. Conversely, in electrical synapses, the separation in performance deteriorates. These results suggest that including evolutionary characteristics in designing learning architectures equal or similar to those in the nematode connectome, particularly regarding chemical and electrical heterophilic/homophilic features, could enhance the learning performances. This concept highlights the potential benefits of mirroring biological structures in artificial intelligence systems.

3.4. Small-world property of the different TNs

The ablation study detailed in Section 3.2 and the findings in Section 3.2.1 reveal that incorporating r-TN with small-world properties into reservoir configurations boosts learning performance with a reduced computational time. This improvement is especially pronounced in tasks demanding short-term memory for nonlinear predictions, paralleling patterns seen in human cortical connectivity [117]. Conversely, in long-term memory tasks, as discussed in Section 3.2.2 and compared in Section 3.1, DNN models $M1$ and $M2$ equipped with s-TN configurations outperform other setups, achieving faster convergence. The r-TN connectomes in $M1$ and $M2$, characterized by their small-world properties similar to the *C.elegans* connectome [118], can suggest a potential link between the small-world property and improved learning performance. The existence or lack of small-world characteristics in a network might account for the varying results observed in reservoir configurations, as depicted in Supplementary Figs. 5 and 6. In these figures, the tensor networks (TNs) are exclusively feed-forward, contrasting with the deep configurations detailed in Section 3.2.3. This observation is also consistent with the results seen in DNN with rewired tensor networks (r-TN), specifically, in the WS (Watts–Strogatz) networks (the blue bands in Fig. 6) the small-world property has been extensively proved [119]. In detail, a stochastic network is considered to have small-world properties if it combines the efficiency of random networks (short average path lengths) with the robustness of regular networks (high clustering), thereby offering a balance between randomness and structural order. Conversely, the presence of a high average path length coupled with a low clustering coefficient in a network often suggests a structure that is more characteristic of a scale-free network. In such networks, there are a few highly connected nodes (hubs) that dominate the network, but local clusters are less prevalent. Our analysis reveals that the nematode connectome shows an average path length of approximately 2.87 with a clustering coefficient of 0.21. In contrast, the simulated connectomes have an average path length of about 3.25 and a clustering coefficient of approximately 0.15, with a p-value of about 0.85×10^{-3}. Meanwhile, random connectomes demonstrate small-world characteristics with an average path length of approximately 1.66 and a clustering coefficient of 0.38, with a p-value of about 0.10×10^{-5} (also see Supplementary Table 4). In summary, the performance of the random networks r-TNs in a reservoir configuration more closely mirrors that of the nematode connectome than the simulated networks s-TNs. On the other hand, the s-TN models, when more closely aligned with the topology of the nematode connectome in DNN configurations (as evidenced by multi-dyadic motif analyses detailed in Section 3.3.2), show the best performance in comparative evaluations.

4. Conclusion

The study compares biological neural networks with bio-inspired artificial ones, focusing on their distinct long-term and short-term memory capabilities, as seen in deep learning and reservoir computing models, respectively. This comparison underscores the complex, efficient, robust, and adaptable nature of biological networks when applied to artificial intelligence systems. The research emphasizes the necessity of considering both the topology and the heterophilic/homophilic properties of neural connections in designing advanced AI models that are connectome-inspired. The research demonstrates how to efficiently integrate biological connectomes into artificial neural networks, utilizing advanced techniques such as deep neural network transformers, deep echo-state transformers, attention-based encoders/decoders, and echo-state encoders/decoders as wrappers. This integration has resulted in improved outcomes in both classification and reconstruction tasks, showcasing scalability from simpler datasets like *Cifar10* to more complex ones such as *Cifar100*. The efficacy of this approach is highlighted by the *M1* model's success in classification tasks within a short-term memory setting (reservoir configuration) and the *M2* model's notable achievements in reconstruction tasks on the MNIST Unsup dataset, utilizing a long-term memory configuration. In the reservoir configuration, our models significantly reduce the number of trainable parameters always reaching a good accuracy. Thus, such an approach not only improves computational efficiency but also maintain the model's generalization capabilities. Integrating evolutionary optimization principles from biological network organizations, which have evolved to process information efficiently, further refines this efficiency. This strategy suggests a promising pathway for developing

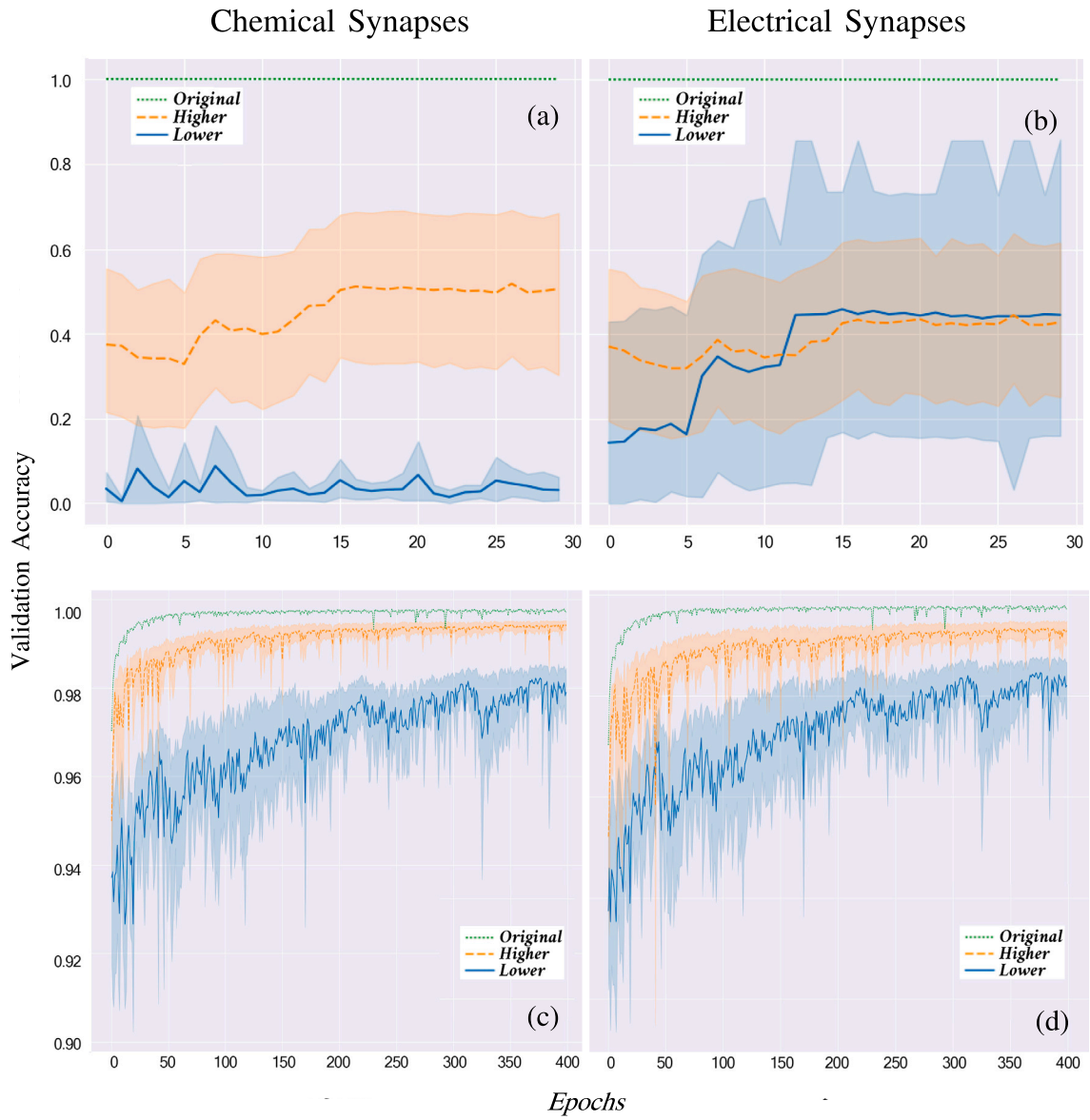


Fig. 9. Comparing Heterophilic Effects Relative to Learning Performance in Elegans-AI Models: Fig. 9 display the validation set's Top-1 average accuracy over the training epochs for Elegans-AI models M1 (**(a-b)**) and M2 (**(c-d)**) on *Cifar10* and *MNIST-Unsup*. The distinct bands (blue and orange) represent the models' performances separated by thresholding of the heterophilic effect measured on nematode and s-TNs. The M1 and M2 performances based on the TN of the nematode are in green. In detail, the MiDEA algorithm evaluated the motif patterns (effects) separately on the electrical (undirected) and chemical (directed) shortest paths of length 3 (three neurons). Successively the s-TNs are grouped based on the normalized intensity of the heterophilic effects, into two different bands (orange range - higher ($\geqslant$0.5), blue range - lower (<0.5)). See also Section 3.3.3.

more resource-efficient and potentially more capable artificial learning systems. Our work outlines models and a methodological approach aimed at enhancing both the scalability and the adaptability of integrating additional natural connectomes, such as that of the full annotated fruit fly larvae [120] while acknowledging the extensive complexity and computational challenges involved. This is in harmony with prior studies on reservoir connectomic-inspired models applied to chaotic time series, as referenced in [121]. Going forward, we plan to further develop our models to investigate specific functional areas within the fruit fly connectome using the **TA1** transformation algorithm. Such a bio-inspired approach promises to pave the way for precise research endeavors targeting specific connectomic functional areas, including the visual or olfactory systems, as in [122]. On one hand, ongoing research is required to fully understand the mechanisms underlying neural network functioning and to advance artificial networks that can closely mimic the complexity and adaptability of biological systems. On the other hand, the structural properties of connectomic-oriented neural networks are crucial for their functional training performance, and they are capable of representing various forms of bio-plausibility, with or without backpropagation, warranting further investigation in future studies. Moreover, our study underscores the significance of varying learning capacities across deep and reservoir configurations. This approach offers valuable insights into evolutionarily optimized neural circuitry. We present this as a form of post-training connectomic explainable AI, grounded in the analysis of multi-dyadic magnitudes, entropic motifs, and small-world properties. This comprehensive analysis contributes to a deeper understanding of how these complex network structures influence learning and memory processes in artificial intelligence systems. The findings of our research indicate a promising direction for future studies in both neuroscience and artificial neural networks highlighting the potential for a more profound integration of biological and artificial systems, suggesting that this approach could

yield significant advancements in our understanding and development of automatic learning systems.

Source code availability

The Python code is available upon motivated request at the GitHub repository in compliance with the Journal's policies and distributed under the Apache License 2.0.

CRediT authorship contribution statement

Francesco Bardozzo: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Andrea Terlizzi:** Writing – original draft, Software, Methodology, Data curation. **Claudio Simoncini:** Writing – original draft, Supervision. **Pietro Lió:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Methodology. **Roberto Tagliaferri:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors wish to thank the CINECA SuperComputing Application and Innovation Department (SCAI) of Bologna (Italy) for granting MARCONI-100. FB oberlunar extends his sincere gratitude to Prof. Maurizio Galluzzo, Roberta Pelachin and Marco Morgan Castoldi for their motivation and belief in this research.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neucom.2024.127598.

References

[1] J.C. Worrell, J. Rumschlag, R.F. Betzel, O. Sporns, B. Mišić, Optimized connectome architecture for sensory-motor integration, Netw. Neurosci. 1 (4) (2017) 415–430.
[2] H. Mizutani, M. Ueno, N. Arakawa, H. Yamakawa, Whole brain connectomic architecture to develop general artificial intelligence, Procedia Comput. Sci. 123 (2018) 308–313.
[3] H. Yamakawa, M. Osawa, Y. Matsuo, Whole brain architecture approach is a feasible way toward an artificial general intelligence, in: Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part I 23, Springer, 2016, pp. 275–281.
[4] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[5] O. Sporns, The human connectome: a complex network, Ann. NY Acad. Sci. 1224 (1) (2011) 109–125.
[6] S. Donachy, Spiking Neural Networks: Neuron Models, Plasticity, and Graph Applications (Master's thesis), Virginia Commonwealth University, 2015.
[7] A. Almomani, M. Alauthman, M. Alweshah, O. Dorgham, F. Albalas, A comparative study on spiking neural network encoding schema: implemented with cloud computing, Cluster Comput. 22 (2) (2019) 419–433.
[8] D. Beniaguev, I. Segev, M. London, Single cortical neurons as deep artificial neural networks, Neuron 109 (17) (2021) 2727–2739.
[9] F. Rosenblatt, Perceptron simulation experiments, Proc. IRE 48 (3) (1960) 301–309.
[10] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2012) 84–90.
[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015, CoRR abs/1512.03385. URL: http://arxiv.org/abs/1512.03385. arXiv:1512.03385.
[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[14] T. Dalgaty, J.P. Miller, E. Vianello, J. Casas, Bio-inspired architectures substantially reduce the memory requirements of neural network models, Front. Neurosci. 15 (2021) 156.
[15] A.-M. Jürgensen, A. Khalili, E. Chicca, G. Indiveri, M.P. Nawrot, A neuromorphic model of olfactory processing and sparse coding in the drosophila larva brain, Neuromorphic Comput. Eng. 1 (2) (2021) 024008.
[16] G. Bouvier, J. Aljadeff, C. Clopath, C. Bimbard, J. Ranft, A. Blot, J.-P. Nadal, N. Brunel, V. Hakim, B. Barbour, Cerebellar learning using perturbations, eLife 7 (2018) e31599.
[17] L. Petrauskas, M. Cucchi, C. Grüner, F. Ellinger, K. Leo, C. Matthus, H. Kleemann, Nonlinear behavior of dendritic polymer networks for reservoir computing, Adv. Electron. Mater. 8 (3) (2022) 2100330.
[18] N. Cazin, M. Llofriu Alonso, P. Scleidorovich Chiodi, T. Pelc, B. Harland, A. Weitzenfeld, J.-M. Fellous, P.F. Dominey, Reservoir computing model of prefrontal cortex creates novel combinations of previous navigation sequences from hippocampal place-cell replay with spatial reward propagation, PLoS Comput. Biol. 15 (7) (2019) e1006624.
[19] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[20] S. Davidson, S.B. Furber, Comparison of artificial and spiking neural networks on digital hardware, Front. Neurosci. 15 (2021) 651141.
[21] S. Woźniak, A. Pantazi, T. Bohnstingl, E. Eleftheriou, Deep learning incorporating biologically inspired neural dynamics and in-memory computing, Nat. Mach. Intell. 2 (6) (2020) 325–336.
[22] M. Ambroise, S. Buccelli, F. Grassia, A. Pirog, Y. Bornat, M. Chiappalone, T. Levi, Biomimetic neural network for modifying biological dynamics during hybrid experiments, Artif. Life Robot. 22 (3) (2017) 398–403.
[23] A. Pantazi, S. Woźniak, T. Tuma, E. Eleftheriou, All-memristive neuromorphic computing with level-tuned neurons, Nanotechnology 27 (35) (2016) 355205.
[24] S.R. Kulkarni, B. Rajendran, Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization, Neural Netw. 103 (2018) 118–127.
[25] S. Hodassman, R. Vardi, Y. Tugendhaft, A. Goldental, I. Kanter, Efficient dendritic learning as an alternative to synaptic plasticity hypothesis, Sci. Rep. 12 (1) (2022) 1–12.
[26] C. Lee, P. Panda, G. Srinivasan, K. Roy, Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning, Front. Neurosci. 12 (2018) 435.
[27] T.P. Lillicrap, A. Santoro, L. Marris, C.J. Akerman, G. Hinton, Backpropagation and the brain, Nat. Rev. Neurosci. 21 (6) (2020) 335–346.
[28] Y. Yan, H. Chu, Y. Jin, Y. Huan, Z. Zou, L. Zheng, Backpropagation with sparsity regularization for spiking neural network learning, Front. Neurosci. 16 (2022).
[29] J. Hernandez, K. Daza, H. Florez, Spiking neural network approach based on caenorhabditis elegans worm for classification, IAENG Int. J. Comput. Sci. 49 (4) (2022).
[30] M.Z. Alom, T.M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M.S. Nasrin, M. Hasan, B.C. Van Essen, A.A. Awwal, V.K. Asari, A state-of-the-art survey on deep learning theory and architectures, Electronics 8 (3) (2019) 292.
[31] C.D. Schuman, T.E. Potok, R.M. Patton, J.D. Birdwell, M.E. Dean, G.S. Rose, J.S. Plank, A survey of neuromorphic computing and neural networks in hardware, 2017, arXiv preprint arXiv:1705.06963.
[32] P. Dayan, L.F. Abbott, et al., Theoretical Neuroscience, Vol. 806, MIT Press, Cambridge, MA, 2001.
[33] M. Tan, Q. Le, Efficientnetv2: Smaller models and faster training, in: International Conference on Machine Learning, PMLR, 2021, pp. 10096–10106.
[34] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, 2020, arXiv preprint arXiv:2010.11929.
[35] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, L. Zhang, Cvt: Introducing convolutions to vision transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 22–31.
[36] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, H. Jégou, Going deeper with image transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 32–42.

[37] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, in: International Conference on Machine Learning, PMLR, 2021, pp. 10347–10357.

[38] A. Gesmundo, J. Dean, An evolutionary approach to dynamic introduction of tasks in large-scale multitask learning systems, 2022, arXiv preprint arXiv:2205.12755.

[39] A. Kosiorek, S. Sabour, Y.W. Teh, G.E. Hinton, Stacked capsule autoencoders, Adv. Neural Inf. Process. Syst. 32 (2019).

[40] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, Adversarial autoencoders, 2015, arXiv preprint arXiv:1511.05644.

[41] Y. Chen, Z. Yun, Y. Ma, B. Olshausen, Y. LeCun, Minimalistic unsupervised learning with the sparse manifold transform, 2022, arXiv preprint arXiv:2209.15261.

[42] T. Ucar, E. Hajiramezanali, L. Edwards, Subtab: Subsetting features of tabular data for self-supervised representation learning, Adv. Neural Inf. Process. Syst. 34 (2021) 18853–18865.

[43] L. Khacef, L. Rodriguez, B. Miramond, 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.

[44] N. Bostrom, Superintelligence, Dunod, 2017.

[45] M. van Harmelen, M. van der Meer, M. Boon, J. Gerbscheid, A. Visser, et al., Hunting a robot controlled by an artificial brain, in: Proceedings of the 27th Belgian-Netherlands Conference on Artificial Intelligence (BNAIC 2015), Hasselt, Belgium, 2015.

[46] H. Cai, Z. Ao, C. Tian, Z. Wu, H. Liu, J. Tchieu, M. Gu, K. Mackie, F. Guo, Brain organoid reservoir computing for artificial intelligence, Nat. Electron. (2023) 1–8.

[47] I. Rabinowitch, What would a synthetic connectome look like? Phys. Life Rev. 33 (2020) 1–15.

[48] I. Rabinowitch, Synthetic connectomes at the interface: Reply to comments on "what would a synthetic connectome look like?", Phys. Life Rev. 33 (2020) 30–33.

[49] S. Sardi, R. Vardi, Y. Meir, Y. Tugendhaft, S. Hodassman, A. Goldental, I. Kanter, Brain experiments imply adaptation mechanisms which outperform common AI learning algorithms, Sci. Rep. 10 (1) (2020) 6923.

[50] M. Della Vecchia, V. Hakim, A. Pagnani, P. DISAT, Biologically plausible learning algorithm for recurrent neural networks, eLife (2021).

[51] D. Hassabis, D. Kumaran, C. Summerfield, M. Botvinick, Neuroscience-inspired artificial intelligence, Neuron 95 (2) (2017) 245–258.

[52] N. Roberts, D.A. Yap, V.U. Prabhu, Deep connectomics networks: Neural network architectures inspired by neuronal networks, 2019, arXiv preprint arXiv:1912.08986.

[53] C. Park, J.S. Kim, Caenorhabditis elegans connectomes of both sexes as image classifiers, Exp. Neurobiol. 32 (2) (2023) 102.

[54] L. Li, A. Talwalkar, Random search and reproducibility for neural architecture search, in: Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 367–377.

[55] M. Chahine, R. Hasani, P. Kao, A. Ray, R. Shubert, M. Lechner, A. Amini, D. Rus, Robust flight navigation out of distribution with liquid neural networks, Science Robotics 8 (77) (2023) eadc8892.

[56] J.K. Lappalainen, F.D. Tschopp, S. Prakhya, M. McGill, A. Nern, K. Shinomiya, S.-y. Takemura, E. Gruntman, J.H. Macke, S.C. Turaga, Connectome-constrained deep mechanistic networks predict neural responses across the fly visual system at single-neuron resolution, 2023, bioRxiv 2023-2003.

[57] J. Liu, Y. Yuan, P. Zhao, G. Liu, H. Huo, Z. Li, T. Fang, Change of motifs in C. elegans reveals developmental principle of neural network, Biochem. Biophys. Res. Commun. 624 (2022) 112–119.

[58] J.M. Conner, K.M. Franks, A.K. Titterness, K. Russell, D.A. Merrill, B.R. Christie, T.J. Sejnowski, M.H. Tuszynski, NGF is essential for hippocampal plasticity and learning, J. Neurosci. 29 (35) (2009) 10883–10889.

[59] T. Fuchsberger, C. Clopath, P. Jarzebowski, Z. Brzosko, H. Wang, O. Paulsen, Postsynaptic burst reactivation of hippocampal neurons enables associative plasticity of temporally discontiguous inputs, eLife 11 (2022) e81071.

[60] A. Krizhevsky, G. Hinton, Learning Multiple Layers of Features from Tiny Images, Technical Report, University of Toronto, Toronto, Ontario, 2009.

[61] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[62] L. Deng, The mnist database of handwritten digit images for machine learning research, IEEE Signal Process. Mag. 29 (6) (2012) 141–142.

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

[64] J. Li, X. Wang, Z. Tu, M.R. Lyu, On the diversity of multi-head attention, Neurocomputing 454 (2021) 14–24.

[65] J.C.R. Whittington, J. Warren, T.E. Behrens, Relating transformers to models and neural representations of the hippocampal formation, in: International Conference on Learning Representations, 2022, URL: https://openreview.net/forum?id=B8DVo9B1YE0.

[66] M. Frey, C.F. Doeller, C. Barry, Probing neural representations of scene perception in a hippocampally dependent task using artificial neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 2113–2121.

[67] F.A. Heinsen, An algorithm for routing vectors in sequences, 2022, arXiv preprint arXiv:2211.11754.

[68] I.A. Kazerouni, G. Dooly, D. Toal, Ghost-UNet: An asymmetric encoder-decoder architecture for semantic segmentation from scratch, IEEE Access 9 (2021) 97457–97465.

[69] A. Makhzani, B.J. Frey, Pixelgan autoencoders, Adv. Neural Inf. Process. Syst. 30 (2017).

[70] H. Jaeger, The "Echo State" Approach to Analysing and Training Recurrent Neural Networks-With an Erratum Note, Technical Report 148, German National Research Center for Information Technology GMD, Bonn, Germany, 2001, p. 13.

[71] C. Gallicchio, A. Micheli, L. Pedrelli, Design of deep echo state networks, Neural Netw. 108 (2018) 33–47.

[72] D.V. Buonomano, W. Maass, State-dependent computations: spatiotemporal processing in cortical networks, Nat. Rev. Neurosci. 10 (2) (2009) 113–125.

[73] I. Farkaš, R. Bosák, P. Gergel', Computational analysis of memory capacity in echo state networks, Neural Netw. 83 (2016) 109–120.

[74] X. Glorot, Y. Bengio, 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.

[75] M.D. Zeiler, Adadelta: an adaptive learning rate method, 2012, arXiv preprint arXiv:1212.5701.

[76] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[77] R. Ma, J. Miao, L. Niu, P. Zhang, Transformed L1 regularization for learning sparse deep neural networks, Neural Netw. 119 (2019) 286–298.

[78] T.N. Kipf, M. Welling, Variational graph auto-encoders, 2016, arXiv preprint arXiv:1611.07308.

[79] L.R. Varshney, B.L. Chen, E. Paniagua, D.H. Hall, D.B. Chklovskii, Structural properties of the caenorhabditis elegans neuronal network, PLoS Comput. Biol. 7 (2) (2011) e1001066.

[80] B.L. Chen, D.H. Hall, D.B. Chklovskii, Wiring optimization can relate neuronal structure and function, Proc. Natl. Acad. Sci. 103 (12) (2006) 4723–4728.

[81] J.G. White, E. Southgate, J.N. Thomson, S. Brenner, et al., The structure of the nervous system of the nematode caenorhabditis elegans, Philos. Trans. R. Soc. Lond. Ser. B Biol. Sci. 314 (1165) (1986) 1–340.

[82] B. Szigeti, P. Gleeson, M. Vella, S. Khayrulin, A. Palyanov, J. Hokanson, M. Currie, M. Cantarelli, G. Idili, S. Larson, OpenWorm: an open-science approach to modeling caenorhabditis elegans, Front. Comput. Neurosci. 8 (2014) 137.

[83] E. Dai, S. Zhou, Z. Guo, S. Wang, Label-wise graph convolutional network for heterophilic graphs, in: Proceedings of the First Learning on Graphs Conference, PMLR 198, 2022, 26:1–26:21.

[84] D.J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: E.P. Xing, T. Jebara (Eds.), Proceedings of the 31st International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, Bejing, China, 2014, URL: http://proceedings.mlr.press/v32/rezende14.pdf.

[85] G. Li, M. Müller, B. Ghanem, V. Koltun, Training graph neural networks with 1000 layers, in: International Conference on Machine Learning, PMLR, 2021, pp. 6437–6449.

[86] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.

[87] J.M. Joyce, Kullback-leibler divergence, in: International Encyclopedia of Statistical Science, Springer, 2011.

[88] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Statistics 1050 (2014) 10.

[89] F. Bardozzo, M.D. Priscoli, T. Collins, A. Forgione, A. Hostettler, R. Tagliaferri, Cross X-AI: Explainable semantic segmentation of laparoscopic images in relation to depth estimation, in: 2022 International Joint Conference on Neural Networks, IJCNN, 2022, pp. 1–8, http://dx.doi.org/10.1109/IJCNN55064.2022.9892345.

[90] J. Park, A.-L. Barabási, Distribution of node characteristics in complex networks, Proc. Natl. Acad. Sci. 104 (46) (2007) 17916–17920.

[91] F. Bardozzo, P. Lió, R. Tagliaferri, A study on multi-omic oscillations in escherichia coli metabolic networks, BMC Bioinform. 19 (7) (2018) 139–154.

[92] F. Bardozzo, P. Lió, R. Tagliaferri, Signal metrics analysis of oscillatory patterns in bacterial multi-omic networks, Bioinformatics 37 (10) (2021) 1411–1419.

[93] S.E. Schaeffer, Graph clustering, Comput. Sci. Rev. 1 (1) (2007) 27–64.

[94] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Rev. Modern Phys. 74 (1) (2002) 47.

[95] A. Goldenberg, A.X. Zheng, S.E. Fienberg, E.M. Airoldi, A survey of statistical network models, Found. Trends Mach. Learn. (2010).

[96] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, N. Houlsby, 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.

[97] P. Foret, A. Kleiner, H. Mobahi, B. Neyshabur, Sharpness-aware minimization for efficiently improving generalization, 2020, arXiv preprint arXiv:2010.01412.

[98] J.T. Springenberg, Unsupervised and semi-supervised learning with categorical generative adversarial networks, 2015, arXiv preprint arXiv:1511.06390.

[99] T. Hinz, S. Wermter, Inferencing based on unsupervised learning of disentangled representations, 2018, arXiv preprint arXiv:1803.02627.

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

[101] M. Abbas, A. El-Zoghabi, A. Shoukry, DenMune: Density peak based clustering using mutual nearest neighbors, Pattern Recognit. 109 (2021) 107589.

[102] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, Adv. Neural Inf. Process. Syst. 29 (2016).

[103] H. Touvron, M. Cord, A. El-Nouby, J. Verbeek, H. Jégou, 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.

[104] C.-H. Tseng, H.-C. Liu, S.-J. Lee, X. Zeng, Perturbed gradients updating within unit space for deep learning, in: 2022 International Joint Conference on Neural Networks, IJCNN, IEEE, 2022, pp. 01–08.

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

[106] M. Chen, H. Peng, J. Fu, H. Ling, Autoformer: Searching transformers for visual recognition, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 12270–12280.

[107] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, Y. Wang, Transformer in transformer, Adv. Neural Inf. Process. Syst. 34 (2021) 15908–15919.

[108] T. Ridnik, G. Sharir, A. Ben-Cohen, E. Ben-Baruch, A. Noy, Ml-decoder: Scalable and versatile classification head, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2023, pp. 32–41.

[109] T. Ridnik, E. Ben-Baruch, A. Noy, L. Zelnik-Manor, Imagenet-21k pretraining for the masses, 2021, arXiv preprint arXiv:2104.10972.

[110] R. Dagli, Astroformer: More data might not be all you need for classification, 2023, arXiv preprint arXiv:2304.05350.

[111] T. Ridnik, H. Lawen, A. Noy, E. Ben Baruch, G. Sharir, I. Friedman, Tresnet: High performance gpu-dedicated architecture, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 1400–1409.

[112] S. Funahashi, Prefrontal cortex and working memory processes, Neuroscience 139 (1) (2006) 251–261.

[113] J.J. Todd, R. Marois, Capacity limit of visual short-term memory in human posterior parietal cortex, Nature 428 (6984) (2004) 751–754.

[114] C.-H. Chang, D. Nemrodov, A.C. Lee, A. Nestor, Memory and perception-based facial image reconstruction, Sci. Rep. 7 (1) (2017) 6499.

[115] C. Adami, J. Qian, M. Rupp, A. Hintze, Information content of colored motifs in complex networks, Artif. Life 17 (4) (2011) 375–390.

[116] D. Ron, Y. Singer, N. Tishby, The power of amnesia: Learning probabilistic automata with variable memory length, Mach. Learn. 25 (2) (1996) 117–149.

[117] Y. Kawai, J. Park, M. Asada, A small-world topology enhances the echo state property and signal propagation in reservoir computing, Neural Netw. 112 (2019) 15–23.

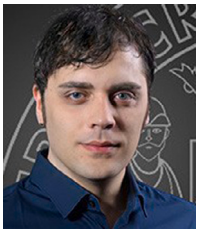
[118] D.S. Bassett, E.T. Bullmore, Small-world brain networks revisited, Neuroscientist 23 (5) (2017) 499–516.

[119] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.

[120] M. Winding, B.D. Pedigo, C.L. Barnes, H.G. Patsolic, Y. Park, T. Kazimiers, A. Fushiki, I.V. Andrade, A. Khandelwal, J. Valdes-Aleman, et al., The connectome of an insect brain, Science 379 (6636) (2023) eadd9330.

[121] J. Morra, M. Daley, Using connectome features to constrain echo state networks, in: 2023 International Joint Conference on Neural Networks, IJCNN, IEEE, 2023, pp. 1–8.

[122] J. Morra, M. Daley, A fully-connected neural network derived from an electron microscopy map of olfactory neurons in drosophila melanogaster for odor classification, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics, SMC, IEEE, 2020, pp. 4504–4509.



Francesco Bardozzo is an Assistant Professor at the University of Salerno, specializing in bio-inspired artificial and computational intelligence, with honor degrees in Computer Science and a Ph.D. in AI from UNISA. His work focuses on advanced deep learning, neuromorphic learning systems, cross explainable AI, and its applications in fields like computational finance, medical imaging, computational biology and generative music. He is a researcher at the Neurone Laboratory of the University of Salerno.

Andrea Terlizzi is a Computer Science M.Sc. student at the University of Salerno, now specializing in Data Science & Machine Learning, after receiving his bachelor's degree in computer science with honors. He has research experience in multiple deep learning fields including remote sensing multimodal models for captioning and zero-shot classification and graph neural networks for protein structural change prediction. Currently, he is conducting research at NeuroneLab (University of Salerno) on neuromorphic, biologically-inspired neural networks; his research interests also include reinforcement learning, graph neural networks and computational neuroscience.

Claudio Simoncini is an Italian Experimental Psychologist and Neuroscientist. His areas of expertise are neurophysiology of the visual system, oculomotor system and decision-making processes. His research focuses on understanding the mechanisms underlying visual perception, how the brain encodes and decodes visual information, and how that information is used in the control of behaviors. His investigation methods involve the use of behavioral experiments and eye movement recordings on humans and monkeys. He worked for several years in France at the CNRS and at the University of Chicago. His works have been published in Nature Neuroscience, Journal of Neuroscience and NeuroImage.

Pietro Liò is a Full Professor at the Department of Computer Science and Technology of the University of Cambridge, and he is a member of the Artificial Intelligence group. His research interest focuses on developing Artificial Intelligence and Computational Biology models to understand disease complexity and address personalized and precision medicine. The current focus is on Graph Neural Network modeling. He has a MA from Cambridge, a Ph.D. in Complex Systems and Non-Linear Dynamics (School of Informatics, dept of Engineering of the University of Firenze, Italy) and a Ph.D. in (Theoretical) Genetics (University of Pavia, Italy). He is a member of Cambridge Centre for AI in Medicine and of Ellis, the European Lab for Learning & Intelligent Systems and Academia Europaea.

Roberto Tagliaferri, a full Professor at the University of Salerno, directs the Ph.D. School in Big Data Management and Data Science, Accounting & Management. Specializing in Artificial Intelligence, Computational Intelligence, Neural Networks, Computer Architectures, and Bioinformatics, he contributes as an Associate Editor to IEEE Transactions on Cybernetics, BMC in Bioinformatics, and the Journal of Translational Medicine. His work focuses on modeling complex systems through Neural Networks and Computational Intelligence, including Deep Learning. Tagliaferri is a senior member of the IEEE "Computational Intelligence" and "System, Man and Cybernetics" societies and of INNS, with significant contributions to AI model design and learning paradigms.