Structure Matters: Deciphering Neural Network's Properties from its Structure

Abstract

Neural networks; both biological and artificial, are commonly represented as graphs with connections between neurons, yet there is little understanding of the relationship between their graph structure and computational properties. Neuroscientists are trying to answer this question in biological neural networks or connectomes; however, there is a big opportunity to explore this in the vast domain of artificial neural networks. We present StructureReps, an architecture-agnostic framework for encoding neural networks as graphs using graph representation learning. By capturing key structural properties, StructureReps reveals strong correlations between network structure and task performance across various architectures. Additionally, this framework has potential applications beyond the decoding of neural network properties.

1. Introduction

Neural networks are powerful computational models consisting of layers of interconnected neurons, where each connection, or edge, forms part of a directed graph that governs the flow of information. In neuroscience, significant effort is devoted to understanding the function of biological neural networks by reconstructing synaptic connectivity in the brain, revealing how structure shapes function Consortium et al. (2021); Lappalainen et al. (2024); Winding et al. (2023). But can we decipher computational properties from structure alone? We seek to answer this question using artificial neural networks as a digital twin. While it's widely recognized that the performance of artificial neural networks is closely tied to their structure LeCun et al. (1998); He et al. (2016); Vaswani (2017), a comprehensive understanding of the relationship between a network's accuracy and its underlying graph structure remains elusive.

Jiao et al. (2022) proposed using relational graphs to interpret neural networks, employing simple graph metrics like clustering coefficient and average path length to represent networks. However, these metrics are limited in capturing network complexity. A learned parametric representation of the graph structure provides a more nuanced understanding of network behavior.

To better capture the full graph structure of neural networks, we adopted a neural approach to graph encoding. This method embeds both geometric and relational information from the network's parameter space, similar to how graph representation learning has revealed key insights across domains such as social networks, molecular compounds, and biological systems Kipf et al. (2018); Rhee et al. (2017); Li et al. (2023).

In this paper, we introduce StructureReps, an architecture-agnostic graph representation learning framework for neural networks. To evaluate its effectiveness, we applied it to encode two different types of neural network and uncovered several key properties, especially insights into their performance on the tasks for which they were trained. Our results demonstrate that StructureReps provides a robust approach for analyzing and understanding network behavior independent of the underlying neural architecture.

2. Method

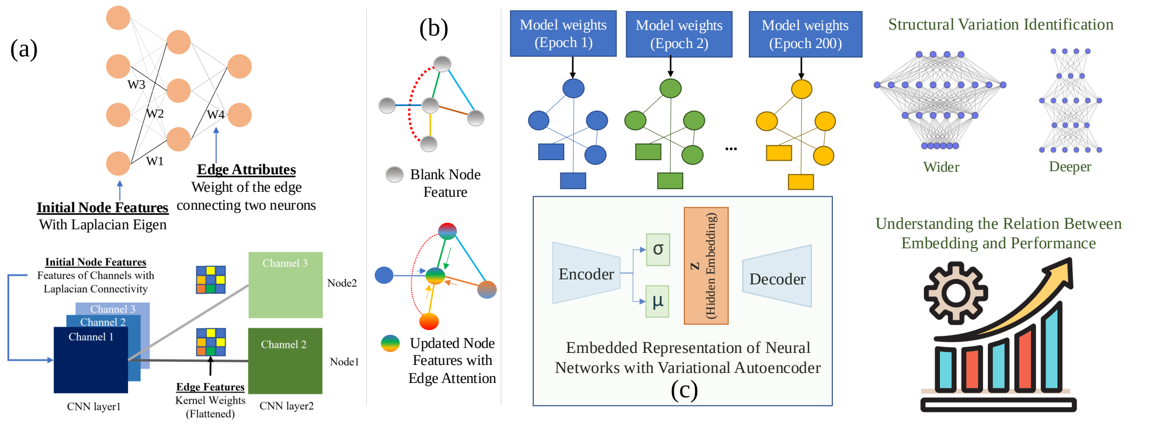

(a)
W3
W2
W1
W4
Edge Attributes
Weight of the edge connecting two neurons
Initial Node Features
With Laplacian Eigen
Initial Node Features
Features of Channels with Laplacian Connectivity
Channel 3
Channel 2
Channel 1
CNN layer1
Edge Features
Kernel Weights (Flattened)
Channel 3
Node2
Channel 2
Node1
CNN layer2
(b)
Blank Node Feature
Updated Node Features with Edge Attention
Model weights (Epoch 1)
Model weights (Epoch 2)
Model weights (Epoch 200)
...
Encoder
σ
μ
z (Hidden Embedding)
Decoder
Embedded Representation of Neural Networks with Variational Autoencoder
(c)
Structural Variation Identification
Wider
Deeper
Understanding the Relation Between Embedding and Performance

Figure 1: (a) Representation of Multi Layer Perceptron (MLP) and Convolutional Neural Network (CNN) as graphs. (b) Initial node features generated with Laplacian positional encoding are updated with an edge-attention layer. (c) With the generated graphs for each epoch, a VGAE is trained to embed the neural networks into a fixed dimension, allowing the decoding of performance and structural variations within the network from its structural information only.

In this section, we focus on three key aspects of our framework: representing different neural network architectures as a graph, extracting and updating node features within these graphs, and finally, discussing the unified representation of neural networks and how it reveals their properties.

Representing Neural Networks as a Graph: Given an n-layer **MLP**, where $(a_1, a_2, \dots, a_n)$ represents the number of neurons in each layer, the MLP is characterized by consecutive weight matrices with shapes $(a_2 \cdot a_1, a_3 \cdot a_2, \dots, a_n \cdot a_{n-1})$. The weight matrix $\mathcal{W}_{k,(k+1)} \in \mathbb{R}^{a_{k+1} \times a_k}$ defines the connectivity between neurons in layers k and $k+1$. Neurons are nodes in the graph, and weight matrices represent edges and their features. For a **CNN**, the channel of each layer is a node, and the filters that connect the channels are edges. A filter with a $k \times k$ kernel has a k^2-dimensional edge feature vector. Filters between channels $(c_1, c_2, \dots, c_n)$ across layers capture connectivity, where weight matrices $\mathcal{F}_{i,(i+1)} \in \mathbb{R}^{c_{i+1} \times c_i \times k \times k}$ define relationships between channels i and $i+1$. Filters (edges) describe transformations between channels (nodes) across consecutive layers.

Node Feature Construction: To encode the graph representations of neural networks, we generate meaningful node features using two methods.
1. Laplacian Eigenvector Positional Encoding: We follow the method of Dwivedi et al. (2023), using Laplacian eigenvectors as proposed by Belkin and Niyogi (2003) to address the lack of distinctive node features. This leverages eigenvectors derived from the Normalized Laplacian of the graph adjacency matrix (see Appendix A1.1) *2. Node Feature Generation with Edge Attention (EA) Layer:* We introduce **Edge Attention(EA)** layer, which creates node features $\mathbf{x}'_j$ for a node j by aggregating edge features $\varepsilon_{i,j}$ from all edges (i, j) connected to j during training. The contribution of each edge is weighted by the

Table 1: Pearson correlations between the training accuracies of different networks and the structural representations obtained through various combinations of encoding layers and features of choice. EA represents the Edge Attention layer, TC represents the Transformer Convolution layer (Shi et al. (2020)), and GC represents the GAT-Conv layer (Veličković et al. (2017)). LF and RF represent Laplacian and Random features. **StructureReps** refers to the custom VGAE (EA+TC+LF). Results are represented in $mean_{varaince}$ format. Figures in **bold** represent the best values.

Graph Encoders	Dense Neural Networks (MLP)			Vision Networks (CNN)	
	Two Moon	California Housing Price Prediction	Forest Covertype	CIFAR-10	ImageNet
EA+TC+LF	$\mathbf{0.74_{0.0020}}$	$\mathbf{0.39_{0.0271}}$	$\mathbf{0.79_{0.1201}}$	$\mathbf{0.94_{0.0001}}$	$\mathbf{0.91_{0.0002}}$
EA+GC+LF	$0.54_{0.0053}$	$0.26_{0.0100}$	$0.67_{0.0126}$	$0.84_{0.0003}$	$0.82_{0.0042}$
GC+LF	$0.10_{0.0583}$	$0.02_{0.0022}$	$0.04_{0.0045}$	$0.63_{0.0070}$	$0.56_{0.0053}$
GC+RF	$0.08_{0.0052}$	$-0.05_{0.0023}$	$0.02_{0.0031}$	$0.31_{0.0018}$	$0.16_{0.0073}$

attention coefficient $\alpha_{i,j}$, which is learned to emphasize the most important edges (See Appendix A1.2). For MLPs, attention is weighted by edge weights, and for CNNs, by kernel weights.

Unified Representation of Neural Networks and Decoding Properties: After defining node and edge features for different architectures, we use a variational graph autoencoder (Kipf and Welling, 2016) to learn the structural representation of the graphs in a self-supervised manner, as shown in Figure 1. This allows us to encode any trained neural network into a fixed-dimensional representation. We show the robustness of our method by decoding network performance on specific tasks solely from structural information. Additionally, we show that our unified representation can understand structural variations within networks of similar parameter space and capture their learning dynamics.

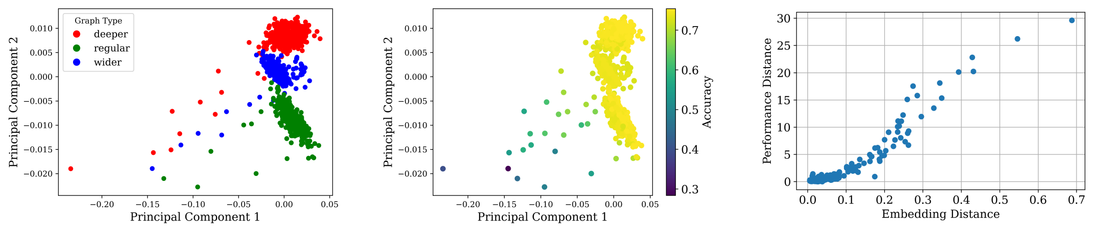
3. Results

In this section, we present a comprehensive overview of the experiments conducted and their corresponding results, providing a rigorous evaluation of StructureReps' effectiveness and viability.

Performance Decoding from Structure:

We perform MLP prediction on three datasets : Two moon classification as used by Liu et al. (2023), California housing price prediction (Pace and Barry (1997)), and Forest covertype prediction (Blackard (1998)) and CNN prediction on two image datasets (CIFAR-10, and a curated version of ImageNet (). For each tasks, we train the model for 200-250 epochs and record the performances. For each epoch, we get a new graph with updated weights according to Section 2. From the graphs, unified embedding representations are extracted using VGAE encoder. We use a linear regressor with 20% validation split on these embeddings to decode the performance of the neural network at any particular epoch. Specifically, we report the Pearson's correlation between the predicted and actual performance scores. Performances of StructureReps and other combinations on this score decoding task are shown in Table 2. We see that, for MLP tasks. We get a Pearson's Correlation of 0.74, 0.39, and 0.79 on the Two Moons, California housing price prediction, and the forest cover datasets with the MLP graph respectively. For vision tasks, we achieve a Pearson's correlation score of 0.94 on CIFAR-10 and 0.91 on ImageNet using StructureReps. These results indicate that with StructureReps, we can effectively decode the prediction performances of different types of neural networks from their structural information.

Identification of Structural Variations and Learning Dynamics: To verify if StructureReps can encode the structural variations of different neural networks while capturing the learning dynamics across each of these networks, we train three variants of MLP networks (regular, deeper, wider) with same parameter count on the Forest Covertype Prediction task. We see in Figure 2(a) that structural variations are clearly identified within the embedding space. Moreover, Figure 2(b) shows that within each structural cluster, performance dynamics (epoch) also shows a gradual shift from lower to higher accuracy(bottom left to top right). This indicates that graphs that have similar performances cluster closely in the embedding space. This is more clearly observed in Figure 2(c), where we show that their performances also get closer as the embedding distance between two neural network graphs decreases.



(a) (b) (c)

Figure 2: (a) Clustering networks based on structural properties, with colors representing different graph types. (b) Shows accuracy variation within clusters (color bar indicates accuracy). (c) Highlights the positive relationship between structural and performance similarity.

4. Conclusion and Discussion

In conclusion, we tested StructureReps on dense networks across three tasks and CNNs on two vision tasks, demonstrating that our framework achieved the strongest correlation between network structure and performance in all cases. Moreover, this correlation will enable to reliably identify the best-performing network from a set of candidates when compared to a base network, showcasing the potential of our approach in neural architecture search.

In future work, StructureReps can be extended to encode a broader range of neural network architectures, enabling a unified representation across diverse network types. Additionally, leveraging the latent representations produced by StructureReps opens the door to exploring important network properties such as human-AI alignment scores, uncertainty estimation, and the calibration state. Moreover, this framework can be applied to encode and decipher the properties of biological neural networks or connectomes, contributing to advancements in structural neurobiology. In parallel, the framework can be utilized to generate neural network “hashes” to ensure secure inference of a network. These future developments will enhance the framework’s ability to provide more comprehensive insights, ultimately improving the reliability and safety of neural networks in real-world applications.

References

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

MICrONS Consortium, J Alexander Bae, Mahaly Baptiste, Caitlyn A Bishop, Agnes L Bodor, Derrick Brittain, JoAnn Buchanan, Daniel J Bumbarger, Manuel A Castro, Brendan Celii, et al. Functional connectomics spanning multiple areas of mouse visual cortex. *BioRxiv*, pages 2021–07, 2021.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Licheng Jiao, Jie Chen, Fang Liu, Shuyuan Yang, Chao You, Xu Liu, Lingling Li, and Biao Hou. Graph representation learning meets computer vision: A survey. *IEEE Transactions on Artificial Intelligence*, 4(1):2–22, 2022.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. PMLR, 2018.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Janne K Lappalainen, Fabian D Tschopp, Sridhama Prakhya, Mason McGill, Aljoscha Nern, Kazunori Shinomiya, Shin-ya Takemura, Eyal Gruntman, Jakob H Macke, and Srinivas C Turaga. Connectome-constrained networks predict neural activity across the fly visual system. *Nature*, pages 1–9, 2024.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Xiao Li, Li Sun, Mengjie Ling, and Yan Peng. A survey of graph neural network based recommendation in social networks. *Neurocomputing*, 549:126441, 2023.

Ziming Liu, Eric Gan, and Max Tegmark. Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *Entropy*, 26(1):41, 2023.

R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

Sungmin Rhee, Seokjun Seo, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *arXiv preprint arXiv:1711.05859*, 2017.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Michael Winding, Benjamin D Pedigo, Christopher L Barnes, Heather G Patsolic, Youngser Park, Tom Kazimiers, Akira Fushiki, Ingrid V Andrade, Avinash Khandelwal, Javier Valdes-Aleman, et al. The connectome of an insect brain. *Science*, 379(6636):eadd9330, 2023.

Appendix

A1. Details of Method

A1.1. Algorithm for Generating Laplacian Eigenvector Positional Encoding

Algorithm 1 Laplacian Eigenvector Positional Encoding

```
Input: Graph $G = (V, E)$, number of eigenvectors $k$, sparse threshold $T$
Output: Positional encodings $\mathbf{PE}$ of graph $G$
1. Compute the Laplacian:
 $\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathcal{M})\mathbf{D}^{-1/2}$ ;    // Symmetric normalized Laplacian matrix
2. Eigenvector Computation:
 if $N < T$ then
   Compute all eigenvectors $\mathbf{V}$ and eigenvalues $\lambda$ such that $\mathbf{L}_{\text{sym}}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
else
   Compute the smallest $k + 1$ eigenvectors using iterative methods
end
3. Eigenvector Selection:
 Sort $\mathbf{V}$ by eigenvalues and select $k$ smallest non-trivial eigenvectors $\mathbf{V}_{\text{selected}} = [\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_{k+1}]$
4. Random Sign Flipping:
 $\mathbf{PE} = \mathbf{V}_{\text{selected}} \odot \mathbf{S}$ ;    // Random sign vector $\mathbf{S} \in \{-1, 1\}^k$
return PE
```

A1.2. Edge Attention Layer

To create the node feature $\mathbf{x}'_j$, we apply the update function in Equation (1), where $\mathcal{N}(j)$ represents set of all node i which has an edge (i, j)

$$\mathbf{x}'_j = \sum_{i \in \mathcal{N}(j)} \alpha_{i,j} \cdot \varepsilon_{i,j} \tag{1}$$

Here, $\alpha_{i,j}$ denotes the attention co-efficient which is calculated as in Equation (2)

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}_{\varepsilon}^{\top} \varepsilon_{i,j}^{\text{proj}}\right)\right)}{\sum_{k \in \mathcal{N}(j)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}_{\varepsilon}^{\top} \varepsilon_{k,j}^{\text{proj}}\right)\right)} \tag{2}$$

We calculate $\varepsilon_{k,j}^{\text{proj}}$ following the transformations as in Equation (3) where $\mathbf{n}(\cdot)$ represents a basic noise function with no learnable parameters.

$$
\begin{aligned}
\varepsilon_{i,j}^{\text{transformed}} &= \text{LeakyReLU}(\boldsymbol{\Theta}_{\varepsilon}\varepsilon_{i,j} + \mathbf{b}_{\varepsilon}) \\
\varepsilon_{i,j}^{\text{noised}} &= \mathbf{n}(\varepsilon_{i,j}) \\
\varepsilon_{i,j}^{\text{proj}} &= \boldsymbol{\Theta}_{\text{lin}}\left(\text{Concat}\left(\varepsilon_{i,j}^{\text{transformed}}, \varepsilon_{i,j}, \varepsilon_{i,j}^{\text{noised}}\right)\right)
\end{aligned} \tag{3}
$$

A1.3. Why do we need other graph neural network layers?

One important question that might arise is why Edge Attention Layer should not be used throughout the encoder. It is important to note that while Edge Attention has learnable parameters, it is ultimately a feature generation layer and not a message-passing layer. As such, it is not meant to process node features but to generate them and should not be used beyond the first layer of any encoder. We use other message-passing layers like GATConv and Transformer Conv to process the features generated by Edge Attention.

A2. Additional Results

Table 2: Pearson correlations between the test accuracies of different networks and the structural representations obtained through various combinations of encoding layers and features of choice.

Graph Encoders	Dense Neural Networks (MLP)			Vision Networks (CNN)	
	Two Moon	California Housing Price Prediction	Forest Covertype	CIFAR-10	ImageNet
EA+TC+LF	$\mathbf{0.71_{0.00231}}$	$\mathbf{0.42_{0.0261}}$	$\mathbf{0.86_{0.0595}}$	$\mathbf{0.81_{0.004}}$	$\mathbf{0.83_{0.0022}}$
EA+GC+LF	$0.52_{0.0062}$	$0.29_{0.0080}$	$0.68_{0.0122}$	$0.73_{0.0014}$	$0.74_{0.0030}$
GC+LF	$0.09_{0.0560}$	$0.06_{0.0027}$	$0.06_{0.0042}$	$0.55_{0.0020}$	$0.42_{0.0012}$
GC+RF	$0.09_{0.0049}$	$-0.06_{0.0023}$	$0.03_{0.0036}$	$0.18_{0.0002}$	$0.11_{0.0082}$