Self-Supervised Graph Representation Learning for Neuronal Morphologies

Anonymous authors
Paper under double-blind review

Abstract

Unsupervised graph representation learning has recently gained interest in several application domains such as neuroscience, where modeling the diverse morphology of cell types in the brain is one of the key challenges. It is currently unknown how many excitatory cortical cell types exist and what their defining morphological features are. Here we present GraphDINO, a purely data-driven approach to learn low-dimensional representations of 3D neuronal morphologies from unlabeled large-scale datasets. GraphDINO is a novel transformer-based representation learning method for spatially-embedded graphs. To enable self-supervised learning on transformers, we (1) developed data augmentation strategies for spatially-embedded graphs, (2) adapted the positional encoding and (3) introduced a novel attention mechanism, AC-Attention, which combines attention-based global interaction between nodes and classic graph convolutional processing. We show, in two different species and across multiple brain areas, that this method yields morphological cell type clusterings that are on par with manual feature-based classification by experts, but without using prior knowledge about the structural features of neurons. Moreover, it outperforms previous approaches on quantitative benchmarks predicting expert labels. Our method could potentially enable data-driven discovery of novel morphological features and cell types in large-scale datasets. It is applicable beyond neuroscience in settings where samples in a dataset are graphs and graph-level embeddings are desired.

1 Introduction

The brain is structured into different areas that contain diverse types of neurons (Ascoli et al., 2008). The morphology of cortical neurons is highly complex with widely varying shapes that are tightly linked to their function. Cell morphology has long been used to classify neurons into cell types (Ramón y Cajal, 1911), but characterizing neuronal morphologies is still a challenging open research question. Morphological analysis has traditionally been carried out by visual inspection (Ascoli et al., 2008; Defelipe et al., 2013) or by computing a set of predefined, quantitatively measurable features such as number of branching points (Uylings & Van Pelt, 2002; Scorcioni et al., 2008; Oberlaender et al., 2012; Polavaram et al., 2014; Markram et al., 2015; Lu et al., 2015; Gouwens et al., 2019). However, both approaches have deficits: expert assessments have a high variance(Defelipe et al., 2013) and the manual definition of morphological features introduces biases (Wang, 2018), thus calling for more unbiased, data-driven approaches to characterize the morphology of neurons.

Recent advances in recording technologies have greatly accelerated data collection and therefore the amount of data available. These developments have opened the floor for data-driven approaches based on unsupervised machine learning methods. One form of data representation that is particularly suitable for neurons is representing the skeleton of a neuron as a tree. In such a tree, the root node represents the neuron's cell body and the node features are their 3D locations. The availability of a number of such skeleton datasets has recently sparked some work on graph-level representation learning (Laturnus & Berens, 2021; Sun et al., 2020; You et al., 2020). Following this line of research, we here present a fully automatic and scalable graph-level representation learning approach.

Our contributions in this paper are fourfold:

1. We propose a new self-supervised model to learn graph-level embeddings for spatial graphs. Unlike previous methods, our approach does not require human annotation or manual feature definition.
2. We introduce a novel attention module that combines transformer-style attention and message passing between neighboring nodes as in graph neural networks.
3. We apply this approach to the classification of excitatory neuronal morphologies and show that it produces clusters that are comparable with known excitatory cell types obtained by manual feature-based classification and expert-labeling.
4. We outperform existing approaches based on manual feature engineering and auto-encoding in predicting expert labels.

Source code will be publicly available.

2 Related Work

2.1 Representation learning for neuronal morphologies

Morphology has been used for a long time to classify neurons by either letting experts visually inspect the cells (Ramón y Cajal, 1911; Defelipe et al., 2013) or by specifying expert-defined features that can be extracted and used as input to a classifier (Oberlaender et al., 2012; Markram et al., 2015; Kanari et al., 2017; Wang, 2018; Kanari et al., 2019; Gouwens et al., 2019) (see Armañanzas & Ascoli (2015) for review). Ascoli et al. (2008) made an effort to unify the used expert-defined features.

With the advent of new technologies for microscopic imaging, electrical recording, and molecular analysis such as Patch-seq (Cadwell et al., 2015) that allow the simultaneous recording of transcriptomy, electrophysiology and morphology of whole cells, several works have explored the prediction of cell types from multiple modalities (Gala et al., 2021) or one modality from the other (Cadwell et al., 2015; Scala et al., 2021; Gouwens et al., 2020).

Several works try to learn a representation of neuronal morphologies. Laturnus & Berens (2021) propose a generative approach involving random walks in graphs to model neuronal morphologies. Schubert et al. (2019) process 2D projections of neuronal morphologies with a convolutional neural network (CNN) to learn low-dimensional representations. Seshamani et al. (2020) extract local mesh features around spines and combine them with traditional Sholl analysis (Sholl, 1953). Gouwens et al. (2019) define a set of morphological features based on graphs and perform hierarchical clustering on them. We use the latter as a baseline for a classical approach with pre-defined features and Laturnus & Berens (2021) as a baseline of a model with learned features.

Concurrently to this work, Zhao et al. (2022) proposed a contrastive graph neural network to learn neuronal embeddings with a focus on retrieval efficiency from large-scale databases. (Elabbady et al., 2022) developed a method to learn representations of neurons based on the subcellular features of the somatic region of the neurons and show that those features are sufficient for classifying cell types on large-scale EM datasets. Chen et al. (2022b) proposed a combination of graph-based processing and manually-defined features to learn embeddings of neuronal morphologies using a LSTM-based network and contrastive learning. We compare to the latter in Section 5.8.

2.2 Graph Neural Networks (GNNs)

Graph neural networks learn node representations by recursively aggregating information from adjacent nodes as defined by the graph's structure. While early approaches date back over a decade Scarselli et al. (2009), recently numerous new variants were introduced for (semi-) supervised settings: relying on convolution over nodes Duvenaud et al. (2015); Hamilton et al. (2017); Kipf & Welling (2017), using recurrence Li et al. (2016), or making use of attention mechanisms Veličković et al. (2018). A representation for the whole graph

is often derived by a readout operation on the node representations, for instance averaging. See Dwivedi et al. (2020) for a recent benchmark on graph neural network architectures.

Transformer-based GNNs. Similar to us, Zhang et al. (2020) and Dwivedi & Bresson (2021) use transformer attention to work with graphs. However, Zhang et al. (2020) compute transformer attention over the nodes of sampled subgraphs, while Dwivedi & Bresson (2021) compute the attention only over local neighbors of nodes, which boils down to a weighted message passing that is conditioned on node feature similarity, and trains supervised. Unlike these previous approaches, we compute the attention between nodes of the global graph and adapt the transformer attention to consider the adjacency matrix of the graph, which allows the model to take into account both the direct neighbors of a node as well as all other nodes in the graph. Concurrently, Ying et al. (2021) and Mialon et al. (2021) propose strategies to adapt positional encodings to graphs to leverage the structural information of the graphs with transformer attention. Rampášek et al. (2022) implements a two-stream architecture, in which transformer attention and message passing are computed in parallel and then combined after each block. Chen et al. (2022a) incorporates structural information into the transformer attention by extracting a subgraph representation around each node before computing the attention.

Self-supervised learning on graphs. Self-supervised learning has proven to be a useful technique for training image feature extractors Oord et al. (2018); Chen et al. (2020); Chen & He (2021); Caron et al. (2021) and has been investigated for learning graph (Li et al., 2016; Hassani & Khasahmadi, 2020; Qiu et al., 2020; You et al., 2020; Xu et al., 2021) and node Veličković et al. (2019) representations. Narayanan et al. (2017) learn graph representations through skip-gram with negative sampling by predicting present subgraphs. You et al. (2020) propose four data augmentations for contrastive learning of graph-level embeddings. Sun et al. (2020) learn graph-level representations in a contrastive way, by predicting if a subgraph and a graph representation originate from the same graph. Similarly, Hassani & Khasahmadi (2020) put node features of one view in contrast with the graph encoding of a second view and vice versa. They build on graph diffusion networks Klicpera et al. (2019) and only augment the structure of the graph but not the initial node features. We use Sun et al. (2020) and You et al. (2020) as a baseline for graph-level unsupervised representation learning. Qiu et al. (2020) propose a generic pre-training method which uses an InfoNCE objective Oord et al. (2018) to learn features by telling augmented versions of one subgraph from other subgraphs with random walks as augmentations. Xu et al. (2021) aim to capture local and global structures for whole-graph representation learning. They rely on an EM-like algorithm to jointly train the assignment of graphs to hierarchical prototypes, the GNN parameters and the prototypes. Zhu et al. (2021) propose adaptive augmentation, which considers node centrality and importance to generate graph views in a contrastive framework. Unlike most prior work, we contrast two global views of a graph in order to learn a whole-graph representation. Our method operates on spatially embedded graphs, in which nodes correspond to points in 3D space. We make use of this knowledge in the choice of augmentations.

3 GraphDINO

We propose GraphDINO, a method for self-supervised representation learning of graphs. It is inspired by recent progress in self-supervised representation learning of images that has been shown to be competitive to supervised learning without relying on labels. The core idea is to enforce that the representations of two augmented versions of the same image are close to each other in latent space.

DINO (Caron et al., 2021) is an implementation of this self-supervised learning framework consisting of two encoders with a transformer backbone. To avoid mode collapse, only one encoder is directly trained through backpropagation (student) while the weights of the other encoder (teacher) are an exponential moving average (ema) of the student's weights. The latent representations given by the encoders are mapped to probability distributions by a multi-layer perceptron (MLP) and subsequent softmax operator over which the cross-entropy loss is computed (Fig. C.4). For further explanation of DINO see Appendix C.1.

GraphDINO adapts this self-supervised framework to the data domain of graphs (Fig. 1). In order to use information given by the connectivity of the graph, we modify the computation of the transformer attention to take the graph adjacency matrix into account and use the graph Laplacian as positional encoding.

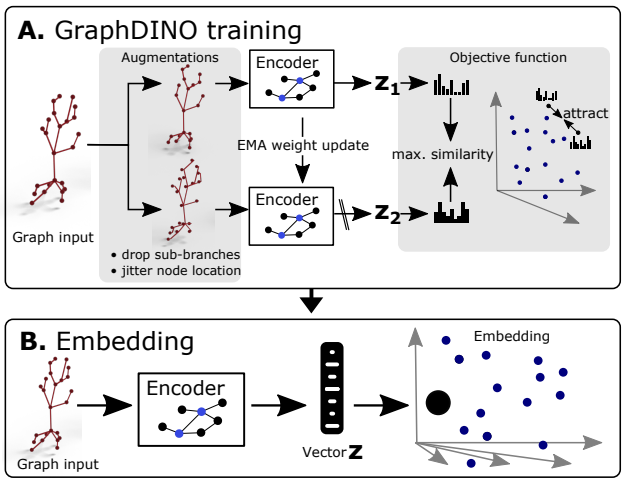


Figure 1: **A.** Self-supervised learning of low dimensional vector embeddings z_1, z_2 that capture the essence of the 3D morphology of individual neurons using GraphDINO. Two augmented "views" of the neuron are input into the network, where the weights of one encoder (bottom) are an exponential moving average (EMA) of the other encoder (top). The objective is to maximize the similarity between the vector embeddings of both views. Vector embeddings of similar neurons are close to each other in latent space. **B.** An individual neuron is represented by its vector embedding as a point in the 32-dimensional vector space.

More specifically, we introduce the following modifications: (1) we incorporate the graph's adjacency matrix into the attention computation; (2) we use the graph Laplacian as positional encoding; (3) we define augmentations suitable for spatial graphs.

Input. Input to the network is the 3D shape of a neuron which is represented as an undirected graph $G = (V, E)$. V is the set of nodes $\{v_i\}_{i=1}^N$ and E the set of undirected edges $E = \{e_{ij} = (v_i, v_j)\}$ that connect two nodes v_i, v_j. The features of each node v_i in the graph are encoded into a token using a linear transformation. These tokens are then used as input to the transformer model, which consists of l multi-head attention modules with h heads each.

Attention bias. Key-value query attention became popular in natural language modelling (Vaswani et al., 2017) and is now used routinely also in image models (Dosovitskiy et al., 2020).

To make use of the information given by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ of the input graph — i.e. the neighborhood of nodes —, we bias the attention towards $\mathbf{A}$ by adding a learned bias to the attention matrix that is conditioned on the input token values:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \sigma\left(\lambda \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \gamma \mathbf{A}\right)\mathbf{V}, \quad \text{with } [\lambda_i, \gamma_i] = \exp(\mathbf{W}x_i), \tag{1}$$

where $\mathbf{K}$, $\mathbf{Q}$, $\mathbf{V}$ are the keys, queries and values which are computed as a learned linear projection of the tokens. $\sigma(\cdot)$ denotes the softmax function. $x_i \in \mathbb{R}^D$ is the token of node v_i, $\mathbf{W} \in \mathbb{R}^{2 \times D}$ is a learned weight matrix, $\lambda, \gamma \in \mathbb{R}^N$ are two factors per node that trade off how much weight is assigned to neighboring nodes versus all other nodes in the graph, and N is the number of nodes.

When $\gamma = 0$ and $\lambda = 1$, the adjacency-conditioned attention (AC-Attention) reduces to regular transformer attention. In the other extreme case ($\lambda = 0$,), the attention matrix is dominated by $\mathbf{A}$ and the transformer attention computation is akin to the message passing algorithm that is commonly used when working with graphs (Scarselli et al., 2009; Duvenaud et al., 2015; Gilmer et al., 2017). But GraphDINO is more flexible than both regular message passing and point-cloud attention since it can decide how much weight is given to the neighbors of a node while maintaining the flexibility to attend to all other nodes in the graph as well.

Positional encoding. Following Dwivedi et al. (2020), we use the normalized graph Laplacian matrix $\mathbf{L}$ as positional encoding, which is computed by $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{U}^T\mathbf{\Lambda}\mathbf{U}$, where $\mathbf{I}$ is the identity matrix, $\mathbf{D}$ the $N \times N$ degree matrix, $\mathbf{A}$ the adjacency matrix, and $\mathbf{U}$ and $\mathbf{\Lambda}$ are the matrices of eigenvectors and eigenvalues, respectively. The positional encodings are the first 32 eigenvectors with largest eigenvalues. Positional encodings are added to the nodes features after tokenization.

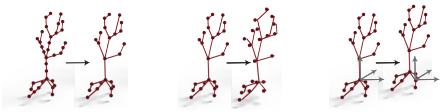
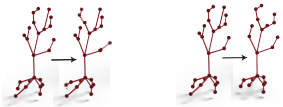
Data augmentation. Data augmentation plays an important role in self-supervised learning and needs to be adapted to the data, since it expresses which invariances should be imposed. Given the spatial neuronal data, we apply the following augmentations: **(1) Subsampling**: We subsample the original graph to a fixed number of n nodes by randomly removing nodes that are not branching points (i.e. nodes connected

to more than two other nodes), and connecting the two neighbors of the removed node. This facilitates batch processing. Furthermore, this augmentation retains the global structure of the neuron, while altering local structure in the two views. **(2) Rotation**: we perform random 3D rotation around the y-axis, that is orthogonal to the pia. **(3) Jittering**: we randomly translate individual node positions by adding Gaussian noise with $\mathcal{N}(0, \sigma_1)$. **(4) Subgraph deletion**: We identify branches that connect leaf nodes to the last upstream parent node in the graph, i.e. terminal branches that do not split into further branches, and randomly delete n of them starting at a random location along the branch, while maintaining the overall graph structure. **(5) Graph position**: we randomly translate the graph as a whole by adding Gaussian noise with $\mathcal{N}(0, \sigma_2)$ to all nodes. Unlike Caron et al. (2021), we do not differentiate between the augmentations seen by the student and the teacher network.

4 Data and Experiments

4.1 Synthetic graphs

Table 1: Overview of data augmentations for spatially-embedded graphs such as neuronal skeletons.

Level	Augmentations
Graph	(1) Subsampling, (2) Rotation, (5) Translation
	
Subgraph	(3) Jittering, (4) Branch deletion
	

To demonstrate that our novel attention mechanism is strictly more powerful than simple all-to-all attention on a graph, we generate a synthetic graph dataset. In this dataset, the five classes share similar node locations but differ in how the nodes are connected. See Appendix A for the detailed generation process. We use this dataset to test the efficacy of our novel attention mechanism, AC-Attention, and the positional encoding.

4.2 Neuronal and tree graphs

We apply GraphDINO to five publicly available neuronal datasets and one non-neuronal dataset.

Blue Brain Project (BBP): Rat somatosensory cortex. Available from the Neocortical Microcircuit Collaboration Portal of the Blue Brain Project[1] (Ramaswamy et al., 2015), the dataset contains 1,389 neurons from juvenile rat somatosensory cortex. We train GraphDINO unsupervised on the 3D dendritic morphologies of all neurons. For evaluation, we use the subset of 616 neurons which have been labeled by experts into cell types and cortical layer. Of these 616 neurons 286 are excitatory that have been assigned to 14 cell types (Markram et al., 2015). See Appendix C.5.1 for more details on the dataset. We use this dataset to evaluate the capability of GraphDINO to learn useful representations of neuronal morphologies that align with know cell types, perform ablation experiments on the novel graph augmentation strategies and compare to previous work using manually-defined features.

M1 PatchSeq: Mouse motor cortex. The M1 PatchSeq dataset contains 275 excitatory and 371 inhibitory cells from M1 in adult mouse primary motor cortex (Scala et al., 2021).[2] The excitatory cells (M1 EXC) have been classified into tufted, untufted and other neurons based on their morphology in a previous study (Laturnus & Berens, 2021). We use this dataset to compare to previous work that learns morphological embeddings in a data-driven way. We train GraphDINO unsupervised on the 3D dendritic morphologies of the 646 neurons. For evaluation, we follow the evaluation protocol and use the same dataset split as Laturnus & Berens (2021). We additionally report the 5-nearest neighbour accuracy of three additional data to estimate the variance due to the chosen split, since the test set is very small (60 neurons) and the balanced

[1] http://microcircuits.epfl.ch/#/main
[2] https://download.brainimagelibrary.org/3a/88/3a88a7687ab66069/

accuracy is strongly influenced by the morphologically heterogeneous "other" class that is only represented by six samples in the test set (Laturnus & Berens, 2021).

Allen Brain Atlas (ACT): Mouse visual cortex. As part of the Allen Cell Types Database, the dataset contains 510 neurons from the mouse visual cortex with a broad coverage of types, layers and transgenic lines.[3] See Allen Institute (2016) for details on how the dataset was recorded. It comes with a classification of each neuron into spiny, aspiny, or sparsely spiny, where spiny are assumed to be excitatory neurons and all else are inhibitory (Gouwens et al., 2019). Additionally, the cortical layer of each neuron is provided.

Brain Image Library (BIL): Whole mouse brain. The Brain Image Library contains 1,741 reconstructed neurons from cortex, claustrum, thalamus, striatum and other brain areas in mice (Peng H, 2021).[4]

Janelia MouseLight (JML): Whole mouse brain. The Janelia MouseLight platform contains 1,200 projection neurons from the motor cortex, thalamus, subiculum, and hypothalamus (Winnubst J, 2019).[5]

Joint training on ACT, BIL and JML. Following Chen et al. (2022b), for joint training of the ACT, BIL and JML datasets, we rotate the neurons such that the first principle component is aligned with the y-axis.Chen et al. (2022b) group the neurons of the three datasets ACT, BIL and JML into eleven classes based on the cortical layer or brain region they originate from. They then evaluate their learned embeddings on a subset of six (for BIL) or four classes (for ACT and JML) that have a broad coverage across the datasets. See Appendix C.5.4 for further details.

Botanical Trees. The Trees dataset (Seidel et al., 2021) is a highly diverse dataset comprised of 391 skeletons of trees stemming from 39 different genuses and 152 species or breedings. The skeletons were extracted from LIDAR scans of the trees. Nodes of the skeletons have a 3D coordinate associated with them. We normalize the data such that the lowest point (start of the tree trunk) is normalized to (0, 0, 0).

4.2.1 Data Preprocessing

Since the objective of GraphDINO is to learn purely from the 3D dendritic morphology of neurons, we normalize each graph such that the soma location is centered at (0, 0, 0) (no cortical depth information is given to the model). Furthermore, axons are removed for all experiments in the paper, because the reconstruction of axonal arbors of excitatory neurons from light microscopy images is difficult due to their small thickness and long ranges that they cover (Kanari et al., 2019) and thus often unreliable. The input nodes V have features $v_i = [x, y, z]$ where $v_i \in \mathbb{R}^3$ are the spatial xyz-coordinates in micrometers [µm].

4.2.2 Training details.

GraphDINO is implemented in PyTorch Paszke et al. (2019) and trained with the Adam optimizer Kingma & Ba (2015). The latent dimensionality of z is 16 for the synthetic graphs and 32 for the neuronal datasets. See C.2 for an overview of the hyperparameters used for training on the different datasets. At inference time, the latent embeddings z are extracted from the student network for the unaugmented graphs. We use scipy for fitting Gaussian Mixture models (GMM) and k-nearest neighbor classifiers (kNN) Pedregosa et al. (2011).

5 Results

We first establish that GraphDINO works on the synthetic graph dataset and show that our novel AC-Attention is necessary for exploiting information from graph connectivity. Second, we show that our novel augmentation strategies are suitable for spatially-embedded graphs that are tree-structured and that classical GNN message-passing is not sufficient when graphs have long-ranging branches. Then, we move

[3] http://celltypes.brain-map.org/
[4] https://download.brainimagelibrary.org/biccn/zeng/luo/fMOST/
[5] http://mouselight.janelia.org/

to the gradually more complex, biological questions of spiny-aspiny differentiation, cell type recovery and consistency with existing labels. To this end, we employ in total five neuronal datasets that encompass two species and range across multiple brain areas. Finally, we compare our model to several previous works based on manually-defined morphological features as well as approaches with learned features. See Appendix B for the application of GraphDINO to a non-neuronal dataset.

5.1 AC-Attention recovers information encoded by graph connectivity

We start by demonstrating the efficacy of our novel AC-Attention module. For this experiment, we use the synthetic graph dataset where classes differ in how nodes are connected whereas the distribution of node positions does not vary across classes. Therefore, considering the graph structure is necessary to differentiate between the classes (more details in Appendix A). We train GraphDINO on the synthetic graph dataset without labels in three configurations: (1) with AC-Attention, (2) with regular transformer attention and (3) with transformer attention and without positional encoding. We asses the quality of the learned embedding using the ground truth labels. A linear classifier on the learned embeddings achieves a test set accuracy of (1) 95% ± 4, (2) 29% ± 6, and (3) 20% ± 1, showing that AC-Attention allows us to capture the structure of spatially embedded graphs when the location of the nodes alone does not provide sufficient information. Removing both AC-Attention and the positional encoding results in the classifier performing at chance level. Using only the positional encoding performs slightly better than chance, because the positional encodings contain some information about node connections through the graph Laplacian. To make full use of the information given by the connectivity of the graphs, using AC-Attention is essential (Fig. 2).

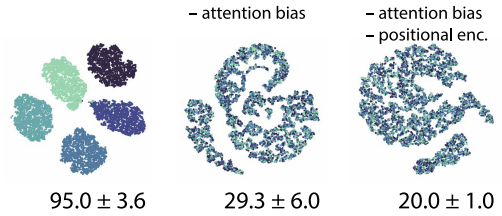


Figure 2: t-SNE embedding of latent representations of 3D synthetic graphs shown for one example run per model. Accuracy averaged over five random seeds and given as mean ± standard deviation. "–" means removing one component from the full model.

5.2 Tailored graph augmentations are well-suited for spatially-embedded graphs

Table 2: Ablation results on the BBP dataset. Cell-type classification accuracy [%] of our model and ablations averaged over three random seeds and given as mean ± standard deviation. "–" indicate removal of an augmentation or model component.

Model	Accuracy
GraphDINO	**65.8** ± 1
– 3D rot.	55.4 ± 1
– node jitter	64.8 ± 2
– graph translation	55.6 ± 2
– drop branch	64.6 ± 1
subsampling: 50 nodes	60.0 ± 0
subsampling: 200 nodes	62.0 ± 3
– adjacency ($\gamma = 0$)	62.8 ± 1
– attention ($\lambda = 0$)	59.8 ± 2

In self-supervised learning, data augmentation is used to obtain two views that define a positive input pair. The augmentations here are chosen to encode invariances that should not change the underlying sample identity. In previous contrastive learning for graphs, these augmentations were for example dropping random edges or masking node features (You et al., 2020). These augmentations are not appropriate for our spatially-embedded graphs that form a tree and whose only node features are their 3D location in space. Thus we designed five novel augmentation techniques specifically for spatially embedded graphs such as neural morphologies or botantical trees: subsampling, rotation, node jittering, branch deletion and graph translation (see Section 3).

To test the importance of the individual graph augmentations we perform a set of ablation experiments using the BBP dataset. We remove one augmentation from our model at a time and evaluate the leave-one-out 5-nearest neighbor accuracy when predicting the expert labels. For the subsampling augmentation we vary the number of retained nodes. Our full model achieves an average accuracy of 65.8% when classifying the excitatory cells into the 12 expert labels (Appendix C.5.1). When removing individual data augmentations the accuracy decreases (Tab. 3). Especially 3D rotation and graph translation are important augmentation strategies whose removal lead to substantial performance deterioration.

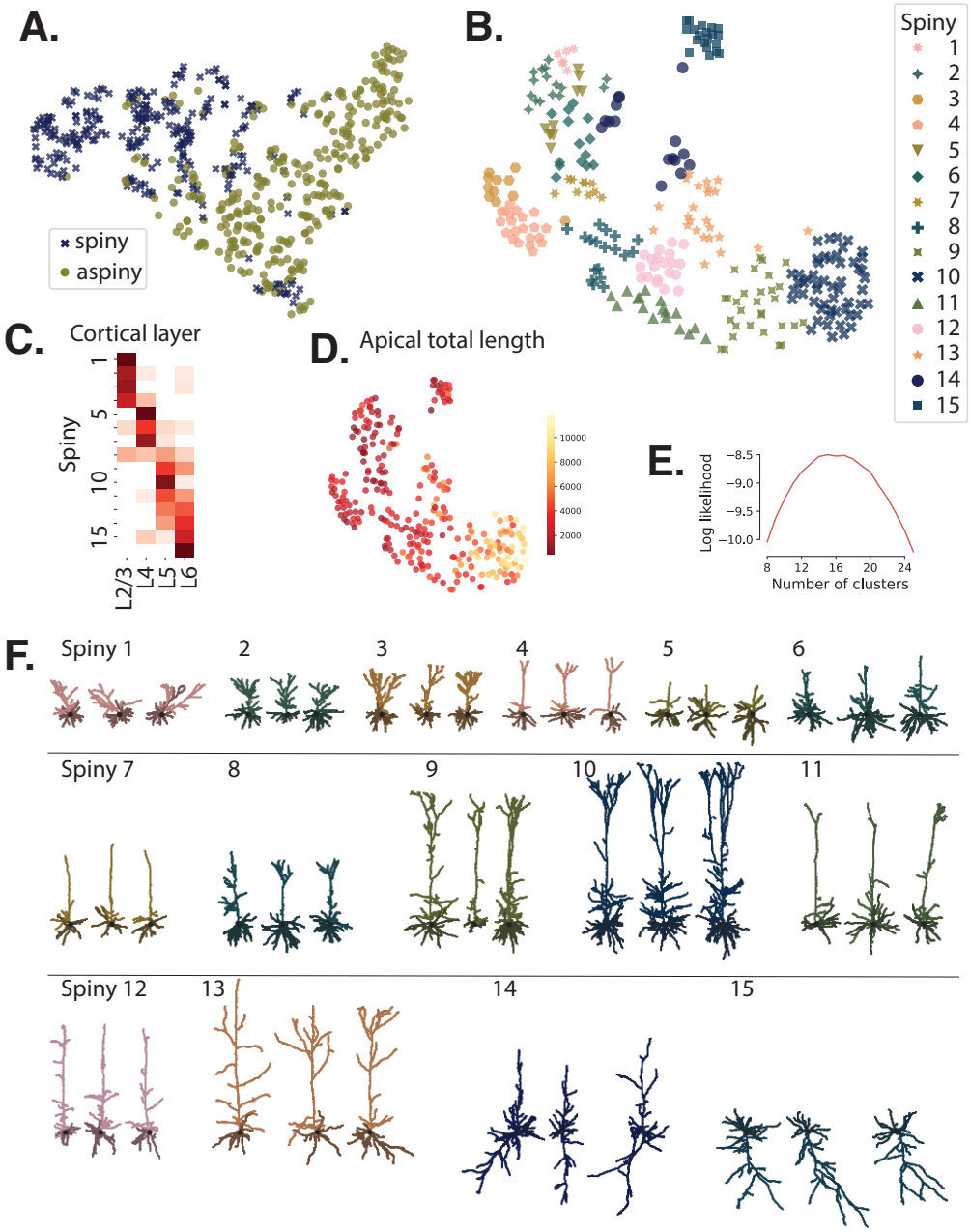


Figure 3: t-SNE embedding (perplexity=30) of latent representation of 3D neuronal morphologies of the BBP dataset showing **A.** a separation into spiny and aspiny neurons ($n = 616$). **B.** t-SNE embedding (perplexity=30) of the latent representations of the morphologies of the spiny neurons colored by the cluster found by our model ($n = 286$). **C.** Relative cortical layer distribution of neurons per cluster across L2/3—L6. Higher values are indicated by red. **D.** As **B.** but neurons colored by their total apical length revealing an organization of the latent space in terms of morphological properties. **E.** Log-likelihood of Gaussian Mixture Model on held-out test set for spiny neurons used to select the optimal number of clusters. **F.** Example neurons for each cluster are shown with apical dendrites in lighter color, while basal dendrites are colored darker. Soma is indicated by black circle.

5.3 Message-passing is not sufficient for long-range graphs

Next, we investigate whether classical message-passing is sufficient to process graphs with long-ranging branches such as neuronal morphologies. Therefore, we train GraphDINO once when only using message passing while removing the global attention (setting $\lambda = 0$ in Eq. 1). This decreases the performance to 59.8% (Tab. 3). Additionally, we train InfoGraph (Sun et al., 2020), as a baseline for an unsupervised method that learns graph-level representations and uses GNN message-passing. InfoGraph achieves accuracy of 48.4% (Tab. 3). Thus, we conclude that using global attention is beneficial in situations where graphs contain long-range branches. Global attention enables information flow between distant (in terms of graph connectivity) nodes that might be close in space or function.

Table 3: Cell-type classification accuracy [%] on the BBP dataset. Performance of our model and InfoGraph (Sun et al., 2020) averaged over three random seeds and given as mean ± standard deviation.

Model	**Accuracy**
InfoGraph (Sun et al., 2020)	48.2 ± 0
GraphDINO	**65.8** ± 1

5.4 Morphological embeddings differentiate between spiny/aspiny cells and layers

To evaluate the capability of GraphDINO to capture essential features of 3D neuronal shapes purely data-driven, we train GraphDINO on the BBP dataset and use t-distributed stochastic neighbor embedding (t-SNE) (van der Maaten & Hinton, 2008) to map the learned embeddings of the BBP dataset into 2D (Fig. 3) for visualization. A clear separation between spiny and aspiny neurons can be observed (see Fig. 3**A**), indicating that our learned representation captures meaningful biological differences of the neuronal morphologies.

Interestingly, some of the spiny neurons end up in the aspiny cluster (Fig. 3**A** bottom right). These are inverted L6 neurons (Fig. 3**F** Cluster 15), whose size and dendritic tree are morphologically similar to the surrounding aspiny neurons that also show a downwards bias in the dendritic tree.

5.5 Morphological embeddings recover known excitatory cell types

To identify cell types, we fit a Gaussian mixture model (GMM) with a diagonal covariance matrix to our learned representation of the spiny neurons. To determine the number of clusters, we fit 1'000 GMMs with different random seeds using five-fold cross-validation for 2—30 clusters. We average over the log-likelihood for each number of clusters over repetitions and folds. We find $n = 15$ to be the optimal number of clusters (Fig. 3**E**).

Having identified the optimal number of clusters, we re-fit the GMM to the full dataset including all spiny neurons. To avoid picking a particularly good or bad random clustering, we fit 100 models and choose the one that has the highest average adjusted rand index (ARI) to all other clusterings.

The spiny neurons cluster nicely into different shapes and layers (Fig. 3**F** and Appendix Fig. D.6), retrieving known excitatory cell types. The first four spiny clusters contain mainly cells from layer 2/3 (L2/3) (Fig. 3**C**) and group them by morphology: Cluster 1 contains wide and short neurons from layer 2/3, while L2/3 neurons in cluster 4 are more elongated with a less pronounced apical tuft (Fig. 3**F**). Clusters 5-7 group cells from layer 4 (L4) (Fig. 3**C**), differentiating between spiny stellate cells (cluster 5) and atufted L4 neurons (cluster 7) (Fig. 3**F**). Within layer 5 and 6, neurons are group by their size, amount of apical tuft and obliques, as well as the direction of the apical-like dendrites: For instance, cluster 10 groups thick-tufted pyramidal cells from layer 5 and cluster 15 contains inverted L6 neurons (Fig. 3**F**).

Most clusters show a strong preference for grouping cells whose soma position is in a certain layer (Fig. 3**C**) even though the model — in contrast to the experts who labeled the cells — does not have access to anatomical knowledge such as cortical layer of origin. One exception are pyramidal L6 cells with upward-directed apicals that separate less well and get rather clustered with L4 and L5 neurons of the same size and similar morphological shape. This is to be expected, as the model only learns to differentiate between different morphologies but has no knowledge about anatomical features such as soma depth.

Table 4: Adjusted rand index (ARI) between identified clusters and expert labels for the learned embeddings from GraphDINO and manually-defined features by (Gouwens et al., 2019) and expert labels, when performing the clustering within cortical layers and across the whole cortex.

Clustering	**Features**	**ARI**
across layers	GraphDINO	0.31
within layers	Gouwens et al. (2019)	0.27
	GraphDINO	**0.46**

Table 5: Balanced accuracy [%] on M1 EXC test set using the learned embeddings from either GraphDINO or MorphVAE (Laturnus & Berens, 2021) (mean ± SEM across three runs and across three data splits, respectively). Percentages in brackets indicate the amount of labels used during training for MorphVAE. GraphDINO is trained without labels.

Model	**Accuracy over runs** (mean ± SEM)	**Accuracy over splits** (mean ± SD)
MorphVAE (100 %)	70 ± 5	-
MorphVAE (0 %)	58± 7	-
Density Map (0 %)	60	-
GraphDINO (0 %)	**68** ± 5	71 ± 9

5.6 Data-driven clusters are consistent with expert labels

To compare our data-driven features to manually-designed features, we compute the adjusted rand index (ARI) between our clusters and the expert-identified cell types on the BBP dataset and compare the performance to the clusters based on morphometrics obtained by Gouwens et al. (2020). We achieve an ARI performance of 0.31 when clustering neurons across all cortical layers together while using significantly less prior information than Gouwens et al. (2019). In comparison, Gouwens et al. (2019) reached an ARI of 0.27 with a feature space specifically designed for spiny neurons and by splitting the neurons into their cortical layer of origin before performing the clustering. This approach reduces the complexity of the problem significantly, since misassignments across layers are excluded by construction. When performing the clustering like Gouwens et al. (2019) only within the layers, we achieve an ARI of 0.46 (Tab. 4).

5.7 Morphological embeddings encode distinct morphological features

Laturnus & Berens (2021) classified the M1 EXC dataset (Scala et al., 2021) into three classes based on presence of an apical tuft (tufted, untufted and others). Following their work, we train a 5-nearest-neighbor classifier on our learned embeddings and show that GraphDINO learns meaningful features to differentiate between the three classes (Tab. 5). Our method outperforms their MorphVAE method as well as a baseline using density maps of the neurons (Laturnus & Berens, 2021). This dataset is rather small and Laturnus & Berens (2021) used only a single train/test split. To estimate how reliable the reported accuracy metrics are, we computed the cross-validated accuracy across multiple different train/test splits, which showed a variability across splits of ±9% (standard deviation; Tab. 5). We conclude that GraphDINO likely outperforms MorphVAE trained unsupervised and performs approximately on par with MorphVAE trained fully supervised.

5.8 Morphological embeddings encode cortical regions

TreeMoCo (Chen et al., 2022b) is an LSTM-based model that was concurrently proposed to perform unsupervised representation learning on neuronal graphs. The model uses as input the simplified skeletons of neurons that only contain the branching points as nodes. They compute 26 manually-selected features in addition to the xyz-coordinates as node features to describe the morphology of the skeletons between branching points. TreeMoCo is trained on a combination of the datasets BIL, JML and ACT and quantitatively evaluated on the task of predicting the brain anatomical region or cortical layer of origin of the neurons on a subset of the neuronal classes. Chen et al. (2022b) remove 955 neurons from the dataset due to "reconstruction errors" and evaluate on a 80-20% training-test split. Since we did not have access to the exact neurons used for training and evaluation both in terms of split and which neurons were removed, we

trained unsupervised on the joint dataset and evaluated using 5-fold cross-validation, i.e. splitting the data into five folds and evaluating each fold, given the other four folds as training data and reporting the average performance across folds. For further details regarding the evaluation, see Appendix C.5.4.

GraphDINO performs on par or better than TreeMoCo and GraphCL when predicting the origin of neurons (Tab. 6). Note that GraphDINO is fully data-driven while TreeMoCo and GraphCL additionally employ manually extracted node features.

Table 6: Cell-type classification on the TreeMoCo dataset. Performance of our model (GraphDINO) averaged over three random seeds and given as mean ± standard deviation. TreeMoCo and GraphCL performance given as the average accuracy over the last five epochs per dataset. *Results taken Fig. C1 of Chen et al. (2022b).

Model	**BIL-6**	**JML-4**	**ACT**	**ACT spiny**
TreeMoCo*	76.9	59.7	53.9	-
GraphCL*	66.3	50.6	55.6	-
GraphDINO	$79_{\pm 1}$	$63_{\pm 6}$	$54_{\pm 5}$	$73_{\pm 6}$

Note that the evaluation reported by Chen et al. (2022b) uses excitatory and inhibitory neurons at the same time. With this approach, morphologies of neurons of the "same" class can look very different (Fig. C.5). A better proxy task to evaluate the encoding capabilities of the models would be to restrict the evaluation to only excitatory cells. For the ACT dataset this information is available. We therefore repeated the evaluation only on this subset (Tab. 6), which should provide a more meaningful baseline for future studies.

6 Limitations

GraphDINO is designed to learn graph-level representations of spatially-embedded tree-structured graphs using self-supervised learning. As we focus on graphs where each node has a location in 3D space and design the data augmentations accordingly, the approach is not expected to work out-of-the-box on graphs that have different node features. AC-Attention is likely to be beneficial in many other scenarios as well, since it can smoothly interpolate between message passing and global attention based on node similarity, but this hypothesis remains to be tested empirically. Data augmentations would need to be adapted to the respective data domain and the respective invariances that should be encoded or supervised learning to be used. The attention mechanism is not tied in any way to the self-supervised learning objective we use.

Computing the full transformer attention matrix has a quadratic complexity and might therefore be computationally infeasible for graphs with a large number of nodes. We solve this problem here by subsampling the neuronal skeletons to a smaller number of nodes, which has the added benefit of being a strong data augmentation that keeps the global morphology of the neuron intact while altering the local structure between the two views. However, this approach might not be suitable for all graph datasets. There has been some work in building attention mechanism that scale linearly with the number of input tokens (Wang et al., 2020; Kitaev et al., 2020; Choromanski et al., 2021), but integrating them with the message passing might not be straightforward.

In terms of neuronal cell type classification, we did not take some features into account that have been previously used to differentiate cell types, such as the shape of the soma (as formerly used for GABAergic interneurons) or spine densities (Ascoli et al., 2008). Future work could focus on incorporating them into our framework. Depending on the type of feature, they could be easily integrated by adding them as features of the graph or as additional node features.

7 Conclusion

Increasingly large and complex datasets of neurons have given rise to the need for unbiased and quantitative approaches to cell type classification. We have demonstrated one such approach that is purely data-driven and self-supervised, and that learns a low-dimensional representation of the 3D shape of a neuron. By using self-supervised learning, we do not pre-specify which cell types to learn and which features to use, thereby reducing bias in the classification process and opening up the possibility to discover new cell types. A similar

approach can also be useful in other domains beyond neuroscience, where samples of the dataset are spatial graphs and graph-level embeddings are desired, such as tree classification in forestry.

References

Allen Institute. Allen cell types database technical white paper: Cell morphology and histology. 2016. URL `http://help.brain-map.org/download/attachments/8323525/CellTypes_Morph_Overview.pdf`.

Rubén Armañanzas and Giorgio A. Ascoli. Towards the automatic classification of neurons. *Trends in Neurosciences*, 38(5):307–318, 2015.

Giorgio Ascoli, Lidia Alonso-Nanclares, Stewart Anderson, Germán Barrionuevo, Ruth Benavides-Piccione, Andreas Burkhalter, Gyorgy Buzsáki, Bruno Cauli, Javier Defelipe, and Alfonso Fairen. Petilla terminology: nomenclature of features of GABAergic interneurons of the cerebral cortex. *Nature reviews. Neuroscience*, 9:557–568, 2008.

Cathryn Cadwell, Athanasia Palasantza, Xiaolong Jiang, Philipp Berens, Qiaolin Deng, Marlene Yilmaz, Jacob Reimer, Shan Shen, Matthias Bethge, Kimberley Tolias, Rickard Sandberg, and Andreas Tolias. Electrophysiological, transcriptomic and morphologic profiling of single neurons using patch-seq. *Nature Biotechnology*, 34, 2015.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9650–9660, 2021.

Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning, 2022a.

Hanbo Chen, Jiawei Yang, Daniel Maxim Iascone, Lijuan Liu, Lei He, Hanchuan Peng, and Jianhua Yao. Treemoco: Contrastive neuron morphology representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proc. of the International Conf. on Machine learning (ICML)*, 2020.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021.

Javier Defelipe, Pedro López-Cruz, Ruth Benavides-Piccione, Concha Bielza, Pedro Larranaga, Stewart Anderson, Andreas Burkhalter, Bruno Cauli, Alfonso Fairen, Dirk Feldmeyer, Gord Fishell, David Fitzpatrick, Tamás Freund, Guillermo Gonzalez Burgos, Shaul Hestrin, Sean Hill, Patrick Hof, Josh Huang, Edward Jones, and Giorgio Ascoli. New insights into the classification and nomenclature of cortical gabaergic interneurons. *Nature reviews. Neuroscience*, 14, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ARXIV*, 2020.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv.org*, 2012.09699, 2021.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv.org*, 2003.00982, 2020.

Leila Elabbady, Sharmishtaa Seshamani, Shang Mu, Gayathri Mahalingam, Casey M Schneider-Mizell, Agnes Bodor, J Alexander Bae, Derrick Brittain, JoAnn Buchanan, Daniel J Bumbarger, et al. Quantitative census of local somatic features in mouse visual cortex. *bioRxiv*, 2022.

Rohan Gala, Agata Budzillo, Fahimeh Baftizadeh, Jeremy Miller, Nathan Gouwens, Anton Arkhipov, Gabe Murphy, Bosiljka Tasic, Hongkui Zeng, Michael Hawrylycz, et al. Consistent cross-modal identification of cortical neurons with coupled autoencoders. *Nature Computational Science*, 1(2):120–127, 2021.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proc. of the International Conf. on Machine learning (ICML)*, pp. 1263–1272, 2017.

Nathan Gouwens, Staci Sorensen, Jim Berg, Changkyu Lee, Tim Jarsky, Jonathan Ting, Susan Sunkin, David Feng, Costas Anastassiou, Eliza Barkan, Kris Bickley, Nicole Blesie, Thomas Braun, Krissy Brouner, Agata Budzillo, Shiella Caldejon, Tamara Casper, Dan Castelli, Peter Chong, and Christof Koch. Classification of electrophysiological and morphological neuron types in the mouse visual cortex. *Nature Neuroscience*, 22, 2019.

Nathan W. Gouwens, Staci A. Sorensen, Fahimeh Baftizadeh, Agata Budzillo, Brian R. Lee, Tim Jarsky, Lauren Alfiler, Katherine Baker, Eliza Barkan, Kyla Berry, Darren Bertagnolli, Kris Bickley, Jasmine Bomben, Thomas Braun, Krissy Brouner, Tamara Casper, Kirsten Crichton, Tanya L. Daigle, Rachel Dalley, Rebecca A. de Frates, Nick Dee, Tsega Desta, Samuel Dingman Lee, Nadezhda Dotson, Tom Egdorf, Lauren Ellingwood, Rachel Enstrom, Luke Esposito, Colin Farrell, David Feng, Olivia Fong, Rohan Gala, Clare Gamlin, Amanda Gary, Alexandra Glandon, Jeff Goldy, Melissa Gorham, Lucas Graybuck, Hong Gu, Kristen Hadley, Michael J. Hawrylycz, Alex M. Henry, DiJon Hill, Madie Hupp, Sara Kebede, Tae Kyung Kim, Lisa Kim, Matthew Kroll, Changkyu Lee, Katherine E. Link, Matthew Mallory, Rusty Mann, Michelle Maxwell, Medea McGraw, Delissa McMillen, Alice Mukora, Lindsay Ng, Lydia Ng, Kiet Ngo, Philip R. Nicovich, Aaron Oldre, Daniel Park, Hanchuan Peng, Osnat Penn, Thanh Pham, Alice Pom, Zoran Popović, Lydia Potekhina, Ramkumar Rajanbabu, Shea Ransford, David Reid, Christine Rimorin, Miranda Robertson, Kara Ronellenfitch, Augustin Ruiz, David Sandman, Kimberly Smith, Josef Sulc, Susan M. Sunkin, Aaron Szafer, Michael Tieu, Amy Torkelson, Jessica Trinh, Herman Tung, Wayne Wakeman, Katelyn Ward, Grace Williams, Zhi Zhou, Jonathan T. Ting, Anton Arkhipov, Uygar Sümbül, Ed S. Lein, Christof Koch, Zizhen Yao, Bosiljka Tasic, Jim Berg, Gabe J. Murphy, and Hongkui Zeng. Integrated morphoelectric and transcriptomic classification of cortical gabaergic cells. *Cell*, 183(4):935–953.e19, 2020.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1025–1035, 2017.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *Proc. of the International Conf. on Machine learning (ICML)*, volume 119, pp. 4116–4126, 2020.

Lida Kanari, Pawel Dlotko, Martina Scolamiero, Ran Levi, Julian C. Shillcock, Kathryn Hess, and Henry Markram. A topological representation of branching neuronal morphologies. *Neuroinformatics*, 16:3 – 13, 2017.

Lida Kanari, Srikanth Ramaswamy, Ying Shi, Sebastien Morand, Julie Meystre, Rodrigo Perin, Marwan Abdellah, Yun Wang, Kathryn Hess, and Henry Markram. Objective morphological classification of neocortical pyramidal cells. *Cerebral Cortex*, 29(4):1719–1735, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

Johannes Klicpera, Stefan Weiß enberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Sophie C. Laturnus and Philipp Berens. Morphvae: Generating neural morphologies from 3d-walks using a variational autoencoder with spherical latent space. In *Proc. of the International Conf. on Machine learning (ICML)*, volume 139, pp. 6021–6031, 2021.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *Proc. of the International Conf. on Learning Representations (ICLR)*, 2016.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *arXiv.org*, 1608.03983, 2016.

Yanbin Lu, Lawrence Carin, Ronald Coifman, William Shain, and Badrinath Roysam. Quantitative arbor analytics: unsupervised harmonic co-clustering of populations of brain cell arbors based on l-measure. *Neuroinformatics*, 13(1):47–63, 2015.

Henry Markram, Eilif Muller, Srikanth Ramaswamy, Michael Reimann, Marwan Abdellah, Carlos Aguado, Anastasia Ailamaki, Lidia Alonso-Nanclares, Nicolas Antille, Selim Arsever, Atenekeng Kahou Guy Antoine, Thomas K Berger, Ahmet Bilgili, Nenad Buncic, Athanassia Chalimourda, Giuseppe Chindemi, Jean-Denis Courcol, Fabien Delalondre, Vincent Delattre, and Felix Schürmann. Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163:456–492, 2015.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers, 2021.

Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv.org*, 1707.05005, 2017.

Marcel Oberlaender, Christiaan P. J. de Kock, Randy M. Bruno, Alejandro Ramirez, Hanno S. Meyer, Vincent J. Dercksen, Moritz Helmstaedter, and Bert Sakmann. Cell Type–Specific Three-Dimensional Structure of Thalamocortical Circuits in a Column of Rat Vibrissal Cortex. *Cerebral Cortex*, 22(10): 2375–2391, 2012.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv.org*, 1807.03748, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011.

Liu L Kuang X Wang Y Qu L Gong H Jiang S Li A Ruan Z Ding L Yao Z Chen C Chen M Daigle TL Dalley R Ding Z Duan Y Feiner A He P Hill C Hirokawa KE Hong G Huang L Kebede S Kuo HC Larsen R Lesnar P Li L Li Q Li X Li Y Li Y Liu A Lu D Mok S Ng L Nguyen TN Ouyang Q Pan J Shen E Song Y Sunkin SM Tasic B Veldman MB Wakeman W Wan W Wang P Wang Q Wang T Wang Y Xiong F Xiong W Xu W Ye M Yin L Yu Y Yuan J Yuan J Yun Z Zeng S Zhang S Zhao S Zhao Z Zhou Z Huang ZJ Esposito L Hawrylycz MJ Sorensen SA Yang XW Zheng Y Gu Z Xie W Koch C Luo Q Harris

JA Wang Y Zeng H Peng H, Xie P. Morphological diversity of single neurons in molecularly defined cell types. *Nature*, 2021. doi: https://doi.org/10.1038/s41586-021-03941-1.

Sridevi Polavaram, Todd A Gillette, Ruchi Parekh, and Giorgio A Ascoli. Statistical analysis and data mining of digital reconstructions of dendritic morphologies. *Frontiers in neuroanatomy*, 8:138, 2014.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proc. of Conf. on Knowledge Discovery and Data Mining (KDD)*, pp. 1150–1160, 2020.

Srikanth Ramaswamy, Jean-Denis Courcol, Marwan Abdellah, Stanislaw R. Adaszewski, Nicolas Antille, Selim Arsever, Guy Atenekeng, Ahmet Bilgili, Yury Brukau, Athanassia Chalimourda, Giuseppe Chindemi, Fabien Delalondre, Raphael Dumusc, Stefan Eilemann, Michael Emiel Gevaert, Padraig Gleeson, Joe W. Graham, Juan B. Hernando, Lida Kanari, Yury Katkov, Daniel Keller, James G. King, Rajnish Ranjan, Michael W. Reimann, Christian Rössert, Ying Shi, Julian C. Shillcock, Martin Telefont, Werner Van Geit, Jafet Villafranca Diaz, Richard Walker, Yun Wang, Stefano M. Zaninetta, Javier DeFelipe, Sean L. Hill, Jeffrey Muller, Idan Segev, Felix Schürmann, Eilif B. Muller, and Henry Markram. The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex. *Frontiers in Neural Circuits*, 9: 44, 2015.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer, 2022.

Santiago Ramón y Cajal. *Histologie du système nerveux de l'homme et des vertébrés.* 1911.

Federico Scala, Dmitry Kobak, Matteo Bernabucci, Yves Bernaerts, Cathryn Cadwell, Jesus Castro, Leonard Hartmanis, Xiaolong Jiang, Sophie Laturnus, Elanine Miranda, Shalaka Mulherkar, Zheng Tan, Zizhen Yao, Hongkui Zeng, Rickard Sandberg, Philipp Berens, and Andreas Tolias. Phenotypic variation of transcriptomic cell types in mouse motor cortex. *Nature*, 598:1–7, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Philipp Schubert, Sven Dorkenwald, Michal Januszewski, Viren Jain, and Joergen Kornfeld. Learning cellular morphology with neural networks. *Nature Communications*, 10:2736, 2019.

Ruggero Scorcioni, Sridevi Polavaram, and Giorgio A Ascoli. L-measure: a web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. *Nature protocols*, 3 (5):866–876, 2008.

Dominik Seidel, Yonten Dorji, Bernhard Schuldt, Emilie Isasa, and Klaus Körber. Dataset: New insights into tree architecture from mobile laser scanning and geometry analysis. *Dryad*, 2021. doi: https://doi.org/10.5061/dryad.2fqz612n6.

Sharmishtaa Seshamani, Leila Elabbady, Casey Schneider-Mizell, Gayathri Mahalingam, Sven Dorkenwald, Agnes Bodor, Thomas Macrina, Daniel Bumbarger, JoAnn Buchanan, Marc Takeno, Wenjing Yin, Derrick Brittain, Russel Torres, Daniel Kapner, Kisuk Lee, Ran Lu, Jingpeng Wu, Nuno daCosta, R. Clay Reid, and Forrest Collman. Automated neuron shape analysis from electron microscopy. *arXiv.org*, 2006.00100, 2020.

D. A. Sholl. Dendritic organization in the neurons of the visual and motor cortices of the cat. *Journal of Anatomy*, 87:387–406, 1953.

Fan-Yun Sun, Jordan Hoffmann, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2020.

Harry BM Uylings and Jaap Van Pelt. Measures for quantifying dendritic arborizations. *Network: computation in neural systems*, 13(3):397, 2002.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 9(86):2579–2605, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *ARXIV*, 2020.

Yun Wang. A simplified morphological classification scheme for pyramidal cells in six layers of primary somatosensory cortex of juvenile rats. *IBRO Reports*, 5, 2018.

Ferreira TA Wu Z Economo MN Edson P Arthur BJ Bruns C Rokicki K Schauder D Olbris DJ Murphy SD Ackerman DG Arshadi C Baldwin P Blake R Elsayed A Hasan M Ramirez D Dos Santos B Weldon M Zafar A Dudman JT Gerfen CR Hantman AW Korff W Sternson SM Spruston N Svoboda K Chandrashekar J Winnubst J, Bas E. Reconstruction of 1,000 projection neurons reveals new cell types and organization of long-range connectivity in the mouse brain. 2019. doi: https://doi.org/10.1016/j.cell.2019.07.042.

Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. Self-supervised graph-level representation learning with local and global structure. In *Proc. of the International Conf. on Machine learning (ICML)*, volume 139, pp. 11548–11558, 2021.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *NIPS*, 2021.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *NIPS*, 2020.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv.org*, 2001.05140, 2020.

Jie Zhao, Xuejin Chen, Zhiwei Xiong, Zheng-Jun Zha, and Feng Wu. Graph representation learning for large-scale neuronal morphological analysis. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2022. doi: 10.1109/TNNLS.2022.3204686.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pp. 2069–2080, 2021.

Appendices

A Synthetic graph dataset

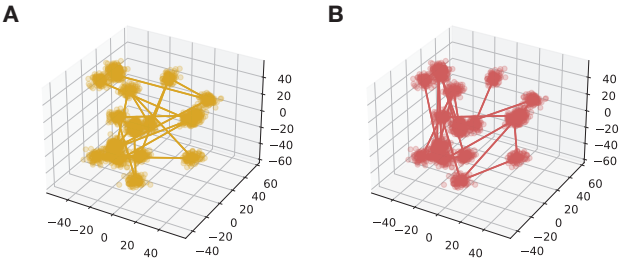


Figure A.1: Example classes 1 (A) and 2 (B) of synthetic graph dataset. Samples within one class share graph connectivity. Samples between classes share mean node locations. Node locations are drawn from $\mathcal{N}(\mu, \sigma)$.

To test whether our model is able to use information encoded in the connectivity of the graphs, we generate a synthetic graph dataset with five classes that differ in connectivity while having similar node locations. We create this synthetic graph dataset by uniformly sampling 20 mean node positions in 3D space in $[-50, 50]^3$. The mean node locations are shared between the five classes to ensure that the presence of a specific node does not encode class membership. For each class, we construct a distinct graph connectivity as follows: We first randomly sample a root node and two children, then we recursively sample one or two children per child (with a branching probability of 50%) until all 20 nodes are connected. Using this method, we generate 100,000 graphs for the training set and 10,000 graphs for validation and test set each (with $\frac{1}{5}$ class probability) by sampling node positions from $\mathcal{N}(\mu, \sigma)$ with μ equal to the above drawn means and $\sigma = 10$.

Tab. C.1, Tab. C.2 and Tab. C.3 list the hyperparamers used for experiments on the synthetic graphs.

t-SNE of the learned latent spaces: To visualize the learned latent space we perform t-SNE with a perplexity of 30 to reduce the embedding to two dimensions (Fig. A.2).

Linear classifier: We train a supervised linear classifier on the extracted embeddings of GraphDINO for 100 epochs and a learning rate of 0.01. To train the classifier, we use the test set that has not been used in training GraphDINO, and split it in 8,000 samples for training the classifier and 2,000 samples for evaluating held-out test set accuracy.

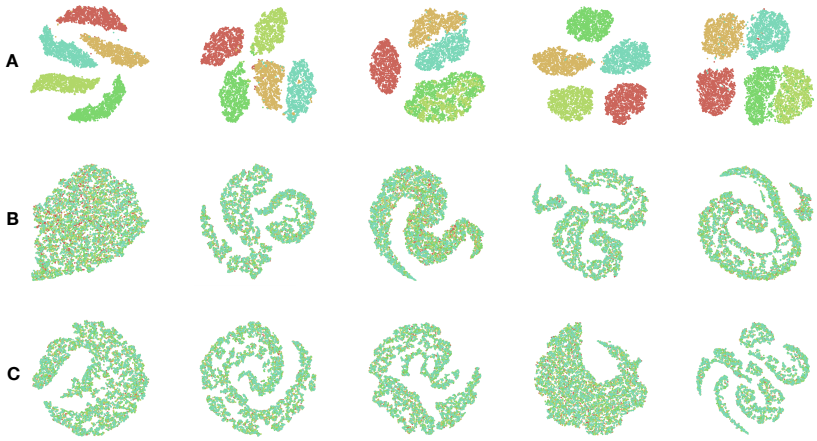


Figure A.2: t-SNE embedding of synthetic graph dataset colored by class membership for **A** five runs of GraphDINO with GraphAttention, **B** five runs of GraphDINO with regular transformer attention, and **C** five runs of GraphDINO with transformer attention and without positional encoding.

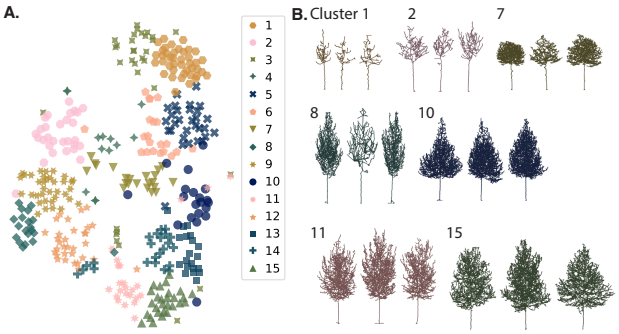


Figure B.3: **A.** t-SNE embedding of Trees dataset colored by cluster membership based on GMM clustering with 15 clusters. **B.** Three example tree morphologies are shown for different clusters.

B Application to different domain: Tree Morphologies

We developed a model that is able to learn graph-level embeddings of spatially-embedded graphs. So far, we have shown that it yields meaningful cell types clusterings of neuronal morphologies. To show that GraphDINO is applicable to data domains beyond neuronal morphologies, we train our model on 3D skeletons of individual trees (from a forest).

The Trees dataset (Seidel et al., 2021) is a highly divers dataset comprised of 391 skeletons of trees stemming from 39 different genuses and 152 species or breedings. The skeletons were extracted from LIDAR scans of the trees. Nodes of the skeletons have a 3D coordinate. We normalize the data such that the lowest point (start of the tree trunk) is normalized to (0, 0, 0).

GraphDINO learns a latent space that orders tree morphologies with respect to their size, crown size and crown shape (Fig. B.3, Fig. D.7).

C Extended Methods

C.1 Background: DINO

DINO (Fig. C.4) (Caron et al., 2021) is a method for self-supervised image representation learning. Similar to previous approaches, it consists of two image encoders which process different views of an image. These views are obtained by image augmentation. The training objective is to enforce both encoders to generate the same output distribution when the same input image is shown. This can be implemented by the cross entropy loss function: $\sum_i -q_i \log p_i$. Both encoders are transformers that share the architecture but differ in their weights: One of the encoders is the student encoder which receives weight updates through gradients of the training objective while the other encoder's (teacher) weights are an exponential moving average of the student's weights. In contrast to some other self-supervised methods, DINO does not require contrastive (negative) samples. To prevent collapse, i.e. predicting the same distribution independent of the input image, two additional operations on the teacher's predictions are crucial: sharpening by adjusting the softmax temperature, and centering using batch statistics. Besides competitive performance on downstream image classification tasks, another key finding of the paper is that object segmentations emerge in the self-attention when applying DINO training on visual transformer image encoders.

C.2 Data preprocessing.

To speed up data loading during training, we reduce the number of nodes in the graph of each neuron to 1000 nodes in the same way as when subsampling and ensure that it contains only one connected component. If there are unconnected components, we connect them by adding an edge between two nodes of two unconnected components that have the least distance between their spatial coordinates.

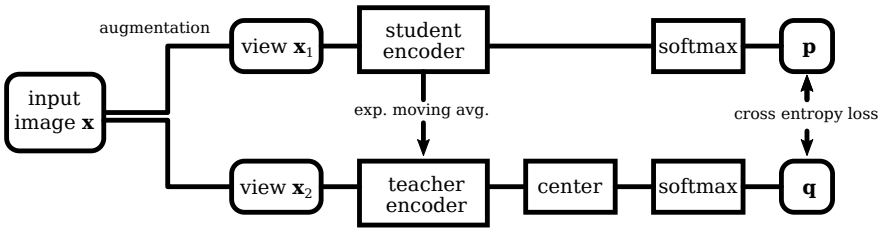


Figure C.4: The DINO method for self-supervised image representation learning (figure adapted from Caron et al. (2021)).

C.3 Training details and hyperparameters

To select hyperparameters we run three grid searches and pick the best hyperparameters according to the lowest average loss over the BBP and M1 PatchSeq dataset.

For the optimization, we run a hyperparameter search over batch size $\in \{32, 64, 128\}$, learning rate $\in \{10^{-3}, 10^{-4}, 10^{-5}\}$, and number of training iterations $\in \{20,000, 50,000, 100,000\}$.

For the augmentation strength, we run a hyperparameter search over jitter variance $\sigma_1 \in \{1.0, 0.1, 0.001\}$, number of deleted branches$n \in \{1, 5, 10\}$, and graph position variance $\sigma_2 \in \{0.1, 1.0, 10.0\}$.

For the architecture, we run a hyperparameter search over latent dimension $\in \{16, 32, 64\}$, number of GraphAttention blocks (depth) $\in \{5, 7, 10\}$, and number of attention heads per block $\in \{2, 4, 8\}$.

C.3.1 Architecture Hyperparameters

Table C.1: Hyperparameters used for the architecture for the different datasets. **PE:** Positional encoding. **T temp:** Softmax temperature of teacher network.

Dataset	Dims p	Dims z	N layers	N heads	MLP dims	PE dims	T temp
Synthetic Graphs	300	16	4	4	16	16	0.04
BBP	1000	32	10	8	64	32	0.06
M1 PatchSeq	1000	64	7	8	64	32	0.06
Joint dataset (BIL, JML, ACT)	1000	32	7	4	64	32	0.06
Trees	1000	32	7	8	64	32	0.06

Tab. C.1 lists the hyperparameters used for the architecture for the different datasets. For the synthetic graph dataset, we downscale the network due to being a simpler dataset. DINO (Caron et al., 2021) uses an output dimensionality of 65,536 for p when training on ImageNet (Deng et al., 2009) (1,000 classes). The number of classes in the neuronal datasets is unknown, but previous literature described 14 – 19 cell types (Gouwens et al., 2019; Markram et al., 2015). Hence, we decrease the number of dimensions of p proportionally to 1,000, approximately retaining the ratio between classes and number of dimensions.

C.3.2 Optimization Hyperparameters

The learning rate is linearly increased to the value given in Tab. C.2 during the first 2,000 iterations and then decayed using a exponential decay with rate 0.5 (Loshchilov & Hutter, 2016).

Table C.2: Hyperparameters used for optimization for the different datasets.

Dataset	**Iterations**	**Batch size**	**Learning rate**
Synthetic Graphs	100,000	512	10^{-4}
BBP	100,000	64	10^{-4}
M1 PatchSeq	50,000	128	10^{-3}
Joint dataset (BIL, JML, ACT)	100,000	128	10^{-3}
Trees	100,000	64	10^{-4}

C.3.3 Augmentation Hyperparameters

Table C.3: Augmentation hyperparameters for the different datasets. **N nodes:** Number of nodes to subsample to. σ_1: Variance of node jittering. **N DB:** Number of deleted branches. σ_2: Variance of graph translation.

Dataset	**N nodes**	σ_1	**N DB**	σ_2
Synthetic Graphs	15	0.1	0	0
BBP	100	0.001	10	10.0
M1 PatchSeq	100	0.1	10	10.0
Joint dataset (BIL, JML, ACT)	200	1.0	5	10.0
Trees	200	0.1	5	10.0

C.3.4 Computation

All trainings were performed on a NVIDIA Quadro RTX 5000 single GPU. Training on the neuronal BBP dataset ran for approximately 10 hours for 100,000 training iterations.

C.4 Inference

To extract the latent representation per sample, we encode the unaugmented graphs subsampled to 200 nodes using the student encoder and extract the latent representation z using the weights of the last iteration of training (no early-stopping is used).

C.5 Evaluation

C.5.1 Evaluation on BBP

For visualization of the latent space, we use t-distributed stochastic neighbor embedding (t-SNE) (van der Maaten & Hinton, 2008) with PCA-initialization, Euclidean distance and a perplexity of 30.

For quantitative evaluation we use the subset of labeled excitatory neurons ($n = 286$) with the following 14 expert labels: L23-PC, L4-PC, L4-SP, L4-SS, L5-STPC, L5-TTPC1, L5-TTPC2, L5-UTPC, L6-BPC, L6-IPC, L6-TPC-L1, L6-TPC-L4, L6-UTPC, L6-HPC (Markram et al., 2015).

For the ablation experiments and the comparison to InfoGraph Sun et al. (2020), we perform k-nearest neighbor (kNN) classification with $k = 5$ in a leave-one-out setting predicting the above listed expert labels with two exceptions: We remoev the L6-HPC cells, since there are only three samples in the dataset, and we group the L5-TTPC1 and L5-TTPC2 into one class L5-TTPC following previous work that found that they rather form a continuum then two separate classes (Gouwens et al., 2019; Kanari et al., 2019).

For the clustering analysis and the comparison to Gouwens et al. (2019), we follow Gouwens et al. (2019) and compute the adjusted rand index between our found clusters and the 14 expert labels. To determine the optimal number of clusters, we use cross-validation to compute the log-likelihood of held-out data of the Gaussian Mixture model and choose the number of clusters with the highest log-likelihood. The optimal

number of clusters is 15 for the BBP dataset. To perform clustering within cortical layers, we chose the number of clusters per layer based on the number of clusters with the majority of cells from the cortex-wide clustering (Fig. 3): four for layer 2/3, layer 5 and layer 6 and three for layer 4.

C.5.2 Comparison to InfoGraph (Sun et al., 2020)

We use the official implementation[6] to train InfoGraph on the BBP dataset. We perform a hyperparameter search for InfoGraph as detailed in the original publication (Sun et al., 2020) and extend it to include more training epochs to train it for approximately the same number of iterations as GraphDINO. In detail, we run a grid search over learning rate (lr) $\in \{10^{-2}, 10^{-3}, 10^{-4}\}$, number of training epochs $\in \{10, 20, 100, 200, 1,000, 2,000\}$ and GNN layers $\in \{4, 8, 12\}$. We select the hyperparameters based on the lowest unsupervised loss. The chosen hyperparameters are: $lr = 0.001$, $epochs = 1,000$ and four GNN layers with a hidden dimensionality of 32.

We evaluate the performance of InfoGraph (Sun et al., 2020) using a kNN classifier analogous to the ablation experiments (Appendix. C.5.1).

C.5.3 Comparison to MorphVAE (Laturnus & Berens, 2021)

We follow the evaluation protocol of Laturnus & Berens (2021) and perform k-nearest neighbor (kNN) classification with $k = 5$ on the learned latent embeddings of the excitatory neurons to predict whether they are untufted, tufted or "other" on the test set ($n = 60$) and report the balanced accuracy. The "other" class only contains six examples in the test set. To get an estimate of the variance that is due to the chosen data split, we additionally evaluate three further data splits and report the average test set performance over the three splits. We report the performance of MorphVAE as given in Tab. 3 of Laturnus & Berens (2021).

C.5.4 Comparison to TreeMoCo (Chen et al., 2022b)

A fair comparison to TreeMoCo proved difficult. We tried to replicate their setting as best as possible from the information given in the paper as well as by inferring it from their code base[7] while trying to set up a more fair benchmark for future works.

We downloaded the three datasets BIL, JML and ATC using the official code base of TreeMoCo and used it to assign the eleven class labels: L1, L2/3, L4, L5, L6, VPM, CP, VPL, SUB, PRE, MG and Others as used by Chen et al. (2022b). However, our cell counts slightly differ from those given in Chen et al. (2022b). More specifically, the JML dataset contained 1,200 neurons instead of 1,107.

Chen et al. (2022b) removed a substantial amount of the neurons (995 of 3,358 neurons) from the datasets due to reconstruction errors. Since we did not have access to the identities of these neurons, we trained GraphDINO unsupervised on all cells with more than 200 nodes ($n_{total} = 3,138$; $n_{BIL} = 1,739$, $n_{JML} = 889$, $n_{ACT} = 510$) and evaluated the proposed classes as assigned by the TreeMoCo code base. We replicated the proposed data preprocessing by centering the somata at (0, 0, 0) and aligning the neurons' first principle component to the y-axis.

Chen et al. (2022b) performs the quantitative evaluation on a 80-20% training-test data split. Since we did not have access to the exact split, we performed five cross-validations instead and report the average accuracy over folds.

According to the paper, Chen et al. (2022b) perform k-nearest neighbor classification ($k = 5$ or $k = 20$ depending on the dataset). We unify the evaluation and report the kNN accuracy with $k = 5$ for all experiments in this paper. For reference, we list the $k = 20$ performance in Tab. C.4. In their code base, the implementation of kNN is weighted, where the neighbors vote is weighted by the cosine similarity of the embeddings. We follow the description in the paper (Chen et al., 2022b) and use the standard kNN classification without weighing the neighbors' votes.

[6] `https://github.com/sunfanyunn/InfoGraph`

[7] We additionally tried to reach out to the authors but did not get a reply.

Table C.4: Cell-type classification on the TreeMoCo dataset. Performance of our model (GraphDINO) averaged over three random seeds and given as mean ± standard deviation when using $k = 20$ for the kNN classifer.

Model	**BIL-6**	**ACT**
Ours	78 ± 2	54± 4

Figure C.5: Example neurons labeled as Isocortex 4 of the ACT dataset.

The performances reported by Chen et al. (2022b) are overfitted on the test set: They pick the best test set performance over epochs (for the three datasets separately) (see Fig. C1 in Chen et al. (2022b)). Additionally, they picked whether to use the latent embedding z or the projection head's output p based on the test set performance per dataset. To give an estimate of the less overfitted performance of TreeMoCo (Chen et al., 2022b) (at least with respect to which epoch to evaluate), we report the averaged performance over the last five epochs given by Fig. C1 (Chen et al., 2022b).

Similarly, the performance of GraphCL (You et al., 2020) as reported by Chen et al. (2022b) is picked as the best test set performance per dataset over training epochs. We therefore report the average accuracy over the last five epochs with the given by Fig. C1 (Chen et al., 2022b).

D Complete cluster visualizations

In the Fig. D.6 and Fig. D.7, we show the cluster assignments of all samples of the excitatory BBP dataset ($n = 286$) and the Trees dataset ($n = 391$), respectively.

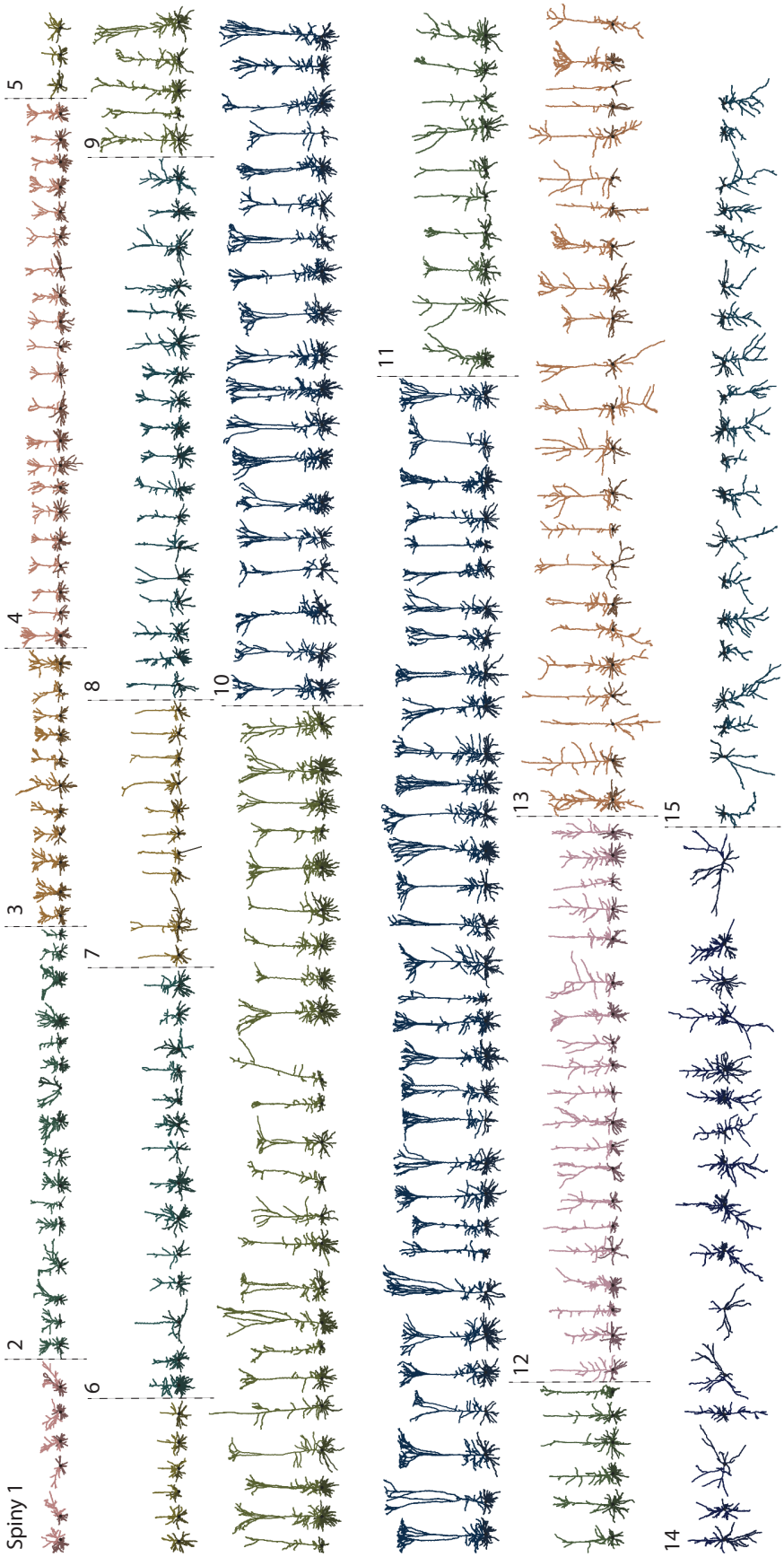


Figure D.6: Clusters of spiny neurons of BBP dataset as identified by GMM based on our learned feature space. Apical dendrites are colored lighter, while basal dendrites are shown in a darker color. Soma is marked by a black circle.

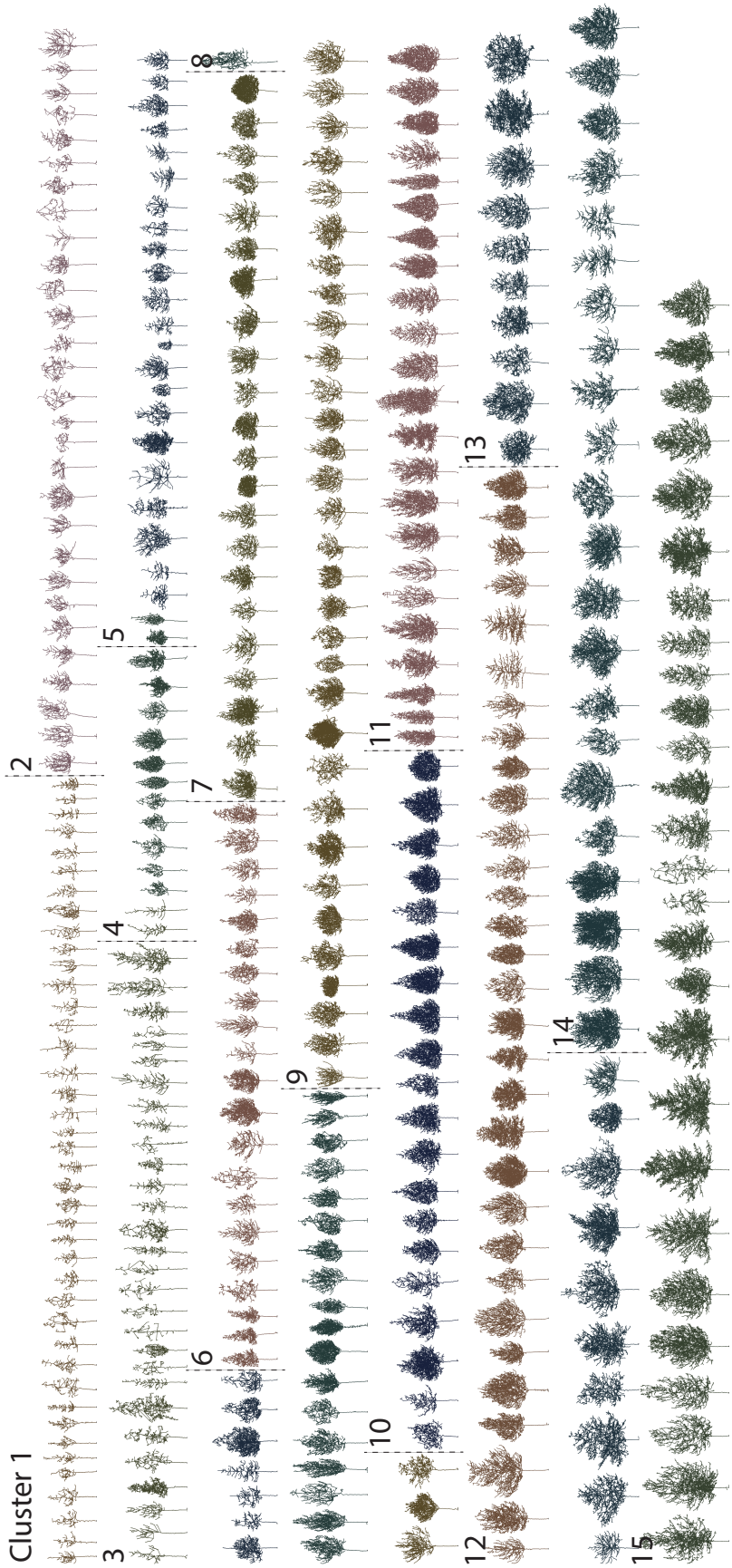


Figure D.7: Clusters of trees as identified by GMM based on our learned feature space.