MorphGrower: A Synchronized Layer-by-layer Growing Approach for Plausible Neuronal Morphology Generation

Nianzu Yang [*1] **Kaipeng Zeng** [*1] **Haotian Lu** [1] **Yexin Wu** [1] **Zexin Yuan** [2] **Danni Chen** [2] **Shengdian Jiang** [3] **Jiaxiang Wu** [4] **Yimin Wang** [2] **Junchi Yan** [1]

Abstract

Neuronal morphology is essential for studying brain functioning and understanding neurodegenerative disorders. As acquiring real-world morphology data is expensive, computational approaches for morphology generation have been studied. Traditional methods heavily rely on expert-set rules and parameter tuning, making it difficult to generalize across different types of morphologies. Recently, MorphVAE was introduced as the sole learning-based method, but its generated morphologies lack plausibility, i.e., they do not appear realistic enough and most of the generated samples are topologically invalid. To fill this gap, this paper proposes **MorphGrower**, which mimicks the neuron natural growth mechanism for generation. Specifically, MorphGrower generates morphologies layer by layer, with each subsequent layer conditioned on the previously generated structure. During each layer generation, MorphGrower utilizes a pair of sibling branches as the basic generation block and generates branch pairs synchronously. This approach ensures topological validity and allows for fine-grained generation, thereby enhancing the realism of the final generated morphologies. Results on four real-world datasets demonstrate that MorphGrower outperforms MorphVAE by a notable margin. Importantly, the electrophysiological response simulation demonstrates the plausibility of our generated samples from a neuroscience perspective. Our code is available at https://github.com/Thinklab-SJTU/MorphGrower.

[*]Equal contribution [1]School of Artificial Intelligence & Department of Computer Science and Engineering & MoE Lab of AI, Shanghai Jiao Tong University [2]Guangdong Institute of Intelligence Science and Technology [3]SEU-ALLEN Joint Center, Institute for Brain and Intelligence, Southeast University [4]XVERSE Technology. Correspondence to: Yimin Wang <ywang@gdiist.cn>, Junchi Yan <yanjunchi@sjtu.edu.cn>.
Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction and Related Works

Neurons are the building blocks of the nervous system and constitute the computational units of the brain (Yuste, 2015). The morphology of neurons plays a central role in brain functioning (Torben-Nielsen & De Schutter, 2014) and has been a standing research area dating back to a century ago (Cajal, 1911). The morphology determines which spatial domain can be reached for a certain neuron, governing the connectivity of the neuronal circuits (Memelli et al., 2013).

A mammalian brain typically has billions of neurons (Kandel et al., 2000), and the computational description and investigation of the brain functioning require a rich number of neuronal morphologies (Lin et al., 2018). However, it can be very expensive to collect quality neuronal morphologies due to its complex collecting procedure with three key steps (Parekh & Ascoli, 2013): *i)* histological preparation, *ii)* microscopic visualization, and *iii)* accurate tracing. Such a procedure is known to be labor-intensive, time-consuming and potentially subject to human bias and error (Schmitz et al., 2011; Choromanska et al., 2012; Yang et al., 2020).

Efforts have been made to generate plausible neuronal morphologies by computational approaches, which are still immature. Traditional neuronal morphology generation methods which mainly aim for generation from scratch, can be classified into two main categories: sampling based (Samsonovich & Ascoli, 2005; da Fontoura Costa & Coelho, 2005; Farhoodi & Kording, 2018) and growth-rule based (Shinbrot, 2006; Krottje & Ooyen, 2007; Stepanyants et al., 2008) methods. Sampling-based methods do not consider the biological processes underlying neural growth (Zubler & Douglas, 2009). They often generate from a simple morphology and iteratively make small changes via e.g. Markov-Chain Monte-Carlo. They suffer from high computation overhead. As for the growth-rule-based methods, they simulate the neuronal growth process according to some preset rules or priors (Koene et al., 2009; Cuntz et al., 2010; Kanari et al., 2022) which can be sensitive to hyperparameters.

Beyond the above traditional methods, until very recently, a learning-based method MorphVAE (Laturnus & Berens,

2021), starts to shed light on this field. Besides its technical distinction to the above non-learning traditional methods, it also advocates a useful generation setting that motivates our work: generating new morphologies by referencing the tree structure of existing real samples, akin to the concept of data augmentation in machine learning. The rationale and value of this 'augmentation' task are that many morphologies' tree structure could be similar to each other due to their same gene expression (Sorensen et al., 2015; Paulsen et al., 2022; Janssen & Budd, 2022), located in the same brain region etc (Oh et al., 2014; Economo et al., 2018; Winnubst et al., 2019). By referencing the tree structure of real morphologies to generate new ones, this approach does not rely on complex handcrafted generation rules, e.g., when to terminate the growth, requiring less human interference.

Specifically, MorphVAE defines the path from soma to a tip (see detailed definitions of soma and tip in Section 2) as a 3D-walk and uses a sequence-to-sequence variational autoencoder to generate those 3D-walks. During generation, the number of 3D-walks to be generated is based on the number of 3D-walks in a sample from the real morphology set, which means generating based on the tree structure of real samples mentioned in the previous paragraph. Then, MorphVAE adopts a post-hoc clustering method on the generated 3D-walks to aggregate some nodes of different 3D-walks, leading to a final generated morphology.

Meanwhile, MorphVAE exhibits certain limitations. Firstly, it chooses the 3D-walk as a basic generation block. However, 3D-walks usually span a wide range in 3D space, which means each 3D-walk is composed of many nodes in most cases. Generating such a long sequence of node 3D coordinates via a seq2seq VAE that decodes nodes autoregressively is difficult due to the accumulated errors and the vanishing gradient (Kong et al., 2023). Secondly, little domain-specific prior knowledge is incorporated into the model design, e.g., the impacts on subsequent branches exerted by previous branches. Thirdly, after the post-hoc clustering, there may exist other nodes that have more than two outgoing edges in the final generated morphology apart from the soma, leading to uncontrolled occurrence of trifurcation or even N-furcation. This contradicts a commonly accepted notion that only the soma node can have more than two child branches (Bray, 1973; Wessells & Nuttall, 1978; Szebenyi et al., 1998). That is to say that morphologies generated by MorphVAE may be invalid in terms of topology (see an example in Appendix C.1).

In contrast to the one-shot generation scheme of MorphVAE, we resort to a progressive way to generate the neuronal morphologies for two reasons: *i)* the growth of a real neuronal morphology from an initial soma is a progressive process that takes a week or more (Budday et al., 2015; Sorrells et al., 2018; Sarnat, 2023). Hence mimicking such dynamics could be of benefit; *ii)* it is much easier to incorporate domain knowledge about neuron growth pattern into generation with a progressive model (Harrison, 1910; Scott & Luo, 2001; Tamariz & Varela-Echavarría, 2015; Shi et al., 2022).

Accordingly, we propose a novel method entitled MorphGrower, which generates high-quality neuronal morphologies at a finer granularity than MorphVAE to augment the available morphology database. Specifically, we adopt a layer-by-layer generation strategy (which in this paper is simplified as layer-wise synchronized though in reality the exact dynamics can be a bit asynchronous) and choose to generate branches in pairs at each layer. In this way, our method can strictly adhere to the rule that only soma can have more than two outgoing branches throughout the generation process, thus ensuring the topological validity of generated morphologies. Moreover, based on the key observation that previous branches have an influence on subsequent branches (Burke et al., 1992; Van Pelt et al., 1997; Cuntz et al., 2007; Purohit & Smith, 2016), we frame each generation step to be conditioned on the intermediate generated morphology. **The highlights of the paper are:**

1) To handle the complexity of morphology (given the authentic morphology as reference), mimicking its biologically growing nature, we devise a layer-by-layer conditional morphology generation scheme called **MorphGrower**. To our best knowledge, this is the first deep model for plausible neuronal morphology generation, particularly in contrast to the peer MorphVAE which generates the whole morphology in one shot and lacks an explicit mechanism to respect the topological validity.

2) Extensive experiments on real-world datasets show that MorphGrower can generate more plausible morphologies than MorphVAE. Particularly, the simulation of electrophysiological responses provides evidence for the plausibility of our generated samples from a neuroscience standpoint. As a side product, it can also generate snapshots of morphologies in different growing stages (see Appendix L.3). We believe the computational recovery of dynamic growth procedure would be an interesting future direction which is currently rarely studied.

2. Preliminaries

A neuronal morphology can be represented as a directed tree (a special graph) $T = (V, E)$. $V = \{v_i\}_{i=1}^{N_v}$ denotes the set of nodes, where $v_i \in \mathbb{R}^3$ are the 3D coordinates of nodes and N_v represents the number of nodes. E denotes the set of directed edges in T. An element $e_{i,j} \in E$ represents a directed edge, which starts from v_i and ends with v_j. The definitions of key terms used in this paper or within the scope of neuronal morphology are presented as follows:

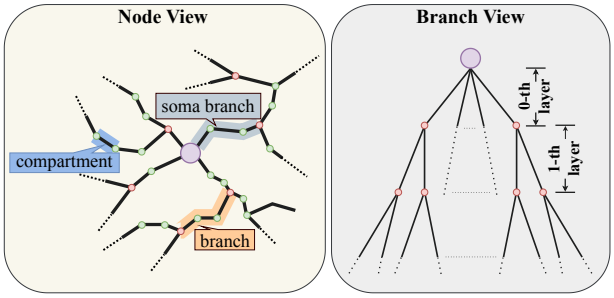


Figure 1: Morphology in node and branch views. The node representing the soma, the nodes signifying bifurcations or tips, and the regular nodes along each branch but not at their ends are highlighted in different colors for distinction.

Definition 2.1 (**Soma & Tip & Bifurcation**). Given a neuronal morphology T, ***soma*** is the root node of T and ***tips*** are those leaf nodes. Soma is the only node that is allowed to have more than two outgoing edges, while tips have no outgoing edge. There is another special kind of nodes called ***bifurcations***. Each bifurcation has two outgoing edges. Soma and bifurcation are two disjoint classes and are collectively referred to as ***multifurcation***.

Definition 2.2 (**Compartment & Branch**). Given a neuronal morphology $T = (V, E)$, each element $e_{i,j} \in E$ is also named as ***compartment***. A ***branch*** is defined as a directed path, e.g. in the form of $e_{i,j} \to e_{j,k} \to \dots e_{p,q} \to e_{q,s}$, which means that this branch starts from v_i and ends at v_s. We can abbreviate such a branch as b_{is}. The beginning node v_i must be a multifurcation (soma or bifurcation) and the ending node must be a bifurcation or a tip. Note that there is no other bifurcation on a branch apart from its two ends. Especially, those branches starting from soma are called ***soma branches*** and constitute the ***soma branch layer***. Besides, we define a pair of branches starting from the same bifurcation as ***sibling branches***.

To facilitate understanding of the above definitions, we provide a node-view topological demonstration of neuronal morphology in the left part of Figure 1. Based on the concept of the branch, as shown in the right part of Figure 1, a neuronal morphology T now can be decomposed into a set of branches and re-represented by a set of branches which are arranged in a Breadth First Search (BFS) like manner, i.e. $T = \{\{b_1^0, b_2^0, \dots, b_{N^0}^0\}, \{b_1^1, b_2^1, \dots, b_{N^1}^1\}, \dots\}$. b_i^k denotes the i-th branch at the k-th layer and N^k is the number of branches at the k-th layer. We specify the soma branch layer as the 0-th layer.

3. Methodology

In this section, we propose MorphGrower to generate high-quality realistic-looking neuronal morphologies. We highlight our model in Figure 2.

3.1. Generation Procedure Formulation

Layer-by-layer Generation Strategy. The number of nodes and branches varies among neuronal morphologies and there exists complex dependency among nodes and edges, making generating T all at once directly challenging. As previously noted, we can represent a morphology via a set of branches and these branches can be divided into different groups according to their corresponding layer number. This implies a feasible solution that we could resort to generating a morphology layer by layer. This layer-by-layer generation strategy is consistent with the natural growth pattern of neurons. In practice, dendrites or axons grow from soma progressively and may diverge several times in the growing process (Harrison, 1910; Scott & Luo, 2001; Tamariz & Varela-Echavarría, 2015; Shi et al., 2022). In contrast, MorphVAE (Laturnus & Berens, 2021) defines the path from soma to tip as a 3D-walk and generates 3D-walks one by one. This strategy fails to consider bifurcations along the 3D-walks. Thus it violates this natural pattern.

Following such a layer-by-layer strategy, a new morphology can be obtained by generating new layers and merging them to intermediate generated morphology regressively. In the following, we further discuss the details of how to generate a certain layer next.

Generating Branches in Pairs. Since the leaf nodes of an intermediate morphology are either bifurcations or tips, branches grow in pairs at each layer except the soma branch layer, implying that N^i is even for $i \geq 1$. As pointed out in previous works (Uylings & Smit, 1975; Kim et al., 2012; Bird & Cuntz, 2016; Otopalik et al., 2017), there exists a complex dependency between sibling branches. If we separate sibling branches from each other and generate each of them individually, this dependency will be hard to model. A natural idea is that one can regard sibling branches as a whole and generate sibling branches in pairs each time, to implicitly model their internal dependency. Following (Laturnus & Berens, 2021), we adopt a seq2seq variational autoencoder (VAE) for generation. Different from the work by (Laturnus & Berens, 2021), which uses a VAE to generate 3D-walks with greater lengths in 3D space than branches, increasing the difficulty of generation, our branch-pair-based method can generate high-quality neuronal morphologies more easily.

Conditional Generation. In addition, a body of literature on neuronal morphology (Burke et al., 1992; Van Pelt et al., 1997; Cuntz et al., 2007; Purohit & Smith, 2016) has consistently shown such an observation in neuronal growth: grown branches could influence their subsequent branches. Hence, taking the structural information of the first i layers into consideration when generating branches at i-th layer is more reasonable and benefits the generation.

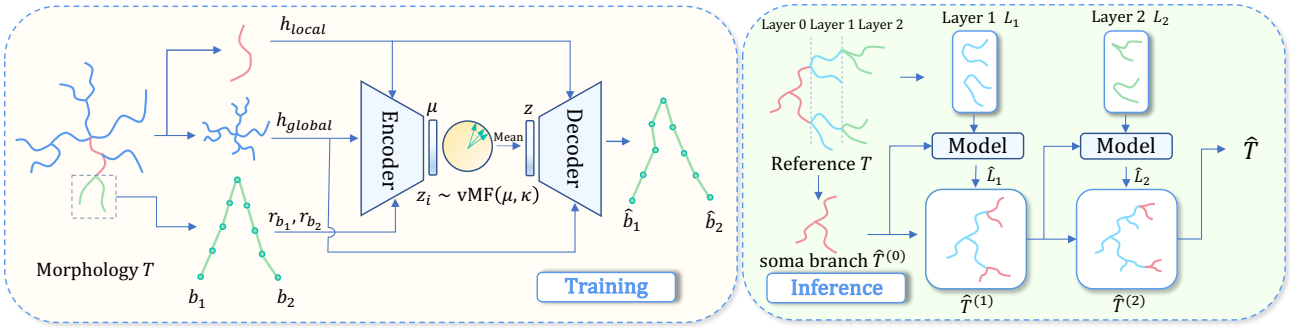


Figure 2: **Overview of MorphGrower.** It takes the branch pair and its previous layers as inputs. The branch pairs as well as the previous layers as conditions determine the mean direction μ of latent space which follows a von-Mises Fisher distribution with fixed variance κ. Latent variables are then sampled from the distribution $\mathbf{z}_i \sim \text{vMF}(\mu, \kappa)$. Finally, the decoder reconstructs the branch pairs from the latent variable $\mathbf{z}$ and the given condition. In inference, the model is called regressively, taking the generated subtree $\hat{T}^{(i)}$ and the $(i+1)$-th layer L_{i+1} of the reference morphology as input and outputting a new layer $\hat{L}_{i+1}$ of the final generated morphology.

To incorporate the structural information of the previous layers into the generation of their subsequent layers, in this paper, we propose to encode the intermediate morphology which has been generated into an embedding and restrict the generation of branch pairs in the following layer to be conditioned on this embedding we obtain. Considering the conditional generation setting, we turn to use a seq2seq conditional variational autoencoder (CVAE) (Sohn et al., 2015) instead. We next present how we encode the structural information of the previous layers in detail.

We split the conditions extracted from the structural information of previous layers into two parts and name them **local** and **global** conditions respectively. Assuming that we are generating the pair highlighted in Figure 2, we define the path from soma to the bifurcation from which the pair to be generated starts as the **local condition** and its previous layers structure as the **global condition**. We provide justifications for the significance of both local and global conditions from a neuronscience perspective as follows and the details of encoding conditions are presented in Section 3.2.

Justifications for the Conditions. ***Local:*** Previous studies (Samsonovich & Ascoli, 2003; López-Cruz et al., 2011) show that the dendrites or axons usually extend away from the soma without making any sharp change of direction, thus reflecting that the orientation of a pair of sibling branches is mainly determined by the overall orientation of the path from the soma to the start point of the siblings. ***Global:*** Dendrites/axons establish territory coverage by following the organizing principle of self-avoidance (Sweeney et al., 2002; Sdrulla & Linden, 2006; Matthews et al., 2007). Self-avoidance refers to dendrites/axons that should avoid crossing, thus spreading evenly over a territory (Kramer & Kuwada, 1983). Since the global condition can be regarded as a set of the local conditions and each local condition can roughly decide the orientation of a corresponding pair of branches, the global condition helps us better organize the branches in the same layer and achieve an even spread.

The Distinction of the Soma Branch Layer. Under the aforementioned methodology formulation, we can observe that the soma branch layer differs from other layers in two folds. Firstly, 2 may not be a divisor of the branch number of the soma branch layer (i.e. $N^0 \mod 2 = 0$ may not hold). Secondly, the soma branch layer cannot be unified to the conditional generation formulation due to that there is no proper definition of conditions for it. Its distinction requires specific treatment.

In some other generation task settings (Liu et al., 2018; Liao et al., 2019), a little prior knowledge about the reference is introduced to the model as hints to enhance the realism of generations. Thus a straightforward solution is to adopt a similar approach: we directly present the soma branches as conditional input to the model, which are fairly small in number compared to all the branches[1]. Another slightly more complex approach is to generate the soma branch layer using another VAE without conditions. In the experimental section, we demonstrate the results of both having the soma branch layer directly provided and not provided.

Now we have given a description of our generation procedure and will present the instantiations next.

3.2. Model Instantiation

We use CVAE to model the distribution over branch pairs conditioned on the structure of their previous layers. The encoder encodes the branch pair as well as the condition

[1] For example, soma branches only account for approximately two percent of the total number in RGC dataset.

to obtain a latent variable $\mathbf{z} \in \mathbb{R}^d$, where $d \in \mathbb{N}^+$ denotes the embedding size. The decoder generates the branch pairs from the latent variable $\mathbf{z}$ under the encoded conditions. Our goal is to obtain an encoder $f_\theta(\mathbf{z}|b_1, b_2, C)$ for a branch pair (b_1, b_2) and condition C , and a decoder $g_\phi(b_1, b_2|\mathbf{z}, C)$ such that $g_\phi(f_\theta(b_1, b_2, C), C) \approx b_1, b_2$. The encoder and decoder are parameterized by θ and ϕ respectively.

Encoding Single Branch. A branch b_i can also be represented by a sequence of 3D coordinates, denoted as $b_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{L(i)}}\}$, where $L(i)$ denotes the number of nodes included in b_i and every pair of adjacent coordinates $(v_{i_k}, v_{i_{k+1}})$ is connected via an edge $e_{i_k,i_{k+1}}$. Notice that the length and the number of compartments constructing a branch vary from branch to branch, so as the start point v_{i_1}, making it difficult to encode the morphology information of a branch thereby. Thus we first translate the branch to make it start from the point $(0, 0, 0)$ in 3D space and then perform a resampling algorithm before encoding, which rebuilds the branches with L new node coordinates and all compartments on a branch share the same length after the resampling operation. We present the details of the resampling algorithm in Appendix A. The branch b_i after resampling can be written as $b'_i = \{v'_{i_1}, v'_{i_2}, \ldots, v'_{i_L}\}$. In the following text, all branches have been obtained after the resampling process. In principle, they should be denoted with an apostrophe. However, to maintain a more concise notation, we omit the use of the apostrophe subsequently.

Following MorphVAE (Laturnus & Berens, 2021), we use Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) to encode the branch morphological information since a branch can be represented by a sequence of 3D coordinates. Each input coordinate will first be embedded into a high-dimension space via a linear layer, i.e. $\mathbf{x}_i = \mathbf{W}_{in} \cdot v'_i$ and $\mathbf{W}_{in} \in \mathbb{R}^{d\times 3}$. The obtained sequence of $\mathbf{x}_i$ is then fed into an LSTM. The internal hidden state $\mathbf{h}_i$ and the cell state $\mathbf{c}_i$ of the LSTM network for $i \geq 1$ are updated by:

$$\mathbf{h}_i, \mathbf{c}_i = \mathrm{LSTM}(\mathbf{x}_i, \mathbf{h}_{i-1}, \mathbf{c}_{i-1}). \tag{1}$$

The initial two states $\mathbf{h}_0$ and $\mathbf{c}_0$ are both set to zero vectors. Finally, for a input branch b, we can obtain the corresponding representation $\mathbf{r}_b$ by concatenating $\mathbf{h}_L$ and $\mathbf{c}_L$, i.e.,

$$\mathbf{r}_b = \mathrm{CONCAT}[\mathbf{h}_L, \mathbf{c}_L]. \tag{2}$$

Encoding Global Condition. The global condition for a branch pair to be generated is obtained from the whole morphological structure of its previous layers. Note that the previous layers – exactly a subtree of the whole morphology, form an extremely sparse graph. Most nodes on the tree have no more than $3k$ k-hop[2] neighbors, thereby limiting the receptive field of nodes. Furthermore, as morphology is a hierarchical tree-like structure, vanilla Graph Neural Networks (Kipf & Welling, 2017) have difficulty in modeling such sparse hierarchical trees and encoding the global condition.

To tackle the challenging problem, instead of treating each coordinate as a node, we regard a branch as a node. Then the re-defined nodes will be connected by a directed edge e_{b_i,b_j} starting from b_i if b_j is the succession of b_i. We can find the original subtree now is transformed into a forest $\mathcal{F} = (V^{\mathcal{F}}, E^{\mathcal{F}})$ composed of N^0 trees, where N^0 is the number of soma branches. $V^{\mathcal{F}}$ and $E^{\mathcal{F}}$ denote the set of nodes and edges in $\mathcal{F}$ respectively. The root of each included tree corresponds to a soma branch and each tree is denser than the tree made up of coordinate nodes before. Meanwhile, the feature of each re-defined node, extracted from a branch rather than a single coordinate, is far more informative than those of the original nodes, making the message passing among nodes more efficient.

Inspired by Tree structure-aware Graph Neural Network (T-GNN) (Qiao et al., 2020), we use a GNN combined with the Gated Recurrent Unit (GRU) (Chung et al., 2014) to integrate the hierarchical and sequential neighborhood information on the tree structure to node representations. Similar to T-GNN, We also perform a bottom-up message passing within the tree until we update the feature of the root node. For a branch b_i located at depth k in the tree[3], its corresponding node feature $\mathbf{h}_{b_i}^{(k)}$ is given by:

$$\mathbf{h}_{b_i}^{(k)} = \begin{cases} \mathrm{GRU}(\mathbf{r}_{b_i}, \sum_{b_j \in \mathcal{N}^+(b_i)} \mathbf{W}\mathbf{h}_{b_j}^{(k+1)}), & \mathcal{N}^+(b_i) \neq \emptyset \\ r_{b_i}, & \mathcal{N}^+(b_i) = \emptyset \end{cases}, \tag{3}$$

where $\mathbf{W}$ is a linear transformation, GRU is shared between different layers, and $\mathcal{N}^+(b_i)$ is the neighbors of b_i who lies in deeper layer than b_i, which are exactly two subsequent branches of b_i. $\mathbf{r}_{bi}$ is obtained by encoding the branch using the aforementioned LSTM. The global feature is obtained by aggregating the features of root nodes be all trees in the forest $\mathcal{F}$:

$$\mathbf{h}_{global} = \mathrm{READOUT}\left\{\mathbf{h}_{b_i}^{(0)} | b_i \in R^{\mathcal{F}}\right\}, \tag{4}$$

where $R^{\mathcal{F}}$ denotes the roots of trees in $\mathcal{F}$ as and mean-pooling is used as the READOUT function.

Encoding Local Condition. For a branch at the l-th layer, we denote the sequence of its ancestor branches as $\mathcal{A} = \{a_0, a_1, \ldots, a_{l-1}\}$ sorted by depth in ascending order where a_0 is a soma branch. As mentioned in Section 3.1, the growth of a branch is influenced by its ancestor branches. The closer ancestor branches might exert more influence on it. Thus we use Exponential Moving Average

[2] Here,"k-hop neighbors of a node v" refers to all neighbors within a distance of k from node v.

[3] Here, "tree" refers to the trees in $\mathcal{F}$ and not to the tree structure of the morphology.

(EMA) (Lawrance & Lewis, 1977) to calculate the local feature. Denoting the feature aggregated from the first k elements of $\mathcal{A}$ as $\mathbf{D}_k^{\mathcal{A}}$ and $\mathbf{D}_0^{\mathcal{A}} = \mathbf{r}_{a_0}$. For $k \geq 1$,

$$\mathbf{D}_k^{\mathcal{A}} = \alpha \mathbf{r}_{a_k} + (1-\alpha)\mathbf{D}_{k-1}^{\mathcal{A}}, \quad 0 \leq \alpha \leq 1, \tag{5}$$

where α is a hyper-parameter. The local condition $\mathbf{h}_{local}$ for a branch at the l-th layer is $\mathbf{D}_{l-1}^{\mathcal{A}}$, i.e.,

$$\mathbf{h}_{local} = \mathbf{D}_{l-1}^{\mathcal{A}}. \tag{6}$$

The global condition ensures consistency within the layer, while the local condition varies among different pairs, only depending on their respective ancestor branches. Therefore, the generation order of branch pairs within the same layer does not have any impact, enabling synchronous generation.

Design of Encoder and Decoder. The encoder models a von-Mises Fisher (vMF) distribution (Xu & Durrett, 2018) with fixed variance κ and mean μ obtained by aggregating branch information and conditions. The encoder takes a branch pair (b_1, b_2) and the corresponding global condition $\mathcal{F}$, local condition $\mathcal{A}$ as input. All these input are encoded into features $\mathbf{r}_{b_1}$ $\mathbf{r}_{b_2}$, $\mathbf{h}_{global}$ and $\mathbf{h}_{local}$ respectively by the aforementioned procedure. The branches after resampling are denoted as $b_1' = \{v_1^{(1)}, v_2^{(1)}, \ldots, v_L^{(1)}\}$ and $b_2' = \{v_1^{(2)}, v_2^{(2)}, \ldots, v_L^{(2)}\}$. Then all the features are concatenated and fed to a linear layer $\mathbf{W}_{lat}$ to obtain the mean $\mu = \mathbf{W}_{lat} \cdot \mathrm{CONCAT}[\mathbf{r}_{b_1}, \mathbf{r}_{b_2}, \mathbf{h}_{global}, \mathbf{h}_{local}]$. We take the average of five samples from $\mathbf{z}_i \sim \mathrm{vMF}(\mu, \kappa)$ via rejection sampling as the latent variable $\mathbf{z}$.

As for the decoder $g_\phi(b_1, b_2 | \mathbf{z}, C = (\mathcal{A}, \mathcal{F}))$, we use another LSTM to generate two branches node by node from the latent variable $\mathbf{z}$ and condition representations, i.e. $\mathbf{h}_{global}$ and $\mathbf{h}_{local}$. First, we concatenate latent variable z as well as the condition features $\mathbf{h}_{global}$ and $\mathbf{h}_{local}$ together and feed it to two different linear layers $\mathbf{W}_{sta1}, \mathbf{W}_{sta2}$ to determine the initial internal state for decoding branch $\hat{b}_1$ and $\hat{b}_2$. After that, a linear projection $\mathbf{W}_{in}' \in \mathbb{R}^{3 \times d}$ is used to project the first coordinates $v_1^{(1)}$ and $v_1^{(2)}$ of each branch into a high-dimension space $\mathbf{y}_1^{(1)} = \mathbf{W}_{in}' \cdot v_1^{(1)}, \mathbf{y}_1^{(2)} = \mathbf{W}_{in}' \cdot v_1^{(2)}$. Then the LSTM predicts $\mathbf{y}_{i+1}^{(t)}$ from $\mathbf{y}_i^{(t)}$ and its internal states repeatedly, where $t \in \{1, 2\}$. Finally, another linear transformation $\mathbf{W}_{out} \in \mathbb{R}^{3 \times d}$ is used to transform each $\mathbf{y}_i^{(t)}$ into coordinate $\hat{v}_i^{(t)} = \mathbf{W}_{out} \cdot \mathbf{y}_i^{(t)}, t \in \{1, 2\}$ and we can get the generated branch pair $\hat{b}_1 = \{\hat{v}_1^{(1)}, \hat{v}_2^{(1)}, \ldots, \hat{v}_L^{(1)}\}$ and $\hat{b}_2 = \{\hat{v}_1^{(2)}, \hat{v}_2^{(2)}, \ldots, \hat{v}_L^{(2)}\}$. In summary:

$$\begin{aligned}
\mu &= \mathbf{W}_{lat} \cdot \mathrm{CONCAT}[\mathbf{r}_{b_1}, \mathbf{r}_{b_2}, \mathbf{h}_{global}, \mathbf{h}_{local}], \\
\mathbf{z}_i &\sim \mathrm{vMF}(\mu, \kappa), \\
\mathbf{z} &= \frac{1}{5}\sum_{i=1}^{5} \mathbf{z}_i, \\
\mathbf{h}_0'^{(1)}, \mathbf{c}_0'^{(1)} &= \mathbf{W}_{sta1} \cdot \mathrm{CONCAT}[\mathbf{z}, \mathbf{h}_{global}, \mathbf{h}_{local}], \\
\mathbf{h}_0'^{(2)}, \mathbf{c}_0'^{(2)} &= \mathbf{W}_{sta2} \cdot \mathrm{CONCAT}[\mathbf{z}, \mathbf{h}_{global}, \mathbf{h}_{local}], \\
\mathbf{y}_1^{(t)} &= \mathbf{W}_{in}' \cdot v_1^{(t)}, \quad t \in \{1, 2\}, \\
\mathbf{h}_{i+1}'^{(t)}, \mathbf{c}_{i+1}'^{(t)} &= \mathrm{LSTM}(\mathbf{y}_i, \mathbf{h}_i'^{(t)}, \mathbf{c}_i'^{(t)}), \quad t \in \{1, 2\}, \\
\hat{v}_i^{(t)} &= \mathbf{W}_{out} \cdot \mathbf{y}_i^{(t)}, \quad t \in \{1, 2\}, \\
\hat{b}_t &= \{\hat{v}_1^{(t)}, \hat{v}_2^{(t)}, \ldots, \hat{v}_L^{(t)}\}, \quad t \in \{1, 2\}.
\end{aligned} \tag{7}$$

We jointly optimize the parameters (θ, ϕ) of the encoder and decoder by maximizing the *Evidence Lower BOund* (ELBO), which is composed of a KL term and a reconstruction term:

$$\begin{aligned}
&\mathcal{L}(B = (b_1, b_2), C = (\mathcal{A}, \mathcal{F}); \theta, \phi) \\
&= \mathbb{E}_{f_\theta(\mathbf{z}|B,C)}[\log g_\phi(B|\mathbf{z}, C)] - \mathrm{KL}(f_\theta(\mathbf{z}|B, C) \| g_\phi(\mathbf{z}|C)),
\end{aligned} \tag{8}$$

where we abbreviate (b_1, b_2) as B. We use a uniform vMF distribution $g_\phi(\mathbf{z}|C) = \mathrm{vMF}(\cdot, 0)$ as the prior. According to the property of vMF distribution, the KL term will become a constant depending on only the choice of κ. Then the loss function in Eq. 8 is reduced to the reconstruction term and can be rewritten as follows, estimated by the sum of mean-squared error between (b_1, b_2) and $(\hat{b}_1, \hat{b}_2)$,

$$\mathcal{L}(B, C; \theta, \phi) = \mathbb{E}_{f_\theta(\mathbf{z}|B,C)}[\log g_\phi(B|\mathbf{z}, C)]. \tag{9}$$

3.3. Sampling New Morphologies

A new morphology can be generated by selecting a morphology T from the real sample set and calling our model regressively based on the tree structure of T. In the generation of the i-th layer, the reference branch pairs on the i-th layer on T and the generated previous $i-1$ layers will be taken as the model input. We use $L_i = \{b_1^i, b_2^i, \ldots, b_{N^i}^i\}$ and $\hat{L}_i = \{\hat{b}_1^i, \hat{b}_2^i, \ldots, \hat{b}_{N^i}^i\}$ to represent the set of reference branches and those branches we generate at the i-th layer respectively. As mentioned in Section 3.1, we choose to start the generation on the basis of the given soma branch layer of T here. In general, the generation process can be formulated as:

$$\begin{aligned}
\hat{T}^{(0)} &= \{L_0\}, \quad \hat{T}^{(i)} = \hat{T}^{(i-1)} \cup \{\hat{L}_i\}, \\
\hat{L}_i &= g_\phi(f_\theta(L_i, \hat{T}^{(i-1)}), \hat{T}^{(i-1)}),
\end{aligned} \tag{10}$$

where $\hat{T}^{(i)}$ is the intermediate morphology after generating the i-th layer and $\hat{L}_i$ is the collection of generated branches at the i-th layer of the new morphology.

Table 1: Performance on the four datasets by the six quantitative metrics. We leave MorphVAE's numbers on MBPL and MAPS blank because it may generate nodes with more than two subsequent branches that conflict with the definition of MBPL and MAPS for bifurcations. **MorphGrower** denotes the version where soma branches are directly provided. Meanwhile, **MorphGrower**† generates soma branches using another unconditional VAE. *Reference* corresponds to the statistical indicators derived from the realistic samples. A closer alignment with *Reference* indicates better performance. The best and the runner-up results are highlighted in **bold** and underline respectively.

Dataset	Method	MBPL / μm	MMED / μm	MMPD / μm	MCTT / %	MASB / °	MAPS / °
VPM	*Reference*	51.33 ± 0.59	162.99 ± 2.25	189.46 ± 3.81	0.936 ± 0.001	65.35 ± 0.55	36.04 ± 0.38
	MorphVAE	41.87 ± 0.66	126.73 ± 2.54	132.50 ± 2.61	0.987 ± 0.001	———	———
	MorphGrower	**48.29 ± 0.34**	**161.65 ± 1.68**	**180.53 ± 2.70**	0.920 ± 0.004	72.71 ± 1.50	**43.80 ± 0.98**
	MorphGrower†	46.86 ± 0.57	159.62 ± 3.19	179.44 ± 5.23	**0.929 ± 0.006**	**59.15 ± 3.25**	46.59 ± 3.24
RGC	*Reference*	26.52 ± 0.75	308.85 ± 8.12	404.73 ± 12.05	0.937 ± 0.003	84.08 ± 0.28	50.60 ± 0.13
	MorphVAE	43.23 ± 1.06	248.62 ± 9.05	269.92 ± 10.25	0.984 ± 0.004	———	———
	MorphGrower	**25.15 ± 0.71**	**306.83 ± 7.76**	**384.34 ± 11.85**	**0.945 ± 0.003**	**82.68 ± 0.53**	**51.33 ± 0.31**
	MorphGrower†	23.32 ± 0.52	287.09 ± 5.88	358.31 ± 8.54	0.926 ± 0.004	76.27 ± 0.86	49.67 ± 0.41
M1-EXC	*Reference*	62.74 ± 1.73	414.39 ± 6.16	497.43 ± 12.42	0.891 ± 0.004	76.34 ± 0.63	46.74 ± 0.85
	MorphVAE	52.13 ± 1.30	195.49 ± 9.91	220.72 ± 12.96	0.955 ± 0.005	———	———
	MorphGrower	**58.16 ± 1.26**	**413.78 ± 14.73**	**473.25 ± 19.37**	0.922 ± 0.002	**73.12 ± 2.17**	**48.16 ± 1.00**
	MorphGrower†	54.63 ± 1.07	398.85 ± 18.84	463.24 ± 22.61	**0.908 ± 0.003**	63.54 ± 2.02	48.77 ± 0.87
M1-INH	*Reference*	45.03 ± 1.04	396.73 ± 15.89	705.28 ± 34.02	0.877 ± 0.002	84.40 ± 0.68	55.23 ± 0.78
	MorphVAE	50.79 ± 1.77	244.49 ± 15.62	306.99 ± 23.19	0.965 ± 0.002	———	———
	MorphGrower	**41.50 ± 1.02**	**389.06 ± 13.54**	**659.38 ± 30.05**	0.898 ± 0.002	**82.43 ± 1.41**	61.44 ± 4.23
	MorphGrower†	37.72 ± 0.96	349.66 ± 11.40	617.89 ± 27.87	**0.876 ± 0.002**	78.66 ± 1.12	**57.87 ± 0.96**

4. Experiments

In this section, we conduct extensive experiments to comprehensively evaluate the performance of our proposed MorphGrower. Due to the limited space, we place some additional results, such as the ablation study, in Appendix J.

4.1. Evaluation Protocols

Datasets. We use four popular datasets all sampled from adult mice: 1) **VPM** is a set of ventral posteromedial nucleus neuronal morphologies (Landisman & Connors, 2007; Peng et al., 2021); 2) **RGC** is a morphology dataset of retinal ganglion cell dendrites (Reinhard et al., 2019); 3) **M1-EXC**; and 4) **M1-INH** refer to the excitatory pyramidal cell dendrites and inhibitory cell axons in M1. We split each dataset into training/validation/test sets by 8:1:1. See Appendix I for more dataset details.

Baseline. To our knowledge, there is no learning-based morphology generator except MorphVAE (Laturnus & Berens, 2021) which pioneers in generating neuronal morphologies resembling the given references. Grid search of learning rate over $\{1e-2, 5e-3, 1e-3\}$ and dropout rate over $\{0.1, 0.3, 0.5\}$ is performed to select the optimal hyperparameters for MorphVAE. For fairness, both MorphVAE and ours use a 64 embedding size. Refer to Appendix H for more configuration details.

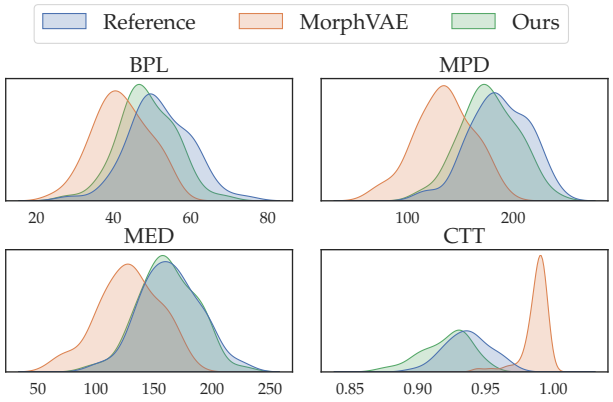


Figure 3: Distributions of four morphological metrics: MPD, BPL, MED and CTT on VPM dataset.

4.2. Quantitative Results on Morphological Statistics

Quantitative Metrics. Rather than the Inception Score (IS) (Salimans et al., 2016) and Fréchet Inception Distance (FID) (Heusel et al., 2017) used in vision, there are tailored metrics as widely-used in computational neuronal morphology generation software tools. These metrics (as defined in L-measure (Scorcioni et al., 2008)) include: **BPL**, **MED**, **MPD**, **CTT**, **ASB**, **APS** which in fact are computed for a single instance. We further compute their mean over the dataset as **MBPL**, **MMED**, **MMPD**, **MCTT**, **MASB** and

Table 2: Classification accuracy (%). Accuracy approaching 50% indicates higher plausibility.

Method \ Dataset	VPM	RGC	M1-EXC	M1-INH
MorphVAE	86.75 ± 06.87	94.60 ± 01.02	80.72 ± 10.58	91.76 ± 12.14
MorphGrower	54.73 ± 03.36	62.70 ± 05.24	55.00 ± 02.54	54.74 ± 01.63

Table 3: Simulated recording statistical characteristics.

Metric	MMF / Hz	MMAPH / mV	MMAHPD / mV	MMAPA / mV
Reference	9.328	43.021	25.666	-77.231
MorphGrower	9.384	44.382	26.606	-77.837
Relative Error	0.60%	3.17%	3.66%	0.78%

MAPS, and use these six metrics as our final metrics. The first four metrics relate to branch length and shape and the other two measure the amplitudes between branches. Detailed metric definitions are presented in Appendix B.

As shown in Table 1, we can observe that the performance of our approach does not achieve a significant boost from re-using the soma branches as initial structures simply for convenience. Even without providing soma branches, MorphGrower still outperforms MorphVAE across all the datasets, except for the MBPL metric on the M1-INH dataset, suggesting the effectiveness of our progressive generation approach compared to the one-shot one in MorphVAE, especially for the four metrics related to branch length and shape. Due to the fact that MorphVAE uses 3D-walks as its basic generation units at a coarser granularity and resamples them, it loses many details of individual branches. In contrast, our MorphGrower uses branch pairs as its basic units at a finer granularity and performs branch-level resampling. A 3D-walk often consists of many branches, and with the same number of sampling points, our method can preserve more of the original morphology details. This is why MorphGrower performs better in the first four metrics. In addition, regarding MASB and MAPS that measure the amplitudes between branches and thus can reflect the orientations and layout of branches, the generation by MorphGrower is close to the reference data. We believe that this is owed to our incorporation of the neuronal growth principle, namely previous branches help in orientating and better organizing the subsequent branches. To further illustrate the performance gain, we plot the distributions of MPD, BPL, MED and CTT over the reference samples and generated morphologies on VPM in Figure 3 (see the other datasets in Appendix L.2).

4.3. Generation Plausibility with Real/Fake Classifier

The above six human-defined metrics can reflect the generation quality to some extent, while we seek a more data-driven way, especially considering the inherent complexity of the morphology data. Inspired by the scheme of generative adversarial networks (GAN) (Goodfellow et al., 2014), we here propose to train a binary classifier with the ratio $1:1$ for using the real-world morphologies as positive and generated ones as negative samples (one classifier for one generation method respectively). Then this classifier serves a role similar to the discriminator in GAN to evaluate the generation plausibility.

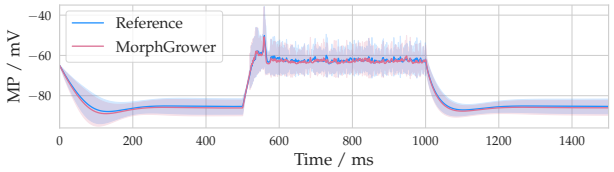


Figure 4: Comparison of averaged electrophysiological recordings on real and generated morphology samples. The term MP on the vertical axis stands for Membrane Potential.

Specifically, features are prepared as input to the classifier. Given a 3-D neuron, we project its morphology onto the xy, xz and yz planes and obtain three corresponding projections. Inspired by the works in vision on multi-view representation (Prakash et al., 2021; Shao et al., 2022), we feed these three multi-view projections to three CNN blocks and obtain three corresponding representations. Here each CNN block adopts ResNet18 (He et al., 2016) as the backbone. Then, we feed each representation to a linear layer and obtain a final representation for each projection. Finally, the representations are concatenated and fed to the classifier, which is implemented with a linear layer followed by a sigmoid function. The whole pipeline is shown in Appendix H.3.

Table 2 reports the classification accuracy on the test set by the same data split as in Section 4.2. It is much more difficult for the neuronal morphologies generated by MorphGrower to be differentiated by the trained classifier, suggesting its plausibility. Moreover, we present the generated neuronal morphologies to neuroscience domain experts and receive positive feedback for their realistic looking.

4.4. Simulation Validation

The generation of neuronal morphology primarily aims to produce a wealth of data for extensive brain science analysis, particularly critical for simulating phenomena like neuronal population discharges and neural network signal propagation in neurosystem dynamics. Thus, it's vital to ensure that the generated neuronal morphologies exhibit electrophysiological responses akin to real samples, as this is a key requirement for their application in broader neuroscience research. Given that subtle morphological variations can significantly influence electrophysiological response, to more comprehensively assess the plausibility of the generated neurons from a neuroscience perspective and to demonstrate their value to neuroscientists, we opt to conduct electro-

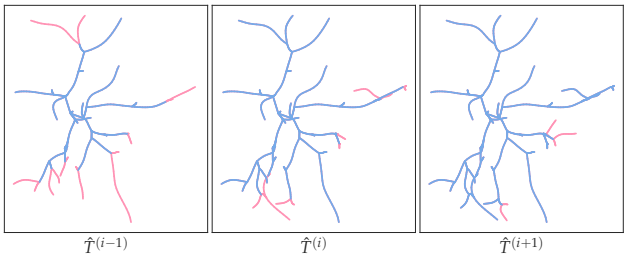


Figure 5: Projections onto xy plane of three adjacent intermediate morphologies of a generated sample from RGC. For each $\hat{T}^{(j)}$, the newly generated layer $\hat{L}_j$ and the intermediate morphology generated after the last step $\hat{T}^{(j-1)}$ are highlighted in pink and blue, respectively.

physiological response simulation as an additional avenue for evaluation. For those interested in the details of these experiments, we have included them in Appendix H.4.

Through electrophysiological responses simulation, we can obtain two sets of electrophysiological recordings, corresponding respectively to real and generated morphologies. Since simulation experiments are time-consuming, we only conduct simulations on the VPM dataset. We average the recordings simulated from both real and generated morphology samples separately, and the two averaged recordings are displayed in Figure 4. We can observe that the curves of the real and generated samples closely align, indicating that their electrophysiological responses exhibit a high degree of similarity. Moreover, we calculate some characteristics of the recordings for both real and generated samples. Table 3 presents the results for four metrics: **MMF**, **MMAPH**, **MMAHPD**, and **MMAPA**, with their definitions provided in Appendix B.3. It shows that the statistical characteristics of these four metrics on our generated samples closely align with those on the real samples. These above findings effectively demonstrate the plausibility of the morphologies generated by our method from a neuroscience perspective.

4.5. Snapshots of the Growing Morphologies

We present three adjacent snapshots of the intermediate generated morphologies of a randomly picked sample from the RGC in Fig. 5. More examples are given in Appendix L.3. We believe such studies are informative for the field, though have been rarely performed before.

5. Conclusion and Outlook

To achieve plausible neuronal morphology generation, we propose a biologically inspired neuronal morphology growing approach based on the given morphology as a reference. During the inference stage, the morphology is generated layer by layer in an auto-regressive manner whereby the domain knowledge about the growing constraints can be effectively introduced. Quantitative and qualitative results show the efficacy of our approach against the state-of-the-art baseline MorphVAE.

Our aspiration is that the fruits of our research will significantly augment the efficiency with which morphological samples of neurons are procured. This enhancement may pave the way for the generation of biologically realistic networks encompassing entire brain regions (Kanari et al., 2022), fostering a deeper understanding of the complex mechanisms that underpin neural computation. A more comprehensive grasp of neural computation could unveil fundamental ingredients of intelligence, potentially acting as a catalyst for future advancements in the realm of artificial intelligence. This could eventually make it possible for the development of artificial agents whose capabilities match those of humans (Zador et al., 2023).

Limitations. There are two aspects limiting our approach, both in terms of data availability. First, we only have morphology information regarding the coordinates of the nodes, hence the generation may inherently suffer from the illness of missing information (e.g. the diameter). Second, the real-world samples for learning do not reflect the dynamic growing procedure and thus learning such dynamics can be challenging. In our paper, we have simplified the procedure by imposing synchronized layer growth which in reality could be asynchronous to some extent with growing randomness. In both cases, certain prior knowledge need to be introduced as preliminarily done in this paper which could be further improved for future work.

Acknowledgments

This work was in part supported by the Guangdong High Level Innovation Research Institute (2021B0909050004), the Key-Area Research and Development Program of Guangdong Province (2021B0909060002), the National Natural Science Foundation of China (62222607, 32071367), and the Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

Impact Statement

This paper introduces a novel approach for generating plausible neuronal morphologies, with the goal of streamlining the acquisition of neuron morphology data crucial for advancements in neuroscience. The proposed method holds promise for providing the data infrastructure essential for simulating neural circuits. Our research primarily emphasizes the generation of foundational data pivotal for brain science investigations. It's worth noting that our work, at present, doesn't directly address broader societal issues, and no adverse social implications have been attributed to our work thus far.

References

Adams, P., Brown, D., and Constanti, A. M-currents and other potassium currents in bullfrog sympathetic neurones. *The Journal of Physiology*, 330(1):537–572, 1982.

Avery, R. B. and Johnston, D. Multiple channel types contribute to the low-voltage-activated calcium current in hippocampal ca3 pyramidal neurons. *Journal of Neuroscience*, 16(18):5567–5582, 1996.

Beauquis, J., Pavía, P., Pomilio, C., Vinuesa, A., Podlutskaya, N., Galvan, V., and Saravia, F. Environmental enrichment prevents astroglial pathological changes in the hippocampus of app transgenic mice, model of alzheimer's disease. *Experimental Neurology*, 239:28–37, 2013.

Bird, A. D. and Cuntz, H. Optimal current transfer in dendrites. *PLoS computational biology*, 12(5):e1004897, 2016.

Bray, D. Branching patterns of individual sympathetic neurons in culture. *The Journal of Cell Biology*, 56(3):702–712, 1973.

Budday, S., Steinmann, P., and Kuhl, E. Physical biology of human brain development. *Frontiers in cellular neuroscience*, 9:257, 2015.

Burke, R., Marks, W., and Ulfhake, B. A parsimonious description of motoneuron dendritic morphology using computer simulation. *Journal of Neuroscience*, 12(6): 2403–2416, 1992.

Cajal, R. S. *Histologie du système nerveux de l'homme & des vertébrés: Cervelet, cerveau moyen, rétine, couche optique, corps strié, écorce cérébrale générale & régionale, grand sympathique*, volume 2. A. Maloine, 1911.

Choromanska, A., Chang, S.-F., and Yuste, R. Automatic reconstruction of neural morphologies with multi-scale tracking. *Frontiers in neural circuits*, 6:25, 2012.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Colbert, C. M. and Pan, E. Ion channel properties underlying axonal action potential initiation in pyramidal neurons. *Nature neuroscience*, 5(6):533–538, 2002.

Conde-Sousa, E., Szücs, P., Peng, H., and Aguiar, P. N3dfix: an algorithm for automatic removal of swelling artifacts in neuronal reconstructions. *Neuroinformatics*, 15:5–12, 2017.

Cuntz, H., Borst, A., and Segev, I. Optimization principles of dendritic structure. *Theoretical Biology and Medical Modelling*, 4(1):1–8, 2007.

Cuntz, H., Forstner, F., Borst, A., and Häusser, M. One rule to grow them all: a general theory of neuronal branching and its practical application. *PLoS computational biology*, 6(8):e1000877, 2010.

da Fontoura Costa, L. and Coelho, R. C. Growth-driven percolations: the dynamics of connectivity in neuronal systems. *The European Physical Journal B-Condensed Matter and Complex Systems*, 47(4):571–581, 2005.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pp. 856–865. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

Destexhe, A., Mainen, Z. F., and Sejnowski, T. J. Synthesis of models for excitable membranes, synaptic transmission and neuromodulation using a common kinetic formalism. *Journal of computational neuroscience*, 1:195–230, 1994.

Economo, M. N., Viswanathan, S., Tasic, B., Bas, E., Winnubst, J., Menon, V., Graybuck, L. T., Nguyen, T. N., Smith, K. A., Yao, Z., et al. Distinct descending motor cortex pathways and their roles in movement. *Nature*, 563(7729):79–84, 2018.

Farhoodi, R. and Kording, K. P. Sampling neuron morphologies. *BioRxiv*, pp. 248385, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., , and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems (NeurIPS)*, pp. 2672–2680, 2014.

Harrison, R. G. The outgrowth of the nerve fiber as a mode of protoplasmic movement. *Journal of Experimental Zoology*, 9(4):787–846, 1910.

Hatcher, A. *Algebraic topology*. Cambridge Univ. Press, Cambridge, 2000. URL https://cds.cern.ch/record/478079.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 770–778, 2016.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems (NeurIPS)*, 30, 2017.

Hines, M. L. and Carnevale, N. T. The neuron simulation environment. *Neural computation*, 9(6):1179–1209, 1997.

Hines, M. L. and Carnevale, N. T. Neuron: a tool for neuroscientists. *The neuroscientist*, 7(2):123–135, 2001.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Janssen, R. and Budd, G. E. Expression of netrin and its receptors uncoordinated-5 and frazzled in arthropods and onychophorans suggests conserved and diverged functions in neuronal pathfinding and synaptogenesis. *Developmental Dynamics*, 2022.

Jefferis, G. S., Potter, C. J., Chan, A. M., Marin, E. C., Rohlfing, T., Maurer, C. R., and Luo, L. Comprehensive maps of drosophila higher olfactory centers: spatially segregated fruit and pheromone representation. *Cell*, 128 (6):1187–1203, 2007.

Kanari, L., Dictus, H., Chalimourda, A., Arnaudon, A., Van Geit, W., Coste, B., Shillcock, J., Hess, K., and Markram, H. Computational synthesis of cortical dendritic morphologies. *Cell Reports*, 39(1):110586, 2022.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S., Hudspeth, A. J., Mack, S., et al. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

Kim, Y., Sinclair, R., Chindapol, N., Kaandorp, J. A., and De Schutter, E. Geometric theory predicts bifurcations in minimal wiring cost trees in biology are flat. *PLoS computational biology*, 8(4):e1002474, 2012.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Koene, R. A., Tijms, B., Van Hees, P., Postma, F., De Ridder, A., Ramakers, G. J., Van Pelt, J., and Van Ooyen, A. Netmorph: a framework for the stochastic generation of large scale neuronal networks with realistic neuron morphologies. *Neuroinformatics*, 7:195–210, 2009.

Köhler, M., Hirschberg, B., Bond, C., Kinzie, J. M., Marrion, N., Maylie, J., and Adelman, J. Small-conductance, calcium-activated potassium channels from mammalian brain. *Science*, 273(5282):1709–1714, 1996.

Kole, M. H., Hallermann, S., and Stuart, G. J. Single ih channels in pyramidal neuron dendrites: properties, distribution, and impact on action potential output. *Journal of Neuroscience*, 26(6):1677–1687, 2006.

Kong, X., Huang, W., and Liu, Y. Conditional antibody design as 3d equivariant graph translation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=LFHFQbjxIiP.

Korngreen, A. and Sakmann, B. Voltage-gated k+ channels in layer 5 neocortical pyramidal neurones from young rats: subtypes and gradients. *The Journal of physiology*, 525(3):621–639, 2000.

Kramer, A. and Kuwada, J. Formation of the receptive fields of leech mechanosensory neurons during embryonic development. *Journal of Neuroscience*, 3(12):2474–2486, 1983.

Krottje, J. K. and Ooyen, A. v. A mathematical framework for modeling axon guidance. *Bulletin of Mathematical Biology*, 69(1):3–31, 2007.

Landisman, C. E. and Connors, B. W. Vpm and pom nuclei of the rat somatosensory thalamus: intrinsic neuronal properties and corticothalamic feedback. *Cerebral cortex*, 17(12):2853–2865, 2007.

Laturnus, S., Kobak, D., and Berens, P. A systematic evaluation of interneuron morphology representations for cell type discrimination. *Neuroinformatics*, 18:591–609, 2020a.

Laturnus, S., von Daranyi, A., Huang, Z., and Berens, P. Morphopy: A python package for feature extraction of neural morphologies. *Journal of Open Source Software*, 5(52):2339, 2020b.

Laturnus, S. C. and Berens, P. Morphvae: Generating neural morphologies from 3d-walks using a variational autoencoder with spherical latent space. In *International Conference on Machine Learning (ICML)*, pp. 6021–6031. PMLR, 2021.

Lawrance, A. and Lewis, P. An exponential moving-average sequence and point process (ema1). *Journal of Applied Probability*, 14(1):98–113, 1977.

Le Bé, J.-V., Silberberg, G., Wang, Y., and Markram, H. Morphological, electrophysiological, and synaptic properties of corticocallosal pyramidal cells in the neonatal rat neocortex. *Cerebral cortex*, 17(9):2204–2213, 2007.

Liao, R., Li, Y., Song, Y., Wang, S., Hamilton, W., Duvenaud, D. K., Urtasun, R., and Zemel, R. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems (NeurIPS)*, 32, 2019.

Lim, Y., Cho, I.-T., Schoel, L. J., Cho, G., and Golden, J. A. Hereditary spastic paraplegia-linked reep1 modulates endoplasmic reticulum/mitochondria contacts. *Annals of neurology*, 78(5):679–696, 2015.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Lin, X., Li, Z., Ma, H., and Wang, X. An evolutionary developmental approach for generation of 3d neuronal morphologies using gene regulatory networks. *Neurocomputing*, 273:346–356, 2018.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems (NeurIPS)*, 31, 2018.

López-Cruz, P. L., Bielza, C., Larrañaga, P., Benavides-Piccione, R., and DeFelipe, J. Models and simulation of 3d neuronal dendritic trees using bayesian networks. *Neuroinformatics*, 9(4):347–369, 2011.

Lu, P. and Werb, Z. Patterning mechanisms of branched organs. *Science*, 322(5907):1506–1509, 2008.

Magistretti, J. and Alonso, A. Biophysical properties and slow voltage-dependent inactivation of a sustained sodium current in entorhinal cortex layer-ii principal neurons: a whole-cell and single-channel study. *The Journal of general physiology*, 114(4):491–509, 1999.

Markram, H., Muller, E., Ramaswamy, S., Reimann, M. W., Abdellah, M., Sanchez, C. A., Ailamaki, A., Alonso-Nanclares, L., Antille, N., Arsever, S., et al. Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163(2):456–492, 2015.

Matthews, B. J., Kim, M. E., Flanagan, J. J., Hattori, D., Clemens, J. C., Zipursky, S. L., and Grueber, W. B. Dendrite self-avoidance is controlled by dscam. *Cell*, 129(3): 593–604, 2007.

Memelli, H., Torben-Nielsen, B., and Kozloski, J. Self-referential forces are sufficient to explain different dendritic morphologies. *Frontiers in Neuroinformatics*, 7:1, 2013.

Oh, S. W., Harris, J. A., Ng, L., Winslow, B., Cain, N., Mihalas, S., Wang, Q., Lau, C., Kuan, L., Henry, A. M., et al. A mesoscale connectome of the mouse brain. *Nature*, 508 (7495):207–214, 2014.

Otopalik, A. G., Goeritz, M. L., Sutton, A. C., Brookings, T., Guerini, C., and Marder, E. Sloppy morphological tuning in identified neurons of the crustacean stomatogastric ganglion. *Elife*, 6:e22352, 2017.

Packer, A. M., McConnell, D. J., Fino, E., and Yuste, R. Axo-dendritic overlap and laminar projection can explain interneuron connectivity to pyramidal cells. *Cerebral cortex*, 23(12):2790–2802, 2013.

Parekh, R. and Ascoli, G. A. Neuronal morphology goes digital: a research hub for cellular and system neuroscience. *Neuron*, 77(6):1017–1038, 2013.

Paulsen, B., Velasco, S., Kedaigle, A. J., Pigoni, M., Quadrato, G., Deo, A. J., Adiconis, X., Uzquiano, A., Sartore, R., Yang, S. M., et al. Autism genes converge on asynchronous development of shared neuron classes. *Nature*, 602(7896):268–273, 2022.

Peng, H., Xie, P., Liu, L., Kuang, X., Wang, Y., Qu, L., Gong, H., Jiang, S., Li, A., Ruan, Z., et al. Morphological diversity of single neurons in molecularly defined cell types. *Nature*, 598(7879):174–181, 2021.

Prakash, A., Chitta, K., and Geiger, A. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7077–7087, 2021.

Purohit, P. K. and Smith, D. H. A model for stretch growth of neurons. *Journal of biomechanics*, 49(16):3934–3942, 2016.

Qiao, Z., Wang, P., Fu, Y., Du, Y., Wang, P., and Zhou, Y. Tree structure-aware graph representation learning via integrated hierarchical aggregation and relational metric learning. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 432–441. IEEE, 2020.

Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

Reinhard, K., Li, C., Do, Q., Burke, E. G., Heynderickx, S., and Farrow, K. A projection specific logic to sampling visual inputs in mouse superior colliculus. *Elife*, 8:e50697, 2019.

Rettig, J., Wunder, F., Stocker, M., Lichtinghagen, R., Mastiaux, F., Beckh, S., Kues, W., Pedarzani, P., Schröter, K. H., and Ruppersberg, J. P. Characterization of a shaw-related potassium channel family in rat brain. *The EMBO Journal*, 11(7):2473–2486, 1992.

Reuveni, I., Friedman, A., Amitai, Y., and Gutnick, M. J. Stepwise repolarization from ca2+ plateaus in neocortical pyramidal cells: evidence for nonhomogeneous distribution of hva ca2+ channels in dendrites. *Journal of Neuroscience*, 13(11):4609–4621, 1993.

Ruddigkeit, L., Van Deursen, R., Blum, L. C., and Reymond, J.-L. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems (NeurIPS)*, 29, 2016.

Samsonovich, A. V. and Ascoli, G. A. Statistical morphological analysis of hippocampal principal neurons indicates cell-specific repulsion of dendrites from their own cell. *Journal of neuroscience research*, 71(2):173–187, 2003.

Samsonovich, A. V. and Ascoli, G. A. Statistical determinants of dendritic morphology in hippocampal pyramidal neurons: a hidden markov model. *Hippocampus*, 15(2): 166–183, 2005.

Sarnat, H. B. Axonal pathfinding during the development of the nervous system. *Annals of the Child Neurology Society*, 2023.

Schmitz, S. K., Hjorth, J. J., Joemai, R. M., Wijntjes, R., Eijgenraam, S., de Bruijn, P., Georgiou, C., de Jong, A. P., van Ooyen, A., Verhage, M., et al. Automated analysis of neuronal morphology, synapse number and synaptic recruitment. *Journal of neuroscience methods*, 195(2): 185–193, 2011.

Scorcioni, R., Polavaram, S., and Ascoli, G. A. 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.

Scott, E. K. and Luo, L. How do dendrites take their shape? *Nature neuroscience*, 4(4):359–365, 2001.

Sdrulla, A. D. and Linden, D. J. Dynamic imaging of cerebellar purkinje cells reveals a population of filopodia which cross-link dendrites during early postnatal development. *The Cerebellum*, 5(2):105–115, 2006.

Shao, H., Wang, L., Chen, R., Li, H., and Liu, Y. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. *arXiv preprint arXiv:2207.14024*, 2022.

Shi, Z., Innes-Gold, S., and Cohen, A. E. Membrane tension propagation couples axon growth and collateral branching. *Science Advances*, 8(35):eabo1297, 2022.

Shinbrot, T. Simulated morphogenesis of developmental folds due to proliferative pressure. *Journal of theoretical biology*, 242(3):764–773, 2006.

Sholl, D. Dendritic organization in the neurons of the visual and motor cortices of the cat. *Journal of anatomy*, 87(Pt 4):387, 1953.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems (NeurIPS)*, 28, 2015.

Sorensen, S. A., Bernard, A., Menon, V., Royall, J. J., Glattfelder, K. J., Desta, T., Hirokawa, K., Mortrud, M., Miller, J. A., Zeng, H., et al. Correlated gene expression and target specificity demonstrate excitatory projection neuron diversity. *Cerebral cortex*, 25(2):433–449, 2015.

Sorrells, S. F., Paredes, M. F., Cebrian-Silla, A., Sandoval, K., Qi, D., Kelley, K. W., James, D., Mayer, S., Chang, J., Auguste, K. I., et al. Human hippocampal neurogenesis drops sharply in children to undetectable levels in adults. *Nature*, 555(7696):377–381, 2018.

Stepanyants, A., Hirsch, J. A., Martinez, L. M., Kisvárday, Z. F., Ferecskó, A. S., and Chklovskii, D. B. Local potential connectivity in cat primary visual cortex. *Cerebral Cortex*, 18(1):13–28, 2008.

Sümbül, U., Song, S., McCulloch, K., Becker, M., Lin, B., Sanes, J. R., Masland, R. H., and Seung, H. S. A genetic and computational approach to structurally classify neuronal types. *Nature communications*, 5(1):3512, 2014.

Sweeney, N. T., Li, W., and Gao, F.-B. Genetic manipulation of single neurons in vivo reveals specific roles of flamingo in neuronal morphogenesis. *Developmental biology*, 247 (1):76–88, 2002.

Szebenyi, G., Callaway, J. L., Dent, E. W., and Kalil, K. Interstitial branches develop from active regions of the axon demarcated by the primary growth cone during pausing behaviors. *Journal of Neuroscience*, 18(19):7930–7940, 1998.

Tamariz, E. and Varela-Echavarría, A. The discovery of the growth cone and its influence on the study of axon guidance. *Frontiers in neuroanatomy*, 9:51, 2015.

Torben-Nielsen, B. and De Schutter, E. Context-aware modeling of neuronal morphologies. *Frontiers in neuroanatomy*, 8:92, 2014.

Uylings, H. and Smit, G. Three-dimensional branching structure of pyramidal cell dendrites. *Brain Research*, 87 (1):55–60, 1975.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Van Pelt, J., Dityatev, A. E., and Uylings, H. B. Natural variability in the number of dendritic segments: model-based inferences about branching during neurite outgrowth. *Journal of Comparative Neurology*, 387(3):325–340, 1997.

Watanabe, T. and Costantini, F. Real-time analysis of ureteric bud branching morphogenesis in vitro. *Developmental biology*, 271(1):98–108, 2004.

Wessells, N. K. and Nuttall, R. P. Normal branching, induced branching, and steering of cultured parasympathetic motor neurons. *Experimental cell research*, 115(1): 111–122, 1978.

Williams, P. A., Thirgood, R. A., Oliphant, H., Frizzati, A., Littlewood, E., Votruba, M., Good, M. A., Williams, J., and Morgan, J. E. Retinal ganglion cell dendritic degeneration in a mouse model of alzheimer's disease. *Neurobiology of aging*, 34(7):1799–1806, 2013.

Winnubst, J., Bas, E., Ferreira, T. A., Wu, Z., Economo, M. N., Edson, P., Arthur, B. J., Bruns, C., Rokicki, K., Schauder, D., et al. Reconstruction of 1,000 projection neurons reveals new cell types and organization of long-range connectivity in the mouse brain. *Cell*, 179(1):268–281, 2019.

Xu, J. and Durrett, G. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4503–4513, 2018.

Yang, J., He, Y., and Liu, X. Retrieving similar substructures on 3d neuron reconstructions. *Brain Informatics*, 7(1): 1–9, 2020.

Yuste, R. From the neuron doctrine to neural networks. *Nature reviews neuroscience*, 16(8):487–497, 2015.

Zador, A., Escola, S., Richards, B., Ölveczky, B., Bengio, Y., Boahen, K., Botvinick, M., Chklovskii, D., Churchland, A., Clopath, C., et al. Catalyzing next-generation artificial intelligence through neuroai. *Nature Communications*, 14(1):1597, 2023.

Zubler, F. and Douglas, R. A framework for modeling the growth and development of neurons and networks. *Frontiers in computational neuroscience*, pp. 25, 2009.

Appendix of MorphGrower

A. Data Preprocess

A.1. Resampling Algorithm

The resampling algorithm is performed on a single branch and aims to solve the uneven sample node density over branches issue. $x \in (a, b)$ means that a coordinate x is on the segment with endpoints a and b where a and b are also coordinates and we define the euclidean distance between a and b as $dist(a, b)$. Then for a branch $b = \{v_1, v_2, \ldots, v_l\}$, after resampling b, we can obtain $b' = \{v'_1, v'_2, \ldots, v'_{l'}\}$, where v_i and v'_i are 3D coordinates and v'_i satisfies the following constraints:

$$\begin{cases} Len = \dfrac{\sum_{i=1}^{l-1} dist(v_i, v_{i+1})}{l' - 1}. \\ \forall\, j \in \{1, 2, \ldots, l'\}, \exists\, i \in \{1, 2, \ldots, l-1\},\ v'_j \in (v_i, v_{i+1}). \\ \forall\, v'_j \in (v_a, v_{a+1}) \text{ and } v'_{j+1} \in (v_b, v_{b+1}),\ \text{if } a \neq b,\ dist(v'_j, v_{a+1}) + \sum_{i=a+1}^{b-1} dist(v_i, v_{i+1}) + dist(v_b, v'_{j+1}) = Len. \\ \forall\, v'_j \in (v_a, v_{a+1}) \text{ and } v'_{j+1} \in (v_b, v_{b+1}),\ \text{if } a = b,\ dist(v'_j, v'_{j+1}) = Len. \end{cases} \tag{11}$$

A.2. Preprocessing of The Raw Neuronal Morphology Samples

Reason for preprocessing. Neuronal morphology is manually reconstructed from mesoscopic resolution 3D image stack, usually imaging from optical sectioning tomography imaging techniques to obtain images, and mechanical slicing yields a lower resolution in the z-axis than the resolution in xy-axis obtained from imaging. Limited by the resolution, the reconstruction algorithm can only connect two points by pixel-by-pixel, and the sampled point coordinates are aligned with the image coordinates, forming a series of jagged straight lines as shown in Figure 6(a). Therefore, directly using the raw data for training is unsuitable and preprocessing is necessary.

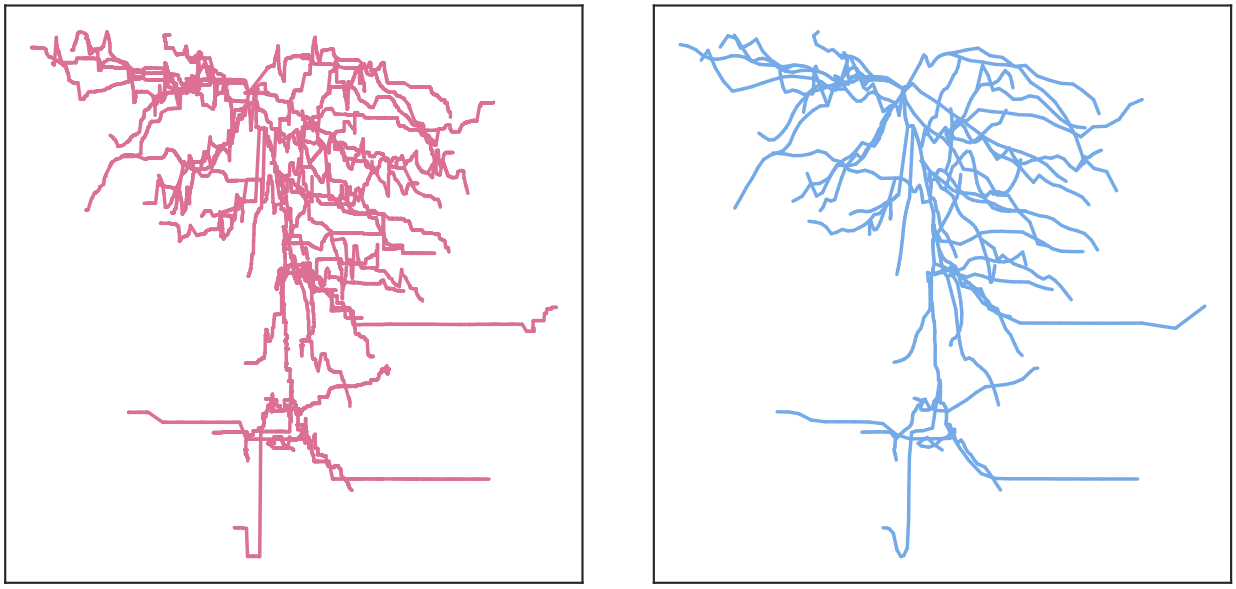
To denoise the raw morphology samples, the preprocessing strategy we adopted in this paper is demonstrated as follows and it contains four key operations:

- **Merging pseudo somas.** In some morphology samples, there may exist multiple somas at the same time, where we call them pseudo somas. Due to that the volume of soma is relatively larger, we use multiple pseudo somas to describe the shape of soma and the coordinates of these pseudo somas are quite close to each other. In line with MorphoPy (Laturnus et al., 2020b), which is a python package tailored for neuronal morphology data, the coordinates of pseudo somas will be merged to the centroid of their convex hull as a final true soma.

- **Inserting Nodes.** As noted in Section 1, only the soma is allowed to have more than two outgoing branches (Bray, 1973; Wessells & Nuttall, 1978; Szebenyi et al., 1998). However, in some cases, two or even more bifurcations would be so close that they may share the same coordinate after reconstruction from images due to some manual errors. Notice that these cases are different from pseudo somas. We solve this problem by inserting points. For a non-soma node v with more than two outgoing branches, we denote the set of the closest nodes on each subsequent branches as $\{v_{\text{close}_1}, v_{\text{close}_2}, \ldots, v_{\text{close}_{N_{\text{close}}}}\}$, where N_{close} represents the number of branches extending away from v. We only reserve two edges $e_{v,v_{\text{close}_1}}$, $e_{v,v_{\text{close}_2}}$ and delete the rest edges $e_{v,v_{\text{close}_i}}$ for $3 \leq i \leq N_{\text{close}}$. Then we insert a new node which is located at the midpoint between node v and v_{close_1} and denote it as v_{mid_1}. Next we reconnect the $v_{\text{close}_3}, \ldots, v_{\text{close}_{N_{\text{close}}}}$ to node v_{mid_1}, i.e. adding edges $e_{v_{\text{mid}_1},v_{\text{close}_i}}$ for $3 \leq i \leq N_{\text{close}}$. We repeat such procedure until there is no other node having more than two outgoing edges apart from the soma.

- **Resampling branches.** We perform resampling algorithm described in Appendix A.1 over branches of each neuronal morphology sample. The goal of this step is two-fold: *i)* distribute the nodes on branches evenly so that the following smoothing step could be easier; *ii)* cut down the number of node coordinates to reduce IO workloads during training.

- **Smoothing branches.** As shown in Figure 6(a), the raw morphology contains a lot of jagged straight line, posing an urgency for smoothing the branches. We smooth the branches via a sliding window. The window size is set to $2w + 1$, where w is hyper-parameter. Given a branch $b_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{L(i)}}\}$, for a node v_{i_j} on b_j, if there are at least w nodes



(a) Before preprocessing. (b) After preprocessing.

Figure 6: An example morphology sample from M1-EXC dataset for illustrating the preprocessing. We demonstrate the projections of the morphology sample onto the xy plane before and after preprocessing here.

on b_i before it and at least w nodes on b_i after it, we directly calculate the average coordinate of v_{i_j}'s previous w nodes, v_{i_j}'s subsequent w nodes and v_{i_j} itself. Then, we obtain a new node v'_{i_j} at the average coordinate calculated. We denote the set of nodes that are involved in calculating the coordinate of v'_{i_j} as $\mathcal{N}_{\text{smooth}}(v_{i_j})$. For those nodes which are close to the two ends of b_i, the number of nodes before or after them may be less than w. To solve this issue, we define $\mathcal{N}_{\text{smooth}}(v_{i_j})$ as follows:

$$\mathcal{N}_{\text{smooth}}(v_{i_j}) = \{v_{i_{j-p}}, v_{i_{j-p+1}}, \ldots, v_{i_{j-1}}, v_{i_j}, v_{i_{j+1}}, \ldots, v_{i_{j+s-1}}, v_{i_{j+s}}\}, \tag{12}$$

where $p = \min\{w, j-1, \eta(L(i)-j)\}$ and $s = \min\{w, L(i)-j, \eta(j-1)\}$. $\eta \in \mathbb{N}^+$ a is hyper-parameter in case of the extreme imbalance between the number of nodes before v_{i_j} and the number of nodes after v_{i_j}. We reserve the two ends of b_i and calculate new coordinates by averaging the coordinates of $\mathcal{N}_{\text{smooth}}(v_{i_j})$ for $2 \leq j \leq L(i)-1$. Finally, we can obtain a smoother branch $b'_i = \{v_{i_1}, v'_{i_2}, \ldots, v'_{i_{L(i)-1}}, v_{i_{L(i)}}\}$.

Remark. Due to the low quality of raw **M1-EXC** and **M1-INH** datasets, the above data preprocessing procedure is applied to both of them. As for the raw **VPM** and **RGC** datasets, their quality is gooed enough. Therefore, we do not apply the above preprocessing procedure to them. We show an example morphology sample before and after preprocessing from M1-EXC dataset in Figure 6.

B. Definitions of Metrics

In this section, we present the detailed definitions of all metrics we adopt for evaluation in this paper.

B.1. Metric Adopted in Section 4.2

Before introducing the metrics, we first present the descriptions of their related quantitative characterizations as follows. Recall that all these adopted quantitative characterizations are defined on a single neuronal morphology.

Branch Path Length (BPL). Given a neuronal morphology sample T, we denote the set of branches it contains as $\mathcal{B}$. For a branch $b_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{L(i)}}\}$, its path length is calculated by $\sum_{j=1}^{L(i)-1} dist(v_{i_j}, v_{i_{j+1}})$. BPL means the mean path

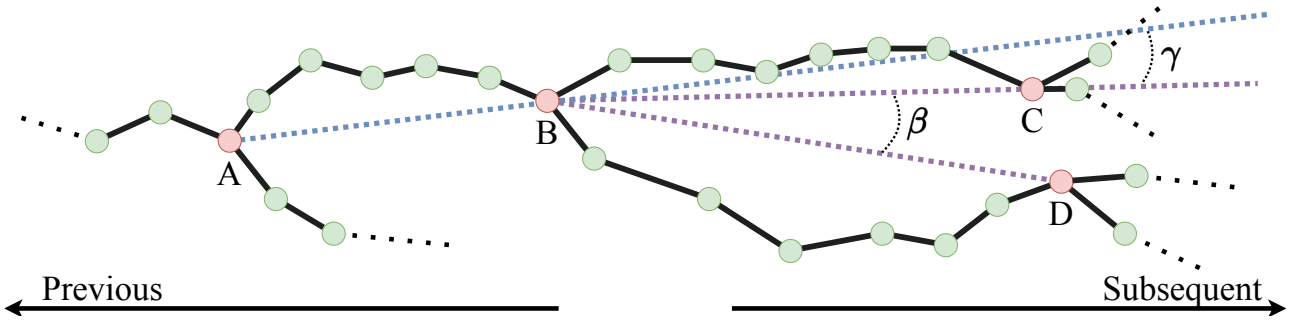


Figure 7: Illustrations for BAR and APS to facilitate understanding the deifinitions of them. Birfurcations are colored red.

length of branches over T, i.e.,

$$\mathrm{BPL}(T) = \frac{1}{|\mathcal{B}|} \sum_{b_i \in \mathcal{B}} \sum_{j=1}^{L(i)-1} dist(v_{ij}, v_{ij+1}). \tag{13}$$

Maximum Euclidean Distance (MED). Given a neuronal morphology sample $T = (V, E)$, we denote the soma as v_{soma}. The $\mathrm{MED}(T)$ is defined as:

$$\mathrm{MED}(T) = \max_{v_i \in V} dist(v_i, v_{soma}). \tag{14}$$

Maximum Path Distance (MPD). Given a neuronal morphology sample $T = (V, E)$, we denote the soma as v_{soma}. For a node $v_i \in V$, $\{v_{soma}, v_{i^1}, v_{i^2}, \ldots, v_{i^{A_v(i)}}, v_i\}$ is the sequence of nodes contained on the path beginning from soma and ending at v_i, where $A_v(i)$ denotes the number of v_i's ancestor nodes except soma. The path distance of v_i is defined as $dist(v_{soma}, v_{i^1}) + \sum_{j=1}^{A_v(i)-1} dist(v_{i^j}, v_{i^{j+1}}) + dist(v_{i^{A_v(i)}}, v_i)$. Therefore, the maximum path distance (MPD) of T is formulated as:

$$\mathrm{MPD}(T) = \max_{v_i \in V} dist(v_{soma}, v_{i^1}) + \sum_{j=1}^{A_v(i)-1} dist(v_{i^j}, v_{i^{j+1}}) + dist(v_{i^{A_v(i)}}, v_i). \tag{15}$$

ConTracTion (CTT). CTT measures the mean degree of wrinkling of branches over a morphology T, and we denote the set of branches included in T as $\mathcal{B}$. For a branch $b_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{L(i)}}\}$, its degree of wrinkling is calculated by $\frac{dist(v_{i_1}, v_{i_{L(i)}})}{\sum_{j=1}^{L(i)-1} dist(v_{i_j}, v_{i_{j+1}})}$. Hence, the definition of CTT is:

$$\mathrm{CTT}(T) = \frac{1}{|\mathcal{B}|} \sum_{b_i \in \mathcal{B}} \frac{dist(v_{i_1}, v_{i_{L(i)}})}{\sum_{j=1}^{L(i)-1} dist(v_{i_j}, v_{i_{j+1}})}. \tag{16}$$

Amplitude between a pair of Sibling Branches (ASB). Given a morphology sample T, we denote the set of bifurcations it contains as $V_{\text{bifurcation}}$. For any bifurcation node $v_i \in V_{\text{bifurcation}}$, there is a pair of sibling branches extending away from v_i. We denote the endpoints of these two branches as v_{i1} and v_{i2}, respectively. The order of sibling branches does not matter here. The vector $\overrightarrow{v_i v_{i1}}$ and $\overrightarrow{v_i v_{i2}}$ can form a amplitude like the amplitude β in Figure 7. ASB represents the mean amplitude size of all such instances over T. Hence, $\mathrm{ASB}(T)$ is defined as:

$$\mathrm{ASB}(T) = \frac{1}{|V_{\text{bifurcation}}|} \sum_{v_i \in V_{\text{bifurcation}}} \angle v_{i1} v_i v_{i2}. \tag{17}$$

Amplitude between of a Previous branch and its Subsequent branch (APS). Given a morphology sample T, we denote the set of bifurcations it contains as $V_{\text{bifurcation}}$. For any bifurcation node $v_i \in V_{\text{bifurcation}}$, there is a pair of sibling branches extending away from v_i. Notice that v_i itself is not only the start point of these but also the ending point of another branch. As demonstrated in Figure 7, node B is a bifurcation. There is a pair of sibling branches extending away from B. Node C and D are the ending points of the sibling branches. In the meanwhile, B is the ending point of the branch whose start point

is A. $\overrightarrow{AB}$ and $\overrightarrow{BC}$ can form a amplitude i.e. the amplitude γ. In general, we denote the start point of the branch ending with v_i as $v_{i_{\text{start}}}$. We use $v_{i_{\text{end1}}}$ and $v_{i_{\text{end2}}}$ to represent the ending points of two subsequent branches. $\text{APS}(T)$ equals to the mean amplitude size of all amplitudes like γ over T and can be formulated as follows with $< \cdot, \cdot >$ representing the amplitude size between two vectors:

$$\text{APS}(T) = \frac{1}{2 \cdot |V_{\text{bifurcation}}|} \sum_{v_i \in V_{\text{bifurcation}}} \angle < \overrightarrow{v_{i_{\text{start}}} v_i}, \overrightarrow{v_i v_{i_{\text{end1}}}} > + \angle < \overrightarrow{v_{i_{\text{start}}} v_i}, \overrightarrow{v_i v_{i_{\text{end2}}}} > . \tag{18}$$

Now, we have presented detailed definitions of all six quantitative characterizations. Recalling that the six metrics adopted are just the expectation of these six characterizations over the dataset, we denote the dataset as $\mathcal{T} = \{T_i\}_{i=1}^{N_T}$, where N_T represents the number of morphology samples. **Then the six metrics are defined as below:**

- **Mean BPL (MBPL):**

$$\text{MBPL} = \frac{1}{N_T} \sum_{i=1}^{N_T} \text{BPL}(T_i). \tag{19}$$

- **Mean MED (MMED):**

$$\text{MMED} = \frac{1}{N_T} \sum_{i=1}^{N_T} \text{MED}(T_i). \tag{20}$$

- **Mean MPD (MMPD):**

$$\text{MMPD} = \frac{1}{N_T} \sum_{i=1}^{N_T} \text{MPD}(T_i). \tag{21}$$

- **Mean CTT (MCTT):**

$$\text{MCTT} = \frac{1}{N_T} \sum_{i=1}^{N_T} \text{CTT}(T_i). \tag{22}$$

- **Mean ASB (MASB):**

$$\text{MASB} = \frac{1}{N_T} \sum_{i=1}^{N_T} \text{ASB}(T_i). \tag{23}$$

- **Mean APS (MAPS):**

$$\text{MAPS} = \frac{1}{N_T} \sum_{i=1}^{N_T} \text{APS}(T_i). \tag{24}$$

B.2. Metric Adopted in Section 4.3

In the discrimination test, we adopt the **Accuracy** as metric for evaluation, which is widely-used in classification tasks. We abbreviate the **true-positives**, **false-negatives**, **true-negatives** and **false-negatives** as **TP**, **FP**, **TN** and **FN**, respectively. Then Accuracy is formulated as:

$$\text{Accuracy} = \frac{|\text{TP}| + |\text{TN}|}{|\text{TP}| + |\text{TN}| + |\text{FP}| + |\text{FN}|}. \tag{25}$$

B.3. Metric Adopted in Section 4.4

Before we delve into the detailed definitions of metrics used in Section 4.4, we first illustrate some key concepts in electrophysiological recordings in Figure 8. Additionally, for each neuronal morphology sample, we can obtain more than one electrophysiological recording (details can be found in Appendix H.4). Recalling that morphology is essentially a tree-like structure, each leaf node in the morphology corresponds to an electrophysiological recording. Given a neuronal morphology T, we denote the number of leaf nodes in T as N_{leaf}, so we have a set of electrophysiological recordings

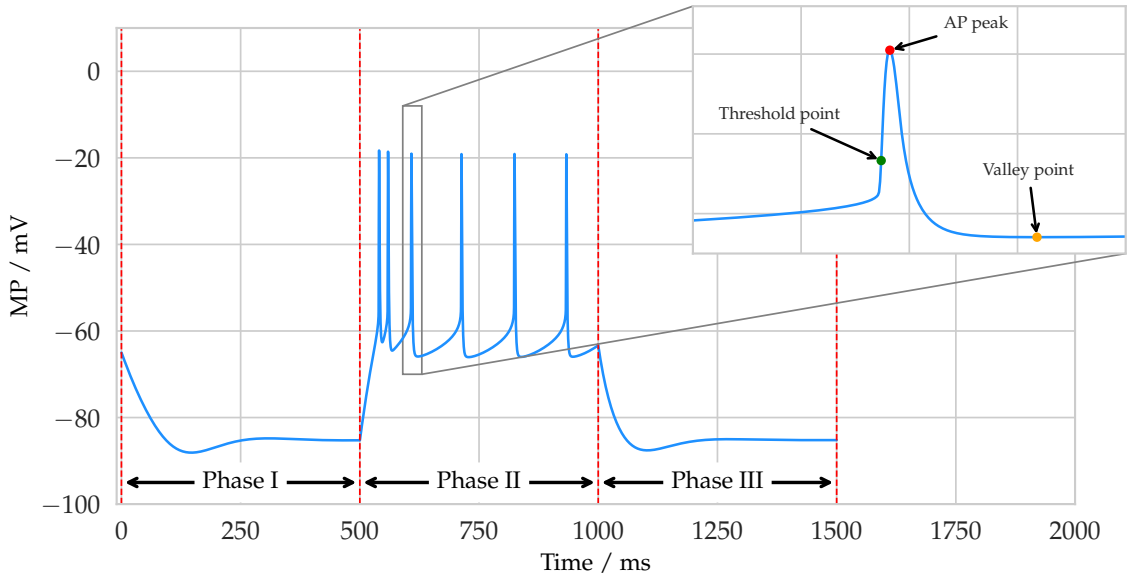


Figure 8: Demonstration of an electrophysiological recording. Here, the recording duration of the whole simulation is up to 1,500 ms for each voltage detection position. **a.** From an overall perspective, all voltage recordings detected at dendritic terminals involve three phases: *i)* The first phase, which is **Phase I** during 0-500 ms, involves the change from initial potential to nearly stabilized potential. *ii)* Upon receiving a current stimulus at 500 ms, which marks the beginning of **Phase II**, the membrane potential quickly rises to the threshold point and triggers **Action Potential (AP)** peaks regularly for 500 ms. *iii)* After the stimulation ends, the membrane potential gradually returns to the resting potential, waiting for the next stimulus input, marking **Phase III**. **b.** Delving deeper into the crucial Phase II, this stage features peaks, with each peak having three key points, namely, the Threshold point, the AP peak, and the Valley point (marked in green, red, and yellow in the figure, respectively). **The AP peak** is straightforward, being the point at the peak position with the largest AP amplitude. **The Threshold point** is identified as the point before reaching the AP peak where the rate of change in AP amplitude is greatest during its ascent. **The Valley point** refers to the lowest point in the descending curve following the AP peak.

$\{W_i\}_i^{N_{leaf}}$ corresponding to T. The metrics used in Section 4.4 are averages calculated across all neuronal morphologies. Below, we first introduce how to calculate related results for a single neuronal morphology:

Mean Frequency (MF). As the name suggests, MF refers to the mean frequency of a signal. In this paper, we directly use the $\mathrm{meanfreq}(\cdot)$[4] function provided in MATLAB to calculate the mean frequency of a signal. The input to the function is an entire electrophysiological recording of a neuronal morphology. Given a morphology T, we calculate the mean frequency for each of the electrophysiological recordings it contains, and then average these values to determine the final mean frequency of T:

$$\mathrm{MF}(T) = \sum_{i=1}^{N_{leaf}} \mathrm{meanfreq}(W_i). \tag{26}$$

Mean Action Potential Height (MAPH). This metric focuses on the peaks in Phase II as shown in Figure 8. For each peak p, we define its AP height (APH), denoted as $\mathrm{APH}(p)$, as the value of the AP amplitude at the peak point minus the value at the corresponding Valley point. For a specific electrophysiological recording W_i contained in a given morphology T, we denote the number of peaks in Phase II of W_i as N_{peak}^i. We first calculate the average AP height of all peaks in a specific electrophysiological recording W_i to determine its mean AP height. Then, we average the mean AP heights of all electrophysiological recordings contained in T to calculate the final mean AP height (MAPH) of T:

$$\mathrm{MAPH}(T) = \sum_{i=1}^{N_{leaf}} \sum_{j=1}^{N_{peak}^j} \mathrm{APH}(p_j). \tag{27}$$

[4] https://www.mathworks.com/help/signal/ref/meanfreq.html

Mean Action HyperPolarization Depth (MAHPD). Consistent with the previous metric MAPH, this metric also focuses on the characteristics of the peaks in Phase II as shown in Figure 8. For each peak p, we define its Action Hyperpolarization Depth, denoted as $\mathrm{AHPD}(p)$, as the result of subtracting the AP amplitude value at the corresponding Valley point from the AP amplitude value at the Threshold point. For a specific electrophysiological recording W_i contained in a given morphology T, we denote the number of peaks in Phase II of W_i as N^i_{peak}. We first calculate the average AHPD of all peaks in a specific electrophysiological recording W_i to determine its mean AHPD. Then, we average the mean AHPD of all electrophysiological recordings contained in T to calculate the final mean AHPD (MAHPD) of T:

$$\mathrm{MAHPD}(T) = \sum_{i=1}^{N_{leaf}} \sum_{j=1}^{N^j_{peak}} \mathrm{AHPD}(p_j). \tag{28}$$

Mean Action Potential Amplitude (MAPA). For each electrophysiological recording, we calculate the average of all the vertical coordinate values on the curve, which are the Action Potential amplitudes at each moment, to obtain the mean Action Potential amplitude of that curve. Since the simulation result for each morphology sample includes multiple electrophysiological recordings, we then calculate the average of the mean Action Potential amplitude for each waveform, which serves as the final mean Action Potential amplitude (MAPA) for a single neuronal morphology T, denoted as $\mathrm{ASB}(T)$.

We have presented detailed definitions of four quantitative characterizations designed for electrophysiological recordings. The four metrics adopted in Section 4.4 are just the expectation of these four characterizations over the dataset, we denote the dataset as $\mathcal{T} = \{T_i\}_{i=1}^{N_T}$, where N_T represents the number of morphology samples. **Then the four metrics are defined as below:**

- **Mean MF (MMF):**

$$\mathrm{MMF} = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathrm{MF}(T_i). \tag{29}$$

- **Mean MAPH (MMAPH):**

$$\mathrm{MMAPH} = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathrm{MAPH}(T_i). \tag{30}$$

- **Mean MAHPD (MMAHPD):**

$$\mathrm{MMAHPD} = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathrm{MAHPD}(T_i). \tag{31}$$

- **Mean MAPA (MMAPA):**

$$\mathrm{MMAPA} = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathrm{MAPA}(T_i). \tag{32}$$

C. More Discussion on MorphVAE

Here we present examples to illustrate some limitations of MorphVAE (Laturnus & Berens, 2021).

C.1. Failure to Ensure Topological Validty of Generated Morphologies

In the generation process, MorphVAE does not impose a constraint that only soma is allowed to have more than two outgoing edges, thereby failing to ensure the topological validty of generated samples. Here, we present an example generated by MorphVAE from VPM dataset, where there are more than one node that have more than two outgoing edges in Figure 9.

Elaboration on Topological Validity.

- Under the majority of circumstances, neurons exhibit bifurcations (Lu & Werb, 2008; Peng et al., 2021), which signifies that branching points, excluding the soma node, typically have only two subsequent branches. Specifically, in the extensive

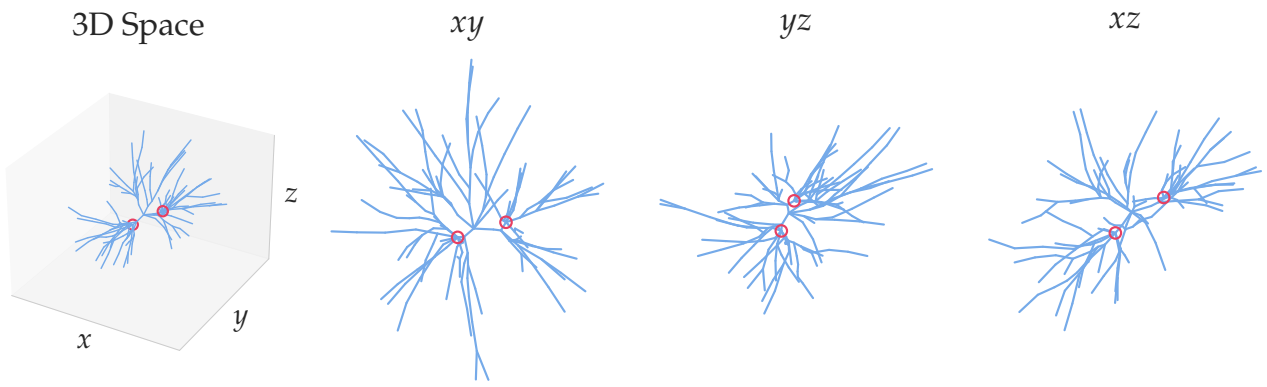


Figure 9: A topologically invalid neuronal morphology generated by MorphVAE. Nodes with more than two outgoing edges are circled in red. The first one demonstrates the morphology in 3D space. The three picture (from left to right) on the right are the projections of the sample onto xy, yz and xz planes, respectively.

Table 4: The topological validity rate of morphologies generated by MorphVAE on each dataset.

Dataset	**VPM**	**M1-EXC**	**M1-INH**	**RGC**
validity rate	0.052 ± 0.019	0.319 ± 0.088	0.038 ± 0.025	0.005 ± 0.010

morphological dataset presented in the work by Peng et al., the authors underscored the importance of treating trifurcations as topological errors that necessitate removal during post-processing quality assessment. Moreover, numerous tools for neuronal morphology analysis, such as the TREES toolbox (Cuntz et al., 2010), operate under the assumption that only binary neuron trees constitute valid and compatible data.

- In the infrequent instances where trifurcations have been observed, they were found to occur exclusively during the growth phase (Watanabe & Costantini, 2004). As a neuron reaches full maturation, these trifurcations often transform into two distinct bifurcations.

Consequently, there is no pragmatic rationale for intentionally generating or preserving trifurcations in a synthetic neuron. The indiscriminate and unregulated incorporation of trifurcations is ill-advised. Indeed, MorphVAE failed to adequately account for this factor, resulting in the generation of neurons with $3/4/5/n$-furcations.

We have computed the Topologically Validity Rate of samples generated by MorphVAE on the four datasets, indicating the proportion of samples that strictly adhere to a binary tree structure. The results are presented in Table 4, where we observe that the validity rate of MorphVAE is generally low. This highlights the significant improvement in topological plausibility offered by MorphGrower. Our generation mechanism ensures that our samples are all topologically valid.

Employing rejection sampling with the baseline MorphVAE to retain only topologically valid samples could be one potential approach. However, based on the results provided, especially on the RGC dataset where the validity rate is less than one percent, implementing rejection sampling might not yield a sufficient number of samples for a comprehensive evaluation.

C.2. Failure to Generate Morphologies with Relatively Longer 3D-walks

Given a authentic morphology, MorphVAE aims to generate a resembling morphology. MorphVAE regards 3D-walks as the basic generation blocks. Note that the length of 3D-walks varies. MorphVAE determines to truncate those relatively longer 3D-walks during the encoding stage. Hence, in the final generated samples, its difficult to find morphologies with relatively longer 3D-walks. We provide two examples from M1-EXC dataset in Figure 10 to show this limitation of MorphVAE.

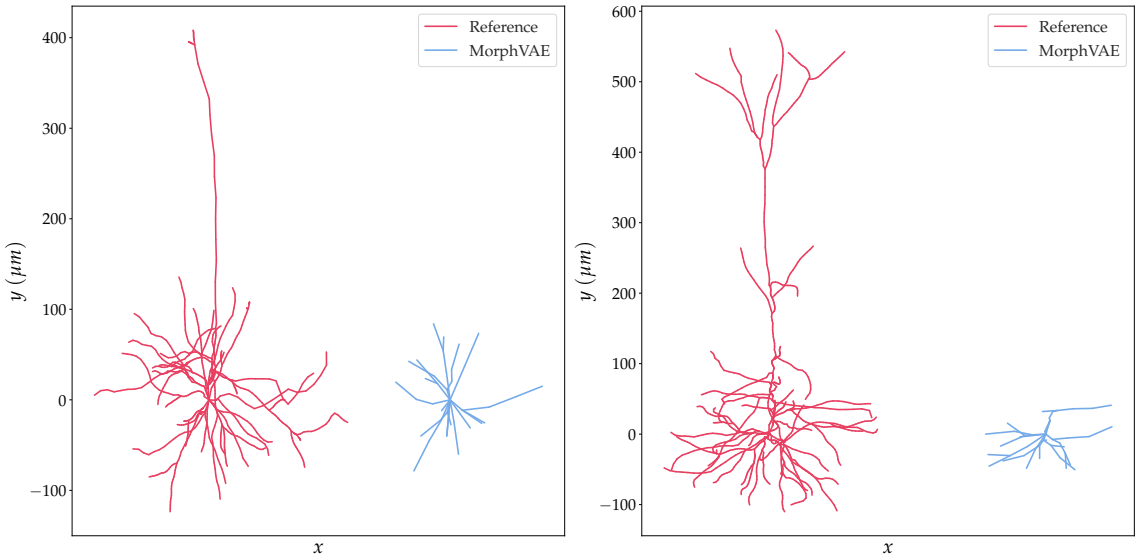


Figure 10: Each picture shows the projection of a pair of reference sample and the generated sample by MorphVAE, which is expected to resemble the target, onto the xy plane. We can see that MorphVAE does fail to generate morphologies with relatively longer 3D-walks.

D. Significant Distinctions from Graph Generation Task

In this section, we aim to **highlight the significant differences between the neural morphology generation task discussed in this paper and the graph generation task.**

Graph generation primarily focuses on creating topological graphs with specific structural properties, whereas neuronal morphology generation aims to generate geometric shapes that conform to specific requirements and constraints within the field of neuroscience. Morphology is defined by a set of coordinate points in 3D space and their connection relationships. It is important to note that two different morphologies can share the same topological structure. For example, as illustrated in Figure 11, the two neurons depicted in the first and second rows respectively possess the same topological structure in terms of algebraic topology (Hatcher, 2000) (i.e., the same Betti number-0 of 1 and the same Betti number-1 of 0), but are entirely distinct in morphology.

When attempting to adapt graph generation methods to the task of generating neuronal morphologies, scalability and efficiency pose significant challenges. Neuronal morphologies typically have far more nodes than the graphs used for training in graph generation. For example, the average number of nodes in the QM9 dataset (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014), which is a commonly used dataset for molecular generation tasks, is **18**, while the average number of nodes in the VPM dataset (Landisman & Connors, 2007; Peng et al., 2021), the dataset used in our paper, is **948**. Graph generation methods require generating an adjacency matrix with $\Theta(n^2)$ size, where n is the number of nodes, resulting in unaffordable computational overhead.

The most significant difference between graph and morphology generation is that every neuron morphology is a tree with special structural constraints. Thus, the absence of loops is a crucial criterion for valid generated morphology samples. We surveyed numerous graph generation-related articles and found that existing algorithms do not impose explicit constraints to ensure loop-free graph samples. This is primarily due to the application scenarios of current graph generation tasks, which mainly focus on molecular generation. Since molecules naturally have the possibility of containing loops, there is no need to consider the no-loop constraint. Furthermore, as stated in Appendix C.1, in most cases, neurons have only bifurcations (Lu & Werb, 2008; Peng et al., 2021) (i.e., branching points except for the soma node have only two subsequent branches). This implies that the trees in this study, except for the soma node, must be at most binary, resulting in even more stringent generated constraints. Nevertheless, current graph generation methods cannot ensure strict compliance with the aforementioned constraints.

Overall, there are significant differences between the graph generation task and the neuronal morphology generation task,

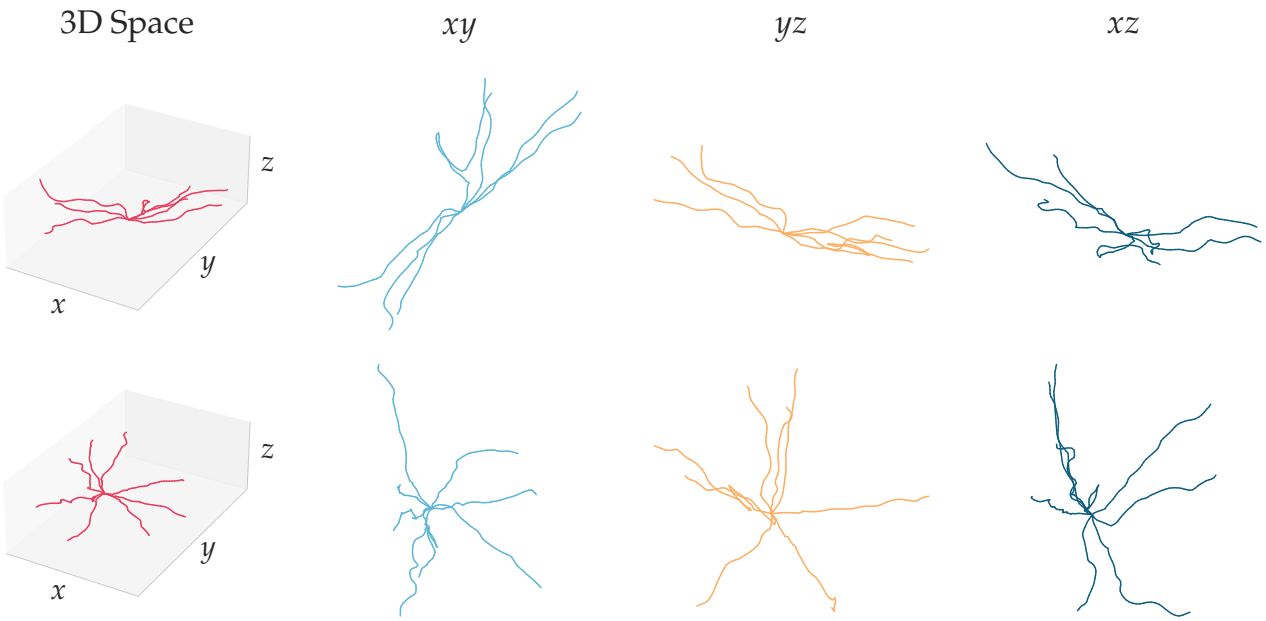


Figure 11: Two distinct neural morphology data are shown in two rows respectively. Both of them are trees composed of branches only connected by the root node (soma).

which make it non-trivial to adapt existing graph generation methods to the latter. This is also why neither MorphVAE nor our paper chose any existing graph generation method as a baseline for comparison.

E. Supplementary Clarifications for Resampling Operation

Although recurrent neural networks (RNNs) are able to encode or generate branches of any length, the second resampling operation is still necessary for the following reasons. Unlike numerous sequence-to-sequence tasks in the natural language processing (NLP) field, neuron branches are composed of 3D coordinates, which are continuous data type rather than discrete type. Therefore, it is difficult to use an approach like outputting an <END> token to halt generation. Additionally, it is challenging to calculate reconstruction loss between branches of different lengths.

In our experiments, we resample each branch to a fixed number of nodes, setting the hyperparameter at 32, which exceeds the original node count for most branches in the raw data. To demonstrate that our resampling operation does not have a significant impact on the morphology of neurons, we conducted the following statistics. We refer to branches with over 32 nodes in the raw data as "large branches". As shown in Table 5, for the majority of branches, we perform upsampling rather than downsampling, thereby preserving the morphology well.

Table 5: Statistics related to large branches.

Dataset	**VPM**	**M1-EXC**	**M1-INH**	**RGC**
total # of branches	26,395	18,088	114,139	232,315
total # of large branches	2,127	0	0	92
percentage $\frac{\text{large}}{\text{all}}$ / %	8.05	0.00	0.00	0.00

We designed an experiment to further investigate the impact of our resampling operation on the original branches, particularly on the large branches. We defined a metric called path length difference (abbreviated as PLD) to reflect the difference between two branches. We use PLD to measure the difference between the same branch before and after resampling. For

two branches $b_1 = \{x^{(1)}, x^{(2)}, \cdots, x^{(i)}\}$ and $b_2 = \{y^{(1)}, y^{(2)}, \cdots, y^{(j)}\}$ the PLD is defined as:

$$\left| \sum_{c=1}^{i-1} \sqrt{\sum_{d=1}^{3} (x_d^{(c)} - x_d^{(c+1)})^2} - \sum_{c=1}^{j-1} \sqrt{\sum_{d=1}^{3} (y_d^{(c)} - y_d^{(c+1)})^2} \right|. \tag{33}$$

The result is shown in Table 6 and since there are no large branches in M1-EXC and M1-INH, the corresponding results are missing.

Table 6: PLD results for branches before and after resampling. Due to the absence of large branches in the M1-EXC and M1-INH datasets, data related to large branches are missing on the two datasets.

Dataset	RGC	VPM	M1-EXC	M1-INH
avg. PLD on all branches	0.13 ± 0.42	0.17 ± 0.26	0.37 ± 0.68	0.30 ± 0.70
avg. path length of all original branches	24.31 ± 25.30	52.20 ± 46.29	60.32 ± 51.03	43.30 ± 43.48
avg. PLD on large branches	7.65 ± 3.16	0.65 ± 0.37	———	———
avg. path length of all large branches	264.29 ± 44.00	139.82 ± 14.02	———	———

From the table above, we can observe that compared to the original branch lengths, the length differences of the branches after resampling are quite small across all datasets, even for large branches. This further indicates that our resampling operation can effectively preserve the original neuronal morphology.

F. Motivation of the Choice of Architectures

In the main text, we have already presented the rationale behind our selection of specific architectures. Here, we provide a concise summary of the motivation behind our choice of key architectures.

F.1. Using LSTM as Our Branch Encoder

Our baseline MorphVAE incorporates LSTM as its branch encoder and decoder. To ensure a fair comparison, we have followed the same approach as our baseline and also employed LSTM as our branch encoder.

Due to our limited data samples, we are uncertain if more complex sequence-based models like the Transformer would be effectively trainable. As an experiment, we replace our LSTM branch encoder with a Transformer, and the results, which are presented in Appendix J.1, indicate that the Transformer does not demonstrate significant advantages over the LSTM in terms of performance. This suggests that the LSTM is already sufficiently powerful for this task.

F.2. Using vMF-VAE

The vMF-VAE with a fixed κ is proposed to prevent the KL collapse typically observed in the Gaussian VAE setting (Xu & Durrett, 2018; Davidson et al., 2018). By fixing κ, the KL distance between the posterior distribution $\mathrm{vMF}(\cdot, \kappa)$ and the prior distribution $\mathrm{vMF}(\cdot, 0)$ remains constant, thus avoiding the KL collapse.

F.3. Treating A Branch as A node and Employing T-GNN-like Networks.

The reasons for this aspect have already been explained in great detail in Section 3.2 *Encode Global Condition* of the main text, so we will not reiterate them here.

G. Future Work

G.1. Inclusion of the Branch Diameter

In our paper, we focus on the three-dimensional spatial coordinates of each node, with each node corresponding to a three-dimensional representation. To incorporate the diameter, we would simply add an extra dimension to the node representation, expanding the original three-dimensional representation to four dimensions. Incorporating the diameter into

our method is relatively straightforward and does not necessitate significant modifications to our model design. Furthermore, diameter variations within neurons are generally smooth. Even when significant changes in diameter occur, as noted in the work by (Conde-Sousa et al., 2017), we generally ascribe these inconsistencies to problems encountered during tissue preprocessing and the staining procedure. This indicates that differences in this dimension are less significant compared to the other three dimensions (spatial coordinates), making the training process for this dimension more stable and easier for the model to learn. Initially, we considered including the diameter in our model learning process, but given that the neuron's 3D geometry is of greater importance than the diameter and remains the primary focus, we ultimately chose not to include it. Additionally, due to limitations in imaging technology, not all neuron datasets contain diameter information, which is another reason why we did not take diameter into account. However, it is important to note that diameter information is essential for characterizing more specific neurons. We believe that as neuronal imaging technology advances, this information will gradually become more available. With sufficient training data and the inclusion of diameter information, our method should be capable of generating more detailed neurons.

G.2. Inclusion of the Spines or Boutons

As mentioned in the limitations section of our paper, the existing data only contains spatial coordinate information, lacking additional input. If we could obtain splines/boutons information, we have a simple and feasible idea: for each spline/bouton, we can learn a feature and then share that feature with the surrounding nodes. This can be concatenated with the features they already encode, forming a new representation for each node. This design can make use of the additional splines/boutons information provided and does not require significant modifications to our model. However, this idea is just an initial simple attempt, and there may be better ways to utilize this extra information. Similarly, to the diameter mentioned earlier, data on spines/boutons is also scarce at this stage. If we have access to a wealth of such data, we believe it could further improve our neuron morphology generation results.

G.3. Use of Dense Imaging Data

We believe that the dense imaging data can serve two main purposes as below:

- Since dense imaging data encompasses a vast amount of information about individual neurons, we are confident that it will facilitate the enhanced generation of single neuron morphologies in the future.
- Dense imaging data illustrates the intricate connections between neurons, which will prove highly valuable for generating simulated neuron populations down the line.

G.4. Other Potential Applications

Our approach is not limited to the neuronal morphology generation task alone. In fact, our method can be adapted to generate data with similar tree-like branching structures in morphology (or even beyond). For instance, retinal capillaries are also scarce in data, and we can attempt to generate more retinal capillary data using our method to address this data insufficiency issue. With additional capillary samples, the related segmentation models can be trained more effectively, promoting the development of biomedical engineering and bioimaging fields. Therefore, the scope of our method's application is not narrow.

Moreover, as emphasized in the introduction section of our paper, the current process of obtaining single neuron morphology data is time-consuming and labor-intensive. Therefore, the motivation behind our method is to design an efficient generation method for single neuron morphology samples. In the future, we can also try to use our method as a basic building block to construct a neuron population, which will help us gain a deeper understanding of information transmission within the nervous system and further our knowledge of brain function.

H. Implementation Details

This section describes the detailed experimental setup or training configurations for MorphVAE and MorphGrower in order for reproducibility. We firstly list the configurations of our environments and packages as below:

- Ubuntu: 20.04.2
- CUDA: 10.1

- Python: 3.7.0
- Numpy: 1.21.6
- Pytorch: 1.12.1
- PyTorch Geometric: 2.1.0
- Scipy: 1.7.3

Experiments are repeated 5 times with mean and standard deviation reported, running on a machine with i9-10920X CPU, RTX 3090 GPU and 128G RAM.

H.1. MorphVAE

Resampling Distance. MorphVAE (Laturnus & Berens, 2021) regards a 3D-walk as a basic building block when generating neuronal morphologies. Following the data preparation steps of MorphVAE, all the morphologies are first resampled at a certain distance and then scaled down according to the resampling distance to make the training and clustering easier. The resampling distances over datasets are all shown in Table 7.

Table 7: The resample distance for each dataset. We adopt the default distance used in the original paper for RGC, M1-EXC and M1-INH. As for VPM, which is not adopted in MorphVAE for evaluation, we set its corresponding resampling distance to 30 µm.

Dataset	**VPM**	**RGC**	**M1-EXC**	**M1-INH**
Resampling Distance / µm	30	30	50	40

Hyper Parameter Selection. Apart from the hyper-parameters mentioned in Section 4, we also perform grid search for the supervised-learning proportion over $\{1, 0.5, 0.1, 0\}$ on all datasets except **VPM** because no cell type label is provided for **VPM** and the classifier is not set up for this dataset. Following the original paper, we set the 3D walk length as 32 and $\kappa = 500$. The teaching-force for training decoder is set to 0.5. As for the distance threshold for agglomerative clustering, we use 0.5 for **VPM**, **M1-EXC**, **M1-INH** and 1.0 for **RGC**.

Pretraining. Following the original paper of MorphVAE, to better encode the branch information, we pretrain MorphVAE on the artificially generated toy dataset. Then model weight of pretrained LSTM is also used to initialize the LSTM part of the single branch encoder in our MorphGrower.

H.2. Our Method

We implement our method in Pytorch and adopt grid search to tune the hyper parameters. The learning rate is searched within $\{5\mathrm{e}-2, 1\mathrm{e}-2, 5\mathrm{e}-3, 1\mathrm{e}-3, 5\mathrm{e}-4, 1\mathrm{e}-4, 5\mathrm{e}-5, 1\mathrm{e}-5\}$ and dropout rate is searched within $\{0.1, 0.3, 0.5, 0.7\}$. In line with MorphVAE, we also the set the teaching-force for training decoder to 0.5. In line with morphvae, we use $\kappa = 500$ for training. The number of nodes rebuilding a branch, i.e. the hyper parameter L, is set to 32. The weight α in Eq. 5 is set to 0.5. Besides, the hyper-parameter m in Eq. 4 is set to 2. We adopt Adam optimizer to optimize learnable parameters.

H.3. Supplementary Descriptions for Section 4.3

Rationale behind evaluating plausibility using the CV method. Neurons have complicated, unique and type-defining structures, making their visual appearances a discriminative characteristics. It is important that a synthetic neuron "looks" real, either by human or by a computer. Thus, as an additional proof, we designed a vision-based classifier and tried to see if the generated neurons can "deceive" it. As reported in Table 2, the classifier cannot well differentiate our generated data from the real ones, while the MorphVAE results can be well identified. Furthermore, our use of three views as inputs to assess the plausibility of the generated neuronal morphology samples is well-founded. This can be supported by many cell typing studies (Jefferis et al., 2007; Sümbül et al., 2014; Laturnus et al., 2020a), researchers adopt density maps as

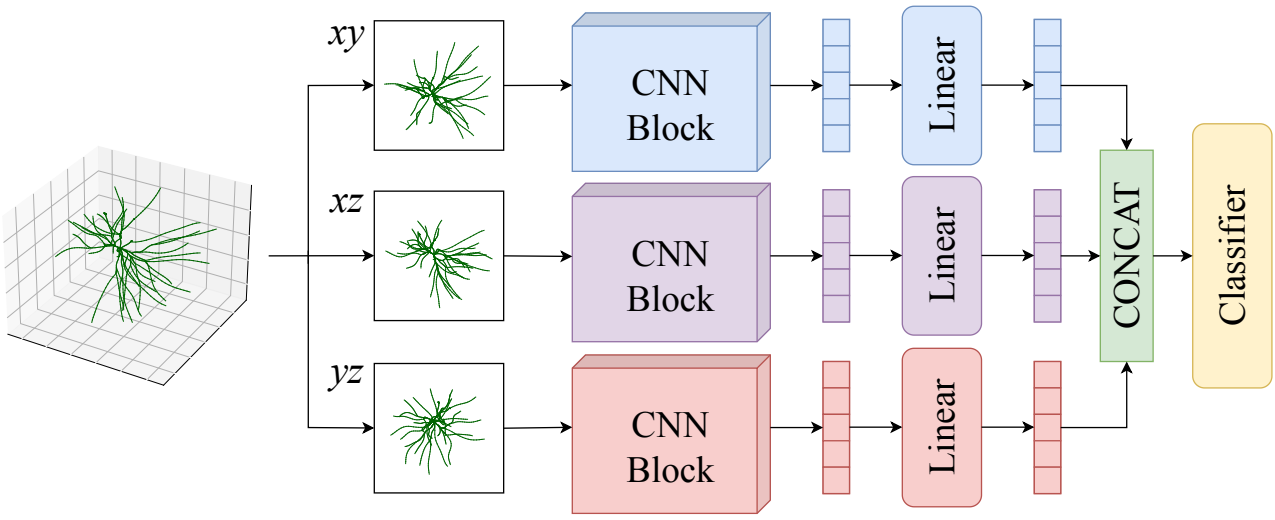


Figure 12: The overview of the pipeline for the discrimination test.

descriptors for samples, where the so-called density map is essentially the projection of a sample's 3D morphology onto a specific coordinate plane.

Learning Objective. We adopt the **Focal-Loss** as the learning objective for training the classifier, which is widely-used in classification tasks. It is a dynamically scaled Cross-Entropy-Loss, where the scaling factor decays to zero as confidence in the correct class increases (Lin et al., 2017).

Pipeline Overview. We demonstrate the whole pipeline in Figure 12.

Hyper Parameter Selection. We search the optimal hyper-parameters by ranging learning rate over $\{1e-3, 5e-3, 1e-4, 5e-4, 1e-5, 5e-5\}$ and dropout rate over $\{0.2, 0.5, 0.8\}$. Adam optimizer is adopted to optimize learnable parameters.

H.4. Supplementary Descriptions for Section 4.4

The evaluation process is based on the NEURON simulator[5] (Hines & Carnevale, 1997; 2001), utilizing the method of controlled variables for experimentation. In terms of simulation parameter settings, this experiment initializes the same electrophysiological parameters for all neurons, including ion channel parameters, and environmental parameters. Throughout the simulation process, the same current stimulation and voltage recording schemes are employed. In this experiment, to make neuronal physical morphology more realistic, diameter tapering is applied to all neurons before simulation, which will directly influence conductance of the dendritic fiber, thereby enhancing the objectivity of electrophysiological response results.

After neuronal morphology generation, a set of electrophysiological parameter initialization will be applied to all neuronal morphologies. Table 8 presents specific information regarding the properties and dynamic mechanisms for different parts of neuronal morphologies in this experiment. To enable the conduction of electrical signals in neurons, passive properties such as membrane capacitance C_m and axial resistance R_a need to be set in the whole neuron. Due to the current leakage on the membrane, parameters of leak potential E_L and leak resistance R_m need to be set, which can be used, combining the fiber diameters, to calculate other properties of the cell, such as resting potential, membrane time constant, and membrane input resistance. Next step is the setting of ion dynamics mechanisms in different parts of neurons to generate action potentials. All ion dynamic models based on the Hodgkin-Huxley formalism, and specific ion types and conductance distribution (Kole et al., 2006) are shown in Table 8. Since the membrane potential is mainly influenced by Na^+ and K^+, dynamic models related to these two types of ion are primarily selected. Similarly, Na^+ leakage potential E_{Na^+} and K^+ leakage potential E_{K^+} will also be set to simulate the activation of ion channels under different cellular states. As calcium-activated potassium channel is also important in the cell membrane, a dynamic model related to Ca^{2+} will be

[5] https://www.neuron.yale.edu/neuron/

introduced. Regarding environmental parameters, this experiment will simulate the environment at 34 °C to mimic the vivo conditions of experimental organisms at room temperature (21 °C). All ion channel models in Table 8 can be downloaded from ModelDB[6]. And all parameter settings of both ion channel and simulation environment have been validated in reported experimental data (Markram et al., 2015) where the electrophysiological parameters of simulated neurons are close to real neurons.

During the simulation, it is necessary to design a stimulation protocol that induces action potentials with moderate frequency in all neurons. On the other hand, a membrane potential recording protocol needs to be designed for overall assessment of neuronal electrophysiological responses. Here, stimulation current is injected in soma for each neuron and one of electrophysiological results is shown in Figure 8. The entire electrophysiological process consists of three phases. Both Phase I and Phase III represent the process of recovery to the resting potential. The difference lies in that Phase I demonstrates a change from the initial potential, while Phase III is the process starting from the potential at the end of the excitation phase which is Phase II. To ensure that the resting potential of neurons can be maintained at approximately be approximately -70 mV (a typical value for neuronal resting potential), before receiving stimulation (Phase I in Figure 8), a hyperpolarizing current is applied to the neurons, which aims to compensates for a liquid junction potential correction of about -10 mV in the simulation environment. Once the neuronal potential stabilizes, a depolarizing current is then applied to excite the neurons and generate regular action potentials (Phase I in Figure 8). Figure 13 and Figure 14 present the stimulation protocol where all currents are stimulated in soma, as well as the amplitude and duration of both hyperpolarizing current and depolarizing current are different. The amplitude settings of hyperpolarizing and depolarizing currents are based on biological experiments (Le Bé et al., 2007), and manual tuning to produce clear observable potential waveforms for all neurons. To observe electrophysiological results across different parts of the entire neuron, potential recordings are performed at all dendritic fiber terminals, as shown in Figure 13 and Figure 8, which facilitates a comprehensive evaluation of the similarity in neuronal electrophysiology.

Diameter tapering is employed before mentioned simulation to provide neurons with more realistic morphology. When physically modeling, the neuron is segmented into a series of cylindrical compartments based on the diameter value at each node. After diameter modifying, the compartments comprising the neuron transform from uniform thickness to varying thickness. The diameter tapering rule employed follows the guidance of TREES toolbox (Cuntz et al., 2010) and is calculated based on dendritic fiber length and branching positions. Firstly, the length of path from each dendritic terminal to soma is calculated. According to the relationship between dendritic length and the diameter tapering function, corresponding normalized quadratic attenuation curve coefficients p_1, p_2 and p_3 are determined. Additionally, a unified scaling factor *scale* and offset value *offset* are manually set. The diameter value for each point on each path can be calculated as follows:

$$\textit{diameter} = \textit{scale} \cdot (p_1 x^2 + p_2 x + p_3) + \textit{offset}, \tag{34}$$

where x represents the distance along the path between the node and the soma. When a node is situated in different paths, such as branching points, the diameter value at that node is averaged based on the results in different paths. The final effect of an exemplar neuronal morphology is shown in Figure 13, where dendritic fibers closer to the soma are thicker, and diametesr of nodes closer to dendritic terminals are smaller.

Table 8: Details of neuronal physiology.

Properties and Ion Channels	Location	Conductance Distribution	Key Parameters
C_m and R_a	whole neuron	uniform distribution	$C_m = 1$ μF/cm^2, $R_a = 100$ Ω·cm
E_L and R_m	soma	uniform distribution	$E_L = -67.129$ mV, $R_m = 10^4$ Ω·cm^2
E_L and R_m	dendrites	uniform distribution	$E_L = -60.296$ mV, $R_m = 10^6$ Ω·cm^2
transient Sodium (NaTa_t) (Colbert & Pan, 2002)	whole neuron	uniform distribution	$E_{Na^+} = 50$ mV, $E_{K^+} = -85$ mV
Kv3.1 (SKv3_1) (Rettig et al., 1992)	whole neuron	uniform distribution	
persistent Sodium (Nap_Et2) (Magistretti & Alonso, 1999)	whole neuron	uniform distribution	
h-current (Ih) (Kole et al., 2006)	dentrites	exponential distribution	
m-current (Im) (Adams et al., 1982)	whole neuron	uniform distribution	
persistent potassium (K_Pst) (Korngreen & Sakmann, 2000)	whole neuron	uniform distribution	
transient potassium (K_Tst) (Korngreen & Sakmann, 2000)	whole neuron	uniform distribution	
SK calcium-activated potassium (SK_E2) (Köhler et al., 1996)	soma	uniform distritbuion	
high voltage-activated calcium (Ca) (Reuveni et al., 1993)	soma	uniform distribution	$E_{Ca^{2+}} = 132$ mV
low voltage-activated calcium (Ca_LVAst) (Avery & Johnston, 1996)	soma	uniform distribution	
dynamics inside calcium concentration (CaDynamics_E2) (Destexhe et al., 1994)	soma	uniform distribution	

[6] http://senselab.med.yale.edu/ModelDB

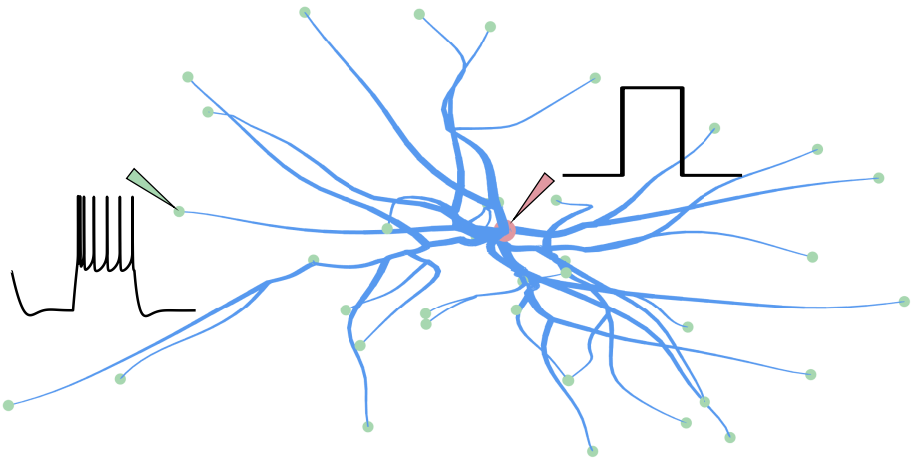


Figure 13: Illustration of a neuronal morphology after diameter tapering. We highlight the positions for current stimulation and potential detection (the soma is marked in red, dendritic terminals are marked in green). The stimulation current waveform and one recorded potential result by injected probes are also demonstrated.

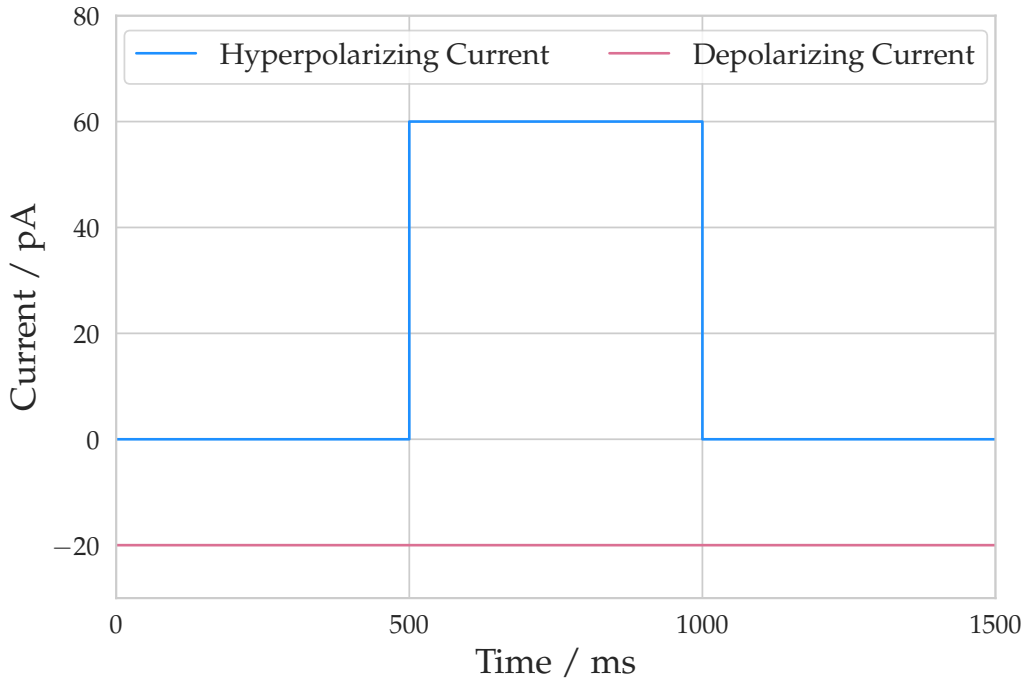


Figure 14: Demonstration of two types of long somatic current injection that determine three phases in the electrophysiological recording.

Table 9: **Datasets Statistics.** "#train/#valid/#test" denotes the number of samples in the training/validation/test set, respectively. "avg. / max # of branches" represents the average and maximum number of branches that a neuron contains respectively.

Dataset	#train	#valid	#test	avg. / max # of branches
VPM	266	57	57	69.46/153
RGC	534	114	115	303.47/2327
M1-EXC	192	41	42	65.77/156
M1-INH	259	55	57	307.65/913

I. More Dataset Information

The statistics of the four datasets are reported in Table 9 and their download links are as follows:

- **VPM**: https://download.brainimagelibrary.org/biccn/zeng/luo/fMOST/
- **RGC**: https://osf.io/b4qtr/wiki/home/
- **M1-EXC & M1-INH**: https://github.com/berenslab/mini-atlas

Remark. Axons in the VPM dataset typically have long-range projections, which can be dozens of times longer than other branches, and they target specific brain regions that they must innervate. Therefore, simultaneous modeling of the axon and dendrite of the VPM dataset is not appropriate, and we focused solely on generating the dendrite part.

J. More Experiment Results

Due to the limited pages for main paper, we present more experimental results in this section. Recall that we use '**MorphGrower**†' to represent the version that does not require the provision of soma branches, while '**MorphGrower**' represents the version where we directly provide soma branches.

J.1. Ablation Study

We conduct additional experiments to analyze the contributions of various model components to the overall performance. This included evaluating the effects of removing the local condition (Eq. 6) or the global condition (Eq. 4), as well as replacing the Exponential Moving Average (EMA) (Eq. 5) with a simpler Mean Pooling of ancestor branch representations for our local condition.

Table 10 empirically indicates that each proposed technique has effectively enhanced the performance of our model. Additionally, we experimented with replacing LSTM with Transformer and found that performance improvement was not significant, suggesting that LSTM is already sufficiently powerful for this task.

J.2. Quantitative Results on Morphological Statistics Excluding Soma Branches

Recall that **MorphGrower** represents the version where we choose to directly give the soma branches instead of also generating them. Section 4.2 evaluates methods in terms of the overall neuronal morphologies, thus including the given reference soma branches when evaluating the performance of our method. Here, we report the quantitative results the with soma branches excluded.

Remark. Here, given a neuronal morphology T, we delete all edges on soma branches. Then the morphology T is decomposed into several isolated nodes and several subtrees. Each root of the subtrees is the ending point of the original soma branches in T. Now, we regard each subtree as a new single neuronal morphology. Notice that the branches in the 1-th layer of the original T now turn to new soma branches in those substrees.

Table 11 summarizes the quantitative results excluding the soma branches. We see that the morphologies generated by MorphGrower also well fits all six selected quantitative characterizations of given reference data.

Table 10: The result of ablation study on the four datasets by six quantitative metrics. The best and the runner-up in each columns are highlighted in **bold** and underline respectively. We leave MorphVAE's numbers on MBPL and MAPS blank because it may generate nodes with more than two subsequent branches that conflict with the definition of MBPL and MAPS for bifurcations. A closer alignment with *Reference* indicates better performance.

Dataset	Method	MBPL / μm	MMED / μm	MMPD / μm	MCTT / %	MASB / °	MAPS / °
VPM	*Reference*	51.33 ± 0.59	162.99 ± 2.25	189.46 ± 3.81	0.936 ± 0.001	65.35 ± 0.55	36.04 ± 0.38
	MorphVAE	41.87 ± 0.66	126.73 ± 2.54	132.50 ± 2.61	0.987 ± 0.001	——	——
	MorphGrower	**48.29 ± 0.34**	161.65 ± 1.68	180.53 ± 2.70	0.920 ± 0.004	72.71 ± 1.50	**43.80 ± 0.98**
	LSTM → Transformers	47.40 ± 0.88	**162.46 ± 3.82**	**180.87 ± 3.09**	0.943 ± 0.010	**70.94 ± 2.77**	53.86 ± 0.99
	- Local Condition	40.90 ± 0.79	137.47 ± 2.63	162.53 ± 3.24	0.911 ± 0.006	74.55 ± 0.88	58.22 ± 0.32
	- Global Condition	44.51 ± 0.78	153.84 ± 3.56	173.58 ± 5.41	0.938 ± 0.003	71.29 ± 3.56	47.06 ± 0.59
	- EMA	45.24 ± 0.22	155.11 ± 1.98	173.68 ± 2.86	**0.936 ± 0.005**	67.79 ± 1.45	46.28 ± 0.08
RGC	*Reference*	26.52 ± 0.75	308.85 ± 8.12	404.73 ± 12.05	0.937 ± 0.003	84.08 ± 0.28	50.60 ± 0.13
	MorphVAE	43.23 ± 1.06	248.62 ± 9.05	269.92 ± 10.25	0.984 ± 0.004	——	——
	MorphGrower	**25.15 ± 0.71**	306.83 ± 7.76	384.34 ± 11.85	**0.945 ± 0.003**	82.68 ± 0.53	51.33 ± 0.31
	LSTM → Transformers	25.10 ± 0.65	**308.35 ± 7.34**	**387.67 ± 10.55**	0.948 ± 0.003	**84.04 ± 0.33**	52.35 ± 0.14
	- Local Condition	23.56 ± 0.74	294.01 ± 8.21	363.86 ± 11.36	0.954 ± 0.003	79.67 ± 1.17	54.44 ± 0.36
	- Global Condition	22.99 ± 0.83	293.87 ± 9.01	354.95 ± 11.85	0.954 ± 0.006	78.19 ± 4.10	**50.96 ± 0.63**
	- EMA	23.38 ± 0.66	295.09 ± 8.76	359.76 ± 8.76	0.951 ± 0.005	78.47 ± 1.84	52.25 ± 0.44
M1-EXC	*Reference*	62.74 ± 1.73	414.39 ± 6.16	497.43 ± 12.42	0.891 ± 0.004	76.34 ± 0.63	46.74 ± 0.85
	MorphVAE	52.13 ± 1.30	195.49 ± 9.91	220.72 ± 12.96	0.955 ± 0.005	——	——
	MorphGrower	**58.16 ± 1.26**	**413.78 ± 14.73**	**473.25 ± 19.37**	**0.922 ± 0.002**	73.12 ± 2.17	**48.16 ± 1.00**
	LSTM → Transformers	56.75 ± 1.49	415.90 ± 4.39	472.30 ± 7.99	0.942 ± 0.005	72.97 ± 1.75	51.06 ± 0.98
	- Local Condition	55.85 ± 1.24	409.66 ± 7.36	464.18 ± 9.07	0.940 ± 0.004	**73.81 ± 1.24**	51.54 ± 0.84
	- Global Condition	55.01 ± 0.65	404.42 ± 8.67	453.58 ± 11.90	0.955 ± 0.007	71.72 ± 1.23	48.48 ± 1.04
	- EMA	55.71 ± 1.24	407.29 ± 16.28	458.49 ± 12.29	0.951 ± 0.008	72.61 ± 4.35	50.20 ± 0.83
M1-INH	*Reference*	45.03 ± 1.04	396.73 ± 15.89	705.28 ± 34.02	0.877 ± 0.002	84.40 ± 0.68	55.23 ± 0.78
	MorphVAE	50.79 ± 1.77	244.49 ± 15.62	306.99 ± 23.19	0.965 ± 0.002	——	——
	MorphGrower	**41.50 ± 1.02**	**389.06 ± 13.54**	**659.38 ± 30.05**	**0.898 ± 0.002**	82.43 ± 1.41	61.44 ± 4.23
	LSTM → Transformers	40.55 ± 0.82	378.98 ± 12.21	645.68 ± 28.83	0.903 ± 0.003	**84.32 ± 0.85**	59.89 ± 0.91
	- Local Condition	39.33 ± 1.02	383.21 ± 10.06	641.00 ± 23.99	0.918 ± 0.003	77.30 ± 10.86	60.53 ± 0.75
	- Global Condition	38.12 ± 0.26	372.66 ± 10.30	613.76 ± 33.09	0.929 ± 0.003	78.29 ± 3.19	**57.21 ± 0.48**
	- EMA	38.97 ± 0.82	383.77 ± 14.04	636.61 ± 40.45	0.921 ± 0.004	80.09 ± 3.24	58.77 ± 0.91

Table 11: Quantitative Results of our MorphGrower on the VPM, RGC, M1-EXC and M1-INH datasets in terms of six quantitative metrics excluding soma branches.

Dataset	Method	MBPL / μm	MMED / μm	MMPD / μm	MCTT / %	MASB / °	MAPS / °
VPM	*Reference*	59.66 ± 0.72	117.74 ± 1.19	133.71 ± 0.71	0.928 ± 0.001	74.93 ± 1.15	29.49 ± 0.58
	MorphGrower	55.53 ± 0.58	116.17 ± 1.67	125.72 ± 1.78	0.920 ± 0.005	82.96 ± 2.01	36.00 ± 1.13
RGC	*Reference*	27.03 ± 0.81	212.73 ± 5.61	276.53 ± 7.72	0.936 ± 0.001	141.09 ± 1.47	49.36 ± 0.67
	MorphGrower	25.56 ± 0.80	210.87 ± 5.34	261.60 ± 7.30	0.945 ± 0.001	138.23 ± 2.30	50.08 ± 0.90
M1-EXC	*Reference*	67.43 ± 1.81	168.29 ± 3.07	198.03 ± 3.03	0.887 ± 0.004	68.39 ± 2.64	30.56 ± 1.25
	MorphGrower	61.79 ± 1.23	166.68 ± 3.93	184.85 ± 4.12	0.930 ± 0.003	68.83 ± 1.91	31.84 ± 1.42
M1-INH	*Reference*	58.54 ± 1.88	181.32 ± 7.84	254.79 ± 11.23	0.888 ± 0.002	89.06 ± 4.16	36.39 ± 2.01
	MorphGrower	54.28 ± 1.62	177.74 ± 7.33	237.60 ± 9.99	0.916 ± 0.004	88.37 ± 3.59	41.78 ± 4.92

J.3. Plausibility Evaluation for the Samples Generated without Soma Branches Given

We conducted an additional evaluation experiment regarding the "plausibility" of generated morphology samples without soma branches directly given. The results are presented in Table 12. Based on the table, it is evident that MorphGrower and MorphGrower† perform similarly in this evaluation, making it challenging to decisively determine which one exhibits superior performance. However, both MorphGrower and MorphGrower† consistently generate samples with higher

Table 12: Classification accuracy (%). The closer accuracy to 50% indicates higher generation plausibility. The best and the runner-up plausible in each columns are highlighted in **bold** and underline respectively.

Method \ Dataset	VPM	RGC	M1-EXC	M1-INH
MorphVAE	86.75 ± 06.87	94.60 ± 01.02	80.72 ± 10.58	91.76 ± 12.14
MorphGrower	**54.73 ± 03.36**	62.70 ± 05.24	**55.00 ± 02.54**	**54.74 ± 01.63**
MorphGrower†	59.47 ± 04.53	**54.99 ± 03.08**	55.26 ± 02.42	55.47 ± 03.43

plausibility compared to the baseline MorphVAE.

J.4. Sensitivity to the embedding size

In this section, we investigate the sensitivity of our method and the baseline MorphVAE to the embedding size.

Figure 15 illustrates the performance of baseline MorphVAE and our MorphVAE on six metrics used in Section 4.2 as the embedding size varies. We selected three embedding sizes: $\{32, 64, 128\}$. It can be observed that MorphVAE's performance on these four embedding size choices is inferior to MorphGrower with the same embedding size.

MorphGrower shows an improvement in performance in metrics such as MBPL, MMED, MMPD, and MCTT as the embedding size increases, but then it decreases as the embedding size further increases. This might be due to the limited amount of data, making it challenging to fit a larger model. On the other hand, in metrics like MASB and MAPS, the model's performance improves as the embedding size increases for these four different choices. This is likely because larger models help capture complex patterns and relationships related to angle correlations in the data.

J.5. Evaluation in terms of FID and KID

FID (Fréchet Inception Distance) and KID (Kernel Inception Distance) are commonly used as evaluation metrics for image generation methods. However, it's important to note that they typically require feature extraction from samples before computation, which is often done using pre-trained networks like ResNet (He et al., 2016) in the realm of computer vision. Yet, neuronal morphology generation isn't an image generation task, and there aren't any pre-trained extractors available for neuron morphology data on extensive datasets. Therefore, strictly speaking, we cannot provide convincing results based on FID and KID.

Nevertheless, we still try our best to present some results from the perspectives of FID and KID. Recalling our main content in Section 4.3, we utilize the network architecture displayed in Figure 12 to train a real *vs.* fake classifier to evaluate the plausibility of generated samples. Here, we continue to consider leveraging computer vision techniques to extract features for morphologies. Please refer to Appendix H.3 for our rationale behind this approach. We adopt the network architecture from Figure 12 to train a feature extractor. Initially, we set up an experiment to further verify if such an architecture indeed can learn useful representations: As neurons from different categories (i.e., from various datasets) have different morphological features, we combine real samples from the four datasets. Using the dataset as a label, our goal is to train a ResNet18-based model for classification. We conduct this experiment with diverse data splits five times at random, and the mean and variance of the classification accuracy across these results are 0.9188 and 0.0220, respectively. Clearly, this model can acquire useful representations that generalize to unseen data.

Subsequently, we employ the newly trained ResNet18-based model as a feature extractor to compute results on both KID and FID metrics. Table 13 showcases the results in terms of KID and FID.

All results were averaged from five experiments. From the results, we can observe that whether it's the KID or FID metric, our method significantly outperforms the baseline MorphVAE across all datasets. Because we adopted the evaluation method of image generation here, it further confirms that the samples generated by our method are visually closer to real samples compared to those produced by the baseline MorphVAE.

K. Notations

We summarize the notations used in this paper in Table 14 to facilitate reading.

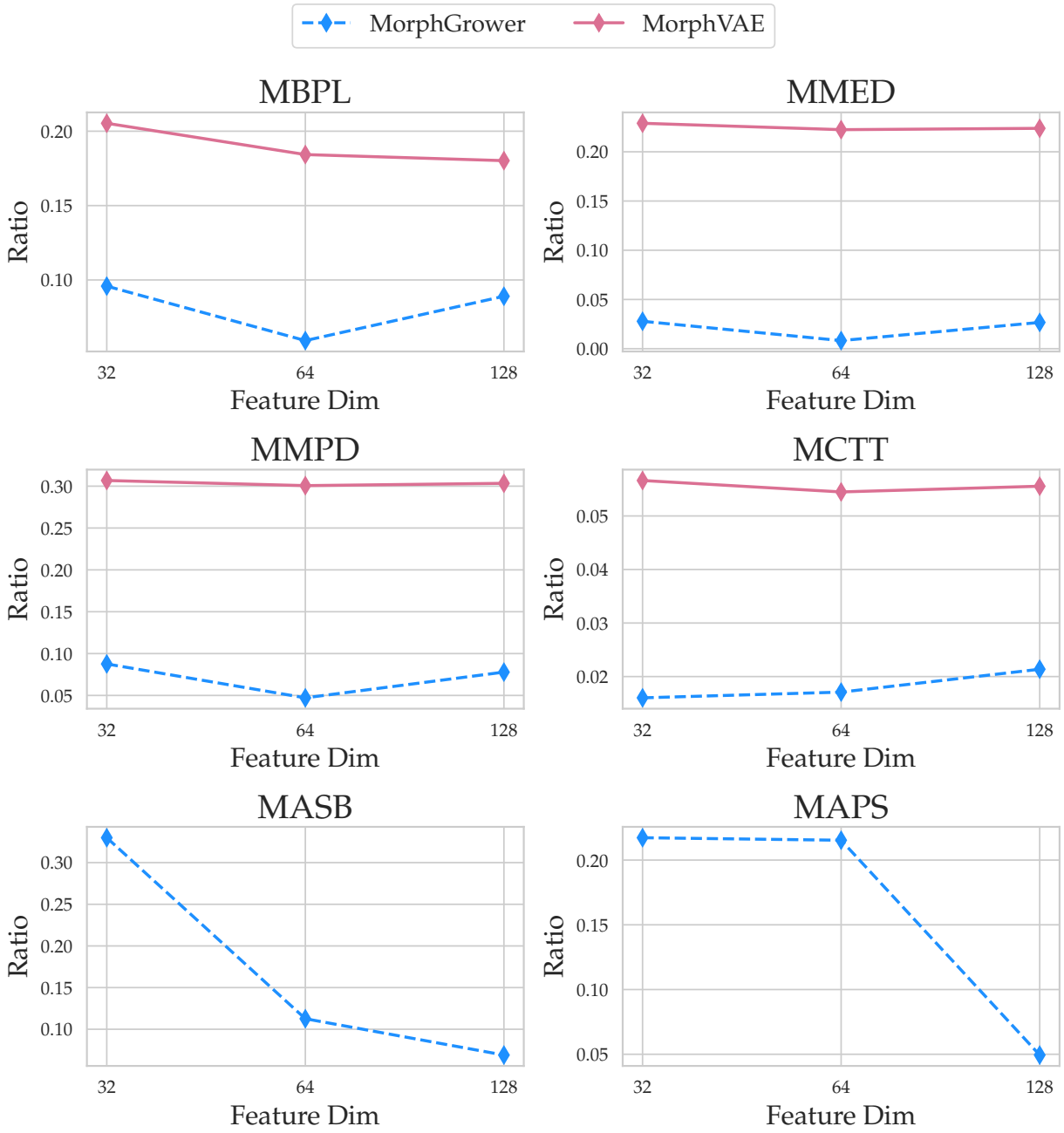


Figure 15: Performance curves of MorphVAE and MorphGrower as the embedding size varies, in terms of MBPL, MMED, MMPD, MCTT, MASB, and MAPS. The horizontal axis represents the embedding size, while the vertical axis indicates the ratio of the deviation from real data metrics to the real data metrics themselves. A lower ratio indicates better performance. We leave MorphVAE's curves on MBPL and MAPS blank because it may generate nodes with more than two subsequent branches that conflict with the definition of MBPL and MAPS for bifurcations.

Table 13: Evaluation of generation performance on the VPM, RGC, M1-EXC and M1-INH datasets in terms of FID and KID.

Metric	Method	VPM	RGC	M1-EXC	M1-INH
FID	**MorphVAE**	85.95 ± 5.13	208.42 ± 4.39	168.63 ± 9.82	1651.17 ± 91.51
	MorphGrower	**23.71 ± 5.06**	**106.16 ± 8.33**	**17.07 ± 5.73**	**62.37 ± 14.38**
KID	**MorphVAE**	5.83 ± 0.76	22.65 ± 1.84	7.86 ± 0.73	356.67 ± 22.07
	MorphGrower	**1.48 ± 0.34**	**9.42 ± 0.85**	**0.10 ± 0.12**	**19.87 ± 7.17**

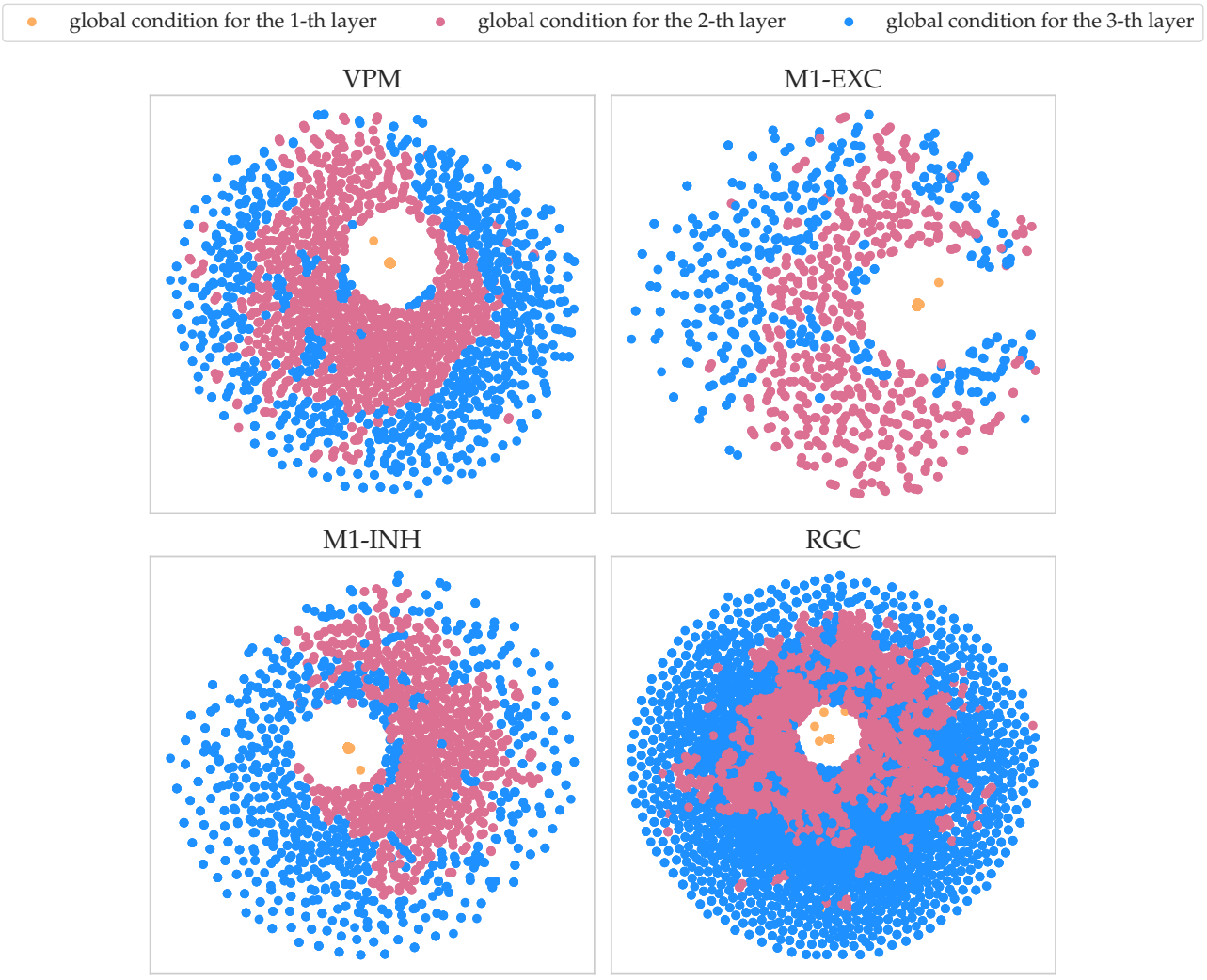


Figure 16: We use t-SNE to visualize the global conditions learned by MorphGrower on four datasets.

L. Visualization

In this section, we present more visualization results.

L.1. Visualization of the Learned Global Condition Using t-SNE

Figure 16 visualizes the global conditions learned by MorphGrower using t-SNE (Van der Maaten & Hinton, 2008). Since the morphologies in all four datasets typically include at least four layers of tree structure (requiring the extraction of global

conditions three times for generating four layers), in Figure 16, we chose to depict the global conditions extracted when generating the 1-th, 2-th, and 3-th layers (in our paper, layer indexing starts from 0). We can see that the global conditions for different layers can be well classified, indicating that MorphGrower can indeed capture certain patterns in the data. Therefore, the representations of the extracted global conditions are meaningful, which further confirms that MorphGrower can effectively model the influence of predecessor branches on subsequent branches.

L.2. Visualization of the Distribution of Four Metrics

We only plot the distributions of MPD, BPL, MED and CTT over the reference samples and generated morphologies on the VPM dataset in the main paper due to the limited space. Here, we plot the their distributions over the other three datasets in Figure 17, Figure 18 and Figure 19.

L.3. Illustration of the Layer-by-layer Generation

We demonstrate the complete sequence of the intermediate generated morphologies and the final generated morphology for the selected example from the RGC dataset in Sec. 4.5 in Fig. 21.

Besides, we also randomly pick a generated morphology from each dataset and demonstrate the sequence of the intermediate generated morphologies and the final generated morphology for each corresponding example in Figure 20, Figure 22 and Figure 23.

We see that MorphGrower can generate snapshots of morphologies in different growing stages. Such a computational recovery of neuronal dynamic growth procedure may be an interesting future research direction and help in further unravelling the neuronal growth mechanism.

L.4. Sholl Analysis

In this study, we utilize the Sholl analysis method (Sholl, 1953) to quantitatively evaluate the complexity of morphological differences between reference and generated neurons. The Sholl intersection profile of a specific neuron is represented as a one-dimensional distribution, which is derived by enumerating the number of branch intersections at varying distances from the soma. This analytical approach enables the quantitative comparison of morphologically distinct neurons and facilitates the mapping of synaptic contact or mitochondrial distribution within dendritic arbors (Lim et al., 2015). Moreover, the Sholl analysis has proven instrumental in illustrating alterations in dendritic morphology resulting from neuronal diseases (Beauquis et al., 2013), degeneration (Williams et al., 2013), or therapeutic interventions (Packer et al., 2013).

We conduct Sholl analysis on each neural morphology data in the testing set of each dataset to investigate dendritic arborization. Prior to analysis, we standardized each morphology data by translating the soma (root node) to the origin $(0, 0, 0)$. Thereafter, we generated 20 concentric spheres with the soma as the center such that they were equidistant from one another and spanned the maximal coordinate range. We assess the number of dendritic intersections between each concentric sphere and different neuron samples and depicted the results as a plot such that the average number of intersections was depicted on the y-axis and the radius of the concentric sphere on the x-axis. The findings are represented in Figure 24, Figure 25, Figure 26 and Figure 27. The visualized distribution indicates that our method closely approximates the real samples in the testing set, whereas the performance of MorphVAE significantly deviates from them.

L.5. More Visualization Results of Generated Morphologies

We present more visualizations of some neuronal morphologies generated by our proposed MorphGrower in Figure 28, Figure 29, Figure 30 and Figure 31. Owing to image size constraints, we were unable to render the 3D image with uniform scaling across all three axes. In reality, the unit scales for the x, y, and z axes in the 3D image differ, leading to some distortion and making the elongation in specific directions less noticeable. The 2D projection images feature consistent unit scaling for each axis, which explains why the observed neuronal morphology appears distinct from the 3D image.

Table 14: Notations.

Notation	Description
T	an neuronal morphology instance
$\hat{T}$	the generated morphology
$\hat{T}^{(i)}$	the top i layer subtree of the final generated morphology
V	the node set of a given neuronal morphology
E	the edge set of a given neuronal morphology
b_i	a branch instance
$\hat{b}_i$	a generated branch using b_i as reference
$\hat{v}_i$	a node instance on the generated morphology
b_{is}	the branch which starts from v_i and ends at v_s
v_i	the 3D coordinates of a node instance
N_v	the node number of a given neuronal morphology
$e_{i,j}$	the directed edge beginning from v_i and ending with v_j
$\mathcal{N}^+(b_i)$	two subsequent branches of b_i
$\mathcal{F}$	the forest composed of branch trees, serves as the global condition
$L(i)$	the number of nodes included in b_i
L_i	the branch collection of the i-th layer
$\hat{L}_i$	the branch collection of the generated morphology at the i-th layer
N^k	the number of branches at the k-th layer
m	the iteration number in Eq. 4
μ	the mean of the vMF distribution
κ	the variance of the vMF distribution
B	the branch pair input
C	the input for encoding the global condition and local contidion
$\mathcal{A}$	the sequence of ancestor branches
$\mathbf{D}_k^{\mathcal{A}}$	the feature extracted from the first k element of $\mathcal{A}$
$\mathbf{h}_i$	the internal hidden state of the LSTM network in encoder
$\mathbf{c}_i$	the internal cell state of the LSTM network in decoder
$\mathbf{h}_i'^{(t)}$	the internal hidden state of LSTM network in decoder
$\mathbf{c}_i'^{(t)}$	the internal hidden state of LSTM network in decoder
$\mathbf{x}_i$	the high dimension embedding projected by $\mathbf{W}_{in}$ in encoder
$\mathbf{y}_i$	the output of LSTM in decoder
$\mathbf{W}_{in}$, $\mathbf{W}'_{in}$	the linear transform that projects 3D coordinate into high dimension space in encoder, decoder
$\mathbf{W}_{sta_1}$, $\mathbf{W}_{sta_2}$	the linear transform to obtain the initial internal state for decoding the branch pair
$\mathbf{W}_{out}$	the linear transform that transforms y_i into 3D coordinate
$\mathbf{W}^{(k)}$	the linear transform in the k-th iteration of GNN while encoding global condition
$\mathbf{r}_b$	the encoded representation of branch b
$\mathbf{r}_{b_i}$	the encoded representation of a specific branch b_i
$\mathbf{h}_{b_i}^{(k)}$	the node feature of branch b_i which is located at k-th layer
k in Eq. 3	depth in the tree (starting from 0)
$\mathbf{h}_{local}$	the encoded representation of local condition
$\mathbf{h}_{global}$	the encoded representation of global condition
θ	the learnable parameters of the encoder
ϕ	the learnable parameters of the decoder
$f_\theta(\cdot)$	the encoder
$g_\phi(\cdot)$	the decoder
$\mathbf{z}$	the latent variable

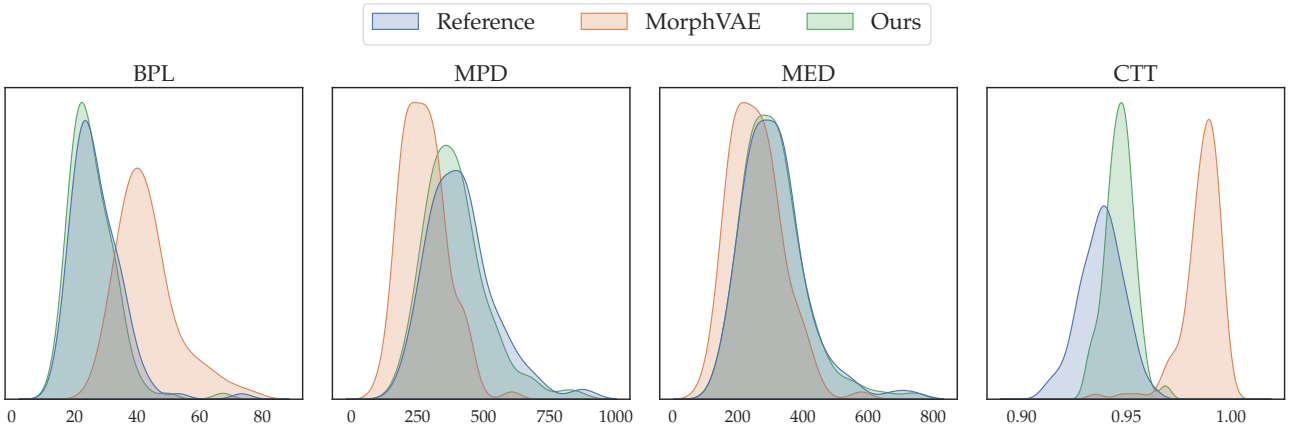


Figure 17: Distributions of the four morphological metrics: MPD, BPL, MED and CTT on the RGC dataset.

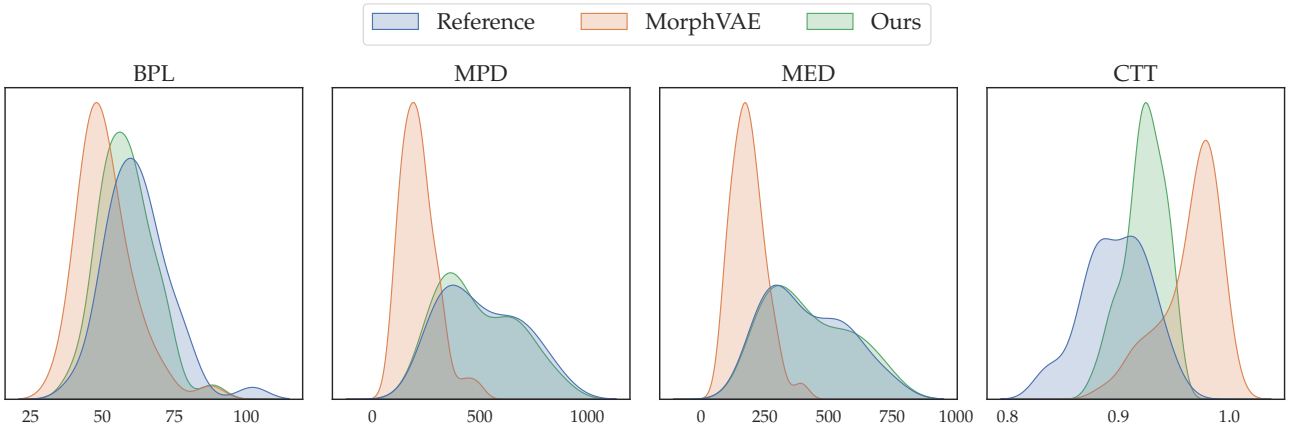


Figure 18: Distributions of the four morphological metrics: MPD, BPL, MED and CTT on the M1-EXC dataset.

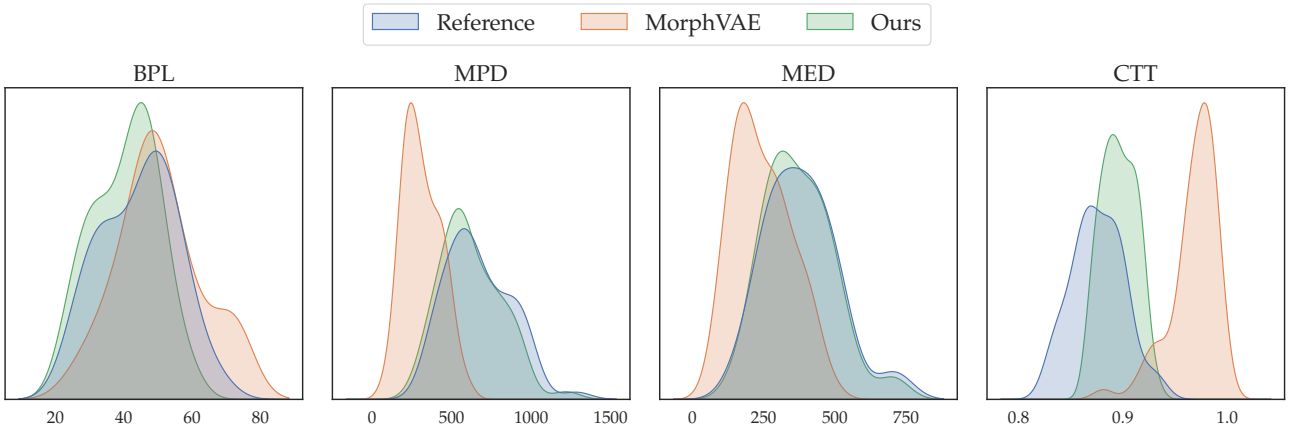


Figure 19: Distributions of the four morphological metrics: MPD, BPL, MED and CTT on the M1-INH dataset.

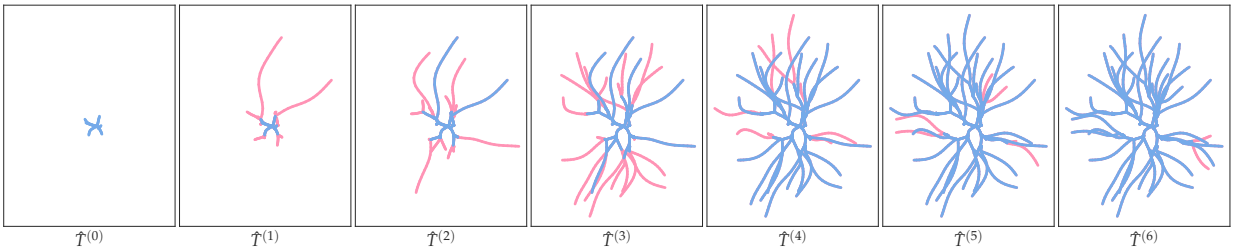


Figure 20: An example from the VPM dataset. We demonstrate the projections onto the xy plane of the sequence of the intermediate morphologies $\{\hat{T}^{(i)}\}_{i=0}^{5}$ and the final generated morphology $\hat{T}^{(6)}$.

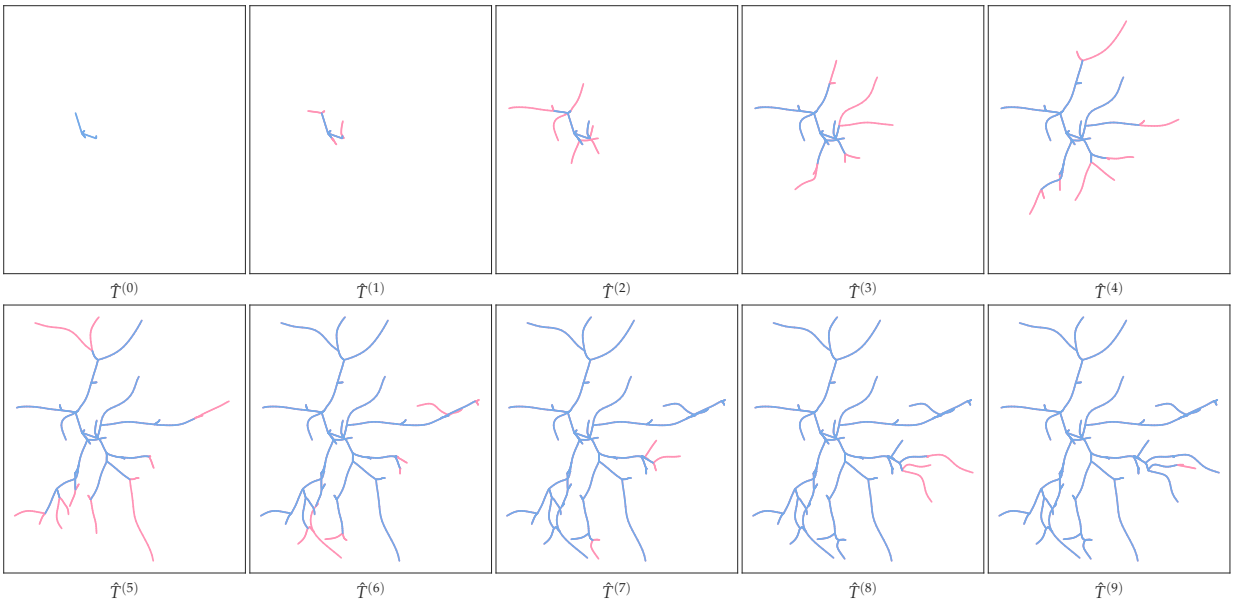


Figure 21: An example from the RGC dataset. We demonstrate the projections onto the xy plane of the sequence of the intermediate morphologies $\{\hat{T}^{(i)}\}_{i=0}^{8}$ and the final generated morphology $\hat{T}^{(9)}$.

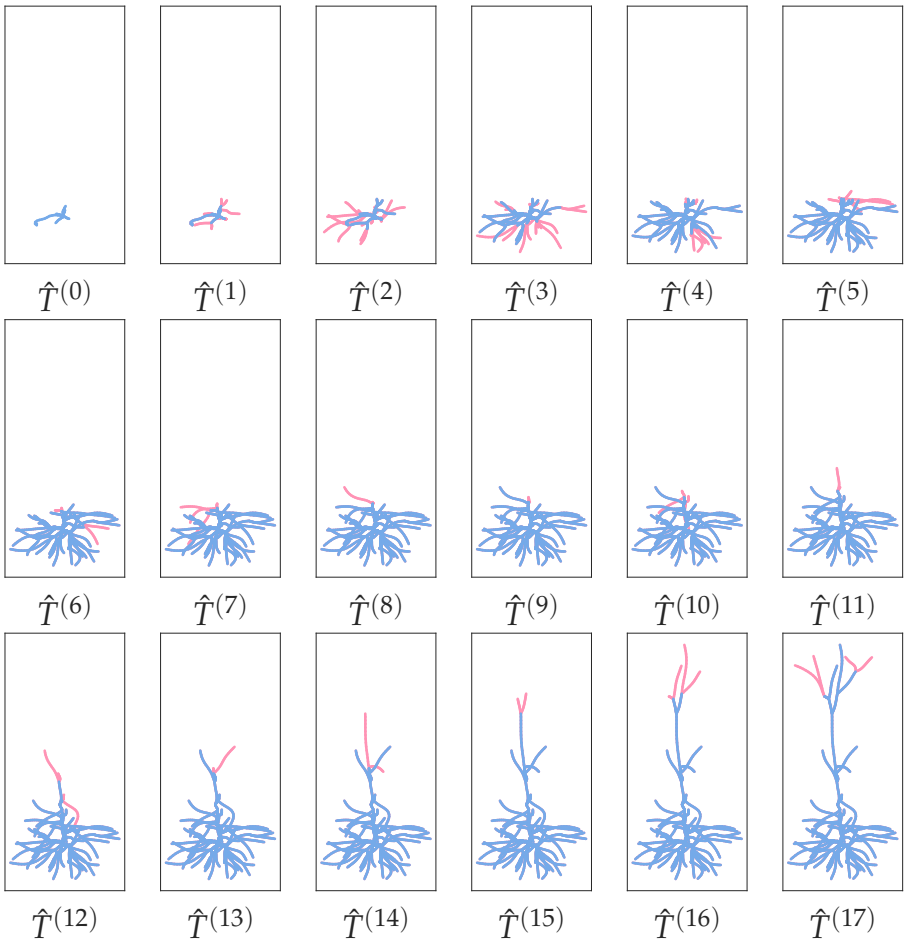


Figure 22: An example from the M1-EXC dataset. We demonstrate the projections onto the xy plane of the sequence of the intermediate morphologies $\{\hat{T}^{(i)}\}_{i=0}^{16}$ and the final generated morphology $\hat{T}^{(17)}$.

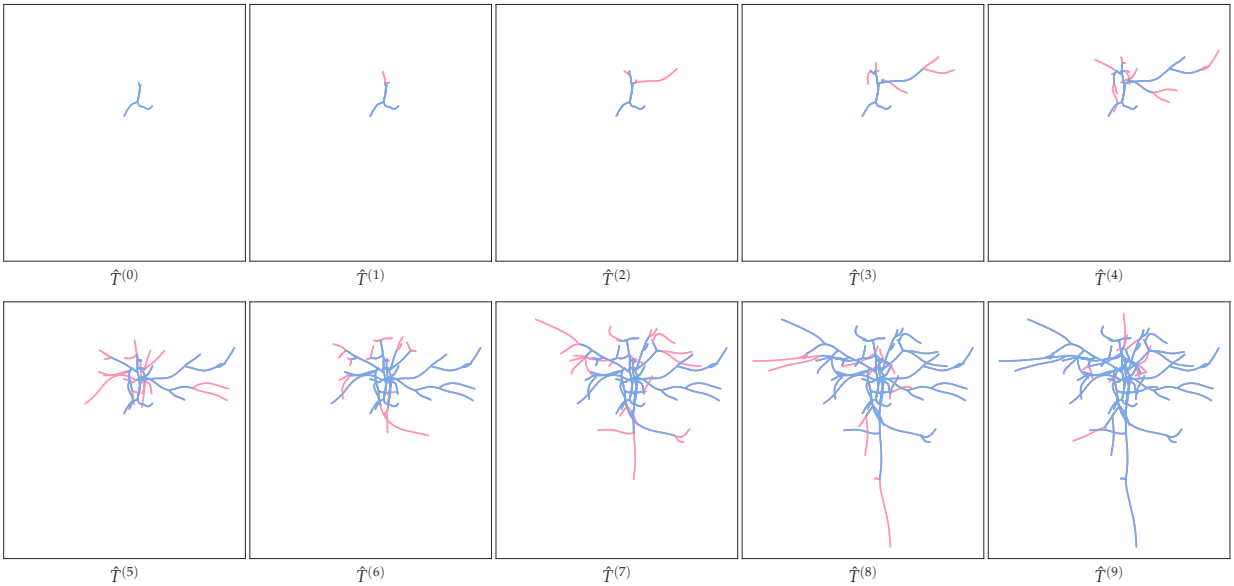


Figure 23: An example from the M1-INH dataset. We demonstrate the projections onto the xy plane of the sequence of the intermediate morphologies $\{\hat{T}^{(i)}\}_{i=0}^{8}$ and the final generated morphology $\hat{T}^{(9)}$.

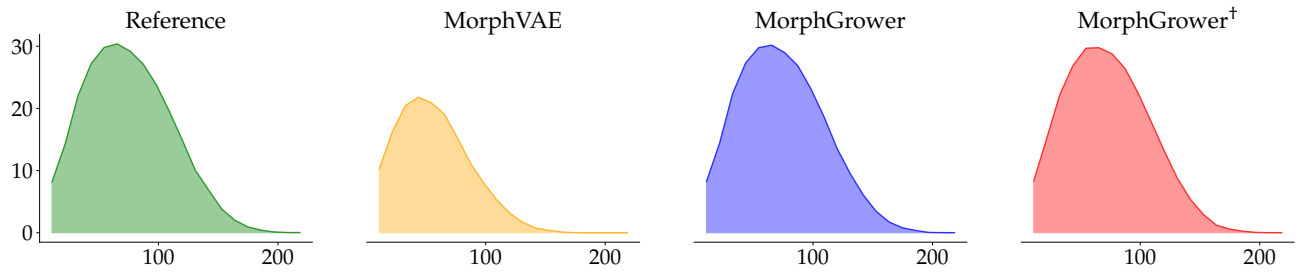


Figure 24: Visualized distribution on the VPM dataset. The horizontal axis represents the radius of the concentric spheres, and the vertical axis represents the average number of intersections between different neuron samples and the concentric spheres.

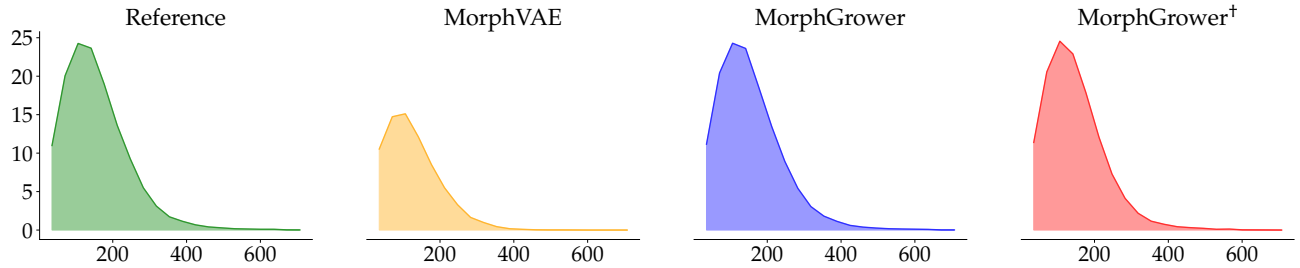


Figure 25: Visualized distribution on the RGC dataset. The horizontal axis represents the radius of the concentric spheres, and the vertical axis represents the average number of intersections between different neuron samples and the concentric spheres.

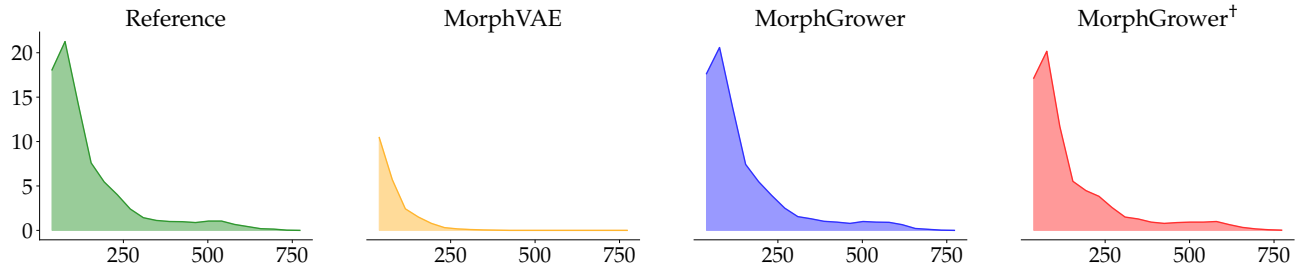


Figure 26: Visualized distribution on the M1-EXC dataset. The horizontal axis represents the radius of the concentric spheres, and the vertical axis represents the average number of intersections between different neuron samples and the concentric spheres.

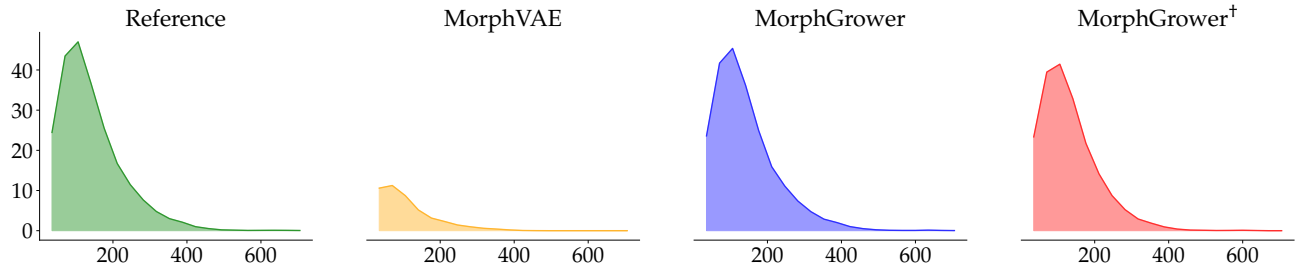


Figure 27: Visualized distribution on the M1-INH dataset. The horizontal axis represents the radius of the concentric spheres, and the vertical axis represents the average number of intersections between different neuron samples and the concentric spheres.

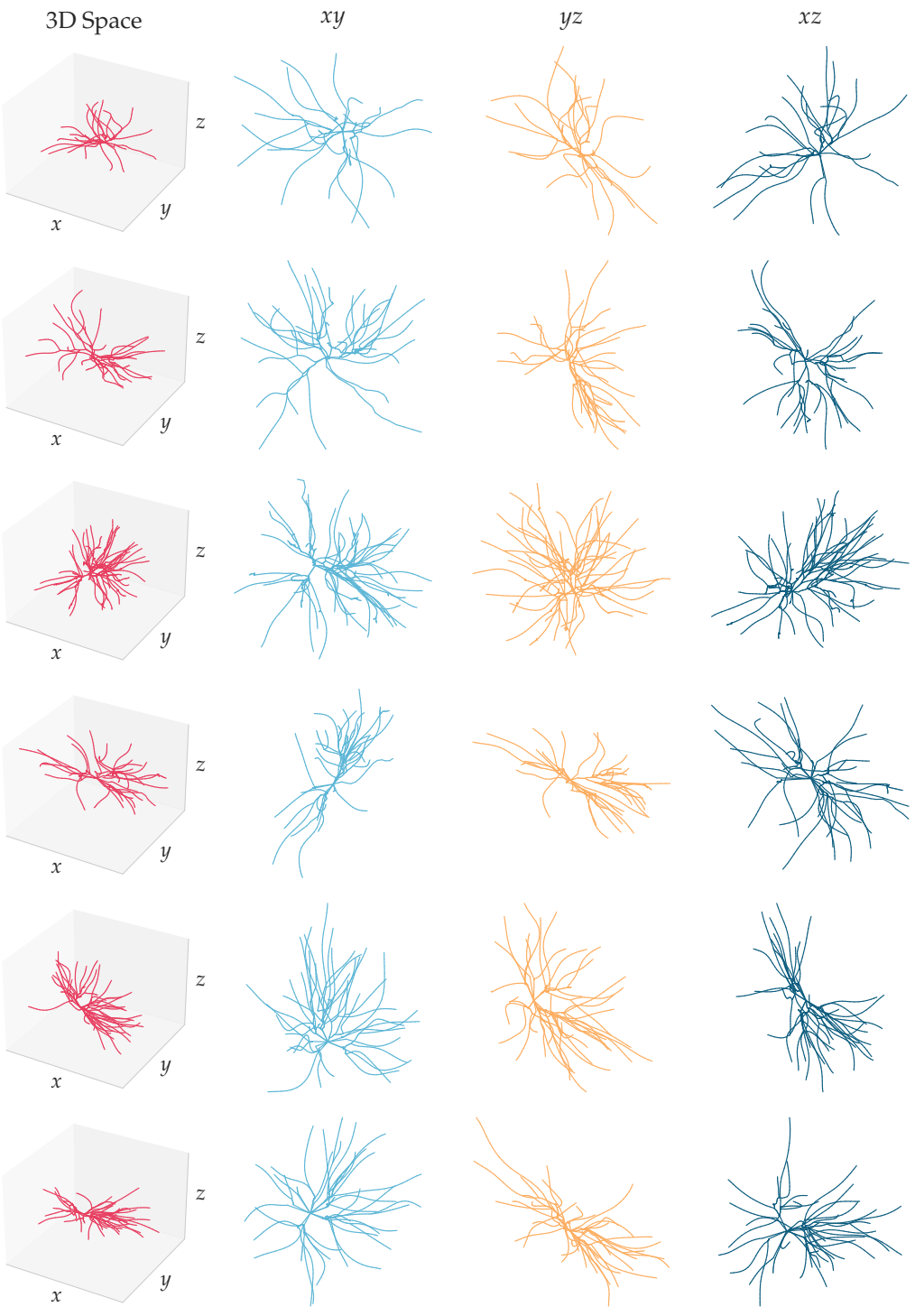


Figure 28: Visualization of some neuronal morphologies generated by **MorphGrower** over the **VPM** dataset. The first column demonstrates the morphologies in 3D space. The three columns (from left to right) on the right are the projections of morphology samples onto xy, yz and xz planes, respectively. A group of pictures on each row corresponds to a generated neuronal morphology.

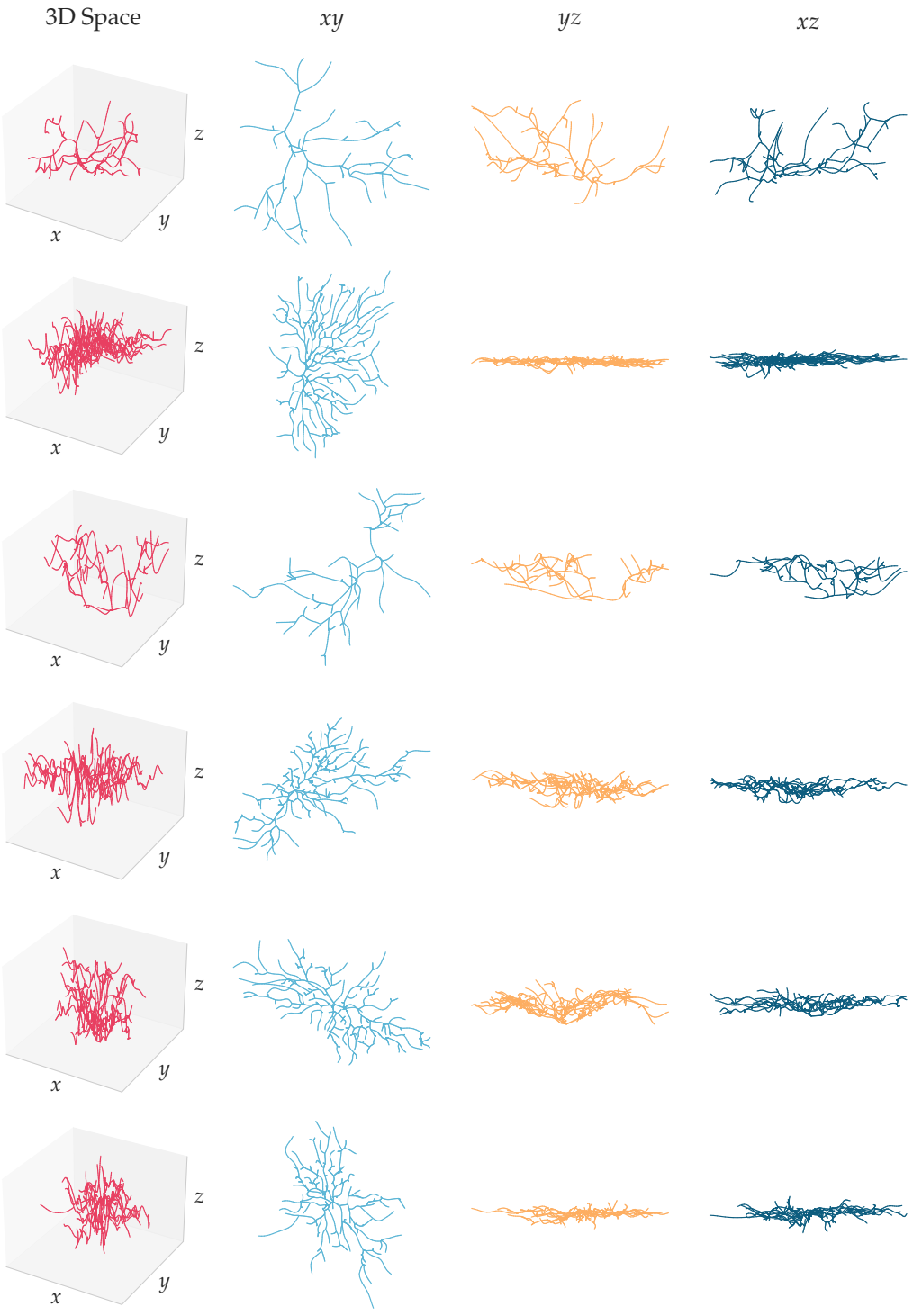


Figure 29: Visualization of some neuronal morphologies generated by **MorphGrower** over the **RGC** dataset. The first column demonstrates the morphologies in 3D space. The three columns (from left to right) on the right are the projections of morphology samples onto xy, yz and xz planes, respectively. A group of pictures on each row corresponds to a generated neuronal morphology.

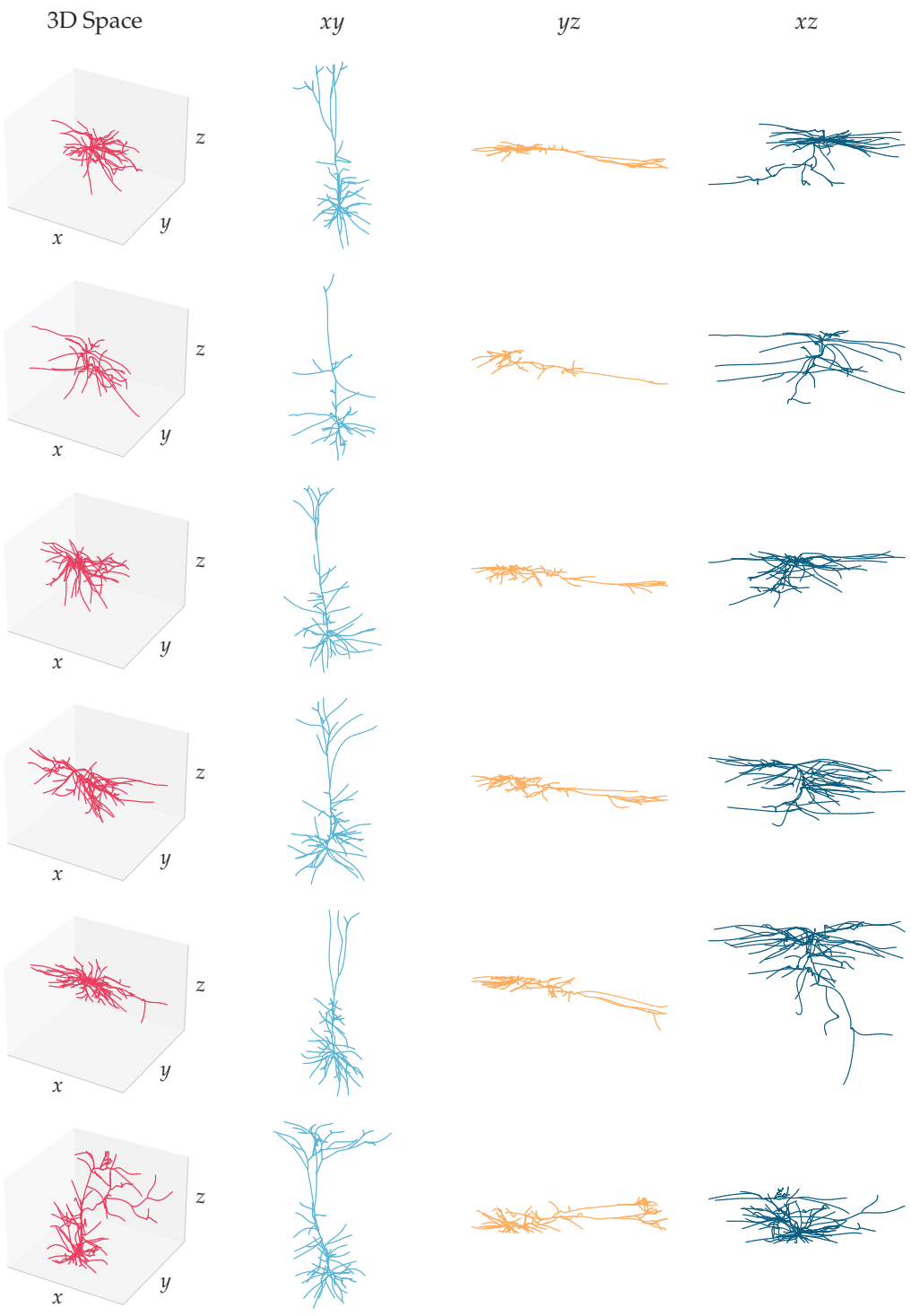


Figure 30: Visualization of some neuronal morphologies generated by **MorphGrower** over the **M1-EXC** dataset. The first column demonstrates the morphologies in 3D space. The three columns (from left to right) on the right are the projections of morphology samples onto xy, yz and xz planes, respectively. A group of pictures on each row corresponds to a generated neuronal morphology.

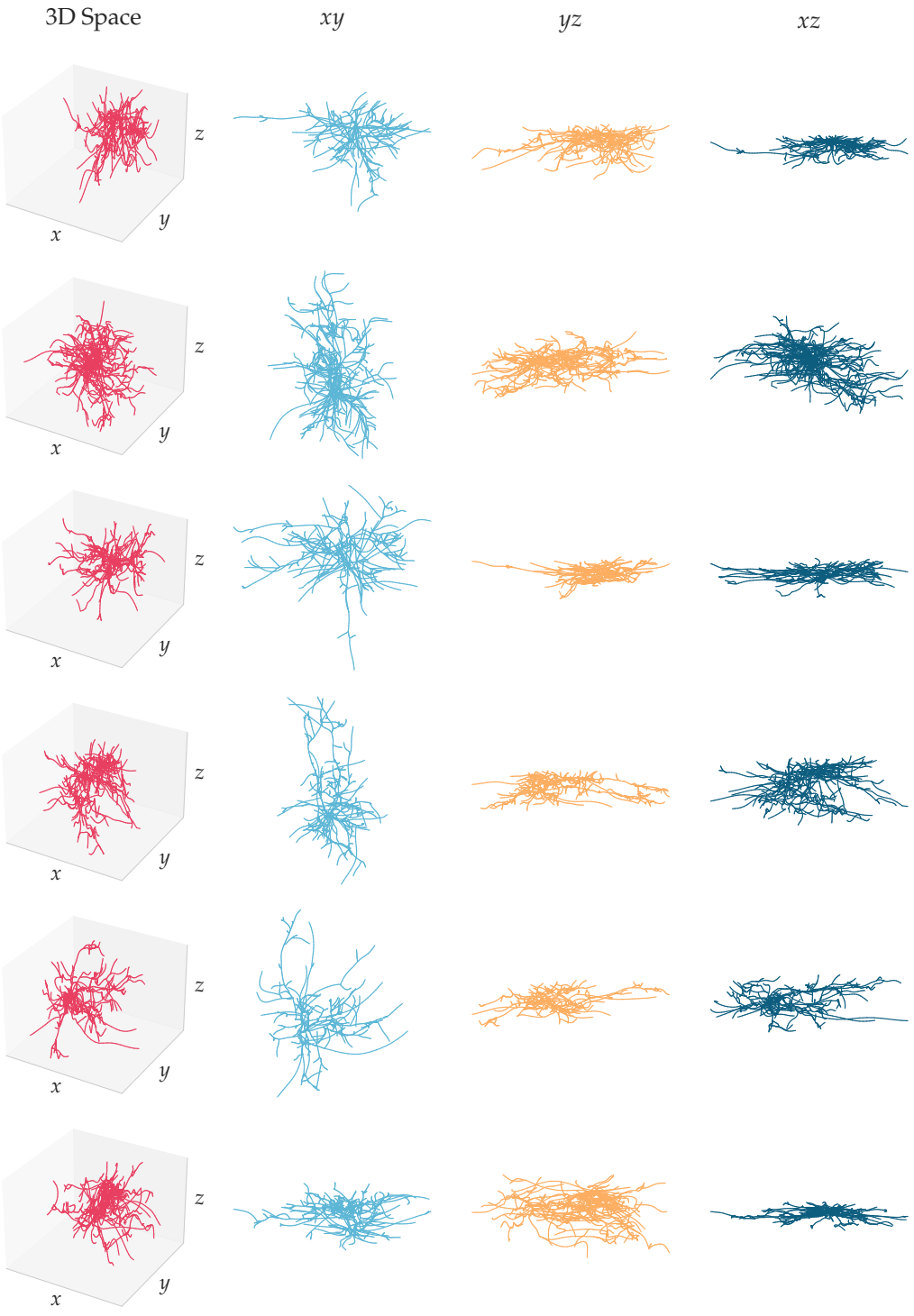


Figure 31: Visualization of some neuronal morphologies generated by **MorphGrower** over the **M1-INH** dataset. The first column demonstrates the morphologies in 3D space. The three columns (from left to right) on the right are the projections of morphology samples onto xy, yz and xz planes, respectively. A group of pictures on each row corresponds to a generated neuronal morphology.