AUTOREGRESSIVE FRONTIER EXPANSION: GROWING TREES WITH GRAPH MACHINE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Tree-like branching structures are common in nature, from botanical trees to neurons, blood vessels and respiratory trees. Their branching shape often reflects function, making structural modelling central to understanding how these systems work. Because acquiring real-world 3D data is often expensive or infeasible, realistic generative models are valuable for simulation and data augmentation. Existing morphology-specific models either constrain how topology is generated or rely on hand-tuned, mechanistic procedures. Generic 3D graph generators, by contrast, do not exploit or enforce the structure of trees. We propose Autoregressive Frontier Expansion, a generative framework that constructs trees through an iterative expansion process, simulating the biological growth of real trees. At each step, a flow-matching model parameterised by an SO(2)-equivariant GNN expands the frontier by predicting whether each active branch bifurcates or terminates. We evaluate our method on cortical neurons and botanical trees in unconditional, class-conditioned, and morphology-guided generation. Across both domains, the generated morphologies agree closely with the reference distributions and, in conditional experiments, with the specified targets.

1 INTRODUCTION

Tree-like (acyclic) branching structures regularly appear in nature: respiratory trees branch to increase surface area for gas exchange; botanical trees branch to improve light capture and resource distribution; and neurons branch to form connections with other neurons. Across biological systems, morphology – the anatomical shape – is often closely tied to physical constraints and functional requirements. For example, neuronal morphologies define the wiring patterns of the brain's circuitry and consequently its function (Torben-Nielsen & De Schutter, 2014; Memelli et al., 2013; Lin et al., 2018). Generating realistic simulations of branching structures is, hence, valuable for understanding such biological systems, especially where data capture is difficult. The brain, for example, contains billions of neurons with diverse morphologies (Markram et al., 2004) that need to be mapped to understand its function. This is resource-intensive (Schmitz et al., 2011), even for volumes as small as a cubic millimetre of a mammalian brain (MICrONS Consortium, 2025). Beyond neuroscience, inpainting incomplete scans of trees directly leads to better estimation of their biomass. In both domains, branching structure is extracted from point clouds and data augmentation with realistic synthetic structures can improve this step.

Several classical approaches have been used to generate branching morphologies (Honda, 1971; Prusinkiewicz & Lindenmayer, 1990; Kanari et al., 2022), but work on learning-based methods remains limited. MorphGrower (Yang et al., 2024) generates neuronal morphologies layer-by-layer, mimicking the growth pattern in nature, but uses a *predefined* branching structure from a seed tree – only the geometry, not the structure, is predicted. Similarly, MorphoGen (Zhu et al., 2025) predicts geometry through a point cloud while the tree is recovered post hoc by a skeletonisation heuristic. In both cases the topology is not explicitly learned but either fixed in advance or recovered afterwards.

In this work, we propose *autoregressive frontier expansion*, an iterative procedure that conditions future branching on the intermediate morphology, mirroring how branching structures grow in nature (Barthélémy & Caraglio, 2007; Niell et al., 2004). In contrast to previous approaches, our method predicts both *topology* and *geometry* jointly. We parameterise this process with an SO(2)-equivariant GNN that accounts for the principal axis of biological structures. Our method is able to generate

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

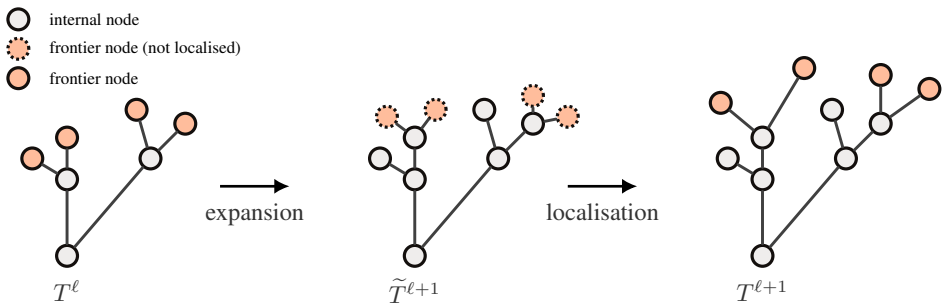


Figure 1: Overview of a generation step. Starting from a partial tree T^ℓ, we (i) **expand** the current frontier (a subset of leaves) by deciding for each frontier node whether it branches or terminates, yielding a tree $\widetilde{T}^{\ell+1}$; then (ii) **localise** by predicting 3D parent-relative offsets for the previous frontier nodes, producing $T^{\ell+1}$. The process repeats until all frontier nodes terminate.

structurally valid trees by construction. Across $26{,}000$ cortical neurons as well as botanical trees, the generated morphologies closely match the corresponding reference distributions. Further, our method follows a target's branching profile when guided by it, making it well suited for data augmentation. Lastly, iterative generation is computationally efficient and scales better with tree size compared to baselines.

1.1 Related Work

Growth-rule based approaches On the classical, non-learning side, growth-rule based methods use a predefined set of rules or probabilistic conditioning on priors to model the growth process of branching morphologies (Yang et al., 2024). Honda (1971) uses preset rules to model a tree as repeated bifurcations of straight segments with fixed branching angles and constant length contraction ratios. Topological Neuron Synthesis (TNS) (Kanari et al., 2022) simulates growth as a probabilistic procedure that generates neuronal trees conditioned on a topological description of their branching structure. The conditioning signal is given by persistence barcodes (Carlsson, 2009), formalised for complex branching morphologies by Kanari et al. (2018) as the Topological Morphology Descriptor (TMD) which we also use in our experiments. Li et al. (2021) combines L-systems with learning and conditions on individual images.

Learned tree generation Existing learned methods for generating neuronal trees include MorphVAE (Laturnus & Berens, 2021), which trains a sequence-to-sequence variational autoencoder on soma-to-tip walks. New trees are sampled by decoding a set of such walks in 3D, clustering nearby points to merge nodes, and connecting consecutive points along each walk to form edges. Because the walks are decoded independently and merged only post hoc, MorphVAE does not guarantee a globally consistent tree topology. MorphGrower (Yang et al., 2024) grows neuronal trees iteratively, similar to our method. However, the branching structure is fixed to that of a seed tree, and only new geometry is generated. Thus, neither method performs the end-to-end joint generation of topology and geometry considered in our experiments. MorphoGen (Zhu et al., 2025) instead generates the complete point cloud before reconstructing the tree topology. A DiT-3D diffusion model (Mo et al., 2023) generates the neuron as an unordered cloud of $2{,}048$ points. A tree is recovered afterwards by L1-medial skeletonisation and a greedy linking heuristic. The connections are therefore added after generation rather than predicted by the model. We evaluate MorphoGen in the unconditional setting; conditional generation is not supported by the model. Lee et al. (2023) directly produce L-system-like strings using a transformer but relies on a large synthetic training set instead of real-world trees.

More broadly, TreeGen (Kollovieh et al., 2025) infers a hierarchical structure based on leaf positions in high-energy physics, thus focusing on the topology. Topology-only tree generation further appears as a subroutine in general learnable graph generation algorithms. For example, Jin et al. (2018) generate molecules by generating and expanding the tree backbone of the molecule and Shirzad et al. (2022) expand the idea to general graphs. Bergmeister et al. (2024) use iterative expansion and diffusion for generation of graphs without node features.

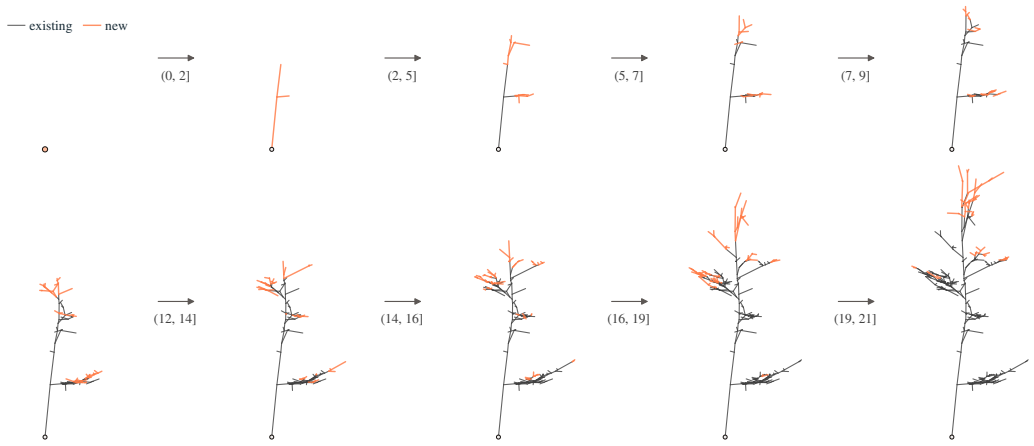


Figure 2: Stepwise generation of a botanical tree

2 METHODOLOGY

A tree T is a tuple (V, E, P) where $V = \{1, \dots, N\}$ is the node set, $E \subset V \times V$ is the edge set, and $P \in \mathbb{R}^{|V|\times 3}$ stores the node coordinates (root-centred). We write $r \in V$ for the root. Each non-root node $v \neq r$ has a unique parent $\pi(v)$. Let $\mathrm{Leaves}(T) \subseteq V$ denote the set of leaves of the tree. For any node-indexed matrix M we write $M(v)$ for the row of node v.

For modelling trees we restrict ourselves to the overarching branching geometry and thus only consider branching points (bifurcations) and leaves (terminations/tips), see Appendix B. We can do so since most information is encoded in the 'orientation and length of new branches' for both botanical trees (Honda, 1971) and neurons (Kanari et al., 2018).

2.1 TREE EXPANSION

We generate trees iteratively as a sequence $\{T^0, ..., T^L\}$. A partial tree at level ℓ is $T^\ell = (V^\ell, E^\ell, P^\ell)$. Our generation operates on an *active frontier* $\mathcal{A}^\ell = V^\ell \setminus V^{\ell-1} \subseteq \mathrm{Leaves}(T^\ell)$, i.e., the leaves created in the previous step. Each iteration consists of two operations: (1) frontier expansion and (2) localisation. See Fig. 1 for an overview and Appendix A for pseudocode.

Frontier expansion. Given T^ℓ, we predict a binary expansion state $\Gamma^\ell(v) \in \{0, 1\}$ for each $v \in \mathcal{A}^\ell$ and attach two children (v_L, v_R) to every v with $\Gamma^\ell(v) = 1$. Together with the pairs of new edges (v, v_L) and (v, v_R) this defines the intermediary expanded tree $\tilde{T}^{\ell+1}$.

Localisation. In the intermediary tree $\tilde{T}^{\ell+1}$ the coordinates of the new leaves $v \in \mathcal{A}^{\ell+1}$ are not yet specified. For each such $v \in \mathcal{A}^{\ell+1}$ we get the coordinates $P^{\ell+1}(v)$ and thus the next partial tree $T^{\ell+1}$ by predicting parent-relative offsets $C^{\ell+1}(v)$:

$$P^{\ell+1}(v) = P^\ell\big(\pi(v)\big) + C^{\ell+1}(v), \tag{1}$$

Offsets $C^{\ell+1}(v)$ are predicted conditioned on the intermediary tree $\tilde{T}^{\ell+1}$. For all nodes not in the frontier $\mathcal{A}^{\ell+1}$, the position does not change, so $P^{\ell+1}(u) = P^\ell(u)$ for $u \notin \mathcal{A}^{\ell+1}$.

2.2 MODELLING AND TRAINING

Joint prediction. Expansion and localisation are two operations, but we model them as a single conditional distribution by shifting the expansion prediction forward by one level: at level ℓ we predict the offsets C^ℓ that place the current frontier *and* the expansion states Γ^ℓ that decide the subset of frontier nodes bifurcating next. Given $\tilde{T}^\ell$, the pair (C^ℓ, Γ^ℓ) determines both T^ℓ and $\tilde{T}^{\ell+1}$, so generation factorises over levels as

$$p_\theta(T^0, \dots, T^L) \;=\; \prod_\ell p_\theta\big(C^\ell,\, \Gamma^\ell \,\big|\, \tilde{T}^\ell\big). \tag{2}$$

Each factor conditions on the entire partial tree generated so far.

Sampling. We initialise $T^0 = (\{r\}, \emptyset, P^0(r) = \mathbf{0})$ and assume the root *has to* expand: $\Gamma^0(r) = 1$. To properly model neuron morphologies, we allow k children at the root (soma) with k provided as input. That is, $\tilde{T}^1$ is given and $\mathcal{A}^1 := \{r_1, r_2, ..., r_k\}$. For $\ell \geq 1$, we repeatedly sample positions C^ℓ and expansion states Γ^ℓ conditioned on $\tilde{T}^\ell$, defining the tree T^ℓ as well as $\tilde{T}^{\ell+1}$ with the new frontier $\mathcal{A}^{\ell+1}$ according to Γ^ℓ. Generation stops when $\Gamma^\ell = \mathbf{0}$, i.e. the frontier becomes empty.

Equivariant GNN. Trees and neurons have an inherent orientation: trees typically grow upward, while neurons are usually oriented perpendicular to the cortical surface (Cajal, 1911). To encode this inductive bias, we introduce an SO(2)-EGNN model (see Appendix D), inspired by the fully E(n)-equivariant model proposed by Satorras et al. (2021). Every learned quantity inside the network is invariant to rotation about that axis. We achieve equivariance by interpreting predicted values in a local frame built from the global principal axis and the incoming branch direction at the parent node.

Flow Matching. The prediction of the positions and expansion states at each level (C^ℓ, Γ^ℓ) follows a flow matching model. Flow matching learns a continuous transformation from a simple noise distribution to the distribution of training examples. During training, the network learns the direction in which intermediate noisy samples should move along this transformation. At generation time, we start from noise and follow these learned directions for a small number of steps to obtain a sample (see Appendix C for more details).

Training sequence. Let T be a tree with root r and T_ℓ the subtree induced by nodes at most ℓ hops from r. Training uses the sequence $T_0, \ldots, T_D$, where D is the maximum depth and T_0 only contains r. Coarsening $T_{\ell+1} \to T_\ell$ removes one layer of nodes which provide ground-truth expansion targets for T_ℓ.

Global conditioning. For generation we consider unconditional as well as class- and tree-conditioned generation. For class-conditioning we use a one-hot encoding of the class to condition each flow step and for tree-specific conditioning we rely on a compact topological descriptor of the complete tree. The descriptor works by assigning a value to each node – path length, or straight-line distance from the root. Revealing parts of the tree according to the value gives a nested sequence of subgraphs, called a filtration. Zero-dimensional persistent homology records when branches appear and merge in the filtration as points in a persistence diagram. The resulting diagrams are converted to persistence images (Adams et al., 2017), concatenated, embedded, and supplied to the network at every flow step (see Appendix E).

3 Experiments

Our experiments cover both conditional and unconditional generation for both neuron morphologies and botanical trees. We also provide ablations to model size and use the botanical trees to show scaling behaviour with increased tree size.

3.1 Setup

Data. For neurons, we use 26,469 cortical pyramidal cells from the MICrONS project, collected with Geckt (2024) and cleaned following Weis et al. (2025). The dataset contains 7 cell-type labels and is split into 22,773 training, 2,529 validation, and 1,167 test cells. In the training split, neurons have 61 nodes on average, a mean depth of 8.2, and a mean soma degree of 7.7. The test split contains the neurons from the manually proofread MICrONS Minnie Column Schneider-Mizell et al. (2025) and thus has the highest quality reconstructions. *Botanical trees*: We use 3,386 QSM reconstructions of broadleaf trees from BioDiv-3DTrees (Griese et al., 2025). The dataset contains six genus labels, and the trees are truncated to depth 10/15/20 and contain 84/195/378 nodes on average. In both datasets, we retain only the root, branch points, and leaves and binarise non-root multifurcations. Further preprocessing details can be found in Section B.

Baselines. We compare with SemlaFlow (Irwin et al., 2024) and MorphoGen (Zhu et al., 2025). SemlaFlow is an E(3)-equivariant flow-matching model developed for molecular generation, achiev-

ing state-of-the-art performance for molecular generation on QM9 (Ramakrishnan et al., 2014). It generates all nodes jointly and predicts their connectivity with a bond head, rather than growing the tree. Unlike our model, it has no distinguished principal axis. MorphoGen (Zhu et al., 2025) is the closest neuron-specific generator. It uses a DiT-3D diffusion model that generates the neuron as an unordered point cloud of $2{,}048$ points, from which a tree is recovered post hoc by medial contraction and a greedy growth-guided linking heuristic. It is unconditional only and is handed neither a node count nor the soma degree.

We post-process baseline generations to ensure valid trees, effectively selecting the largest connected component (Section G). Our model creates valid trees by construction. We provide SemlaFlow with the reference node count and, for neuronal data, our model with the soma degree.

Metrics. We assess how well generated populations reproduce both the *individual morphological properties* and the *joint distribution* of the reference data. For individual properties, we use 1-Wasserstein (W_1) to measure distances between distributions, normalised by the corresponding reference standard deviation (W_1/σ_{ref}, see Section F). For joint comparisons, we use (1) a nine-dimensional morphometric vector of graph statistics (Section F), and (2) TMD persistence images under the path- and radial-from-root filtrations (Section E). For each representation, we compute MMD^2, which measures disagreement between joint distributions of generated and reference samples. To account for finite-sample variation, we subtract the MMD^2 between the same reference set and an equally sized, disjoint sample of training morphologies; we report this baseline-adjusted discrepancy as ΔMMD^2 (closer to zero is better). For the morphometric vector, we additionally compute *density* and *coverage*: density indicates whether generated samples lie in regions supported by the reference data, whereas coverage measures how much of the reference distribution they represent.

Reporting conventions. Samples are drawn with a red circle indicating the root, and branches coloured by depth relative to the projection plane. Structural validity is measured on raw outputs, morphology metrics use the method-specific post-processing that ensures valid trees.

3.2 Unconditional Generation

For unconditional generation in Table 1, we ask whether the generations are valid morphologies (i.e. graph-theoretic trees) and how closely the generated population matches the reference. By design, all our samples are valid trees, as are MorphoGen's given by a spanning tree over the predicted points. In contrast, 23.4% of SemlaFlow's raw unconditional outputs are not trees, and 34.0% of the outputs are modified by the postprocessing which contracts degree-2 nodes.

On population statistics, our method outperforms both baselines on the structural properties and on the joint distances to the reference. The density and coverage scores, 0.88 and 0.80 respectively, show that the generated samples lie in regions supported by the reference data and cover much of the reference set. MorphoGen performs poorly on both the marginal and the joint evaluations: it over-branches, producing more than twice the reference node count with branches less than half as long. Its density and coverage are near zero, and it has the largest MMD among the evaluated methods. SemlaFlow is closer to the reference than MorphoGen but remains below our model on density, coverage, and every axis-independent marginal statistic (Fig. 3, bottom).

Our main systematic gap is the under-generation of tall neurons: only 24.2% of generated cells have an axial extent beyond $400\,\mu\text{m}$, compared with 33.5% of the reference cells. This recurs in both conditional settings below, although the gap is smaller. A set of samples from our model is shown in Figure 3. More extensive samples from Reference, SemlaFlow, MorphoGen, and our model are shown in Fig. 9 in the appendix.

Smaller model size. We experiment with narrower models based on a node-feature width F of 64, 128, and the default 256 in Table 1. We compare $1{,}167$ samples from each method with an equally sized sample of training morphologies. Even at $F = 64$ with only 1.57M parameters – $14\times$ fewer than SemlaFlow and $21\times$ fewer than MorphoGen – our model still outperforms both. Increasing the width improves the joint distances and marginals, while coverage is already saturated at $F = 64$.

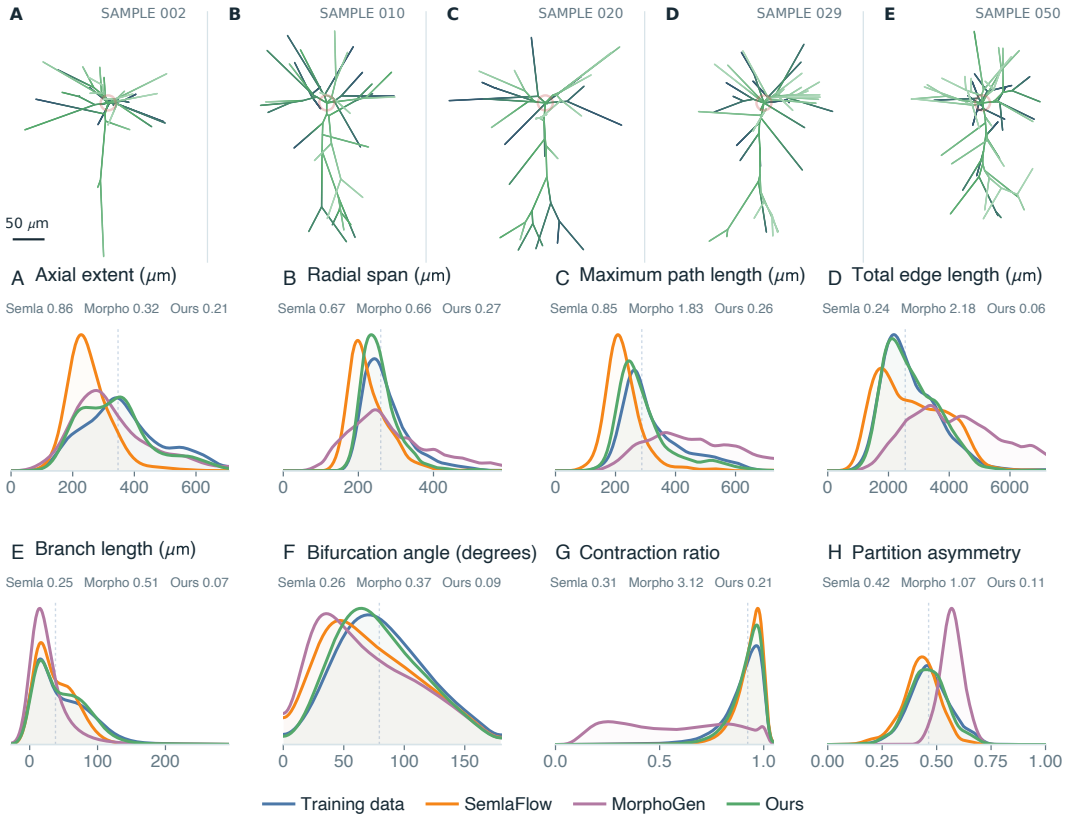


Figure 3: **Unconditional neuronal morphology generation. Top:** Five representative generations from our model. **Bottom:** Distributions of eight morphology statistics for the training reference, SemlaFlow, MorphoGen, and our model. Dashed lines mark training-reference medians; grey values give the normalised W_1 to the training reference (lower is better). Within-tree distributions weight each morphology equally, and horizontal ranges are determined from the training reference.

3.3 Class-Conditioned Generation

Conditioning on the seven cell-type labels yields the expected class differences (Fig. 4): compact, bushy arbours for 23P and 4P cells, tall apical trunks for 5P-ET and 5P-NP, and long, sparsely branched layer-6 cells shown next to reference cells.

The class structure also holds up quantitatively. We evaluate maximum path length, branch length, bifurcation angle, contraction, and partition asymmetry. Node count is excluded because it is sup-

Table 1: **Unconditional neuronal morphology generation.** Distributional metrics on 1,167 samples. The real–real row compares two disjoint training samples of the same size. Mean marginal W_1 averages the six rotation-invariant statistics defined in Section F. Default for our method is $F = 256$.

Method	Params.	Structure	Distributional agreement				
		Valid tree (%) ↑	Morph. ΔMMD^2 ↓	TMD ΔMMD^2 ↓	Coverage ↑	Density → ref.	Mean norm. marginal W_1 ↓
Real–real	—	100.0	0.0000	0.0000	0.971	0.976	0.039
SemlaFlow	22.3M	76.6	0.2521	0.1038	0.223	0.155	0.389
MorphoGen	32.6M	**100.0**	0.8436	0.2077	0.006	0.018	1.515
Ours ($F = 64$)	1.57M	**100.0**	0.0557	0.0146	**0.808**	0.827	0.175
Ours ($F = 128$)	5.69M	**100.0**	0.0471	0.0108	0.805	0.862	0.163
Ours (default)	21.6M	**100.0**	**0.0393**	**0.0092**	0.798	**0.876**	**0.133**

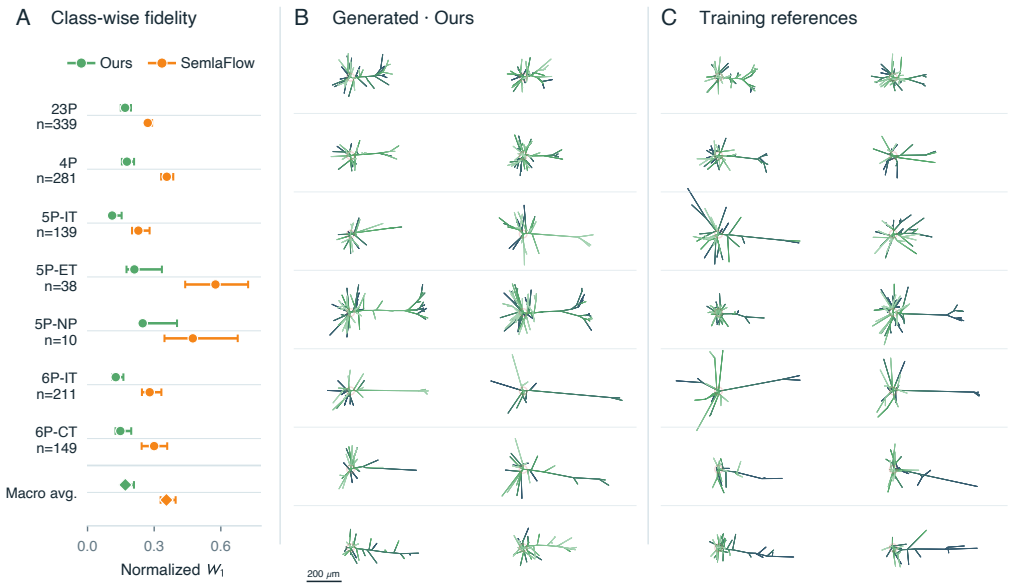


Figure 4: **Class-conditioned neuronal morphology generation.** **(A)** Class-wise morphology discrepancy for our model and SemlaFlow relative to the training trees. The values represent normalised 1-Wasserstein distance, averaged over five rotation-invariant statistics (lower is better). Whiskers show 95% bootstrap intervals, resampling the generated morphologies while holding the sampled training reference fixed. n is the number of generated morphologies. **(B)** Two selected generations from our model per class. **(C)** Two selected training-set references per class.

plied to SemlaFlow, while directional extents are excluded because SemlaFlow has no fixed vertical axis. Averaged over the seven classes and five statistics, the normalised W_1 is 0.170 for our model and 0.355 for SemlaFlow. MorphoGen only supports unconditional generation and is thus not included. Fig. 8 reports all statistics for each class separately. For our method, within-tree statistics such as branch length and bifurcation angle are matched within one standard deviation in every class.

3.4 Morphology-Guided Generation

Our method allows us to condition directly on the morphology of a given tree based on the fixed-length TMD vector. Figure 5 shows two TMD-conditioned generations from our model compared to the ground-truth morphology. The generations retain the target's overall shape, with a long main branch and most branching near the top. Their path persistence diagrams, which the TMD condition directly represents (see Section E), align closely. The distributions of branch length, distance from the root, and branch order also agree well, while branching angles have slightly larger differences.

Table 2: **TMD-conditioned generation.** Each method generates one morphology for each of 1,167 held-out targets. Entries in the *persistence diagrams* and *within-tree distributions* are means over target–generation pairs. The former compares normalised persistence diagrams, and the latter compares distributions of within-tree attributes including branch length, sibling angle, and branch order. The reference row pairs each target with a different held-out reference using a fixed-seed random derangement, providing a no-conditioning baseline.

Method	Validity	Persistence diagrams		Within-tree distributions		
	Valid tree (%) ↑	Path PD W_1 ↓	Radial-root PD W_1 ↓	Branch length W_1 (μm) ↓	Sibling angle W_1 (°) ↓	Branch order W_1 ↓
Reference	100.0	5.914	6.453	16.15	8.34	1.190
SemlaFlow	63.2	3.158	**2.453**	18.01	10.61	0.668
Ours	**100.0**	**2.939**	2.803	**13.38**	**9.50**	**0.548**

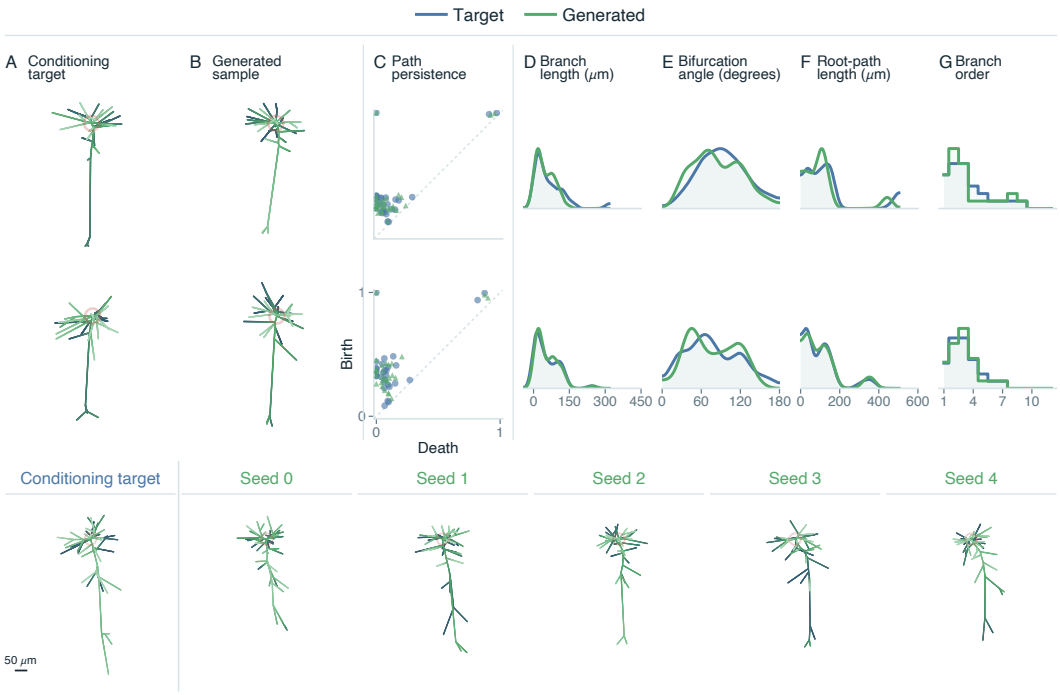


Figure 5: **TMD-conditioned generation. Top:** Each row shows a conditioning target, one generated sample, their path-based persistence diagrams, and within-tree distributions of branch length, bifurcation angle and branch order. **Bottom:** 5 independent samples conditioned on the same target.

This suggests that TMD primarily controls overall structure rather than finer geometric details. The bottom of Fig. 5 illustrates this with five generations from the same target, retaining the overall structure while differing in smaller details. See Fig. 11 for more examples.

Table 2 evaluates TMD-conditioned generation across the full test set. Our method produces a valid tree in every case, compared to 63.2% for SemlaFlow, and gives lower errors for path persistence, branch length, branching angle. SemlaFlow performs slightly better on radial persistence. Both persistence-diagram distances are roughly halved relative to a randomly paired held-out tree (top row). This indicates that the generations follow the specified target rather than simply matching the overall population. Appendix Table 5 provides a stricter comparison, where conditional generations are compared to the target and a root-degree matched random sample. Averaged over five seeds, the target is closer in about 91% of comparisons. In 97.9% of cases, our generator produces five unique generations out of the five seeds.

3.5 Botanical Trees

We evaluate unconditional and TMD-conditioned generation on botanical trees (Table 3). Our model gives the lower discrepancy on all four morphometrics in both regimes, with a mean normalised W_1 of 0.236 unconditionally and 0.141 under TMD guidance – between $2.8\times$ and $4.0\times$ better than SemlaFlow. SemlaFlow degrades on validity: at most 12.5% of its raw outputs are trees at all, against 76.6% on neurons. As with neurons, morphology guidance is the strongest regime. Performance is therefore maintained across two structurally and biologically distinct domains, supporting frontier expansion as a general model of branching structure rather than one tuned to a single morphology.

Scaling with tree size. Figure 6 illustrates the increasingly detailed structures generated at maximum depths of 10, 15, and 20 hops from the root. For the quantitative comparison, we evaluate unconditional generation on capped test sets (Table 4). To make the largest set feasible for SemlaFlow, we exclude the same 199 trees exceeding 1,110 nodes at every depth. Our model gives the lower discrepancy on all four morphometrics at every depth: its mean normalised W_1 ranges from



Figure 6: **Generated botanical trees across depth caps.** Three representative samples are shown for each maximum topological depth $d_{\max} \in \{10, 15, 20\}$.

Table 3: **Botanical-tree generation.** Population-level discrepancies on botanical trees restricted to depth 10 (full dataset). Morphology entries are tree-balanced 1-Wasserstein distances normalised by the reference standard deviation W_1/σ_{ref} (lower is better). Mean averages the distances. Bold marks the better result per training regime.

Regime	Method	Valid tree (%) ↑	Max. path length ↓	Branch length ↓	Bifurcation angle ↓	Partition asymmetry ↓	Mean ↓
Unconditional	SemlaFlow	3.9	1.280	0.245	0.297	0.822	0.661
	Ours	**100.0**	**0.443**	**0.167**	**0.177**	**0.157**	**0.236**
TMD-conditioned	SemlaFlow	12.5	0.982	0.172	0.417	0.683	0.564
	Ours	**100.0**	**0.165**	**0.087**	**0.238**	**0.076**	**0.141**

0.27 at D10 to 0.38 at D20, whereas for SemlaFlow it ranges from 0.66 to 1.84. Sampling time grows from 0.192 to 0.617 seconds per tree for our method, compared with 0.562 to 12.317 seconds for SemlaFlow.

Table 4: **Unconditional botanical-tree generation across depth caps.** Columns are W_1/σ_{ref} discrepancies (lower is better), Mean averages the four discrepancies, and Time is seconds per tree. Bold marks the better result within each depth.

Depth	Method	Max. path length ↓	Branch length ↓	Bifurcation angle ↓	Partition asymmetry ↓	Mean ↓	Time (s/tree)
D10	SemlaFlow	1.314	0.275	0.330	0.737	0.664	0.562
	Ours	**0.542**	**0.178**	**0.197**	**0.148**	**0.266**	**0.192**
D15	SemlaFlow	2.347	0.441	0.272	1.871	1.233	3.714
	Ours	**0.639**	**0.227**	**0.200**	**0.139**	**0.301**	**0.390**
D20	SemlaFlow	2.572	0.361	0.452	3.964	1.837	12.317
	Ours	**0.702**	**0.250**	**0.198**	**0.366**	**0.379**	**0.617**

4 Conclusion

Autoregressive frontier expansion generates 3D tree structures by predicting topology and geometry one level at a time. On cortical neurons and botanical trees, it produces valid trees by construction, matches reference morphology distributions more closely than the evaluated baselines, and scales to larger trees. Conditioning on cell type or a topological description enables class-level and target-specific control. These results indicate that iterative tree construction is a useful inductive bias for generative models of branching structures. Further questions include generation in the context of neighbouring trees and neurons, which the existing conditioning targets do not yet cover.

AI USE STATEMENT

We used large language models for language editing, including spell-checking, condensing, and clarifying text. We also used AI tools for coding assistance, including implementing, debugging, and testing research and evaluation code. All AI-assisted work was reviewed by the authors. The research ideas, experimental design, and interpretation of the results are entirely our own.

REFERENCES

Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18 (8):1–35, 2017.

Daniel Barthélémy and Yves Caraglio. Plant architecture: a dynamic, multilevel and comprehensive approach to plant form, structure and ontogeny. *Annals of botany*, 99(3):375–407, 2007.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *International Conference on Learning Representations*, volume 2024, pp. 27698–27736, 2024.

Santiago Ramón Cajal. *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.

Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

Herbert Edelsbrunner, John Harer, et al. Persistent homology-a survey. *Contemporary mathematics*, 453(26):257–282, 2008.

Dean Geckt. Neuron swc editor: A browser-based tool for visualizing and editing neuronal morphology, 2024. URL `https://github.com/deangeckt/swc_editor`. `https://deangeckt.github.io/swc_editor/`.

Nils Griese, Martin Ritzert, and Nils Nölke. A large dataset of labelled single tree point clouds, qsms and tree graphs. *Scientific Data*, 2025.

Hisao Honda. Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body. *Journal of theoretical biology*, 31(2):331–338, 1971.

Ross Irwin, Alessandro Tibo, Jon Paul Janet, and Simon Olsson. Semlaflow–efficient 3d molecular generation with latent attention and equivariant flow matching. *arXiv preprint arXiv:2406.07266*, 2024.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Lida Kanari, Paweł Dłotko, Martina Scolamiero, Ran Levi, Julian Shillcock, Kathryn Hess, and Henry Markram. A topological representation of branching neuronal morphologies. *Neuroinformatics*, 16(1):3–13, 2018.

Lida Kanari, Hugo Dictus, Athanassia Chalimourda, Alexis Arnaudon, Werner Van Geit, Benoit Coste, Julian Shillcock, Kathryn Hess, and Henry Markram. Computational synthesis of cortical dendritic morphologies. *Cell Reports*, 39(1), 2022.

Marcel Kollovieh, Nils Fleischmann, Filippo Guerranti, Bertrand Charpentier, and Stephan Günnemann. Treegen: A bayesian generative model for hierarchies. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Sophie C. Laturnus and Philipp Berens. Morphvae: Generating neural morphologies from 3d-walks using a variational autoencoder with spherical latent space. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6021–6031. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/laturnus21a.html.

Jae Joong Lee, Bosheng Li, and Bedrich Benes. Latent l-systems: Transformer-based tree generator. *ACM Transactions on Graphics*, 43(1):1–16, 2023.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Bosheng Li, Jacek Kałużny, Jonathan Klein, Dominik L Michels, Wojtek Pałubicki, Bedrich Benes, and Sören Pirk. Learning to reconstruct botanical trees from single images. *ACM Transactions on Graphics (TOG)*, 40(6):1–15, 2021.

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

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Henry Markram, Maria Toledo-Rodriguez, Yun Wang, Anirudh Gupta, Gilad Silberberg, and Caizhi Wu. Interneurons of the neocortical inhibitory system. *Nature reviews neuroscience*, 5(10):793–807, 2004.

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

MICrONS Consortium. Functional connectomics spanning multiple areas of mouse visual cortex. *Nature*, 640(8058):435–447, 2025.

Shentong Mo, Enze Xie, Ruihang Chu, Lanqing Hong, Matthias Nießner, and Zhenguo Li. DiT-3D: Exploring plain diffusion transformers for 3d shape generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Cristopher M Niell, Martin P Meyer, and Stephen J Smith. In vivo imaging of synapse formation on a growing dendritic arbor. *Nature neuroscience*, 7(3):254–260, 2004.

Przemyslaw Prusinkiewicz and Aristid Lindenmayer. Graphical modeling using l-systems. In *The Algorithmic Beauty of Plants*, pp. 1–50. Springer, 1990.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):140022, 2014.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Sabine K Schmitz, JJ Johannes Hjorth, Raoul MS Joemai, Rick Wijntjes, Susanne Eijgenraam, Petra de Bruijn, Christina Georgiou, Arthur PH de Jong, Arjen van Ooyen, Matthijs Verhage, et al. Automated analysis of neuronal morphology, synapse number and synaptic recruitment. *Journal of neuroscience methods*, 195(2):185–193, 2011.

Casey M Schneider-Mizell, Agnes L Bodor, Derrick Brittain, JoAnn Buchanan, Daniel J Bumbarger, Leila Elabbady, Clare Gamlin, Daniel Kapner, Sam Kinn, Gayathri Mahalingam, et al. Inhibitory specificity from a connectomic census of mouse visual cortex. *Nature*, 640(8058):448–458, 2025.

Hamed Shirzad, Hossein Hajimirsadeghi, Amir H Abdi, and Greg Mori. Td-gen: graph generation using tree decomposition. In *International Conference on Artificial Intelligence and Statistics*, pp. 5518–5537. PMLR, 2022.

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

Marissa A Weis, Stelios Papadopoulos, Laura Hansel, Timo Lüddecke, Brendan Celii, Paul G Fahey, Eric Y Wang, J Alexander Bae, Agnes L Bodor, Derrick Brittain, et al. An unsupervised map of excitatory neuron dendritic morphology in the mouse visual cortex. *Nature communications*, 16 (1):3361, 2025.

Nianzu Yang, Kaipeng Zeng, Haotian Lu, Yexin Wu, Zexin Yuan, Danni Chen, Shengdian Jiang, Jiaxiang Wu, Yimin Wang, and Junchi Yan. Morphgrower: A synchronized layer-by-layer growing approach for plausible neuronal morphology generation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ZTN866OsGx.

Tianfang Zhu, Hongyang Zhou, and Anan Li. MorphoGen: Efficient unconditional generation of long-range projection neuronal morphology via a global-to-local framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13021–13031, 2025.

APPENDIX

The appendix includes pseudocode for our method (Section A), details on data preprocessing (Section B), details on the flow-matching procedure (Section C), the SO(2)-equivariant architecture (Section D), our conditioning vector (Section E), metrics (Section F), postprocessing for our baselines (Section G), as well as additional results and samples (Sections H and I).

A PSEUDOCODE FOR AUTOREGRESSIVE FRONTIER EXPANSION

Algorithm 1: Tree generation procedure

Input: root degree k

$V^1 = \{r, r_1, \dots, r_k\};\ E^1 = \{(r, r_i)\}_{i=1}^{k};\ P^0(r) = \mathbf{0}$ `// root at the origin`

$\mathcal{A}^1 = \{r_1, \dots, r_k\};\ \ell = 0$ `// k unplaced children: this is` $\tilde{T}^1$

while $\mathcal{A}^{\ell+1} \neq \emptyset$ **:**

  $\ell = \ell + 1$

  Sample $(C^\ell, \Gamma^\ell) \sim p_\theta(\cdot \mid \tilde{T}^\ell)$ `// flow matching, Appendix C`

  $P^\ell = P^{\ell-1}$

  foreach $v \in \mathcal{A}^\ell$ **:**

    $P^\ell(v) = P^{\ell-1}(\pi(v)) + C^\ell(v)$ `// localise:` $\tilde{T}^\ell \to T^\ell$

  $V^{\ell+1} = V^\ell;\ E^{\ell+1} = E^\ell;\ \mathcal{A}^{\ell+1} = \emptyset$

  foreach $v \in \mathcal{A}^\ell$ *with* $\Gamma^\ell(v) = 1$ **:**

    $V^{\ell+1} = V^{\ell+1} \cup \{v_L, v_R\};\ E^{\ell+1} = E^{\ell+1} \cup \{(v, v_L), (v, v_R)\}$

    $\mathcal{A}^{\ell+1} = \mathcal{A}^{\ell+1} \cup \{v_L, v_R\}$ `// expand:` $T^\ell \to \tilde{T}^{\ell+1}$

return T^ℓ

Pseudocode for our autoregressive tree generation. The root is initialised with k unplaced children, where k is an input to the sampler. In each iteration the offsets C^ℓ and expansion states Γ^ℓ of the current frontier $\mathcal{A}^\ell$ are sampled jointly given the intermediary tree $\tilde{T}^\ell$. The offsets place the frontier relative to its parents, and every node with $\Gamma^\ell(v) = 1$ receives two children, which form the next frontier. The process continues until no node expands and the frontier is empty.

B PREPROCESSING AND TREE REPRESENTATION

We assume trees to consist of binary branching nodes and leaves only. We thus preprocess all trees to their branching skeletons by contracting all degree-2-paths such that only branching nodes and leaves remain. Degree-2-paths encode the local shape of branches rather than structure. We leave the extension of our method to also predict those local branch shapes as future work. Performing this simplification also benefits GNNs and message passing: contracting degree-2 paths shortens the graph distances that message passing must span. In an unreduced MICrONS tree, even a 10-layer GNN may fail to propagate information from one branching node to the next.

We further assume that apart from the root (to be able to generate neuron morphologies, see next paragraph) trees are binary, which simplifies the decision space for the learned expansion target. A three-way junction is split losslessly by inserting a node at the intersection; at wider junctions we keep the two thickest children and drop the rest, which removes roughly 1.25% of neuron nodes. Handling junctions with more than two children natively is kept for future work.

The root is exempt from this constraint: a soma carries 1 to $K_{\max}$ primary dendrites, and we set $K_{\max} = 23$, the largest number observed in the corpus, so that no neuron is excluded by the bound. In our data the mean soma degree is 7.7. Botanical trees have root degree 1 or 2, the base of the trunk being a single cylinder. The rank of each root child among its siblings is supplied to the model as an input feature; Section D.5 explains why it is needed – all root children share a single local frame, so rank is the only signal that distinguishes them.

For the scaling experiment, we use the botanical trees as they are much larger and deeper than neurons, reaching 70 hops and 3k nodes on average (max 326 hops and 27k nodes), whereas neurons

have 8 hops and 61 nodes (max 35 hops and 537 nodes). We thus use the first 10/15/20 hops of each of the trees and only consider trees where size at 20 hops does not exceed 1,110 nodes to be able to run SemlaFlow. The cap removes around 6% of trees which would otherwise again have up to 3k nodes, and now reaches 378 nodes on average. In contrast to the full D10 trees used for unconditional generation in Table 3 with an average number of nodes of 84, the cap globally dropped the densest trees such that the average number of nodes in Table 4 for D10 is 78.

C Flow Matching

At each expansion step, the newly created frontier $\mathcal{A}^\ell$ consists of leaves whose coordinates and *next-step* expansion state must be generated. As described in the main paper, both predictions are performed jointly using flow matching. Concretely, we parametrise this as a conditional flow matching problem (Lipman et al., 2022; Liu et al., 2022) on the *frontier state*, $X^\ell = [\check{C}^\ell, \Gamma^\ell] \in \mathbb{R}^{|\mathcal{A}^\ell| \times 4}$, conditioned on the full intermediate expanded tree $\tilde{T}^\ell$. For flow matching we treat Γ as continuous.

In this section, we drop the tree expansion level ℓ from the superscript to make the notation easier to read. In the subscript we now write the *flow time* $t \in [0, 1]$, so $P_t^\ell(v)$ is the position of node v in round ℓ at flow time t. Note that t runs from noise to data: $t = 0$ is a draw from the prior and $t = 1$ is the data. The offsets $C(v)$ of Eq. (1) are expressed in world axes for improved readability. In practice, our method works in *local coordinates* $\check{C}(v)$ to facilitate learning and to incorporate SO(2)-equivariance, see Sections D.3 and D.4.

Probability path. We use the linear (optimal-transport) interpolant between a prior sample X_0 and the data X_1:

$$X_t = (1 - t)\, X_0 + t\, X_1, \qquad X_0 \sim \mathcal{N}(0, \Sigma), \qquad t \sim \mathcal{U}[0, 1], \tag{3}$$

drawn independently for each frontier node v. Here Σ is diagonal and shared by every node: it has unit variance on the expansion coordinate, and per-axis scales set to the spread of the ground-truth offsets along each frame axis, in each corpus's normalised coordinates. In training we sample a single t per graph. Thus, each training example consists of a tree T sampled at some level ℓ together with a flow time t, and we aim to predict the velocity of the path at X_t.

Velocity field. Because the path in Eq. (3) is a straight line, its velocity is constant along the path and given in closed form by

$$u(X_t, t) \;=\; \frac{\mathrm{d}X_t}{\mathrm{d}t} \;=\; X_1 - X_0. \tag{4}$$

Network and Inputs. Our velocity network is the SO(2)-EGNN model. It reads node coordinates only to form the rotation- and translation-invariant edge quantities of Section D.1, and never updates them. We construct P_t by overwriting only the frontier node coordinates:

$$P_t(v) = \begin{cases} P_1(v), & v \notin \mathcal{A}, \\ P_1(\pi(v)) + C_t(v), & v \in \mathcal{A}. \end{cases} \tag{5}$$

All non-frontier nodes are already placed and provide geometric context. $\check{C}_t(v)$ is converted back to the global frame as input to the invariant network. However, note that P_t is a transient input, not part of the integrated state: it is rebuilt from P_1 at every step and discarded, while the state itself never leaves frame coordinates.

Let H denote deterministic graph inputs: node features and constructed geometric edge features. We provide t as a broadcast node feature and use the interpolated expansion state Γ_t as an additional node feature on frontier nodes (and 0 elsewhere). Then

$$v_\theta : \; (P_t, H, \Gamma_t, t) \mapsto \widehat{V}, \tag{6}$$

giving velocity predictions on the frontier state, $\widehat{V} = [\widehat{V}_{\check{C}}, \widehat{V}_\Gamma]$. Here $\widehat{V}_{\check{C}}(v)$ is a triple of frame coordinates, in the same basis as $\check{C}_t(v)$, and $\widehat{V}_\Gamma(v)$ is a scalar.

Algorithm 2: Flow-matching sampling of the frontier state

Input: intermediate tree $\tilde{T}^\ell$, frontier $\mathcal{A}$, steps K

local frames $(\mathbf{f}_v, \mathbf{s}_v, \hat{\mathbf{u}}) \leftarrow$ from placed parents `// built once, Section D.3`

$X \sim \mathcal{N}(0, \Sigma)$ `// prior sample in frame coords, Eq. (3)`

for $k = 0$ **to** $K - 1$ **:**

  $t_k = k/K$

  foreach $v \in \mathcal{A}$ **:**

    $P_{t_k}(v) = P(\pi(v)) + C_{t_k}(v)$ `// decode` $X_{\check{C}}$, `Eq. (12)`

  $H \leftarrow$ recompute branch angles, edge features for $\mathcal{A}$ `// frames locked`

  $\widehat{V} = v_\theta(P_{t_k}, H, X_\Gamma, t_k)$

  $X = X + \frac{1}{K}\widehat{V}$ `// explicit Euler, in frame coords`

foreach $v \in \mathcal{A}$ **:**

  $P^\ell(v) = P(\pi(v)) + C(v)$ `// final placement`

return P^ℓ, $\quad \Gamma^\ell = \mathbf{1}[X_\Gamma > 0]$ `// to build` $\tilde{T}^{\ell+1}$

Training Objective. We regress the velocity with an MSE loss on the frontier nodes only. Let θ denote the parameters of the SO(2)-EGNN.

$$\mathcal{L}(\theta) = \mathbb{E}_{t, X_0}\left[\sum_{v \in \mathcal{A}} \left\|\widehat{V}_{\check{C}}(v) - \left(\check{C}_1(v) - \check{C}_0(v)\right)\right\|_2^2 + \sum_{v \in \mathcal{A}} \left(\widehat{V}_\Gamma(v) - \left(\Gamma_1(v) - \Gamma_0(v)\right)\right)^2\right], \quad (7)$$

where the ground-truth target $\check{C}_1(v)$ is the observed offset read off in the local frame at v. Prediction and target are thus in a common basis and the loss needs no change of coordinates. The binary expansion label is mapped to $\Gamma_1 \in \{-1, +1\}$ rather than $\{0, 1\}$, so that it is symmetric about the prior mean; the decision threshold at sampling time is therefore 0.

Sampling. At inference we draw X_0 from the prior and integrate Eq. (4) forward with explicit Euler steps on a uniform grid $t_k = k/K$,

$$X_{t_{k+1}} = X_{t_k} + \tfrac{1}{K}\, v_\theta(P_{t_k}, H, \Gamma_{t_k}, t_k), \qquad k = 0, \ldots, K-1, \quad (8)$$

with $K = 10$ steps. Algorithm 2 gives the procedure in full. We note that for the algorithm, the integrated state X stays in frame coordinates throughout; the frame is used only to hand the network world coordinates to measure, and to place the node at the end. It is built once from the already-placed parent positions and held fixed across all K steps, so the basis its inputs are measured in and the basis its outputs are decoded in are the same, and neither drifts with the state being integrated (Section D.3). Only frontier-adjacent geometry is recomputed between steps.

D SO(2)-EGNN

D.1 Invariant edge features

The original EGNN achieves E(n) equivariance by using only pairwise squared distances as geometric input to edge messages, deliberately avoiding directional information that would break rotational symmetry (Satorras et al., 2021). For structures with a preferred axis, we instead decompose displacements into axis-parallel and perpendicular components.

For each directed edge $(i \to j)$ with coordinates $\mathbf{x}_i, \mathbf{x}_j$, we compute the relative displacement $\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and decompose it with respect to a rotation axis $\hat{\mathbf{u}}$, fixed per corpus. We form the axial component $d_{ij}^{\parallel} = \mathbf{r}_{ij}^\top \hat{\mathbf{u}}$ and the perpendicular residual $\mathbf{r}_{ij}^{\perp} = \mathbf{r}_{ij} - d_{ij}^{\parallel}\hat{\mathbf{u}}$, then take $\rho_{ij} = \|\mathbf{r}_{ij}^{\perp}\|_2$. The pair $(\rho_{ij}, d_{ij}^{\parallel})$ is invariant to rotations around $\hat{\mathbf{u}}$. We enrich these scalars with two branch angles, shared by both directions of the tree edge between v and its parent. Let $\mathbf{r}_v = P(v) - P(\pi(v))$ be the branch, $\hat{\mathbf{r}}_v^{\perp}$ its unit projection onto the plane perpendicular to $\hat{\mathbf{u}}$, and $(\mathbf{f}_v, \mathbf{s}_v, \hat{\mathbf{u}})$ the local frame of Section D.3. Then

$$\cos\psi_v = \mathbf{f}_v^\top \hat{\mathbf{r}}_v^{\perp}, \qquad \sin\psi_v = \mathbf{s}_v^\top \hat{\mathbf{r}}_v^{\perp}, \qquad \cos\phi_v = \frac{\mathbf{r}_v^\top \hat{\mathbf{u}}}{\|\mathbf{r}_v\|_2}, \quad (9)$$

so ψ_v is the azimuth of the branch about $\hat{\mathbf{u}}$, measured from the incoming direction at the parent, and ϕ_v its tilt from the axis. Both are invariant to rotations about $\hat{\mathbf{u}}$, since the frame co-rotates with the branch.

D.2 Message passing

Every tree edge is used in both directions. Writing $\mathbf{e}_{ij}$ for the invariant edge features above, one layer updates the node features $\mathbf{h}$ as

$$\mathbf{m}_{ij} = \mathrm{MLP}_e\big(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij}\big), \qquad \mathbf{m}_j = \sum_{(i\to j)\in E} \mathbf{m}_{ij}, \qquad \mathbf{h}_j \leftarrow \mathbf{h}_j + \mathrm{MLP}_h\big(\mathrm{LN}(\mathbf{h}_j), \mathbf{m}_j\big), \tag{10}$$

where $\mathrm{MLP}_e, \mathrm{MLP}_h$ are two-layer perceptrons and LN is layer normalisation. This is the EGNN layer of Satorras et al. (2021) with the squared distance replaced by $\mathbf{e}_{ij}$ and the coordinate update removed: coordinates are read once to form $\mathbf{e}_{ij}$ and never modified. Since messages only travel along tree edges, we precede every second layer with an induced-set attention block (Lee et al., 2019), in which a small set of learned per-graph tokens attends to all node features and the nodes attend back; it acts on invariant features only, so invariance is preserved.

D.3 Local frames

For a node v whose parent $\pi(v)$ itself has a parent $\pi(\pi(v))$, let $\mathbf{w}_v = P(\pi(v)) - P(\pi(\pi(v)))$ be the incoming branch direction at the parent. Projecting it into the plane perpendicular to $\hat{\mathbf{u}}$ and normalising gives

$$\mathbf{f}_v = \frac{\mathbf{w}_v - (\mathbf{w}_v^\top \hat{\mathbf{u}})\,\hat{\mathbf{u}}}{\|\mathbf{w}_v - (\mathbf{w}_v^\top \hat{\mathbf{u}})\,\hat{\mathbf{u}}\|_2}, \qquad \mathbf{s}_v = \hat{\mathbf{u}} \times \mathbf{f}_v, \tag{11}$$

so that $\mathbf{f}_v \perp \hat{\mathbf{u}}$ by construction. We call $(\mathbf{f}_v, \mathbf{s}_v, \hat{\mathbf{u}})$ the *local frame* at v: an in-plane forward direction inherited from the parent branch, a sideways direction completing the perpendicular plane, and the corpus axis. A triple $\check{\mathbf{c}}$ of coordinates in this frame denotes the world vector

$$C(v) = \check{c}_1\,\mathbf{f}_v + \check{c}_2\,\mathbf{s}_v + \check{c}_3\,\hat{\mathbf{u}}. \tag{12}$$

For trees, the basis always contains up (the principal axis). The first horizontal axis aligns with the parent-branch direction (projected to the horizontal plane), and the last axis completes the orthonormal frame.

D.4 Equivariance

Because the features of Section D.1 are all invariant, so is everything the network computes from them: the head emits a triple (a_v, b_v, c_v) of *frame* coordinates, together with the expansion velocity γ_v. Equivariance is supplied by the per-node local frame, against which Eq. (12) decodes those coordinates into geometry.

Let Q be a rotation about $\hat{\mathbf{u}}$, applied to every node coordinate. The edge quantities of Section D.1 are unchanged by construction, hence so are all node features and the head outputs. The prior of Eq. (3) is defined in frame coordinates with Σ fixed, so it carries no dependence on orientation either, and the Euler update of Algorithm 2 adds invariant increments to an invariant state: the integrated state X is therefore invariant. The frame vectors, by contrast, are built from coordinates and co-rotate, $\mathbf{f}_v \mapsto Q\mathbf{f}_v$ and $\mathbf{s}_v \mapsto Q\mathbf{s}_v$, with $\hat{\mathbf{u}}$ fixed by Q. Placement decodes that invariant state against the co-rotating frame (Eq. (12)), so each offset maps to $Q\,C(v)$; with the root at the origin, the generated positions co-rotate with the input.

D.5 The root

The root has no grandparent, so Eq. (11) has no incoming direction to work with, and its k children are created in a single step rather than one at a time ($1 \le k \le K_{\max}$). To resolve both facts, **all k root children share a single frame**, whose forward direction $\mathbf{f}_0$ is aligned during training with the child whose *entire subtree* reaches furthest along $-\hat{\mathbf{u}}$. At sampling no child is yet placed, so $\mathbf{f}_0$ is instead a single azimuth drawn uniformly at random per root; its absolute value is precisely the SO(2) degree of freedom the model does not fix. With the frame shared, no geometric signal

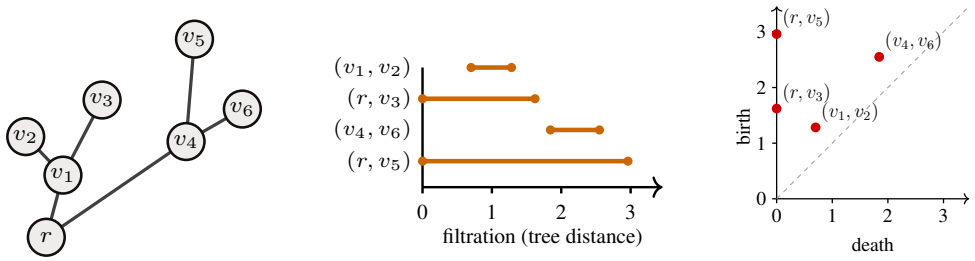


Figure 7: **(Left)** A tree with root r. The nodes are embedded in $\mathbb{R}^2$. Their coordinates induce Euclidean edge lengths, which define a distance-to-root filtration. **(Middle)** A barcode showing the resulting H_0 persistence under this filtration, with the elder rule deciding which branch survives each merge. The bars are annotated by (v_d, v_b), the death and birth nodes respectively. Note that the barcode itself is unlabelled. **(Right)** The corresponding persistence diagram. Points close to the diagonal represent short-lived features, while points further away represent more persistent features.

distinguishes one root child from another – their frames are identical and their parent is the same node. We therefore supply each root child's rank among its siblings as a $K_{\max}$-wide one-hot node feature.

E Topological Morphological Conditioning

To generate a tree with a *specified* structural profile, we additionally condition on a compact topological descriptor of global morphology (Edelsbrunner et al., 2008). This makes the model a controllable source of morphologies: given a descriptor computed from a real cell, it can produce further samples sharing that cell's branching profile – a form of targeted augmentation an unconditional sampler cannot provide. We do not evaluate downstream augmentation here; our experiments assess only how closely a generated morphology matches the profile it was conditioned on. Below we describe the relevant topological background in the context of tree morphologies.

Filtrations. Let $T = (V, E, P)$ be a tree with root r. Each node $v \in V$ has coordinates $P(v) \in \mathbb{R}^3$. A *filtration function* is a map $f : V \to \mathbb{R}$. Given f, we define the *superlevel set filtration* of T as the nested sequence of induced subgraphs $\{T_\alpha\}_{\alpha \in \mathbb{R}}$ where T_α contains all nodes v with $f(v) \geq \alpha$. Both filtration functions we use grow away from the root, so sweeping α downwards grows the tree inwards from the tips.

Barcode and elder rule. In this work we only use 0-dimensional persistent homology (PH_0), which tracks how connected components appear and merge along the filtration. A component is born at the filtration value α when a new connected component appears in T_α. Two components merge at filtration value α when they become connected in T_α. When two components merge, we apply the *elder rule*: the older component – the one born at the larger filtration value – persists, and the younger component dies. The multiset of intervals with endpoints b_i and d_i, the birth and death values of component i, is called the *persistence barcode* of T under filtration f. See Figure 7 for an example.

Persistence diagram. A barcode can equivalently be represented as a multiset of points in $\mathbb{R}^2$, called the *persistence diagram*, where each point (b_i, d_i) corresponds to one bar of the barcode. Points close to the diagonal represent short-lived features, while points further away represent more persistent features. Since $b_i > d_i$ under a superlevel filtration, we follow the convention of plotting death on the horizontal axis, placing the points above the diagonal.

Filtration choices used in this work. We use two filtrations, both anchored at the root and both computed with the TMD construction of Kanari et al. (2018):

- *Path from root* (see Figure 7): the geodesic distance from the root along the tree, with each edge weighted by its Euclidean length. This captures the branching structure, yielding a

compact summary of the global branching allocation across depth – *where* and *how much* branching occurs.

- *Radial from root*: the straight-line distance $\|P(v) - P(r)\|_2$ from the root. Where the path filtration measures reach *along* the tree, this measures reach *through space*, so the pair distinguishes trees that branch at the same topological depth but at different spatial extents.

Both filtrations are invariant to rotation about the root and therefore introduce no dependence on the distinguished axis of the SO(2)-EGNN (Appendix D). Filtration values are min-max normalised to $[0, 1]$ before the barcode is computed, so the descriptor encodes the *shape* of the branching profile rather than absolute scale.

Vectorisation. To provide this morphological information to the network with a fixed dimension, we convert persistence diagrams into persistence images. A *persistence image* (PI) (Adams et al., 2017) is a fixed-dimensional vector representation of a persistence diagram obtained by placing Gaussian kernels at each point and discretising the resulting density on a grid. We use a 16×16 grid per filtration, so the PIs for the two filtrations flatten and concatenate into a single 512-dimensional conditioning vector, which is embedded and broadcast to nodes as a graph-level condition. Conditioning on this descriptor produces trees whose branching structure and spatial extent follow the target profile.

F METRICS

This section defines the metrics used in the results tables. We use $T = (V, E, P)$. Let $(x_v, y_v, z_v) = P(v)$ denote the position of a node $v \in V$, let r be the root, and $\hat{\mathbf{u}}$ the unit vector along the chosen principal axis ('up'). For nodes $u, v \in V$, let $\boldsymbol{\delta}_{uv} = P(v) - P(u)$ denote the displacement from u to v, and let $d_{uv} = \|\boldsymbol{\delta}_{uv}\|_2$ denote its Euclidean length. Let $\mathcal{B}$ denote the set of nodes with at least two children, and $C(u)$ the children of u. We use the following tree statistics:

Axial Extent: Extent of the tree along the principal axis ('height' for botanical trees):

$$\mathrm{AxialExtent}(T) = \max_{v \in V} \big(P(v) \cdot \hat{\mathbf{u}}\big) - \min_{v \in V} \big(P(v) \cdot \hat{\mathbf{u}}\big)$$

Radial Span: Project the tree into the XY-plane, then take the longest distance between any two points in the projection

$$\mathrm{Diameter}(T) = \max_{u,v \in V} \big\| (x_u, y_u) - (x_v, y_v) \big\|_2$$

Maximum Branch Order: Maximum length of a path from the root to a leaf (in hops):

$$\mathrm{MaxBranchOrder}(T) = \max_{v \in V} \mathrm{len}(r, v),$$

where $\mathrm{len}(u, v)$ is the distance in hops between u and v.

Maximum Path Length: Maximum distance from the root to any node along the edges:

$$\mathrm{MaxPathLength}(T) = \max_{v \in V} \sum_{(u,w) \in \mathrm{path}(r,v)} d_{uw}.$$

Total Edge Length: Sum of the Euclidean lengths of all edges:

$$\mathrm{TotalEdgeLength}(T) = \sum_{(u,v) \in E} d_{uv}.$$

Partition Asymmetry: For every inner node $u \in \mathcal{B}$ we compute its asymmetry as $\frac{|a-b|}{a+b-2}$ where a and b are the number of leaves in the subtree rooted at each of the children of u. We subtract 2 in the denominator since those leaves (one on each side) are unavoidable for any inner node by definition. Averaging over all $u \in \mathcal{B}$ gives the partition asymmetry for T.

Mean Branch Length: Mean Euclidean length of all edges:

$$\mathrm{MeanBranchLength}(T) = \frac{1}{|E|} \sum_{(u,v)\in E} d_{uv}$$

Critical-Branch Length: Let $V_{\mathrm{crit}} = \{r\} \cup \{v \in V : |C(v)| \neq 1\}$ contain the root, branching nodes, and leaves. A critical branch $b = (v_0, \ldots, v_k)$ is a maximal path whose endpoints are in V_{crit} and whose internal nodes are not. Its length is

$$\mathrm{BranchLength}(b) = \sum_{i=1}^{k} d_{v_{i-1},v_i}.$$

Mean Bifurcation Angle: Mean angle, in degrees, across sibling-branch pairs at all branching nodes:

$$\mathrm{MeanBifurcationAngle}(T) = \frac{1}{|\mathcal{B}|} \sum_{u\in\mathcal{B}} \frac{180}{\pi} \arccos\left(\frac{\boldsymbol{\delta}_{u,c_1(u)} \cdot \boldsymbol{\delta}_{u,c_2(u)}}{d_{u,c_1(u)}\, d_{u,c_2(u)}}\right)$$

using $C(u) = (c_1(u), c_2(u))$ since we only consider binary trees. For the root which may have more than two children, we average over all pairs of angles between siblings.

Critical-Branch Bifurcation Angle: For two critical branches b_i and b_j starting at the same node u, let $t(b_i)$ and $t(b_j)$ denote their endpoints. Their angle is

$$\theta(b_i, b_j) = \frac{180}{\pi} \arccos\left(\frac{\boldsymbol{\delta}_{u,t(b_i)} \cdot \boldsymbol{\delta}_{u,t(b_j)}}{d_{u,t(b_i)}\, d_{u,t(b_j)}}\right).$$

Mean Radial Distance to Root: Mean straight-line distance from non-root nodes to the root:

$$\mathrm{MeanRadialToRoot}(T) = \frac{1}{|V|-1} \sum_{v\in V\setminus\{r\}} d_{rv}$$

Root-to-Leaf Contraction Ratio: Let $\mathcal{L}$ denote the set of non-root leaves. For each $\ell \in \mathcal{L}$, the contraction ratio is its straight-line distance from the root divided by its root-to-leaf path length:

$$\mathrm{Contraction}(\ell; T) = \frac{d_{r\ell}}{\displaystyle\sum_{(u,v)\in \mathrm{path}(r,\ell)} d_{uv}}.$$

Mean Contraction: Mean ratio of straight-line root-to-leaf distance to path length:

$$\mathrm{MeanContraction}(T) = \frac{1}{|\mathcal{L}|} \sum_{\ell\in\mathcal{L}} \frac{d_{r\ell}}{\displaystyle\sum_{(u,v)\in \mathrm{path}(r,\ell)} d_{uv}},$$

where $\mathcal{L}$ is the set of non-root leaves.

Sholl Critical Radius: To compute this we take a sphere around the root and count how many edges cross that sphere. We then compute at which radius this count is maximised. By normalising with the distance between the root and the furthest node in the tree, the radius becomes comparable between different-size trees.

Evaluation Representations From these statistics, we form two evaluation representations. For joint-distribution comparisons, each tree is represented by the nine-dimensional morphometric vector comprising axial extent, radial span, maximum branch order, partition asymmetry, mean branch length, mean bifurcation angle, mean radial distance to the root, mean contraction, and Sholl critical radius. This representation is used for morphometric MMD, density, and coverage.

For marginal comparisons, we use six selected rotation-invariant statistics: maximum path length, total edge length, critical-branch length, critical-branch bifurcation angle, root-to-leaf contraction ratio, and partition asymmetry. These statistics do not depend on a chosen global axis, allowing comparison with methods whose outputs have no distinguished orientation.

MMD^2 We compare the generated and ground-truth distributions using the squared maximum mean discrepancy (MMD) with a Gaussian RBF kernel:

$$\mathrm{MMD}^2(P,Q) = \mathbb{E}_{x,x'\sim P}[k(x,x')] + \mathbb{E}_{y,y'\sim Q}[k(y,y')] - 2\,\mathbb{E}_{x\sim P,\,y\sim Q}[k(x,y)],$$
$$k(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right).$$

It measures how much the within-distribution similarities exceed cross-distribution similarities and becomes zero if the two distributions P and Q match.

We estimate MMD^2 using its unbiased finite-sample estimator, $\widehat{\mathrm{MMD}}_u^2$. Let G, R, and R' be equally sized sets, where G contains generated morphologies and R and R' are disjoint samples of real training morphologies. We report the baseline-adjusted discrepancy

$$\Delta\,\mathrm{MMD}^2 = \widehat{\mathrm{MMD}}_u^2(G,R) - \widehat{\mathrm{MMD}}_u^2(R',R).$$

Values close to zero indicate agreement at the level observed between real samples; small negative values can arise from finite-sample estimation.

G Baseline Post-Processing

This section describes how raw outputs are processed before scoring. We distinguish a method's own tree construction, which is judged as its output, from repair that we add for comparability. Every added step is neutral or in the baseline's favour.

SemlaFlow. The raw output is a set of coordinates with pairwise bond logits, and the predicted adjacency is not constrained to be a tree. We keep the largest connected component, take its minimum spanning tree by Euclidean edge length, choose as root the unique maximum-degree node when one exists and otherwise the base of the principal axis, prune non-root nodes with more than two children down to the two largest subtrees, and contract non-root degree-2 nodes. This yields rooted, connected, and critical trees that are binary except for the root.

The repair is required for a fair evaluation. Root-path and branch-order statistics assume that every node is reachable from one root, persistence calculations require an acyclic graph, and the reference data contain only critical binary nodes and potentially a soma with more than two children. We use those repaired graphs to compute the morphology metrics. For structural validity we resort to the raw outputs to properly measure in how many cases such postprocessing was necessary.

MorphoGen. The raw output is a $2{,}048$-point cloud on the unit sphere. Tree construction is MorphoGen's published method and its cost is not subtracted: L1-medial contraction, farthest-point subsampling to $1{,}200$ points, soma detection as a local density peak, greedy outward growth of a spanning tree (linker caps non-root nodes at two children) and pruning of short terminal branches. Scale is assigned to each generated tree by a draw from the training-split scale distribution. The resulting tree, which carries about 32 points per branch, is then contracted to a critical skeleton by removing degree-2 nodes.

We make four changes to the released MorphoGen code: two align the implementation with the method described in the paper, and two tune hyperparameters for MICrONS. First, the released soma radius of 5 is in micrometres but is applied to unit-sphere data, so every density ball contains every point and the root is effectively arbitrary; we use a radius calibrated on real clouds, which reduces the normalised soma-placement error from 0.32 to 0.06. Second, the released code caps all nodes including the root at two children although the paper explicitly exempts the soma; we allow $K_{\max} = 23$. The two hyperparameters are the seed-edge weight and the branch-pruning length which are swept jointly on the training split and confirmed on validation; on reference clouds they reduce the mean normalised W_1 over seven morphometrics from 2.51 to 0.61. We use this optimised configuration when reporting on MorphoGen in the main text.

H Additional Quantitative Results

This section collects supplementary analyses of the neuron experiments. We examine class-conditioned population structure and test whether repeated TMD-conditioned samples remain specific to their assigned targets.

H.1 Variability of TMD-Conditioned Sampling

As already indicated in Fig. 5, there is some variability in the generation. Here we quantify this by producing five generations for each of the test neurons with TMD-conditioning. Table 5 shows that five distinct topologies are generated for 97.9% of targets, and the node count varies for 98.7%. The persistence-diagram distance among the five variants is half the distance to alternative neurons with the same root degree. Note that the rows of the table refer to different channels of the persistence diagram (path and radial-root). Larger sets of repeated samples are shown in Fig. 11.

Table 5: **Repeated sampling under TMD-conditioning.** Persistence-diagram distances to the assigned target and to root-degree-matched alternatives across five generations per test neuron. "assigned closer" is the percentage of matched comparisons in which the assigned target has the lower distance. Variability between samples is much lower than to arbitrary neurons showing the effect of conditioning.

TMD channel	Assigned target W_1	Matched alternative W_1	Assigned/matched ratio ↓	Assigned closer (%) ↑
Path PD	2.916	5.543	0.526 ± 0.015	$91.1 \pm 0.8\%$
Radial-root PD	2.775	5.666	0.490 ± 0.015	$90.8 \pm 0.8\%$

H.2 Class-Conditioned Population Structure

In Fig. 8 we report errors by class for class-conditional generation. As before, our model receives the soma degree while SemlaFlow gets a reference node count which thus only produces an error when the postprocessing removes nodes after generation, for example when selecting the largest component from a forest. Our method tends to outperform SemlaFlow over all classes and metrics except for node count as described above and partition asymmetry where the image is mixed. Also on the axial extent for 23P neurons SemlaFlow is slightly better, while especially on 5P-NP which are hard for SemlaFlow we see a strong advantage of our method. Overall, the advantage of our method is visible across cell types and there are clearly easier (5P-IT, 6P-IT) and harder (4P, 5P-ET) cell types for both models. Also both models tend to work well on the same cell types, except for 5P-NP where SemlaFlow struggles while our method works reasonably well.

I Additional Samples

The following galleries expand the examples in the main text, showing unconditional samples across methods, class-conditioned samples, and repeated generations from the same TMD conditions. Since SemlaFlow and MorphoGen do not have a principal axis, they are plotted such that the root-to-centroid axis is used as such ('up'). Colour encodes signed depth, and translucent red rings mark the roots.



Figure 8: **Error by cell type.** Class-conditioned generation errors for our model (top) and SemlaFlow (bottom). Entries are class-wise W_1 normalised by the reference standard deviation (lower is better). Both models work well on the same cell types and ours clearly outperforms SemlaFlow, except for node count that is given to SemlaFlow as input and partition asymmetry where errors are generally low.



Figure 9: **Unconditional generation for neurons.** Reference morphologies, post-processed SemlaFlow generations, MorphoGen generations, and generations from our model are shown. Neurons are drawn root-centred and to scale within each method.

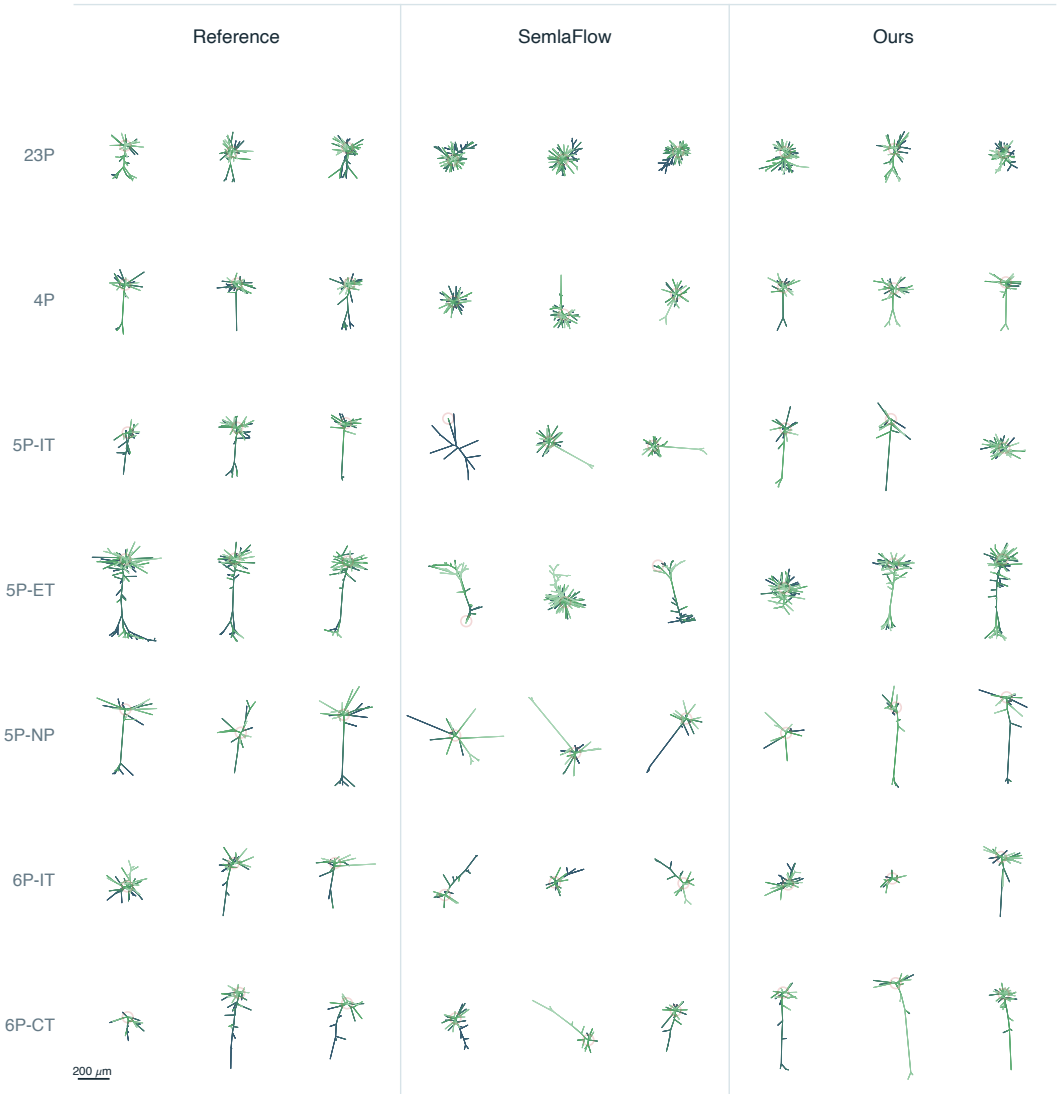


Figure 10: **Class-conditioned neuron samples.** Each row corresponds to one of the seven cortical pyramidal-cell classes; columns compare training examples, SemlaFlow generations, and generations from our model. The displayed samples are selected per source and class rather than treated as paired reconstructions.

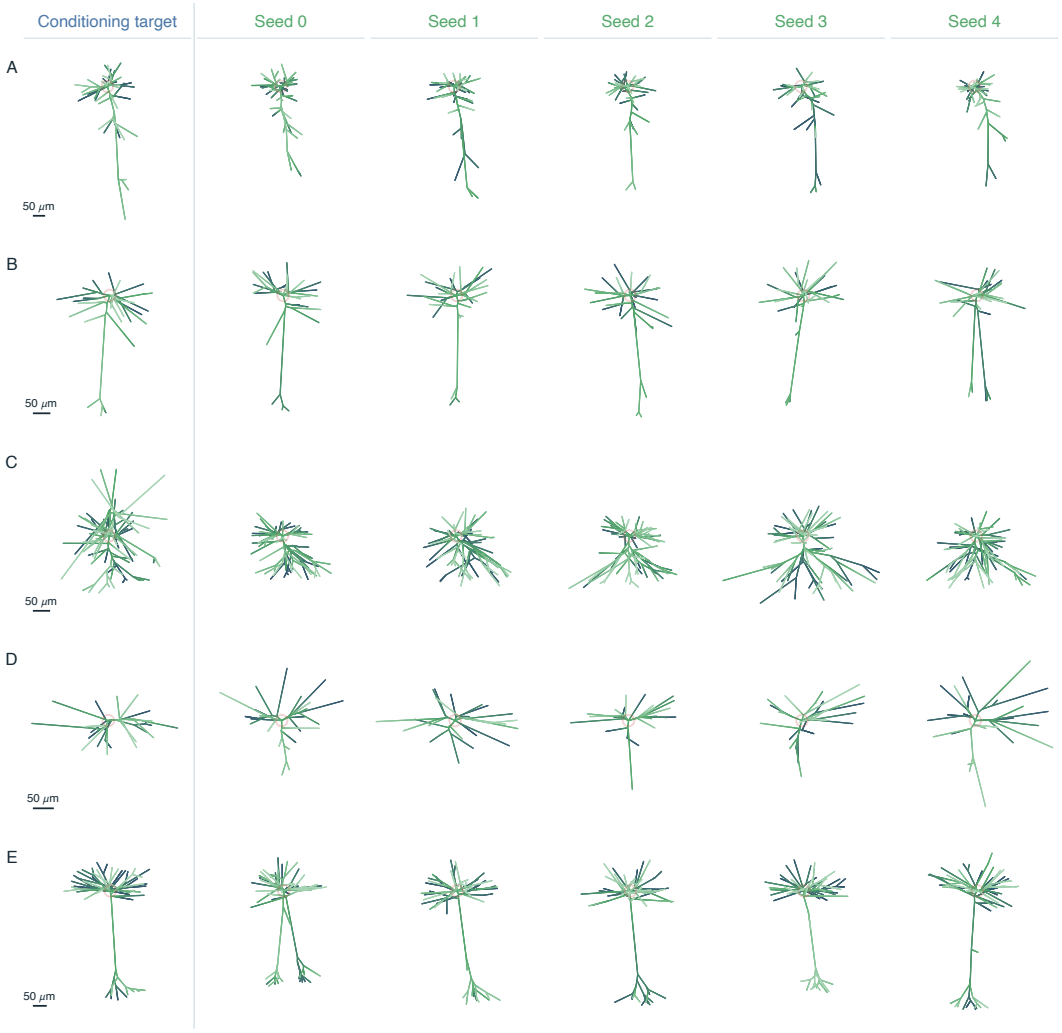


Figure 11: **Stochastic variation under TMD-conditioning.** Each row shows one held-out conditioning target followed by five independent generations for that target. The generated samples preserve the target's coarse dendritic organisation while varying in branch geometry and topology. Row C shows a more difficult condition for which the generations are visibly more compact than the target. B4 and E0 are unrealistic since they exhibit two apical dendrites instead of just one.